MAILAM ENGINEERING COLLEGE

Mailam (Po), Tindivanam (Tk.), Villupuram (Dt.) Pin: 604 304
Approved by AICTE, New Delhi, Affiliated to Anna University, Chennai and
Accredited by National Board of Accreditation (NBA) & TATA Consultancy Services (TCS)

CS3492- DATABASE MANAGEMENT SYSTEMS
[REGULATION-2021]

STUDY MATERIAL

DEPARTMENT OF INFORMATION TECHNOLOGY

NAME OF THE STUDENT:……

REGISTER NUMBER:……

YEAR / SEM:……

ACADEMIC YEAR:……

PREPARED BY
Mrs. C. Ramya, AP/IT
Ms. M. Subathra, AP/IT

MAILAM ENGINEERING COLLEGE

MAILAM (PO), Villupuram (Dt.) Pin: 604 304

(Approved by AICTE New Delhi, Affiliated to Anna University Chennai

& Accredited by National Board of Accreditation (NBA) New Delhi)

MAILAM ENGINEERING COLLEGE

DEPARTMENT OF INFORMATION TECHNOLOGY

STAFF NAME: Mrs.C.RAMYA, AP/ IT, Ms.M.SUBATHRA, AP/IT

CLASS/SEM: II-IT /IV

SUBJECT CODE/NAME: CS3492/DATABASE MANAGEMENT SYSTEMS

SYLLABUS

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

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

TEXT BOOKS:

1. Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Seventh Edition, McGraw Hill, 2020.
2. Ramez Elmasri, Shamkant B. Navathe, "Fundamentals of Database Systems", Seventh Edition, Pearson Education, 2017

REFERENCES:

1. C.J.Date, A.Kannan, S.Swamynathan, "An Introduction to Database Systems", Eighth Edition, Pearson Education, 2006.

PREPARED BY

Mrs. C.Ramya, AP/IT

Ms.M.Subathra, AP/IT

II Batch Co-Ordinator
Mrs. G.Vasanthi, AP/IT

VERIFIED BY
Dr.S.Kalaivany,
Prof & HOD/IT

PRINCIPAL

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.

1. **What is database system? (April/May 2010)**
   The Database and the software together are called as database system.

2. **Define Database management system. (Nov/Dec 2008)**
   - Database management system (DBMS) is a 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.

3. List any eight applications of DBMS. (April\May-2019)

- Banking
- Airlines
- Universities
- Credit card transactions
- Tele communication
- Finance
- Sales
- Manufacturing
- Human resources

4. What are the advantages of using a DBMS?

The advantages of using a DBMS are

a) Controlling redundancy

b) Restricting unauthorized access

c) Providing multiple user interfaces

d) Enforcing integrity constraints.

e) Providing backup and recovery

5. Give the levels of data abstraction.(April/may 2014,Nov/Dec 2017)

a) Physical level

b) Logical level

c) View level

6. Define null values.

In some cases a particular entity may not have an applicable value for an attribute or if we do not know the value of an attribute for a particular entity. In these cases null value is used.

7. Define the terms i) Key attribute ii) Value set.

- **Key attribute**:

  An entity type usually has an attribute whose values are distinct from each individual entity in the collection. Such an attribute is called a key attribute.

- **Value set:**

  Each simple attribute of an entity type is associated with a value set that specifies the set of values that may be assigned to that attribute for each individual entity.

8. Define weak and strong entity sets.

- **Weak entity set:** entity set that do not have key attribute of their own are called weak entity sets.
- **Strong entity set:** Entity set that has a primary key is termed a strong entity set

9. What does the cardinality ratio specify?

Mapping cardinalities or cardinality ratios express the number of entities to which another entity can be associated. Mapping cardinalities must be one of the following:

- ✓ One to one
- ✓ One to many
- ✓ Many to one
- ✓ Many to many

Relationships in R, the participation of entity set E in relationship R is said to be partial.

10. Define the terms i) DDL ii) DML.

- **DDL:** Data base schema is specified by a set of definitions expressed by a special language called a data definition language.
- **DML:** A data manipulation language is a language that enables users to access or manipulate data as organized by the appropriate data model.

11. Write short notes on relational model.

The relational model uses a collection of tables to represent both data and the relationships among those data. The relational model is an example of a record based model.

12. Define tuple and attribute.

Attributes: column headers

Tuple: Row present in the table.

13. Define the term relation.

Relation is a subset of a Cartesian product of list domains.

14. Define tuple variable.

Tuple variable is a variable whose domain is the set of all tuples.

15. Define the term Domain.

For each attribute there is a set of permitted values called the domain of that attribute.

16. What is a candidate key?

Minimal super keys are called candidate keys.

17. What is a primary key?

Primary key is a candidate key chosen by the database designer as the principal means of identifying an entity in the entity set.

18. What is a super key?

A super key is a set of one or more attributes that collectively allows us to identify uniquely an entity in the entity set.

19. What are the levels of Data independence? (Nov/Dec-2010)

- **Physical Data Independence:** Changing the schema at the physical level without affecting the schema at logical level
- **Logical Data Independence:** Changing the schema at Logical level ,without affecting schema at view level

20. What are the functions of Authorization& Integrity Manager?

Tests for satisfaction of integrity constraints and checks the authority of users to access data.

21. What are the functions of Transaction Manager & File Manager?

- **Transaction Manager:**

  It ensures that database remains in a consistent state despite system failure and the concurrent transaction executions proceed without conflicting.

- **File Manager:**

  It manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

22. What are the functions ofBuffer Manager?

It is responsibility for fetching data from disk storage to main memory & deciding what data to cache in main memory.

The storage manager implements several data structure such as,

- **Data files -** Store the database itself.

- **Data dictionary -** Stores the metadata about the structure of the Database (i.e) Schema of the database.
- **Indices -** It provides fast access to data items. The Database index provides pointers to those data items that hold a particular value.

23. What are the functions of the Database administrator? (Nov/Dec2010)

(i) Schema definition
(ii) Storage structure and access method definition
(iii) Schema and physical organization modification
(iv) Granting of authorization for data access
(v) Routine Maintenance

24. What are the different types of data model? (April/May2011) (A\M-2019)

(i) Relational Model
(ii) Network Model
(iii) Hierarchical Model
(iv) Entity Relationship Model
(v) Object Based Model
(vi) Semi-structured Model

25. List the difference between File Processing and database System.(or) Explain the purpose of Database system. (Nov/Dec2008) (Apr/May 2015) (Nov/Dec 2016) (Nov/Dec 2019)

File Processing System	Database System
Data independence is not there	Data independence is there
It is difficult to access the data	It is easy to access the data
Data integrity& Security is less	Data integrity &Security is more
It produces concurrent anomalies.	Data can be accessed Concurrently

26. Define atomicity in transaction management. (MAY/JUNE 2013)

- Either all operations of the transaction are reflected properly in the database or none are. This is also known as the 'all or nothing property'.

28. Define data model. (Nov/Dec 2011)(Apr/May 2019)

- A data model is a collection of conceptual tools for describing data, data relationships, data semantics and consistency constraints.
- Data Model is a Structure below the Database.

29. Explain the basic structure of a relational database with example.

- A relational database is one which consists of rows & columns. A relational model uses a collection of tables to represent both data & relationship among those data.

Enroll no	Name	Dept
11010	Vinoth	IT
11021	Kanna	ECE
11023	Kumar	EEE
11008	Peter	CSE

30. With and example explain what are derived attribute is? (Nov/ Dec. 2011)

- An attribute whose value is derived from the value of other attributes.
- **Example:** Age can be derived from Date of Birth

31. What are the functions of DBA? (April/May 2010)

- Schema definition
- Storage structure and access method definition
- Granting of authorization for data access.
- Routine Maintenance.

32. Define the two levels of data independence. (Nov/ Dec 2010)

- **Physical Data Independence:** Modification in physical level should not affect the logical level.
- **Logical Data Independence:** Modification in logical level should affect the view level.

34. Explain the basic structure of a relational database with an example. (April/May 2010)

- The structure of a relational database consists of the attributes (column name) and a set of permitted values for those attributes called as domain (column value).

35. What are the parts of SQL language?

The SQL language has several parts:

- Data - definitition language
- Data manipulation language
- View definition
- Transaction control
- Embedded SQL
- Integrity
- Authorization

36. What are the categories of SQL command? (Apr/May 2012)

SQL commands are divided in to the following categories:

- Data - definition language
- Data manipulation language
- Data Query language
- Data control language
- Data administration statements
- Transaction control statements

37. What are the three classes of SQL expression?

SQL expression consists of three clauses:

- Select
- From
- Where

38. What is meant by relational model?

- Relational model stores data in the form of table
- Each table corresponds to application entity and each row represents an instance of that entity.

