

Database Management Systems

- SIMPLIFIED APPROACH
- 2 MARKS QUESTIONS WITH ANSWERS
- CHAPTERWISE SOLVED AU QUESTIONS DEC 2002 to MAY 2018



Sub Code : CS8492



**TECHNICAL
PUBLICATIONS**TM

An Up-Thrust for Knowledge

A. A. Puntambekar

Strictly as per Revised Syllabus of
ANNA UNIVERSITY
Choice Based Credit System (CBCS)
Semester - IV (CSE / IT)

DATABASE MANAGEMENT SYSTEMS

Mrs. Anuradha A. Puntambekar
M.E. (Computer)
Formerly Assistant Professor in
P.E.S. Modern College of Engineering,
Pune



DATABASE MANAGEMENT SYSTEMS

Semester - IV (CSE / IT)

First Edition : January 2019

© Copyright with Author

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 :

 **TECHNICAL PUBLICATIONS**™ Amit Residency, Office No.1, 412, Shaniwar Peth, Pune - 411030, M.S. INDIA
Ph.: +91-020-24495496/97, Telefax : +91-020-24495497
Email : sales@technicalpublications.org Website : www.technicalpublications.org

ISBN 978-93-332-2129-0



PREFACE

The importance of **Database Management Systems** is well known in various engineering fields. Overwhelming response to my books on various subjects inspired me 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 chapters in this 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 this 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.

I wish to express my 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 my whole family. I 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.

Author

A. A. Puntambekar

Dedicated to God.

SYLLABUS

Database Management Systems - (CS8492)

UNIT I Relational Databases

Purpose of Database System - Views of data - Data Models- Database System Architecture - Introduction to relational databases - Relational Model - Keys - Relational Algebra - SQL fundamentals - Advanced SQL features - Embedded SQL - Dynamic SQL. **(Chapter - 1)**

UNIT II Database Design

Entity-Relationship model - E-R Diagrams - Enhanced-ER Model - ER-to-Relational Mapping - Functional Dependencies - Non-loss Decomposition - First, Second, Third Normal Forms, Dependency Preservation - Boyce/Codd Normal Form - Multi-valued Dependencies and Fourth Normal Form - Join Dependencies and Fifth Normal Form. **(Chapter - 2)**

UNIT III Transactions

Transaction Concepts - ACID Properties - Schedules - Serializability - Concurrency Control - Need for Concurrency - Locking Protocols - Two Phase Locking - Deadlock - Transaction Recovery - Save Points - Isolation Levels - SQL Facilities for Concurrency and Recovery. **(Chapter - 3)**

UNIT IV Implementation Techniques

RAID - File Organization - Organization of Records in Files - Indexing and Hashing - Ordered Indices - B+ tree Index Files - B tree Index Files - Static Hashing - Dynamic Hashing - Query Processing Overview - Algorithms for SELECT and JOIN operations - Query optimization using Heuristics and Cost Estimation. **(Chapter - 4)**

UNIT V Advanced Topics

Distributed Databases : Architecture, Data Storage, Transaction Processing - Object-based Databases : Object Database Concepts, Object-Relational features, ODMG Object Model, ODL, OQL - XML Databases : XML Hierarchical Model, DTD, XML Schema, XQuery - Information Retrieval : IR Concepts, Retrieval Models, Queries in IR systems. **(Chapter - 5)**

TABLE OF CONTENTS

Unit - I

Chapter - 1 Relational Databases	(1 - 1) to (1 - 72)
1.1 Introduction to Database Management	1 - 2
1.2 Purpose of Database System	1 - 2
1.3 Views of Data	1 - 5
1.3.1 Data Abstraction	1 - 6
1.3.2 Instances and Schema	1 - 7
1.3.3 Database Languages	1 - 8
1.4 Data Models.....	1 - 9
1.5 Database System Architecture	1 - 12
1.6 Data Independence	1 - 15
1.7 Introduction to Relational Databases	1 - 16
1.8 Relational Model.....	1 - 17
1.9 Keys.....	1 - 19
1.10 Integrity Constraints	1 - 22
1.10.1 Entity Integrity Rule.	1 - 22
1.10.2 Referential Integrity Rule	1 - 23
1.11 Database integrity.....	1 - 24
1.12 Relational Algebra.....	1 - 25
1.12.1 Relational Operations.	1 - 26
1.13 SQL Fundamentals	1 - 36
1.13.1 Data Abstraction	1 - 36
1.13.2 Basic Schema Definition.	1 - 37
1.13.3 Basic Structure of SQL Queries	1 - 39
1.13.3.1 Queries on Multiple Relations	1 - 39
1.13.4 Additional Basic Operations.	1 - 43

1.13.5 Domain and Key Constraint	1 - 46
1.13.6 String Operations	1 - 48
1.13.7 Set Operations	1 - 50
1.13.8 Aggregate Functions.	1 - 52
1.13.8.1 Basic Aggregation	1 - 52
1.13.8.2 Use of Group By and Having Clause	1 - 54
1.13.9 Nested Queries	1 - 56
1.13.10 Modification of Databases	1 - 57
1.14 Advanced SQL Features	1 - 63
1.14.1 Embedded SQL	1 - 63
1.15 Dynamic SQL.....	1 - 65
1.16 Two Marks Questions with Answers.....	1 - 66

Unit - II

Chapter - 2 Database Design (2 - 1) to (2 - 60)

2.1 Introduction to Entity Relationship Model	2 - 2
2.1.1 Design Phases	2 - 2
2.1.2 ER Model	2 - 3
2.2 Mapping Cardinality.....	2 - 6
2.3 ER Diagrams	2 - 7
2.3.1 Mapping Cardinality Representation using ER Diagram	2 - 9
2.3.2 Ternary Relationship.	2 - 10
2.3.3 Binary and Ternary Relationships	2 - 10
2.3.4 Weak Entity Set.	2 - 12
2.4 Enhanced ER Model	2 - 13
2.4.1 Specialization and Generalization	2 - 13
2.4.2 Constraints on Specialization/Generalization	2 - 14
2.4.3 Aggregation	2 - 16
2.5 Examples based on ER Diagram	2 - 16
2.6 ER to Relational Mapping	2 - 23

2.6.1 Mapping of Entity Set to Relationship	2 - 23
2.6.2 Mapping Relationship Sets(without Constraints) to Tables	2 - 23
2.6.3 Mapping Relationship Sets(With Constraints) to Tables	2 - 24
2.6.4 Mapping Weak Entity Sets to Relational Mapping.....	2 - 26
2.6.5 Mapping of Specialization / Generalization (EER Construct) to Relational Mapping.....	2 - 27
2.7 Concept of Relational Database Design.....	2 - 28
2.8 Functional Dependencies	2 - 28
2.8.1 Computing Closure Set of Functional Dependency	2 - 29
2.8.2 Canonical Cover or Minimal Cover	2 - 32
2.9 Concept of Redundancy and Anomalies	2 - 34
2.10 Decomposition.....	2 - 35
2.10.1 Non-loss Decomposition or Loss-less Join	2 - 37
2.10.2 Dependency Preservation	2 - 39
2.11 Normal Forms	2 - 41
2.11.1 First Normal Form.....	2 - 41
2.11.2 Second Normal Form	2 - 42
2.11.3 Third Normal Form	2 - 44
2.12 Boyce / Codd Normal Form (BCNF)	2 - 47
2.13 Multivalued Dependencies and Fourth Normal Form	2 - 51
2.14 Join Dependencies and Fifth Normal Form.....	2 - 53
2.15 Two Marks Questions with Answers.....	2 - 56

Unit - III

Chapter - 3 Transactions	(3 - 1) to (3 - 58)
3.1 Transaction Concepts	3 - 2
3.2 ACID Properties	3 - 2
3.3 Transaction States	3 - 4
3.4 Schedules	3 - 5

3.5 Serializability.....	3 - 6
3.5.1 Conflict Serializability	3 - 8
3.5.2 View Serializability.....	3 - 11
3.6 Transaction Isolation and Atomicity.....	3 - 18
3.6.1 Recoverable Schedule	3 - 18
3.6.2 Cascadeless Schedule	3 - 20
3.7 Introduction to Concurrency Control	3 - 20
3.8 Need for Concurrency.....	3 - 21
3.9 Locking Protocols.....	3 - 24
3.9.1 Why Do we Need Locks ?	3 - 24
3.9.2 Simple Lock Based Protocol	3 - 25
3.10 Two Phase Locking.....	3 - 26
3.10.1 Types of Two Phase Locking	3 - 30
310.2 Lock Conversion	3 - 32
3.11 Time Stamp Based Protocol.....	3 - 34
3.12 Dead Lock	3 - 36
3.13 Transaction Recovery	3 - 42
3.13.1 Failure Classification	3 - 42
3.13.2 Storage	3 - 42
3.13.3 Recovery with Concurrent Transactions	3 - 43
3.13.4 Shadow Copy Technique.....	3 - 44
3.13.5 Log Based Recovery Approach	3 - 44
3.14 Save Points.....	3 - 49
3.15 Isolation Levels	3 - 51
3.16 SQL Facilities for Concurrency and Recovery	3 - 52
3.17 Two Marks Questions with Answers	3 - 54

Unit - IV

Chapter - 4 Implementation Techniques (4 - 1) to (4 - 58)

4.1 RAID	4 - 2
4.1.1 RAID Levels	4 - 2
4.2 File Organization.....	4 - 7
4.3 Organization of Records in File	4 - 11
4.3.1 Sequential File Organization	4 - 12
4.3.2 Multi-table Clustering File Organization	4 - 13
4.4 Indexing and Hashing.....	4 - 14
4.5 Ordered Indices	4 - 15
4.5.1 Primary and Clustered Indices	4 - 15
4.5.2 Dense and Sparse Indices	4 - 16
4.5.3 Single and Multilevel Indices	4 - 17
4.5.4 Secondary Indices	4 - 19
4.6 B+ Tree Index Files.....	4 - 20
4.6.1 Insertion Operation	4 - 21
4.6.2 Deletion Operation	4 - 26
4.7 B Tree Index Files.....	4 - 30
4.8 Hashing	4 - 33
4.8.1 Basic Terms used in Hashing.	4 - 33
4.9 Static Hashing	4 - 35
4.9.1 Open Hashing.	4 - 36
4.10 Dynamic Hashing	4 - 37
4.10.1 Expendible Hashing	4 - 37
4.11 Query Processing Overview.....	4 - 48
4.12 Measure of Query Cost.....	4 - 50
4.13 Algorithms for SELECT Operation	4 - 51
4.14 Algorithms for JOIN Operation	4 - 52

4.15 Query Optimization using Heuristics and Cost Estimation	4 - 54
4.15.1 Heuristic Estimation	4 - 54
4.15.2 Cost based Estimation.....	4 - 55
4.16 Two Marks Questions with Answers	4 - 56

Unit - V

Chapter - 5 Advance Topics	(5 - 1) to (5 - 44)
5.1 Architecture of Distributed Databases	5 - 2
5.1.1 Homogeneous and Heterogeneous Databases.....	5 - 3
5.2 Data Storage	5 - 4
5.2.1 Data Replication.....	5 - 5
5.2.2 Data Fragmentation.....	5 - 5
5.3 Transaction Processing	5 - 7
5.3.1 Two Phase Commit Protocol.....	5 - 8
5.3.2 Three Phase Commit Protocol	5 - 12
5.4 Object Database Concepts.....	5 - 13
5.4.1 Complex Data Types	5 - 13
5.4.2 Object Oriented Data Model	5 - 13
5.4.2.1 Object Structure	5 - 14
5.4.2.2 Object Classes	5 - 15
5.4.2.3 Inheritance	5 - 15
5.4.2.4 Multiple Inheritance	5 - 16
5.4.2.5 Object Identity	5 - 17
5.4.2.6 Object Containment.	5 - 18
5.4.3 Object Oriented Languages.....	5 - 19
5.4.4 Persistent Programming Languages.....	5 - 19
5.4.4.1 Persistence of Object	5 - 20
5.4.4.2 Object Identity and Pointers	5 - 20
5.4.4.3 Storage and Access of Persistent Objects	5 - 20
5.5 Object Relational Features	5 - 21

5.6 ODMG Object Model	5 - 22
5.6.1 ODL	5 - 23
5.6.2 OQL	5 - 24
5.7 XML Hierarchical Model	5 - 25
5.7.1 Types of XML Documents	5 - 26
5.8 DTD	5 - 27
5.9 XML Schema	5 - 30
5.10 XQuery	5 - 32
5.11 IR Concepts	5 - 34
5.11.1 Modes of Interaction	5 - 36
5.11.2 IR Processing	5 - 37
5.12 Retrieval Models	5 - 38
5.13 Queries in IR Systems	5 - 41
5.14 Two Marks Questions with Answers	5 - 42

UNIT - I

1

Relational Databases

Syllabus

Purpose of Database System - Views of data - Data Models- Database System Architecture - Introduction to relational databases - Relational Model - Keys - Relational Algebra - SQL fundamentals - Advanced SQL features - Embedded SQL - Dynamic SQL.

Contents

1.1	<i>Introduction to Database Management</i>
1.2	<i>Purpose of Database System</i> May-07, 12, Dec.-04, Marks 8
1.3	<i>Views of Data.....</i> May-16 Marks 16
1.4	<i>Data Models.....</i> Dec.-14, Marks 8
1.5	<i>Database System Architecture</i> May-12, 13, 14, 16, 17, <i>.....</i> Dec.-08, 15, 17, Marks 16
1.6	<i>Data Independence</i>
1.7	<i>Introduction to Relational Databases</i>
1.8	<i>Relational Model</i>
1.9	<i>Keys.....</i> May-06, 07, 12, Dec.-06, Marks 4
1.10	<i>Integrity Constraints</i> Dec.-05, Marks 10
1.11	<i>Database integrity</i>
1.12	<i>Relational Algebra.....</i> May-03,04,05,14,15,16,17,18, <i>.....</i> Dec.-02,07,08,11,15,16,17 . Marks 16
1.13	<i>SQL Fundamentals.....</i> Dec.-15, Marks 16
1.14	<i>Advanced SQL Features</i>
1.15	<i>Dynamic SQL.....</i> May-17, Dec.-17, Marks 11
1.16	<i>Two Marks Questions with Answers</i>

Part I : Introduction of DBMS**1.1 Introduction to Database Management**

- **Definition :** A Database Management System (DBMS) is a collection of **interrelated data** and various **programs** that are used to handle that 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 -
 - (i) **Define the structure** for storage of information.
 - (ii) Provide mechanism for **manipulation of information**.
- In addition, the database systems must ensure the **safety of information** stored.

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.

1.2 Purpose of Database System**AU : May-07, 12, Dec.-04**

- 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.
- 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 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 correct 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.

Characteristics of Database Systems

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.

Advantages of Database Systems

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.

Disadvantages of Database Systems

- 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

University Questions

1. Compare file system with database system	AU : May-07, Marks 8, May-12, Marks 2
2. What are the advantages and disadvantages of DBMS ?	AU : Dec.-04, Marks 4

1.3 Views of Data

AU : May-16

- 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 **helps 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.

1.3.1 Data Abstraction

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

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

i) Physical level :

- o This is the lowest level.
- o This level describes how actually the data are stored.
- o This level describes complex low level data structures.

2) Logical level :

- o This is the next higher level, which describes the what data are stored in database.
- o This level also describes the relationship among the data.
- o The logical level thus describes then entire database in terms of small number of relatively simple structures.

The database administrators use logical level of abstraction for deciding what information to keep in database.

3) View level :

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

Fig. 1.3.1 shows the relationship among the three levels of abstraction.

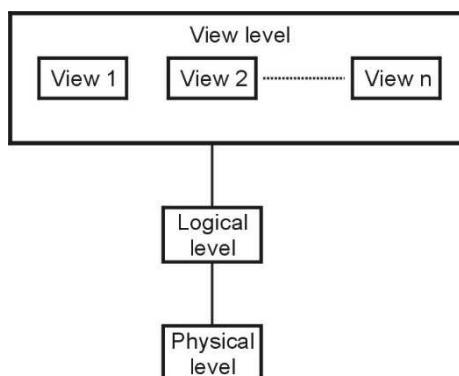


Fig. 1.3.1 : Levels of data abstraction

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

- o 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.
- o 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.
- o 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.

1.3.2 Instances and Schemas

Schema : The overall design of the database is called schema

For example - In a program we do variable declaration and assignment of values to the variable. The variable declaration is called schema and the value assigned to the variable is called instance. The schema for the student record can be

RollNo	Name	Marks
--------	------	-------

Instances : When information is inserted or deleted from the database then the database gets changed. The collection of information at particular moment is called **instances**. For example - following is an instance of **student** database

RollNo	Name	Marks
10	AAA	43
20	BBB	67

Types of Schema : The database has several schema based on the levels of abstraction.

- (1) **Physical Schema :** The physical schema is a database design described at the physical level of abstraction.
- (2) **Logical Schema :** The logical schema is a database design at the logical level of abstraction.
- (3) **Subschema :** A database may have several views at the view level which are called subschemas.

1.3.3 Database Languages

There are two types of languages supported by database systems. These are -

(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 which is used for **creating** and modifying the structures of tables, views, indexes and so on.
- DDL is also used to specify additional properties of data.
- Some of the common commands used in DDL are -**CREATE, ALTER, DROP**.
- The main 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**.

University Question

1. Briefly explain about views of data.

AU : May-16, Marks 16

1.4 Data Models

AU : Dec.-14

- **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 :

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

SeatNo	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.

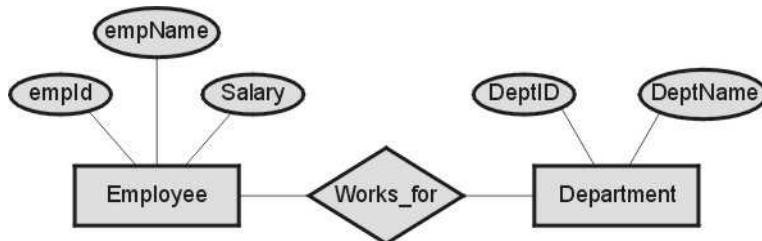
(iv) Easy design,maintenance and usage : The relational models can be designed logically hence they are easy to maintain and use.

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 :

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



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.

University Question

1. Write short note on : Data model and its types.

AU : Dec.-14, Marks 8

1.5 Database System Architecture

AU : May-12, 13, 14, 16, 17, Dec.-08, 15, 17

- The typical structure of typical DBMS is based on relational data model as shown in Fig. 1.5.1. (Refer page 1-14).
- Consider the top part of Fig. 1.5.1. 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 -
 - Authorization and integrity manager :** Validates the users who want to access the data and tests for integrity constraints.
 - Transaction manager :** Ensures that the database remains in consistent despite of system failures and concurrent transaction execution proceeds without conflicting.
 - File manager :** Manages allocation of space on disk storage and representation of the information on disk.
 - 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.
- Storage manager implements several data structures such as -
 - Data files :** Used for storing database itself.
 - Data dictionary :** Used for storing metadata, particularly schema of database.
 - Indices :** Indices are used to provide fast access to data items present in the database

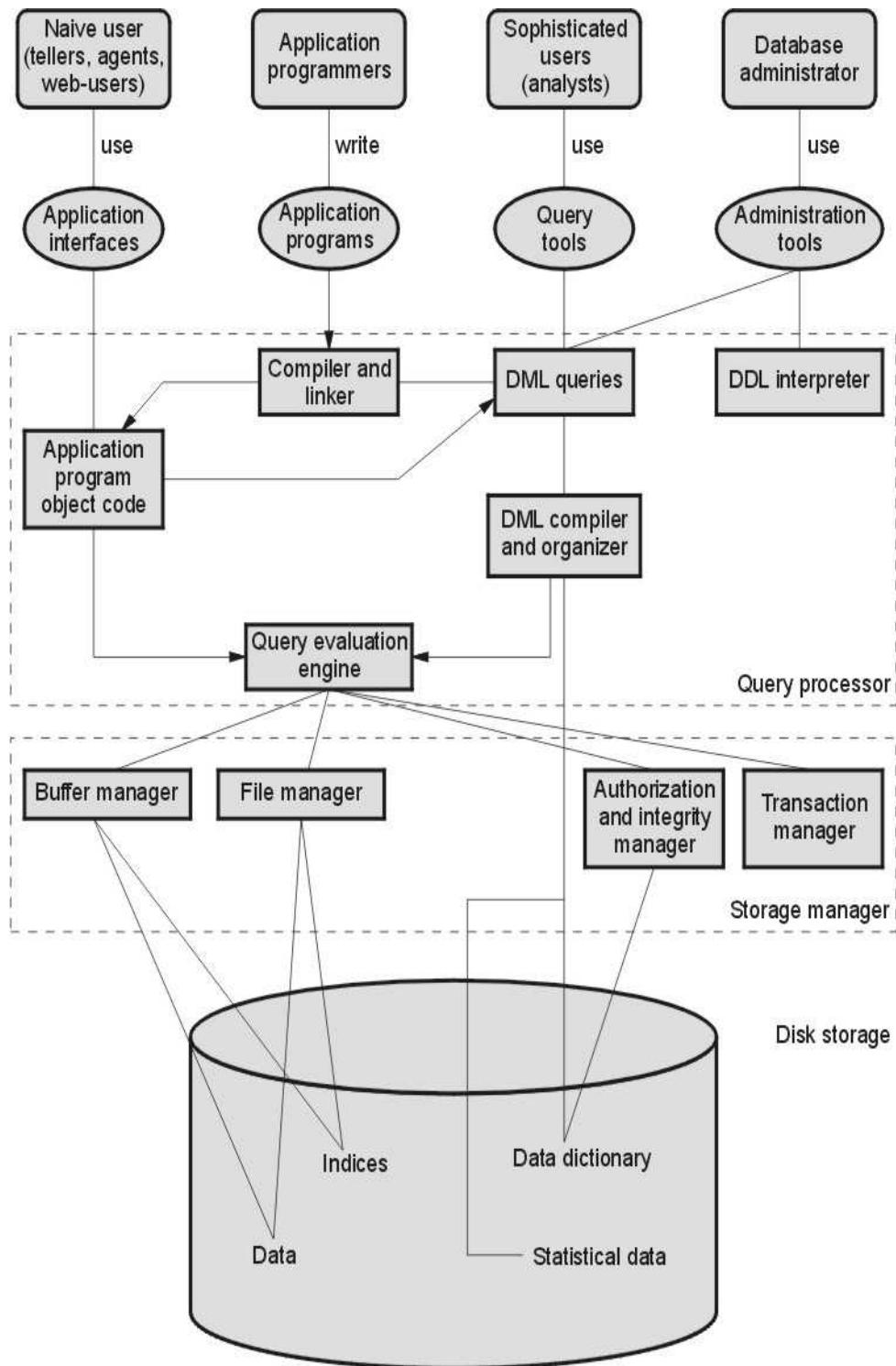


Fig. 1.5.1 Architecture of database

University Questions

1. Explain the overall architecture of database system in detail.

AU : May-14,17, Dec.-17, Marks 8, May-16, Marks 16

2. With the help of a neat block diagram explain basic architecture of a database management system.

AU : May-12, May 13, Marks 16, Dec.-15, Marks 8

Q. Explain component modules of a DBMS and their interactions with the architecture

AU : Dec 08, Marks 10

1.6 Data Independence

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

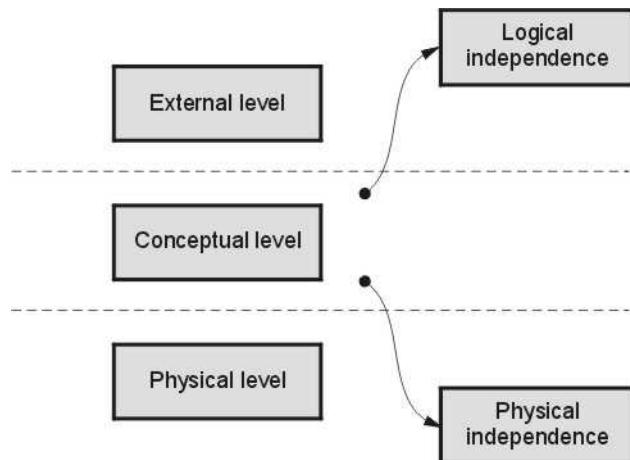


Fig. 1.6.1 Data independence

- 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.
- 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.

Part II Relational Databases

1.7 Introduction to Relational Databases

- 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. 1.7.1 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. 1.7.2 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. 1.7.3 Admission

From this third table we can easily find out that the course to which the RollNo 001 is admitted is computer Science.

1.8 Relational Model

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 1.8.1 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

1.9 Keys

AU : May-06, 07, 12, Dec.-06

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. 1.9.1 Student

The superkey can be represented as follows

Superkey	(RollNo, Phone, Name) Superkey			(Name, Marks) Not a Superkey
RegNo.	RollNo	Phone	Name	Marks
R101	001	1111111111	AAA	88
R102	002	2222222222	BBB	83
R103	003	3333333333	CCC	98
R104	004	4444444444	DDD	67

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

Candidate key	Candidate key			
RegNo.	RollNo	Phone	Name	Marks
R101	001	1111111111	AAA	88
R102	002	2222222222	BBB	83
R103	003	3333333333	CCC	98
R104	004	4444444444	DDD	67

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	1111111111	AAA	88
R102	002	2222222222	BBB	83
R103	003	3333333333	CCC	98
R104	004	4444444444	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

Candidate Key=Super Key – 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.

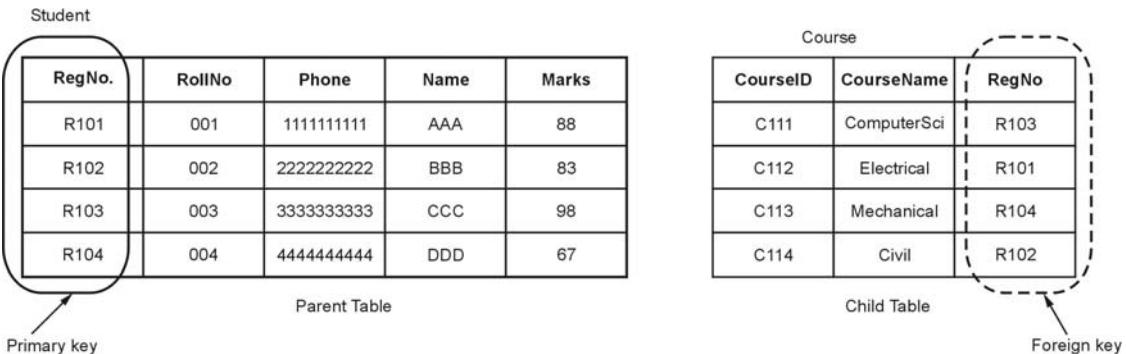
4) Alternate key : The alternate key is a candidate key which is not chosen by the database designer to uniquely identify the tuples. For example –

Primary key Alternate key

RegNo.	RollNo	Phone	Name	Marks
R101	001	1111111111	AAA	88
R102	002	2222222222	BBB	83
R103	003	3333333333	CCC	98
R104	004	4444444444	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'

University Question

1. Explain distinction among the terms primary key, candidate key, foreign key and super key with suitable example

AU : May-06, 07, 12, Dec.-06, Marks 4

1.10 Integrity Constraints

AU : Dec.-05

Database integrity means correctness or accuracy of data in the database. A database may have number of integrity constraints. For example –

- (i) The Employee ID and Department ID must consists of two digits.
- (ii) Every Employee ID must start with letter.

The integrity constraints are classified based on the concept of primary key and foreign key. Let us discuss the classification of constraints based on primary key and foreign key as follows –

1.10.1 Entity Integrity Rule

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.

1.10.2 Referential Integrity Rule

- Referential integrity refers to the **accuracy and consistency** of data within a relationship.
- 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”.
- **Example :** Consider the situation where you have two tables : **Employees** and **Managers**. The **Employees** table has a foreign key attribute entitled **ManagedBy**, which points to the record for each employee’s manager in the **Managers** table.

Referential integrity enforces the **following three rules :**

- i) You cannot add a record to the **Employees** table unless the **ManagedBy** attribute points to a valid record in the **Managers** table. Referential integrity prevents the insertion of incorrect details into a table. Any operation that doesn't satisfy referential integrity rule fails.
- ii) If the primary key for a record in the **Managers** table changes, all corresponding records in the **Employees** table are modified.
- iii) If a record in the **Managers** table is deleted, all corresponding records in the **Employees** table are deleted.

Advantages of Referential Integrity

Referential integrity offers following advantages :

- i) Prevents the entry of duplicate data.
- ii) Prevents one table from pointing to a nonexistent field in another table.
- iii) Guaranteed consistency between "partnered" tables.
- iv) Prevents the deletion of a record that contains a value referred to by a foreign key in another table.
- v) Prevents the addition of a record to a table that contains a foreign key unless there is a primary key in the linked table.

University Question

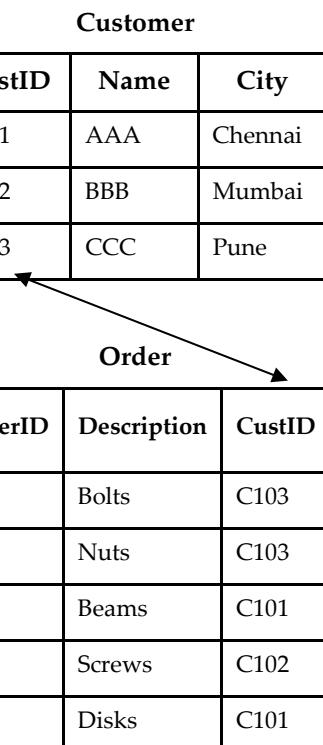
1. Discuss the entity Integrity and referential integrity constraints. Why are they important ? Explain them with suitable examples.

AU : Dec.-05, Marks 10

1.11 Database integrity

- The foreign key is a key in one table that refers to the primary key of another table.
- The foreign key is basically used to link two tables. For example –

Consider Customer table as follows –



Customer

CustID	Name	City
C101	AAA	Chennai
C102	BBB	Mumbai
C103	CCC	Pune

Order

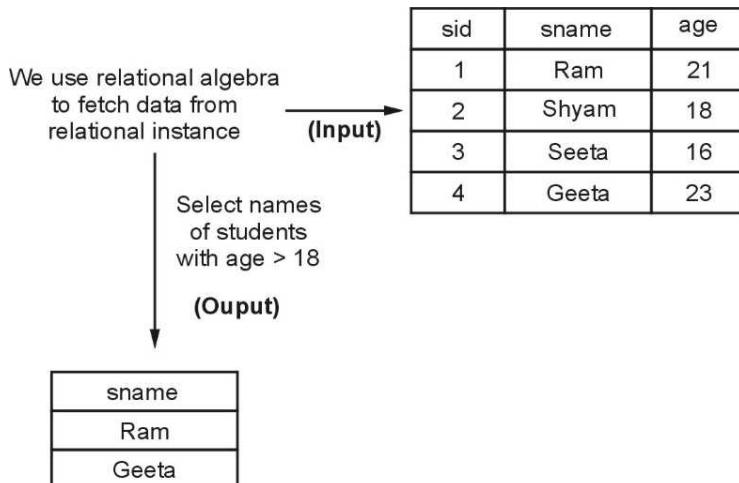
OrderID	Description	CustID
111	Bolts	C103
222	Nuts	C103
333	Beams	C101
444	Screws	C102
555	Disks	C101

- Note that the "CustID" column in the "Order" table points to the "CustID" column in the "Customer" table.
- The "CustID" column in the "Customer" table is the PRIMARY KEY in the "Customer" table.
- The "CustID" column in the "Order" table is a FOREIGN KEY in the "Order" table.
- The table containing the foreign key is called the **child table**, and the table containing the primary key is called the **referenced or parent 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.

1.12 Relational Algebra AU : May-03,04,05,14,15,16,17,18, Dec.-02,07,08,11,15,16,17

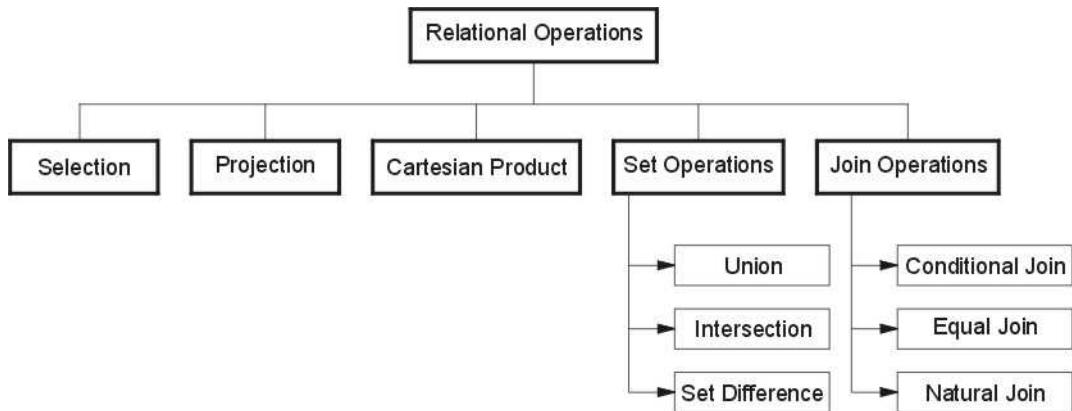
- There are two formal query languages associated with relational model and those are relational algebra and relational calculus.
- Definition :** Relational algebra is a procedural query language which is used to access database tables to read data in different ways.
- The queries present in the relational algebra are denoted **using operators**.
- Every operator in relational algebra accepts relational instances (tables) as input and returns relational instance as output. For example :



- Each **relational algebra is procedural**. That means Each relational query describes a step-by-step procedure for computing the desired answer, based on the order in which operators are applied in the query.
- A sequence of relational algebra operations forms a relational algebra expression, whose result will also be a relation that represents the result of a database query. The By composing the operators in relational expressions the complex relation can be defined.

1.12.1 Relational Operations

Various types of relational operations are as follows -



(1) Selection :

- This operation is used to fetch the **rows or tuples** from the table(relation).
- **Syntax :** The syntax is
 $\sigma_{\text{predicate}}(\text{relation})$
- where σ represents the select operation. The predicate denotes some logic using which the data from the relation(table) is selected.
- For example - Consider the relation student as follows

sid	sname	age	gender
1	Ram	21	Male
2	Shyam	18	Male
3	Seeta	16	Female
4	Geeta	23	Female

Fig.1.12.1 Student Table

Query : Fetch students with age more than 18

We can write it in relational algebra as

$$\sigma_{\text{age} > 18}(\text{Student})$$

The output will be -

sname
Ram
Geeta

We can also specify conditions using and, or operators.

$\sigma_{\text{age} > 18 \text{ and } \text{gender} = \text{'Male'}}$ (**Student**)

sname
Ram

(2) Projection :

- Project operation is used to project only a certain set of attributes of a relation. That means if you want to see only the names all of the students in the Student table, then you can use Project operation.
- Thus to **display particular column** from the relation, the projection operator is used.
- It will only project or show the columns or attributes asked for, and will also remove duplicate data from the columns.
- **Syntax:**

$\Pi C_1, C_2 \dots (r)$

where C_1, C_2 etc. are attribute names(column names).

- **For example** - Consider the Student table given in Fig. 1.12.2.

Query : Display the name and age all the students

This can be written in relational algebra as

$\Pi_{\text{sname}, \text{age}}(\text{Student})$

Above statement will show us only the Name and Age columns for all the rows of data in Student table.

sname	age
Ram	21
Shyam	18
Seeta	16
Geeta	23

Fig. 1.12.2

(3) Cartesian product :

- This is used to combine data from two different relations(tables) into one and fetch data from the combined relation.
- **Syntax : $A \times B$**
- **For example :** Suppose there are two tables named Student and Reserve as follows

Student			Reserve		
sid	sname	age	sid	isbn	day
1	Ram	21	1	005	07-07-18
2	Shyam	18	2	005	03-03-17
3	Seeta	16	3	007	08-11-16
4	Geeta	23			

- **Query :** Find the names of all the students who have reserved isbn = 005. To satisfy this query we need to extract data from two table. Hence the cartesian product operator is used as

$(\sigma_{\text{Student.sid} = \text{Reserve.sid} \wedge \text{Reserve.isbn} = 005} (\text{Student} \times \text{Reserve}))$

As an output we will get

sid	sname	age	sid	isbn	day
1	Ram	21	1	005	07-07-18
2	Shyam	18	2	005	03-03-18

Note : that although the sid column is same, it is repeated.

- **(4) Set operations :** Various set operations are - union, intersection and set-difference.

Let us understand each of these operations with the help of examples.

(i) Union:

- This operation is used to fetch data from two relations(tables) or temporary relation(result of another operation).
- For this operation to work, the relations(tables) specified should have **same number of attributes(columns)** and **same attribute domain**. Also the **duplicate tuples are automatically eliminated** from the result.
- **Syntax :** $A \cup B$
- where A and B are relations.
- **For example :** If there are two tables student and book as follows –

Student		
sid	sname	age
1	Ram	21
2	Shyam	18
3	Seeta	16
4	Geeta	23

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

- **Query :** We want to display both the student name and book names from both the tables then

$$\Pi_{\text{sname}}(\text{Student}) \cup \Pi_{\text{bname}}(\text{Book})$$

(ii) Intersection :

- This operation is used to fetch data from both tables which is common in both the tables.
- **Syntax :** $A \cap B$
where A and B are relations.
- Example – Consider two tables – Student and Worker

Student

Name	Branch
AAA	ComputerSci
BBB	Mechanical
CCC	Civil
DDD	Electrical

Worker

Name	Salary
XXX	3000
AAA	2000
YYY	1500
DDD	2500

- **Query :** If we want to find out the names of the students who are working in a company then

$$\Pi_{\text{name}}(\text{Student}) \cap \Pi_{\text{name}}(\text{Worker})$$

Name
AAA
DDD

- **Set-Difference :** The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

Syntax : A – B

For Example : Consider two relations Full_Time_Employee and Part_Time_Employee, if we want to find out all the employee working for Fulltime, then the set difference operator is used -

$$\Pi_{\text{EmpName}}(\text{Full_Time_Employee}) - \Pi_{\text{EmpName}}(\text{Part_Time_Employee})$$

(5) Join : The join operation is used to **combine information from two or more relations**. Formally join can be defined as a cross-product followed by selections and projections, joins arise much more frequently in practice than plain cross-products. The join operator is used as \bowtie

There are three types of joins used in relational algebra

- i) **Conditional join :** This is an operation in which information from two tables is combined using **some condition** and this condition is specified along with the join operator.

$$A \bowtie_c B = \sigma_c(A \times B)$$

Thus \bowtie is defined to be a cross-product followed by a selection. Note that the condition c can refer to attributes of both A and B. The condition C can be specified using $<$, $<=$, $>$, $<=$ or $=$ operators.

For example consider two table student and reserve as follows -

Student		
sid	sname	age
1	Ram	21
2	Shyam	18
3	Seeta	16
4	Geeta	23

Reserve		
sid	isbn	day
1	005	07-07-18
2	005	03-03-17
3	007	08-11-16

If we want the names of students with sid(Student) = sid(Reserve) and isbn = 005, then we can write it using Cartesian product as -

$$(\sigma_{(\text{Student.sid} = \text{Reserve.sid}) \wedge (\text{Reserve.isbn} = 005)}(\text{Student} \times \text{Reserve}))$$

Here there are two conditions as

- i) $(\text{Student.sid} = \text{Reserve.sid})$ and ii) $(\text{Reserve.isbn} = 005)$ which are joined by \wedge operator.

Now we can use \bowtie_c instead of above statement and write it as –

$$(\text{Student} \bowtie_{(\text{Student.sid} = \text{Reserve.sid}) \wedge (\text{Reserve.isbn} = 005)} \text{Reserve})$$

The result will be -

sid	sname	age	isbn	day
1	Ram	21	005	07-07-18
2	Shyam	18	005	03-03-18

- ii) **Equijoin** : This is a kind of join in which there is **equality condition** between **two attributes(columns) of relations(tables)**. For example - If there are two table Book and Reserve table and we want to find the book which is reserved by the student having isbn 005 and name of the book is 'DBMS', then :

Book			Reserve		
isbn	bname	Author	sid	isbn	day
005	DBMS	XYZ	1	005	07-07-18
006	OS	PQR	2	005	03-03-17
007	DAA	ABC	3	007	08-11-16

$(\sigma_{bname} = 'DBMS' (Book \bowtie_{(Book.isbn = Reserve.isbn)} Reserve))$

Then we get

isbn	bname	Author	sid	day
005	DBMS	XYZ	1	07-07-18
005	DBMS	XYZ	2	03-03-18

- iii) **Natural Join** : When there are **common columns** and we have to **equate** these **common columns** then we use natural join. The symbol for natural join is simply \bowtie without any condition. For example, consider two tables -

Book			Reserve		
isbn	bname	Author	sid	isbn	day
005	DBMS	XYZ	1	005	07-07-18
006	OS	PQR	2	005	03-03-17
007	DAA	ABC	3	007	08-11-16

Now if we want to list the books that are reserved, then that means we want to match **Books.isbn** with **Reserve.isbn**. Hence it will be simply

Books \bowtie Reserve

(6) Rename operation : This operation is used to rename the output relation for any query operation which returns result like Select, Project etc. Or to simply rename a relation(table). The operator ρ (rho) is used for renaming.

Syntax : ρ (RelationNew, RelationOld)

For example : If you want to create a relation Student_names with **sid** and **sname** from **Student**, it can be done using rename operator as :

$\rho(\text{Student_names}, (\Pi_{\text{sid}, \text{sname}}(\text{Student}))$

(7) Divide operation

The division operator is used when we have to evaluate queries which contain the keyword **ALL**.

It is denoted by A/B where A and B are instances of relation.

For example - Find all the customers having accounts in all the branches. For that consider two tables - Customer and Account as

Customer		Account
Name	Branch	Branch
A	Pune	Pune
B	Mumbai	Mumbai
A	Mumbai	
C	Pune	

Now A/B will give us

Name
A

Here We check all the branches from Account table against all the names from Customer table. We can then find that only customer A has all the accounts in all the branches.

Formal Definition of Division Operation : The operation A/B is define as the set of all x values (in the form of unary tuples) such that for every y value in (a tuple of) B, there is a tuple $\langle x, y \rangle$ in A.

Example 1.12.1 Consider following databases reserves(sid, bid, day) sailors (sid, sname, rating, age) boats (bid, bname, color)

- (i) Find the names of sailors who have reserved boat number 103
- (ii) Find the names of sailors who have reserved a red boat

- (iii) Find the id of sailors with age over 20 who have not reserved red boat
 (iv) Find the names of sailors who have reserved at least one boat

Solution :

- (i) $(\Pi_{\text{surname}}((\sigma_{\text{bid}=103} \text{ Reserves}) \bowtie \text{ Sailors}))$
- (ii) $(\Pi_{\text{surname}}((\sigma_{\text{color}=\text{'red'}} \text{ Boats}) \bowtie \text{ Reserves} \bowtie \text{ Sailors}))$
- (iii) $(\Pi_{\text{sid}}((\sigma_{\text{age}>20} \text{ Sailors}) - \Pi_{\text{sid}}((\sigma_{\text{color}=\text{'red'}} \text{ Boats}) \bowtie \text{ Reserves}))$
- (iv) $(\Pi_{\text{surname}}(\text{Sailors} \bowtie \text{ Reserves}))$

Example 1.12.2 Consider the following expressions, which use the result of a relational algebra operation as the input to another operation. For each expression explain in words what the expression does : a) $\sigma_{\text{year} \geq 2009}(\text{takes}) \bowtie \text{ Student}$ b) $\sigma_{\text{year} \geq 2009}(\text{takes} \bowtie \text{ Student})$ c) $\Pi_{\text{ID}, \text{name}, \text{course-id}}(\text{student} \bowtie \text{ takes})$

Solution :

- a. Select each student who takes at least one course in 2009, display the student information along with the information about what the courses the student took.
- b. Select each student who takes at least one course in 2009, display the student information along with the information about what the courses the student took but the selection must be before join operation.
- c. Display the ID, Name and Course_id of all the students who took any course in the university.

Example 1.12.3 Consider following relational database

branch(branch_name, branch_city, assets)
customer (customer_name, customer_street, customer_city)
loan (loan_number, branch_name, amount)
borrower (customer_name, loan_number)
account (account_number, branch_name, balance)
depositor (customer_name, account_number)

i) Find the names of all branches located in "Chennai".
 ii) Find the names of all borrowers who have a loan in branch "ABC".

Solution :

- i) $\Pi_{\text{branch_name}(\sigma_{\text{branch_city}=\text{'Chennai'}})}(\text{branch})$
- ii) $\Pi_{\text{customer_name}(\sigma_{\text{branch_name}=\text{'ABC'}})}(\text{borrower} \bowtie \text{ loan})$

Example 1.12.4 | *author (author_id, first_name, last_name)**author_pub(author_id, pub_id, author_position)**book(book_id, book_title, month, year, editor)**pub(pub_id, title, book_id)*

- Give the relational algebra expression that returns names of all the authors that are book editors*
- Give the relational algebra expression that returns names of all the authors that are not book editors*
- Write a relational algebra expression that returns the names of all authors who have at least one publication in the database.*

Solution :i) $(\Pi_{\text{first_name}, \text{last_name}}(\text{author} \text{ } \text{author_id} = \text{editor} \bowtie \text{book}))$ ii) $(\Pi_{\text{first_name}, \text{last_name}}((\Pi_{\text{author_id}}(\text{author}) - \Pi_{\text{editor}}(\text{book})) \times \text{author}))$ iii) $(\Pi_{\text{first_name}, \text{last_name}}(\text{author} \times \text{author_pub}))$ **Example 1.12.5** | Consider the following schema :*Supplier(sid, sname, address)**Parts(pid, pname, color)**Catalogue(sid, pid, cost)**Write the relational algebraic queries for the following :*

- Find the sids of supplier who supply some red or some green parts*
- Find the sids of supplier who supply every red or some green parts*
- Find the pids of parts supplied by at least two different suppliers*

Solution :i) $\rho(R1, \Pi_{\text{sid}}((\Pi_{\text{pid}} \sigma_{\text{color}=\text{'red'}} \text{Parts}) \bowtie \text{Catalogue}))$ $\rho(R2, \Pi_{\text{sid}}((\Pi_{\text{pid}} \sigma_{\text{color}=\text{'red'}} \text{Parts}) \bowtie \text{Catalogue}))$ $R1 \cup R2$ ii) $\rho(R1, \Pi_{\text{sid}, \text{pid}} \text{Catalogue}) / (\Pi_{\text{pid}} \sigma_{\text{color}=\text{'red'}} \text{Parts})$ $\rho(R2, \Pi_{\text{sid}}((\Pi_{\text{pid}} \sigma_{\text{color}=\text{'red'}} \text{Parts}) \bowtie \text{Catalogue}))$ $R1 \cup R2$ iii) $\rho(R1, \text{Catalogue})$ $\rho(R2, \text{Catalogue})$ $(\Pi_{R1.\text{pid}} \sigma_{R1.\text{pid} = R2.\text{pid} \wedge R1.\text{sid} \neq R2.\text{sid}} (R1 \times R2))$

Example 1.12.6 Consider the relational database

employee (person-name, street, city)

works (person-name, company-name, salary)

company (company-name, city)

manages (person-name, manager-name)

where primary keys are underlined.

(a) Find the names of all employees who work for First Bank Corporation

(b) Find the names, street address, and cities of residence of all employees who work for First Bank Corporation and earn more than 200,000 per annum.

(c) Find the names of all employees in this database who live in the same city as the company for which they work.

AU : Dec.-06, Marks 8

Solution :

- a) $\Pi_{\text{person-name}}(\sigma_{\text{company-name} = \text{"First Bank Corporation"}}(\text{works}))$
- b) $\Pi_{\text{person-name}, \text{street}, \text{city}}(\sigma_{\text{company-name} = \text{"First Bank Corporation"} \wedge \text{salary} > 200000} (\text{works} \bowtie \text{employee}))$
- c) $\Pi_{\text{person-name}}(\text{works} \bowtie \text{employee} \bowtie \text{company})$

University Questions

1. Explain select, project, cartesian product and join operations in relational algebra with an example

AU : May-18, Marks 13, Dec.-16, Marks 6

2. List operations of relational algebra and purpose of each with example

AU : May-17, Marks 5

3. Differentiate between foreign key constraints and referential Integrity constraints with suitable example.

AU : Dec.-17, Marks 6

4. Explain various operations in relational algebra with examples

AU : May 03, Marks 10, Dec-07, Marks 8, Dec.- 08, Marks 10, May-14, Marks 16

5. Explain all join operations in relational algebra

AU : May 05, Marks 8

6. Briefly explain relational algebra

AU : May 04, Marks 8

7. What is rename operation in relational algebra ? Illustrate your answer with example

AU : Dec 02, Marks 2

Part III Structured Query Language(SQL)**1.13 SQL Fundamentals****AU : Dec.-15**

- Structure Query Language(SQL) is a database query language used for storing and managing data in Relational DBMS.
- Various parts of SQL are –
 - **Data Definition Language(DDL)** : It consists of a set of commands for defining relation schema, deleting relations, and modifying relation schemas.
 - **Data Manipulation Language(DML)** : It consists of set of SQL commands for inserting tuples into relational schema, deleting tuples from or modifying tuples in databases.
 - **Integrity** : The SQL DDL includes commands for specifying integrity constraints. These constraints must be satisfied by the databases.
 - **View definition** : The SQL DDL contains the commands for defining views for database.
 - **Transaction control** : The SQL also includes the set of commands that indicate beginning and ending of the transactions.
 - **Embedded SQL and Dynamic SQL** : There is a facility of including SQL commands in the programming languages like C,C++, COBOL or Java.
 - **Authorization** : The SQL DDL includes the commands for specifying access rights to relations and views.

