

SUBJECT CODE : 310241

As per Revised Syllabus of
SAVITRIBAI PHULE PUNE UNIVERSITY

Choice Based Credit System (CBCS)

T.E. (Computer) Semester - V

DATABASE MANAGEMENT SYSTEMS

Anuradha A. Puntambekar

M.E. (Computer)

Formerly Assistant Professor in
P.E.S. Modern College of Engineering,
Pune.

Dr. Pramod Patil

Ph.D. (Computer Engineering)

M.E. (Computer Engineering), B.E. (CSE)

Professor in Computer Engineering

Dr. D.Y. Patil Institute of Technology,
Pimpri Pune.



DATABASE MANAGEMENT SYSTEMS

Subject Code : 310241

T.E. (Computer Engineering) Semester - V

© Copyright with Authors

All publishing rights (printed and ebook version) reserved with Technical Publications. No part of this book should be reproduced in any form, Electronic, Mechanical, Photocopy or any information storage and retrieval system without prior permission in writing, from Technical Publications, Pune.

Published by :



Amit Residency, Office No.1, 412, Shaniwar Peth,
Pune - 411030, M.S. INDIA, Ph.: +91-020-24495496/97
Email : sales@technicalpublications.org Website : www.technicalpublications.org

Printer :

Yogiraj Printers & Binders
Sr.No. 10/1A,
Ghule Industrial Estate, Nanded Village Road,
Tal. - Haveli, Dist. - Pune - 411041.

ISBN 978-93-91567-34-7



9 789391 567347

SPPU 19

PREFACE

The importance of **Database Management Systems** is well known in various engineering fields. Overwhelming response to our books on various subjects inspired us to write this book. The book is structured to cover the key aspects of the subject **Database Management Systems**.

The book uses plain, lucid language to explain fundamentals of this subject. The book provides logical method of explaining various complicated concepts and stepwise methods to explain the important topics. Each chapter is well supported with necessary illustrations, practical examples and solved problems. All the chapters in the book are arranged in a proper sequence that permits each topic to build upon earlier studies. All care has been taken to make students comfortable in understanding the basic concepts of the subject.

Representative questions have been added at the end of each section to help the students in picking important points from that section.

The book not only covers the entire scope of the subject but explains the philosophy of the subject. This makes the understanding of this subject more clear and makes it more interesting. The book will be very useful not only to the students but also to the subject teachers. The students have to omit nothing and possibly have to cover nothing more.

We wish to express our profound thanks to all those who helped in making this book a reality. Much needed moral support and encouragement is provided on numerous occasions by our whole family. We wish to thank the **Publisher** and the entire team of **Technical Publications** who have taken immense pain to get this book in time with quality printing.

Any suggestion for the improvement of the book will be acknowledged and well appreciated.

Authors
A. A. Puntambekar
Pramod Patil

Dedicated to God.

SYLLABUS

Database Management Systems - 310241

Credit :	Examination Scheme :
03	Mid-Sem (TH) : 30 Marks End-Sem (TH) : 70 Marks

Unit I Introduction to Database Management Systems and ER Model

Introduction, Purpose of Database Systems, Database-System Applications, View of Data, Database Languages, Database System Structure, Data Models. **Database Design and ER Model** : Entity, Attributes, Relationships, Constraints, Keys, Design Process, Entity-Relationship Model, ER Diagram, Design Issues, Extended E-R Features, converting ER and EER diagram into tables. **(Chapter - 1)**

Unit II SQL and PL/SQL

SQL : Characteristics and Advantages, SQL Data Types and Literals, DDL, DML, DCL, TCL, SQL Operators. **Tables** : Creating, Modifying, Deleting, Updating. **SQL DML Queries** : SELECT Query and clauses, Index and Sequence in SQL. **Views** : Creating, Dropping, Updating using Indexes, Set Operations, Predicates and Joins, Set membership, Tuple Variables, Set comparison, Ordering of Tuples, Aggregate Functions, SQL Functions, Nested Queries. **PL/SQL** : Concept of Stored Procedures and Functions, Cursors, Triggers, Assertions, Roles and Privileges. **(Chapter - 2, 3)**

Unit III Relational Database Design

Relational Model : Basic concepts, Attributes and Domains, CODD's Rules. **Relational Integrity** : Domain, Referential Integrities, Enterprise Constraints. **Database Design** : Features of Good Relational Designs, Normalization, Atomic Domains and First Normal Form, Decomposition using Functional Dependencies, Algorithms for Decomposition, 2NF, 3NF, BCNF. **(Chapter - 4)**

Unit IV Database Transaction Management

Introduction to Database Transaction, Transaction states, ACID properties, Concept of Schedule, Serial Schedule. **Serializability** : Conflict and View, Cascaded Aborts, Recoverable and Non-recoverable Schedules. **Concurrency Control** : Lock-based, Time-stamp based Deadlock handling. **Recovery methods** : Shadow-Paging and Log-Based Recovery, Checkpoints. **Log-Based Recovery** : Deferred Database Modifications and Immediate Database Modifications. **(Chapter - 5)**

Unit V NoSQL Databases

Introduction to Distributed Database System, Advantages, Disadvantages, CAP Theorem.

Types of Data : Structured, Unstructured Data and Semi-Structured Data.

NoSQL Database : Introduction, Need, Features. **Types of NoSQL Databases** : Key-value store, document store, graph, wide column stores, BASE Properties, Data Consistency model, ACID Vs BASE, Comparative study of RDBMS and NoSQL. **MongoDB** (with syntax and usage) : CRUD Operations, Indexing, Aggregation, MapReduce, Replication, Sharding. **(Chapter - 6)**

Unit VI Advances in Databases

Emerging Databases : Active and Deductive Databases, Main Memory Databases, Semantic Databases.

Complex Data Types :

Semi-Structured Data, Features of Semi-Structured Data Models. **Nested Data Types** : JSON, XML.

Object Orientation : Object-Relational Database System, Table Inheritance, Object-Relational Mapping. **Spatial Data** : Geographic Data, Geometric Data. **(Chapter - 7)**

TABLE OF CONTENTS

Unit - I

Chapter - 1 Introduction to Database Management Systems and ER Model (1 - 1) to (1 - 68)

Part I : Introduction to DBMS

1.1	Introduction	1 - 2
1.1.1	Characteristics of Database Approach.....	1 - 2
1.2	Purpose of Database Systems.....	1 - 2
1.3	Advantages of DBMS over File Processing Systems.....	1 - 3
1.3.1	Advantages of DBMS	1 - 3
1.3.2	Disadvantages of DBMS.....	1 - 3
1.3.3	File Processing System Vs. DBMS	1 - 4
1.4	Database-System Applications.....	1 - 5
1.5	View of Data.....	1 - 6
1.6	Data Abstraction	1 - 7
1.7	Database Languages	1 - 9
1.8	Database System Structure.....	1 - 10
1.8.1	Overall Structure of DBMS.....	1 - 10
1.8.2	Architecture of DBMS	1 - 12
1.9	Data Models.....	1 - 14
1.10	Data Independence.....	1 - 19
1.11	Database Users	1 - 20

Part II : Data Modelling

1.12	Database Design and ER Model	1 - 21
1.12.1	Entity and Entity Sets.....	1 - 21
1.12.2	Relationships.....	1 - 22

1.12.3	Attributes.....	1 - 22
1.12.3.1	Types of Attributes.....	1 - 23
1.13	Constraints	1 - 24
1.13.1	Types of Cardinality	1 - 24
1.14	Keys	1 - 26
1.15	Design Process	1 - 26
1.16	ER Diagram.....	1 - 28
1.17	Conventions	1 - 29
1.17.1	Mapping Cardinality Representation.....	1 - 29
1.17.2	Ternary Relationship.....	1 - 30
1.17.3	Weak Entity Set.....	1 - 31
1.18	Design Issues	1 - 32
1.19	Extended E-R Features	1 - 34
1.19.1	Specialization and Generalization.....	1 - 34
1.19.2	Constraints on Specialization / Generalization.....	1 - 35
1.19.3	Aggregation	1 - 37
1.20	Converting ER and EER Diagram into Tables.....	1 - 37
1.20.1	Mapping of Entity Set to Relationship.....	1 - 37
1.20.2	Mapping Relationship Sets(without Constraints) to Tables	1 - 38
1.20.3	Mapping Relationship Sets(With Constraints) to Tables	1 - 39
1.20.4	Mapping Weak Entity Sets to Relational Mapping	1 - 41
1.20.5	Mapping of Specialization / Generalization(EER Construct)	
	to Relational Mapping	1 - 41
1.21	Examples based on ER Diagram	1 - 43
	Multiple Choice Questions with Answers	1 - 62

Unit - II

Chapter - 2	Structured Query Language	(2 - 1) to (2 - 82)
--------------------	----------------------------------	----------------------------

2.1	Introduction to Structured Query Language.....	2 - 3
2.1.1	Characteristics and Advantages.....	2 - 3
2.2	SQL Data Types and Literals	2 - 3

2.3	DDL, DML, DCL and TCL Structure.....	2 - 4
2.4	Tables	2 - 6
2.4.1	Creating Table.....	2 - 6
2.4.2	Insertion of Data into the Table.....	2 - 7
2.4.3	Modifying the Record from the Table	2 - 7
2.4.4	Deleting Record from the Table	2 - 8
2.5	SQL DML Queries	2 - 8
2.5.1	SELECT Query.....	2 - 9
2.5.2	WHERE	2 - 10
2.5.3	Clauses.....	2 - 11
2.6	Logical Operators	2 - 14
2.7	String Operations	2 - 16
2.8	The BETWEEN Operator.....	2 - 18
2.9	Built-In Functions	2 - 18
2.10	NULL Values	2 - 20
2.11	EXISTS, NOT EXISTS and UNIQUE	2 - 22
2.12	Defining Constraints.....	2 - 24
2.13	Renaming Attributes	2 - 28
2.14	Tuple Variables.....	2 - 29
2.15	Schema Change Statements	2 - 29
2.15.1	The DROP Command	2 - 29
2.15.2	The ALTER Command.....	2 - 30
2.16	Indexes	2 - 32
2.17	Aggregate Functions	2 - 33
2.18	Set Operations	2 - 35
2.19	Nested Queries	2 - 37
2.20	Join Operation.....	2 - 39
2.21	Views.....	2 - 43
2.22	Examples Based on SQL	2 - 48

3.1 Basics of PL/SQL	3 - 2
3.1.1 How to Set Environment for Executing PL/SQL Scripts ?	3 - 2
3.2 Writing First PL/SQL Script	3 - 4
3.3 Block Structure of PL/SQL	3 - 9
3.4 PL/SQL Data Types	3 - 10
3.5 PL/SQL Variables	3 - 11
3.6 PL/SQL Constants	3 - 14
3.7 Control Statements	3 - 14
3.7.1 IF Statement	3 - 14
3.7.2 General Loop	3 - 16
3.7.3 For Loop	3 - 17
3.7.4 While Loop	3 - 18
3.7.5 CASE Statement	3 - 19
3.8 Handling Database Tables using PL/SQL	3 - 20
3.9 Examples on PL/SQL	3 - 24
3.10 Concept of Stored Procedures	3 - 27
3.10.1 Procedures without Parameter	3 - 28
3.10.2 Procedures with Parameters	3 - 29
3.10.3 Stored Procedure for Handling Database Table	3 - 30
3.11 Functions	3 - 31
3.11.1 PL/SQL Stored Function for Table	3 - 33
3.12 Cursors	3 - 35
3.13 Triggers	3 - 41
3.14 Assertions	3 - 47
3.15 Roles and Privileges	3 - 48
3.16 Exceptions	3 - 49
Multiple Choice Questions with Answers	3 - 54

Unit - III

Chapter - 4 Relational Database Design

(4 - 1) to (4 - 74)

Part I : Relational Model

4.1 Basic Concepts.....	4 - 3
4.2 Attributes and Domains	4 - 4
4.3 CODD's Rules	4 - 6

Part II : Relational Integrity

4.4 Keys	4 - 7
4.5 Constraints	4 - 11
4.6 Enterprise Constraints.....	4 - 14

Part III : Database Design

4.7 Features of Good Relational Designs	4 - 14
4.8 Data Redundancy and Update Anomalies	4 - 15
4.9 Normalization.....	4 - 16
4.10 Atomic Domains and First Normal Form.....	4 - 17
4.11 Decomposition using Functional Dependencies	4 - 18
4.11.1 Inference Rules	4 - 20
4.11.2 Keys and Functional Dependencies	4 - 23
4.12 Equivalence and Minimal Cover.....	4 - 25
4.13 Algorithms for Decomposition	4 - 35
4.14 Lossless Join.....	4 - 37
4.15 Dependency Preservation	4 - 39
4.16 Second Normal Form (2NF).....	4 - 44
4.17 Third Normal Form (3NF)	4 - 48
4.18 BCNF	4 - 61
Multiple Choice Questions with Answers	4 - 72

Unit - IV

Chapter - 5 Database Transaction Management (5 - 1) to (5 - 66)

Part I : Transaction Management

5.1 Introduction to Database Transaction	5 - 2
5.2 Transaction States	5 - 3
5.3 ACID Properties	5 - 5
5.4 Concept of Schedule.....	5 - 6
5.5 Serializability : Conflict and View	5 - 7
5.5.1 Conflict Serializability.....	5 - 9
5.5.2 View Serializability	5 - 20
5.6 Recoverable and Non-recoverable Schedules	5 - 27
5.6.1 Recoverable Schedule.....	5 - 28
5.6.2 Cascadeless Schedule	5 - 29

Part II : Concurrency Control

5.7 Concurrency Control	5 - 30
5.8 Need for Concurrency	5 - 30
5.9 Lock-based Protocol.....	5 - 33
5.9.1 Why Do We Need Lock ?	5 - 33
5.9.2 Working of Lock	5 - 33
5.9.3 Two Phase Locking Protocol	5 - 35
5.9.3.1 Types of Two Phase Locking.....	5 - 39
5.10 Time-stamp based Protocol	5 - 42
5.11 Deadlocks Handling.....	5 - 46
5.12 Recovery Concepts.....	5 - 52
5.12.1 Purpose of Database Recovery	5 - 52
5.12.2 Failure Classification	5 - 53
5.12.3 Storage.....	5 - 54
5.13 Recovery Methods	5 - 55

5.14 Shadow-paging.....	5 - 55
5.15 Check Points.....	5 - 56
5.16 Log-based Recovery : Deferred and Immediate Update	5 - 57
5.16.1 Concept of Log.....	5 - 57
5.16.2 REDO and UNDO Operation.....	5 - 58
5.16.3 Write Ahead Logging Rule	5 - 58
5.16.4 Deferred Database Modification	5 - 58
5.16.5 Immediate Database Modification	5 - 60
Multiple Choice Questions with Answers	5 - 63

Unit - V

Chapter - 6	NoSQL Databases	(6 - 1) to (6 - 40)
6.1	Introduction to Distributed Database System	6 - 2
6.2	CAP Theorem	6 - 3
6.3	Types of Data : Structured, Unstructured Data and Semi-Structured Data ...	6 - 4
6.4	NoSQL Database.....	6 - 4
6.4.1	Introduction.....	6 - 4
6.4.2	Need	6 - 5
6.4.3	Features	6 - 5
6.5	Types of NoSQL Databases.....	6 - 5
6.5.1	Key-Value Store	6 - 6
6.5.2	Document Store.....	6 - 6
6.5.3	Graph	6 - 6
6.5.4	Wide Column Store.....	6 - 7
6.6	BASE Properties.....	6 - 8
6.7	ACID Vs BASE.....	6 - 8
6.8	Comparative Study of RDBMS and NoSQL.....	6 - 8
6.9	MongoDB	6 - 9
6.9.1	Data Types	6 - 11
6.9.2	MongoDB Installation	6 - 12

6.9.3	CRUD Operations	6 - 19
6.9.4	Indexing	6 - 29
6.9.5	Aggregation	6 - 32
6.9.6	Map Reduce.....	6 - 34
6.9.7	Replication	6 - 36
6.9.8	Sharding.....	6 - 37
	Multiple Choice Questions with Answers	6 - 38

Unit - VI

Chapter - 7	Advances in Databases	(7 - 1) to (7 - 22)
--------------------	------------------------------	----------------------------

7.1	Emerging Databases.....	7 - 2
7.1.1	Active and Deductive Databases	7 - 2
7.1.2	Main Memory Databases	7 - 4
7.1.3	Semantic Databases.....	7 - 4
7.2	Complex Data Types.....	7 - 5
7.2.1	Semi-Structured Data	7 - 5
7.2.2	Features of Semi-Structured Data Models	7 - 7
7.3	Nested Data Types	7 - 7
7.3.1	JSON.....	7 - 7
7.3.2	XML.....	7 - 9
7.3.3	Difference between XML and HTML.....	7 - 10
7.3.4	Example of XML	7 - 11
7.3.5	Building Blocks of XML Document.....	7 - 12
7.3.6	Concept of Namespace.....	7 - 12
7.3.7	Document Type Definition (DTD)	7 - 13
7.4	Object Orientation	7 - 16
7.4.1	Object-Relational Database System	7 - 17
7.4.2	Table Inheritance.....	7 - 17
7.4.3	Object-Relational Mapping.....	7 - 18
7.5	Spatial Data	7 - 19

7.5.1	Geometric Data.....	7 - 19
7.5.2	Geographic Data	7 - 20
	Multiple Choice Questions with Answers	7 - 21
	Solved Model Question Papers	(M - 1) to (M - 4)

UNIT - I

1

Introduction to Database Management Systems and ER Model

Syllabus

Introduction, Purpose of Database Systems, Database-System Applications, View of Data, Database Languages, Database System Structure, Data Models. Database Design and ER Model : Entity, Attributes, Relationships, Constraints, Keys, Design Process, Entity-Relationship Model, ER Diagram, Design Issues, Extended E-R Features, converting ER and EER diagram into tables.

Contents

Part I : Introduction to DBMS

1.1	Introduction	
1.2	Purpose of Database Systems	
1.3	Advantages of DBMS over File Processing Systems	May-19, Marks 5
1.4	Database-System Applications	
1.5	View of Data	May-19, Oct.-19, Marks 5
1.6	Data Abstraction	Dec.-17, May-19, Marks 5
1.7	Database Languages	
1.8	Database System Structure	May-18, Oct.-18, Dec.-18,19, Marks 5
1.9	Data Models	Nov.-19, Marks 5
1.10	Data Independence	Nov.-19, Oct.-19, Aug.-17 Marks 5
1.11	Database Users	May-19, Nov.-18, Marks 5

Part II : Data Modelling

1.12	Database Design and ER Model	May-18, Nov.-17,19,..... Marks 5
1.13	Constraints	
1.14	Keys	May-19, Oct.-19, Marks 5
1.15	Design Process	
1.16	ER Diagram	Aug.-17, Marks 2
1.17	Convention	Oct.-19, Marks 2
1.18	Design Issues	
1.19	Extended E-R Features	
1.20	Converting ER and EER Diagram into Tables	Nov.-18, Marks 4
1.21	Examples based on ER Diagram	
	Multiple Choice Questions	

Part I : Introduction to DBMS

1.1 Introduction

- **Definition :** A Database Management System (DBMS) is collection of **interrelated data** and various **programs** that are used to handle the data.
- The **primary goal** of DBMS is to provide a way to **store and retrieve** the required information from the database in convenient and efficient manner.
- For managing the data in the database two important **tasks** are conducted -
 - **Define the structure** for storage of information.
 - Provide mechanism for **manipulation of information**.
- In addition, the database systems must ensure the **safety of information** stored.

1.1.1 Characteristics of Database Approach

Following are the characteristics of database system :

- 1) Representation of some aspects of **real world applications**.
- 2) Systematic **management of information**.
- 3) Representing the data by **multiple views**.
- 4) Efficient and easy implementation of various **operations** such as insertion, deletion and updation.
- 5) It maintains **data for some specific purpose**.
- 6) It represents **logical relationship** between records and data.

1.2 Purpose of Database Systems

- Earlier database systems are created in response to manage the commercial data. These data is typically stored in files. To allow users to manipulate these files various programs are written for
 - 1) Addition of new data
 - 2) Updating the data
 - 3) Deleting the data.
- As per the addition of new need, separate application programs were required to write. Thus as the time goes by, the system acquires more files and more application programs.
- This typical **file processing system** is supported by conventional operating system. Thus the file processing system can be described as –

- The system that stores the permanent records in files and it needs different application programs to extract or add the records.

1.3 Advantages of DBMS over File Processing Systems

SPPU : May-19, Marks 5

1.3.1 Advantages of DBMS

Following are the advantages of DBMS :

- 1) DBMS **removes the data redundancy** that means there is no duplication of data in database.
- 2) DBMS allows to **retrieve the desired data** in required format.
- 3) **Data** can be **isolated** in separate tables for convenient and efficient use.
- 4) Data can be **accessed efficiently** using a simple query language.
- 5) The **data integrity** can be maintained. That means – the constraints can be applied on data and it should be in some specific range.
- 6) The **atomicity** of data can be maintained. That means, if some operation is performed on one particular table of the database, then the change must be reflected for the entire database.
- 7) The DBMS allows **concurrent access** to multiple users by using the synchronization technique.
- 8) The **security policies** can be applied to DBMS to allow the user to access only desired part of the database system.

1.3.2 Disadvantages of DBMS

- 1) **Complex design** : Database design is complex, **difficult and time consuming**.
- 2) **Hardware and software cost** : **Large amount of investment** is needed to setup the required hardware or to repair software failure.
- 3) **Damaged part** : If one part of database is corrupted or damaged, then **entire database** may get affected.
- 4) **Conversion cost** : If the current system is in conventional file system and if we need to convert it to database systems then large amount of cost is incurred in purchasing different tools, and adopting different techniques as per the requirement.
- 5) **Training** : For designing and maintaining the database systems, the people need to be trained.

1.3.3 File Processing System Vs. DBMS

- Earlier database systems are created in response to manage the commercial data. These data is typically stored in files. To allow users to manipulate these files, various programs are written for
 - 1) Addition of new data
 - 2) Updating the data
 - 3) Deleting the data.
- As per the need for addition of new data, separate application programs were required to write. Thus as the time goes by, the system acquires more files and more application programs.
- This typical **file processing system** is supported by conventional operating system. Thus the file processing system can be described as -
- “The system that stores the permanent records in files and it needs different application programs to extract or add the records”.
- Before introducing database management system, this file processing system was in use. However, such a system has many drawbacks. Let us discuss them.

Disadvantages of Traditional File Processing System

The traditional file system has following disadvantages :

- 1) **Data redundancy** : Data redundancy means duplication of data at several places. Since different programmers create different files and these files might have different structures, there are chances that some information may appear repeatedly in some or more format at several places.
- 2) **Data inconsistency** : Data inconsistency occurs when various copies of same data may no longer get matched. For example changed address of an employee may be reflected in one department and may not be available (or old address present) for other department.
- 3) **Difficulty in accessing data** : The conventional file system does not allow to retrieve the desired data in efficient and convenient manner.
- 4) **Data isolation** : As the data is scattered over several files and files may be in different formats, it becomes difficult to retrieve the desired data from the file for writing the new application.
- 5) **Integrity problems** : Data integrity means data values entered in the database fall within a specified range and are of specific format. With the use of several files enforcing such constraint on the data becomes difficult.

- 6) **Atomicity problems** : An atomicity means particular operation must be carried out entirely or not at all with the database. It is difficult to ensure atomicity in conventional file processing system.
- 7) **Concurrent access anomalies** : For efficient execution, multiple users update data simultaneously, in such a case data need to be synchronized. As in traditional file systems, data is distributed over multiple files, one cannot access these files concurrently.
- 8) **Security problems** : Every user is not allowed to access all the data of database system. Since application program in file system are added in an ad hoc manner, enforcing such security constraints become difficult.

Database systems offer solutions to all the above mentioned problems.

Difference between Database System and Conventional File System

Sr. No.	Database systems	Conventional file systems
1.	Data redundancy is less .	Data redundancy is more .
2.	Security is high .	Security is very low .
3.	Database systems are used when security constraints are high .	Conventional file systems are used where there is less demand for security constraints .
4.	Database systems define the data in a structured manner. Also there is well defined co-relation among the data.	File systems define the data in un-structured manner. Data is usually in isolated form.
5.	Data inconsistency is less in database systems.	Data inconsistency is more in file systems.
6.	User is unknown to the physical address of the data used in database systems.	User locates the physical address of file to access the data in conventional file systems.
7.	We can retrieve the data in any desired format using database systems.	We cannot retrieve the data in any desired format using file systems.
8.	There is ability to access the data concurrently using database systems.	There is no ability to concurrently access the data using conventional file system.

Review Question

1. Define DBMS. Explain advantages of DBMS over file system.

SPPU : May-19, End Sem, Marks 5

1.4 Database-System Applications

There are wide range of applications that make use of database systems. Some of the applications are -

- 1) **Accounting** : Database systems are used in maintaining information employees, salaries, and payroll taxes.
- 2) **Manufacturing** : For management of supply chain and tracking production of items in factories database systems are maintained.
- 3) For maintaining customer, product and purchase information the databases are used.
- 4) **Banking** : In banking sector, for customer information, accounts and loan and for performing banking applications the DBMS is used.
- 5) For purchase on credit cards and generation of monthly statements database systems are useful.
- 6) **Universities** : The database systems are used in universities for maintaining student information, course registration, and accounting.
- 7) **Reservation systems** : In airline / railway reservation systems, the database is used to maintain the reservation and schedule information.
- 8) **Telecommunication** : In telecommunications for keeping records of the calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about communication networks the database systems are used.

1.5 View of Data

SPPU : May-19, Oct.-19, Marks 5

- Database is a collection of interrelated data and set of programs that allow users to access or modify the data.
- Abstract view of the system is a view in which the system hides certain details of how the data are stored and maintained.
- The **main purpose** of database systems is to provide users with abstract view of the data.
- The view of the system help the user to **retrieve data** efficiently.
- For simplifying the user interaction with the system there are several levels of abstraction - these levels are - Physical level, logical level and view level.

Review Question

1. *Elaborate the need of database views. Also explain the situations where in the view created are updateable views.*

SPPU : May-19, End Sem, Oct.-19, In Sem, Marks 5

1.6 Data Abstraction

SPPU : Dec.-17, May-19, Marks 5

Definition of data abstraction : Data abstraction means retrieving only the required amount of information about the system and **hiding** background details.

There are several levels of abstraction that simplify user interactions with the system. These are :

1) Physical level :

- This is the **lowest level**.
- This level describes how the data are **stored**.
- The database administrators decide how to store data at the physical level.
- This level describes complex low-level data structures.

2) Logical level :

- This is the next **higher level**, which describes what data are stored in the database?.
- This level also describes the **relationship between the data**.
- The logical level thus describes the entire database in terms of a small number of relatively simple structures.
- The database administrators use a logical level of abstraction for deciding what information to keep in the database.

3) View level :

- This is the **highest level** of abstraction that describes only part of the entire database.
- The view level can provide access to only part of the database.
- This level helps in **simplifying the interaction** with the system.
- It can provide **multiple views** of the same system.
- For example - A Clerk at the reservation system can see only part of the database and access the passenger's required information.

Fig. 1.6.1 shows the relationship between the three levels of abstraction.

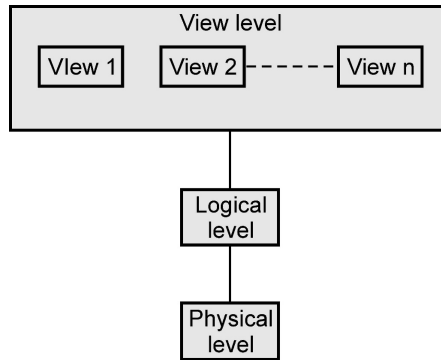


Fig. 1.6.1 : Levels of data abstraction

For example : Consider following record

```

type employee = record
    empID : numeric(10)
    empname : char(20)
    dept_no : numeric(10)
    salary : numeric(8,2)
end
  
```

This code defines a new record **employee** with four fields. Each field is associated with field name and its **type**. There are several other records such as **department** with fields dept_no, dept_name, building customer with fields cust_id, cust_name

- At the **physical level**, the record - **customer**, **employee**, **department** can be described as block of **consecutive storage locations**. Many database systems hide lowest level storage details from database programmer.
- The type definition of the records is decided **at the logical level**. The **programmer** work of the record at this level, similarly **database administrators** also work at this level of abstraction.
- There is specific view of the record is allowed at the view level. For instance - customer can view the name of the employee, or id of the employee but cannot access employee's salary.

Review Question

1. For the database system to be usable, it must retrieve data efficiently. The need of efficiency has led designers to use complex data structures to represent data in the database. Developers hides this complexity from the database system users through several levels of abstraction. Explain those levels of abstraction in detail.

SPPU : Dec.-17, May-19, End Sem, Marks 5

1.7 Database Languages

There are three types of languages supported by database systems.

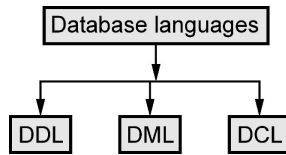


Fig. 1.7.1 Types of database languages

(1) DDL

- Data Definition Language (DDL) is a specialized language used to specify a database schema by a set of definitions.
- It is a language used for **creating** and **modifying** the structures of tables, views, indexes, etc.
- DDL is also used to specify additional properties of data.
- Some of the common commands used in DDL are **-CREATE, ALTER, DROP**.
- The **primary use** of CREATE command is to build a new table. Using ALTER command, the users can add up some additional column and drop existing columns. Using DROP command, the user can delete table or view.

(2) DML

- DML stands for Data Manipulation Language.
- This language enables users to access or manipulate data as organized by appropriate data model.
- The types of access are -
 - **Retrieval** of information stored in the database
 - **Insertion** of new information into the database.
 - **Deletion** of information from the database.
 - **Modification** of information stored in database.
- There are two types of DML -
 - **Procedural DML** - Require a user to specify what data are needed and how to get those data.
 - **Declarative DML** - Require a user to specify what data are needed without specifying how to get those data.
- **Query** is a statement used for requesting the retrieval of information. This retrieval of information using some specific language is called **query language**.

(3) DCL

- The Data Control Language (DCL) is used to control access to data stored in the database. This is also called as authorization.
- The typical command used in DCL are GRANT and REVOKE.
 - **GRANT** : This command is used to give access rights or privileges to the database.
 - **REVOKE** : The revoke command removes user access rights or privileges to the database objects

1.8 Database System Structure

SPPU : May-18, Oct.-18, Dec.-18,19, Marks 5

1.8.1 Overall Structure of DBMS

Three Schema Architecture

- **Definition** : Database schema is a collection of database objects like tables, views, indexes and so on associated with one particular database username. This username is called the **schema owner**.
- For example Student Schema can be owner of STUDENT and MARKS tables. The Course schema can be the owner of SUBJECT table.
- The **goal** of three-schema architecture is to separate the user application from the physical database.
- The architecture of database is divided into **three levels** based on three types of schema - internal schema, conceptual schema or external schema.

1. Internal level :

- It contains **internal schema**.
- This schema represents the **physical storage structure of database**.
- This schema is maintained by the software and user is not allowed to modify it.
- This level is closest to the physical storage. It typically describes the record layout of the files and types of files, access paths etc.

2. Conceptual level :

- It contains **conceptual schema**.
- This schema **hides the details** of internal level.
- This level is also called as logical level as it contains the constructs used for designing the database.

- It contains information like table name, their columns, indexes and constraints, database operations.
- A representational data model is used to describe conceptual schema when a database system is implemented.

3. External level :

- It contains the external schema or user views.
- At this level, the user will get to see only the data stored in the database. Either they will see whole data values or any specific records. They will not have any information about how they are stored in the database.

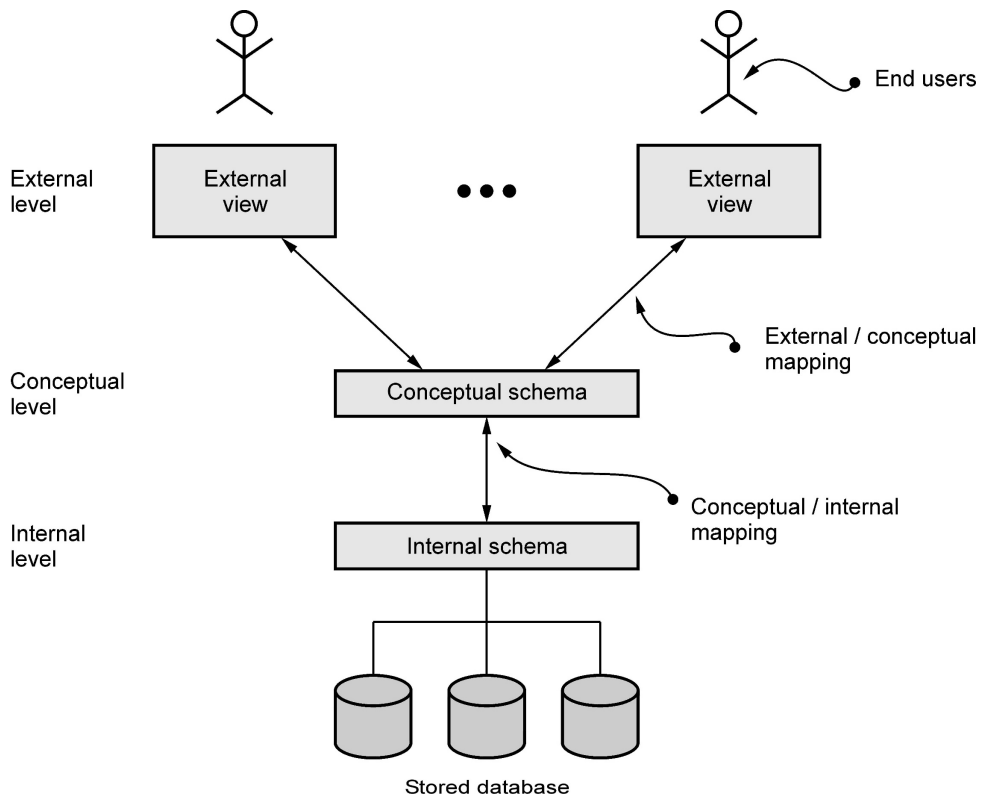


Fig. 1.8.1 Three schema architecture

- The processes of transforming requests and results between levels are called **mappings**.
- In the three schema architecture there are two mappings –
 - 1) External - Conceptual Mapping and
 - 2) Conceptual - Internal Mapping

1.8.2 Architecture of DBMS

- The typical structure of typical DBMS is based on relational data model as shown in Fig. 1.8.2.

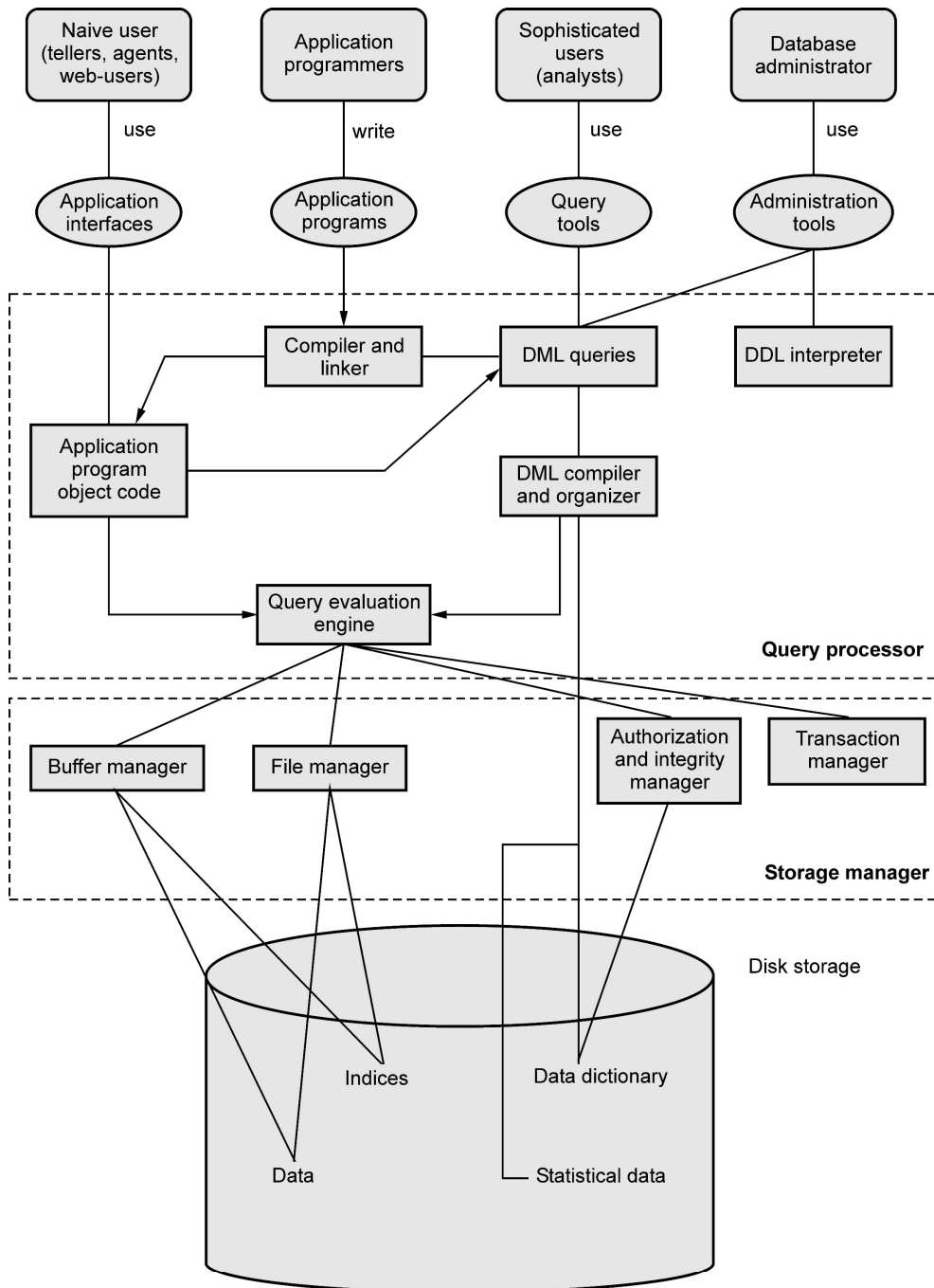


Fig. 1.8.2 Architecture of database

- Consider the top part of Fig. 1.8.2. It shows **application interfaces** used by naïve users, application programs created by application programmers, query tools used by sophisticated users and administration tools used by database administrator
- The lowest part of the architecture is for **disk storage**.
- The two important components of database architecture are - **Query processor and storage manager**.

Query processor :

- The interactive query processor helps the database system to simplify and facilitate access to data. It consists of DDL interpreter, DML compiler and query evaluation engine.
- With the following components of query processor, various functionalities are performed -
 - i) **DDL interpreter** : This is basically a translator which interprets the DDL statements in data dictionaries.
 - ii) **DML compiler** : It translates DML statements query language into an evaluation plan. This plan consists of the instructions which query evaluation engine understands.
 - iii) **Query evaluation engine** : It executes the low-level instructions generated by the DML compiler.
- When a user issues a query, the parsed query is presented to a query optimizer, which uses information about how the data is stored to produce an efficient execution plan for evaluating the query. An execution plan is a blueprint for evaluating a query. It is evaluated by **query evaluation engine**.

Storage Manager :

- Storage manager is the component of database system that provides interface between the low level data stored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible for storing, retrieving, and updating data in the database. The storage manager components include -
 - i) **Authorization and integrity manager** : Validates the users who want to access the data and tests for integrity constraints.
 - ii) **Transaction manager** : Ensures that the database remains in consistent despite of system failures and concurrent transaction execution proceeds without conflicting.
 - iii) **File manager** : Manages allocation of space on disk storage and representation of the information on disk.

- iv) **Buffer manager** : Manages the fetching of data from disk storage into main memory. The buffer manager also decides what data to cache in main memory. Buffer manager is a crucial part of database system.
- o Storage manager implements several data structures such as -
 - i) **Data files** : Used for storing database itself.
 - ii) **Data dictionary** : Used for storing metadata, particularly schema of database.
 - iii) **Indices** : Indices are used to provide fast access to data items present in the database

Review Questions

1. Draw and explain overall structure of database system.

SPPU : May-18, Dec.-18, End Sem, Marks 5

2. Different components of database management systems like query processor, storage manager, transaction manager etc. are functional for processing the query submitted by user. Explain the functions of each component in view of getting query output.

SPPU : Oct.-18, In Sem, Marks 5

3. Draw the overall database system structure. Explain storage manager, transaction manager and query processor in detail.

SPPU : Dec.-19, End Sem, Marks 5

1.9 Data Models

SPPU : Nov.-19, Marks 5

- **Definition** : It is a collection of conceptual tools for describing data, relationships among data, semantics (meaning) of data and constraints.
- Data model is a **structure below the database**.
- Data model provides a way to describe the design of database at physical, logical and view level.
- There are various data models used in database systems and these are as follows -

(1) Relational model :

- o The relation model consists of **collection of tables** which stores data and also represents the relationship among the data.
- o The **table** is also known as **relation**.
- o The table contains one or more **columns** and each column has unique name.
- o Each table contains **record of particular type**, and each record type defines a fixed **number of fields or attributes**.

- **For example** – The following figure shows the relational model by showing the relationship between Student and Result database. For example – Student **Ram** lives in city **Chennai** and his marks are **78**. Thus the relationship between these two databases is maintained by the **SeatNo.** Column

Seat No	Name	City
101	Ram	Chennai
102	Shyam	Pune

SeatNo	Marks
101	78
102	95

Advantages :

- (i) **Structural independence** : Structural independence is an ability that allows us to make changes in one database structure without affecting other. The relational model have structural independence. Hence making required changes in the database is convenient in relational database model.
- (ii) **Conceptual simplicity** : The relational model allows the designer to simply focus on logical design and not on physical design. Hence relational models are conceptually simple to understand.
- (iii) **Query capability** : Using simple query language (such as SQL) user can get information from the database or designer can manipulate the database structure.
- (iv) **Easy design, maintenance and usage** : The relational models can be designed logically hence they are easy to maintain and use.

Disadvantages :

- i) Relational model **requires powerful hardware** and large data storage devices.
- ii) May lead to **slower processing time**.
- iii) Poorly designed systems lead to **poor implementation** of database systems.

(2) Entity relationship model :

- As the name suggests the entity relationship model uses collection of basic objects called **entities** and **relationships**.
- The entity is a thing or object in the real world.
- The entity relationship model is widely used in database design.
- For example - Following is a representation of Entity Relationship model in which the relationship **works_for** is between entities **Employee** and **Department**.

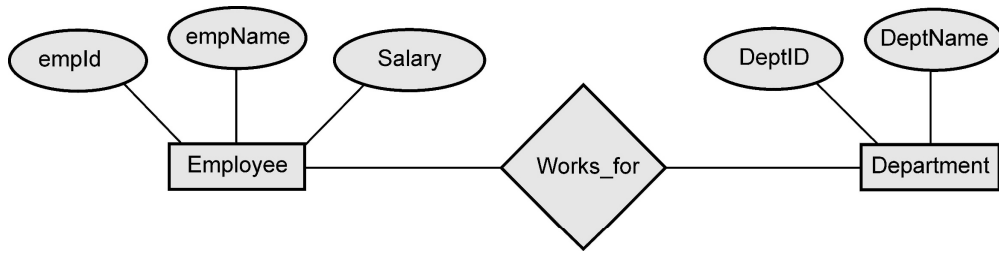


Fig. 1.9.1

Advantages :

- i) **Simple** : It is simple to draw ER diagram when we know entities and relationships.
- ii) **Easy to understand** : The design of ER diagram is very logical and hence they are easy to design and understand.
- iii) **Effective** : It is effective communication tool.
- iv) **Integrated** : The ER model can be easily integrated with Relational model.
- v) **Easy conversion** : ER model can be converted easily into other type of models.

Disadvantages :

- i) **Loss of information** : While drawing ER model some information can be hidden or lost.
- ii) **Limited relationships** : The ER model can represent limited relationships as compared to other models.
- iii) **No representation for data manipulation** : It is not possible to represent data manipulation in ER model.
- iv) **No industry standard** : There is no industry standard for notations of ER diagram.

(3) Object Based Data Model :

- o The **object oriented languages** like C++, Java, C# are becoming the dominant in software development.
- o This led to object based data model.
- o The object based data model combines **object oriented features** with relational data model.

Advantages :

- i) **Enriched modelling** : The object based data model has capability of modelling the real world objects.

- ii) **Reusability** : There are certain features of object oriented design such as inheritance, polymorphism which help in reusability.
- iii) **Support for schema evolution** : There is a tight coupling between data and applications, hence there is strong support for schema evolution.
- iv) **Improved performance** : Using object based data model there can be significant improvement in performance using object based data model.

Disadvantages :

- i) **Lack of universal data model** : There is no universally agreed data model for an object based data model, and most models lack a theoretical foundation.
- ii) **Lack of experience** : In comparison with relational database management the use of object based data model is limited. This model is more dependent on the skilled programmer.
- iii) **Complex** : More functionalities present in object based data model make the design complex.

(4) Semi-structured data model :

- o The semi-structured data model permits the **specification of data** where individual data items of same type may have **different sets of attributes**.
- o The Extensible Markup Language (**XML**) is widely used to represent semi-structured data model.

Advantages

- i) Data is not constrained by fixed schema.
- ii) It is **flexible**.
- iii) It is **portable**.

Disadvantage

- i) **Queries** are **less efficient** than other types of data model.

(5) Hierarchical Model

- In this model each entity has only **one parent** but can have **several children**. At the top of hierarchy there is only one node called **root**. Refer Fig. 1.9.2.
- This model represents the relationship in 1 : N types. That means one university can have multiple courses. One course can have multiple projects and so on.

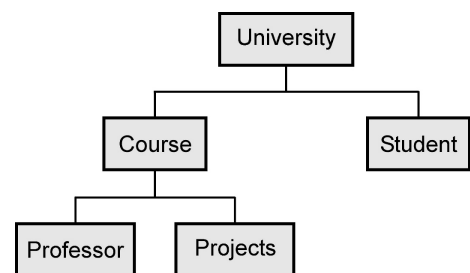


Fig. 1.9.2 Hierarchical model

Advantage

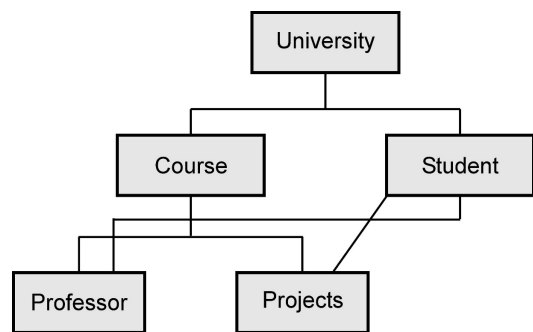
1. This model groups the data into tables and defines the relationship between the tables.

Disadvantages

1. For searching any data, we have to start from the root and move downwards and visit each child node. Thus traversing through each node is required.
2. For addition of some information about child node, sometimes the parent information need to be modified.
3. It fails to handle many to many relationship (M : N) efficiently.
It can cause duplication and data redundancy.

(6) Network Model

- This is enhanced version of hierarchical model. It overcomes the drawback of hierarchical model. It helps to address M : N relationship. That means, this model is not having single parent concept. Any child in this model can have multiple parents. Refer Fig. 1.9.3.
- The **main difference** between network model and hierarchical model is to allow many to many relationship.

**Fig. 1.9.3 Network model****Advantages**

1. **Capability to handle more Relationships** : Since the network model allows many to many relationship, it helps in modeling the real life situations.
2. **Ease of data access** : The data access is easier and flexible than hierarchical model.
3. **Data Integrity** : In network model every member is associated with some other member in the model.
4. **Conformance to Standards** : The network model structure can be designed as per the standards.

Disadvantages

1. **Complex to implement** : For all the records the pointers need to be maintained, hence the database structure becomes complex.
2. **Complicated Operations** : The simple operations such as insertion, deletion and modification becomes complex due to adjustment of multiple pointer.

3. **Difficult to change structure** : The structural changes are difficult.

1.10 Data Independence

SPPU : Nov.-19, Oct.-19, Aug.-17, Marks 5

- **Definition** : Data independence is an ability by which one can change the data at one level without affecting the data at another level. Here level can be **physical**, **conceptual** or **external**.
- Data independence is one of the important characteristics of database management system.
- By this property, the structure of the database or the values stored in the database can be easily modified by without changing the application programs.
- There are two types of data independence :

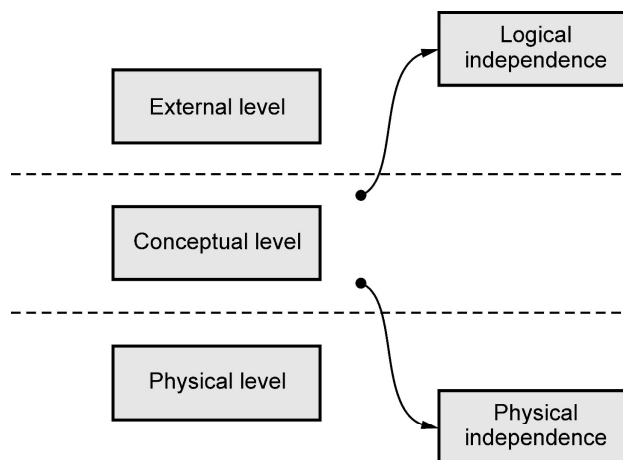


Fig. 1.10.1 Data independence

1. **Physical Independence** : This is a kind of data independence which allows the modification of physical schema without requiring any change to the conceptual schema. For example - if there is any change in memory size of database server then it will not affect the logical structure of any data object.
 2. **Logical Independence** : This is a kind of data independence which allows the modification of conceptual schema without requiring any change to the external schema. For example - Any change in the table structure such as addition or deletion of some column does not affect user views.
- By these data independence the time and cost acquired by changes in any one level can be reduced and abstract view of data can be provided to the user.

Review Question

1. Explain with example what is physical data independence. Also explain its importance.

SPPU : Aug.-17, In Sem, Marks 5

1.11 Database Users

SPPU : May 19, Nov.-18, Marks 5

There **four different types** of database system **users** differentiated by the way they interact with the system. Different types of user interfaces for different types of users are -

- i) **Naïve users** : This type of users interact with the system with the help of previously created program (known as application program). Typically a form interface is used by this type of user to interact with the system. For example - we may fill up the booking form for booking a ticket on an online system.
- ii) **Application programmers** : These are computer professionals who write application programs. Normally Rapid Application Development (RAD) tools are used to quickly design forms and reports.
- iii) **Sophisticated users** : These are the type of users who interact with the system without writing programs. These users may submit the database query to retrieve the desired information or tools from data analysis software. Database analysts fall in this category of database users.
- iv) **Specialized users** : Specialized users are sophisticated users who write specialized database application that does not fit into the traditional data-processing framework. Among these applications are computer aided-design systems, knowledge-base and expert systems etc.

Responsibilities of DBA

Database Administrator (DBA) is a person who has a **central control** over both data and programs that access data in DBMS. The **functions of DBA** are -

- i) **Schema definition** : DBA creates a database Schema using DDL statements.
- ii) **Schema and physical organization modification** : In order to improve the overall performance of database management system DBA carries out changes in schema or physical organization.
- iii) **Granting authorization for data access** : Granting authorization for data access means giving special permissions to the users for accessing the database. This task is done by DBA so that privacy of database can be maintained.
- iv) **Maintenance** : Maintenance involves different types of tasks such as monitoring jobs and their performances, taking periodic back-ups, making free space available for normal operations or upgrading disk space as per the requirements. These all tasks are carried out by DBA.

Example 1.11.1 Explain the problems that may arrive if the DBA does not discharge the responsibilities properly.

SPPU : May 19, End Sem, Marks 5

Solution : Following are the problems that may arrive if DBA does not discharge the responsibilities properly -

- 1) The database can not perform without file manger interaction. If nothing is stored in the files then obviously we can not retrieve anything.
- 2) The consistency in database must operations must be maintained. If it is not, then it will create major problems. For instance – account balance may go below the minimum allowed, employees can earn too much overtime and so on.
- 3) If authorization for the authentic user is not done, then unauthorized users may access the database or users authorized to access part of the database may be able to access parts of the database for which they lack authority.
- 4) Data can be lost permanently.
- 5) Consistency constraints may be violated despite proper integrity enforcement in each transaction. For example, incorrect bank balances might be reflected due to simultaneous withdrawals and deposits, and so on.

Part II: Data Modelling

1.12 Database Design and ER Model

SPPU : May-18, Nov.-17,19, Marks 5

- Data Modelling in database management system is based on Entity Relationship modelling(ER Model)
- Entity Relational model is a model for identifying entities to be represented in the database and representation of how those entities are related.
- ER data model represents the overall **logical structure** of database.
- The E-R model is very useful in mapping the meanings and interactions of real-world entities onto a conceptual schema or database.
- The ER model consists of three basic concepts –

1. Entities

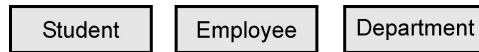
2. Relationships

3. Attributes

1.12.1 Entity and Entity Sets

- **Entity :** An entity is an object that exists and is distinguishable from other objects. For example - Student named “Poonam” is an entity and can be identified by her name. The entity can be concrete or abstract. The concrete entity can be - Person,

Book, Bank. The abstract entity can be like - holiday, concept entity is represented as a box.



- **Entity set** : The entity set is a set of entities of the same types. For example - All students studying in class X of the School. The entity set need not be disjoint. Each entity in entity set have the same set of attributes and the set of attributes will distinguish it from other entity sets. No other entity set will have exactly the same set of attributes.

1.12.2 Relationships

- **Relationship**: Relationship is an association among two or more entities.
- **Relationship Set**: The **relationship set** is a collection of similar relationships. For example - Following Fig. 1.12.1 shows the relationship **works_for** for the two entities **Employee** and **Departments**.

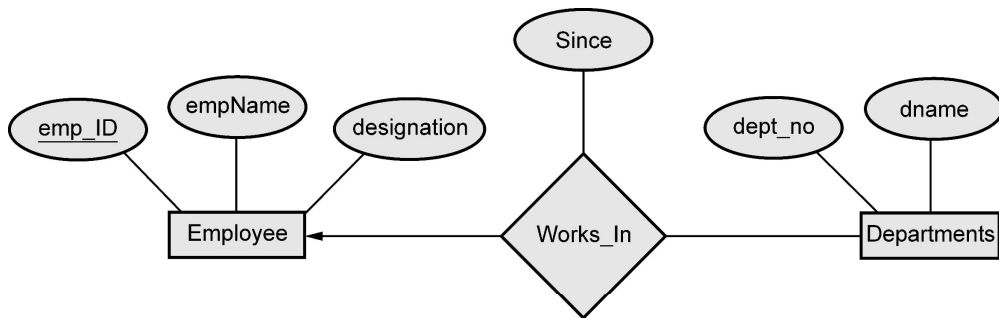


Fig. 1.12.1 : Relation set

- The association between entity sets is called as **participation**. that is, the entity sets E_1, E_2, \dots, E_n participate in relationship set R .
- The function that an entity plays in a relationship is called that entity's **role**.

1.12.3 Attributes

- Attributes define the properties of a data object of entity. For example: if student is an entity, his ID, name, address, date of birth, class are its attributes. The attributes help in determining the unique entity. Refer Fig. 1.12.2 for Student entity set with attributes - ID, name, address. Note that entity is shown by rectangular box and attributes are shown in oval. The primary key is underlined.

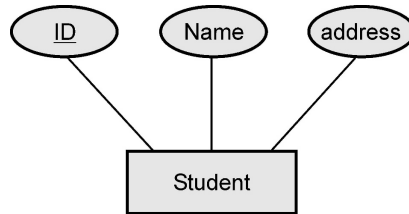


Fig. 1.12.2 : Student entity set with attributes

1.12.3.1 Types of Attributes

Following are the types of attributes -

1) Simple and Composite Attributes :

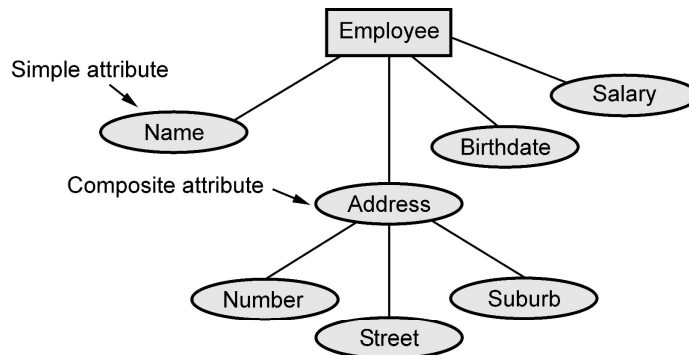
1) Simple attributes are attributes that are drawn from the atomic value domains

For example - Name = {Parth} ; Age = {23}

2) Composite attributes: Attributes that consist of a hierarchy of attributes

For example – Address may consists of “Number”, “Street” and “Suburb”

Hence, Address = {59 + 'JM Road' + 'ShivajiNagar'}

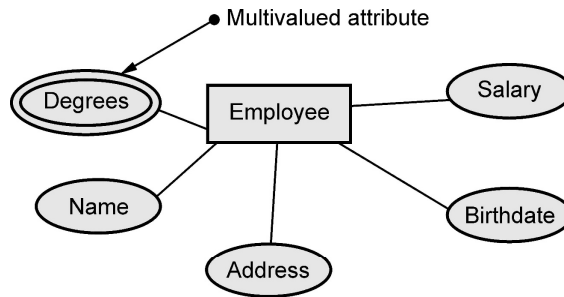


2) Single valued and multivalued :

- There are some attributes that can be represented using a single value. For example - StudentID attribute for a Student is specific only one studentID.

- Multivalued attributes : Attributes that have a set of values for each entity. It is represented by concentric ovals

For example - Degrees of a person: ' BSc' , 'MTech', 'PhD'



3) Derived attribute :

Derived attributes are the attributes that contain values that are calculated from other attributes. To represent derived attribute there is dotted ellipse inside the solid ellipse. For example – Age can be derived from attribute DateOfBirth. In this situation, DateOfBirth might be called Stored Attribute.

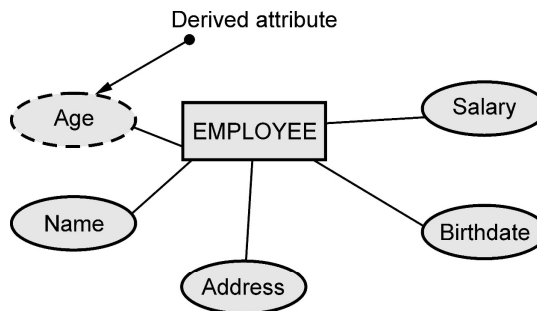


Fig. 1.12.3

1.13 Constraints

- Relationship types have certain rules that limit the possible combination of entities that can take part in relationship. These rules or restrictions are called **structural constraints**.
- The common type of structural constraint is represented by the **cardinality ratio**.
- The **cardinality ratio** for a binary relationship specifies the **maximum** number of relationship instances that an entity can participate in.

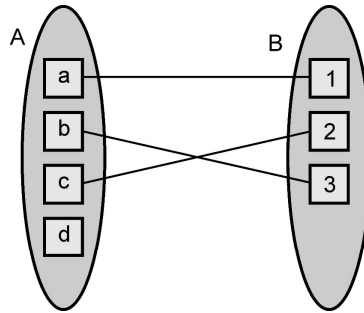
1.13.1 Types of Cardinality

Mapping Cardinality represents the number of entities to which another entity can be associated via a relationship set.

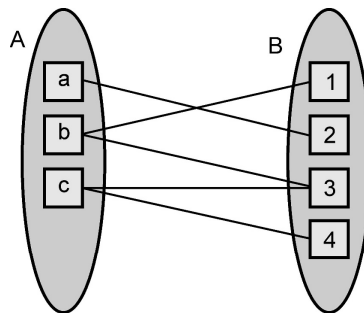
The mapping cardinalities are used in representing the binary relationship sets.

Various types of mapping cardinalities are -

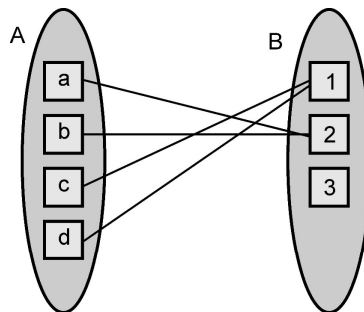
1. **One to One** : An entity A is associated with at least one entity on B and an entity B is associated with at one entity on A. This can be represented as



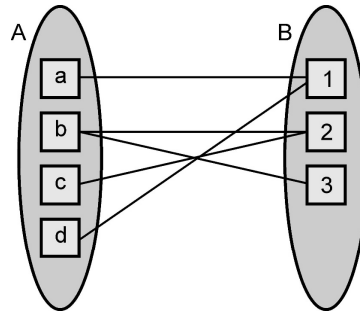
2. **One to Many** : An entity in A is associated with any number of entities in B. An entity in B, however, can be associated with at most one entity in A.



3. **Many to One** : An entity in A is associated with at most one entity in B. An entity in B, however, can be associated with any number of entities in A.



4. **Many to many** : An entity in A is associated with any number (zero or more) of entities in B, and an entity in B is associated with any number (zero or more) of entities in A.



1.14 Keys

SPPU : May-19, Oct.-19, Marks 5

- Keys are used to identify entities uniquely from the given entity set.
- A key can be an attribute or a set of attributes that help us to identify the entity uniquely.
- Keys also help to identify relationships uniquely, and thus distinguish relationships from each other.
- The primary key of an entity set allows us to distinguish among the various entities of the set.
- For example - If a Student table contains the information about various students as given below -

RollNo	Name	City	Course
101	Ram	Pune	Computer
102	Sita	Pune	Electronics
103	Laxman	Chennai	Mechanical

In above table, **RollNo** is a primary key because, it uniquely identifies the student record.

Review Question

1. Distinguish between super key, candidate key and primary key.

SPPU : May-19, End Sem, Oct.-19, In Sem, Marks 5

1.15 Design Process

Following are the six steps of database design process. The ER model is most relevant to first three steps

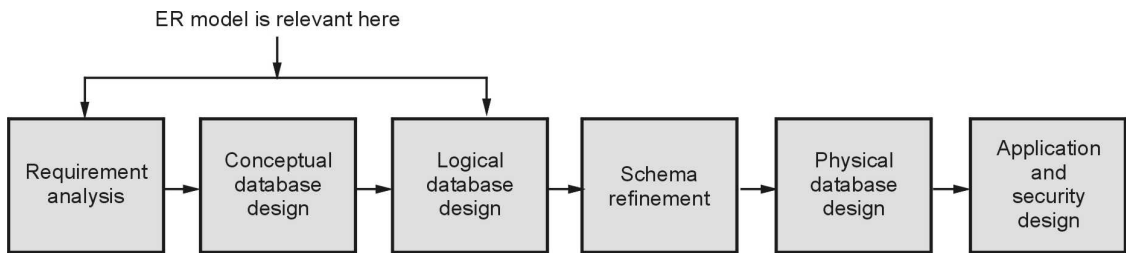


Fig. 1.15.1 : Database design process

Step 1 : Requirement analysis :

- In this step, it is necessary to understand what data need to be stored in the database, what applications must be built, what are all those operations that are frequently used by the system.
- The requirement analysis is an **informal** process and it requires proper **communication** with user groups.
- There are several methods for **organizing and presenting information** gathered in this step.
- Some automated tools can also be used for this purpose.

Step 2 : Conceptual database design :

- This is a steps in which **E-R Model** i.e. Entity Relationship model is built.
- E-R model is a **high level data model** used in database design.
- The **goal** of this design is to create a simple description of data that matches with the requirements of users.

Step 3 : Logical database design :

- This is a step in which ER model in **converted to relational database schema**, sometimes called as the logical schema in the relational data model.

Step 4 : Schema refinement :

- In this step, **relational database schema is analyzed** to identify the potential problems and to refine it.
- The schema refinement can be done with the help of **normalizing and restructuring the relations**.

Step 5 : Physical database design :

- In this step, the design of database is **refined further**.

- The tasks that are performed in this step are - building **indexes** on tables and **clustering** tables, redesigning some parts of schema obtained from earlier design steps.

Step 6 : Application and security design :

- Using design methodologies like UML(Unified Modeling Language) the design of the database can be accomplished.
- The **role of each entity** in every process must be reflected in the application task.
- For each role, there must be the provision for **accessing** the some part of database and **prohibition of access** to some other part of database.
- Thus some **access rules** must be enforced on the application(which is accessing the database) to protect the **security features**.

1.16 ER Diagram


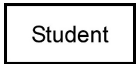
SPPU : Aug.-17, Marks 2

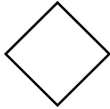

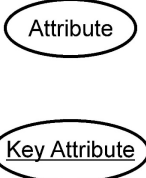
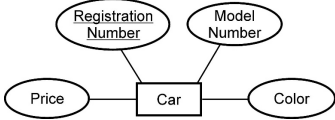

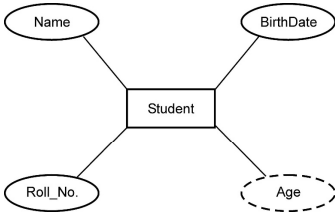

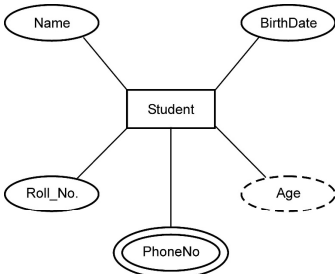

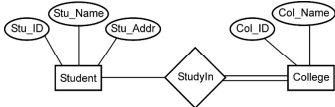
- Entity Relational model is a model for identifying entities to be represented in the database and representation of how those entities are related.
- The ER data model specifies enterprise schema that represents the overall **logical structure** of a database graphically.
- E-R diagrams are used to model real-world objects like a person, a car, a company and the relation between these real-world objects.

Features of ER model

- E-R diagrams are used to represent E-R model in a database, which makes them easy to be converted into relations (tables).
- E-R diagrams provide the purpose of real-world modeling of objects which makes them intently useful.
- E-R diagrams require **no technical knowledge** and no hardware support.
- These diagrams are very **easy to understand** and easy to create even by a naive user.
- It gives a standard solution of **visualizing** the **data logically**.

Various Components used in ER Model are -

Component	Symbol	Example
Entity : Any real-world object can be represented as an entity about which data can be stored in a database. All the real world objects like a book, an organization, a		

<p>product, a car, a person are the examples of an entity.</p>		
<p>Relationship : Rhombus is used to setup relationships between two or more entities.</p>		
<p>Attribute : Each entity has a set of properties. These properties of each entity are termed as attributes. For example, a car entity would be described by attributes such as price, registration number, model number, color etc</p>		
<p>Derived attribute : Derived attributes are those which are derived based on other attributes, for example, age can be derived from date of birth.</p> <p>To represent a derived attribute, another dotted ellipse is created.</p>		
<p>Multivalued attribute : An attribute that can hold multiple values is known as multivalued attribute. We represent it with double ellipses in an E-R Diagram. E.g. A person can have more than one phone numbers so the phone number attribute is multivalued.</p>		
<p>Total participation : Each entity is involved in the relationship. Total participation is represented by double lines.</p>		

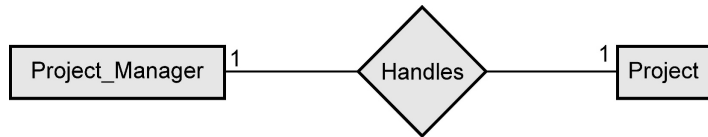
1.17 Conventions

SPPU : Oct.-19, Marks 2

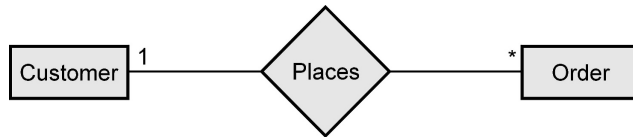
1.17.1 Mapping Cardinality Representation

There are four types of relationships that are considered for key constraints.

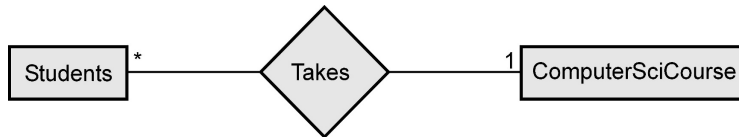
- i) **One to one relation** : When entity A is associated with at the most one entity B then it shares one to one relation. For example - There is one project manager who manages only one project.



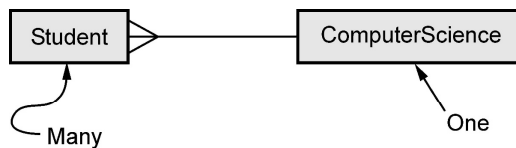
ii) **One to many** : When entity A is associated with more than one entities at a time then there is one to many relation. For example - One customer places order at a time.



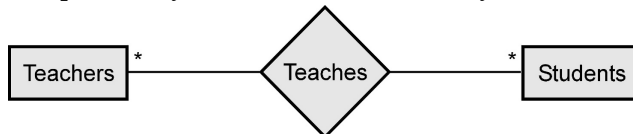
iii) **Many to one** : When more than one entities are associated with only one entity then there is many to one relation. For example - Many student take a ComputerSciCourse.



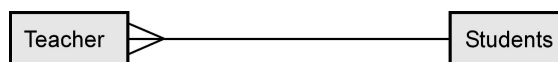
Alternate representation can be



iv) **Many to many** : When more than one entities are associated with more than one entities. For example - Many teachers can teach many students.

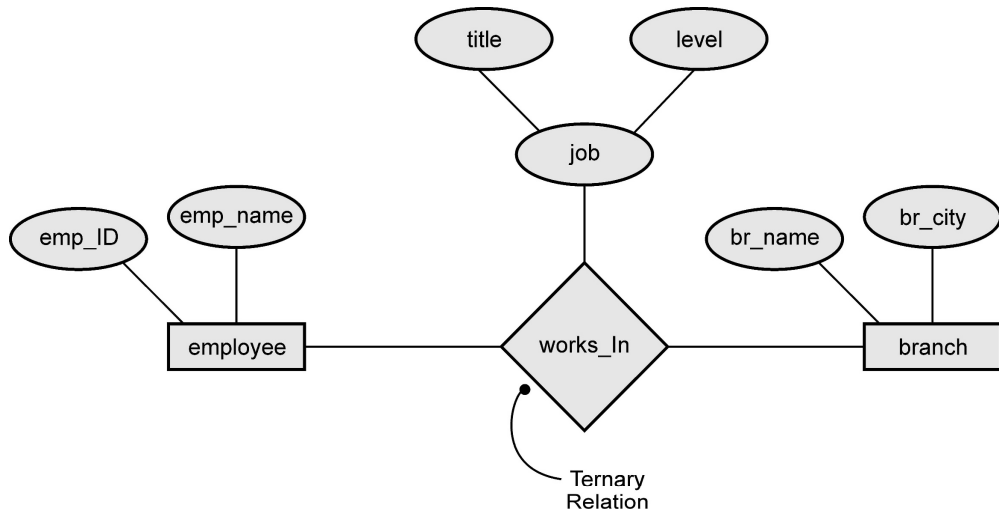


Alternate representation can be



1.17.2 Ternary Relationship

The relationship in which three entities are involved is called ternary relationship. For example -



1.17.3 Weak Entity Set

- A weak entity is an entity that cannot be uniquely identified by its attributes alone. The entity set which does not have sufficient attributes to form a primary key is called as weak entity set.

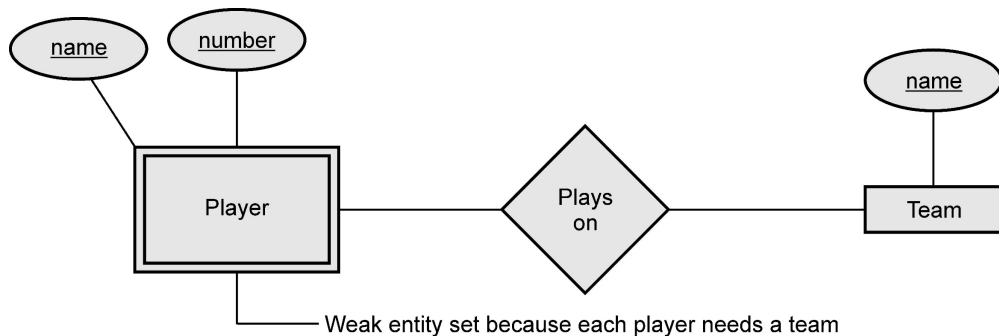


Fig. 1.17.1 : Weak entity set

- **Strong Entity Set**

The entity set that has primary key is called as strong entity set

Weak entity rules

- A weak entity set has **one or more many-one** relationships to other (supporting) entity sets.
- The key for a weak entity set is its own underlined attributes and the keys for the supporting entity sets. For example - player-number and team-name is a key for Players.

Difference between Strong and Weak Entity Set

Sr. No.	Strong entity set	Weak entity set
1	It has its own primary key.	It does not have sufficient attribute to form a primary key on its own.
2.	It is represented by rectangle	It is represented by double rectangle.
3.	It represents the primary key which is underlined.	It represents the partial key or discriminator which is represented by dashed underline.
4.	The member of strong entity set is called as dominant entity set	The member of weak entity set is called subordinate entity set.
5.	The relationship between two strong entity sets is represented by diamond symbol.	The relationship between strong entity set and weak entity set is represented by double diamond symbol.
6.	The primary key is one of the attributes which uniquely identifies its member.	The primary key of weak entity set is a combination of partial key and primary key of the strong entity set.

1.18 Design Issues

- The ER diagram can be designed in several different ways for representing the same system application. Different representations may not always be exactly equivalent.

(1) Use of Entity Vs. Attribute

- Use of entity or attribute in the ER diagram depends upon the real-world application. For example - Following is an example, in which phone_number is represented as attribute. But we can represent phone_number as entity instead of attribute. The advantage of such representation is that it is possible to maintain the multiple phone numbers for a customer. Moreover, we can maintain some extra information such as customer's **location** if the phone_number is represented as entity instead of attribute.

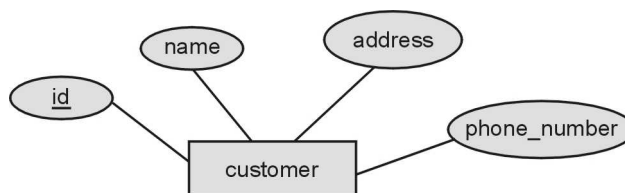


Fig. 1.18.1 : Representation of Phone_number as attribute

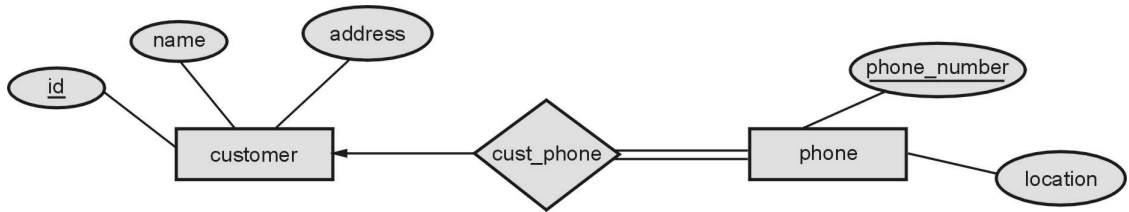
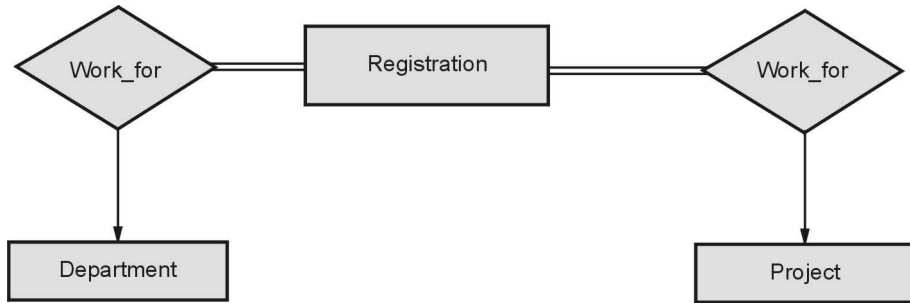


Fig. 1.18.2 : Representation of Phone_number as entity

(2) Use of Entity Set Vs. Relationship Sets

- Representing a particular as entity set or relationship is a common problem, many times designer face. For example as shown in the following figure , an – Employee works for a Department and an Employee works for a project, These two relationships can be represented by making Registration as an entity set and representing works_for relationship set for both department and project.

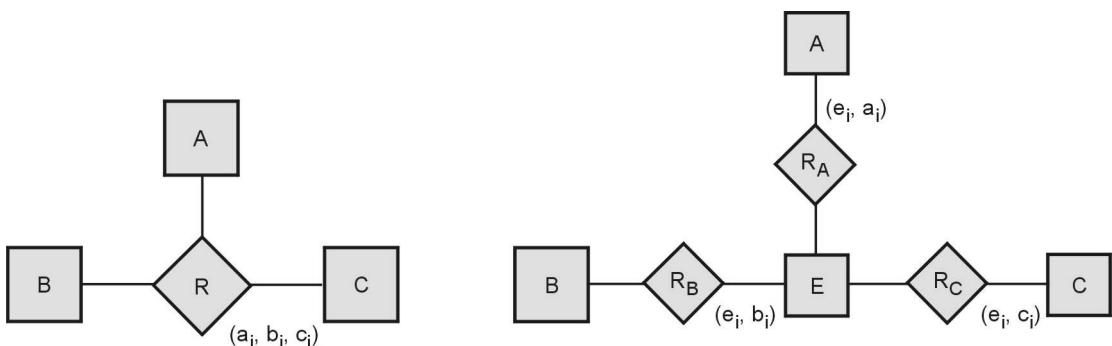


(3) Binary Vs. n-ary Relationship Sets

Generally, the relationships described in the databases are binary relationships. However, non-binary relationships can be represented by several binary relationships.

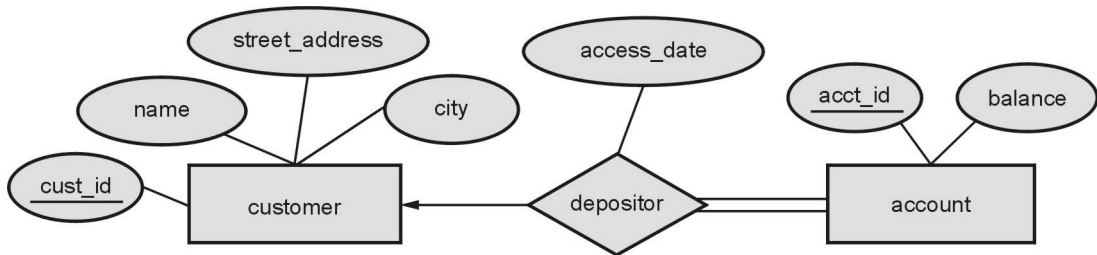
We can specify some mapping cardinalities on relationships with degree > 2

For example –



(4) Placing Relationship Attributes

- The relationship set can have descriptive attribute. For example – in the following fig. the relationship depositor has descriptive attribute named **access_date**. The decision of placing the specified attribute as a relationship or entity attribute should possess the characteristics of the real world enterprise that is being modelled.



1.19 Extended E-R Features

1.19.1 Specialization and Generalization

- Some entities have relationships that form hierarchies. For instance, Employee can be an hourly employee or contracted employee.
- In this relationship hierarchies, some entities can act as superclass and some other entities can act as subclass.
- Superclass** : An entity type that represents a general concept at a high level, is called superclass.
- Subclass** : An entity type that represents a specific concept at lower levels, is called subclass.
- The subclass is said to inherit from superclass. When a subclass inherits from one or more superclasses, it inherits all their attributes. In addition to the inherited attributes, a subclass can also define its own specific attributes.
- The process of making subclasses from a general concept is called **specialization**. This is **top-down** process. In this process, the sub-groups are identified within an entity set which have attributes that are not shared by all entities.
- The process of making superclass from subclasses is called **generalization**. This is a **bottom up** process. In this process multiple sets are synthesized into high level entities.
- The symbol used for specialization/ Generalization is



- **For example** – There can be two subclass entities namely **Hourly_Emps** and **Contract_Emps** which are subclasses of **Employee** class. We might have attributes **hours_worked** and **hourly_wage** defined for **Hourly_Emps** and an attribute **contractid** defined for **ContractEmps**.

Therefore, the attributes defined for an **Hourly_Emps** entity are the attributes for **Employees** plus **Hourly_Emps**. We say that the attributes for the entity set **Employees** are inherited by the entity set **Hourly_Emps** and that **Hourly-Emps ISA** (read is a) **Employees**. It can be represented by following Fig. 1.19.1.

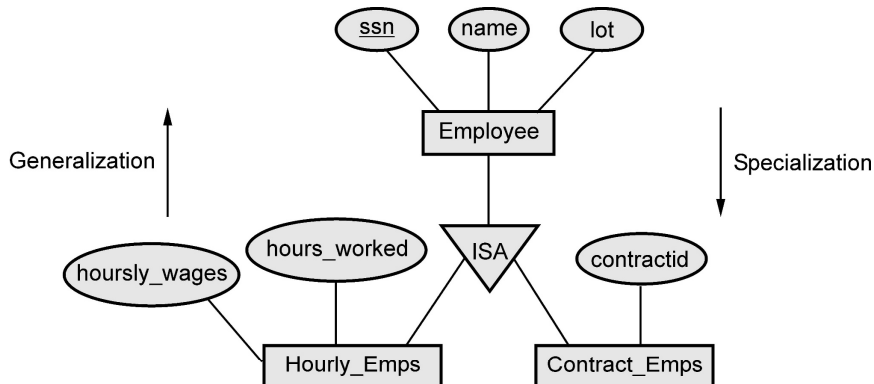


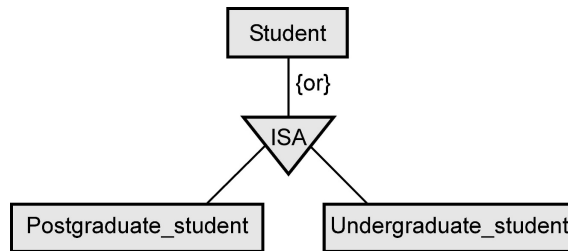
Fig. 1.19.1 Example of Generalization and Specialization

1.19.2 Constraints on Specialization / Generalization

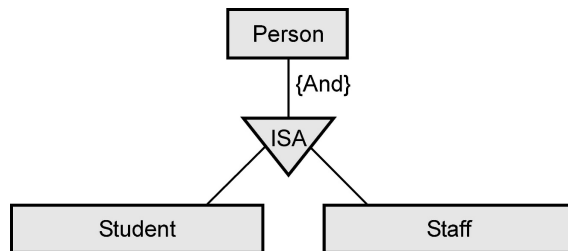
There are four types of constraints on specialization/generalization relationship. These are -

- 1) **Membership constraints** : This is a kind of constraints that involves determining which entities can be members of a given lower-level entity. There are two types of membership constraints -
 - i) **Condition defined** : In condition-defined lower-level entity sets, membership is evaluated on the basis of whether or not an entity satisfies an explicit condition or predicate. For example - Consider the high-level entity Set **Employee** that has attribute **Employee_type**. All Employee entities are evaluated on defining **Employee_type** attribute. All entities that satisfy the condition student type = "ContractEmployee" are included in Contracted Employee. Since all the lower-level entities are evaluated on the basis of the same attribute this type of generalization is said to be **attribute-defined**.
 - ii) **User defined** : This is kind of entity set that in which the membership is manually defined.
- 2) **Disjoint constraints** : The disjoint constraint only applies when a superclass has

more than one subclass. If the subclasses are disjoint, then an entity occurrence can be a member of only one of the subclasses. For entity Student has either **Postgraduate_Student** entity or **Undergraduate_Student**

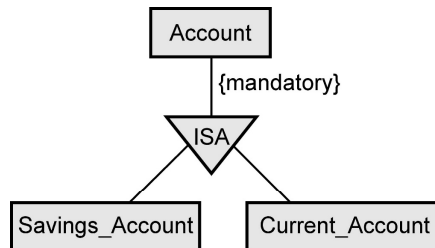


- 3) **Overlapping** : When some entity can be a member of more than one subclasses. For example - Person can be both a Student or a Staff. The **And** can be used to represent this constraint.

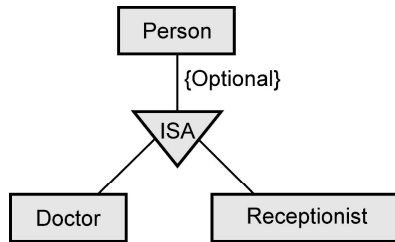


- 4) **Completeness** : It specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within the generalization/specialization. This constraint may be one of the following -

- i) **Total generalization or specialization** : Each higher-level entity must belong to a lower-level entity set. For example - Account in the bank must either Savings account or Current Account. The **mandatory** can be used to represent this constraint.



- ii) **Partial generalization or specialization** : Some higher-level entities may not belong to any lower-level entity set.



1.19.3 Aggregation

A feature of the entity relationship model that allows a relationship set to participate in another relationship set. This is indicated on an ER diagram by drawing a dashed box around the aggregation.

For example - We treat the relationship set work and the entity sets employee and project as a higher-level entity set called work.