39. What is join? Why should we go for join? What are the types?

- Join is an operation by which the result is obtained by combining more than one table and by using a single SQL commands
- **TYPES:**

1. **Equi join –** Returns rows from both the tables that satisfies the equal to condition.
2. **Non equi join -** Returns rows from both the tables that satisfies the non equal to condition such as <, <=, >, >=, <>.
3. **Self join –** Returns rows from the singe table by comparing itself based on some condition.
4. **Inner join**-Returns rows from both the tables including all the attributes based on certain condition
5. **Outer join**

   Types:

   i. **Left outer join –** Returns all the rows from the left table and the rows from the right table thatmatches with the left table.

   ii. **Right outer join-** Returns all the rows from the right table and the rows from the left table that matches with the right table.

iii. **Full outer join –** Returns all the rows from both right and left table.

40. Define relational algebra.

- The relational algebra is a procedural query language.
- It consists of a set of operations that take one or two relation as input and produce a new relation as output.

41. What is a SELECT operation?

- The select operation selects tuples that satisfy a given predicate. We use the lowercase letter σ to denote selection.

42. What is the use of rename operation?

- Rename operation is used to rename both relations and a attributes. It uses the as clause, taking the form: Old-name as new-name

43. List the string operations supported by SQL.

- Pattern matching Operation
- Concatenation
- Extracting character strings
- Converting between uppercase and lower case letters.

44. List the set operations of SQL.

- Union
- Union all
- Intersect
- Intersect all
- The except

45. What is the use of Union, intersection and except operation?

1. Union:
- Includes all tuples that are either in relation r1 or in r2 and in both r1 and r2.
- Avoids duplicates.

2. Union all:
- Includes all tuples that are either in relation r1 or in r2 and in both r1 and r2.
- Allows duplicates.

3. Intersect:
- Includes all tuples that are in both relation r1 and r2.
- Avoids duplicates.

4. Intersect all:
- Includes all tuples that are in both relation r1and r2.
- Allows duplicates.

5. Except:
- Includes tuples that are in relation r1 and not in r2.

46. What are aggregate functions? And list the aggregate functions supported by SQL.

- Aggregate functions are functions that take a collection of values as input and return a single value.
- Aggregate functions supported by SQL are
    - Average: avg
    - Minimum: min
    - Maximum: max
    - Total: sum
    - Count: count

47. What is the use of group by clause?

- Group by clause is used to apply aggregate functions to a set of tuples.
- The attributes given in the group by clause are used to form groups.
- Tuples with the same value on all attributes in the group by clause are placed in one group.
- **Example :** Select count(*) from student group by department

48. What is the use of sub queries?

- A sub query is a select-from-where expression that is nested with in another query.
- A common use of sub queries is to perform tests for set membership, make set comparisions, and determine set cardinality.

49. What is view in SQL? How is it defined?

- Any relation that is not part of the logical model, but is made visible to a user as a virtual relation is called a view. We define view in SQL by using the create view command.
- **Syntax** to create view command is

  Create view <view name><attributes> as <query expression>

50. List the table modification commands in SQL.

- Deletion
- Insertion
- Updates
- Update of a view

51. List out the statements associated with a database transaction.

- Commit work
- Rollback work

52. What is transaction?

Transaction is a unit of program executions that accesses and possibly updates various data items in the database.

53. List the SQL domain Types.

SQL supports the following domain types.

1. Char(n), 2.varchar(n), 3.int, 4.numeric (p, d), 5.float (n) 6. date.

54. What is the use of integrity constraints?

- Integrity constraints ensure that changes made to the database by authorized users do not result in a loss of data consistency.
- Thus integrity constraints guard against accidental damage to the database.

55. What is trigger?

- Triggers are statements that are executed automatically by the system as the side effect of a modification to the database.
- **Syntax:**

  Create or replace trigger < trigger name >< before / after >

  <insert / update / delete > on < table name >for < each row > where< condition >;

56. What are referential integrity constraints? Explain with an example.

- A value that appears in one relation for a given set of attributes also appears for a certain set of attributes in another relation.
- The definition of a table has a declaration

  "foreign key(attribute) references <table name>"
- **Example:**

1. Create table account(accno number primary key, custnamevarchar(50));
2. Create table branch(accno number foreign key(accno) references account, bnamevarchar(50));

   Where, accno is a primary key in table account and a foreign key in table branch.

57. Write the purpose of triggers. (May/June 2013)

- Triggers are useful mechanisms for alerting humans or for starting certain tasks automaticallywhen certain conditions are met.

58. List the requirements needed to design a trigger.

The requirements are

- ✓ Specifying when a trigger is to be executed.
- ✓ Specify the actions to be taken when the trigger executes.

59. Give the forms of triggers.

- The triggering event can be insert or delete.
- For updated the trigger can specify columns.
- The referencing old row as clause
- The referencing new row as clause
- The triggers can be initiated before the event or after the event.

60. Name the various privileges in SQL.

- Delete
- Select
- Insert
- update

61. Mention the various user privileges.

- All privileges directly granted to the user or role.
- All privileges granted to roles that have been granted to the user or role

62. Give the limitations of SQL authorization.

- The code for checking authorization becomes intermixed with the rest of the application code.
- Implementing authorization through application code rather than specifying it declaratively in SQL makes it hard to ensure the absence of loop holes

63. What is a PROJECT operation?

- The project operation is a unary operation that returns its argument relation with certain attributes left out. Projection is denoted by pie (π).

64. What are the parts of trigger?

- The parts of trigger are
  - ✓ event
  - ✓ Condition
  - ✓ action

- **EVENT**

  It is an SQL statement which causes the trigger to occur.

- **CONDITION**

  It is a logical expression to be satisfied for the trigger to occur

- **ACTION**

  The statement which is executed after a trigger has occurred.

64. What is an embedded SQL?

Embedded SQL is a method of combining the computing power of a programming language .

65. What is domain integrity? Give example. (MAY 2008)

- The domain integrity states that every element from a relation should respect the type and restrictions of its corresponding attribute.
- A type can have a variable length which needs to be respected.
- Restrictions could be the range of values that the element can have, the default value if none is provided, and if the element can be NULL.
- **Ex:** stud_no number

67. List out the various relational algebra operators. (DEC 2010)

Relational algebra operators

1. Fundamental operators

- Select
- Project
- Rename
- Union
- Set difference
- Cartesian product

2. Additional operators

- Intersect
- Division
- Join

68. Give the usage of the rename operation with an example. (MAY 2010)

Rename is used to change the name of the attribute or table name.

1. IN SQL

a. Rename the column name

Syntax:

ALTER TABLE table name **RENAME COLUMN** OldName **TO** NewName

Example:

Alter table account **rename column** accno **to** ano;

a. Rename the table name

Syntax:

ALTER TABLE tablename**RENAME** OldName**TO**NewName

Example:

Alter table account **rename** account **to** account_details**;**

69. What do you mean by weak entity set? (MAY 2010)

In a relational database, a weak entity set is an entity set that cannot be uniquely identified by its attributes alone; therefore, it must use a foreign key in conjunction with its attributes to create a primary key.

70. Consider the following relation:

EMP(ENO, NAME, DATE_OF_BIRTH, SEX, DATE_OF_JOINING, BASIC_PAY, DEPT). Develop an SQL query that will find and display the average BASIC_PAY in each DEPT. (DEC 2011)

Answer:

Select avg (basic_pay**) from** emp **group by** dept;

71. What is embedded SQL? What are its advantages? (MAY 2011)

- **Embedded SQL** is a method of combining the computing power of a programming language.
- A language in which SQL queries are embedded is referred to as a host language, and the SQL structures permitted in the host language comprise embedded SQL.
- EXEC SQL statement is used to identify embedded SQL request to the preprocessor.

Advantages of Embedded SQL:

- Embedded SQL provides several advantages over a call-level interface:
- Embedded SQL is easy to use because it is simply Transact-SQL with some added features that facilitate using it in an application.
- It is an ANSI/ISO-standard programming language.
- It requires less coding to achieve the same results as a call-level approach.
- Embedded SQL is essentially identical across different host languages. Programming conventions and syntax change very little. Therefore, to write applications in different languages, you need not learn new syntax.
- The precompiler can optimize execution time by generating stored procedures for the Embedded SQL statements.

72. Is it possible for several attributes to have the same domain? Illustrate your answer with suitable example? NOV/DEC 2015

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 Attribute to have same domain.

For Example:

Student domain has attribute Roll no,name,Address.

73. What are the disadvantages of File processing System? Apr/May 2016

The file processing system has the following major disadvantages:

- Data redundancy and inconsistency.
- Integrity Problems.
- Security Problems
- Difficulty in accessing data.
- Data isolation.

74. Give a Brief Description on DCLCommand.NOV-DEC-2014

Data Control Language:

It is used to control access to data in a database.

It includes:

Grant: used to allow specified users to perform specified tasks.

Revoke: used to cancel previously granted or denied permissions.

75. Why does SQL allow duplicate tuples in a table or in a query result? NOV/DEC-2015

Every individual scalar value in the database must be logically addressable by specifying the name of the containing table, the name of the containing column and the primary key value of the containing row.