1.13.1 Data Abstraction

The Basic data types used in SQL are –

- (1) **char(n)**: For representing the fixed length character string this data type is used.
For instance – to represent name, designation, coursename, we use this data type. Instead of char we can also use **character**. The n is specified by the user.
- (2) **varchar(n)** : The varchar means character varying. That means – for denoting the variable length character strings this data type is used. The n is user specified maximum character length.
- (3) **int** : For representing the numeric values without precision, the int data type is used.
- (4) **numeric** : For representing, a fixed point number with user-specified precision this data type is used. The number consists of m digits plus sign k digits are to the right of precision. For instance the numeric(3,2) allows 333.11 but it does not allow 3333.11

(5) **smallint** : It is used to store small integer value. It allows machine dependent subset of integer type.

(6) **real** : It allows the floating point, double precision numbers.

(7) **float(n)** : For representing the floating point number with precision of at least n digits this data type is used.

1.13.2 Basic Schema Definition

In this section, we will discuss various SQL commands for creating the schema definition.

There are three types of SQL Languages -

1. DDL commands : DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in database.

Examples of DDL commands are :

- **CREATE** – is used to create the database or its objects such as table, function, views and so on.
- **DROP** – is used to delete objects from the database.
- **ALTER**–is used to alter the structure of the database.
- **TRUNCATE**–is used to remove all records from a table, including all spaces allocated for the records are removed.
- **COMMENT** –is used to add comments to the data dictionary.
- **RENAME** –is used to rename an object existing in the database.

2. DML commands : DML stands for Data Manipulation Language. These commands deal with manipulation of data present in the database.

Examples of DML commands are :

- **SELECT** – is used to retrieve data from the a database.
- **INSERT** – is used to insert data into a table.
- **UPDATE** – is used to update existing data within a table.
- **DELETE** – is used to delete records from a database table.

3. DCL commands : It stands for Data Control Language. It includes commands such as GRANT and REVOKE which mainly deals with the rights, permissions and other controls of the database system.

Examples of DCL commands :

- GRANT-gives user's access privileges to database.
- REVOKE-withdraw user's access privileges given by using the GRANT command.

Let us discuss various commonly used SQL commands that help in building the basic schema.

(1) Create Table : The database relation can be created by using the **create table** command

Syntax

```
create table table_name;
```

Example

```
create table Student
(RollNo int,
Name varchar(10),
Marks numeric(3,2),
Primary key(RollNo));
```

The primary key attribute must be non null and unique.

(2) Insert : The insert command is used to insert data into the table. There are two syntaxes of inserting data into SQL

Syntax

- i) **Insert into** table_name (column1, column2, column3, ...)
values (value1, value2, value3, ...);
- ii) **insert into** table_name
values (value1, value2, value3, ...);

Example

```
(i) insert into Student(RollNo,Name,Makrs) values(101,'AAA',56.45)
(ii) insert into Student values(101,'AAA',56.45)
```

(3) Delete : This command is used to delete the existing record.

Syntax

```
delete from table_name
where condition;
```

Example

```
Delete from student
where RollNo=10
```

(4) Alter: The **alter table** statement is used to add, delete, or modify columns in an existing table.

The **alter table** statement is also used to add and drop various constraints on an existing table.

Syntax for adding a columns

```
alter table table_name  
add column_name datatype;
```

Example

```
Alter table student  
Add address varchar(20)  
Syntax for dropping column  
Alter table table_name  
drop column column_name;
```

Example

```
Alter table student  
drop column address;
```

1.13.3 Basic Structure of SQL Queries

The **basic form** of SQL queries is

SELECT-FROM-WHERE. The syntax is as follows :

```
SELECT [DISTINCT] target-list  
FROM relation-list  
WHERE qualification
```

- **SELECT** : This is one of the fundamental query command of SQL. It is similar to the projection operation of relational algebra. It selects the attributes based on the condition described by WHERE clause.
- **FROM** : This clause takes a relation name as an argument from which attributes are to be selected/projected. In case more than one relation names are given, this clause corresponds to Cartesian product.
- **WHERE** : This clause defines predicate or conditions, which must match in order to qualify the attributes to be projected.
- **Relation-list** : A list of relation names(tables)
- **target-list** : A list of attributes of relations from relation list(tables)
- **qualification** : Comparisons of attributes with values or with other attributes combined using AND, OR and NOT.
- **DISTINCT** 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.

Example

```
SELECT sname  
FROM Student  
WHERE age>18
```

- The above query will return names of all the students from student table where age of each student is greater than 18

1.13.3.1 Queries on Multiple Relations

Many times it is required to access multiple relations(tables) to operate on some information. For example consider two tables as **Student** and **Reserve**.

sid	sname	age
1	Ram	21
2	Shyam	18
3	Seeta	16
4	Geeta	23

sid	isbn	day
1	005	07-07-18
2	007	03-03-18
3	009	

Query : Find the names of students who have reserved the books with book isbn

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

Use of SQL Join

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.

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

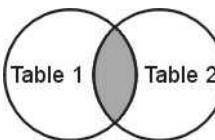
Student		
sid	cid	sname
1	101	Ram
2	101	Shyam
3	102	Seeta
4	NULL	Geeta

Reserve	
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 table1 with each row of Table2 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

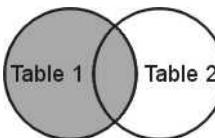
```
SELECT *
FROM Student Inner Join City on Student.cid=City.cid
```

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 :

- 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

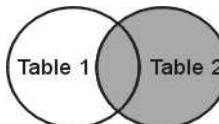
```

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 :

- 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

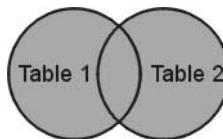
```

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 :

- 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

1.13.4 Additional Basic Operations

1) The Rename Operation : 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

sid	sname	age
1	Ram	21
2	Shyam	18
3	Seeta	16
4	Geeta	23

Reserve

sid	isbn	day
1	005	07-07-18
2	007	03-03-18
3	009	05-05-18

```
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) Attribute Specification in Select clause : The symbol * is used in select clause to denote **all attributes**. For example – To select all the records from Student table we can write

```
Select* from Student
```

3) Ordering the display of tuples : For displaying the records in particular order we use **order by** clause.

The general syntax with ORDER BY is

```
SELECT column_name(s)
FROM table_name
WHERE condition
ORDER 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 names of students from highest marks to lowest

```
Select sname
From Student
Order By marks
```

We can also use the **desc** for descending order and **asc** for ascending order. For example :

In order to display names of the students in descending order of city – we can specify

```
Select sname
From Student
Order by city desc;
```

(4) Where clause Predicate :

- (i) 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;
```

Similarly we can make use of the comparison operators for various attributes. 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 sname
From Student , Reserve
Where (Student.sid,Reserve.isbn)=(Reserve.sid,005);
```

- (ii) We can use AND, OR and NOT operators in the Where clause. For filtering the records based on more than one condition, the AND and OR operators can be used. The NOT operator is used to demonstrate when the condition is not TRUE.

Consider following sample database – **Students** database, for applying AND, OR and NOT operators

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

Syntax of AND

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

Example : Find the student having name “AAA” and lives in city “Pune”

```
SELECT *
FROM Students
Where sname='AAA' AND city='Pune'
```

Output

sid	sname	marks	city
1	AAA	60	Pune

Syntax OR

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

Example : Find the student having name “AAA” OR lives in city “Pune”

```
SELECT *
FROM Students
Where sname='AAA' OR city='Pune'
```

Output

sid	sname	marks	city
1	AAA	60	Pune
3	CCC	90	Pune

Syntax NOT

```
SELECT column1, column2, ..
FROM table_name
WHERE NOT condition
```

Example : Find the student who do not have city “Pune”

```
SELECT *
FROM Students
Where NOT city='Pune'
```

Output

sid	sname	marks	city
2	BBB	70	Mumbai
4	DDD	55	Mumbai

1.13.5 Domain and Key Constraint

Domain Constraint

- A domain is defined as the set of all unique values permitted for an attribute. For example, a domain of date is the set of all possible valid dates, a domain of Integer is all possible whole numbers, and a domain of day-of-week is Monday, Tuesday ... Sunday.
- This in effect is defining rules for a particular attribute. If it is determined that an attribute is a date then it should be implemented in the database to prevent invalid dates being entered.
- Domain constraints are user defined data type and we can define them like this :
- Domain constraint = Data type + Constraints
- The constraints can be specified using NOT NULL / UNIQUE / PRIMARY KEY / FOREIGN KEY / CHECK / DEFAULT.
- For example –

```
Create domain id_value integer
constraint id_test
check(value > 100); ← cheking if stud_id value is greater than 100

create table student (
stu_id id_value PRIMARY KEY,
stu_name CHAR(30),
stu_age integer
);
```

Key Constraint

- A key constraint is a statement that a certain minimal subset of the fields of a relation is a unique identifier for a tuple.
- For example - Consider the students relation and the constraint that no two students have the same student id. This IC is an example of a key constraint.
- The **definition** of key constraints contain **two parts** -
 - Two distinct tuples in a legal instance (an instance that satisfies all Integrity Constraints including the key constraint) **cannot have identical values** in all the fields of a key.
 - No subset of the set of fields in a key is a **unique identifier** for a tuple.
- The **first part** of the definition means that, in any legal instance, the values in the key fields uniquely identify a tuple in the instance. When specifying a key constraint, the DBA or user must be sure that this constraint will not prevent them from storing a 'correct' set of tuples. For example, several students may have the same name, although each student has a unique student id. If the name field is declared to be a key, the DBMS will not allow the Students relation to contain two tuples describing different students with the same name.

- The **second part** of the definition means, for example, that the set of fields {RollNo, Name} is not a key for Students, because this set properly contains the key {RollNo}. The set {RollNo, Name} is an example of a superkey, which is a set of fields that contains a key.
- The key constraint can be specified using SQL as follows -
 - In SQL, we can declare that a subset of the columns of a table constitute a key by using the UNIQUE constraint.
 - At most one of these candidate keys can be declared to be a primary key, using the PRIMARY KEY constraint. For example -

```
CREATE TABLE Student(RollNo integer,
                     Name CHAR(20),
                     age integer,
                     UNIQUE(Name,age),
                     CONSTRAINT StudentKey PRIMARY KEY(RollNo))
```

This definition says that **RollNo** is a **Primary key** and Combination of **Name and age** is also a key.

1.13.6 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

1.13.7 Set Operations

- 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.

Consider Following relations –

Student		
sid	sname	age
1	Ram	21
2	Shyam	18
3	Seeta	16
4	Geeta	23

Reserve		
sid	isbn	day
1	005	07-07-18
2	005	03-03-17
3	007	08-11-16

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

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'
```

1.13.8 Aggregate Functions

- 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 :

1.13.8.1 Basic Aggregation

- 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

Test	
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

1.13.8.2 Use of Group By and Having Clause**(i) 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)
```

The general syntax with ORDER BY is

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

Output

SUM(marks)	city
150	Pune
125	Mumbai

(ii) 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')
```

Output

- The result will be as follows –

SUM(marks)	city
150	Pune
125	Mumbai

1.13.9 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 :

- 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 cname='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.

1.13.10 | Modification of Databases

The modification of database is an operation for making changes in the existing databases. Various operations of modification of database are – insertion, deletion and updation of databases.

1. Deletion : The **delete** command is used to delete the existing record.

Syntax

```
delete from table_name  
where condition;
```

Example

```
delete from student  
where RollNo=10
```

- 2. Insertion :** The insert command is used to insert data into the table. There are two syntaxes of inserting data into SQL

Syntax

```
(i) Insert into table_name (column1, column2, column3, ...)  
values (value1, value2, value3, ...);  
(ii) insert into table_name  
values (value1, value2, value3, ...);
```

Example

```
(i) insert into Student(RollNo,Name,Makrs) values(101,'AAA',56.45)  
(ii) insert into Student values(101,'AAA',56.45)
```

- 3. Update :** The update statement is used to modify the existing records in the table.

```
update table_name  
set column1=value1, column2=value2,...  
where condition;
```

Example:

```
Delete student  
Set Name='WWW'  
where RollNo=101
```

Example 1.13.1 Write the DDL, DML, DCL for the students database. Which contains student details:name, id,DOB, branch, DOJ.

Course details : Course name, Course id, Stud.id, Faculty name, id, marks

AU : Dec.-17, Marks 15

Solution :

DDL Commands

```
CREATE TABLE Student  
(  
stud_name varchar(20),  
stud_id int(3),  
DOB varchar(15),  
branch varchar(10),  
DOJ varchar(15),  
);  
CREATE TABLE Course  
(
```

```
course_name varchar(20),  
course_id int(5),  
stud_id int(3),  
facult_name varchar(20),  
faculty_id varchar(5),  
marks real  
);
```

DML Commands

The commands which we will use here are insert and select. The insert command is used to insert the values into database tables. Using the select command, the database values can be displayed.

(1) Inserting values into Student table

```
insert into Student(stud_name,stud_id,DOB,branch,DOJ)  
values('AAA',11,'01-10-1999' , 'computers','5-3-2018')
```

```
insert into Student(stud_name,stud_id,DOB,branch,DOJ)  
values('BBB',12,'24-5-1988' , 'Mechanical','17-2-2016')
```

```
insert into Student(stud_name,stud_id,DOB,branch,DOJ)  
values('CCC',13,'8-1-1990' , 'Electrical','22-9-2017')
```

(2) Inserting values into Course table

```
insert into Course(course_name,course_id,stud_id,faculty_name,faculty_id,marks)  
values('Basic',101,11,'Archana' , 'F001','50')
```

```
insert into Course(course_name,course_id,stud_id,faculty_name,faculty_id,marks)  
values('Intermediate',102,12,'Rupali' , 'F002','70')
```

```
insert into Course(course_name,course_id,stud_id,faculty_name,faculty_id,marks)  
values('Advanced',103,13,'Sunil' , 'F003','100')
```

(3) Displaying records of Student table

Select * from Student;

(4) Displaying records of Course table

Select * from Course;

DCL Commands

The DCL command is used to control privileges in Database. To perform any operation in the database, such as for creating tables, sequences or views, a user needs privileges.

We will use the command **GRANT**.

To allow a user to create tables in the database, we can use the below command,

Grant create table to user1;

Example 1.13.2 C Write the following queries in relational algebra and SQL

- (i) Find the names of employee who have borrowed a book published by McGraw Hill
- (ii) Find the names of employees who have borrowed all books published by McGraw-Hill

AU : May 17, Marks 10

Solution :

We will assume the databases as –

member(memb_no, name, dob)
books(isbn, title, authors, publisher)
borrowed(memb_no, isbn, date)

(i) Relational Algebra :

$\Pi_{\text{name}}((\sigma_{\text{publisher}=\text{'McGraw Hill'}} \text{books}) \bowtie \text{borrowed} \bowtie \text{member})$

SQL :

```
SELECT name
FROM member
WHERE meber.memb_no=borrowed.memb_no
      AND books.isbn=borrowed.isbn
      AND books.publisher='McGraw Hill';
```

(ii) Relational Algebra

$\rho (\text{Tempname}, (\pi_{\text{memb_no, isbn}} \text{borrowed}) / \pi_{\text{isbn}} (\sigma_{\text{publisher}=\text{'McGraw Hill'}} \text{books}))$
 $\pi_{\text{name}}(\text{Tempname} \bowtie \text{member})$

SQL :

```
SELECT distinct M.name
FROM Member M,
WHERE NOT EXISTS
(
  (SELECT isbn
   FROM books
   WHERE publisher = 'McGrawHill'
  )
EXCEPT
  (SELECT isbn
   FROM borrowed R
```

```

    WHERE R.memb_no = M.memb_no
)
)

```

Example 1.13.3 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.

AU : Dec.-15, Marks 4 + 4 + 4 + 4

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 1.13.4 Consider a student registration database comprising of the below given table schema.

Student File

Student Number	Student Name	Address	Telephone
----------------	--------------	---------	-----------

Course File

Course Number	Description	Hours	Professor Number
---------------	-------------	-------	------------------

Professor File

Professor Number	Name	Office
------------------	------	--------

Registration File

Student Number	Course Number	Date
----------------	---------------	------

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 ?

AU : May 15, Marks 16

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
```

University Questions

- | | |
|---|------------------------------|
| 1. Explain aggregate functions in SQL with example. | AU : May 18, Marks 13 |
| 2. Write DDL, DML,DCL commands for the students database. | AU : Dec 17, Marks 7 |
| 3. Explain about SQL fundamentals. | AU : May 16, Marks 8 |
| 4. Explain about Data Definition Language. | AU : May 16, Marks 8 |
| 5. Explain the six clauses in the syntax of SQL query and show what type of constructs can be specified in each of the six clauses. Which of the six clauses are required and which are optional. | AU : Dec 15, Marks 16 |
| 6. Explain- DDL and DML | AU : Dec 14, Marks 8 |

1.14 Advanced SQL Features

1.14.1 Embedded SQL

- The programming module in which the SQL Statements are embedded is called Embedded SQL module.
- It is possible to embed SQL statements inside the programming language such as C, C++, PASCAL,Java and so on.
- It allows the application languages to communicate with DB and get requested result.
- The high level languages which supports embedding SQLs within it are also known as **host language**.

- An embedded SQL program must be processed by a special preprocessor prior to compilation. The preprocessor replaces embedded SQL requests with host-language declarations and procedure calls that allow runtime execution of the database accesses. Then, the resulting program is compiled by the host-language compiler. This is the main distinction between embedded SQL and JDBC or ODBC.

Example of Embedded SQL – Following program prompts the user for an order number, retrieves the customer number, salesperson, and status of the order, and displays the retrieved information on the screen.

```
int main() {
    EXEC SQL INCLUDE SQLCA;
    EXEC SQL BEGIN DECLARE SECTION;
    int OrderID;      /* Employee ID (from user)      */
    int CustID;       /* Retrieved customer ID      */
    char SalesPerson[10] /* Retrieved salesperson name   */
    char Status[6]     /* Retrieved order status      */
    EXEC SQL END DECLARE SECTION;

    /* Set up error processing */
    EXEC SQL WHENEVER SQLERROR GOTO query_error;
    EXEC SQL WHENEVER NOT FOUND GOTO bad_number;

    /* Prompt the user for order number */
    printf ("Enter order number: ");
    scanf_s("%d", &OrderID);

    /* Execute the SQL query */
    EXEC SQL SELECT CustID, SalesPerson, Status
        FROM Orders
        WHERE OrderID = :OrderID
        INTO :CustID, :SalesPerson, :Status;

    /* Display the results */
    printf ("Customer number: %d\n", CustID);
    printf ("Salesperson: %s\n", SalesPerson);
    printf ("Status: %s\n", Status);
    exit();

query_error:
    printf ("SQL error: %ld\n", sqlca->sqlcode);
    exit();

bad_number:
    printf ("Invalid order number.\n");
    exit();
}
```

Features of Embedded SQL

- (1) It is easy to use.
- (2) It is ANSI/ISO standard programming language.
- (3) It requires less coding
- (4) The precompiler can optimize execution time by generating stored procedures for the Embedded SQL statements.
- (5) It is identical over different host languages, hence writing applications using different programming languages is quite easy.

University Questions

1. What is the need of embedded SQL.

AU : May 17, Dec 17, Marks 2

2. What is embedded SQL ? Give an example

AU : Dec 16, Marks 5, May-14, Dec 14, Marks 8

1.15 Dynamic SQL

AU : May-17, Dec.-17

- Dynamic SQL is a programming technique which allows to build the SQL statements dynamically at runtime.
- Dynamic SQL statements are not embedded in the source program but stored as strings of characters that are manipulated during a program's runtime.
- These SQL statements are either entered by a programmer or automatically generated by the program.
- Dynamic SQL statements also may change from one execution to the next without manual intervention.
- Dynamic SQL facilitates automatic generation and manipulation of program modules for efficient automated repeating task preparation and performance.
- Dynamic SQL facilitates the development of powerful applications with the ability to create database objects for manipulation according to user input.
- The simplest way to execute a dynamic SQL statement is with an EXECUTE IMMEDIATE statement. This statement passes the SQL statement to the DBMS for compilation and execution.

Example 1.15.1 Consider the relation student(Reg.No.,name,mark, and grade). Write embedded dynamic SQL program in C language to retrieve all the students' records whose mark is more than 90.

AU : May 17, Marks 11, Dec 17, Marks 6

Solution :

```
int main() {
/* Begin program */
EXEC SQL INCLUDE SQLCA;
```

```

EXEC SQL BEGIN DECLARE SECTION
int Reg_No;
char name[10][10];
float marks;
char grade;

EXEC SQL END DECLARE SECTION
EXEC SQL WHENEVER SQLERROR STOP
EXEC SQL SELECT Reg_No,name,marks,grade
FROM Student
WHERE marks>90
INTO :Reg_No,:name,:marks,:grade;
/* Display the results */
printf ("Registration number: %d\n", Reg_No);
printf ("Name: %s\n", name);
printf ("Marks: %f\n", marks);
printf ("Grade: %c\n", grade);
exit();
EXEC SQL DISCONNECT
/* End program */
}

```

1.16 Two Marks Questions with Answers

Q.1 What is Database Management System ? Why do we need a DBMS ?

AU : May 05, Dec - 08

Ans. :

- 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.

Q.2 What is the purpose of database management system ?

AU : Dec 14

Ans. : The purpose of database management system is –

- **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.

Q.3 List any two advantages of database systems

AU : Dec 07

Ans. : 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.

Q.4 Define data abstraction**AU : May 05**

Ans. : Data abstraction means retrieving only required amount of information of the system and hiding background details.

Q.5 What are three levels of data abstraction ?**AU : Dec 02, 04, May 14, Dec 17**

Ans. : The three levels of data abstraction are –

1. Physical Level
2. Logical Level
3. View Level

Q.6 Is it possible for several attributes to have same domain ? Illustrate your answer with suitable example**AU : Dec 04, Dec 15**

Ans. : A domain is the set of legal values that can be assigned to an attribute. Each attribute in a database must have a well-defined domain; we can't mix values from different domains in the same attribute. Hence it is not possible for several attributes to have same domain.

For example - Student domain has attributes RollNo, Name, Address. Similarly Employee domain has EmpID, Ename, Salary, Address. We can not define the same domain for defining several attributes.

Q.7 Write the characteristic that distinguish the database approach with File based approach**AU : May 15, Dec 16****OR What are main differences between file processing system and a DBMS ?****AU : May 06, Dec 06**

Ans. : Refer section 1.2

Q.8 Discuss briefly three major disadvantages of keeping organizational information in a file processing system**AU : Dec 04, May 16**

Ans. : Refer Section 1.2

Q.9 What is data model ?**AU : Dec 11**

Ans. :

- 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.

Q.10 What are different types of data models ?**AU : May 12****Ans. :** Various types of data models are –

- (1) Relational Data Model (2) Entity Relational Data Model
- (3) Object Based Data Model (4) Semi-structured Data Model

Q.11 Name the categories of SQL commands**AU : May 12****Ans. :** The categories of SQL commands are –

- (1) Data Definition Language (DDL)
- (2) Data Manipulation Language (DML)
- (3) Data Control Language (DCL)

Q.12 What is data definition language ? Give example**AU : Dec 16, May 18****Ans. :**

- Data Definition Language (DDL) is a specialized language used to specify a database schema by a set of definitions.
- It is a language which is used for **creating** and modifying the structures of tables, views, indexes and so on.
- Some of the common commands used in DDL are -**CREATE, ALTER, DROP**.

Q.13 Give brief description of DCL command**AU : Dec 14**

Ans. : DCL stands for Data Control Language. It includes commands such as GRANT and REVOKE which mainly deals with the rights, permissions and other controls of the database system.

Q.14 Define the term tuple**AU : Dec 05****Ans. :** Tuple means a row present in the table**Q.15 Why does SQL allow duplicate tuples in a table or in a query result ?****AU : Dec 15****Ans. :**

- Data can be the same. Two people may have the same name. Since SQL is a database where you store your data and data can be duplicate.

- But we can apply primary key constraints, Unique constraints or Distinct keyword to identify the record uniquely

Q.16 Why key is essential? Write the different types of keys

AU : Dec 04

Ans. :

- 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.

Q.17 Define primary key. Give example.

AU : May 09

Ans. :

- The primary key is a candidate key chosen by the database designer to identify the tuple in the relation uniquely.
- For example – Consider a Student database as Student (RollNo,Name,Address). The primary key for this database is RollNo. The primary is underlined.

Q.18 Define foreign key. Give example

AU : May 18

Ans. :

- Foreign key is a single attribute or collection of attributes in one table that refers to the primary key of other table.
- For example - Consider a Student database as Student (RollNo,Name,Address) and Course(CourseId, CourseName, RollNo). Here RollNo is a foreign key

Q.19 What is the difference between primary key and foreign key ?

AU : Dec 05

Ans. :

Primary Key	Foreign Key
Primary key is a column or a set of columns that can be used to uniquely identify a row in a table	Foreign key is a column or a set of columns that refer to a primary key or a candidate key of another table.
A table can have a single primary key,	A table can have multiple foreign keys that can reference different tables.

Q.20 What is referential integrity ?

AU : May 04,08

Ans. :

- 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”.

- **Example :** Consider the situation where you have two tables : **Employees** and **Managers**. The **Employees** table has a foreign key attribute entitled **ManagedBy**, which points to the record for each employee's manager in the **Managers** table.

Q.21 What is domain integrity? Give example

AU : Dec 08

Ans. : Domain integrity ensures that all the data items in a column fall within a defined set of valid values. Each column in a table has a defined set of values, such as the set of all numbers for zip (five-digit), the set of all character strings for name.

Q.22 What are different types of integrity constraints used in designing relational databases

AU : Dec 07

Ans. : Different types of integrity constraints are –

- (1) Entity Integrity Constraint
- (2) Referential Integrity Constraint
- (3) Domain Integrity Constraint
- (4) Key Integrity Constraint

Q.23 List the reasons why null value might be introduced into the database **AU : May 06**

Ans. : NULL is a special value provided by database in two cases – i) When field values of some tuples are unknown(For e.g. city name is not assigned) and ii) inapplicable(For e.g. middle name is not present).

Q.24 List various operators used in relational algebra

AU : May 06

Ans. : Various operators used in Relational algebra are –

- (1) Selection Operator(σ)
- (2) Projection Operator(Π)
- (3) Cartesian Product(\times)
- (4) Rename Operator(ρ)

Q.25 Describe briefly any two undesirable properties that a database design may have ?

AU : Dec 02

Ans. : The two undesirable properties that a database design may have –

- (1) Repetition of data
- (2) In-ability of representation of certain information in database.

Q.26 Specify with suitable examples, the different types of keys used in database management system.

AU : Dec 02

Ans. : Refer section 1.9

Q.27 Define data independence.

AU : May 08

Ans. : 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.

Q.28 Distinguish between Physical and logical data independence

AU : May 03

Ans. : Refer Section 1.6

Q.29 What is meant by instance and Schema of the database

AU : May 04, Dec 05

Ans. :

- When information is inserted or deleted from the database then the database gets changed. The collection of information at particular moment is called **instances**.
- The overall design of the database is called schema

Q.30 Differentiate between Dynamic SQL and Static SQL

AU : Dec 14, May 15, Dec 15, Dec 16, Dec 17

Ans:

Sr.No.	Static SQL	Dynamic SQL
1	SQL statements are compiled at compile time.	SQL statements are compiled at run time.
2	It is more efficient.	It is less efficient.
3	It is less flexible.	It is more flexible.
4	It is used in the situations where data is distributed uniformly	It is used in situations where data is distributed non uniformly.



Notes

UNIT - II

2

Database Design

Syllabus

Entity-Relationship model - E-R Diagrams - Enhanced-ER Model - ER-to-Relational Mapping - Functional Dependencies - Non-loss Decomposition - First, Second, Third Normal Forms, Dependency Preservation - Boyce/Codd Normal Form - Multi-valued Dependencies and Fourth Normal Form - Join Dependencies and Fifth Normal Form.

Contents

2.1 Introduction to Entity Relationship Model	
2.2 Mapping Cardinality	
2.3 ER Diagrams	
2.4 Enhanced ER Model	
2.5 Examples based on ER Diagram	
2.6 ER to Relational Mapping May-17, Marks 13
2.7 Concept of Relational Database Design	
2.8 Functional Dependencies	
2.9 Concept of Redundancy and Anomalies	
2.10 Decomposition Dec.-17, Marks 7
2.11 Normal Forms Dec.-14, 15, May-18 Marks 16
2.12 Boyce / Codd Normal Form (BCNF)	
2.13 Multivalued Dependencies and Fourth Normal Form	May-14, Dec.-16 Marks 16
2.14 Join Dependencies and Fifth Normal Form	
2.15 Two Marks Questions with Answers	

Part I Entity Relationship Model

2.1 Introduction to Entity Relationship Model

Entity Relational model is a model for identifying entities to be represented in the database and representation of how those entities are related.

Let us first understand the design process of database design.

2.1.1 Design Phases

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

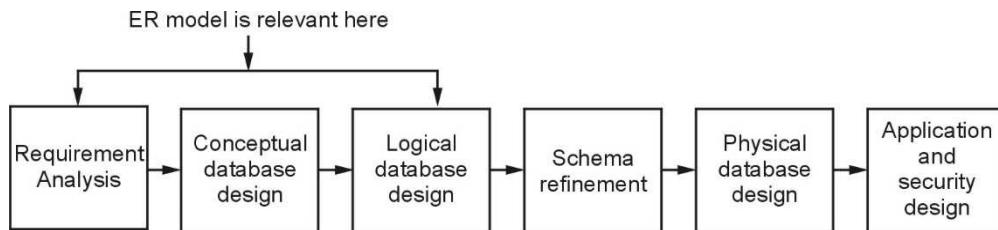


Fig. 2.1.1 : Database design process

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 is **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**.

2.1.2 ER Model

The ER data model specifies enterprise schema that represents the overall logical structure of a database.

The E-R model is very useful in mapping the meanings and interactions of real-world entities onto a conceptual schema.

The ER model consists of three basic concepts –

1) 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.

Student	Employee	Department
---------	----------	------------

- **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.

2) Relationship Sets

Relationship is an association among two or more entities.

The **relationship set** is a collection of similar relationships. For example - Following Fig. 2.1.2 shows the relationship **works_for** for the two entities **Employee** and **Departments**.

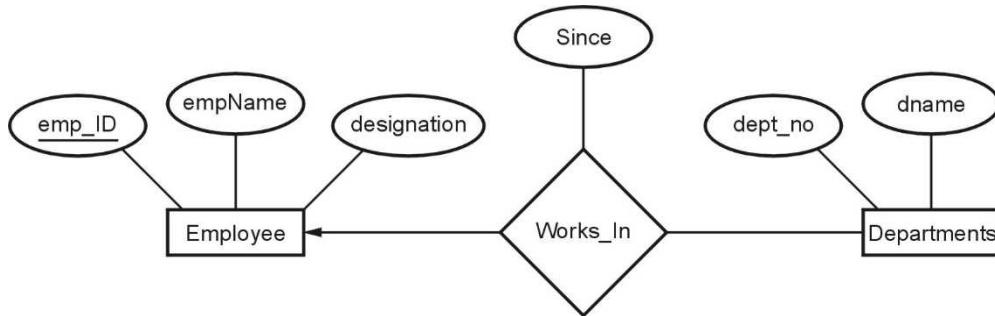


Fig. 2.1.2 : Relation set

The association between entity sets is called as **participation**. that is, the entity sets E₁, E₂, ..., E_n participate in relationship set R.

The function that an entity plays in a relationship is called that entity's **role**.

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. 2.1.3 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.

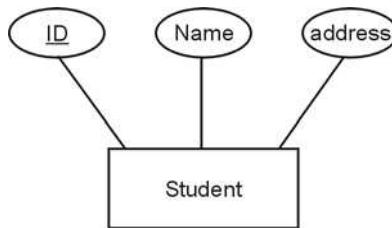


Fig. 2.1.3 : Student entity set with attributes

Types of Attributes

1) Simple and Composite Attributes :

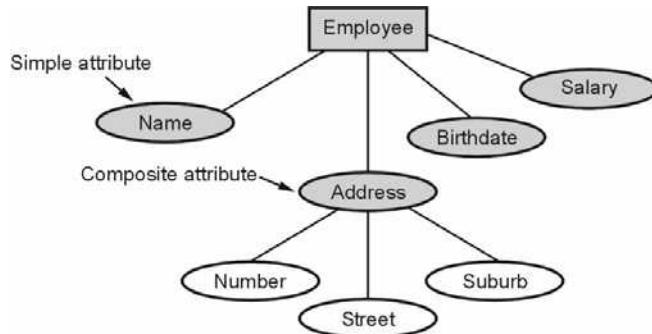
1) Simple attributes are attributes that are drawn from the atomic value domains

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”

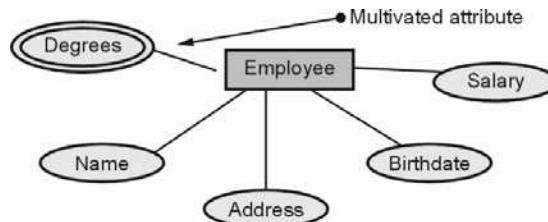
→ Address = {59 + ‘JM Road’ + ‘ShivajiNagar’}



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.

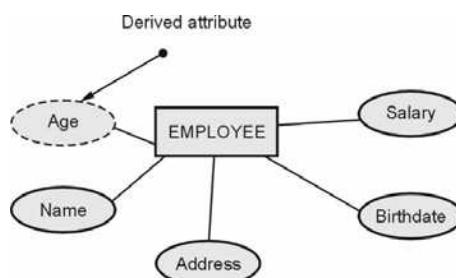


Fig. 2.1.4

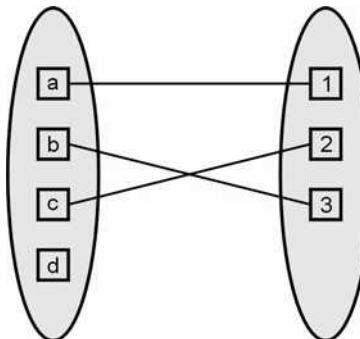
2.2 Mapping Cardinality

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

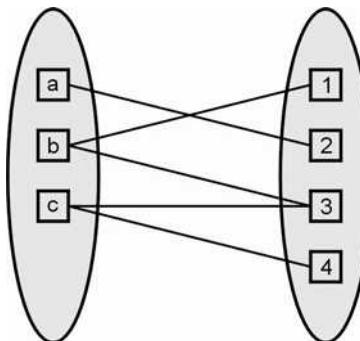
The mapping cardinalities are used in representing the binary relationship sets.

Various types of mapping cardinalities are -

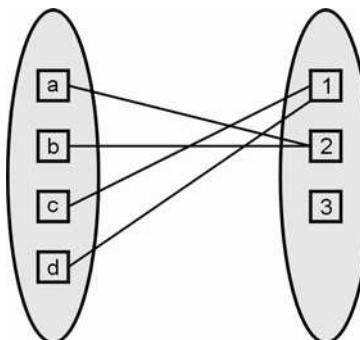
- 1) **One to One** : An entity A is associated with at least one entity on B and an entity B is associated with at one entity on A. This can be represented as



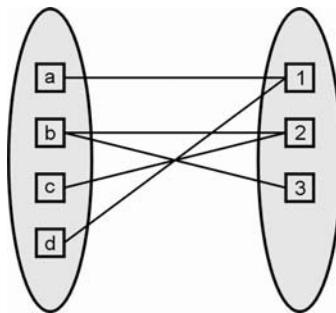
- 2) **One to Many** : An entity in A is associated with any number of entities in B. An entity in B, however, can be associated with at most one entity in A.



- 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.



2.3 ER Diagrams

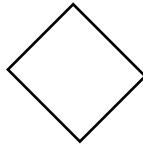
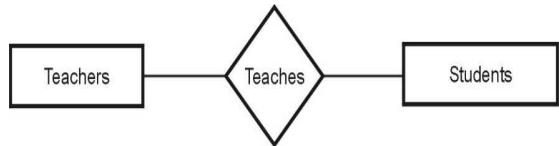
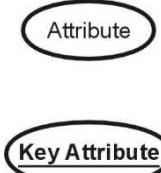
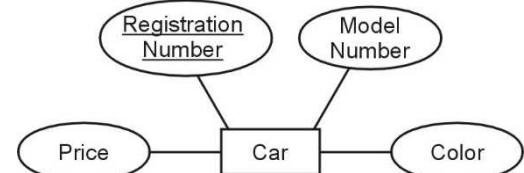
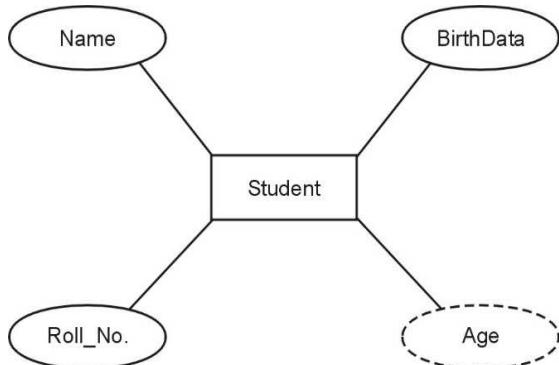
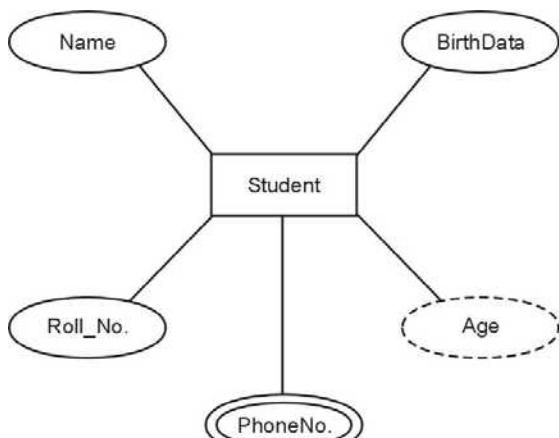
An E-R diagram can express the overall logical structure of a database graphically. E-R diagrams are used to model real-world objects like a person, a car, a company and the relation between these real-world objects.

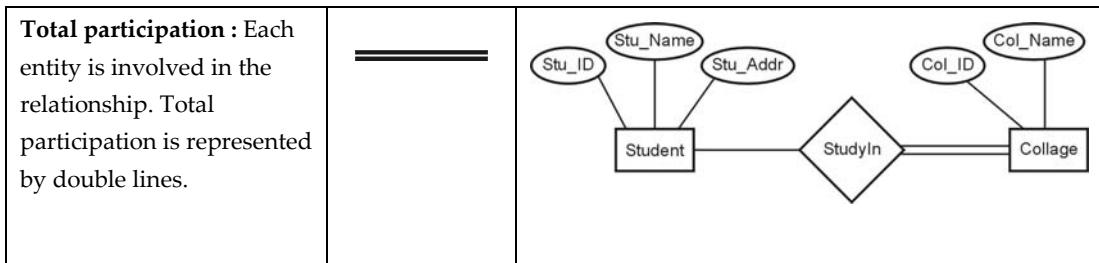
Features of ER model

- i) E-R diagrams are used to represent E-R model in a database, which makes them easy to be converted into relations (tables).
- ii) E-R diagrams provide the purpose of real-world modeling of objects which makes them intently useful.
- iii) E-R diagrams require no technical knowledge and no hardware support.
- iv) These diagrams are very easy to understand and easy to create even by a naive user.
- v) 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 product, a car, a person are the examples of an entity.		

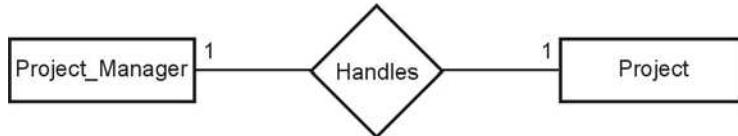
Relationship : Rhombus is used to setup relationships between two or more entities.		
Attribute : Each entity has a set of properties. These properties of each entity are termed as attributes. For example, a car entity would be described by attributes such as price, registration number, model number, color etc		
Derived attribute : Derived attributes are those which are derived based on other attributes, for example, age can be derived from date of birth. To represent a derived attribute, another dotted ellipse is created inside the main ellipse		
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.		



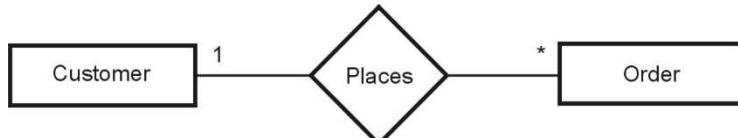
2.3.1 Mapping Cardinality Representation using ER Diagram

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

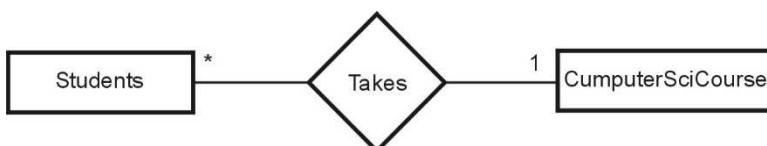
- i) **One to one relation :** When entity A is associated with at the most one entity B then it shares one to one relation. For example - There is one project manager who manages only one project.



- 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.



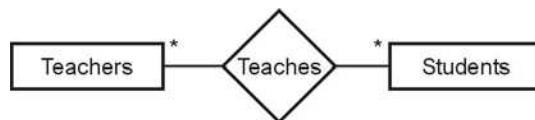
- ii) **Many to one :** When more than one entities are associated with only one entity then there is many to one relation. For example - Many student take a ComputerSciCourse.



Alternate representation can be



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

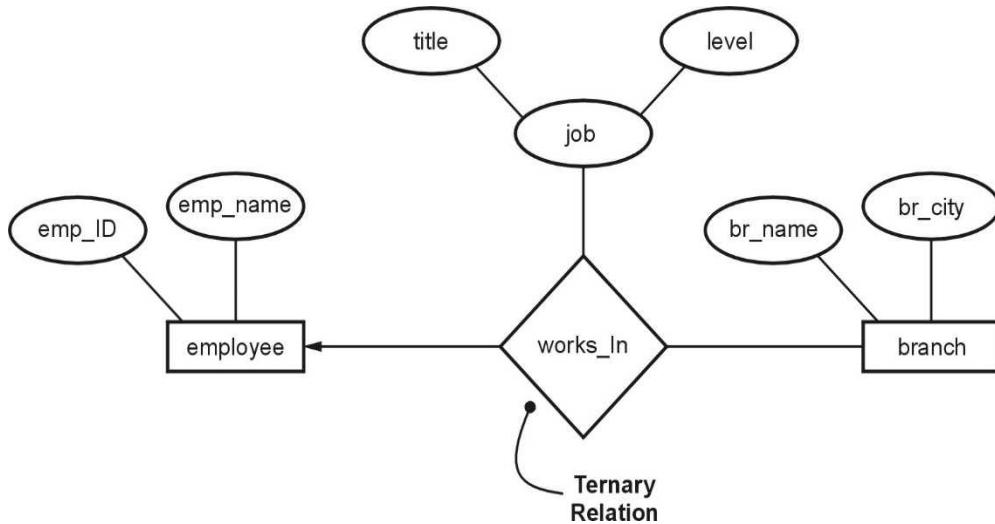


Alternate representation can be



2.3.2 Ternary Relationship

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



2.3.3 Binary and Ternary Relationships

- Although binary relationships seem natural to most of us, in reality it is sometimes necessary to connect three or more entities. If a relationship connects three entities, it is called ternary or "3-ary."
- Ternary relationships are required when binary relationships are not sufficient to accurately describe the semantics of an association among three entities.
- For example - Suppose, you have a database for a company that contains the entities, PRODUCT, SUPPLIER, and CUSTOMER. The usual relationships might be PRODUCT/ SUPPLIER where the company buys products from a supplier - a normal binary relationship. The intersection attribute for PRODUCT/SUPPLIER is wholesale_price

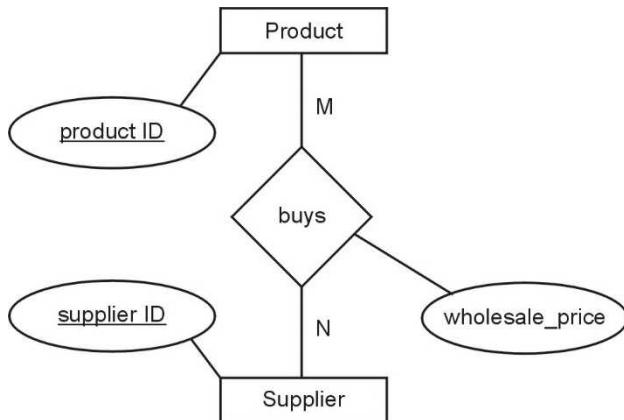


Fig. 2.3.1 : A binary relationship of PRODUCT and SUPPLIER and an intersection attribute, wholesale_price

- Now consider the CUSTOMER entity, and that the customer buys products. If all customers pay the same price for a product, regardless of supplier, then you have a simple binary relationship between CUSTOMER and PRODUCT. For the CUSTOMER/ PRODUCT relationship, the intersection attribute is retail_price.

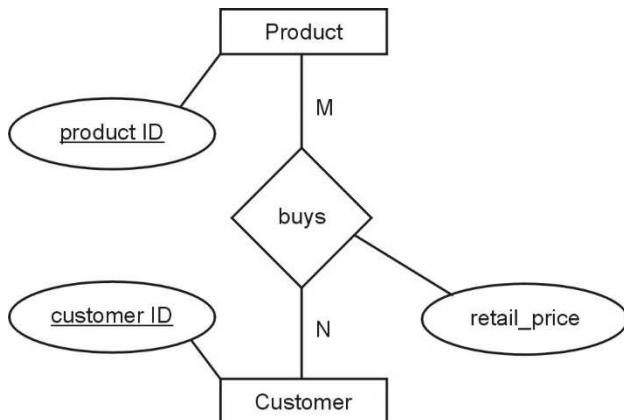


Fig. 2.3.2 : A binary relationship of PRODUCT and CUSTOMER and an Intersection attribute, retail_price

- Single ternary relation :** Now consider a different scenario. Suppose the customer buys products but the price depends not only on the product, but also on the supplier. Suppose you needed a customerID, a productID, and a supplierID to identify a price. Now you have an attribute that depends on three things and hence you have a relationship between three entities (a ternary relationship) that will have the intersection attribute, price.

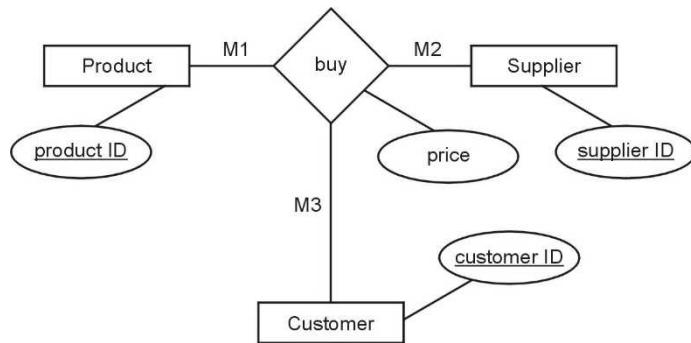


Fig. 2.3.3 : Ternary relation

2.3.4 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.

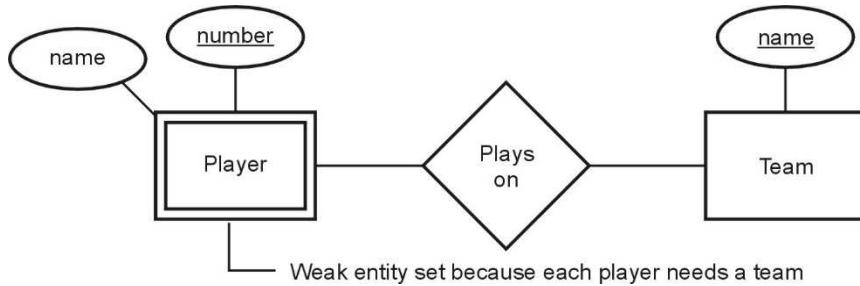


Fig. 2.3.4 : Weak entity set

- 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.

2.4 Enhanced ER Model

2.4.1 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. 2.4.1.