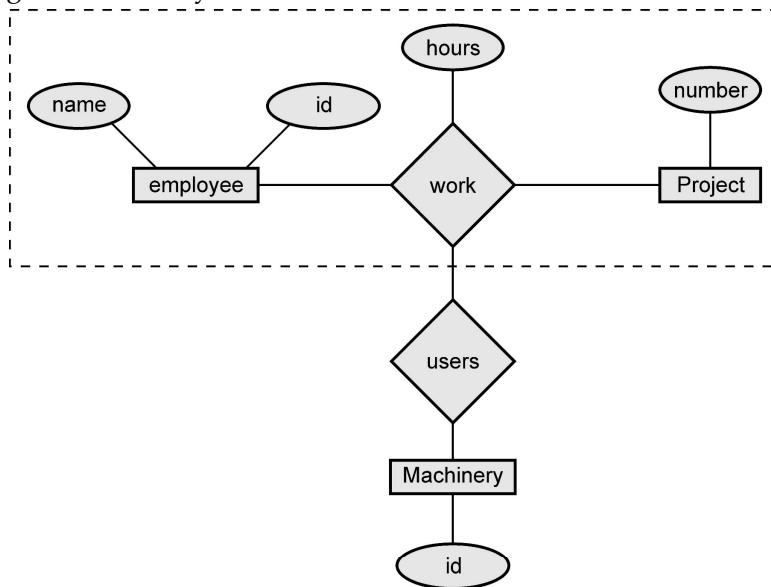


Fig. 1.19.2 : ER model with aggregation

1.20 Converting ER and EER Diagram into Tables

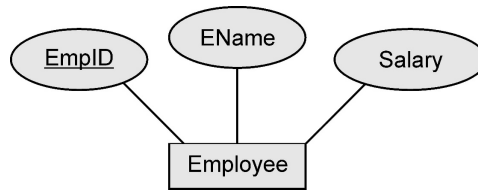
SPPU : Nov.-18, Marks 4

In this section we will discuss how to map various ER model constructs to Relational Model construct.

1.20.1 Mapping of Entity Set to Relationship

- An entity set is mapped to a relation in a straightforward way.
- Each attribute of entity set becomes an attribute of the table.
- The primary key attribute of entity set becomes an entity of the table.

- For example - Consider following ER diagram.



The converted employee table is as follows -

EmpID	EName	Salary
201	Poonam	30000
202	Ashwini	35000
203	Sharda	40000

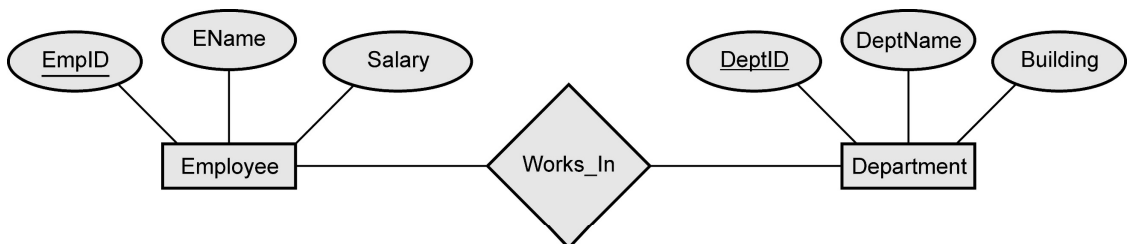
The SQL statement captures the information for above ER diagram as follows -

```
CREATE TABLE Employee( EmpID CHAR(11),
  EName CHAR(30),
  Salary INTEGER,
  PRIMARY KEY(EmpID))
```

1.20.2 Mapping Relationship Sets(without Constraints) to Tables

- Create a table for the relationship set.
- Add all primary keys of the participating entity sets as fields of the table.
- Add a field for each attribute of the relationship.
- Declare a primary key using all key fields from the entity sets.
- Declare foreign key constraints for all these fields from the entity sets.

For example - Consider following ER model

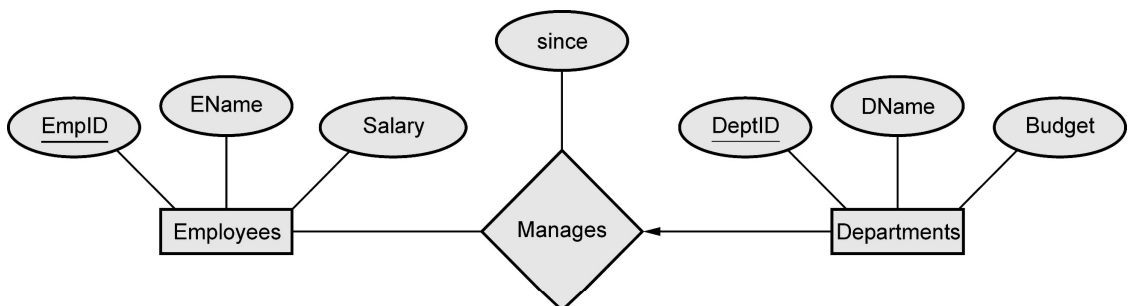


The SQL statement captures the information for relationship present in above ER diagram as follows -

```
CREATE TABLE Works_In (EmpID CHAR(11),
    DeptID CHAR(11),
    EName CHAR(30),
    Salary INTEGER,
    DeptName CHAR(20),
    Building CHAR(10),
    PRIMARY KEY(EmpID,DeptID),
    FOREIGN KEY (EmpID) REFERENCES Employee,
    FOREIGN KEY (DeptID) REFERENCES Department
)
```

1.20.3 Mapping Relationship Sets(With Constraints) to Tables

- If a relationship set involves **n** entity sets and some **m** of them are linked **via arrows** in the ER diagram, the key for anyone of these **m** entity sets **constitutes a key** for the relation to which the relationship set is mapped.
- Hence we have **m** candidate keys, and one of these should be designated as the **primary key**.
- There are two approaches used to convert a relationship sets with key constraints into table.
- **Approach 1 :**
 - By this approach the relationship associated with more than one entities is separately represented using a table. For example - Consider following ER diagram. Each **Dept has at most one manager**, according to the **key constraint on Manages**.



Here the constraint is each department has at the most one manager to manage it. Hence no two tuples can have same DeptID. Hence there can be a separate table named **Manages** with DeptID as Primary Key. The table can be defined using following SQL statement

```
CREATE TABLE Manages(EmpID CHAR(11),
                    DeptID INTEGER,
                    Since DATE,
                    PRIMARY KEY(DeptID),
                    FOREIGN KEY (EmpID) REFERENCES Employees,
                    FOREIGN KEY (DeptID) REFERENCES Departments)
```

- **Approach 2 :**

- In this approach , it is preferred **to translate a relationship set with key constraints.**
- It is a superior approach because, it avoids creating a distinct table for the relationship set.
- The idea is to include the information about the relationship set in the table corresponding to the entity set with the key, taking advantage of the key constraint.
- This approach eliminates the need for a separate Manages relation, and queries asking for a department's manager can be answered without combining information from two relations.
- The only drawback to this approach is that space could be wasted if several departments have no managers.
- The following SQL statement, defining a **Dep_Mgr** relation that captures the information in both **Departments** and **Manages**, illustrates the second approach to translating relationship sets with key constraints :

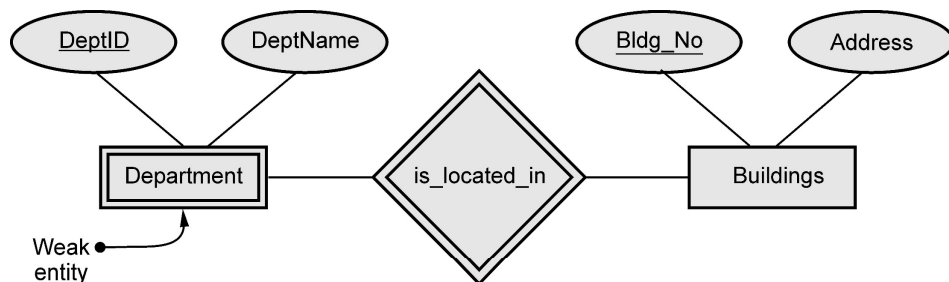
```
CREATE TABLE Dep_Mgr ( DeptID INTEGER,
                    DName CHAR(20),
                    Budget REAL,
                    EmpID CHAR (11),
                    since DATE,
                    PRIMARY KEY (DeptID),
                    FOREIGN KEY (EmpID) REFERENCES Employees)
```

1.20.4 Mapping Weak Entity Sets to Relational Mapping

A weak entity can be identified uniquely only by considering the primary key of another (owner) entity. Following steps are used for mapping Weak Entity Set to Relational Mapping

- Create a table for the weak entity set.
- Make each attribute of the weak entity set a field of the table.
- Add fields for the primary key attributes of the identifying owner.
- Declare a foreign key constraint on these identifying owner fields.
- Instruct the system to automatically delete any tuples in the table for which there are no owners

For example - Consider following ER model



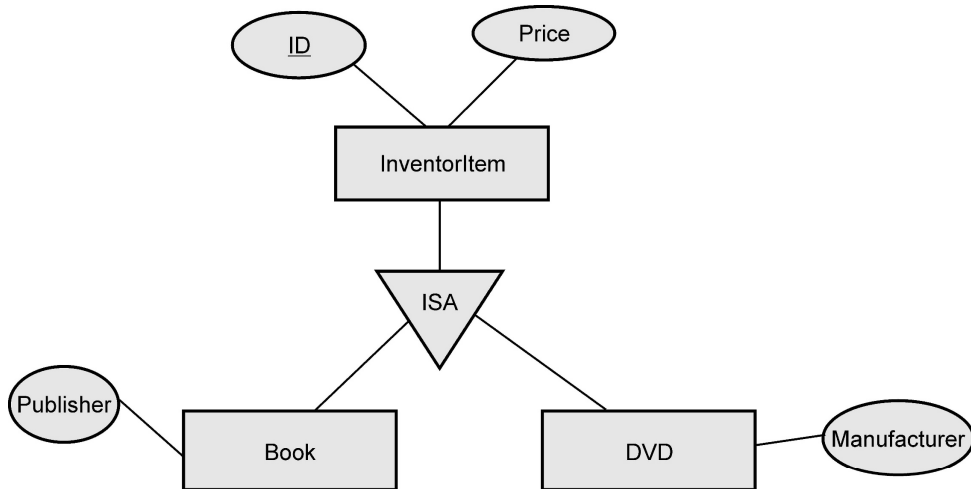
Following SQL Statement illustrates this mapping

```

CREATE TABLE Department(DeptID CHAR(11),
    DeptName CHAR(20),
    Bldg_No CHAR(5),
    PRIMARY KEY (DeptID,Bldg_No),
    FOREIGN KEY(Bldg_No) References Buildings on delete cascade
)
  
```

1.20.5 Mapping of Specialization / Generalization(EER Construct) to Relational Mapping

The specialization/Generalization relationship(Enhanced ER Construct) can be mapped to database tables(relations) using three methods. To demonstrate the methods, we will take the – InventoryItem, Book, DVD



Method 1 : All the entities in the relationship are mapped to individual tables

```

InventoryItem(ID , Price)
Book(ID,Publisher)
DVD(ID, Manufacturer)
  
```

Method 2 : Only subclasses are mapped to tables. The attributes in the superclass are duplicated in all subclasses. For example -

```

Book(ID, Price,Publisher)
DVD(ID, Price,Manufacturer)
  
```

Method 3 : Only the superclass is mapped to a table. The attributes in the subclasses are taken to the superclass. For example -

```

InventoryItem(ID , Price,Publisher,Manufacturer)
  
```

This method will introduce **null** values. When we insert a **Book** record in the table, the **Manufacturer** column value will be null. In the same way, when we insert a **DVD** record in the table, the **Publisher** value will be null.

Review Question

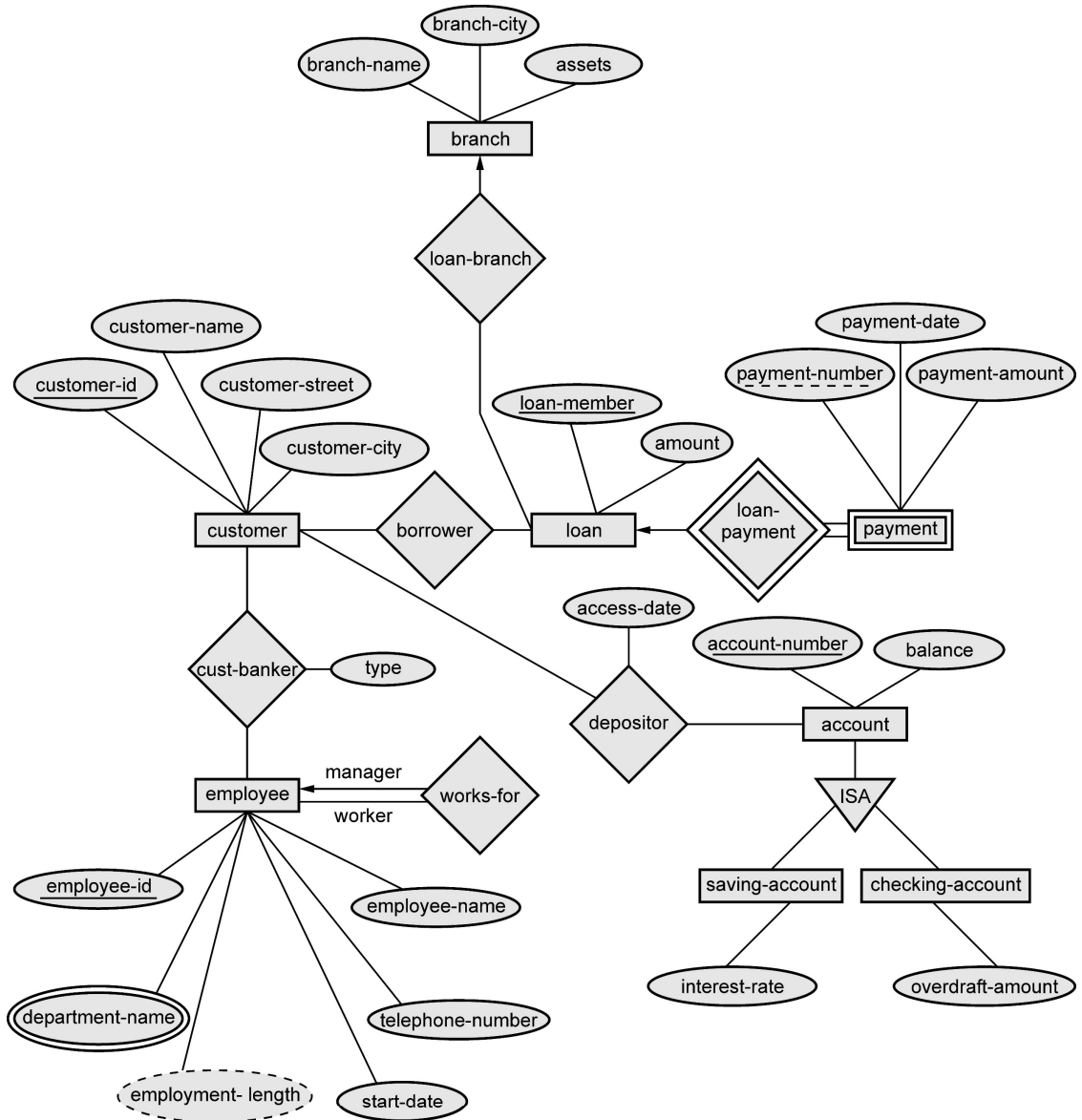
1. Writing short note on – Mapping ISA relationship of E-R diagram to tables

SPPU : Nov.-18, End Sem, Marks 4

1.21 Examples based on ER Diagram

Example 1.21.1 Draw an ER diagram for banking system(Home-Loan Application)

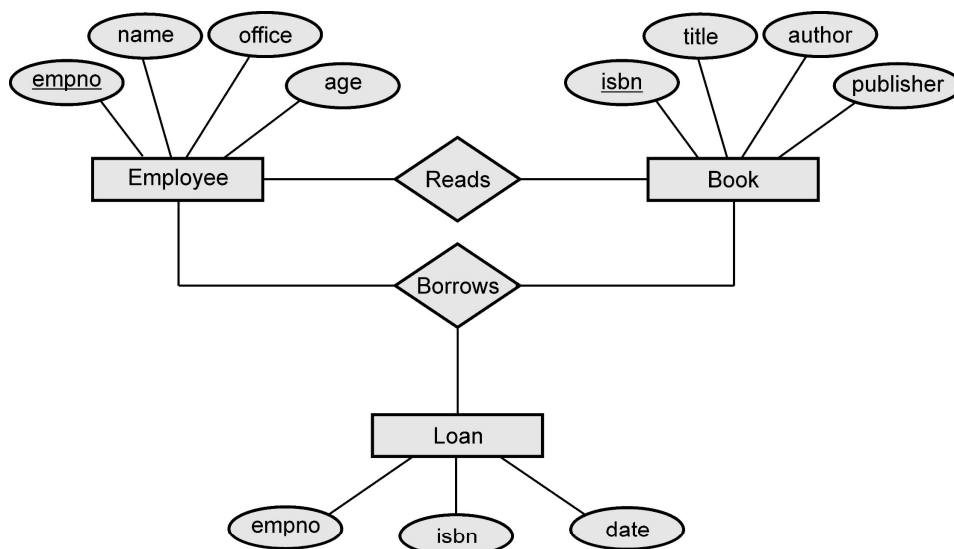
Solution :



Example 1.21.2 Consider the relation schema given in following Figure. Design and draw an ER diagram that capture the information of this schema.

Employee(empno,name,office,age)
Books(isbn,title,authors,publisher)
Loan(empno,isbn,date)

Solution :



Example 1.21.3 A car rental company maintains a database for all vehicles in its current fleet. For all vehicles, it includes the vehicle identification number license number, manufacturer, model, date of purchase and color. Special data are included for certain types of vehicles.

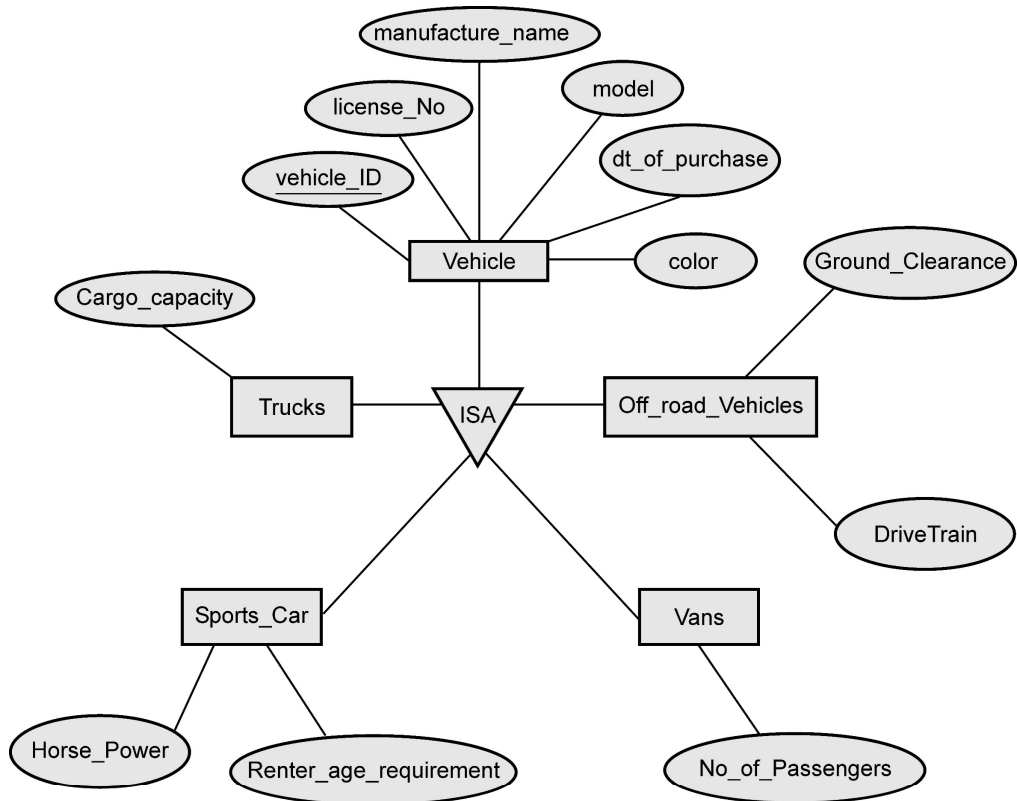
Trucks : Cargo capacity

Sports cars : horsepower, renter age requirement

Vans : number of passengers

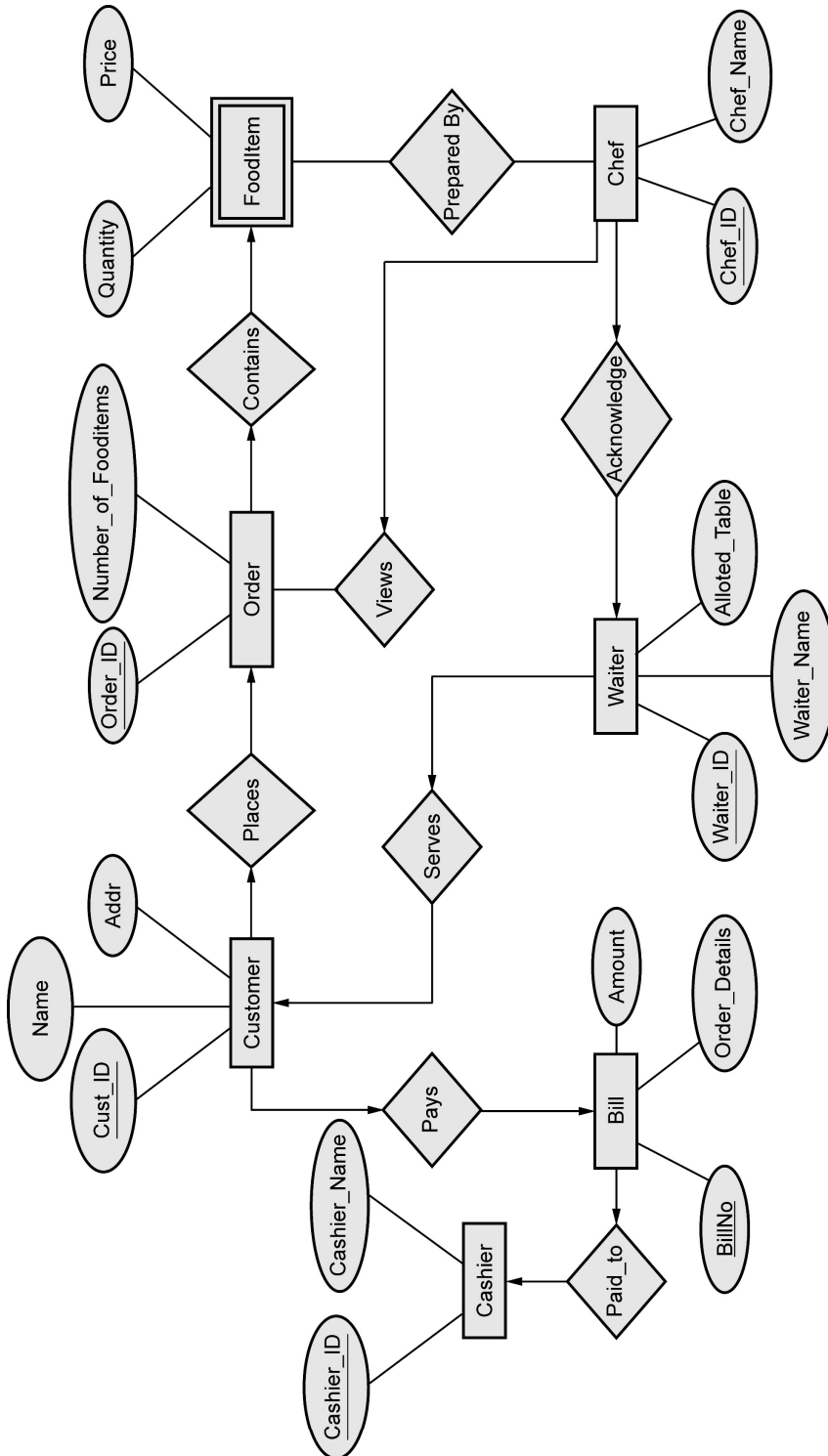
Off-road vehicles : ground clearance, drivetrain (four-or two-wheel drive)

Construct an ER model for the car rental company database.

Solution :

Example 1.21.4 Draw E-R diagram for the "Restaurant Menu Ordering System", which will facilitate the food items ordering and services within a restaurant. The entire restaurant scenario is detailed as follows. The customer is able to view the food items menu, call the waiter, place orders and obtain the final bill through the computer kept in their table. The Waiters through their wireless tablet PC are able to initialize a table for customers, control the table functions to assist customers, orders, send orders to food preparation staff (chef) and finalize the customer's bill. The Food preparation staffs (chefs), with their touch-display interfaces to the system, are able to view orders sent to the kitchen by waiters. During preparation they are able to let the waiter know the status of each item, and can send notifications when items are completed. The system should have full accountability and logging facilities, and should support supervisor actions to account for exceptional circumstances, such as a meal being refunded or walked out on.

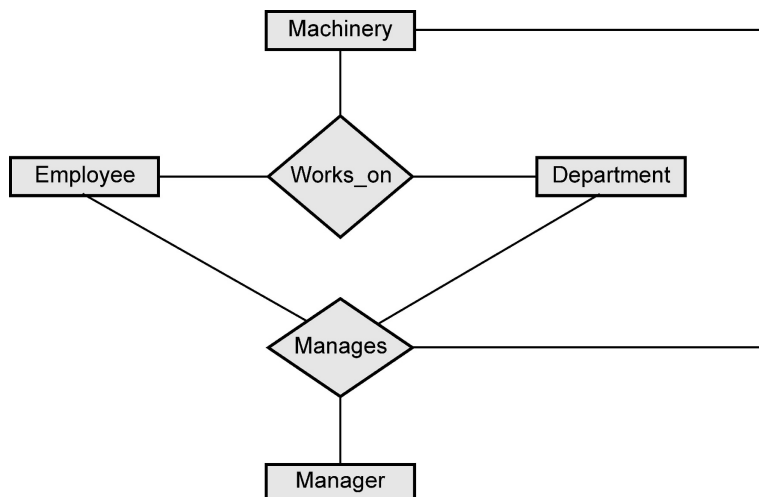
Solution :



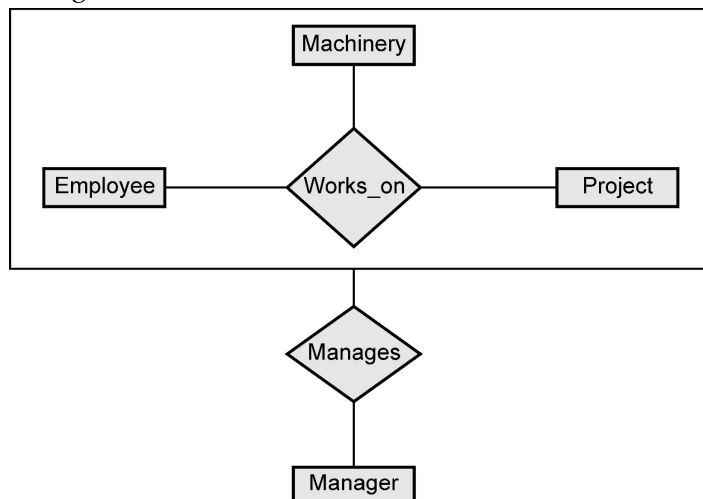
Example 1.21.5 What is aggregation in ER model ? Develop an ER diagram using aggregation that captures following information : Employees work for projects. An employee working for particular project uses various machinery. Assume necessary attributes. State any assumptions you make. Also discuss about the ER diagram you have designed.

Solution : Aggregation : Refer section 1.19.3.

ER Diagram : The ER diagram for above described scenario can be drawn as follows -



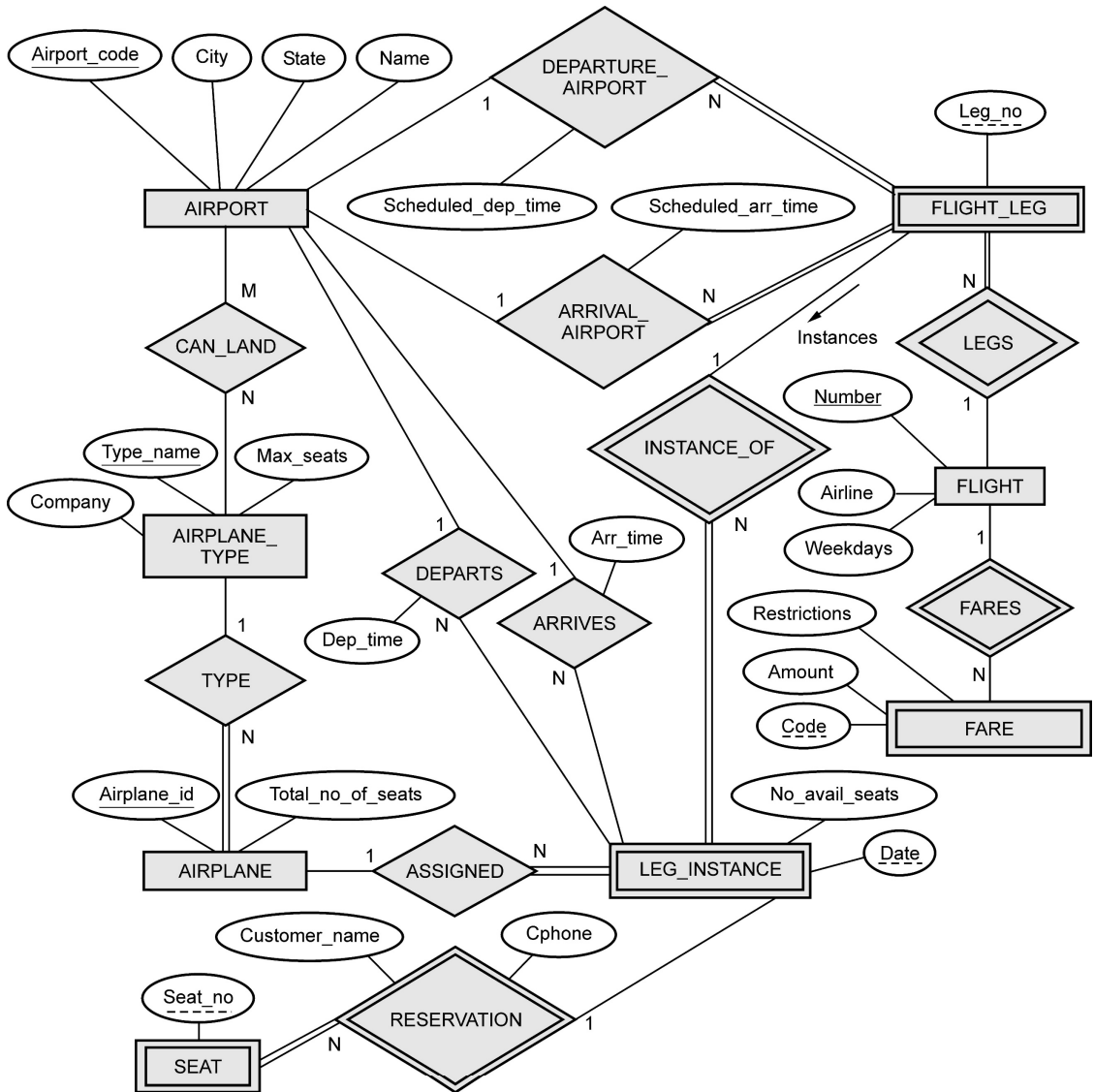
The above ER model contains the redundant information, because every Employee, Project, Machinery combination in **works_on** relationship is also considered in **manages** relationship. To avoid this redundancy problem we can make use of aggregation relationship in ER diagram as follows -



We can then create a binary relationship **manages** for between **Manager** and **(Employee, Project, Machinery)**.

Example 1.21.6 Draw an ER diagram of Airline reservation system taking into account at least five entities. Indicate all keys, constraints and assumptions that are made

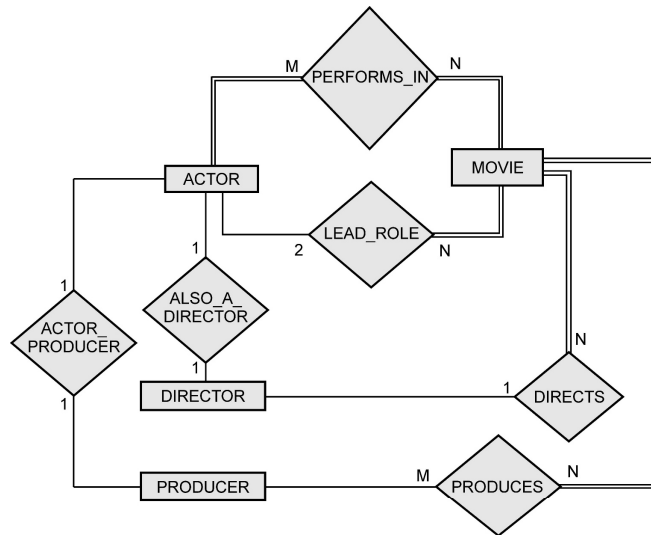
Solution :



A LEG is nonstop portion of flight. The LEG_INSTANCE is a particular occurrence of a LEG on particular date.

Example 1.21.7 Draw an ER diagram of movie database. Assume your own entities (minimum 4) attributes and relationships.

Solution :



Example 1.21.8 Draw an ER diagram to represent the Election Information system based on the following description.

In Indian national election, a state is divided into a number of constituencies depending upon the population of the state. Several candidates contest elections in each constituency. Record the number of votes obtained by each candidate. The system also maintains the voter list and a voter normally belongs to a particular constituency.

Note that the party details must also be taken care in the design.

Solution :

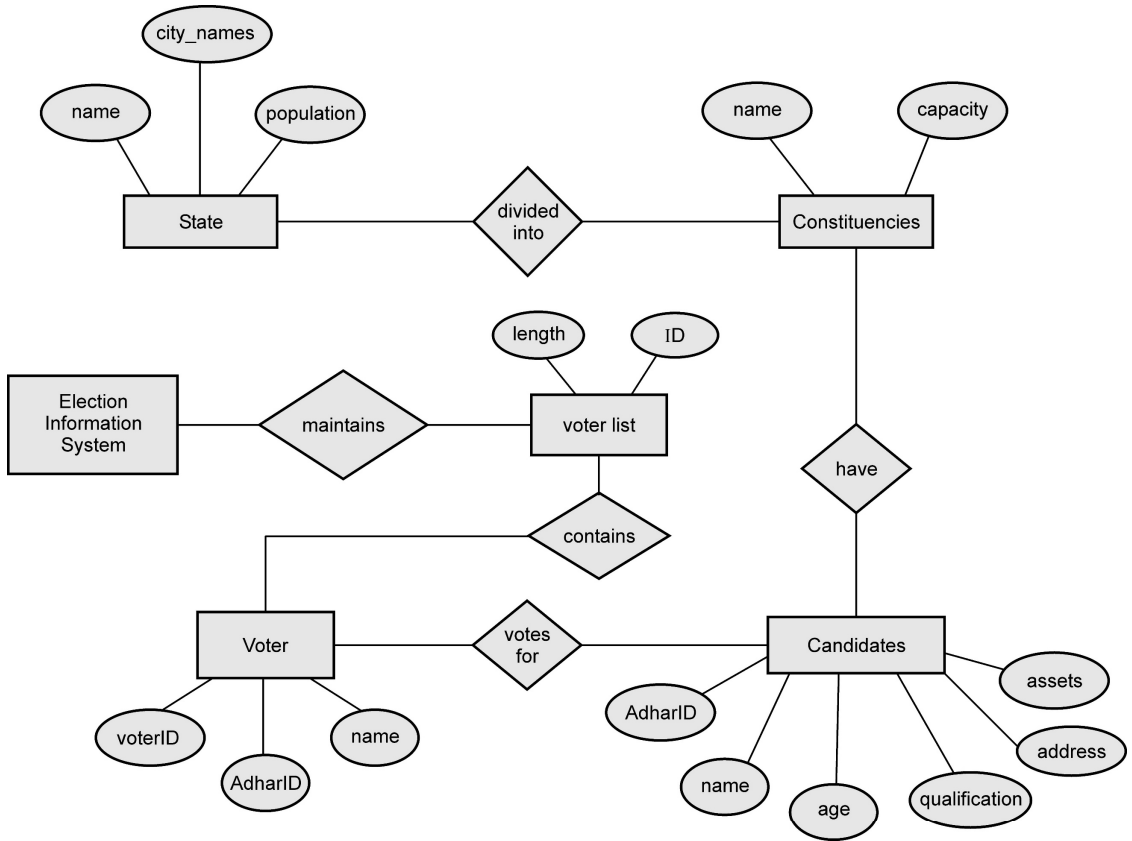
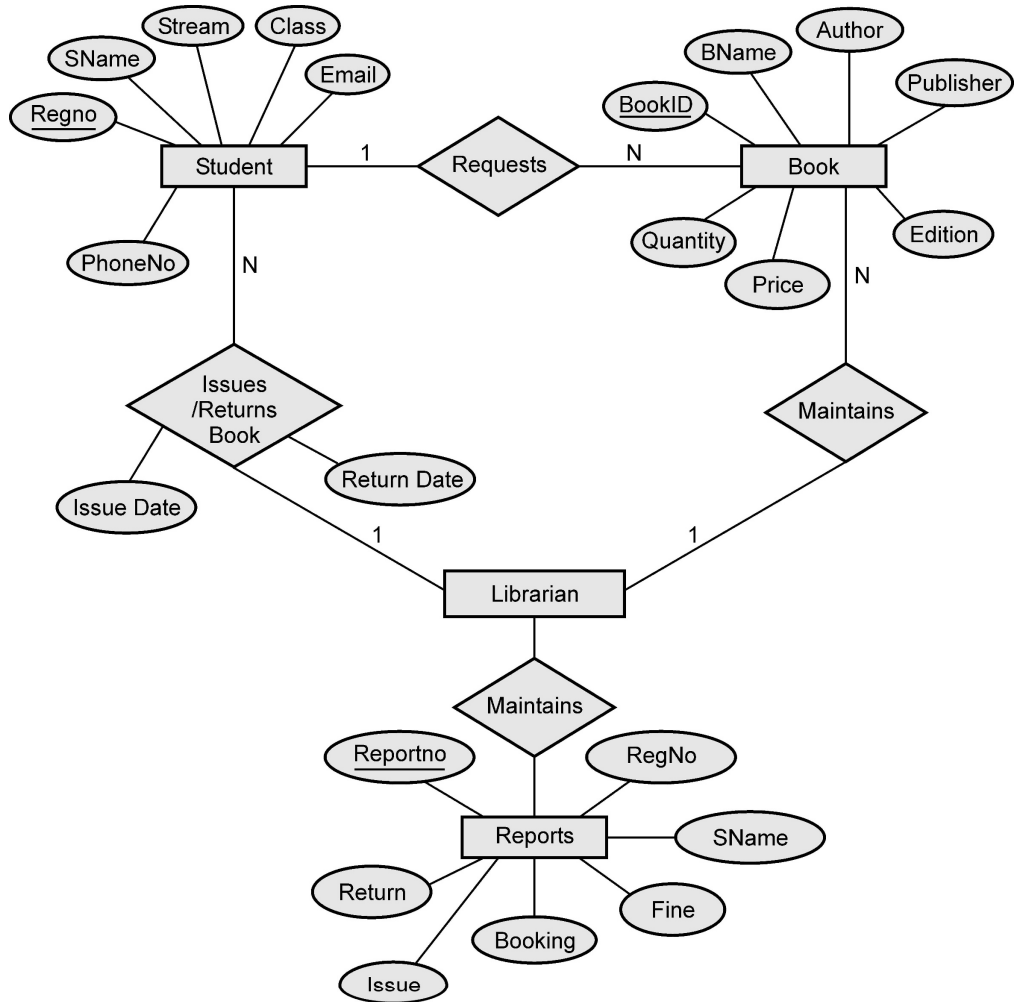


Fig. 1.21.1 Election information system

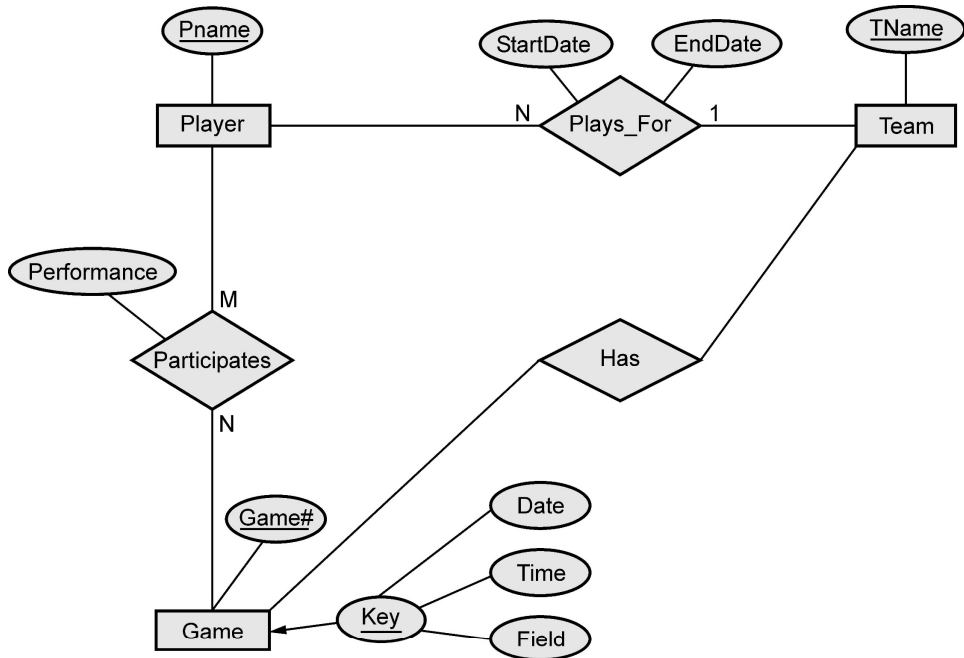
Example 1.21.9 Construct an E R diagram for library management system

Solution :



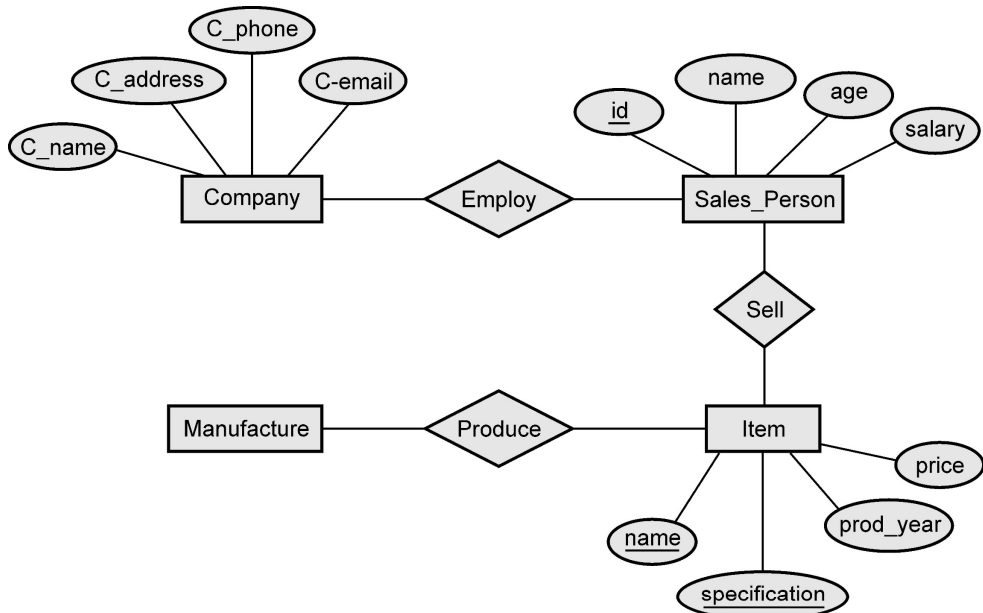
Example 1.21.10 A database is to be constructed to keep track of the teams and games of a sport league. A team has number of players not all of whom participate in each game. It is desired to keep track of the players participating in each game for each team, the positions they played in that game, and the result of the game. Design an ER diagram completely with attributes, keys and constraints for the above description.

Solution :



Example 1.21.11 Draw an ER diagram for a small marketing company database. Assume suitable data.

Solution :

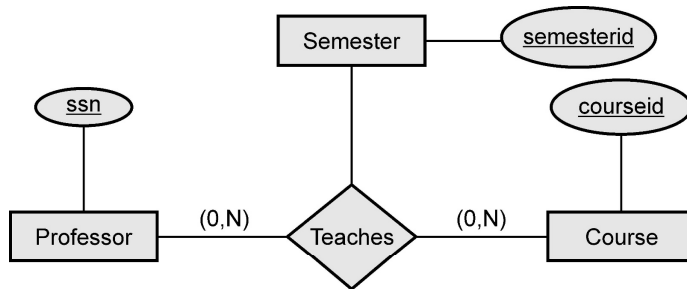


Example 1.21.12 A university database contains information about professors (identified by social security number, or SSN) and courses (identified by courseid). Professors teach courses; each of the following situations concerns the Teaches relationship set. For each situation, draw an ER diagram -

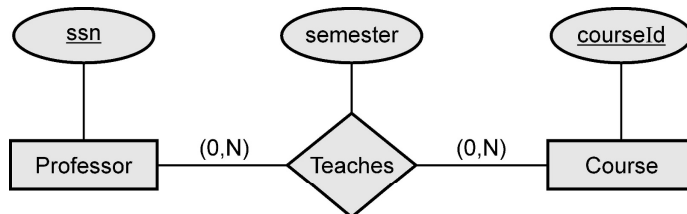
1. Professors can teach the same course in several semesters, and each ordering must be recorded.
2. Professors can teach the same course in several semesters, and only the most recent such ordering needs to be recorded (Assume this condition applies in all subsequent questions).
3. Every professor must teach some course.
4. Every professor teaches exactly one course.
5. Every professor teaches exactly one course and every course must be taught by some professor.

Solution :

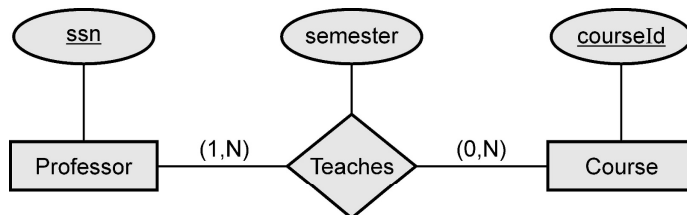
1.



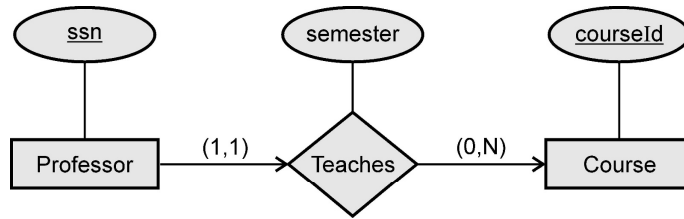
2.



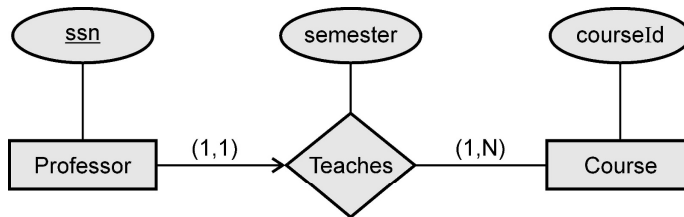
3.



4.

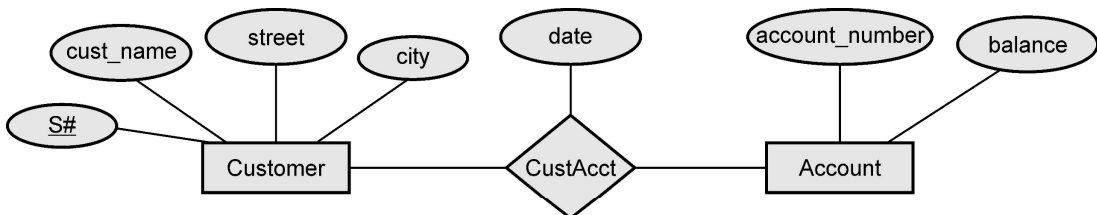


5.



Example 1.21.13 Construct an ER diagram of customer account relationship. Customer entity with attributes of S#, Customer name, street, customer city, and account entity with attributes account no, and balance. The customer account relationship with date attributes.

Solution :



Example 1.21.14 A university registrar's office maintains data about the following entities :

- (1) courses, including number, title, credits, syllabus, and prerequisites;
- (2) course offerings, including course number, year, semester, section number, instructor(s), timings, and classroom;
- (3) students, including student-id, name, and program; and
- (4) instructors, including identification number, name, department, and title.

Further, the enrollment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modeled.

(i) Construct an E-R diagram for the registrar's office. Document all assumptions that you make about the mapping constraints.

SPPU : Aug.-17, May-19, End Sem, Marks 10

OR

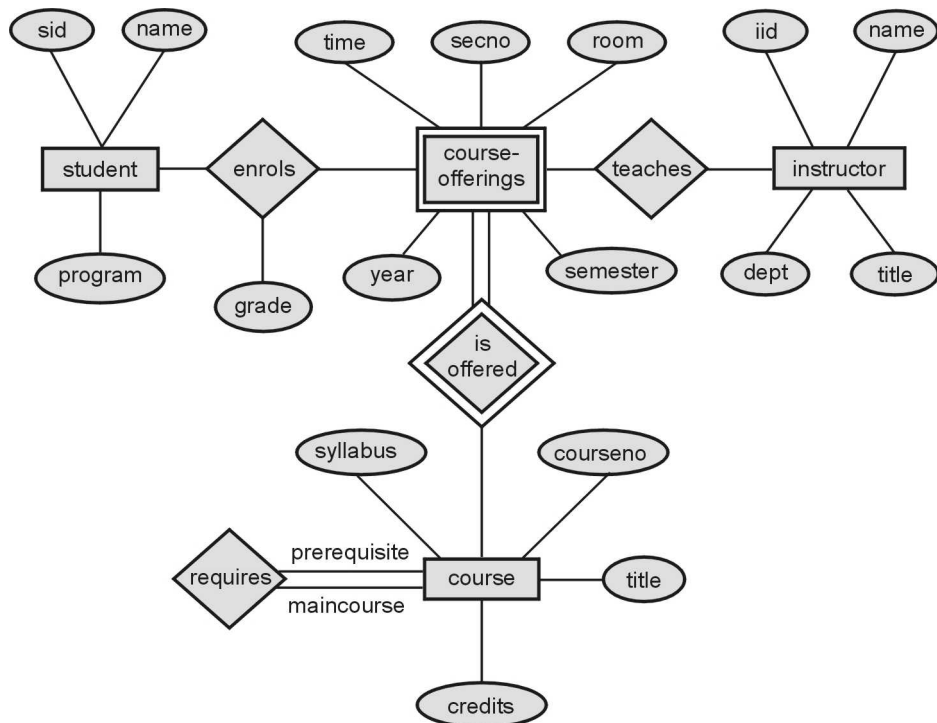
Consider a database used to record the marks that students get in different exams of different course offerings.

Construct an E - R diagram that models exams as entities, uses a ternary relationship, for the above database.

SPPU : Dec.-17, End Sem, Marks 5

Solution :

(i)



Example 1.21.15 Assuming you as a part of designer and development team, propose E-R model using E-R diagram for the following data requirements of banking system. Also convert and represent E-R model into tables characteristics of banking requirements are given below.

The bank is organized into branches. Each branch is located in a particular city.

Bank customers are identified by customer_id.

Bank employees are identified by emp-id

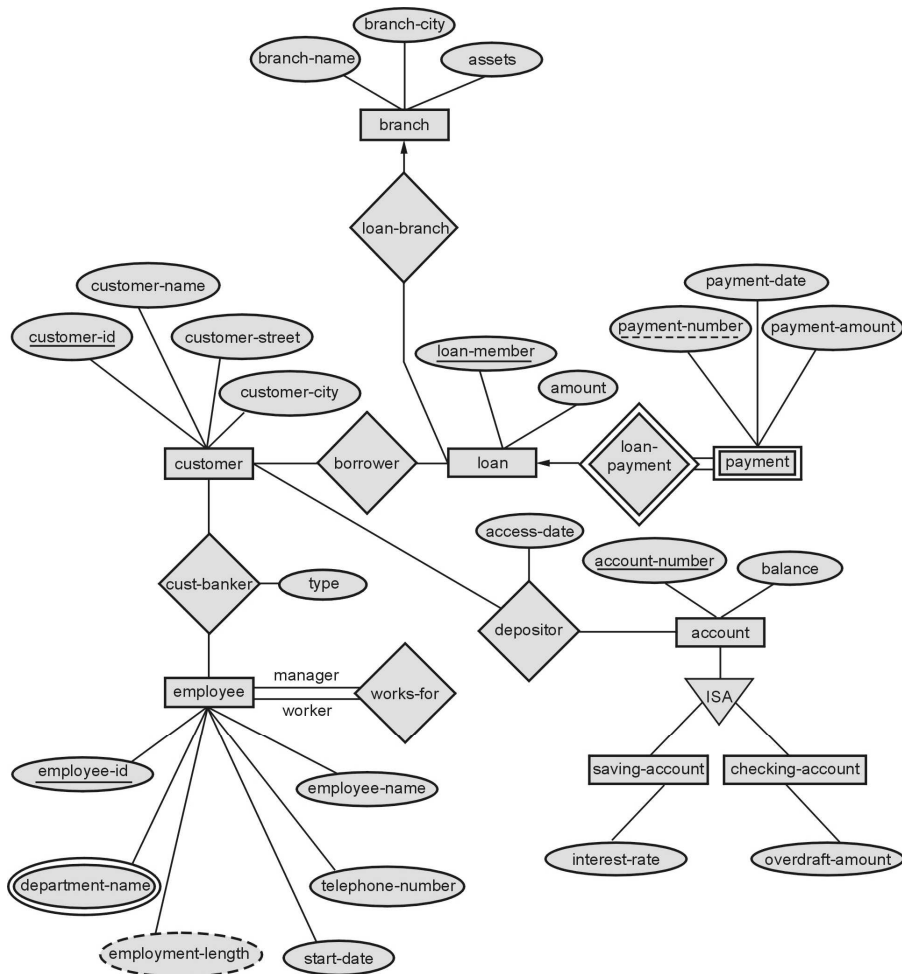
The bank offers two types of accounts saving and current. Accounts can be held by more than one customer and a customer can have more than one account.

A loan originates at a particular branch and can be held by one or more customers.

Additional requirement can be assumed if required, but assumptions should be clearly mentioned.

SPPU : Oct.-18, In Sem, Marks 10

Solution :



Example 1.21.16 Ramesh's family owns and operates a 100-acre farm for several generations.

Since the farm business is growing, Ramesh is thinking to build a database that would make easier the management of the activities in the farm. He is considering the following requirements for the database :

i) For each livestock classification group (for example, cow, horse etc.), Ramesh keeps track of the following: identification number, classification, total number of livestock per classification group (for example, number of cows, number of horses etc.)

ii) For each crop the following information is recorded Crop identification number and classification.

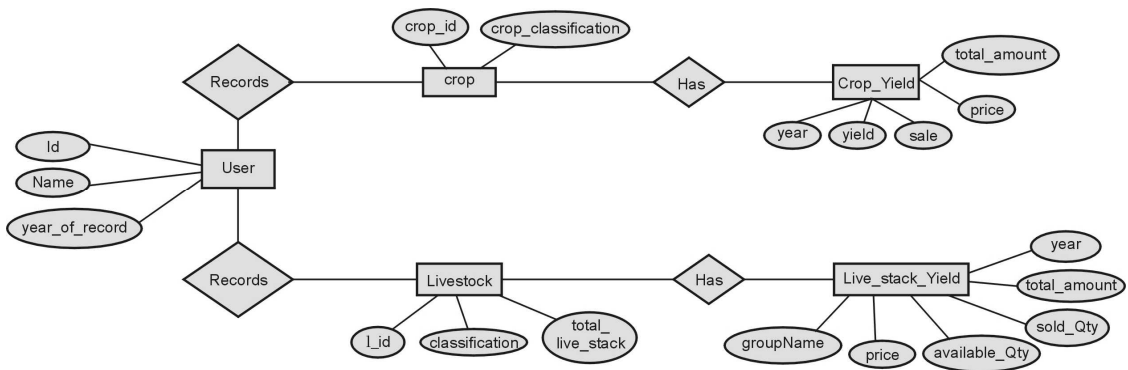
iii) Ramesh has recorded the yield of each crop classification group during the last ten years. The records consist of the year, yield, sales, price of the crop and the amount of money earned.

iv) Ramesh has recorded the yield of each livestock classification group during the last ten years. The records consist of the following historical data : the year, (historical) selling price per head, number of livestock in the end of the year, number of livestock sold during one-year period, and the total amount of money earned.

Draw an E-R diagram for this application. Specify the key attribute of each entity type.

SPPU : Dec.-18, End Sem, Marks 5

Solution :



Example 1.21.17 Assuming you as a part of design and development of team of an organization, propose E-R model using E-R diagram for the following data requirements. Also convert and represent E-R model into tables :

Company organized into DEPARTMENT. Each department has unique name and a particular employee who manages the department. Start date for the manager is recorded. Department may have several locations.

A department control a number of PROJECT. Projects have a unique name, number and a

single location.

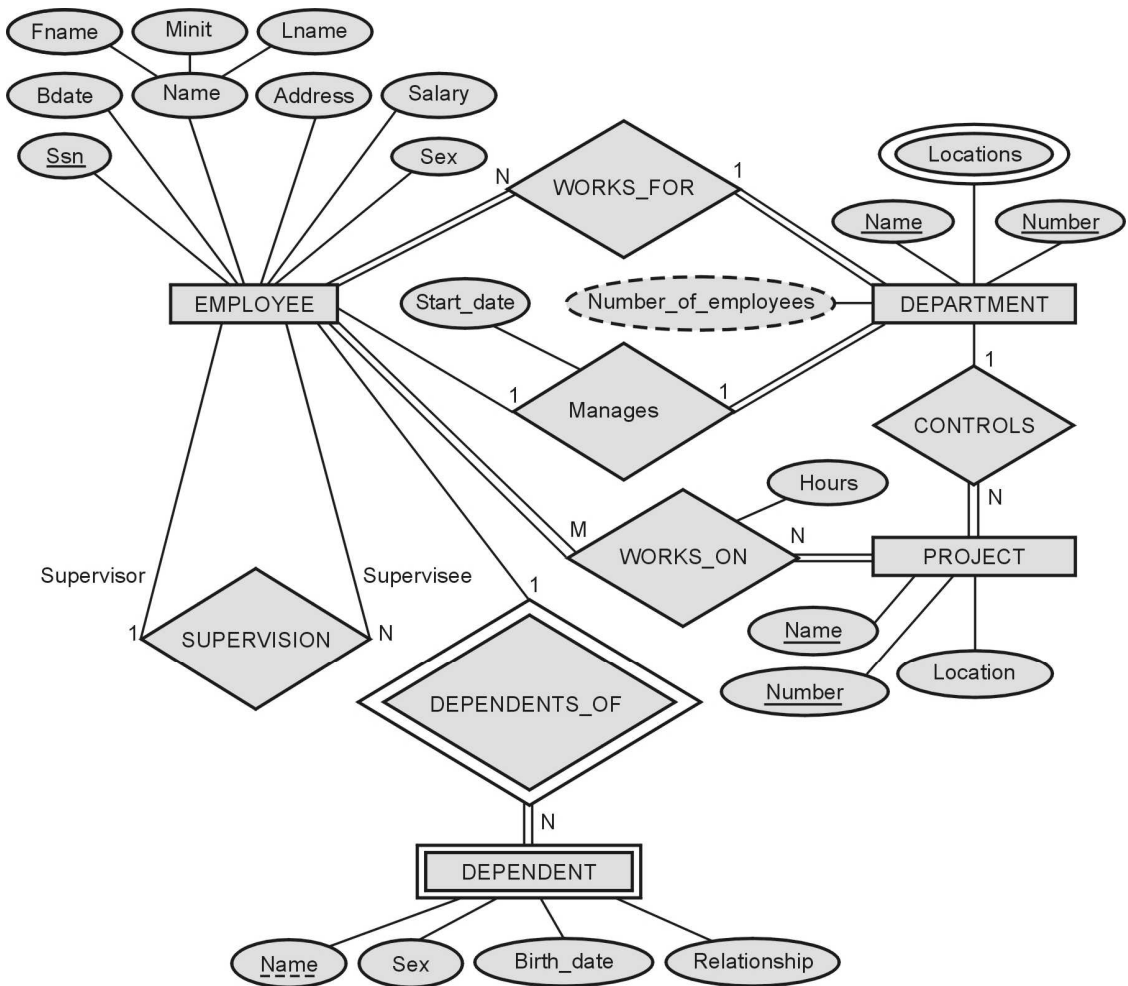
Company's EMPLOYEE name, ssno, address, salary, sex and birth date are recorded. An employee is assigned to one department, but may work for several projects (not necessarily controlled by her dept).

Number of hours/week an employee works on each project is recorded.

Employee's DEPENDENT are tracked for health insurance purposes (dependent name, birthdate, relationship to employee).

SPPU : May-19, End Sem, Oct.-19, In Sem, Marks 10

Solution :



The above ER diagram is converted into tables as

EMPLOYEE

ssn	Fname	Lname	Minit	Bdate	Address	sex	salary
-----	-------	-------	-------	-------	---------	-----	--------

DEPARTMENT

DNum	DName	Mng_ssn	Mng_start_date
------	-------	---------	----------------

DEPT_LOCATIONS

Dept_num	Dept_location
----------	---------------

PROJECT

Proj_Name	Proj_Num	Proj_Location	DNo
-----------	----------	---------------	-----

WORKS_ON

Emp_SSN	Proj_No	Hours
---------	---------	-------

DEPENDENT

Emp_SSN	Depend_Name	Sex	Bdate	Relationship
---------	-------------	-----	-------	--------------

Example 1.21.18 Assume we have the following application that models soccer teams, the games they play and the players in each team. In the design, we want to capture the following :

We have a set of teams, each team has an ID (unique identifier), name, main stadium and to which city this team belongs.

Each team has many players and each player belongs to one team. Each player has a number (unique identifier), name, DoB, start year and shirt number that he uses.

Teams play matches, in each match there is a host team and a guest team. The match takes places in the stadium of the host team.

For each match we need to keep track of the following :

The date on which the game is played.

The final result of the match.

The players participated in the match. For each player, how many goals he scored, whether or not he took yellow card and whether or not he took red card.

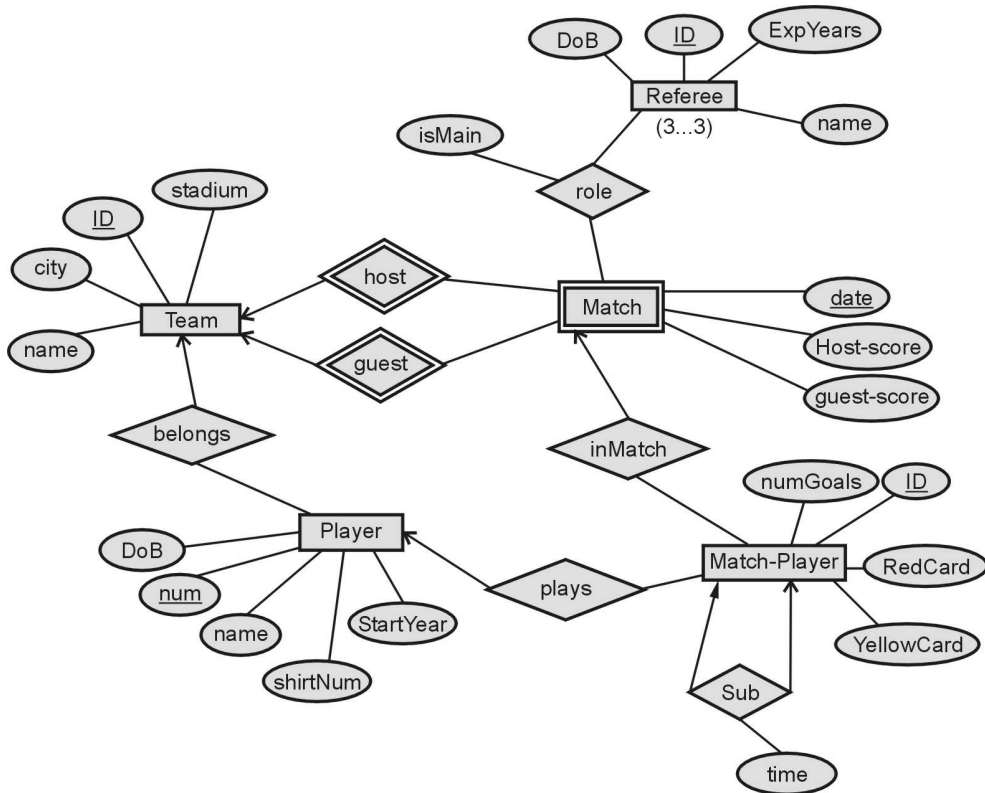
During the match, one player may substitute another player. We want to capture this substitution and the time at which it took place.

Each match has exactly three referees. For each referee we have an ID (unique identifier), name, DoB, years of experience. One referee is the main referee and the other two are assignment referee.

Design an ER diagram to capture the above requirements. State any assumptions you have that effects your design.

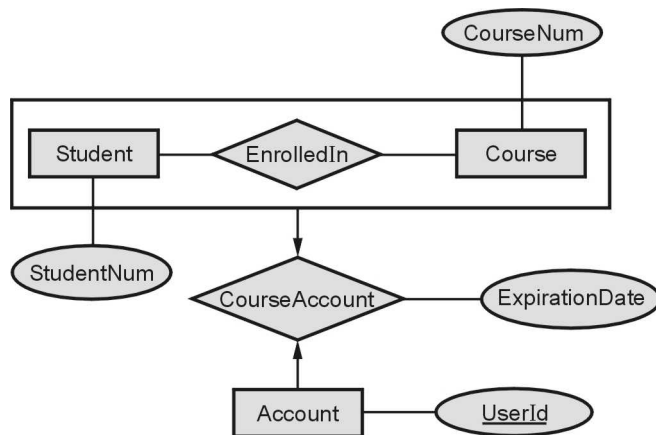
SPPU : Dec.-19, End Sem, Marks 5

Solution :



Example 1.21.19 Translate the following entity-relationship diagram to relational tables.

SPPU : Dec.-19, End Sem, Marks 5



Solution :

Student

<u>StudentNum</u>

Course

<u>CourseNum</u>

Account

<u>UserId</u>

EnrolledIn

<u>StudentNum</u>	<u>CourseNum</u>
-------------------	------------------

CourseAccount

<u>UserId</u>	StudentNum	CourseNum	ExpirationDate
---------------	------------	-----------	----------------

Example 1.21.20 *A weak entity set can always be made into strong entity set by adding to its attributes; the primary key attributes of its identifying entity set. Outline what sort of redundancy will result if we do so while converting into tables.*

SPPU : Aug.-17, Oct.-18, In Sem, Marks 5

Solution : By adding appropriate primary key to a weak entity set can result into the strong entity set.

If we add primary key attributes to the weak entity set, they will be present in both the entity set and the relationship set and both needs to be the same. Hence there will be redundancy.

Unit - I
Multiple Choice Questions

Q.1 A DBMS provides users with the conceptual representation of _____.

- a register b data
 c logical view d physical view

Q.2 DBMS helps to achieve _____.

- a data Independence b centralized Control of Data
 c neither a nor b d both a and b

Q.3 In view of total database content is _____.

- a conceptual view b internal view
 c external view d physical view

Q.4 The main purpose of DBMS is to provide _____ view of data to user.

- a completer b abstract
 c partial d none of these

Q.5 _____ means to hide certain details of how data is stored.

- a Data Integrity b Data independence
 c Data abstraction d Data separation

Q.6 How many levels of data abstraction are there ?

- a One b Two
 c Three d Four

Q.7 A _____ view of data expresses the way a user thinks about data _____.

- a logical view b physical view
 c both d none

Q.8 A physical view of data refers to the way data is handled at a _____ its storage and retrieval.

- a high level b low level
 c medium level d all of these

Q.9 Architecture of the database can be viewed as _____.

- a two levels b three levels
 c four levels d one level

- Q.10** In the architecture of a database system external level is the _____.
- a physical level b logical level
 c conceptual level d view level
- Q.11** In hierarchical model records are organized as _____.
- a lists b links
 c tree d graph
- Q.12** There are ____ levels of data independence.
- a one b two
 c three d four
- Q.13** The ability to modify the schema of database in one level without affecting the schema definition in higher level is called as _____.
- a data isolation b data abstraction
 c data hiding d data independence
- Q.14** Which of the following is record based on logical model ?
- a Network Model b Object Oriented Model
 c E-R Model d None of these
- Q.15** The DDL is used to specify the _____.
- a conceptual schemas b internal schemas
 c both d none
- Q.16** DCL stands for _____.
- a Data Control Language b Data Console Language
 c Data Console Level d Data Control Level
- Q.17** Which of the following is / are the DDL statements ?
- a Create b Drop
 c Alter d All of the above
- Q.18** Which are the three levels of abstraction ?
- a Physical b Logical
 c External d All of these
- Q.19** The statement in SQL which allows to change the definition of a table is _____.
- a create b alter
 c select d update

Q.20 Which of the following is NOT a basic element of all versions of the E - R model ?

- a Entities b Relationships
 c Attributes d Primary key

Q.21 Data independence means _____.

- a data is defined separately and not included in programs
 b programs are not dependent on the physical attributes of data.
 c programs are not dependent on the logical attributes of data
 d both (b) and (c)

Q.22 E-R model uses this symbol to represent weak entity set _____.

- a dotted rectangle b diamond
 c doubly outlined rectangle d none of these

Q.23 _____ express the number of entities to which another entity can be associated via a relationship set.

- a Mapping cardinality b Relational cardinality
 c Participation constraints d None of the mentioned

Q.24 In E-R diagram derived attribute is represented by _____.

- a rectangle b circle
 c dashed Ellipse d diamond

Q.25 DBA stands for _____.

- a Data Building Administrator b Database Access
 c Database Authentication d Database Administrator

Q.26 _____ represents the number of entities to which another entity can be associated

- a Degree b Cardinality
 c Modality d None of these

Q.27 Data Model is collection of conceptual tools for describing _____.

- a Data b Schema
 c constraints d All of the above

Q.28 Which of the following is example of Object based logical model ?

- a Relational Model b Hierarchical Model
 c Network Model d Entity Relationship Model

Q.29 Entity Relationship model consists of collection of basic objects called _____ and relationship among these objects.

- a functions b models
 c entity d all of these

Q.30 ER model was introduced by _____.

- a E.F. Codd b P.P. Chen
 c Bjarne Stroustrup d None of these

Q.31 ER diagram an ellipse represents _____.

- a weak entity b relationship
 c attribute d entity class

Q.32 In ER diagram the relationship is represented by _____.

- a rectangle b ellipse
 c diamond d circle

Q.33 Relationship among entities of a single class is called as _____.

- a IS-A relationship b recursive relationship
 c HAS -A relationship d none of these

Q.34 The relationship used for connecting entities of different types when identifiers are different _____.

- a HAS-A relationship b IS-A relationship
 c recursive relationship d none of these

Q.35 Which is not an example of strong entity type ?

- a Employee b Department
 c Emp_ID d Course

Q.36 If person is an entity then "Ankita" and "Prajakta" is the entity _____.

- a characteristics b field
 c identifier d instance

Q.37 Properties that describe the characteristics of entities are called :

- a entities b attributes
 c identifiers d relationships

Q.38 A student RollNumber, Name and Marks are all examples of _____.

- a entities b attributes
 c relationships d none of these

Q.39 An attribute which consists of a group of attributes is called as _____.

- a composite attribute b single valued attribute
 c multi-valued attribute d none of these

Q.40	The attribute "age" is calculated from "date_of_Birth". The attribute "age" is ____.		
<input type="checkbox"/> a	single valued	<input type="checkbox"/> b	multi valued
<input type="checkbox"/> c	composite	<input type="checkbox"/> d	derived
Q.41	The most common type of relationship used in data modeling is ____.		
<input type="checkbox"/> a	unary	<input type="checkbox"/> b	binary
<input type="checkbox"/> c	ternary	<input type="checkbox"/> d	none of these
Q.42	An entity type whose existence depends on another entity type is called ____ entity.		
<input type="checkbox"/> a	strong	<input type="checkbox"/> b	weak
<input type="checkbox"/> c	dependent	<input type="checkbox"/> d	mutually dependent
Q.43	Every weak entity set can be converted into a strong entity set by :		
<input type="checkbox"/> a	Using generalization	<input type="checkbox"/> b	Adding appropriate attributes
<input type="checkbox"/> c	Using aggregation	<input type="checkbox"/> d	None of the above
Q.44	In a one-to-many relationship, the entity that is on the one side of the relationship is called a(n) _____ entity.		
<input type="checkbox"/> a	parent	<input type="checkbox"/> b	child
<input type="checkbox"/> c	instance	<input type="checkbox"/> d	subtype
Q.45	Which of the following is NOT a basic element of all versions of the E-R model ?		
<input type="checkbox"/> a	Entities	<input type="checkbox"/> b	Attributes
<input type="checkbox"/> c	Relationships	<input type="checkbox"/> d	Primary keys
Q.46	The entity employee has three candidate keys : 1) EmpID 2) Email 3) Date_of_joining 4) Designation. Suggest best primary key for this entity :		
<input type="checkbox"/> a	EmpID	<input type="checkbox"/> b	Email
<input type="checkbox"/> c	Date_of_joining	<input type="checkbox"/> d	Designation

Answers Keys for Multiple Choice Questions :

Q.1	b	Q.2	d	Q.3	a	Q.4	b
Q.5	c	Q.6	c	Q.7	a	Q.8	b
Q.9	b	Q.10	d	Q.11	c	Q.12	b
Q.13	d	Q.14	a	Q.15	a	Q.16	a
Q.17	d	Q.18	d	Q.19	b	Q.20	d

Q.21	d	Q.22	c	Q.23	a	Q.24	c
Q.25	d	Q.26	b	Q.27	d	Q.28	d
Q.29	c	Q.30	b	Q.31	c	Q.32	c
Q.33	b	Q.34	a	Q.35	c	Q.36	d
Q.37	b	Q.38	b	Q.39	c	Q.40	d
Q.41	b	Q.42	b	Q.43	b	Q.44	a
Q.45	d	Q.46	a				



Notes

2

Structured Query Language

Syllabus

SQL : Characteristics and Advantages, SQL Data Types and Literals, DDL, DML, DCL, TCL, SQL Operators. **Tables** : Creating, Modifying, Deleting, Updating. **SQL DML Queries** : SELECT Query and clauses, Index and Sequence in SQL. **Views** : Creating, Dropping, Updating using Indexes, Set Operations, Predicates and Joins, Set membership, Tuple Variables, Set comparison, Ordering of Tuples, Aggregate Functions, SQL Functions, Nested Queries.

Contents

- 2.1 Introduction to Structured Query Language (SQL)
- 2.2 SQL Data Types and Literals **Nov.-17**, Marks 5
- 2.3 DDL, DML, DCL and TCL Structure
- 2.4 Tables
- 2.5 SQL DML Queries
- 2.6 Logical Operators
- 2.7 String Operations
- 2.8 The BETWEEN Operator
- 2.9 Built-In Functions
- 2.10 NULL Values
- 2.11 EXISTS, NOT EXISTS and UNIQUE
- 2.12 Defining Constraints
- 2.13 Renaming Attributes
- 2.14 Tuple Variables

2.15	Schema Change Statements	Nov.-18,	Marks 5
2.16	Indexes	Aug. 17, Dec. 18, May 19,	Marks 5
2.17	Aggregate Functions	Nov.-17,	Marks 2
2.18	Set Operations		
2.19	Nested Queries		
2.20	Join Operation	Nov.-17,19, May-19,	Marks 6
2.21	Views	July-18, Oct.-18, Nov.-19,	Marks 8
2.22	Examples Based on SQL	Aug 17, Dec. 17,18, May 18,19, Oct 18,19,	Marks 5

2.1 Introduction to Structured Query Language (SQL)

- SQL stands for **Structured Query Language**.
- It is the language of databases and almost all companies use databases to store their data.
- SQL makes use of query. A Query is a set of instruction given to the database management system. It tells any database what information we would like to get from the database.
- SQL is **case- insensitive**. However it has become standard in SQL community to use **all capital** letters for SQL keywords.
- SQL is a standard language for Relational Database Management System (RDBMS).
- There are various RDBMS software that are popularly used. It includes MySQL, Oracle, MS ACCESS, Microsoft SQL Server, Sybase and so on.

2.1.1 Characteristics and Advantages

- 1) SQL is a standard computer language for creating and manipulating databases.
- 2) SQL is very simple and easy to learn.
- 3) SQL allows the users to create,update,delete and retrieve data from the database.
- 4) SQL is used to create view, stored procedures and functions in a database.
- 5) SQL allows the users to set the permissions on the tables, procedures and views in the database.

2.2 SQL Data Types and Literals

SPPU : Nov.-17, Marks 5

Various data types used in SQL are -

1) Numeric data types

- Integer numbers : INT, INTEGER, SMALLINT, BIGINT
- Floating-point (real) numbers : REAL, DOUBLE , FLOAT
- Fixed-point numbers : DECIMAL(n,m), DEC(n,m), NUMERIC(n,m), NUM(n,m)

2) Character-string data types

- Fixed length : CHAR(n), CHARACTER(n)
- Varying length : VARCHAR(n), CHAR VARYING(n), CHARACTER VARYING(n), LONG VARCHAR

3) Large object data types

- Characters : CLOB, CHAR LARGE OBJECT , CHARACTER LARGE OBJECT

- Bits : BLOB, BINARY LARGE OBJECT

4) Boolean data type

- Values of TRUE or FALSE or NULL

5) DATE data type

- Ten positions
- Components are YEAR, MONTH, and DAY in the form YYYY-MM-DD

6) Additional Data type

a) TIMESTAMP data type

It includes the DATE and TIME fields.

b) INTERVAL data type

It specifies a relative value that can be used to increment or decrement an absolute value of a date, time, or timestamp.

2.3 DDL, DML, DCL and TCL Structure

There are four types of SQL commands -

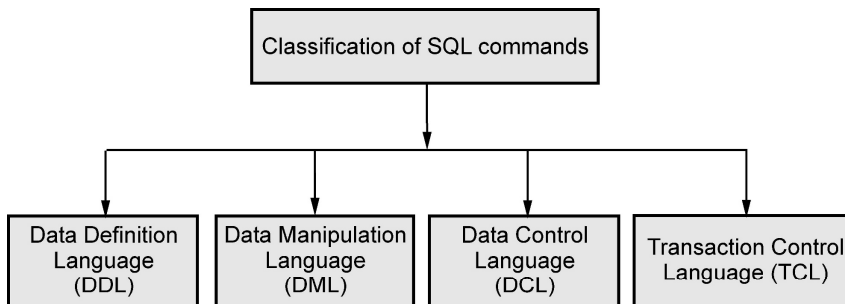


Fig. 2.3.1

Data Definition Language

Sr.No.	Command	Purpose
1.	CREATE	This command is used to create database, tables, views or any other database objects.
2.	ALTER	It modifies the existing tables.
3.	DROP	This command deletes complete table, view or any other database object.

Data Manipulation Language

Sr. No.	Command	Purpose
1.	SELECT	This command is used to retrieve either all or desired records from one or more tables.
2.	INSERT	For inserting the records in the table, this command is used.
3.	UPDATE	For updating one or more fields of the table, this command is used.
4.	DELETE	This command is used for deleting the desired record

Data Control Language

Sr. No.	Command	Purpose
1.	GRANT	This command is used to give access rights or privileges to the database.
2.	INVOKE	The revoke command removes user access rights or privileges to the database objects.

Transaction Control Language (TCL)

Sr. No.	Command	Purpose
1.	COMMIT	This command is used to save permanently any transaction to database
2.	ROLLBACK	The ROLLBACK command is used to undo transactions that have not already saved to database.

Difference between DDL and DML

DDL	DML
DDL stands for Data Definition Language.	DML stands for Data Manipulation Language.
DDL commands are used to define database structure.	DML commands are used for managing data within the database.
It works on whole table.	It works on one or more rows.
It cannot be classified further.	It can be classified as – procedural and non-procedural language.
Changes made by DDL commands cannot be rolled back.	Changes by DML commands can be rolled back.
Example - CREATE, ALTER, DROP commands.	Example - SELECT, INSERT, UPDATE, and DELETE commands.

- **Terminology** : In relational model we use the terms relation tuple and attribute.

Same terms are used in SQL as follows

Relation Model	SQL
Relation	Table
Tuple	Row
Attribute	Column

2.4 Tables

2.4.1 Creating Table

- A database can be considered as a container for tables and a table is a grid with rows and columns to hold data.
- Individual statements in SQL are called **queries**.
- We can execute SQL queries for various tasks such as creation of tables, insertion of data into the tables, deletion of record from table, and so on.

In this section we will discuss how to create a table.

Step 1 : We normally create a database using following SQL statement

Syntax

```
CREATE DATABASE database_name;
```

Example

```
CREATE DATABASE Person_DB
```

Step 2 : The table can be created inside the database as follows –

```
CREATE TABLE table_name (
    col1_name datatype,
    col2_name datatype,
    ...
    coln_name datatype
);
```

Example

```
CREATE TABLE person_details (
    AadharNo int,
    FirstName VARCHAR(20),
    MiddleName VARCHAR(20),
    LastName VARCHAR(20),
    Address VARCHAR(30),
    City VARCHAR(10)
)
```

The blank table will be created with following structure

Person_details

AadharNo	FirstName	MiddleName	LastName	Address	City
----------	-----------	------------	----------	---------	------

2.4.2 Insertion of Data into the Table

- We can insert data into the table using INSERT statement.

Syntax

```
INSERT INTO table_name (col1,col2,...,coln)
VALUES (value1,value2,...,valuen)
```

Example

```
INSERT INTO person_details (AadharNo, FirstName, MiddleName, LastName, Address, City)
VALUES (111,'AAA','BBB','CCC','M.G. Road','Pune')
```

The above query will result into –

AadharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. Road	Pune

2.4.3 Modifying the Record from the Table

- For modifying the existing record of a table, update query is used.

Syntax

```
UPDATE table_name
SET col1=value1, col2=value2,...
WHERE condition;
```

The WHERE command is used to specify some condition. Based on this condition the data present in the table can be displayed or can be updated or deleted.

Example

Consider following table

Person_details table

AadharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. Road	Pune
222	DDD	EEE	FFF	Shivaji nagar	Pune
333	GGG	HHH	III	Chandani Chowk	Delhi
444	JJJ	KKK	LLL	Viman Nagar	Mumbai

If we execute following query

```
UPDATE person_details
SET city='Chennai'
WHERE AadharNo=333
```

The result will be

AadharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. Road	Pune
222	DDD	EEE	FFF	Shivaji nagar	Pune
333	GGG	HHH	III	Chandani chowk	Chennai
444	JJJ	KKK	LLL	Viman nagar	Mumbai

2.4.4 Deleting Record from the Table

- We can delete one or more records based on some condition. The syntax is as follows –

Syntax

```
DELETE FROM table_name WHERE condition;
```

Example

```
DELETE FROM person_details
WHERE AadharNo=333
```

The result will be -

AadharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. road	Pune
222	DDD	EEE	FFF	Shivaji nagar	Pune
444	JJJ	KKK	LLL	Viman nagar	Mumbai

We can delete all the records from table. But in this deletion, all the records get deleted without deleting table. For that purpose the SQL statement will be

```
DELETE FROM person_details;
```

2.5 SQL DML Queries

- DML stands for Data manipulation Language
- The basic operations under DML queries are SELECT, INSERT, UPDATE, and DELETE

We have already discussed INSERT, DELETE and UPDATE operations in previous section. Let us now discuss the SELECT operation.

2.5.1 SELECT Query

- The Select statement is used to fetch the data from the database table.
- The result returns the data in the form of table. These result tables are called **resultsets**.
- We can use the keyword **DISTINCT**. It is an optional keyword indicating that the answer should **not contain duplicates**. Normally if we write the SQL without DISTINCT operator then it does not eliminate the duplicates.

Syntax

```
SELECT col1, col2, ...,coln FROM table_name;
```

Example

```
SELECT AadharNo, FirstName, Address, City FROM person_details
```

The result of above query will be

AadharNo	FirstName	City
111	AAA	Pune

- If we want to select all the records present in the table we make use of * character.

Syntax

```
SELECT * FROM table_name;
```

Example

```
SELECT * FROM person_details;
```

The above query will result into

AadharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. road	Pune

- **Use of DISTINCT Keyword:** The keyword DISTINCT is used along with the SELECT statements.
- It is used to obtain unique values from the table. This query does not allow duplication of element.

Syntax :

```
SELECT DISTINCT Column-name FROM table-name;
```

Example:

Consider following database table **Student**

Roll No	Name	City
1	Ankita	Pune
2.	Rohit	Hyderabad
3.	Prajakta	Chennai
4.	Sunil	Pune
5.	Sharda	Chennai

SQL Statement

```
SELECT DISTINCT City
FROM Student;
```

This will result into

Pune
Hyderabad
Chennai

2.5.2 WHERE

The WHERE command is used to specify some condition. Based on this condition the data present in the table can be displayed or can be updated or deleted.

Syntax

```
SELECT col1,col2, ...,coln
FROM table_name
WHERE condition;
```

Example

Consider following table

AadharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. road	Pune
222	DDD	EEE	FFF	Shivaji nagar	Pune
333	GGG	HHH	III	Chandani chowk	Delhi
444	JJJ	KKK	LLL	Viman nagar	Mumbai

If we execute the following query

```
SELECT AadharNo
FROM person_details
WHERE city='Pune';
```

The result will be –

AadharNo	City
111	Pune
222	Pune

If we want records of all those person who live in city Pune then we can write the query using WHERE clause as

```
SELECT *
FROM person_details
WHERE city='Pune';
```

The result of above query will be

AadharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. road	Pune
222	DDD	EEE	FFF	Shivaji nagar	Pune

2.5.3 Clauses

Most commonly used clauses in SQL statements are Order by, Group by and Having. Let us discuss them along with syntax and examples.

(1) Order By

- Many times we need the records in the table to be in sorted order.
- If the records are arranged in increasing order of some column then it is called **ascending order**.
- If the records are arranged in decreasing order of some column then it is called **descending order**.
- For getting the sorted records in the table we use **ORDER BY** command.
- The ORDER BY keyword sorts the records in ascending order by default.

Syntax

```
SELECT col1, col2,...,coln
FROM table_name
ORDER BY col1,col2,... ASC|DESC
```

Here ASC is for ascending order display and DESC is for descending order display.