PART-B

1. Explain the purpose of Database system. (APRIL/MAY 2010)

The database system arose in response to early methods of file –processing system.

File processing system:

- It is supported by conventional OS. It stores permanent records in various files if needs different application programs to extract records from and add records to appropriate file.
- The file processing system has number of disadvantage. To avoid these disadvantages we need database system.

1. Data Redundancy and Inconsistency

REDUNDANCY:

- Duplication of data in several files.
- Leads to higher storage and access cost.

INCOSISTENCY:

- Changes made in one data is not reflected in all the copies of the same data.

2. Difficult In Accessing Data:

- Does not allow needed data to be retrieved in convenient & efficient manner.

3. Data Isolation:

- Since data are scattered & files are in different format writing new application programs to retrieve appropriate data is difficult.

4. Integrity Problem:

- The data values stored in the database must satisfy certain type of consistency constraints
- .The constraints are enforced by adding appropriate code in programs.
- When new constraints are added it is difficult to change programs to enforce them.

5. Atomicity Problem:

- If any failure occurs the data is to be restored to the consistent state that existed prior to failure .
- It must be atomic happen entirely or not at all.

6. Concurrent Anamolies:

- Interaction of multiple user to update the data simultaneously is possible and may result in inconsistent data.

7. Security Problems:

- Not every user of database system is allowed to access data.
- Enforcing security constraint is difficult in file processing system.

2. Explain the different views of Data.(MAY 2016) (or) Explain the difference between physical level, conceptual level & view level of data abstraction.(MAY 2011) (Nov/Dec 2019)

- *Introduction*
- *Dataabstraction*
  - *Physicallevel*
  - *Logicallevel*
  - *Viewlevel*
- *Instances &schema*
  - *Physicalschema*
  - *Logicalschema*
  - *Subschema*

Introduction

The major purpose of a database system is to provide users with an **abstract view of data** (i.e.) the system **hide the details** of how the data is stored and maintained.

Data Abstraction:

- **Figure 1.1** describes Developers hide the complexity of database from users, to simplify users interaction with the system
- Three **levels of abstraction** are

- ✓ Physical level
- ✓ Logical level
- ✓ view level

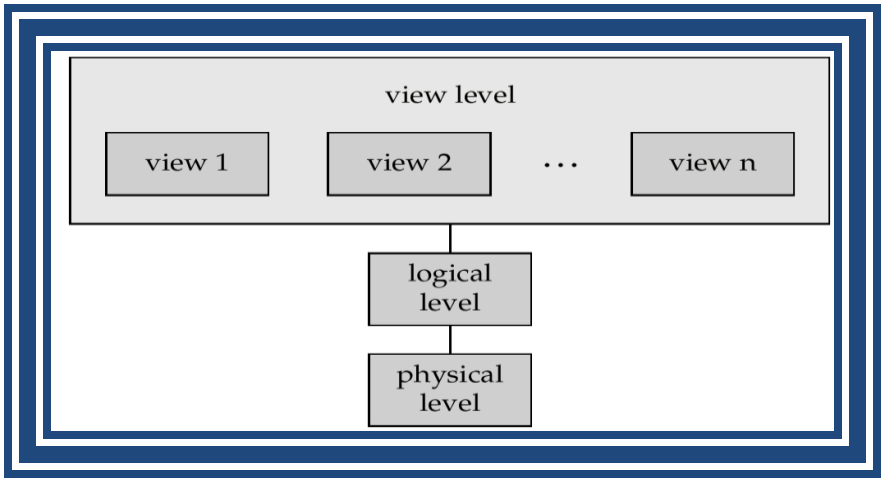


Figure 1.1 : views of Data

Physical Level:

- This is the **lowest levels** of abstraction.
- This level describes **how the data** are actually stored.
- This level describes complex low level data structure in details.

Logical Level:

- This **is Next higher level** of abstraction
- Describe **what data** are stored and what relationship exist among the data
- The logical level thus Describe the database in terms of small number of relatively simple structure.
- **Database administrators** decide what information to be kept in database.
- Database Administrator use logical level of abstraction.

View Level:

- This is **highest level** of abstraction.
- Describe **only part of the entire database.**
- It simplifies **the user interaction** with system.
- It provides many views for the same database.

INSTANCE & SCHEMA:

Instance: The **collection of information** stored in a database at a particular moment.

Schema: The **overall design of database** is called database **schema.**

- **Physical schema:** describes database design at physical level.
- **Logical schema:** describes database design at logical level
- **Subschema:** describe different views of the database.

Data Independence:-

- Changing the schema at one level, without affecting the schema at higher levels.

Physical Data Independence :-

- ✓ Changing the schema at **physical level**, without affecting the schema at **logical level.**

Logical Data Independence:-

- ✓ Changing the schema at **logical level,** without affecting the schema at **view level.**

3. What is a Data model? Explain the various types of data models.(Nov/DEC-2010)(APR -2019)(NOV2014)

- ***Data Model Definition***
- ***Types***
  1. ***Network data model***
  2. ***Hierarchical data model***
  3. ***Relational model***
  4. ***Entity relationship model***
  5. ***Object based data model***
  6. ***Semi structured data model***

Data Model Definition

- It is a **collection of conceptual tools** for describing **data, data relationship, data semantics and consistency constraints.**
- It provides a way to describe the design of a database at a physical, logical &new level.

TYPES:

- Network data model
- Hierarchical data model
- Relational model
- Entity relationship model
- Object based data model
- Semi structured data model

Network Data Model:

- In **Figure 1.2** network model the data are represented by **collection of records** and their relationship is represented by **links**.
- Network database consists of collection of records connected to one another through links.
- Each **records is a collection of fields or attributes**& each of which contain only one data value.
- **a link** is an associated between two records.

Ex:

Customer record is defined as

Type customer =**record**

Customer_name:string;

Customer_street:string;

Customer_city:string;

End

Account records is defined as

Type account=**record**

Acc_number : string;

Balance :integer;

End

In network model the two records are represented as

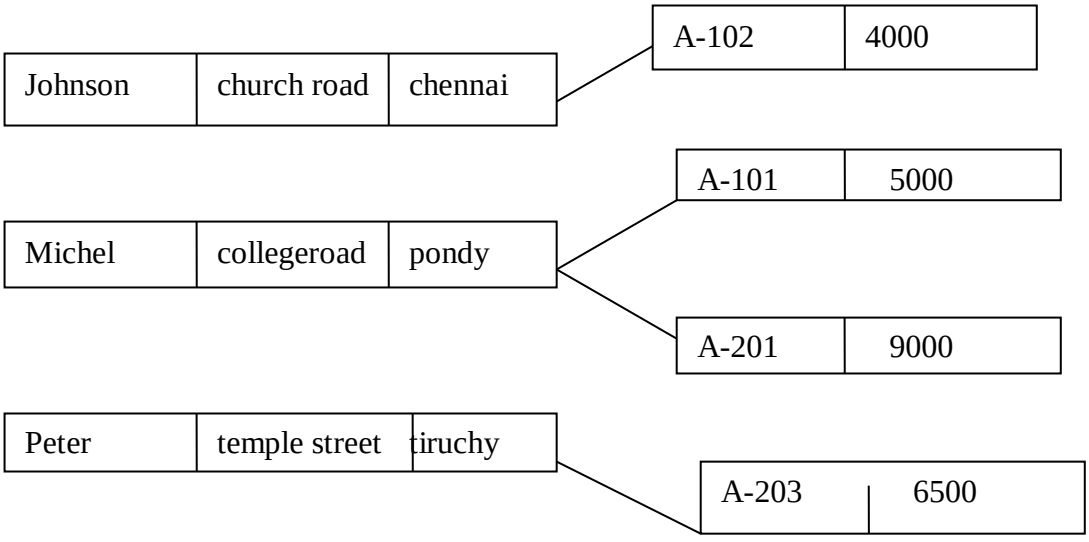


Figure 1.2 : Network Data Model

The sample database shows that Johnson has account A-102 & Michael has account A-101 &A201 & Peter Has Account a-203.