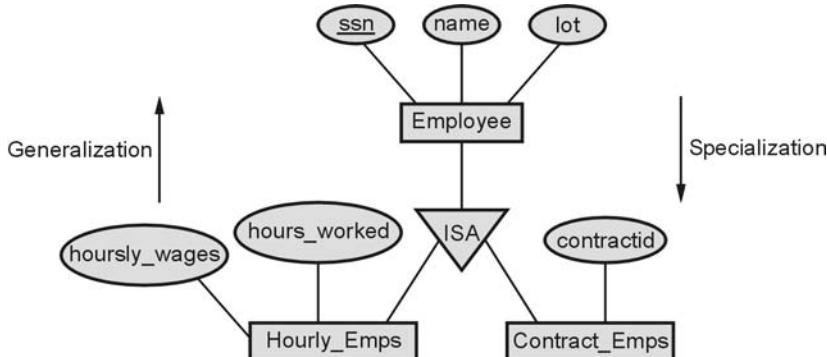
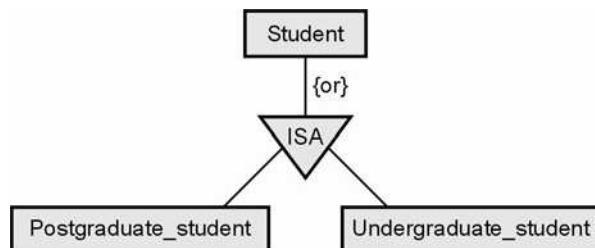


Fig. 2.4.1

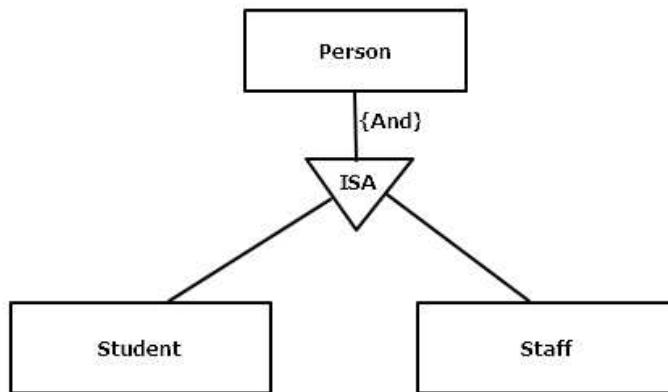
2.4.2 Constraints on Specialization/Generalization

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

- 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 -
 - 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**.
 - User defined :** This is kind of entity set that in which the membership is manually defined.
- 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**

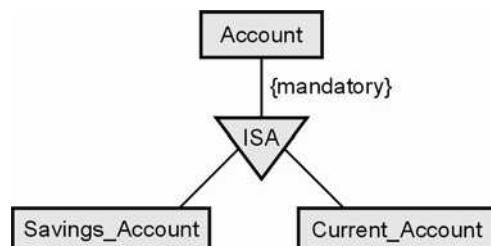


- 3) **Overlapping** : When some entity can be a member of more than one subclasses. For example - Person can be both a Student or a Staff. The **And** can be used to represent this constraint.

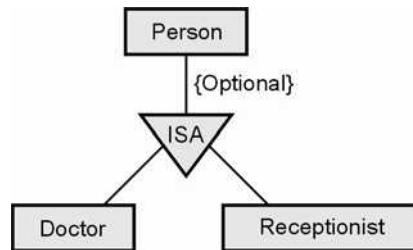


- 4) **Completeness** : It specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within the generalization/specialization. This constraint may be one of the following -

- 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.



2.4.3 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.

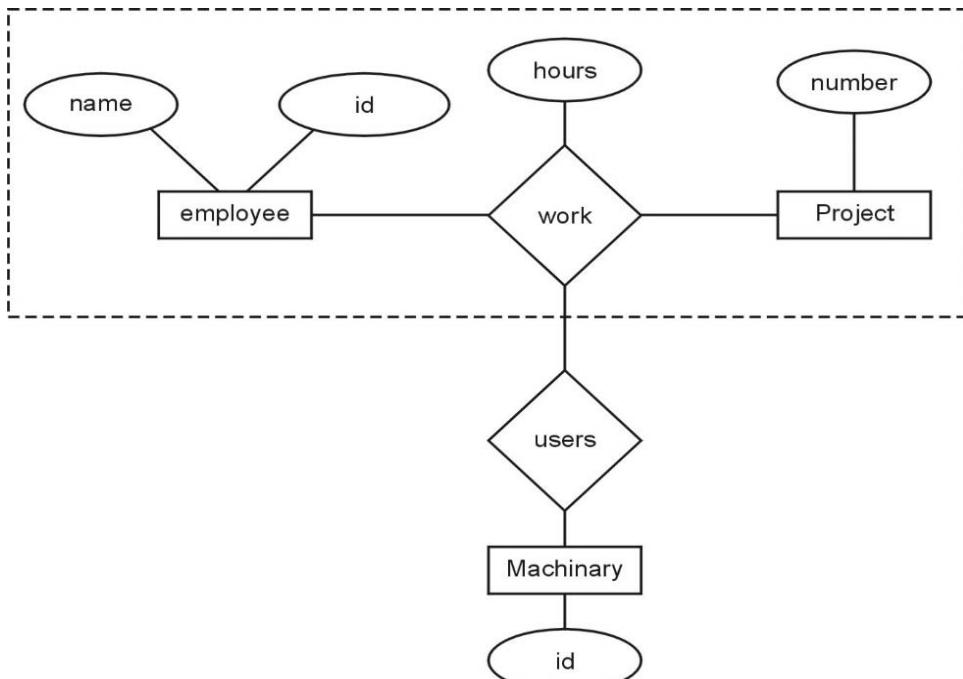


Fig. 2.4.2 : ER model with aggregation

2.5 Examples based on ER Diagram

Example 2.5.1 Draw the ER diagram for banking systems (home loan applications).

AU : Dec.-17, Marks 8

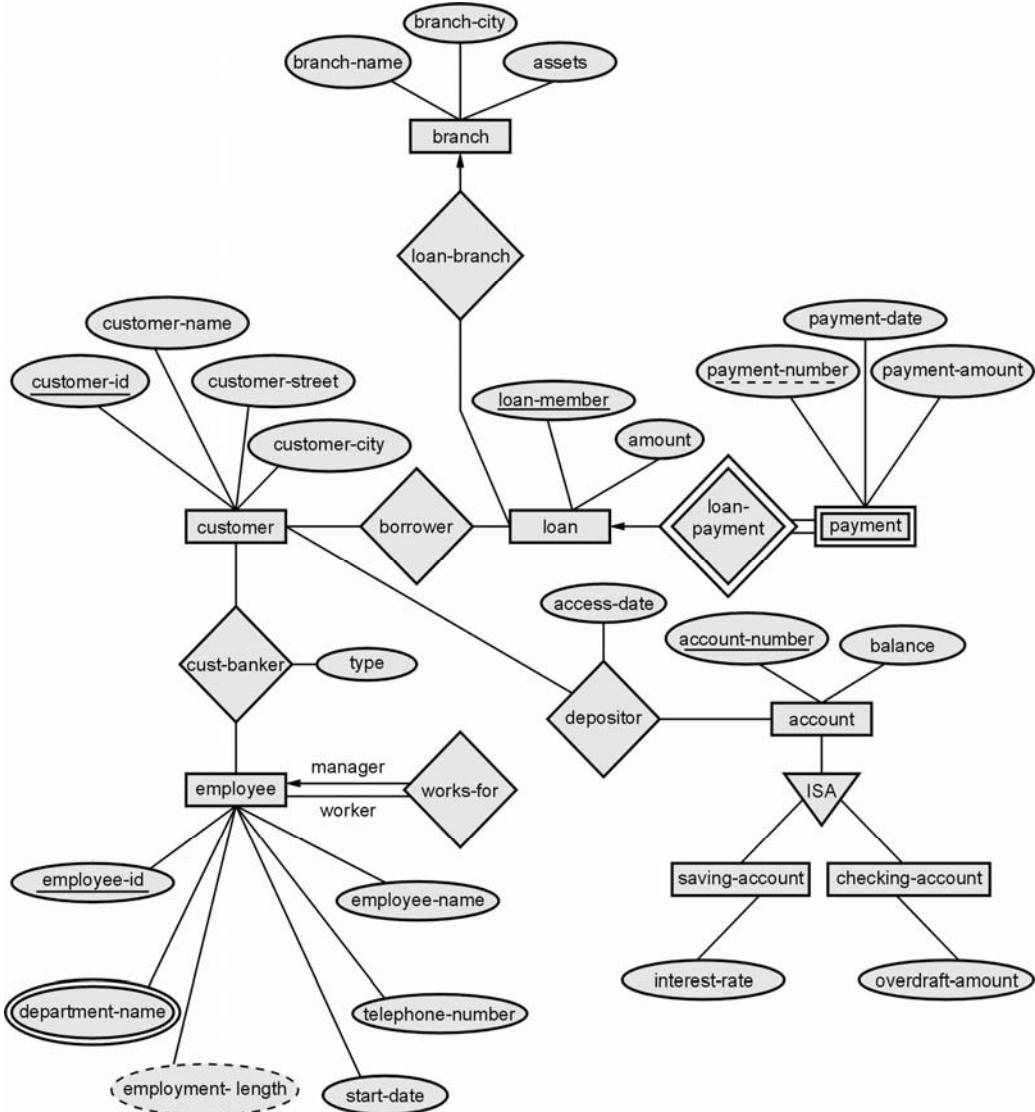
OR Draw an ER diagram corresponding to customers and loans.

AU : May.-14, Marks 8

OR Write short notes on : E-R diagram for banking system .

AU : Dec.-14, Marks 8

Solution :

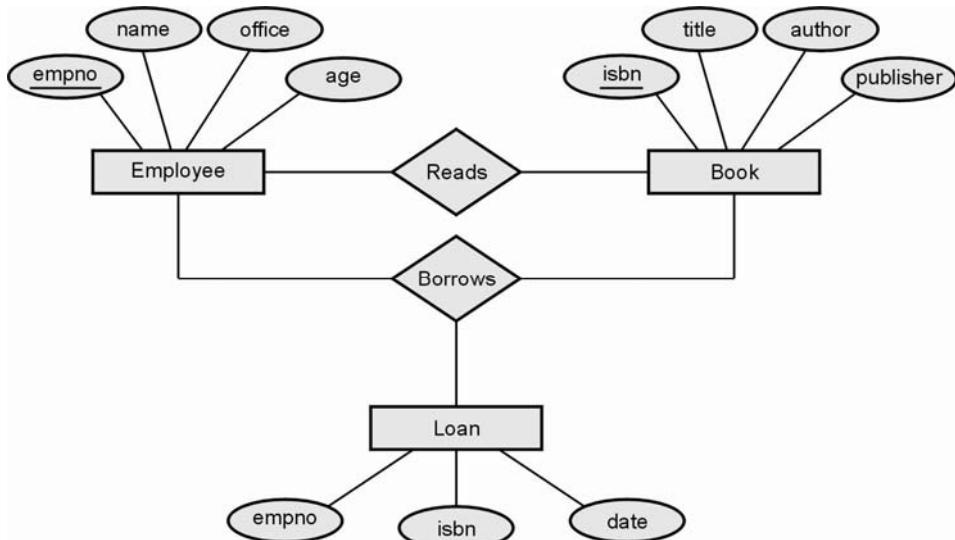


Example 2.5.2 Consider the relation schema given in Figure. Design and draw an ER diagram that capture the information of this schema.

AU : May-17, Marks 5

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

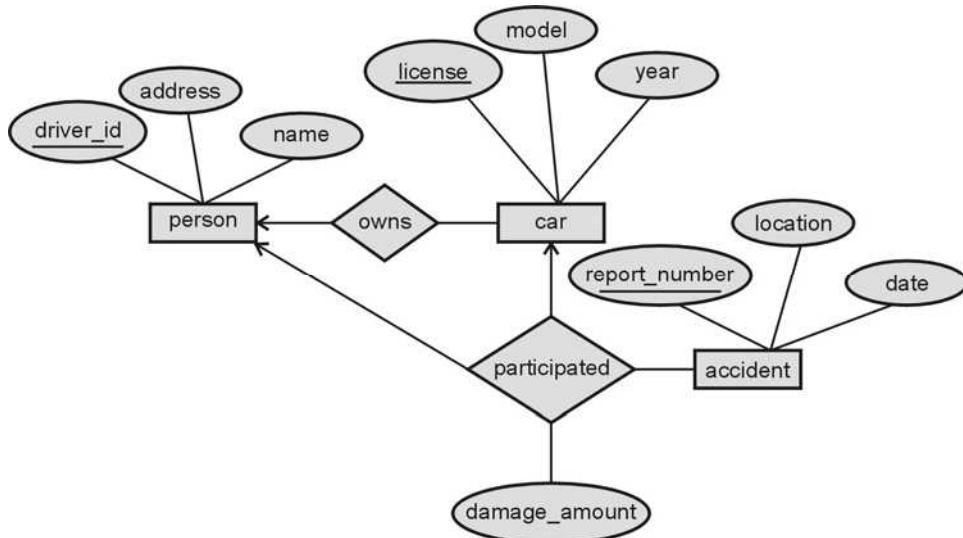
Solution :



Example 2.5.3 Construct an E-R diagram for a car insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents. Each insurance policy covers one or more cars and has one or more premium payments associated with it. Each payment is for particular period of time and has an associated due date and date when the payment was received.

AU : Dec.-16, Marks 7

Solution :



Example 2.5.4 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

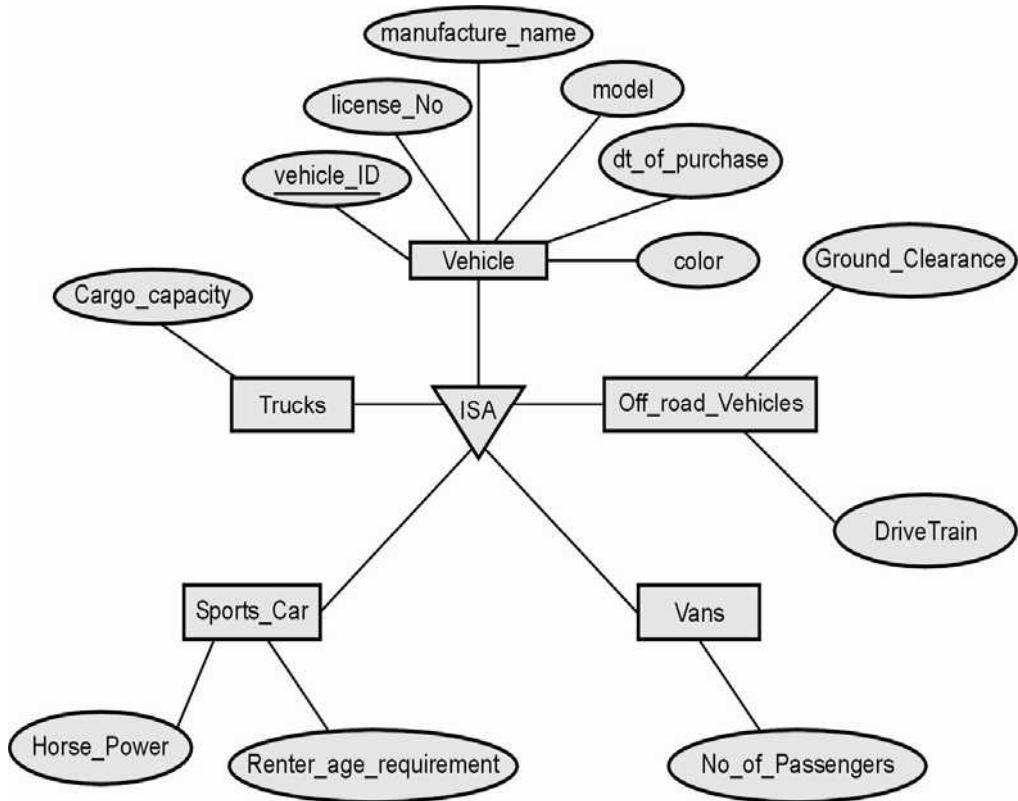
Vans : number of passengers

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

Construct an ER model for the car rental company database.

AU : Dec.-15, Marks 16

Solution :

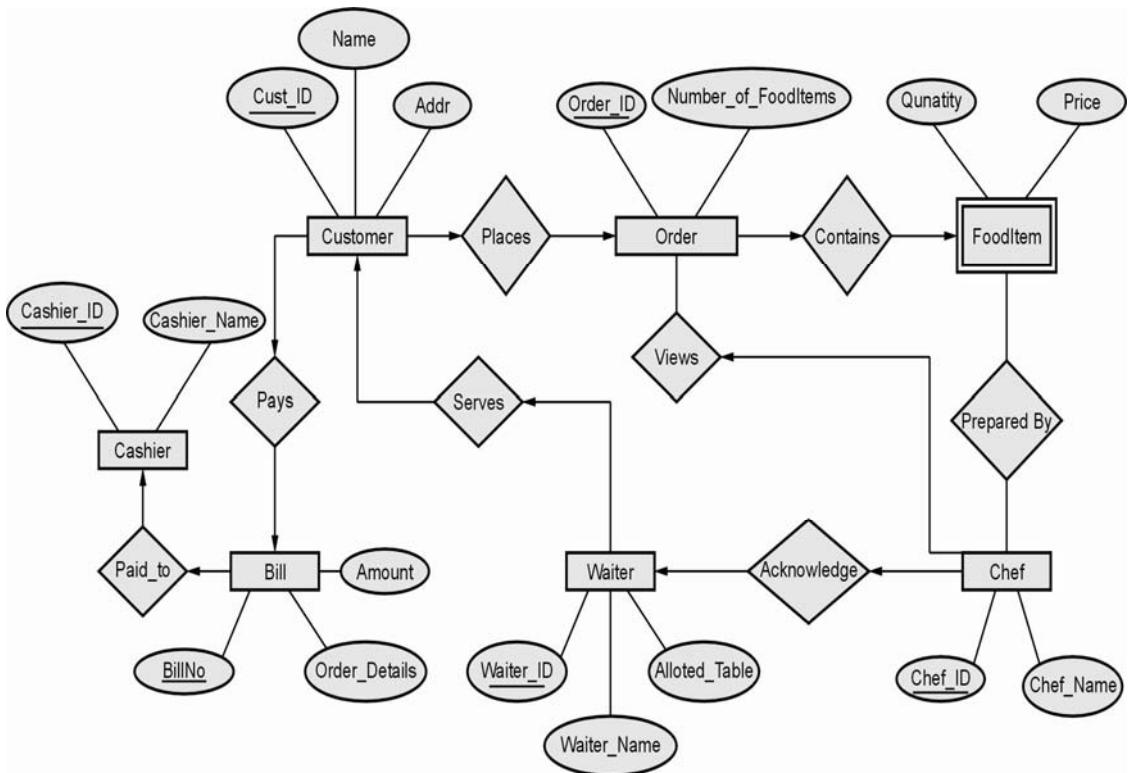


Example 2.5.5 Draw E-R diagram for the "Restaurant Menu Ordering System", which will facilitate the food items ordering and services within a restaurant. The entire restaurant scenario is detailed as follows. The customer is able to view the food items menu, call the waiter, place orders and obtain the 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.

AU : May-15, Marks 16

Solution :



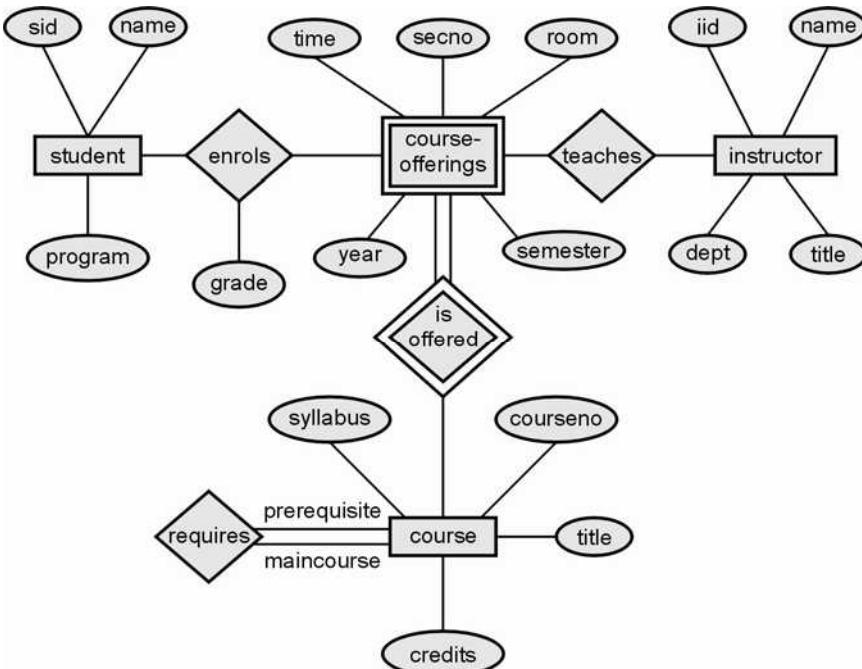
Example 2.5.6 A university registrar's office maintains data about the following entities :

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

AU : Dec.-13, Marks 10

Solution :

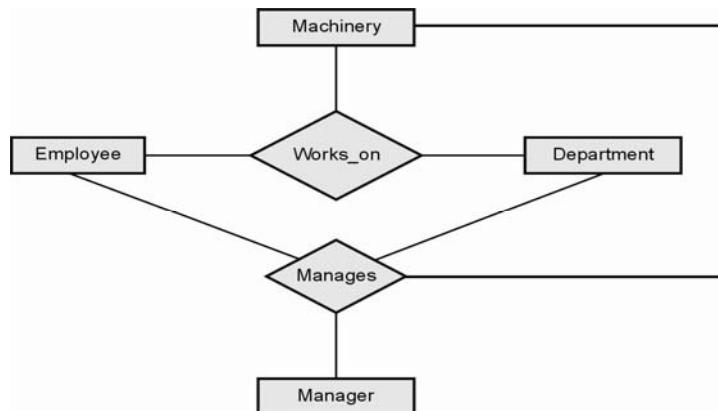


Example 2.5.7 What is aggregation in ER model ? Develop an ER diagram using aggregation that captures following information : Employees work for projects. An employee working for particular project uses various machinery. Assume necessary attributes. State any assumptions you make. Also discuss about the ER diagram you have designed.

AU : Dec.-11, Marks 8

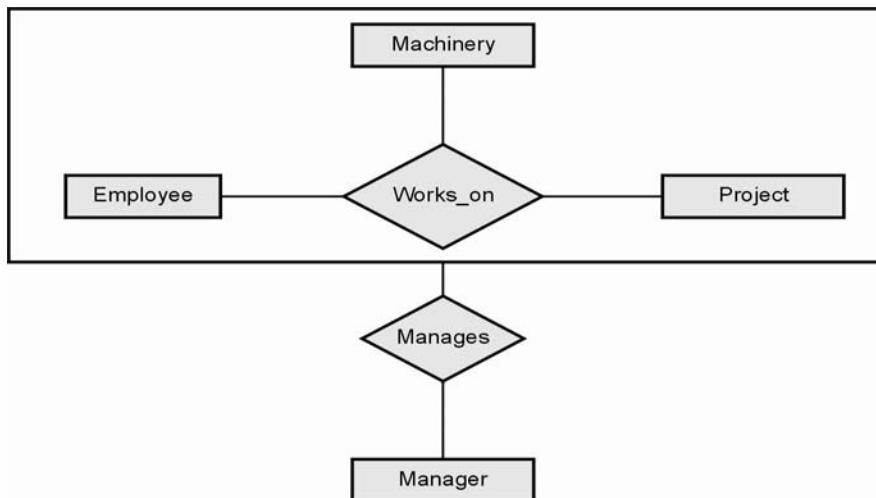
Solution : Aggregation : Refer section 2.4.3.

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 -

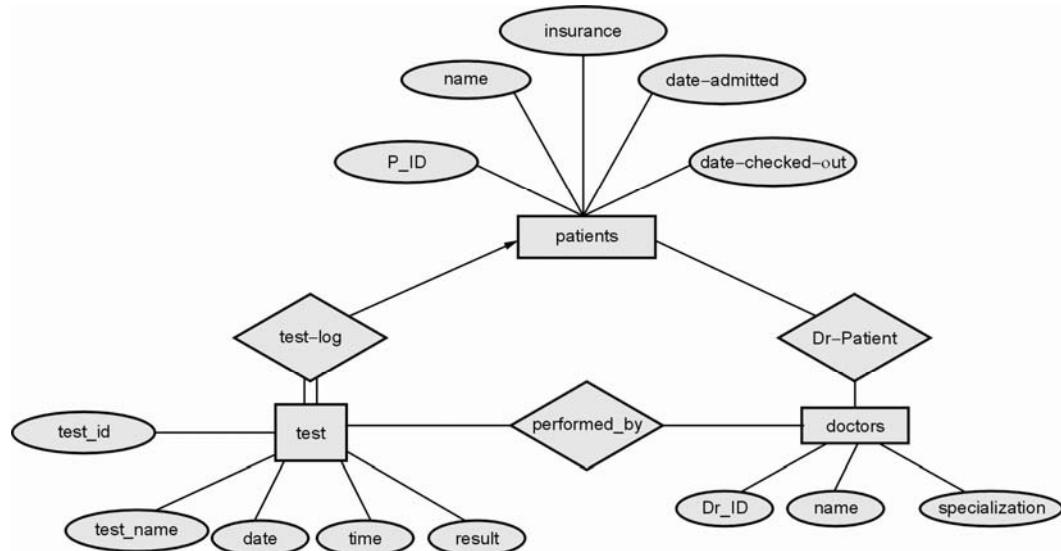


We can then create a binary relationship **manages** for between **Manager** and (**Employee, Project, Machinery**).

Example 2.5.8 Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examinations conducted.

AU : Dec.-07, Marks 8

Solution :



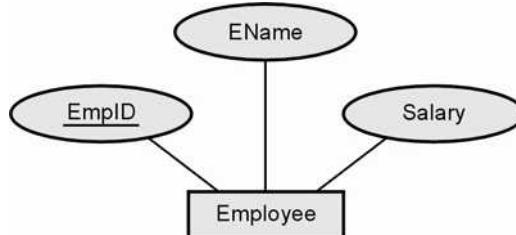
2.6 ER to Relational Mapping

AU : May-17, Marks 13

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

2.6.1 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 -

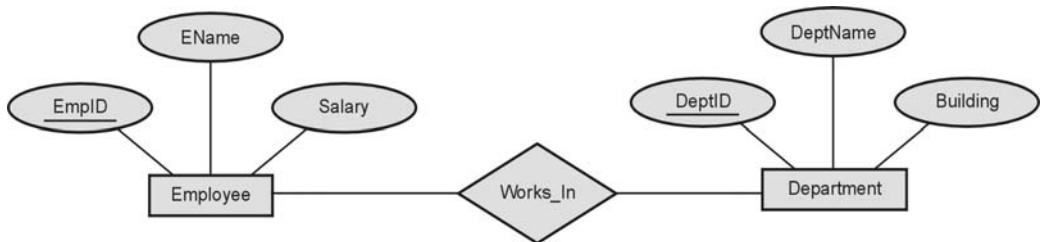
```

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

2.6.2 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

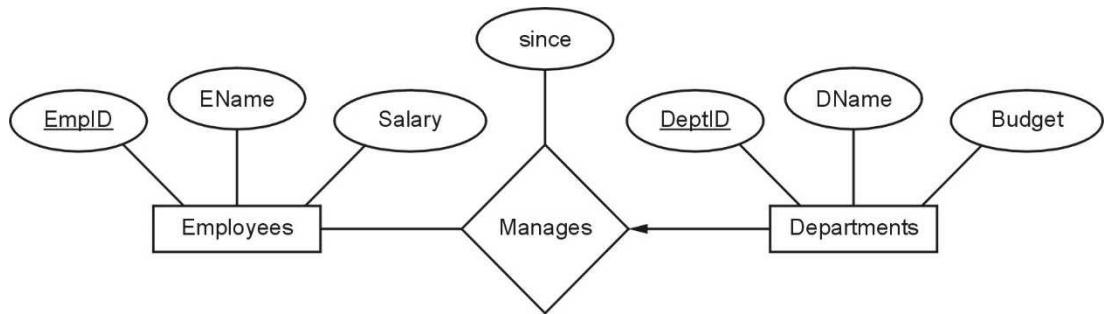


The SQL statement captures the information for relationship present in above ER diagram as follows -

```
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
)
```

2.6.3 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 Managers**.



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 :

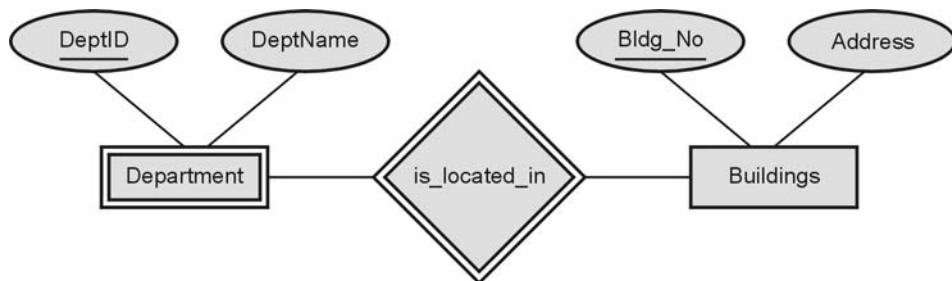
```
CREATE TABLE Dep_Mgr ( DeptID INTEGER,
                      DName CHAR(20),
                      Budget REAL,
                      EmpID CHAR (11),
                      since DATE,
                      PRIMARY KEY (DeptID),
                      FOREIGN KEY (EmpID) REFERENCES Employees)
```

2.6.4 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 Weka 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

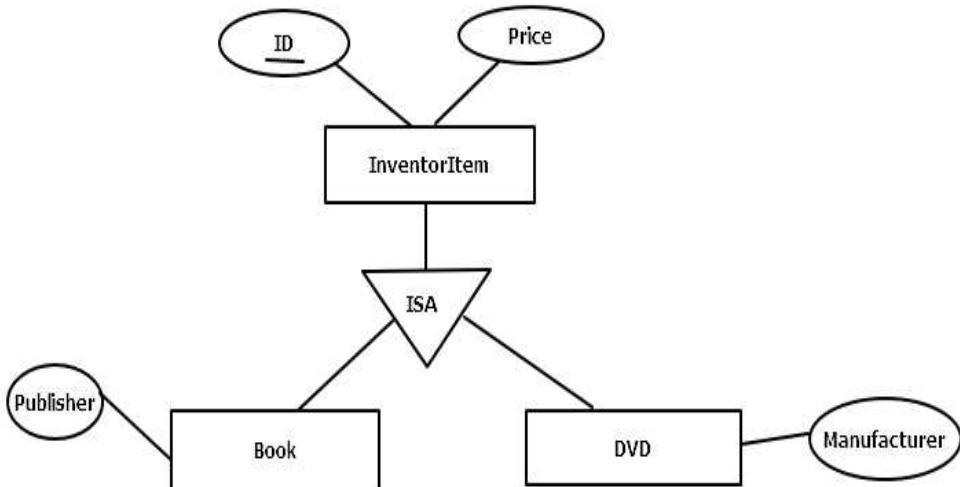


Following SQL Statement illustrates this mapping

```
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
                     )
```

2.6.5 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 , name)
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,name,Publisher)
DVD(ID, name,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 , name,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.

Example 2.6.1 Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examinations conducted. Also construct appropriate tables for the ER diagram you have drawn.

Solution :

ER Diagram - Refer example 2.5.8.

Relational Mapping

```

patients (P_id, name, insurance, date-admitted, date-checked-out)
doctors (Dr_id, name, specialization)
test (testid, testname, date, time, result)
doctor-patient (P_id, Dr_id)
test-log (testid, P_id) performed-by (testid, Dr_id)

```

University Question

1. Discuss the correspondence between the ER model construct and the relational model constructs. Show how each ER model construct can be mapped to the relational model. Discuss the option for mapping EER construct.

AU : May-17, Marks 13

Part II Relational Database Design

2.7 Concept of Relational Database Design

- 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.
- For achieving these goals, the database design needs to be **normalized**. That means we have to check whether the schema is in normal form or not.
- For checking the normal form of the schema, it is necessary to check the functional dependencies and other data dependencies that exist within the schema.

Hence before letting us know what the normalization means, it is necessary to understand the concept of functional dependencies.

2.8 Functional Dependencies

Definition : Let P and Q be sets of columns, then: P functionally determines Q, written $P \rightarrow Q$ if and only if any two rows that are equal on (all the attributes in) P must be equal on (all the attributes in) Q.

In other words, the functional dependency holds if

$$T1.P = T2.P, \text{ then } T1.Q = T2.Q$$

Where notation $T_1.P$ projects the tuple T_1 onto the attribute in P .

For 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. 2.8.1 : Table which holds functional dependency i.e. $R \rightarrow B$

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. 2.8.2 : 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.

2.8.1 Computing Closure Set of Functional Dependency

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
- iv) **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 $X \rightarrow Z$

Example 2.8.1 Compute the closure of the following set of functional dependencies for a relation scheme $R(A,B,C,D,E)$, $F=\{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A\}$

Solution : Consider F as follows

$A \rightarrow BC$

$CD \rightarrow E$

$B \rightarrow D$

$E \rightarrow A$

The closure can be written for each attribute of relation as follows

- $(A)^+ =$
 - Step 1** : $\{A\}$ -> the attribute itself
 - Step 2** : $\{ABC\}$ as $A \rightarrow BC$
 - Step 3** : $\{ABCD\}$ as $B \rightarrow D$
 - Step 4** : $\{ABCDE\}$ as $CD \rightarrow E$
 - Step 5** : $\{ABCDE\}$ as $E \rightarrow A$ and A is already present

Hence $(A)^+ = \{ABCDE\}$
- $(B)^+ =$
 - Step 1** : $\{B\}$
 - Step 2** : $\{BD\}$ as $B \rightarrow D$
 - Step 3** : $\{BD\}$ as there is no BD pair on LHS of F

Hence $(B)^+ = \{BD\}$
- $(C)^+ =$
 - Step 1** : $\{C\}$
 - Step 2** : $\{C\}$ as there is no single C on LHS of F

Hence $(C)^+ = \{C\}$
- $(D)^+ =$
 - Step 1** : $\{D\}$
 - Step 3** : $\{D\}$ as there is no BD pair on LHS of F

Hence $(D)^+ = \{D\}$

- $(E)^+ =$ **Step 1 : {E}**

Step 2 : {EA} as $E \rightarrow A$

Step 3 : {EABC} as $A \rightarrow BC$

Step 4 : {EABCD} as $B \rightarrow D$

Step 5 : {EABCD} as $CD \rightarrow E$ and E is already present

By rearranging we get $\{ABCDE\}$

Hence $(E)^+ = \{ABCDE\}$

- $(CD)^+ =$ **Step 1:{CD}**

Step 2 :{CDE}

Step 3 :{CDEA}

Step 4 :{CDEAB}

By rearranging we get $\{ABCDE\}$

Hence $(CD)^+ = \{ABCDE\}$

Example 2.8.2 Compute the closure of the following set of functional dependencies for a relation scheme $R(A,B,C,D,E)$, $F=\{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A\}$ and Find the candidate key.

Solution : For finding the closure of functional dependencies - Refer example 2.8.1.

We can identify candidate from the given relation schema with the help of functional dependency. For that purpose we need to compute the closure set of attribute. Now we will find out the closure set which can completely identify the relation $R(A,B,C,D)$.

Let, $(A)^+ = \{ABCDE\}$

$$(B)^+ = \{BD\}$$

$$(C)^+ = \{C\}$$

$$(D)^+ = \{D\}$$

$$(E)^+ = \{ABCDE\}$$

$$(CD)^+ = \{ABCDE\}$$

Clearly, only $(A)^+, (E)^+$ and $(CD)^+$ gives us $\{ABCD\}$ i.e. complete relation R . Hence these are the candidate keys.

2.8.2 Canonical Cover or 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 2.8.3 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->D

B->E

Step 2 : Find the **redundant entries** and **delete** them. This can be done as follows -

- **For A->C :** We find $(A)^+$ by assuming that we delete A->C temporarily. We get $(A)^+ = \{A\}$. Thus from A it is not possible to obtain C by deleting A->C. This means we can not delete A->C
- **For AC->D :** We find $(AC)^+$ by assuming that we delete AC->D temporarily. We get $(AC)^+ = \{AC\}$. Thus by such deletion it is not possible to obtain D. This means we can not delete AC->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->AE

This is a **minimal cover** or **Canonical cover** of functional dependencies.

2.9 Concept of Redundancy and Anomalies

Definition : 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.

EmplID	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.

2.10 Decomposition

AU : Dec.-17, Marks 7

- 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

- Hence, the above table can be decomposed into two Schema S and T as follows :

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 :

- Some queries become more **expensive**.
- 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.

2.10.1 Non-loss Decomposition or Loss-less Join

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)

$$\text{Att}(R1) \cap \text{Att}(R2) \rightarrow \text{Att}(R1)$$

or $\text{Att}(R1) \cap \text{Att}(R2) \rightarrow \text{Att}(R2)$

Example 2.10.1 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 $(A)^+ = \{A,B,C\} \in$ attributes of R1. Thus the third condition gets satisfied.

This shows that the given decomposition is a **lossless join**.

Example 2.10.2 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 2.10.3 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 $R_1 \bowtie R_2$ 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 $R_1 \bowtie R_2 \not\subseteq R$. Hence it is not lossless decomposition.

2.10.2 Dependency Preservation

- **Definition :** A Decomposition $D = \{R_1, R_2, R_3, \dots, R_n\}$ of R is dependency preserving for a set F of Functional dependency if $(F_1 \cup F_2 \cup \dots \cup F_m) = F$.
- If decomposition is not dependency-preserving, some dependency is lost in the decomposition.

Example 2.10.4 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 $R_1 = (A, C)$ and $R_2 = (B, C)$. 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

$$F^+ = (F_1 \cup F_2)^+$$

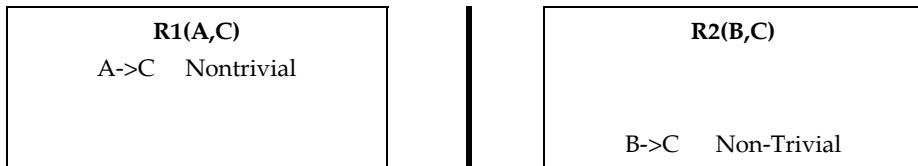
Step 2 : We have with us the $F^+ = \{A \rightarrow B \text{ and } B \rightarrow C\}$

Step 3 : Let us find $(F_1)^+$ for relation R_1 and $(F_2)^+$ for relation R_2

R1(A,C)	
A->A	Trivial
C->C	Trivial
A->C	~ In $(F)^+ A \rightarrow B \rightarrow C$ and it is Nontrivial
	AC->AC Trivial
A->B	but is not useful as B is not part of R1 set
	We can not obtain C->A

R2(B,C)	
B->B	Trivial
C->C	Trivial
B->C	~ In $(F)^+ B \rightarrow C$ and it is Non-Trivial
	BC->BC Trivial
	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



$$(F_1 \cup F_2)^+ = \{A \rightarrow C, B \rightarrow C\} \neq \{A \rightarrow B, B \rightarrow C\} \text{ i.e. } (F)^+$$

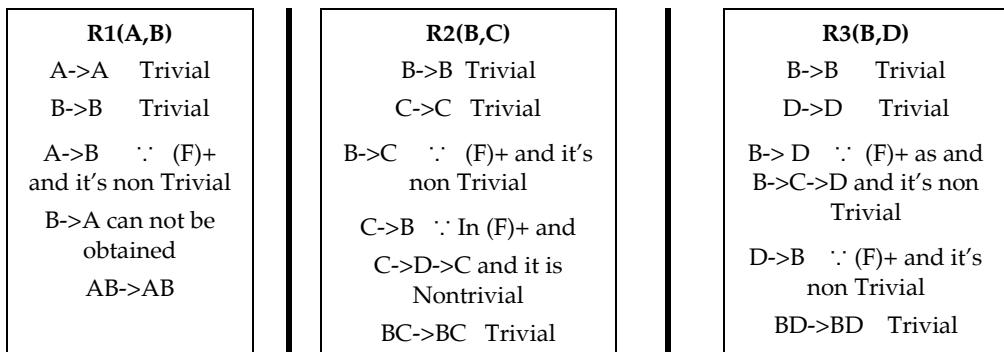
Thus the condition specified in step 1 i.e. $F^+ = (F_1 \cup F_2)^+$ is **not true**. Hence it is **not dependency preserving decomposition**.

Example 2.10.5 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.

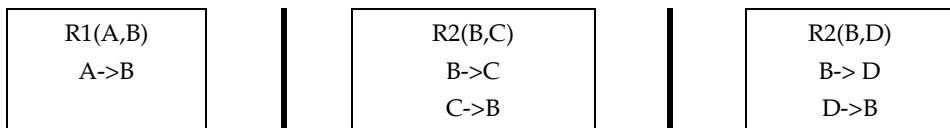
Solution :

Step 1 : Let $(F)^+ = \{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow B\}$.

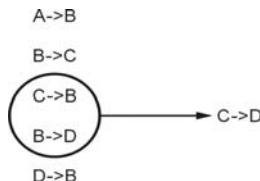
Step 2 : We will find $(F_1)^+, (F_2)^+, (F_3)^+$ for relations $R1(A,B)$, $R2(B,C)$ and $R3(B,D)$ as follows -



Step 3 : We will eliminate all the trivial relations and useless relations. Hence we can obtain $R1 \cup R2 \cup R3$ as



Step 4 : As from above FD's we get



Step 5 : This proves that $F^+ = (F_1 \cup F_2 \cup F_3)^+$. Hence given **decomposition is dependency preserving**.

University Question

1. Differentiate between lossless join decomposition and dependency preserving decomposition.

AU : Dec.-17, Marks 7

2.11 Normal Forms

AU : Dec.-14, 15, May-18, Marks 16

- 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)
- The normalization is important because it allows database to take up **less disk space**.
- It also help in increasing the **performance**.

2.11.1 First Normal Form

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

2.11.2 Second Normal Form

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->CD,A->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->C, the non prime attribute C is dependent upon A which is actually a part of candidate key AB. Hence due to A->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 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

2.11.3 Third Normal Form

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

- $\{\text{RegID}\}$
- $\{\text{RegID}, \text{RollNo}\}$
- $\{\text{RegID}, \text{Sname}\}$
- $\{\text{RollNo}, \text{Sname}\}$
- $\{\text{RegID}, \text{RollNo}, \text{Sname}\}$

Candidate Keys

- $\{\text{RegID}\}$
- $\{\text{RollNo}\}$

Third Normal Form

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 :

- i) X is a super key of table
- ii) 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	Rajasthan
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 \rightarrow sname
 sid \rightarrow zipcode
 zipcode \rightarrow cityname
 cityname \rightarrow 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 2.11.1 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,

$$A \rightarrow DE \text{ (given)}$$

$$\therefore A \rightarrow D, A \rightarrow E$$

$$\text{As } D \rightarrow IJ, A \rightarrow IJ$$

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

$$AB \rightarrow ABDEIJ$$

$$\text{But } AB \rightarrow C \text{ (given)}$$

$$\therefore AB \rightarrow ABCDEIJ$$

$$B \rightarrow F \text{ (given)} \quad F \rightarrow GH \quad \therefore B \rightarrow GH \text{ (transitivity)}$$

$$\therefore AB \rightarrow AGH \text{ is also true}$$

$$\text{Similarly } AB \rightarrow AF \quad \therefore B \rightarrow F \text{ (given)}$$

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

$$R_1 = (\underline{A}, \underline{B}, C)$$

$$R_2 = (\underline{A}, D, E)$$

$$R_3 = (\underline{D}, I, J)$$

$$R_4 = (\underline{B}, E)$$

$$R_5 = (E, G, H).$$

University Questions

1. What is database normalization ? Explain the first normal form, second normal form and third normal form. AU : May-18, Marks 13; Dec.-15, Marks 16
2. What are normal forms. Explain the types of normal form with an example. AU : Dec.-14, Marks 16

2.12 Boyce / Codd Normal Form (BCNF)

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) \rightarrow Teacher
 - Teacher \rightarrow course
- The above table is not in BCNF because of the dependency **teacher \rightarrow 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 2.12.1 Consider a relation(A,B,C,D) having following FDs. $\{AB \rightarrow C, AB \rightarrow D, C \rightarrow A, B \rightarrow D\}$. 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

prime attributes = {A,B,C}

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 $X \rightarrow Y$ the X should be candidate key or super key.

From above FDs consider $C \rightarrow D$ 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 $X \rightarrow Y$ 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 $AB \rightarrow C$ or $AB \rightarrow D$ the AB is a candidate key. Condition for 3NF is satisfied.
- o Consider $C \rightarrow A$. In this FD the C is not candidate key but A is a prime attribute. Condition for 3NF is satisfied.
- o 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.

- o For $AB \rightarrow C$ or $AB \rightarrow D$ the AB is a **complete candidate key**. Condition for 2NF is satisfied.

- Consider C->A. In this FD the C is not candidate key. Condition for 2NF is satisfied.
- Now consider B->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 2.12.2 Consider a relation R(ABC) with following FD A->B, B->C and C->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->Y the X should be candidate key or super key.

From above FDs

- Consider A->B in which A is a candidate key or super key. Condition for BCNF is satisfied.
- Consider B->C in which B is a candidate key or super key. Condition for BCNF is satisfied.
- Consider C->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 2.12.3 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->B, B->A.

From the above FDs

- Consider A->B in which A is a candidate key or super key. Condition for BCNF is satisfied.

- o Consider $B \rightarrow A$ in which B is a candidate key or super key. Condition for BCNF is satisfied.
- o 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.
- o 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.

2.13 Multivalued Dependencies and Fourth Normal Form

AU : May-14, Dec.-16, Marks 16

Concept of Multivalued Dependencies

- A table is said to have multi-valued dependency, if the following conditions are true,
 - 1) For a dependency $A \rightarrow B$, if for a single value of A, **multiple values** of B exists, then the table may have multi-values dependency.
 - 2) Also, a table should have **at-least 3 columns** for it to have a multi-valued dependency.
 - 3) And, for a relation R(A,B,C), if there is a multi-valued **dependency between**, A and B, then B and C should be independent of each other.

If all these conditions are true for any relation(table), it is said to have multi-valued dependency.

- In simple terms, if there are two columns A and B - and for column A if there are multiple values of column B then we say that MVD exists between A and B
- The multivalued dependency is denoted by \twoheadrightarrow
- If there exists a multivalued dependency then the table is **not in 4th normal form**.
- **For example :** Consider following table for information about student

Student

sid	Course	Skill
1	C C++	English German
2	Java	English French

Here sid =1 leads to multiple values for courses and skill. Following table shows this

sid	Course	Skill
1	C	English
1	C++	German
1	C	German
1	C++	English
2	Java	English
2	Java	French

Here **sid** and **course** are dependent but the **Course** and **Skill** are independent. The multivalued dependency is denoted as :

$\text{sid} \twoheadrightarrow \text{Course}$

$\text{sid} \twoheadrightarrow \text{Skill}$

Fourth Normal Form

Definition : For a table to satisfy the Fourth Normal Form, it should satisfy the following two conditions :

- 1) It should be in the Boyce-Codd Normal Form(BCNF).
- 2) And, the table should not have any multi-valued dependency.

For example : Consider following student relation which is not in 4NF as it contains multivalued dependency.

Student Table

sid	Course	Skill
1	C	English
1	C++	German
1	C	German
1	C++	English
2	Java	English
2	Java	French

Now to convert the above table to 4NF we must decompose the table into following two tables.

Student_Course Table**Key :** (sid,Course)

sid	Course
1	C
1	C++
2	Java

Student_Skill Table**Key :** (sid,Skill)

sid	Skill
1	English
1	German
2	English
2	French

Thus the tables are now in 4NF.

University Questions

1. Explain first normal form, second normal form, third normal form and BCNF with example.

AU : Dec.-16, Marks 13

2. Explain Boyce Codd Normal form and fourth normal form with suitable example.

AU : May-14, Marks 16**2.14 Join Dependencies and Fifth Normal Form****Concept of Join Dependencies**

- Join decomposition is a further generalization of Multivalued dependencies.
- If the join of R1 and R2 over C is equal to relation R, then we can say that a Join Dependency (JD) exists.
- Where R1 and R2 are the decompositions R1(A, B, C) and R2(C, D) of a given relations R (A, B, C, D).
- Alternatively, R1 and R2 are a lossless decomposition of R.
- A JD $\bowtie \{R_1, R_2, \dots, R_n\}$ is said to hold over a relation R if R1, R2, ..., Rn is a lossless-join decomposition.

- o The $*(A, B, C, D), (C, D)$ will be a JD of R if the join of join's attribute is equal to the relation R.
- o Here, $*(R_1, R_2, R_3)$ is used to indicate that relation R1, R2, R3 and so on are a JD of R.

Concept of Fifth Normal Form

The database is said to be in 5NF if -

- i) It is in 4th Normal Form
- ii) If we can decompose table further to eliminate redundancy and anomalies and when we rejoin the table we should not be losing the original data or get a new record(**join Dependency Principle**)

The fifth normal form is also called as **project join normal form**

For example - Consider following table

Seller	Company	Product
Rupali	Godrej	Cinthol
Sharda	Dabur	Honey
Sharda	Dabur	HairOil
Sharda	Dabur	Rosewater
Sunil	Amul	Icecream
Sunil	Britania	Biscuits

Here we assume the keys as{Seller, Company, Product}

The above table has multivalued dependency as

$\text{Seller} \twoheadrightarrow \{\text{Company}, \text{Product}\}$. Hence table is not in 4th Normal Form. To make the above table in 4th normal form we decompose above table into two tables as

Seller_Company

Seller	Company
Rupali	Godrej
Sharda	Dabur
Sunil	Amul
Sunil	Britania

Seller_Product

Seller	Product
Rupali	Cinthol
Sharda	Honey
Sharda	HairOil
Sharda	RoseWater
Sunil	Icecream
Sunil	Biscuits

The above table is in 4th Normal Form as there is no multivalued dependency. But it is not in 5th normal form because if we join the above two table we may get

Seller	Company	Product
Rupali	Godrej	Cinthol
Sharda	Dabur	Honey
Sharda	Dabur	HairOil
Sharda	Dabur	Rosewater
Sunil	Amul	Icecream
Sunil	Amul	Biscuits
Sunil	Britania	Icecream
Sunil	Britania	Biscuits
Newly added records which are not present in original table		

To avoid the above problem we can decompose the tables into three tables as Seller_Company, Seller_Product, and Company Product table

Seller_Company		Seller_Product		Company_Product	
Seller	Company	Seller	Product	Company	Product
Rupali	Godrej	Rupali	Cinthol	Godrej	Cinthol
Sharda	Dabur	Sharda	Honey	Dabur	Honey
Sunil	Amul	Sharda	HairOil	Dabur	HairOil
Sunil	Britania	Sharda	RoseWater	Dabur	RoseWater
		Sunil	Icecream	Amul	Icecream
		Sunil	Biscuit	Britania	Biscuit

Thus the table is in 5th normal form.

2.15 Two Marks Questions with Answers

Q.1 Explain Entity Relationship model.

AU : May-16

- Ans. :**
- The ER data model specifies enterprise schema that represents the overall logical structure of a database.
 - The E-R model is very useful in mapping the meanings and interactions of real-world entities onto a conceptual schema.

Q.2 Give the limitations of E-R model ? How do you overcome this ?

AU : May-07

- Ans. :**
- 1) **Loss of information content :** Some information be lost or hidden in ER model
 - 2) **Limited relationship representation :** ER model represents limited relationship as compared to another data models like relational model etc.
 - 3) **No representation of data manipulation :** It is difficult to show data manipulation in ER model.
 - 4) **Popular for high level design :** ER model is very popular for designing high level design.

Q.3 List the design phases of Entity Relationship model.

- Ans. :**
- 1) Requirement Analysis,
 - 2) Conceptual Database Design,
 - 3) Logical Database Design,
 - 4) Schema Refinement,
 - 5) Physical Database Design,
 - 6) Application and Security Design.

Q.4 What is an entity ?

AU : May-14

- Ans. :**
- 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. Entity is represented as a box, in ER model.

Q.5 What do you mean by derived attributes ?

- Ans. :**
- 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.

Q.6 What is a weak entity ? Give example.

AU : Dec.-16, May-18

- Ans. :** Refer section 2.3.4

Q.7 What are the problems caused by redundancy ?

AU : Dec.-17

Ans. : **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.

Q.8 Define functional dependency.

AU : Dec 04,05, May 05,14,15

Ans. : Let P and Q be sets of columns, then : P functionally determines Q, written $P \rightarrow Q$ if and only if any two rows that are equal on (all the attributes in) P must be equal on (all the attributes in) Q.

In other words, the functional dependency holds if

$$T1.P = T2.P, \text{ then } T1.Q=T2.Q$$

Where notation $T1.P$ projects the tuple $T1$ onto the attribute in P.

Q.9 Why certain functional dependencies are called trivial functional dependencies ?

AU : May-06,12

Ans. : • A functional dependency $FD : X \rightarrow Y$ is called trivial if Y is a subset of X.

This kind of dependency is called trivial because it can be derived from common sense. If one "side" is a subset of the other, it's considered trivial. The left side is considered the determinant and the right the dependent.

- **For example** - $\{A,B\} \rightarrow B$ is a trivial functional dependency because B is a subset of A,B. Since $\{A,B\} \rightarrow B$ includes B, the value of B can be determined. It's a trivial functional dependency because determining B is satisfied by its relationship to A,B

Q.10 Define normalization.

AU : May -14

Ans. : 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)

Q.11 State anomalies of 1NF.

AU : Dec.-15

Ans. : All the insertion, deletion and update anomalies are in 1NF relation

Q.12 What is multivalued dependency ?

AU : Dec. -06

Ans. : A table is said to have multi-valued dependency, if the following conditions are true,

- 1) For a dependency $A \rightarrow B$, if for a single value of A, **multiple values** of B exists, then the table may have multi-values dependency.
- 2) Also, a table should have **at-least 3 columns** for it to have a multi-valued dependency.
- 3) And, for a relation $R(A,B,C)$, if there is a multi-valued **dependency between A and B**, then B and C should be independent of each other.

Q.13 Describe BCNF and describe a relation which is in BCNF.

AU : Dec. -02

Ans. : Refer section 2.12.

Q.14 Why 4NF in normal form is more desirable than BCNF ?

AU : Dec. -14

Ans. :

- 4NF is more desirable than BCNF because it reduces the repetition of information.
- If we consider a BCNF schema not in 4NF we observe that decomposition into 4NF does not lose information provided that a lossless join decomposition is used, yet redundancy is reduced.

Q.15 Give an example of a relation schema R and set of dependencies such that R is in BCNF but not in 4NF.

AU : May -12

Ans. : Consider relation $R(A,B,C,D)$ with dependencies

 $AB \rightarrow C$
 $ABC \rightarrow D$
 $AC \rightarrow B$

Here the only key is AB. Thus each functional dependency has superkey on the left. But MVD has non-superky on its left. So it is not 4NF.

Q.16 Show that if a relation is in BCNF, then it is also in 3NF.

AU : Dec.-12

Ans. :

- Boyce and Codd Normal Form is a **higher version** of the Third Normal form.
- A 3NF table which **does not have multiple overlapping** candidate keys is said to be in BCNF. When the table is in BCNF then it doesn't have partial functional dependency as well as transitive dependency.
- Hence it is true that if relation is in BCNF then it is also in 3NF.

Q.17 Why it is necessary to decompose a relation ?

AU : May-07

- Ans. :**
- Decomposition is the process of breaking down one table into multiple tables.
 - The decomposition is used for eliminating redundancy.

Q.18 Explain atleast two desirable properties of decomposition.

AU : May-03,17, Dec.-05

Ans. :

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.

Q.19 Explain with simple example lossless join decomposition.

AU : May-03

Ans. : Refer section 2.10.1.



Notes

UNIT - III

3

Transactions

Syllabus

Transaction Concepts - ACID Properties - Schedules - Serializability - Concurrency Control - Need for Concurrency - Locking Protocols - Two Phase Locking - Deadlock - Transaction Recovery - Save Points - Isolation Levels - SQL Facilities for Concurrency and Recovery.

Contents

3.1	<i>Transaction Concepts</i>	Dec.-14	Marks 4
3.2	<i>ACID Properties</i>	May-14, 18	Marks 8
3.3	<i>Transaction States</i>	Dec.-11, May-14,18	Marks 8
3.4	<i>Schedules</i>		
3.5	<i>Serializability</i>	Dec.-15, May-15,18	Marks 8
3.6	<i>Transaction Isolation and Atomicity</i>		
3.7	<i>Introduction to Concurrency Control</i>		
3.8	<i>Need for Concurrency</i>	May-17	Marks 13
3.9	<i>Locking Protocols.....</i>	Dec-15,17, May-16	Marks 16
3.10	<i>Two Phase Locking.....</i>	May-14,18, Dec.-16	Marks 7
3.11	<i>Time Stamp Based Protocol</i>		
3.12	<i>Dead Lock.....</i>	Dec-14,15,16, May-09	Marks 16
3.13	<i>Transaction Recovery</i>		
3.14	<i>Save Points</i>		
3.15	<i>Isolation Levels</i>		
3.16	<i>SQL Facilities for Concurrency and Recovery</i>	May-14	Marks 8
3.17	<i>Two Marks Questions with Answers</i>		

Part I : Introduction Transactions**3.1 Transaction Concepts****AU : Dec.-14, Marks 4**

Definition : 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 **four steps** denote **one transaction**.

In a database, each transaction should maintain ACID property to meet the consistency and integrity of the database.

University Question

1. Write a short note on – Transaction Concept

AU : Dec.-14, Marks 4**3.2 ACID Properties****AU : May-14, 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.

University Questions

1. Explain with an example the properties that must be satisfied by transaction

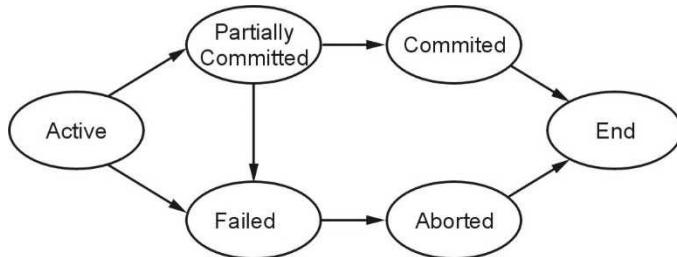
AU : May-18, Marks 7

2. Explain the ACID properties of transaction

AU : May-14, Marks 8

3.3 Transaction States**AU : Dec.-11, May-14,18, Marks 8**

Each transaction has following five states :

**Fig. 3.3.1 Transaction States**

- 1) **Active** : This is the first state of transaction. For example : insertion, deletion or updation 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 3.3.1 Define a transaction. Then discuss the following with relevant examples :

1. A read only transaction
2. A read write transaction
3. An aborted transaction

AU : Dec.-11, Marks 8

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	
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)	

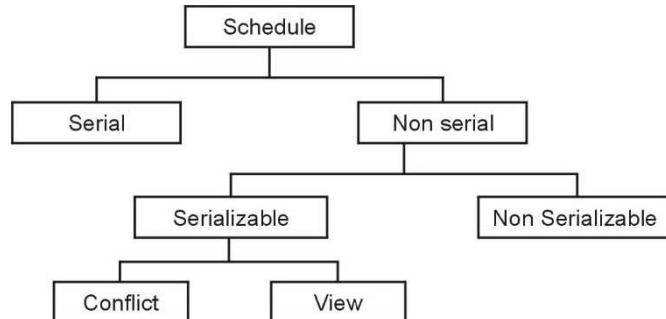
University Questions

1. During execution, a transaction passes through several states, until it finally commits or aborts. List all possible sequences of states through which transaction may pass. Explain why each state transaction may occur ? AU : May-18, Marks 6

2. With a neat sketch explain the states of transaction. AU : May-14, Marks 8

3.4 Schedules

Schedule is an order of multiple transactions executing in concurrent environment. Following figure represents the types of schedules.

**Fig. 3.4.1 : 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 T_1 and T_2

T ₁	T ₂
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,

T ₁	T ₂
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.

3.5 Serializability

AU : Dec.-15, May-15, 18, Marks 8

- 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.

- For example :

T_1	A	B	T_2
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 T_1 will read the values from database as $A=100$, $B=100$ and modify the values of A and B. But transaction T_2 will read the modified value i.e. 90 and will modify it to 80 and perform write operation. Thus at the end of transaction T_1 value of A will be 90 but at end of transaction T_2 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

Serial Schedule	Serializable Schedule
No concurrency is allowed in serial schedule.	Concurrency is allowed in serializable schedule.
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	
T1	T2	T1	T2
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(A)	Write(B)	

- There are two types of serializabilities : conflict serializability and view serializability

3.5.1 Conflict Serializability

Definition : Suppose T_1 and T_2 are two transactions and I_1 and I_2 are the instructions in T_1 and T_2 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 $T_i \rightarrow T_j$ 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 $\text{write}(Q)$ before T_j executes $\text{write}(Q)$.

- 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 $\text{write}(Q)$ before T_j executes $\text{read}(Q)$, then draw edge from T_i to T_j .
2. T_i executes $\text{read}(Q)$ before T_j executes $\text{write}(Q)$, then draw edge from T_i to T_j
3. T_i executes $\text{write}(Q)$ before T_j executes $\text{write}(Q)$, 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 3.5.1 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



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 to T_2 .



Step 5 : Repeat the step 4 while reading from top to bottom. Finally the **precedence graph** will be as follows

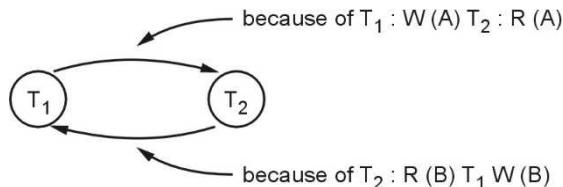


Fig. 3.5.1 : Precedence graph

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 3.5.2 Check whether following schedule is conflict serializable or not. If it is not conflict serializable then find the serializability order.

T_1	T_2	T_3
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 :

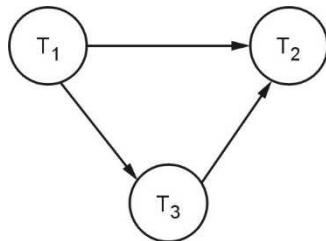


Fig. 3.5.2 Precedence graph

Step 2 : 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 3 : 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 4 : Find the vertex which has no incoming edge which is T_1 . Finally find the vertex having no outgoing edge which is T_2 . So in between them is T_3 . Hence the order will be $T_1 - T_3 - T_2$

3.5.2 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 S_1 and S_2 consisting of transactions

T_1 and T_2 respectively, then schedules S_1 and S_2 are said to be view equivalent schedule if it satisfies following three conditions :

- o If transaction T_1 reads a data item A from the database initially in schedule S_1 , then in schedule S_2 also, T_1 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.
- o If data item A has been updated at last by transaction T_i in schedule S_1 , then in schedule S_2 also, the data item A must be updated at last by transaction T_i .
- o If transaction T_i reads a data item that has been updated by the transaction T_j in schedule S_1 , then in schedule S_2 also, transaction T_i must read the same data item that has been updated by transaction T_j . In other words the Write-Read sequence must be same.

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 3.5.3 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(C)		
		W(B)
W(B)		

Solution : i) The initial read operation is performed by T_2 on data item A or by T_1 on data item C. Hence we will begin with T_2 or T_1 . We will choose T_2 at the beginning.

ii) The final write is performed by T_1 on the same data item B. Hence T_1 will be at the last position.

iii) The data item C is written by T_3 and then it is read by T_1 . Hence T_3 should appear before T_1 .

Thus we get the order of schedule of view serializability as $T_2 - T_1 - T_3$

Example 3.5.4 Consider 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)