Example

Consider following table

AadharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. road	Pune
222	DDD	EEE	FFF	Shivaji nagar	Pune
333	GGG	HHH	III	Chandani chowk	Delhi
444	JJJ	KKK	LLL	Viman nagar	Mumbai

```
SELECT *
FROM person_details
ORDER BY AadharNo DESC;
```

The above query will result in

AadharNo	FirstName	MiddleName	LastName	Address	City
444	JJJ	KKK	LLL	Viman nagar	Mumbai
333	GGG	HHH	III	Chandani chowk	Delhi
222	DDD	EEE	FFF	Shivaji nagar	Pune
111	AAA	BBB	CCC	M.G. road	Pune

(2) Group By

- The GROUP BY clause is a SQL command that is used to group rows that have the same values.
- The GROUP BY clause is used in the SELECT statement.
- Optionally it is used in conjunction with aggregate functions.
- The queries that contain the GROUP BY clause are called grouped queries
- This query returns a single row for every grouped item.
- **Syntax :**

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
```

- **Example :** Consider the Student table as follows -

sid	sname	marks	city
1	AAA	60	Pune
2	BBB	70	Mumbai
3	CCC	90	Pune
4	DDD	55	Mumbai

Query : Find the total marks of each student in each city

```
SELECT SUM(marks), city
FROM Student
GROUP BY city
```

The result will be as follows –

SUM(marks)	city
150	Pune
125	Mumbai

(3) Having

- HAVING filters records that work on summarized GROUP BY results.
- HAVING applies to summarized group records, whereas WHERE applies to individual records.
- Only the groups that meet the HAVING criteria will be returned.
- HAVING requires that a GROUP BY clause is present.
- WHERE and HAVING can be in the same query.
- **Syntax :**

```
SELECT column-names
FROM table-name
WHERE condition
GROUP BY column-names
HAVING condition
```

- **Example** : Consider the Student table as follows -

sid	sname	marks	city
1	AAA	60	Pune
2	BBB	70	Mumbai
3	CCC	90	Pune
4	DDD	55	Mumbai
5	EEE	84	Chennai

Query : Find the total marks of each student in the city named 'Pune' and 'Mumbai' only

```
SELECT SUM(marks), city
FROM Student
GROUP BY city
HAVING city IN('Pune','Mumbai')
```

- The result will be as follows –

SUM(marks)	city
150	Pune
125	Mumbai

2.6 Logical Operators

- Using WHERE clause we can use the operators such as AND, OR and NOT.
- AND operator displays the records if all the conditions that are separated using AND operator are true.
- OR operator displays the records if any one of the condition separated using OR operator is true.
- NOT operator displays a record if the condition is NOT TRUE.

Consider following table

AadharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. road	Pune
222	DDD	EEE	FFF	Shivaji nagar	Pune

333	GGG	HHH	III	Chandani chowk	Delhi
444	JJJ	KKK	LLL	Viman nagar	Mumbai

Syntax of AND

```
SELECT col1, col2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

Example of AND

If we execute following query –

```
SELECT AadharNo, FirstName, City
FROM person_details
WHERE AadharNo=222 AND City='Pune';
```

The result will be –

AadharNo	FirstName	City
222	DDD	Pune

Syntax of OR

```
SELECT col1, col2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

Example of OR

```
SELECT AadharNo, FirstName, City
FROM person_details
WHERE City='Pune' OR City='Mumbai';
```

The result will be –

AadharNo	FirstName	City
111	AAA	Pune
222	DDD	Pune
444	JJJ	Mumbai

Syntax of NOT

```
SELECT col1, col2, ...
FROM table_name
WHERE NOT condition;
```

Example of NOT

```
SELECT AadharNo, FirstName, City
FROM person_details
WHERE NOT City='Pune';
```

The result will be

AadharNo	FirstName	City
333	GGG	Delhi
444	JJJ	Mumbai

2.7 String Operations

- For string comparisons, we can use the comparison operators =, <, >, <=, >=, <> with the ordering of strings determined alphabetically as usual.
- SQL also permits a variety of functions on character strings such as concatenation using operator ||, extracting substrings, finding length of string, converting strings to upper case (using function **upper(s)**) and lowercase (using function **lower(s)**), removing spaces at the end of string (using function (trim(s)) and so on.
- Pattern matching can also be performed on strings using two types of special characters –
 - Percent(%): It matches zero, one or multiple characters
 - Underscore(_): The _ character matches any single character.
- The percentage and underscore can be used in combinations.
- Patterns are case sensitive. That means upper case characters do not match lowercase characters or vice versa.
- **For instance :**
 - 'Data%' matches any string beginning with "Data", For instance it could be with "Database", "DataMining", "DataStructure"
 - '___' matches any string of exactly three characters.
 - '___%' matches any string of at least length 3 characters.
- The **LIKE** clause can be used in **WHERE** clause to search for specific patterns.
- **For example –** Consider following **Employee Database**

EmpID	EmpName	Department	Date_of_Join
1	Sunil	Marketing	1-Jan
2	Mohsin	Manager	2-Jan
3	Supriya	Manager	3-Jan
4	Sonia	Accounts	4-Jan
5	Suraj	Sales	5-Jan
6	Archana	Purchase	6-Jan

(1) Find all the employee with EmpName starting with “s”**SQL Statement :**

```
SELECT * FROM Employee
WHERE EmpName LIKE 's%'
```

Output

EmpID	EmpName	Department	Date_of_Join
1	Sunil	Marketing	1-Jan
3	Supriya	Manager	3-Jan
4	Sonia	Accounts	4-Jan
5	Suraj	Sales	5-Jan

(2) Find the names of employee whose name begin with S and end with a**SQL Statement :**

```
SELECT EmpName FROM Employee
WHERE EmpName LIKE 'S%a'
```

Output

EmpName
Supriya
Sonia

(3) Find the names of employee whose name begin with S and followed by exactly four characters

```
SELECT EmpName FROM Employee
WHERE EmpName LIKE 'S_____'
```

Output

EmpName
Sunil
Sonia
Suraj

2.8 The BETWEEN Operator

- The **between** operator can be used to simplify the where clause which is used to denote the value be less than or equal to some value and greater than or equal to some other value.
- For example – if we want the names of the students whose marks are between 80 and 90 then SQL statement will be

```
SELECT name
FROM Students
WHERE marks BETWEEN 80 and 90;
```

2.9 Built-In Functions

- In SQL a built-in function is a piece for programming that takes zero or more inputs and returns a value.
- An example of a built-in functions is ABS(), which when given a value calculates the absolute (non-negative) value of the number.

Query

```
SELECT ABS(-9) Result;
```

Output

Result
9

Mathematical Functions

Function	Value Returned
ABS (m)	Absolute value of m
MOD (m, n)	Remainder of m divided by n
POWER (m, n)	m raised to the nth power
ROUND (m [, n])	m rounded to the nth decimal place
TRUNC (m [, n])	m truncated to the nth decimal place
SIN (n)	sine (n)

COS (n)	cosine (n)
TAN (n)	tan (n)
SQRT (n)	positive square root of n
EXP (n)	e raised to the power n
LOG (n2, n1)	logarithm of n1, base n2
CEIL (n)	smallest integer greater than or equal to n
FLOOR (n)	greatest integer smaller than or equal to n
SIGN (n)	-1 if n < 0, 0 if n = 0, and 1 if n > 0

String Functions

Function	Value Returned
INITCAP (s)	First letter of each word is changed to uppercase and all other letters are in lower case.
LOWER (s)	All letters are changed to lowercase.
UPPER (s)	All letters are changed to uppercase.
CONCAT (s1, s2)	Concatenation of s1 and s2. Equivalent to s1 s2
LTRIM (s [, set])	Returns s with characters removed up to the first character not in set; defaults to space
RTRIM (s [, set])	Returns s with final characters removed after the last character not in set; defaults to space
REPLACE (s, search_s [, replace_s])	Returns s with every occurrence of search_s in s replaced by replace_s; default removes search_s
SUBSTR (s, m [, n])	Returns a substring from s, beginning in position m and n characters long; default returns to end of s.
LENGTH (s)	Returns the number of characters in s.

For example

```
SELECT CONCAT('Bill', 'Gates') as "NAME"
```

Date and Time Functions

Function	Value Returned
ADD_MONTHS (d, n)	Date d plus n months
LAST_DAY (d)	Date of the last day of the month containing d
MONTHS_BETWEEN (d, e)	Number of months by which e precedes d
NEW_TIME (d, a, b)	The date and time in time zone b when date d is for time zone a
NEXT_DAY (d, day)	Date of the first day of the week after d
SYSDATE	Current date and time
GREATEST (d1, d2, ..., dn)	Latest of the given dates
LEAST (d1, d2, ..., dn)	Earliest of the given dates

For example

```
SELECT SYSDATE();
```

Will result in

```
2019-06-27 10:02:39
```

2.10 NULL Values

- In SQL NULL is an important value. NULL is nothing but a missing value. But it has three different interpretations -
 - 1) **Unknown Value** : If student's age is not known then the NULL represents the unknown value.
 - 2) **Unavailable Value** : If a persons' phone number is not available then it can be represented by NULL value in the database
 - 3) **Not applicable Attribute** : If some value is not applicable to current database systems then it can be represented by NULL. For instance - Parent's designation is not applicable in student database system.
- When a record with NULL in one of its attributes is involved in a comparison operation, result is UNKNOWN.

- SQL uses a **three-valued** logic with values TRUE, FALSE, and UNKNOWN instead of the standard two-valued (Boolean) logic with values TRUE or FALSE. Using the logical operators AND, OR and NOT the result of these three values is represented in the following table -

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

NOT		
TRUE	FALSE	
FALSE	TRUE	
UNKNOWN	UNKNOWN	

- For example - The result of FALSE AND TRUE is FALSE
- SQL allows queries that check whether an attribute value is NULL. For that, instead of using =, <> (not equal to), it compares attribute value to NULL using the operators IS or IS NOT.
- For instance - Retrieve names of all students who do not opt for any sports.

```
SELECT FName, LName
FROM STUDENT
WHERE sports_type IS NULL
```

2.11 EXISTS, NOT EXISTS and UNIQUE

- The **EXISTS** operator is used to test for the existence of any record in a subquery. The **EXISTS** operator returns true if the subquery returns one or more records.

- Syntax**

```
SELECT column_name(s)
FROM table_name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE condition);
```

- Example - Consider two tables namely Student and Student_Score table as follows -

Student Table

RollNo	FName	LName
1.	Anil	Kumble
2.	Virat	Kohli
3.	Suresh	Raina

Student_Score Table

Subject	TestNo	Grade	RollNo
Maths	1	10	1
Maths	2	9.5	1
Maths	3	9.75	1
Science	4	9.5	1
Science	5	8.5	1
Maths	6	9.25	2
Maths	7	9.8	2
Science	8	7	2

Let's say we want to get all students that have received a 10 grade in the subject 'Maths'.

The SQL query using EXISTS can be written as follows –

```

SELECT
  RollNo, FName, LName
FROM
  Student
WHERE EXISTS (
  SELECT Student_Score.RollNo
  FROM
    Student_Score
  WHERE
    Student_Score.RollNo = Student.RollNo AND
    Student_Score.grade = 10 AND
    Student_Score.Subject = 'Maths'
)

```

After running the above query the result will be -

RollNo	FName	LName
1.	Anil	Kumble

The outer query selects the student row columns we are interested in returning to the client. However, the WHERE clause is using the EXISTS operator with an associated inner subquery.

- The EXISTS operator returns true if the subquery returns at least one record and false if no row is selected.
- The NOT EXISTS operator returns true if the underlying subquery returns no record. However, if a single record is matched by the inner subquery, the NOT EXISTS operator will return false, and the subquery execution can be stopped.
- The UNIQUE(Q) function returns TRUE if there are no duplicate tuples in the result of query Q.
- **Syntax**

```

SELECT UNIQUE column_name1, column_name2, ..., column_namen
FROM table_name;

```

- **Example** - Suppose there is a Student table as follows –

Student

RollNo	Name	City	Age
1	AAA	Pune	19
2	BBB	Mumbai	19
3	CCC	Chennai	20
4	DDD	Delhi	20

If we execute following query

```
SELECT Age FROM Student;
```

Then the result will be

Age
19
19
20
20

Now if we execute following query

```
SELECT UNIQUE Age FROM Student;
```

The result will be

Age
19
20

2.12 Defining Constraints

- We can specify rules for data in a table.
- When the table is created at that time we can define the constraints.
- The constraint can be column level i.e. we can impose constraint on the column and table level i.e we can impose constraint on the entire table.
- There are various types of constraints that can be defined are as follows -

(1) Primary key : The primary key constraint is defined to uniquely identify the records from the table.

The primary key must contain unique values. Hence database designer should choose primary key very carefully.

For example

Consider that we have to create a **perons_details** table with AdharNo, FirstName, MiddleName, LastName, Address and City.

Now making AdharNo as a primary key is helpful here as using this field it becomes easy to identify the records correctly.

The result will be

```
CREATE TABLE person_details (  
AadharNo int,  
FirstName VARCHAR(20),  
MiddleName VARCHAR(20),  
LastName VARCHAR(20),  
Address VARCHAR(30),  
City VARCHAR(10),  
PRIMARY KEY(AadharNo)  
);
```

We can create a composite key as a primary key using CONSTRAINT keyword. For example

```
CREATE TABLE person_details (  
AadharNo int NOT NULL,  
FirstName VARCHAR(20),  
MiddleName VARCHAR(20),  
LastName VARCHAR(20) NOT NULL,  
Address VARCHAR(30),  
City VARCHAR(10),  
CONSTRAINT PK_person_details PRIMARY KEY(AadharNo, LastName)  
);
```

(2) Foreign Key

- Foreign key is used to link two tables.
- Foreign key for one table is actually a primary key of another table.
- The table containing foreign key is called child table and the table containing candidate primary key is called parent key.
- Consider

Employee Table

EmpID	LastName	FirstName	Age
1	Khanna	Rajesh	30
2	Joshi	Sharman	23
3	Kapoor	Tushar	20

Dept Table :

DeptID	DeptName	EmpID
1	Accounts	3
2	Production	3
3	Sales	2
4	Purchase	1

- Notice that the "EmpID" column in the "Dept" table points to the "EmpID" column in the "Employee" table.
- The "EmpID" column in the "Employee" table is the PRIMARY KEY in the "Employee" table.
- The "EmpID" column in the "Dept" table is a FOREIGN KEY in the "Dept" table.
- The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.
- The FOREIGN KEY constraint also prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the table it points to.
- The purpose of the foreign key constraint is to enforce referential integrity but there are also performance benefits to be had by including them in your database design.

The table Dept can be created as follows with foreign key constraint.

```
CREATE TABLE DEPT (
    DeptID      int
    DeptName VARCHAR(20),
    EmpID int,
    PRIMARY KEY(DeptID),
    FOREIGN KEY(EmpID) REFERENCES EMPLOYEE(EmpID)
);
```

(3) Unique

Unique constraint is used to prevent same values in a column. In the EMPLOYEE table, for example, you might want to prevent two or more employees from having an identical **designation**. Then in that case we must use unique constraint.

We can set the constraint as unique at the time of creation of table, or if the table is already created and we want to add the unique constraint then we can use ALTER command.

For example –

```
CREATE TABLE EMPLOYEE(  
    EmpID INT NOT NULL,  
    Name VARCHAR (20) NOT NULL,  
    Designation VARCHAR(20) NOT NULL UNIQUE,  
    Salary DECIMAL (12, 2),  
    PRIMARY KEY (EmpID)  
);
```

If table is already created then also we can add the unique constraint as follows –

```
ALTER TABLE EMPLOYEE  
MODIFY Designation VARCHAR(20) NOT NULL UNIQUE;
```

(4) NOT NULL

- By default the column can have NULL values.
- NULL means unknown values.
- We can set the column values as non NULL by using the constraint NOT NULL.
- For example –

```
CREATE TABLE EMPLOYEE(  
    EmpID INT NOT NULL,  
    Name VARCHAR (20) NOT NULL,  
    Designation VARCHAR(20) NOT NULL,  
    Salary DECIMAL (12, 2) NOT NULL,  
    PRIMARY KEY (EmpID)  
);
```

(5) CHECK

The CHECK constraint is used to limit the value range that can be placed in a column.

For example

```
CREATE TABLE parts (  
    part_no int PRIMARY KEY,  
    description VARCHAR(40),  
    price DECIMAL(10 , 2 ) NOT NULL CHECK(cost > 0)  
);
```

(6) IN operator

The IN operator is just similar to OR operator.

It allows to specify multiple values in WHERE clause.

Syntax

```
SELECT col1,col2,...  
FROM table_name  
WHERE column-name IN (value1, value2, ...);
```

Example

Consider following table

Employee

empID	empName	Salary	DeptID
1	AAA	1000	D101
2	BBB	2000	D102
3	CCC	3000	D103
4	DDD	4000	D104
5	EEE	5000	D105

```
SELECT * FROM Employee
WHERE empID IN (1, 3);
```

The result will be

empID	empName	Salary	DeptID
1	AAA	1000	D101
2	BBB	2000	D102
3	CCC	3000	D103

2.13 Renaming Attributes

The SQL **AS** is used to assign temporarily a new name to a table column or table(relation) itself. One reason to rename a relation is to replace a long relation name with a shortened version that is more convenient to use elsewhere in the query. For example - "Find the names of students and isbn of book who reserve the books".

Student			Reserve		
sid	sname	age	sid	isbn	day
1	Ram	21	1	005	07-07-18
2	Shyam	18	2	007	03-03-18
3	Seeta	16	3	009	05-05-18
4	Geeta	23			

```
Select S.sname,R.isbn
From Student as S, Reserve as R
Where S.sid=R.sid
```

In above case we could shorten the names of tables Student and Reserve as S and R respectively.

Another reason to rename a relation is a case where we wish to compare tuples in the same relation. We then need to take the **Cartesian product** of a relation with itself. For example –

If the query is - Find the names of students who reserve the book of isbn 005. Then the SQL statement will be -

```
Select S.sname,R.isbn
From Student as S, Reserve as R
Where S.sid=R.sid and S.isbn=005
```

2.14 Tuple Variables

Tuple variables can be used in SQL and are defined in the FROM clause.

For example –

```
SELECT DISTINCT cname, C.courseID
FROM Student AS S, Course AS C
WHERE S.courseID = C.courseID
```

Note that we have used the tuple variables as S.courseID and C.courseID. The keyword AS is optional here. Thus tuple variables can be used throughout the expression in SQL.

2.15 Schema Change Statements

SPPU : Nov.-18, (End Sem), Marks 5

Schema can be changed by adding or dropping tables, attributes and constraints. These commands are also known as **schema evolution commands**.

There are two commands that change the schema and those are DROP and ALTER.

Let us discuss these commands in detail

2.15.1 The DROP Command

- The DROP command is used to remove the object (table, domains and constraints) from the database. There are two options for the DROP command - CASCADE and RESTRICT.
- To use the RESTRICT option, the user must first individually drop each element in the schema, then drop the schema itself. That means, the schema is dropped only if it has no elements in it, otherwise the DROP command can not be executed.
- Otherwise to remove completely some database schema CASCADE option is chosen. For example – to remove the **Student_database**

```
DROP SCHEMA Student_database CASCADE;
```

- If the table is to be deleted then the SQL command would be -

```
DROP TABLE Student CASCADE;
```

- The DROP TABLE command not only deletes all the records in the table if successful, but also removes the table definition from the catalog. If it is desired to delete only the records but to leave the table definition for future use, then the DELETE command. For example -
- The following SQL statement deletes all rows in the "Students" table, without deleting the table :

```
DELETE FROM Students;
```

2.15.2 The ALTER Command

There are SQL commands for alteration of table. That means we can add new column or delete some column from the table using these alteration commands.

Syntax for Adding columns

```
ALTER TABLE table_name  
ADD column_name datatype;
```

Example

Consider following table

AadharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. Road	Pune
222	DDD	EEE	FFF	Shivaji nagar	Pune
333	GGG	HHH	III	Chandani chowk	Delhi
444	JJJ	KKK	LLL	Viman nagar	Mumbai

If we execute following command

```
ALTER TABLE Customers  
ADD Email varchar(30);
```


Then the result will be as follows -

AadharNo	FirstName	MiddleName	LastName	Address	City	Email
111	AAA	BBB	CCC	M.G. road	Pune	NULL
222	DDD	EEE	FFF	Shivaji nagar	Pune	NULL
333	GGG	HHH	III	Chandani chowk	Delhi	NULL
444	JJJ	KKK	LLL	Viman nagar	Mumbai	NULL

Syntax for Deleting columns

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

Example

Consider following table -

AadharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. road	Pune
222	DDD	EEE	FFF	Shivaji nagar	Pune
333	GGG	HHH	III	Chandani chowk	Delhi
444	JJJ	KKK	LLL	Viman nagar	Mumbai

If we execute following command

```
ALTER TABLE Customers
DROP COLUMN Address;
```

Then the result will be as follows -

AadharNo	FirstName	MiddleName	LastName	City
111	AAA	BBB	CCC	Pune
222	DDD	EEE	FFF	Pune
333	GGG	HHH	III	Delhi
444	JJJ	KKK	LLL	Mumbai

2.16 Indexes**SPPU : Aug. 17, In Sem, Dec. 18, May 19, End Sem, Marks 5**

Index is a pointer to the data in the database table. These are also called as **lookup tables**. With the help of indexing data retrieval becomes fast and efficient. The concept of index is just similar to the index at the back of the book which contains the keywords. Using these keywords it is easy to locate the desired record quickly from the database table.

1) Creating an Index**Syntax for creating an Index**

```
CREATE INDEX index_name ON table_name;
```

Example

Consider following Book table

isbn	bname	Author
005	DBMS	XYZ
006	OS	PQR
007	DAA	ABC

We can create an index named **idx_isbn** using the field **isbn** of the **Book** table.

```
CREATE INDEX idx_isbn  
ON Book(isbn);
```

Creating Index using UNIQUE**Syntax**

```
CREATE UNIQUE INDEX index_name  
ON table_name (column1, column2, ...);
```

2) Creating Index on Multiple Columns

We can create index using more than one columns of the table. For example – For the above given Book table we can create an index as

```
CREATE INDEX idx_isbn  
ON Book(bname, Author);
```

3) Dropping the Index

The DROP INDEX statement is used to delete an index in a table.

The syntax is

```
DROP INDEX index_name
```

Example 2.16.1 *It is easy to create index on all attributes of any relation, why it is not recommended to create index on all attributes ?*

SPPU : Dec 18, May 19, End Sem, Marks 5

Solution : The index is used to access the desired records efficiently. If we create index on a single attribute then the record can be accessed quickly. As every index requires some disk space and if we create index on all the attributes of any relation then it such index will require more memory.

Secondly, every time when we add or update the record then we have to recalculate the indexes and it will take a lot of time and will cause to perform less efficiently.

Hence it is not recommended to create index on all attributes

Review Question

1. *What is index created on table column ? How performance of SELECT query is improved if index is created on table ?*

SPPU : Aug 17, In Sem, Marks 5

2.17 Aggregate Functions

SPPU : Nov.-17, (End Sem), Marks 2

- An aggregate function allows you to perform a calculation on a set of values to return a single scalar value.
- SQL offers five built-in aggregate functions :
 1. Average : avg
 2. Minimum : min
 3. Maximum : max
 4. Total : sum
 5. Count :
- The aggregate functions that accept an expression parameter can be modified by the keywords DISTINCT or ALL. If neither is specified, the result is the same as if ALL were specified.

DISTINCT	Modifies the expression to include only distinct values that are not NULL
ALL	Includes all rows where expression is not NULL

- Syntax of all the Aggregate Functions

```
AVG( [ DISTINCT | ALL ] expression)
COUNT(*)
COUNT( [ DISTINCT | ALL ] expression )
MAX( [ DISTINCT | ALL ] expression)
MIN( [ DISTINCT | ALL ] expression)
SUM( [ DISTINCT | ALL ] expression)
```

- The **avg** function is used to compute average value. For example – To compute average marks of the students we can use

```
SQL Statement
SELECT AVG(marks)
FROM Students
```

- The **Count** function is used to count the total number of values in the specified field. It works on both numeric and non-numeric data type. COUNT (*) is a special implementation of the COUNT function that returns the count of all the rows in a specified table. COUNT (*) also considers Nulls and duplicates. For example Consider following table

id	value
11	100
22	200
33	300
NULL	400

SQL Statement

```
SELECT COUNT(*)  
FROM Test  
Output  
4  
SELECT COUNT(ALL id)  
FROM Test  
Output  
3
```

- The **min** function is used to get the minimum value from the specified column. For example – Consider the above created **Test table**

```
SQL Statement  
SELECT Min(value)  
FROM Test  
Output  
100
```

- The max function is used to get the maximum value from the specified column. For example - Consider the above created **Test table**

```
SQL Statement  
SELECT Max(value)  
FROM Test  
Output  
400
```

- The sum function is used to get total sum value from the specified column. For example - Consider the above created **Test table**

```
SQL Statement  
SELECT sum(value)  
FROM Test
```

Output

1000

Example 2.17.1 Consider, the following database,*Student(RollNo, Name, Address)**Subject(Sub_code, Sub_Name)**Marks (Roll_no, Sub_code, Marks)**Write following queries in SQL.**Find average marks of each student, along with the name of Student***SPPU : Nov.-17, (End Sem), Marks 2****Solution:**

```
SELECT Name, AVG(Marks)
FROM Student,Marks
WHERE Student.Roll_No=Marks.Roll_No
```

2.18 Set Operations

Set is a collection of elements on which union, intersection and difference operations can be performed.

1) Union : To use this UNION clause, each SELECT statement must have

- i) The same number of columns selected
- ii) The same number of column expressions
- iii) The same data type and
- iv) Have them in the same order

This clause is used to **combine two tables** using UNION operator. It replaces the OR operator in the query. The union operator **eliminates duplicate** while the union all query will retain the duplicates.

Syntax :

The basic syntax of a UNION clause is as follows -

```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
UNION
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```

Here, the given condition could be any given expression based on your requirement.

Example : Find the names of the students who have reserved the 'DBMS' book or 'OS' Book.

The query can then be written by considering the Student, Reserve and Book table as

```
SELECT S.sname
FROM Student S, Reserve R, Book B
WHERE S.sid=R.sid AND R.isbn=B.isbn AND B.bname='DBMS'
UNION
SELECT S.sname
FROM Student S, Reserve R, Book B
WHERE S.sid=R.sid AND R.isbn=B.isbn AND B.bname='OS'
```

- 2) **Intersect** : The common entries between the two tables can be represented with the help of Intersect operator. It replaces the AND operator in the query.

Syntax :

The basic syntax of a INTERSECT clause is as follows-

```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
INTERSECT
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```

Example : Find the students who have reserved both the 'DBMS' book and 'OS' Book

The query can then be written by considering the Student, Reserve and Book table as

```
SELECT S.sid, S.sname
FROM Student S, Reserve R, Book B
WHERE S.sid=R.sid AND R.isbn=B.isbn AND B.bname='DBMS'
INTERSECT
SELECT S.sname
FROM Student S, Reserve R, Book B
WHERE S.sid=R.sid AND R.isbn=B.isbn AND B.bname='OS'
```

- 3) **Except** : The EXCEPT clause is used to represent the set-difference in the query. This query is used to represent the entries that are present in one table and not in other.

Syntax :

The basic syntax of a EXCEPT clause is as follows-

```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
EXCEPT
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```

Example : Find the students who have reserved both the 'DBMS' book but not reserved 'OS' Book

The query can then be written by considering the Student, Reserve and Book table as

```
SELECT S.sid, S.sname
FROM Student S, Reserve R, Book B
WHERE S.sid=R.sid AND R.isbn=B.isbn AND B.bname='DBMS'
EXCEPT
SELECT S.sname
FROM Student S, Reserve R, Book B
WHERE S.sid=R.sid AND R.isbn=B.isbn AND B.bname='OS'
```

2.19 Nested Queries

In nested queries, a **query is written inside a query**. The result of inner query is used in execution of outer query.

There are two types of nested queries :

(i) Independent Query (ii) Corelated Query

i) Independent Query :

- In independent nested queries, query execution starts from innermost query to outermost queries.
- The execution of inner query is independent of outer query, but the result of inner query is used in execution of outer query.
- Various operators like IN, NOT IN, ANY, ALL etc are used in writing independent nested queries.
- For example - Consider three tables namely **Student**, **City** and **Student_City** as follows -

Student		
sid	sname	phone
1	Ram	1111
2	Shyam	2222
3	Seeta	3333
4	Geeta	4444

City	
cid	cname
101	Pune
102	Mumbai
103	Chennai

Student_City	
sid	cid
1	101
1	103
2	101
3	102
4	102
4	103

- **Example 1** - If we want to find out **sid** who live in city 'Pune' or 'Chennai'. We can then write independent nested query using IN operator. Here we can use the IN operator allows you to specify multiple values in a WHERE clause. The IN operator is a shorthand for multiple OR conditions.

Step 1 : Find cid for cname='Pune' or 'Chennai'. The query will be

```
SELECT cid
FROM City
WHERE cname='Pune' or 'Chennai'
```

Step 2 : Using cid obtained in step 1 we can find the sid. The query will be

```
SELECT sid
FROM Student_City
WHERE cid IN
(
  SELECT cid
  FROM City
  WHERE cname='Pune' or cname='Chennai'
)
```

The inner query will return a set with members 101 and 103 and outer query will return those **sid** for which **cid** is equal to any member of set (101 and 103 in this case). So, it will return 1, 2 and 4.

Example 2 : If we want to find out sname who live in city 'Pune' or 'Chennai'.

```
SELECT sname
FROM Student
WHERE sid IN
( SELECT sid
  FROM Student_City
```



```

WHERE cid IN
( SELECT cid
  FROM City
  WHERE cname='Pune' or cname='Chennai'
)
)

```

ii) Co-related Query :

In co-related nested queries, the output of inner query depends on the row which is being currently executed in outer query. For example

If we want to find out sname of Student who live in city with cid as 101, it can be done with the help of co-related nested query as :

```

SELECT sname
FROM Student S
WHERE EXISTS
( SELECT *
  FROM Student_City SC
  WHERE S.sid=SC.sid and SC.cid=101
)

```

Here for each row of **Student S**, it will find the rows from **Student_City** where $S.sid = SC.sid$ and $SC.cid=101$.

If for a **sid** from **Student S**, atleast a row exists in **Student_City SC** with $cid=101$, then inner query will return true and corresponding **sid** will be returned as output.

2.20 Join Operation

SPPU : Nov.-17,19, May-19, Marks 6

The SQL Joins clause is used to combine records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each.

Various types of join operations are –

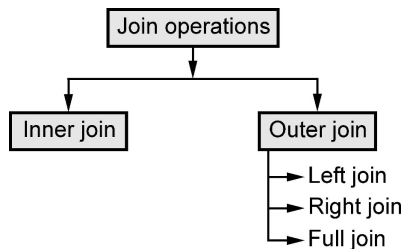


Fig. 2.20.1 Types of join operations

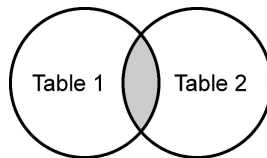
Example : Consider two tables for using the joins in SQL. Note that **cid** is common column in following tables.

sid	cid	sname
1	101	Ram
2	101	Shyam
3	102	Seeta
4	NULL	Geeta

cid	cname
101	Pune
102	Mumbai
103	Chennai

1) Inner Join :

- The most important and frequently used of the joins is the INNER JOIN. They are also known as an EQUIJOIN.
- The INNER JOIN creates a new result table by combining column values of two tables (Table1 and Table2) based upon the join-predicate.
- The query compares each row of Table 1 with each row of Table 2 to find all pairs of rows which satisfy the join-predicate.
- When the join-predicate is satisfied, column values for each matched pair of rows of A and B are combined into a result row. It can be represented as :



- **Syntax :** The basic syntax of the INNER JOIN is as follows.

```
SELECT Table1.column1, Table2.column2...
FROM Table1
INNER JOIN Table2
ON Table1.common_field = Table2.common_field;
```

- **Example :** For above given two tables namely Student and City, we can apply inner join. It will return the record that are matching in both tables using the common column cid.

The query will be

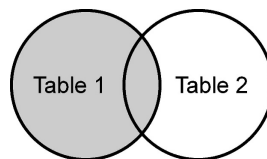
```
SELECT *
FROM Student Inner Join City on Student.cid=City.cid
```

The result will be

sid	cid	sname	cid	cname
1	101	Ram	101	Pune
2	101	Shyam	101	Pune
3	102	Seeta	102	Mumbai

2) Left Join(Outer Join) :

- The SQL LEFT JOIN returns all rows from the left table, even if there are no matches in the right table. This means that if the ON clause matches 0 (zero) records in the right table; the join will still return a row in the result, but with NULL in each column from the right table.
- This means that a left join returns all the values from the left table, plus matched values from the right table or NULL in case of no matching join predicate.
- It can be represented as –



- **Syntax :** The basic syntax of a LEFT JOIN is as follows.

```
SELECT
SELECT Table1.column1, Table2.column2...
FROM Table1
LEFT JOIN Table2
ON Table1.common_field = Table2.common_field;
```

- **Example :** For above given two tables namely Student and City, we can apply Left join. It will Return all records from the left table, and the matched records from the right table using the common column cid. The query will be

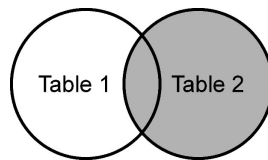
```
SELECT *
FROM Student Left Join City on Student.cid=City.cid
```

The result will be

sid	cid	sname	cid	cname
1	101	Ram	101	Pune
2	101	Shyam	101	Pune
3	102	Seeta	102	Mumbai
4	NULL	Geeta	NULL	NULL

3) Right Join(Outer Join) :

- The SQL RIGHT JOIN returns all rows from the right table, even if there are no matches in the left table.
- This means that if the ON clause matches 0 (zero) records in the left table; the join will still return a row in the result, but with NULL in each column from the left table.
- This means that a right join returns all the values from the right table, plus matched values from the left table or NULL in case of no matching join predicate.
- It can be represented as follows :



- **Syntax :** The basic syntax of a RIGHT JOIN is as follow -

```
SELECT Table1.column1, Table2.column2...
FROM Table1
RIGHT JOIN Table2
ON Table1.common_field = Table2.common_field;
```

- **Example :** For above given two tables namely Student and City, we can apply Right join. It will return all records from the right table, and the matched records from the left table using the common column cid. The query will be

```
SELECT *
FROM Student Right Join City on Student.cid=City.cid
```

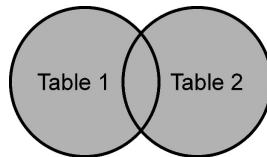
The result will be –

sid	cid	sname	cid	cname
1	101	Ram	101	Pune
2	101	Shyam	101	Pune
3	102	Seeta	102	Mumbai
NULL	NULL	NULL	103	Chennai

4) Full Join (Outer Join) :

- The SQL FULL JOIN combines the results of both left and right outer joins.
- The joined table will contain all records from both the tables and fill in NULLs for missing matches on either side.

- It can be represented as,



- **Syntax** : The basic syntax of a FULL JOIN is as follows :

```
SELECT Table1.column1, Table2.column2...
FROM Table1 FULL JOIN Table2 ON Table1.common_field = Table2.common_field;
```

The result will be -

- **Example** : For above given two tables namely **Student and City**, we can apply full join. It will return returns rows when there is a match in one of the tables using the common column cid. The query will be -

```
SELECT *
FROM Student Full Join City on Student.cid=City.cid
```

The result will be -

sid	cid	sname	cid	cname
1	101	Ram	101	Pune
2	101	Shyam	101	Pune
3	102	Seeta	102	Mumbai
4	NULL	Geeta	NULL	NULL
NULL	NULL	NULL	103	Chennai

2.21 Views

SPPU : July-18, Oct.-18, (In Sem), Nov.-19, Marks 8

- Views in SQL are kind of **virtual tables**.
- A view also has rows and columns as they are in a real table in the database.
- We can create a view by selecting fields from one or more tables present in the database.
- A view can either have all the rows of a table or specific rows based on certain condition.

Creating View

We can create a view using CREATE VIEW statement. The syntax is

```
CREATE VIEW name_of_view AS
SELECT column1,column2,...
FROM table_name1,table_name2,...
WHERE condition;
```

Example

- i) **Creating a view using single table** : Consider table Employee and create a view EmployeeDetails whose Salary is < 10000

EmpID	EName	Salary
101	Archana	20000
102	Madhura	5000
103	Poonam	8000
104	Sharda	15000
105	Monika	7000

```
CREATE VIEW EmployeeDetails(EmpID, EName) AS
SELECT E.EmpID, E.EName
FROM Employee E
WHERE E.Salary > 10000
```

The **Output** will be -

EmpID	EName
102	Madhura
103	Poonam
105	Monika

- ii) **Creating a view from multiple table** : In this example we will create a view from two tables Employee and Department.

Employee

EmpID	EName	Salary
101	Archana	20000
102	Madhura	5000
103	Poonam	8000
104	Sharda	15000
105	Monika	7000

Department

EmpID	DName
101	Accounts
104	Sales
105	Sales

Now we need to create a view named Employee_dept_Details in which the names and salary of employees belonging to Sales department is displayed. The SQL statement will then be –

```
CREATE VIEW Employee_dept_Details(EName,Salary) AS
SELECT E.EName,E.Salary
FROM Employee E, Department D
WHERE E.EmpID=D.EmpID AND D.DName='Sales'
```

The output will be

EName	Salary
Sharda	15000
Monika	7000

View Update

- The SQL UPDATE VIEW command can be used to modify the data of a view.
- All views are not updatable. So, UPDATE command is not applicable to all views.
- An updatable view is one which allows performing a UPDATE command on itself without affecting any other table.

- **Syntax for updating view**

```
UPDATE < view_name >  
SET <column1>=<value1>,<column2>=<value2>,....  
WHERE <condition>;
```

- **Example**

```
UPDATE view Employee_dept_Details  
SET Salary=1000  
WHERE EName='Monika';  
This view can be viewed by using following query  
SELECT * FROM Employee_dept_Details;
```

Rules for updating the views

The view can be updated in following conditions :

- (1) The view can be defined based on one and only one table.
- (2) The SELECT statement should not have DISTINCT keyword.
- (3) The view should not have all NOT NULL values.
- (4) The view should not be created from nested and complex queries.
- (5) The SELECT statement which is used to create the view should not include GROUP BY clause or ORDER BY clause.
- (6) The view should not have any field made out of aggregate functions.
- (7) Any selected output fields of the view must not use constants, strings or value expressions.

Dropping View

For deleting a view, the DROP command is used.

Syntax

```
DROP VIEW view_name;
```

Example

```
DROP VIEW Employee_dept_Details;
```

Difference between View and Table

- Table is a database object that contains the data in row and column form. View is also database object and it is virtual table which is built on the top of the other tables.
- The table contains the data while view does not hold data itself.
- A table is designed with a limited number of columns and an unlimited number of rows while a view is designed as a virtual table that is extracted from a database.

Example 2.21.1 Consider following company database :

EMP(Name, SSn, Salary, Supersn, dno)

DEPT (dnum, dname, mgrssn)

DEPT_LOC (dnum, dlocation)

PROJECT(Pname, Pnumber, Plocation, dnum)

WORKS_ON(Essn, dept_name, sex)

Write SQL queries for the following:

- i) Retrieve the names of all employees who work in the department that has the employee with the highest salary among all employees.
- ii) Retrieve the names of employees who make atleast 10000 more than the employee who is paid the least in the company.
- iii) A view that has the employee name, supervisor name and employee salary for each employee who works in the Research' department.
- iv) A view that has the project name, controlling department name, number of employees and total hours worked per week on the project for each project with more than one employee working on it.

Solution : (i)

```
SELECT Name FROM EMP WHERE dno =
      ( SELECT dno FROM EMP WHERE Salary =
        ( SELECT MAX(Salary) FROM EMP ) )
```

(ii)

```
SELECT Name FROM EMP WHERE Salary >= 10000 +
      ( SELECT MIN(Salary) FROM EMP )
```

(iii)

```
CREATE VIEW RESEARCH_EMPLOYEE_INFO (Lname, Fname, Supervisor, Salary) AS
SELECT E.Lname, E.Fname, S.Lname, E.Salary FROM
EMPLOYEE E, EMPLOYEE S, DEPARTMENT D
WHERE Dname = 'Reasearch' AND
D.Dnumber = E.Dno AND E.super_ssn = S.ssn;
```

(iv)

```
CREATE VIEW PROJECT_INFO AS
SELECT Pname, Dname, COUNT(WO.Essn), SUM(WO.Hours)
FROM PROJECT P, DEPARTMENT D, WORKS_ON WO
WHERE
P.Dnum = D.Dnumber AND
P.Pnumber = WO.Pno GROUP BY Pno
HAVING COUNT(WO.Essn) > 1
```

Review Question

1. What is the use of database view ? What are the mandatory requirements of updating the values in view ? Explain with example.

SPPU : Oct.-18, (In Sem), Marks 5**2.22 Examples Based on SQL**

Example 2.22.1 For the following database, identify primary key and foreign key where-ever applicable and solve the given queries in SQL.

Item (ino, description, unit_price)

Supplier (sno, sname, address)

Supplied (sno, ino, sdate, qty, per_unit_discount)

1. Find supplier name for the suppliers who supply every item.
2. Find distinct item names for the items supplied with total discount > 500.
3. Pair of supplier names supplying same items on same dates.

Solution :

- The primary key for Item table is **ino**.
- The primary key for supplier table is **sno**
- The primary key for supplied table is **(sno,ino)**

1) Find supplier name for the suppliers who supply every item.

```
SELECT S.sname
FROM Supplier S
WHERE NOT EXISTS (
    ( SELECT * FROM Item I
      WHERE NOT EXISTS
        ( SELECT *
          FROM Supplied SP
          WHERE SP.sno = S.sno
            AND SP.ino=I.ino )
    )
)
```

2) Find distinct item names for the items supplied with total discount > 500.

```
SELECT DISTINCT description
FROM Item, Supplied
WHERE Item.ino=Supplied.ino
AND Supplied.per_unit_discount >500
```

3) Pair of supplier names supplying same items on same dates.

```
SELECT sname
FROM Supplier
WHERE Supplied.sno=Supplier.sno
AND
Item.ino=Supplied.ino
```

Example 2.22.2 Write SQL statements for the following (any five)

Consider the following database

pilot (pid, pname)

flight (fid, ftype, capacity)

route (pid, fid, from_city, to_city)

i) List the details of flights having capacity more than 300.

ii) List the flights between 'Surat' and 'Mumbai'.

iii) List the names of the pilots who fly from 'Pune'.

iv) List the route on which, pilot named 'Mr Kapoor' flies

v) List the pilots whose names, starts with letter 'A' '%' but does not end with letter 'A'.

vi) List the name of pilots who fly 'boing 737' type of flights.

Solution :

(i) List the details of flights having capacity more than 300.

```
SELECT * FROM flight
WHERE capacity>300
```

(ii) List the flights between 'Surat' and 'Mumbai'.

```
SELECT fid
FROM flight, route
WHERE
flight.fid=route.fid AND
route.from_city='Surat' AND route.to_city='Mumbai';
```

(iii) List the names of the pilots who fly from 'Pune'.

```
SELECT pname
FROM pilot,route
WHERE pilot.pid=route.pid
AND route.from_city='Pune';
```

(iv) List the route on which, pilot named 'Mr Kapoor' flies.

```
SELECT from_city, to_city
FROM route, pilot
WHERE pilot.pid=route.pid
AND pilot.pname='Mr.Kapoor';
```

(v) List the pilots whose names, starts with letter 'A' '%' but does not end with letter 'A'.

```
SELECT pname
FROM pilot
WHERE pname LIKE 'A%' MINUS
SELECT pname FROM pilot
WHERE pname LIKE '%A';
```

(vi) List the name of pilots who fly 'boing 737' type of flights.

```
SELECT pname
FROM pilot, flight, route
WHERE flight.ftype = 'boing 737' AND
pilot.pid = route.pid AND
route.fid = flight.fid;
```

Example 2.22.3 Consider the relation Database.

Person (SSN, Name, city)

Car (License_no, year, Model, SSN)

Accident(drive_no, SSN, license_no, accidentyear, damage_Amt)

Query :

1. Find out total no of cars that had accident in 1988.
2. Find the Name of driver who did not have an accident in 'Delhi'.
3. Find the car, who don't have total damage of more than ₹1000.
4. Find the cars sold in 2006 and whose owner are from 'Vadodara'
5. How many different models of car are used by 'Mr.abc'
6. Find the lucky persons who have not met any accident yet.

Solution :**1. Find out total no of cars that had accident in 1988.**

```
SELECT count(License_no)
FROM Car, Accident
WHERE Accident.accidentyear=1988
```

2. Find the Name of driver who did not have an accident in 'Delhi'.

```
SELECT Name
FROM Person,Accident
WHERE Person.SSN=Accident.SSN AND Peson.city<> 'Delhi'
```

3. Find the car, who don't have total damage of more than ` 1000.

```
SELECT License_no
FROM Car, Accident
WHERE Accident.damage_amt<1000 AND
Car.License_no=Accident.licence_no;
```

4. Find the cars sold in 2006 and whose owner are from 'Vadodara'

```
SELECT license_no
FROM Car, Person
WHERE Car.SSN = Person. SSN AND
Car.year = 2006 and Person.city = 'Vadodara';
```

5. How many different models of car are used by 'Mr.abc'

```
SELECT count(model)
FROM Car, Person
WHERE Car.SSN = Person. SSN AND
Person.Name = 'Mr.abc';
```

6. Find the lucky persons who have not met any accident yet.

```
SELECT Name
FROM Person
WHERE person.SSN NOT IN (SELECT SSN FROM Accident);
```

Example 2.22.4 We have following relations :

Supplier (S#, sname, status, city)

Parts (P#, pname, color, weight, city)

SP(S#, P#, quantity)

Answer the following queries in SQL,

- i) Find name of supplier for city = 'Delhi'.*
- ii) Find suppliers whose name start with 'AB'.*
- iii) Find all suppliers whose status is 10, 20 or 30.*
- iv) Find total number of city of all suppliers.*
- v) Find s# of supplier who supplies 'red' part.*
- vi) Count number of supplier who supplies 'red' part.*
- vii) Sort the supplier table by sname.*

Solution : i) Find name of supplier for city = 'Delhi'.

```
SELECT sname
FROM Supplier
WHERE city='Delhi'
```

ii) Find suppliers whose name start with 'AB'.

```
SELECT sname
FROM Supplier
WHERE sname='AB%'
```

iii) Find all suppliers whose status is 10, 20 or 30.

```
SELECT sname
FROM Supplier
WHERE status BETWEEN 10 AND 30
```

iv) Find total number of city of all suppliers.

```
SELECT count(*) FROM (SELECT DISTINCT CITY FROM Supplier);
```

v) Find s# of supplier who supplies 'red' part.

```
SELECT DISTINCT supplier.S#  
FROM Supplier,Parts, SP  
WHERE Supplier.S# = SP.S#  
AND SP.P# = Parts.P# AND Parts.color = 'red';
```

vi) Count number of supplier who supplies 'red' part.

```
SELECT count(*)  
FROM (SELECT DISTINCT Supplier.S#  
FROM Supplier,Parts, SP  
WHERE Supplier.S# = SP.S#  
AND SP.P# = Parts.P# AND Parts.color = 'red');
```

vii) Sort the supplier table by sname.

```
SELECT *  
FROM Supplier  
ORDER BY sname;
```

Example 2.22.5 We have following relations :

Supplier (S#, sname, status, city)

Parts (P#, pname, color, weight, city)

SP(S#, P#, quantity)

Answer the following queries in SQL,

i) Delete records in supplier table whose status is 40.

ii) Add one field in supplier table.

Solution :

i) Delete records in supplier table whose status is 40.

```
DELETE FROM Supplier  
WHERE status=40
```

ii) Add one field in supplier table.

```
ALTER TABLE Supplier  
ADD(PhoneNo number);
```

Example 2.22.6 We have following relations :

Supplier (S#, sname, status, city)

Parts (P#, pname, color, weight, city)

SP(S#, P#, quantity)

Answer the following queries in SQL,

i) Find name of parts whose color is 'red'.

ii) Find parts name whose weight less than 10 kg.

iii) Find all parts whose weight from 10 to 20 kg.

- iv) Find average weight of all parts.*
- v) Find S# of supplier who supply part 'p2'.*
- vi) Find name of supplier who supply maximum parts.*
- vii) Sort the parts table by pname.*

Solution :**i) Find name of parts whose color is 'red'.**

```
SELECT pname
FROM Parts
WHERE Parts.color = 'red';
```

ii) Find parts name whose weight less than 10 kg.

```
SELECT pname
FROM Parts
WHERE Parts.weight < 10;
```

iii) Find all parts whose weight from 10 to 20 kg.

```
SELECT pname, weight
FROM Parts
WHERE Parts.weight BETWEEN 10 AND 20;
```

iv) Find average weight of all parts.

```
SELECT AVG (weight)
FROM Parts;
```

v) Find S# of supplier who supply part 'p2'.

```
SELECT S#
FROM SP
WHERE p# = 'p2';
```

vi) Find name of supplier who supply maximum parts.

```
SELECT sname, MAX(quantity)
FROM Supplier, Parts, SP
WHERE Supplier.S# = SP.S#
AND SP.P# = P.P#
GROUP BY sname;
```

vii) Sort the parts table by pname.

```
SELECT *
FROM Parts
ORDER BY pname;
```

Example 2.22.7 We have following relations :

Supplier (S#, sname, status, city)

Parts (P#, pname, color, weight, city)

SP(S#, P#, quantity)

Answer the following query in SQL,

Delete records in parts table whose color is 'blue'.

Solution : (i) DELETE FROM Parts

```
WHERE color='blue'
```

Example 2.22.8 Consider following schema and write SQL for given statements.

student (rollno, name, branch)

exam(rollno, subject_code, obtained_marks, paper_code)

papers(paper_code, paper_satter_name, university)

i) Display name of student who got first class in subject '130703'.

ii) Display name of all student with their total mark.

iii) Display list number of student in each university.

iv) Display list of student who has not given any exam.

Solution :

i) Display name of student who got first class in subject '130703'.

```
SELECT name
FROM student,exam
WHERE exam.obtained_marks>60 AND exam.subject_code='130703' AND
student.rollno=exam.rollno
```

ii) Display name of all student with their total mark.

```
SELECT name,SUM(obtained_marks)
FROM student,exam
WHERE student.rollno=exam.rollno
```

iii) Display list number of student in each university.

```
SELECT COUNT(Rollno)
FROM student, exam, papers
WHERE student.rollno=exam.rollno AND exam.paper_code=papers.paper_code;
```

iv) Display list of student who has not given any exam.

```
SELECT name
FROM student NOT IN
(SELECT rollno
FROM student, exam
WHERE student.rollno=exam.rollno)
```


Example 2.22.9 Write down the query for the following table where primary keys are underlined.

Person(ss#, name, address)

Car(license, year, model)

Accident(date, driver, damage-amount)

Owns(ss#, license)

Log(license, date, driver)

1) Find the total number of people whose cars were involved in accidents in 2009.

2) Find the number of accidents in which the cars belonging to "S.Sudarshan"

3) Add a new customer to the database.

4) Add a new accident recorded for the santro belonging to "KORTH"

Solution :

1) Find the total number of people whose cars were involved in accidents in 2009.

```
SELECT count(ss#)
FROM person ,owns,accident,log
WHERE person.ss#=owns.ss# AND
owns.licence=log.licence AND
log.driver=accident.driver AND
accident.date >= '01-jan-2009' and accident.date <='31-dec-09;
```

2) Find the number of accidents in which the cars belonging to "S.Sudarshan"

```
SELECT COUNT(accident.date)
FROM person ,owns,accident,log
WHERE log.date=accident.date AND log.driver=accident.driver AND
own.licence=log.licence AND
person.ss#=own.ss# AND
person.name = 's.sudarshan';
```

3) Add a new customer to the database.

```
INSERT INTO person (ss#, name, address) values (101, 'Madhav', 'Pune');
INSERT INTO owns (ss#, license) values (101, 'L101');
INSERT INTO car (license, year, model) values ('L101', 2017, 'Honda');
INSERT INTO log (license, date, driver) values ('L101', NULL, 'Shankar');
```

4) Add a new accident recorded for the santro belonging to "KORTH"

```
INSERT INTO accident (date, driver, damage-amount) VALUES ('31-Dec-2016', 'Shankar', 10000);
INSERT INTO log (license, date, driver) VALUES ('L111', '31-Dec-2016', 'Shankar');
INSERT INTO car (license, year, model) VALUES ('L111', '2005', 'SANTRO');
INSERT INTO person (ss#, name, address) VALUES (111, 'Korth', 'Mumbai');
INSERT INTO owns (ss#, license) VALUES (111, 'L111');
```

Example 2.22.10 Consider the employee data. Give an expression in SQL for the following query :

Employee(employee_name, street, city)

Works(employee_name, company_name, salary)

Company(company_name, city)

Manages(employee_name, manager_name)

- 1) Find the name of all employees who work for State Bank.
- 2) Find the names and cities of residence of all employees who work for State Bank.
- 3) Find all employee in the database who do not work for State Bank.
- 4) Find all employee in the database who earn more than every employee of UCO bank.

Solution :

1) Find the name of all employees who work for State Bank.

```
SELECT employee_name
FROM Works
WHERE company_name='State Bank';
```

2) Find the names and cities of residence of all employees who work for State Bank.

```
SELECT employee_name, city
FROM Employee, Works
WHERE company_name='State Bank' AND
Works.employee_name=Employee.employee_name;
```

3) Find all employee in the database who do not work for State Bank.

```
SELECT employee_name
FROM Works
WHERE company_name <> 'State Bank';
```

4) Find all employee in the database who earn more than every employee of UCO bank.

```
SELECT employee_name from Works
WHERE salary > (SELECT MAX(salary) FROM Works
WHERE company_name='UCO bank');
```

Example 2.22.11 Consider following schema and write SQL for given statements.

Student(Rollno, Name, Age, Sex, City).

Student_marks(Rollno, Sub1, Sub2, Sub3, Total, Average)

Write query to

- i) Calculate and store total and average marks from sub1, sub2 and sub3.
- ii) Display name of students who got more than 60 marks in subject Sub1.
- iii) Display name of students with their total and average marks.
- iv) Display name of students who got equal marks in subject sub2.

Solution :**i) Calculate and store total and average marks from sub1, sub2 and sub3.**

```
UPDATE Student_marks
SET Total=sub1+sub2+sub3,
Average= (sub1+sub2+sub3)/3;
```

ii) Display name of students who got more than 60 marks in subject Sub1.

```
SELECT Name
FROM Student, Student_marks
WHERE Student.Rollno=Student_marks.Rollno AND Student_marks.sub1>60
```

iii) Display name of students with their total and average marks.

```
SELECT Name, Total, Average
FROM Student, Student_marks
WHERE Student.Rollno=Student_marks.Rollno
```

iv) Display name of students who got equal marks in subject sub2.

```
SELECT Name
FROM Student S, Student_marks SM1, Student_marks SM2
WHERE S.Rollno=SM1.Rollno AND SM1.sub2=SM2.sub2;
```

Example 2.22.12 *We have following relations.*

EMP (empno, ename, jobtitle, manager no, hiredate, sal, comm, dept no)

DEPT (dept no, dname, loc)

- i) The employees who are getting salary greater than 3000 for those persons belongings to the department 20*
- ii) Employees who are not getting any commission.*
- iii) Find how many job titles are available in employee table.*
- iv) Display total salary spent for each job category.*
- v) Display number of employees working in each department and their department name.*
- vi) List ename whose manager is NULL.*
- vii) List all employee names and their salaries, whose salary lies between 1500/- and 3500/- both inclusive.*

Solution : i) The employees who are getting salary greater than 3000 for those persons belongings to the department 20

```
SELECT ename
FROM EMP
WHERE EMP.sal>3000 AND EMP.dept_no=20
```

ii) Employees who are not getting any commission.

```
SELECT ename
FROM EMP
WHERE EMP.comm IS NULL;
```

iii) Find how many job titles are available in employee table.

```
SELECT count(jobtitle)
FROM EMP
```

iv) Display total salary spent for each job category.

```
SELECT ename,SUM(sal)
FROM EMP
GROUP BY jobtitle;
```

v) Display number of employees working in each department and their department name.

```
SELECT COUNT(EMP.eno), DEPT.dname
FROM EMP, DEPT
WHERE EMP.dept_no=DEPT.dept_no
GROUP BY DEPT.dept_no;
```

vi) List ename whose manager is NULL.

```
SELECT ename
FROM EMP
WHERE manager_no IS NULL;
```

vii) List all employee names and their salaries, whose salary lies between 1500/- and 3500/- both inclusive.

```
SELECT ename, sal
FROM EMP
WHERE sal >= 1500 AND sal <= 3500;
```

Example 2.22.13 *We have following relations.*

EMP (empno, ename, jobtitle, manager no, hiredate, sal, comm, dept no)

DEPT (dept no, dname, loc)

Answer the following queries in SQL

i) Find the employees working in the department 10, 20, 30 only.

ii) Find employees whose names start with letter A or letter a.

iii) Find employees along with their department name.

iv) Find employees whose manager is KING.

v) Find the employees who are working in smith's department.

vi) Find the employees who get salary more than Allen's salary.

vii) Display employees who are getting maximum salary in each department.

Solution : i) Find the employees working in the department 10, 20, 30 only.

```
SELECT empno
FROM EMP
WHERE dept_no BETWEEN 10 AND 30
```

ii) Find employees whose names start with letter A or letter a.

```
SELECT ename
FROM EMP
WHERE ename='A%' OR ename='a%';
```

iii) Find employees along with their department name.

```
SELECT EMP.ename,DEPT.dname
FROM EMP, DEPT
WHERE EMP.dept_no=DEPT.dept_no ;
```

iv) Find employees whose manager is KING.

```
SELECT ename
FROM EMP
WHERE managerno = (SELECT empno FROM EMP WHERE ename='KING');
```

v) Find the employees who are working in smith's department.

```
SELECT ename
FROM EMP,DEPT
WHERE EMP.dept_no=DEPT.dept_no AND ename='Smith';
```

vi) Find the employees who get salary more than Allen's salary.

```
SELECT ename
FROM EMP
WHERE sal > (SELECT sal FROM EMP WHERE ename='Allen');
```

vii) Display employees who are getting maximum salary in each department.

```
SELECT ename, MAX(sal)
FROM EMP
GROUP BY dept_no;
```

Example 2.22.14 Write queries for the following table.

T1 (Empno, Ename, Salary, Designation),

T2 (Empno, Deptno.)

i) Display all rows for salary greater than 5000

ii) Display the deptno for the ename='shyam'.

iii) Add a new column deptname in table T2.

iv) Change the designation of ename='ram' from 'clerk' to 'senior clerk'.

v) Find the total salary of all the rows.

vi) Display Empno, Ename, Deptno and Deptname.

vii) Drop the table T1.

Solution : i) Display all rows for salary greater than 5000

```
SELECT *  
FROM T1  
WHERE Salary > 5000
```

ii) Display the deptno for the ename='shyam'.

```
SELECT deptno  
FROM T1, T2  
WHERE T1.Empno = T2.Empno AND T1.Ename = 'shyam';
```

iii) Add a new column deptname in table T2.

```
ALTER TABLE T2  
ADD (deptname VARCHAR(20));
```

iv) Change the designation of ename='ram' from 'clerk' to 'senior clerk'.

```
UPDATE T1  
SET T1.designation = 'Senior Clerk'  
WHERE T1.ename = 'ram';
```

v) Find the total salary of all the rows.

```
SELECT SUM(Salary)  
FROM T1;
```

vi) Display Empno, Ename, Deptno and Deptname.

```
SELECT E.Empno, E.Ename, D.Deptno, D.Deptname  
FROM T1 E, T2 D  
WHERE E.Empno = D.Empno;
```

vii) Drop the table T1.

```
DROP Table T1;
```

Example 2.22.15 Solve following queries with following table, where underlined attribute is primary key.

primery key.

Person(SS#, name, address)

Car(license, year, model)

Accident(date, driver, damage-amount)

Owns(SS#, license)

Log(licence, date, driver)

i) Find the name of person whose license number is '12345'.

ii) Display name of driver with number of accidents done by that driver.

iii) Add a new accident by 'Ravi' for 'BMW' car on 01/01/2013 for damage amount of 1.5 lakh rupees.

Solution :**i) Find the name of person whose license number is '12345'.**

```
SELECT name
FROM Person, Owns, Car
WHERE Person.SS#=Owns.SS# AND
Car.license=Owns.license AND
Car.license='12345';
```

ii) Display name of driver with number of accidents done by that driver.

```
SELECT driver,COUNT(*)
FROM Accident
GROUP BY driver
```

iii) Add a new accident by 'Ravi' for 'BMW' car on 01/01/2013 for damage amount of 1.5 lakh rupees.

```
INSERT INTO Person (SS#, name, address) VALUES(111,'Ravi','Mumbai');
INSERT INTO Car (license,year,model) VALUES('L111','2008','BMW');
INSERT INTO Owns(SS#, license) VALUES(111, 'L111');
INSERT INTO Accident('01/01/2013','Ravi',150000);
INSERT INTO Log('L111', '01/01/2013','Ravi');
```

Example 2.22.16 *For Supplier - Parts database*

Supplier(S#, sname, status, city)

Parts(P#, pname,color,weight,city)

SP(S#, P#, quantity)

Answer the following queries in SQL.

i) Display the name of supplier who lives in 'Ahemdabad'.

ii) Display the parts name which is not supplied yet.

iii) Find all suppliers whose status is either 20 or 30.

Solution :**i) Display the name of supplier who lives in 'Ahemdabad'.**

```
SELECT sname
FROM Supplier
WHERE city='Ahemdabad';
```

ii) Display the parts name which is not supplied yet.

```
SELECT pname
FROM Parts, SP
WHRE SP.P# <> Parts.P#
```

iii) Find all suppliers whose status is either 20 or 30.

```
SELECT sname
FROM Supplier
WHERE status = 20 or status = 30;
```

Example 2.22.17 For Supplier - Parts database

Supplier(S#, sname, status, city)

Parts(P#, pname,color,weight,city)

SP(S#, P#, quantity)

Answer the following queries in SQL.

- i) Find the name of parts having 'Red' colour.*
- ii) Delete parts whose weight is more than 100 gram.*
- iii) Count how many times each supplier has supplied part 'P2'.*
- iv) How much times shipment is for more than 100 quantities ?*

Solution :**i) Find the name of parts having 'Red' colour.**

```
SELECT pname
FROM Parts
WHERE color='Red';
```

ii) Delete parts whose weight is more than 100 gram.

```
DELETE FROM Parts
WHERE weight>100
```

iii) Count how many times each supplier has supplied part 'P2'.

```
SELECT S#, COUNT(*)
FROM SP
WHERE P#='P2'
```

iv) How much times shipment is for more than 100 quantities?

```
SELECT S#, COUNT(*)
FROM SP
WHERE quantity>100;
```

Example 2.22.18 Consider following schema and write SQL for given statements.

student(RollNo, Name, Age, Sex, City)

Student_marks(RollNo, Sub1, Sub2, Sub3, Total, Average)

Write query to

- i) Display name and city of students whose total marks are greater than 225.*
- ii) Display name of students who got more than 60 marks in each subject.*
- iii) Display name of city from where more than 10 students come from.*
- iv) Display a unique pair of male and female students.*

Solution :**i) Display name and city of students whose total marks are greater than 225.**

```
SELECT name, city
FROM student,student_marks
WHERE student.RollNo = student_marks.RollNo AND student_marks.Total>225
```

ii) Display name of students who got more than 60 marks in each subject.

```
SELECT name
FROM student,student_marks
WHERE student.RollNo = student_marks.RollNo AND
student_marks.Sub1>60 OR student_marks.Sub2>60 OR student_marks.Sub3>60;
```

iii) Display name of city from where more than 10 students come from.

```
SELECT city
FROM student
WHERE count(RollNo)>10
```

iv) Display a unique pair of male and female students.

```
SELECT S1.name
FROM Student S1, Student S2
WHERE S1.Name=S2.Name AND S1.sex='M' S2.sex='F'
```

Example 2.22.19 C Write queries for the following tables :

T1 (Empno, Ename, Salary, Designation)

T2 (Empno, Deptno.)

- 1) Display all the details of the employee whose salary is lesser than 10 K.
- 2) Display the Deptno in which employee Seeta is working.
- 3) Add a new column Deptname in table T2.
- 4) Change the designation of Geeta from 'Manager' to 'Senior Manager'.
- 5) Find the total salary of all the employees.
- 6) Display Empno, Ename, Deptno and Deptname
- 7) Drop the table T1.

Solution :**1) Display all the details of the employee whose salary is lesser than 10 K.**

```
SELECT *
FROM T1
WHERE Salary<10000
```

2) Display the Deptno in which employee Seeta is working.

```
SELECT Deptno
FROM T2, T1
WHERE T1.Empno=T2.Empno AND T1.Ename='Seeta';
```

3) Add a new column Deptname in table T2.

```
ALTER TABLE T2
ADD (Deptname VARCHAR(20));
```

4) Change the designation of Geeta from 'Manager' to 'Senior Manager'.

```
UPDATE T1
SET Designation = 'Senior Manager'
WHERE Ename='Geeta';
```

5) Find the total salary of all the employees.

```
SELECT SUM(Salary)
FROM T1
```

6) Display Empno, Ename, Deptno and Deptname

```
SELECT T1.Empno, T1.Ename, T2.Deptno, T2.Deptname
FROM T1, T2
```

7) Drop the table T1.

```
DROP TABLE T1
```

Example 2.22.20 We have following relations :

EMP(empno, ename, jobtitle, managerno, hiredate, sal, comm, deptno)

DEPT(deptno, dname, loc)

Answer the following queries in SQL.

- i) Find the Employees working in the department 10, 20, 30 only.
- ii) Find Employees whose names start with letter A or letter a.
- iii) Find Employees along with their department name.
- iv) Insert data in EMP table.
- v) Find the Employees who are working in Smith's department
- vi) Update Department name of Department No = 10
- vii) Display employees who are getting maximum salary in each department

Solution :

i) Find the Employees working in the department 10, 20, 30 only.

```
SELECT ename
FROM EMP
WHERE deptno BETWEEN 10 AND 30
```

ii) Find Employees whose names start with letter A or letter a.

```
SELECT ename
FROM EMP
WHERE ename='A%' OR ename='a%';
```

iii) Find Employees along with their department name.

```
SELECT EMP.ename,DEPT.dname
FROM EMP, DEPT
WHERE EMP.deptno=DEPT.deptno ;
```

iv) Insert data in EMP table.

```
INSERT INTO EMP( empno, ename, jobtitle, managerno, hiredate, sal, comm, deptno)
VALUES ('E111','AAA','Manager',M123,01-01-2010,20000,2000,'D111');
```

v) Find the Employees who are working in Smith's department

```
SELECT ename
FROM EMP,DEPT
WHERE EMP.deptno=DEPT.deptno AND ename='Smith';
```

vi) Update Department name of Department No = 10

```
UPDATE DEPT
SET dname='Accounts'
WHERE deptno=10;
```

vii) Display employees who are getting maximum salary in each department

```
SELECT ename, MAX(sal)
FROM EMP
GROUP BY deptno;
```

Example 2.22.21 Consider following Hotel database, primary keys are underlined :

hotel(hotel_no,name,type,price)

room(room-no,hotel-no,type,price)

booking(hotel-no,guest-no,date-from,date-to,room-no)

guest(guest-no,name,address)

Give an expression in SQL for each of the following queries

- (1) List the names and addresses of all guests in London, alphabetically ordered by name.
- (2) List out hotel name and total number of rooms available
- (3) List the details of all the rooms at the Grosvenor Hotel, including the name of the guest staying in the room, if the room is occupied.
- (4) List all guests currently staying at the Grosvenor Hotel.
- (5) List the rooms that are currently unoccupied at the Grosvenor Hotel.
- (6) List the number of rooms in each hotel in London.
- (7) List out all guests who have booked room for three or more days.

Solution :**(1) List the names and addresses of all guests in London, alphabetically ordered by name.**

```
SELECT name, address
FROM guest
WHERE address LIKE '%London%'
ORDER BY name;
```

(2) List out hotel name and total number of rooms available

```
SELECT name, COUNT(room-no)
FROM hotel, room
WHERE hotel.hotel_no=room.hotel_no
GROUP by hotel_no;
```

(3) List the details of all the rooms at the Grosvenor Hotel, including the name of the guest staying in the room, if the room is occupied.

```
SELECT r.* FROM Room r LEFT JOIN
(SELECT g.guestName, h.hotelNo, b.roomNo
     FROM guest g, booking b, hotel h
WHERE g.guest_no = b.guest_no AND
b.hotel_no = h.hotel_no AND
h.name= 'Grosvenor Hotel' AND
date_from <= CURRENT_DATE AND
date_to >= CURRENT_DATE)
AS XXX ON r.hotel_no = XXX.hotel_no AND r.room_no = XXX.room_no;
```

(4) List all guests currently staying at the Grosvenor Hotel.

```
SELECT * FROM guest
WHERE guest_no =
(SELECT guest_no FROM booking
WHERE
date-from <= CURRENT_DATE AND date-to >= CURRENT_DATE AND
hotel_no =
(SELECT hotel_no FROM hotel
WHERE name = 'Grosvenor Hotel'));
```

(5) List the rooms that are currently unoccupied at the Grosvenor Hotel.

```
SELECT (r.hotel_no, r.room_no, r.type, r.price)
FROM room r, hotel h
WHERE r.hotel_no = h.hotel_no AND
h.name = 'Grosvenor Hotel' AND
NOT EXIST
(SELECT *
FROM booking b, hotel h
WHERE (date_from <= 'CURRENT_DATE'
AND date_to >= 'CURRENT_DATE')
AND r.hotel_no=b.hotel_no
AND r.room_no=b.room_no
```

```
AND r.hotel_no=h.hotel_no
AND name = 'Grosvenor Hotel');
```

(6) List the number of rooms in each hotel in London.

```
SELECT hotel_no, COUNT(room_no) AS count
FROM room r, hotel h
WHERE r.hotel_no = h.hotel_no AND city = 'London'
GROUP BY hotel_no;
```

(7) List out all guests who have booked room for three or more days.

```
SELECT guest-no, name
FROM guest g ,booking b
WHERE b.date-to Minus b.date-from >=3
```

Example 2.22.22 Consider following schema and write SQL for given statements

employee(employee-name,street,city)

works(employee-name, company-name, salary)

company(company-name, city)

manages(employee-name,manager-name)

(1) Find the names of all employees who work for first bank corporation.

(2) Give all employees of first bank corporation a 10-percent raise.

(3) Find the names and cities of residence of all employees who work for first bank corporation.

(4) Find the names and street addresses, cities of residence of all employees who work for First Bank Corporation and earn more than \$10,000

(5) Find all employees in the database who live in the same cities as the companies for which they work.

(6) Find all employees in the database who do not work for First Bank Corporation.

(7) Find the company and number of employees in company that has more than 30 employees

Solution :

(1) Find the names of all employees who work for First Bank Corporation.

```
SELECT employee_name
FROM works
WHERE company_name='First Bank Corporation';
```

(2) Give all employees of First Bank Corporation a 10-percent raise.

```
UPDATE works
SET salary=salary*1.1
WHERE company_name='First Bank Corporation';
```

(3) Find the names and cities of residence of all employees who work for First Bank Corporation.

```
SELECT employee_name,city
FROM employee
WHERE employee_name IN
(SELECT employee_name
FROM works
WHERE company_name='First Bank Corporation');
```

(4) Find the names and Street addresses, cities of residence of all employees who work for First Bank Corporation and earn more than \$10,000

```
SELECT employee_name,city
FROM employee
WHERE employee_name IN
(SELECT employee_name
FROM works
WHERE company_name='First Bank Corporation' AND salary>10000);
OR
SELECT E.employee_name, E.street, E.city
FROM employee as E, works as W
WHERE
E.employee_name=W.employee_name AND
W.company_name='First Bank Corporation' AND W.salary>10000
```

(5) Find all employees in the database who live in the same cities as the companies for which they work.

```
SELECT E.employee_name
FROM employee as E, works as W, company as C
where E.employee_name=W.employee_name AND
E.city=C.city AND
W.company_name=C.company_name;
```

(6) Find all employees in the database who do not work for First Bank Corporation.

```
SELECT employee_name
FROM works
WHERE company_name <>'First Bank Corporation';
```

(7) Find the company and number of employees in company that has more than 30 employees

```
SELECT company_name,COUNT(employee_name)
FROM employee E, company C,works W
WHERE E.employee_name=W.employee_name AND
W.company_name=C.company_name
HAVING COUNT(employee_name)>30;
```

Example 2.22.23 Consider following relations and write SQL queries for given statements

Assume suitable constraints

Instructor(ID, Name, Dept_name, Salary)

Teaches(ID, Course_id, Sec_id, Semester(even/odd), Year)

(1) Write SQL query to create Instructor table

(2) Find the average salary of the instructor in computer department.

(3) Find the number of instructors in each department who teach a course in even semester of 2016

(4) Find the names of instructor with salary amounts between 30000 and 50000

Solution : (1) Write SQL query to create Instructor table

```
CREATE TABLE Instructor
(ID CHAR,
Name VARCHAR(20),
Dept_name VARCHAR(15),
Salary numeric(8,2)
);
```

(2) Find the average salary of the instructor in computer department.

```
SELECT AVG(Salary)
FROM Instructor
WHERE Dept_name='Computer';
```

(3) Find the number of instructors in each department who teach a course in even semester of 2016

```
SELECT COUNT(DISTINCT ID)
FROM Teaches
WHERE Semester='even' AND Year=2016
```

(4) Find the names of instructor with salary amounts between 30000 and 50000

```
SELECT Name
FROM Instructor
WHERE Salary >=30000 AND Salary <=50000
```

Example 2.22.24 Consider following schema and write SQL for given statements

Client_master(clientno,name,address,city,pincode,state,baldue)

Product_master(productno,name,profitpercent,unitmeasure,sellprice,costprice)

Sales_master(Salesmanno,name,address,city,pincode,state,salary,tgtotget,remarks)

(1) Find out the names of all clients

(2) List all the clients who are located in Mumbai

(3) Delete all salesmen from salesman_master whose salaries are equal to ₹3500

(4) Destroy the table client_master along with data

- (5) List the names of all clients having 'a' as the second letter in their names.
 (6) Count the number of products having cost price is less than or equal to 500.
 (7) Calculate the average, minimum and maximum Sell price of product

Solution :

```
(1) SELECT name FROM Client_master;
(2) SELECT * FROM Client_master
    WHERE city='Mumbai'
(3) DELETE FROM Salesman_master
    WHERE salary=3500;

(4) DROP TABLE Client_master;
(5) SELECT name
FROM Client_master
WHERE name like ' _a%';
(6) SELECT count(productno)
FROM product_master
WHERE costprice <= 500
(7) SELECT AVG(sellprice), MIN(sellprice), MAX(sellprice)
FROM product_master;
```

Example 2.22.25 Assume the following table.

Degree (degcode, name, subject)

Candidate (seatno, degcode, name, semester, month, year, result)

Marks (seatno, degcode, semester, month, year, papcode, marks)

[degcode – degree code, name – name of the degree (Eg. MSc.), subject – subject of the course (Eg. Physis), papcode – paper code (Eg. A1)]

Solve the following queries using SQL;

Write a SELECT statement to display,

- (i) all the degree codes which are there in the candidate table but not present in degree table in the order of degcode.
 (ii) the name of all the candidates who have got less than 40 marks in exactly 2 subjects.
 (iii) the name, subject and number of candidates for all degrees in which there are less than 5 candidates.
 (iv) the names of all the candidate who have got highest total marks in MSc. Maths.

Solution : (i) SELECT C.degcode

```
FROM Candidate C,
WHERE NOT EXISTS
    (SELECT D.degcode
    FROM Degree D
    WHERE D.degcode=C.degcode)
ORDER by C.degcode
```


(ii) SELECT C.name

```
FROM Candidate C, Degree D, Marks M
WHERE
C.seatno=M.seatno AND C.degcode=D.degcode AND C.degcode=M.degcode AND M.marks < 40
GROUP BY C.seatno
HAVING count(D.subject)=2;
```

(iii) SELECT D.name,D.subject,count(*)

```
FROM degree D, Candidate C
WHERE D.degcode=C.degcode
HAVING( SELECT count(*) FROM Candidate <5);
```

(iv) SELECT C.name

```
FROM Candidate C, Degree D, Marks M
WHERE
D.degname='MSc' AND D.subject='Maths' AND C.degcode=D.degcode AND C.seatno=M.seatno
AND
M.marks = (SELECT max(M.marks) FROM Marks M)
```

Example 2.22.26 Consider a student registration database comprising of the below given table schema.

Student File

<i>Student Number</i>	<i>Student Name</i>	<i>Address</i>	<i>Telephone</i>
-----------------------	---------------------	----------------	------------------

Course File

<i>Course Number</i>	<i>Description</i>	<i>Hours</i>	<i>Professor Number</i>
----------------------	--------------------	--------------	-------------------------

Professor File

<i>Professor Number</i>	<i>Name</i>	<i>Office</i>
-------------------------	-------------	---------------

Registration File

<i>Student Number</i>	<i>Course Number</i>	<i>Date</i>
-----------------------	----------------------	-------------

Consider a suitable sample of tuples / records for the above mentioned tables and write DML statements (SQL) to answer for the queries listed below.

- i) Which courses does a specific professor teach ?
- ii) What courses are taught by two specific professors ?
- iii) Who teaches a specific course and where is his/her office ?
- iv) For a specific student number, in which courses is the student registered and what is his/her name ?
- v) Who are the professors for a specific student ?
- vi) Who are the students registered in a specific course ?

Solution :**(i)**

```
SELECT P.name,C.description
FROM Professor P, Course C
WHERE P.ProfessorNumber=C.ProfessorNumber
HAVING count(DISTINCT P.name)=2
```

(ii)

```
SELECT P.name,C.description
FROM Professor P, Course C
WHERE P.ProfessorNumber=C.ProfessorNumber
```

(iii)

```
SELECT P.name,P.office, C.description
FROM Professor P, Course C
WHERE P.ProfessorNumber=C.ProfessorNumber
```

(iv)

```
SELECT S.StudentNumber,S.StudentNumber,C.Description
FROM Student S, Course C, Registration R
WHERE S.StudentNumber=R.StudentNumber AND C.CourseNumber=R.CourseNumber
```

(v)

```
SELECT S.StudentName, P.Name
FROM Student S, Course C, Professor P, Registration R
WHERE C.ProfessorNumber=P.ProfessorNumber
      AND C.CourseNumber=R.CourseNumber
      AND S.StudentNumber=R.StudentNumber
GROUP BY P.ProfessorNumber
```

(vi)

```
SELECT S.StudentName, C.Description
FROM Student S, Course C, Registration R
WHERE S.StudentNumber=R.StudentNumber
      AND R.CourseNumber=C.CourseNumber
GROUP BY C.CourseNumber
```

Example 2.22.27 Consider following RESORT database

RESORT(resortno, resortname, resorttype, resortaddr,resortcity,numsuite)

SUITE (suiteno, resortno, suiteprice)

RESERVATION (reservationno, resorttype, resortaddr, resortcity , numsuite)

VISITOR(visitno, firstname, lastname, visitoraddr)

- i) Write the SQL to list full details of all the resorts on Los Angeles.
- ii) Write the SQL to list full details of all resorts having number of suites more than 30.
- iii) Write the SQL to list visitors in ascending order by firstname.

Solution : (i) SELECT * FROM RESORT WHERE resortcity ='Los Angeles';
 (ii) SELECT * FROM RESORT WHERE numsuite>30;
 (iii) SELECT * FROM VISITOR ORDER BY firstname;

Example 2.22.28 Consider the following database and answer the queries-

WORKS(Pname,Cname,Salary)

LIVES(Pname, Street, City)

LOCATED_IN(Cname, City)

Write the following queries in SQL :

- (i) List the names of the people who work for the company 'Wipro' along with cities they live in.
- (ii) Find the names of the persons who do not work for 'Infosys'
- (iii) Find the people whose salaries are more than that of all of the 'oracle' employees.
- (iv) Find the persons who works and lives in the same city.

Solution :

(i) SELECT L.Pname, L.City

FROM Works W, Lives L

Where W.Cname = "Wipro" and W.Pname = L.Pname;

(ii) SELECT Pname

FROM WORKS

WHERE Cname != 'Infoosy';

(iii) SELECT Pname

FROM WORKS

WHERE Salary > ALL

(SELECT Salary

FROM WORKS

WHERE Cname = "Oracle");

(iv) SELECT W.Pname

FROM WORKS W, LIVES L, LOCATED_IN I

WHERE W.Cname = I.Cname AND W.Pname = L.Pname

AND I.City = L.City

Example 2.22.29 Write the SQL queries for the following relational schema :

Sailors(Sid, Sname, Rating, Age)

Boats(Bid, Bname,color)

Reserve(Sid, Bid, Day)

- i) Retrieve the Sailor's name who have reserved red and green boat
- ii) Retrieve the no of boats which are not reserved.
- iii) Retrieve the Sailors name who have reserved boat number 103
- iv) Retrieve the Sailors name who have reserved all boats.

Solution : (i)

```
SELECT S.sname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid=R.sid
AND R.bid=B.bid
AND ( B.color='red' AND B.color='green')
```

(ii)

```
SELECT Sname
FROM Sailors
WHERE Sid NOT IN
(SELECT Sid
FROM Reserve, Sailors
WHERE Reserve.sid=Sailors.sid )
```

(iii)

```
SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid=R.sid AND bid=103
(iv) SELECT S.sname
FROM Sailors S
WHERE NOT EXISTS (( SELECT B.bid
FROM Boats B )
EXCEPT
( SELECT R.bid
FROM Reserves R
WHERE R.sid=S.sid ))
```

Example 2.22.30 Write SQL queries for the following relational schema

CUSTOMER(CID, CNAME,EMAIL,ADDR,PHONE)

ITEM(ITEM_NO,ITEM_NAME,PRICE, BRAND)

SALES(CID,ITEM_NO,#ITEMS, AMOUNT,SALE_DATE)

SUPPLIER(SID, SNAME, SPHONE, SADDR)

SUPPLY(SID, ITEM_NO, SUPPLY_DATE, QTY)

- (i) List the items purchased by Customer 'Prasanth'
- (ii) Retrieve items supplied by all suppliers starting from 1st Jan 2019 to 30th Jan 2019.

- (iii) Get the details of customers whose total purchase of items worth more than 5000 rupees.
 (iv) List total sales amount, total items, average sale amount of all items.
 (v) Display customers who have not purchased any items

Solution : (i)

```
SELECT ITEM_NAME
FROM ITEM I,CUSTOMER C and SALES S
WHERE C.CID= S.CID AND I.ITEM_NO = S.ITEM_NO AND C.CNAME='Prasanth'
```

(ii)

```
SELECT ITEM_NAME
FROM ITEM, SUPPLIER, SUPPLY
WHERE ITEM.ITEM_NO=SUPPLY.ITEM_NO AND
AND SUPPLY.SID=SUPPLIER.SID AND
SUPPLY.SUPPLY_DATE '1 JAN 2019' AND '30 JAN 2019'
```

(iii)

```
SELECT CID, CNAME, EMAIL,ADDR,PHONE
FROM CUSTOMER,SALES
WHERE CUSTOMER.CID=SALES.CID AND SALES.AMOUNT>=5000
```

(iv)

```
SELECT
SUM(AMOUNT),#item,AVG(AMOUNT)
FROM SALES
WHERE SALES.ITEM_NO = ITEM.ITEM_NO
```

(v)

```
SELECT CID, CNAME, EMAIL,ADDR,PHONE
FROM CUSTOMER
WHERE CID
NOT IN (
SELECT CID
FROM SALES
)
```

Example 2.22.31 Consider following database

Student(Roll_No, Name, Address)

Subject(Sub_code, Sub_name)

Marks(Roll_no, Sub_code, Marks)

Write following queries in SQL

1) Find average marks of each student along with the name of student

2) Find how many students have failed in the subject "DBMS"

(note:failed means obtained less than 40 marks)

Solution : (1)

```
SELECT Name, AVG(Marks)
FROM Student,Marks
WHERE Student.Roll_No=Marks.Roll_No
```

(2)

```
SELECT COUNT(Roll_NO)
FROM Subject,Marks
WHERE Marks.Marks<40 AND Subject.sub_code=Marks.Sub_code AND Sub_name="DBMS"
```

Example 2.22.32 Consider the following six relations for an order processing database application in a company.

customer (cust #, cname, city)

order (order #, odate, cust #, ord-Amt)

order-item (order #, item #, Qty)

item (item #, unit-price)

shipment (order #, warehouse #, ship-date)

Warehouse (warehouse #, city)

Here, *old-Amt* refers to total amount of an order; *odate* is the date the order was placed; *ship-date* is the date an order is shipped from the warehouse. Assume that an order can be shipped from several warehouses.

Specify the following in SQL queries

- i) List the order # and ship-date for all orders shipped from warehouse number 'W2'.
- ii) List the warehouse information from which the customer named 'Jose Lopez' was supplied his orders.
- iii) List the orders that were not shipped within 30 days of ordering.
- iv) List the order # for orders that were shipped from all warehouses that the company has in New York.

Solution:**i)**

```
SELECT Order#, ship-date
FROM shipment
WHERE warehouse# = 'W2'
```

ii)

```
SELECT order.order#, shipment.warehouse#
FROM customer, shipment, order
WHERE order.order# = shipment.order#
      AND customer.cust# = order.cust#
      AND customer.cname = 'Jose Lopez'
```

iii)

```
SELECT order#, odate, cust#, ord-Amt
FROM order, shipment,
WHERE order.order#=shipment.order# AND shipment.ship-date>30
```

iv)

```
SELECT shipment.order#
FROM shipment, Warehouse
WHERE shipment.warehouse# = Warehouse.warehouse#
AND Warehouse.city = "New York"
```

Example 2.22.33 Consider the following database schema :

Physician (reg_no, name, tel_no, city)

Patient(p_name, street, city)

Visit (p_name, reg_no, date_of_visit, fee)

Write SQL queries for following requirements (any two)

i) Find the name and city of patients who visited a physician on 13 July 2017.

ii) Get the name of the physician and the total no. of patients visited him.

iii) Get the details of date wise fees collected at clinic.