Hierarchical Model:

- In **Figure 1.3** describes Hierarchical model consists of **a collection of records** that are connected to each other through **links**.
- Records are organized as **a collection of trees.**

Ex:

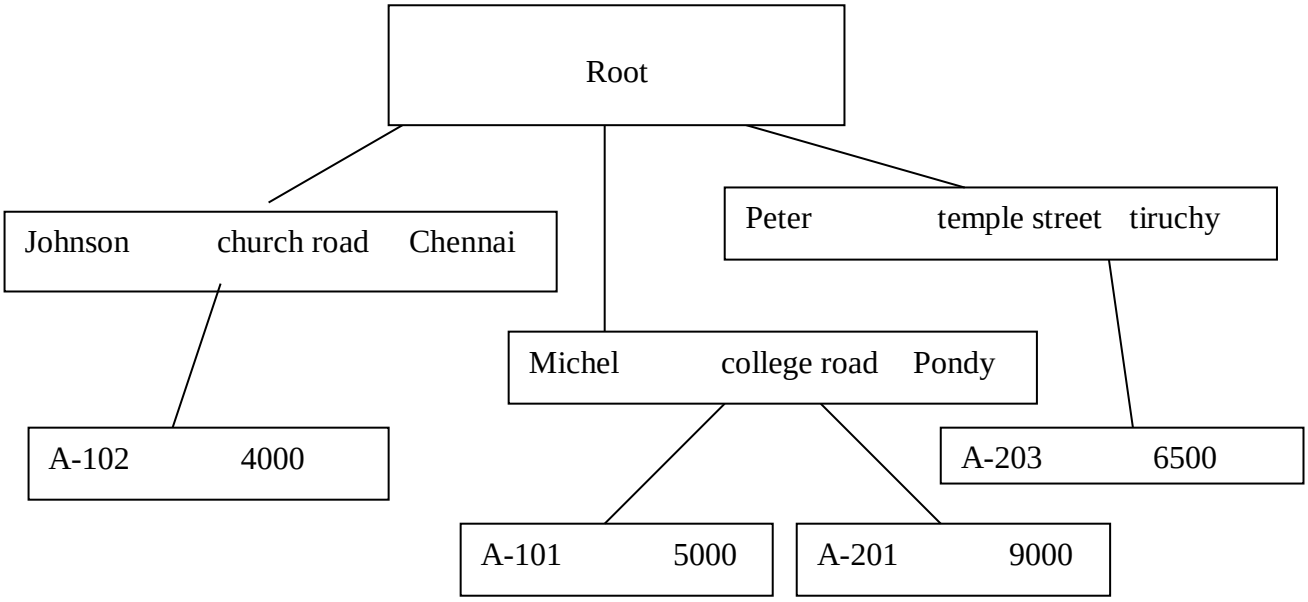


Figure 1.3: Hierarchical Model

Relational Model:

- It uses a **collection of tables** to represent both data and relationship among data.
- Each table has **multiple columns** and each column has **a unique name.**
- Tables are known as **relation,** each table contains **record** each record defines fixed no of fields the attribute of the records type.
- It is the **most widely** used as a model.

Eg:

Enrollno	NAME	DEPT
11010	Vinoth	IT
11021	Kanna	ECE
11023	Kumar	EEE
11008	Peter	CSE

Figure 1.4: Relational Model

Entity Relationship Model:

- It uses **a collection of basic objects** called entities and relationship among these object.
- **An entity is a thing or object** in the real world that is distinguished from other object.
- The entity relationship model is widely used in **database design**.
- **Eg:** In Figure 1.5, Customer and account is entity and depositor is relationship

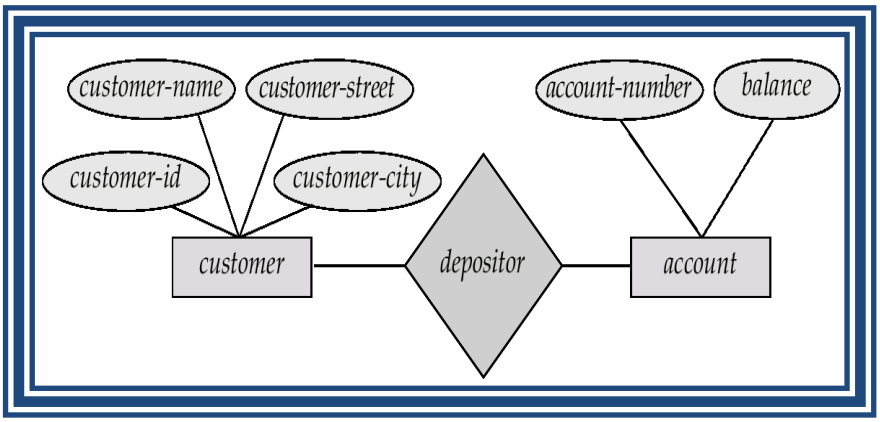


Figure 1.5: Entity Relationship Model:

Object Based Data Model:-

- The object oriented language like C++,java, C# are becoming the dominant in software development.
- The object based data model combines object oriented features with relational data model.
- It is of two types

1. **Object oriented data model**
   - ✓ It is the extension of ER model with notion of encapsulation methods & object identity.
2. **Object relational data model**
   - ✓ Combines features of object oriented data model and relational data model.

Semi structured Data Model:

- It permits the specification of data where individual data item of the same type may have **different set of attributes**.
- The **extensible markup language (XML)** widely used to represent semi structured data.

4. Explain the component modules of a DBMS and their interactions with the architecture. (DEC2008)

Explain the overall architecture of database system in detail.(MAY 2017)

Explain the database system structure with a neat diagram.(MAY 2010)

Explain the Database System Architecture in detail? (DEC 2009/MAY 16)

With the help of a neat block diagram explain the basic architecture of database management system. (DEC2015)

- *Introduction*
- *Functional Components of databasesystem*
  1. *Storage Manager*
     - *Authentication & IntegrityManager*
     - *File Manager*
     - *BufferManager*
     - *Transaction Manager*
  2. *QueryProcessor*
     - *DDLInterpreter*
     - *DMLCompiler*
     - *Query EvaluationEngine*
  3. *Architecture of DatabaseSystem*
     - *Two TierArchitecture*
     - *Three TierArchitecture*
  4. *Database System StructureDiagram*
  5. *Types of Database users andAdministrators*

Introduction

- Database system is partitioned into modules that deal with each of the responsibilities of the overall system.
- In Figure 1.6 shows that The functional components of the database system are,
  1. Storage Manager
  2. Query Processor

1. Storage Manager:

- It is a component of database system that provides the **interface** between the low-level data stored in the database and the **application programs and queries** submitted to the system.

- Storage manager **translates the DML statements into low-level file system commands.**
- It is responsibility for the interaction with file manager.
- Thus, the storage manager is responsible for **storing, retrieving and updating data is the database.**

Components of storage manager are,

- ✓ **Authorization & Integrity Manager**

  Tests for satisfaction of integrity constraints and checks the authority of users to access data.

- ✓ **Transaction Manager:**

  It ensures that database remains in a consistent state despite system failure and the concurrent transaction executions proceed without conflicting.

- ✓ **File Manager:**

  It manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

- ✓ **Buffer Manager:**

  It is responsibility for fetching data from disk storage to main memory & deciding what data to cache in main memory.

The storage manager implements several **data structure** such as,

- ✓ **Data files** - Store the database itself.
- ✓ **Data dictionary -** Stores the metadata about the structure of the Database (i.e) Schema of the database.
- ✓ **Indices** - It provides fast access to data items.

2. Query processor:

- It helps the database system to **simplify and facilitate access to data**.

Components of query processor are

- ✓ **DDL Interpreter:**

  Interprets DDL statements and records the definitions in **data dictionary.**

- ✓ **DML Compiler:**

  Translates DML statements into an **evaluation plan** consisting of low-level instruction that the query evaluation engine understands.

- It also performs query optimization (i.e) it picks the lowest cost evaluation plan from among the alternates

✓ **Query Evaluation Engine:**

- **Executes low-level instructions** generated by the DML compiler.

Architecture of DatabaseSystem

- Database systems can be centralized as **client –server.** Based on this database applications are portioned into
  - ✓ Two Tier architecture
  - ✓ Three tier architecture