```

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

$A \vee B = A$ 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

$A \vee B = A$ 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 3.5.5 Test serializability of the following schedule :

- i) $r_1(x);r_3(x);w_1(x);r_2(x);w_3(x)$ ii) $r_3(x);r_2(x);w_3(x);r_1(x);w_1(x)$

Solution : i) $r_1(x);r_3(x);w_1(x);r_2(x);w_3(x)$

The r_1 represents the read operation of transaction T_1 , w_3 represents the write operation on transaction T_3 and so on. Hence from given sequence the schedule for three transactions can be represented as follows :

T_1	T_2	T_3
$r_1(x)$		
		$r_3(x)$
$w_1(x)$		
	$r_2(x)$	
		$w_3(x)$

Step 1 : We will use the precedence graph method to check the serializability. As there are three transactions, three nodes are created for each transaction.

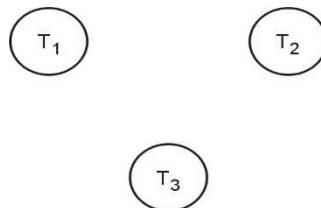


Fig. 3.5.3 Nodes

Step 2 : We will read from top to bottom. Initially we read $r_1(x)$ and keep on moving bottom in search of write operation. Here all the transactions work on same data item i.e. x . Now we get a write operation in T_3 as $w_3(x)$. Hence the dependency is from T_1 to T_3 . Therefore we draw edge from T_1 to T_3 .

Similarly, for $r_3(x)$ we get $w_1(x)$ pair. Hence there will be edge from T_3 to T_1 . Continuing in this fashion we get the precedence graph as

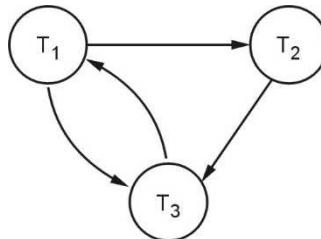


Fig. 3.5.4 Precedence Graph

Step 3 : As cycle exists in the above precedence graph, we conclude that it is **not serializable**.

ii) $r_3(x);r_2(x);w_3(x);r_1(x);w_1(x)$

From the given sequence the schedule can be represented as follows :

T_1	T_2	T_3
		$r_3(x)$
	$r_2(x)$	
		$w_3(x)$
$r_1(x)$		
$w_1(x)$		

Step 1 : Read the schedule from top to bottom for pair of operations. For $r_3(x)$ we get $w_1(x)$ pair. Hence edge exists from T_3 to T_1 in precedence graph.

There is a pair from $r_2(x) : w_3(x)$. Hence edge exists from T_2 to T_3 .

There is a pair from $r_2(x) : w_1(x)$. Hence edge exists from T_2 to T_1 .

There is a pair from $w_3(x) : r_1(x)$. Hence edge exists from T_3 to T_1 .

Step 2 : The precedence graph will then be as follows –

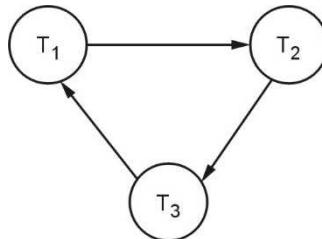
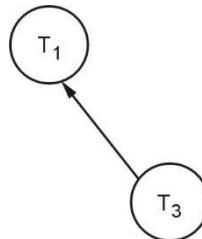


Fig. 3.5.5 Precedence graph

Step 3 : As there is no cycle in the above graph, the given schedule is **serializable**.

Step 4 : The serializability order for consistent schedule will be obtained by applying topological sorting on above drawn precedence graph. This can be achieved as follows,

Sub-Step 1 : Find the node having no incoming edge. We obtain T_2 is such a node. Hence T_2 is at the beginning of the serializability sequence. Now delete T_2 . The Graph will be



Sub-Step 2 : Repeat sub-Step 1, We obtain T_3 and T_1 nodes as a sequence.

Thus we obtain the sequence of transactions as T_2, T_3 and T_1 . Hence the serializability order is

$r_2(x);r_3(x);w_3(x);r_1(x);w_1(x)$

Example 3.5.6 Consider the following schedules. The actions are listed in the order they are scheduled, and prefixed with the transaction name.

$S1 : T1 : R(X), \quad T2 : R(X), \quad T1 : W(Y), \quad T2 : W(Y) \quad T1 : R(Y), \quad T2 : R(Y)$

$S2 : T3 : W(X), \quad T1 : R(X), \quad T1 : W(Y), \quad T2 : R(Z), \quad T2 : W(Z) \quad T3 : R(Z)$

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 ?

AU : May-15, Marks 2 + 7 + 7

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 S1: From above given example in the top to bottom scanning we find the conflict as

- **T1 : W(Y), T2 : W(Y) and**
- **T2 : W(Y), T1 : 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

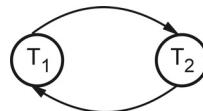


Fig. 3.5.6 Precedence graph for S1

For S2: The conflicts are

- **T3 : W(X), T1 : R(X)**
- **T2 : W(Z) T3 : R(Z)**

Hence the precedence graph is as follows -

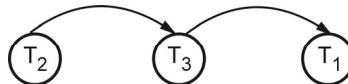


Fig. 3.5.7 Precedence graph for S2

(ii)

- S1 is **not conflict-serializable** since the dependency graph has a cycle.
- S2 is conflict-serializable as the dependency graph is acyclic. The order $T_2-T_3-T_1$ is the only equivalent **serial order**.

(iii)

- S1 is **not view serializable**.
- S2 is trivially view-serializable as it is conflict serializable. The **only serial order allowed** is

T2-T3-T1.

University Questions

1. Explain Conflict serializability and view serializability.

AU : May-18, Marks 6, Dec.-15, Marks 8

3.6 Transaction Isolation and Atomicity

- 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.

3.6.1 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	

- 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

3.6.2 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

3.7 Introduction to 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.

3.8 Need for Concurrency

AU : May-17, Marks 13

- 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 T2.
- (3) During transaction T1, the salary is incremented by ₹ 200
- (4) During transaction T2, the salary is incremented by ₹ 500

T ₁	T ₂	
Read		Salary = ₹ 1000
	Read	Salary = ₹ 1000
Update Increment salary by ₹ 200		Salary = ₹ 1200
	Update Increment salary by ₹ 500	Salary = ₹ 1500

This update is lost

Time

Only this update is successful

The result of the above sequence is that the update made by transaction T1 is completely lost. Therefor 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

T ₁	T ₂
R(A)	
A=A+50	
W(A)	R(A)
	Dirty read
	A=A-20
	W(A)
	Commit
Commit	

For example – Consider following transactions -

Assume initially salary is = ₹ 1000

The diagram illustrates a database row over time. The row has three columns: T₁, T₂, and a third column. A vertical arrow labeled 'Time' points downwards. The timeline marks t₁, t₂, and t₃. At t₁, T₁ reads the row, which contains 'Salary = ₹ 1000'. At t₂, T₂ updates the row, changing it to 'Update Salary = Salary + 200' (Salary = ₹ 1200). At t₃, T₂ performs a 'Rollback', returning the row to its state at t₁ (Salary = ₹ 1000). A dashed box highlights the period from t₂ to t₃, during which T₁'s read is labeled as a 'Dirty Read'.

T ₁	T ₁	
...	...	Salary = ₹ 1000
	Update Salary = Salary + 200	Salary = ₹ 1200
Read		Salary = ₹ 1200
	Rollback	Salary = ₹ 1000

- (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 –

The diagram illustrates a database row over time. The row has three columns: T₁, T₂, and a third column. A vertical arrow labeled 'Time' points downwards. The timeline marks t₁, t₂, t₃, and t₄. At t₁, T₁ reads the row, which contains 'Salary = ₹ 1000'. At t₂, T₂ updates the row, changing it to 'Update salary from ₹ 1000 to ₹ 1200'. At t₃, T₂ commits the update. At t₄, T₁ reads the row again, which now contains 'Salary = ₹ 1200'. The second read by T₁ is highlighted as a 'Non-repeatable read'.

T ₁	T ₂	
Read		Salary = ₹ 1000
	Update salary from ₹ 1000 to ₹ 1200	Salary = ₹ 1200
	Commit	
Read		Salary = ₹ 1200

- (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		"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.

University Question

1. Discuss the violations caused by each of the following: dirty read, non repeatable read and phantoms with suitable example

AU : May-17, Marks 13

3.9 Locking Protocols

AU : May-16, Dec.-15, 17, Marks 16

3.9.1 Why Do we Need Locks ?

- 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.

3.9.2 Simple Lock Based Protocol

- 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. 3.9.1 Types of locks

- 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**.
 - 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. 3.9.2 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 held 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 T_1 transaction and Read the data item A in transaction T_2 . The scenario can be represented with the help of locks and concurrency control manager as follows :

	T_1	T_2	Concurrency control manager
Exclusive Lock	Lock-X(A)		
			Grant X(A,T1) because in T1 there is write operation.
	R(A)		
	A=A-50		
	W(A)		
	Unlock(A)		
Shared Lock		Lock-S(A)	
			Grant S(A,T2) because in T2 there is Read operation
		R(A)	
		Unlock(A)	

University Questions

1. State and explain the lock based concurrency control with suitable example

AU : Dec-17, Marks 13, May-16, Marks 16

2. What is Concurrency control ? How is implemented in DBMS ? Illustrate with suitable example.

AU : Dec-15, Marks 8

3.10 Two Phase Locking

AU : May-14, 18, Dec.-16, Marks 7

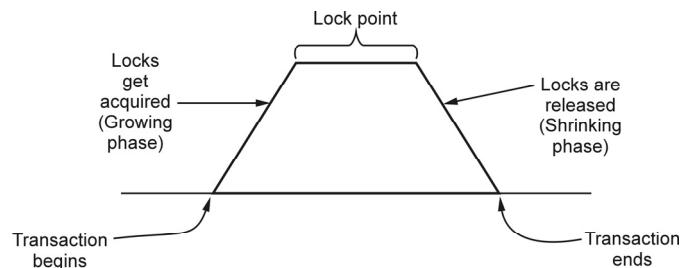
- The two phase locking is a protocol in which there are two phases :
- i) **Growing phase (Locking phase) :** It is a phase in which the transaction may obtain locks but does not release any lock.

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

- Lock(A)
- Lock(B)
- Lock(C)
-
- ...
- ...

Lock Point



Unlock(A)

Unlock(B)
Unlock(C)

For example –

Consider following transactions

T1	T2
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 3.10.1 Prove that two phase locking guarantees serializability.

Solution:

- o Serializability is mainly an issue of handling write operation. Because any inconsistency may only be created by **write** operation.
- o Multiple reads on a database item can happen parallelly.
- o 2-Phase locking protocol restricts this unwanted read/write by applying **exclusive lock**.
- o 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.

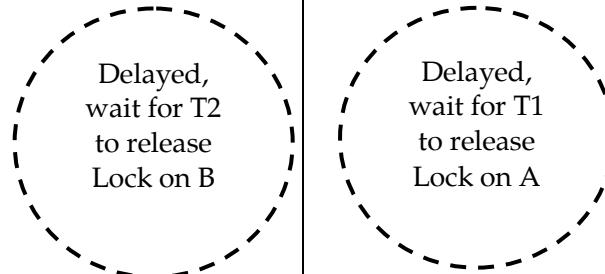
The last lock position is in T_1 , then it is in T_2 . Hence the serializability will be $T_1 \rightarrow T_2$ based on lock points. Hence The **serializability sequence** can be $R1(A);R2(A);R1(B);W1(B)$

Limitations of Two Phase Locking Protocol

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

T1	T2
Lock-X(A)	Lock-X(B)
Read(A)	Read(B)
A=A-50	B=B+100
Write(A)	Write(B)
	

(2) Cascading Rollback : Cascading rollback is a situation in which a single transaction failure leads to a series of transaction rollback. For example –

T1	T2	T3
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.

3.10.1 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 T_1 has exclusive lock, then T_1 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 T_1 , 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 3.10.2 Consider the following two transactions :

$T1:read(A)$

$Read(B);$

If A=0 then B=B+1;

$Write(B)$

$T2:read(B); read(A)$

If B=0 then A=A+1

$Write(A)$

Add lock and unlock instructions to transactions $T1$ and $T2$, so that they observe two phase locking protocol. Can the execution of these transactions result in deadlock ?

AU : Dec.-16, Marks 6

Solution :

T1	T2
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.

T1	T2
Lock-S(A)	Lock-S(B)
Read(A)	Read(B)
Lock-X(B)	Lock-X(A)
Now it will wait for T2 to release exclusive lock on A	Now it will wait for T1 to release exclusive lock on B

3.10.2 Lock Conversion

Lock conversion is a mechanism in two phase locking mechanism - which allows conversion of shared lock to exclusive lock or exclusive lock to shared lock.

Method of Conversion :

First Phase :

- can acquire a lock-S on item
- can acquire a lock-X on item
- can convert a lock-S to a lock-X (**upgrade**)

Second Phase :

- can release a lock-S
- can release a lock-X
- can convert a lock-X to a lock-S (**downgrade**)

This protocol assures serializability. But still relies on the programmer to insert the various locking instructions.

For example – Consider following two transactions –

T_1	T_2
R(A)	R(A)
R(B)	R(B)
...	
R(C)	
...	
W(A)	

Here if we start applying locks, then we must apply the **exclusive lock** on data item A, because we have to read as well as write on data item A. Another transaction T2 does not get shared lock on A until transaction T1 performs write operation on A. Since transaction T1 needs exclusive lock only at the end when it performs write operation on A, it is better if T1 could initially lock A in shared mode and then later change it to exclusive mode lock when it performs write operation. In such situation, the lock conversion mechanism becomes useful.

When we convert the shared mode lock to exclusive mode then it is called **upgrading** and when we convert exclusive mode lock to shared mode then it is called **downgrading**.

Also note that upgrading takes place only in **growing phase** and downgrading takes place only in shrinking phase. Thus we can refine above transactions using lock conversion mechanism as follows –

T_1	T_2
Lock-S(A)	
R(A)	
	Lock-S(A)
	R(A)
Lock-S(B)	
R(B)	
	Lock-S(B)
	R(B)
	Unlock(A)
	Unlock(B)
...	

Lock-S(C)	
R(C)	
...	
Upgrade(A)	
W(A)	
Unlock(A)	
Unlock(B)	
Unlock(C)	

University Questions

1. What is concurrency control ? Explain two phase locking protocol with an example

AU : May-18, Marks 7

2. Illustrate two phase locking protocol with an example.

AU : May-14, Dec.-16, Marks 6

3.11 Time Stamp Based Protocol

- The time stamp ordering protocol is a scheme in which the order of transaction is 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.
- There are two types of time stamps -
 - Read Time stamp i.e. . **R-timestamp(A)** - It is the youngest time stamp that performed on read operation.
 - Write Time stamp i.e. **W-timestamp(A)** - It is the youngest time stamp that performed on write operation.
- These timestamps are updated whenever a new **read(A)** or **write(A)** instruction is executed.
- The timestamp-ordering protocol ensures that any conflicting read and write operations are executed in timestamp order. This protocol operates as follows :

1. Suppose that transaction T_i issues **read(A)**.

- If $TS(T_i) < W\text{-timestamp}(A)$, then T_i needs to read a value of A that was already overwritten.

Hence, the read operation is rejected, and T_i is rolled back.

For example : Consider two transactions T_1 and T_2 with timestamps 10 sec and 20 sec. That means younger transaction has already performed write operation and now older transaction is issuing the **Read (A)** request. This is as shown below

T_1 (10 sec) older transaction	T_2 (20 sec) younger transaction
	W(A)
R(A)	

Then W-timestamp(A) for transaction T_2 = 20 sec. and TS(T_1) for R(A) = 10. i.e.

$TS(T_1) < W\text{-timestamp}(A)$. Hence reject R(A) of T_1 and rollback.

- b. If $TS(T_i) \geq W\text{-timestamp}(A)$, then the read operation is executed, and R-timestamp(A) is set to the maximum of R-timestamp(A) and TS(T_i). For example :

T_1 (20 sec) younger transaction	T_2 (10 sec) older transaction
	W(A)
R(A)	

Set R-timestamp (A) = 20 sec.

2. Suppose that transaction T_i issues write (A).

- a. If $TS(T_i)$ (older transaction) $<$ R-timestamp (A) (younger transaction), and then if T_i is attempting to Write an obsolete value of A. Then the system rejects the write operation and rolls T_i back.

For example : Consider two transactions T_1 and T_2 with timestamps 10 sec and 20 sec.

T_1 (10 sec) older transaction	T_2 (20 sec) younger transaction
	R(A)
W(A)	

Here $R_2\text{-TS}(A)=20$ sec and $TS(T_1)=10$ sec i.e. $TS(T_1) < R_2\text{-Timestamp}(A)$, it is invalid operation and then we must reject W(A) in T_1 and must roll back.

- b. If $TS(T_i) < W\text{-timestamp}(A)$, then T_i is attempting to write an obsolete value of A. Hence, the system rejects this write operation and rolls T_i back.

For example : Consider two transactions T_1 and T_2 with timestamps 10 sec and 20 sec. That means younger transaction has already performed write operation and now older transaction is issuing the write request. This is as shown below

T_1 (10 sec) older transaction	T_2 (20 sec) younger transaction
	W(A)
W(A)	

Here W-timestamp(A) of T_2 =20 sec and TS(T_1) for write request =10 sec i.e.

$TS(T_1) < W\text{-timestamp}(A)$, it is invalid operation and then we must reject W(A) in T_1 and must roll back.

- c. Otherwise, the system executes the write operation and sets W-timestamp(A) to $TS(T_i)$.

For example just change the timestamps of T_1 and T_2 . This can be as shown below -

T_1 (20 sec) younger transaction	T_2 (10 sec) older transaction
	R(A) or W(A)
W(A)	

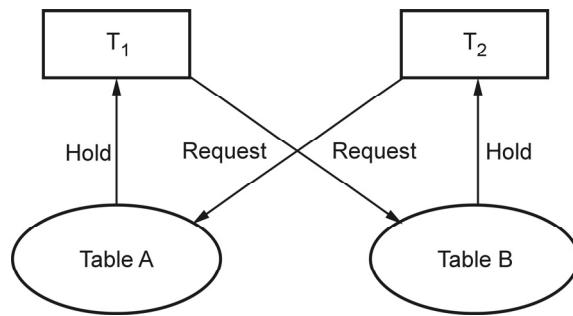
This is allowed and the write operation of T_1 will be executed.

3.12 Dead Lock

AU : May-09, Dec. 14, 15, 16, Marks 16

Deadlock is a specific concurrency problem in which two transactions depend on each other for something.

For example – Consider that transaction T1 holds a lock on some rows of table A and needs to update some rows in the B table. Simultaneously, transaction T2 holds locks on some rows in the B table and needs to update the rows in the A table held by Transaction T1.



Now, the main problem arises. Now Transaction T1 is waiting for T2 to release its lock and similarly, transaction T2 is waiting for T1 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 processso 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 –

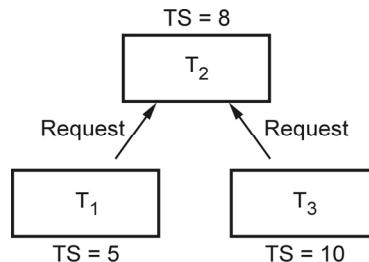
(i) Wait-Die :

- 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 .



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 T1 needs a resource held by T2 and T3 also needs the resource held by T2, with TS(T1)=5, TS(T2)=8 and TS(T3)=10, then T1 being older waits and T3 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.

T1	T2
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 T1 has **Read** operation and then T2 has **Write** operation then draw an edge T1->T2.

Rule 2: If T1 has **Write** operation and then T2 has **Read** operation then draw an edge T1->T2

Rule 3: If T1 has **Write** operation and then T2 has **Write** operation then draw an edge T1->T2

Let us draw wait-for graph

Step 1 : Draw vertices for all the transactions



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 -

T1	T2
R(A)	
	R(A)
W(A)	
R(B)	
	W(A)
W(B)	



Step 3 :

T1	T2
R(A)	
	R(A)
W(A) ←	
R(B)	
	W(A)
W(B)	



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 3.12.1 Give an example of a scenario where two phase locking leads to deadlock.

AU : May-09, Marks 4

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.

University Questions

1. Outline deadlock handling mechanisms.

AU : Dec-16, Marks 7

2. What is deadlock ? How does it occur ? How transactions can be written to

(i) Avoid deadlock.

(ii) Guarantee correct execution.

Illustrate with suitable example.

AU : Dec.-15, Marks 16

3. Write short note on- Deadlock

AU : Dec.-14, Marks 4

3.13 Transaction Recovery

- An integral part of a database system is a recovery scheme that can **restore the database** to the consistent state that existed before the failure.
- The recovery scheme must also provide **high availability**; that means, it must minimize the time for which the database is not usable after a failure.

3.13.1 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.

3.13.2 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.

3.13.3 Recovery with Concurrent Transactions

There are four ways for recovery with concurrent transactions :

- 1) **Interaction with concurrency control** : In this scheme recovery depends upon the concurrency control scheme which is used for the transaction. If a transaction gets failed, then rollback and undo all the updates performed by transaction.
- 2) **Transaction Rollback** : In this scheme, if the transaction gets failed, then the failed transaction can be rolled back with the help of **log**. The system scans the log backward, and with the help of log entries the system can restore the data items.
- 3) **Checkpoints** : The checkpoints are used to reduce the number of log records.
- 4) **Restart recovery** : When the system recovers from the crash it constructs two lists : Undo-list and Redo-list. The undo list consists of transactions to be undone. The redo list consists of transactions to be redone. These two lists are handled as follows

Step 1 : Initially both the lists are empty.

Step 2 : The system scans the log entries from backwards. It scans each entry until it finds first **checkpoint record**.

3.13.4 Shadow Copy Technique

- In the **shadow-copy scheme**, a transaction that wants to update the database first creates a complete **copy of the database**.
- All updates are done on the new database copy, leaving the original copy, the shadow copy, untouched.
- If at any point the transaction has to be aborted, the system merely deletes the new copy. The old copy of the database has not been affected.
- The current copy of the database is identified by a pointer, called **db-pointer**, which is stored on disk.
- If the transaction **partially commits**, it is committed as follows :
 - First, the operating system is asked to make sure that all pages of the new copy of the database have been written out to disk.
 - After the operating system has written all the pages to disk, the database system updates the pointer **db-pointer** to point to the new copy of the database
 - The new copy then becomes the current copy of the database.
 - The old copy of the database is then deleted.
 - The transaction is said to have been **committed** at the point where the **updated db-pointer** is written to disk.
 - The disk system guarantees that it will update **db-pointer** atomically, as long as we make sure that **db-pointer** lies entirely in a single sector.

Where is shadow copy technique used ?

- Shadow copy schemes are commonly used by text editors.
- Shadow copying can be used for small databases.

3.13.5 Log Based Recovery Approach

- Log is the most commonly used structure for recording the modifications that are 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. 3.13.1. This structure is for <update> operation

Transaction ID(T_i)	Data Item Name	Old Value of Data Item	New Value of Data Item
-------------------------	----------------	------------------------	------------------------

- For example : The sample log file is

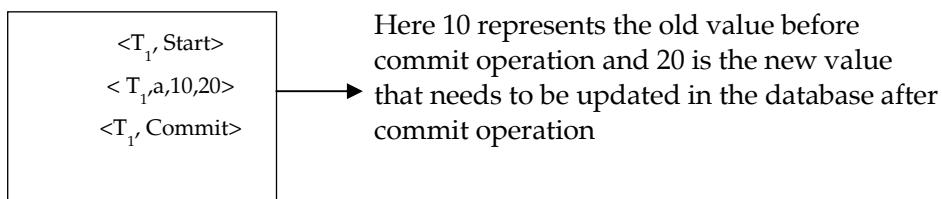


Fig. 3.13.1 Sample Log File

- 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.

1. 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

- a) Just after write (B, b)
- b) Just after write (C, 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 $< T_1, \text{commit} >$ gets executed. Therefore $A = 90$, $B = 210$ and $C = 300$ are the values for database.

c) Just after $< T_2, \text{commit} >$

The log records are as follows :

$< T_1, \text{Start} >$

$< T_1, A, 90 >$

$< T_1, B, 210 >$

$< T_1, \text{Commit} >$

$< T_2, \text{Start} >$

$< T_2, 6, 280 >$

$< T_2, \text{Commit} >$

———— 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 $A = 90$, $B = 210$, $C = 280$.

2. 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

3.14 Save Points

The COMMIT, ROLLBACK, and SAVEPOINT are collectively considered as Transaction Commands

- (1) **COMMIT** : The COMMIT command is used to save permanently any transaction to database.

When we perform, Read or Write operations to the database then those changes can be undone by rollback operations. To make these changes permanent, we should make use of **commit**

- (2) **ROLLBACK** : The ROLLBACK command is used to undo transactions that have not already saved to database. For example

Consider the database table as

RollNo	Name
1	AAA
2	BBB
3	CCC
4	DDD
5	EEE

Fig.3.14.1 Student Table

Following command will delete the record from the database, but if we immediately performs ROLLBACK, then this deletion is undone.

For instance –

```
DELETE FROM Student
```

```
WHERE RollNo =2;
```

```
ROLLBACK;
```

Then the resultant table will be

RollNo	Name
1	AAA
2	BBB
3	CCC
4	DDD
5	EEE

(3) SAVEPOINT : A SAVEPOINT is a point in a transaction when you can roll the transaction back to a certain point without rolling back the entire transaction. The SAVEPOINT can be created as

```
SAVEPOINT savepoint_name;
```

Then we can ROLLBACK to SAVEPOIT as

```
ROLLBACK TO savepoint_name;
```

For example – Consider Student table as follows –

RollNo	Name
1	AAA
2	BBB
3	CCC
4	DDD
5	EEE

Fig.3.14.2 Student Table

Consider Following commands

```
SQL> SAVEPOINT S1
```

SQL>DELETE FROM Student

Where RollNo=2;

SQL> SAVEPOINT S2

SQL>DELETE FROM Student

Where RollNo=3;

SQL> SAVEPOINT S3

SQL>DELETE FROM Student

Where RollNo=4

SQL> SAVEPOINT S4

SQL>DELETE FROM Student

Where RollNo=5

SQL> ROLLBACK TO S3;

Then the resultant table will be

RollNo	Name
1	AAA
2	BBB
3	CCC

Thus the effect of deleting the record having RollNo 2, and RollNo3 is undone.

3.15 Isolation Levels

- The consistency of the database is maintained with the help of isolation property(one of the property from ACID properties) of transaction.
- The transaction should take place in a system in such a way that it is the only transaction that is accessing the resources in a database system at particular instance.
- Isolation levels defines the degree to which a transaction must be isolated from the data modifications made by any other transaction in the database system.
- There are four levels of transaction isolation defined by SQL -
- **Serializable :**
 - This is the **Highest isolation level**.
 - **Serializable execution is defined** to be an execution of operations in which concurrently executing transactions **appears to be serially executing**.

- **Repeatable Read :**
 - This is the **most restrictive isolation level**.
 - The transaction **holds read locks on all rows** it references.
 - It holds **write locks on all rows** it inserts, updates, or deletes.
 - Since other transaction cannot read, update or delete these rows, it avoids non repeatable read.
 - **Read Committed :**
 - This isolation level allows only **committed data to be read**.
 - Thus it does **not allows dirty read** (i.e. one transaction reading of data immediately after written by another transaction).
 - The transaction hold a **read or write lock** on the current row, and thus prevent other rows from reading, updating or deleting it.
 - **Read Uncommitted :**
 - It is **lowest isolation level**.
 - In this level, one transaction may read not yet committed changes made by other transaction.
 - This level **allows dirty reads**.
- In this level **transactions are not isolated** from each other.

3.16 SQL Facilities for Concurrency and Recovery

AU : May-14, Marks 8

It is possible to achieve concurrency control and recovery using SQL statements. Following is an illustration of implementation of isolation level

The different isolation levels can be set in SQL as follows

(1) READ UNCOMMITTED : It permits dirty reads, nonrepeatable reads, phantoms.

In SQL, we can write

SET TRANSACTION

ISOLATION LEVEL READ UNCOMMITTED

(2) READ COMMITTED : It permits nonrepeatable reads and phantoms, but prohibits dirty reads

In SQL, we can write

SET TRANSACTION

ISOLATION LEVEL READ COMMITTED

(3) REPEATABLE READ : It permits phantoms, but not dirty reads, nor nonrepeatable reads.

In SQL, we can write

SET TRANSACTION

ISOLATION LEVEL REPEATABLE READ

(4) **SERIALIZABLE** : In this isolation level there is true two phase locking, and it keeps all transactions serializable.

In SQL, we can write

SET TRANSACTION

ISOLATION LEVEL SERIALIZABLE

For example : Let us discuss the Implementation of READ COMMITTED isolation level using SQL facilities.

Step 1 : Create Table

```
CREATE TABLE Emp(ID int,Name char(50),Salary int)
```

Step 2 : Insert values into the table

```
INSERT INTO Emp(ID,Name,Salary)
```

```
VALUES ( 1,'Sharda',1000)
```

```
INSERT INTO Emp(ID,Name,Salary)
```

```
VALUES( 2,'Ashwini',2000)
```

```
INSERT INTO Emp(ID,Name,Salary)
```

```
VALUES( 3,'Madhura',3000)
```

The Table will be

ID	Name	Salary
1	Sharda	1000
2	Ashwini	2000
3	Madhura	3000

Step 3 : In select query it will take only committed values of table. If any transaction is opened and incompletely completed on table in others sessions then select query will wait till no transactions are pending on same table. Read Committed is the default transaction isolation level.