SPPU : Aug 17, End Sem, Marks 5

Solution :

```
(i) SELECT p_name, city
FROM Patient, Visit
WHERE Patient.p_name=Visit.p_name AND date_of_visit='13-July-2017'
(ii) SELECT name, count(*)
FROM Physician, Visit
WHERE Physician.reg_no = Visit.reg_no
GROUP BY Physician.reg_no
(iii) SELECT *
FROM Visit
GROUP BY date_of_visit
```

Example 2.22.34 Consider insurance database with following schema :

person (driver-id, name, address)

car (license, model, year)

accident (report-no, date, location)

owns (driver-id, license)

participated (driver-id, car, report-no, damage-amount)

Write a query in SQL for following requirements (any two) :

- i) Find the total no. of people who owned cars that were involved in accidents in 2016.
- ii) Retrieve the name of person whose address contains Pune.
- iii) Find the name of persons having more than two cars

SPPU : Dec 17, End Sem, Marks 5

Solution :

- i)

```
SELECT count(DISTINCT name)
FROM person, accident, participated
WHERE accident.report-no = participated.report-no AND participated.driver-id =
person.driver-id AND date BETWEEN DATE '2016-00-00' AND DATE '2016-12-31'
```
- ii)

```
SELECT name
FROM person
WHERE address LIKE %Pune'
```
- iii)

```
SELECT name
FROM person
WHERE 2 <
(SELECT count(*)
FROM person,car,owns
WHERE car.license = owns.license AND owns.driver-id=person.driver-id
```

Example 2.22.35 Schema definition for supplier-and-parts database.

Supplier = (supplier_number, supplier_name, status, city)

Parts = (part_number, part_name, colour, weight, city)

Shipments = (supplier_number, part_number, quantity)

Write SQL query for following requirements (any 2).

- i) Find shipment information (supplier_number, supplier_name, part_number, part_name, quantity) for those having quantity less than 150.
- ii) List supplier_number, supplier_name, part_number, part_name for those suppliers who made shipment of parts whose quantity is larger than the average quantity.
- iii) Find aggregate quantity of part_number 'A692' of colour 'GREEN' for which shipment made by supplier_number who reside in 'MUMBAI'

SPPU : May 18, End Sem, Marks 5

Solution :

- i)

```
SELECT supplier_number,supplier_name,part_number,part_name,quantity
FROM Supplier, Parts, Shipments
WHERE Supplier.supplier_number = Shipments.supplier_number AND Parts.part_number
=Shipments.part_number
AND Shipments.quantity<150
```
- ii)

```
SELECT supplier_number, supplier_name, part_number, part_name
FROM Supplier, Parts
WHERE Supplier.supplier_number = Shipments.supplier_number AND Part.part_number=
```



```

Shipments.part_number AND
Shipments.quantity > (SELECT AVG(quantity) FROM Shipments)
iii) SELECT COUNT(part_number)
FROM Parts, Supplier, Shipments
WHERE Supplier.supplier_number = Shipments.supplier_number AND Part.part_number =
Shipments.part_number AND
Parts.part_number = 'A692' AND Parts.colour = 'GREEN' AND Supplier.city = 'MUMBAI'

```

Example 2.22.36 Consider the following database schema :

Loan (loan_no, Branch_name, Amount)

Borrower (Cust_name, Loan_no).

Write SQL queries for following requirements (any two) :

- i) Find all customers who have a loan from bank. Find their names, loan nos. and Loan amount.
- ii) Find names of customers in alphabetical order who have a loan at Pune branch.
- iii) Find all loan nos. for loan made at Pune branch with loan amount greater than 20000.

SPPU : Oct 18, End Sem, Marks 5

Solution :

```

(i) SELECT Cust_name Loan_no, Amount
FROM Loan, Borrower
WHERE Loan.Loan_no = Borrower.Loan_no
(ii) SELECT Cust_name
FROM Borrower, Loan
WHERE Loan.Branch_name = 'Pune'
ORDER BY Cust_name ASC
(iii) SELECT Loan_no
FROM Loan, Borrower
WHERE Loan.Loan_no = Borrower.Loan_no AND Loan.Branch_name = 'Pune' AND
Loan.Amount > 20000

```

Example 2.22.37 Consider Employee database with following schema :

Employee (Emp_Id, First_Name, Last_Name, Salary, Joining_Date, Department)

Bonus (Emp_Ref_Id, Bonus_Amount, Bonus_Date)

Designation (Emp_Ref_Id, Emp_Designation, Affected_From)

Write queries in SQL for following requirements (any 2):

- i) To fetch the departments that have less than five people in it.
- ii) To print the name of employees having the highest salary in each department.
- iii) Write an SQL query to print details of the employee who are also Managers.

SPPU : Dec 18, End Sem, Marks 5

Solution :

- ```
(i) SELECT Department, COUNT(Emp_Id) as 'Number of People'
 FROM Employee GROUP BY Department
 HAVING COUNT(Emp_Id) <5
(ii) SELECT E.Department,t.First_Name,t.Salary
 FROM
 (SELECT MAX(Salary) AS TotalSalary, Department FROM Employee GROUP BY Department)
 AS temp
 INNER JOIN ON temp.Department=E.Department AND temp.TotalSalary=E.Salary;
(iii) SELECT DISTINCT E.FIRST_NAME, D.Emp_Designation
 FROM Employee E, Designation D
 WHERE E.Emp_Id = D.Emp_Ref_Id
 AND D.Emp_Designation IN ('Manager');
```

**Example 2.22.38** Consider following schema :

*account (acct-no, branch - name, balance)*

*Depositor (cust - name, acct - no)*

*borrower (cust-name, loan-no)*

*loan (loan - no, branch - name, amount)*

Write following queries using SQL (any 2)

i) Find names of all customers who have a loan at the redwood branch.

ii) Find all customers who are having an account and loan or both.

iii) Find average account balance at each branch.

**SPPU : May 19, End Sem, Marks 5**

**Solution :**

- ```
(i) SELECT DISTINCT cust-name
    FROM borrower, loan
    WHERE borrower.loan-no = loan.loan-no AND loan.branch-name = 'redwood'
(ii) SELECT cust-name FROM Depositor
    UNION
    SELECT cust-name FROM borrower
(iii) SELECT branch-name, AVG(balance)
    FROM account
    GROUP BY branch-name
```

Example 2.22.39 Consider the following database schema :

Emp(E_number, E_name, Dept_no)

Dept(Dept_no, Dept_name)

Address(Dept_name, Dept_location)

Write SQL queries for following requirements (any 2)

- i) Display the name of department for the employee having E_number 'E1011'.
- ii) Display the location of department where employee 'Ramesh' is working.
- iii) Display total no. of employees working in each department.

SPPU : Oct 19, In Sem, Marks 5

Solution :

- i)

```
SELECT Dept-name
FROM Dept, Emp
WHERE Dept.Dept_no = Emp.Dept_no AND E_number = 'E1011'
```
- ii)

```
SELECT Dept_location
FROM Address A, Dept D, Emp E
WHERE E.E_name = 'Ramesh' AND E.Dept_no = D.Dept_no AND
Dept.Dept_name = A.Dept_name
```
- iii)

```
SELECT COUNT(Emp_no)
FROM Emp E, Dept D
WHERE E.Dept_no = D.Dept_no
```

Example 2.22.40 Emp(E_number, E_name, Dept_no)

Dept(Dept_no, Dept_name).

Consider the schema given above. Consider above tables are created without considering the Dept_no as primary key in Dept table and foreign key in Emp table. Assuming tables are already created write SQL queries for following requirements.

- i) Create primary key in dept table considering above situations.
- ii) Create foreign key considering EMP as child table and dept as master table also consider the above situation.
- iii) Add column salary with appropriate data type in EMP table.

SPPU : Oct 19, In Sem, Marks 5

Solution :

- i)

```
ALTER TABLE Dept
ADD PRIMARY KEY(Dept_no)
```
- ii)

```
ALTER TABLE Emp
ADD CONSTRAINT FK_deptno
FOREIGN KEY (Dept_no) REFERENCES Dept(Dept_no);
```
- iii)

```
ALTER TABLE Emp
ADD Salary int
```



UNIT - II

3

PL / SQL

Syllabus

Concept of Stored Procedures and Functions, Cursors, Triggers, Assertions, Roles and Privileges.

Contents

3.1	Basics of PL/SQL	
3.2	Writing First PL/SQL Script	
3.3	Block Structure of PL/SQL	
3.4	PL/SQL Data Types	
3.5	PL/SQL Variables	
3.6	PL/SQL Constants	
3.7	Control Statements	
3.8	Handling Database Tables using PL/SQL	
3.9	Examples on PL/SQL.....	Aug.-17, Oct.-18,19, May-19, Dec.-19, Marks 5
3.10	Concept of Stored Procedures	
3.11	Functions Aug.-17, Marks 5
3.12	Cursors Dec.-17, Marks 5
3.13	Triggers Dec.-17, 18, 19, Marks 5
3.14	Assertions	
3.15	Roles and Privileges	
3.16	Exceptions Dec.-18, May-19, Marks 5
	Multiple Choice Questions	

3.1 Basics of PL/SQL

- PL/SQL stands for Procedural Language extensions to the Structured Query Language (SQL).
- PL/SQL is a combination of SQL along with the procedural features of programming languages.
- **Oracle** uses a PL/SQL engine to process the PL/SQL statements.
- PL/SQL includes procedural language elements like conditions and loops. It allows declaration of constants and variables, procedures and functions, types and variables of those types and triggers.

3.1.1 How to Set Environment for Executing PL/SQL Scripts ?

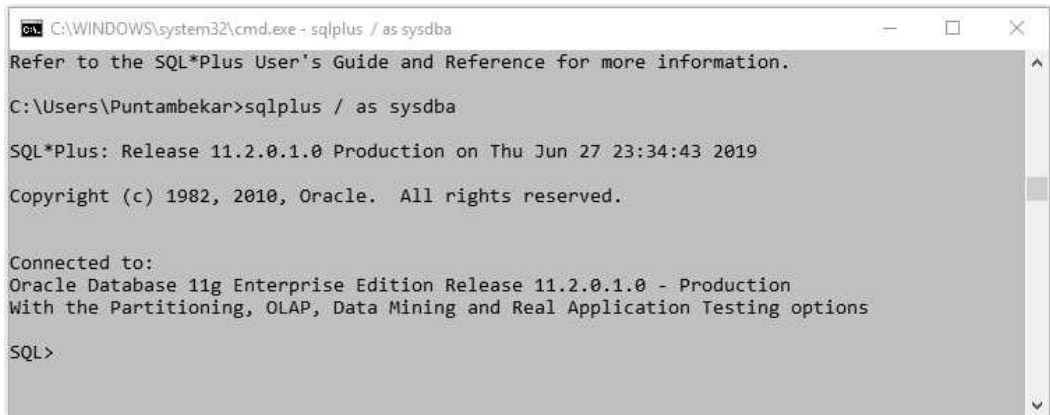
We need to install **Oracle** for executing the PL/SQL script. We can install it on Linux as well as on Windows (32 bit or 64 bit) platform.

For that purpose go to the web site <https://www.oracle.com>.

On completion of installation, Open command prompt window and issue the following commands at the prompt

```
Set ORACLE-SID=orcl
sqlplus /as sysdba
```

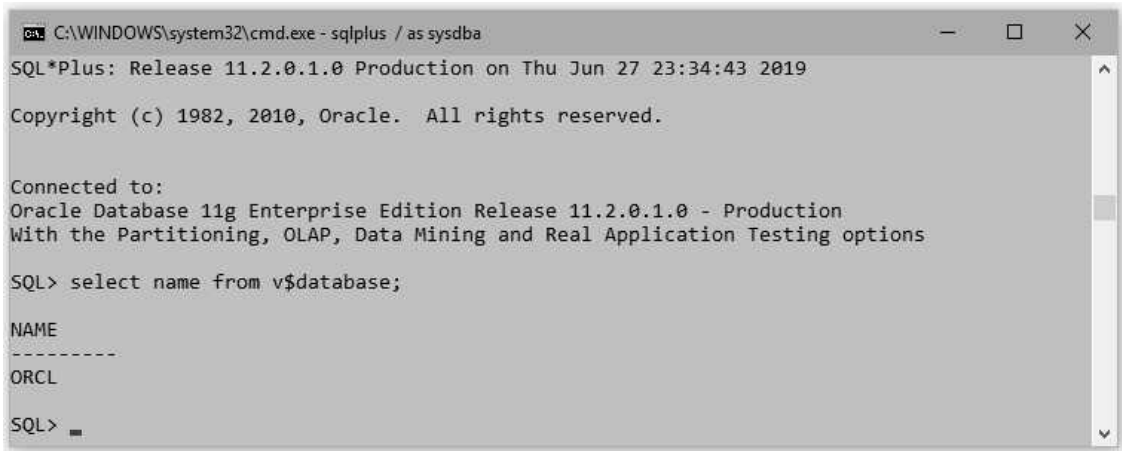
The command prompt window will display SQL prompt as follows -



```
C:\WINDOWS\system32\cmd.exe - sqlplus / as sysdba
Refer to the SQL*Plus User's Guide and Reference for more information.
C:\Users\Puntambekar>sqlplus / as sysdba
SQL*Plus: Release 11.2.0.1.0 Production on Thu Jun 27 23:34:43 2019
Copyright (c) 1982, 2010, Oracle. All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
SQL>
```

You can test the Oracle installation by typing the SELECT query as follows.



```
C:\WINDOWS\system32\cmd.exe - sqlplus / as sysdba
SQL*Plus: Release 11.2.0.1.0 Production on Thu Jun 27 23:34:43 2019
Copyright (c) 1982, 2010, Oracle. All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> select name from v$database;

NAME
-----
ORCL

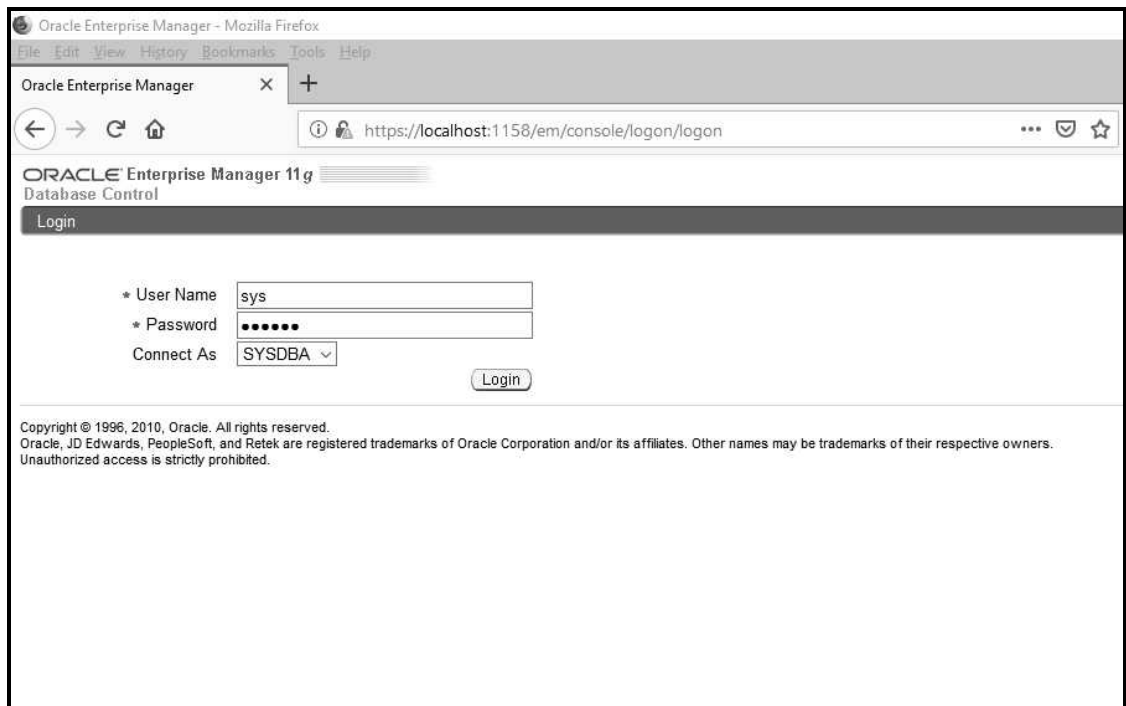
SQL>
```

Then type **quit** and then exit at the SQL prompt to close this command prompt window. We can also check the installation is correct or not at GUI level as follows -

Just open the Web browser and type the URL as,

<https://localhost:1158/em>

Following type of web page appears in your browser.

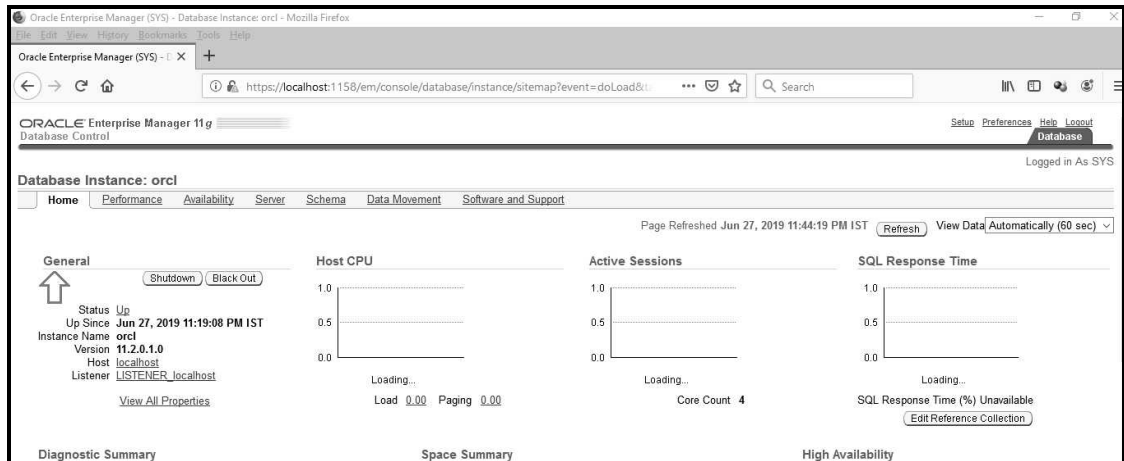


Username as **sys**

Password as **oracle**(you can choose any password of your choice)

Connect as SYSDBA

Click Login button and then the Web page appears as follows -



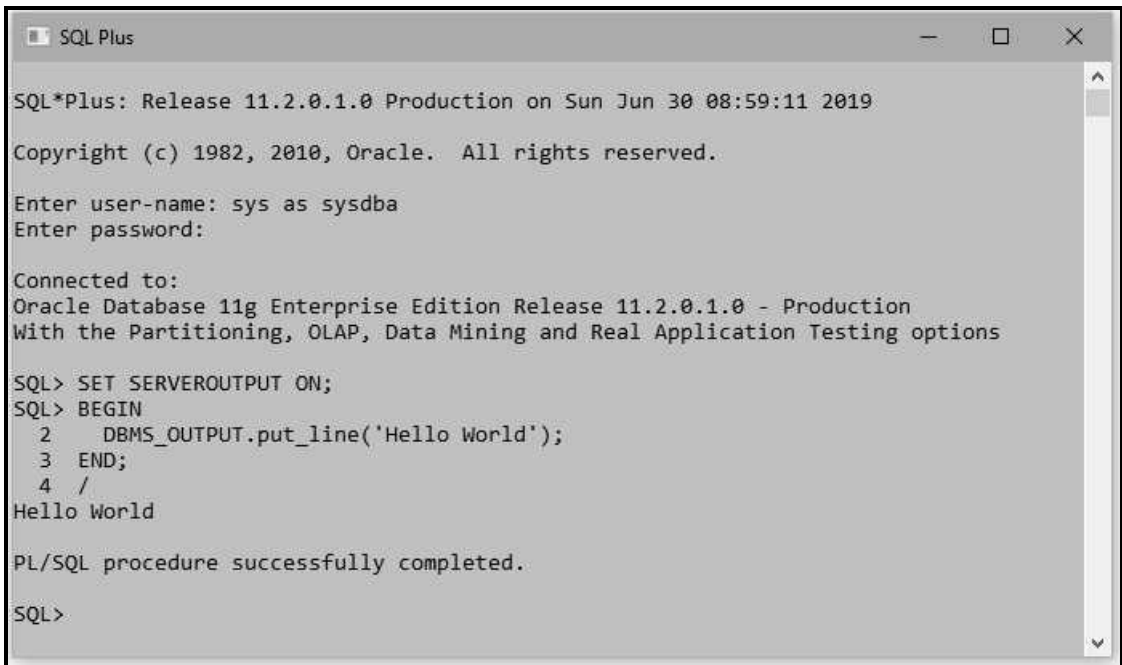
That it!! Your Oracle is installed on your machine.

Troubleshooting : Sometimes at the start of installation itself, it gives error about CREATE file on Command prompt, to resolve this error, just disable your Antivirus Software during the installation process.

3.2 Writing First PL/SQL Script

(1) Writing and Executing the PL/SQL script on SQL PLUS

- SQL Plus, which is a command-line interface for executing SQL statement and PL/SQL blocks provided by Oracle Database.
- Open SQL PLUS command prompt.
- Type user name and password. I am using the user name : sys as sysdba and password as oracle(This is the password which I set during installation.).
- Then the SQL prompt appears. Now we can execute our PL/SQL code from here.



```
SQL*Plus: Release 11.2.0.1.0 Production on Sun Jun 30 08:59:11 2019
Copyright (c) 1982, 2010, Oracle. All rights reserved.

Enter user-name: sys as sysdba
Enter password:

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> SET SERVEROUTPUT ON;
SQL> BEGIN
  2  DBMS_OUTPUT.put_line('Hello World');
  3  END;
  4  /
Hello World

PL/SQL procedure successfully completed.

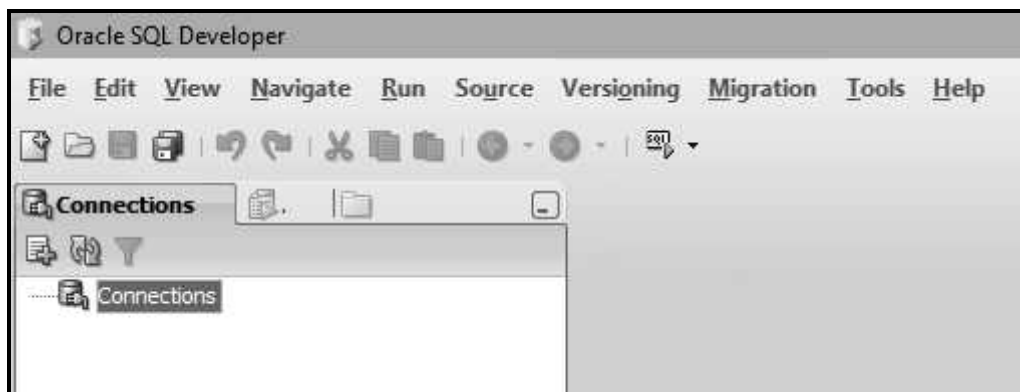
SQL>
```

(2) Writing the PL/SQL script on Oracle SQL Developer

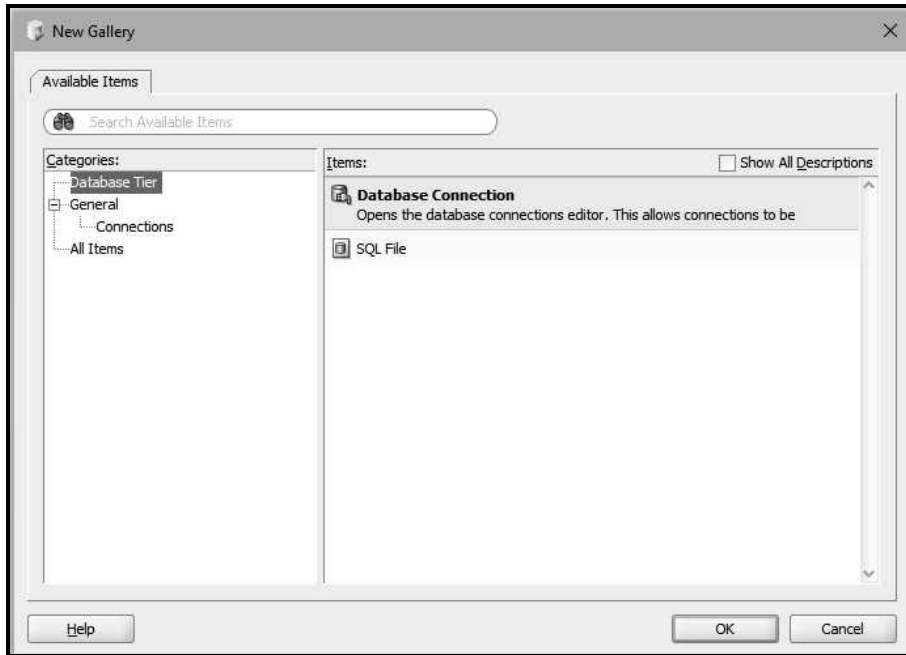
You can write PL/SQL script at the SQL Prompt or you can store the script in separate file and execute it.

For storing the script in separate file we need to use **Oracle SQL Developer**.

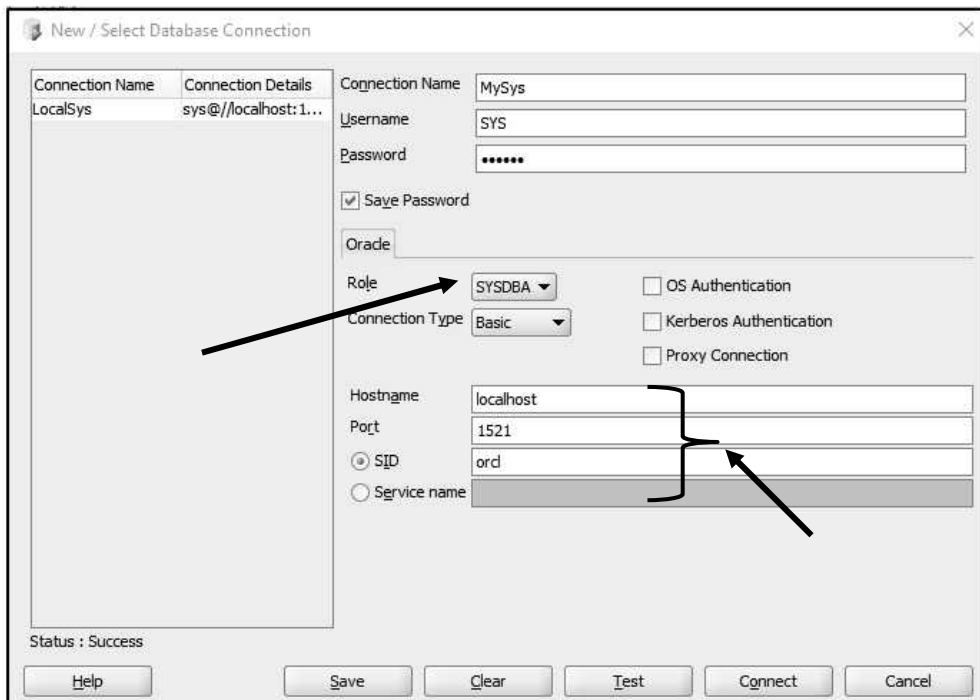
After installation of Oracle Package, this developer tool comes integrated with it. Click on it.



Click on File -> New, Following window will appear

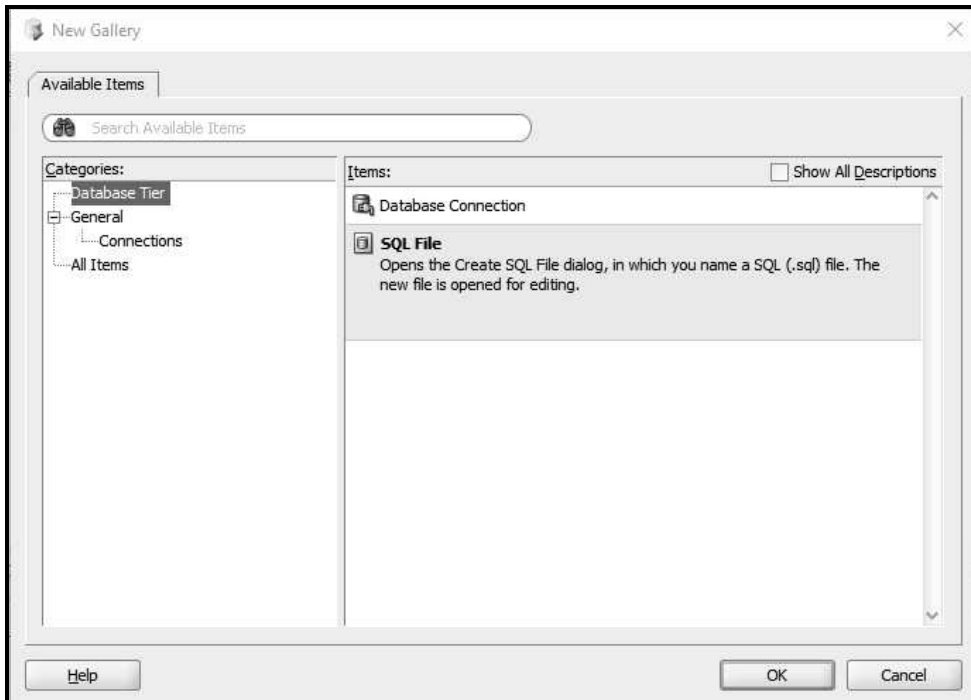


Now Enter user name and password as follows. You can give Connection Name of your choice. Then click Test button. On successful connection we get Success message. Then Click Connect button.

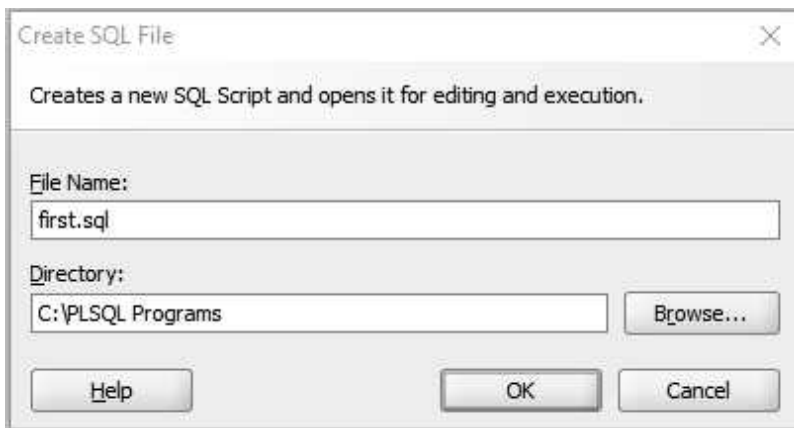


Now Click on **File->New**.

Following window appears. Click on **SQL File** and then Click **OK** button



Set the filename of your choice. For instance - I have created a PL/SQL file as follows -



The script is as follows -

```
SET SERVEROUTPUT ON;
BEGIN
    DBMS_OUTPUT.put_line('Hello World');
END;
/
```

Explanation of Script

(1) The first line of PL/SQL script is

```
SET SERVEROUTPUT ON;
```

This allows the user to see the output, when the script is executed on specified connection.

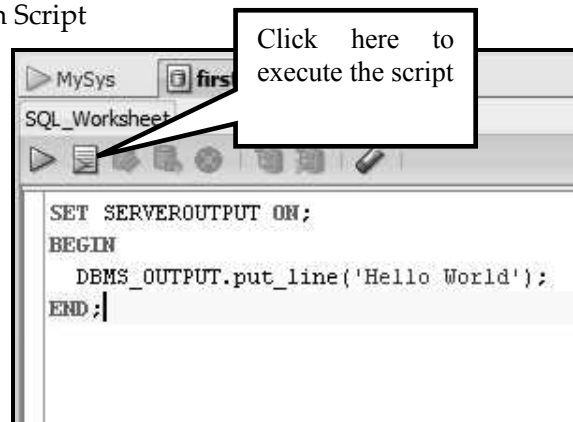
(2) Then BEGIN statement indicated the beginning of the execution block.

(3) The execution block is within the BEGIN and END statements. You can write any executable PL/SQL statements here.

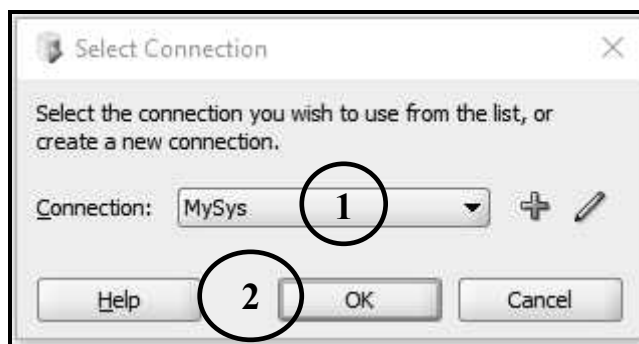
(4) For displaying the desired message we use DBMS_OUTPUT.PUT_LINE. The DBMS_OUTPUT is a built-in package that allows us to display output.

Execution of Script

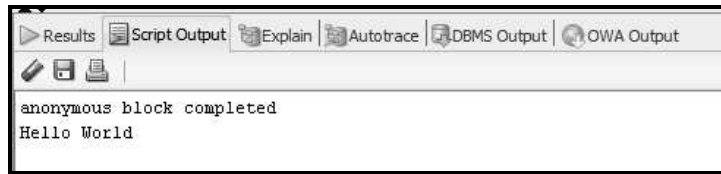
Click on Execution Script



The window for Select Connection will appear, Select the Connection and click OK button.



The output will be displayed at the bottom of the Editor window as follows



3.3 Block Structure of PL/SQL

- PL/SQL is a procedural language.
- The code is written in small blocks in PL/SQL.
- Broadly there are two types of blocks -

(1) Anonymous Block

- A block without a name is an anonymous block.
- An anonymous block is not saved in the Oracle Database server, so it is just for one-time use.
- PL/SQL anonymous blocks are useful for testing purposes.

(2) Named Block

- A PL/SQL block has a name.
- A **Function** or a **Procedure** is an example of a named block.
- A named block is saved into the Oracle Database server first and then can be reused.
- A PL/SQL block consists of three sections : declaration, executable, and exception-handling sections. In a block, the executable section is mandatory while the declaration and exception-handling sections are optional. This structure of PL/SQL block is as follows

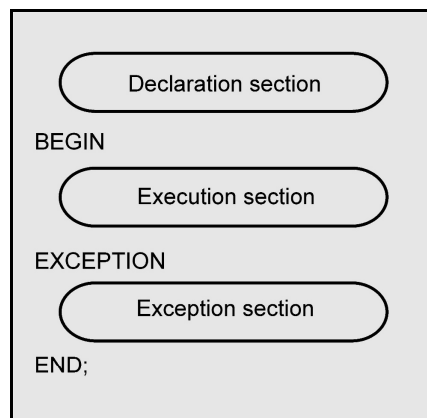


Fig. 3.3.1 Block structure of PL/SQL

1) Declaration section

A PL/SQL block has a declaration section where you declare variables, allocate memory for cursors and define data types.

2) Executable section

A PL/SQL block has an executable section. An executable section starts with the keyword BEGIN and ends with the keyword END. The executable section must have a least one executable statement, even if it is the NULL statement which does nothing.

3) Exception-handling section

A PL/SQL block has an exception-handling section that starts with the keyword EXCEPTION. The exception-handling section is where you catch and handle exceptions raised by the code in the execution section.

3.4 PL/SQL Data Types

Following are the types of PL/SQL data types

1. **Scalar** : These data types don't include any internal components. It includes data types such as NUMBER, DATE, BOOLEAN, etc.
2. **Large Objects (LOB)** : This type of data type stores objects that are relatively large in size and stored separately from other data types such as text, graphic images, video clips, sound, etc.
3. **Composite** : These type of data types have internal components that can be accessed individually. It includes records and collections.
4. **Reference** : As the name sounds, it includes pointers that refer to the location of the other data items.

Following are some commonly used data types in PL/SQL

Data Type	Description
Numeric	Numeric values on which arithmetic operations are performed. It includes sub types such as number, decimal, real, float, etc.
Character	Character values on which character operations such as strings are performed. It includes sub types such as char, varchar, varchar2, nvarchar2, etc.
Date and Time	This data type is for displaying the date and time values. The default date format is DD-MM-YY.
Boolean	These include logical values on which logical operations are performed. The logical values are the Boolean values TRUE and FALSE and the value NULL.

Number	<p>Number(Precision, Scale). Precision is the total number of digits and scale is the number of digits to the right of the decimal point. You cannot use constants or variables to specify precision and scale; you must use integer literals.</p> <p>To declare floating-point numbers, for which you cannot specify precision or scale because the decimal point can float to any position, use the following form without precision and scale :</p> <p>NUMBER</p> <p>The maximum precision that can be specified for a NUMBER value is 38 decimal digits. If you do not specify precision, it defaults to 39 or 40, or the maximum supported by your system, whichever is less.</p> <p>Scale, which can range from -84 to 127, determines where rounding occurs.</p>
Float	ANSI and IBM specific floating-point type with maximum precision of 126 binary digits (approximately 38 decimal digits).
Integer	ANSI and IBM specific integer type with maximum precision of 38 decimal digits.
Real	Floating-point type with maximum precision of 63 binary digits (approximately 18 decimal digits).
Varchar2	Variable-length character string with maximum size of 32,767 bytes.
Rowid	Physical row identifier, the address of a row in an ordinary table.
Blob	This data types is used to store large binary objects in the database. Memory Capacity : 8 to 128 TB.
Clob	This data type is used to store large blocks of character data in the database. Memory Capacity : 8 to 128 TB.

3.5 PL/SQL Variables

- The variables in PL/SQL are declared in declaration section. The syntax of variable declaration is as follows -

Syntax

Variable-Name DataType(Precision/Dimension)

Example

```
Person_name varchar2(30);
```

Here Person_name is a variable name and varchar2 is the data type with 30 as dimension.

- Constraints are associated with the variables defined in the code block. A constraint is a condition that is placed on the variable. Two commonly used constraints are –
 - **Constant** - This constraint will ensure that the value is not changed after a value is initially assigned to a variable. If a statement tries to change the variable value, an error will be displayed.
 - **Not Null** - This constraint will ensure that the variable always contains a value. If the statement attempts to assign an empty or a null value to that particular variable, the program will be error prone and will get abnormal termination of the program or the exception section will execute, if included in the program code.
 - **For Example -**

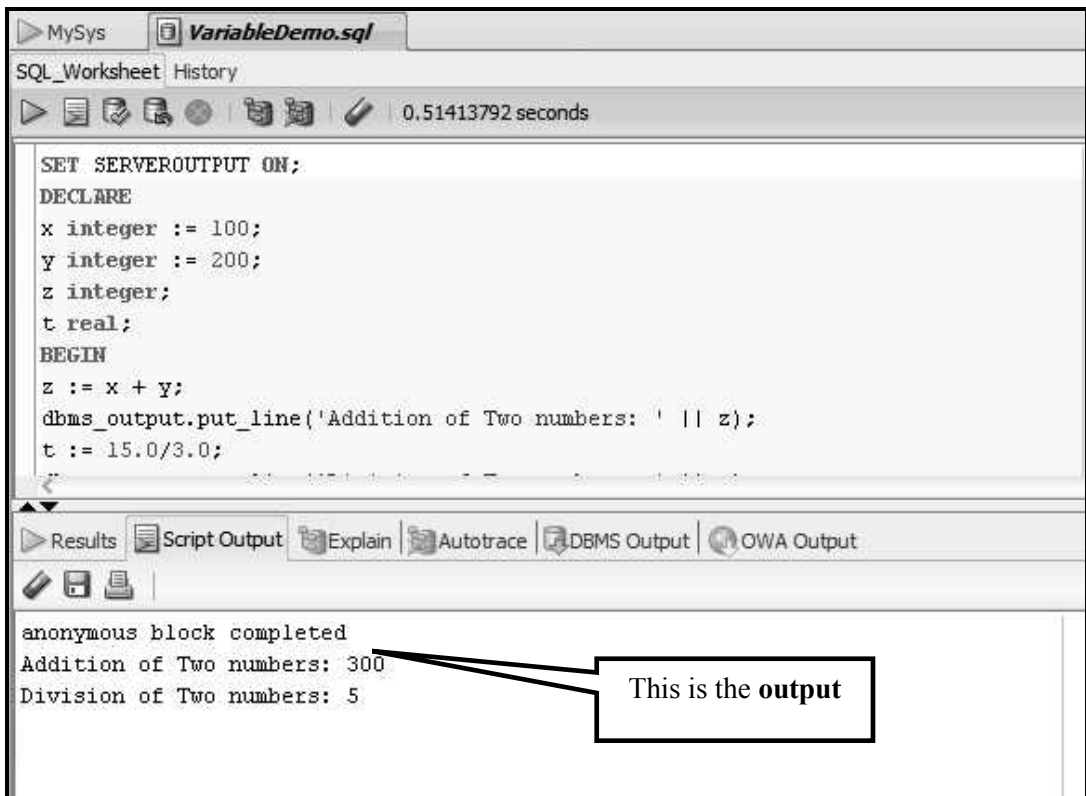
```
PI constant number(5,2)=3.1415
```

- **Variable Initialization** : Variable can be initialized using :=. This sign includes a 'colon' with a succeeding 'equal to' sign. This particular sign assigns the parameter on the right hand side of the sign to the parameter or the variable on the left hand side of the sign.

Programming Example

```
SET SERVEROUTPUT ON;
DECLARE
x integer := 100;
y integer := 200;
z integer;
t real;
BEGIN
z := x + y;
dbms_output.put_line('Addition of Two numbers: ' || z);
t := 15.0/3.0;
dbms_output.put_line('Division of Two numbers: ' || t);
END;
/
```


Following Screenshot demonstrates above program



The screenshot shows a SQL IDE window titled 'MySys' with a file named 'VariableDemo.sql'. The main editor contains the following PL/SQL code:

```
SET SERVEROUTPUT ON;
DECLARE
x integer := 100;
y integer := 200;
z integer;
t real;
BEGIN
z := x + y;
dbms_output.put_line('Addition of Two numbers: ' || z);
t := 15.0/3.0;
```

Below the editor, the 'Results' tab is active, displaying the output of the program:

```
anonymous block completed
Addition of Two numbers: 300
Division of Two numbers: 5
```

A callout box with the text 'This is the output' points to the output text in the Results tab.

There are two types of variable scope :

1. Local Variables
2. Global Variables

For example

```
DECLARE
-- Global Variables
a integer := 100;
b integer := 200;
dbms_output.put_line('a: ' || a);
dbms_output.put_line('b: ' || b);
BEGIN
-- Local Variables
c integer := 300;
d integer := 400;
dbms_output.put_line('c: ' || c);
dbms_output.put_line('d: ' || d);
END;
```

3.6 PL/SQL Constants

Constant is a value that remains unchanged. It is declared using the keyword CONSTANT. It requires initial value that does not get changed.

Syntax

```
constant_name CONSTANT datatype := VALUE;
```

Example

```
PI CONSTANT NUMBER :=3.1415;
```

3.7 Control Statements

The control states determine the flow of execution of PL/SQL block. The control statements are of two types – conditional statements and loop statements.

3.7.1 IF Statement

The If-then-else type of statement is used to specify the condition in PL/SQL block. There are various ways by which the if statement can be written. Their syntaxes are as given below

(1) Syntax- IF-THEN

```
IF condition  
THEN  
Statement: {when condition is true, this statement executes}  
END IF;
```

(2) Syntax- IF-THEN-ELSE

```
IF condition  
THEN  
    {statements to execute when condition is true}  
ELSE  
    {statements to execute when condition is False}  
END IF;
```

(3) Syntax- If-THEN-ELSIF

```
IF condition#1  
THEN  
    {statements to execute when condition#1 is True}  
ELSIF condition#2  
THEN  
    {statements to execute when condition#2 is True}  
END IF;
```

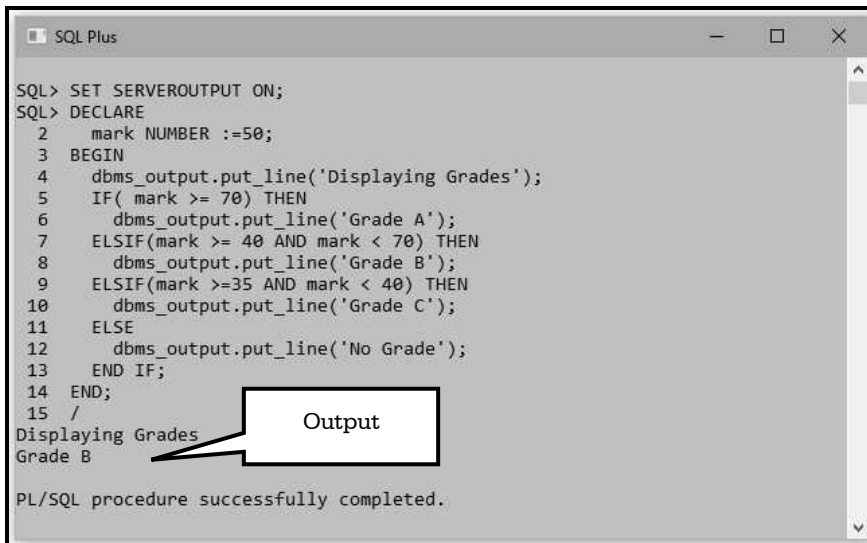
(4) Syntax- If-THEN-ELSIF -ELSE

```
IF condition#1
THEN
  {statements to execute when condition#1 is True}
ELSIF condition#2
THEN
  {statements to execute when condition#2 is True}
ELSE
  {statements to execute when condition#1 and Condition#1 is False}
END IF;
```

Programming Example

```
SET SERVEROUTPUT ON;
DECLARE
  mark NUMBER :=50;
BEGIN
  dbms_output.put_line('Displaying Grades');
  IF( mark >= 70) THEN
    dbms_output.put_line('Grade A');
  ELSIF(mark >= 40 AND mark < 70) THEN
    dbms_output.put_line('Grade B');
  ELSIF(mark >=35 AND mark < 40) THEN
    dbms_output.put_line('Grade C');
  ELSE
    dbms_output.put_line('No Grade');
  END IF;
END;
/
```

Output(Using SQL PLUS)



```
SQL Plus
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2  mark NUMBER :=50;
  3  BEGIN
  4  dbms_output.put_line('Displaying Grades');
  5  IF( mark >= 70) THEN
  6  dbms_output.put_line('Grade A');
  7  ELSIF(mark >= 40 AND mark < 70) THEN
  8  dbms_output.put_line('Grade B');
  9  ELSIF(mark >=35 AND mark < 40) THEN
 10  dbms_output.put_line('Grade C');
 11  ELSE
 12  dbms_output.put_line('No Grade');
 13  END IF;
 14  END;
 15  /
Displaying Grades
Grade B
PL/SQL procedure successfully completed.
```

Output

Output(Using Oracle SQL Developer)

```
anonymous block completed
Displaying Grades
Grade B
```

3.7.2 General Loop

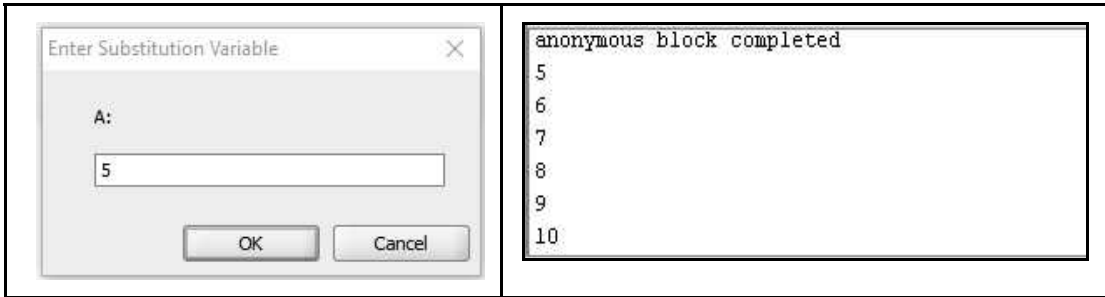
- A General Loop in PL/SQL is used to execute a set of statements at least once before the termination of the loop.
- An EXIT condition has to be specified in the loop; otherwise the looping process will get into a never ending loop, also known as an Infinite Loop
- On encountering with EXIT statement, the loop exits.
- In a PL/SQL Loop, the statements are enclosed between the keywords LOOP and END LOOP.
- In every Loop, the statements are executed and then control restarts from the top of the loop until a certain condition is satisfied.

Syntax

```
LOOP
  Statement1;
  Statement2;
  EXIT;
END LOOP;
```

Programming Example

```
SET SERVEROUTPUT ON;
DECLARE
  a integer;
BEGIN
  a:=&a;
  LOOP
    dbms_output.put_line(a);
    a:=a+1;
  EXIT WHEN a>10;
  END LOOP;
END;
/
```

Output (Using ORACLE SQL DEVELOPER)**Output(Using SQL PLUS)**

```
SQL Plus
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2   a integer;
  3   BEGIN
  4   a:=&a;
  5   LOOP
  6     dbms_output.put_line(a);
  7     a:=a+1;
  8   EXIT WHEN a>10;
  9   END LOOP;
 10  END;
 11  /
Enter value for a: 5
old 4: a:=&a;
new 4: a:=5;
5
6
7
8
9
10

PL/SQL procedure successfully completed.
SQL>
```

3.7.3 For Loop

- For loop executes sequence of statements for specific number of times
- There are starting and ending values in between which the statement executes.
- The counter is incremented by one each time.

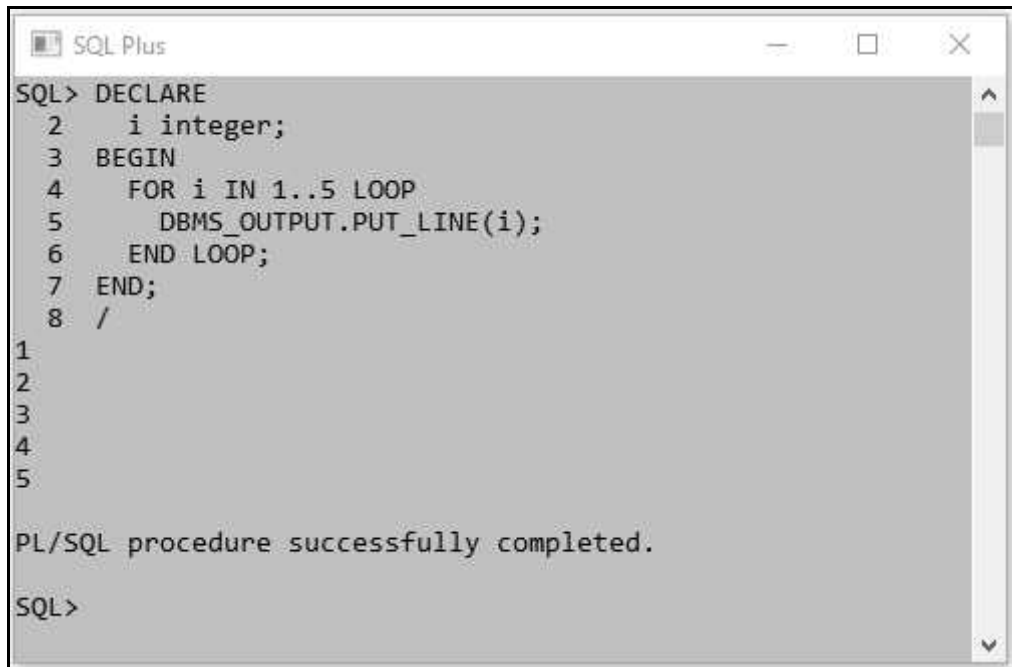
Syntax

```
FOR counter IN num1....num2 LOOP
statement1;
statement2;
END LOOP;
```

Programming Example

```
DECLARE
  i integer;
BEGIN
  FOR i IN 1..5 LOOP
    DBMS_OUTPUT.PUT_LINE(i);
  END LOOP;
END;
/
```

Output



```
SQL Plus
SQL> DECLARE
  2  i integer;
  3  BEGIN
  4  FOR i IN 1..5 LOOP
  5  DBMS_OUTPUT.PUT_LINE(i);
  6  END LOOP;
  7  END;
  8  /
1
2
3
4
5
PL/SQL procedure successfully completed.
SQL>
```

3.7.4 While Loop

- This is a kind of loop that repeats the sequence of statements while given condition is true.
- Before entering the loop the condition is tested and if it is true then only the control executes the loop body. It executes until the condition becomes false.

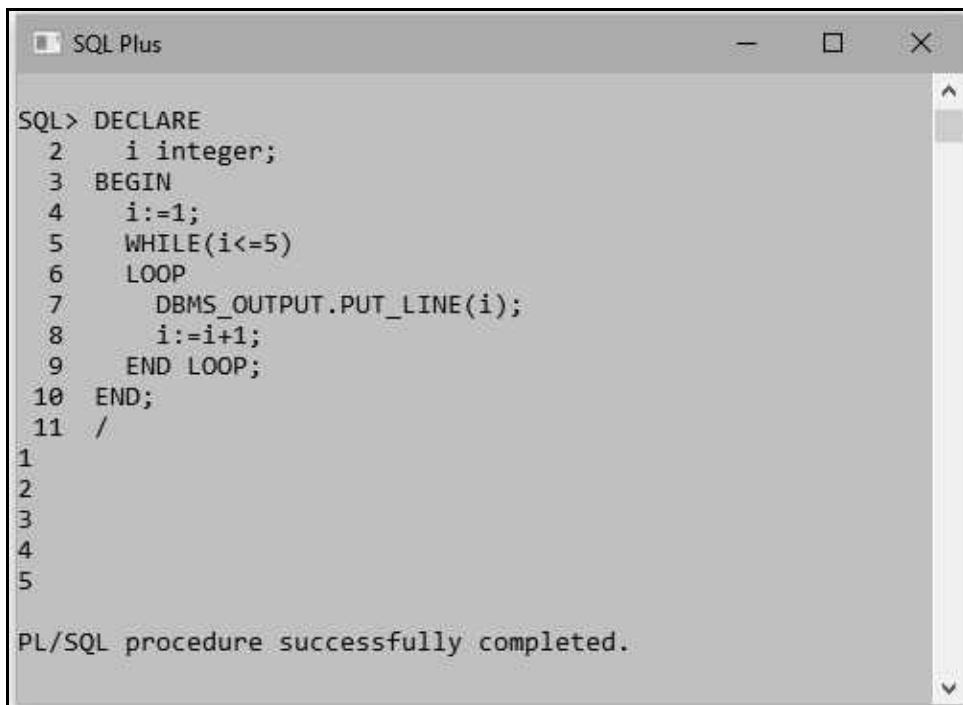
Syntax

```
WHILE <condition>
LOOP statement1;
  statement2;
END LOOP;
```

Programming Example

```
DECLARE
  i integer;
BEGIN
  i:=1;
  WHILE(i<=5)
  LOOP
    DBMS_OUTPUT.PUT_LINE(i);
    i:=i+1;
  END LOOP;
END;
/
```

Output



```
SQL Plus
SQL> DECLARE
 2  i integer;
 3  BEGIN
 4  i:=1;
 5  WHILE(i<=5)
 6  LOOP
 7  DBMS_OUTPUT.PUT_LINE(i);
 8  i:=i+1;
 9  END LOOP;
10  END;
11  /
1
2
3
4
5
PL/SQL procedure successfully completed.
```

3.7.5 CASE Statement

- The CASE statement is just similar to switch case statement in C.
- It allows you to execute the sequence of statements based on selector. The selector can be a variable, or a function or some expression.
- A CASE statement is evaluated from top to bottom. If it get the condition TRUE, then the corresponding THEN clause is executed and the execution goes to the END CASE clause.

Syntax

```
CASE [expression]
WHEN condition1 THEN statement1
  WHEN condition2 THEN statement2
  ...
  WHEN conditionn THEN statementn
ELSE someStatement
END
```

Programming Example

```
SET SERVEROUTPUT ON;
DECLARE
  grade CHAR(1) := 'A';
BEGIN
  CASE grade
  WHEN 'A' THEN
    DBMS_OUTPUT.PUT_LINE('Distinction');
  WHEN 'B' THEN
    DBMS_OUTPUT.PUT_LINE('First Class');
  WHEN 'C' THEN
    DBMS_OUTPUT.PUT_LINE('Higher Second Class');
  WHEN 'D' THEN
    DBMS_OUTPUT.PUT_LINE('Second Class');
  WHEN 'F' THEN
    DBMS_OUTPUT.PUT_LINE('Pass');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Fail');
  END CASE;
END;
/
```

Output

```
Distinction
```

3.8 Handling Database Tables using PL/SQL

We can create a database table in Oracle database using two ways either using SQL prompt or using Oracle SQL Developer.

(1) Creating a Database Table Using SQL PLUS

Open SQL PLUS window, type the user-name and password. I am using HR schema under which the table will be created. The password which I have set during installation of ORACLE is HR. So by entering correct user name and password the SQL prompt will appear. Just give the CREATE table query for creating the table.

Following screenshot illustrates creation of table named T with two fields - **roll** and **name**.

```
SQL Plus
Enter user-name: HR
Enter password:

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> CREATE TABLE T(
 2  roll          NUMBER(2),
 3  name         VARCHAR2(20)
 4  );

Table created.

SQL>
```

Using DESC command the table structure can be displayed.

```
SQL Plus

SQL> DESC T
Name                                Null?    Type
-----
ROLL                                NUMBER(2)
NAME                                VARCHAR2(20)

SQL>
```

For inserting the values into the table we use INSERT command as follows -

```
SQL Plus

SQL> INSERT INTO T VALUES(10, 'ABC');

1 row created.

SQL>
```

The table can then be displayed using SELECT command as follows -

```
SQL Plus

SQL> SELECT * FROM T;

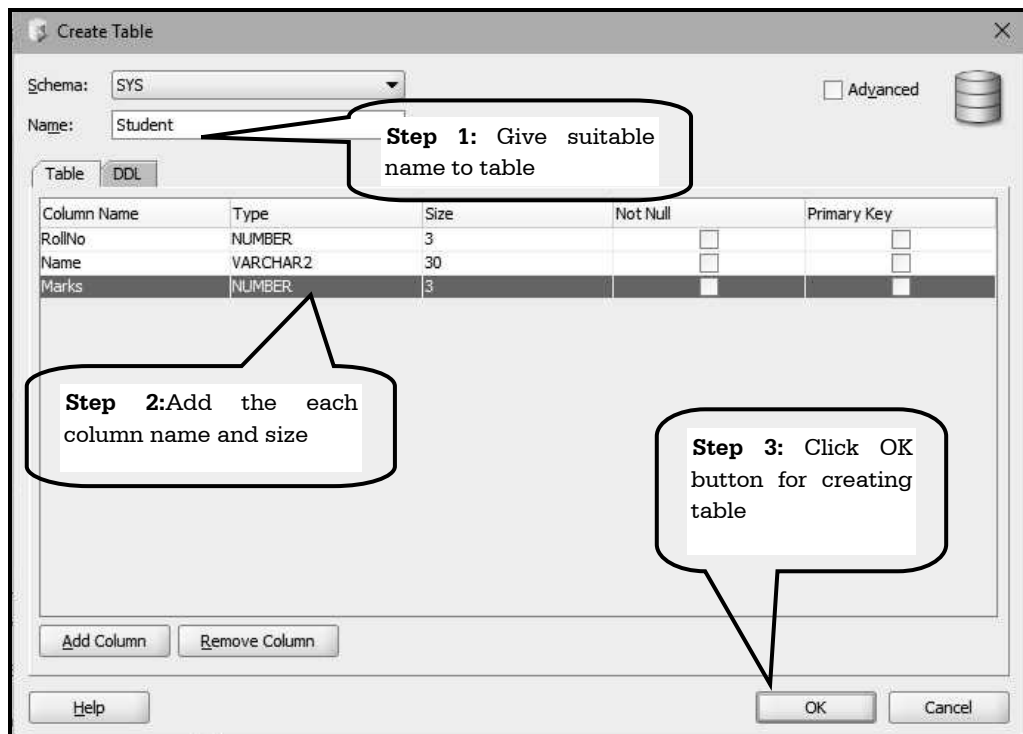
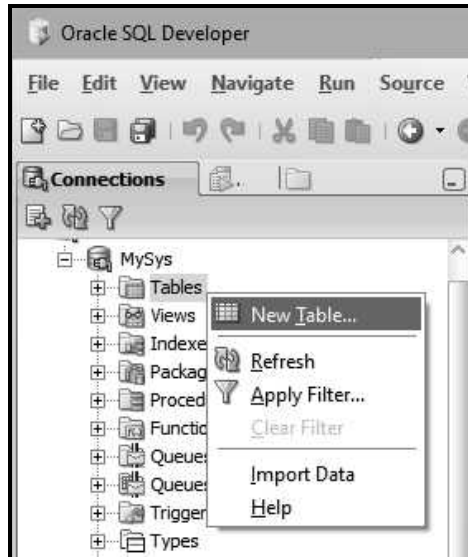
  ROLL NAME
-----
   10 ABC

SQL>
```

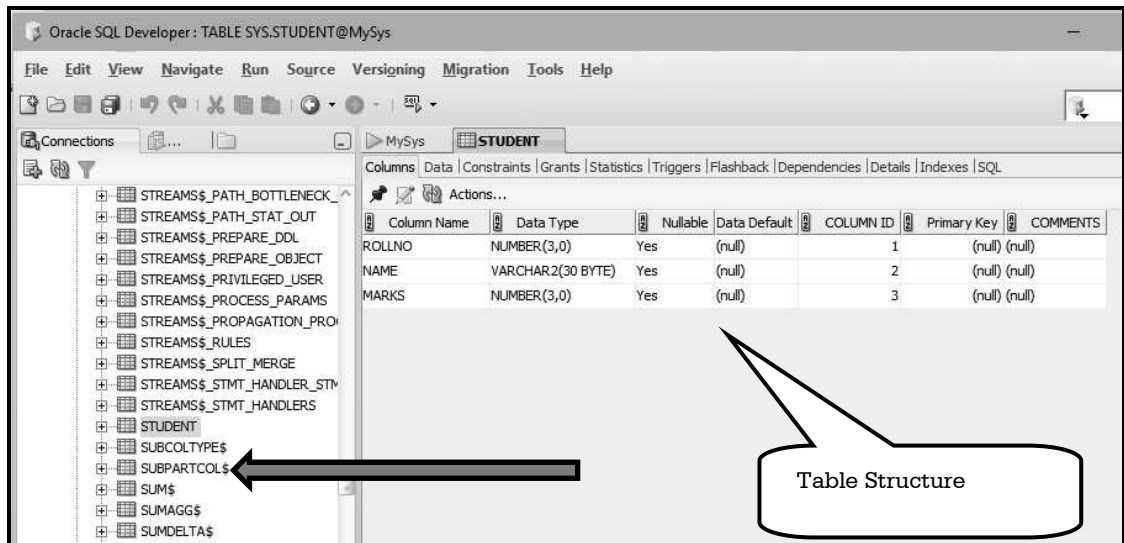
(2) Creating a Database Table Using SQL Developer

Open Oracle SQL Developer and follow these steps -

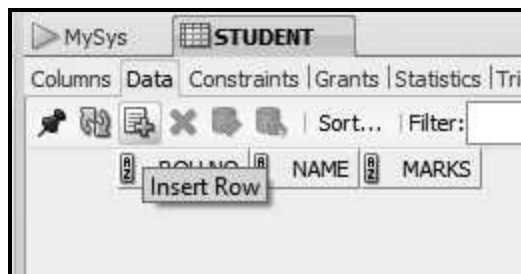
Step 1 : Expand your Database Connection node. Find out the node Table. Just Right click on it, click on New Table. Here I have already created a connection by name MySys (Refer section A.3(2) to know how to create a connection), so I am creating a table under this schema. Here is the screenshot.



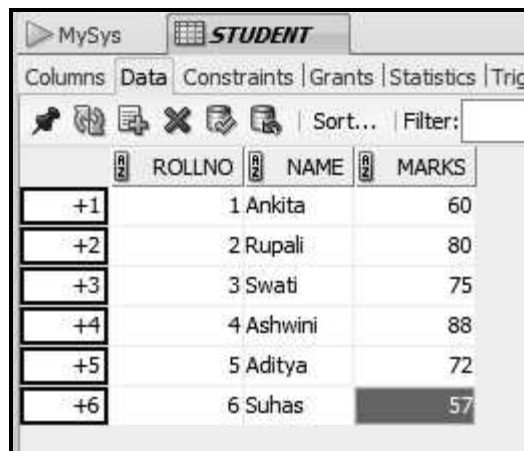
Now you can see your table structure by Selecting Student Table from Table node of your database Connection List. Just refer following screenshot.



Just click on **Data** tab for insertion of data into the table -



The data can be inserted into the table by clicking to enter the value to each column -



Then click Save button. The data will be inserted into the table.

3.9 Examples on PL/SQL

Example 3.9.1 Write PL/SQL block of code which accepts the roll.no. from user, the attendance of roll no entered by user will be checked in stud_att (Roll_no, Att) table. Attendance of Roll no entered is displayed on screen.

SPPU : Aug.-17, End Sem, Marks 5

Solution :

Create Table

```
CREATE TABLE TESTUDENTS(Roll_no NUMBER(3)PRIMARY KEY, att NUMBER(5));
```

Insert Data

```
INSERT INTO "SYS"."TESTUDENTS" (ROLL_NO, ATT) VALUES ('1', '130')
INSERT INTO "SYS"."TESTUDENTS" (ROLL_NO, ATT) VALUES ('2', '155')
INSERT INTO "SYS"."TESTUDENTS" (ROLL_NO, ATT) VALUES ('3', '180')
INSERT INTO "SYS"."TESTUDENTS" (ROLL_NO, ATT) VALUES ('4', '100')
INSERT INTO "SYS"."TESTUDENTS" (ROLL_NO, ATT) VALUES ('5', '98')
INSERT INTO "SYS"."TESTUDENTS" (ROLL_NO, ATT) VALUES ('6', '190')
```

The table can be viewed as follows –

ROLL_NO	ATT
1	130
2	155
3	180
4	100
5	98
6	190

PL/SQL Program(Without Function)

```
SET SERVEROUTPUT ON;
DECLARE
temp_rollno NUMBER(3);
temp_att NUMBER(5);
total_days NUMBER(5) := 200;
BEGIN
temp_rollno := &temp_rollno;
SELECT att INTO temp_att FROM TESTUDENTS WHERE roll_no = temp_rollno;
DBMS_OUTPUT.put_line('Roll No = ' || temp_rollno || ' Attendance = ' || temp_att);
END;
/
```

Output

```
anonymous block completed
Roll No = 5 Attendance = 98
```

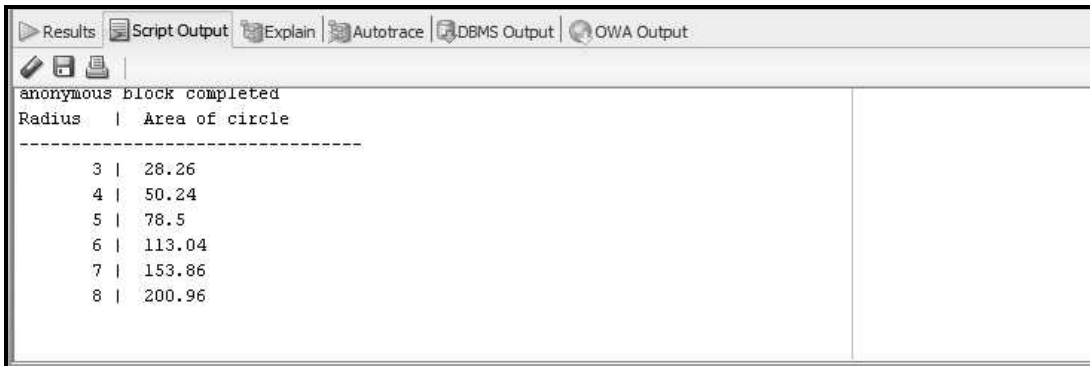
Example 3.9.2 Write a PL/SQL code block for displaying the area of circle with radius values from 3 to 8. (Formula to calculate area of circle is $\text{Pi} \times \text{radius} \times \text{radius}$).

SPPU : Oct.-18, In Sem, Marks 5

Solution :

```
SET SERVEROUTPUT ON;
DECLARE
-- The radius is declared in 'radius' variable and result in 'area' variable
radius NUMBER;
area NUMBER(10,2);
BEGIN
DBMS_OUTPUT.put_line('Radius | Area of circle ');
DBMS_OUTPUT.put_line('-----');
FOR radius IN 3..8 LOOP
area:=3.14* radius* radius;
DBMS_OUTPUT.put_line(' || radius || ' | ' || area);
END LOOP;
END;
/
```

Output



Radius	Area of circle
3	28.26
4	50.24
5	78.5
6	113.04
7	153.86
8	200.96

Example 3.9.3 Write the PL/SQL block of code to calculate the factorial value of a number.

SPPU : May-19, Dec.-19, End Sem, Marks 5

Solution :

```
SET SERVEROUTPUT ON;
DECLARE
-- The factorial f is initialized to 1
f NUMBER :=1;
--user inputs the number here in variable n
n NUMBER :=&1;
num NUMBER := n;
BEGIN
WHILE n>0 LOOP
  f:=n*f;
  n:=n-1;
END LOOP;
DBMS_OUTPUT.put_line('Factorial of ' || num || ' is ' || f);
END;
/
```

Output

Factorial of 7 is 5040

Example 3.9.4 Write PL/SQL block of code for following requirement :

Student_fees (PRN, S_name, class, fees_paid).

Accept the PRN of student from user, check the fees paid by student, if fees paid is less than 30,000 then display the message on screen not paid full fees and display the total fees due. If fees_paid is greater than or equal to 30,000 then display message no fees due.

SPPU : Oct.-19, In Sem, Marks 5

Solution : Step 1 : Create Table using following SQL command

```
CREATE TABLE STUDENT_FEES
(PRN NUMBER(5),
```

```

S_NAME VARCHAR2(30),
CLASS VARCHAR2(20),
FEES_PAID NUMBER(10)
};

```

Insert some data into this table. The sample table can be viewed as

PRN	S_NAME	CLASS	FEES_PAID
101	AAA	TEComp	30000
102	BBB	SEMech	25000
103	CCC	BECivil	35000
104	DDD	FE	28000

Step 2 : Following is the PL/SQL code that checks whether the fees paid fully or not.

PL/SQL Program

```

SET SERVEROUTPUT ON;
DECLARE
temp_PRN NUMBER(5);
temp_fees NUMBER(10);
BEGIN
temp_PRN := &temp_PRN;
SELECT fees_paid INTO temp_fees FROM STUDENT_FEES WHERE PRN = temp_PRN;
DBMS_OUTPUT.put_line('PRN = ' || temp_prn);
DBMS_OUTPUT.put_line('Fees(Rs.) = ' || temp_fees);
IF(temp_fees<30000) THEN
DBMS_OUTPUT.put_line('The Feees not Paid Fully');
ELSE
DBMS_OUTPUT.put_line('No Fees Due');
END IF;
END;
/

```

Output(Run1)

```

PRN = 101
Fees(Rs.) = 30000
No Fees Due

```

3.10 Concept of Stored Procedures

- Stored procedure is a type of subprogram in PL/SQL block. It is a group of statements that can be called by its name.
- This is a subprogram that does not return a value directly.
- A procedure is created with the CREATE OR REPLACE PROCEDURE statement.

Syntax

```
CREATE or REPLACE Procedure Procedure_Name  
[(Parameter_Name [IN | OUT | IN OUT ] Type [...])]  
[IS | AS]  
BEGIN  
Procedure_Body  
END;
```

- Where
 - **Procedure_Name** is used to specify the name of the procedure.
 - **CREATE** keyword is used to develop a new procedure and [OR REPLACE] option allows us to modify an existing procedure.
 - The optional **parameter list** contains name and types of parameters.
- There are three **ways to pass parameters in procedures** -
 - **IN** represents that argument value will be passed from outside the procedure. It is a read-only parameter. Parameters are passed by reference. Inside the procedure or a sub-program, an IN Parameter acts as a constant. It cannot be assigned a value. It is the default mode of parameter passing.
 - An **OUT** parameter returns a value to the calling program. OUT represents that this parameter will be used to return a value outside of the procedure.
 - An **IN OUT** parameter passes an initial value to the sub-program and returns an updated value to the caller. We can read and write values using this parameter.
- The executable part is included in the procedure body.
- We can create a procedure without passing the parameters or by passing the parameters.
- We can execute the procedure using **Execute** keyword.
- A standalone procedure can be called by using the EXECUTE keyword or by calling or mentioning the procedure by its name from a PL/SQL block.

Syntax

```
Execute [Procedure-Name];
```

3.10.1 Procedures without Parameter

Let us write a simple procedure for displaying some welcome message.

Step 1 : Create a simple sql file that contains a procedure

```
SET SERVEROUTPUT ON;  
CREATE OR REPLACE PROCEDURE MyMessage ← Creation of procedure
```



```
IS
BEGIN
  dbms_output.put_line('Welcome friend!!!');
END;
/
EXECUTE MyMessage; ← This is for executing the procedure
```

Step 2 : Save and compile the code. The output will be as follows -

```
PROCEDURE MyMessage Compiled.
anonymous block completed
Welcome friend!!!
```

Another way of execution of procedure

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE MyMessage ← Creation of procedure
IS
BEGIN
  dbms_output.put_line('Welcome friend!!!');
END;
/
BEGIN ← This is for calling the procedure
  MyMessage;
END;
/
```

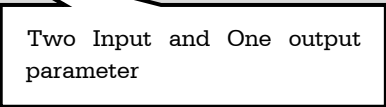
Save and compile above code to get the output.

3.10.2 Procedures with Parameters

Following example show the procedure for addition of two numbers. The result is stored in third variable which is an output variable.

Step 1 : We will create a procedure and store it in an sql file as follows

```
CREATE OR REPLACE PROCEDURE AddProc(x IN NUMBER, y IN NUMBER, z OUT NUMBER)
IS
BEGIN
  z:=x+y;
END;
/
```



Two Input and One output parameter

Save and compile above code.

Step 2 : The call to the procedure can be made from another file. The code for calling the procedure for execution is as follows -

```
SET SERVEROUTPUT ON;
DECLARE
a NUMBER;
b NUMBER;
c NUMBER;

BEGIN
a := 10;
b := 20;
AddProc(a,b,c);
dbms_output.put_line('Addition of two numbers:| |c);
END;
```

Save and compile above code

Step 3 : The output will be as follows -

```
anonymous block completed
Addition of two numbers:30
```

3.10.3 Stored Procedure for Handling Database Table

Using a stored procedure we can access the database table and can perform insertion, deletion and updation of data.

Following is a simple example of procedure using which we insert some record in student table.

Step 1 : Write a procedure as follows

```
CREATE OR REPLACE PROCEDURE InsertStudent(roll IN NUMBER,
name IN VARCHAR2,
marks IN NUMBER,
branch IN VARCHAR2)
IS
BEGIN
INSERT INTO student VALUES(roll,name,marks,branch);
END;
/
```

Save and compile the above code.

Step 2 : Now call the procedure. The code is as follows

```
SET SERVEROUTPUT ON;
BEGIN
  InsertStudent(7,'Akash',78,'CIVIL');
  DBMS_OUTPUT.PUT_LINE('One row inserted in Student Table!!!');
END;
/
```

Save and compile the above code.

Step 3 : The output can be as follows

```
anonymous block completed
One row inserted in Student Table!!!
```

Step 4 : You can cross-verify the above execution by opening the student table. It should display new row inserted into it.

ROLLNO	NAME	MARKS	BRANCH
1	Ankita	60	CS
2	Rupali	80	Mechanical
3	Swati	75	Electrical
4	Ashwini	88	Civil
5	Aditya	72	IT
6	Suhas	57	Electronics
7	Akash	78	CIVIL

Inserted new row

3.11 Functions

SPPU : Aug.-17, Marks 5

- Stored function is a named block or subprogram in PL/SQL.
- In PL/SQL, a function takes one or more parameter and returns one value.
- The syntax for a function in PL/SQL is as follows :

```
CREATE [Or REPLACE] Function Function_Name
[(Parameter,...)]
Return Datatype
[IS | AS]
[Declaration Section]
BEGIN
```

```
[Executable Section]
END Function_Name;
```

Let us now understand how to write a function in PL/SQL with the help of example

Step 1 : Open New file in Oracle SQL Developer and write following code for creating a simple function for addition of two numbers.

AdditionFunction.sql

```
CREATE OR REPLACE FUNCTION Addion_fun(a IN NUMBER, b IN NUMBER)
RETURN NUMBER IS
c NUMBER;
BEGIN
c := a+b;
RETURN c;
END;
/
```

Save this file and compile it.



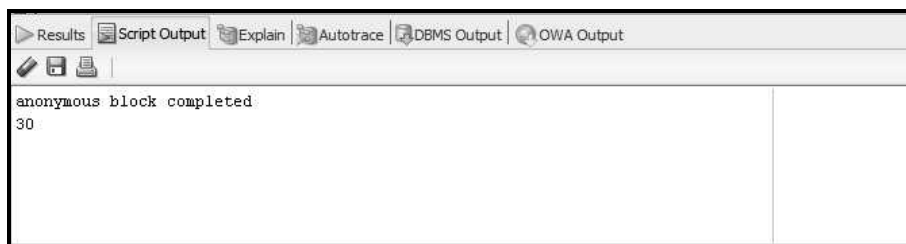
Step 3 : Create another sql file for calling the above function in an anonymous block.

CallAdditionFun.sql

```
SET SERVEROUTPUT ON;
BEGIN
DBMS_OUTPUT.PUT_LINE(addion_Fun(10,20));
END;
/
```

Save above code and compile it.

Step 4 : On execution of the above code you get the output as



3.11.1 PL/SQL Stored Function for Table

We can write PL/SQL stored Function can be used to perform some functionality on the database tables. Following example shows that we can write a function to find total number of students from **STUDENT** table. Here is a stepwise illustration -

Step 1 : Create a student table as follows -

ROLLNO	NAME	MARKS	BRANCH
1	Ankita	60	CS
2	Rupali	80	Mechanical
3	Swati	75	Electrical
4	Ashwini	88	Civil
5	Aditya	72	IT
6	Suhas	57	Electronics

Step 2 : Now create an sql file and create a function in it.

DisplayTotalFun.sql

```
CREATE OR REPLACE FUNCTION Total_Students
RETURN NUMBER IS
  t NUMBER(2) := 0;
BEGIN
  SELECT COUNT(*) INTO t
  FROM student;
  RETURN t;
END;
/
```

Save and compile above program

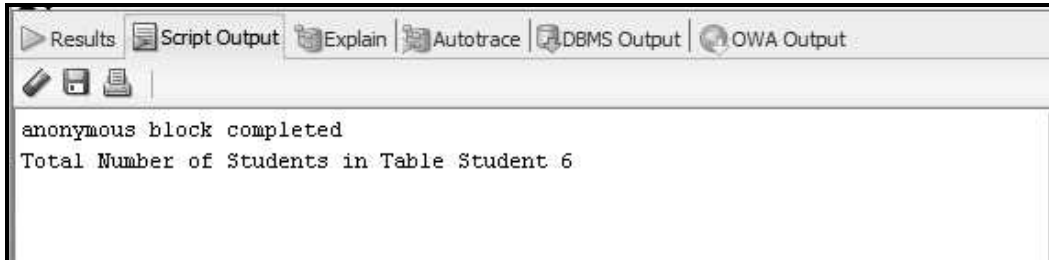
```
FUNCTION Total_Students Compiled.
```

Step 3 : Now we will write sql file for calling the above function.

CallFun.sql

```
SET SERVEROUTPUT ON;
DECLARE
  t NUMBER(2);
BEGIN
  t := Total_Students();
  DBMS_OUTPUT.PUT_LINE('Total Number of Students in Table Student ' || t);
END;
/
```

Save above code and compile it.