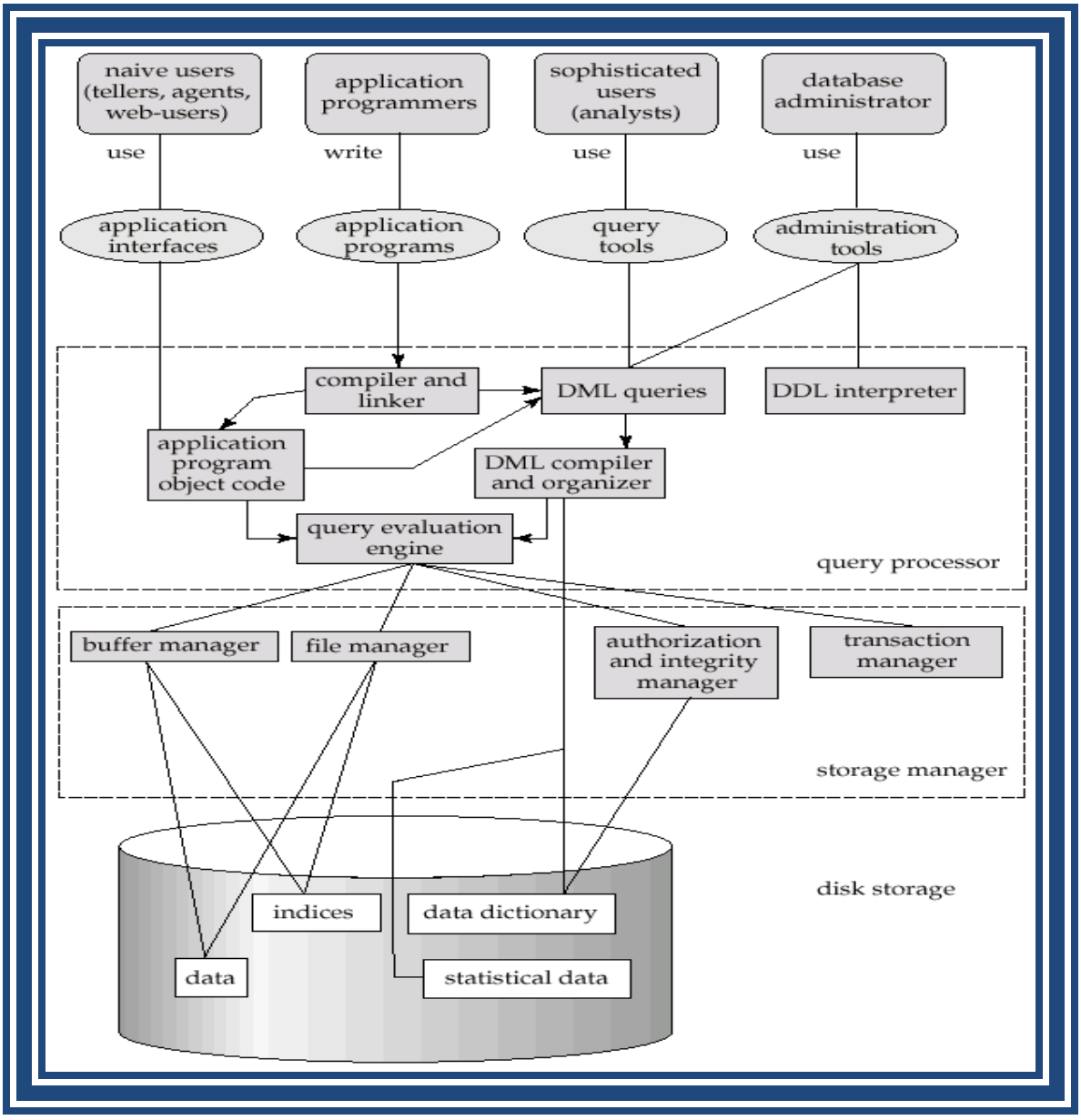


Figure 1.6: Database System Architecture

Two-tier architecture:

- **Application** resides at the **client machine** where it invokes **database system functionality** at the **server machine** through **query language** statements.
- E.g. client programs using ODBC/JDBC to communicate with a database.

Three-tier architecture:

- Client machine acts as merely a front end and does not contain any direct database calls.
- The client end communicates with an application server through forms interface and the application server in turn communicates with the database system to access data.
- E.g. web-based applications and applications built using **"middleware".**

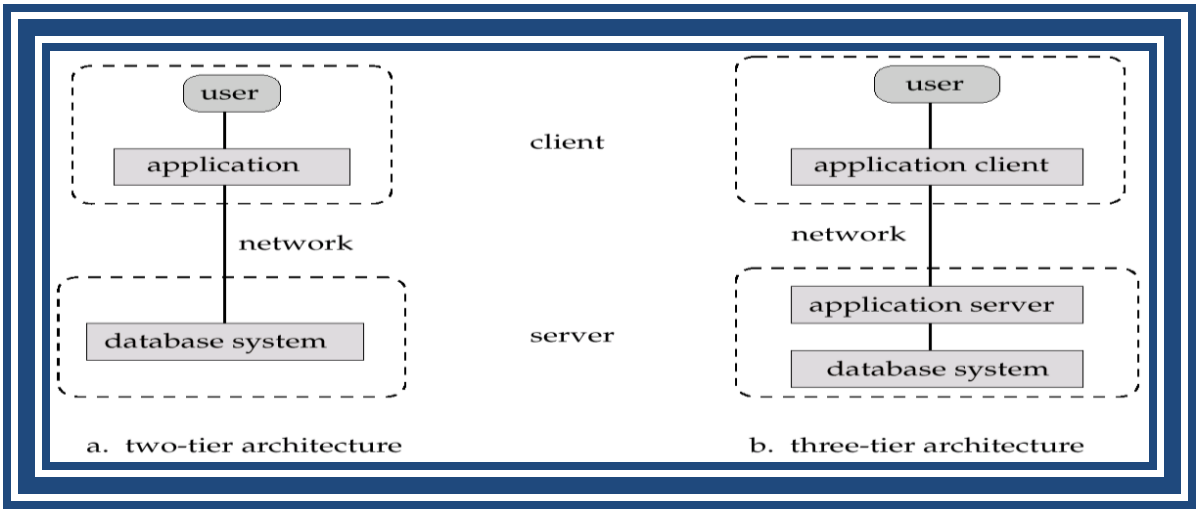


Figure 1.7 Two-tier architecture and Three-tier architecture:

Types of Database User & Administrator

- **People** who **work with a database** can be categorized as **database users or database administrator**.

Different types of users are

1. Naive Users

- **Unsophisticated users** who interact with the system by invoking one of the **application programs that have been written previously.**
- The user interface is from interface for naïve users.

2. Application Programmers

- They are **computer professionals** who **write application programs**.
- **Rapid application development** tools are tools that enable an application programmers to construct forms and reports with minimal programming effort

3. Sophisticated Users

- Interact with the system **without writing programs**.
- They form their request using **database query language.**

4. Specialized Users:

- These are **sophisticated users** who **write specialized database applications** that do not fit into traditional data processing framework
- **EX:** CAD design, audio, video.

5. Database Administrator

- A person who has **central control of both the data and the program** that access those data is called as database administrator.
- The functions of database Administrator

1. Schema Definition:

✓The Database Administrator creates original database schema by executing a set of data definition statements in the DDL.

2. Storage structure and Access method definition

3. Schema And Physical Organization Modification

✓The Database Administrator performs changes to the schema and physical organization to reflect the changing need of the organization.

4. Granting of authorization for data access

✓It grants different types of authorization, so that the database administrator can regulate which parts of the database various users can access.

5. Routine Maintenance

(i) Periodically backing up the database.

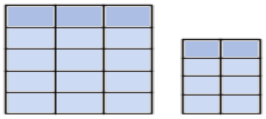
(ii)Ensuring that enough free disk space is available for normal operations.

(iii) Monitoring the jobs running and ensuring that performance is not degraded.

5. Write Short notes on Introduction to Relational Database.

- A relational database is one which consists of rows & columns.
- A relational model uses a collection of tables to represent both data & relationship among those data.
- Each table has multiple columns & each column has unique name.
- In record based model, the database is structured into fixed format records of a particular type.

- Each record type defines a fixed number of fields or attributes.
- It is the primary data model for commercial data processing application
- The data is stored in two-dimensional tables (rows and columns).
- The data is manipulated based on the relational theory of mathematics.



- Data and relationships are represented by a collection of tables.
- Each table has a number of columns with unique names, e.g. customer, account.

Example:

Name	Street	City	number
Lowery	Maple	Queens	900
Shiver	North	Bronx	556
Shiver	North	Bronx	647
Hodges	Sidehill	Brooklyn	801
Hodges	Sidehill	Brooklyn	647

Name	Balance
900	55
556	100000
647	105366
801	10533

6. Explain the Relational Model in detail?

- Relation model definition
- Relational model terminologies
- The relational model has 3 aspects
  1. Structural aspect
  2. Integrity aspect
  3. Manipulative aspect

Relation model definition

- It uses a **collection of tables** to represent both data and relationship among data.
- Each table has **multiple columns** and each column has a **unique name**.
- Tables are known as **relations**, each table contains **record**, and each record defines fixed no. of **fields** (ie) the **attribute** of the records type.
- It is the **most widely used** data model.

Relational Model terminologies

1. **Attribute**: Each column in a Table. Attributes are the properties which define a relation. e.g., Student_Rollno, NAME,etc.
2. **Tables** – In the Relational model the, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes.
3. **Tuple** – It is nothing but a single row of a table, which contains a single record.
4. **Relation Schema:** A relation schema represents the name of the relation with its attributes.
5. **Degree:** The total number of attributes which in the relation is called the degree of the relation.
6. **Cardinality:** Total number of rows present in the Table.
7. **Column:** The column represents the set of values for a specific attribute.
8. **Relation instance :** Relation instance is a finite set of tuples in the RDBMS system. Relation instances never have duplicate tuples.
9. **Relation key :** Every row has one, two or multiple attributes, which is called relation key.
10. **Attribute domain** : Every attribute has some pre-defined value and scope which is known as attribute domain

The relational model has 3 aspects

1. Structural aspect
2. Integrity aspect
3. Manipulative aspect

1. STRUCTURAL ASPECT:

The data in the database is perceived by the user as tables

2. INTEGRITY ASPECT:

The table must satisfy integrity constrains

3. MANIPULATIVE ASPECT:

It includes certain operators like select, Project, join and project for manipulation of tables

SELECT - Extract specified rows from the table

PROJECT - Extracts specified columns from the table

JOIN - Combines two tables into 1 on the basis of common values in a common Column.