Session 1:

```
BEGIN TRAN
```

```
UPDATE emp SET Salary=999 WHERE ID=1
```

```
WAITFOR DELAY '00:00:15'
```

```
COMMIT
```

Session 2 :

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED
```

```
SELECT Salary FROM emp WHERE ID=1
```

Step 4 : If we run above mentioned two sessions concurrently then

The output will be 999

Step 5 : Following SQL facilities demonstrate the **REPEATABLE READ** isolation level. In this technique, select query data of table that is used under transaction of isolation level "Repeatable Read" can not be modified from any other sessions till transaction is completed.

Assume that we have already created database and inserted some values in it.

Session 1 :

```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
```

```
BEGIN TRAN
```

```
SELECT * FROM emp WHERE ID IN(1,2)
```

```
WAITFOR DELAY '00:00:15'
```

```
SELECT * FROM Emp WHERE ID IN (1,2)
```

```
ROLLBACK
```

Session 2 :

```
UPDATE emp SET Salary=999 WHERE ID=1
```

If we run above two sessions concurrently, then Update command in session 2 will wait till session 1 transaction is completed because emp table row with ID=1 has locked in session1 transaction.

University Question

1. Write a note on – SQL facilities.

AU : May-14, Marks 8

3.17 Two Marks Questions with Answers

Q.1 What is a transaction ?

AU : May-04, Dec.05

Ans. : A transaction can be defined as a **group of tasks** that form a single logical unit.

Q.2 What does time to commit mean ?

AU : May-04

Ans. :

- The COMMIT command is used to save permanently any transaction to database.

- When we perform, Read or Write operations to the database then those changes can be undone by rollback operations. To make these changes permanent, we should make use of **commit**

Q.3 What are the various properties of transaction that the database system maintains to ensure integrity of data.

AU : Dec.-04

OR

Q.4 What are ACID properties ?

AU : May-05,06,08,13,15,Dec.-07,14,17

Ans. : In a database, each transaction should maintain ACID property to meet the consistency and integrity of the database. These are

- (1) Atomicity (2) Consistency (3) Isolation (4) Durability

Q.5 Give the meaning of the expression ACID transaction.

AU : Dec.-08

Ans. : The expression ACID transaction represents the transaction that follows the ACID Properties.

Q.6 State the atomicity property of a transaction.

AU : May-09,13

Ans. : 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.

Q.7 What is meant by concurrency control ?

AU : Dec.-15

Ans. : A mechanism which ensures that simultaneous execution of more than one transactions does not lead to any database inconsistencies is called **concurrency control mechanism**.

Q.8 State the need for concurrency control.

AU : Dec.-17

OR

Q.9 Why is it necessary to have control of concurrent execution of transactions ? How is it made possible ?

AU : Dec.-02

Ans. : Following are the purposes of concurrency control –

- o To ensure isolation :
- o To resolve read-write or write-write conflicts :
- o To preserve consistency of database :

Q.10 List commonly used concurrency control techniques.

AU : Dec.-11

Ans. : The commonly used concurrency control techniques are –

- i) Lock ii) Timestamp
- iii) Snapshot Isolation

Q.11 What is meant by serializability? How it is tested?**AU : May-14,18, Dec.-14,16**

Ans. : Serializability is a concept that helps to identify which non serial schedule and find the transaction equivalent to serial schedule.

It is tested using precedence graph technique.

Q.12 What is serializable schedule ?**AU : May-17**

Ans. : 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 T_1 and T_2

T_1	T_2
R(A)	
W(A)	
R(B)	
W(B)	
	R(A)
	W(A)
	R(B)
	W(B)

All the operations of transaction T_1 on data items A and then B executes and then in transaction T_2 all the operations on data items A and B execute. The R stands for Read operation and W stands for write operation.

Q.13 When are two schedules conflict equivalent ?**AU : Dec.-08**

Ans. : Two schedules are conflict equivalent if :

- They contain the same set of the transaction.
- every pair of conflicting actions is ordered the same way.

For example –

Non Serial Schedule

T1	T2
Read(A)	
Write(A)	
	Read(A)
	Write(A)
Read(B)	

Serial Schedule

T1	T2
Read(A)	
Write(A)	
Read(B)	
Write(B)	
	Read(A)

Write(B)			Write(A)
	Read(B)		Read(B)
	Write(B)		Write(B)

Schedule S2 is a serial schedule because, in this, all operations of T1 are performed before starting any operation of T2. Schedule S1 can be transformed into a serial schedule by swapping non-conflicting operations of S1.

Hence both of the above the schedules are conflict equivalent.

Q.14 Define two phase locking.

AU : May-13

Ans. : 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.

Q.15 What is the difference between shared lock and exclusive lock?

AU : May-18

Ans:

Shared Lock	Exclusive Lock
Shared lock is used for when the transaction wants to perform read operation.	Exclusive lock is used when the transaction wants to perform both read and write operation.
Multiple shared lock can be set on a transactions simultaneously.	Only one exclusive lock can be placed on a data item at a time.
Using shared lock data item can be viewed.	Using exclusive lock data can be inserted or deleted.

Q.16 What type of lock is needed for insert and delete operations.

AU : May-17

Ans. : The exclusive lock is needed to insert and delete operations.

Q.17 What benefit does strict two-phase locking provide ? What disadvantages result ?

AU : May-06, 07, Dec.-07

Ans. : Benefits :

1. This ensure that any data written by an uncommitted transaction are locked in exclusive mode until the transaction commits and preventing other transaction from reading that data .
2. This protocol solves dirty read problem.

Disadvantage:

1. Concurrency is reduced.

Q.18 What is rigorous two phase locking protocol ?**AU : Dec.-13**

Ans. : This is stricter two phase locking protocol. Here all locks are to be held until the transaction commits.

Q.19 Differentiate strict two phase locking and rigorous two phase locking protocol.**AU : May-16****Ans. :**

- In Strict two phase locking protocol all the exclusive mode locks be held until the transaction commits.
- The rigorous two phase locking protocol is stricter than strict two phase locking protocol. Here all locks are to be held until the transaction commits.

Q.20 Define deadlock.**AU : May-08,09,14**

Ans. : 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.

Q.21 List four conditions for deadlock.**AU : Dec.-16**

Ans. : 1. Mutual exclusion condition 2. Hold and wait condition 3.No preemption condition 4.Circular wait condition

Q.22 Why is recovery needed ?**AU : May-09****Ans. :**

- A recovery scheme that can **restore the database** to the consistent state that existed before the failure.
- Due to recovery mechanism, there is high availability of database to its users.



4

Implementation Techniques

Syllabus

RAID - File Organization - Organization of Records in Files - Indexing and Hashing - Ordered Indices - B+ tree Index Files - B tree Index Files - Static Hashing - Dynamic Hashing - Query Processing Overview - Algorithms for SELECT and JOIN operations - Query optimization using Heuristics and Cost Estimation.

Contents

4.1	RAID.....	May-07, 15, 16, 17, Dec.-06, 13, 14, 15, 16, Marks 16
4.2	File Organization	Dec.-08, Marks 10
4.3	Organization of Records in File	
4.4	Indexing and Hashing	Dec.-11,..... Marks 6
4.5	Ordered Indices.....	Dec.-04, 08, 15, May-06, Marks 10
4.6	B+ Tree Index Files.....	May-03, 06, 16, Dec.-17, ... Marks 16
4.7	B Tree Index Files.....	May-08, Dec.-12, 14, Marks 16
4.8	Hashing.....	Dec.- 04, 05, May-05, 14, Marks 8
4.9	Static Hashing.....	May-06, Marks 8
4.10	Dynamic Hashing.....	May-04, 07, 18, Dec.-08, 17 .. Marks 13
4.11	Query Processing Overview.....	May-14, 16, 18, Marks 16
4.12	Measure of Query Cost	
4.13	Algorithms for SELECT Operation	
4.14	Algorithms for JOIN Operation	
4.15	Query Optimization using Heuristics and Cost Estimation.....	Dec.-13, 16, May-15, Marks 16
4.16	Two Marks Questions with Answers	

Part I : File Organization and Indexing**4.1 RAID****AU : May-07,15,16,17, Dec.-06,13,14,15,16, Marks 16**

- RAID stands for Redundant Array of Independent Disks. This is a technology in which multiple secondary disks are connected together to increase the performance, data redundancy or both.
- For achieving the data redundancy - in case of disk failure, if the same data is also backed up onto another disk, we can retrieve the data and go on with the operation.
- It consists of an array of disks in which multiple disks are connected to achieve different goals.
- The **main advantage** of RAID, is the fact that, to the operating system the array of disks can be presented as a single disk.

Need for RAID

- RAID is a technology that is used to **increase the performance**.
- It is used for **increased reliability** of data storage.
- An array of multiple disks accessed in parallel will give greater throughput than a single disk.
- With multiple disks and a suitable redundancy scheme, your system can stay up and running when a disk fails, and even while the replacement disk is being installed and its **data restored**.

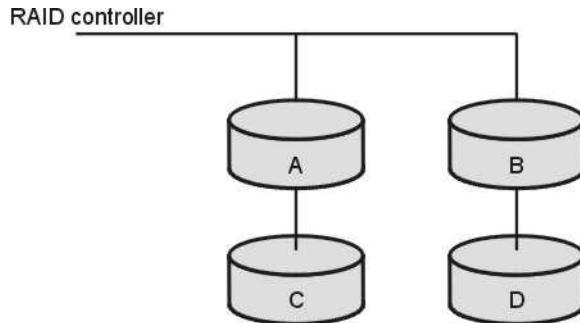
Features

- (1) RAID is a technology that contains the set of physical disk drives.
- (2) In this technology, the operating system views the separate disks as a single logical disk.
- (3) The data is distributed across the physical drives of the array.
- (4) In case of disk failure, the parity information can be used to recover the data.

4.1.1 RAID Levels**Level : RAID 0**

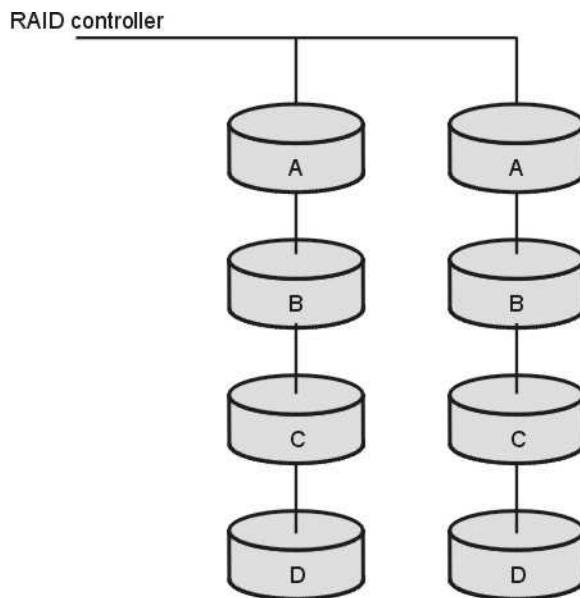
- In this level, data is broken down into blocks and these blocks are stored across all the disks.
- Thus **striped array of disks** is implemented in this level. For instance in the following figure blocks "A B" form a stripe.

- There is no duplication of data in this level so once a block is lost then there is no way recover it.
- The main **priority of this level is performance and not the reliability.**



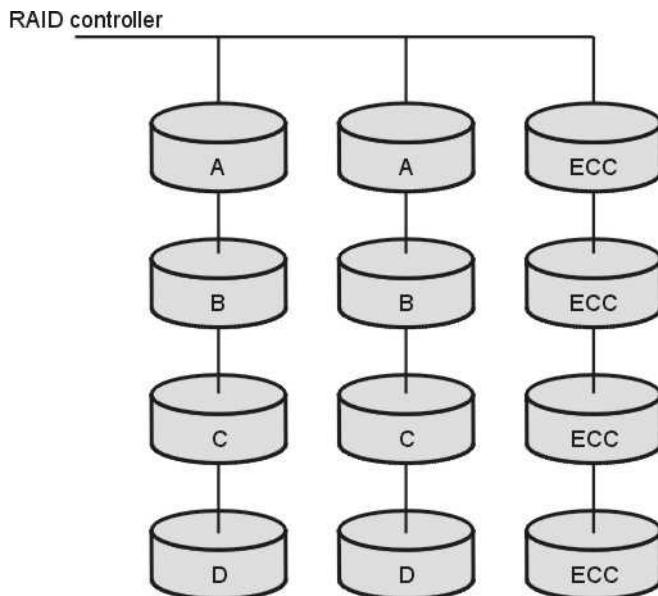
Level : RAID 1

- This level makes use of **mirroring**. That means all data in the drive is duplicated to another drive.
- This level provides 100% redundancy in case of failure.
- Only half space of the drive is used to store the data. The other half of drive is just a mirror to the already stored data.
- The main advantage of this level is fault tolerance. If some disk fails then the other automatically takes care of lost data.



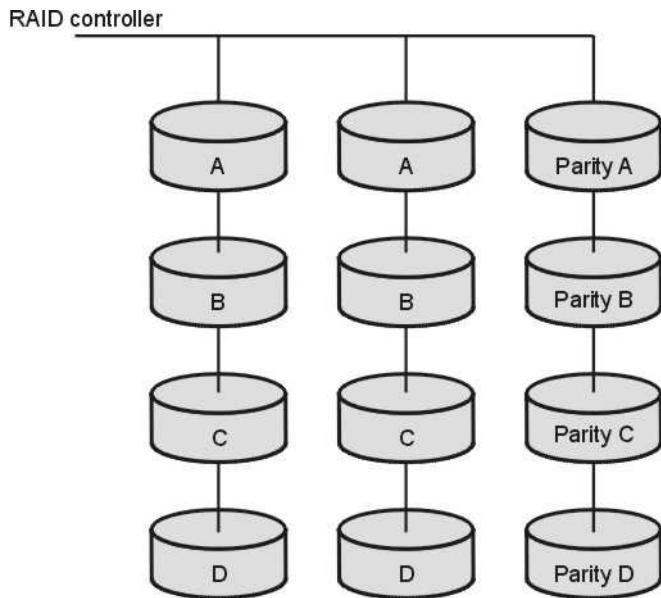
Level : RAID 2

- This level makes use of **mirroring as well as stores Error Correcting Codes (ECC)** for its data striped on different disks.
- The data is stored in separate set of disks and ECC is stored another set of disks.
- This level has a complex structure and high cost. Hence it is not used commercially.

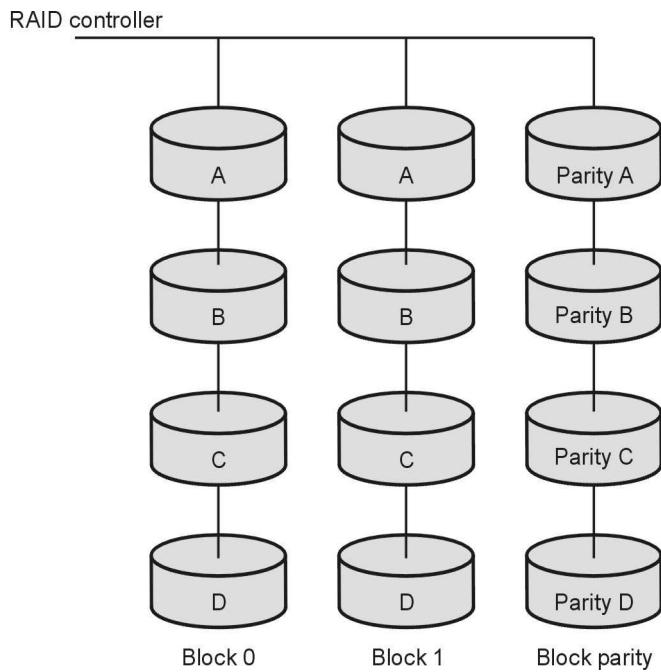


Level : RAID 3

- This level consists of byte-level stripping with dedicated parity. In this level, the parity information is stored for each disk section and written to a dedicated parity drive.
- We can detect single errors with a **parity bit**. Parity is a technique that checks whether data has been lost or written over when it is moved from one place in storage to another.
- In case of disk failure, the parity disk is accessed and data is reconstructed from the remaining devices. Once the failed disk is replaced, the missing data can be restored on the new disk.

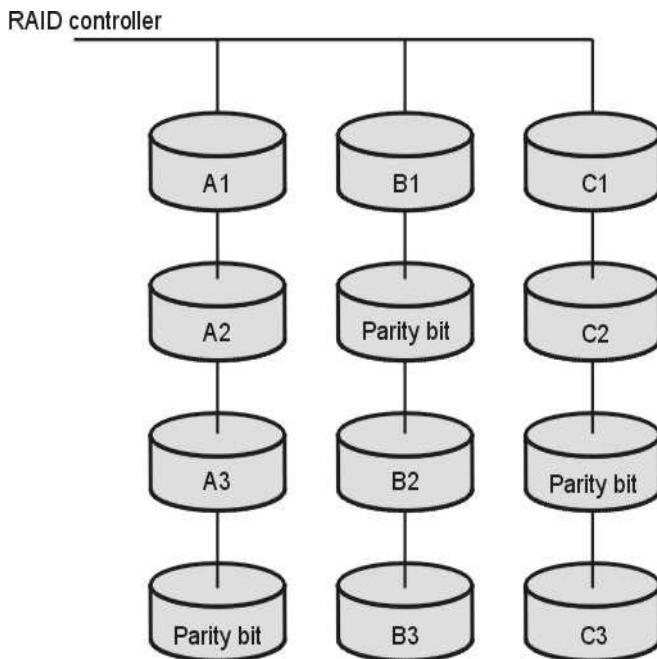
**Level : RAID 4**

- o RAID 4 consists of block-level stripping with a parity disk.
- o Note that level 3 uses byte-level striping, whereas level 4 uses block-level striping.



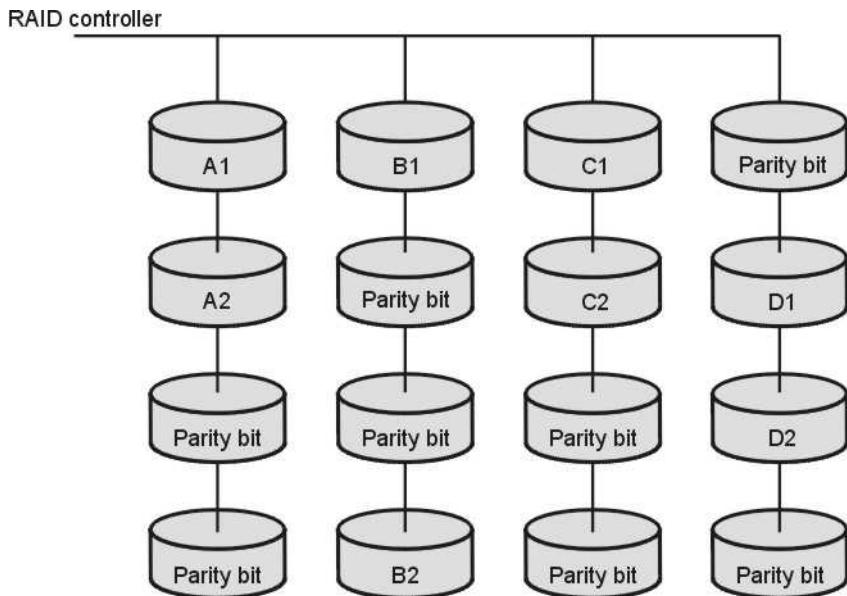
Level : RAID 5

- RAID 5 is a modification of RAID 4.
- RAID 5 writes whole data blocks onto different disks, but the **parity bits** generated for data block stripe are **distributed among all the data disks** rather than storing them on a different dedicated disk.



RAID : Level 6

- RAID 6 is a extension of Level 5
- RAID 6 writes whole data blocks onto different disks, but the **two independent parity bits** generated for data block stripe are **distributed among all the data disks** rather than storing them on a different dedicated disk.
- Two parities provide **additional fault tolerance**.
- This level requires **at least four disks** to implement RAID.



The factors to be taken into account in choosing a RAID level are :

Monetary cost of extra disk-storage requirements.

1. Performance requirements in terms of number of I/O operations.
2. Performance when a disk has failed.
3. Performance during rebuild

University Questions

1. Explain what a RAID system is ? How does it improve performance and reliability. Discuss the level 3 and level 4 of RAID. **AU : May-17, Marks (3+4+6)**

2. Explain the concept of RAID. **AU : Dec.-16, Marks 6, Dec.-14, Marks 8**

3. Briefly explain RAID and RAID levels.

AU : Dec.-06, Marks 10, Dec.-13, May-15, 16, Marks 16

4. What is RAID ? List the different levels in RAID technology and explain its features.

AU : May-07, Dec.-15, Marks 8

4.2 File Organization

AU : Dec.-08, Marks 10

- A file organization is a method of **arranging records** in a file when the file is stored on disk.
- A file is organized logically as a sequence of records.
- Record is a sequence of fields.
- There are two types of records used in file organization (1) Fixed Length Record
(2) Variable Length Record.

(1) Fixed Length Record

- A file where each records is of the same length is said to have fixed length records.
- Some fields are always the same length (e.g. PhoneNumber is always 10 characters).
- Some fields may need to be 'padded out' so they are the correct length.
- For example -