```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output
anonymous block completed
Total Number of Students in Table Student 6

```

Example 3.11.1 Write a PL/SQL function that returns the name of the student whose roll number is given by user.

Solution :

Step 1 : Create a student table. The sample table is as follows -

ROLLNO	NAME	MARKS	BRANCH
1	Ankita	60	CS
2	Rupali	80	Mechanical
3	Swati	75	Electrical
4	Ashwini	88	Civil
5	Aditya	72	IT
6	Suhas	57	Electronics

Step 2 : Write a function for retrieving the name of the student from the STUDENT table

```

CREATE OR REPLACE FUNCTION Display(roll NUMBER)
RETURN VARCHAR2 IS
  s_name VARCHAR2(20);
BEGIN
  SELECT name INTO s_name
  FROM student
  WHERE ROLLNO=roll;
  RETURN s_name;
END;
/

```

Save and compile above code.


Step 3 : Now call the above function

```

SET SERVEROUTPUT ON;
DECLARE
  s_name VARCHAR2(20);
BEGIN
  s_name := Display(2);
  DBMS_OUTPUT.PUT_LINE('Name Student with Rollno 2 is: ' || s_name);
END;
/

```

Save and compile above file. You will get following output



```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output
anonymous block completed
Name Student with Rollno 2 is: Rupali

```

Example 3.11.2 Write PL/SQL block of code which accepts the roll.no. from user, the attendance of roll no entered by user will be checked in stud_att (Roll_no, Att) table. Attendance of Roll no entered is displayed on screen.

SPPU : Aug.-17, End Sem, Marks 5

Solution : PL/SQL Program

Step 1 : Create a function in a sql file as

example.sql

```

CREATE OR REPLACE FUNCTION Display(roll NUMBER)
RETURN NUMBER IS
temp_att NUMBER(5);
BEGIN
SELECT att INTO temp_att FROM TESTUDENTS WHERE roll_no = roll;
RETURN temp_att;
END;
/

```

Step 2 : Following is a driver program that calls above created function

driver.sql

```

SET SERVEROUTPUT ON;
DECLARE
temp_rollno NUMBER(3);
temp_att NUMBER(5);
total_days NUMBER(5) := 200;
BEGIN
temp_rollno := &temp_rollno;
temp_att := Display(temp_rollno);
DBMS_OUTPUT.put_line('Roll No = ' || temp_rollno || ' Attendance = ' || temp_att);
END;
/

```

Output

```
Roll No = 4 Attendance = 100
```

3.12 Cursors

SPPU : Dec.-17, Marks 5

- When an SQL statement is processed, Oracle creates a memory area known as **context area**. A cursor is a pointer to this context area.

- It contains all information needed for processing the statement.
- In PL/SQL, the context area is controlled by cursor.
- A cursor contains information on a SELECT statement and the rows of data accessed by it.
- The cursor is used to fetch and process the rows returned by SQL statement one at a time.
- There are two types of cursors -

(1) Implicit cursor (2) Explicit cursor

(1) Implicit cursor

- Whenever Oracle executes an SQL statement such as SELECT INTO, INSERT, UPDATE and DELETE, it creates an implicit cursor.
- The Implicit Cursor is the Default Cursor in PL/SQL block.
- Oracle provides some attributes known as Implicit cursor's attributes to check the status of DML operations. Some of them are as follows -
 - **%ROWCOUNT** : It returns the number of rows influenced by an UPDATE, DELETE or an INSERT statement.
 - **%NOTFOUND** : It returns TRUE if an UPDATE, DELETE or an INSERT statement influenced zero rows or a SELECT INTO statement returned no rows. Else, it returns a FALSE.
 - **%FOUND** : It returns TRUE if an UPDATE, DELETE or an INSERT statement influenced one or more rows or a SELECT INTO statement returned one or more rows. Else, it returns FALSE.
 - **%ISOPEN** : It returns FALSE for Implicit cursors as Oracle closes the PL/SQL cursor by default after processing its associated PL/SQL statement.

Programming Example : Following is an example program that uses attributes of implicit cursor to demonstrate number of rows affected when SELECT query is executed for Student table

```
SET SERVEROUTPUT ON;
DECLARE
s_name student.name%type;
BEGIN
SELECT name INTO s_name
FROM student
WHERE marks >= 70;
DBMS_OUTPUT.PUT_LINE('Number of rows processed: ' || sql%rowcount);
END;
/
```

Output

Number of rows processed:5

(2) Explicit cursor

- Explicit cursors are used when you are executing a SELECT statement query that will return more than one row.
- Cursors can process one record at a given point of time even though it stores more than one record.
- An explicit cursor is defined in the declaration section of the PL/SQL Block.
- Explicit cursors are used if you need to have better control over the Context Area via Cursor.

Syntax for Declaration of Explicit Cursor

```
CURSOR cursor_name  
IS  
SELECT statement
```

For example

```
CURSOR MyCursor  
IS  
SELECT * FROM Student;
```

The explicit cursor works in four stages -

- 1) **Declaration of cursor** : The declaration of cursor is in declaration section of PL/SQL block. The name of the cursor requires to be defined along with the SELECT Statement.

Syntax

```
CURSOR cursor-name IS  
SELECT statement;
```

- 2) **Open the cursor** : Opening the cursor also means allocating the memory for the cursor in the context area which thereby makes it sufficient to fetch and store records in it.

Syntax

```
OPEN cursor-name;
```

- 3) **Fetch the cursor** : Fetching the cursor involves retrieval of data using the fetch statement. It is used to help the cursor process and access records or rows at a time.

Syntax

```
FETCH cursor_name INTO variable_list;
```

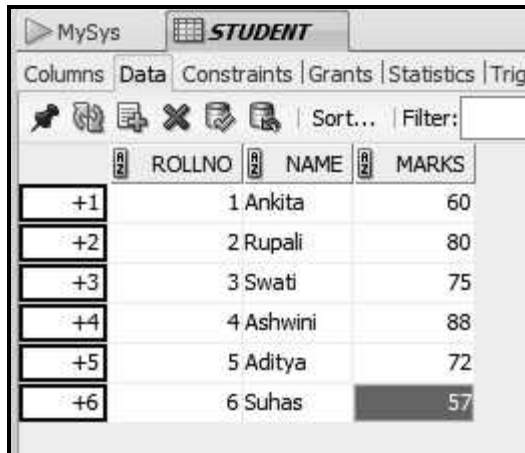
- 4) **Close the cursor** : Once the work associated for a cursor to be completed is accomplished, it is necessary to release the allocated memory of the cursor to let other tasks occupy memory.

Syntax

```
Close cursor_name;
```

Now let us understand how to write a simple explicit cursor and execute it.

Step 1 : Create a database table for using cursor. I have already created a student table in section 3.8 (2). It is as follows :



	ROLLNO	NAME	MARKS
+1	1	Ankita	60
+2	2	Rupali	80
+3	3	Swati	75
+4	4	Ashwini	88
+5	5	Aditya	72
+6	6	Suhas	57

Step 2 : Under your current connection type(In my case it is **MySys**), Open **File->New File**. Give suitable name to your file which stores the cursor program. I have given the name as **CursorDemo.sql**

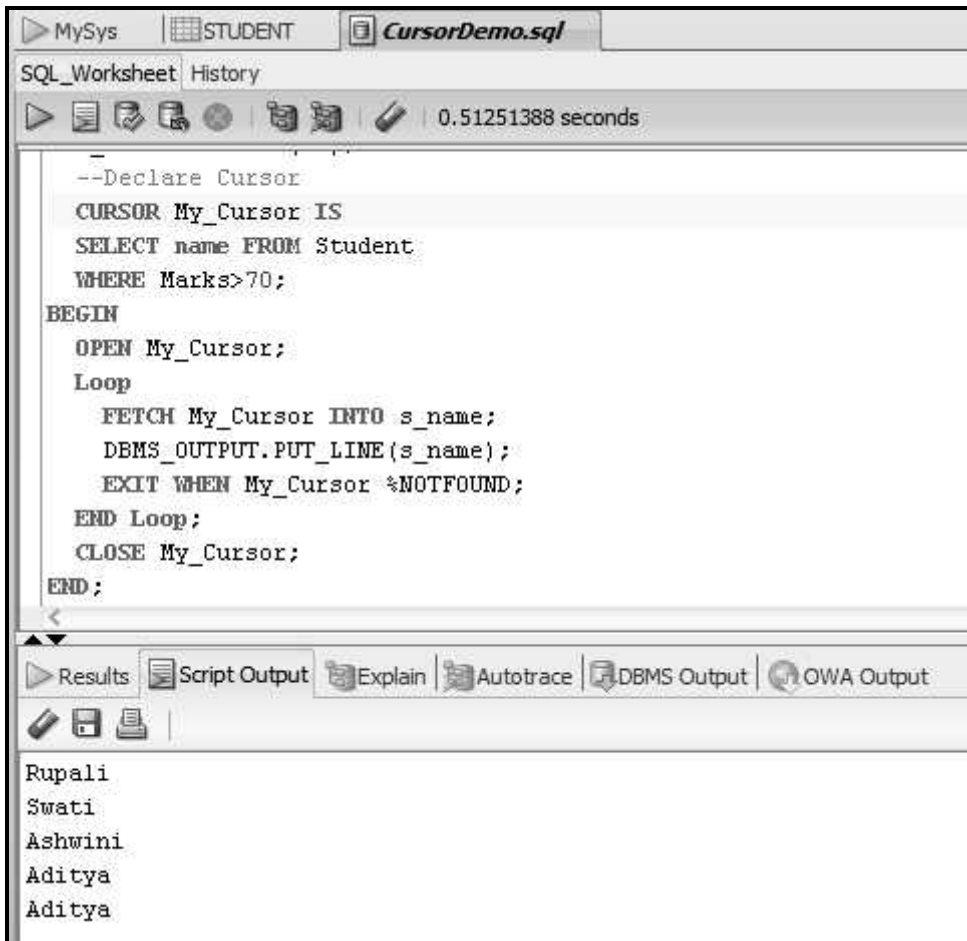
Step 3 : Write the program as follows

```
SET SERVEROUTPUT ON;
DECLARE
s_name VARCHAR2(30);
--Declare Cursor
CURSOR My_Cursor IS
SELECT name FROM Student
WHERE Marks>70;
BEGIN
OPEN My_Cursor;
Loop
FETCH My_Cursor INTO s_name;
DBMS_OUTPUT.PUT_LINE(s_name);
EXIT WHEN My_Cursor %NOTFOUND;
END Loop;
CLOSE My_Cursor;
END;
```

Step 4 : Save your file and execute script by either hitting F5 or by Run Script



The output will be obtained as a list of names of those students who have marks > 70. Here is the output



The screenshot shows a SQL IDE window titled 'CursorDemo.sql'. The main editor contains the following PL/SQL code:

```
--Declare Cursor
CURSOR My_Cursor IS
SELECT name FROM Student
WHERE Marks>70;
BEGIN
OPEN My_Cursor;
Loop
  FETCH My_Cursor INTO s_name;
  DBMS_OUTPUT.PUT_LINE(s_name);
  EXIT WHEN My_Cursor %NOTFOUND;
END Loop;
CLOSE My_Cursor;
END;
```

Below the code editor, the 'Results' tab is active, displaying the output of the cursor loop:

```
Rupali
Swati
Ashwini
Aditya
Aditya
```

Program Explanation : In above code,

- The cursor is created to display the names of students who have scored more than 70 marks.
- The name of the cursor is My_Cursor. The SELECT query associated with it is `SELECT name FROM STUDENT WHERE Marks > 70;`
- Then we open the My_Cursor.
- Inside the LOOP, we fetch students' name from student table into variable s_name and each time the s_name is displayed. This loop is exited when no record is found. For that purpose the attribute %NOTFOUND is used.
- Finally the cursor is closed.

Example 3.12.1 The organization has decided to increase the salary of employees by 10 % of existing salary, whose existing salary is less than ₹ 10000/-. Write a PL/SQ block to update the salary as per above requirement, display an appropriate message based on the number of rows affected by this update (using implicit cursor status variables).

SPPU : Dec.-17, End Sem, Marks 5

Solution :

Step 1 : Create an employee table as follows -

R#	EMP_NO	EMP_NAME	SALARY
1	101	AAA	25000
2	102	BBB	12000
3	103	CCC	7000
4	104	DDD	5000
5	105	EEE	22000
6	106	FFF	9000

PL/SQL Program

```
SET SERVEROUTPUT ON;
```

```
BEGIN
```

```
UPDATE emp_table
```

```
SET salary = salary+(salary*0.1)
```

```
WHERE salary < 10000;
```

SQL query to increase the salary(which is less than 10000) by 10%.

```
DBMS_OUTPUT.put_line('Number of records updated= ' || sql%rowcount);
```

```
END;
```

```
/
```

Use of implicit cursor variable

Output

```
Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output
anonymous block completed
Number of records updated= 3
```

Now If we open the database table we get the updated values of the salary as

	EMP_NO	EMP_NAME	SALARY
1	101	AAA	25000
2	102	BBB	12000
3	103	CCC	7700
4	104	DDD	5500
5	105	EEE	22000
6	106	FFF	9900

3.13 Triggers

SPPU : Dec.-17, 18, 19, Marks 5

- Trigger is something that is invoked automatically when some event occurs.
- PL/SQL triggers are **block structures or pre-defined programs**, which may be in-built or even explicitly developed by the programmers for a particular task.
- Trigger is **stored into database** and invoked repeatedly, when specific condition match.
- Triggers are stored programs, which are automatically executed or fired when some event occurs.
- Triggers are associated with response-based events such as a,
 - Database Definition Language (DDL) statement such as CREATE, DROP or ALTER .
 - Database Manipulation Language (DML) statement such as UPDATE, INSERT or DELETE.
 - Any other database operation such as a Startup, Shutdown, Logging in and Logging Out.

Syntax

```
CREATE OR REPLACE TRIGGER Trigger_Name
BEFORE or AFTER or INSTEAD OF
INSERT or UPDATE or DELETE
of Column_Name
ON Table_Name
[REFERENCING OLD AS O New AS N]
FOR EACH ROW
WHEN (Condition)
DECLARE
```

```

Declaration Section
BEGIN
Execution Section
END;

```

Where

- CREATE [OR REPLACE] TRIGGER trigger_name: It creates or replaces an existing trigger with some trigger_name.
- {BEFORE | AFTER | INSTEAD OF} : This specifies when the trigger would be executed. The INSTEAD OF clause is used for creating trigger on a view.
- {INSERT [OR] | UPDATE [OR] | DELETE} : This specifies the DML operation.
- [OF col_name] : This specifies the column name that would be updated.
- [ON table_name] : This specifies the name of the table associated with the trigger.
- [REFERENCING OLD AS o NEW AS n]: This allows you to refer new and old values for various DML statements, like INSERT, UPDATE, and DELETE.
- [FOR EACH ROW] : This specifies a row level trigger, i.e., the trigger would be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.
- WHEN (condition) : This provides a condition for rows for which the trigger would fire. This clause is valid only for row level triggers.

Programming Example

Let us now see how to use trigger with the help of some example -

Example 3.13.1 Write a trigger for insertion of a row into a person table. On firing the trigger name of the user who performed insertion operation should be displayed.

Solution :

```

CREATE TABLE PersonTab (
  pname VARCHAR2(20)
);

```

Here table named **PersonTab** is created

```

SET SERVEROUTPUT ON;
CREATE OR REPLACE TRIGGER MyTrigger
BEFORE INSERT ON PersonTab
FOR EACH ROW
ENABLE
DECLARE
  usr_name VARCHAR2(20);
BEGIN
  SELECT user INTO usr_name FROM dual;

```

Code for actual trigger

```

DBMS_OUTPUT.PUT_LINE('Inserted a new row by user: '||usr_name);
END;
/
INSERT INTO PersonTab VALUES('Sharda');

```

SQL query on execution of which the trigger gets fired.

Output

```

*Cause:
*Action:
TRIGGER MyTrigger Compiled.
1 rows inserted
Inserted a new row by user: HR

```

You can cross-verify the insertion of data into the table by opening the corresponding table.

For instance -

PNAME
1 Sharda

Example 3.13.2 Write PL/SQL trigger for following requirement :

Event : Deletion of row from stud (roll_no, name, class) table.

Action : After deletion of values from stud table, values should be inserted into cancel_admission (roll_no, name) table

Note : For every row to be deleted, action should be performed.

SPPU : Dec.-17, End Sem, Marks 5

Solution :

Step 1 : Create table **stud_table** using following SQL statement

```

CREATE TABLE stud_table(
roll_no NUMBER(5),
sname VARCHAR2(30),
sclass VARCHAR2(20)
);

```

Step 2 : Insert data into table using following commands

```

INSERT INTO stud_table VALUES(111,'AAA','TECOMP');
INSERT INTO stud_table VALUES(222,'BBB','SEMECH');
INSERT INTO stud_table VALUES(333,'CCC','BECIVIL');
INSERT INTO stud_table VALUES(444,'DDD','TEETC');

```

Step 3 : Create table **cancel_admission** using following SQL statement

```

CREATE TABLE cancel_admission(
roll_no NUMBER(5),
sname VARCHAR2(30)
);

```

Step 4 : Following is a PL/SQL program that deletes data from **stud_table** and insert the deleted data into **cancel_admission** table

TriggerDemo1.sql

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE TRIGGER Stud_AD
AFTER DELETE ON stud_table
FOR EACH ROW
ENABLE

BEGIN
INSERT INTO cancel_admission
(roll_no,
 sname)
VALUES
( :old.roll_no,
 :old.sname);
END;
/
DELETE FROM stud_table WHERE roll_no=222
```

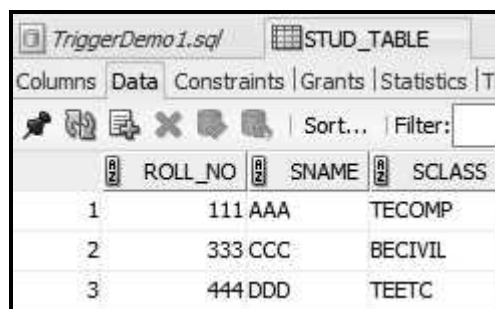
Output

On compiling above trigger we get the output as follows -

```
TRIGGER Stud_AD Compiled.
1 rows deleted
```

The execution of this trigger can be verified by observing the data from student_table and cancel_admission table.

Stud_table Table



	ROLL_NO	SNAME	SCLASS
1	111	AAA	TECOMP
2	333	CCC	BECIVIL
3	444	DDD	TEETC

Cancel_admission Table

ROLL_NO	SNAME
1	222 BBB

Example 3.13.3 Consider the schema : *student_fees_detail* (*name*, *total_fees_deposited*, *till_date*)

Answer the following :

- i) Write a SQL query to display the total fees deposited by students whose minimum 3 character name starts with aj.
- ii) Write database trigger to preserve the old values of student fees details before updating in table.

SPPU : Dec.-18,19, End Sem, Marks 5

Solution :

i) SQL Query is as follows -

```
SELECT name,total_fees_deposited
FROM student_fees_detail
WHERE LEN(name)>2 AND name LIKE 'aj%'
```

ii) Step 1 : First of all we will create the table **stud_fees_detail** using following SQL statement.

```
CREATE TABLE stud_fees_detail
(
  name VARCHAR2(30),
  fees_deposited NUMBER(5),
  till_date DATE
);
```

Step 2 : Insert values the values into this table using following SQL statement -

```
INSERT INTO stud_fees_detail VALUES('AAA',2000,'27-JUNE-19');
INSERT INTO stud_fees_detail VALUES('BBB',5000,'16-May-20');
INSERT INTO stud_fees_detail VALUES('CCC',1000,'01-August-19');
INSERT INTO stud_fees_detail VALUES('DDD',3000,'28-March-20');
INSERT INTO stud_fees_detail VALUES('EEE',2500,'12-November-20');
```

Step 3 : Simply create a table **backup_fees** using following SQL statement -

```
CREATE TABLE backup_fees
(
  name VARCHAR2(30),
  fees_deposited NUMBER(5),
  till_date DATE
);
```

Step 4 : The PL/SQL script for the updating values is as shown below

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE TRIGGER Stud_fees_AU
AFTER UPDATE ON stud_fees_detail
FOR EACH ROW
ENABLE

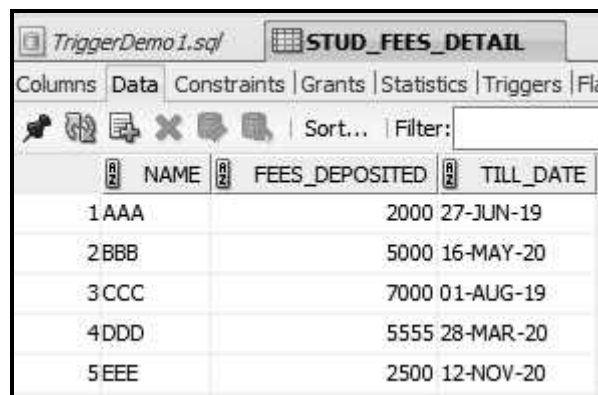
BEGIN
  INSERT INTO backup_fees
  (name,
  fees_deposited,
  till_date
  )
  VALUES
  ( :old.name,
  :old.fees_deposited,
  :old.till_date
  );
END;
/

UPDATE stud_fees_detail
SET fees_deposited=7000
WHERE name='CCC'

UPDATE stud_fees_detail
SET fees_deposited=5555
WHERE name='DDD'
```

Output

Stud_fees_detail Table(This table is updated as trigger executes)



The screenshot shows a database table viewer for the 'STUD_FEES_DETAIL' table. The table has three columns: 'NAME', 'FEES_DEPOSITED', and 'TILL_DATE'. The data is as follows:

ID	NAME	FEES_DEPOSITED	TILL_DATE
1	AAA	2000	27-JUN-19
2	BBB	5000	16-MAY-20
3	CCC	7000	01-AUG-19
4	DDD	5555	28-MAR-20
5	EEE	2500	12-NOV-20

Back_fees Table

NAME	FEES_DEPOSITED	TILL_DATE
1 CCC	1000	01-AUG-19
2 DDD	3000	28-MAR-20

3.14 Assertions

- An assertion is a predicate expressing a condition we wish the database to always satisfy.
- When created, the expression must be true.
- DBMS checks the assertion after any change that may violate the expression.
- **Syntax for creating an assertion**

```
CREATE ASSERTION <assertion-name> CHECK <predicate>
```

It must return true

- **Example**

Consider following relation of customer

```
customer(customer_name, customer_street, customer_city)
```

If we want that the customer city should not be NULL then the Assertion can be written as follows

```
CREATE ASSERTION city_not_null Check
( NOT EXISTS
(Select *
  From customer
  Where customer_city is null));
```

Difference between Assertion and Trigger

Sr. No.	Assertion	Trigger
1.	Assertions do not modify the data. They only check certain condition.	Triggers are powerful than assertions because they can check the condition as well as they can modify the data.
2.	Assertions are not linked to specific tables in the database and not linked to specific events.	Triggers are linked to specific tables and specific events.

3.	All assertions can be implemented as triggers.	All triggers cannot be implemented as assertions.
4.	Oracle does not support assertion.	Oracle support Triggers.

3.15 Roles and Privileges

Roles

- Role is a set of privileges that can be granted to users. The roles are used for administrating the database. The privileges can be added to the role and the role can be granted to the user.
- The role can be created in Oracle using CREATE ROLE command. For example -

```
CREATE ROLE data_manager;
```

- The IDENTIFIED BY clause is used for authentication of the user. It adds the security layer to the role. For instance - In following example, the role data_manager is created and it is identified using the password 'password 1234'.

```
CREATE ROLE data_manager.  
IDENTIFIED BY password1234
```

- The privileges or permissions to particular roles can be given with the command GRANT. For example -

```
GRANT create table,create view  
TO data_manager;
```

Advantages of role management

- It is possible to assign multiple roles to the user.
- One can assign password to particular role for security purpose.
- Rather than assigning privileges at one time to some user, we can create the role and assign privileges to the role and then grant that role to multiple users.
- We can add or delete some privileges to the role and all the users are assigned with that role get those privileges or loss that privileges automatically.

Privileges

- Privileges are the access rights provided to the user on database object.
- Some of the examples of privileges are -
 - Connect to database
 - Create a table
 - Select a row from another user's table.

- There are two types of privileges -
 1. **System privilege** : This allows user to CREATE, ALTER or DROP the database objects such as table, views and so on.
 2. **Object privilege** : This allows user to SELECT, INSERT, UPDATE, or DELETE data from database objects.

3.16 Exceptions

SPPU : Dec.-18, May-19, Marks 5

Exception is an unusual situation that when occurs interrupts the normal flow of execution. The exception handling mechanism allows to handle this runtime error situation in such a way that the normal flow of execution may not get interrupted.

There are two types of exceptions in PL/SQL

- 1) **System Defined Exceptions** : These exceptions are predefined in PL/SQL which get raised WHEN certain database rule is violated. For example - Consider following Student table

RollNo	Name	Marks
101	Shilpa	55
102	Trupti	66
103	Pradnya	43
104	Sharda	42
105	Vijaya	45

We can write the exception as follows -

```

DECLARE
s_roll customers.RollNo%type := 100;
s_name customers.Name%type;
s_marks customers.Marks%type;
BEGIN
SELECT Name INTO s_name
FROM student
WHERE RollNo = s_RollNo;
DBMS_OUTPUT.PUT_LINE ('Name: ' || s_name);
EXCEPTION
WHEN no_data_found THEN
dbms_output.put_line('Student is not present!');
WHEN others THEN
dbms_output.put_line('Error!!!');
END;
/

```

After execution of above PL/SQL program we get following output
Student is not present!

Explanation : The student with roll number 100 is not present in the Student table, hence is the output.

Here NO_DATA_FOUND is a pre-defined exception which is raised when a SELECT INTO statement returns no rows.

2) User defined Exception : PL/SQL facilitates their users to define their own exceptions according to the need of the program. A user-defined exception can be raised explicitly, using either a RAISE statement or the procedure DBMS_STANDARD.RAISE_APPLICATION_ERROR.

Syntax

```
DECLARE  
my-exception EXCEPTION;
```

Example

```
DECLARE  
s_roll customers.RollNo%type := 100;  
s_name customers.Name%type;  
s_marks customers.Marks%type;  
my_exception Exception;  
BEGIN  
IF s_roll <= 0 THEN  
    RAISE my_exception;  
ELSE  
    SELECT Name INTO s_name  
FROM student  
    WHERE RollNo = s_RollNo;  
    DBMS_OUTPUT.PUT_LINE ('Name: ' || s_name);  
END IF;  
EXCEPTION  
WHEN my_exception THEN  
    dbms_output.put_line('Roll number must be greater than zero');  
WHEN no_data_found THEN  
    dbms_output.put_line('Student is not present!');  
WHEN others THEN  
    dbms_output.put_line('Error!!!');  
END;  
/
```

Example 3.16.1 Write a PL/SQL block using user defined exception for following requirements :

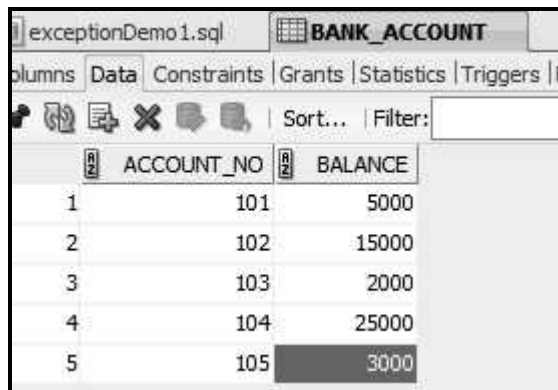
The Bank_Account table records the current balance for an account, which is updated whenever, any deposits or withdrawals takes place. If the withdrawal attempted is more than the current balance held in the account. The user defined exception is raised, displaying an appropriate message.

Note : Assume table :- Bank_Account (Account_No, Balance)

SPPU : Dec.-18, End Sem, Marks 5

Solution :

Step 1 : Table named **Bank_Account** is created as follows -



ACCOUNT_NO	BALANCE
1	101
2	102
3	103
4	104
5	105

Step 2 : Now create PL/SQL Program as follows

```

SET SERVEROUTPUT ON;
DECLARE
temp_account_no NUMBER(5);
temp_balance NUMBER(10);
withdraw_amount NUMBER(10) := 0;
Insufficient_balance EXCEPTION;
BEGIN

temp_account_no := &temp_account_no;
withdraw_amount := &withdraw_amount;
SELECT balance INTO temp_balance FROM bank_account WHERE account_no =
temp_account_no;
DBMS_OUTPUT.put_line('Account No = ' || temp_account_no || ' Balanace = ' || temp_balance);
IF(temp_balance < withdraw_amount) THEN
RAISE Insufficient_balance;
END IF;
temp_balance := temp_balance - withdraw_amount;
DBMS_OUTPUT.put_line('Current Balance is: ' || temp_balance);

EXCEPTION WHEN Insufficient_balance THEN
DBMS_OUTPUT.put_line('Insufficient Balance!!!');
END;
/

```

Output(Run 1)

```
anonymous block completed
Account No = 103 Balance = 2000
Insufficient Balance!!!
```

Output(Run 2)

```
anonymous block completed
Account No = 104 Balance = 25000
Current Balance is: 20000
```

Example 3.16.2 Write a PL/SQL block for following requirement and handle the exceptions.

Roll no. of student will be entered by user. Attendance of roll no. entered by user will be checked in student table. If attendance is less than 75 % then display the message "Term not granted" and set the status in student table as "D". Otherwise display message "Term granted" and set the status in student table as "ND".

SPPU : May-19, End Sem, Marks 5

Solution :

```
SET SERVEROUTPUT ON;
DECLARE
temp_rollno NUMBER(3);
temp_att NUMBER(5);
total_days NUMBER(5) := 200;
attend_ex EXCEPTION;
BEGIN
temp_rollno := &temp_rollno;
SELECT att INTO temp_att FROM stud_att WHERE roll_no = temp_rollno;
DBMS_OUTPUT.put_line('Roll No = ' || temp_rollno || ' Attendance = ' || temp_att || 'days outof
200');
IF(temp_att < (total_days * 0.75)) THEN
RAISE attend_ex;
ELSE
DBMS_OUTPUT.put_line('Term Granted!!!');
UPDATE stud_att
SET status = 'ND'
WHERE roll_no = temp_rollno;
END IF;

EXCEPTION WHEN attend_ex THEN
DBMS_OUTPUT.put_line('Term not Granted!!!');
UPDATE stud_att
SET status = 'D'
WHERE roll_no = temp_rollno;
END;
/
```

Output

On subsequent execution of the above PL/SQL code following output can be obtained.

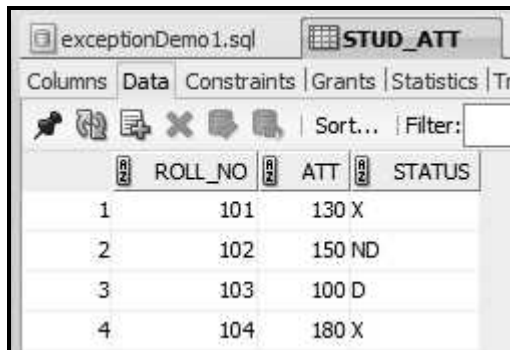

```

anonymous block completed
Roll No = 103 Attendance = 100days outof 200
Term not Granted!!!

anonymous block completed
Roll No = 102 Attendance = 150days outof 200
Term Granted!!!

```

We can verify the above execution by opening the `stud_att` table. Note that the status is updated as per the required condition.



	ROLL_NO	ATT	STATUS
1	101	130	X
2	102	150	ND
3	103	100	D
4	104	180	X

Example 3.16.3 Write PL/SQL code block that raise a user defined exception when business rule is violated. Business rule for client - master table specifies when the value of bal-due field is less than 0 handle the exception.

SPPU : May-19, End Sem, Marks 5

Solution :

```

SET SERVEROUTPUT ON;
DECLARE
temp_client_id NUMBER(5);
temp_balance NUMBER(10);
Insufficient_balance EXCEPTION;
BEGIN
SELECT bal_due INTO temp_balance FROM client_master WHERE bal_due < 0;
IF(temp_balance<0) THEN
RAISE Insufficient_balance;
END IF;
DBMS_OUTPUT.put_line('Success');

EXCEPTION WHEN Insufficient_balance THEN
DBMS_OUTPUT.put_line('Business Rule Violated!!!');
END;
/

```

Unit - II
Multiple Choice Questions

- Q.1** What is the full form of SQL ?
- a Standard Query Language b Structured Query Language
 c Simple Query Language d Specification Query Language
- Q.2** The SELECT query belongs to following category of SQL Command :
- a DDL b DML
 c DCL d None of these
- Q.3** The database language used for defining the schema of the database is called _____.
- a DDL b DML
 c DCL d None of these
- Q.4** DDL stands for _____.
- a Data Decoration Language b Data Definition Language
 c Dual Data Language d None of these
- Q.5** Which of the following is not a valid SQL type ?
- a Decimal b Float
 c Numeric d Character
- Q.6** Which operator tests column for the absence of data ?
- a EXIST operator b IS NULL operator
 c NOT operator d None of these
- Q.7** In SQL following command is used to change table's storage characteristics ?
- a MODIFY b CHANGE
 c ALTER d All of these
- Q.8** Which operator is used to compare a value to specified list of values ?
- a ANY b BETWEEN
 c ALL d IN
- Q.9** Which SQL keyword is used to retrieve maximum value ?
- a MOST b MAX
 c TOP d UPPER

Q.10 In SQL which command is used to SELECT only one copy of each set of duplicable rows

- _____.
- a SELECT DISTINCT b SELECT UNIQUE
 c SELECT DEFINITE d None of these

Q.11 The FROM clause is used to specify _____.

- a range for search condition b actual search condition
 c table from which data is searched. d all of the above

Q.12 Which of the following SQL command is used to retrieve data ?

- a DELETE b INSERT
 c RETRIVE d SELECT

Q.13 Which of the following command is used to add data to database table ?

- a Add b Insert
 c Append d All of the above

Q.14 Which of the following is an aggregate function ?

- a LEFT b RIGHT
 c MAX d JOIN

Q.15 In SQL, which command is used to issue multiple CREATE TABLE, CREATE VIEW and GRANT statements in a single transaction ?

- a CREATE PACKAGE b CREATE SCHEMA
 c CREATE CLUSTER d All of the above

Q.16 The intersect operation _____.

- a automatically eliminates duplicates
 b automatically eliminates duplicates, if we provide all clause with intersect
 c never eliminates duplicates
 d none of these

Q.17 Which of the following join is also called as an inner join ?

- a Self-join b Non Equi-join
 c Equi-join d None of these

Q.18 The virtual table created as a result of SELECT statement is called _____.

- a view b synonym
 c sequence d transaction

Q.19 Which of the following aggregate functions ignore NULL values ?

a MAX

b COUNT

c SUM

d All of the above

Q.20 Following keyword is used with wildcards ____.

a IN

b BETWEEN

c LIKE

d UNIQUE

Q.21 What is the meaning of LIKE '%0%0%' ?

a begins with two 0's

b ends with two 0's

c two 0's at any position

d none of these

Answers Keys for Multiple Choice Questions :

Q.1	b	Q.2	b	Q.3	a	Q.4	b
Q.5	a	Q.6	b	Q.7	c	Q.8	d
Q.9	b	Q.10	a	Q.11	c	Q.12	d
Q.13	b	Q.14	c	Q.15	b	Q.16	a
Q.17	c	Q.18	a	Q.19	d	Q.20	c
Q.21	c						



4

Relational Database Design

Syllabus

Relational Model : Basic concepts, Attributes and Domains, CODD's Rules. **Relational Integrity** : Domain, Referential Integrities, Enterprise Constraints. **Database Design** : Features of Good Relational Designs, Normalization, Atomic Domains and First Normal Form, Decomposition using Functional Dependencies, Algorithms for Decomposition, 2NF, 3NF, BCNF.

Contents

Part I : Relational Model

4.1	Basic Concepts		
4.2	Attributes and Domains		
4.3	CODD's Rules Aug.-17	
	 Oct.-19 Marks 5

Part II : Relational Integrity

4.4	Keys		
4.5	Constraints May-18, Dec.-19 Marks 5
4.6	Enterprise Constraints		

Part III : Database Design

4.7	Features of Good Relational Designs Dec.-18,	
	 Aug.-17 Marks 5
4.8	Data Redundancy and Update Anomalies		
4.9	Normalization Dec.-1 Marks 5
4.10	Atomic Domains and First Normal Form		
4.11	Decomposition using Functional Dependencies Oct.-19	
	 May-18 Marks 5
4.12	Equivalence and Minimal Cover		
4.13	Algorithms for Decomposition		
4.14	Lossless Join May-18 Marks 6

4.15 *Dependency Preservation*

4.16 *Second Normal Form (2NF)*..... **Oct.-19**, *Marks 7*

4.17 *Third Normal Form (3NF)*..... **Oct.-18, 19,**

..... **Aug.-17,**

..... **Dec.-18**, *Marks 7*

4.18 *BCNF*

..... **May-18,**

..... **Dec.-18,19**,..... *Marks 5*

Multiple Choice Questions

Part I : Relational Model

4.1 Basic Concepts

- Relation database is a **collection of tables** having unique names.
- For example - Consider the example of **Student table** in which the information about the student is stored.

RollNo	Name	Phone
001	AAA	1111111111
002	BBB	2222222222
003	CCC	3333333333

Fig. 4.1.1 Student table

The above table consists of three column headers **RollNo**, **Name** and **Phone**. Each row of the table indicates the information of each student by means of his Roll Number, Name and Phone number.

Similarly consider another table named **Course** as follows –

CourseID	CourseName	Credits
101	Mechanical	4
102	Computer Science	6
103	Electrical	5
104	Civil	3

Fig. 4.1.2 Course table

Clearly, in above table the columns are **CourseID**, **CourseName** and **Credits**. The **CourseID 101** is associated with the course named **Mechanical** and associated with the course of mechanical there are **4 credit points**. Thus the relation is represented by the table in the relation model. Similarly we can establish the relationship among the two tables by defining the third table. For example – Consider the table **Admission** as

RollNo	CourseID
001	102
002	104
003	101

Fig. 4.1.3 Admission table

From this third table we can easily find out that the course to which the **RollNo 001** is admitted is **computer Science**.

4.2 Attributes and Domains

There are some commonly used terms in Relational Model and those are -

Table or relation : In relational model, table is a collection of data items arranged in rows and columns. The table cannot have duplicate data or rows. Below is an example of student table

Roll No	Name	Marks	Phone
001	AAA	88	1111111111
002	BBB	83	2222222222
003	CCC	98	3333333333
004	DDD	67	4444444444

Tuple or record or row : The single entry in the table is called tuple. The tuple represents a set of related data. In above **Student** table there are four tuples. One of the tuple can be represented as

001	AAA	88	1111111111
-----	-----	----	------------

Attribute or columns : It is a part of table that contains several records. Each record can be broken down into several small parts of data known as attributes. For example the above table consists of four attributes such as **RollNo,Name,Marks** and **Phone**.

Relation schema : A relation schema describes the structure of the relation, with the name of the relation (i.e. name of table), its attributes and their names and type.

Relation Instance : It refers to specific instance of relation i.e. containing a specific set of rows. For example – the following is a relation instance – which contains the records with marks above 80.

RollNo	Name	Marks	Phone
001	AAA	88	1111111111
002	BBB	83	2222222222
003	CCC	98	3333333333

Domain : For each attribute of relation, there is a set of permitted values called domain. For example – in above table, the domain of attribute **Marks** is set of all possible permitted marks of the students. Similarly the domain of **Name** attribute is all possible names of students.

That means Domain of Marks attribute is (88,83,98)

Atomic : The domain is atomic if elements of the domain are considered to be indivisible units. For example in above **Student** table, the attribute **Phone** is non-atomic.

NULL attribute : A null is a special symbol, independent of data type, which means either unknown or inapplicable. It does not mean zero or blank. For example - Consider a salary table that contains NULL

Emp#	Job Name	Salary	Commission
E10	Sales	12500	32090
E11	Null	25000	8000
E12	Sales	44000	0
E13	Sales	44000	Null

Degree : It is nothing but total number of columns present in the relational database. In given Student table –

Roll No	Name	Marks	Phone
001	AAA	88	1111111111
002	BBB	83	2222222222
003	CCC	98	3333333333

The degree is 4.

Cardinality : It is total number of tuples present in the relational database. In above given table the cardinality is 3

Example 4.2.1 Find out following for given Staff table

i) No of Columns

ii) No of tuples

iii) Different attributes

iv) Degree

v) Cardinality

StaffID	Name	Sex	Designation	Salary	DOJ
S001	John	M	Manager	50000	1 Oct. 2012
S002	Ram	M	Executive	20000	20 Jan. 2015
S003	Meena	F	Supervisor	40000	12 Aug. 2011

Solution :

i) No of Columns = 6

ii) No of Tuples = 3

iii) Different attributes are StaffID, Name, Sex, Designation, Salary, DOJ

iv) Degree = Total number of columns = 6

v) Cardinality = Total number of rows = 3

4.3 CODD's Rules**SPPU : Aug.-17, Oct.-19, Marks 5**

Codd proposed 13 rules for relational database management system, which are popularly known as **Codd's 12 rule** : These rules are as follows –

Rule 0 : This rule states for a database to be relational, it must use its relational capabilities to manage the database.

Rule 1 : The Information rule - All information in an RDBMS is represented logically only by storing the values in tables.

Rule 2 : The Guaranteed Access rule - Each item of data in an RDBMS is guaranteed to be logically accessible by specifying the table name, primary key value, and column name.

Rule 3 : The Systematic Treatment of Null Values rule - Null values are supported in a fully relational DBMS for representing missing information or inapplicable information in a systematic way which is independent of the data type.

Rule 4 : The Dynamic Online Catalog Based on the Relational Model rule - Database dictionary which is called as catalog-is the structure description of the complete Database and it must be stored online. This Catalog must be governed by same rules as rest of the database. The same query language should be used on catalog as used to query database.

Rule 5 : The Comprehensive Data Sublanguage rule - At least one well structured, well-defined language must be there which can access all the data present in the database.

Rule 6 : The View Updating rule - All views of the data which are theoretically updatable must be updatable in practice by the DBMS.

Rule 7 : Relational level operation - The High-level Insert, Update, and Delete rule: There must be insert, delete and update operations at each level of relations.

Rule 8 : The Physical Data Independence rule - Physical storage should not matter the system. Whenever any changes are made in either storage representations or access methods then it should not affect the application.

Rule 9 : The Logical Data Independence rule - If any changes are made in table structure then the logical view of the user should not get affected. For Rule example - if a table is split into two tables internally, the view of the table to the user should be an entire table and not the split tables.

Rule 10 : The Integrity Independence rule - The Integrity constraints must be defined by the RDBMS stored in the system and it should not be enforced by the external application programs.

Rule 11 : The Distribution Independence rule - An RDBMS must have distribution independence. That means, even if database is scattered geographically, user should get a feel as if it is stored in one piece at one location.

Rule 12 : The Non-sub-version rule - If low-level language is allowed to access the system, then that low-level language must not be able to subvert or bypass the integrity rules which are expressed in a higher-level language.

Review Questions

1. One of the rule designed by Codd's for good relational database management system is integrity independence, which states that all integrity constraints can be independently modified without the need of any change in the application. Justify the significance of rule in relational database management system. **SPPU : Aug.-17, In Sem, Marks 5**
2. Twelve rules are proposed by codd, which according to him, a database must obey in order to be regarded as a true relational database. One of the rule is comprehensive data sub language rule. A database can only be accessed using a language having linear syntax that supports data definition, data manipulation and transaction management operations. Explain in brief above rule. Also state its significance. **SPPU : Oct.-19, In Sem, Marks 5**

Part II : Relational Integrity

4.4 Keys

Keys are used to specify the tuples distinctly in the given relation.

Various types of keys used in relational model are – Superkey, Candidate Keys, primary keys, foreign keys. Let us discuss them with suitable example

1) Super Key(SK) :

It is a set of one or more attributes within a table that can uniquely identify each record within a table. For example – Consider the Student table as follows –

Reg No.	Roll No	Phone	Name	Marks
R101	001	1111111111	AAA	88
R102	002	2222222222	BBB	83
R103	003	3333333333	CCC	98
R104	004	4444444444	DDD	67

Fig. 4.4.1 Student

The superkey can be represented as follows

RegNo.	RollNo	Phone	Name	Marks
R101	001	111111111	AAA	88
R102	002	222222222	BBB	83
R103	003	333333333	CCC	98
R104	004	444444444	DDD	67

Annotations in the diagram:
 - A solid oval labeled "Superkey" encloses the first two columns (RegNo., RollNo).
 - A solid oval labeled "(RollNo, Phone, Name) Superkey" encloses the second, third, and fourth columns.
 - A dashed oval labeled "(Name, Marks) Not a Superkey" encloses the fourth and fifth columns.

Clearly using the (RegNo) and (RollNo,Phone,Name) we can identify the records uniquely but (Name, Marks) of two students can be same, hence this combination not necessarily help in identifying the record uniquely.

2) Candidate Key(CK) :

The candidate key is a subset of superset. In other words candidate key is a single attribute or least or minimal combination of attributes that uniquely identify each record in the table. For example - in above given **Student** table, the candidate key is RegNo, (RollNo,Phone). The candidate key can be

RegNo.	RollNo	Phone	Name	Marks
R101	001	111111111	AAA	88
R102	002	222222222	BBB	83
R103	003	333333333	CCC	98
R104	004	444444444	DDD	67

Annotations in the diagram:
 - A solid oval labeled "Candidate key" encloses the first column (RegNo.).
 - A solid oval labeled "Candidate key" encloses the second and third columns (RollNo, Phone).

Thus every candidate key is a superkey but every superkey is not a candidate key.

3) Primary Key(PK) :

The primary key is a candidate key chosen by the database designer to identify the tuple in the relation uniquely. For example – Consider the following representation of primary key in the student table

Primary key

RegNo.	RollNo	Phone	Name	Marks
R101	001	111111111	AAA	88
R102	002	222222222	BBB	83
R103	003	333333333	CCC	98
R104	004	444444444	DDD	67

Other than the above mentioned primary key, various possible primary keys can be **(RollNo)**, **(RollNo,Name)**, **(RollNo, Phone)**

The relation among super key, candidate key and primary can be denoted by

$$\text{Candidate Key} = \text{Super Key} - \text{Primary Key}$$

Rules for Primary Key

- i) The primary key may have one or more attributes.
- ii) There is **only one primary key** in the relation.
- iii) The value of primary key attribute can not be NULL.
- iv) The value of primary key attribute does not get changed.

4) Alternate key :

The alternate key is a candidate key which is not chosen by the database designer to uniquely identify the tuples. For example –

Primary key

Alternate key

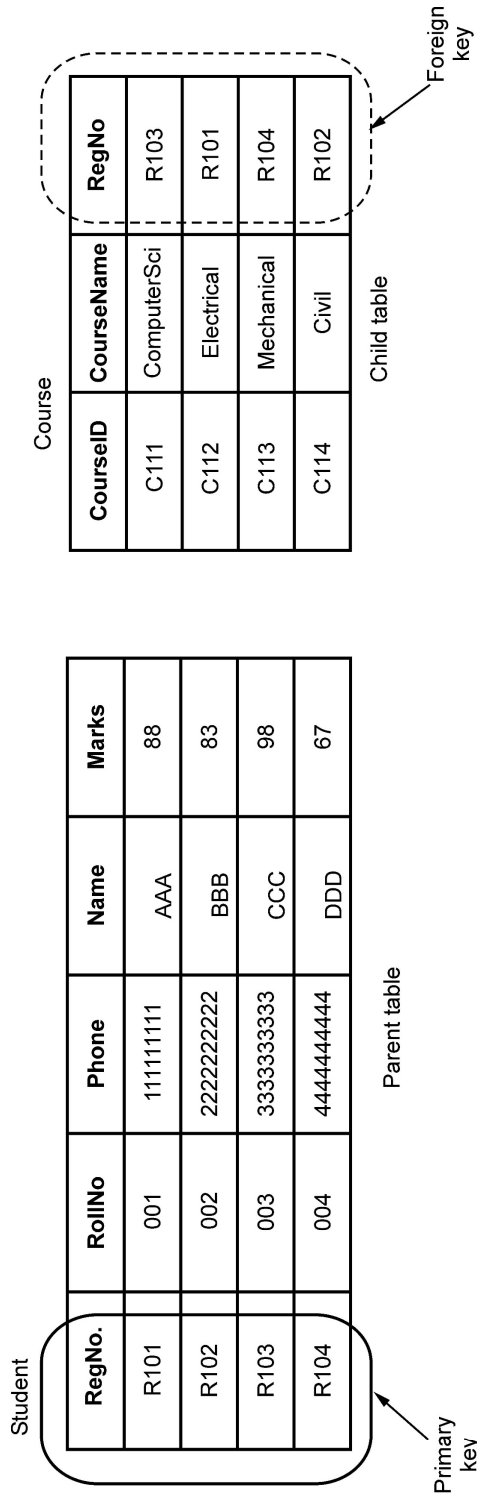
RegNo.	RollNo	Phone	Name	Marks
R101	001	111111111	AAA	88
R102	002	222222222	BBB	83
R103	003	333333333	CCC	98
R104	004	444444444	DDD	67

5) Foreign key :

Foreign key is a single attribute or collection of attributes in one table that refers to the primary key of other table.

- Thus foreign keys refer to primary key.
- The table containing the primary key is called **parent table** and the table containing foreign key is called **child table**.

- Example -



From above example, we can see that two tables are linked. For instance we could easily find out that the 'Student CCC has opted for ComputerSci course'

4.5 Constraints

SPPU : May-18, Dec.-19, Marks 5

Constraints mean some rules or restrictions that are set on the database.

There are three main types of constraints.

1. Domain Constraint
2. Key Constraint or NULL Constraint
3. Integrity Constraint
 - i) Entity Integrity Constraint
 - ii) Referential Integrity Constraint

1. Domain Constraint

- Domain constraint defines the domain or set of values for an attribute.
- The data type of domain includes string, character, integer, time, date, currency, etc. The value of the attribute must be available in the corresponding domain.
- For example - Consider the Student table as follows.

Roll No	Name	Marks	Phone
001	AAA	88	111111111
001	BBB	83	222222222
003	1234	98	333333333
004	DDD	67	444444444

Name cannot be numeric value.
It must be a string.
Hence this is not allowed !!!

The above relation does not satisfy the domain constraint.

2. Key Constraint or Null Constraint

- Keys are used to identify particular record from the table. Primary key is normally used to identify the record uniquely.

- Hence the key constraint can be stated as -
 - All values of primary key must be unique.
 - The value of primary key must not be NULL.
- For example - Consider the Student table as follows. For this relation, the **Roll No** is a primary key. It is expected to find the desired record using this primary key.

	Roll No	Name	Marks	Phone
Duplicate values ? This is not allowed !!!	001	AAA	88	111111111
	001	BBB	83	222222222
	003	CCC	98	333333333
	004	DDD	67	444444444

The above relation does not satisfy key constraint as the primary key is not having unique value.

3. Integrity Constraint

- Integrity constraints are rules that are to be applied on database columns to ensure the validity of data.
- For example -
 - i) The Employee ID and Department ID must consist of two digits.
 - ii) Every Employee ID must start with letter.

i) Entity Integrity Constraint

- **This rule states that** "In the relations, the value of attribute of primary key can not be null".
- The NULL represents a value for an attribute that is currently unknown or is not applicable for this tuple. The Nulls are always **to deal with incomplete or exceptional data**.
- The primary key value helps in uniquely identifying every row in the table. Thus if the users of the database want to retrieve any row from the table or perform any action on that table, they must know the value of the key for that row. Hence it is necessary that the primary key should not have the NULL value.

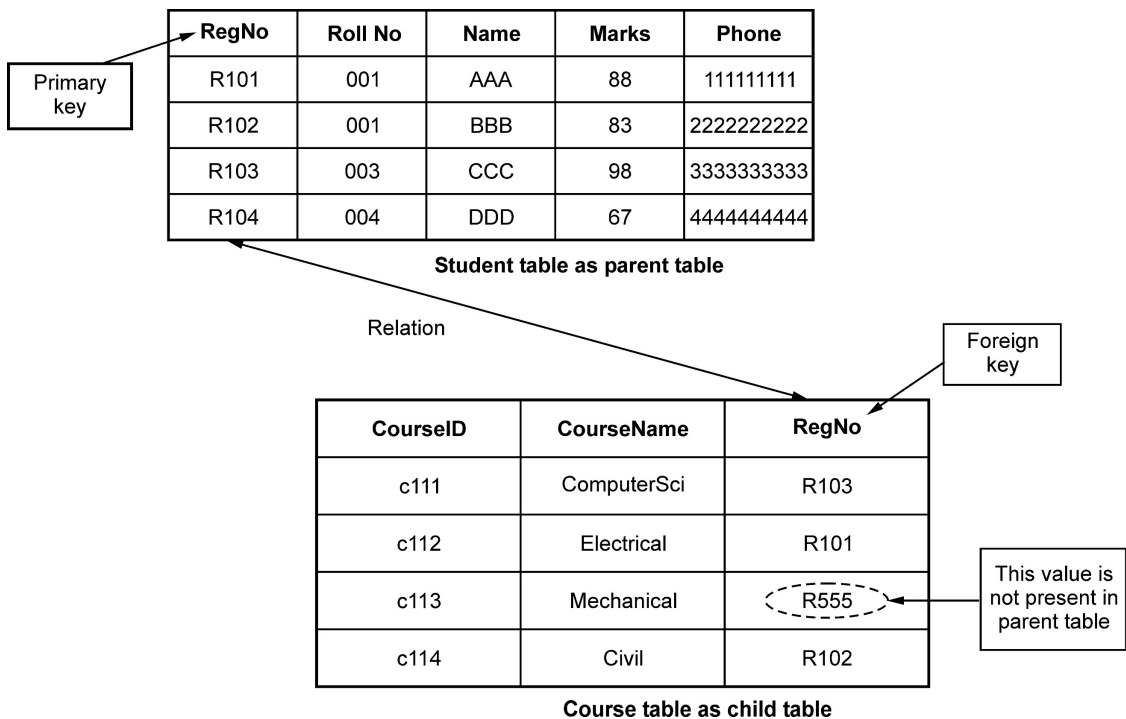
- For example -

Roll No	Name	Marks	Phone
001	AAA	88	111111111
001	BBB	83	222222222
003	CCC	98	333333333
	DDD	67	444444444

NULL is not allowed !!!

ii) Referential Integrity Constraint

- In relationships, data is linked between two or more tables.
- This is achieved by having the foreign key (in the associated table) reference a primary key value (in the primary - or parent - table). Because of this, we need to ensure that data on both sides of the relationship remain intact.
- **The referential integrity rule states that “whenever a foreign key value is used it must reference a valid, existing primary key in the parent table”.**
- For example - Consider two tables



In above relation, the registration no. R555 is not existing still if it is present in the course table, then we say that it is not following referential integrity constraint.

Review Question

1. Explain the concepts of referential integrity constraint and entity integrity constraint with example.

SPPU : May-18, Dec.-19, End Sem, Marks 5

4.6 Enterprise Constraints

Enterprise constraints are also called as semantic constraints.

The enterprise constraints are basically the additional rules specified by users or database administrators. These constraints are normally based on multiple tables.

Examples of enterprise constraints are –

- 1) The salary of teacher should not exceed the salary of Principal.
- 2) A Student can not opt for more than two courses at a time.
- 3) A class can have maximum 50 students.

Part III : Database Design

4.7 Features of Good Relational Designs

SPPU : Dec.-18, Aug.-17, Marks 5

There are two primary goals of relational database design –

- i) To generate a set of relation schemas that allows us to store information without unnecessary redundancy and
- ii) To allow us to retrieve information easily.

Example 4.7.1 Explain what is meant by repetition of information and inability to represent information. Explain why each of these properties may indicate a bad relational database design.

SPPU ; Dec.-18, End Sem, Marks 5

Solution : Repetition of information and inability to represent the required information are considered to be bad features of relational design. Consider following schema

EmpID	EName	Salary	DeptID	DeptName	DeptLoc
1	AAA	10000	101	XYZ	Pune
2	BBB	20000	101	XYZ	Pune
3	CCC	30000	101	XYZ	Pune
4	DDD	40000	102	PQR	Mumbai

Now if we want to insert a record 5,EEE,50000 for the DeptID 101, then again there will be repetition of information about DeptID, DeptName and DeptLoc i.e. (101,XYZ,Pune). That means if we want to perform some operation (insertion, updation, deletion) on the relational schema then it should not cause repetition of information.

The above scenario indicates bad database design.

Inability to represent information is a condition where a relationship exists among only a proper subset of the attributes in a relation. This is bad relational database design because all the unrelated attributes must be filled with null values otherwise a tuple without the unrelated information cannot be inserted into the relation.

Review Question

1. Explain different features of good relational database design.

SPPU : Aug.-17, In Sem, Marks 5

4.8 Data Redundancy and Update Anomalies

Definition : Data redundancy is a condition created in database in which same piece of data is held at two different places.

Redundancy is at the root of several problems associated with relational schemas.

Problems caused by redundancy : Following problems can be caused by redundancy-

- i) **Redundant storage :** Some information is stored repeatedly.
- ii) **Update anomalies :** If one copy of such repeated data is updated then inconsistency is created unless all other copies are similarly updated.
- iii) **Insertion anomalies :** Due to insertion of new record repeated information get added to the relation schema.
- iv) **Deletion anomalies :** Due to deletion of particular record some other important information associated with the deleted record get deleted and thus we may lose some other important information from the schema.

Example : Following example illustrates the above discussed anomalies or redundancy problems.

Consider following Schema in which all possible information about Employee is stored.

EmpID	EName	Salary	DeptID	DeptName	DeptLoc
1	AAA	10000	101	XYZ	Pune
2	BBB	20000	101	XYZ	Pune
3	CCC	30000	101	XYZ	Pune
4	DDD	40000	102	PQR	Mumbai

Redundancy!!!

- 1) **Redundant storage** : Note that the information about DeptID, DeptName and DeptLoc is repeated.
- 2) **Update anomalies** : In above table if we change **DeptLoc** of Pune to Chennai, then it will result **inconsistency** as for DeptID 101 the DeptLoc is Pune. Or otherwise, we need to **update multiple copies** of DeptLoc from Pune to Chennai. Hence this is an update anomaly.
- 3) **Insertion anomalies** : For above table if we want to add new tuple say (5, EEE,50000) for **DeptID 101** then it will cause repeated information of (101, XYZ,Pune) will occur.
- 4) **Deletion anomalies** : For above table, if we delete a record for **EmpID 4**, then automatically information about the **DeptID 102, DeptName PQR** and **DeptLoc Mumbai** will get deleted and one may not be aware about **DeptID 102**. This causes deletion anomaly.

4.9 Normalization

SPPU : Dec.-17, Marks 5

- Normalization is the process of reorganizing data in a database so that it meets **two basic requirements** :
 - 1) There is **no redundancy** of data (all data is stored in only one place) and
 - 2) **Data dependencies** are logical (all related data items are stored together)

Need for normalization

- 1) **It eliminates redundant data.**
- 2) **It reduces chances of data error.**
- 3) The normalization is important because it allows database to take up **less disk space.**

- 4) It also help in increasing the **performance**.
- 5) It improves the data integrity and consistency.

Review Question

1. What is normalization ? What is the need of normalized database ?

SPPU : Dec.-17, End Sem, Marks 5

4.10 Atomic Domains and First Normal Form

By atomic value, we mean that each value in the domain is indivisible.

The first normal form rule defines that all the attributes in a relation must have atomic domains. The values in an atomic domain are indivisible units.

The table is said to be in 1NF if it follows following rules -

- i) It should only have single (atomic) valued attributes/columns.
- ii) Values stored in a column should be of the same domain.
- iii) All the columns in a table should have unique names.
- iv) And the order in which data is stored, does not matter.

Consider following student table

Student

sid	sname	Phone
1	AAA	11111 22222
2	BBB	33333
3	CCC	44444 55555

As there are multiple values of phone number for sid 1 and 3, the above table is not in 1NF. We can make it in 1NF. The conversion is as follows -

sid	sname	Phone
1	AAA	11111
1	AAA	22222
2	BBB	33333
3	CCC	44444
3	CCC	55555

Review Question

1. Suggest and explain three different techniques to achieve 1NF using suitable example.

4.11 Decomposition using Functional Dependencies**SPPU : Oct.-19, May-18, Marks 5**

Definition : A functional dependency $A \rightarrow B$ in a relation holds if two tuples having same value of attribute A also have the same value for attribute B. It is denoted by $A \rightarrow B$ where A is called determinant and B is called dependent.

For example - Consider Student table as follows -

Roll	Name	City
1	AAA	Mumbai
2	BBB	Pune
3	CCC	Gandhinagar

Here

Roll \rightarrow Name **hold**

But

Name \rightarrow City **does not hold**

- In above table, student roll number is unique hence each student's name and city can be uniquely identified using his roll number.
- But using name we cannot uniquely identify his/her city because there can be same names of the students. Similarly using city name we can not identify the student uniquely. As in the same city may belong to multiple students.

Another example -

Consider a relation in which the roll of the student and his/her name is stored as follows :

R	N
1	AAA
2	BBB
3	CCC
4	DDD
5	EEE

Fig. 4.11.1 : Table which holds functional dependency i.e. $R \rightarrow N$

Here, $R \rightarrow N$ is true. That means the functional dependency holds true here. Because for every assigned RollNumber of student there will be unique name. For instance : The name of the Student whose RollNo is 1 is AAA. But if we get two different names for the same roll number then that means the table does not hold the functional dependency. Following is such table -

R	N
1	AAA
2	BBB
3	CCC
1	XXX
2	YYY

Fig. 4.11.2 : Table which does not hold functional dependency

In above table for RollNumber 1 we are getting two different names - "AAA" and "XXX". Hence here it does not hold the functional dependency.

Trivial FD : The functional dependency $A \rightarrow B$ is trivial if B is a subset of A.

For example $(A,B) \rightarrow A$

Non Trivial FD : The functional dependency $A \rightarrow B$ is non trivial if B is not a subset of A.

For example $\{A,B\} \rightarrow C$

Example 4.11.1 For the given below relation $R(A,B,C,D,E)$ and its instance, check whether FDs given hold or not. Give reasons.

i) $A \rightarrow B$ ii) $B \rightarrow C$ iii) $D \rightarrow E$ iv) $CD \rightarrow E$

A	B	C	D	E
a1	b1	c1	d1	e1
a1	b2	c1	d1	e1
a2	b2	c1	d2	e3
a2	b3	c3	d2	e2

Solution : Association among attributes is known as Functional Dependencies (FD). AFD $X \rightarrow Y$ require that the value of X uniquely determines the value of Y where X and Y are set of attributes.

For example,

Roll_No \rightarrow Name : the value of Roll_No uniquely determines the Name.

Now from, the given relation and its instance -

- i) The FD $A \rightarrow B$ does not hold because – a1 has two different values b1 and b2. Similarly a2 has two different values and those are b2 and b3.
- ii) The FD $B \rightarrow C$ holds true.
- iii) $D \rightarrow E$ does not hold true because d2 gives two different values e3 and e2.
- iv) $CD \rightarrow E$ hold true as (c1,d1) gives e1 , (c1,d2) gives e3 and (c3,d2) gives e2. All are uniquely identified.

4.11.1 Inference Rules

The closure set is a set of all functional dependencies implied by a given set F. It is denoted by F^+

The closure set of functional dependency can be computed using basic three rules which are also called as **Armstrong's Axioms**.

These are as follows -

- i) **Reflexivity** : If $X \supseteq Y$, then $X \rightarrow Y$
- ii) **Augmentation** : If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z
- iii) **Transitivity** : If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

In addition to above axioms some additional rules for computing closure set of functional dependency are as follows -

- **Union** : If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$
- **Decomposition** : If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$

Let us understand how to apply Armstrong's axioms for finding the closure of set of functional dependencies -

Example 4.11.2 Given FD's for relation $R\{A,B,C,D,E,F\}$, Find closure of FD set by applying Armstrong's Axioms.

$A \rightarrow B, A \rightarrow C, CD \rightarrow E, CD \rightarrow F, B \rightarrow E$

Solution :

Step 1 : $A \rightarrow$ gives A attribute itself by reflexivity. It is called trivial production.

$A \rightarrow B$ and $A \rightarrow C$, Hence by union rule $A \rightarrow BC$

$A \rightarrow B$ and $B \rightarrow E$, Hence by transitivity rule $A \rightarrow E$

$\therefore (A)^+ = \{A, B, C, E\}$

Step 2 : $B \rightarrow$ gives B itself

And $B \rightarrow E$

$\therefore (B)^+ = \{B, E\}$

Step 3 : $CD \rightarrow$ gives CD itself. It is trivial.

$CD \rightarrow E$, $CD \rightarrow F$, Hence by union rule $CD \rightarrow EF$

$\therefore (CD)^+ = \{C, D, E, F\}$

So by omitting trivial productions, we get

$\therefore F^+ = \{A \rightarrow BC, A \rightarrow E, B \rightarrow E, CD \rightarrow EF\}$

Example 4.11.3 Compute the closure of the following set F of functional dependencies for relational schema $R = (A, B, C, D, E)$ $A \rightarrow BC$, $CD \rightarrow E$, $B \rightarrow D$, $E \rightarrow A$

Solution : The closure of F is denoted by F^+ and it can be computed in following steps

Step 1 : As $A \rightarrow BC$ is given we get

$A \rightarrow B$ and $A \rightarrow C$

By decomposition rule

Step 2 : As

$A \rightarrow B$

(Refer step 1)

$B \rightarrow D$

(given)

$A \rightarrow D$

(transitivity rule)

Step 3 :

$A \rightarrow CD$ because $A \rightarrow C$ and $A \rightarrow D$

(From step 1 and Step 2 applying union rule)

Step 4 :

$CD \rightarrow E$

(given)

$A \rightarrow E$

(transitive rule as $A \rightarrow CD$, $CD \rightarrow E$
hence $A \rightarrow E$)

Step 5 :

Since $A \rightarrow A$,
 $A \rightarrow ABCDE$

we have (reflexive)
 from the above steps (union)

Step 6 :

Since $E \rightarrow A$, $E \rightarrow ABCDE$

(transitive)

Step 7 :

Since $CD \rightarrow E$, $CD \rightarrow ABCDE$

(transitive)

Step 8 :

Since $B \rightarrow D$ and $BC \rightarrow CD$, $BC \rightarrow ABCDE$ (augmentative, transitive)

Step 9 :

Also, $C \rightarrow C$, $D \rightarrow D$, $BD \rightarrow D$

Thus any functional dependency with A , E , BC , or CD on the left hand side of the arrow is in F^+

Example 4.11.4 Give Armstrong's axioms and using it find the closure of following FD set.

$A \rightarrow B$, $AB \rightarrow C$, $D \rightarrow AC$, $D \rightarrow E$

Solution :

Step 1 : Consider the rules $D \rightarrow AC$ and $D \rightarrow E$

$\therefore D \rightarrow ACE$

(Union rule)

Step 2 : Consider $AB \rightarrow C$ then we get

$\therefore A \rightarrow C$ and $B \rightarrow C$

(decomposition rule)

Thus $F^+ = \{A \rightarrow C, B \rightarrow C, D \rightarrow ACE\}$

Example 4.11.5 $R = \{A, B, C, D, E, F\}$ and FDs are $A \rightarrow BC$ $E \rightarrow CF$ $B \rightarrow E$ $CD \rightarrow EF$ compute closure of $\{A, B\}^+$

Solution :

Step 1 : $A \rightarrow BC$

$\therefore A \rightarrow B$ and $A \rightarrow C$

(decomposition)

So we add A, B, C in closure set

Step 2 : $E \rightarrow CF$

$\therefore E \rightarrow C$ and $E \rightarrow F$ (decomposition)

Step 3 : $B \rightarrow E \therefore B \rightarrow C, F$

So we add E and F in closure set

Hence

$$\{A, B\}^+ = \{A, B, C, E, F\}$$

Example 4.11.6 Consider schema EMPLOYEE(E-ID, E-NAME, E-CITY, E-STATE) and

$FD = \{E-ID \rightarrow E-NAME, E-ID \rightarrow E-CITY, E-ID \rightarrow E-STATE, E-CITY \rightarrow E-STATE\}$

(1) Find attribute of closure for $(E-ID)^+$

(2) Find $(E-NAME)^+$

Solution :

1) Finding $(E-ID)^+$ means finding the closure. In this process we try to find out, all the attributes that can be derived from E-ID.

As $E-ID \rightarrow E-NAME, E-ID \rightarrow E-CITY, E-ID \rightarrow E-STATE$, We add E-NAME, E-ID, E-CITY, E-STATE to $(E-ID)^+$

As $E-ID \rightarrow E-CITY, E-CITY \rightarrow E-STATE$, we add E-STATE to $(E-ID)^+$

$\therefore (E-ID)^+ = \{E-ID, E-NAME, E-CITY, E-STATE\}$

2) E-NAME derives no rule. Hence $(E-NAME)^+ = \{E-NAME\}$

4.11.2 Keys and Functional Dependencies

- For a given relation $R = \{A_1, A_2, A_3, \dots, A_n\}$ K is a key of R then if closure $(K)^+ = \{A_1, A_2, \dots, A_n\}$ and no subset of K i.e. X such that $(X)^+ = \{A_1, A_2, \dots, A_n\}$
- In other words there are two conditions -
 - 1) The $(K)^+$ contains all the attributes of relations R -
 - 2) All subset X of K, $(X)^+$ never contains all the attributes of R i.e. $(X)^+ \neq \{A_1, A_2, \dots, A_n\}$
- If only one subset of R satisfy above condition then it is known as **primary key**.
- If more than one subset of R satisfies above condition then all these subsets are recognized as **candidate keys**. In that case one of the candidate key is also considered as **primary key**.
- A superset of candidate key K is known as **superkey**.

Example 4.11.7 Give $R = \{A, B, C, G, H, I\}$. The following set F of functional dependencies holds

$A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H$

Compute AG^+ . Is AG candidate key?

Solution :

Step 1 : $A \rightarrow B, A \rightarrow C$, Hence add A, B, C to the set of AG^+ .

Step 2 : $A \rightarrow B, B \rightarrow H$, hence add H to the set of AG^+

Step 3 : $CG \rightarrow I$, hence add I to the set of AG^+ . Also add G to the set.

Thus $(AG)^+ = \{A, B, C, G, H, I\} = \text{Relation } R$

Hence $(AG)^+$ is a candidate key.

Example 4.11.8 Compute the closure of $R(A, B, C, D, E)$ with the following set of functional dependencies $A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A$

List the candidate keys of R

Solution :

Step 1 : $A \rightarrow BC$ hence add A, B, C to $(A)^+$

$A \rightarrow BC$ can be decomposed into $A \rightarrow B$ and $A \rightarrow C$. Also $B \rightarrow D$. Thus $A \rightarrow D$ is also true by transitivity rule.

Hence add D to $(A)^+$

$A \rightarrow C, A \rightarrow D \therefore$ By union rule $A \rightarrow CD$.

As $CD \rightarrow E$ add E to $(A)^+$

$\therefore (A)^+ = \{A, B, C, D, E\}$

Step 2 : Consider $(B)^+ = \{B, D\} \neq R$ hence it is not a candidate key

Step 3 : Consider $(BC)^+ = \{B, C, D, E, A\} = \{A, B, C, D, E\} = R$. Hence it is a candidate key

Step 4 : Consider $CD \rightarrow E, E \rightarrow A$, hence $(CD)^+ = \{A, B, C, D, E\}$.

Hence it is a candidate key

Step 5 : Consider $E \rightarrow A, A \rightarrow BC, B \rightarrow D, CD \rightarrow E$.

Hence $(E)^+ = \{A, B, C, D, E\}$ is a candidate key.

Thus we get the candidate keys as $\{A, BC, CD, E\}$

Example 4.11.9 Consider schema $R = (A,B,C,G,H,I)$ and the set F of functional dependencies

$\{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$. Use $(F)^+$ Prove $(AG)^+ \rightarrow I$

Solution :

Step 1 : As $A \rightarrow B$

$B \rightarrow H$

$\therefore A \rightarrow H$ (Transitivity rule)

Step 2 : $CG \rightarrow H$

$CG \rightarrow I$

$\therefore CG \rightarrow HI$ (Union rule)

Step 3 : $A \rightarrow C$

$CG \rightarrow I$

$AG \rightarrow I$ (Pseudo transitive rule)

Thus $AG \rightarrow I$ is proved

$(F)^+ = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H, A \rightarrow H, CG \rightarrow HI, AG \rightarrow I\}$

Review Questions

1. Explain in brief with suitable example full functional dependency and partial dependency.

SPPU : Oct.-19, In Sem, Marks 3

2. What is the impact of insert, update and delete anomaly on overall design of database ? How normalization is used to remove these anomalies ?

SPPU : May-18, End Sem, Marks 5

4.12 Equivalence and Minimal Cover

- A set of functional dependencies E is said to be covered by F if every FD in E is also in F^+ , i.e. every dependency in F can be inferred from E and vice versa.
- Two sets of FDs E and F are equivalent if $E^+ = F^+$.
- If E and F are equivalent only if both $E^+ = F^+$
 - 1) E covers F
 - 2) F covers E holds

Example 4.12.1 Here are two sets of FDs for $R(A,B,C,D,E)$. Are they equivalent ?

1) $A \rightarrow B$ 2) $A \rightarrow BC$

$AB \rightarrow C$ $D \rightarrow AE$

$D \rightarrow AC$

$D \rightarrow E$

Solution : Two sets of FDs are said to be equivalent if

Rule 1 : If $FD2 \supset FD1$. That means all FDs of $FD1$ can be derived from all the FDs of $FD2$.

Rule 2 : If $FD1 \supset FD2$. That means all FDs of $FD2$ can be derived from all the FDs of $FD1$.

Rule 3 : If both rule 1 and rule 2 are true then $FD1 = FD2$.

For given two sets

Step 1 : We will first check if all the FDs of $FD1$ are present in $FD2$

$A \rightarrow B$ is present in $FD1$.	$A \rightarrow BC$ is in $FD2$, that also means $A \rightarrow B$ and $A \rightarrow C$ by decomposition rule
Similarly $AB \rightarrow C$ is in $FD1$ $\therefore (A)^+ = \{A,B,C\}$	Hence $(A)^+ = \{A,B,C\}$
$\therefore (AB)^+ = \{A,B,C\}$	As $D \rightarrow AE$ then by decomposition rule,
$D \rightarrow AC$	$D \rightarrow A, D \rightarrow E$
i.e $D \rightarrow A, D \rightarrow C$ by decomposition rule.	As $A \rightarrow B$, then by transitivity rule $D \rightarrow A$, $A \rightarrow B, D \rightarrow B$
The $D \rightarrow A, A \rightarrow B$ and C	$\therefore (D)^+ = \{A,B,C,D,E\}$
$D \rightarrow A, D \rightarrow B, D \rightarrow C$ by transitivity rule $D \rightarrow E$ is given $\therefore (D)^+ = \{A,B,C,D,E\}$	

Step 2 : We will first check if all the FDs of $FD2$ are present in $FD1$

$\therefore (A)^+ = \{A,B,C\}$	$(A)^+ = \{A,B,C\}$
$\therefore (AC)^+ = \{A,B,C\}$	$\therefore (D)^+ = \{A,B,C,D,E\}$
$\therefore (D)^+ = \{A,B,C,D,E\}$	

Thus from Step 1 and Step 2, $FD2 \supset FD1$ and $FD1 \supset FD2$. Hence both the sets are equivalent.

Example 4.12.2 Consider two sets of functional dependency.

$F = \{A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H\}$ and $G = \{A \rightarrow CD, E \rightarrow AH\}$. Are they equivalent ?

Solution : We will find the FDs of both F and G and then check for $F \supseteq G$ and $G \supseteq F$.

Step 1 : For F

Using G functional dependencies -

$$(A)^+ = \{A,C,D\} \leftarrow \text{As } A \rightarrow CD \text{ is in G}$$

$$(AC)^+ = \{A,C,D\} \leftarrow \text{As } A \rightarrow CD \text{ is in G}$$

$$(E)^+ = \{A,C,D,E,H\} \leftarrow \text{As } E \rightarrow AH, A \rightarrow CD \text{ is in G}$$

Using F functional dependencies -

$$(A)^+ = \{A,C,D\} \leftarrow \text{As } A \rightarrow C \text{ and } AC \rightarrow D \text{ is in F}$$

$$(AC)^+ = \{A,C,D\} \leftarrow AC \rightarrow D \text{ is in F}$$

$$(E)^+ = \{A,C,D,E,H\} \leftarrow \text{As } E \rightarrow AD, E \rightarrow H \text{ and } A \rightarrow C \text{ is in F}$$

Step 2 : For G

Using F functional dependencies -

$$(A)^+ = \{A,C,D\} \leftarrow \text{As } A \rightarrow C \text{ and } AC \rightarrow D \text{ is in F}$$

$$(E)^+ = \{A,C,D,E,H\} \leftarrow \text{As } E \rightarrow AD, E \rightarrow H \text{ is in F}$$

Using G functional dependencies -

$$(A)^+ = \{A,C,D\} \leftarrow \text{As } A \rightarrow CD \text{ is in G}$$

$$(E)^+ = \{A,C,D,E,H\} \leftarrow \text{As } E \rightarrow AH \text{ and } A \rightarrow CD \text{ is in G}$$

Step 3 : From both these steps

$G \supseteq F$ and $F \supseteq G$

Hence F and G are equivalent.

Example 4.12.3 Given below are two sets of FDs for a relation $R(A,B,C,D,E)$, Are they equivalent ?

i) $A \rightarrow B, AB \rightarrow C, D \rightarrow AC, D \rightarrow E$ ii) $A \rightarrow BC, D \rightarrow AE$

Solution : We will assume these relations as F and G. That means -

F : $A \rightarrow B, AB \rightarrow C, D \rightarrow AC, D \rightarrow E$

G : $A \rightarrow BC, D \rightarrow AE$

Now we will find the F^+ and G^+ as follows

Step 1 : For F

Using G functional dependencies -

$$(A)^+ = \{A,B,C\}$$

$$(AB)^+ = \{A,B,C\}$$

$$(D)^+ = \{D,A,C,E,B\}$$

Using F functional dependencies -

$$(A)^+ = \{A,B,C\}$$

$$(AB)^+ = \{A,B,C\}$$

$$(D)^+ = \{D,A,C,E,B\}$$

Step 2 : For G

Using F functional dependencies -

$$(A)^+ = \{A,B,C\}$$

$$(D)^+ = \{D,A,C,E,B\}$$

Using G functional dependencies -

$$(A)^+ = \{A,B,C\}$$

$$(D)^+ = \{D,A,C,E,B\}$$

Step 3 : From both these steps

$G \supseteq F$ and $F \supseteq G$

Hence F and G are equivalent.

Minimal cover

Formal definition : A minimal cover for a set F of FDs is a set G of FDs such that :

- 1) Every dependency in G is of the form $X \rightarrow A$, where A is a single attribute.
- 2) **The closure F^+ is equal to the closure G^+ .**
- 3) If we obtain a set H of dependencies from G by deleting one or more dependencies or by deleting attributes from a dependency in G, then $F^+ \neq H^+$.

Concept of extraneous attributes

Definition : An attribute of a functional dependency is said to be extraneous if we can remove it without changing the closure of the set of functional dependencies. The formal definition of extraneous attributes is as follows :

Consider a set F of functional dependencies and the functional dependency $\alpha \rightarrow \beta$ in F

- Attribute A is extraneous in α if $A \in \alpha$ and F logically implies $(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}$
- Attribute A is extraneous in β if $A \in \beta$ and the set of functional dependencies $(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha \rightarrow (\beta - A))\}$ logically implies F .

Algorithm for computing canonical cover for set of functional dependencies F

$F_C = F$

repeat

Use the union rule to replace any dependencies in F_C of the form

$\alpha_1 \rightarrow \beta_1$ and $\alpha_1 \rightarrow \beta_2$ and $\alpha_1 \rightarrow \beta_1 \beta_2$

Find a functional dependency $\alpha \rightarrow \beta$ in F_C with an extraneous attribute either in α or in β .

/* The test for extraneous attributes is done using F_C , not F */

If an extraneous attribute is found, delete it from $\alpha \rightarrow \beta$ in F_C .

until (F_C does not change)

Example 4.12.4 Consider the following functional dependencies over the attribute set $R(ABCDE)$ for finding minimal cover $FD = \{A \rightarrow C, AC \rightarrow D, B \rightarrow ADE\}$.

Solution :

Step 1 : Split the FD such that R.H.S contain single attribute. Hence we get

$A \rightarrow C$

$AC \rightarrow D$

$B \rightarrow A$

$B \rightarrow D$

$B \rightarrow E$

Step 2 : Find the **redundant entries** and **delete** them. This can be done as follows -

- **For $A \rightarrow C$:** We find $(A)^+$ by assuming that we delete $A \rightarrow C$ temporarily. We get $(A)^+ = \{A\}$. Thus from A it is not possible to obtain C by deleting $A \rightarrow C$. This means we can not delete $A \rightarrow C$.
- **For $AC \rightarrow D$:** We find $(AC)^+$ by assuming that we delete $AC \rightarrow D$ temporarily. We get $(AC)^+ = \{AC\}$. Thus by such deletion it is not possible to obtain D . This means we can not delete $AC \rightarrow D$.

- **For B → A :** We find $(B)^+$ by assuming that we delete B → A temporarily. We get $(B)^+ = \{BDE\}$. Thus by such deletion it is not possible to obtain A. This means we can not delete B → A.
- **For B → D :** We find $(B)^+$ by assuming that we delete B → D temporarily. We get $(B)^+ = \{BEACD\}$. This shows clearly that even if we delete B → D we can obtain D. This means we can delete B → A. Thus it is redundant.
- **For B → E :** We find $(B)^+$ by assuming that we delete B → E temporarily. We get $(B)^+ = \{BDAC\}$. Thus by such deletion it is not possible to obtain E. This means we can not delete B → E.

To summarize we get now

A → C

AC → D

B → A

B → E

Thus R.H.S gets simplified.

Step 3 : Now we will simplify L.H.S.

Consider AC → D. Here we can split A and C. For that we find closure set of A and C.

$$(A)^+ = (AC)$$

$$(C)^+ = (C)$$

Thus C can be obtained from both A as well as C. That also means we need not have to have AC on L.H.S. Instead, only A can be allowed and C can be eliminated. Thus after simplification we get

A → D

To summarize we get now

A → C

A → D

B → A

B → E

Thus L.H.S gets simplified.

Step 3 : The simplified L.H.S. and R.H.S can be combined together to form

A → CD

$B \rightarrow AE$

This is a **minimal cover** or **canonical cover** of functional dependencies.

Example 4.12.5 A relation $R(A,C,D,E,H)$ satisfies the following FDs $A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H$. Find the canonical cover for this set of FD's.

Solution : For obtaining canonical cover we have to find the redundant entries from both LHS and RHS and eliminate them.

Step 1 : Suppose we minimize LHS first, then go through each production rule one by one considering LHS.

$A \rightarrow C$, Keep it as it is.

$AC \rightarrow D$, Here $A \rightarrow C$ and $A \rightarrow D$, So we remove $A \rightarrow C$, hence $A \rightarrow D$ is kept by eliminating C from LHS.

$E \rightarrow AD$, keep it as it is as E is a single attribute at LHS.

$E \rightarrow H$, keep it as it is

Step 2 : Now we will minimize RHS.

$A \rightarrow C$, keep it as it is

$A \rightarrow D$, keep it as it is

$E \rightarrow AD$. That means $E \rightarrow A$ and $E \rightarrow D$.

As $A \rightarrow D$ is also present in the FD, so we get $E \rightarrow A$ and $A \rightarrow D$. Thus $E \rightarrow D$ is transitive. Hence neglect it. So we keep $E \rightarrow A$ only

$E \rightarrow H$, Keep it as it is.

Step 3 : From steps 1 and 2, we get minimal cover of FD as

$A \rightarrow C$

$A \rightarrow D$

$E \rightarrow A$

$E \rightarrow H$

Hence the canonical for is

$A \rightarrow CD$

$E \rightarrow AH$

Example 4.12.6 What is functional dependency? Find the minimal cover using the minimal cover algorithm for the following functional dependency

$F = \{AB \rightarrow D, B \rightarrow C, AE \rightarrow B, A \rightarrow D, D \rightarrow EF\}$

Solution :

Step 1 : We will make right hand sides atomic

$AB \rightarrow D$

$B \rightarrow C$

$AE \rightarrow B$

$A \rightarrow D$

$D \rightarrow E$

$D \rightarrow F$

Step 2 : Now we will remove redundant FDs using RHS

- For $AB \rightarrow D$. Now compute $(AB)^+$ without considering the $AB \rightarrow D$ i.e. $\{G - (AB \rightarrow D)\}$

We get $(AB)^+ = \{ABCDEF\}$. That means we can remove $AB \rightarrow D$ as it is redundant entry.

Hence grammar is

$B \rightarrow C$

$AE \rightarrow B$

$A \rightarrow D$

$D \rightarrow E$

$D \rightarrow F$

- For $B \rightarrow C$ compute $(B)^+$ by considering $\{G - (B \rightarrow C)\}$

$(B)^+ = \{AEBDF\}$. As C is not present in this set. That means $B \rightarrow C$ is not redundant. So we can not remove it.

Hence grammar is

$B \rightarrow C$

$AE \rightarrow B$

$A \rightarrow D$

$D \rightarrow E$

$D \rightarrow F$

- For $AE \rightarrow B$, we will compute $(AE)^+$ under $\{G - (AE \rightarrow B)\}$

$(AE)^+ = \{AEDF\}$ as B is not present in $(AE)^+$. So we cannot remove $AE \rightarrow B$ from grammar. Hence grammar will be

$B \rightarrow C$

$AE \rightarrow B$

$A \rightarrow D$

$D \rightarrow E$

$D \rightarrow F$

For $A \rightarrow D$, we will compute $(A)^+$ under $(G - (A \rightarrow D))$

$(A)^+ = \{AEBC\}$. As D is not present in $(A)^+$, hence we can not eliminate $A \rightarrow D$. The grammar is

$B \rightarrow C$

$AE \rightarrow B$

$A \rightarrow D$

$D \rightarrow E$

$D \rightarrow F$

- For $D \rightarrow E$, compute $(D)^+$ under $(G - (D \rightarrow E))$

$(D)^+ = \{DF\}$. As E is not present in $(D)^+$, We cannot remove $D \rightarrow E$

- For $D \rightarrow F$, compute $(D)^+$ under $(G - (D \rightarrow F))$

$(D)^+ = \{DE\}$. As F is not present in $(D)^+$, We cannot remove $D \rightarrow F$. Finally the grammar

is

$B \rightarrow C$

$AE \rightarrow B$

$A \rightarrow D$

$D \rightarrow E$

$D \rightarrow F$

Step 3 : Remove redundant entries based on RHS

$B \rightarrow C$, $A \rightarrow D$, $D \rightarrow E$ and $D \rightarrow F$ as LHS is atomic.

Now we consider $AE \rightarrow B$

For A : compute E^+ with respect to $(G - (AE \rightarrow B) \cup (E \rightarrow B))$

E^+ using $\{B \rightarrow C, E \rightarrow B, A \rightarrow D, D \rightarrow E, D \rightarrow F\} = EBC$

E^+ doesn't contain A , so A not redundant in $AE \rightarrow B$

For E : compute A^+ with respect to $(G - (AE \rightarrow B) \cup (A \rightarrow B))$

A^+ using $\{B \rightarrow C, A \rightarrow B, A \rightarrow D, D \rightarrow E, D \rightarrow F\} = ABDEFC$

A^+ contains E , so E is redundant in $AE \rightarrow B$

Hence we consider $AE \rightarrow B$ as $A \rightarrow B$.

Finally minimal closure is $\{B \rightarrow C, A \rightarrow B, A \rightarrow D, D \rightarrow E, D \rightarrow F\}$

Example 4.12.7 Using the minimal cover algorithm, find the minimal cover for the following FDs :

$F = \{AB \rightarrow C, A \rightarrow D, BD \rightarrow C, D \rightarrow BG, AE \rightarrow F\}$.

Solution : We will make right hand side atomic

$AB \rightarrow C$

$A \rightarrow D$

$BD \rightarrow C$

$D \rightarrow B$

$D \rightarrow G$

$AE \rightarrow F$

Step 2 : Now we will remove redundant FDs using RHS

- For $AB \rightarrow C$ we compute $(AB)^+$ without considering the rule $AB \rightarrow C$ i.e. $(G - (AB \rightarrow C))$. We get $(AB)^+ = \{ABDBGC\}$ i.e C is present by other way also in the set. Hence we can **remove $AB \rightarrow C$** as it is redundant entry.
- For $A \rightarrow D$, we compute $(A)^+$ using $(G - (A \rightarrow D))$. We get $(A)^+ = \{A\}$. We can not get D. So it is not a redundant entry and we can not remove it.
- For $BD \rightarrow C$, We compute $(BD)^+$ using $(G - (BD \rightarrow C))$. We get $(BD)^+ = \{BDG\}$. This is also not a redundant entry and we can not remove it.
- For $D \rightarrow B$. Let us compute $(D)^+$ using $(G - (D \rightarrow B))$. We get $(D)^+ = \{DG\}$. This again indicates that we can not get B without the rule $D \rightarrow B$. Hence it is not a redundant entry and we can not remove it.
- Similarly, we can conclude For $D \rightarrow G$ and $AE \rightarrow F$ as not redundant entries.
- Finally, the grammar will be

$A \rightarrow D$

$BD \rightarrow C$

$D \rightarrow B$

$D \rightarrow G$

$AE \rightarrow F$

Step 2 : Now we will remove redundant entries based on LHS.

The $A \rightarrow D, D \rightarrow B, D \rightarrow G$ remain as it is in minimal cover as LHS is atomic.

Now consider $BD \rightarrow C$, We can replace this by $B \rightarrow C$ and eliminate D , as $D \rightarrow B$ is present.

Similarly consider $AE \rightarrow F$. But as we cannot replace it either by $A \rightarrow F$ or $E \rightarrow F$. So it is not redundant.

The minimal cover is $\{A \rightarrow D, D \rightarrow B, D \rightarrow G, B \rightarrow C, AE \rightarrow F\}$.

4.13 Algorithms for Decomposition

- Decomposition is the process of **breaking down one table into multiple tables**.
- **Formal definition of decomposition is -**
- A decomposition of relation schema R consists of replacing the relation schema by two relation schema that each contain a subset of attributes of R and together include all attributes of R by storing projections of the instance.
- For example - Consider the following table

Employee_Department table as follows -

Eid	Ename	Age	City	Salary	Deptid	DeptName
E001	ABC	29	Pune	20000	D001	Finance
E002	PQR	30	Pune	30000	D002	Production
E003	LMN	25	Mumbai	5000	D003	Sales
E004	XYZ	24	Mumbai	4000	D004	Marketing
E005	STU	32	Hyderabad	25000	D005	Human Resource

We can decompose the above relation Schema into two relation schemas as **Employee (Eid, Ename, Age, City, Salary)** and **Department (Deptid, Eid, DeptName)** as follows -

Employee Table

Eid	Ename	Age	City	Salary
E001	ABC	29	Pune	20000
E002	PQR	30	Pune	30000
E003	LMN	25	Mumbai	5000
E004	XYZ	24	Mumbai	4000
E005	STU	32	Hyderabad	25000

Department Table

Deptid	Eid	DeptName
D001	E001	Finance
D002	E002	Production
D003	E003	Sales
D004	E004	Marketing
D005	E005	Human Resource

- The decomposition is used for eliminating redundancy.
- **For example :** Consider following relation **Schema R** in which we assume that the grade determines the salary, the redundancy is caused

Schema R

Name	eid	deptname	Grade	Salary
AAA	121	Accounts	2	8000
AAA	132	Sales	3	7000
BBB	101	Marketing	4	7000
CCC	106	Purchase	2	8000

Redundancy!!!

- Hence, the above table can be decomposed into two Schema S and T as follow :

Schema S			
Name	eid	deptname	Grade
AAA	121	Accounts	2
AAA	132	Sales	3
BBB	101	Marketing	4
CCC	106	Purchase	2

Schema T	
Grade	Salary
2	8000
3	7000
4	7000
2	8000

Problems related to decomposition :

Following are the potential problems to consider :

- 1) Some queries become more **expensive**.
- 2) Given instances of the decomposed relations, we may not be able to reconstruct the corresponding instance of the original relation!

- 3) Checking some dependencies may require joining the instances of the decomposed relations.
- 4) There may be loss of information during decomposition.

Properties associated with decomposition

There are two properties associated with decomposition and those are -

- 1) **Loss-less join or non loss decomposition** : When all information found in the original database is preserved after decomposition, we call it as loss less or non loss decomposition.
- 2) **Dependency preservation** : This is a property in which the constraints on the original table can be maintained by simply enforcing some constraints on each of the smaller relations.

4.14 Lossless Join

SPPU : May-18, Marks 6

The lossless join can be defined using following three conditions :

- i) **Union** of attributes of R1 and R2 must be equal to attribute of R. Each attribute of R must be either in R1 or in R2.

$$\text{Att}(R1) \cup \text{Att}(R2) = \text{Att}(R)$$

- ii) **Intersection** of attributes of R1 and R2 must not be NULL.

$$\text{Att}(R1) \cap \text{Att}(R2) \neq \Phi$$

- iii) **Common attribute** must be a key for at least one relation (R1 or R2)

$$\text{Att}(R1) \cap \text{Att}(R2) \rightarrow \text{Att}(R1)$$

$$\text{or } \text{Att}(R1) \cap \text{Att}(R2) \rightarrow \text{Att}(R2)$$

Example 4.14.1 Consider the following relation $R(A, B, C, D)$ and FDs $A \rightarrow BC$, is the decomposition of R into $R1(A, B, C)$, $R2(A, D)$. Check if the decomposition is lossless join or not.

Solution :

Step 1 : Here $\text{Att}(R1) \cup \text{Att}(R2) = \text{Att}(R)$ i.e $R1(A,B,C) \cup R2(A,D) = (A,B,C,D)$ i.e R .

Thus first condition gets satisfied.

Step 2 : Here $R1 \cap R2 = \{A\}$. Thus $\text{Att}(R1) \cap \text{Att}(R2) \neq \Phi$. Here the second condition gets satisfied.

Step 3 : $\text{Att}(R1) \cap \text{Att}(R2) \rightarrow \{A\}$. Now $(A)^+ = \{A,B,C\}$ i.e. attributes of R1. Thus the third condition gets satisfied.

This shows that the given decomposition is a **lossless join**.

Example 4.14.2 Consider the following relation $R(A, B, C, D, E, F)$ and FDs $A \rightarrow BC$, $C \rightarrow A$, $D \rightarrow E$, $F \rightarrow A$, $E \rightarrow D$ is the decomposition of R into $R1(A, C, D)$, $R2(B, C, D)$ and $R3(E, F, D)$. Check for lossless.

Solution :

Step 1 : $R1 \cup R2 \cup R3 = R$. Here the first condition for checking lossless join is satisfied as $(A,C,D) \cup (B,C,D) \cup (E,F,D) = \{A,B,C,D,E,F\}$ which is nothing but R .

Step 2 : Consider $R1 \cap R2 = \{CD\}$ and $R2 \cap R3 = \{D\}$. Hence second condition of intersection not being Φ gets satisfied.

Step 3 : Now, consider $R1(A, C, D)$ and $R2(B, C, D)$. We find $R1 \cap R2 = \{CD\}$

$(CD)^+ = \{ABCDE\} \in$ attributes of $R1$ i.e. $\{A, C, D\}$. Hence condition 3 for checking lossless join for $R1$ and $R2$ gets satisfied.

Step 4 : Now, consider $R2(B, C, D)$ and $R3(E, F, D)$. We find $R2 \cap R3 = \{D\}$.

$(D)^+ = \{D, E\}$ which is neither complete set of attributes of $R2$ or $R3$.

[Note that F is missing for being attribute of $R3$].

Hence it is not **lossless join decomposition**. Or in other words we can say it is a **lossy decomposition**.

Example 4.14.3 Suppose that we decompose schema $R = (A,B,C,D,E)$ into (A,B,C) (C,D,E)
Show that it is not a lossless decomposition.

Solution :

Step 1 : Here we need to assume some data for the attributes A, B, C, D , and E . Using this data we can represent the relation as follows -

Relation R

A	B	C	D	E
a	1	x	p	q
b	2	x	r	s

Relation R1 = (A,B,C)

A	B	C
a	1	x
b	2	x

Relation R2 = (C,D,E)

C	D	E
x	p	q
x	r	s

Step 2 : Now we will join these tables using natural join, i.e. the join based on common attribute C. We get $R1 \bowtie R2$ as

A	B	C	D	E
a	1	x	p	q
a	1	x	r	s
b	2	x	p	q
b	2	x	r	s

Here we get more rows or tuples than original relation R

Clearly $R1 \bowtie R2 \neq R$. Hence it is not lossless decomposition.

Review Question

1. List the properties of decomposition. Explain lossless join with example.

SPPU : May-18, End Sem, Marks 6

4.15 Dependency Preservation

- **Definition** : A decomposition $D = \{R1, R2, R3, \dots, Rn\}$ of R is dependency preserving for a set F of functional dependency if - $(F1 \cup F2 \cup \dots \cup Fm) = F$.
- If decomposition is not dependency-preserving, some dependency is lost in the decomposition.

Example 4.15.1 Consider the relation R (A, B, C) for functional dependency set $\{A \rightarrow B \text{ and } B \rightarrow C\}$ which is decomposed into two relations $R1 = (A, C)$ and $R2 = (B, C)$. Then check if this decomposition dependency preserving or not.

Solution : This can be solved in following steps :

Step 1 : For checking whether the decomposition is dependency preserving or not we need to check following condition

$$F^+ = (F1 \cup F2)^+$$

Step 2 : We have with us the $F^+ = \{ A \rightarrow B \text{ and } B \rightarrow C \}$

Step 3 : Let us find $(F1)^+$ for relation R1 and $(F2)^+$ for relation R2

R1(A,C)	R2(B,C)
A → A Trivial	B → B Trivial
C → C Trivial	C → C Trivial
A → C ∴ In (F)+ A → B → C and it is Nontrivial	B → C ∴ In (F)+ B → C and it is
AC → AC Trivial	Non-Trivial
A → B but is not useful as B is not part of R1 set	BC → BC Trivial
We can not obtain C → A	We can not obtain C → B

Step 4 : We will eliminate all the trivial relations and useless relations. Hence we can obtain R1 and R2 as

R1(A,C)	R2(B,C)
A → C Nontrivial	B → C Non-Trivial

$$(F1 \cup F2)^+ = \{A \rightarrow C, B \rightarrow C\} \neq \{A \rightarrow B, B \rightarrow C\} \text{ i.e. } (F)^+$$

Thus the condition specified in step 1 i.e. $F^+ = (F1 \cup F2)^+$ is **not true**. Hence it is **not dependency preserving decomposition**.

Example 4.15.2 Let relation $R(A, B, C, D)$ be a relational schema with following functional dependencies $\{A \rightarrow B, B \rightarrow C, C \rightarrow D \text{ and } D \rightarrow B\}$. The decomposition of R into (A, B) , (B, C) and (B, D) . Check whether this decomposition is dependency preserving or not.

Solution :

Step 1 : Let $(F)^+ = \{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow B\}$.

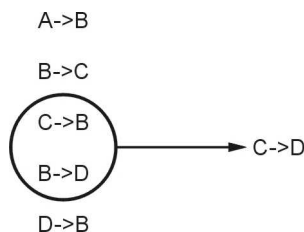
Step 2 : We will find $(F1)^+$, $(F2)^+$, $(F3)^+$ for relations $R1(A,B)$, $R2(B,C)$ and $R3(B,D)$ as follows -

<p style="text-align: center;">R1(A,B)</p> <p>A->A Trivial</p> <p>B->B Trivial</p> <p>A->B ∴ $(F)^+$</p> <p>and it's non Trivial</p> <p>B->A can not be obtained</p> <p>AB->AB</p>	<p style="text-align: center;">R2(B,C)</p> <p>B->B Trivial</p> <p>C->C Trivial</p> <p>B->C ∴ $(F)^+$ and it's non Trivial</p> <p>C->B ∴ In $(F)^+$ and</p> <p>C->D->C and it is Nontrivial</p> <p>BC->BC Trivial</p>	<p style="text-align: center;">R3(B,D)</p> <p>B->B Trivial</p> <p>D->D Trivial</p> <p>B->D ∴ $(F)^+$ as and</p> <p>B->C->D and it's non Trivial</p> <p>D->B ∴ $(F)^+$ and it's non Trivial</p> <p>BD->BD Trivial</p>
---	--	--

Step 3 : We will eliminate all the trivial relations and useless relations. Hence we can obtain $R1 \cup R2 \cup R3$ as

<p style="text-align: center;">R1(A,B)</p> <p>A->B</p>	<p style="text-align: center;">R2(B,C)</p> <p>B->C</p> <p>C->B</p>	<p style="text-align: center;">R2(B,D)</p> <p>B->D</p> <p>D->B</p>
--	---	---

Step 4 : As from above FD's we get



Step 5 : This proves that $F^+ = (F1 \cup F2 \cup F3)^+$. Hence given **decomposition is dependency preserving.**

Example 4.15.3 Given relation $r(X,Y,W,Z,P,Q)$ and the set $F = \{XY \rightarrow W, XW \rightarrow P, PQ \rightarrow Z, XY \rightarrow Q\}$. Consider the decomposition $R1(Z,P,Q)$, $R2(X,Y,Z,P,Q)$. Is this decomposition lossless or lossy? Use lossless join algorithm.

Solution :

Step 1 : Construct a table with six columns for given six attributes i.e. X, Y, W, Z, P, Q and two rows for given two relation i.e. R1 and R2.

	X	Y	W	Z	P	Q
R1						
R2						

Step 2 : Fill in the entries as follows -

R1 (Z, P, Q) and R2 (X, Y, Z, P, Q)

Consider R1 having attributes Z, P, Q so put ' α ' in those row and put ' β ' other remaining rows same as R2 having attributes X, Y, Z, P, Q so put ' α ' in those row and put ' β ' other remaining rows.

	X	Y	W	Z	P	Q
R1	β_{1X}	β_{1Y}	β_{1W}	α_Z	α_P	α_Q
R2	α_X	α_Y	β_{2W}	α_Z	α_P	α_Q

Step 3 :

- i) Considering $XY \rightarrow W$, we check if the two rows of the table have the same value under the columns XY that make up the determinant of the FD. Since the rows do not have identical values, the table will remain unchanged and we repeat step 3 by considering another FD.
- ii) Considering $XW \rightarrow P$, we check if the two rows of the table have the same value under the columns XW that make up the determinant of the FD. Since the rows do not have identical of values, the table will remain unchanged and we repeat step 3 by considering another FD.
- iii) Considering $PQ \rightarrow Z$, we check if the two rows of the table have the same value under the columns PQ that make up the determinant of the FD. Since the rows R1 and R2 have values of α_P and α_Q under the columns PQ. Therefore, we need to equate all the corresponding entries for the rows under column Z. The values in column Z are already equal no changes are necessary Repeat step 3 again.
- iv) Considering $XY \rightarrow Q$, we check if the two rows of the table have the value under the columns XY that make up determinant of the FD. Since rows do not have identical values, the table will remain unchanged. Now, there are no more FDs to consider.

Step 4 : Since there is no row in the table that has all Φ in it's entries the decomposition is lossy. That is original table cannot be recovered from the join of relations R1 and R2.

Example 4.15.4 Suppose we decompose the scheme $R=(A,B,C,D,E)$ into (A,B,C) and (A,D,E) .

Show that this decomposition is the lossless decomposition if following functional dependencies hold : $A \rightarrow BC, CD \rightarrow E, B \rightarrow D$

Show that decomposition is dependency preserving decomposition.

Solution :

Step 1 : Construct table with five columns for given five attributes i.e. A, B, C, D, E and two rows for given two relations i.e. R1 (A, B, C) and R2 (A, D, E).

	A	B	C	D	E
R1					
R2					

Step 2 : The attributes of the scheme of R1 are A, B, C. Therefore, we place α_A , α_B and α_C under these columns respectively. The remaining entries of this row are filled with β_{1D} and β_{1E} . Same as for R2 have attributes A, D, E. Therefore, we place α_A , α_D and α_E under these columns respectively. The remaining entries of this row are filled with β_{2B} and β_{2C} .

	A	B	C	D	E
R1	α_A	α_B	α_C	β_{1D}	β_{1E}
R2	α_A	β_{2B}	β_{2C}	α_D	α_E

Step 3 :

- i) Considering $A \rightarrow BC$ we check if the two rows of the table have the same value under the columns that make up the determinant of FD. Rows R1 and R2 has values α_A under column A. Therefore, we need to equate all the corresponding entries for these rows under columns B and C.

Since entry under column B is α_B (for row R1) and column C is α_C (for row R2).

	A	B	C	D	E
R1	α_A	α_B	α_C	β_{1D}	β_{1E}
R2	α_A	α_B	α_C	α_D	α_E

- ii) Considering $CD \rightarrow E$, we check for rows that have the same value in the columns C and D. Here, rows do not same values, the table remain unchanged and we repeat step 3 by another FD.

iii) Considering $B \rightarrow D$, we check for rows that have the same value in the columns B here rows have same values α_B under the column B. Therefore, we need to equate all the corresponding entries for these rows under column D. Since entry under column D is α_D for row R1.

	A	B	C	D	E
R1	α_A	α_B	α_C	α_D	β_{1E}
R2	α_A	α_B	α_C	α_D	α_E

Now there are no more FDs to consider.

Step 4 : Since row R2 has become $\alpha_A, \alpha_B, \alpha_C, \alpha_D, \alpha_E$ i.e. R2 has all α values. Hence decomposition is lossless.

4.16 Second Normal Form (2NF)

SPPU : Oct.-19, Marks 7

Before understanding the second normal form let us first discuss the concept of partial functional dependency and prime and non prime attributes.

Concept of partial functional dependency

Partial dependency means that a nonprime attribute is functionally dependent on part of a candidate key.

For example : Consider a relation $R(A,B,C,D)$ with functional dependency

$\{AB \rightarrow CD, A \rightarrow C\}$

Here (AB) is a candidate key because

$$(AB)^+ = \{ABCD\} = \{R\}$$

Hence $\{A,B\}$ are prime attributes and $\{C,D\}$ are non prime attribute. In $A \rightarrow C$, the non prime attribute C is dependent upon A which is actually a part of candidate key AB. Hence due to $A \rightarrow C$ we get partial functional dependency.

Prime and non prime attributes

- **Prime attribute :** An attribute, which is a part of the candidate-key, is known as a prime attribute.
- **Non-prime attribute :** An attribute, which is not a part of the prime-key, is said to be a non-prime attribute.
- **Example : Consider a Relation $R = \{A,B,C,D\}$ and candidate key as AB, the Prime attributes : A, B.**

Non Prime attributes : C, D

The second normal form

For a table to be in the Second Normal Form, following conditions must be followed

- i) It should be in the First Normal form.
- ii) It should not have partial functional dependency.

For example : Consider following table in which every information about a the Student is maintained in a table such as student id(sid), student name(sname), course id(cid) and course name(cname).

Student_Course

sid	sname	cid	cname
1	AAA	101	C
2	BBB	102	C++
3	CCC	101	C
4	DDD	103	Java

This table is not in 2NF. For converting above table to 2NF we must follow the following steps -

Step 1 : The above table is in 1NF.

Step 2 : Here **sname** and **sid** are associated similarly **cid** and **cname** are associated with each other. Now if we delete a record with **sid = 2**, then automatically the course C++ will also get deleted. Thus,

sid->sname or **cid->cname** is a partial functional dependency, because **{sid,cid}** should be essentially a candidate key for above table. Hence to bring the above table to 2NF we must decompose it as follows :

Student

sid	sname	cid
1	AAA	101
2	BBB	102
3	CCC	101
4	DDD	103

Here candidate key is
(sid, cid)
and
(sid, cid)->sname

Course

cid	cname
101	C
102	C++
101	C
103	Java

Here candidate key is cid
here cid -> cname

Thus now table is in 2NF as there is no partial functional dependency.

Example 4.16.1 Study the relation given below and state what level of normalization can be achieved and normalize it upto that level.

Order no.	Order date	Item lines		
		Item code	Quantity	Price/Unit
1456	26-12-1999	3687	52	50.4
		4627	38	60
		3214	20	20.00
1886	04-03-1999	4629	45	20.25
		4627	30	60.20
1788	04-04-1999	4627	40	60.20

Solution :

Reason for the given relation being unnormalized

1. Observe order for many items.
2. Item lines has many attributes-called composite attributes.
3. Each tuple has variable length.
4. Difficult to store due to non-uniformity.
5. Given item code difficult to find qty-ordered and hence called Unnormalized relation.

For conversion to first normal form -

- Identify the composite attributes, convert the composite attributes to individual attributes.
- Duplicate the common attributes as many times as lines in composite attribute.
- Every attribute now describes single property and not multiple properties, some data will be duplicated.
- Now this is called First normal form (1NF) also called flat file.

Order no.	Order date	Item lines		
		Item code	Quantity	Price/Unit
1456	26-12-1999	3687	52	50.4
1456	26-12-1999	4627	38	60
1456	26-12-1999	3214	20	20.00
1886	04-03-1999	4629	45	20.25
1886	04-03-1999	4627	30	60.20
1788	04-04-1999	4627	40	60.20

Fig. 4.16.1 Table in first normal form

- The above table has insertion, deletion and update anomalies. For instance - if we delete order no. 1886, then the item code 4629 gets lost. Similarly if we update 4627, then all instances of 4627 need to be changed.
- We need to convert 2NF if it is in 1NF. The non-key attributes are functionally dependent on key attribute and if there is a composite key then no non-key attribute is functionally depend on one part of the key.
- The table can be converted to 2NF as follows -

Orders

OrderNo	OrderDate
1456	26-12-1999
1886	04-03-1999
1788	04-04-1999

Order details

OrderNo	ItemCode	Qty
1456	3687	52
1886	4629	45
1788	4627	40

Prices

ItemCode	Price/Unit
3687	50.4
4627	60
3214	20
4629	20.25

Review Question

1. Explain why database normalization is required for good relational database design ? Explain with example requirements of second normal form.

SPPU : Oct.-19, In Sem, Marks 7**4.17 Third Normal Form (3NF)****SPPU : Oct.-18, 19, Aug.-17, Dec.-18, Marks 7**

Before understanding the third normal form let us first discuss the concept of transitive dependency, super key and candidate key.

Concept of transitive dependency

A functional dependency is said to be transitive if it is indirectly formed by two functional dependencies. For example -

$X \rightarrow Z$ is a transitive dependency if the following functional dependencies hold true :

$X \rightarrow Y$

$Y \rightarrow Z$

Concept of super key and candidate key

Superkey : A super key is a set or one of more columns (attributes) to uniquely identify rows in a table.

Candidate key : The minimal set of attribute which can uniquely identify a tuple is known as candidate key. For example consider following table

RegID	RollNo	Sname
101	1	AAA
102	2	BBB
103	3	CCC
104	4	DDD

Superkeys

- {RegID}
- {RegID, RollNo}
- {RegID, Sname}
- {RollNo, Sname}
- {RegID, RollNo, Sname}

Candidate keys

- {RegID}
- {RollNo}

Third normal form

A table is said to be in the third normal form when,

- It is in the second normal form.(i.e. it does not have partial functional dependency).
- It doesn't have transitive dependency.

Or in other words

In other words 3NF can be defined as : A table is in 3NF if it is in 2NF and for each functional dependency

$X \rightarrow Y$

at least one of the following conditions hold :

- X is a super key of table.
- Y is a prime attribute of table.

For example : Consider following table **Student_details** as follows -

sid	sname	zipcode	cityname	state
1	AAA	11111	Pune	Maharashtra
2	BBB	22222	Surat	Gujarat
3	CCC	33333	Chennai	Tamilnadu
4	DDD	44444	Jaipur	Rajastan
5	EEE	55555	Mumbai	Maharashtra

Here

Super keys : {sid},{sid,sname},{sid,sname,zipcode}, {sid,zipcode,cityname}... and so on.

Candidate keys : {sid}

Non-prime attributes : {sname,zipcode,cityname,state}

The dependencies can be denoted as

sid->sname

sid->zipcode

zipcode->cityname

cityname->state

The above denotes the transitive dependency. Hence above table is not in 3NF. We can convert it into 3NF as follows :

Student

sid	sname	zipcode
1	AAA	11111
2	BBB	22222
3	CCC	33333
4	DDD	44444
5	EEE	55555

Zip

zipcode	cityname	state
11111	Pune	Maharashtra
22222	Surat	Gujarat
33333	Chennai	Tamilnadu
44444	Jaipur	Rajasthan
55555	Mumbai	Maharashtra

Example 4.17.1 Consider the relation $R = \{A, B, C, D, E, F, G, H, I, J\}$ and the set of functional dependencies $F = \{A, B \rightarrow C, A \rightarrow \{D, E\}, B \rightarrow F, F \rightarrow \{G, H\}, D \rightarrow \{I, J\}\}$

1. What is the key for R ? Demonstrate it using the inference rules.
2. Decompose R into 2NF, then 3NF relations.

Solution : Let,

$$A \rightarrow DE \text{ (given)}$$

$$\therefore A \rightarrow D, A \rightarrow E \text{ (decomposition rule)}$$

$$\text{As } D \rightarrow IJ, A \rightarrow IJ$$

Using union rule we get

$$A \rightarrow DEIJ$$

$$\text{As } A \rightarrow A$$

$$\text{we get } A \rightarrow ADEIJ$$

Using augmentation rule we compute AB

$$AB \rightarrow ABDEIJ$$

$$\text{But } AB \rightarrow C \text{ (given)}$$

$$\therefore AB \rightarrow ABCDEIJ$$

$$B \rightarrow F \text{ (given)} \quad F \rightarrow GH \quad \therefore B \rightarrow GH \text{ (transitivity)}$$

$$\therefore AB \rightarrow AGH \text{ is also true}$$

$$\text{Similarly } AB \rightarrow AF \quad \because B \rightarrow F \text{ (given)}$$

Thus now using union rule

$$AB \rightarrow ABCDEFGHIJ$$

\therefore AB is a key

The table can be converted to 2NF as

$$R_1 = (\underline{A}, \underline{B}, C)$$

$$R_2 = (\underline{A}, D, E, I, J)$$

$$R_3 = (\underline{B}, F, G, H)$$

The above 2NF relations can be converted to 3NF as follows

$$R_1 = (\underline{A}, \underline{B}, C)$$

$$R_2 = (\underline{A}, D, E)$$

$$R_3 = (\underline{D}, I, J)$$

$$R_4 = (\underline{B}, E)$$

$$R_5 = (E, G, H).$$

Example 4.17.2 A software contract and consultancy firm maintains details of all the various projects in which its employees are currently involved. These details comprise :

- Employee number

- *Employee name*
- *Date of birth*
- *Department code*
- *Department name*
- *Project code*
- *Project description*
- *Project supervisor*

Assume the following :

- *Each employee number is unique.*
- *Each department has a single department code.*
- *Each project has a single code and supervisor.*
- *Each employee may work on one or more projects.*
- *Employee names need not necessarily be unique.*
- *Project code, project description and project supervisor are repeating fields.*

Normalise this data to third normal form.

Solution :

Un-Normalized Form

Employee Number, Employee Name _Date of Birth_ Department Code_ Department Name_ Project Code_ Project Description_ Project Supervisor

1NF

Employee Number, Employee Name _Date of Birth
Department Code, Department Name
Employee Number, Project Code, Project Description_ Project Supervisor

2NF

Employee Number, Employee Name _Date of Birth_ Department Code_ Department Name
Employee Number, Project Code,
Project Code, Project Description, Project Supervisor

3NF

Employee Number, Employee Name _Date of Birth_ *Department Code
Department Code, Department Name
Employee Number, Project Code
Project Code, Project Description, Project Supervisor

Example 4.17.3 What is normalization? Normalize below given relation upto 3NF

STUDENT.

StudID	StudName	City	Pincode	ProjectID	ProjectName	Course	Content
S101	Ajay	Surat	326201	P101	Health	Programming	C++, Java, C
S102	Vijay	Pune	325456	P102	Social	WEB	HTML, PHP, ASP

Solution : For converting the given schema to first normal form, we will arrange it in such a way that have each tuple contains single record. For that purpose we need to split the schema into two tables namely **Student** and **Projects**.

1NF

Student

StudID	StudName	Pincode	City
S101	Ajay	326201	Surat
S102	Vijay	325456	Pune

Projects

StudID	ProjectID	ProjName	Course	Content
S101	P101	Health	Programming	C++
S101	P101	Health	Programming	Java
S101	P101	Health	Programming	C
S102	P102	Social	WEB	HTML
S102	P102	Social	WEB	PHP
S102	P102	Social	WEB	ASP

2NF

For a table to be in 2NF, there should not be any partial dependency.

Student

StudID	StudName	Pincode	City
S101	Ajay	326201	Surat
S102	Vijay	325456	Pune

Project

StudID	ProjectID	ProjName	CourseID
S101	P101	Health	C101
S101	P101	Health	C102
S101	P101	Health	C103
S102	P102	Social	C104
S102	P102	Social	C105
S102	P102	Social	C106

CourseDetails

CourseID	Course	Content
C101	Programming	C++
C102	Programming	Java
C103	Programming	C
C104	WEB	HTML
C105	WEB	PHP
C106	WEB	ASP

3NF

There was a transitive dependency in 2NF tables because city is associated with student ID and city depends upon zip code. Hence the transitive dependency is removed to convert table into 3NF. The required 3NF schema is as below -

Student

StudID	StudName	Pincode
S101	Ajay	326201
S102	Vijay	325456

Student_Address

Pincode	City
326201	Surat
325456	Pune

Project

StudID	ProjectID	ProjName	CourseID
S101	P101	Health	C101
S101	P101	Health	C102
S101	P101	Health	C103
S102	P102	Social	C104
S102	P102	Social	C105
S102	P102	Social	C106

CourseDetails

CourseID	Course	Content
C101	Programming	C++
C102	Programming	Java
C103	Programming	C
C104	WEB	HTML

C105	WEB	PHP
C106	WEB	ASP

Example 4.17.4 What is the need for normalization ? Consider the relation : $Emp\text{-}proj = \{ssn, Pnumber, Hours, Ename, Pname, Plocation\}$

Assume $\{ssn, Pnumber\}$ as primary key.

The dependencies are:

$\{ssn, Pnumber\} \rightarrow Hours$

$Ssn \rightarrow Ename$

$Pnumber \rightarrow \{Pname, Plocation\}$

Normalize the above relation to 3NF.

Solution : Need for normalization - Refer section 4.9.

Consider the given dependencies

(1) $\{ssn, Pnumber\} \rightarrow Hours$

(2) $ssn \rightarrow Ename$

(3) $Pnumber \rightarrow \{Pname, Plocation\}$

The dependencies 2 and 3 represents the partial dependency. Hence we convert the relation into second normal form by splitting the given $Emp\text{-}proj$ into three relations

$Emp = \{\underline{ssn}, Ename\}$

$Proj = \{\underline{Pnumber}, Pname, Plocation\}$

$Works = \{\underline{ssn}, \underline{Pnumber}, Hours\}$

Example 4.17.5 Consider the following relation for $CARSALE(CAR\text{-}NO, Date\text{-}Sold, Salesman\text{-}no, Commission, Discount)$.

Assume a car can be sold by multiple salesman and hence primary is $(CAR\text{-}NO, Salesman\text{-}no)$

Additional dependencies are

$Date_Sold \rightarrow Discount$

$Salesman_no \rightarrow Commission$

i) Is this relation in 1NF, 2NF, 3NF ? Why and Why not ?

ii) How would you normalize this completely ?

Solution :

i) Let us check the database schema against each normal form.

First normal form : As the relation have no multivalued attributes or nested relations, the given relation is in 1st normal form.

Second normal form : This relation is not in second normal form because the attribute **commission** is dependent on part of primary key **Salesman-no**.

Third normal form : This relation is not in third normal form because firstly it is not in 2nd normal form and there should be transitive dependency of a nonkey attribute on primary key.

(CAR-No, Salesman-no) → Date_sold → Discount

(ii) To normalize this relation we will decompose it into

R1={CAR-No, Salesman-no, Date_sold}

R2={Date_sold, Discount}

R3={Salesman-no, commission}

The functional dependency is as follows -

F1={(CAR-No, Salesman-no) → Date_sold}

F2={Date_sold → Discount}

F3={Salesman-no → commission}

Example 4.17.6 Normalize the below relation upto 3NF

<i>Module</i>	<i>Dept</i>	<i>Lecturer</i>	<i>Text</i>
M1	D1	L1	T1
M1	D1	L1	T2
M2	D1	L1	T1
M2	D1	L1	T3
M3	D1	L2	T4
M4	D2	L3	T1
M4	D2	3	T5
M5	D2	L4	T6

Solution : The given relation is already in 1st normal form. But it has Insert, delete and update anomalies. Because -

1) Insert anomalies : We can not add a module(M) with no texts(T).

2) **Delete anomalies** : If we remove M3, we remove L2 as well.

3) **Update anomalies** : To change lecturer for M1, we have to change two rows.

Hence we will convert it to second normal form.

Step 1 : We can define the functional dependency FD as

{Module, Text}->{Lecturer, Dept}

But

{Module}->{Lecturer, Dept}

That means Lecturer and Dept are partially dependent on the primary key. Hence for conversion of first normal form to second normal form we will decompose the give table into two tables as

Table 2a

Module	Dept	Lecturer
M1	D1	L1
M2	D1	L1
M3	D1	L2
M4	D2	L3
M5	D2	L4

Table 2b

Module	Text
M1	T1
M1	T2
M2	T1
M2	T3
M3	T4
M4	T1
M4	T5
M5	T6

The relation is now in second normal form.

Step 3 : The table 2a has Insert, Delete and Update anomalies. Because -

1) **INSERT anomalies** : We can't add lecturers who teach no modules.

2) **UPDATE anomalies** : To change the department for L1 we must alter two rows.

3) **DELETE anomalies** : If we delete M3 we delete L2 as well.

Hence, to eliminate these anomalies, we decompose table 2a into two tables and convert it to third normal form.

Step 4 : Hence we get

Table 3a

Lecturer	Dept.
L1	D1
L2	D1
L3	D2
L4	D2

Table 3b

Module	Lecturer
M1	L1
M2	L1
M3	L2
M4	L3
M5	L4

Step 5 : Thus now the complete relation is decomposed into three tables and it is in third normal form. It is summarized as below

Table 3a

Lecturer	Dept.
L1	D1
L2	D1
L3	D2
L4	D2

Table 3b

Module	Lecturer
M1	L1
M2	L1
M3	L2
M4	L3
M5	L4

Table 2b

Module	Text
M1	T1
M1	T2
M2	T1
M2	T3
M3	T4
M4	T1
M4	T5
M5	T6

Example 4.17.7 *Students_Detail (Stud_id, Stud_name, Zip, City)*

Consider above schema, check whether it is in 3NF, if not justify and propose the schema in 3NF.

SPPU : Oct.-18, In Sem, Marks 5

Solution : The given schema is not in 3NF because there exists following transitive dependencies.

Because city is associated with Stud_id and city depends upon the zip code. When this dependency is removed then the table will be in 3NF. It is as follows –

Student

Stud_id	Stud_name	Zip

Student_address

Zip	City

Example 4.17.8 *Consider a table having structure student (Roll_no, Branch_code, Marks_obtained, Exam_name, Total_marks).*

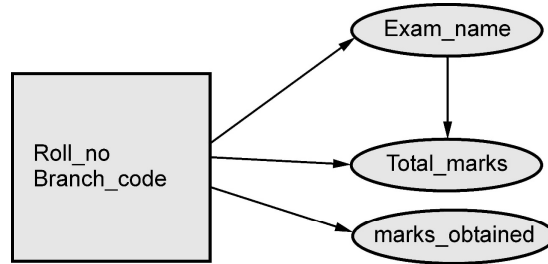
Note following points :

- i) Composite primary key for student table is (Roll_no, Branch_code).
- ii) Branch_code column stores the code of branch for which students have taken admission.
- iii) Exam name attribute is depend on both roll_no and branch_code.
- iv) Total marks attribute is depend on exam_name attribute.

Considering above requirement state whether the table created is in third normal form or not ? Why ? If not in third normal for propose the database design for above requirements which is in third normal form.

SPPU : Oct.-19, In Sem, Marks 7

Solution : Clearly the above table is not there in third normal form as there exists transitive dependencies. This is because, **exam_name** attribute depend upon the primary key **roll_no** and **branch_code** and **total_marks** are associated with **exam_name** attribute.



To convert the above schema to third normal form we need to decompose the schema into different relations –

Student_info

Roll_no	branch_code	Exam_name
---------	-------------	-----------

Exam_info

Exam_name	Total_marks	Marks_obtained
-----------	-------------	----------------

Review Questions

1. Explain what is normalization ? Explain with example requirements of Third Normal Form.

SPPU : Aug.-17, In Sem, Marks 5

2. Explain why Database normalization is required for good relational database design ? Explain with example requirements of different normal forms like 1NF, 2NF and 3NF.

SPPU : Dec.-18, End Sem, Marks 5

4.18 BCNF

SPPU ; May-18, Dec.-18,19, Marks 5

Boyce and Codd Normal Form is a **higher version** of the Third Normal form. This form deals with certain type of anomaly that is not handled by 3NF.

A 3NF table which **does not have multiple overlapping** candidate keys is said to be in BCNF.

Or in other words,

For a table to be in BCNF, following conditions must be satisfied :

- i) R must be in 3rd Normal Form
- ii) For each functional dependency ($X \rightarrow Y$), X should be a super key. In simple words if Y is a prime attribute then X can not be non prime attribute.

For example - Consider following table that represents that a student enrollment for the course -

Enrollment table

sid	Course	Teacher
1	C	Ankita
1	Java	Poonam
2	C	Ankita
3	C++	Supriya
4	C	Archana

From above table following observations can be made :

- One student can enroll for multiple courses. For example student with sid = 1 can enroll for C as well as Java.
- For each course, a teacher is assigned to the student.
- There can be multiple teachers teaching one course for example course C can be taught by both the teachers namely - Ankita and Archana.
- The candidate key for above table can be (sid, course), because using these two columns we can find,
- The above table holds following dependencies
 - (sid, course)->Teacher
 - Teacher->course
- The above table is not in BCNF because of the dependency teacher->course. Note that the teacher is not a superkey or in other words, teacher is a non prime attribute and course is a prime attribute and non-prime attribute derives the prime attribute.
- To convert the above table to BCNF we must decompose above table into student and course tables

Student

sid	Teacher
1	Ankita
1	Poonam
2	Ankita
3	Supriya
4	Archana

Course

Teacher	Course
Ankita	C
Poonam	Java
Ankita	C
Supriya	C++
Archana	C

Now the table is in BCNF

Example 4.18.1 Consider a relation(A,B,C,D) having following FDs.{AB->C, AB->D, C->A, B->D}. Find out the normal form of R.

Solution :

Step 1 : We will first find out the candidate key from the given FD.

$$(AB)^+ = \{ABCD\} = R$$

$$(BC)^+ = \{ABCD\} = R$$

$$(AC)^+ = \{AC\} \neq R$$

There is no involvement of D on LHS of the FD rules. Hence D can not be part of any candidate key. Thus we obtain two candidate keys **(AB)⁺** and **(BC)⁺**. Hence

$$\text{prime attributes} = \{A,B,C\}$$

$$\text{Non prime attributes} = \{D\}$$

Step 2 : Now, we will start checking from reverse manner, that means from BCNF, then 3NF, then 2NF.

Step 3 : For R being in BCNF for X->Y the X should be candidate key or super key.

From above FDs consider C->D in which C is not a candidate key or super key. Hence given relation is not in BCNF.

Step 4 : For R being in 3NF for X->Y either i) the X should be candidate key or super key or

ii) Y should be prime attribute. (For prime and non prime attributes refer step 1)

- o For AB->C or AB->D the AB is a candidate key. Condition for 3NF is satisfied.

- Consider $C \rightarrow A$. In this FD the C is not candidate key but A is a prime attribute. Condition for 3NF is satisfied.
- Now consider $B \rightarrow D$. In this FD, the B is not candidate key, similarly D is not a prime attribute. Hence condition for 3NF fails over here.

Hence given relation is not in 3NF.

Step 5 : For R being in 2NF following condition **should not occur.**

Let $X \rightarrow Y$, if X is a proper subset of candidate key and Y is a non prime attribute. This is a case of partial functional dependency.

For relation to be in 2NF there should not be any partial functional dependency.

- For $AB \rightarrow C$ or $AB \rightarrow D$ the AB is a **complete candidate key**. Condition for 2NF is satisfied.
- Consider $C \rightarrow A$. In this FD the C is not candidate key. Condition for 2NF is satisfied.
- Now consider $B \rightarrow D$. In this FD, the B is a part of candidate key (AB or BC), similarly D is not a prime attribute. That means partial functional dependency occurs here.

Hence condition for 2NF fails over here.

Hence given relation is not in 2NF.

Therefore we can conclude that the given relation **R is in 1NF.**

Example 4.18.2 Consider a relation $R(ABC)$ with following FD $A \rightarrow B$, $B \rightarrow C$ and $C \rightarrow A$. What is the normal form of R ?

Solution :

Step 1 : We will find the candidate key

$$(A)^+ = \{ABC\} = R$$

$$(B)^+ = \{ABC\} = R$$

$$(C)^+ = \{ABC\} = R$$

Hence A, B and C all are candidate keys

Prime attributes = {A,B,C}

Non prime attribute{}

Step 2 : For R being in BCNF for $X \rightarrow Y$ the X should be candidate key or super key.

From above FDs

- Consider $A \rightarrow B$ in which A is a candidate key or super key. Condition for BCNF is satisfied.
- Consider $B \rightarrow C$ in which B is a candidate key or super key. Condition for BCNF is satisfied.
- Consider $C \rightarrow A$ in which C is a candidate key or super key. Condition for BCNF is satisfied.

This shows that the given relation **R is in BCNF.**

Example 4.18.3 Consider table $R(A,B,C,D,E)$ with FDs as $A \rightarrow B$, $BC \rightarrow E$, and $ED \rightarrow A$. The table is in which normal form? Justify your answer.

Solution :

Step 1 : We will first find out the candidate keys for given relation R

$$(ACD)^+ = \{A,B,C,D,E\}$$

$$(BCD)^+ = \{A,B,C,D,E\}$$

$$(CDE)^+ = \{A,B,C,D,E\}$$

Step 2 : Let $A \rightarrow B$, the ACD is candidate key and A is a partial key, B is a prime attribute (i.e. it is also part of candidate key). Hence $A \rightarrow B$ is not a partial functional dependency.

Similarly in $BC \rightarrow E$ and $ED \rightarrow A$,

E and A are prime-attributes and hence both are not partial functional dependencies.

Hence R is in 2NF.

Step 3 : According to 3NF, every non-prime attribute must be dependent on the candidate key.

In the given functional dependencies, all dependent attributes are prime-attributes. Hence the relation R is in 3NF.

Step 4 : For R being in BCNF for $X \rightarrow Y$ the X should be candidate key or super key.

The table is not in BCNF, none of A, BC and ED contain a key.

Example 4.18.4 A college maintains details of its lecturers' subject area skills. These details comprise :

- Lecturer number
- Lecturer name
- Lecturer grade
- Department code
- Department name
- Subject code
- Subject name
- Subject level

Assume that each lecturer may teach many subjects but may not belong to more than one department.

Subject code, subject name and subject level are repeating fields.

Normalise this data to third normal form.

Solution :

Unnormalized form

Lecturer Number, Lecturer Name, Lecturer Grade, Department Code, Department Name, Subject Code, Subject Name, Subject Level

1NF

LecturerNumber	Lecturer Name	Lecturer Grade	Department Code	Department Name
----------------	---------------	----------------	-----------------	-----------------

<u>Lecturer Number</u>	<u>Subject Code</u>	Subject Name	Subject Level
------------------------	---------------------	--------------	---------------

2NF

Lecturer Number	<u>_Lecturer Name</u>	<u>_Lecturer Grade</u>	<u>_Department Code</u>	<u>_Department Name</u>
-----------------	-----------------------	------------------------	-------------------------	-------------------------

<u>Lecturer Number</u>	<u>Subject Code</u>
------------------------	---------------------

<u>Subject Code</u>	<u>_Subject Name</u>	<u>_Subject Level</u>
---------------------	----------------------	-----------------------

3NF

<u>Lecturer Number</u>	Lecturer Name	Lecturer Grade
<u>Department Code</u>	_Department Name	
<u>Lecturer Number</u>	<u>Subject Code</u>	
<u>Subject Code</u>	Subject Name	Subject Level

Example 4.18.5 Prove that any relational schema with two attributes is in BCNF.

Solution : Here, we will consider $R=\{A,B\}$ i.e. a relational schema with two attributes. Now various possible FDs are $A \rightarrow B$, $B \rightarrow A$.

From the above FDs

- Consider $A \rightarrow B$ in which A is a candidate key or super key. Condition for BCNF is satisfied.
- Consider $B \rightarrow A$ in which B is a candidate key or super key. Condition for BCNF is satisfied.
- Consider both $A \rightarrow B$ and $B \rightarrow A$ with both A and B is candidate key or super key. Condition for BCNF is satisfied.
- No FD holds in relation R. In this $\{A,B\}$ is candidate key or super key. Still condition for BCNF is satisfied.

This shows that any relation R is in BCNF with two attributes.

Example 4.18.6 Prove the statement "Every relation which is in BCNF is in 3NF but the converse is not true."

Solution : For a relations to be in 3NF

A table is said to be in the Third Normal Form when,

- i) It is in the Second Normal form. (i.e. it does not have partial functional dependency)
- ii) It doesn't have transitive dependency.

Or in other words

In other words 3NF can be defined as : A table is in 3NF if it is in 2NF and for each functional dependency $X \rightarrow Y$

at least one of the following conditions hold :

- iii) X is a super key of table
- iv) Y is a prime attribute of table

For a relation to be in BCNF

- 1) It should be in 3NF
- 2) A 3NF table which **does not have multiple overlapping** candidate keys is said to be in BCNF.

For proving that the table can be in 3NF but not in BCNF consider Following relation $R(\text{Student, Subject, Teacher})$. Consider following are FDs

(Subject, Student) \rightarrow Teacher

Because subject and student combination gives unique teacher.

Teacher \rightarrow Subject

Because each teacher teaches only Subject.

(Teacher, Student) \rightarrow Subject

- So, this relation is in 3NF as every non-key attribute is non-transitively fully functional dependent on the primary key.
- But it is not in BCNF. Because this is a case of **overlapping of candidate keys** because there are two composite candidate keys :
 - (Subject, Student)
 - (Teacher, Student)

And student is a common attribute in both the candidate keys.

So we need to normalize the above table to BCNF. For that purpose we must set Teacher to be a candidate key

The decomposition of above takes place as follows

$R_1(\text{Student, Teacher})$

$R_2(\text{Teacher, Subject})$

Now table is in 3NF, as well as in BCNF.

This show that the relation Every relation which is in BCNF is in 3NF but the converse is not true.

Example 4.18.7 Consider relational schema $R(A,B,C)$ with FD's $AB \rightarrow C$ and $C \rightarrow A$. Show that schema R is in 3NF but not in BCNF. Determine minimal keys of R .

Solution : A table is said to be in 3rd normal form when,

- i) It is in 2nd normal form.
- ii) It doesn't have transitive dependency or in other words L.H.S must be a candidate key or R.H.S is prime attribute.

For 1st Condition :

Step 1 : For 2nd Normal form condition :

Step A : It is in 1st normal form

Step B : Table not contain partial dependency i.e. all the non-prime attributes should be fully functionally dependent on candidate key.

Given relation $R(A, B, C)$

$$\text{FD} : AB \rightarrow C, C \rightarrow A$$

We will first find out the candidate key from the given FD.

$$(AB)^+ = \{ABC\} = R,$$

$$(CB)^+ = \{ABC\} = R,$$

$$(C)^+ = \{CA\} \neq R$$

They we obtain $(AB)^+$ and $(CB)^+$ candidate key.

Hence, prime attributes = $\{ABC\}$

None prime attributes = $\{C\}$

Hence, For $AB \rightarrow C$

AB is candidate key and its present at R.H.S for $C \rightarrow A$.

C is also candidate key and its present at R.H.S. There is no partial dependency. Hence it is in 2nd normal form.

Step 2 : Candidate key = $(AB)^+$ and $(CB)^+$

Prime attributes (ABC)

Non prime attribute () Null

For given DF's $AB \rightarrow C, C \rightarrow A$

In $AB \rightarrow C$, L.H.S is AB i.e. it is candidate key and In $C \rightarrow A$, L.H.S is C i.e. it is not candidate key but A is present at R.H.S i.e. it is prime attributes.

Hence, given relation is present in 3NF.

Now find given relation is present in BCNF or not.

Step 3 : Boyce / Codd normal form (BCNF) :

For a table to be in BCNF, following conditions must be satisfied.

i) R must be in 3rd normal form

ii) For each functional dependency ($X \rightarrow Y$), X should be a super key or candidate key.

In above 1st condition becomes true it is in 3rd normal form. Now check for 2nd condition. From given FDs

$AB \rightarrow C$ and $C \rightarrow A$ in which AB is candidate key but C is not candidate key. Hence 2nd condition becomes false so given relation is not present in BCNF.

Example 4.18.8 What is the difference between 3NF and BCNF ?

Solution :

Sr. No.	3NF	BCNF
1.	3NF stands for Third Normal Form.	BCNF stands for Boyce Codd Normal Form.
2.	The table is in 3NF if it is in 2NF and for each functional dependency $X \rightarrow Y$ at least following condition hold: (i) X is a superkey, (ii) Y is prime attribute of table.	The table is in BCNF if it is in 3rd normal form and for each relation $X \rightarrow Y$ X should be super key.
3.	3NF can be obtained without sacrificing all dependencies.	Dependencies may not be preserved in BCNF.
4.	Lossless decomposition can be achieved in 3NF.	Lossless decomposition is hard to obtain in BCNF.
5.	3NF can be achieved without losing any information from the old table.	For obtaining BCNF we may lose some information from old table.

Example 4.18.9 Consider following relation -

Book(book_title, authurname, book_type, listprice, author_affiliation, publisher)

Suppose the following functional dependencies exist –

book_title → *publisher, book_type*

book_type → *listprice*

authurname → *author_affiliation*

i) What normal form is the relation in ? Explain your answer.

ii) Apply normalization until you can not decompose the relations further. State the reasons behind each decomposition.

Solution :

- (i) The key for this relation is (Book_title, Authurname). This relation is in 1NF and not in 2NF as no attributes are Fully Functionally Dependent on the key.

Step 1 : We will first decompose it in **2nd Normal Form** by following decomposition.

R1(book_title, authurname)

R2(book_title, publisher, book_type, listprice)

R3(authurname, authoraffiliation)

Reason : This decomposition eliminates partial functional dependencies.

Step 2 : Now we will decompose it further to bring the table in **3rd Normal form**

R1(book_title, authurname)

R2(book_title, publisher, book_type, listprice)

R21(book_title, publisher, book_type)

R22(book_type, listprice)

R3(authurname, authoraffiliation)

Reason : This decomposition eliminates transitive functional dependency of listprice.

Review Question

1. Explain 3NF and BCNF. Also enlist their differences.

SPPU : May-18, Dec.-18,19, End Sem, Marks 5

Unit - III
Multiple Choice Questions

Q.1 Foreign key is one in which the ____ of one relation is referenced in another relation.

- a candidate key b constraint
 c primary key d foreign key

Q.2 The relationship between the two tables are created using ____.

- a primary key b check constraints
 c candidate key d foreign key

Q.3 Constraints are preferred methods for enforcing ____.

- a data abstraction b data access
 c data inheritance d data integrity

Q.4 A table that displays data redundancies yields ____ anomalies.

- a insertion b deletion
 c update d all of these

Q.5 The different classes of relations created by the technique for preventing modification anomalies are called ____.

- a normal forms b referential integrity constraints
 c functional dependencies d none of the above

Q.6 For some relations, changing the data can have undesirable consequences called ____.

- a referential integrity constraints b modification anomalies
 c normal forms d transitive dependencies

Q.7 The three rules that are used to find the logically implied functional dependencies are called as ____.

- a closure rules b Codd's rules
 c Armstrong's rule d none of these

Q.8 FD stands for ____.

- a Functional Dependency b Facilitate Dependency
 c Functional Data d Facilitate Data

Q.9 Which key is referencing a primary key in a table ?

- a Primary key b Candidate key
 c Foreign key d All of these

Q.10 If $K \rightarrow R$ then K is said to be the _____ of R.

- a candidate key b foreign key
 c super key d none of these

Q.11 If there is more than one key for relation schema in DBMS then each key in relation schema is classified as _____.

- a prime key b super key
 c candidate key d primary key

Q.12 The form of dependency in which the set of attributes that are neither a subset of any of the keys nor the candidate key is classified as _____.

- a transitive dependency b full functional dependency
 c partial dependency d prime functional dependency

Q.13 The property of normalization of relations which guarantees that functional dependencies are represented in separate relations after decomposition is classified as _____.

- a nonadditive join property
 b independency reservation property
 c dependency preservation property
 d additive join property

Q.14 $X \rightarrow Y$ is trivial if _____.

- a $X \subset Y$ b $Y \subset X$
 c $X \supseteq Y$ d none of these

Q.15 A table has fields F1, F2, F3, F4, F5 with the following functional dependencies

$F1 \rightarrow F3$,

$F2 \rightarrow F4$

$(F1, F2) \rightarrow F5$

In terms of normalization, this table is in _____.

- a 1NF b 2NF
 c 3NF d none of these

Q.16 Which normal form is considered adequate for relational database design ?

- a 2NF b 3NF
 c 4NF d BCNF

Q.17 Lossless, dependency preserving decomposition into ___ normal form is always possible.

- a 2NF b 3NF
 c 4NF d BCNF

Q.18 After normalization, the original table can be obtained by _____.

- a delete operation b join operation
 c retrieve operation d cascade operation

Q.19 FDs are the types of constraints that are based on _____.

- a data b tables
 c keys d none of these

Q.20 A property which ensures that each functional dependency is represented in some individual relational resulting after decomposition is called _____.

- a loss less join b dependency preservation
 c both a and b d none of the above

Q.21 F+ is called as ___ of F.

- a recursion b closure
 c next version d none of these

Q.22 A functional dependency must first satisfy the second normal form to satisfy the third normal form.

- a True b False

Answers Keys for Multiple Choice Questions :

Q.1	c	Q.2	d	Q.3	d	Q.4	d
Q.5	a	Q.6	b	Q.7	c	Q.8	a
Q.9	a	Q.10	c	Q.11	c	Q.12	a
Q.13	c	Q.14	a	Q.15	a	Q.16	b
Q.17	b	Q.18	b	Q.19	c	Q.20	c
Q.21	b	Q.22	a				



UNIT - IV

5

Database Transaction Management

Syllabus

Introduction to Database Transaction, Transaction states, ACID properties, Concept of Schedule, Serial Schedule. **Serializability** : Conflict and View, Cascaded Aborts, Recoverable and Non-recoverable Schedules. **Concurrency Control** : Lock-based, Time-stamp based Deadlock handling. **Recovery methods** : Shadow-Paging and Log-Based Recovery, Checkpoints. **Log-Based Recovery** : Deferred Database Modifications and Immediate Database Modifications.

Contents

Part I : Transaction Management

- 5.1 Introduction to Database Transaction
- 5.2 Transaction statesDec 17, Marks 8
- 5.3 ACID PropertiesDec.-18,19, May-18,19, Marks 8
- 5.4 Concept of Schedule
- 5.5 Serializability : Conflict and ViewDec.-17,18,19, May-18, Marks 9
- 5.6 Recoverable and Non-Recoverable Schedules

Part II : Concurrency Control

- 5.7 Concurrency Control
- 5.8 Need for Concurrency
- 5.9 Lock based ProtocolDec.-17,18,19, May-18,19, .. Marks 9
- 5.10 Time-stamp based ProtocolMay-19, Dec.-17, Marks 9
- 5.11 Deadlocks Handling
- 5.12 Recovery Concepts
- 5.13 Recovery Methods
- 5.14 Shadow-paging
- 5.15 Check Points
- 5.16 Log-based Recovery : Deferred and Immediate Update
.....May-18,19, Marks 8

Multiple Choice Questions

Part I : Transaction Management

5.1 Introduction to Database Transaction

- **Definition of Transaction :** A transaction can be defined as a **group of tasks** that form a single logical unit. For example - Suppose we want to withdraw ₹ 100 from an account then we will follow following operations :
 - 1) Check account balance
 - 2) If sufficient balance is present request for withdrawal.
 - 3) Get the money
 - 4) Calculate Balance = Balance – 100
 - 5) Update account with new balance.

The above mentioned **five steps** denote **one transaction**.

- A database is a collection of named data items.
- **Granularity or size of data items :** The size of data items can be a field, a record or a whole disk block.
- Basic operations on data item A are -

1. read_item(X) 2. write_item(X)

1. **read_item(X) :** This is a reading operation in which database item named X is read into a programming variable. We can name the program variable as X for simplification.
 2. **write_item(X) :** This operation writes the value of program variable X into the database item which is also named as X.
- Basic unit of data transfer from the disk to the computer main memory is one block.
 - **read_item(X) command includes the following steps :**
 - Step 1 :** Find the address of the disk block that contains item X.
 - Step 2 :** Copy that disk block into a buffer in main memory (if that disk block is not already in some main memory buffer).
 - Step 3 :** Copy item X from the buffer to the program variable named X.
 - **write_item(X) command includes the following steps :**
 - Step 1 :** Find the address of the disk block that contains item X.
 - Step 2 :** Copy that disk block into a buffer in main memory (if that disk block is not already in some main memory buffer).
 - Step 3 :** Copy item X from the program variable named X into its correct location in the buffer.
 - Step 4 :** Store the updated block from the buffer back to disk.

- **Example of sample transaction performing read and write operations**