Consider for ex:

There are 2 tables (i) Emp table (ii) Department table

Emp

Emp_no	Emp_name	Dept_no	Salary
E1	Latha	D1	40K
E2	Chitra	D2	42K
E3	Prabha	D3	30K
E4	Vani	D4	35K

Department

Dept_no	Dept_name	Budget
D1	Marketing	10M
D2	Development	20M
D3	Research	5M

SELECT OPERATION:

Select from Department where budget>8M

Dept_no	Dept_name	Budget
D1	Marketing	10M
D2	Development	20M

PROJECT OPERATION

Project Department over dept_no, budget

Dept_no	Budget
D1	10M
D2	20M
D3	5M

JOIN OPERATION:

Join department and emp over dept_no

Dept_no	Dept_name	Budget	Emp_no	Emp_name	Salary
D1	Marketing	10M	E1	Latha	40K
D2	Development	20M	E2	Chitra	42K
D3	Research	5M	E3	Prabha	30K

7. Explain a bout various types of keys.

- Keys definition
- Types of Keys
  1. Super key:
  2. Candidate key:
  3. Primary key:
  4. Foreign key:
  5. Referential integrity

Keys definition

- Key plays an important role in **relational database**
- it is used for identifying **unique rows** from table.
- It also establishes **relationship among tables**.
- It is the value of the attribute value of an entity such that they can uniquely identify the entity.

Types of Keys

1. Super key:

- It is a set of one or more attributes (columns), that taken collectively allows us identify uniquely a tuple in the relation.
- Super Key is a superset of Candidate key
- Example: Reg_no of students entity ie a superkey.

2. Candidate key:

- The candidate keys in a table are defined as the set of keys that is minimal and can uniquely identify any data row in the table.
- A candidate key can never be NULL or empty.
- Example student_id and phone both are candidate keys for table **Student**.

3. Primary key:

- It is a key that can **uniquely identify each record** in a table.
- Each **table** must have a **primary key** and a primarykey must be **unique**.
- **No part** of the primary key can be **null.**
- It is a candidate key which acts as a principal means of identifying tuples within a relation.
- For the table **Student** we can make the student_id column as the primary key.

4. Foreign key:

- It is a field in the table that is the **primary key of another table.**
- The foreign key is useful in **linking together two tables.**
- Data should be entered in the foreign key column with great care, as wrongly entered data can invalidate the relationship between the two tables.
- A relation r1 may include among its attributes, the primary key of another relation r2. This attribute is called foreign key from r1 referencing r2.

R1 – referencing relation

R2 – referenced relation

5. Referential integrity:

- It requires that the values appearing in specified attribute of any tuple in referencing relation also appear in specified attributes of at least one tuple in referenced relation.

8. Explain select, project and Cartesian product operations in relational algebra with suitable example. NOV 2016

Discuss in detail the operators SELECT, PROJECT, UNION with suitable examples. MAY 2013 & DEC 2011

What are the relational algebra operations supported in SQL? Write the SQL statement for each operation. MAY 2011

What are the relational algebra operations supported in SQL? Write the SQL statement for each operation. MAY 2011

Explain in detail Relational Algebra? DEC 2011

Discuss in detail the operators SELECT, PROJECT, UNION with suitable examples. MAY 2013

List the operations of relational algebra and the purpose of each with example. MAY 2017

- *Definition*
- *Fundamental relational algebra operations*
  1. *Unary Operations:*
     - *Select [σ]*
     - *Project [∏]*
     - *Rename [ρ]*
  2. *Binary operations:*
     - *Union[**U**]*
     - *Set Difference[-]*
     - *Cartesian Product[**X**]*
- *Additional relational algebra operations*
  - *Set Intersection operation (∩)*
  - *Natural Join Operation (⋈)*
    - *Theta Join (⋈Θ)*
    - *Outer Join*
      - *Left Outer Join ⟕*
      - *Right Outer Join ⟖*
      - *Full Outer Join ⟗*

Definition

- The relational algebra is a **collection of operators** that take **one or more relations as their input or operands** and **returns a single relation** as their output or result.
- Relational algebra is **a procedural language** consisting of a set of operators.
- Each operator takes one or more relations as its input and produces one relation as its output.
- The seven basic relational algebra operations are Selection, Projection, Joining, Union, Intersection, Difference and Division.
- It is important to note that these operations do not alter the database. The relation produced by an operation is available to the user but it is not stored in the database by the operation.
- Selection (also **called Restriction**) The SELECT operator selects all tuples from some relation, so that some attributes in each tuple satisfy some condition.

Fundamental Relational Algebra Operations:

1. Unary Operations:
   - Select [σ]
   - Project [$\prod$]
   - Rename [ρ]
2. Binary operations:
   - Union[**U**]
   - Set Difference[-]
   - Cartesian Product[**X**]

1.Unary Operations:

Select operation: [σ]

- The select operation **selects tuple** that satisfy a **given predicate.**
- It is denoted by $\sigma_{<\text{select condition}>}$**(R)**
- **EX:**
  - $\sigma_{b\_name=\text{"Perryridge"}}(loan)$
  - $\sigma_{amount>1200}(loan)$
  - $\sigma_{b\_name=\text{"Perryridge"}\wedge amount>loan}(loan)$

(Project operation[$\prod$]

- The project operation **selects columns or attributes** that **satisfy certain condition.**
- Duplicate rows are eliminated.
- It is denoted by $\prod_{<\text{attribute list}>(}$**R)**
- **Ex:** Consider loan relation

L_no	B_name	Amt
L-11	A	1000
L-12	B	1500
L-13	C	900
L-14	D	800
L-15	B	1100

(i) $\sigma_{b\_name=\text{"B"}}(loan)$

L_no	B_name	Amt
L-12	B	1500
L-15	B	1100

(ii) ∏ loan_no, amount (loan)

L_no	Amt
L-11	1000
L-12	1500
L-13	900
L-14	800
L-15	1100

Composition of Relational Operations:

$\prod_{loan\_no}(\sigma_{b\_name="B"}(loan))$

L_no
L-12
L-15

Rename operation [ρ]

- Given a relational algebra expression E, the expression $\rho_{x(E)}$, **returns the result of expression E** under the name X.
- **Example :** $(\sigma_{account\_bal<d.bal}(account \times \rho_{d(account)}))$

2. Binary operations

The Union Operation [U] (E1 U E2)

- The union operations **display all the records** that appear in either or both of the two relations.
- To perform union operation, two conditions hold
  - ✓ The relation r and s must have same number attributes.
  - ✓ The domains of i^{th} attribute of r and i^{th} attribute of S must be same for all i.

Ex:

Borrower

CNAME	CNO
A	L-103
A	L-104
C	L-105
E	L-106

Depositor

C_name	A_no
A	A_2
B	A_3
C	A_4
B	A_5
D	A_6

$\prod_{c\_name}$**(borrower) U** $\prod_{c\_name}$**(depositor)**

C_name
A
B
C
D
E

- The values selected are discrete and values are eliminated.

Set Difference operation (E1-E2):

- Allows us to find tuples that are in **one relation but are not in another**.
- The expressions r-s produces a relation containing those tuples in r, but not in s.
- **Example:** $\prod_{C\_name}$(depositor) - $\prod_{C\_name}$(borrower)

C_name
B
D

- Two conditions hold
  - ✓ The relation **r and s** must have same number attributes.
  - ✓ The domains of **i^{th} attribute of r and i^{th} attribute of S must be same** for all i.

The Cartesian Product operation (X):

- The Cartesian product, allows us to **combine information from any two relation.** It is represented as r1*r2 where r1 and r2 are relations

Borrower:

C_name	A_no
A	L_20
B	L_21

Loan:

A_no	Amt
L_20	1000
L_21	500
L_22	1500

Example: r=borrower**X**loan is

Output

C_name	Borrower. A_no	Loan. A_no	Amt
A	L_20	L_20	1000
A	L_20	L_21	500
A	L_20	L_22	1500
B	L_21	L_20	1000
B	L_21	L_21	500
B	L_21	L_22	1500

Additional Relational Algebra Operations:

- Set Intersection operation **(∩)**
- Natural Join Operation (⋈)
- Division Operation (÷)

Set Intersection operation (Π)

$$r \cap s = r - (r - s)$$

x: ∏ A_no(borrower) ∩ ∏ A_no(loan)

L_no
L_20
L_21

Natural Join Operation ⋈

- The natural join forms a Cartesian product of its two arguments, performs a selection forcing equality on those attributes that appear in both relation schemas and finally **removes duplicate attributes.**
- Natural join is a **binary operator**.
- Natural join between two or more relations will result set of all combination of tuples where they have equal common attribute.