```
type Employee = record
    EmpNo varchar(4)
    Ename varchar(10)
    Salary integer(5)
    Phone varchar(10)
End
```

EmpNo	EName	Salary	Phone
1111	AAA	1000	1111111111
2222	BBB	2000	2222222222
3333	CCC	3000	3333333333
4444	DDD	4000	4444444444

For instance the first record of example file can be stored as

1	1	1	1	A	A	A										1
0	0	0		1	1	1	1	1	1	1	1	1	1	1		

Thus total 29 bytes are required to store.

Advantage :

- Access is fast because the computer knows where each record starts.

Disadvantage :

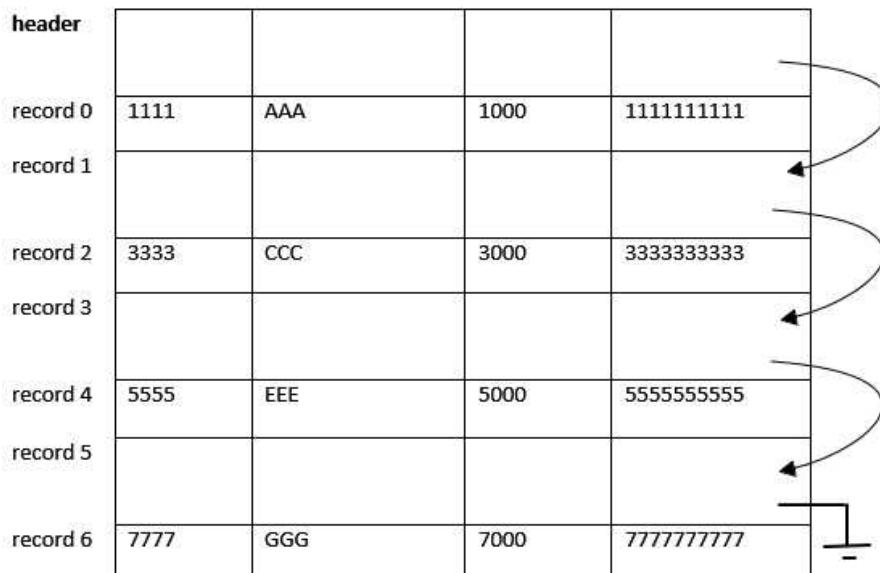
- (1) Due to fixed size, some larger sized record may cross the block boundaries. That means part of record will be stored in one block and other part of the record may be stored in some another block. Thus we may require two block access for each read or write.
- (2) It is difficult to delete the record from this structure. If some intermediate record is deleted from the file then the vacant space must be occupied by next subsequent records.
- When a record is deleted, we could move the record that came after it into the space formerly occupied by the deleted record. But this may require moving of multiple records to occupy the vacant space. This is an undesirable solution to fill up the vacant space of deleted records.

- Another approach is to use a **file header**. At the beginning of the file, we allocate a certain number of bytes as a file header. The header will contain information such as address of the first record whose contents are deleted. We use this first record to store the address of the second available record and so on. Thus the stored record addresses are referred as **pointer** while the deleted records thus form a linked list which is called as **free list**.

For example - Consider the employee record in a file is -

	Emp No	Ename	Salary	Phone
record 0	1111	AAA	1000	1111111111
record 1	2222	BBB	2000	2222222222
record 2	3333	CCC	3000	3333333333
record 3	4444	DDD	4000	4444444444
record 4	5555	EEE	5000	5555555555
record 5	6666	FFF	6000	6666666666
record 6	7777	GGG	7000	7777777777

The representation of records maintaining free list after deletion of record 1,3 and 5



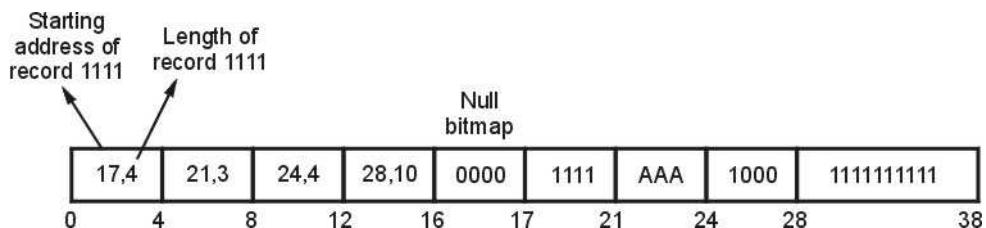
- On insertion of a new record, we use the record pointed to by the header. We change the header pointer to point to the next available record. If no space is available, we add the new record to the end of the file.

(2) Variable Length Record

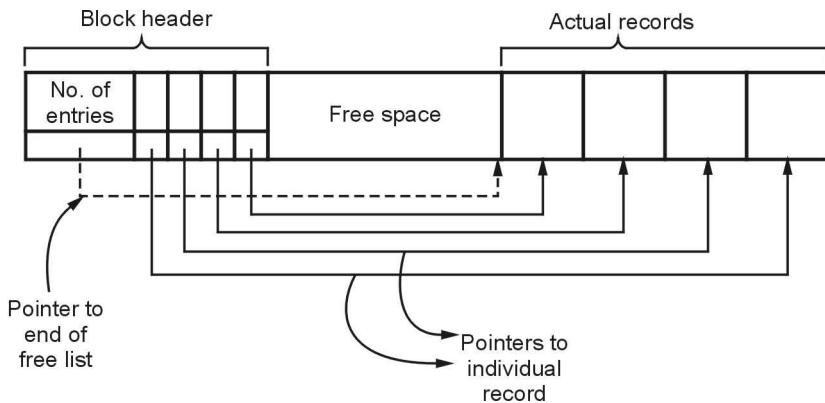
- Variable-length records arise in a database in several ways:
 - (i) Storage of multiple record types in a file.
 - (ii) Record types that allow variable lengths for one or more fields.
 - (iii) Record types that allow repeating fields such as arrays or multisets.
- The representation of a record with variable-length attributes typically has **two parts** :
 - a) an initial part with **fixed length attributes**. This initial part of the record is represented by a pair (offset, length). The offset denotes the starting address of the record while the length represents the actual length of the length.
 - b) followed by data for variable length attributes.
- For example - Consider the employee records stored in a file as

record 0	1111	AAA	1000	1111111111
record 1	2222	BBB	2000	2222222222
record 2	3333	CCC	3000	3333333333
record 3	4444	DDD	4000	4444444444
record 4	5555	EEE	5000	5555555555
record 5	6666	FFF	6000	6666666666
record 6	7777	GGG	7000	7777777777

- The variable length representation of first record is,



- The figure also illustrates the use of a null bitmap, which indicates which attributes of the record have a null value.
- The variable length record can be stored in blocks. A specialized structure called **slotted page structure** is commonly used for organizing the records within a block. This structure is as shown by following Fig. 4.2.1.

**Fig. 4.2.1**

- This structure can be described as follows -
 - At the beginning of each block there is a block header which contains -
 - Total number of entries (i.e. records).
 - Pointer to end of free list.
 - Followed by an array of entries that contain size and location of each record.
 - The actual records are stored in contiguous block. Similarly the free list is also maintained as continuous block.
 - When a new record is inserted then the space is allocated from the block of free list.
 - Records can be moved around within a page to keep them contiguous with no empty space between them; entry in the header must be updated.

University Question

1. Describe different types of file organization. Explain using a sketch of each of the, with their advantages and disadvantages.

AU : Dec.-08, Marks 10

4.3 Organization of Records in File

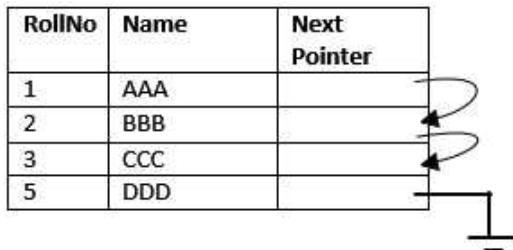
There are three commonly used approaches of organizing records in file -

- (1) **Heap File Organization** : Any record can be placed anywhere in the file where there is a space. There is no ordering for placing the records in the file. Generally single file is used.
- (2) **Sequential File Organization** : Records are stored in sequential order based on the value of search key.

- (3) Hashing File Organization :** A hash function is used to obtain the location of the record in the file. Based on the value returned by hash function, the record is stored in the file.

4.3.1 Sequential File Organization

- The sequential file organization is a simple file organization method in which the records are stored based on the search key value.
- For example – Consider following set of records stored according to the RollNo of student. Note that we assume here that the RollNo is a search key.



Now if we want to insert following record

4	EEE
---	-----

Then, we will insert it in sorted order of RollNo and adjust the pointers accordingly.

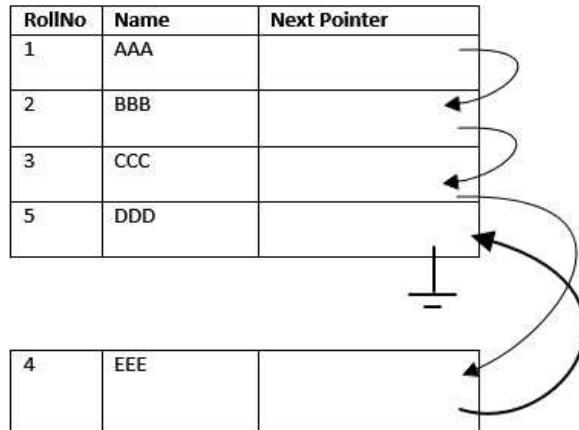


Fig. 4.3.1

- However, maintaining physical sequential order is very difficult as there can be several insertions and deletions.
- We can maintain the deletion by next pointer chain.
- For insertion following rules can be applied -
 - If there is free space insert record there.

- o If no free space, insert the record in an overflow block.
- o In either case, pointer chain must be updated.

4.3.2 Multi-table Clustering File Organization

In a multitable clustering file organization, records of several different relations are stored in the same file.

For example - Following two tables **Student** and **Course**

Sname	Marks
Ankita	55
Ankita	67
Ankita	86
Prajkta	91

Sname	Cname	City
Ankita	ComputerSci	Chennai
Prajkta	Electronics	Pune

The multitable clustering organization for above tables is,

Ankita	ComputerSci	Chennai
Ankita	55	
Ankita	67	
Ankita	86	
Prajkta	Electronics	Pune
Prajkta	91	

This type of file organization is good for join operations such as Student \bowtie Course. This file organization results in variable size records.

The pointer chain can be added to the above records to keep track of address of next record. It can be as shown in following Fig. 4.3.2

Ankita	ComputeSci	Chennai	
Ankita	55		
Ankita	67		
Ankita	86		
Prajkta	Electronics	Pune	
Prajkta	91		

Fig. 4.3.2

4.4 Indexing and Hashing

AU : Dec.-11, Marks 10

- An index is a **data structure** that **organizes data records** on the disk to make the **retrieval of data efficient**.
- The search key for an index is collection of one or more fields of records using which we can efficiently retrieve the data that satisfy the search conditions.
- The indexes are required to speed up the search operations on file of records.
- There are two types of indices -
 - **Ordered Indices** : This type of indexing is based on sorted ordering values.
 - **Hash Indices** : This type of indexing is based on uniform distribution of values across range of buckets. The address of bucket is obtained using the hash function.
- There are several techniques of for using indexing and hashing. These techniques are evaluated based on following factors -
 - **Access Types** : It supports various types of access that are supported efficiently.
 - **Access Time** : It denotes the time it takes to find a particular data item or set items.
 - **Insertion Time** : It represents the time required to insert new data item.
 - **Deletion Time** : It represents the time required to delete the desired data item.
 - **Space overhead** : The space is required to occupy the index structure. But allocating such extra space is worth to achieve improved performance.

Example 4.4.1 Since indices speed query processing. Why might they not be kept on several search keys? List as many reasons as possible.

AU : Dec.-11, Marks 6

Solution : Reasons for not keeping several search indices include :

- a. Every index requires additional CPU time and disk I/O overhead during inserts and deletions.
- b. Indices on non-primary keys might have to be changed on updates, although an index on the primary key might not (as updates typically do not modify the primary key attributes).
- c. Each extra index requires additional storage space.

- d. For queries which involve conditions on several search keys, efficiency might not be bad even if only some of the keys have indices on them.

Therefore database performance is improved less by adding indices when many indices already exist.

4.5 Ordered Indices

AU : Dec.-04, 08, 15, May-06, Marks 10

4.5.1 Primary and Clustered Indices

Primary Index :

- An index on a set of fields that includes the primary key is called a primary index. The primary index file should be always in sorted order.
- The primary indexing is always done when the data file is arranged in sorted order and primary indexing contains the primary key as its search key.
- Consider following scenario in which the primary index consists of few entries as compared to actual data file.

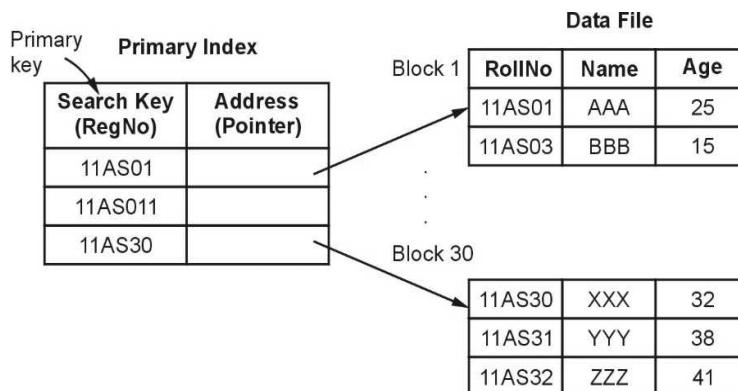


Fig. 4.5.1 : Example of primary index

- Once if you are able to locate the first entry of the record containing block, other entries are stored continuously. For example if we want to search a record for RegNo 11AS32 we need not have to search for the entire data file. With the help of primary index structure we come to know the location of the record containing the RegNo 11AS30, now when the first entry of block 30 is located, then we can easily locate the entry for 11AS32.
- We can apply binary search technique. Suppose there are $n = 300$ blocks in a main data file then the number of accesses required to search the data file will be $\log_2 n + 1 = (\log_2 300) + 1 \approx 9$
- If we use primary index file which contains at the most $n = 3$ blocks then using binary search technique, the number of accesses required to search using the primary index file will be $\log_2 n + 1 = (\log_2 3) + 1 \approx 3$

- This shows that using primary index the access time can be deduced to great extent.

Clustered Index :

- In some cases, the index is created on non-primary key columns which may not be unique for each record. In such cases, in order to identify the records faster, we will group two or more columns together to get the unique values and create index out of them. This method is known as **clustering index**.
- When a file is organized so that the ordering of data records is the same as the ordering of data entries in some index then say that index is **clustered**, otherwise it is an **unclustered index**.
- Note that, the data file need to be in sorted order.
- Basically, records with similar characteristics are grouped together and indexes are created for these groups.
- For example**, students studying in each semester are grouped together. i.e.; 1st semester students, 2nd semester students, 3rd semester students etc. are grouped.

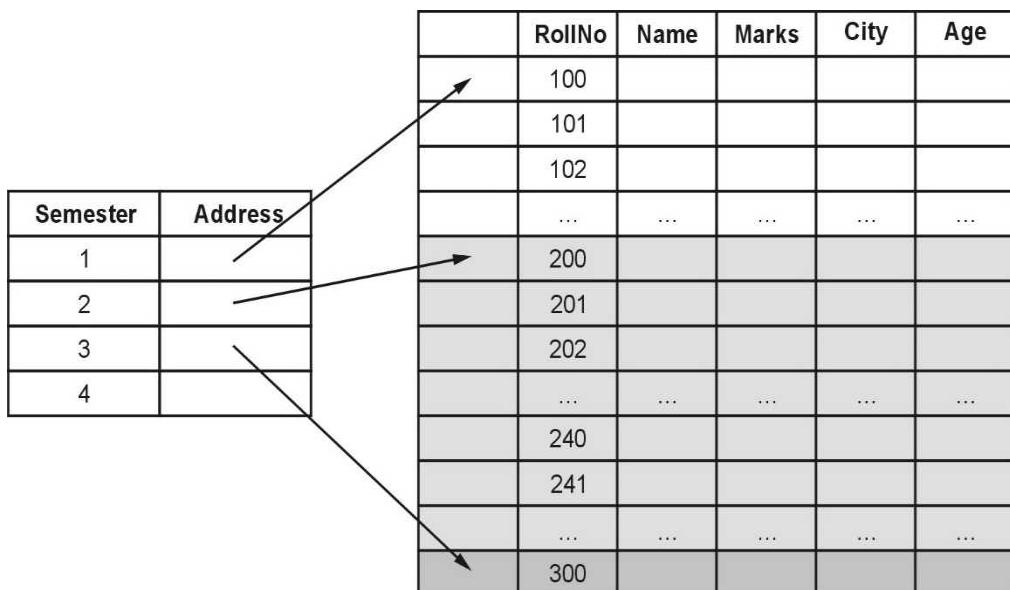


Fig. 4.5.2 Clustered Index

4.5.2 Dense and Sparse Indices

There are two types of ordered indices :

1) Dense index :

- An index record appears for every search key value in file.
- This record contains search key value and a pointer to the actual record.

- For example :



Fig. 4.5.3 : Dense index

2) Sparse index :

- Index records are created only for some of the records.
- To locate a record, we find the index record with the largest search key value less than or equal to the search key value we are looking for.
- We start at that record pointed to by the index record, and proceed along the pointers in the file (that is, sequentially) until we find the desired record.
- For example -

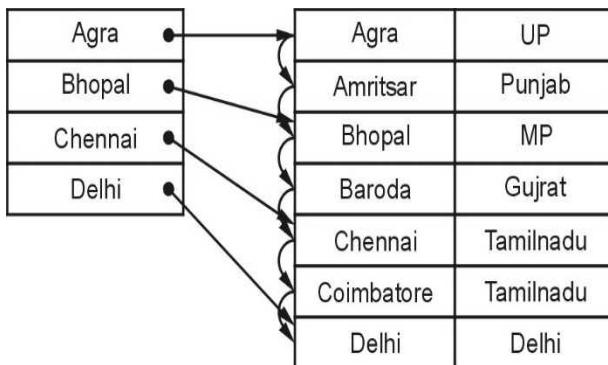


Fig. 4.5.4 : Sparse index

4.5.3 Single and Multilevel Indices

Single level indexing :

- A single-level index is an auxiliary file that makes it more efficient to search for a record in the data file.
- The index is usually specified on one field of the file (although it could be specified on several fields).

- Each index can be in the following form.

Search Key	Pointer to Record
------------	-------------------

- The index file usually occupies considerably less disk blocks than the data file because its entries are much smaller.
- A binary search on the index yields a pointer to the file record.
- The types of single level indexing can be primary indexing, clustering index or secondary indexing.
- Example : Following Fig. 4.5.5 represents the single level indexing -

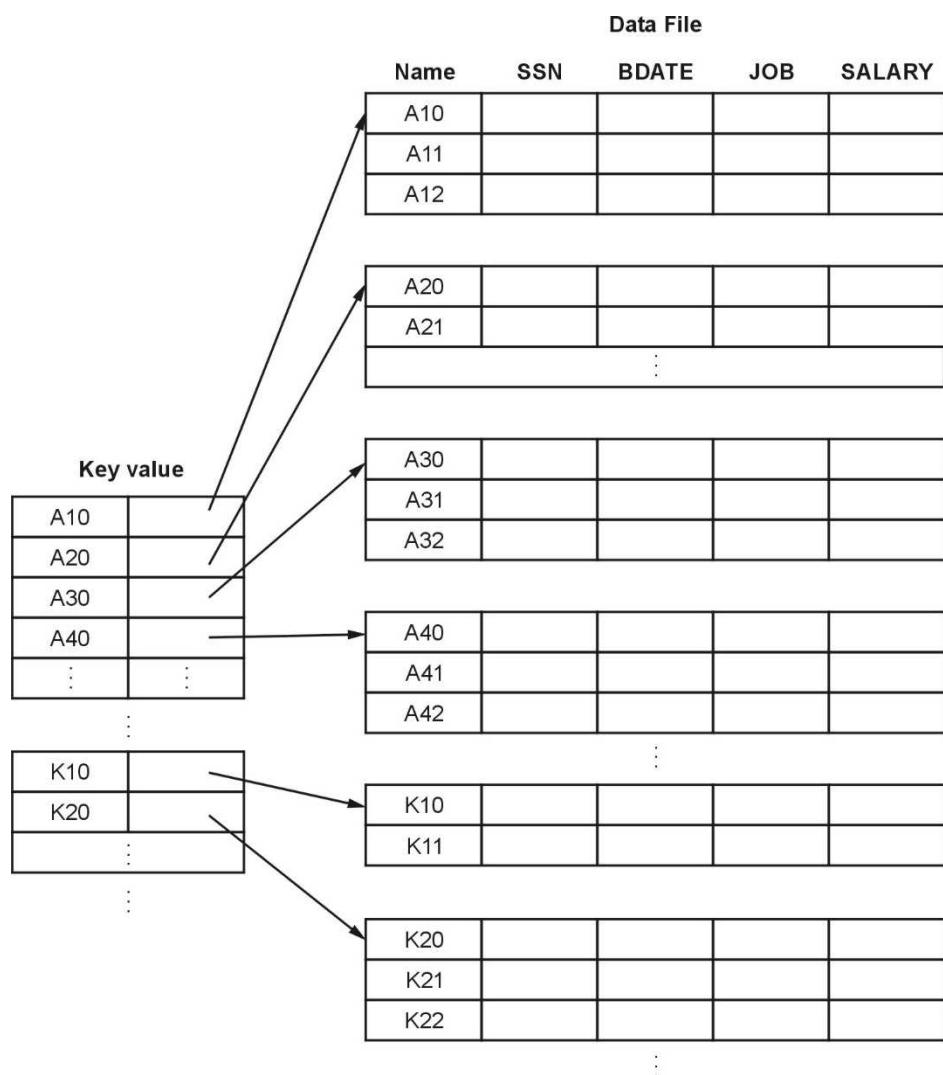


Fig. 4.5.5 : Single level indexing

Multilevel indexing :

- There is an immense need to keep the index records in the main memory so as to speed up the search operations. If single-level index is used, then a large size index cannot be kept in memory which leads to multiple disk accesses.
- Multi-level Index helps in breaking down the index into several smaller indices in order to make the outermost level so small that it can be saved in a single disk block, which can easily be accommodated anywhere in the main memory.
- The multilevel indexing can be represented by following Fig. 4.5.6

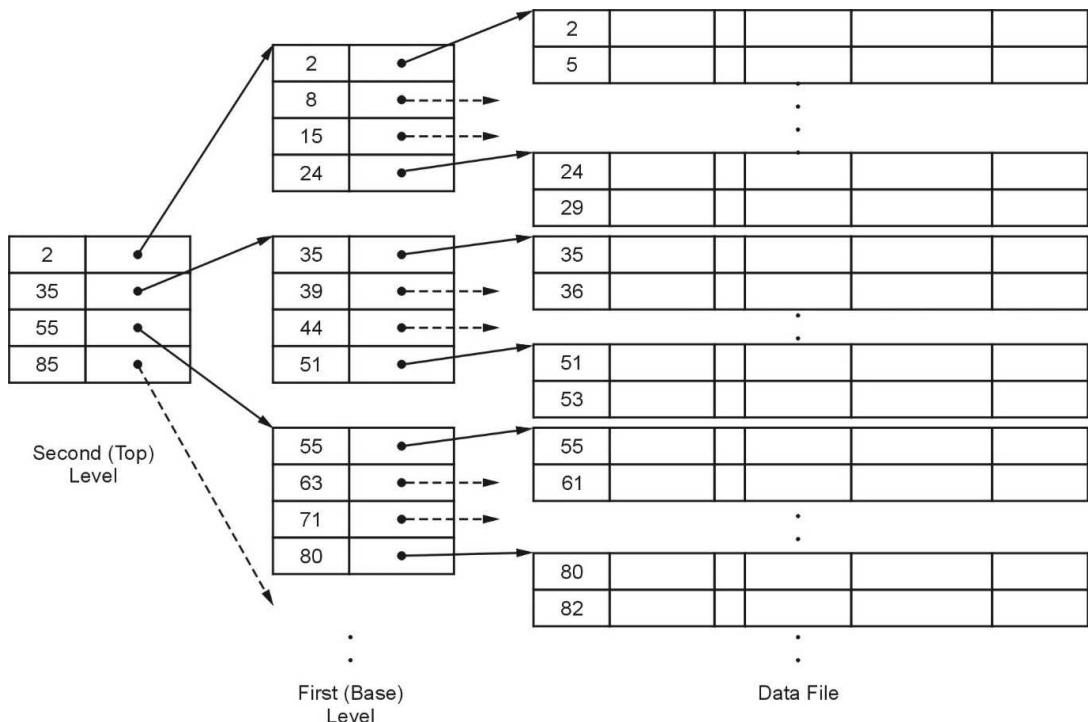


Fig. 4.5.6 Multilevel indexing

4.5.4 Secondary Indices

- In this technique two levels of indexing are used in order to reduce the mapping size of the first level and in general.
- Initially, for the first level, a large range of numbers is selected so that the mapping size is small. Further, each range is divided into further sub ranges.
- It is used to optimize the query processing and access records in a database with some information other than the usual search key.

- For example -

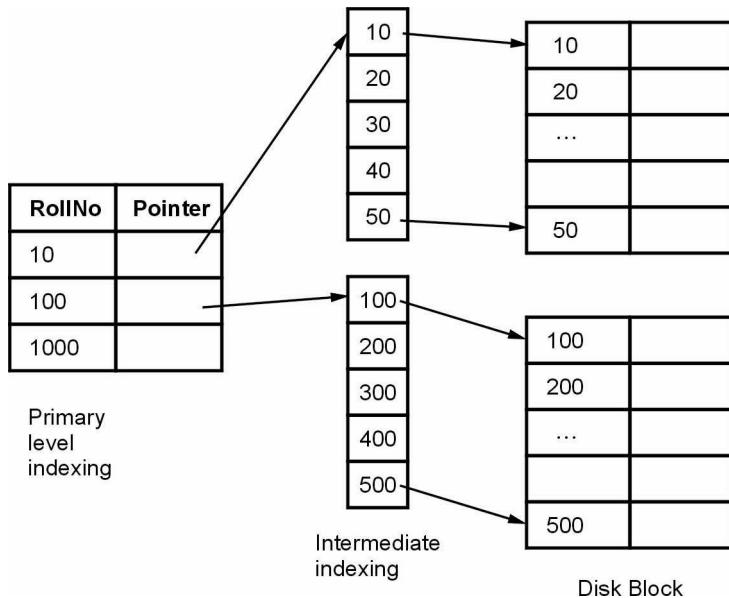


Fig. 4.5.7 Secondary Indexing

University Questions

1 Illustrate indexing and hashing techniques with suitable examples.

AU : Dec.-15, Marks 8

[Note: For hashing technique : Refer section 4.8]

2. What is the use of an index structure and explain the concept of ordered indices. AU : Dec 04, Marks 8

3. What is the difference between primary index and secondary index ?

AU : May 06, Marks 8

4 Explain the index schemas used in database systems

AU : Dec.- 08, Marks 10

4.6 B+ Tree Index Files

AU : May-03, 06,16, Dec.-17, Marks 16

- The B+ tree is similar to binary search tree. It is a balanced tree in which the internal nodes direct the search.
- The leaf nodes of B+ trees contain the **data entries**.

Structure of B+ Tree

- The typical node structure of B+ node is as follows -

P ₁	K ₁	P ₂	K ₂	...	P _{n-1}	K _{n-1}	P _n
----------------	----------------	----------------	----------------	-----	------------------	------------------	----------------

- It contains up to n - 1 search-key values K₁, K₂, ..., K_{n-1}, and n pointers P₁, P₂, ..., P_n.
- The search-key values within a node are kept in sorted order; thus, if i < j, then K_i < K_j.

- To retrieve all the leaf pages efficiently we have to link them using **page pointers**. The sequence of leaf pages is also called as **sequence set**.
- Following Fig. 4.6.1 represents the example of B+ tree.

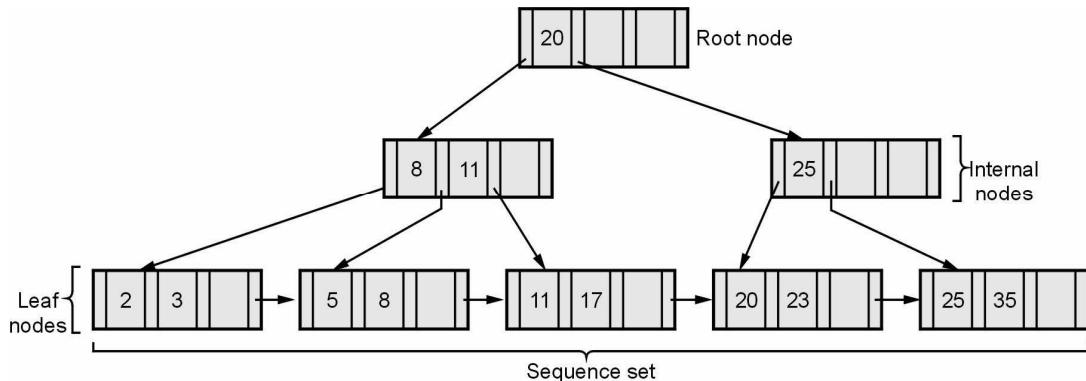


Fig. 4.6.1 B+ Tree

- The B+ tree is called dynamic tree because the tree structure can grow on insertion of records and shrink on deletion of records.

Characteristics of B+ Tree

Following are the characteristics of B+ tree.

- 1) The B+ tree is a balanced tree and the operations insertions and deletion keeps the tree balanced.
- 2) A minimum occupancy of 50 percent is guaranteed for each node except the root.
- 3) Searching for a record requires just traversal from the root to appropriate leaf.

4.6.1 Insertion Operation

Algorithm for insertion :

Step 1 : Find correct leaf L.

Step 2 : Put data entry onto L.

- i) If L has enough space, done!
- ii) Else, must split L (into L and a new node L2)
 - Allocate new node
 - Redistribute entries evenly
 - Copy up middle key.
 - Insert index entry pointing to L2 into parent of L.

Step 3 : This can happen recursively

- i) To split index node, redistribute entries evenly, but push up middle key. (Contrast with leaf splits.)

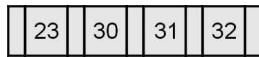
Step 4 : Splits “grow” tree; root split increases height.

- i) Tree growth: gets wider or one level taller at top.

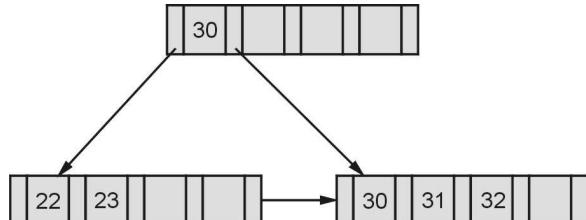
Example 4.6.1 Construct B+ tree for following data. 30,31,23,32,22,28,24,29, where number of pointers that fit in one node are 5.

Solution : In B+ tree each node is allowed to have the number of pointers to be 5. That means at the most 4 key values are allowed in each node.

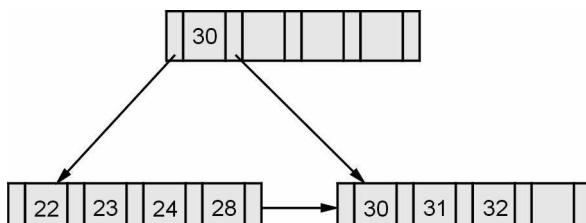
Step 1 : Insert 30,31,23,32. We insert the key values in ascending order.



Step 2 : Now if we insert 22, the sequence will be 22 , 23, 30, 31, 32. The middle key 30, will go up.

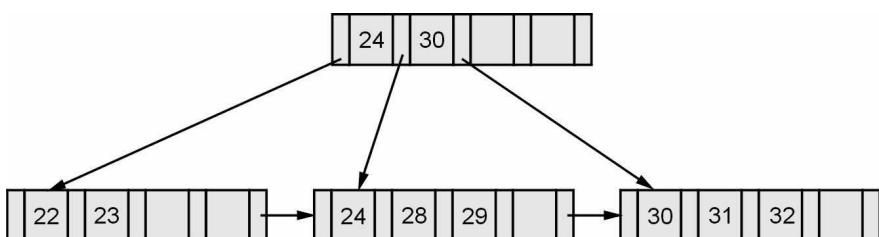


Step 3 : Insert 28,24. The insertion is in ascending order.



Step 4 : Insert 29. The sequence becomes 22, 23, 24, 28, 29. The middle key 24 will go up.

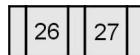
Thus we get the B+ tree.



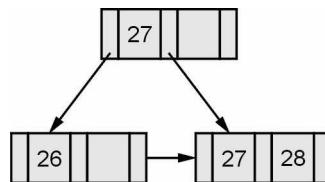
Example 4.6.2 Construct B+ tree to insert the following (order of the tree is 3) 26,27,28,3,4,7,9,46,48,51,2,6

Solution : Order means maximum number of children allowed by each node. Hence order 3 means at the most 2 key values are allowed in each node.

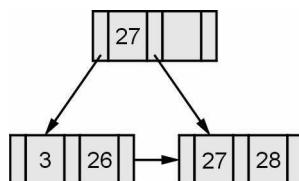
Step 1 : Insert 26, 27 in ascending order



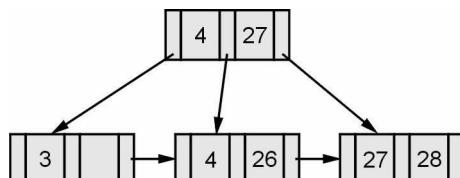
Step 2 : Now insert 28. The sequence becomes 26,27,28. As the capacity of the node is full, 27 will go up. The B+ tree will be,



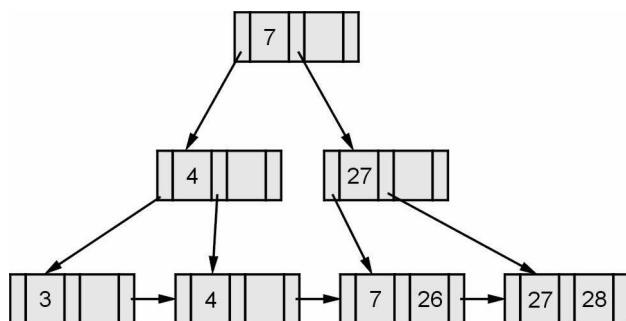
Step 3 : Insert 3. The partial B+ Tree will be,



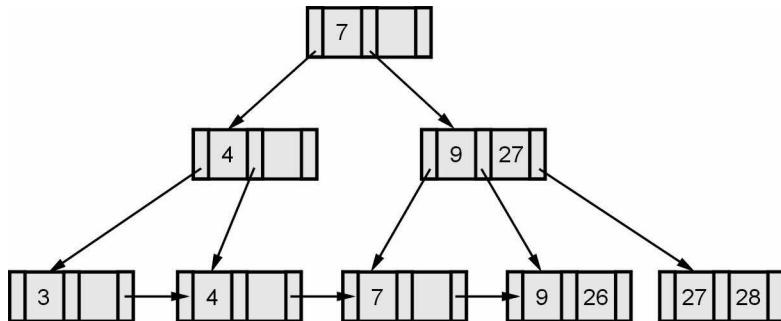
Step 4 : Insert 4. The sequence becomes 3,4, 26. The 4 will go up. The partial B+ tree will be -



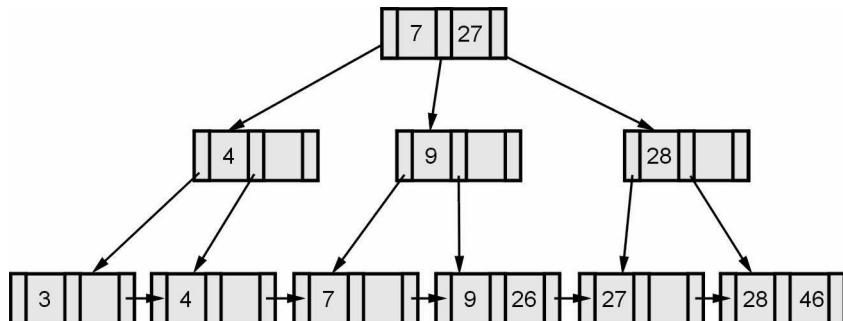
Step 5 : Insert 7. The sequence becomes 4,7,26. The 7 will go up. Again from 4,7,27 the 7 will go up. The partial B+ Tree will be,



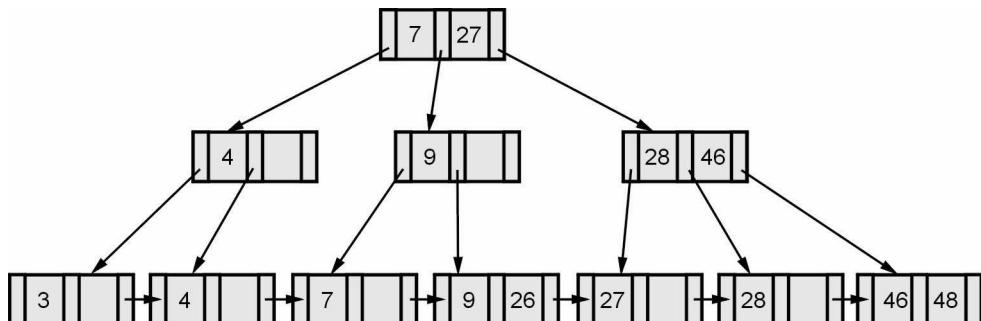
Step 6 : Insert 9. By inserting 7,9,26 will be the sequence. The 9 will go up. The partial B+ tree will be,



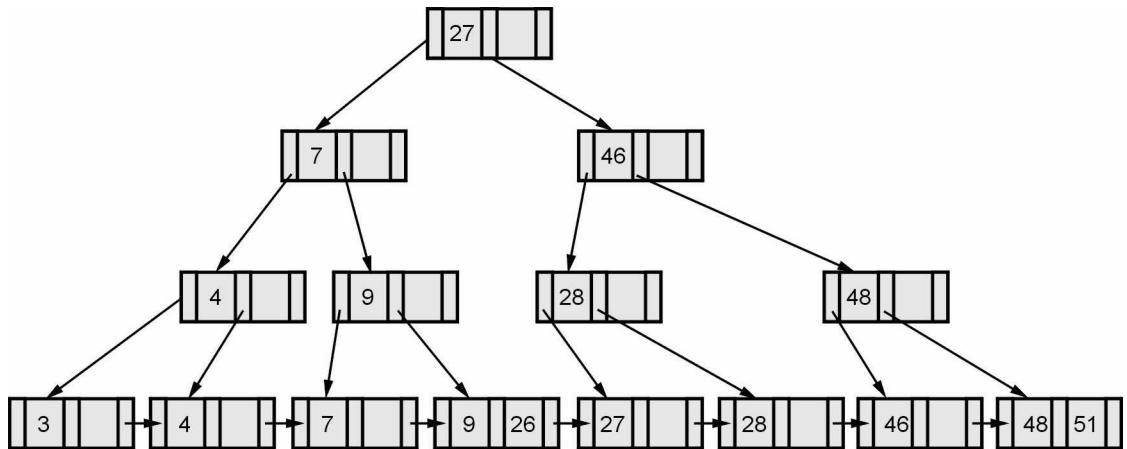
Step 7 : Insert 46. The sequence becomes 27,28,46. The 28 will go up. Now the sequence becomes 9, 27, 28. The 27 will go up and join 7. The B+ Tree will be,



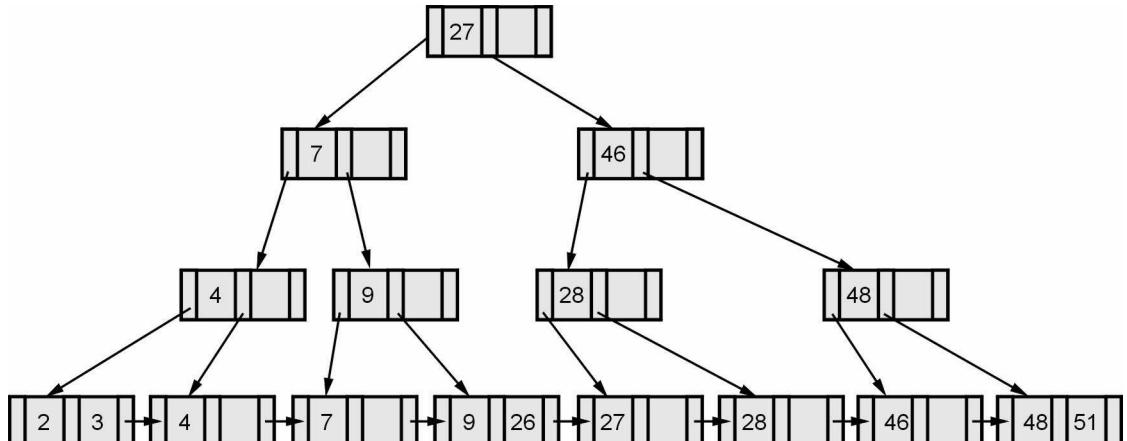
Step 8 : Insert 48. The sequence becomes 28,46,48. The 46 will go up. The B+ Tree will become,



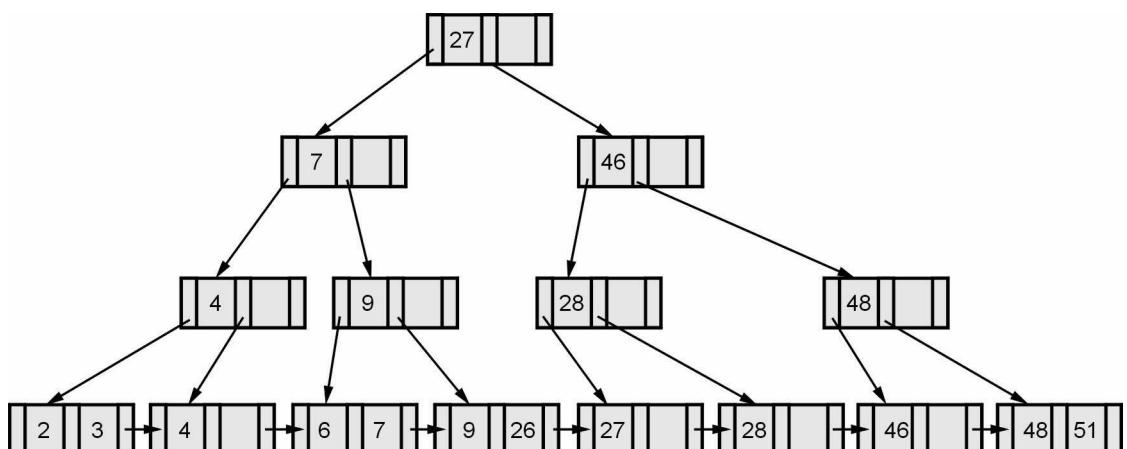
Step 9 : Insert 51. The sequence becomes 46,48,51. The 48 will go up. Then the sequence becomes 28, 46, 48. Again the 46 will go up. Now the sequence becomes 7,27, 46. Now the 27 will go up. Thus the B+ tree will be



Step 10 : Insert 2. The insertion is simple. The B+ tree will be,



Step 11 : Insert 6. The insertion can be made in a vacant node of 7(the leaf node). The final B+ tree will be,



4.6.2 Deletion Operation

Algorithm for deletion :

Step 1 : Start at root, find leaf L with entry, if it exists.

Step 2 : Remove the entry.

- i) If L is at least half-full, done!
- ii) If L has only d-1 entries,

- Try to **re-distribute**, borrowing keys from sibling.

(adjacent node with same parent as L).

- If redistribution fails, **merge L and sibling**.

Step 3 : If merge occurred, must delete entry (pointing to L or sibling) from parent of L.

Step 4 : Merge could propagate to root, decreasing height.

Example 4.6.3 Construct B+ Tree for the following set of key values (2,3,5,7,11,17,19,23,29,31). Assume that the tree is initially empty and values are added in ascending order. Construct B+ tree for the cases where the number of pointers that fit one node is four. After creation of B+ tree perform following series of operations :

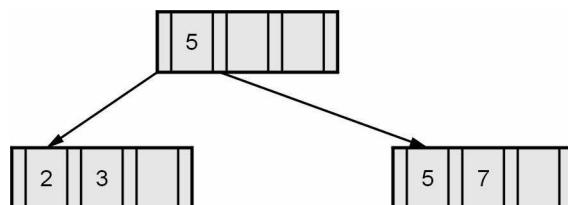
(a) Insert 9. (b) Insert 10. (c) Insert 8. (d) Delete 23. (e) Delete 19.

Solution : The number of pointers fitting in one node is four. That means each node contains at the most three key values.

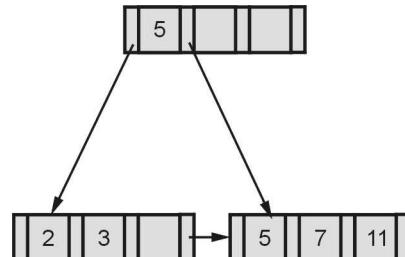
Step 1 : Insert 2, 3, 5.



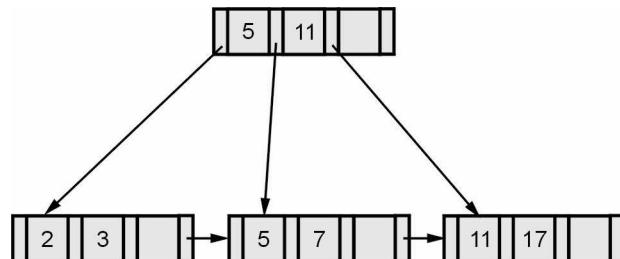
Step 2 : If we insert 7, the sequence becomes 2, 3, 5, 7. Since each node can accommodate at the most three key, the 5 will go up, from the sequence 2, 3, ⑤, 7.



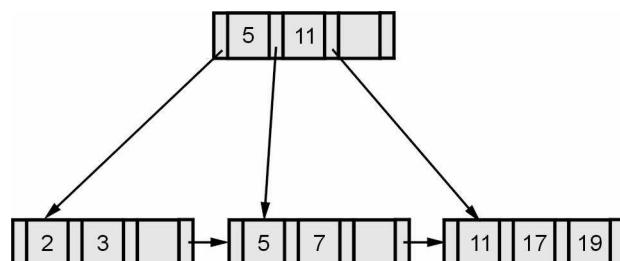
Step 3 : Insert 11. The partial B+ tree will be,



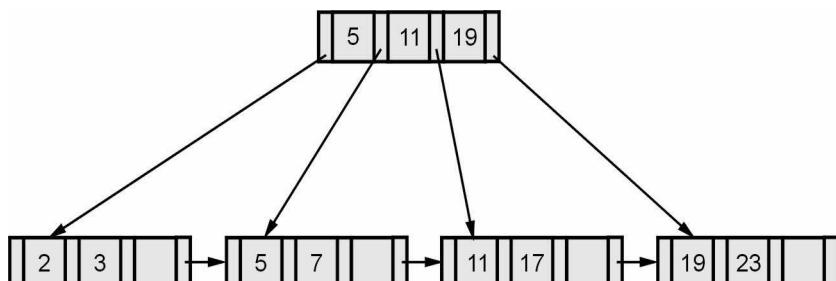
Step 4 : Insert 17. The sequence becomes 5,7, 11,17. The element 11 will go up. Then the partial B+ tree becomes,



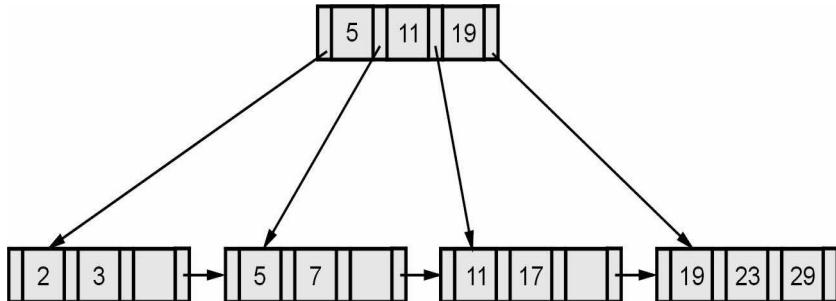
Step 5 : Insert 19.



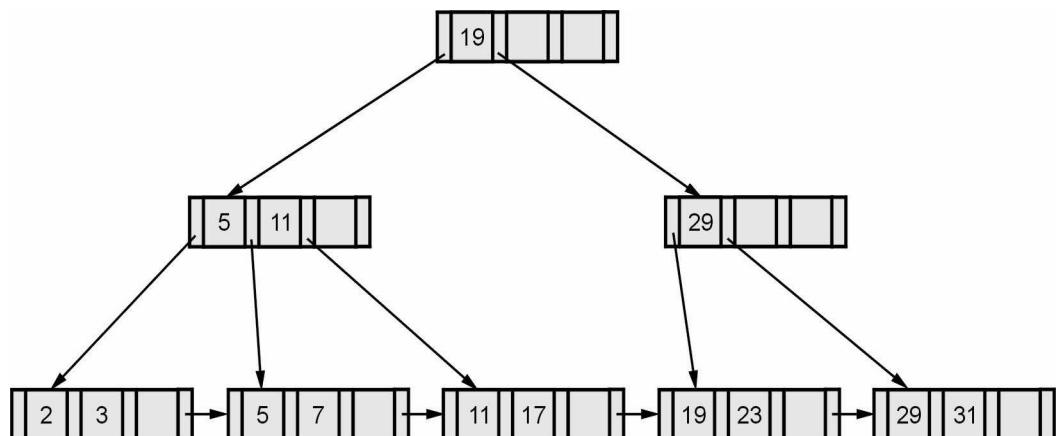
Step 6 : Insert 23. The sequence becomes 11,17,19,23. The 19 will go up.



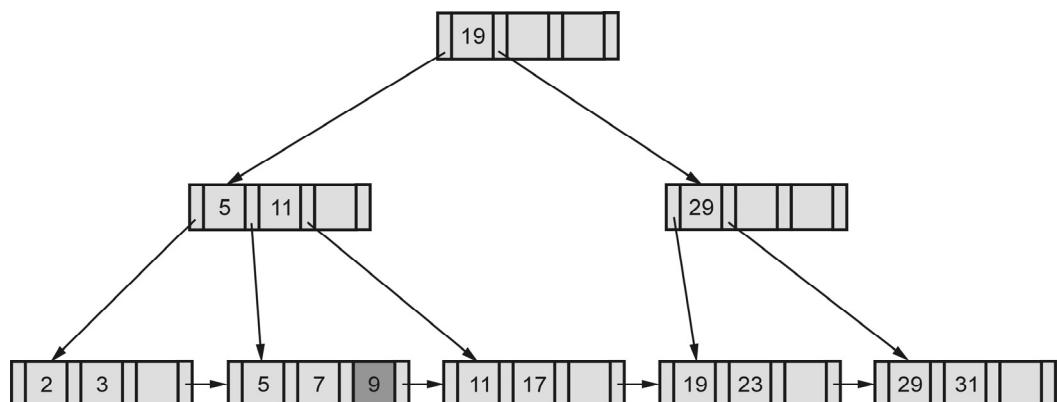
Step 7 : Insert 29. The partial B+ tree will be,



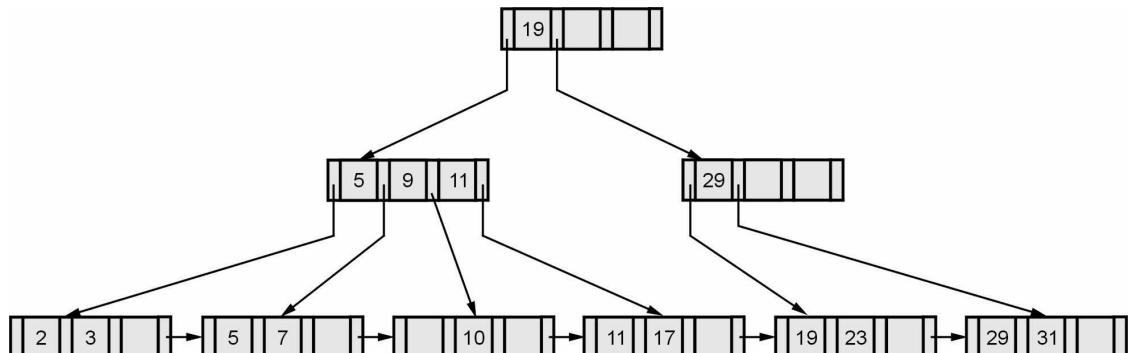
Step 8 : Insert 31. The sequence becomes 19,23,29, 31. The **29 will go up**. Then at the upper level the sequence becomes 5,11,19,29. Hence again **19 will go up** to maintain the capacity of node (it is four pointers= three key values at the most). Hence the complete B+ tree will be,



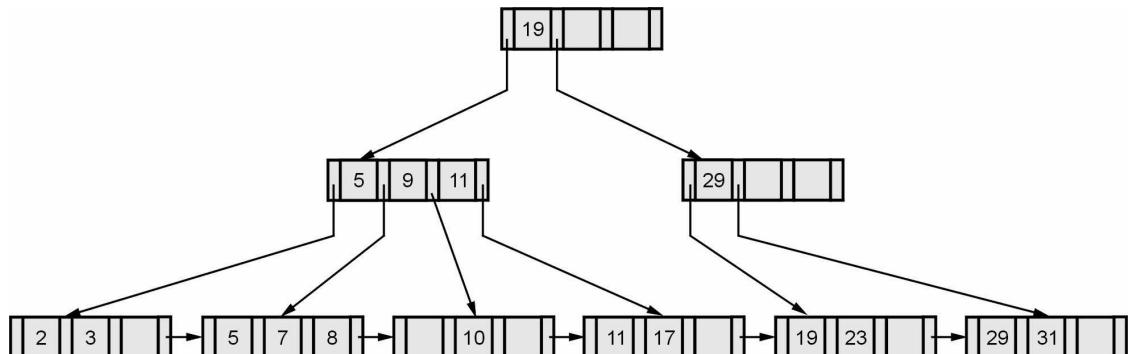
(a) Insertion of 9 : It is very simple operation as the node containing 5,7 has one space vacant to accommodate. The B+ tree will be,



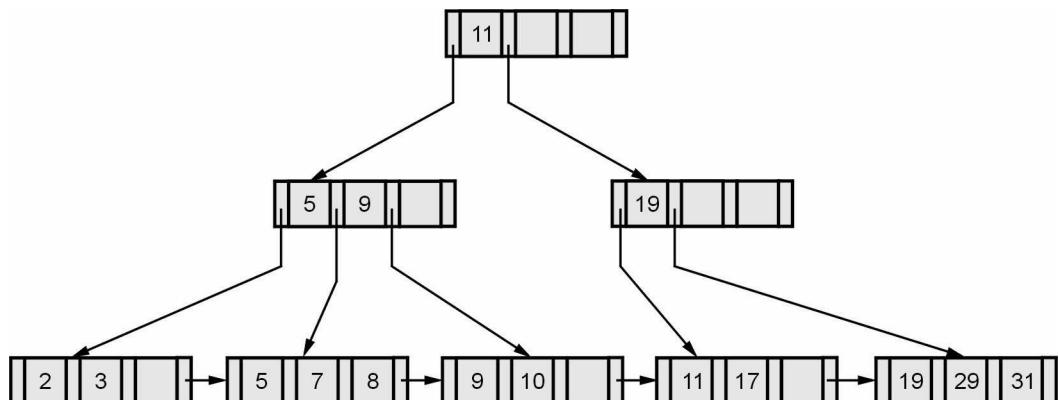
- (b) Insert 10 :** If we try to insert 10 then the sequence becomes 5,7,9,10. The 9 will go up.
The B+ tree will then become –



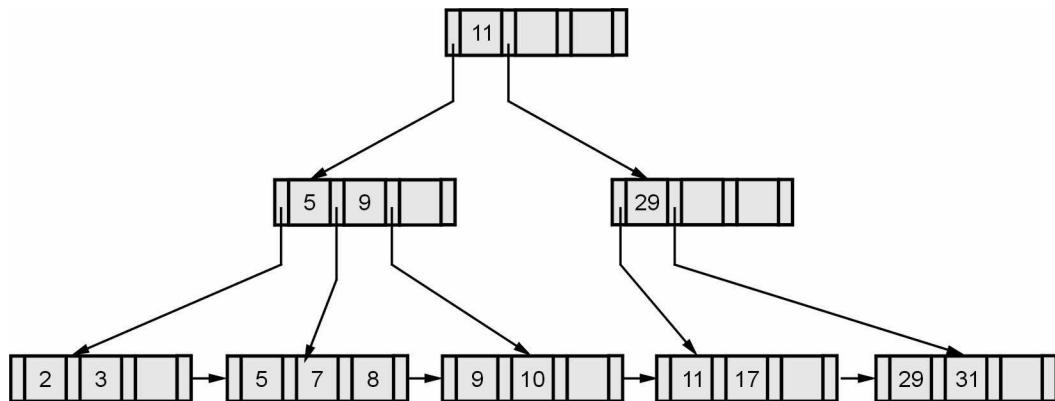
- (c) Insert 8 :** Again insertion of 8 is simple. We have a vacant space at node 5,7. So we just insert the value over there. The B+ tree will be-



- (d) Delete 23 :** Just remove the key entry of 23 from the node 19,23. Then merge the sibling node to form a node 19,29,31. Get down the entry of 11 to the leaf node. Attach the node of 11,17 as a left child of 19.



- (e) **Delete 19 :** Just delete the entry of 19 from the node 19,29,31. Delete the internal node key 19. Copy the 29 up as an internal node as it is an inorder successor node.



Merits of B+ Index Tree Structure

1. In B+ tree the data is stored in leaf node so searching of any data requires scanning only of leaf node alone.
2. Data is ordered in linked list.
3. Any record can be fetched in equal number of disk accesses.
4. Range queries can be performed easily as leaves are linked up.
5. Height of the tree is less as only keys are used for indexing.
6. Supports both random and sequential access.

Demerits of B+ Index Tree Structure

1. Extra insertion of non leaf nodes.
2. There is space overhead.

University Questions

- 1 Explain the B+ tree indexes on multiple keys with suitable example.
- 2 Briefly explain about B+ index file with example.
3. What are the merits and demerits of B+ tree index structures.
4. Describe structure of B+ tree and list the characteristics of B+ tree.

AU : Dec.-17, Marks 7

AU : May-16, Marks 16

AU : May-03, Marks 4

AU : May-03, Marks 6, May-06, Marks 8

4.7 B Tree Index Files

AU : May-08, Dec.-12, 14, Marks 16

- B-tree indices are similar to B+-tree indices.
- The primary distinction between the two approaches is that a B-tree eliminates the redundant storage of search-key values.

- B-tree is a specialized multiway tree used to store the records in a disk.
- There are number of subtrees to each node. So that the height of the tree is relatively small. So that only small number of nodes must be read from disk to retrieve an item. The goal of B-trees is to get fast access of the data.
- A B-tree allows search-key values to appear only once (if they are unique), unlike a B+-tree, where a value may appear in a nonleaf node, in addition to appearing in a leaf node.

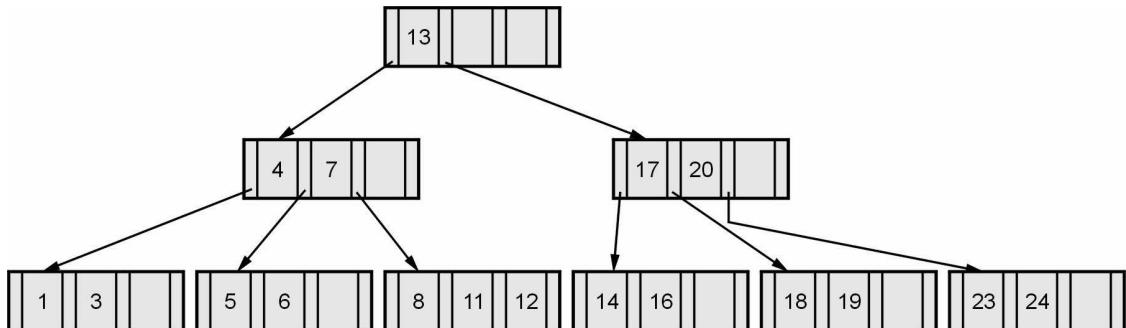


Fig. 4.7.1 B-Tree

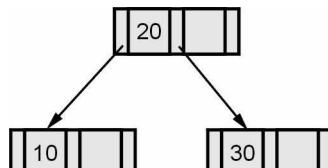
Example 4.7.1 Create B tree of order 3 for following data: 20,10,30,15,12,40,50.

Solution : The B tree of order 3 means at the most two key values are allowed in each node of B-Tree.

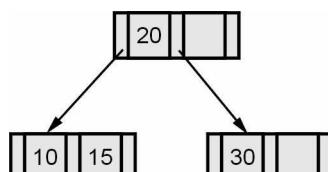
Step 1 : Insert 20,10 in ascending order



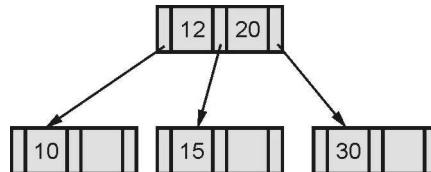
Step 2 : If we insert the value 30. The sequence becomes 10,20,30. As only two key values are allowed in each node (being order 3), the 20 will go up.



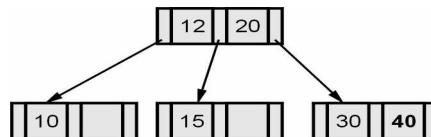
Step 3 : Now insert 15.



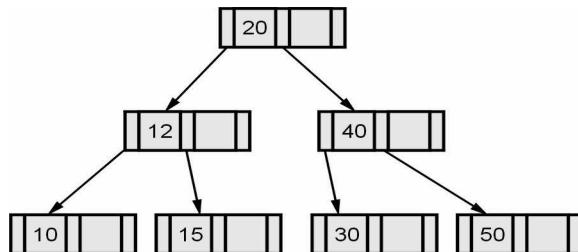
Step 4 : Insert 12. The sequence becomes 10, 12, 15. The middle element 12 will go up.



Step 5 : Insert 40



Step 6 : Insert 50. The sequence becomes 30,40,50. The 40 will go up. But again it forms 12,20,40 sequence and then 20 will go up. Thus the final B Tree will be,



This is the final B-Tree

Difference between B Tree and B+ Tree

B Trees	B+ Trees
In a B-tree you can store both keys and data in the internal and leaf nodes .	In a B+ tree you have to store the data in the leaf nodes only.
It wastes space.	It does not waste space.
The leaf node cannot store using linked list .	The leaf nodes are connected using linked list.
Searching becomes difficult in B-tree as data cannot be found in the leaf node.	Searching of any data in a B+ tree is very easy because all data is found in leaf nodes.
The B-tree does not store redundant search key .	The B-tree stores redundant search key .

University Questions

1. Explain detail about i) B+ tree index ii) B tree index files

AU : Dec.-14, Marks 16, May-08, Marks 10

2. Write down detailed notes on ordered indices and B-tree index files

AU : Dec.-12, Marks 16

4.8 Hashing

AU : Dec.- 04, 05, May-05, 14, Marks 8

- Hash file organization method is the one where data is stored at the data blocks whose address is generated by using **hash function**.
- The memory location where these records are stored is called as **data block** or **bucket**. This bucket is capable of storing one or more records.
- The hash function can use any of the column value to generate the address. Most of the time, hash function uses **primary key** to generate the hash index - address of the data block.
- Hash function can be **simple mathematical function** to any complex mathematical function.
- For example - Following figure represents the records of student can be searched using hash based indexing. In this example the hash function is based on the **age** field of the record. Here the index is made up of data entry k^* which is **actual data record**. For a hash function the **age** is converted to binary number format and the last two digits are considered to locate the student record.

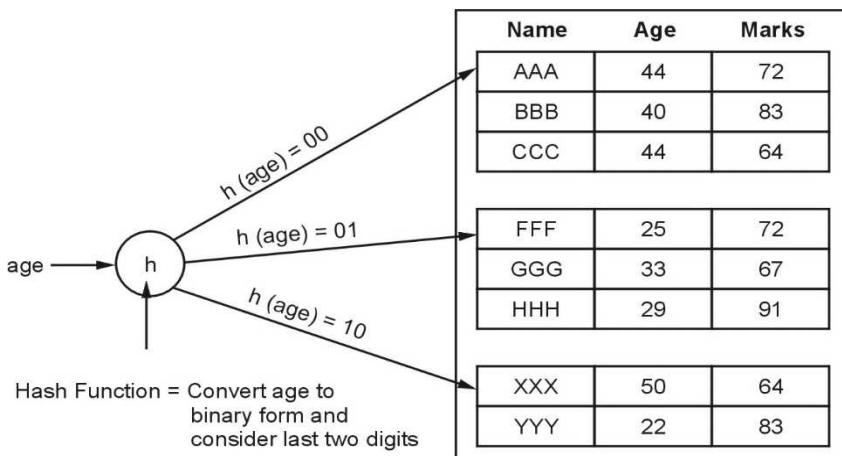


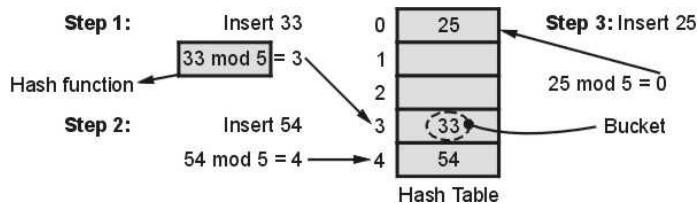
Fig. 4.8.1 : Hash based indexing

4.8.1 Basic Terms used in Hashing

- 1) **Hash Table** : Hash table is a data structure used for storing and retrieving data quickly. Every entry in the hash table is made using Hash function.
- 2) **Hash function :**
 - Hash function is a function used to place data in hash table.
 - Similarly hash function is used to retrieve data from hash table.
 - Thus the use of hash function is to implement hash table.

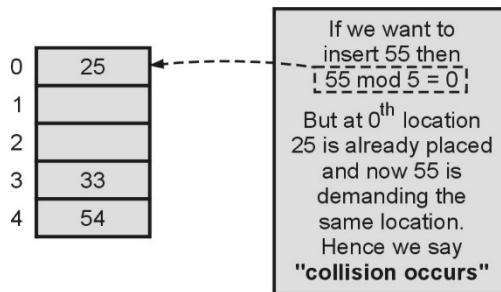
For example : Consider hash function as key

mod 5. The hash table of size 5.



- 3) **Bucket :** The hash function $H(key)$ is used to map several dictionary entries in the hash table. Each position of the hash table is called bucket.
- 4) **Collision :** Collision is situation in which hash function returns the same address for more than one record.

For example :



- 5) **Probe :** Each calculation of an address and test for success is known as a probe.
- 6) **Synonym :** The set of keys that has to the same location are called synonyms. For example - In above given hash table computation 25 and 55 are synonyms.
- 7) **Overflow :** When hash table becomes full and new record needs to be inserted then it is called overflow.

For example -

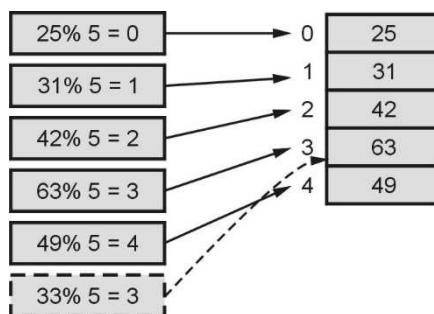


Fig. 4.8.2 : Overflow situation

University Questions

- 1 Write a note on hashing.
- 2 Describe hash file organization.

AU : May-14, Dec.- 05, Marks 8
AU : Dec.-04, May-05, Marks 8

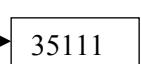
4.9 Static Hashing

AU : May-06, Marks 8

- In this method of hashing, the resultant data **bucket address will be always same**.

Index	Stud_RollNo
0	34780
1	34781
2	34782
3	34783
4	34784
5	34785
6	34786
7	34787
8	34788
9	34789

- That means, if we want to generate address for **Stud_RollNo = 34789**. Here if we use mod 10 hash function, it always result in the same bucket address 9. There will not be any changes to the bucket address here.
- Hence number of data buckets in the memory for this static hashing remains constant throughout. In our example, we will have ten data buckets in the memory used to store the data.



Index	Stud_RollNo
0	34780
1	34781
2	34782
3	34783
4	34784
5	34785
6	34786
7	34787
8	34788
9	34789

Table 4.9.1

- If there is **no space** for some data entry then we can allocate **new overflow page**, put the data record onto that page and add the page to **overflow chain** of the bucket. For example if we want to add the Stud_RollNo= 35111 in above hash table then as there is no space for this entry and the hash address indicate to place this record at index 1, we create overflow chain as shown in Table 4.9.1.

Example 4.9.1 Why is hash structure not the best choice for a search key on which range of queries are likely ?

AU : May-06, Marks 8

Solution :

- A range query cannot be answered efficiently using a hash index, we will have to read all the buckets.
- This is because key values in the range do not occupy consecutive locations in the buckets, they are distributed uniformly and randomly throughout all the buckets.

Advantages of Static Hashing

- (1) It is simple to implement.
- (2) It allows speedy data storage.

Disadvantages of Static Hashing

There are two major disadvantages of static hashing :

- 1) In static hashing, there are fixed number of buckets. This will create a problematic situation if the number of records grow or shrink.
- 2) The ordered access on hash key makes it inefficient.

4.9.1 Open Hashing

The **open hashing** is a form of **static hashing** technique. When the collision occurs, that means if the hash key returns the same address which is already allocated by some data record, then the **next available data block** is used to enter new record instead of overwriting the old record. This technique is also called as **linear probing**. For example

Consider insertion of record 105 in the hash table below with the hash function $h(\text{key}) \bmod 10$.

Index	Stud_RollNo
0	10
1	1
2	22
3	
4	
5	55
6	106
7	
8	88
9	19

The 105 is probed at next empty data block as follows -

Index	Stud_RollNo
0	10
1	1
2	22
3	
4	
5	55
6	106
7	105
8	88
9	19

Advantages :

- 1) It is **faster** technique.
- 2) It is **simple** to implement.

Disadvantages:

- 1) It forms clustering, as the record is just inserted to next free available slot.
- 2) If the hash table gets full then the next subsequent records can not be accommodated.

4.10 Dynamic Hashing

AU : May-04,07,18, Dec.-08,17, Marks 13

- The problem with static hashing is that it does not expand or shrink dynamically as the size of the database grows or shrinks.
- Dynamic hashing provides a mechanism in which data buckets are added and removed dynamically and on-demand.
- The most commonly used technique of dynamic hashing is extendible hashing.

4.10.1 Extendible Hashing

The extendible hashing is a dynamic hashing technique in which, if the bucket is overflow, then the number of buckets are doubled and data entries in buckets are re-distributed.

Example of extendible hashing :

In extendible hashing technique the **directory** of pointers to bucket is used. Refer following Fig. 4.10.1

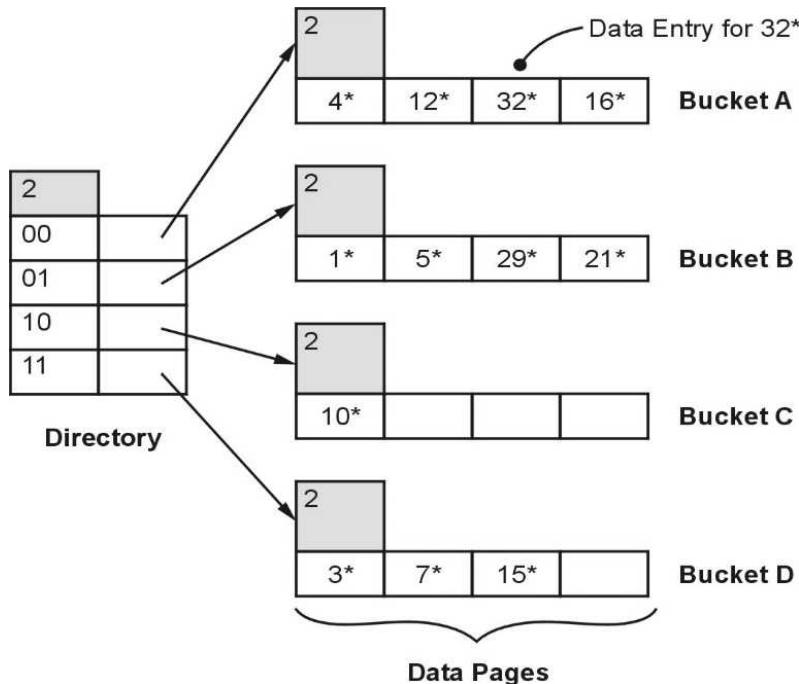


Fig. 4.10.1 Extendible hashing

To locate a data entry, we apply a hash function to search the data we us **last two digits of binary representation of number**. For instance binary representation of 32^* = 10000000. The last two bits are 00. Hence we store 32^* accordingly.

Insertion operation :

- Suppose we want to insert 20^* (binary 10100). But with 00, the bucket A is full. So we must **split the bucket** by allocating new bucket and redistributing the contents, across the old bucket and its split image.
- For splitting, we consider last three bits of $h(r)$.
- The redistribution while insertion of 20^* is as shown in following Fig. 4.10.2

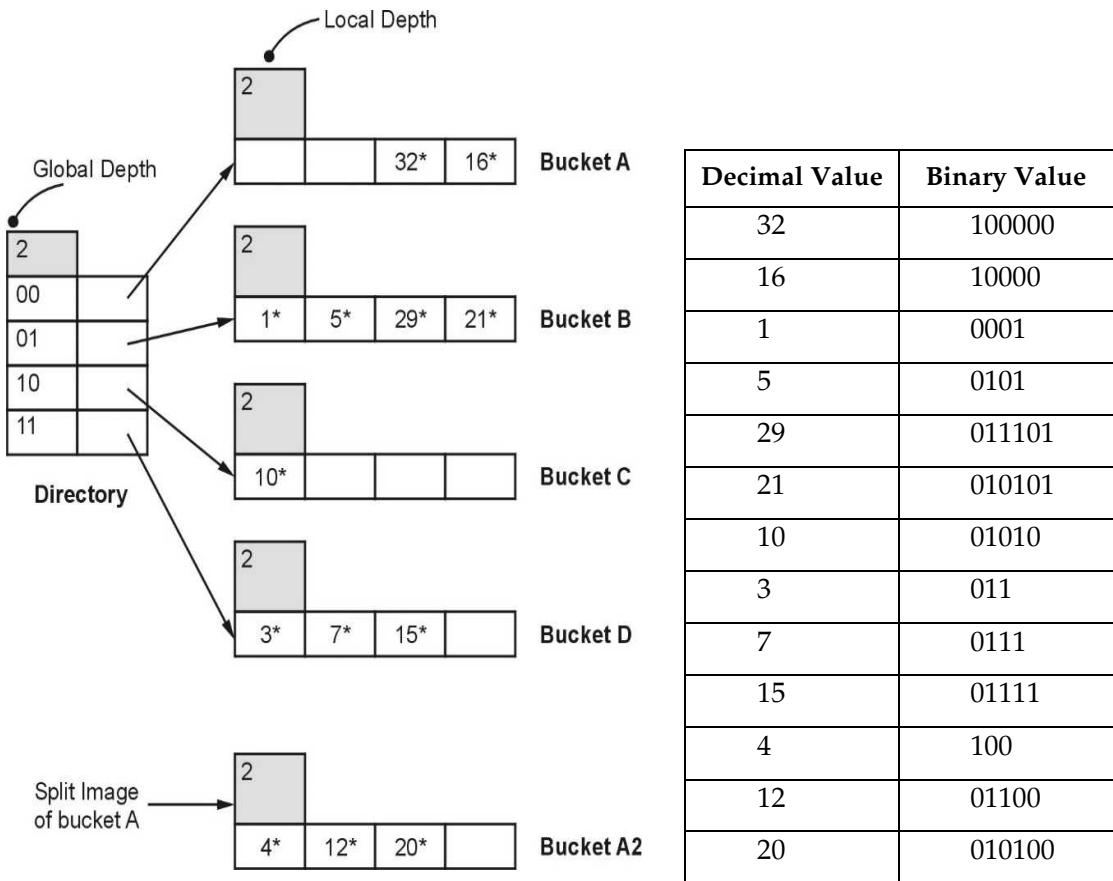


Fig. 4.10.2 : During insertion process

The split image of bucket A i.e. A2 and old bucket A are based on last two bits i.e. 00. Here we need two data pages, to adjacent additional data record. Therefore here it is necessary to **double the directory** using three bits instead of two bits. Hence,

- There will be binary versions for buckets A and A2 as 000 and 100.
- In extendible hashing, last bits d is called **global depth** for directory and d is called **local depth** for data pages or buckets. After insertion of 20*, the global depth becomes 3 as we consider last three bits and local depth of A and A2 buckets become 3 as we are considering last three bits for placing the data records. Refer Fig. 4.10.3.

(**Note :** Student should refer binary values given in Fig. 4.10.2, for understanding insertion operation)

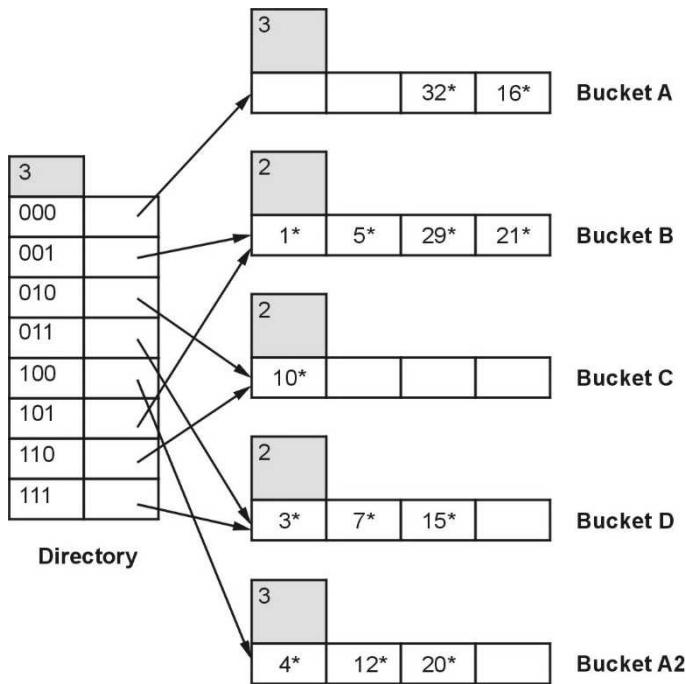


Fig. 4.10.3 : After insertion of 20*

- Suppose if we want to insert 11*, it belongs to bucket B, which is already full. Hence let us split bucket B into old bucket B and split image of B as B2.
- The local depth of B and B2 now becomes 3.
- Now for bucket B, we get and $1 = 001$
 $11 = 10001$
- For bucket B2, we get

$$5 = 101$$

$$29 = 11101$$

and $21 = 10101$

After insertion of 11* we get the scenario as follows,

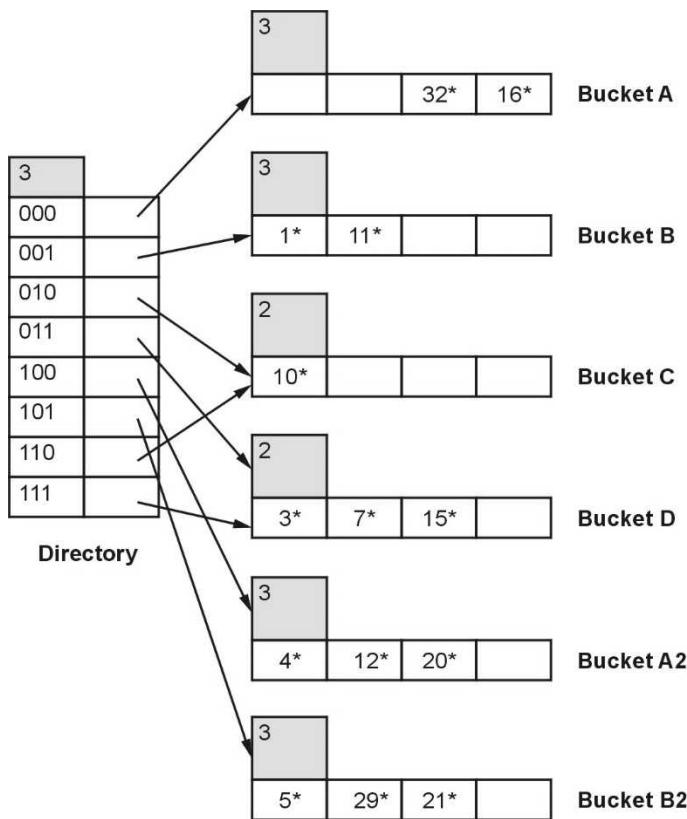


Fig. 4.10.4 : After insertion of 11*

Example 4.10.1 The following key values are organized in an extendible hashing technique. 1 3 5 8 9 12 17 28. Show the extendible hash structure for this file if the hash function is $h(x) = x \bmod 8$ and buckets can hold three records. Show how extendible hash structure changes as the result of each of the following steps :

Insert 2

Insert 24

Delete 5

Delete 12

Solution :

Step 1 : Initially we assume the hash function based on last two bits, of result of hash function.

$$1 \bmod 8 = 1 = 001$$

$$3 \bmod 8 = 3 = 011$$

$$5 \bmod 8 = 5 = 101$$

$8 \bmod 8 = 0 = 000$

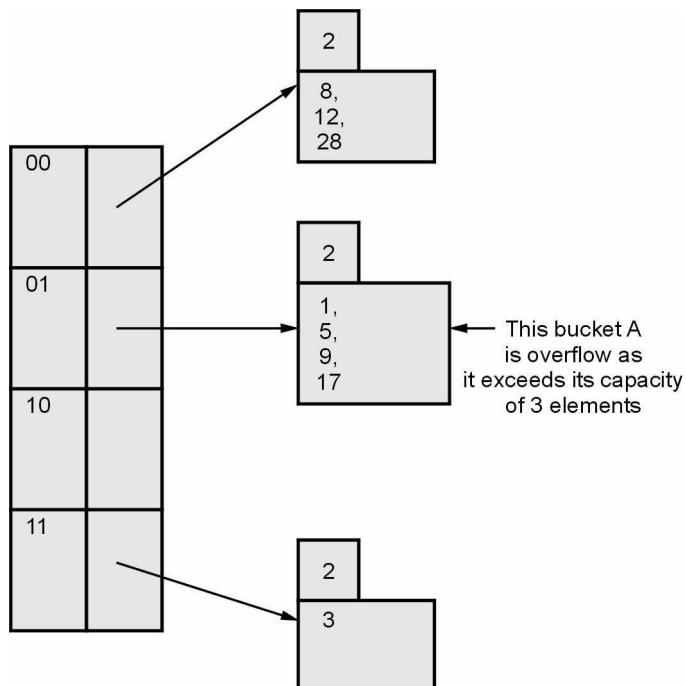
$9 \bmod 8 = 1 = 001$

$12 \bmod 8 = 4 = 100$

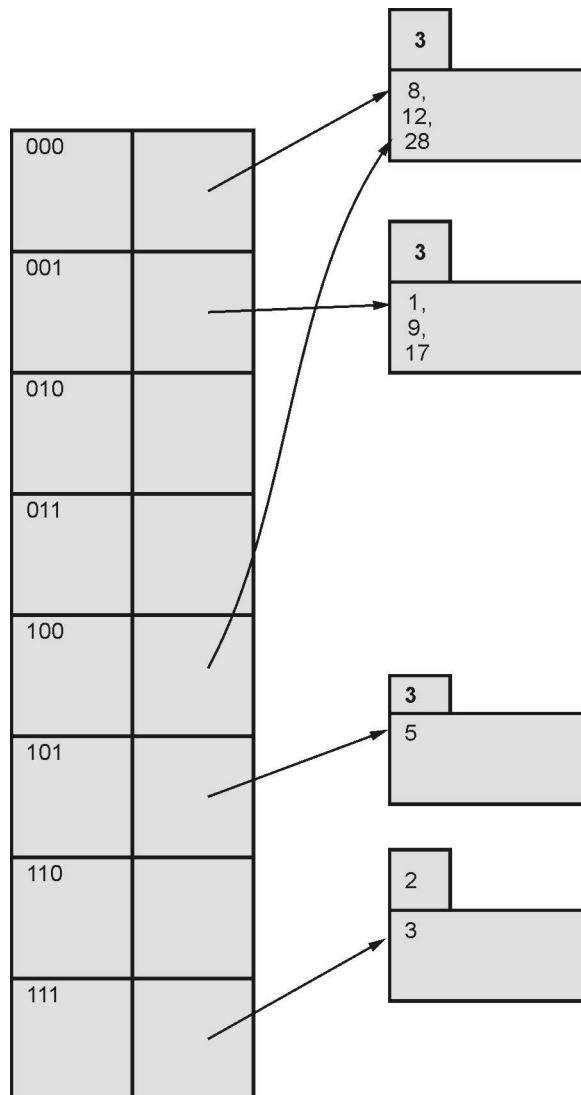
$17 \bmod 8 = 1 = 001$

$28 \bmod 8 = 4 = 100$

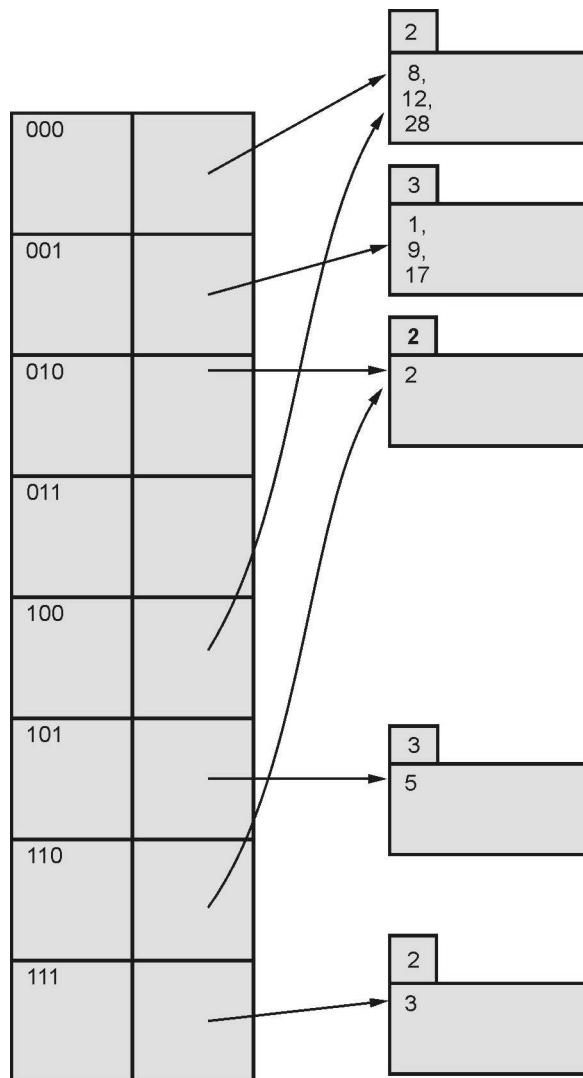
The extendible hash table will be,



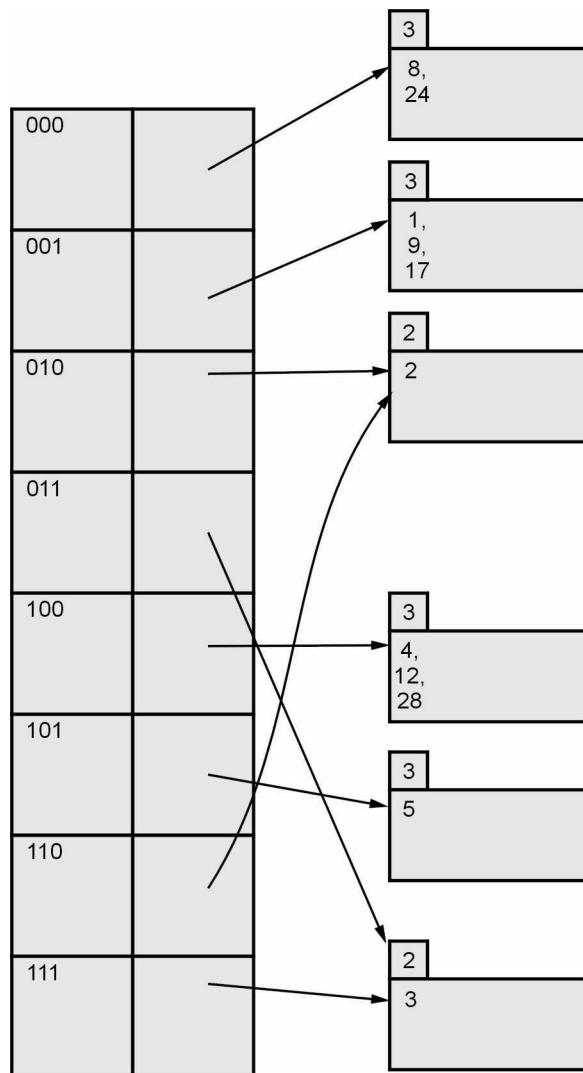
Hence we will extend the table by assuming the bit size as 3. The above indicated bucket A will split based on 001 and 101.



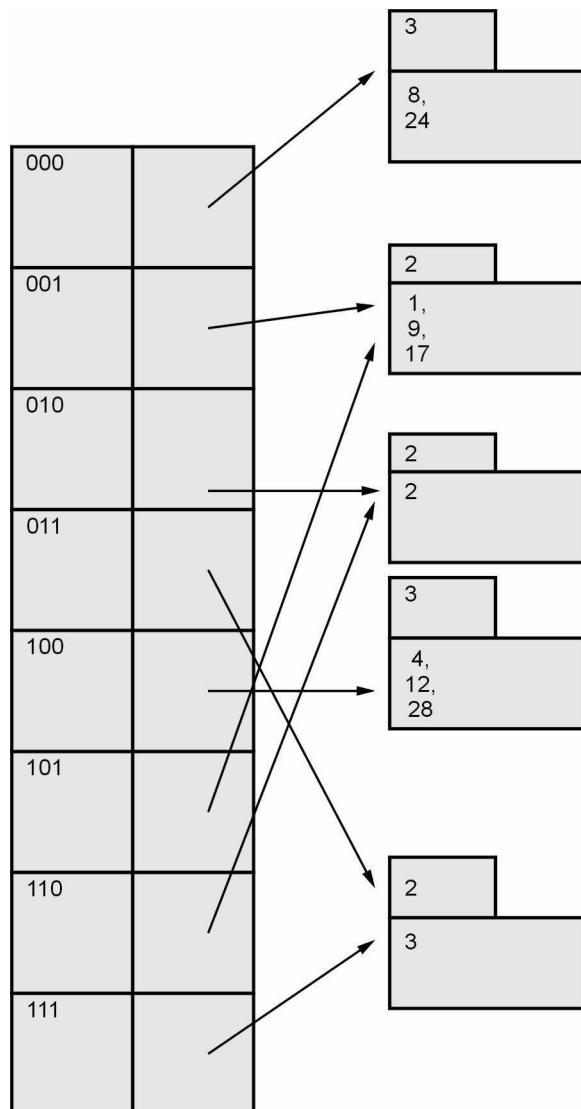
- a) **Insert 2** will be $2 \bmod 8 = 2 = 010$. If we consider last two digits i.e. 10 then there is no bucket. So we get,



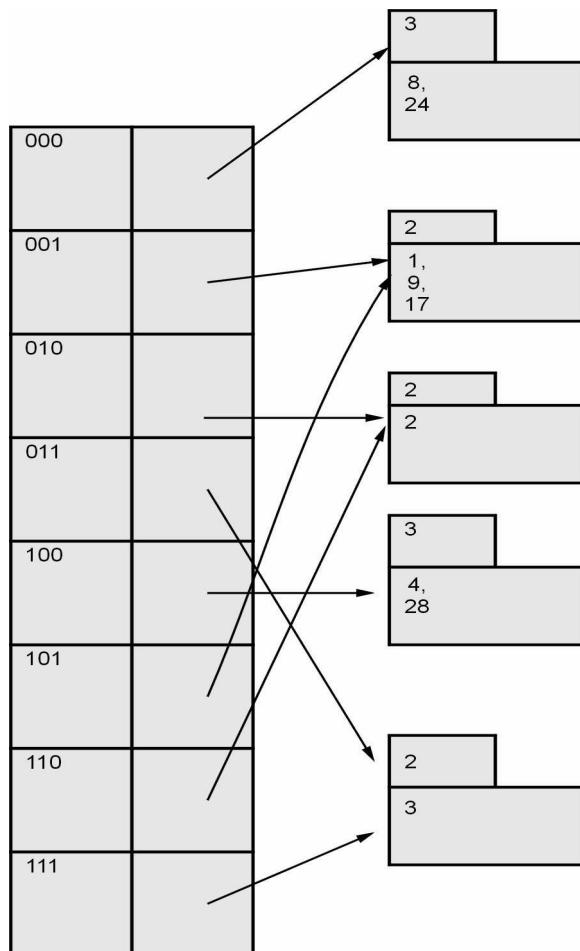
- b) **Insert 24 :** $24 \bmod 8 = 0 = 000$. The bucket in which 24 can be inserted is 8, 12, 28. But as this bucket is full we split it in two buckets based on digits 000 100.



- c) **Delete 5 :** On deleting 5, we get one bucket pointed by 101 as empty. This will also result in reducing the local depth of the bucket pointed by 001. Hence we get,



(d) Delete 12 : We will simply delete 12 from the corresponding bucket there can not be any merging of buckets on deletion. The result of deletion is as given below -



Difference between Static and Dynamic Hashing

Sr. No.	Static Hashing	Dynamic Hashing
1.	The number of buckets are fixed .	The number of buckets are not fixed .
2.	Chaining is used	There is no need of chaining .
3.	Open hashing and Closed hashing are forms of static hashing.	Extendible hashing and linear hashing are forms of dynamic hashing.
4.	Space overhead is more.	Minimum space overhead due to dynamic nature.
5.	As file grows the performance of static hash function decreases .	There is no degradation in performance when the file grows.
6.	The bucket address table is not required.	The bucket address table is required.
7.	The bucket is directly accessed .	The bucket address table is used to access the bucket.

University Questions

1. What is hashing? Explain static hashing and dynamic hashing with an example.

AU : May-18, Marks 13

2. Explain the distinction between static and dynamic hashing. Discuss the relative merits of each technique in database applications.

AU : Dec.-17, Marks 13

3. Describe briefly static and dynamic hashing.

AU : May-04, Marks 8, Dec.-08, Marks 6

4. Explain various hashing techniques.

AU : May-07, Marks 8

Part II : Query Processing**4.11 Query Processing Overview**

AU : May-14, 16, 18, Marks 16

- Query processing is a collection of activities that are involved in extracting data from database.
- During query processing there is translation high level database language queries into the expressions that can be used at the physical level of filesystem.
- There are three basic steps involved in query processing and those are –

1. Parsing and Translation

- In this step the query is translated into its internal form and then into **relational algebra**.
- Parser checks syntax and verifies relations.
- For instance - If we submit the query as,