```
read_item(X);
X=X+M;
Write_item(X);
```

- **Transaction Notations :**

The transaction notations focuses on read and write operations. For example – Following are two transactions denoted by T1 and T2

```
T1:b1;r1(X);w1(X);r1(Y);W1(Y);e1;
T2:b2;r2(X);r2(Y);e2
```

The r and w represents the read and write operations. The b1 and b2 represents the beginning and e1 and e2 represents ending of transaction.

5.2 Transaction States

SPPU : Dec.-17, Marks 8

Each transaction has following five states :

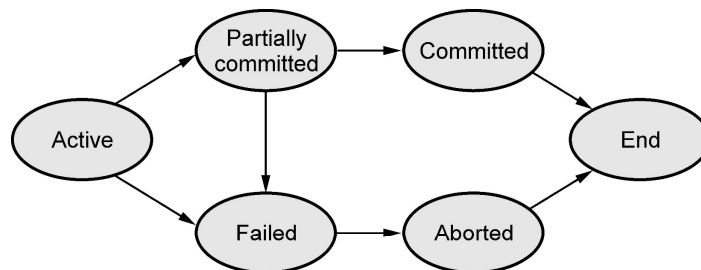


Fig. 5.2.1 Transaction states

- 1) **Active** : This is the first state of transaction. For example : Insertion, deletion or update of record is done here. But data is not saved to database.
- 2) **Partially committed** : When a transaction executes its final operation, it is said to be in a partially committed state.
- 3) **Failed** : A transaction is said to be in a failed state if any of the checks made by the database recovery system fails. A failed transaction can no longer proceed further.
- 4) **Aborted** : If a transaction is failed to execute, then the database recovery system will make sure that the database is in its previous consistent state. If not, it brings the database to consistent state by aborting or rolling back the transaction.
- 5) **Committed** : If a transaction executes all its operations successfully, it is said to be committed. This is the last step of a transaction, if it executes without fail.

Example 5.2.1 Define a transaction. Then discuss the following with relevant examples :

1. A read only transaction
2. A read write transaction
3. An aborted transaction

Solution :

1) Read only transaction

T1
Read(A)
Read(B)
Display(A – B)

2) A read write transaction

T1
Read(A)
A=A+100
Write(A)

3)

T1	T2	Description
Read(A)		Assume A=100
A=A+50		A=150
Write(A)		
	Read(A)	A=150
	A=A+100	A=250
RollBack		A=100 (restore back to original value which is before Transaction T1)
	Write(A)	

Review Question

1. Transaction during its execution should be in one of the different states at any point of time, explain the different states of transactions during its execution.

SPPU : Dec 17, End Sem, Marks 8

5.3 ACID Properties**SPPU : Dec.-18,19, May-18,19, Marks 8****1) Atomicity :**

- This property states that each transaction must be considered as a **single unit and must be completed fully or not completed at all.**
- No transaction in the database is left half completed.
- Database should be in a state either before the transaction execution or after the transaction execution. It **should not be in a state 'executing'**.
- For example - In above mentioned withdrawal of money transaction all the five steps must be completed fully or none of the step is completed. Suppose if transaction gets failed after step 3, then the customer will get the money but the balance will not be updated accordingly. The state of database should be either at before ATM withdrawal (i.e customer without withdrawn money) or after ATM withdrawal (i.e. customer with money and account updated). This will make the system in consistent state.

2) Consistency :

- The database must remain in consistent state after performing any transaction.
- **For example :** In ATM withdrawal operation, the balance must be updated appropriately after performing transaction. Thus the database can be in consistent state.

3) Isolation :

- In a database system where **more than one transaction** are being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system.
- No transaction will affect the existence of any other transaction.
- **For example :** If a bank manager is checking the account balance of particular customer, then manager should see the balance either before withdrawing the money or after withdrawing the money. This will make sure that each individual transaction is completed and any other dependent transaction will get the consistent data out of it. Any failure to any transaction will not affect other transaction in this case. Hence it makes all the transactions consistent.

4) Durability :

- The database should be **strong enough** to handle any **system failure**.
- If there is any set of insert /update, then it should be able to handle and commit to the database.
- If there is any **failure**, the database should be able to **recover** it to the consistent state.
- For example : In ATM withdrawal example, if the system failure happens after customer getting the money then the system should be strong enough to update Database with his new balance, after system recovers. For that purpose the system has to keep the **log of each transaction and its failure**. So when the system recovers, it should be able to know when a system has failed and if there is any pending transaction, then it should be updated to Database.

Review Question

1. State and explain in brief the ACID properties. During execution of transaction, a transaction passes through several states, until it finally commits or aborts. List all possible sequences of states through which a transaction may pass. Explain why each state transition occurs.

SPPU : May-18,19, Dec.-18,19, End Sem, Marks 8

5.4 Concept of Schedule

Schedule is an order of multiple transactions executing in concurrent environment. Following Fig. 5.4.1 represents the types of schedules.

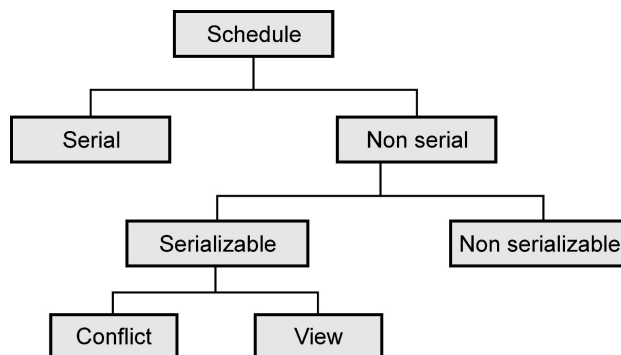


Fig. 5.4.1 : Types of schedule

Serial schedule : The schedule in which the transactions execute one after the other is called **serial schedule**. It is consistent in nature. For example : Consider following two transactions T1 and T2.

T1	T2
R(A)	
W(A)	
R(B)	
W(B)	
	R(A)
	W(A)
	R(B)
	W(B)

All the operations of transaction T1 on data items A and then B executes and then in transaction T2 all the operations on data items A and B execute. The **R** stands for read operation and **W** stands for write operation.

Non serial schedule : The schedule in which operations present within the transaction are intermixed. This may lead to conflicts in the result or inconsistency in the resultant data.

For example -

Consider following two transactions,

T1	T2
R(A)	
W(A)	
	R(A)
	W(B)
R(A)	
W(B)	
	R(B)
	W(B)

The above transaction is said to be non serial which result in inconsistency or conflicts in the data.

5.5 Serializability : Conflict and View

SPPU : Dec.-17,18,19, May-18, Marks 9

- When multiple transactions run concurrently, then it may lead to inconsistency of data (i.e. change in the resultant value of data from different transactions).

- Serializability is a concept that helps to identify which non serial schedule and find the transaction equivalent to serial schedule.

T1	A	B	T2
Initial Value	100	100	
A=A - 10			
W(A)			
B=B+10			
W(B)			
	90	110	
			A=A-10
			W(A)
	80	110	

- In above transactions initially T1 will read the values from database as A = 100, B = 100 and modify the values of A and B. But transaction T2 will read the modified value i.e. 90 and will modify it to 80 and perform write operation. Thus at the end of transaction T1 value of A will be 90 but at end of transaction T2 value of A will be 80. Thus conflicts or inconsistency occurs here. This sequence can be converted to a sequence which may give us consistent result. This process is called **serializability**.

Difference between serial schedule and serializable schedule

Sr. No.	Serial schedule	Serializable schedule
1	No concurrency is allowed in serial schedule.	Concurrency is allowed in serializable schedule.
2	In serial schedule, if there are two transactions executing at the same time and no interleaving of operations is permitted, then following can be the possibilities of execution - i) Execute all the operations of transactions T ₁ in a sequence and then execute all the operations of transactions T ₂ in a sequence. ii) Execute all the operations of transactions T ₂ in a sequence and then execute all the operations of transactions T ₁ in a sequence.	In serializable schedule, if there are two transactions executing at the same time and interleaving of operations is allowed there can be different possible orders of executing an individual operation of the transactions.

Example of serial schedule		Example of serializable schedule	
T ₁	T ₂	T ₁	T ₂
Read(A)		Read(A)	
A=A-50		A=A-50	
Write(A)		Write(A)	
Read(B)			Read(B)
B=B+100			B=B+100
Write(B)			Write(B)
	Read(A)	Read(B)	
	A=A+10	Write(B)	
	Write(A)		

- There are two types of serializabilities : Conflict serializability and view serializability.

5.5.1 Conflict Serializability

Definition : Suppose T₁ and T₂ are two transactions and I₁ and I₂ are the instructions in T₁ and T₂ respectively. Then these two transactions are said to be conflict serializable, if both the instruction access the data item d, and at least one of the instruction is write operation.

What is conflict ? : In the definition **three conditions** are specified for a conflict in conflict serializability -

- 1) There should be **different transactions**
- 2) The operations must be performed on **same data** items
- 3) **One of the operation** must be the **Write (W)** operation.

- We can test a given schedule for conflict serializability by constructing a **precedence graph** for the schedule, and by searching for absence of cycles in the graph.
- Precedence graph is a directed graph, consisting of G = (V,E) where V is set of vertices and E is set of edges. The set of vertices consists of all the transactions participating in the schedule. The set of edges consists of all edges T_i → T_j for which one of three conditions holds :

1. T_i executes write(Q) before T_j executes read(Q).

2. T_i executes read(Q) before T_j executes write(Q).
 3. T_i executes write(Q) before T_j executes write(Q).
- A serializability order of the transactions can be obtained by finding a linear order consistent with the partial order of the precedence graph. This process is called **topological sorting**.

Testing for serializability

- Following method is used for testing the serializability : To test the conflict serializability we can draw a graph $G = (V, E)$ where $V =$ Vertices which represent the number of transactions.

$E =$ Edges for conflicting pairs.

Step 1 : Create a node for each transaction.

Step 2 : Find the conflicting pairs (RW, WR, WW) on the same variable (or data item) by different transactions.

Step 3 : Draw edge for the given schedule. Consider following cases

1. T_i executes write(Q) before T_j executes read(Q), then draw edge from T_i to T_j .
2. T_i executes read(Q) before T_j executes write(Q) , then draw edge from T_i to T_j .
3. T_i executes write(Q) before T_j executes write(Q), , then draw edge from T_i to T_j .

Step 4 : Now, if precedence graph is cyclic then it is a non conflict serializable schedule and if the precedence graph is acyclic then it is conflict serializable schedule.

Example 5.5.1 Consider the following two transactions and schedule (time goes from top to bottom). Is this schedule conflict-serializable ? Explain why or why not.

T_1	T_2
R(A)	
W(A)	
	R(A)
	R(B)
R(B)	
W(B)	

Solution :

Step 1 : To check whether the schedule is conflict serializable or not we will check from top to bottom. Thus we will start reading from top to bottom as,

$T_1 : R(A) \rightarrow T_1 : W(A) \rightarrow T_2 : R(A) \rightarrow T_2 : R(B) \rightarrow T_1 : R(B) \rightarrow T_1 : W(B)$

Step 2 : We will find **conflicting operations**. Two operations are called as **conflicting operations** if all the following conditions hold true for them -

- i) Both the **operations belong to different transactions**.
- ii) Both the **operations are on same data item**.
- iii) **At least one** of the two operations is a **write operation**.

From above given example in the top to bottom scanning we find the conflict as

$T_1 : W(A) \rightarrow T_2 : R(A)$.

- i) Here note that there are two different transactions T_1 and T_2 ,
- ii) Both work on same data item i.e. A and
- iii) One of the operation is write operation.

Step 3 : We will build a precedence graph by drawing one node from each transaction. In above given scenario as there are two transactions, there will be two nodes namely T_1 and T_2 .



Fig. 5.5.1

Step 4 : Draw the edge between conflicting transactions. For example in above given scenario, the conflict occurs while moving from $T_1 : W(A)$ to $T_2 : R(A)$. Hence edge must be from T_1 and T_2 .

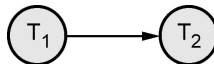


Fig. 5.5.2

Step 5 : Repeat the step 4 while reading from top to bottom. Finally the precedence graph will be as follows

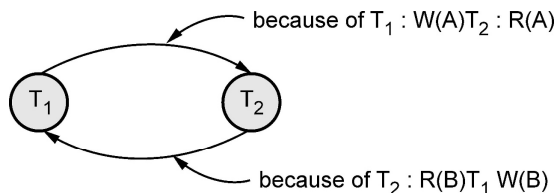


Fig. 5.5.3 : Precedence graph

Step 6 : Check if any cycle exists in the graph. Cycle is a path using which we can start from one node and reach to the same node. If the is cycle found then schedule is not conflict serializable. In the step 5 we get a graph with cycle, that means given schedule is not conflict serializable.

Example 5.5.2 Check whether following schedule is conflict serializable or not. If it is not conflict serializable then find the serializability order.

T1	T2	T3
R(A)		
	R(B)	
		R(B)
	W(B)	
W(A)		
		W(A)
	R(A)	
	W(A)	

Solution :

Step 1 : We will read from top to bottom, and build a precedence graph for conflicting entries. We will build a precedence graph by drawing one node from each transaction. In above given scenario as there are three transactions, there will be two nodes namely T1 T2, and T3

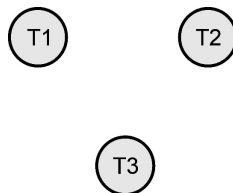
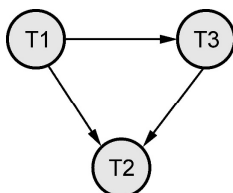


Fig. 5.5.4

Step 2 : The conflicts are found as follows –

T1	T2	T3
R(A)		ⓐ
	R(B)	
ⓑ	ⓓ	R(B)
	W(B)	
W(A)		W(A)
ⓔ	R(A)	
	W(A)	

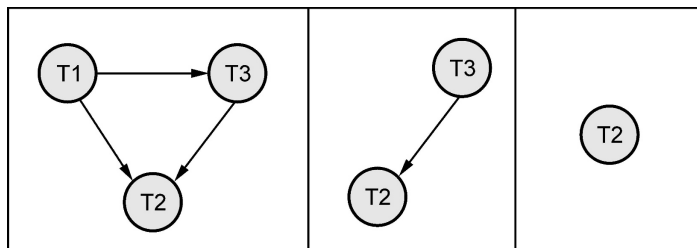
Step 3 : The precedence graph will be as follows –



Step 4 : As there is **no cycle** in the precedence graph, the given sequence is **conflict serializable**. Hence we can convert this non serial schedule to serial schedule. For that purpose we will follow these steps to find the serializable order.

Step 5 : A **serializability order** of the transactions can be obtained by finding a linear order consistent with the partial order of the precedence graph. This process is called topological sorting.

Step 6 : Find the vertex which has no incoming edge which is T1. If we delete T1 node then T3 is a node that has no incoming edge. If we delete T3, then T2 is a node that has no incoming edge.



Thus the nodes can be deleted in a order T1, T3 and T2. Hence the order will be T1-T3-T2

Example 5.5.3 Check whether the below schedule is conflict serializable or not.

$\{B2, r2(X), b1, r1(X), W1(X), r1(Y), W1(Y), W2(X), e1, C1, e2, C2\}$

Solution : b2 and b1 represents **begin transaction 2** and **begin transaction 1**. Similarly, e1 and e2 represents **end transaction 1** and **end transaction 2**.

We will rewrite the schedule as follows -

T ₁	T ₂
	r2(X)
r1(X)	
W1(X)	
r1(Y)	
W1(Y)	
	W2(X)

Step 1 : We will find conflicting operations. Two operations are called as conflicting operations if all the following conditions hold true for them -

- i) Both the **operations belong to different transactions**.
- ii) Both the operations are on **same data item**.
- iii) **At least one** of the two operations is a **write operation**

The conflicting entries are as follows -

T ₁	T ₂
	R ₂ (X)
R ₁ (X)	
W ₁ (X)	
R ₁ (Y)	
W ₁ (Y)	
	W ₂ (X)

Step 2 : Now we build a precedence graph for conflicting entries.

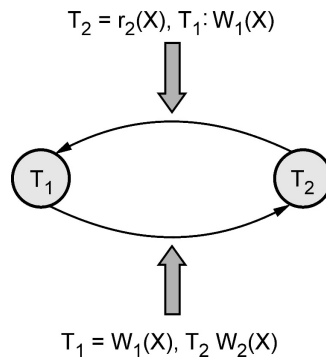


Fig. 5.5.5

As there are two transactions only two nodes are present in the graph.

Step 3 : We get a graph with cycle, that means given schedule is **not conflict serializable**.

Example 5.5.4 Consider the three transactions T1, T2, and T3 and schedules S1 and S2 given below. Determine whether each schedule is serializable or not ? If a schedule is serializable write down the equivalent serial schedule(S).

T1: R1(x) R1(z);W1(x);

T2: R2(x);R2(y);W2(z);W2(y)

T3:R3(x);R3(y);W3(y);

S1: R1(x);R2(z);R1(z);R3(x);R3(y);W1(x);W3(y);R2(y);W2(z);W2(y);

S2: R1(x);R2(z);R3(x);R1(z);R2(y);R3(y);W1(x);W2(z);W3(y);W2(y);

Solution :

Step 1 : We will represent the schedule S1 as follows

T1	T2	T3
R1(x)		
	R2(z)	
R1(z)		
		R3(x)
		R3(y)
W1(x)		
		W3(y)
	R2(y)	
	W2(z)	
	W2(y)	

Step (a) : We will find conflicting operations. Two operations are called as conflicting operations if all the following conditions hold true for them -

- i) Both the **operations belong to different transactions.**
- ii) Both the operations are on **same data item.**
- iii) **At least one** of the two operations is a **write operation**

T1	T2	T3
R1(x)		
	R2(z)	
R1(z)		
		R3(x)
		R3(y)
W1(x)		
		W3(y)
	R2(y)	
	W2(z)	
	W2(y)	

Step (b) : Now we will draw precedence graph as follows -

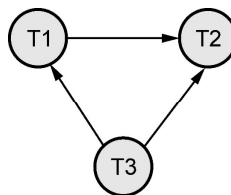


Fig. 5.5.6 Precedence graph

As there is **no cycle** in the precedence graph, the given sequence is **conflict serializable**. Hence we can convert this non serial schedule to serial schedule. For that purpose we will follow these steps to find the serializable order.

Step (c) : A **serializability order** of the transactions can be obtained by finding a linear order consistent with the partial order of the precedence graph. This process is called **topological sorting**.

Step (d) : Find the vertex which has no incoming edge which is T3. If we delete T3, then T1 is the edge that has no incoming edge. Finally find the vertex having no outgoing edge which is T2. Hence the order will be T3- T1-T2

Step 2 : We will represent the schedule S2 as follows -

T1	T2	T3
R1(x)		
	R2(z)	
		R3(x)
R1(z)		
	R2(y)	
		R3(y)
W1(x)		
	W2(z)	
		W3(y)
	W2(y)	

We will find **conflicting operations**. Two operations are called as conflicting operations if all the following conditions hold true for them -

- i) Both the **operations belong to different transactions**.
- ii) Both the operations are on **same data item**.
- iii) **At least one** of the two operations is a **write operation**

The conflicting entries are as follows -

T1	T2	T3
R1(x)		
	R2(z)	
		R3(x)
R1(z)		
	R2(y)	
		R3(y)
W1(x)		
	W2(z)	
		W3(y)
	W2(y)	

Step (b) : Now we will draw precedence graph as follows -

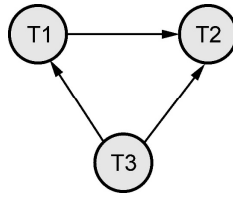


Fig. 5.5.7 Precedence Graph

As there is **no cycle** in the precedence graph, the given sequence is **conflict serializable**. Hence we can convert this non serial schedule to serial schedule. For that purpose we will follow these steps to find the serializable order.

Step (c) : A serializability order of the transactions can be obtained by finding a linear order consistent with the partial order of the precedence graph. This process is called topological sorting.

Step (d) : Find the vertex which has no incoming edge which is T3. Finally find the vertex having no outgoing edge which is T2. So in between them is T1. Hence the order will be T3- T1-T2

Example 5.5.5 Consider the transaction, transaction and transaction are any hypothetical transactions working on data item Q. Schedule explaining the execution of and are given below. Decide whether following schedule is conflict serializable or not ? Justify your answer.

T3	T4	T6
read (Q)		
	write (Q)	
write (Q)		
		write (Q)

SPPU : Dec 17, End Sem, Marks 9

Solution : We will find the conflicting entries as follows -

T3	T4	T6
read (Q)		
	write (Q)	
write (Q)		
		write (Q)

Arrows in the table above indicate conflicts: from T3 read(Q) to T4 write(Q), from T4 write(Q) to T3 write(Q), and from T4 write(Q) to T6 write(Q).

The precedence graph is as follows –

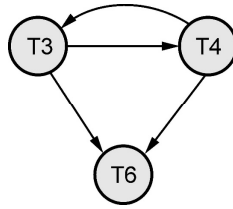


Fig. 5.5.8 Precedence Graph

As there exists **cycle**, the given schedule is **not conflict serializable**.

Example 5.5.6 Explain the concept of conflict serializability. Decide whether following schedule is conflict serializable or not. Justify your answer.

T1	T2
read (A)	
write (A)	
	read (A)
	write (A)
read (B)	
write (B)	
	read (B)
	write (B)

SPPU : May 18, End Sem, Marks 9

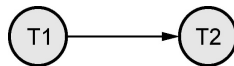
Solution :

Step 1 : We will read from top to bottom, and build a precedence graph for conflicting entries.

The conflicting entries are as follows –

T1	T2
read (A)	
write (A)	
	read (A)
	write (A)
read (B)	
write (B)	
	read (B)
	write (B)

Step 2 : Now we will build precedence graph as follows –



Step 3 : There is **no cycle** in the precedence graph. That means this schedule is conflict serializable. Hence we can convert this non serial schedule to serial schedule. For that purpose we will follow the following steps to find the serializable order.

- 1) Find the vertex which has no incoming edge which is T1.
- 2) Then find the vertex having no outgoing edge which is T2. In between them there is no other transaction.
- 3) Hence the order will be T1-T2

5.5.2 View Serializability

- If a given schedule is found to be view equivalent to some serial schedule, then it is called as a **view serializable schedule**.
- **View Equivalent Schedule :** Consider two schedules S1 and S2 consisting of transactions T1 and T2 respectively, then schedules S1 and S2 are said to be view equivalent schedule if it satisfies following three conditions :
 - If transaction T1 reads a data item A **from** the database initially in schedule S2, then in schedule S2 also, T1 must perform the initial read of the data item X from the database. This is same for all the data items. In other words - the initial reads must be same for all data items.
 - If data item A has been updated at last by transaction Ti in schedule S1, then in schedule S2 also, the data item A must be updated at last by transaction Ti.
 - If transaction Ti reads a data item that has been updated by the transaction Tj in schedule S1, then in schedule S2 also, transaction Ti must read the same data item that has been updated by transaction Tj. In other words the Write-Read sequence must be same.
 - **Difference between conflict serializability and view serializability**

Conflict serializability	View serializability
Every conflict serializable is view serializable.	Every view serializable schedule is not necessarily conflict serializable.
It is easy to test conflict serializability.	It is complex to test view serializability.

Steps to check whether the given schedule is view serializable or not

Step 1 : If the schedule is conflict serializable then it is surely view serializable because conflict serializability is a restricted form of view serializability.

Step 2 : If it is not conflict serializable schedule then check whether there exist any blind write operation. The blind write operation is a write operation without reading a value. If there does not exist any blind write then that means the given schedule is not view serializable. In other words if a blind write exists then that means schedule may or may not be view conflict.

Step 3 : Find the view equivalence schedule

Example 5.5.7 Consider the following schedules for checking if these are view serializable or not.

T ₁	T ₂	T ₃
		W(C)
	R(A)	
	W(B)	R(B)
R(C)		
		W(B)
W(B)		

Solution :

- i) The initial read operation is performed by T₂ on data item A or by T₁ on data item C. Hence we will begin with T₂ or T₁. We will choose T₂ at the beginning.
- ii) The final write is performed by T₁ on the same data item B. Hence T₁ will be at the last position.
- iii) The data item C is written by T₃ and then it is read by T₁. Hence T₃ should appear before T₁. Thus we get the order of schedule of view serializability as T₂ – T₁ – T₃

Example 5.5.8 Consider the following schedules for checking if these are view serializable or not.