Example: Borrower ⋈ loan

OUTPUT

C_name	Borrower. A_no	Loan. A_no	Amt
A	L_20	L_20	1000
B	L_21	L_21	500

Outer join:

- It is an extension of join operation to deal with **missing information**.

- It is used to **avoid loss of data.**
- The various types of joins are
  - (i) Left Outer Join ⟕
  - (ii) Right Outer Join ⟖
  - (iii) Full Outer join ⟗

Left Outer Join (⟕)

- In this it takes all the tuples in the **left relation that did not match with any tuple in the right relation**, pads the tuples with NULL for all other attributes from right relation.

EX: Employee

E_name	Street	City
Rupesh	Mission	Pondy
Bala	Busy	Chennai
Risha	IG	Pondy
Hemesh	Nehru	Chennai

Pt_Works

E_name	B_name	Salary
Rupesh	SBI	50000
Risha	UCO	45000
Hemesh	ICICI	47000
Kavinaya	UTI	44000

EX: Employee ⟕ Pt_Works

E_name	Street	City	B_name	salary
Rupesh	Mission	Pondy	SBI	50000
Risha	IG	Pondy	UCO	45000
Hemesh	Nehru	Chennai	ICICI	47000
Bala	Busy	Chennai	NULL	NULL

Right Outer Join (⟖)

- In this it pads **tuples from the right relation that did not match any from left relation with NULL** an adds them to the result of natural join.

Ex: Employee ⟖ Pt_works

E_name	Street	City	B_name	salary
Rupesh	Mission	Pondy	SBI	50000
Risha	IG	Pondy	UCO	45000

Hemesh	Nehru	Chennai	ICICI	47000
Kavinaya	NULL	NULL	UTI	44000

Full outer join (⟗)

In this it does both lefter outer join and left outer join operations

Ex: Employee ⟗ Pt_works

E_name	Street	City	B_name	salary
Rupesh	Mission	Pondy	SBI	50000
Risha	IG	Pondy	UCO	45000
Hemesh	Nehru	Chennai	ICICI	47000
Kavinaya	NULL	NULL	UTI	44000
Bala	Busy	Chennai	NULL	NULL

9. Explain the SQL fundamentals in detail.(NOV/DEC 2014, APR/MAY 2015, APR/MAY 2016) (Or) Describe the six clauses in the syntax of SQL query and show what types of constructs can be specified in each six clauses. Which of the six clauses are required and which are optional? NOV/DEC 2015

Explain about Data Definition Language? APR/MAY 2016

- ***Parts of SQL***
  1. ***Data Definition Language (DDL)***
     - *Definition*
     - *Commands*
       - *CreateTable*
       - *AlterTable*
       - *DropTable*
  2. ***Data Manipulation Language (DML)***
     - *Definition*
     - *Commands*
       - *Select*
       - *Insert*
       - *Delete*
       - *Update*
  3. ***Data Control Language(DCL)***
     - *Definition*
     - *Commands*
       - *Grant*
       - *Revoke*

4. Transaction Control Language(TCL)

- *Definition*
- *Commands*
  - *Commit*
  - *Rollback*

The SQL fundamentals has several parts

1. Data Definition Language(DDL)
2. Data Manipulation Language(DML)
3. Data Control Language(DCL)
4. Transaction Control Language(TCL)

1. DATA DEFINITION LANGUAGE(DDL)

- It is used to create a table, alter the structure of a table and also drop the table.
- DDL specifies thefollowing:
  - Schema for each relation
  - Domain of values Integrity Constrains
  - Set of indices to be maintained for each relation
  - Physical storage media
- **DDL Commands** are

- Create table
- Alter table
  - Add Attribute
  - Modify Attribute
  - Drop Attribute
- Drop table
- Truncate table
- Desc table
- Rename table

1. CREATE TABLE

- It is used to **create a table**.

Syntax

Create table <tablename> (attributes datatypes);

Example:

Create table account (accno varchar (10), Bname char(15),balance number, primary key(accno));

2. DROP TABLE

- This command is used to delete a table**.**

Syntax:

drop table <tablename>;

Example:

drop table account;

3. ALTER TABLE –

- It is used to add, modify or remove attributes from the table.

- **Add**
  - **Syntax:**

Alter table <tablename> add (attributes);

  - **Example:** *Alter table account add (dept varchar (10));*

- **Modify**
  - **Syntax:**

Alter table <tablename> modify (attributes);

  - **Example:** *Alter table account modify (deptvarchar (20));*

- **Drop**
  - **Syntax:**

Alter table<tablename> drop (attributes);

  - **Example:** *Alter table account drop (dept);*

TRUNCATE COMMAND

- This command is used to delete the records but retain the structure.

Syntax

Truncate table <tablename>;

Example

Truncate table *account;*

DESC COMMAND

- It is used to View the table structure .

Syntax

desc <tablename>;

Example

desc account;

RENAME COMMAND

- It is used to Rename a table

Syntax

Rename (old_table_name) to (new_table_name);

Example

Rename account to acc;

2. DATA MANIPULATION LANGUAGE:(DML)

- Data manipulation language comprises the SQL statements, which **modify stored data** but **not the schema or database objects.**
- It is used to retrieve and manipulate **data in a relational database**.
- **DML Commands** are

select
insert
delete
update

INSERT TABLE:

- An SQL INSERT statement **adds one or more records** to any single table in a relational database.

Syntax1:

insert into table name values (value1, [value2, ...])

Example: *insert into phone_book values ('john doe', '555-1212');*

UPDATE TABLE

- An **SQLUPDATE** statement **changes** the data of one or more records in a **table.**
- Either all the rows can be updated, or a subset may be chosen using a **condition.**

Syntax:

Update table_name set column_name = value [, column_name =value ...] [where condition]

Example:

- *update t set c1 = 1 where c2 ='a'*
- *update t set c1 = 9, c3 = 4 where c2 ='a'*

SELECT TABLE:

- A SELECT statement **retrieves zero or more rows** from one or more database tables or database views.
- The SELECT statement has many optional clauses:
  - ✓ FROM
  - ✓ WHERE specifies which rows to retrieve.
  - ✓ GROUP BY groups rows sharing a property so that an aggregate function can be applied to each group.
  - ✓ HAVING selects among the groups defined by the GROUP BY clause.
  - ✓ ORDER BY specifies an order in which to return the rows.

Syntax:

select *from <table name> where predicate;

Example:

Table name: book

ISBN	TITLE	PUB_YEAR	UNIT_PRICE	AUTHOR_NAME	PUBLISHER_NAME
1001	Oracle	2004	399	Arora	PHI
1002	DBMS	2004	400	Basu	Technical
2001	DOS	2003	250	Sinha	Nirali
2002	ADBMS	2004	450	Basu	Technicalo
2003	Unix	2000	300	Kapoor	SciTech

✓ ***Displaying all fields***

*select * from book;*

Output

ISBN	TITLE	PUB_YEAR	UNIT_PRICE	AUTHOR_NAME	PUBLISHER_NAME
1001	Oracle	2004	399	Arora	PHI
1002	DBMS	2004	400	Basu	Technical
2001	DOS	2003	250	Sinha	Nirali
2002	ADBMS	2004	450	Basu	Technical
2003	Unix	2000	300	Kapoor	SciTech

✓ ***Displaying selected Fields***

select publisher_name from book;

Output

PUBLISHER_NAME
PHI
Technical
Nirali
Technicalo
SciTech

✓ ***Displaying distinct values***

select distinct publisher_name from book;

Output

PUBLISHER_NAME
PHI
Technical
Nirali
SciTech

✓ ***Where clause*** -It is used to select specific rows satisfying given predicate.

select title from book where pub_year='2004';

Output

TITLE
Oracle
DBMS
ADBMS

- ✓ ***Between*** - Between Comparison operator to specify that a value be less than or equal to some value and greater than or equal to some other value.

Select title from book where unit_price between 300 and 400;

(Or)

Select title from book where unit_price >=300 and unit_price <=400;

Output

TITLE
Oracle
DBMS
Unix

Order by:

- The ORDER BY clause identifies which columns are used to sort the resulting data, and in which direction they should be sorted (options are ascending or descending).
- **Example:** select title,unit_price from book order by title;

Output

TITLE	UNIT_PRICE
ADBMS	450
DBMS	400
DOS	250
Oracle	399
Unix	300

Group By:

- The GROUP BY clause is used to **group the rows based on certain criteria.**
- GROUP BY is often used in conjunction with SQL aggregation functions or to eliminate duplicate rows from a result set.
- The WHERE clause is applied before the GROUP BY clause.