```
SELECT RollNo, name
FROM Student
HAVING RollNo=10
```

Then it will issue a syntactical error message as the correct query should be

```
SELECT RollNo, name
FROM Student
HAVING RollNo=10
```

Thus during this step the syntax of the query is checked so that only correct and verified query can be submitted for further processing.

2. Optimization :

- During this process the query evaluation plan is prepared from all the relational algebraic expressions.
- The query cost for all the evaluation plans is calculated.
- Amongst all equivalent evaluation plans the one with lowest cost is chosen.

- Cost is estimated using statistical information from the database catalog, such as the number of tuples in each relation, size of tuples, etc.

3. Evaluation

- The query-execution engine takes a query-evaluation plan, executes that plan, and returns the answers to the query.

The above described steps are represented by following Fig. 4.11.1 –

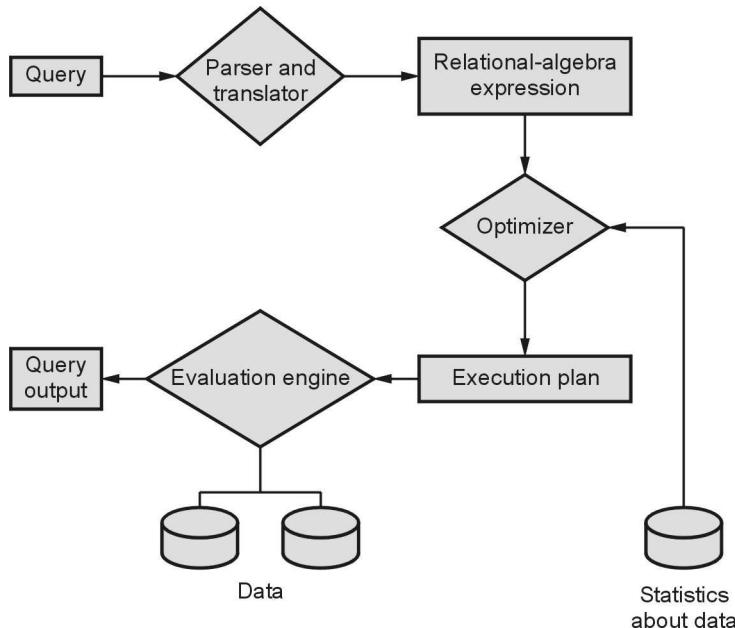


Fig. 4.11.1 Query processing

For example - If the SQL query is,

```

SELECT balance
FROM account
WHERE balance < 1000
    
```

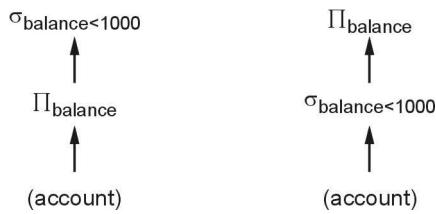
Step 1 : This query is first verified by the **parser and translator** unit for correct syntax. If so then the relational algebra expressions can be obtained. For the above given queries there are two possible relational algebra

$$(1) \sigma_{\text{balance} < 1000}(\pi_{\text{balance}}(\text{account}))$$

$$(2) \pi_{\text{balance}} (\sigma_{\text{balance} < 1000} (\text{account}))$$

Step 2 :

Query Evaluation Plan : To specify fully how to evaluate a query, we need not only to provide the relational-algebra expression, but also to annotate it with instructions specifying how to evaluate each operation. For that purpose, using the order of evaluation of queries, two query evaluation plans are prepared. These are as follows

**Fig. 4.11.2 Query evaluation plans**

Associated with each query evaluation plan there is a **query cost**. The query optimization selects the query evaluation plan having minimum query cost.

Once the query plan is chosen, the **query is evaluated** with that plan and the **result of the query is output**.

University Questions

1. Briefly explain about query processing.

AU : May-14, 16, Marks 16

2. What is query optimization? Outline the steps in query optimization.

AU : May-18, Marks 13

4.12 Measure of Query Cost

- There are multiple possible evaluation plans for a query, and it is important to be able to compare the alternatives in terms of their estimated cost and choose the best plan.
- There are many factors that contribute to query cost are -
 - Disk access
 - CPU time to execute the query
 - Cost of communication in distributed and parallel database system.
- The cost of access data from disk is an important cost. Normally disk access is relatively slow as compared to in-memory operations. Moreover, the CPU speed is much faster than the disk speed. Hence the time spent in disk is relatively dominant factor in query execution.
- Computing CPU access time is comparatively harder, hence we will not consider the CPU time to execute the query.
- Similarly the cost of communication does not matter for simple large databases present in the centralized database system.
- Typically disk access is the predominant cost and is also relatively easy to estimate, taking into account:
 - Number of seeks × average-seek-cost
 - Number of blocks read × average-block-read-cost
 - Number of blocks written × average-block-write-cost

- Cost to write a block is greater than cost to read a block because data is read back after being written to ensure that the write was successful.
- We use **number of block transfers** from disk and **number of disk seeks** to estimate the Query cost.
- Let,
 - b be the number to blocks
 - S be the number of Seek
 - t_T is average time required to transfer a block of data, in seconds
 - t_S is average block access time in seconds.

Then query cost can be computed using following formula $b*t_T + S*t_S$

4.13 Algorithms for SELECT Operation

For selection operation, the file scan is an important activity. Basically file scan is a based on searching algorithms. These searching algorithms locate and retrieve the records that fulfills a selection condition.

Let us discuss various algorithms used for SELECT Operation based in file scan

Algorithm A1 : Linear Search

- Scan each file block and test all records to see whether they satisfy the selection condition

$$\text{Cost} = b_r \text{ block transfers} + 1 \text{ seek}$$

Where,

b_r denotes number of blocks containing records from relation r

- If selection is on a key attribute, can stop on finding record

$$\text{Cost} = (b_r/2) \text{ block transfers} + 1 \text{ seek}$$

Advantages of Linear Search

- Linear search works even-if there is no selection condition specified.
- For linear search, there is no need to have records in the file in ordered form.
- Linear search works regardless of indices.

Algorithm A2 : Binary Search

- Applicable if selection is an equality comparison on the attribute on which file is ordered.
- Assume that the blocks of a relation are stored contiguously.
- Cost estimate is nothing but the number of disk blocks to be scanned.

Cost of locating the first tuple by a binary search on the blocks = $\lceil \log_2(b_r) \rceil \times (t_T + t_S)$

Where,

b_r denotes number of blocks containing records from relation r

t_r is average time required to transfer a block of data, in seconds

t_s is average block access time, in seconds

- If there are multiple records satisfying the selection add transfer cost of the number of blocks containing records that satisfy selection condition.

4.14 Algorithms for JOIN Operation

- JOIN operation is the most time consuming operation to process.
- There are Several different algorithms to implement joins
 1. Nested-loop join
 2. Block nested-loop join
 3. Indexed nested-loop join
 4. Merge-join
 5. Hash-join

Choice of a particular algorithm is based on cost estimate.

Algorithm For Nested Loop Join

This algorithm is for computing $r \bowtie \theta s$

Let, r is called the outer relation and s the inner relation of the join.

```
for each tuple  $t_r$  in  $r$  do begin
  for each tuple  $t_s$  in  $s$  do begin
    test pair  $(t_r, t_s)$  to see if they satisfy the join condition  $\theta$ 
    if  $\theta$  is satisfied, then, add  $(t_r, t_s)$  to the result.
  end
end
```

- This algorithm requires no indices and can be used with any kind of join condition.
- It is expensive since it examines every pair of tuples in the two relations.
- In the **worst case**, if there is enough memory only to hold one block of each relation, the estimated cost is
- $n_r \times b_s + b_r$ block transfers, plus $n_r + b_r$ seeks
- If the smaller relation fits entirely in memory, use that as the inner relation
 - Reduces cost to $b_r + b_s$ block transfers and 2 seeks
- **For example** - Assume the query CUSTOMERS \bowtie ORDERS (with join attribute only being CName)

Number of records of customer : 10000 order : 5000

Number of blocks of customer : 400 order : 100

Formula Used :

(1) $n_r \times b_s + b_r$ block transfers,

(2) $n_r + b_r$ seeks

r is outer relation and s is inner relation.

With order as outer relation:

$n_r = 5000, b_s = 400, b_r = 100$

$5000 \times 400 + 100 = 2000100$ block transfers and

$5000 + 100 = 5100$ seeks

With customer as the outer relation:

$n_r = 10000, b_s = 100, b_r = 400$

$10000 \times 100 + 400 = 1000400$ block transfers and

$10000 + 400 = 10400$ seeks

If smaller relation (order) fits entirely in memory, the cost estimate will be:

$b_r + b_s = 500$ block transfers

Algorithm For Block Nested Loop Join

Variant of nested-loop join in which every block of inner relation is paired with every block of outer relation.

This algorithm is for computing $r \bowtie \theta s$

Let, r is called the outer relation and s the inner relation of the join.

```

for each block  $B_r$  of r do
  for each block  $B_s$  of s do
    for each tuple  $t_r$  in  $B_r$  do begin
      for each tuple  $t_s$  in  $B_s$  do begin
        test pair  $(t_r, t_s)$  to see if they satisfy the join condition  $\theta$ 
        if  $\theta$  is satisfied, then, add  $(t_r, t_s)$  to the result.
      end
    end
  end
end

```

Worst case estimate: $b_r \times b_s + b_r$ block transfers + $2 \times b_r$ seeks

Each block in the inner relation s is read once for each block in the outer relation.

Best case : $b_r + b_s$ block transfers + 2 seeks

(3) Merge Join

- In this operation, both the relations are sorted on their join attributes. Then merged these sorted relations to join them.
- Only equijoin and natural joins are used.
- The cost of merge join is,

$b_r + b_s$ block transfers + $\lceil b_r / b_b \rceil + \lceil b_s / b_b \rceil$ seeks + the cost of sorting, if relations are unsorted.

(4) Hash Join

- In this operation, the hash function h is used to partition tuples of both the relations.
- h maps A values to $\{0, 1, \dots, n\}$, where A denotes the attributes of r and s used in the join.
- Cost of hash join is :

$$3(b_r + b_s) + 4 \times n \text{ block transfers} + 2(\lceil b_r / b_b \rceil + \lceil b_s / b_b \rceil) \text{ seeks}$$

If the entire build input can be kept in main memory no partitioning is required
Cost estimate goes down to $b_r + b_s$.

4.15 Query Optimization using Heuristics and Cost Estimation

AU : Dec. -13, 16, May-15, Marks 16

4.15.1 Heuristic Estimation

- Heuristic is a rule that leads to least cost in most of cases.
- Systems may use heuristics to reduce the number of choices that must be made in a cost-based fashion.
- Heuristic optimization transforms the query-tree by using a set of rules that typically improve execution performance. These rules are
 1. Perform selection early (reduces the number of tuples)
 2. Perform projection early (reduces the number of attributes)
 3. Perform most restrictive selection and join operations before other similar operations (such as cartesian product).
- Some systems use only heuristics, others combine heuristics with partial cost-based optimization.

Steps in Heuristic Estimation

Step 1 : Scanner and parser generate initial query representation

Step 2 : Representation is optimized according to heuristic rules

Step 3 : Query execution plan is developed

For example : Suppose there are two relational algebra -

$$(1) \sigma_{\text{city} = \text{"Pune"}} (\pi_{\text{cname}} \text{Branch}) \bowtie \text{Account} \bowtie \text{Customer}$$

$$(2) \pi_{\text{cname}}(\sigma_{\text{city} = \text{"Pune}}(\text{Branch} \bowtie \text{Account} \bowtie \text{Customer}))$$

The query evaluation plan can be drawn using the query trees as follows -

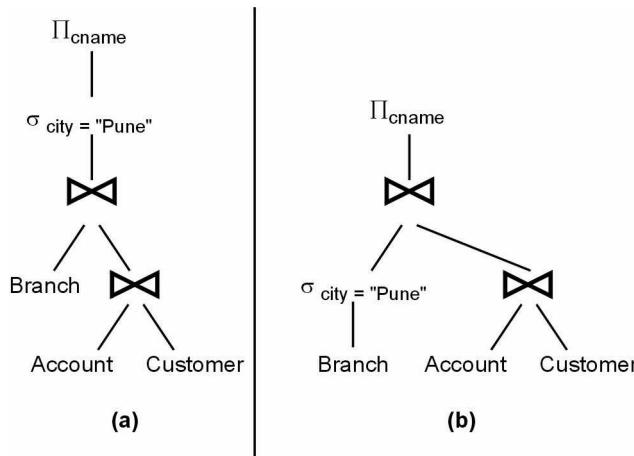


Fig. 4.15.1 Query evaluation plan

Out of the above given query evaluation plans, the Fig. 4.15.1 (b) is much faster than Fig. 4.15.1 (a) because – in Fig. 4.15.1 (a) the join operation is among Branch, Account and Customer, whereas in Fig. 4.15.1 (b) the join of (Account and Customer) is made with the selected tuple for City="Pune". Thus the output of entire table for join operation is much more than the join for some selected tuples. Thus we get choose the optimized query.

4.15.2 Cost based Estimation

- A cost based optimizer will look at all of the possible ways or scenarios in which a query can be executed.
- Each scenario will be assigned a ‘cost’, which indicates how efficiently that query can be run.
- Then, the cost based optimizer will pick the scenario that has the least cost and execute the query using that scenario, because that is the most efficient way to run the query.
- Scope of query optimization is a query block. Global query optimization involves multiple query blocks.
- Cost components for query execution
 - Access cost to secondary storage
 - Disk storage cost
 - Computation cost

- Memory usage cost
- Communication cost
- Following information stored in DBMS catalog and used by optimizer
 - File size
 - Organization
 - Number of levels of each multilevel index
 - Number of distinct values of an attribute
 - Attribute selectivity
- RDBMS stores histograms for most important attributes

University Questions

- | | |
|---|-------------------------------|
| <i>1. Explain query optimization with an example</i> | AU : Dec.-16, Marks 8 |
| <i>2. Discuss about the Join order optimization and Heuristic optimization algorithms.</i> | AU : May-15, Marks 16 |
| <i>3. Give a detailed description about query processing and optimization. Explain the cost estimation of query optimization.</i> | AU : Dec.-13, Marks 16 |

4.16 Two Marks Questions with Answers

Q.1 What is the need for RAID?

AU : May-13

Ans. : Refer section 4.1

Q.2 Define Software and hardware RAID systems

AU : May-16

Ans. : **Hardware RAID** : The hardware-based array manages the RAID subsystem independently from the host. It presents a single disk per RAID array to the host.

Software RAID : Software RAID implements the various RAID levels in the kernel disk code. It offers the cheapest possible solution, as expensive disk controller cards.

Q.3 What are ordered indices ?

AU : June-09, Dec. -11,17, May-14

Ans. : This is type of indexing which is based on sorted ordering values. Various ordered indices are primary indexing, secondary indexing.

Q.4 What are the two types of ordered indices ?

AU : Dec.-06

Ans. : Two types of ordered indices are - Primary indexing and secondary indexing. The primary indexing can be further classified into dense indexing and sparse indexing and single level indexing and multilevel indexing.

Q.5 Give the comparison between ordered indices and hashing

AU : Dec.-06

Ans. :

- (1) If range of queries are common, ordered indices are to be used.
- (2) The buckets containing records can be chained in sorted order in case of ordered indices.
- (3) Hashing is generally better at retrieving records having a specified value of the key.
- (4) Hash function assigns values randomly to buckets. Thus, there is no simple notion of "next bucket in sorted order."

Q.6 What are the causes of bucket overflow in a hash file organization ?

Ans. : Bucket overflow can occur for following reasons -

- (1) **Insufficient buckets** : For the total number of buckets there are insufficient number of buckets to occupy.
- (2) **Skew** : Some buckets are assigned more records than are others, so a bucket might overflow even while other buckets still have space. This situation is known as bucket skew.

Q.7 What can be done to reduce the occurrences of bucket overflows in a hash file organization ?

AU : May-07, June-09, Dce.-12

Ans. :

- (1) A bucket is a unit of storage containing one or more records (a bucket is typically a disk block).
- (2) The file blocks are divided into M equal-sized buckets, numbered bucket0, bucket1... bucketM-1. Typically, a bucket corresponds to one (or a fixed number of) disk block.
- (3) In a hash file organization we obtain the bucket of a record directly from its search-key value using a hash function, h (K).
- (4) To reduce overflow records, a hash file is typically kept 70-80% full.
- (5) The hash function h should distribute the records uniformly among the buckets; otherwise, search time will be increased because many overflow records will exist.

Q.8 Distinguish between dense and sparse indices.

AU : May-08, June-09

Ans. : Refer section 4.5.2.

Q.9 When is it preferable to use a dense index rather than a sparse index ? Explain your answer.

AU : Dec.-11

Ans. :

1. It is preferable to use a dense index instead of a sparse index when the file is not sorted on the indexed field.
2. Or when the index file is small compared to the size of memory.

Q.10 How does B-tree differs from a B+ tree? Why is a B+ tree usually preferred as an access structure to a data file?

AU : Dec.-08

Ans. : Refer section 4.7.

Q.11 What are the disadvantages of B tree over B+ tree

AU : Dec.-16

Ans. :

(1) Searching of a key value becomes difficult in B-tree as data can not be found in the leaf node.

(2) The leaf node can not store linked list and thus wastes the space.

Q.12 Mention different hashing techniques.

AU : May-12

Ans. : Two types of hashing techniques are –

i) Static hashing ii) Dynamic hashing.

Q.13 List the mechanisms to avoid collision during hashing

AU : Dec.-16

Ans. : Collision Resolution techniques are :

(1) Separate chaining

(2) Open addressing techniques : (i) Linear probing (ii) Quadratic probing

Q.14 What is the basic difference between static hashing and dynamic hashing?

AU : May-13, 15, Dec.-14, 15

Ans. : Refer section 4.10.

Q.15 What is the need for query optimization?

AU : May - 15

Ans. : Query optimization is required for fast execution of long running complex queries.

Q.16 Which cost component are used most commonly as the basis for cost function

AU : May-17

Ans. : Disk access or secondary storage access is considered most commonly as a basis for cost function.

Q.17 What is query execution plan ?

AU : May-17

Ans. : To specify fully how to evaluate a query, we need not only to provide the relational-algebra expression, but also to annotate it with instructions specifying how to evaluate each operation. This annotated structure is called query execution plan.



5**Advanced Topics*****Syllabus***

Distributed Databases : Architecture, Data Storage, Transaction Processing - Object-based Databases : Object Database Concepts, Object-Relational features, ODMG Object Model, ODL, OQL - XML Databases : XML Hierarchical Model, DTD, XML Schema, XQuery - Information Retrieval: IR Concepts, Retrieval Models, Queries in IR systems.

Contents

5.1	<i>Architecture of Distributed Databases.....</i>	<i>May-03, 14, 16, 17, Dec.-04, 07, 16, 17,.....</i>	Marks 16
5.2	<i>Data Storage</i>	<i>Dec.-04, May-06,</i>	Marks 8
5.3	<i>Transaction Processing.....</i>	<i>Dec.-12, 14, May-18,</i>	Marks 8
5.4	<i>Object Database Concepts</i>	<i>May-18</i>	Marks 13
5.5	<i>Object Relational Features.....</i>	<i>Dec.-16,.....</i>	Marks 13
5.6	<i>ODMG Object Model</i>		
5.7	<i>XML Hierarchical Mode</i>		
5.8	<i>DTD.....</i>	<i>Dec.-16.....</i>	Marks 15
5.9	<i>XML Schema</i>		
5.10	<i>XQuery.....</i>	<i>May-17</i>	Marks 5
5.11	<i>IR Concepts</i>		
5.12	<i>Retrieval Models</i>		
5.13	<i>Queries in IR Systems</i>		
5.14	<i>Two Marks Questions with Answers</i>		

Part I : Distributed Databases

5.1 Architecture of Distributed Databases

AU : May-03,14,16,17, Dec.-04,07,16,17, Marks 16

Definition of Distributed Databases : 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 run on each site is independent of each other. Refer Fig. 5.1.1.

The transactions can access data at one or more sites.

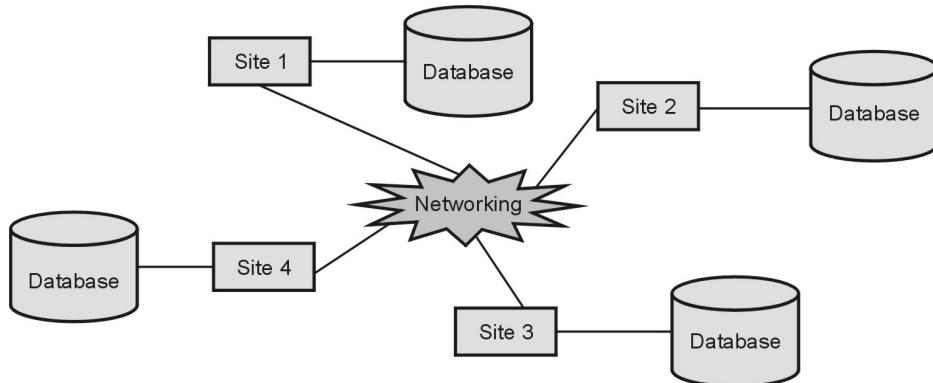


Fig. 5.1.1 Distributed Database System

Features of Distributed Database System

- (1) The distributed databases appear as a single database to the user.
- (2) Data is stored at different sites.
- (3) The sites are interconnected by network.
- (4) Different sites cooperate each other during transaction processing.
- (5) Distributed database management system has full functionality of database management system.

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.

5.1.1 Homogeneous and Heterogeneous Databases

There are two types of distributed databases –

(1) Homogeneous Databases

- The homogeneous databases are kind of database systems in which all sites have identical software running on them. Refer Fig. 5.1.2

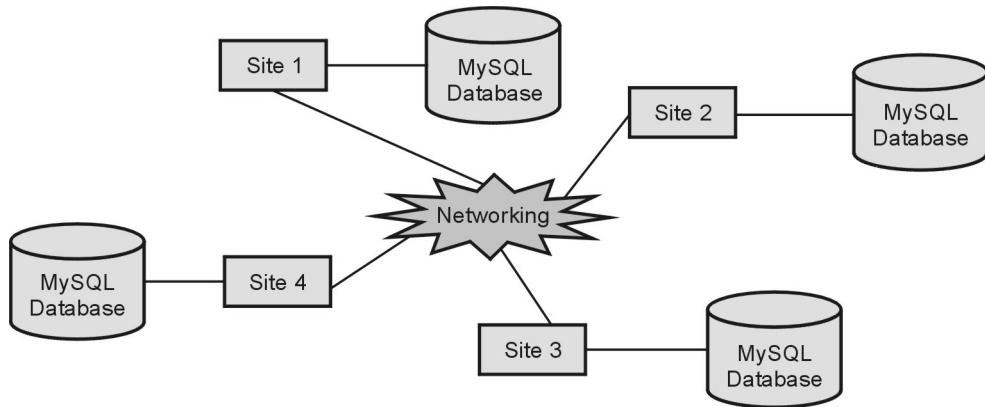
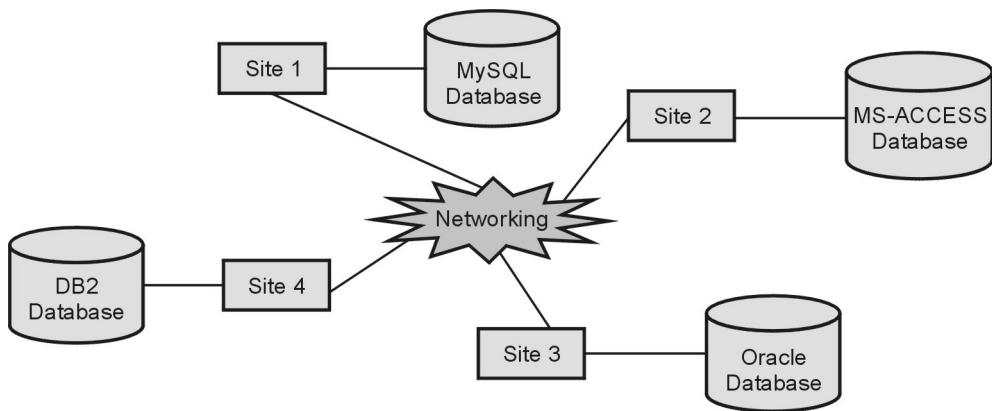


Fig. 5.1.2 Homogeneous databases

- In this system, all the sites are aware of the other sites present in the system and they all cooperate in processing user's request.
- Each site present in the system, surrenders part of its autonomy in terms of right to change schemas or software.
- The homogeneous database system appears as a single system to the user.

(2) Heterogeneous Databases

- The heterogeneous databases are kind of database systems in which different sites have different schema or software. Refer Fig. 5.1.3.

**Fig. 5.1.3 Heterogeneous databases**

- The participating sites are not aware of other sites present in the system.
- These sites provide limited facilities for cooperation in transaction processing.

University Questions

1. What are the various features of distributed database versus centralized database system?

AU : Dec.-17, Marks 6, May-17, Marks 8

2. Explain the architecture of a distributed database.

AU : Dec.-16, Marks 7

3. Explain about distributed databases and their characteristics, functions and advantages and disadvantages.

AU : Dec.-07, May-14, Marks 8, May-16, Marks 16

4. Explain design of distributed database.

AU : Dec.-04, Marks 8

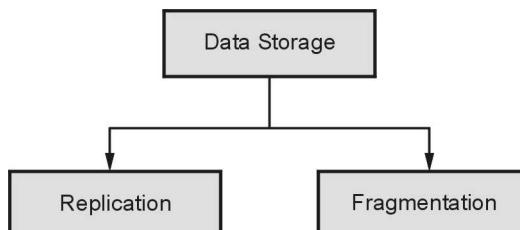
5. Discuss homogeneous and heterogeneous databases reference to distributed databases.

AU : May-03, Marks 8

5.2 Data Storage

AU : Dec.-04, May-06, Marks 8

There are two approaches of storing relation r in distributed database –



- (1) **Replication :** System maintains multiple copies of data, stored in different sites, for faster retrieval and fault tolerance.
- (2) **Fragmentation :** Relation is partitioned into several fragments stored in distinct sites.

5.2.1 Data Replication

- **Concept :** Data replication means storing a copy or replica of a relation fragments in two or more sites.
- There are two methods of data replication – (1) Full Replication (2) Partial Replication.
 - **Full Replication :** In this approach the **entire relation** is stored at **all the sites**. In this approach full redundant databases are those in which every site contains a copy of entire database.
 - **Partial Replication :** In this approach only some fragments of relation are replicated on the sites.

Advantages

- (1) **Availability :** Data replication facilitates increased availability of data.
- (2) **Parallelism :** Queries can be processed by several sites in parallel
- (3) **Faster Accessing :** The relation r is locally available at each site, hence data accessing becomes faster.

Disadvantages

- (1) **Increased cost of update :** The major disadvantage of data replication is increased cost of updated. That means each replica of relation r must be updated from all the sites if user makes a request for some updates in relation.
- (2) **Increased complexity in concurrency control :** It becomes complex to implement the concurrency control mechanism for all the sites containing replica.

5.2.2 Data Fragmentation

- **Concept :** Data fragmentation is a division of relation r into fragments $r_1, r_2, r_3, \dots, r_n$ which contain sufficient information to reconstruct relation r .
- There are two approaches of data fragmentation – (1) Horizontal Fragmentation and (2) Vertical Fragmentation.
 - **Horizontal Fragmentation :** In this approach, each tuple of r is assigned to one or more fragments. If relation R is fragmented in r_1 and r_2 fragments, then to bring these fragments back to R we must use union operation. That means $R = r_1 \cup r_2$
 - **Vertical Fragmentation :** In this approach, the relation r is fragmented based on one or more columns. If relation R is fragmented into r_1 and r_2 fragments using vertical fragmentation then to bring these fragments back to original relation R we must use join operation. That means $R = r_1 \bowtie r_2$
- For example – Consider following relation r

Student(RollNo, Marks, City)

The values in this schema are inserted as

RollNo	Marks	City
R101	55	Pune
R102	66	Chennai
R103	77	Mumbai

Fig. 5.2.1 Student Table

Horizontal Fragmentation 1 :

SELECT * FROM Student WHERE Marks >50 AND City='Pune'

We will get

R101	55	Pune
------	----	------

Horizontal Fragmentation 2 :

SELECT * FROM Student WHERE Marks >50 AND City='Mumbai'

We will get

R103	77	Mumbai
------	----	--------

Vertical Fragmentation 1 :

SELECT * FROM RollNo

R101
R102
R103

Vertical Fragmentation 2 :

SELECT * FROM city

Pune
Chennai
Mumbai

University Questions

1. What are data fragmentations ? Explain various approaches for fragmenting a relation with example.

AU : May-06, Marks 6

2. Explain in detail various approaches used for storing a relation in distributed databases.

AU : Dec.-04, Marks 8

5.3 Transaction Processing

AU : Dec.-12,14, May-18, Marks 8

In distributed system transaction initiated at one site can access or update data at other sites. Let us discuss various basic concepts used during transaction processing in distributed systems –

- **Local and Global Transactions :**

Local transaction T_i is said to be local if it is initiated at site S_i and can access or update data at site S_i only.

Global transaction T_i initiated by site S_i is said to be global if it can access or update data at site S_i, S_j, S_k and so on.

- **Coordinating and Participating Sites :**

The site at which the transaction is initiated is called coordinating site. The participating sites are those sites at which the sub-transactions are executing. For example

– If site S_1 initiates the transaction T_1 then it is called coordinating site. Now assume that transaction T_1 (initiated at S_1) can access site S_2 and S_3 . Then sites S_2 and S_3 are called participating sites.

To access the data on site S_2 , the transaction T_1 needs another transaction T_{12} on site S_2 similarly to access the data on site S_3 , the transaction T_2 needs some transaction say T_{13} on site S_3 . Then transactions T_{12} and T_{13} are called sub-transactions. The above described scenario can be represented by following Fig. 5.3.1.

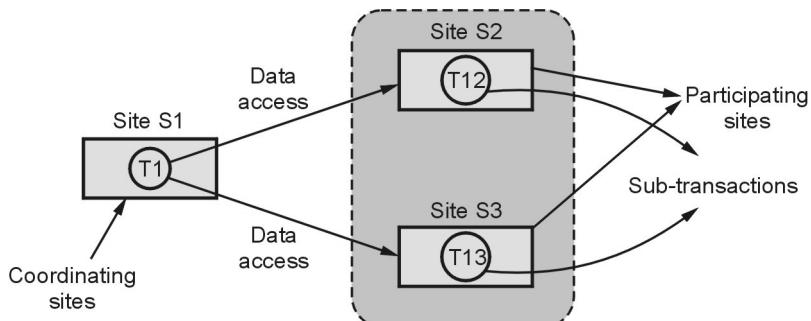


Fig. 5.3.1

- **Transaction Manager :**

The transaction manager manages the execution of those transactions (or subtransactions) that access data stored in a local site.

The tasks of transaction manager are –

- (1) To maintain the log for recovery purpose.
- (2) Participating in coordinating the concurrent execution of the transactions executing at that site.

- **Transaction Coordinator :**

The transaction coordinator coordinates the execution of the various transactions (both local and global) initiated at that site.

The tasks of Transaction coordinator are –

- (1) Starting the execution of transactions that originate at the site.
- (2) Distributing subtransactions at appropriate sites for execution

Let TC denotes the transaction coordinator and TM denotes the transaction manager, then the system architecture can be represented as

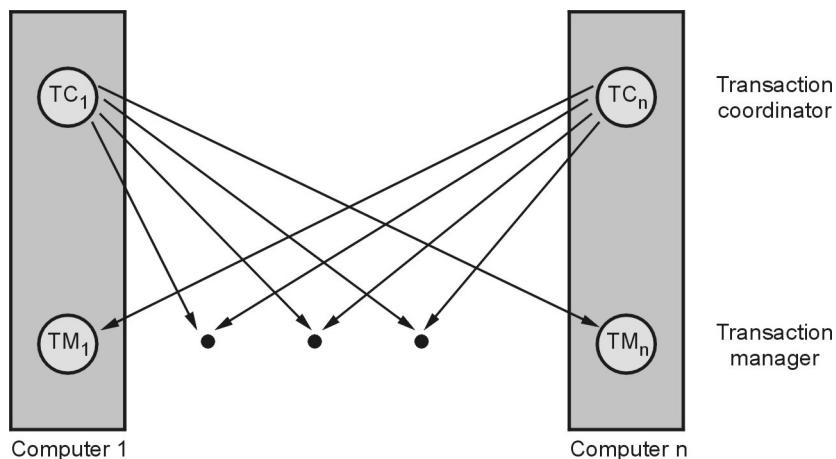


Fig. 5.3.2 System Architecture

5.3.1 Two Phase Commit Protocol

- The atomicity is an important property of any transaction processing. What is this atomicity property?? This property means either the transaction will execute completely or it won't execute at all.
- The Commit protocol ensures the atomicity across the sites in following ways –
 - i) a transaction which executes at multiple sites must either be committed at all the sites, or aborted at all the sites.
 - ii) not acceptable to have a transaction committed at one site and aborted at another
- There are two types of important sites involving in this protocol –
 - One Coordinating Site

- o One or more Participating sites

Two Phase Commit Protocol

This protocol works in two phases – i) Voting phase and ii) Decision Phase.

Phase 1 : Obtaining Decision or Voting Phase

Step 1 : Coordinator site Ci asks all participants to **prepare** to commit transaction Ti.

- o Ci adds the records <prepare T> to the log and writes the log to stable storage.
- o It then sends **prepare T** messages to all participating sites at which T will get executed

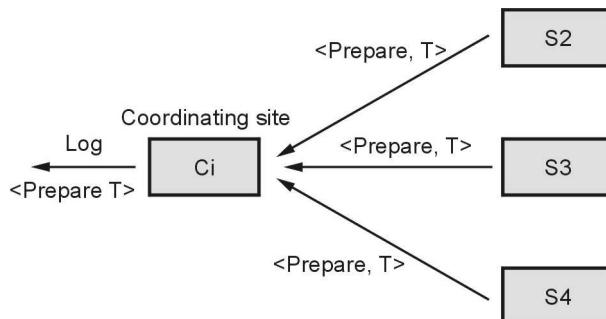


Fig. 5.3.3

Step 2 : Upon receiving message, transaction manager at participating site determines if it can commit the transaction

- o If not, add a record <no T> to the log and send **abort T** message to coordinating site Ci. Even

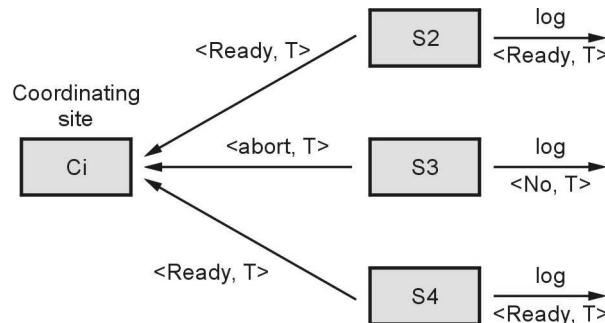


Fig. 5.3.4

- o If the transaction can be committed, then:
 - add the record <ready T> to the log
 - force all records for T to stable storage
 - send **ready T** message to Ci

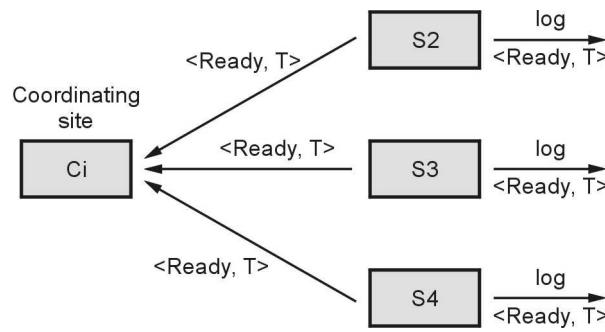
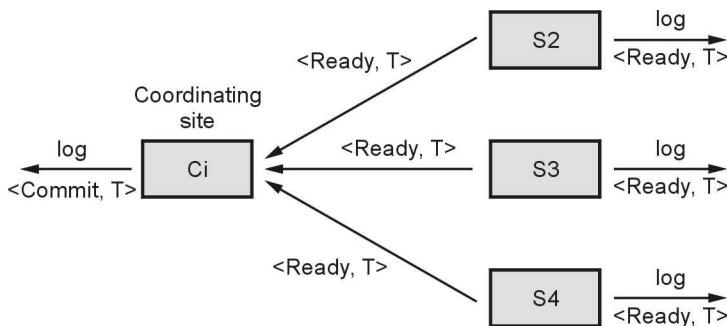


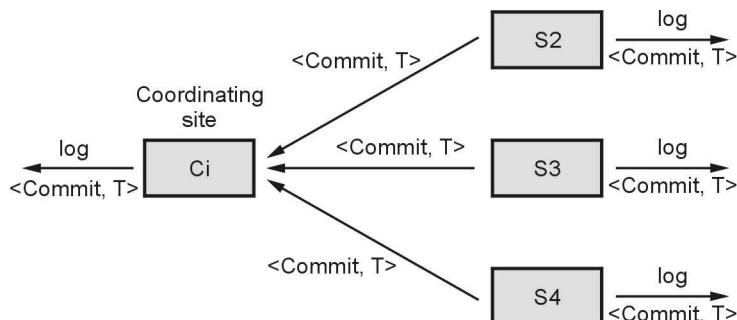
Fig. 5.3.5

Phase 2 : Recoding Decision Phase

- T can be committed if Ci received a ready T message from all the participating sites : otherwise T must be aborted.
- Coordinator adds a decision record, <commit T> or <abort T>, to the log and forces record onto stable storage. Once the record stable storage it is irrevocable (even if failures occur)



- Coordinator sends a message to each participant informing it of the decision (commit or abort)
- Participants take appropriate action locally.



Failure of Site

There are various cases at which failure may occur

(1) Failure of Participating Sites

- If any of the participating sites gets failed then when participating site S_i recovers, it examines the log entry made by it to take the decision about executing transaction.
 - If the log contains **<commit T>** record: participating site executes **redo (T)**
 - If the log contains **<abort T>** record: participating site executes **undo (T)**
 - If the log contains **<ready T>** record: participating site must consult *Coordinating site* to take decision about execution of transaction T .
 - If T committed, **redo (T)**
 - If T aborted, **undo (T)**
- If the log of participating site contains no record then that means S_i gets failed before responding to **Prepare T** message from coordinating site. In this case it must **abort T**

(2) Failure of Coordinator

- If coordinator fails while the commit protocol for T is executing then participating sites must take decision about execution of transaction T :
 - i) If an active participating site contains a **<commit T>** record in its **log**, then **T must be committed.**
 - ii) If an active participating site contains an **<abort T>** record in its **log**, then **T must be aborted.**
 - iii) If some active participating site does not contain a **<ready T>** record in its **log**, then the failed coordinator C_i cannot have decided to commit T . Can therefore abort T .
 - iv) If none of the above cases holds, then all participating active sites must have a **<ready T>** record in their logs, but no additional control records (such as **<abort T>** or **<commit T>**). In this case active sites **must wait** for coordinator site C_i to recover, to find decision.

Two phase locking protocol has blocking problem.

What is blocking Problem ?

It is a stage at which active participating sites may have to wait for failed coordinator site to recover.

The solution to this problem is to use three phase locking protocol.

5.3.2 Three Phase Commit Protocol

- The three phase locking is an extension of two phase locking protocol in which eliminates the blocking problem.
- Various assumptions that are made for three phase commit protocol are –
 - No network partitioning.
 - At any point at least one site must be up.
 - At the most k sites(participating as well as coordinating) can fail.
- Phase 1 :** This phase is similar to phase 1 of two phase protocol. That means Coordinator site Ci asks all participants to **prepare** to commit transaction Ti. The coordinator then makes the decision about commit or abort based on the response from all the participating sites.
- Phase 2 :** In phase 2 coordinator makes a decision as in 2Phase Commit which is called the **pre-commit** decision $\langle \text{Pre-commit}, T \rangle$, and records it in multiple (at least K) participating sites.
- Phase 3 :** In phase 3, coordinator sends commit/abort message to all participating sites.
- Under three phase protocol, the knowledge of pre-commit decision can be used to commit despite coordinator site failure. That means if the coordinating site in case gets failed then one of the participating site becomes the coordinating site and consults other participating sites to know the **Pre-commit** message which they possess. Thus using this pre-commit message the decision about commit/abort is taken by this new coordinating site.
- This protocol avoids blocking problem as long as less than k sites fail.

Advantage of Three Phase commit protocol

- (1) It avoid blocking problem

Drawbacks of Three Phase commit protocol

- (1) The overhead is increased.

University Questions

1. Explain with diagrammatic illustration the architecture of a distributed database management system. AU : May-18, Marks 5

2. Write short note on distributed transactions. AU : Dec.-14, Marks 8

3. Discuss in detail about two phase commit protocol. AU : Dec.-12, May 16, Marks 8

Part II : Object Based Databases**5.4 Object Database Concepts****AU : May-18, Marks 13**

- Traditional applications are designed for business applications such as inventory, employee, university, bank, library, air-line reservation systems, and so on.
- These applications require relatively simple data types that are well suited to the relational data model. But database systems were applied to a wider range of applications, such as computer-aided design and geographical information systems. Hence such systems require complex data type and processing.
- The **object based database** provide the solution to model the real world object and their behavior. It is an alternative to relational database model.

5.4.1 Complex Data Types

- Complex data types have grown more important in recent years to model some real world data attributes. For instance - **Address** can be viewed as a – i) Single string, or ii) Separate attributes for each part or iii) Composite attributes.
- Normally it is also convenient to store multivalued attributes as-it is, without creating a separate relation to store the values in first normal form.
- Complex data types can be used in the database applications such as –
 - Computers Aided Design(CAD)
 - Hypertext database
 - Multimedia and image databases.

5.4.2 Object Oriented Data Model

- Object oriented paradigm encapsulate data and corresponding methods in a single entity called **object**.
- The object oriented data model is a **logical data model** just like ER model. Loosely object can be thought as an entity of E-R model.
- Object oriented data model **helps in adapting object oriented paradigm** to database systems.
- In object oriented database, information is represented in the form of objects.
- Object oriented databases are exactly same as object oriented programming languages. If we can combine the features of relational model (transaction, concurrency, recovery) to object oriented databases, the resultant model is called as **object oriented database model**.

- The core object oriented Data model consists of following **object oriented concepts** –
 - **Object and Object Identifier** : Any real world entity is modeled as object. Associated with each object there is an unique ID maintained which is called **object identifier(OID)**. Object identifiers used to uniquely identify objects. Object identifiers are unique. No two objects have the same identifier. each object has only one object identifier.
 - **Attribute and method** : Every object has **state** and **behavior**. The **state** is a set of values for attributes of the object. The behavior is set of methods. For example – Object named **circle** has state **radius** and **computing its area** is a behavior.
 - **Class** : Class is a means of grouping all the objects which share the same set of attributes and methods. An object must belong to only one class as an instance of that class (instance-of relationship). A class is similar to an abstract data type.
 - **Class Hierarchy and Inheritance** : The class hierarchy is used to represent the Inheritance property of object oriented paradigm. The inheritance is a mechanism in which a new class is derived from existing class.

5.4.2.1 Object Structure

- Object is a fundamental unit of object oriented programming in which data value and code is encapsulated in a single unit.
- Conceptually all interactions among the objects and rest of the system are carried out by using **messages**. Thus the interface among various objects and rest of the system is **messages**. Messages can be implemented as procedure invocations.
- In general, an object has associated with it:
 - A set of **variables** that contain the data for the object. The value of each variable is itself an object.
 - A set of **messages** to which the object responds.
 - A set of **methods**, each of which is a body of code to implement each message; a method returns a value as the response to the message.
- Methods are programs written in general-purpose language with the following features –
 - Only variables in the object itself may be referenced directly.
 - Data in other objects are referenced only by sending messages.
 - Methods can be **read-only** or **update** methods. Read-only methods do not change the value of the object.

- For example – the object **Circle** can have methods such as **compute_area()**

5.4.2.2 Object Classes

- Similar objects are grouped into **classes**. In other words, object is an instance of a class. For example