```

T1 : read(A)
    read(B)
    if A=0 then B:=B+1;
    write(B)
T2 : read(B);
    read(A);
    if B=0 then A:=A+1;
    write(A)

```

Let consistency requirement be $A=0 \vee B=0$ with $A=B=0$ the initial values.

- 1) Show that every serial execution involving these two transactions preserves the consistency of the Database ?
- 2) Show a concurrent execution of T_1 and T_2 that produces a non serializable schedule ?
- 3) Is there a concurrent execution of T_1 and T_2 that produces a serializable schedule ?

Solution : 1) There are two possible executions : $T_1 \rightarrow T_2$ or $T_2 \rightarrow T_1$

Consider case $T_1 \rightarrow T_2$ then

A	B
0	0
0	1
0	1

$A \vee B = A \text{ OR } B = F \vee T = T$. This means consistency is met.

Consider case $T_2 \rightarrow T_1$ then

A	B
0	0
1	0
1	0

$A \vee B = A \text{ OR } B = F \vee T = T$. This means consistency is met.

- 2) The concurrent execution means interleaving of transactions T_1 and T_2 . It can be

T_1	T_2
R(A)	
	R(B)
	R(A)
R(B) If A=0 then B=B+1	If B=0 then A=A+1 W(A)
W(B)	

This is a non-serializable schedule.

- 3) There is no concurrent execution resulting in a serializable schedule.

Example 5.5.9 Consider the following schedules. The actions are listed in the order they are scheduled, and prefixed with the transaction name.

$S_1 : T_1 : R(X), T_2 : R(X), T_1 : W(Y), T_2 : W(Y) T_1 : R(Y), T_2 : R(Y)$

$S_2 : T_3 : W(X), T_1 : R(X), T_1 : W(Y), T_2 : R(Z), T_2 : W(Z) T_3 : R(Z)$

For each of the schedules, answer the following questions :

- What is the precedence graph for the schedule ?
- Is the schedule conflict-serializable ? If so, what are all the conflict equivalent serial schedules ?
- Is the schedule view-serializable ? If so, what are all the view equivalent serial schedules ?

Solution :

i) We will find **conflicting operations**. Two operations are called as conflicting operations if all the following conditions hold true for them -

- Both the **operations belong to different transactions**.
- Both the operations are on **same data item**.

At least one of the two operations is a write operation

For S_1 : From above given example in the top to bottom scanning we find the conflict as

- $T_1 : W(Y), T_2 : W(Y)$ and
- $T_2 : W(Y), T_1 : R(Y)$

Hence we will build the precedence graph. Draw the edge between conflicting transactions. For example in above given scenario, the conflict occurs while moving from $T_1 : W(Y)$ to $T_2 : W(Y)$. Hence edge must be from T_1 to T_2 . Similarly for second conflict, there will be the edge from T_2 to T_1 .



Fig. 5.5.9 Precedence graph for S_1

For S_2 : The conflicts are

- $T_3 : W(X), T_1 : R(X)$
- $T_2 : W(Z) T_3 : R(Z)$

Hence the precedence graph is as follows -

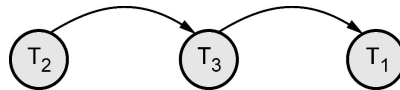


Fig. 5.5.10 Precedence graph for S_2

ii)

- S_1 is **not conflict-serializable** since the dependency graph has a cycle.
- S_2 is conflict-serializable as the dependency graph is acyclic. The order $T_2-T_3-T_1$ is the only equivalent **serial order**.

iii)

- S_1 is not view serializable.
- S_2 is trivially view-serializable as it is conflict serializable. The only serial order allowed is $T_2-T_3-T_1$.

Example 5.5.10 Check whether following schedule is view serializable or not. Justify your answer. (Note : T1 and T2 are transactions). Also explain the concept of view equivalent schedules and conflict equivalent schedule considering the example schedule given below

T1	T2
read (A) A: = A - 50	
write (A)	
	read (A) temp: = A*0.1 A: = A - temp
	write (A)
read (B) B: = B + 50	
write (B)	
	read (B) B: = B + temp
	write (B)

SPPU : Dec 18,19, End Sem, Marks 9

Solution :

Step 1 : We will first find if the given schedule is conflict serializable or not. For that purpose, we will find the conflicting operations. These are as shown below -

T1	T2
read (A)	
A := A - 50	
write (A)	
	read (A)
	temp := A*0.1
	A := A - temp
	write (A)
read (B)	
B := B + 50	
write (B)	
	read (B)
	B := B + temp
	write (B)

The precedence graph is as follows –

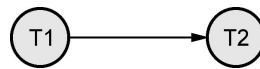


Fig. 5.5.11 Precedence graph

As there exists no cycle, the schedule is conflict serializable. The possible serializability order can be T1-T2

Now we check it for view serializability. As we get the serializability order as T1 – T2, we will find the view equivalence with the given schedule as serializable schedule.

Let S be the given schedule as given in the problem statement. Let the serializable schedule is S'={T1,T2}. These two schedules are represented as follows

T1	T2
read (A)	
A := A - 50	
write (A)	
	read (A)
	temp := A*0.1
	A := A - temp
	write (A)
read (B)	
B := B + 50	
write (B)	
	read (B)
	B := B + temp
	write (B)

Schedule S

T1	T2
read (A)	
A := A - 50	
write (A)	
read (B)	
B := B + 50	
write (B)	
	read (A)
	temp := A*0.1
	A := A - temp
	write (A)
	read (B)
	B := B + temp
	write (B)

Schedule S'

Now we will check the equivalence between them using following conditions –

1. Initial Read

In **schedule S** initial read on A is in transaction T1. Similarly initial read on B is in transaction T1.

Similarly in **schedule S'**, initial read on A is in transaction T1. Similarly initial read on B is in transaction T1.

2. Final Write

In **schedule S** final write on A is in transaction T2. Similarly final write on B is in transaction T2.

In **schedule S'** final write on A is in transaction T2. Similarly final write on B is in transaction T2

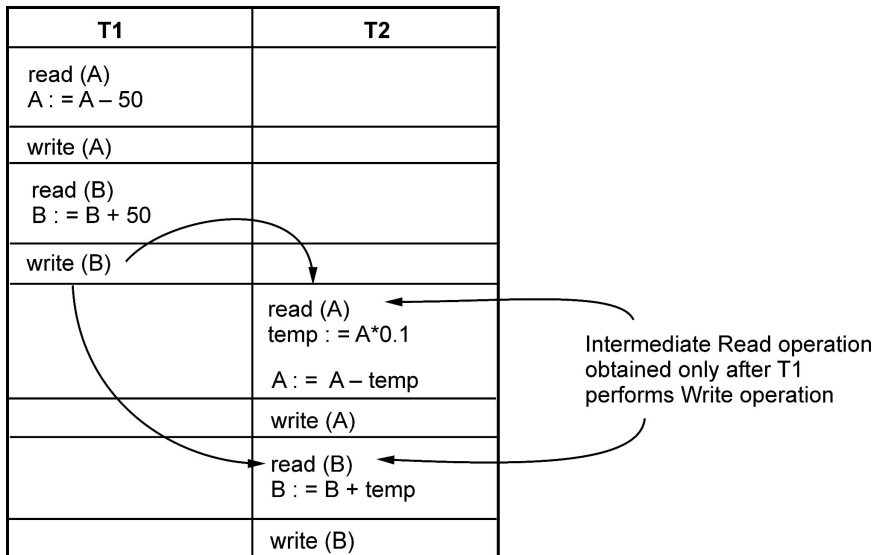
3. Intermediate Read

Consider **schedule S** for finding intermediate read operation.

T1	T2
read (A) A := A - 50	
write (A)	
	read (A) temp := A*0.1 A := A - temp write (A)
read (B) B := B + 50	
write (B)	
	read (B) B := B + temp write (B)

Intermediate Read obtained only after T1 performs Write operation

Similarly consider **schedule S'** for finding intermediate read operation.



In both the **schedules S and S'**, the intermediate read operation is performed by T2 only after T1 performs write operation.

Thus all the above three conditions get satisfied. Hence given schedule is **view serializable**.

Review Question

1. Explain the concept of conflict serializability with example. Since every conflict-serializable schedule is view serializable, why do we emphasize conflict serializability rather than view serializability?

SPPU : Dec 18,19, End Sem, Marks 8

5.6 Recoverable and Non-recoverable Schedules

- The serializable schedule can be made consistent by applying conflict serializability or view serializability.
- The serializable order makes a transaction isolation. But during the execution of concurrent transactions in a given schedule, some of the transaction may get failed(may be due to hardware or software failure)
- If the transaction gets failed we need to undo the effect of this transaction to ensure the atomicity property of transaction. This makes the schedule acceptable even after the failure.
- The schedule can be made acceptable by two techniques namely - Recoverable Schedule and Cascadeless schedule.

5.6.1 Recoverable Schedule

Definition : A recoverable schedule is one where, for each pair of transactions T_i and T_j such that T_j reads a data item previously written by T_i , the commit operation of T_i appears before the commit operation of T_j .

For example : Consider following schedule, consider $A = 100$

T_1	T_2
R(A)	
$A=A+50$	
W(A)	
	R(A)
	$A=A-20$
	W(A)
	Commit
Some transaction...	
Commit	

Failure

- The above schedule is inconsistent if failure occurs after the commit of T_2 .
- It is because T_2 is **dependable transaction** on T_1 . A transaction is said to be dependable if it contains a **dirty read**.
- The dirty read is a situation in which one transaction reads the data immediately after the write operation of previous transaction.

T_1	T_2
R(A)	
$A=A+50$	
W(A)	
	R(A)
	$A=A-20$
	W(A)
	Commit
Commit	

Dirty read

- Now if the dependable transaction i.e. T_2 is committed first and then failure occurs then if the transaction T_1 makes any changes then those changes will not be known to the T_2 . This leads to non recoverable state of the schedule.
- To make the schedule recoverable we will apply the rule that - commit the independent transaction before any dependable transaction.
- In above example independent transaction is T_1 , hence we must commit it before the dependable transaction i.e. T_2 .
- The recoverable schedule will then be -

T_1	T_2
R(A)	
$A=A+50$	
W(A)	
	R(A)
	$A=A-20$
	W(A)
Commit	
	Commit

5.6.2 Cascadeless Schedule

Definition : If in a schedule, a transaction is not allowed to read a data item until the last transaction that has written that data item is committed or aborted, then such a schedule is known as a cascadeless schedule.

The cascadeless schedule allows only **committed Read** operation. For example :

T_1	T_2	T_3
R(A)		
$A=A+50$		
W(A)		
Commit		
	R(A)	
	$A=A-20$	
	W(A)	
	Commit	
		R(A)
		W(A)

In above schedule at any point if the failure occurs due to commit operation before every Read operation of each transaction, the schedule becomes recoverable and atomicity can be maintained.

Part II : Concurrency Control

5.7 Concurrency Control

- One of the fundamental properties of a transaction is **isolation**.
- When several transactions execute concurrently in the database, however, the isolation property may no longer be preserved.
- A database can have multiple transactions running at the same time. This is called **concurrency**.
- To preserve the isolation property, the system must control the interaction among the concurrent transactions; this control is achieved through one of a variety of mechanisms called **concurrency control schemes**.
- **Definition of concurrency control** : A mechanism which ensures that simultaneous execution of more than one transactions does not lead to any database inconsistencies is called **concurrency control mechanism**.
- The concurrency control can be achieved with the help of various protocols such as - lock based protocol, deadlock handling, multiple granularity, timestamp based protocol, and validation based protocols.

5.8 Need for Concurrency

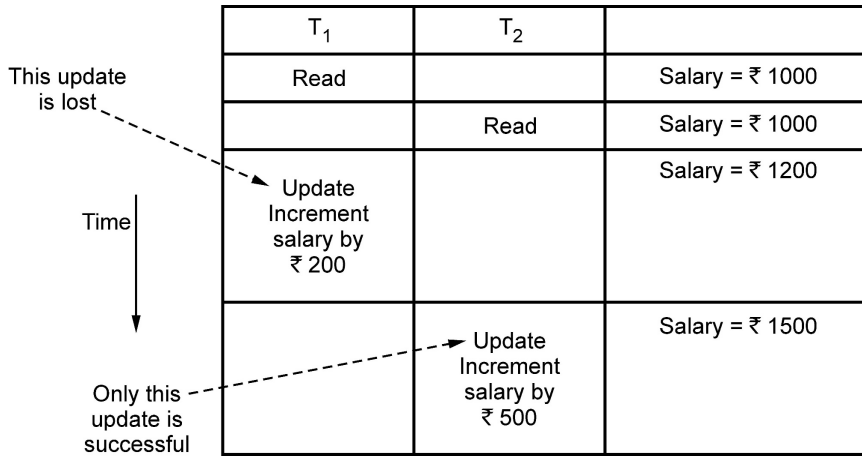
- Following are the purposes of concurrency control –
 - **To ensure isolation**
 - **To resolve read-write or write-write conflicts**
 - **To preserve consistency of database**
- Concurrent execution of transactions over shared database creates several data integrity and consistency problems - these are

1) Lost Update Problem : This problem occurs when two transactions that access the same database items have their operations interleaved in a way that makes the value of some database item incorrect.

For example - Consider following transactions

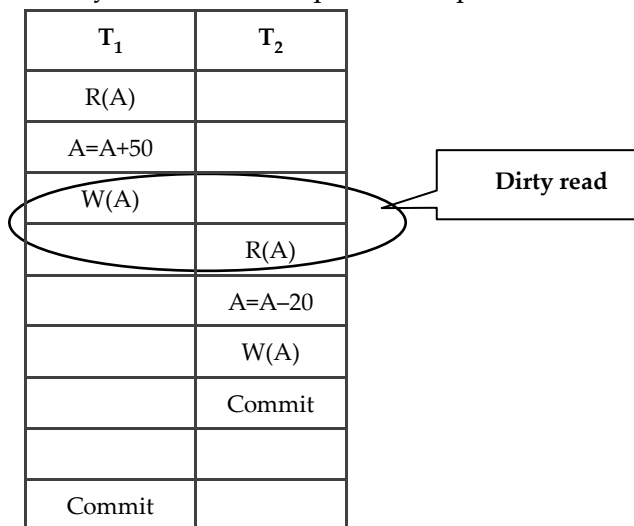
1) Salary of employee is read during transaction T1.

- 2) Salary of employee is read by another transaction T_2 .
- 3) During transaction T_1 , the salary is incremented by ₹ 200
- 4) During transaction T_2 , the salary is incremented by ₹ 500



The result of the above sequence is that the update made by transaction T_1 is completely lost. Therefore this problem is called as lost update problem.

2) Dirty Read or Uncommitted Read Problem : The dirty read is a situation in which one transaction reads the data immediately after the write operation of previous transaction



For example - Consider following transactions -
Assume initially salary is = ₹ 1000

	T ₁	T ₂	
Time ↓	Salary = ₹ 1000
t ₁		Update Salary = Salary + 200	Salary = ₹ 1200
t ₂	Read		Salary = ₹ 1200
t ₃		Rollback	Salary = ₹ 1000

Dirty read

- 1) At the time t₁, the transaction T₂ updates the salary to ₹ 1200
- 2) This salary is read at time t₂ by transaction T₁. Obviously it is ₹ 1200
- 3) But at the time t₃, the transaction T₂ performs Rollback by undoing the changes made by T₁ and T₂ at time t₁ and t₂.
- 4) Thus the salary again becomes = ₹ 1000. This situation leads to **Dirty Read or Uncommitted Read** because here the read made at time t₂ (**immediately after update of another transaction**) becomes a dirty read.

3) Non-repeatable read problem

This problem is also known as **inconsistent analysis problem**. This problem occurs when a particular transaction sees two different values for the same row within its lifetime. For example –

	T ₁	T ₂	
Time ↓	Read		Salary = ₹ 1000
t ₁		Update salary from ₹ 1000 to ₹ 1200	Salary = ₹ 1200
t ₂		Commit	
t ₃	Read		Salary = ₹ 1200
t ₄			

- 1) At time t₁, the transaction T₁ reads the salary as ₹ 1000
- 2) At time t₂ the transaction T₂ reads the same salary as ₹ 1000 and updates it to ₹1200
- 3) Then at time t₃, the transaction T₂ gets committed.
- 4) Now when the transaction T₁ reads the same salary at time t₄, it gets different value than what it had read at time t₁. Now, transaction T₁ cannot repeat its reading operation. Thus inconsistent values are obtained.

Hence the name of this problem is non-repeatable read or inconsistent analysis problem.

4) Phantom read problem

The phantom read problem is a special case of non repeatable read problem.

This is a problem in which one of the transaction makes the changes in the database system and due to these changes another transaction can not read the data item which it has read just recently. For example –

	T ₁	T ₂	
t ₁	Read		Salary = ₹ 1000
t ₂		Read	Salary = ₹ 1000
t ₃	Delete salary		No salary
t ₄		Read	"Can not find salary"

- 1) At time t₁, the transaction T₁ reads the value of salary as ₹ 1000
- 2) At time t₂, the transaction T₂ reads the value of the same salary as ₹ 1000
- 3) At time t₃, the transaction T₁ deletes the variable salary.
- 4) Now at time t₄, when T₂ again reads the salary it gets error. Now transaction T₂ can not identify the reason why it is not getting the salary value which is read just few time back.

This problem occurs due to changes in the database and is called phantom read problem.

5.9 Lock based Protocol

SPPU : Dec.-17,18,19, May-18,19, Marks 9

5.9.1 Why Do We Need Lock ?

- One of the method to ensure the **isolation** property in transactions is to require that data items be accessed in a mutually exclusive manner. That means, while one transaction is accessing a data item, no other transaction can modify that data item.
- The most common method used to implement this requirement is to allow a transaction to access a data item only if it is currently holding a **lock on that item**.
- Thus the lock on the operation is required to ensure the isolation of transaction.

5.9.2 Working of Lock

- **Concept of protocol** : The lock based protocol is a mechanism in which there is exclusive use of **locks** on the data item for current transaction.

- **Types of locks** : There are two types of locks used –



Fig. 5.9.1 Types of locks

- i) **Shared lock** : The shared lock is used for reading data items only. It is denoted by Lock-S. This is also called as **read lock**.
 - ii) **Exclusive lock** : The exclusive lock is used for both read and write operations. It is denoted as Lock-X. This is also called as **write lock**.
- The **compatibility matrix** is used while working on set of locks. The **concurrency control manager** checks the compatibility matrix before granting the lock. If the two modes of transactions are compatible to each other then only the lock will be granted.
 - In a set of locks may consists of shared or exclusive locks. Following matrix represents the compatibility between modes of locks.

	S	X
S	T	F
X	F	F

Fig. 5.9.2 Compatibility matrix for locks

Here T stands for True and F stands for False. If the control manager get the compatibility mode as True then it grant the lock otherwise the lock will be denied.

- **For example** : If the transaction T_1 is holding a shared lock in data item A, then the control manager can grant the shared lock to transaction T_2 as compatibility is True. But it cannot grant the exclusive lock as the compatibility is false. In simple words if transaction T_1 is reading a data item A then same data item A can be read by another transaction T_2 but cannot be written by another transaction.
- Similarly if an exclusive lock (i.e. lock for read and write operations) is hold on the data item in some transaction then no other transaction can acquire shared or exclusive lock as the compatibility function denotes F. That means of some transaction is writing a data item A then another transaction can not read or write that data item A.

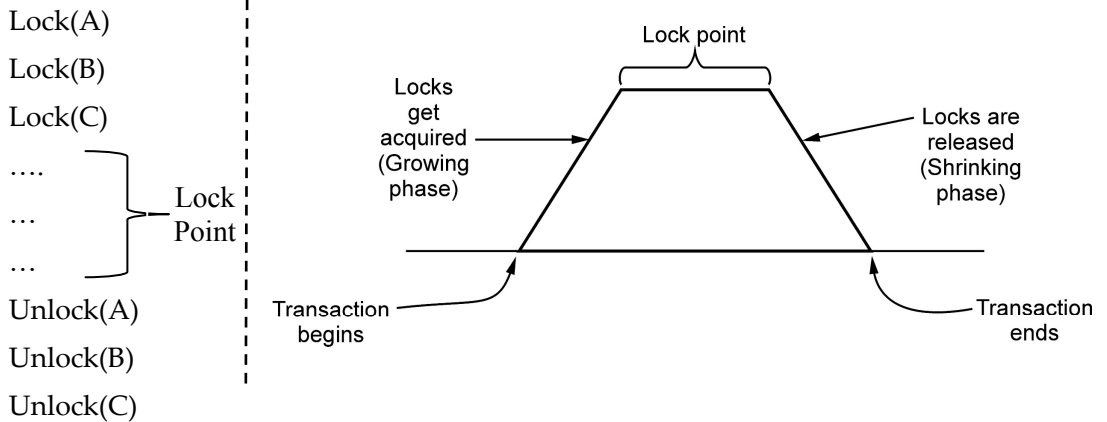
- Hence the **rule of thumb** is
 - i) **Any number** of transactions can hold **shared lock** on an item.
 - ii) But **exclusive lock** can be hold by **only one transaction**.
- **Example of a schedule denoting shared and exclusive locks** : Consider following schedule in which initially $A=100$. We deduct 50 from A in T1 transaction and Read the data item A in transaction T2. The scenario can be represented with the help of locks and concurrency control manager as follows :

	T ₁	T ₂	Concurrency control manager
	Lock-X(A)		
Exclusive lock			Grant X(A,T ₁) because in T ₁ there is write operation.
	R(A)		
	A=A - 50		
	W(A)		
	Unlock(A)		
Shared lock		Lock-S(A)	
			Grant S(A,T ₂) because in T ₂ there is read operation
		R(A)	
		Unlock(A)	

5.9.3 Two Phase Locking Protocol

- The two phase locking is a protocol in which there are two phases :
 - i) **Growing phase (Locking phase)** : It is a phase in which the transaction may obtain locks but does not release any lock.
 - ii) **Shrinking phase (Unlocking phase)** : It is a phase in which the transaction may release the locks but does not obtain any new lock.
- **Lock Point** : The **last lock position** or **first unlock position** is called lock point.

- For example -



Consider following transactions

T ₁	T ₂
Lock-X(A)	Lock-S(B)
Read(A)	Read(B)
A=A-50	Unlock-S(B)
Write(A)	
Lock-X(B)	
Unlock-X(A)	
B=B+100	Lock-S(A)
Write(B)	Read(A)
Unlock-X(B)	Unlock-S(A)

The important rule for being a two phase locking is - **All lock operations precede all the unlock operations.**

In above transactions T₁ is in two phase locking mode but transaction T₂ is not in two phase locking. Because in T₂, the shared lock is acquired by data item B, then data item B is read and then the lock is released. Again the lock is acquired by data item A , then the data item A is read and the lock is then released. Thus we get lock-unlock-lock-unlock sequence. Clearly this is not possible in two phase locking.

Example 5.9.1 Prove that two phase locking guarantees serializability.

Solution :

- Serializability is mainly an issue of handling write operation. Because any inconsistency may only be created by **write** operation.
- Multiple reads on a database item can happen parallelly.
- 2-Phase locking protocol restricts this unwanted read/write by applying **exclusive lock**.
- Moreover, when there is an **exclusive lock** on an item it will **only be released in shrinking phase**. Due to this restriction there is no chance of getting any inconsistent state.

The serializability using two phase locking can be understood with the help of following example :

Consider two transactions

T ₁	T ₂
R(A)	
	R(A)
R(B)	
W(B)	

Step 1 : Now we will **apply two phase locking**. That means we will **apply locks in growing** and shrinking phase

T ₁	T ₂
Lock-S(A)	
R(A)	
	Lock-S(A)
	R(A)
Lock-X(B)	
R(B)	
W(B)	
Unlock-X(B)	
	Unlock-S(A)

Note that above schedule is serializable as it prevents interference between two transactions.

The serializability order can be obtained based on the **lock point**. The lock point is either last lock operation position or first unlock position in the transaction.

The last lock position is in T_1 , then it is in T_2 . Hence the serializability will be $T_1 \rightarrow T_2$ based on lock points. Hence The **serializability sequence** can be $R_1(A); R_2(A); R_1(B); W_1(B)$

Advantages of two phase locking

- (1) It ensures serializability.

Disadvantages of two phase locking protocol

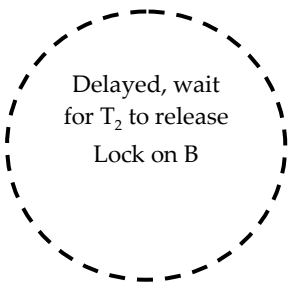
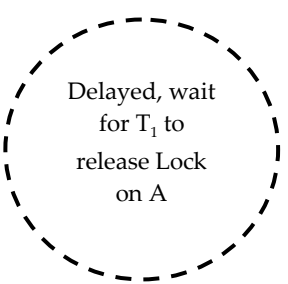
- (1) It leads to dealocks.
- (2) It leads to cascading rollback.

Problems in two phase locking

The two phase locking protocol leads to two problems - Deadlock and cascading roll back.

- 1) **Deadlock** : The deadlock problem can not be solved by two phase locking. Deadlock is a situation in which when two or more transactions have got a lock and waiting for another locks currently held by one of the other transactions.

For example

T_1	T_2
Lock-X(A)	Lock-X(B)
Read(A)	Read(B)
$A=A-50$	$B=B+100$
Write(A)	Write(B)
 <p>Delayed, wait for T_2 to release Lock on B</p>	 <p>Delayed, wait for T_1 to release Lock on A</p>

- 2) **Cascading Rollback** : Cascading rollback is a situation in which a single transaction failure leads to a series of transaction rollback. For example -

T ₁	T ₂	T ₃
Read(A)		
Read(B)		
C=A+B		
Write(C)		
	Read(C)	
	Write(C)	
		Read(C)

When T₁ writes value of C then only T₂ can read it. And when T₂ writes the value of C then only transaction T₃ can read it. But if the transaction T₁ gets failed then automatically transactions T₂ and T₃ gets failed.

The simple two phase locking does not solve the cascading rollback problem. To solve the problem of cascading Rollback two types of two phase locking mechanisms can be used.

5.9.3.1 Types of Two Phase Locking

- 1) **Strict two phase locking** : The strict 2PL protocol is a basic two phase protocol **but all the exclusive mode locks be held until the transaction commits**. That means in other words all the **exclusive locks** are **unlocked** only after the **transaction is committed**. That also means that if T₁ has exclusive lock, then T₁ will release the exclusive lock only after commit operation, then only other transaction is allowed to read or write. For example - Consider two transactions

T ₁	T ₂
W(A)	
	R(A)

If we apply the locks then

T ₁	T ₂
Lock-X(A)	
W(A)	
Commit	
Unlock(A)	
	Lock-S(A)
	R(A)
	Unlock-S(A)

Thus only after commit operation in T₁, we can unlock the exclusive lock. This ensures the strict serializability.

Thus compared to basic two phase locking protocol, the advantage of strict 2PL protocol is it ensures strict serializability.

2) Rigorous two phase locking : This is stricter two phase locking protocol. Here all locks are to be held until the transaction commits. The transactions can be serialized in the order in which they commit.

Example - Consider transactions

T ₁
R(A)
R(B)
W(B)

If we apply the locks then

T ₁
Lock-S(A)
R(A)
Lock-X(B)
R(B)
W(B)
Commit
Unlock(A)
Unlock(B)

Thus the above transaction uses rigorous two phase locking mechanism.

Example 5.9.2 Consider the following two transactions :

T_1 : read(A)Read(B);

If A=0 then B=B+1;

Write(B)

T_2 : read(B); read(A)

If B=0 then A=A+1

Write(A)

Add lock and unlock instructions to transactions T_1 and T_2 , so that they observe two phase locking protocol. Can the execution of these transactions result in deadlock ?

Solution :

T_1	T_2
Lock-S(A)	Lock-S(B)
Read(A)	Read(B)
Lock-X(B)	Lock-X(A)
Read(B)	Read(A)
if A=0 then B=B+1	if B=0 then A=A+1
Write(B)	Write(A)
Unlock(A)	Unlock(B)
Commit	Commit
Unlock(B)	Unlock(A)

This is lock-unlock instruction sequence help to satisfy the requirements for strict two phase locking for the given transactions.

The execution of these transactions result in deadlock. Consider following partial execution scenario which leads to deadlock.

T_1	T_2
Lock-S(A)	Lock-S(B)
Read(A)	Read(B)
Lock-X(B)	Lock-X(A)
Now it will wait for T_2 to release exclusive lock on A	Now it will wait for T_1 to release exclusive lock on B

Review Questions

1. A transaction may be waiting for more time for an Exclusive (X) lock on an item, while a sequence of other transactions request and are granted as Shared (S) lock on the same item. What is this problem ? How it is solved by two phase lock protocol ?

SPPU : Dec.-17, End Sem, Marks 8

2. Explain the two phase lock protocol for concurrency control. Also explain its two versions : strict two phase lock protocol and rigorous two phase lock protocol.

SPPU : May-18, End Sem, Marks 8

3. Explain the Two Phase lock Protocol and show how it ensures conflict serializability. Two Phase lock protocol does not ensure freedom from deadlock explain with necessary example. Also explain its two versions : strict two phase lock protocol and rigorous two phase lock protocol.

SPPU : Dec.-18, 19, End Sem, Marks 9

4. What benefit does rigorous two-phase locking provide ? How does it compare with other forms of two - phase locking ?

SPPU : May-19, End Sem, Marks 9

5.10 Time-stamp based Protocol

SPPU : May-19, Dec.-17, Marks 9

- The time stamp ordering protocol is a scheme in which the order of transaction is decided in advance based on their timestamps. Thus the schedules are serialized according to their timestamps.
- The timestamp-ordering protocol ensures that any conflicting read and write operations are executed in timestamp order.
- A larger timestamp indicates a more recent transaction or it is also called as **younger transaction** while lesser timestamp indicates **older transaction**.
- Assume a collection of data items that are accessed, with read and write operations, by transactions.
- For each data item X the DBMS maintains the following values : –
 - **RTS(X)** : The timestamp on which object X was last read (by some transaction T_i , i.e., $RTS(X)=TS(T_i)$) [Note that : RTS stands for Read Time Stamp]
 - **WTS(X)** : The timestamp on which object X was last written (by some transaction T_j , i.e., $WTS(X)=TS(T_j)$) [**Note that** : WTS stands for Write Time Stamp]
- For the following algorithms we use the following assumptions : A data item X in the database has a **RTS(X)** and **WTS(X)**. These are actually the timestamps of read and write operations performed on data item X at latest time.
- A transaction T attempts to perform some action (read or write) on data item X on some timestamp and we call that timestamp as **TS(T)**.
- By timestamp ordering algorithm we need to decide whether transaction T has to be aborted or T can continue execution.

Basic Timestamp Ordering Algorithm

Case 1 (Read) : Transaction T issues a read(X) operation

- i) If $TS(T) < WTS(X)$, then read(X) is rejected. T has to abort and be rejected.
- ii) If $WTS(X) \leq TS(T)$, then execute read(X) of T and update $RTS(X)$.

Case 2 (Write) : Transaction T issues a write(X) operation

- i) If $TS(T) < RTS(X)$ or if $TS(T) < WTS(X)$, then write is rejected
- ii) If $RTS(X) \leq TS(T)$ or $WTS(X) \leq TS(T)$, then execute write(X) of T and update $WTS(X)$.

Example for Case 1 (Read operation)

- (i) Suppose we have two transactions T_1 and T_2 with timestamps 10 sec. and 20 sec. respectively.

10 Sec T_1	20 Sec T_2
R(X)	
	W(X)
R(X)	

$RTS(X)$ and $WTS(X)$ is initially = 0

Then $RTS(X) = 10$, when transaction T_1 executes

After that $WTS(X) = 20$ when transaction T_2 executes

Now if read operation R(X) occurs on transaction T_1 at $TS(T_1) = 10$ then

$TS(T_1)$ i.e. $10 < WTS(X)$ i.e. 20, hence we have to reject second read operation on T_1 i.e.

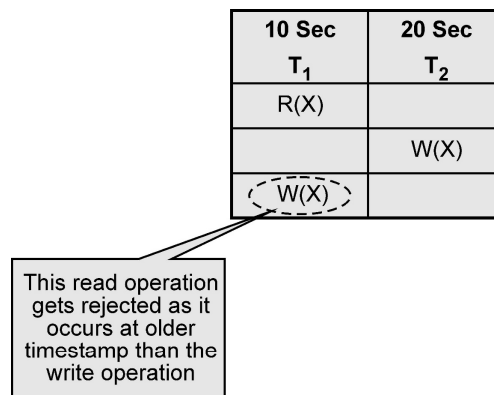


Fig. 5.10.1

- ii) Suppose we have two transactions T_1 and T_2 with timestamps 10 sec. and 20 sec. respectively.

10 Sec	20 Sec
T_1	T_2
W(X)	
	R(X)

RTS(X) and WTS(X) is initially = 0

Then $WTS(X) = 10$ as transaction T_1 executes.

Now if Read operation $R(X)$ occurs on transaction T_2 at $TS(T_2) = 20$ then

$TS(T_2)$ i.e. $20 > WTS(X)$ which is 10, hence we accept read operation on T_2 . The transaction T_2 will perform read operation and now RTS will be updated as,

$RTS(X) = 20$

Example for Case 2 (Write Operation)

- i) Suppose we have two transactions T_1 and T_2 with timestamps 10 sec. and 20 sec. respectively.

10 Sec.	20 Sec.
T_1	T_2
R(X)	
	W(X)
W(X)	

RTS(X) and WTS(X) is initially = 0

Then $RTS(X) = 10$, when transaction T_1 executes

After that $WTS(X) = 20$ when transaction T_2 executes

Now if write operation $W(X)$ occurs on transaction T_1 at $TS(T_1) = 10$ then

$TS(T_1)$ i.e. $10 < WTS(X)$, hence we have to reject second write operation on T_1 i.e.

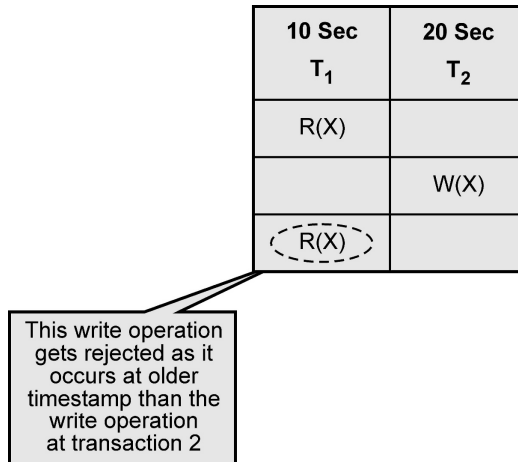


Fig. 5.10.2

- ii) Suppose we have two transactions T_1 and T_2 with timestamps 10 sec. and 20 sec. respectively.

10 Sec T_1	20 Sec T_2
W(X)	
	W(X)

RTS(X) and WTS(X) is initially = 0

Then $WTS(X) = 10$ as transaction T_1 executes.

Now if write operation $W(X)$ occurs on transaction T_2 at $TS(T_2) = 20$ then

$TS(T_2)$ i.e. $20 > WTS(X)$ which is 10, hence we accept write operation on T_2 . The transaction T_2 will perform write operation and now WTS will be updated as,

$WTS(X) = 20$

Advantages and disadvantages of time stamp ordering

Advantages

- 1) Schedules are serializable
- 2) No waiting for transaction and hence there is no deadlock situation.

Disadvantages

- 1) Schedules are not recoverable once transactions occur.
- 2) Same transaction may be continuously aborted or restarted.

Review Question

1. Suppose a transaction, issues a read command on data item Q . How time-stamp based protocol decides whether to allow the operation to be executed or not using time-stamp based protocol of concurrency control.

SPPU : Dec.-17, May-19, End Sem, Marks 9

5.11 Deadlocks Handling

Deadlock is a specific concurrency problem in which two transactions depend on each other for something.

For example - Consider that transaction T_1 holds a lock on some rows of table **A** and needs to update some rows in the **B** table. Simultaneously, transaction T_2 holds locks on some rows in the **B** table and needs to update the rows in the **A** table held by transaction T_1 .

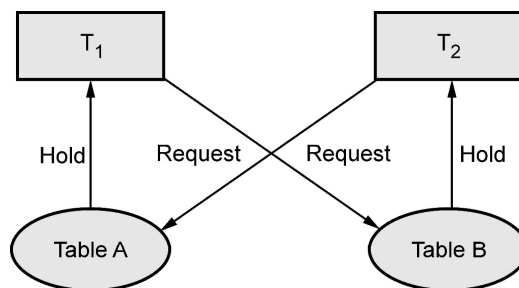


Fig. 5.11.1

Now, the main problem arises. Now transaction T_1 is waiting for T_2 to release its lock and similarly, transaction T_2 is waiting for T_1 to release its lock. All activities come to a halt state and remain at a standstill. This situation is called **deadlock** in DBMS.

Definition : Deadlock can be formally defined as - " A system is in deadlock state if there exists a set of transactions such that every transaction in the set is waiting for another transaction in the set. "

There are **four conditions for a deadlock to occur**

A deadlock may occur if all the following conditions holds true.

1. **Mutual exclusion condition :** There must be at least one resource that cannot be used by more than one process at a time.
2. **Hold and wait condition :** A process that is holding a resource can request for additional resources that are being held by other processes in the system.
3. **No preemption condition :** A resource cannot be forcibly taken from a process. Only the process can release a resource that is being held by it.
4. **Circular wait condition :** A condition where one process is waiting for a resource that is being held by second process and second process is waiting for third process

...so on and the last process is waiting for the first process. Thus making a circular chain of waiting.

Deadlock can be handled using two techniques -

1. Deadlock prevention 2. Deadlock detection and deadlock recovery

1. Deadlock prevention :

For large database, deadlock prevention method is suitable. A deadlock can be prevented if the resources are allocated in such a way that deadlock never occur. The DBMS analyzes the operations whether they can create deadlock situation or not, If they do, that transaction is never allowed to be executed.

There are two techniques used for deadlock prevention -

i) Wait-Die :

- In this scheme, if a transaction requests for a resource which is already held with a conflicting lock by another transaction then the DBMS simply checks the timestamp of both transactions. It **allows the older transaction to wait** until the resource is available for execution.
- Suppose there are two transactions T_i and T_j and let $TS(T)$ is a timestamp of any transaction T . If T_2 holds a lock by some other transaction and T_1 is requesting for resources held by T_2 then the following actions are performed by DBMS :
 - Check if $TS(T_i) < TS(T_j)$ - If T_i is the older transaction and T_j has held some resource, then T_i is allowed to wait until the data-item is available for execution. That means if the older transaction is waiting for a resource which is locked by the younger transaction, then the older transaction is allowed to wait for resource until it is available.
 - Check if $TS(T_i) > TS(T_j)$ - If T_i is older transaction and has held some resource and if T_j is waiting for it, then T_j is killed and restarted later with the random delay but with the same timestamp.

Timestamp is a way of assigning priorities to each transaction when it starts. If timestamp is lower then that transaction has higher priority. **That means oldest transaction has highest priority.**

For example -

Let T_1 is a transaction which requests the data item acquired by transaction T_2 . Similarly T_3 is a transaction which requests the data item acquired by transaction T_2 .

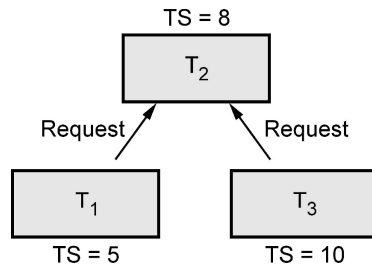


Fig. 5.11.2

Here $TS(T_1)$ i.e. Time stamp of T_1 is less than $TS(T_3)$. In other words T_1 is older than T_3 . Hence T_1 is made to **wait** while T_3 is **rolledback**.

ii) Wound-wait :

- In wound wait scheme, if the older transaction requests for a resource which is held by the younger transaction, then older transaction forces younger one to kill the transaction and release the resource. After some delay, the younger transaction is restarted but with the same timestamp.
- If the older transaction has held a resource which is requested by the younger transaction, then the younger transaction is asked to wait until older releases it.

Suppose T_1 needs a resource held by T_2 and T_3 also needs the resource held by T_2 , with $TS(T_1) = 5$, $TS(T_2) = 8$ and $TS(T_3) = 10$, then T_1 being older waits and T_3 being younger dies. After the some delay, the younger transaction is restarted but with the same timestamp.

This ultimately prevents a deadlock to occur.

To summarize

	Wait-Die	Wound-wait
Older transaction needs a data item held by younger transaction	Older transaction waits	Younger transaction dies.
Younger transaction needs a data item held by older transaction	Younger transaction dies	Younger transaction dies.

2. Deadlock detection :

- In deadlock detection mechanism, an algorithm that examines the state of the system is invoked periodically to determine whether a deadlock has occurred or not. If deadlock is occurrence is detected, then the system must try to recover from it.

- Deadlock detection is done using wait for graph method.

Wait for graph

- In this method, a graph is created based on the transaction and their lock. If the created **graph has a cycle or closed loop, then there is a deadlock.**
- The wait for the graph is maintained by the system for every transaction which is waiting for some data held by the others. The system keeps checking the graph if there is any cycle in the graph.
- This graph consists of a pair $G = (V, E)$, where V is a set of vertices and E is a set of edges.
- The set of vertices consists of all the transactions in the system.
- When transaction T_i requests a data item currently being held by transaction T_j , then the edge $T_i \rightarrow T_j$ is inserted in the wait-for graph. This edge is removed only when transaction T_j is no longer holding a data item needed by transaction T_i .

For example - Consider following transactions, We will draw a wait for graph for this scenario and check for deadlock.

T_1	T_2
R(A)	
	R(A)
W(A)	
R(B)	
	W(A)
W(B)	

We will use three rules for designing the wait-for graph -

Rule 1 : If T_1 has **Read** operation and then T_2 has **Write** operation then draw an edge $T_1 \rightarrow T_2$.

Rule 2 : If T_1 has **Write** operation and then T_2 has **Read** operation then draw an edge $T_1 \rightarrow T_2$.

Rule 3 : If T_1 has **Write** operation and then T_2 has **Write** operation then draw an edge $T_1 \rightarrow T_2$.

Let us draw wait-for graph

Step 1 : Draw vertices for all the transactions



Fig. 5.11.3

Step 2 : We find the Read-Write pair from two different transactions reading from top to bottom. If such a pair is found then we will add the edges between corresponding directions. For instance -

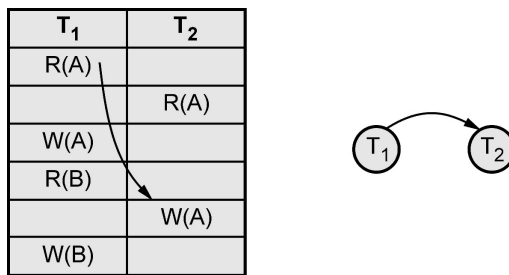


Fig. 5.11.4

Step 3 :

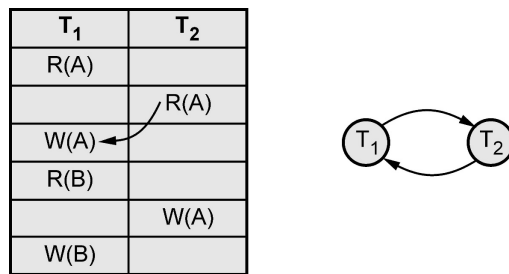


Fig. 5.11.5

As **cycle is detected** in the wait-for graph there is no need to further process. The **deadlock** is present in this transaction scenario.

Example 5.11.1 Give an example of a scenario where two phase locking leads to deadlock.

Solution : Following scenario of execution of transactions can result in deadlock.

These two instructions cause a deadlock situation.

T ₁	T ₂
Lock - S (A)	
	Lock - S (B)
	Read (B)
Read (A)	
Lock - X (B)	
	Lock - X (A)

In above scenario, transaction T₁ makes an exclusive lock on data item B and then transaction T₂ makes an exclusive lock on data item A. Here unless and until T₁ does not give up the lock (i.e. unlock) on B; T₂ cannot read / write it. Similarly unless and until T₂ does not give up the lock on A; T₁ cannot read or write on A.

This is a purely deadlock situation in two phase locking.

Example 5.11.2 What is deadlock ? Consider following sequence of actions listed in order they are submitted to DBMS

Sequence : S1:R1(A)W2(B);R1(B);R3(C);w2(c);W4(B);W3(A)

Draw waits-for graph in case of deadlock situation

Solution :

Step 1 : We will rewrite the schedule as,

T1	T2	T3	T4
R1(A)			
	W2(B)		
R1(B)			
		R3(C)	
	W2(C)		
			W4(B)
		W3(A)	

Step 2 :

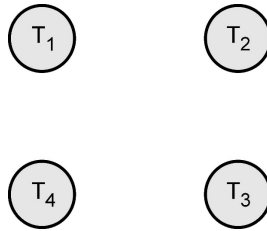


Fig. 5.11.6

As there are four transactions, there will be four nodes.

Step 3 : We find the Read-Write pair from two different transactions reading from top to bottom. If such a pair is found then we will add the edges between corresponding directions. For instance -

T1	T2	T3	T4
R1(A)			
	W2(B)		
R1(B)			
		R3(C)	
	W2(C)		
			W4(B)
		W3(A)	

Hence the wait-for graph will be,

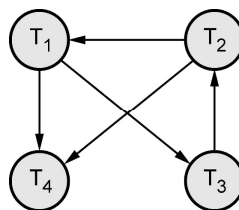


Fig. 5.11.7

The cycle is detected in wait-for graph as T₁ → T₃ → T₂. That means deadlock is present in this transaction scenario.

5.12 Recovery Concepts

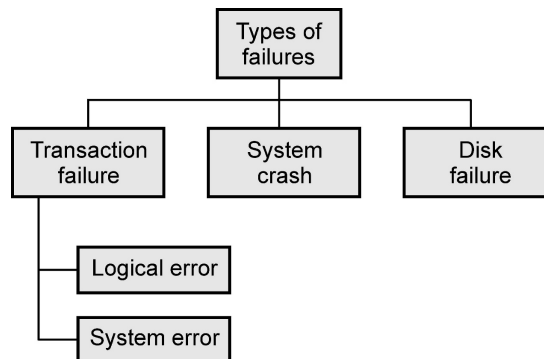
5.12.1 Purpose of Database Recovery

- The purpose of recovery is to bring the database into the last consistent stage prior to occurrence of failure.

- The recovery must preserve all the ACID properties of transaction. The ACID properties are - Atomicity, consistency, isolation and durability.
- Thus recovery ensures high availability of the database for transaction purpose.
- For example - If the system crashes before the amount transfer from one account to another then either one or both the accounts may have incorrect values. Here the database must be recovered before the modification takes place.

5.12.2 Failure Classification

Various types of failures are –



1) Transaction failure : Following are two types of errors due to which the transaction gets failed.

- **Logical error :**

- i) This error is caused due to internal conditions such as bad input, data not found, overflow of resource limit and so on.
- ii) Due to logical error the transaction can not be continued.

- **System error :**

- i) When the system enters in an undesired state and then the transaction can not be continued then this type of error is called as system error.

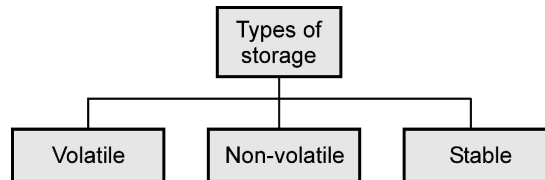
2) System crash : The situation in which there is a hardware malfunction, or a bug in the database software or the operating system, and because of which there is a loss of the content of volatile storage, and finally the transaction processing come to a halt is called system crash.

3) Disk failure : A disk block loses its content as a result of either a head crash or failure during a data-transfer operation. The backup of data is maintained on the secondary disks or DVD to recover from such failure.

5.12.3 Storage

A DBMS stores the data on external storage because the amount of data is very huge and must persist across program executions.

The storage structure is a memory structure in the system. It has following categories –



1) Volatile :

- Volatile memory is a primary memory in the system and is placed along with the CPU.
- These memories can store only small amount of data, but they are very fast. For example - main memory, cache memory.
- A volatile storage cannot survive system crashes.
- That means data in these memories will be lost on failure.

2) Non volatile :

- Non volatile memory is a secondary memory and is huge in size. For example : Hard disk, Flash memory, Magnetic tapes.
- These memories are designed to withstand system crashes.

3) Stable :

- Information residing in stable storage is never lost.
- To implement stable storage, we replicate the information in several nonvolatile storage media (usually disk) with independent failure modes.

Stable Storage Implementation

- Stable storage is a kind of storage on which the information residing on it is never lost.
- Although stable storage is theoretically impossible to obtain it can be approximately built by applying a technique in which **data loss is almost impossible**.
- That means the information is **replicated** in several nonvolatile storage media with **independent failure modes**.
- Updates must be done with care to ensure that a failure during an update to stable storage does not cause a loss of information.

5.13 Recovery Methods

Following are commonly used crash recovery methods –

- 1) Shadow paging
- 2) Log-based recovery
 - a. Deferred Update
 - b. Immediate Update
- 3) Checkpointing

5.14 Shadow-paging

- Shadow paging is a recovery scheme in which database is considered to be made up of number of fixed size disk pages.
- A directory or a page table is constructed with n number of pages where each i th page points to the i th database page on the disk. (Refer Fig. 5.14.1)

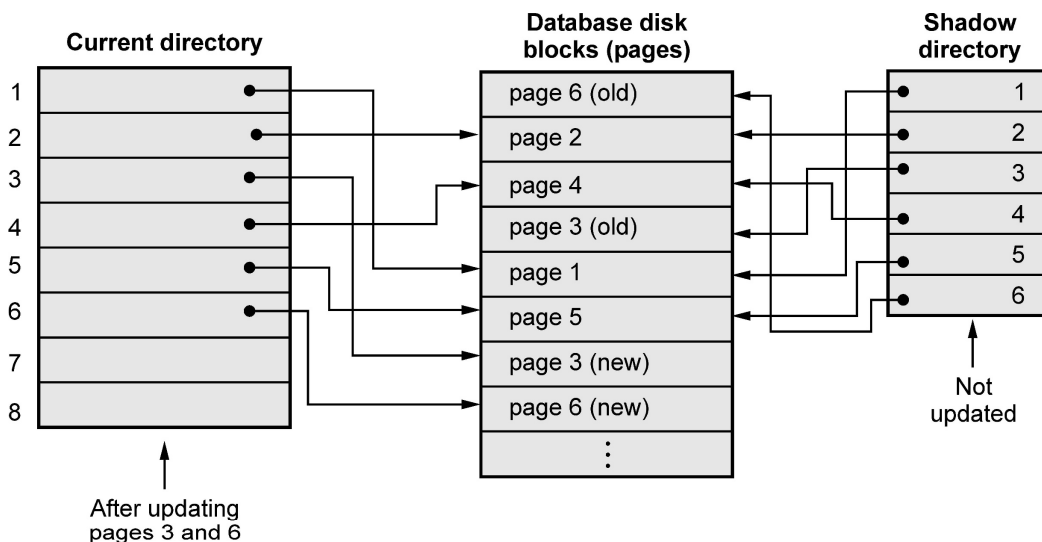


Fig. 5.14.1 Demonstration of shadow paging

- The directory can be kept in the main memory.
- When a transaction begins executing, the current directory-whose entries point to the most recent or current database pages on disk-is copied into a another directory called shadow directory.
- The shadow directory is then saved on disk while the current directory is used by the transaction.
- During the execution of transaction, the shadow directory is never modified.

- When a write operation is to be performed then the **new copy** of modified database page is created but the old copy of database page is never overwritten. This newly created database page is written somewhere else.
- The current directory will point to newly modified web page and the shadow page directory will point to the old web page entries of database disk.
- When the failure occurs then the modified database pages and current directory is discarded.
- The state of database before the failure occurs is now available through the shadow directory and this state can be recovered using shadow directory pages.
- This technique does not require any UNDO/REDO operation.

5.15 Check Points

- Checkpoint is a mechanism where all the previous logs are removed from the system and stored permanently in a storage disk.
- Checkpoint declares a point before which the DBMS was in consistent state, and all the transactions were committed.
- The recovery system reads the logs backwards from the end to the last checkpoint.
- Performing a checkpoint consists of the following operations :
 - Suspending executions of transactions temporarily;
 - Writing (force-writing) all modified database buffers of committed transactions out to disk;
 - Writing a checkpoint record to the log; and
 - Writing (force-writing) all log records in main memory out to disk.
- A checkpoint record usually contains additional information, including a list of transactions active at the time of the checkpoint.
- Many recovery methods (including the deferred and immediate update methods) need this information when a transaction is **rolled back**, as all transactions active at the time of the checkpoint and any subsequent ones may need to be redone.
- Since checkpoints cause some loss in performance while they are being taken, their frequency should be reduced if fast recovery is not critical.
- If we need fast recovery check-pointing frequency should be increased. If the amount of stable storage available is less, frequent check-pointing is unavoidable.

5.16 Log-based Recovery : Deferred and Immediate Update

SPPU : May-18,19, Marks 8

Before discussing the recovery algorithms(deferred and immediate update), let us see the concept of Log and REDO and UNDO operations.

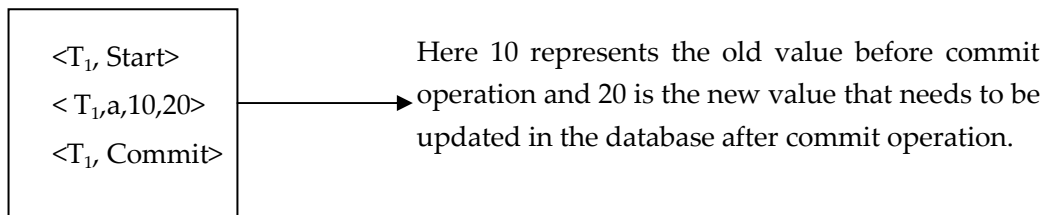
5.16.1 Concept of Log

- Log is the most commonly used structure for recording the modifications that is to be made in the actual database. Hence during the recovery procedure a log file is maintained.
- A log record maintains four types of operations. Depending upon the type of operations there are four types of log records -
 1. <Start> Log record : It is represented as $\langle T_i, \text{Start} \rangle$
 2. <Update> Log record
 3. <Commit> Log record : It is represented as $\langle T_i, \text{Commit} \rangle$
 4. <Abort> Log record : It is represented as $\langle T_i, \text{Abort} \rangle$
- The log contains various fields as shown in following Fig. 5.16.1. This structure is for <update> operation

Transaction ID(T_i)	Data Item Name	Old Value of Data Item	New Value of Data Item
-------------------------	----------------	------------------------	------------------------

Fig. 5.16.1

- For example : The sample log file is



- The log must be maintained on the stable storage and the entries in the log file are maintained before actually updating the physical database.
- There are two approaches used for log based recovery technique - Deferred Database Modification and Immediate Database Modification.

5.16.2 REDO and UNDO Operation

During transaction execution, the updates are recorded only in the log and in the cache buffers. After the transaction reaches its commit point and the log is force written to disk and the updates are recorded in the database.

In order to maintain the atomicity of transaction, the operations can be redone or undone.

UNDO : This is an operation in which we restore all the old values (BFIM - BeFore Modification Image) onto the disk. This is called **roll-back operation**.

REDO : This is an operation in which all the modified values(AFIM - After Modification Image) are restored onto the disk. This is called **roll-forward operation**.

These operations are recorded in the log as they happen.

Difference between UNDO and REDO

Sr. No.	UNDO	REDO
1.	Makes a change go away.	Reproduces a change.
2.	Used for rollback and read consistency.	Used for rolling forward the changes.
3.	Protects the database from inconsistent reads.	Protects from data loss.

5.16.3 Write Ahead Logging Rule

- Before a block of data in main memory is output to the database, all log records pertaining to data in that block must have been output to stable storage. This rule is called the **Write-Ahead Logging (WAL)**
- This rule is necessary because - In the event of a crash or ROLLBACK, the original content contained in the rollback journal is played back into the database file to revert the database file to its original state.

5.16.4 Deferred Database Modification

- In this technique, the database is not updated immediately.
- Only log file is updated on each transaction.
- When the transaction reaches to its commit point, then only the database is physically updated from the log file.
- In this technique, if a transaction fails before reaching to its commit point, it will not have changed database anyway. Hence there is no need for the UNDO operation. The REDO operation is required to record the operations from log file to physical database. Hence deferred database modification technique is also called as **NO UNDO/REDO algorithm**.

- For example :

Consider two transactions T_1 and T_2 as follows :

T_1	T_2
Read (A, a)	Read (C, c)
$a = a - 10$	$c = c - 20$
Write (A, a)	Write (C, c)
Read (B, b)	
$b = b + 10$	
Write (B, b)	

If T_1 and T_2 are executed serially with initial values of $A = 100$, $B = 200$ and $C = 300$, then the state of log and database if crash occurs

- Just after write (B, b)
- Just after write (C, c)
- Just after $\langle T_2, \text{commit} \rangle$

The result of above 3 scenarios is as follows :

Initially the log and database will be

Log	Database
$\langle T_1, \text{Start} \rangle$	
$\langle T_1, A, 90 \rangle$	
$\langle T_1, B, 210 \rangle$	
$\langle T_1, \text{Commit} \rangle$	
	$A = 90$
	$B = 210$
$\langle T_2, \text{Start} \rangle$	
$\langle T_2, C, 280 \rangle$	
$\langle T_2, \text{Commit} \rangle$	
	$C = 280$

a) Just after write (B, b)

Just after write operation, no commit record appears in log. Hence no write operation is performed on database. So database retains only old values. Hence $A = 100$ and $B = 200$ respectively.

Thus the system comes back to original position and no redo operation take place.

The incomplete transaction of T_1 can be deleted from log.

b) Just after write (C, c)

The state of log records is as follows

Note that crash occurs before T_2 commits. At this point T_1 is completed successfully, so new values of A and B are written from log to database. But as T_2 is not committed, there is no redo (T_2) and the incomplete transaction T_2 can be deleted from log.

The redo (T_1) is done as $\langle T_1, \text{commit} \rangle$ gets executed. Therefore $A = 90$, $B = 210$ and

$C = 300$ are the values for database.

c) Just after $\langle T_2, \text{commit} \rangle$

The log records are as follows :

$\langle T_1, \text{Start} \rangle$

$\langle T_1, A, 90 \rangle$

$\langle T_1, B, 210 \rangle$

$\langle T_1, \text{Commit} \rangle$

$\langle T_2, \text{Start} \rangle$

$\langle T_2, 6, 280 \rangle$

$\langle T_2, \text{Commit} \rangle$

← **Crash occurs here**

Clearly both T_1 and T_2 reached at commit point and then crash occurs. So both redo (T_1) and redo (T_2) are done and updated values will be $A = 90$, $B = 210$, $C = 280$.

5.16.5 Immediate Database Modification

In this technique, the database is updated during the execution of transaction even before it reaches to its commit point.

If the transaction gets failed before it reaches to its commit point, then the a **ROLLBACK Operation** needs to be done to bring the database to its earlier consistent state. That means the effect of operations need to be undone on the database. For that purpose both Redo and Undo operations are both required during the recovery. This technique is known as **UNDO/ REDO** technique.

For example : Consider two transaction T_1 and T_2 as follows :

T_1	T_2
Read(A, a)	Read(C, c)
a = a - 10	c = c - 20
Write(A, a)	Write(C, c)
Read(B, b)	
b = b + 10	
Write(B, b)	

Here T_1 and T_2 are executed serially. Initially $A = 100$, $B = 200$ and $C = 300$

If the crash occurs after

i) Just after Write(B, b) ii) Just after Write(C, c) iii) Just after $\langle T_2, \text{Commit} \rangle$

Then using the immediate Database modification approach the result of above three scenarios can be elaborated as follows :

The contents of **log** and **database** is as follows :

Log	Database
$\langle T_1, \text{Start} \rangle$	
$\langle T_1, A, 100, 90 \rangle$	A = 90
$\langle T_1, B, 200, 210 \rangle$	B = 210
$\langle T_1, \text{Commit} \rangle$	
$\langle T_2, \text{Start} \rangle$	
$\langle T_2, C, 300, 280 \rangle$	C = 280
$\langle T_2, \text{Commit} \rangle$	

The recovery scheme uses two recovery techniques -

- i) UNDO (T_i) :** The transaction T_i needs to be undone if the log contains $\langle T_i, \text{Start} \rangle$ but does not contain $\langle T_i, \text{Commit} \rangle$. In this phase, it restores the values of all data items updated by T_i to the old values.
- ii) REDO (T_i) :** The transaction T_i needs to be redone if the log contains both $\langle T_i, \text{Start} \rangle$ and $\langle T_i, \text{Commit} \rangle$. In this phase, the data item values are set to the new values as per the transaction. After a failure has occurred log record is consulted to determine which transaction need to be redone.

- a) **Just after Write (B, b)** : When system comes back from this crash, it sees that there is $\langle T_1, \text{Start} \rangle$ but no $\langle T_1, \text{Commit} \rangle$. Hence T_1 must be undone. That means old values of A and B are restored. Thus old values of A and B are taken from log and both the transaction T_1 and T_2 are re-executed.
- b) **Just after Write (C, c)** : Here both the redo and undo operations will occur.
- c) **Undo** : When system comes back from this crash, it sees that there is $\langle T_2, \text{Start} \rangle$ but no $\langle T_2, \text{Commit} \rangle$. Hence T_2 must be undone. That means old values of C is restored.
Thus old value of C is taken from log and the transaction T_2 is re-executed.
- d) **Redo** : The transaction T_1 must be done as log contains both the $\langle T_1, \text{Start} \rangle$ and $\langle T_1, \text{Commit} \rangle$
So $A = 90, B = 210$ and $C = 300$
- e) **Just after $\langle T_2, \text{Commit} \rangle$** : When the system comes back from this crash, it sees that there are two transaction T_1 and T_2 with both start and commit points. That means T_1 and T_2 need to be redone. So $A = 90, B = 210$ and $C = 280$

Example 5.16.1 Suppose there is a database system that never fails. Is a recovery manager require for this system ? Why ?

Solution :

- (1) Yes. Even-though the database system never fails, the recovery manager is required for this system.
- (2) During the transaction processing some transactions might be aborted. Such transactions must be rolled back and then the schedule is continued further.
- (3) Thus to perform the rollbacks of aborted transactions recovery manager is required.

Review Question

1. To ensure atomicity despite failures we uses recovery methods. Explain in detail log-based recovery method.

SPPU : May-18,19 End Sem, Marks 8

Q.19 No more lock request can be asked in ___ phase.

a shrinking phase

b growing phase

c running phase

d initial phase

Q.20 ___ is a specific concurrency problem wherein two transactions depend on each other for something.

a conflict

b deadlock

c dirty read

d uncommitted transaction

Answer Keys for Multiple Choice Questions :

Q.1	d	Q.2	c	Q.3	a	Q.4	b
Q.5	d	Q.6	c	Q.7	b	Q.8	c
Q.9	b	Q.10	c	Q.11	b	Q.12	c
Q.13	d	Q.14	a	Q.15	c	Q.16	a
Q.17	c	Q.18	a	Q.19	a	Q.20	b

□□□

6

NoSQL Databases

Syllabus

Introduction to Distributed Database System, Advantages, Disadvantages, CAP Theorem.

Types of Data: Structured, Unstructured Data and Semi-Structured Data.

NoSQL Database : Introduction, Need, Features. Types of NoSQL Databases : Key-value store, document store, graph, wide column stores, BASE Properties, Data Consistency model, ACID Vs BASE, Comparative study of RDBMS and NoSQL. MongoDB (with syntax and usage): CRUD Operations, Indexing, Aggregation, MapReduce, Replication, Sharding.

Contents

- 6.1 *Introduction to Distributed Database System*
- 6.2 *CAP Theorem*
- 6.3 *Types of Data : Structured, Unstructured Data and Semi-Structured Data*
- 6.4 *NoSQL Database*
- 6.5 *Types of NoSQL Databases*
- 6.6 *BASE Properties*
- 6.7 *ACID Vs BASE*
- 6.8 *Comparative Study of RDBMS and NoSQL*
- 6.9 *MongoDB*

Multiple Choice Questions

6.1 Introduction to Distributed Database System

A distributed database system consists of loosely coupled sites (computer) that share no physical components and each site is associated a database system.

The software that maintains and manages the working of distributed databases is called distributed database management system.

The database system that runs on each site is independent of each other. Refer Fig. 6.1.1.

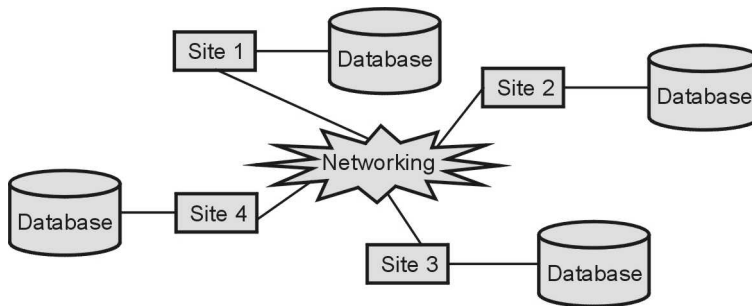


Fig. 6.1.1 Distributed database systems

The transactions can access data at one or more sites.

Advantages of Distributed Database System

- 1) There is **fast data processing** as several sites participate in request processing.
- 2) **Reliability and availability** of this system is high.
- 3) It possess **reduced operating cost**.
- 4) It is **easier to expand** the system by adding more sites.
- 5) It has improved **sharing ability and local autonomy**.

Disadvantages of Distributed Database System

- 1) The system becomes **complex** to manage and control.
- 2) The **security issues** must be carefully managed.
- 3) The system require **deadlock handling** during the transaction processing otherwise the entire system may be in inconsistent state.
- 4) There is need of some **standardization** for processing of distributed database system.

Difference between Centralized DBMS and Distributed DBMS

Distributed DBMS	Centralized DBMS
The database files are stored at geographically different locations across the network.	The database is stored at centralized location.
As data is distributed over the network , it requires time to synchronize data and thus difficult to maintain.	A centralized database is easier to maintain and keep updated since all the data are stored in a single location.
If one database fails, user can have access to other database files.	If the centralized database fails, then there is no access to a database.
It can have data replication as database is distributed. Hence there can be some data inconsistency.	It have single database system, hence there is no data replication. Therefore there is no data inconsistency.

Uses of distributed system :

- 1) Often distributed databases are used by organizations that have numerous offices in different geographical locations. Typically an individual branch is interacting primarily with the data that pertain to its own operations, with a much less frequent need for general company data. In such a situation, distributed systems are useful.
- 2) Using distributed system, one can give permissions to single sections of the overall database, for better internal and external protection.
- 3) If we need to add a new location to a business, it is simple to create an additional node within the database, making distribution highly scalable.

Review Questions

1. What is distributed database system ? Enlist its advantages and disadvantages.
2. Give the difference between distributed DBMS and centralized DBMS.

6.2 CAP Theorem

- Cap theorem is also called as brewer's theorem.
- The CAP Theorem is comprised of three components (hence its name) as they relate to distributed data stores :
 - **Consistency** : All reads receive the most recent write or an error.
 - **Availability** : All reads contain data, but it might not be the most recent.

- **Partition tolerance** : The system continues to operate despite network failures (i.e.; dropped partitions, slow network connections, or unavailable network connections between nodes.)
- The CAP theorem states that it is not possible to guarantee all three of the desirable properties - consistency, availability, and partition tolerance at the same time in a distributed system with data replication.

Review Question

1. Write a short note on CAP theorem.

6.3 Types of Data : Structured, Unstructured Data and Semi-Structured Data

Sr. No.	Structured data	Semi-structured data	Unstructured data
1.	It is having fixed and organized form of data.	It is combination of structured and unstructured data.	It is not predefined or organized form of data.
2.	It is schema dependent and less flexible.	It is more flexible than structured data but less flexible than unstructured data.	It is the most flexible data.
3.	Structured query languages are used to access the data present in the schema.	The tags and elements are used to access the data.	Only textual queries are possible.
4.	Storage requirement for data is less.	Storage requirements for the data is significant.	Storage requirements for the data is huge.
5.	Examples : Phone numbers, Customer Names, Social Security numbers.	Examples : Server logs, Tweets organized by hashtags, emails sorted by the inbox, sent or draft folders.	Examples : Emails and messages, Image files, Open ended survey answers.

6.4 NoSQL Database

6.4.1 Introduction

- NoSQL stands for not only SQL.
- It is nontabular database system that store data differently than relational tables. There are various types of NoSQL databases such as document, key-value, wide column and graph.

- Using NoSQL we can maintain flexible schemas and these schemas can be scaled easily with large amount of data.

6.4.2 Need

The NoSQL database technology is usually adopted for following reasons -

- 1) The NoSQL databases are often used for handling big data as a part of fundamental architecture.
- 2) The NoSQL databases are used for storing and modelling structured, semi-structured and unstructured data.
- 3) For the efficient execution of database with high availability, NoSQL is used.
- 4) The NoSQL database is non-relational, so it scales out better than relational databases and these can be designed with web applications.
- 5) For easy scalability, the NoSQL is used.

6.4.3 Features

- 1) The NoSQL does not follow any relational model.
- 2) It is either schema free or have relaxed schema. That means it does not require specific definition of schema.
- 3) Multiple NoSQL databases can be executed in distributed fashion.
- 4) It can process both unstructured and semi-structured data.
- 5) The NoSQL have higher scalability.
- 6) It is cost effective.
- 7) It supports the data in the form of key-value pair, wide columns and graphs.

Review Question

1. What is NoSQL ? What is the need for it. Enlist various feature of NoSQL.

6.5 Types of NoSQL Databases

There are four types of NoSQL databases and those are -

1. Key-value store
2. Document store
3. Graph based
4. Wide column store

Let us discuss them in detail.

6.5.1 Key-Value Store

- Key-value pair is the simplest type of NoSQL database.
- It is designed in such a way to handle lots of data and heavy load.
- In the key-value storage the key is unique and the value can be JSON, string or Binary objects.
- For example -

```
{Customer:
  [
    {"id":1, "name":"Ankita"},
    {"id":2,"name":"Kavita"}
  ]
}
```

Here **id**, **name** are the keys and 1,2, "Ankita", "Prajakta" are the values corresponding to those keys.

Key-value stores help the developer to store schema-less data . They work best for Shopping Cart Contents.

The DynamoDB, Riak, Redis are some famous examples of key-value store.

6.5.2 Document Store

- The document store make use of key-value pair to store and retrieve data.
- The document is stored in the form of XML and JSON.
- The document stores appear the most natural among NoSQL database types.
- It is most commonly used due to flexibility and ability to query on any field.
- For example -

```
{
  "id" : 101,
  "Name" : "AAA",
  "City" : "Pune"
}
```

MongoDB and CouchDB are two popular document oriented NoSQL database.

6.5.3 Graph

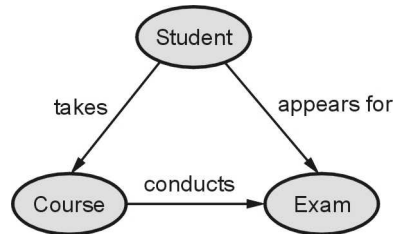
The graph database is typically used in the applications where the relationships among the data elements is an important aspect.

The connections between elements are called links or relationships. In a graph database, connections are first-class elements of the database, stored directly. In relational databases, links are implied, using data to express the relationships.

The graph database has two components -

- 1) **Node** : The entities itself. For example - People, student,
- 2) **Edge** : The relationships among the entities.

For example -



Graph base database is mostly used for social networks, logistics, spatial data. The graph databases are - Neo4J, Infinite Graph, OrientDB.

6.5.4 Wide Column Store

- Wide column store model is similar to traditional relational database. In this model, the columns are created for each row rather than having predefined by the table structure.
- In this model number of columns are not fixed for each record.
- Columns databases can quickly aggregate the value of a given column.
- For example -

Row ID	Columns...		
1	Name	City	
	Ankita	Pune	
2	Name	City	email
	Kavita	Mumbai	kavita123@gmail.com

The column store databases are widely used to manage data warehouses, business intelligence, HBase, Cassandra are examples of column based databases.

Review Question

1. Explain different types of NoSQL databases.

6.6 BASE Properties

The relational database strongly follow the ACID properties(Atomicity, Consistency, Isolation and Durability) while the NoSQL database follows BASE properties.

Base properties consists of

- 1) **Basically Available** : It means the system is guaranteed to be available in the event of failure.
- 2) **Soft State** : It means, even without an input the system state may change.
- 3) **Eventual Consistency** : The system will become consistent over time.

6.7 ACID Vs BASE

Sr. No.	ACID	BASE
1.	It stands for atomicity consistency isolation and durability.	It stands for basic availability soft state eventual consistency.
1.	It shows consistency.	It represents weak consistency.
2.	Availability is less important.	Availability of system is more important.
3.	Evolution is difficult.	Evolution is easy.
4.	It posses expensive joins and relationships.	It is free from joins and relationships.
5.	It has high maintenance costs.	It has low maintenance cost.
6.	It provides vertical scaling.	It provides horizontal scaling.

Review Question

1. Give the difference between ACID and BASE properties in DBMS.

6.8 Comparative Study of RDBMS and NoSQL

Sr. No.	RDBMS	NoSQL
1.	The relational database system is based on relationships among the tables.	It is non-relational database system. It can be used in distributed environment.
2.	It is vertically scalable.	It is horizontally scalable.
3.	It has predefined schema.	It does not have schema or it may have relaxed schema.

4.	It uses SQL to query the database.	It uses unstructured query language.
5.	It is a table based database.	It is document based, graph based or key-value pair.
6.	It emphasizes on ACID properties (Atomicity, consistency, isolation and durability)	It follows Brewers CAP theorem (Consistency, availability and partition tolerance)
7.	Schema is fixed or rigid.	Schema is dynamic.
8.	Pessimistic.	Optimistic.
9.	Examples : MySQL, Oracle, PostgreSQL	Examples : MangoDB, BigTable, Redis

Review Question

1. Give the difference between RDBMS and NoSQL.

6.9 MongoDB

- MongoDB is an **open source, document based database**.
- It is developed and supported by a company named 10gen which is now known as **MongoDB Inc**.
- The first ready version of MongoDB was released in March 2010.

Why MongoDB is needed ?

There are so many efficient RDBMS products available in the market, then why do we need MongoDB? Well, all the modern applications require Big data, faster development and flexible deployment. This need is satisfied by the document based database like MongoDB.

Features of MongoDB

- 1) It is a **schema-less, document based database system**.
- 2) It provides **high performance** data persistence.
- 3) It supports **multiple storage engines**.
- 4) It has a **rich query language support**.
- 5) MongoDB provides **high availability** and **redundancy** with the help of replication. That means it creates multiple copies of the data and sends these copies to a different server so that if one server fails, then the data is retrieved from another server.

- 6) MongoDB provides horizontal **scalability** with the help of **sharding**. Sharding means to distribute data on multiple servers.
- 7) In MongoDB, every field in the document is indexed as primary or secondary. Due to which data can be searched very efficiently from the database.

SQL Structure Vs. MongoDB

Following figure shows the terms in SQL are treated differently in MongoDB. In MongoDB the data is not stored in tables, instead of that, there is a concept called **collection** which is analogous to the **tables**. In the same manner the **rows** in RDBMS are called **documents** in MongoDB, likewise the **columns** of the record in RDBMS are called **fields**.

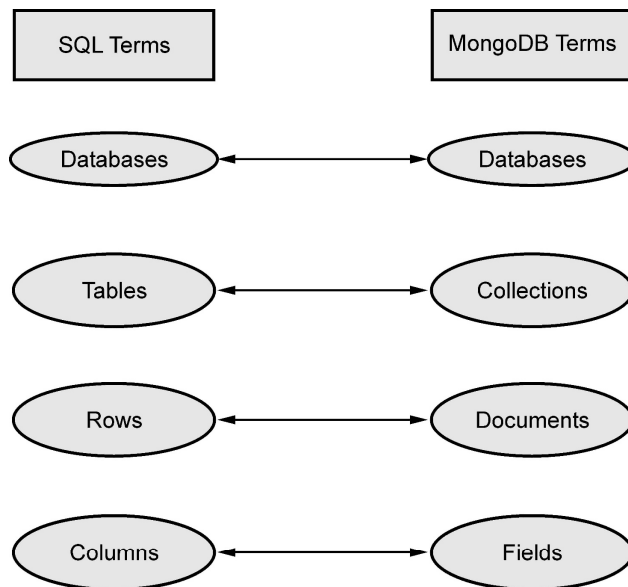


Fig. 6.9.1

Consider a student database as follows -

To the left hand side we show the database in the form of table and to the right hand side the database is shown in the form of collection.

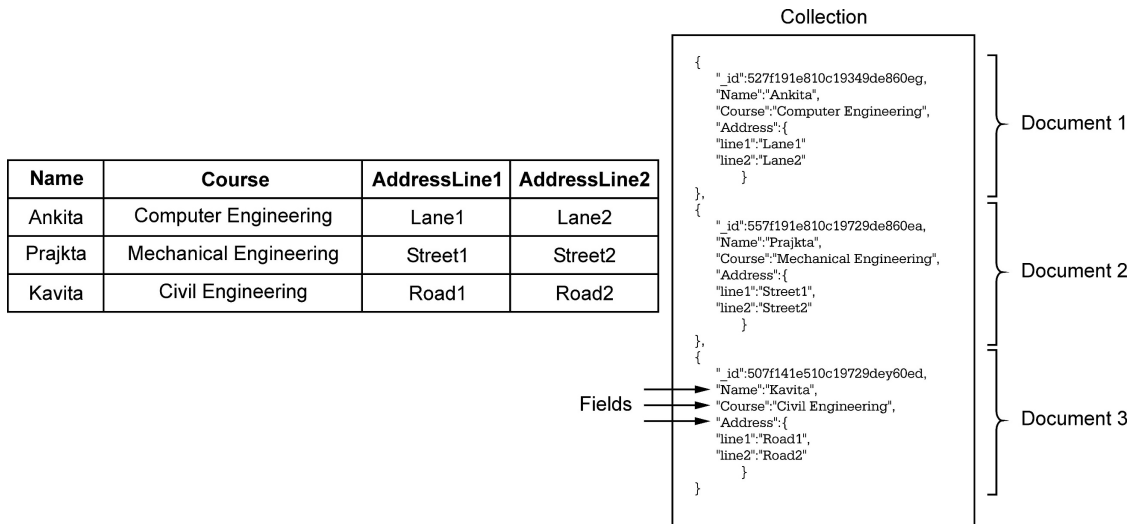


Fig. 6.9.2 Representation of relational and document based schema

6.9.1 Data Types

Following are various types of data types supported by MongoDB.

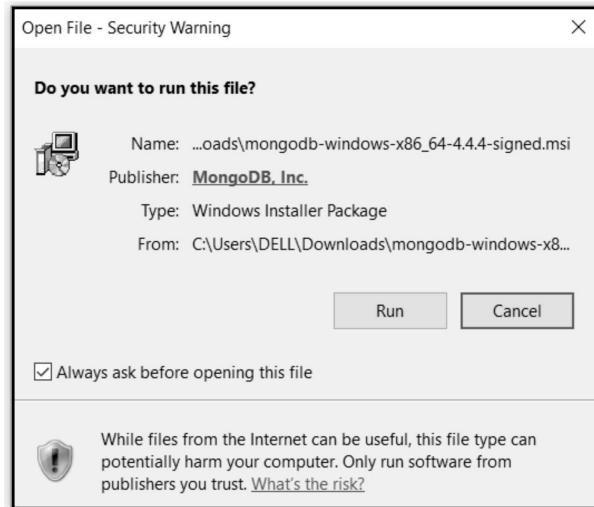
- 1) **Integer** : This data type is used for storing the numerical value.
- 2) **Boolean** : This data type is used for implementing the Boolean values i.e. true or false.
- 3) **Double** : Double is used for storing floating point data.
- 4) **String** : This is the most commonly used data type used for storing the string values.
- 5) **Min/Max keys** : This data type is used to compare a value against the lowest or highest BSON element.
- 6) **Arrays** : For storing an array or list of multiple values in one key, this data type is used.
- 7) **Object** : The object is implemented for embedded documents.
- 8) **Symbol** : This data type is similar to string data type. This data type is used to store specific symbol type.
- 9) **Null** : For storing the null values this data type is used.
- 10) **Date** : This data type is used to store current date or time. We can also create our own date or time object.
- 11) **Binary data** : In order to store binary data we need to use this data type.
- 12) **Regular expression** : This data type is used to store regular expression.

6.9.2 MongoDB Installation

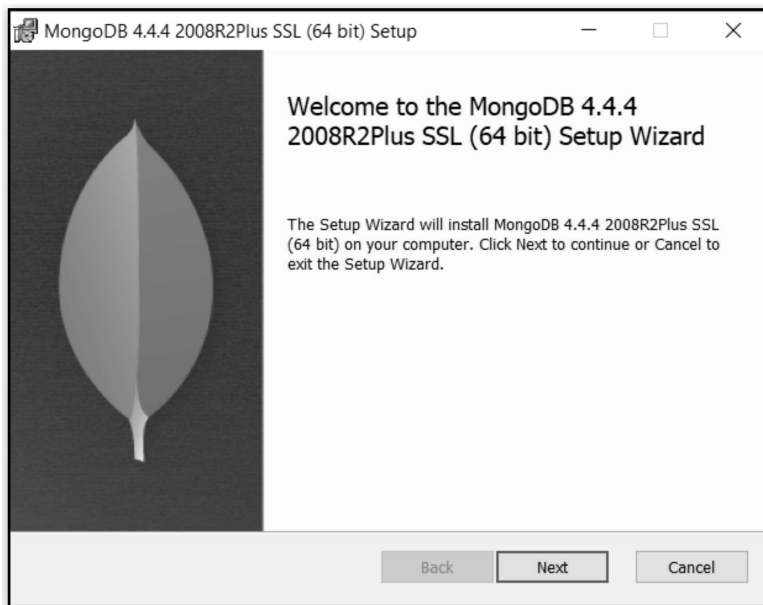
For installing the MongoDB go to the web site

<https://www.mongodb.com/try/download/community>

Choose **Software->Community Server**. The executable file gets downloaded. Click on **Run** to execute it.

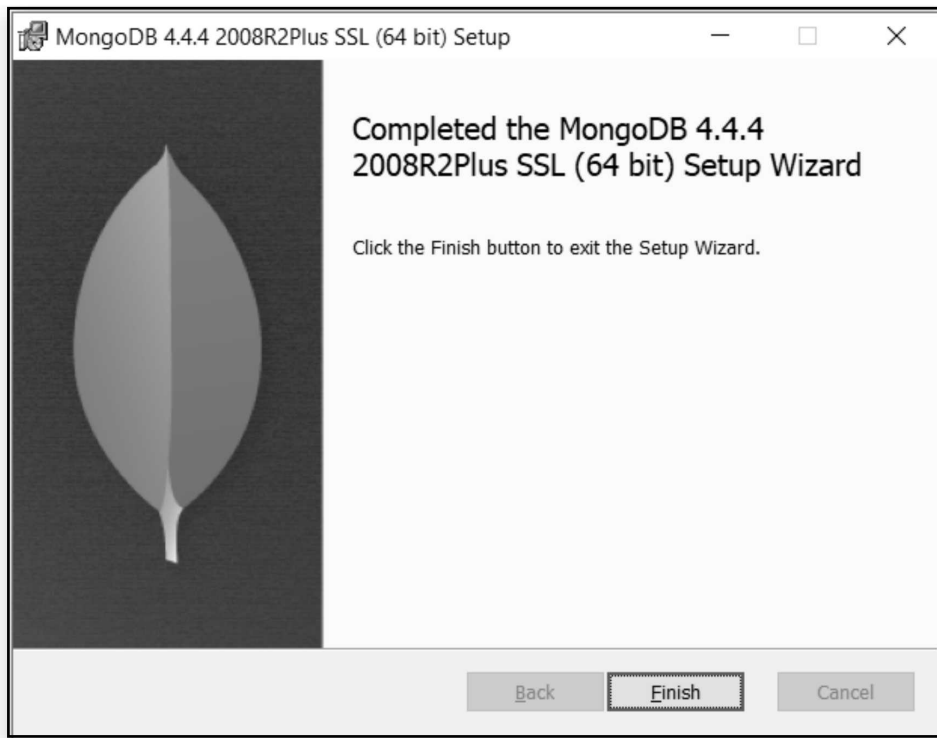


When the installation process starts following window gets popped up



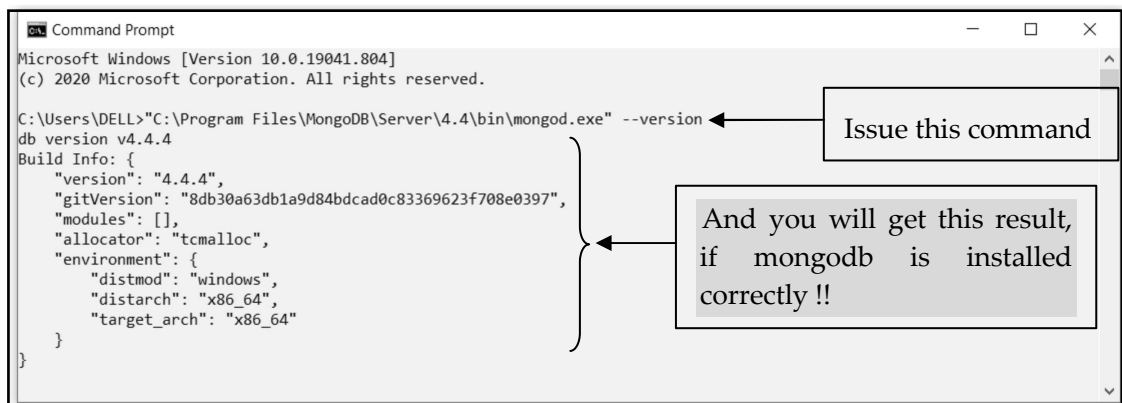
Just click on **Next** button and choose the **Complete** option for installation. Follow the normal procedure of installation by clicking the Next button.

Finally you will get following window on successful installation.



Click on Finish button, to complete the installation process.

In order to verify whether it is installed correctly or not, go to command prompt window and execute the command for mongod.exe file's version. It is illustrated as follows -

A screenshot of a Windows Command Prompt window. The title bar says "Command Prompt". The text in the window shows the command prompt path and the execution of the command:

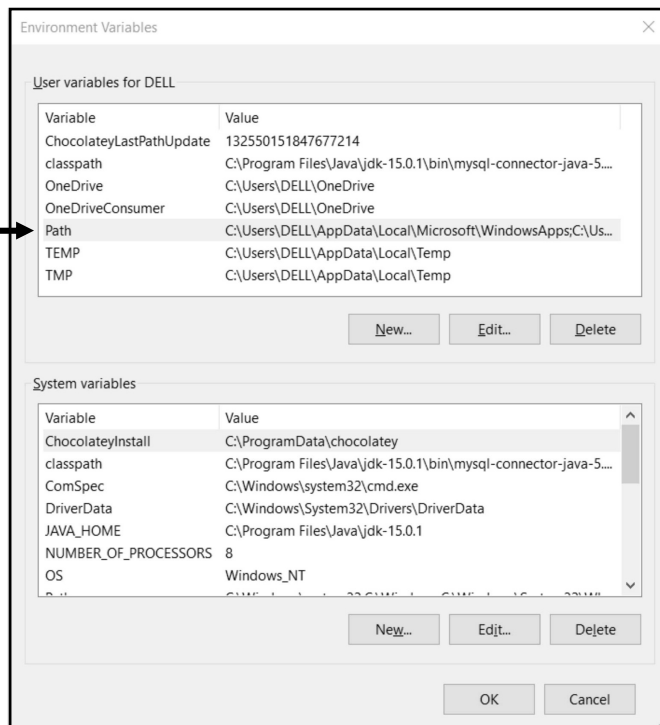
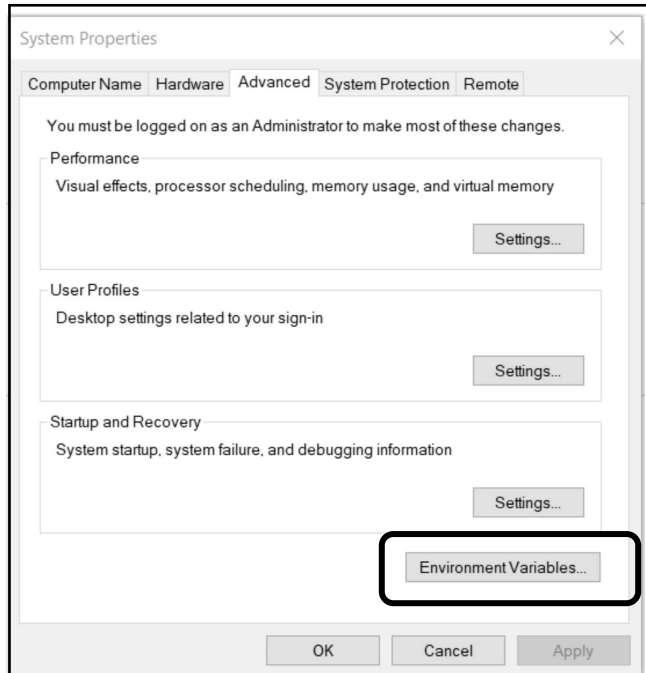
```
C:\Users\DELL>"C:\Program Files\MongoDB\Server\4.4\bin\mongod.exe" --version
```

 The output is:

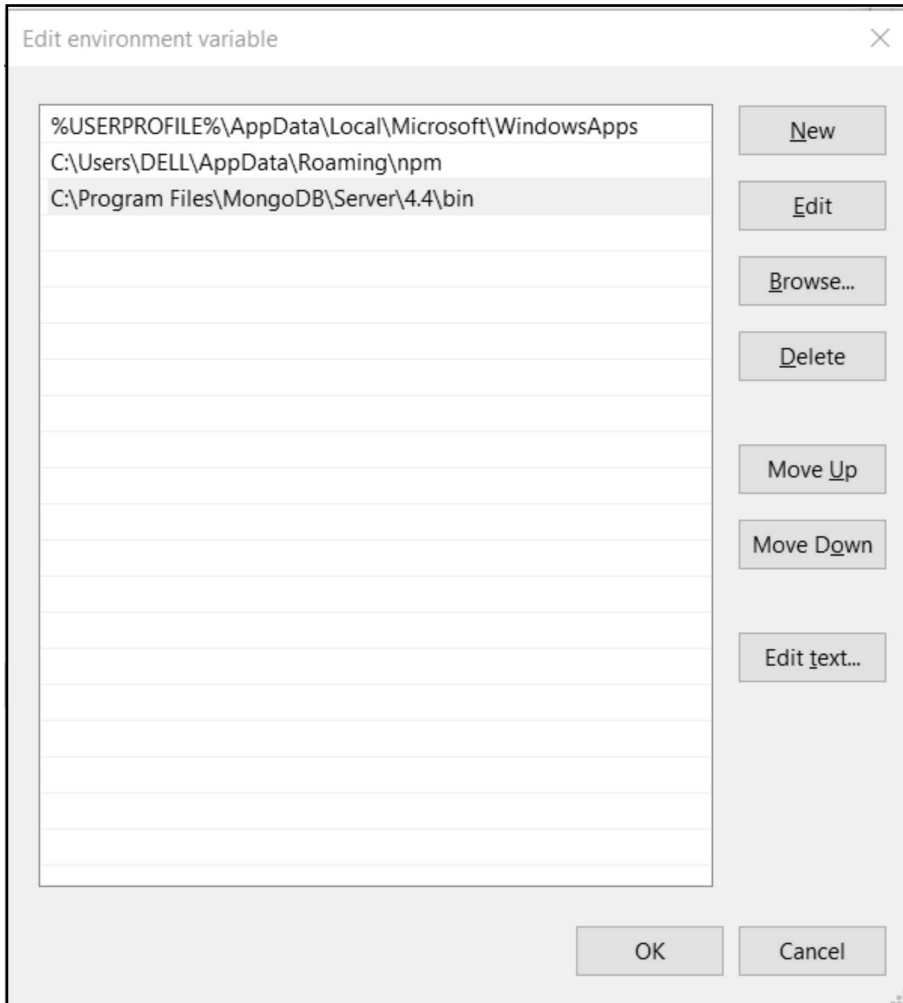
```
db version v4.4.4
Build Info: {
  "version": "4.4.4",
  "gitVersion": "8db30a63db1a9d84bdcad0c83369623f708e0397",
  "modules": [],
  "allocator": "tcmalloc",
  "environment": {
    "distmod": "windows",
    "distarch": "x86_64",
    "target_arch": "x86_64"
  }
}
```

 There are two callout boxes with arrows pointing to the command and the output. The first box says "Issue this command" and points to the command line. The second box says "And you will get this result, if mongodb is installed correctly!!" and points to the output text.

To set the environment variable, Open System Properties and click on **Environment Variables**



Then click on the **path** variable and set the path of MongoDB by clicking **New** button.

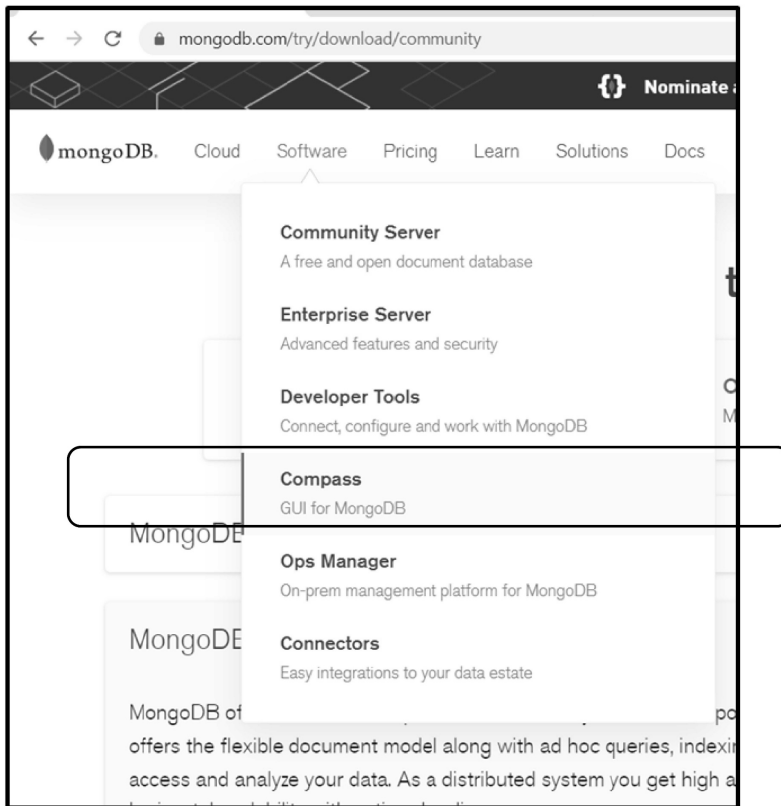


Then simply continue clicking ok button and just come out of the environment variable window.

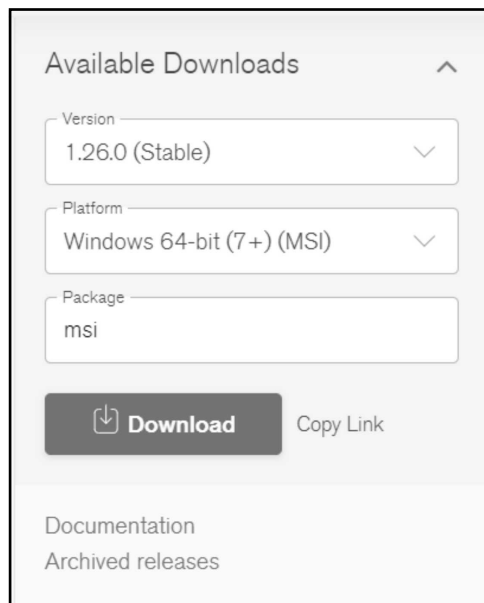
Restart your command prompt window and now simply issue the command **mongod** and then **mongo** at the command prompt window. It will recognise this command and > prompt will appear

Installing Mongodb Compass (Graphical Tool)

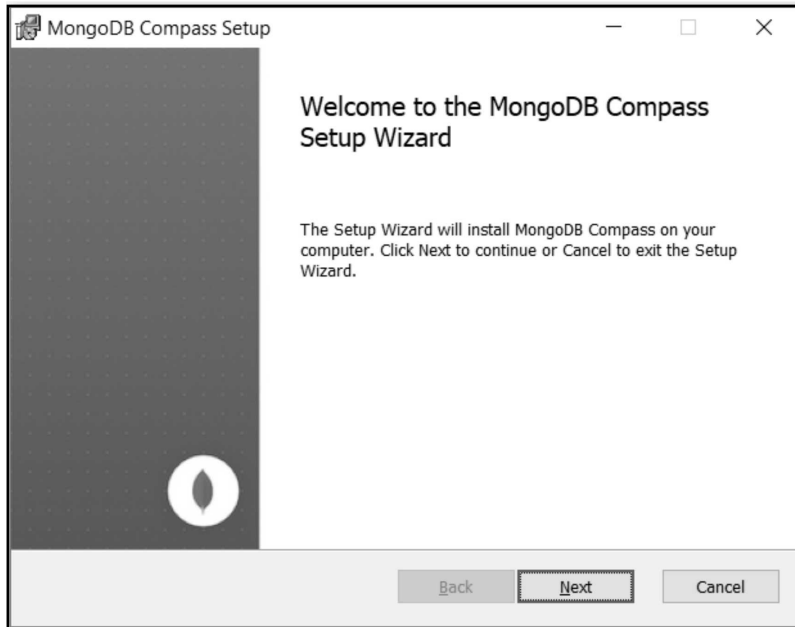
This tool is very useful for handling MongoDB database with the help of simple graphical user interface.



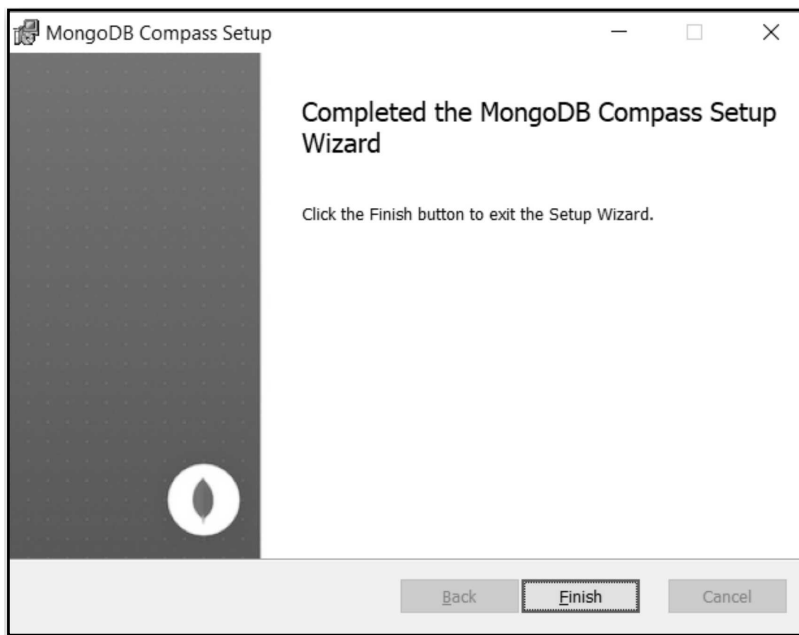
Select any suitable version as per your operating system. As mine is a Windows operating system of 64 bit, I have chosen following option.



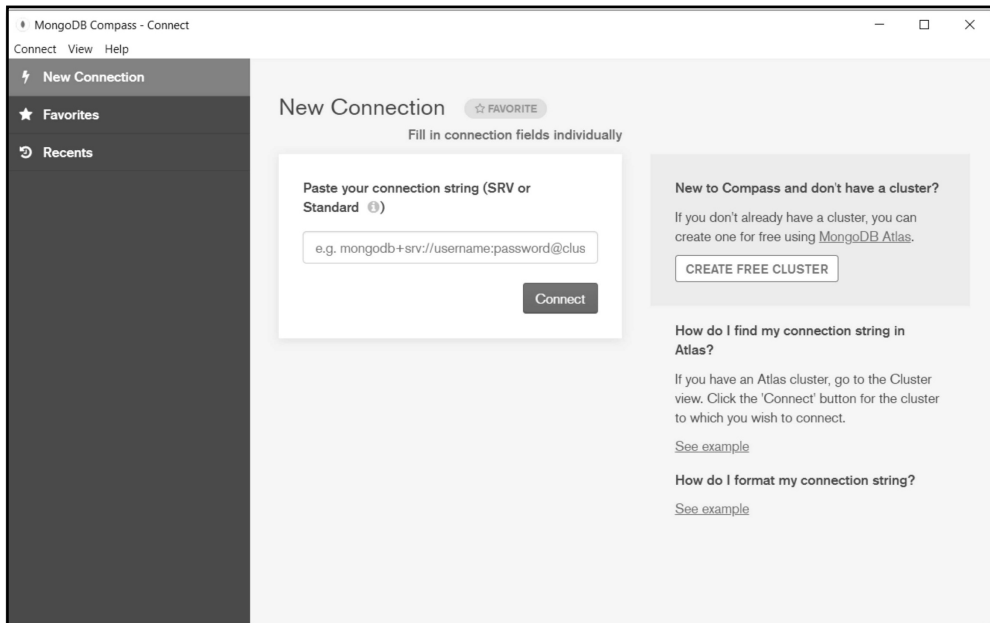
Click on the Download button. The exe file will get downloaded. Double click the installer file which is downloaded in your PC and the installation process for MongoDB Compass will start.



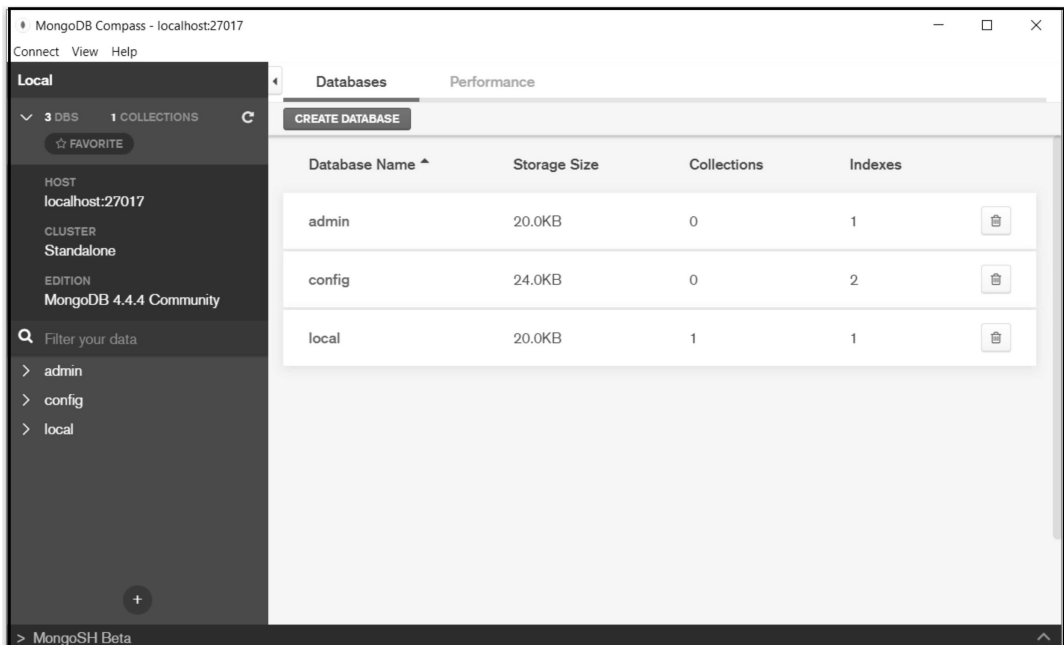
Simply go on clicking **Next** button, and then click on the install button on the subsequent window. Finally you will get the installation completion screen.



Just click Finish button. Just click the **Start** button of Windows, locate MongoDB Compass Application and simply click it to start the GUI for Mongo DB. You will get following GUI



If we click the connect button then we get following screen



6.9.3 CRUD Operations

The CRUD operation is a short form of the **Create, Read, Update** and **Delete** operations. These operations are the basic operations that are normally performed on any database.

Database Commands

In this section we will discuss how to create and handle database in MongoDB using various commands.

(1) Create Database

Open the command prompt and type the command `mongo` for starting the mongoDB. The `>` prompt will appear. For creating a database we need to use the “use” command.

Syntax

```
use Database_name
```

For example



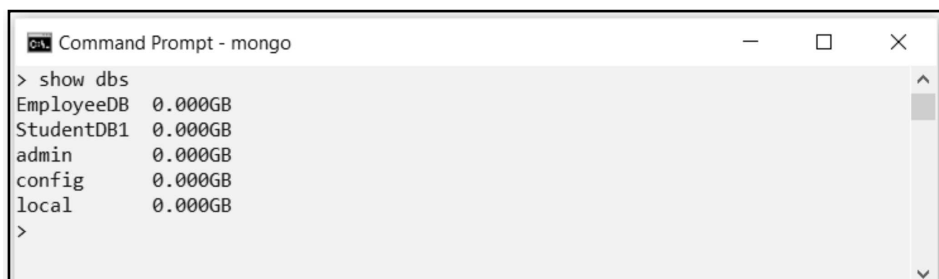
```
Command Prompt - mongo
---
> use mystudents
switched to db mystudents
>
```

To check the currently selected database, use the command **db**



```
Command Prompt - mongo
> db
mystudents
>
```

We can see the list of databases present in the MongoDB using the command **show dbs**

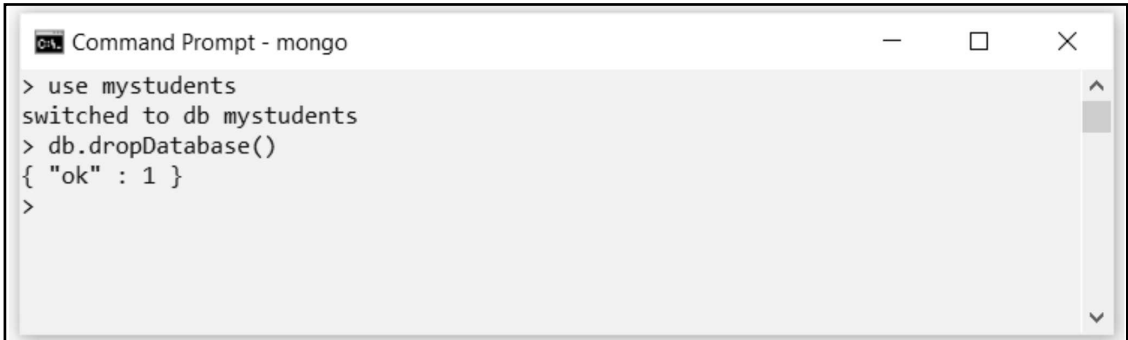


```
Command Prompt - mongo
> show dbs
EmployeeDB  0.000GB
StudentDB1  0.000GB
admin       0.000GB
config      0.000GB
local       0.000GB
>
```

Note that in above listing we can not see the **mystudents** database. This is because we have not inserted any **document** into it. To get the name of the database in the listing by means of **show** command, there should be some record present in the database.

(2) Drop Database

The **dropDatabase()** command is used to delete the database. For example



```
Command Prompt - mongo
> use mystudents
switched to db mystudents
> db.dropDatabase()
{ "ok" : 1 }
>
```

(3) Create Collection

There are two approaches of creating a collection

Method 1 : We can create a collection directly when we insert a document.

Syntax

```
db.collection_name.insert({key1:value1,key2:value2})
```

For example -



```
Command Prompt - mongo
> use EmployeeDB
switched to db EmployeeDB
> db.myemp.insert({"empname":"AAA","Salary":10000})
WriteResult({ "nInserted" : 1 })
>
```

We can cross-verify whether the collection is created or not by using following command

```

Command Prompt - mongo
> db.myemp.find()
{ "_id" : ObjectId("6053126c1ac6ebe32be47c1b"), "empname" : "AAA", "Salary" : 10000 }
>

```

Note that the one document(analogous to row) is getting inserted in the collection named **myemp**.

Method 2 : We can create **collection** explicitly using **createCollection** command.

Syntax

```
db.createCollection(name,options)
```

where

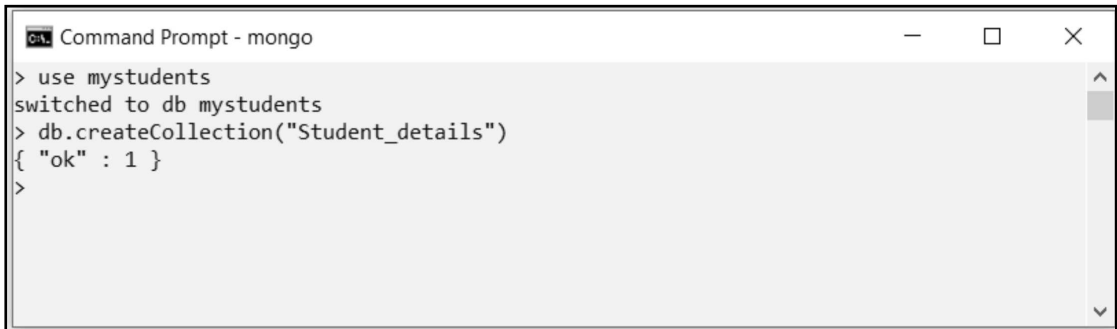
name is the name of collection

options is an optional field. This field is used to specify some parameters such as size, maximum number of documents and so on.

Following is a list of such options.

Field	Type	Description
capped	Boolean	Capped collection is a fixed size collection. It automatically overwrites the oldest entries when it reaches to maximum size. If it is set to true, enabled a capped collection. When you specify this value as true, you need to specify the size parameter.
autoIndexID	Boolean	This field is required to create index id automatically. Its default value is false.
size	Number	This value indicates the maximum size in bytes for a capped collection.
Max	Number	It specifies the maximum number of documents allowed in capped collection.

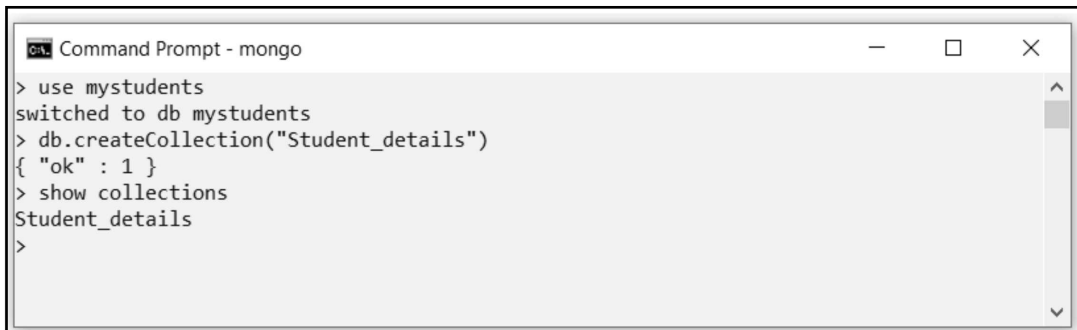
For example - Following command shows how to create collection in a database using explicit command



```
cmd Command Prompt - mongo
> use mystudents
switched to db mystudents
> db.createCollection("Student_details")
{ "ok" : 1 }
>
```

(4) Display Collection (Read Operation)

To check the created collection use the command "show collections" at the command prompt



```
cmd Command Prompt - mongo
> use mystudents
switched to db mystudents
> db.createCollection("Student_details")
{ "ok" : 1 }
> show collections
Student_details
>
```

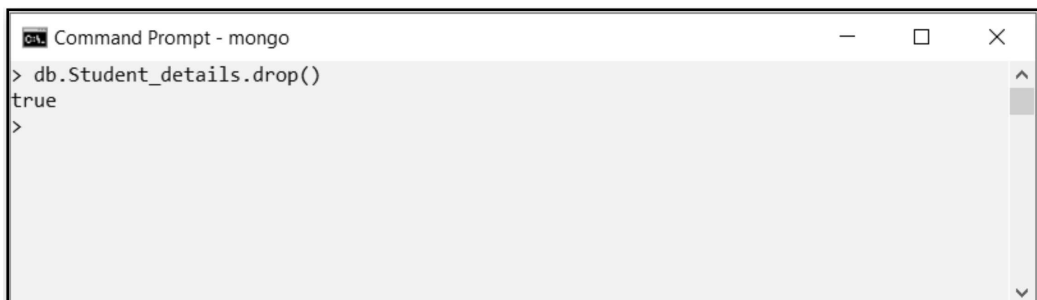
(5) Drop Collection (Delete Operation)

The drop collection command is actually used to remove the collection completely from the database. The drop command removes a collection completely from database.

Syntax

```
db.collection_name.drop()
```

For example



```
cmd Command Prompt - mongo
> db.Student_details.drop()
true
>
```

We can verify the deletion of the collection by using “show collections” in the database.

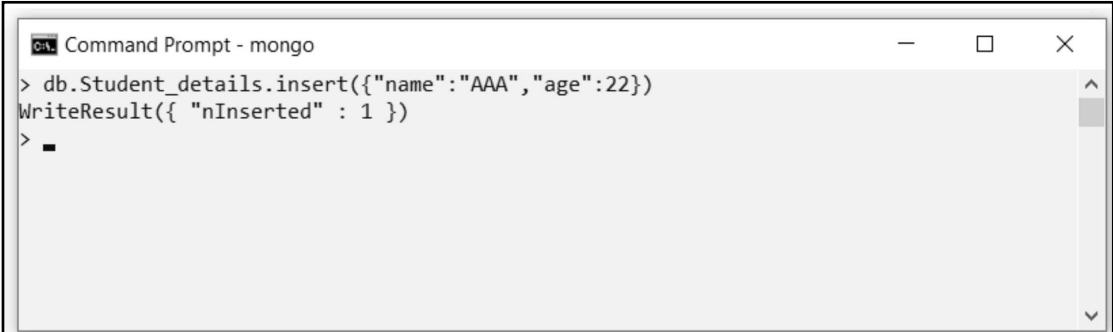
(6) Insert Documents

The document is inserted within the collection. The document is analogous to rows in database.

Syntax

```
db.collection_name.insert({key,value})
```

For example



```
Command Prompt - mongo
> db.Student_details.insert({"name":"AAA","age":22})
WriteResult({ "nInserted" : 1 })
>
```

We can verify this insertion by issuing following command



```
Command Prompt - mongo
> db.Student_details.find()
{ "_id" : ObjectId("6053357d1ac6ebe32be47c1c"), "name" : "AAA", "age" : 22 }
>
```

Inserting Multiple Documents

It is possible to insert multiple documents at a time using a single command. Following screenshot shows how to insert multiple documents in the existing collection.

```
Command Prompt - mongo
> var allStudents =
... [
... {
... "name":"BBB",
... "age":20
... },
... {
... "name":"CCC",
... "age":19
... },
... {
... "name":"DDD",
... "age":21
... },
... ];
> db.Student_details.insert(allStudents);
```

Then you will get

```
Command Prompt - mongo
> db.Student_details.insert(allStudents);
BulkWriteResult({
  "writeErrors" : [ ],
  "writeConcernErrors" : [ ],
  "nInserted" : 3,
  "nUpserted" : 0,
  "nMatched" : 0,
  "nModified" : 0,
  "nRemoved" : 0,
  "upserted" : [ ]
})
>
```

In above screenshot, as you can see that it shows number 3 in front of **nInserted**. This means that the 3 documents have been inserted by this command.

To verify the existence of these documents in the collection you can use **find** command as follows -



```
Command Prompt - mongo
> db.Student_details.find()
{ "_id" : ObjectId("6053357d1ac6ebe32be47c1c"), "name" : "AAA", "age" : 22 }
{ "_id" : ObjectId("605339001ac6ebe32be47c1d"), "name" : "BBB", "age" : 20 }
{ "_id" : ObjectId("605339001ac6ebe32be47c1e"), "name" : "CCC", "age" : 19 }
{ "_id" : ObjectId("605339001ac6ebe32be47c1f"), "name" : "DDD", "age" : 21 }
>
```

(7) Delete Documents

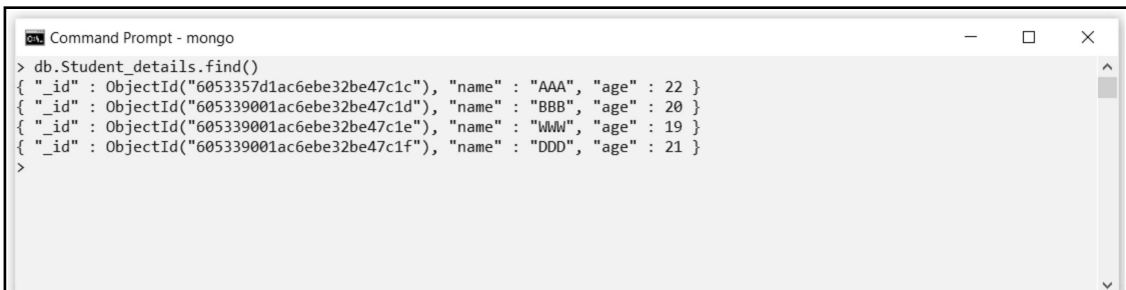
For deleting the document the **remove** command is used. This is the simplest command.

Syntax

```
db.collection_name.remove(delete_criteria)
```

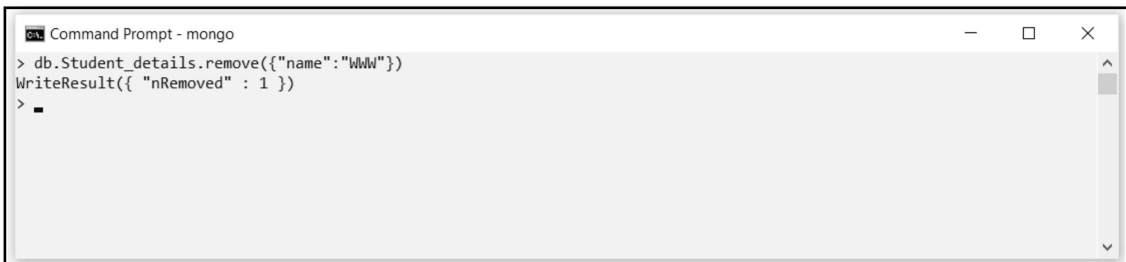
For example -

First of all we can find out the documents present in the collection using **find()** command



```
Command Prompt - mongo
> db.Student_details.find()
{ "_id" : ObjectId("6053357d1ac6ebe32be47c1c"), "name" : "AAA", "age" : 22 }
{ "_id" : ObjectId("605339001ac6ebe32be47c1d"), "name" : "BBB", "age" : 20 }
{ "_id" : ObjectId("605339001ac6ebe32be47c1e"), "name" : "WWW", "age" : 19 }
{ "_id" : ObjectId("605339001ac6ebe32be47c1f"), "name" : "DDD", "age" : 21 }
>
```

Now to delete a record with name "WWW" we can issue the command as follows -



```
Command Prompt - mongo
> db.Student_details.remove({"name":"WWW"})
WriteResult({"nRemoved" : 1 })
>
```

Now using **find()** command we can verify if the desired data is deleted or not.

```
Command Prompt - mongo
> db.Student_details.find()
{ "_id" : ObjectId("6053357d1ac6ebe32be47c1c"), "name" : "AAA", "age" : 22 }
{ "_id" : ObjectId("605339001ac6ebe32be47c1d"), "name" : "BBB", "age" : 20 }
{ "_id" : ObjectId("605339001ac6ebe32be47c1f"), "name" : "DDD", "age" : 21 }
>
```

Deleting only one Document

Sometimes the delete criteria matches for more than one records and in such situation, we can forcefully tell the MongoDB to delete only one document.

Syntax

```
db.collection_name.remove(delete_criteria, justOne)
```

The **justOne** is a Boolean value it can be 1 or 0. If we pass this value as 1 then only one document will get deleted.

For example

```
Command Prompt - mongo
> db.Student_details.remove({"age":20},1)
WriteResult({ "nRemoved" : 1 })
>
```

Now it can be verified using **find()** command as follows -

```
Command Prompt - mongo
> db.Student_details.find()
{ "_id" : ObjectId("6053357d1ac6ebe32be47c1c"), "name" : "AAA", "age" : 22 }
{ "_id" : ObjectId("605339001ac6ebe32be47c1f"), "name" : "DDD", "age" : 21 }
{ "_id" : ObjectId("6054586e5783af51edeb3774"), "name" : "EEE", "age" : 20 }
>
```

Note that there were two records that were matching with age = 20 with name “BBB” and “EEE”, but since we have passed **justOne** attribute as 1, we get the result by deleting the single record having name “BBB”

Remove all the documents

It is possible to remove all the documents present in the collection with the help of single command.

Syntax

```
db.collection_name.remove({})
```

For example



```
Command Prompt - mongo
> db.Student_details.remove({})
WriteResult{"nRemoved" : 3 }
> db.Student_details.find()
>
```

(8) Update Documents

For updating the document we have to provide some criteria based on which the document can be updated.

Syntax

```
db.collection_name.update(criteria,update_data)
```

For example - Suppose the collection “Student_details” contain following documents



```
Command Prompt - mongo
> db.Student_details.find()
{ "_id" : ObjectId("6053357d1ac6ebe32be47c1c"), "name" : "AAA", "age" : 22 }
{ "_id" : ObjectId("605339001ac6ebe32be47c1d"), "name" : "BBB", "age" : 20 }
{ "_id" : ObjectId("605339001ac6ebe32be47c1e"), "name" : "CCC", "age" : 19 }
{ "_id" : ObjectId("605339001ac6ebe32be47c1f"), "name" : "DDD", "age" : 21 }
>
```

And we want to change the name "CCC" to "WWW", then the command can be issued as

```
Command Prompt - mongo
> db.Student_details.update({"name":"CCC"},{$set:{"name":"WWW"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

This can be verified as

```
Command Prompt - mongo
> db.Student_details.find()
{ "_id" : ObjectId("6053357d1ac6ebe32be47c1c"), "name" : "AAA", "age" : 22 }
{ "_id" : ObjectId("605339001ac6ebe32be47c1d"), "name" : "BBB", "age" : 20 }
{ "_id" : ObjectId("605339001ac6ebe32be47c1e"), "name" : "WWW", "age" : 19 }
{ "_id" : ObjectId("605339001ac6ebe32be47c1f"), "name" : "DDD", "age" : 21 }
>
```

Thus the document gets updated.

By default the **update** command updates a single document. But we can update multiple documents as well. For that purpose we have to add {multi:true}

For example

```
db.Student_details.update({"age":21},{$set:{"age":23}},{multi:true})
```

(9) Sorting

We can use the **sort()** method for arranging the documents in ascending or descending order based on particular field of document.

Syntax

For displaying the documents in **ascending order** we should **pass value 1**

```
db.collection_name.find().sort({field_name:1})
```

If we **pass -1** then the document will be displayed in the **descending order** of the field.

For example

Suppose the collection contains following documents

```
Command Prompt - mongo
> db.Student_details.find()
{ "_id" : ObjectId("60545e5c5783af51edeb3775"), "name" : "AAA", "age" : 22 }
{ "_id" : ObjectId("60545e695783af51edeb3776"), "name" : "BBB", "age" : 21 }
{ "_id" : ObjectId("60545e735783af51edeb3777"), "name" : "CCC", "age" : 23 }
{ "_id" : ObjectId("60545e7d5783af51edeb3778"), "name" : "DDD", "age" : 20 }
{ "_id" : ObjectId("60545f035783af51edeb3779"), "name" : "EEE", "age" : 21 }
{ "_id" : ObjectId("60545f125783af51edeb377a"), "name" : "FFF", "age" : 24 }
{ "_id" : ObjectId("60545f1a5783af51edeb377b"), "name" : "GGG", "age" : 22 }
>
```

Now to sort the data in descending order

```
Command Prompt - mongo
> db.Student_details.find().sort({"age":-1})
{ "_id" : ObjectId("60545f125783af51edeb377a"), "name" : "FFF", "age" : 24 }
{ "_id" : ObjectId("60545e735783af51edeb3777"), "name" : "CCC", "age" : 23 }
{ "_id" : ObjectId("60545e5c5783af51edeb3775"), "name" : "AAA", "age" : 22 }
{ "_id" : ObjectId("60545f1a5783af51edeb377b"), "name" : "GGG", "age" : 22 }
{ "_id" : ObjectId("60545e695783af51edeb3776"), "name" : "BBB", "age" : 21 }
{ "_id" : ObjectId("60545f035783af51edeb3779"), "name" : "EEE", "age" : 21 }
{ "_id" : ObjectId("60545e7d5783af51edeb3778"), "name" : "DDD", "age" : 20 }
>
```

If we want to display data in ascending order we issue following command

```
Command Prompt - mongo
> db.Student_details.find().sort({"age":1})
{ "_id" : ObjectId("60545e7d5783af51edeb3778"), "name" : "DDD", "age" : 20 }
{ "_id" : ObjectId("60545e695783af51edeb3776"), "name" : "BBB", "age" : 21 }
{ "_id" : ObjectId("60545f035783af51edeb3779"), "name" : "EEE", "age" : 21 }
{ "_id" : ObjectId("60545e5c5783af51edeb3775"), "name" : "AAA", "age" : 22 }
{ "_id" : ObjectId("60545f1a5783af51edeb377b"), "name" : "GGG", "age" : 22 }
{ "_id" : ObjectId("60545e735783af51edeb3777"), "name" : "CCC", "age" : 23 }
{ "_id" : ObjectId("60545f125783af51edeb377a"), "name" : "FFF", "age" : 24 }
>
```

6.9.4 Indexing

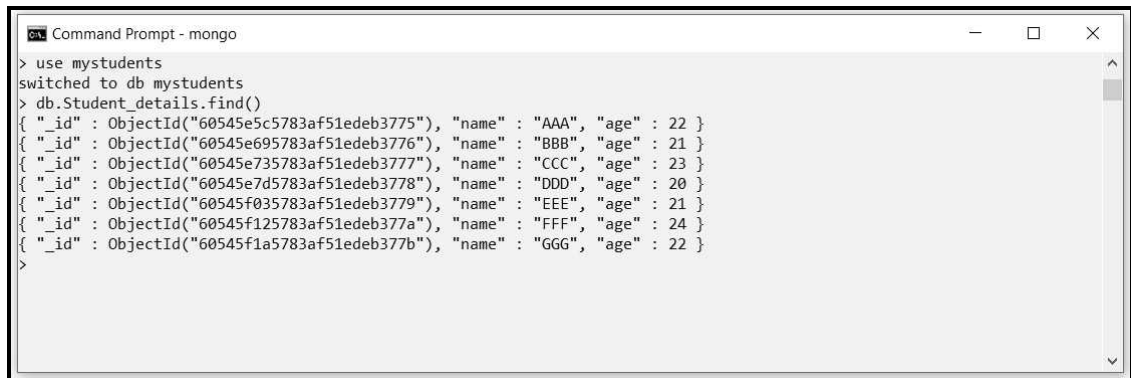
For efficient execution of queries the indexing is used. If we use the query without indexes then the execution of that query will be very slow.

Definition of index : Index is a special data structure that store small part of collection's data in such a way that we can use it in querying.

The index store the values of index fields outside the table or collection and keep track of their location in the disk.

Demonstration :

In order to create an index, we will use the database named **mystudents** (It is created in section 6.9.1). The collection **Student_details** will be used for creating index.



```
Command Prompt - mongo
> use mystudents
switched to db mystudents
> db.Student_details.find()
{ "_id" : ObjectId("60545e5c5783af51edeb3775"), "name" : "AAA", "age" : 22 }
{ "_id" : ObjectId("60545e695783af51edeb3776"), "name" : "BBB", "age" : 21 }
{ "_id" : ObjectId("60545e735783af51edeb3777"), "name" : "CCC", "age" : 23 }
{ "_id" : ObjectId("60545e7d5783af51edeb3778"), "name" : "DDD", "age" : 20 }
{ "_id" : ObjectId("60545f035783af51edeb3779"), "name" : "EEE", "age" : 21 }
{ "_id" : ObjectId("60545f125783af51edeb377a"), "name" : "FFF", "age" : 24 }
{ "_id" : ObjectId("60545f1a5783af51edeb377b"), "name" : "GGG", "age" : 22 }
>
```

1) Index creation

Syntax for creating index is

```
db.<collection>.createIndex({KEY:1})
```

The key determines the field on the basis of which the index is created. After the colon the direction of the key(1 or -1) is used to indicate ascending or descending order.

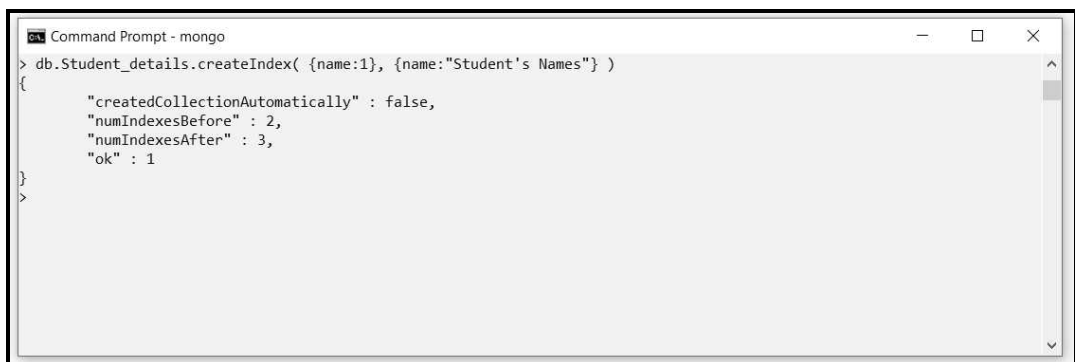
The MongoDB will generate index names by concatenating the indexed keys with the direction of each key with underscore as separator. For example if the index is on the field **name** and the order as **1** then the index will be created as **name_1**.

We can also use name option to define custom index name while creating the index.

For example -

```
> db.Student_details.createIndex( {name:1}, {name:"Student's Names"} )
```

The result will be as follows -



```
Command Prompt - mongo
> db.Student_details.createIndex( {name:1}, {name:"Student's Names"} )
{
  "createdCollectionAutomatically" : false,
  "numIndexesBefore" : 2,
  "numIndexesAfter" : 3,
  "ok" : 1
}
>
```

Thus index gets created on the collection **Student_details**.

2) Find Index

We can find all the available indexes in the MongoDB by using **getIndexes** method.

Syntax

```
db.<collection>.getIndexes()
```

For example –

```
>db.Student_details.getIndexes()
```

The result will be



```
Command Prompt - mongo
> db.Student_details.getIndexes()
[
  {
    "v" : 2,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_"
  },
  {
    "v" : 2,
    "key" : {
      "name" : 1
    },
    "name" : "Student's Names"
  }
]
```

Note that the output contains the default_id index and user created index.

3) Drop Index

To delete an index we use **dropIndex** method.

Syntax

```
db.<collection>.dropIndex(Index Name)
```

For example

```
db.Student_details.dropIndex("Student's Names")
```

The result will be



```
Command Prompt - mongo
> db.Student_details.dropIndex("Student's Names")
{ "nIndexesWas" : 2, "ok" : 1 }
>
```

4) Compound Index

We can create index on multiple fields in MongoDB document. For example

```
db.Student_details.createIndex({name:1, age:-1})
```

It will sort by name in ascending order and age in descending order. If the names are same then the descending order of age can be noticed.

6.9.5 Aggregation

Aggregation is an operation used to process the data that results the computed results. The aggregation groups the data from multiple documents and operate on grouped data to get the combined result. The MongoDB aggregation is equivalent to count(*) and with **group by** in sql. MongoDB supports the concept of aggregation framework. The typical pipeline of aggregation framework is as follows -

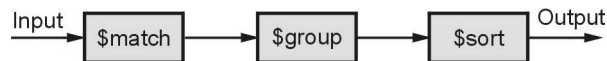


Fig. 6.9.3 Aggregation framework

- 1) **\$match() stage** - filters those documents we need to work with, those that fit our needs
- 2) **\$group() stage** - does the aggregation job
- 3) **\$sort() stage** - sorts the resulting documents the way we require (ascending or descending)

The input of the pipeline can be one or several collections. The pipeline then performs successive transformations on the data and the result is obtained.

Syntax for aggregate operation

```
db.collection_name.aggregate(aggregate_operation)
```

Demo Example

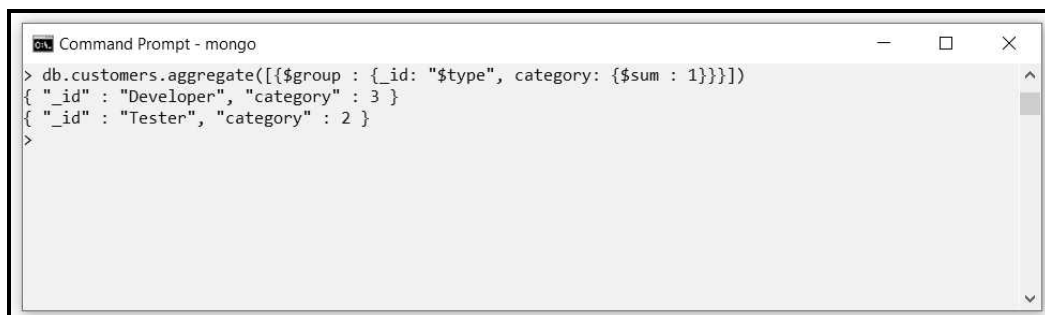
For demonstration purpose, I have created a database named **CustomerDB** inside which there is a collection document named **customers**. Some data is already inserted into it. The contents of **customers** document are as shown below -

```
Command Prompt - mongo
> use CustomerDB
switched to db CustomerDB
> db.customers.find()
{ "_id" : ObjectId("610e57c1128e95223b716fcf"), "name" : "AAA", "type" : "Developer" }
{ "_id" : ObjectId("610e57ff128e95223b716fd0"), "name" : "BBB", "type" : "Tester" }
{ "_id" : ObjectId("610e5834128e95223b716fd1"), "name" : "CCC", "type" : "Tester" }
{ "_id" : ObjectId("610e585c128e95223b716fd2"), "name" : "DDD", "type" : "Developer" }
{ "_id" : ObjectId("610e587d128e95223b716fd3"), "name" : "EEE", "type" : "Developer" }
>
```


Now issue the following command

```
> db.customers.aggregate([{$group : {_id: "$type", category: {$sum : 1}}]})
```

The result will be –



```
Command Prompt - mongo
> db.customers.aggregate([{$group : {_id: "$type", category: {$sum : 1}}]})
{ "_id" : "Developer", "category" : 3 }
{ "_id" : "Tester", "category" : 2 }
>
```

The above command will display total categories of the customers - In our database there are two types of customers - "Developer" and "Tester". There are 3 developers and 2 testers in the collection document. The aggregate function is applied on the **\$group**

Similarly if we want to find only **Developers** from the collection document **customers** then we use **\$match** for aggregate function. The demonstration is as follows -



```
Command Prompt - mongo
> db.customers.aggregate([{$match : {type: "Developer"}}]})
{ "_id" : ObjectId("610e57c1128e95223b716fcf"), "name" : "AAA", "type" : "Developer" }
{ "_id" : ObjectId("610e585c128e95223b716fd2"), "name" : "DDD", "type" : "Developer" }
{ "_id" : ObjectId("610e587d128e95223b716fd3"), "name" : "EEE", "type" : "Developer" }
>
```

Expressions used by Aggregate function

Expression	Description
\$sum	Summates the defined values from all the documents in a collection
\$avg	Calculates the average values from all the documents in a collection
\$min	Return the minimum of all values of documents in a collection
\$max	Return the maximum of all values of documents in a collection
\$addToSet	Inserts values to an array but no duplicates in the resulting document
\$push	Inserts values to an array in the resulting document
\$first	Returns the first document from the source document
\$last	Returns the last document from the source document

6.9.6 Map Reduce

Map reduce is a data processing programming model that helps in performing operations on large data sets and produce aggregate results.

Map reduce is used for large volume of data. The syntax for map reduce is

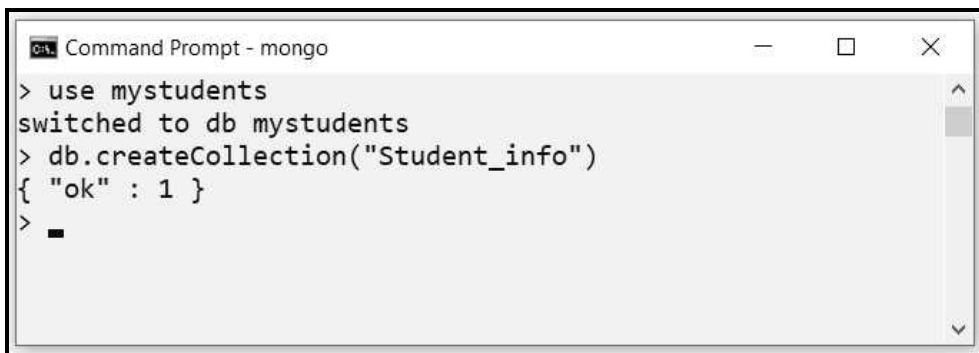
```
>db.collection.mapReduce(  
  function() {emit(key,value);}, ← map function  
  function(key,values) {return reduceFunction}, { ← reduce function  
    out: collection, ← the collection is created in which the result of mapReduce can be stored  
    query: document,  
    sort: document,  
    limit: number  
  }  
)
```

Where

- 1) **map Function** : It uses emit() function in which it takes two parameters key and value key. Here the key is on which we make groups(such as group by name, or age) and the second parameter is on which aggregation is performed like avg(), sum() is calculated on.
- 2) **reduce Function** : This is a function in which we perform aggregate functions like avg(), sum()
- 3) **out** : It will specify the collection name where the result will be stored.
- 4) **query** : We will pass the query to filter the resultset.
- 5) **sort** : It specifies the optional sort criteria.
- 6) **limit** : It specifies the optional maximum number of documents to be returned.

Demo Example :

Step 1 : Create a collection inside the database **mystudents**. The collection is created using name **Student_info**.



```
Command Prompt - mongo  
> use mystudents  
switched to db mystudents  
> db.createCollection("Student_info")  
{ "ok" : 1 }  
> █
```

Step 2 : Now we will insert documents inside the collection **Student_info** using following command

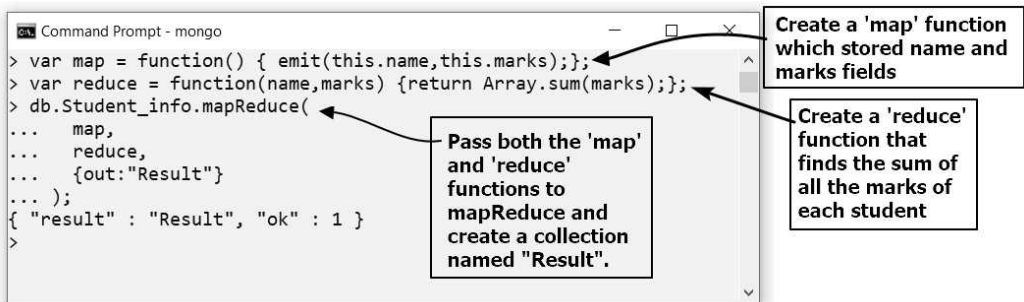
```
db.Student_info.insertMany( [
{name:"Ankita", marks:96},
  {name:"Ankita", marks:86},
  {name:"Ankita", marks:92},
  {name:"Kavita", marks:87},
  {name:"Kavita", marks:74},
  {name:"Kavita", marks:86 }
])
```

Now in order to display the contents of the collection we issue the find() command.



```
Command Prompt - mongo
> db.Student_info.find().pretty() ← Issue this command to display the documents in the collection
{
  "_id" : ObjectId("6110ce2a85089e2d2c7fac22"),
  "name" : "Ankita",
  "marks" : 96
}
{
  "_id" : ObjectId("6110ce2a85089e2d2c7fac23"),
  "name" : "Ankita",
  "marks" : 86
}
{
  "_id" : ObjectId("6110ce2a85089e2d2c7fac24"),
  "name" : "Ankita",
  "marks" : 92
}
{
  "_id" : ObjectId("6110ce2a85089e2d2c7fac25"),
  "name" : "Kavita",
  "marks" : 87
}
{
  "_id" : ObjectId("6110ce2a85089e2d2c7fac26"),
  "name" : "Kavita",
  "marks" : 74
}
{
  "_id" : ObjectId("6110ce2a85089e2d2c7fac27"),
  "name" : "Kavita",
  "marks" : 86
}
>
```

Step 3 : Now we will apply mapReduce function

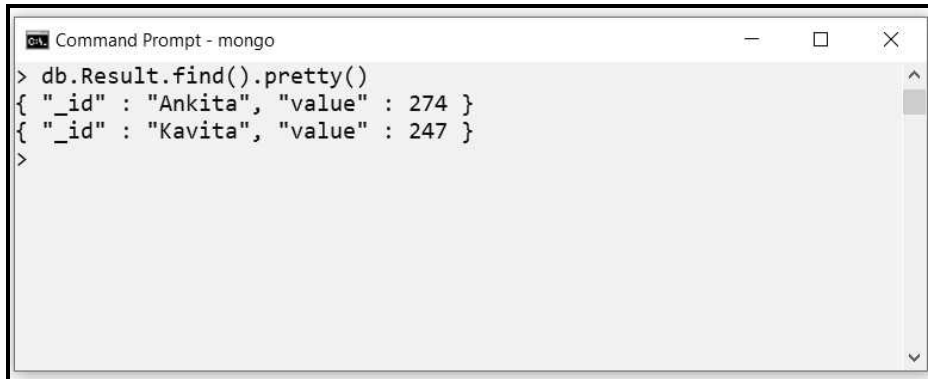


```
Command Prompt - mongo
> var map = function() { emit(this.name,this.marks);};
> var reduce = function(name,marks) {return Array.sum(marks);};
> db.Student_info.mapReduce(
...   map,
...   reduce,
...   {out:"Result"}
... );
{ "result" : "Result", "ok" : 1 }
>
```

Annotations:

- Create a 'map' function which stored name and marks fields
- Create a 'reduce' function that finds the sum of all the marks of each student
- Pass both the 'map' and 'reduce' functions to mapReduce and create a collection named "Result".

Step 4 : The output of the **mapReduce** can be seen with help of **find()** command. It is illustrated by following screenshot -



```
Command Prompt - mongo
> db.Result.find().pretty()
{ "_id" : "Ankita", "value" : 274 }
{ "_id" : "Kavita", "value" : 247 }
>
```

Advantages of mapReduce

- 1) MapReduce allows the developer to store complex result in separate collection.
- 2) MapReduce provides the tools to create incremental aggregation over large collections.
- 3) It is flexible.

6.9.7 Replication

Replication is process of making data available across multiple data servers

The replication is mainly used for security purpose. In case of sever failure and hardware failure the replication is used to restore the data.

The replication in MongoDB is carried out with the help of **replica sets**.

The replica sets are combination of various MongoDB instances having single primary node and multiple secondary nodes. The secondary node automatically copies the changes made to primary in order to maintain the data across all servers. Refer Fig. 6.9.4.

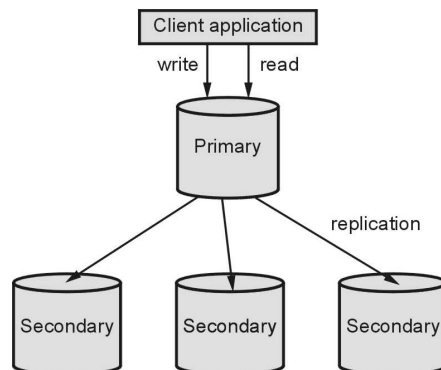


Fig. 6.9.4 Concept of replication

If the primary node gets failed then the secondary node will take primary node's role to provide continuous availability of data. In this case the primary node selection is made by the process called **replica set elections**. In this process the most suitable secondary node will be selected as new primary node.

Benefits of Replication

- 1) Replication is used for data availability.
- 2) It is helpful for handling the situations such as hardware failure and server crash.
- 3) It enhances the performance as data is available across multiple machines and servers.

6.9.8 Sharding

- Sharding is a concept in MongoDB, which splits large data sets into small data sets across multiple MongoDB instances.
- It is not replication of data, but amassing different data from different machines.
- Sharding allows horizontal scaling of data stored in multiple shards or multiple collections. Logically all the shards work as one collection.
- Sharding works by creating a **cluster of MongoDB** instances which is made up of three components.
 - **Shard** : It is a a mongodb instance that contains the sharded data. The combination of multiple shards create a complete data set.
 - **Router** : This is a mongodb instance which is basically responsible for directing the commands sent by client to appropriate server.
 - **Config server** : This is mongodb instance which holds the information about various mongodb instances which hold the shard data.

Review Questions

1. Explain how to perform CRUD operations in MongoDB with illustrative examples.
2. What is replication ? Enlist the advantages of replication in database systems.
3. What is the purpose of mapReduce. Explain it with suitable example.
4. Explain the concept of sharding.

- Q.8** What is the aim of NoSQL ?
- a) NoSQL databases allow storing non-structured data.
 - b) NoSQL provides an alternative to SQL databases to store textual.
 - c) NoSQL is a new data format to store large datasets.
 - d) NoSQL is not suitable for storing structured data.
- Q.9** _____ represent column in NoSQL.
- a) Database
 - b) Field
 - c) Document
 - d) Collection
- Q.10** Which of the following is a NoSQL database type ?
- a) SQL
 - b) Document databases
 - c) JSON
 - d) All of the mentioned
- Q.11** _____ stores are used to store information about networks, such as social connections.
- a) Key-value
 - b) Wide-column
 - c) Document
 - d) Graph
- Q.12** Which of the following is wide-column store ?
- a) Cassandra
 - b) Riak
 - c) MongoDB
 - d) Redis
- Q.13** What is the core principle of NoSQL ?
- a) High availability
 - b) Low availability
 - c) Both high & low availability
 - d) None of above
- Q.14** MongoDB written in _____.
- a) C
 - b) C++
 - c) JavaScript
 - d) All of the above
- Q.15** MongoDB stores all documents in _____.
- a) tables
 - b) collections
 - c) rows
 - d) all of the mentioned
- Q.16** Which of the following format is supported by MongoDB ?
- a) SQL
 - b) XML
 - c) BSON or JSON like
 - d) All of the mentioned
- Q.17** Instead of primary key MongoDB uses _____.
- a) Mongo key_id
 - b) Mongo_id
 - c) Default key_id
 - d) None of the above

7

Advances in Databases

Syllabus

Emerging Databases : Active and Deductive Databases, Main Memory Databases, Semantic Databases.

Complex Data Types : Semi-Structured Data, Features of Semi-Structured Data Models. **Nested Data Types** : JSON, XML. **Object Orientation** : Object-Relational Database System, Table Inheritance, Object-Relational Mapping. **Spatial Data** : Geographic Data, Geometric Data.

Contents

7.1 Emerging Databases

7.2 Complex Data Types

7.3 Nested Data Types

7.4 Object Orientation

7.5 Spatial Data

Multiple Choice Questions

7.1 Emerging Databases

- Data is growing rapidly. Day by day it is becoming complex to handle such huge amount of data properly.
- In order to use the data efficiently the database systems are supported by modern tools and techniques. Emerging databases are those databases that are heavily influenced both by the evolution of software applications, and by advances in computing hardware and operating system design.
- In this chapter we will get introduced by advances in databases by learning the concept of active and deductive databases, complex and nested data types, software technologies such as JSON and XML, object oriented database management systems and spatial databases.

7.1.1 Active and Deductive Databases

Active Databases

- Active databases are the databases which consists of triggers. The situation and action rules are embedded in the active databases. The active databases are able to react automatically to the situations in the database. The trigger is a technique for specifying certain types of active rules. The commercial databases such as Oracle, DB2, Microsoft SQLServer allows the use of triggers.

Generalized Model for Active Database

The general model for active database is considered as **Event-Condition-Action(ECA)** model. This model has three components –

- i) Event :** The events are database update operations that are performed explicitly on the databases.
- ii) Condition :** The condition determines whether the rule action should be executed. If the action is not specified then the action will be executed automatically on occurrence of the event.
- iii) Action :** The action is usually a sequence of SQL statements. It could be a database transaction or external program that will be executed on occurrence of condition.

```

CREATE TABLE PersonTab (
pname VARCHAR2(20)
);

CREATE OR REPLACE TRIGGER MyTrigger
BEFORE INSERT ON PersonTab
FOR EACH ROW
ENABLE
DECLARE
usr_name VARCHAR2(20);
BEGIN
SELECT user INTO usr_name FROM dual;
DBMS_OUTPUT.PUT_LINE('Inserted a new row by user: ' || usr_name);
END;
/

INSERT INTO PersonTab VALUES('Sharda');

```

Here table named **PersonTab** is created

Code for actual trigger

SQL query on execution of which the trigger gets fired.

Deductive Database

- Deductive database is a database system that can make deductions based on rules and facts stored in the database.
- Deductive databases use the concept of logic programming for specifying the rules and the facts. Prolog is a popular programming language which is based on the concept of logic programming.
- There are two types of specifications used in deductive databases –
 - 1) **Facts** : Facts are specified as the same way the relations are specified in the Relational Database except it is not necessary to include the attribute names. The meaning of an attribute value in a tuple is determined solely by its position in the tuple.
 - 2) **Rules** : They specify “virtual relations” that are not actually stored but that can be formed from the facts by applying deduction mechanisms based on the rule specifications.

For example –

```
/*Facts*/
```

```
man(anand)
```

```
man(arun)
```

```
woman(anuradha)
```

```
woman(jayashree)
```

```
parent(anand, parth)
```

```
parent(anuradha,parth)
```

```
parent(arun,anuradha)
```

```
parent(jayashree,anuradha)
```

```
/* General Rule */
```

```
father(F,C):-man(F),parent(F,C)
```

```
mother(M,C):-woman(M),parent(M,C)
```

Now if we fire a query `?father(X,parth)`

Then the answer is `X=anand`. That means 'anand is father of parth'.

7.1.2 Main Memory Databases

- Main memory database system is a kind of database system in which data resides permanently on main physical memory.
- The backup copy of such database is maintained on the disk.
- Access to main memory is much faster than the disk access, hence transactions gets completed quickly. It is essential to have backup copy of the database because if the main memory gets failed then the entire database system gets lost.
- Some popularly used main memory database management systems are CSQL, TimesTen

7.1.3 Semantic Databases

- Semantic database management system is a system that stores the meaning of information as facts about objects.

- Semantic databases represent the data models in which data is arranged in some logical manner.
- It is a conceptual database system that include semantic information that adds basic meaning of the data and relationships among the data.
- The semantic database systems allow easy development of application programs and also it becomes easy to maintain such database system when data is updated.

Features

1. Semantic database systems are exceptionally usable and flexible.
2. They have shorter application design and programming cycle.
3. It provides user control via an intuitive structure of information.
4. It empowers the end-users to pose complex, ad-hoc, decision support queries.
5. These are highly efficient database systems.

Review Questions

1. *Explain active and deductive database systems.*
2. *Write short note on – semantic databases.*

7.2 Complex Data Types

7.2.1 Semi-Structured Data

- Semi structured data is a kind of data that can't be organized in relational databases. This is a kind of data which does not have structural framework but it might have some structural properties.
- Semi-structured data is basically a combination of structured and unstructured data. For example - Facebook that organizes information by User, Friends, Groups, Marketplace, etc., but the comments and text contained in these categories is unstructured.
- The semi-structured data does not reside in relational database. By applying some processes we can store them in relational database.
- XML(eXtensible Markup Language) is widely used to store and exchange semi-structured data.
- Examples of semi-structured data are -
 1. Emails
 2. Web pages

3. XML documents
4. Zipped files
5. Binary executables

Advantages

1. The semi-structured data is flexible as schema can be changed accordingly.
2. Data is portable.
3. The heterogeneous information can be stored in semi-structured data.
4. There is no need to express the requirements in pure SQL only.

Disadvantages

1. As there is no fixed schema, storing the data is complex.
2. Querying the database is less efficient as the data is in semi-structured manner.
3. Interpretation of relationship among data is difficult.

Difference between Structured and Unstructured Data

Sr. No.	Structured data	Semi-structured data	Unstructured data
1.	It is having fixed and organized form of data.	It is combination of structured and unstructured data.	It is not predefined or organized form of data.
2.	It is schema dependent and less flexible.	It is more flexible than structured data but less flexible than unstructured data.	It is the most flexible data.
3.	Structured query languages are used to access the data present in the schema	The tags and elements are used to access the data.	Only textual queries are possible.
4.	Storage requirement for data is less.	Storage requirements for the data is significant.	Storage requirements for the data is huge.
5.	Examples : Phone numbers, Customer Names, Social Security numbers.	Examples : Server logs, Tweets organized by hashtags, emails sorted by the inbox, sent or draft folders.	Examples : Emails and messages, Image files, Open ended survey answers.

7.2.2 Features of Semi-Structured Data Models

1. The semi-structured data can not be stored in the form of rows and columns in databases.
2. The semi structured data does not obey the tabular structure of data models. But it has some structure.
3. Semi-structured data contains tags and elements which is used to group data and describe how data is stored.
4. The entities can be grouped together based on their properties.
5. The entities in the same group may or may not have the same attributes(properties).
6. As the semi-structured data does not have well defined structure, it can not be programmed easily by the traditional programming languages.

Review Question

1. Enlist the features of semi-structured data model.

7.3 Nested Data Types

- Nested data types are structured data types for some common data patterns. Nested data types support arrays,structs and maps.
- For example – array is a homogeneous collection of elements of same data type elements. For instance : phone number can be stored as [98-230-11111]

```
struct employee
{
    Name string,
    id int,
    struct address
    {
        House-no int,
    Street string,
        City string
    }
}
```

7.3.1 JSON

- JSON stands for JavaScript object notation.
- Using JSON we can store and retrieve data. This text based open standard format.
- It is extended from JavaScript language.

Features of JSON

1. It is text based, lightweight data interchange format.
2. It is language independent.
3. It is easy to read and write.
4. It is easy for machines to parse and generate.
5. It uses the conventions that are familiar to the languages like C, C++, Java, JavaScript, Perl, Python and so on.

Structure of JSON

JSON is built on **two structures** :

1. A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
2. An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

JSON Object

- JSON object holds the Key value pair.
- Each key is represented as string and value can be of any datatype.
- The key and value are separated by colon.
- Syntax

```
{ string : value, .....}
```

- For example

```
"Age":38
```

- Each key value pair is separated by comma.
- The object is written within the { } curly brackets.

1) JSON object containing value of different data types

```
{  
  "student": {  
    "name":    AAA",    ← String value  
    "roll_no": 10,      ← Numeric value  
    "Indian":  true     ← Boolean value  
  }  
}
```


2) JSON Nested Object

```
{
  "student": {
    "name": "AAA",
    "roll_no": 10,
    "address": {
      "Street": "Shivaji Nagar",
      "City": "Pune",
      "Pincode": 411005
    }
  }
}
```

7.3.2 XML

- XML stands for eXtensible Markup Language.
- This scripting language is similar to HTML. That means, this scripting language contains various tags. But these tags are not predefined tags, in-fact user can define his own tags.
- Thus HTML is designed for **representation of data** on the web page whereas the XML is designed for **transport or to store data**.

Uses of XML

1. XML is used to display the meta contents i.e. XML describes the content of the document.
2. XML is useful in exchanging data between the applications.
3. The data can be extracted from database and can be used in more than one application. Different applications can perform different tasks on this data.

Advantages of XML

1. XML document is human readable and we can edit any XML document in simple text editors.
2. The XML document is language neutral. That means a Java program can generate an XML document and this document can be parsed by Perl.
3. Every XML document has a tree structure. Hence complex data can be arranged systematically and can be understood in simple manner.
4. XML files are independent of an operating system.

Goals of XML

Following are the goals of XML -

1. User must be able to define and use his own tag. This allows us to restrict the use of the set of tags defined by proprietary vendors.
2. Allow user to build his own tag library based on his web requirement.
3. Allow user to define the formatting rules for the user defined tags.
4. XML must support for storage or transport of data.

Features of XML

Following are some features which are most suitable for creating web related applications.

- XML is **EX**tesible **M**arkup Language intended for **transport** or storage of the data.
- The most important feature of XML is that user is able to **define and use his own** tag.
- XML contains only data and does not contain any formatting information. Hence document developers **can decide how to display the data**.
- XML permits the document writer to create the tags for any type of information. Due to this virtually any kind of information can be such as mathematical formulae, chemical structures, or some other kind of data can be described using XML.
- Searching sorting, rendering or manipulating the XML document is possible using Extended Stylesheet Language (XSL).
- The XML document can be validated using some external tools.
- Some commonly used web browsers like Internet Explorer and Firefox Mozilla provide support to the tags in XML. Hence XML is not at all vendor specific or browser specific.

7.3.3 Difference between XML and HTML

Sr. No.	HTML	XML
1.	HTML stands for Hypertext Markup Language.	XML stands for eXtensible Markup Language.
2.	HTML is designed to display data with the focus on look and feel of data.	XML is used to transport and store data, with focus on what data is.
3.	HTML is case insensitive.	XML is case sensitive.

4.	HTML has predefined tags.	XML has custom tags can be defined and the tags are invented by the author of the XML document.
5.	As HTML is for displaying the data it is static.	As XML is for carrying the data it is dynamic.
6.	It can not preserve white space.	It can preserve the white space.

7.3.4 Example of XML

```

<bank>
  <account>
    <account-number> S101 </account-number>
    <branch-name> Shivaji Nagar </branch-name>
    <balance> 5000 </balance>
  </account>
  <account>
    <account-number>C102 </account-number>
    <branch-name> Model Colony </branch-name>
    <balance> 4000 </balance>
  </account>
  <customer>
    <customer-name> Ram Kumar </customer-name>
    <customer-street> Fergusson Road</customer-street>
    <customer-city> Pune </customer-city>
  </customer>
  <customer>
    <customer-name> Shiv Prasad </customer-name>
    <customer-street> Main Road </customer-street>
    <customer-city> Nasik </customer-city>
  </customer>
  <depositor>
    <account-number> S101 </account-number>
    <customer-name> RamKumar </customer-name>
  </depositor>
  <depositor>
    <account-number> C102 </account-number>
    <customer-name> Shiv Prasad </customer-name>
  </depositor>
</bank>

```

Fig. 7.3.1 XML representation of bank information

7.3.5 Building Blocks of XML Document

Various building blocks of XML are -

1. Elements

The basic entity is **element**. The elements are used for defining the tags. The elements typically consist of opening and closing tag. Mostly only one element is used to define a single tag.

2. Attribute

The attributes are generally used to specify the values of the element. These are specified within the double quotes.

For example -

```
<flag type="True">
```

The type attribute of the element flag is having the value True.

3. CDATA

CDATA stands for Character Data. This character data will be parsed by the parser.

4. PCDATA

It stands for Parsed Character Data (i.e. text).

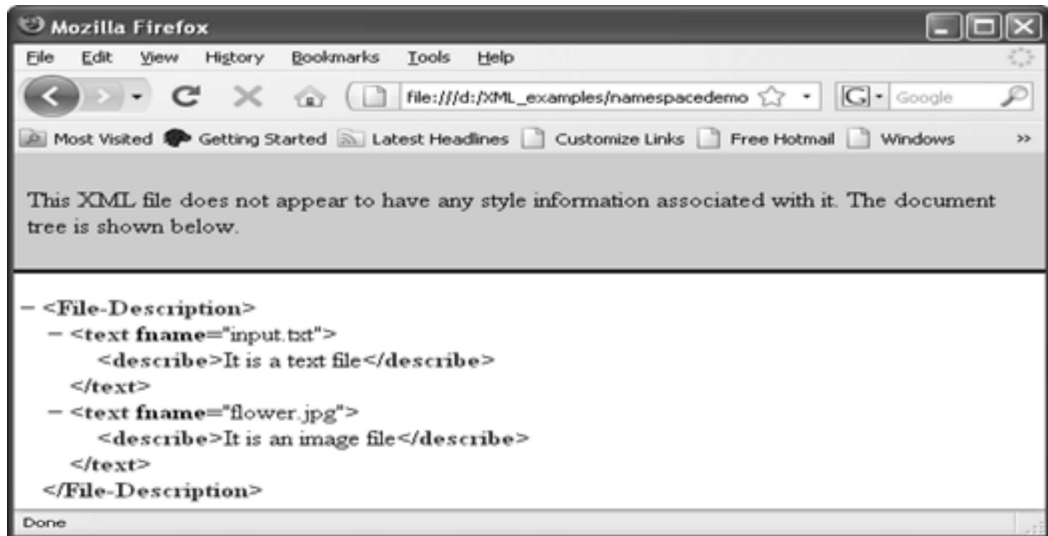
7.3.6 Concept of Namespace

- Sometimes we need to create two different elements by the same name. The xml document allows us to create **different elements** which are having the **common name**. This technique is known as **namespace**.
- In some web documents it becomes necessary to have the same name for two different elements. Here different elements mean the elements which are intended for different purposes. In such a case support for namespace technique is very much helpful.
- For example : Consider the following xml document -

namespacedemo.xml

```
<File-Description>
  <text fname="input.txt">
    <describe>It is a text file</describe>
  </text>
  <text fname="flower.jpg">
    <describe>It is an image file</describe>
  </text>
</File-Description>
```

The above document does not produce any error although the element **text** is used for two different attribute values. The output will be,



7.3.7 Document Type Definition (DTD)

- The document type definition is used to define the basic building block of any xml document.
- Using DTD we can specify the various elements types, attributes and their relationship with one another.
- Basically DTD is used to specify the set of rules for structuring data in any XML file.
- **For example :** If we want to put some information about some students in XML file then, generally we use tag **student** followed by his/her **name, address, standard and marks**. That means we are actually specifying the manner by which the information should be arranged in the XML file. And for this purpose the Document Type Definition is used.
- There are two ways by which DTD can be defined.

Internal DTD

Consider following xml document –

XML Document [DTDDemo1.xml]

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE student [
<!ELEMENT student (name,address,std,marks)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT std (#PCDATA)>
<!ELEMENT marks (#PCDATA)>
]>
<student>
<name>Anand</name>
  <address>Pune</address>
  <std>Second</std>
  <marks>70 percent</marks>
</student>

```

Here we are defining the DTD internally

Output

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```

- <student>
  <name>Anand</name>
  <address>Pune</address>
  <std>Second</std>
  <marks>70 percent</marks>
</student>

```

External DTD

In this type, an external DTD file is created and its name must be specified in the corresponding XML file. Following XML document illustrates the use of external DTD.

Step 1 : Creation of DTD file [student.dtd]

Open some suitable text editor or a notepad. Type following code into it -

```

<!ELEMENT student (name,address,std,marks)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT std (#PCDATA)>
<!ELEMENT marks (#PCDATA)>

```

Now save this file as *student.dtd*

Step 2 : Creation of XML document -

XML Document [DTDDemo.xml]

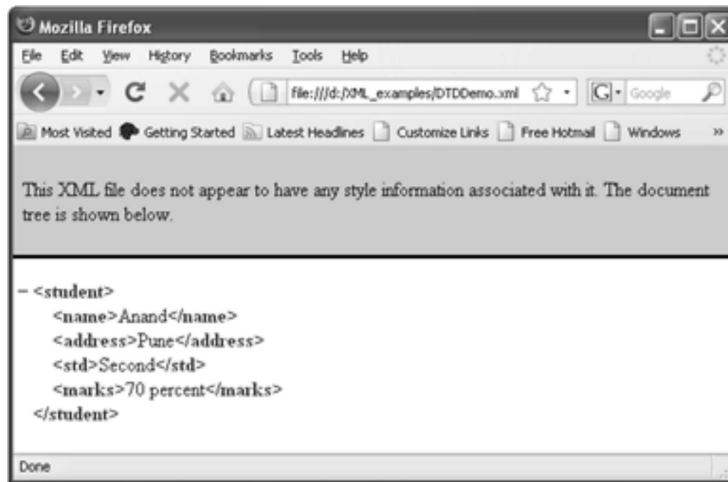
Now create a XML document as follows -

```
<?xml version="1.0"?>
<!DOCTYPE student SYSTEM "student.dtd">
<student>
  <name>Anand</name>
  <address>Pune</address>
  <std>Second</std>
  <marks>70 percent</marks>
</student>
```

The external DTD file is created

Step 3 : Using some web browser open the XML document.

Output



Merits of DTD

1. DTDs are used to define the structural components of XML document.
2. These are relatively simple and compact.
3. DTDs can be defined inline and hence can be embedded directly in the XML document.

Demerits of DTD

1. The DTDs are very basic and hence cannot be much specific for complex documents.
2. The language that DTD uses is not an XML document. Hence various frameworks used by XML cannot be supported by the DTDs.

3. The DTD cannot define the type of data contained within the XML document. Hence using DTD we cannot specify whether the element is numeric, or string or of date type.
4. There are some XML processor which do not understand DTDs.
5. The DTDs are not aware of namespace concept.

Review Questions

1. What is JSON ? Explain it with suitable example.
2. What is XML? Enlist its features.

7.4 Object Orientation

- An object-oriented database (OODBMS) or object database management system (ODBMS) is a database that is based on object-oriented programming (OOP). The data is represented and stored in the form of objects.
- OODBMS are also called object databases or object-oriented database management systems.

Challenges in Designing OODBMS

1. There is **no universally agreed data** model for OODBMS. Hence no universal standard is followed while designing the OODBMS based applications.
2. The OODBMS is **complex** due to handling of complex datatypes, information hiding, inheritance and other object oriented features.
3. The OODBMS **does not provide view mechanism**. Therefore the developer can not have limited view of the physical database system.
4. There is **lack of security** mechanism in maintaining OODBMS.
5. The OODBMS has **scalability and resource usage limitation**, including database server.
6. As **amount of data** generated and collected explodes, OODBMS is struggling to keep up.
7. Although there are benefits to **decentralize data management** and communication among them using data objects, it presents challenges as well. How will the data be distributed? What are the best methods of decentralization? There is a major challenge in designing and managing the database of such systems.
8. In order to handle OODBMS, it **requires skilled people** who know object oriented technology thoroughly well.

Difference between OODBMS and DBMS

OODBMS	DBMS
OODBMS stands for Object Oriented Database Management System that supports creating and modelling of data as objects.	DBMS is any Database Management System. The most popular DBMS are relational database management systems in which we store everything as a relation between entities.
Handles larger and complex data than RDBMS.	Handles comparatively simpler data.
Data Handling - Stores data as well as methods to use it.	Data Handling - RDBMS stores only data.
OODBMS is object-oriented.	RDBMS is table-oriented.
OODBMS uses inheritance and encapsulation to reduce data redundancy.	Normalization is used to eliminate data redundancy in RDBMS.
Examples - Object Database, Objectivity/DB, ObjectStore, Cache, and, ZODB	Examples - MSSQL, MySQL, and Oracle

7.4.1 Object-Relational Database System

- Object relational database system is a database system which is based on object relation model. It provides the migration path for the users of relational databases who wish to use object oriented features.
- There are two commonly addressed issues for supporting object relational database systems -
 1. Build object oriented database systems(OODBMS) that natively support object oriented type system and allows direct access to data from object oriented programming language using the native type system of the language.
 2. Automatically convert data from the native type system of programming language to relational representation and vice versa.This data conversion is specified as **object relational mapping**.

7.4.2 Table Inheritance

- Table inheritance is a feature where there are sub-tables in Relational database management systems that inherit the schema of parent tables.

- For example – Following statements demonstrate how to achieve table inheritance

```
create table people of Person; ← Parent table  
create table students of Student ← Child table  
    under people;  
create table teachers of Teacher ← Child table  
    under people;
```

- As a result of above statements, every attribute present in the **people** table is also present in the subtables **students** and **teachers**. Similarly every tuple present in the **students** or **teachers** become implicitly present in **people**.
- Using the keyword **only** in the query we can restrict the access to particular table. For example – if we want to find the tuples that are present in the **people** table but not in its subtables then we should use “**only people**”

7.4.3 Object-Relational Mapping

- Object oriented relational mapping is an approach in which object oriented programming language is integrated with databases.
- The object-relational mapping allows a programmer to define **mapping** between **tuples** in **database** relations and **objects** in the programming language.
- An object can be retrieved based on the selection condition on its attributes.
- The object oriented programs can be used to create an object, update the objects and delete the object. Similarly mapping from object to relations is possible by updating, deleting, and inserting tuples in the database.
- Hibernate is an open source **object relational mapping tool** for the Java platform.

Advantages

1. It supports object oriented features such as inheritance, polymorphism, association and so on.
2. Instead of plain data, the object relational mapping return objects. These objects can be easily accessed, programmed and permanently stored in memory.
3. Object relational mapping systems also provide query languages that allow programmer to write queries directly on the object model. Such queries are translated into SQL queries for underlying relational database.

Disadvantages

1. The query language for object relational mapping systems have limited capabilities.
2. There is significant overhead on the object relation system when there occurs bulk database update.

Review Questions

1. Explain the concept of table inheritance with suitable example.
2. Explain object relational mapping. Give its advantages and disadvantages.

7.5 Spatial Data

- Spatial data means data **related to space**.
- The special data in the database system allows it to store, index and query the data based on the basis of **special locations**.
- Spatial data represents location, size and shape of an object on the earth. For example - City, location of school, or location of hospital is represented with the help of spatial data.

7.5.1 Geometric Data

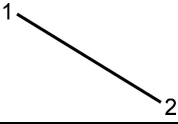
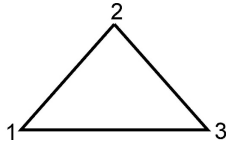
- Various geometric data constructs can be represented in a database in normalized fashion.
- Following is a list of various geometric constructions and the description on how to store the information of these geometric constructs in the database systems –

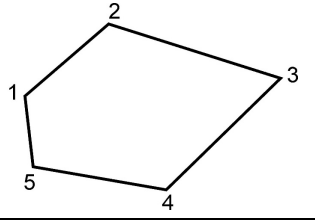
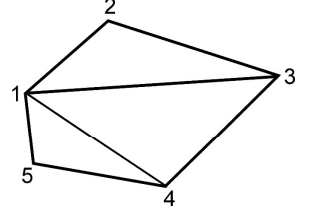
Line : The line segment is represented by co-ordinates of its endpoints.

Curve : Approximate a curve by partitioning it into a storage. Create a list of vertices in order. Represent each segment as a separate tuple that also carries with it the identifier of the curve.

Closed polygons :

- i) It is represented as list of vertices in order starting vertex is the same as ending vertex.
- ii) Represent boundary edges as separate tuples with identifier.
- iii) Use **triangulation** – Divide polygon into triangles. Use identifier for each triangles.

Line segment		$\{(a_1, b_1), (a_2, b_2)\}$
Triangle		$\{(a_1, b_1), (a_2, b_2), (a_3, b_3)\}$

Polygon		$\{(a_1, b_1), (a_2, b_2), (a_3, b_3), (a_4, b_4), (a_5, b_5)\}$
Polygon		$\{(a_1, b_1), (a_2, b_2), (a_3, b_3), ID_1\}$ $\{(a_1, b_1), (a_3, b_3), (a_4, b_4), ID_2\}$ $\{(a_1, b_1), (a_4, b_4), (a_5, b_5), ID_3\}$

- The 3D shapes are represented in the similar manner as of 2D shapes. But it has extra z component.

7.5.2 Geographic Data

- Geographic data are spatial in nature. For example – maps and satellite images are geometric data.
- Maps not only provides the location but along with location it also provides detailed information about the location.

Applications of geographic data

- 1) Web based road map services which allows us to use map data for vehicle navigation.
- 2) Vehicle navigation systems store information about roads and services for use of drivers.
- 3) Global Positioning System (GPS) information broadcasts from GPS satellites to find the current location of user with some accuracy.

Representation of geographic data

- There are two types of geographic data
1. **Raster data** : Raster data consists of bit maps or pixel maps in two or more dimensions. For example – 2D raster image : Satellite image of cloud cover, where each pixel stores the cloud visibility in a particular area.
 2. **Vector data** : The vector data is constructed from basic geometric objects such as points, line segments, polygon, triangles and so on. The vector data is always used to represent the map data. For instance – roads can be considered as two dimensional(2D) and represented by lines and curves. Rivers may be represented as complex curves. The regions and lakes can be represented using polygons.

Review Question

1. Explain geographic data in detail.

Unit - VI

Multiple Choice Questions

Q.1 XML stands for _____.

- | | |
|---|---|
| <input type="checkbox"/> a Extra Markup Language | <input type="checkbox"/> b Excellent Markup Links |
| <input type="checkbox"/> c Extended Markup Language | <input type="checkbox"/> d Extended Marking Links |

Q.2 Which statement is true ?

- a All the statements are true.
- b All XML elements must have a closing tag.
- c All XML elements must be lower case.
- d All XML documents must have a DTD.

Q.3 XML is designed to _____ and store data.

- | | |
|--------------------------------------|--|
| <input type="checkbox"/> a design | <input type="checkbox"/> b verify |
| <input type="checkbox"/> c transport | <input type="checkbox"/> d none of these |

Q.4 What does DTD stand for ?

- | | |
|--|---|
| <input type="checkbox"/> a Direct Type Definition | <input type="checkbox"/> b Document Type Definition |
| <input type="checkbox"/> c Dynamic Type Definition | <input type="checkbox"/> d Dynamic Tag Definition |

Q.5 What is the full form of JSON ?

- | | |
|--|---|
| <input type="checkbox"/> a JavaScripts Object Notification | <input type="checkbox"/> b JavaScript Object Notation |
| <input type="checkbox"/> c Java Object Notation | <input type="checkbox"/> d None of these |

Q.6 Which is the file extension of JSON ?

- | | |
|---------------------------------|----------------------------------|
| <input type="checkbox"/> a .jsn | <input type="checkbox"/> b .json |
| <input type="checkbox"/> c .jn | <input type="checkbox"/> d .js |

Q.7 What is JSON data ?

- | | |
|--|--|
| <input type="checkbox"/> a Exchanging data | <input type="checkbox"/> b Non-exchanging data |
| <input type="checkbox"/> c Changing data | <input type="checkbox"/> d None of these |

Q.8 What are two main structures compose JSON ?

- | | |
|--|--|
| <input type="checkbox"/> a Arrays and structures | <input type="checkbox"/> b Keys and values |
| <input type="checkbox"/> c Classes and objects | <input type="checkbox"/> d Pointers and References |

Solved Model Question Paper (In Sem)

Database Management Systems

T.E. (Computer) Semester - V (As Per 2019 Pattern)

Time : 1 Hour]

[Maximum Marks : 30

N.B : i. Attempt Q.1 or Q.2, Q.3 or Q.4.

ii. Neat diagrams must be drawn wherever necessary.

iii. Figures to the right side indicate full marks.

iv. Assume suitable data, if necessary.

- Q.1** a) What is DBMS ? Enlist the characteristics of DBMS. (Refer section 1.1) [4]
b) Explain advantages of DBMS over file system. (Refer section 1.3) [3]
c) Draw and explain overall structure of database system. (Refer section 1.8) [8]

OR

- Q.2** a) Explain mapping cardinality representation in ER diagram with the help of illustrative examples. (Refer section 1.17.1) [5]
b) Construct an E R diagram for library management system. (Refer example 1.21.9) [10]

- Q.3** a) i) Explain aggregate functions used in SQL with suitable examples. (Refer section 2.17) [6]

ii) Consider, the following database,

Student(RollNo, Name, Address)

Subject(Sub_code, Sub_Name)

Marks (Roll_no, Sub_code, Marks)

Write following queries in SQL.

Find average marks of each student, along with the name of Student.

(Refer example 2.17.1) [3]

- b) Write SQL statements for the following (any five)

Consider the following database

pilot (pid, pname)

flight (fid, ftype, capacity)

route (pid, fid, from_city, to_city)

- i) List the details of flights having capacity more than 300.
 - ii) List the flights between 'Surat' and 'Mumbai'.
 - iii) List the names of the pilots who fly from 'Pune'.
 - iv) List the route on which, pilot named 'Mr Kapoor' flies
 - v) List the pilots whose names, starts with letter 'A' %' but does not end with letter 'A'.
 - vi) List the name of pilots who fly 'boing 737' type of flights.
- (Refer example 2.22.2)** [6]

OR

- Q.4 a)** What are different set Operations used in SQL ? Explain. **(Refer section 2.18)** [5]
- b)** Write the PL/SQL block of code to calculate the factorial value of a number. **(Refer example 3.9.3)** [4]
- c)** Write PL/SQL trigger for following requirement :
- Event : Deletion of row from stud (roll_no, name, class) table.
- Action : After deletion of values from stud table, values should be inserted into cancel_admission (roll_no, name) table
- Note : For every row to be deleted, action should be performed.
- (Refer example 3.13.2)** [6]

Solved Model Question Paper (End Sem)
Database Management Systems
T.E. (Computer) Semester - V (As Per 2019 Pattern)

Time : $2\frac{1}{2}$ Hours]

[Maximum Marks : 70

- N.B :**
- i. Attempt Q.1 or Q.2, Q.3 or Q.4, Q.5 or Q.6, Q.7 or Q.8.
 - ii. Neat diagrams must be drawn wherever necessary.
 - iii. Figures to the right side indicate full marks.
 - iv. Assume suitable data, if necessary.

- Q.1 a)** Explain in brief - Codd's rules. **(Refer section 4.3)** [10]
- b)** Explain the concept of referential integrity constraint with example. **(Refer section 4.5)** [8]

OR

- Q.2 a)** Give $R = \{A, B, C, G, H, I\}$. The following set F of functional dependencies holds
 $A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H$
Computer AG^+ . Is AG candidate key? **(Refer example 4.11.7)** [6]
- b)** What is the difference between 3NF and BCNF? **(Refer example 4.18.8)** [6]
- c)** $Students_Detail (Stud_id, Stud_name, Zip, City)$
Consider above schema, check whether it is in 3NF, if not justify and propose the schema in 3NF. **(Refer example 4.17.7)** [6]
- Q.3 a)** State and explain ACID properties. **(Refer section 5.3)** [5]
- b)** Consider the three transactions T_1, T_2 , and T_3 and schedules S_1 and S_2 given below. Determine whether each schedule is serializable or not? If a schedule is serializable write down the equivalent serial schedule(S).
 $T_1: R_1(x) R_1(z); W_1(x);$
 $T_2: R_2(x); R_2(y); W_2(z); W_2(y)$
 $T_3: R_3(x); R_3(y); W_3(y);$
 $S_1: R_1(x); R_2(z); R_1(z); R_3(x); R_3(y); W_1(x); W_3(y); R_2(y); W_2(z); W_2(y);$
 $S_2: R_1(x); R_2(z); R_3(x); R_1(z); R_2(y); R_3(y); W_1(x); W_2(z); W_3(y); W_2(y);$
(Refer example 5.5.4) [7]
- c)** Explain the concept of cascadeless schedule. **(Refer section 5.6)** [5]

OR

- Q.4 a)** Explain the concept of concurrency control. **(Refer section 5.7)** [2]
- b)** What is two phase locking protocol? Explain it in detail. **(Refer section 5.9.3)** [8]
- c)** Write short note on - Shadow paging. **(Refer section 5.14)** [7]
- Q.5 a)** Explain in brief - CAP theorem. **(Refer section 6.2)** [4]
- b)** What are different types of NoSQL database? Explain in detail. **(Refer section 6.5)** [8]
- c)** What is distributed system? Give its advantages and disadvantages.
(Refer section 6.1) [6]

OR

- Q.6 a)** Explain the CRUD operations used in MongoDB. **(Refer section 6.9.3)** [8]
- b)** Explain the concept of aggregation in MongoDB. **(Refer section 6.9.5)** [6]
- c)** Give the difference between ACID and BASE properties. **(Refer section 6.7)** [4]

- Q.7** a) *What is active and deductive databases. (Refer section 7.1.1)* [6]
b) *Explain the structure of JSON with example. (Refer section 7.3.1)* [6]
c) *Write short note on - Semi-structured data. (Refer section 7.2.1)* [5]

OR

- Q.8** a) *Write XML representation of bank information. (Refer section 7.3.4)* [10]
b) *Write short note on - spatial data. (Refer section 7.5)* [7]





Made in India