```
class Student {
    /*variables*/
    Int RollNo;
    String Name;
    String address;
    /*Messages*/
    String get_name();
    String get_address();
    Int get_RollNo();
}
```

Methods can be defined separately. For example

```
int get_rollNo()
{
    Return RollNo;
}
```

5.4.2.3 Inheritance

In object oriented database schema, there can be large number of classes. Some classes are similar and can be derived from old existing classes. The classes are placed into specialization or Is-a hierarchy.

For example – Consider following ER model -

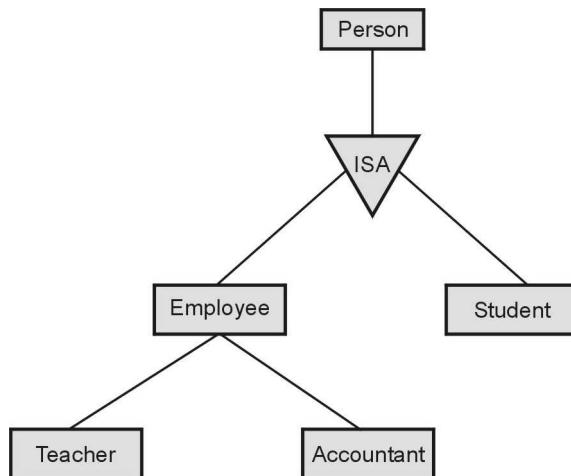


Fig. 5.4.1 Class Hierarchy

The class hierarchy can be defined in pseudo-code in which the variables associated with each class are as follows

```

class Person
{
    String name;
    String address;
}
class Student isa Person //Inheritance: Deriving child class
{
    int RollNo;
    String Course_taken;
}
class Employee isa Person
{
    date join-date;
    int salary;
}
class Teacher isa Employee
{
    String Subject;
}
class Accountant isa Employee
{
    int office_no;
}

```

- The keyword **isa** is used to indicate that a class is a specialization of another class. The specialization of a class are called subclasses. For example - Employee is a subclass of Person. Conversely, Person is a superclass of class Employee.
- Class hierarchy and inheritance of properties from more general classes.
- The important benefit of using inheritance is code-reusability. This can be achieved by means of substitutability property.
- **Substitutability** means any method of a class, say person, can be invoked equally well with any object belonging to any subclass, such as subclass Teacher of Person.

5.4.2.4 Multiple Inheritance

- In most cases, tree-structured organization of classes is adequate to describe applications. In such cases, all superclasses of a class are ancestors of descendants of another in the hierarchy. However, there are situations that cannot be represented well in a tree-structured class hierarchy.
- To avoid the conflicts between two occurrences, we use multiple inheritance.

- Multiple inheritance is an ability of class to inherit variables from multiple super classes. The class – subclass relationship is represented by directed acyclic graph(DAG). For example –

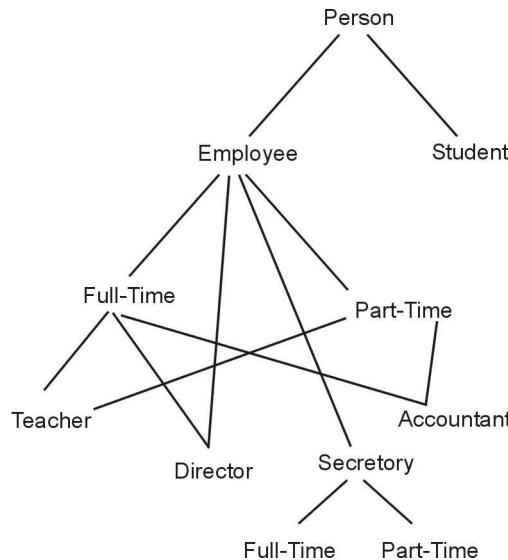


Fig. 5.4.2 Multiple inheritance

- Handling name conflicts :** When multiple inheritance is used, there is potential ambiguity if the same variable or method can be inherited from more than one superclass.

5.4.2.5 Object Identity

- For identifying the RDBMS record uniquely in the table, we use primary key associated with it. In object oriented database management system we use **Object Id** to identify the record.
- Associated with each object there is an unique ID maintained which is called **object identifier(OID)**.
- An object retains its identity even if some or all the values of the variables or definitions of methods change over time.
- The concept of object identity does not apply to tuples of relational database. It is a stronger notion of identity.
- The difference between primary of RDBMS and object ID of OODBMS is that primary key is explicit; that means it is visible to the user whereas the object ID is implicit; that means it is hidden from external world.
- It has several forms of identity :
 - Value :** The data value is used for identity. For example the primary key of the tuple in a relational database.

- **Name** : The user supplied name is used for identity. For example file name can be used for identity of object.
- **Built-in** : A notion of identity is built-into the data model or programming languages, and no user-supplied identifier is required.
- Object identity is typically implemented via a unique, **system-generated OID**. The value of the **OID is not visible to the external user**, but is **used internally** by the system to identify each object uniquely and to create and manage inter-object references.

5.4.2.6 Object Containment

The objects that contain another objects are called **composite objects** or **complex objects**.

There can be multiple levels of object containment. These levels can be represented using **containment hierarchy**.

For example - The following Fig. 5.4.3 represents the design of simple document. The document contains the text in the form of paragraphs. Each paragraph contains word. Each word is made up of some characters.

Also note that there can be some size and style for each paragraph, word and each character.

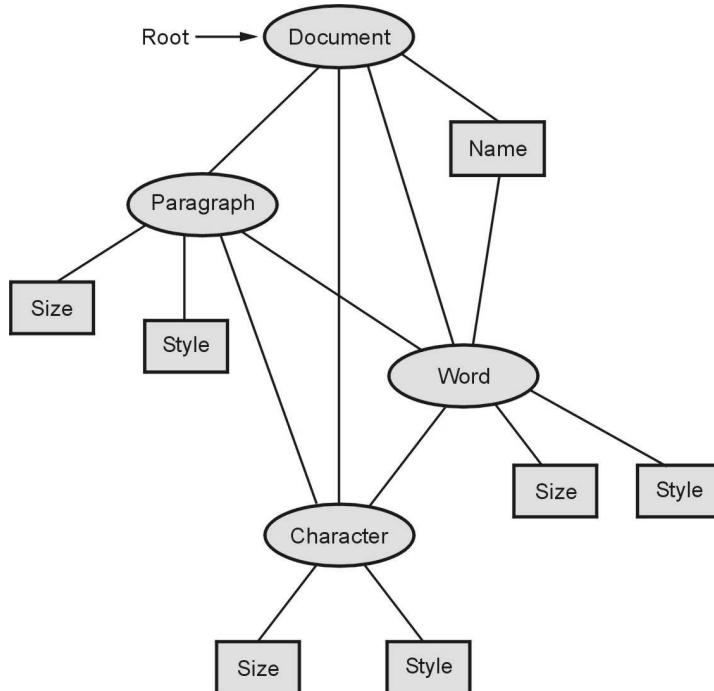


Fig. 5.4.3 Containment Hierarchy

- The links between classes must be interpreted as **is-part-of**, rather than the is-a interpretation of links in an inheritance hierarchy.
- Containment allows data to be viewed at different granularities by different users. For example - document can be viewed by a reader for simply reading purpose, the size and style of paragraphs can be viewed by DTP operator or proof editor for editing purpose.
- The containment hierarchy can be used to find all object contained in **document** object.
- In certain applications, an object may be contained in several objects. In such cases, the containment relationship is represented by a DAG rather than by a hierarchy.

5.4.3 Object Oriented Languages

- The expression of object-orientation can be done in one of two ways –
 1. The concepts of object-orientation can be used purely as a design tool, and are encoded into, For example - a relational database. Example - ER modeling.
 2. The concepts of object-orientation are incorporated into a language that is used to manipulate the database.
- There are several possible languages into which the concepts can be integrated.
 - **Object-relational systems** - Add complex types and object-orientation to relational language.
 - **Persistent programming languages** - Extend object-oriented programming language to deal with databases by adding concepts such as persistence and collections

5.4.4 Persistent Programming Languages

- Persistent Programming languages allow objects to be created and stored in a database, and used directly from a programming language.
- The persistent data is a data that continue to exist even after the program that created it has terminated.
- The persistent programming language allows data to be manipulated directly from the programming language and there is no need to go through SQL.
- Using persistent programming language, the format changes are carried out transparently by system.
- Without a persistent language format changes becomes a burden on the programmer. As it may cause more code to be written.
- This language allows objects to be manipulated in-memory.

Drawbacks

- If the programming errors occur then it might damage the database system.
- Due to complexity of programming languages, high level of optimization can not be achieved.
- It does not support declarative query as well as relational database.

5.4.4.1 Persistence of Object

Persistence of object can be achieved by following approaches –

- (1) **Persistence by Class** : Declare all objects of the class to be persistent to achieve the persistence of the objects.
- (2) **Persistence by Creation** : Introduce new syntax to create persistent objects.
- (3) **Persistence by marking** : An object that is to persist beyond program execution is marked as persistent before program termination.
- (4) **Persistence by Reference** : One or more objects can be explicitly declared as (root) persistent objects. All other objects are persistent if they are referred, directly or indirectly, from a root persistent object.

Thus it is easy to make the entire data structure persistent by merely declaring the root of the structure as persistent.

5.4.4.2 Object Identity and Pointers

- A persistent object is assigned a persistent object identifier.
- There are several degrees of permanence of object identity
 - **Intraprocedure** : Identity persists only during the execution of a single procedure, e.g., local variables within procedures.
 - **Intraprogram** : Identity persists only during the execution of a single program or query, e.g., global variables in programming languages, and main memory or virtual memory pointers.
 - **Interprogram** : Identity persists from one program execution to another, e.g., pointers to file system data on disk but may change if the way data is stored in the file system is changed.
 - **Persistent** : Identity persists not only among program executions but also among structural reorganizations of the data. This is the persistent form of identity required for object-oriented systems.
- In persistent extension of C++, object identifiers are implemented as "persistent pointers" which can be viewed as a pointer to an object in the database.

5.4.4.3 Storage and Access of Persistent Objects

- **Storage** : The storage of code and data of persistent object can be done as follows

- The source code that implements the methods should be stored in the database as a part of the schema, along with type definition.
- The data is stored individually for each object.
- **Access :** For finding the persistent objects –
 - Give **names** to objects like we give names to files. But it works only for small sets of objects.
 - Expose object identifiers(**OID**) or **persistent pointers** to the objects.
 - Store the collections of object and allow programs to **iterate over the collections** to find required objects. The collections can be modeled as objects of a **collection type**.

University Question

1. Explain the necessary characteristics a system must satisfy to be considered as an object oriented database management system.

AU : May-18, Marks 13

5.5 Object Relational Features

AU : Dec.-16, Marks 13

The relational model with object database enhancement is called as object relational model.

Following are some object database features that can be included in SQL

- (1) For specifying the complex objects some **type constructors** can be added. For example – **row type** corresponds to tuple in the schema, **array type** is for specifying the collection. Some other types of constructors are **set**, **list**, and **bag**.
- (2) The mechanism of for specifying the **object identity** through the use of **reference type** is be included.
- (3) The relationships can be specified using **reference**.
- (4) **Encapsulation of operation** is provided through the mechanism of user defined types(UDT). Similarly the concept of user defined routines allow the definition of general methods.
- (5) The mechanism of **inheritance** can be provided using the keyword **UNDER**.

Example 5.5.1 Suppose that you have been hired as a consultant to choose a database system for your client's application. For each of the following, applications, state what type of database system (relational, persistent programming language-based OODB, object relational; do not specify a commercial product) you would recommend. Justify your recommendation.

- (i) A computer-aided design system for a manufacturer of airplanes.
- (ii) A system to track contributions made to candidates for public office.
- (iii) An information system to support the making of movies.

AU : Dec.-16, Marks 13

Solution :

(i) A Computer-Aided Design (CAD) system for a manufacturer of airplanes

- An **OODB system** would be suitable for this.
- That is **because** CAD requires complex data types, and being computation oriented, CAD tools are typically used in a programming language environment needing to access the database.

(ii) A system to track contributions made to candidates for public office

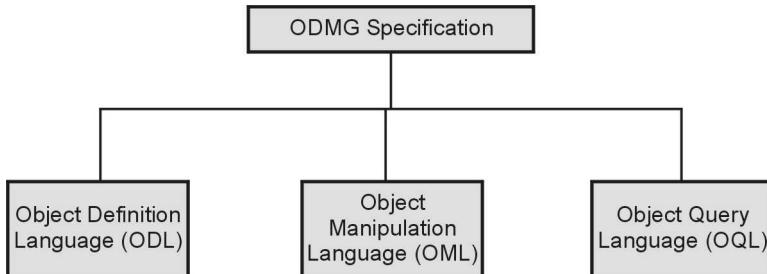
- A **relational system** would be used for this.
- **Because** data types are expected to be simple, and it requires a powerful querying mechanism.

(iii) An information system to support the making of movies

- For this system, will be extensive use of multimedia and other complex data types. But queries are probably simple.
- Hence an **object relational system** can be used to model this type of system.

5.6 ODMG Object Model

- The ODMG stands for Object Database Management Group.
- This group has come up with the specification for using object oriented database in specific manner. This specification is popularly known as ODMG object Model.
- The ODMG specification has three parts -



ODL : The Object Definition Language is similar to Data Definition Language(DDL) in SQL.

OQL : The object Query Language is similar to the concept of Data Manipulation Language(DML) but it provides the support for SQL SELECT and JOIN form of queries and does not provide support for SQL INSERT, UPDATE and DELETE.

OML : It is an extension to OQL but it is in generalized form that can be used in object oriented languages. It is language specific. It provides support for SQL INSERT, UPDATE or DELETE operations.

5.6.1 ODL

(1) Declaring Classes Using ODL

For declaring classes using ODL we need to use –

- (i) keyword **interface**
- (ii) The name of the class
- (iii) The list of attributes of the class declared using the keyword **attribute**

For example –

```
interface Student
{
    attribute integer RollNo;
    attribute string Name;
    attribute string address;
    attribute string course_id;
};
```

(2) Declaring Relationships

The SQL makes use of foreign key concept to establish relationships between two tables. In the similar manner ODL makes use of keyword **relationship** to declare the relationship among two relational schema.

Suppose we have two classes namely X and Y. While declaring class X we want to establish relationship with class Y then we add following statement in class X

relationship Y *x_y_relation*;

where *x_y_relation* is a name of relationship.

Thus class X is linked with class Y.

For example –

```
interface Student
{
    attribute integer RollNo;
    attribute string Name;
    attribute string address;
    attribute string course_id;
    relationship Course Stud_Course_rel;
};
```

(3) Declaring Key

As we know that the keys are used to identify the tuple in the relationship. Some of the attribute(column) can be declared as a key to identify the record uniquely. In ODL we use the keyword **key** to make particular attribute a key.

For example -

```
interface Student(Key RollNo)
{
    attribute integer RollNo;
    attribute string Name;
    attribute string address;
    attribute string course_id;
};
```

5.6.2 OQL

- Object Query Language (OQL) is a query language standard for object-oriented databases modeled after SQL.
- The following rules apply to OQL statements
 - All complete statements must be terminated by a semi-colon.
 - A list of entries in OQL is usually separated by commas but not terminated by a comma(,).
 - Strings of text are enclosed by matching quotation marks.
- Basic From of OQL: SELECT, FROM, WHERE

Syntax

```
SELECT <list of values>
FROM <list of collections and variable assignments>
WHERE <condition>
```

Where the SELECT clause extracts those elements of a collection meeting a specific condition. By using the keyword DISTINCT duplicated elements in the resulting collection get eliminated. Collections in FROM can be either extents (persistent names - sets) or expressions that evaluate to a collection (a set).

Example : Give the names of people who are older than 30 years old:

```
SELECT SName: p.name
FROM p in People
WHERE p.age > 30
```

DOT notations and Path expressions

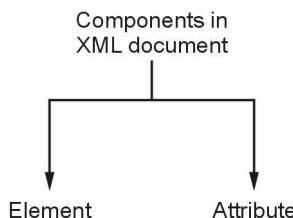
We use the dot notation and path expressions to access components of complex values.
For example

```
ta.salary -> real
t.students -> set of tuples of type tuple(name: string, fee: real) representing students
t.salary -> real
```

Part III : XML Databases

5.7 XML Hierarchical Model

- 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**.
- There are **two key components** in XML document - **Element** and **Attribute**.



(1) Element

- In XML 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.
- The **syntax** of writing any element for **opening tag** is
`<element name>`
- The **syntax** of writing any closing element for **closing tag** is
`</element name >`
- **For example -**
 - `<Student> I am learning Computer Engineering </Student>`
 - Every XML must have a single **root element**.
- The other elements are called the container elements. There can be one or more elements inside the container elements such elements are called the **child elements**.
- An **empty tag** can be defined by putting a / (forward slash) before closing bracket.
- **For example -**
`<Student/>`

XML Document

```
<Person> ← Root Element
    <Personal-Info> ← Container Element
        <Name>Anuradha</Name> ← Child Element
        <City> Pune</City>
    </Personal-Info>
    <Hobby>
        <first>Reading</first>
        <second>Programming</second>
        <third>Singing</third>
    </Hobby>
</Person>
```

The above XML document can be represented as

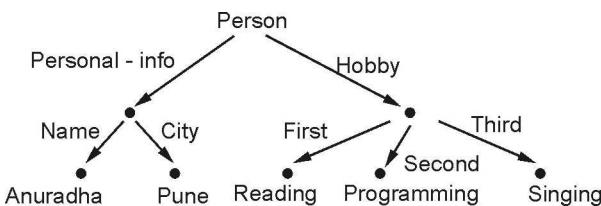


Fig. 5.7.1 Sample tree model or hierarchical model

That is why XML is called Tree model or hierarchical model.

(2) Attribute

- The attributes must be enclosed within **double quotes** or a **single quote**.
- For example -

```
<Person flag="true"> ← attribute
<name>Parth</name>
<grade>B</grade>
</Person>
```

5.7.1 Types of XML Documents

There are three main types of XML documents –

(1) Data Centric XML documents :

- These documents have many small data items that follow specific structure.
Hence data can be extracted from specific structure.
- These documents follow predefined schema that defines tag names.

(2) Document Centric XML documents

- These are type of XML documents that contain large amounts of text, such as articles of book.
- There are very few or no structured data elements in these documents.

(3) Hybrid XML documents

- These documents contain the parts that contain unstructured data.
- These documents may or may not have predefined schema.

XML documents that do not follow predefined schema of element names and corresponding tree structure are known as **schemaless XML Documents**

5.8 DTD

AU : Dec.-16, Marks 15

- The DTD stands for **Document Type Definition**.
- 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.
- 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). Any parsable character data should not contain the markup characters. The markup characters are < or > or &. If we want to use less than, greater than or ampersand characters then make use of < ; > or &

Example

Consider following xml document -

```
<?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>
```

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>
```

Example 5.8.1 : Create a DTD for a catalog of four stroke motorbikes, where each motor bike has the following child elements -make, model, year, color, engine, chassis number and accessories. The engine element has the child elements engine number, number of cylinders, type of fuel. The accessories element has the attributes like disc brake,auto-start and radio, each of which is required and has the possible values yes and no.

Solution :

automobile.dtd

```
<!ELEMENT Catalog (MotorBike)*>
<!ELEMENT MotorBike (make, model, year,color, engine,chassis-num,accessories ) >
<!ELEMENT make (#PCDATA)>
<!ELEMENT model (#PCDATA)>
<!ELEMENT year (#PCDATA)>
```

```
<!ELEMENT color (#PCDATA)>
<!ELEMENT engine (engin-num,cylinders-num,fuel-type)>
<!ELEMENT chassis-num (#PCDATA)>
<!ELEMENT accessories (#PCDATA)>
<!ATTLIST accessories disc-brake (yes|no) #REQUIRED
    auto-start (yes|no) #REQUIRED
    radio (yes|no) #REQUIRED>
```

Example 5.8.2 : Give the DTD or XML Schema for an XML representation of the following nested-relational schema :

Emp = (ename, ChildrenSet setof(Children), SkillsSet setof(Skills))
Children = (name, Birthday)
Birthday = (day, month, year)
Skills = (type, ExamsSet setoff(Exams))
Exams = (year, city).

AU : Dec.-16, Marks 15

Solution : The DTD can be as given below

```
<!DOCTYPE db [
<!ELEMENT emp (ename, children*, skills*)>
<!ELEMENT children (name, birthday)>
<!ELEMENT birthday (day, month, year)>
<!ELEMENT skills (type, exams+)>
<!ELEMENT exams (year, city)>
<!ELEMENT ename( #PCDATA )>
<!ELEMENT name( #PCDATA )>
<!ELEMENT day( #PCDATA )>
<!ELEMENT month( #PCDATA )>
<!ELEMENT year( #PCDATA )>
<!ELEMENT type( #PCDATA )>
<!ELEMENT city( #PCDATA )>
] >
```

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.

5.9 XML Schema

- The XML schemas are used to represent the **structure of XML document**.
- The goal or purpose of XML schema is to define the building blocks of an XML document. These can be used as an alternative to XML DTD.
- The XML schema language is called as **XML Schema Definition (XSD) language**.
- XML schema defines elements, attributes, elements having child elements, order of child elements. It also defines fixed and default values of elements and attributes.
- XML schema also allows the developer to use **data types**.

How to write a simple schema ?

Step 1 : We will first write a simple xsd file in which the desired structure of the XML document is defined. I have named it as *StudentSchema.xsd*

This file contains the complex type with four child simple type elements.

XML Schema [StudentSchema.xsd]

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Student">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name" type="xs:string"/>
        <xs:element name="address" type="xs:string"/>
        <xs:element name="std" type="xs:string"/>
        <xs:element name="marks" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Script Explanation :

- 1) The **xs** is qualifier used to identify the schema elements and types.
- 2) The document element of schema is **xs:schema**. The **xs:schema** is the root element. It takes the attribute **xmlns:xs** which has the value **http://www.w3.org/2001/XMLSchema**. This declaration indicates that document should follow the rules of XML schema. The XML schema rules are defined by the W3 recommendation in year 2001.

- 3) Then comes **xs:element** which is used to define the xml element. In above case the element **Student** is of complex type who have four child elements : *name, address, std and marks*. All these elements are of simple type **string**.

Step 2 : Now develop XML document in which the desired values to the XML elements can be given. I have named this file as **MySchema.xml**

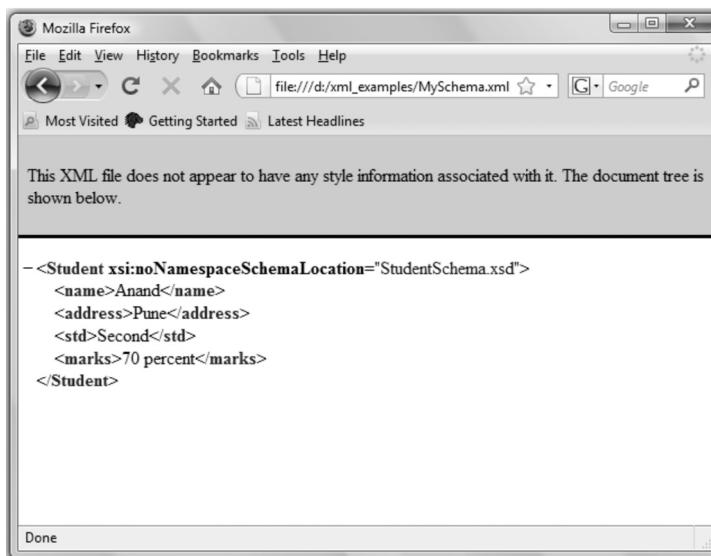
XML Document [MySchema.xml]

```
<?xml version="1.0" encoding="UTF-8"?>
<Student xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="StudentSchema.xsd">
<name>Anand</name>
<address>Pune</address>
<std>Second</std>
<marks>70 percent</marks>
</Student>
```

Script Explanation :

- 1) In above XML document, the first line is typical in which the version and encoding is specified.
- 2) There is **xmlns:xsi** attribute of document which indicates that XML document is an instance of XML schema. And it has come from the namespace "http://www.w3.org/2001/XMLSchema-instance."
- 3) To tie this XML document with the some Schema definition we use the attribute **xsi:noNamespace SchemaLocation**. The value that can be passed to this attribute is the name of xsd file "StudentSchema.xsd".
- 4) After this we can define the child elements.

Step 3 : See the output in the browser window as follows -



Advantages of schema

1. The schemas are more specific.
2. The schema provide the support for data types .
3. The XML schema is written in XML itself and has a large number of built in and derived types.
4. The XML schema is the W3C recommendation. Hence it is supported by various XML validator and XML processors.

Disadvantages of schema

1. The XML schema is complex to design and hard to learn.
2. The XML document cannot be if the corresponding schema file is absent.
3. Maintaining the schema for large and complex operations sometimes slows down the processing of XML document.

Difference between DTD and XML Schema

Sr. No.	DTD	Schema
1.	DTD are basic for defining the structural components of XML.	Schema are specific for complex operations while defining the structural components of XML.
2.	DTD does not have ability to define the type of data.	Schemas have ability to define the type of data.
3.	Simple and small applications can be built using DTD	Large number of complex applications can be built using schema.
4.	DTDs allow inline definitions	XML schema does not allow inline definitions.
5.	DTD is not strongly typed language.	XML schema is strongly typed.

5.10 XQuery

AU : May-17, Marks 5

- XQuery is used to query the XML database. It is just similar to SQL to database.
- XQuery is supported by all major databases.
- Its main task is to get information out of XML databases.
- **Example of XQuery**

Step 1: Consider following simple XML document(this document is saved by name **students.xml**) that stores student information

```
<students>
  <student>
```

```

<name>AAA</name>
<age>15</age>
<student>
  <name>BBB</name>
  <age>16</age>
<student>
  <name>CCC</name>
  <age>17</age>
<student>
  <name>DDD</name>
  <age>18</age>
</student>
<student>
  <name>EEE</name>
  <age>19</age>
<student>
  <name>FFF</name>
  <age>20</age>
</students>

```

Step 2 : We can write the XQuery as follows –

```

for $x in doc("students.xml")/students/student  ← Here x is a variable
where $x/age>16   ← Query
return $x/name    ← returns names of students satisfying the query

```

This query will return the names of students having the age >16.

Step 3 : The result will be –

CCC
DDD
EEE
FFF

Example 5.10.1 : Consider following nested-relational schema:

Emp = (ename, ChildrenSet setof(Children), SkillsSet setof(Skills))

Children = (name, Birthday)

Birthday = (day, month, year)

Skills = (type, ExamsSet setoff(Exams))

Exams = (year, city).

(1) Find the names of all employees who have a child who has a birthday in March.

(2) Find those employees who took an examination for the skill type “painting” in

the city “Chennai”.

(3) List all skill types in Emp.

For performing above mentioned tasks write XQuery statements.

Solution : Assume we have created DTD for given schema (Refer Example 5.8.2 for DTD)

(1) Find the names of all employees who have a child who has a birthday in March.

```
for $e in /db/emp,
$m in distinct($e/children/birthday/month)
where $m = 'March'
return $e/ename
```

(2) Find those employees who took an examination for the skill type “painting” in the city “Chennai”.

```
for $e in /db/emp
$s in $e/skills[type='painting']
$exam in $s/exams
where $exam/city= 'Chennai'
return $e/ename
```

(3) List all skill types in Emp.

```
for $t in distinct (/db/emp/skills/type)
return $e/ename
```

Advantages of XQuery

- Using XQuery both hierarchical and tabular data can be retrieved.
- XQuery can be used to query tree and graphical structure.
- XQuery can be directly used to build web pages.
- XQuery can be used to transform XML documents.

University Question

1. Explain the process of querying XML data with an example.

AU : May-17, Marks 5

Part IV : Information Retrieval

5.11 IR Concepts

What is Structured and Unstructured Data?

- Structured data is a form of data in which the information is in most organized form. For example- Student record such as

$$\text{Student(RollNo, Name, Address, Marks)}$$
- Unstructured data is more like **human language**. It doesn't fit nicely into relational databases. For example - emails, text documents (Word docs, PDFs,

etc.), social media posts, videos, audio files, and images all of these represent unstructured form of data.

- Useful information is locked up in all unstructured data. The information in emails and social media, for example, holds important insight that can be used for operational intelligence, marketing intelligence, and more.
- There are various ways to retrieve useful information from unstructured data. The most popularly used technique is use of **queries**.
- **Definition of Information Retrieval :** Information Retrieval(IR) is the process of retrieving documents from a collection in response to a query submitted by a user.

Concept of Query

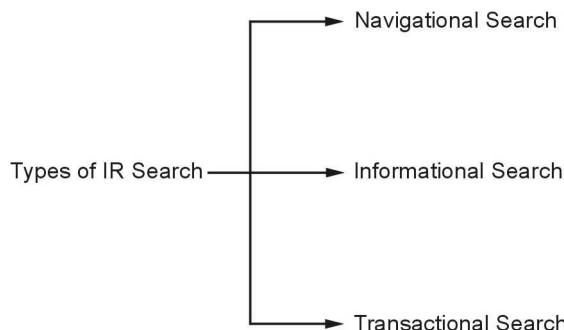
- Information Retrieval system is beyond database systems that do not limit its user to use specific query language, nor do they expect its user to know the structure of schema.
- User can make use of free form of search request which is called as query to retrieve information from IR system. This query is also called as keyword search query.

Characteristics of IR Systems

- The IR system is characterized at different levels : i) By types of users ii) Types of data iii) Types of information need.
- **Types of Users :**
 - There are two types of users – expert user and layperson user.
 - The **expert user** is a kind of user who is searching for specific information that is clear in his mind. He/She also knows the form of query to be submitted to get the desired information. For example – User who wants to get the information about particular book.
 - The **layperson** user is a user with generic information need. This type of user can not create highly relevant queries for search. For example – Student searching for information on particular topic.
- **Types of Data :**
 - Search systems can be tailored to specific types of data.
 - For example, the problem of retrieving information about a specific topic may be handled more efficiently by customized search systems that are built to collect and retrieve only information related to that specific topic.
 - These domain specific or vertical IR systems are not as large as generic WWW. The information can be retrieved efficiently from such IR systems.

- **Types of Information Need :**

- Normally three types of search can be made for information retrieval.



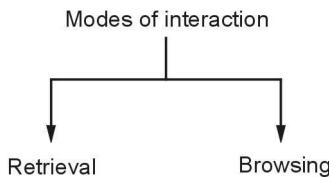
- **Navigational Search :** It refers to finding a particular piece of information that user needs quickly. For example – Finding the site of 'Anna University'
- **Informational Search :** It refers to finding current information about some topic. For example information finding about current News.
- **Transactional Search :** It refers to reach a site in which further interaction happen. For instance – making online reservation, purchasing product online.

Difference between Database Systems and Information Retrieval Systems

Sr. No.	Database System	IR System
1.	It makes use of structured data.	It makes use of unstructured data.
2.	It is schema driven.	There is no fixed schema in IR system.
3.	Relational data model is used.	Free-form Query model is used.
4.	Query returns data.	Search request returns list or pointers to documents that may contain the desired information.
5.	Results are based on exact matching.	Results are based on approximate matching.

5.11.1 Modes of Interaction

- There are two main modes of interaction with IR system – 1. Retrieval and 2. Browsing

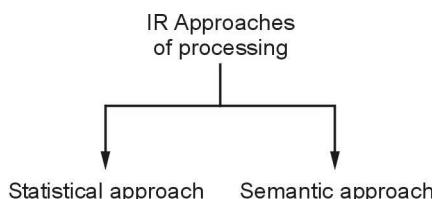


- **Definition of Retrieval :** Retrieval is concerned with the extraction of relevant information from a repository of documents through an IR query.
- **Definition of Browsing:** Browsing signifies the activity of a user visiting or navigating through similar or related documents based on the user's assessment of relevance. It is flexible mode of interaction.
- Consider following scenario –“ A user types a query **Programming languages** and he gets list of documents containing the term Programming language. While going through these pages he/she might click on the word **Java**. Further while going through the information on Java, he might come across the word **web service** and then he could pick up the web service for weather forecast which might be of his/her interest.” This complete scenario of obtaining the desired information is called **web browsing**.
- **Hyperlinks** are used to interconnect Web pages and are mainly used for browsing.
- **Anchor texts** are text phrases within documents used to label hyperlinks and are very relevant to browsing.
- **Web search** combines both the activities i.e. retrieval and browsing.
- **Web search engine** maintains an **indexed repository of web pages**. The most relevant web pages are returned to the user if possible in descending order of their relevance.(That means most relevant web page is list at the top, then less relevant, then lesser relevant and so on).

5.11.2 IR Processing

IR Processing Approaches

There are two main approaches of information Retrieval -



(1) Statistical Approach :

- In this approach, the documents are first **analyzed** and **broken down into chunks** of text.
- Each word(chunk) is **counted** for its relevance.
- These words are then **compared** against the query to test the significant **degree of match**.

- Based on this matching, the **ranked list of documents** containing these words is **presented to the user**.

(2) Semantic Approach :

- In this approach the **knowledge base technique of information retrieval** is used.
- The syntactical, lexical, sentential, discourse based and pragmatic level of words are used to prepare knowledge base for understanding.
- In practice, semantic approaches also apply some form of statistical analysis to improve retrieval process.

Generic IR Framework

Following Fig. 5.11.1 represents the various stages involved in an IR processing.

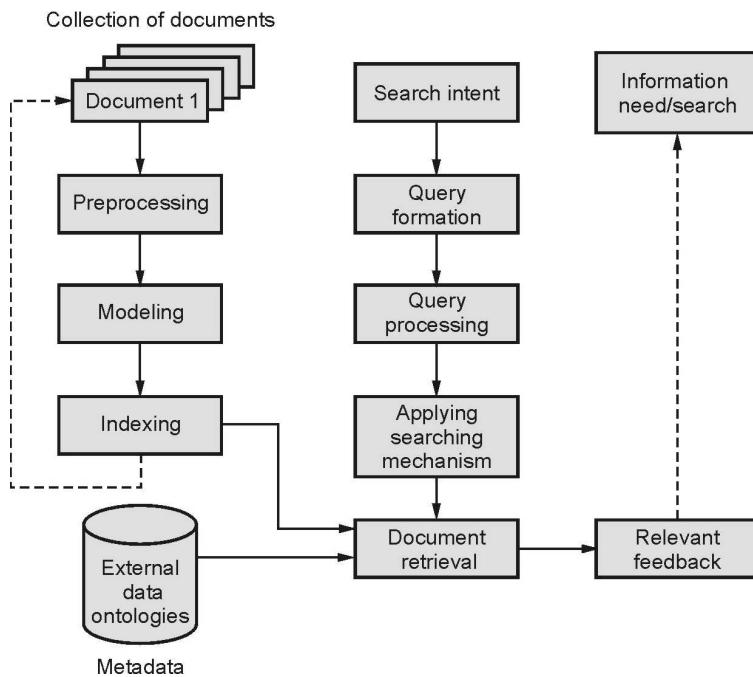


Fig. 5.11.1 Generic IR framework

The stages at the left – such as preprocessing, modeling, indexing are all offline process which prepare a set of documents for efficient retrieval.

The steps involved in query formation, query processing, searching mechanism, document retrieval, and relevance feedback are shown on the right.

5.12 Retrieval Models

(1) Boolean Model

- In this model, the queries are using operators such as AND, OR , NOT.

- The documents are represented as set of terms.
- In this model retrieval and relevance are based on “exact match” with query. The partial matches retrieved documents not ranked.
- There is no concept such as ranking of documents. All the retrieved documents are considered as equally important.
- These models make it easy to associate metadata information and write queries that match the contents of the documents as well as other properties of documents, such as date of creation, author, and type of document.

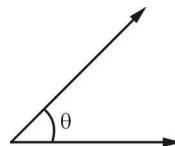
(2) Vector Space Model

- It is an algebraic model for representing text documents.
- The vector space model provides a framework in which term weighting, ranking of retrieved documents, and relevance feedback are possible.
- In this model documents and queries are both vectors.

$$\bar{d} = (w_{i,1}, w_{i,2}, \dots, w_{i,j})$$

Where each $w_{i,j}$ is a weight for term j in document i.

- This model is based on bags of words representation.
- Similarity of a document vector to a query vector = cosine of the angle between them.



$$\text{Sim}(d_i, q) = \cos \theta (x \cdot y = |x| |y| \cos \theta)$$

$$\frac{d_i \cdot q}{|d_i| |q|} = \sum_j \frac{w_{i,j} \times w_{q,j}}{\sqrt{\sum_j w_{i,j}^2 \sum_j w_{q,j}^2}}$$

- Cosine is a normalized dot product.
- Documents ranked by decreasing cosine value $\text{sim}(d,q) = 1$ when $d = q$ $\text{sim}(d,q) = 0$ when d and q share no terms.
- In the classic vector space model, the term-specific weights in the document vectors are products of local and global parameters. The model is known as **term frequency-inverse document frequency model**. The weight vector for document d is $v_d = [w_{1,d}, w_{2,d}, \dots, w_{N,d}]^T$

$$W_{t,d} = tf_{t,d} \cdot \log \frac{|D|}{|\{d' \in D | t \in d'\}|}$$

and

- $tf_{t,d}$ is term frequency of term t in document d (a local parameter)
- $\log \frac{|D|}{|\{d' \in D | t \in d'\}|}$ is inverse document frequency (a global parameters). $|D|$ is the total number of documents in the document set; $|\{d' \in D | t \in d'\}|$ is the number of documents containing the term t .

(3) Probabilistic Model

- The vector space model assumes that those documents closer to the query in cosine space are more relevant to the query vector.
- In the probabilistic model, a more concrete and definitive approach is taken: ranking documents by their estimated probability of relevance with respect to the query and the document.
- In the probabilistic framework, the IR system has to decide whether the documents belong to the relevant set or the non-relevant set for a query.
- To make this decision, it is assumed that a predefined relevant set and nonrelevant set exist for the query, and the task is to calculate the probability that the document belongs to the relevant set and compare that with the probability that the document belongs to the nonrelevant set.
- Given the document representation D of a document, estimating the relevance R and nonrelevance NR of that document involves computation of conditional probability $P(R|D)$ and $P(NR|D)$. These conditional probabilities can be calculated using **Bayes' Rule**
 - $P(R|D) = P(D|R) \times P(R)/P(D)$
 - $P(NR|D) = P(D|NR) \times P(NR)/P(D)$

(4) Semantic Model

- In semantic models, the process of matching documents to a given query is based on concept level and semantic matching instead of index term (keyword) matching.
- This allows retrieval of relevant documents that share meaningful associations with other documents in the query result.
- Semantic approaches include different levels of analysis, such as morphological, syntactic, and semantic analysis, to retrieve documents more effectively.

- In **morphological analysis** the parts of query such as noun, verbs, adjective and so on are analyzed.
- In **syntactical analysis** complete phrases in the document are parsed and then analyzed.
- In **semantic analysis** to resolve the ambiguities in the words the synonyms are used. For this analysis complex knowledgebase and information retrieval heuristic is used.
- These systems often require techniques from **artificial intelligence** and **expert systems**. Knowledge bases like **Cyc** and **WordNet** have been developed for use in knowledge-based IR systems based on semantic models.

5.13 Queries in IR Systems

Various types of queries used for information retrieval are as given below -

(1) Keyword Queries

- This is the simplest and most commonly used form of queries.
- Various keywords are used for searching the desired documents. These keywords are combined using AND, OR operators.
- The query containing keyword “Java Programming” will return the documents containing the words “Java” and “Programming”
- Some systems remove the most commonly used words such as a , the , of and so on.
- All retrieval models support for keyword queries.

(2) Boolean Queries

- Some IR systems allow using of boolean operators such as AND, OR, NOR for formulation of query.
- AND(or dot operator) means both the terms are required, OR(+ operator) means either term1 or term2 must be present and NOT(–) means list the documents not containing particular term.
- Complex Boolean queries can be built out of these operators and their combinations, and they are evaluated according to the classical rules of Boolean algebra.
- No ranking is possible, because a document either satisfies such a query (is “relevant”) or does not satisfy it.
- A document is retrieved for a Boolean query if the query is logically true as an exact match in the document.

(3) Phrase Queries

- A phrase query consists of a sequence of words that makes up a phrase. The phrase is generally enclosed within double quotes.
- Each retrieved document must contain at least one instance of the exact phrase.
- It is more restrictive form of queries. For example “Java Programming” will result the list of documents containing both the terms Java and Programming.

(4) Proximity Queries

- Proximity search refers to a search that accounts for how close within a record multiple terms should be to each other.
- Each search engine can define proximity operators differently, and the search engines use various operator names such as NEAR, ADJ(adjacent), or AFTER.

(5) Wildcard Queries

- Wildcard searching is generally meant to support regular expressions and pattern matching-based searching in text.
- For example the word data* will return Database, dataset, datastructure and so on.

(6) Natural Language Queries

- There are a few natural language search engines that aim to understand the structure and meaning of queries written in natural language text, generally as a question or narrative.
- The system tries to formulate answers for such queries from retrieved results.
- Some search systems are starting to provide natural language interfaces to provide answers to specific types of questions. Such questions are usually easier to answer.

5.14 Two Marks Questions with Answers**Q.1 Define Distributed Database Management system.****AU : May-08,18, Dec.-16**

Ans. : A distributed database system consists of loosely coupled sites(computer) that share no physical components and each site is associated a database system.

Q.2 What are two approaches to store a relation in the distributed database?**AU : May-04**

Ans. : (1) **Replication** : System maintains multiple copies of data, stored in different sites, for faster retrieval and fault tolerance.

(2) **Fragmentation** : Relation is partitioned into several fragments stored in distinct sites.

Q.3 What are various fragmentations? State various fragmentations with example.**AU : Dec.-17**

Ans. : There are two types of fragmentations – horizontal fragmentation and vertical fragmentation

Example – Refer section 5.2.2.

Q.4 What are the advantages of distributed databases ?**AU : Dec.-04, May-08**

Ans. :

- (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.

Q.4 List out the reasons for development of distributed databases.**AU : May-06**

Ans: Following are the reasons for development of distributed databases –

- (1) To control the data present at geographically different sites.
- (2) To obtain highly available and reliable data processing systems

Q.5 Difference between Homogeneous and Heterogeneous Schema

Ans. :

Homogeneous Schema	Heterogeneous Schema
It contains identical software.	It contains different software.
It shares global schema.	It has different schemas.
Each site provides part of its autonomy in terms of right to change or software.	Each site maintains its own right to change the schema or software.
Due to same schema- there is no problems in query processing.	Due to different schemas, there are lot of problems in query processing.

Q.6 What are the advantages of fragmentation ?

Ans. :

- (1) It allows parallel processing on fragments of a relation.
- (2) It allows a relation to be split so that tuples are located where they are most frequently accessed.

Q.7 Give an example of two phase commit protocol.

AU : Dec.-15

Ans. : Refer Section 5.3.1.

Q.8 How does the concept of an object in the object oriented model differ from the concept of an entity in an entity relationship model ?

AU : Dec.-16

Ans. : • An entity is simply a collection of variables or data items.

- An object is an encapsulation of data as well as the methods (code) to operate on the data.
- The data members of an object are directly visible only to its methods.
- The outside world can gain access to the object's data only by passing pre-defined messages to it, and these messages are implemented by the methods.

Q.9 State the functions of XML Schema.

AU : Dec.-17

Ans. : • The XML schemas are used to represent the **structure of XML document**.

- The goal or purpose of XML schema is to define the building blocks of an XML document. These can be used as an alternative to XML DTD.



For Semester - IV (CSE/IT) . . .



FOR COLLEGE ORDER

contact : Mr. Subhash :
044-26150904, 7904404794

Technical Books Distributors

7/170, 3rd street, Imayam Colony,
Near SBOA Matriculation School,
Anna Nagar West, **Chennai** - 600101.
Ph.- (044)26150904/ 43838653
Mobile - 08956955466

jayashreedeshpande10@gmail.com

FOR BOOK SELLER's ORDER

contact : Bishan :
9444001782

Technical Publications

7/169, 3rd street, Imayam Colony,
Near SBOA Matriculation School,
Anna Nagar West, **Chennai** - 600101.
Mobile - 09444001782, 04448510421
Bishan - 08610195454

sales@technicalpublications.org

ONLINE SELLER



ISBN 978-93-332-2129-0



Find us



Website : www.technicalpublications.org

<https://www.facebook.com/technicalpublications>

Amit Residency, Office No.1, 412, Shaniwar Peth, Pune - 411030, M.S. INDIA.
Ph.: + 91-020-24495496/97, Telefax : + 91-020-24495497
Email : sales@technicalpublications.org Website : www.technicalpublications.org