- **Example:**

  select publisher_name,sum(unit_price)" TOTAL BOOK AMOUNT" from book group by publisher_ name;

Output

PUBLISHER_NAME	TOTAL BOOK AMOUNT
Nirali	250
PHI	399
SciTech	300
Technical	850

Having:

- Having is equivalent to the where clause
- It is used to specify the search criteria or search condition when group by clause is specified.
- **Example:**

select publisher_name,sum(unit_price)" TOTAL BOOK AMOUNT" from book group by publisher_ name having publisher_ name <>'PHI';

Output

PUBLISHER_NAME	TOTAL BOOK AMOUNT
Nirali	250
SciTech	300
Technical	850

String operation:

- SQL Specifies string by enclosing them in single quotes.
- Pattern matching is the commonly used operation on strings with Operator like
  - **Percentage (%)-**matches any substring
  - **Underscore(_)-**matches any character
- Pattern are case sensitive uppercase do not match with the lowercase

Example:

Select author_name from book where author_name like 'Ba%';

Output

AUTHOR_NAME
Basu

Aggregate functions:

- It takes collection of values as input and provides a Single value as output.
- The aggregate functions include the following
  - ✓ Average – avg
  - ✓ Minimum – min
  - ✓ Maximum – max
  - ✓ Total –sum
  - ✓ Count – count

Average

- The AVG function is used to calculate the average value of the numeric type.
- AVG function returns the average of all non-Null values.

Syntax

AVG() (or) AVG([ALL|DISTINCT] expression)

Example

Select avg(unit_price)"Average Price" from book;

Output

AVERAGE PRICE
359.8

Minimum

- MIN function is used to find the minimum value of a certain column.
- This function determines the smallest value of all selected values of a column.

Syntax

MIN() (or) MIN([ALL|DISTINCT] expression)

Example

select max(unit_rice)"Minimum Price" from book;

Output

MINIMUM PRICE
250

Maximum

- MAX function is used to find the maximum value of a certain column.
- This function determines the largest value of all selected values of a column.

Syntax

MAX() (or) MAX([ALL|DISTINCT] expression)

Example

select max(unit_price)"Maximum Price" from book;

Output

Maximum Price
450

Total

- Sum function is used to calculate the sum of all selected columns.
- It works on numeric fields only.

Syntax

SUM() (or) SUM([ALL|DISTINCT] expression)

Example

select sum(unit_price)"Total" from book;

Output

Total
1799

Count

- **COUNT function** is used to Count the number of rows in a database table.
- It can work on both numeric and non-numeric data types.

Syntax

COUNT(*) (or) COUNT([ALL|DISTINCT] expression)

Example

select count(unit_price)"No of Books" from book;

Output

No of Books
5

NULL values:

- ✓ It is used to indicate absence of information about the value of anattribute.
- ✓ **Example:** Select lno from loan where amount is Null;

Set operation:

- ✓ Union
- ✓ Intersect
- ✓ Except

Consider two tables: i) Depositor (customer_name,Account_no)

CUSTOMER_NAME	ACCOUNT_NO
John	1001
Sita	1002
Vishal	1003
Ram	1004

ii)Borrower(customer_name,Loan_no)

CUSTOMER_NAME	LOAN_NO
John	2001
Tonny	2003
Rohit	2004
Vishal	2002

Union operation:

Union clause merges the output of two or more queries into a single set of rows and columns.

Example

(select customer_name from depositor) union (select customer_name from Borrower)

Output

CUSTOMER_NAME
John
Sita
Vishal
Ram
Rohit
Tonny

Intersect operation

The intersect clause outputs only rows produced by both the queries intersected i.e the intersect operation returns common records from output of both queries.

Example

(select customer_name from depositor) intersect (select customer_name from Borrower)

Output

CUSTOMER_NAME
John
Vishal

Except operation:

The except also called as **Minus** outputs rows that are in first table but not in second table.

Example

(select customer_name from depositor) except (select customer_name from Borrower)

Output

CUSTOMER_NAME
John
Sita

DELETE TABLE:

- ✓ The DELETE statement **removes** one or more records from a **table**.

Syntax:

```
delete  from table_name [where condition];
```

- ✓ Any rows that match the WHERE condition will be removed from the table.
- ✓ If the WHERE clause is omitted, all rows in the table are removed

Example:

- *delete from trees where height <80;*
- *delete from book;*

3. DATA CONTROL LANGUAGE (DCL)

- A data control language (DCL) is a programming language used to **control access to data stored in a database.**
- It is a component of **Structured Query Language (SQL).**
- DCL commands include:
  - **GRANT** used to allow specified users to perform specified tasks.
  - **REVOKE** used to cancel previously granted or denied permissions.
- The operations for which privileges may be granted to or revoked from a user or role may include **SELECT, INSERT, UPDATE, and DELETE.**

1. GRANT

GRANT authorizes one or more users to perform an operation or a set of operations on an object.

Syntax:

grant <pivilege list > on < object > to < user id >[with grant option];

Description :

- Privilege List – SELECT, INSERT, DELETE, UPDATE, EXECUTE etc.
- Objects – TYPE < Type name >, TABLE < Table name >
- User ID – User ID or Role Name. IT can be replaced by PUBLIC – meaning all user known to the system.
- WITH GRANT OPTION - Can grant those privileges on that object to some other users.

Example :

- ***Grant*** *insert, delete* ***on*** *employee* ***to*** *user1,user2*
- ***Grant*** *select* ***on*** *employee, department* ***to*** *a3* ***with grant option;***

2. REVOKE:

It used to revoke privileges from users or roles.

Syntax:

revoke <pivilege list > on < object > from < user id>

Description:

- Privilege List – SELECT, INSERT, DELETE, UPDATE, EXECUTE etc.

- Objects – TYPE < Type name >, TABLE < Table name >
- User ID – User ID or Role Name. IT can be replaced by PUBLIC – meaning all user known to the system.

Example:

- **REVOKE** INSERT, DELETE **ON** EMPLOYEE, DEPARTMENT **FROM** A2;

4. TRANSACTION CONTROL LANGUAGE (TCL)

- A Transaction Control Language (TCL) is used to **control transactional processing** in adatabase.
- A transaction is **logical unit of work** that comprises one or more SQL statements, usually a group of **Data Manipulation Language (DML)** statements.
- TCL commands include:

COMMIT

- The commit command saves all transaction to the database since the last Commit or rollback command.

Syntax:

commit

ROLLBACK

- The Rollback Command is the Transaction Control Command Used to **undo all changes** of a transaction that have not already been saved to the database.

Syntax:

rollback to savepoint-name;

SAVEPOINT

- To **divide the transaction** into smaller sections.
- Save points offer a mechanism to roll back portions of transactions.

Syntax:

savepoint savepoint-name;

Example

1. *create table tbl_1(idint);*
2. *insert into tbl_1(id)values(1);*
3. *insert into tbl_1(id)values(2);*
4. *commit;*
5. *update tbl_1 set id=200 whereid=1;*

6. *savepointid_1upd;*
7. *update tbl_1 set id=1000 whereid=2;*
8. *rollback toid_1upd;*
9. *select id fromtbl_1;*

11. What is a trigger? What is the need of it in DBMS? (APRIL-2011)

Triggers:

- Trigger is a statement that is executed automatically by the system as a side effect of a modification to the database.

Need For Triggers:

- It is a useful mechanism for alerting humans or for starting certain tasks automatically when certain conditions are met
- To design a trigger mechanism, we must
  - ✓ Specify the conditions under which the trigger is to be executed.
  - ✓ Specify the actions to be taken when the trigger executed.
- Trigger is stored in the database and it is automatically when a triggering event occurs.

Triggering Events:

- Triggering event can be insert, delete or update.
- Triggers on update can be restricted to specific attributes.
- Values of attributes before and after an update can be referenced.
  - Referencing old row as delete and updates
  - Referencing new row as inserts and updates
- Triggers can be activated before an event, which can serve as extra constraints.

Syntax:

Create or Replace trigger <trigger name>
Before / After triggering event on <table view>
For each row
When (Condition)
Declare statements
Begin
Executable statements
Exception
End;

Example:

SQL> create table product_history (pidnumber(5), pname varchar2(20), sup_name varchar2(20), price number(7,2));
Table Created.

SQL> create table product (pidnumber(5), pname varchar2(20), sup_name varchar2(20), price number(7,2));
Table Created.

SQL> insert into product values (12, 'mobile', 'nokia', 2000);
SQL> select * from product_history;

No row selected.

SQL> select * from product;

Pid	Pname	Sup_name	Price
12	Mobile	Nokia	2000

Note: Trigger Query

```
SQL> create or replace Trigger price_history
     Before update of price on product
     For each row
     Begin
     Insert into product_history values(:old.pid, :old.pname, :old.sup_name,
:old.price);
     End;

     Trigger created.
```

SQL> update product set price = 800 where pid=12;

1 row updated.

SQL> select * from product;

Pid	Pname	Sup_name	Price
12	Mobile	Nokia	800

SQL> select * from product_history;

Pid	Pname	Sup_name	Price
12	Mobile	Nokia	2000

13. Explain in detail the embedded and dynamic SQL.(or) Advanced SQL features.DEC 2010 / NOV 2016(NOV 2008) (APR 2019)

1. ***Embedded SQL***
- *Definition*
- *Phases*
- *Example*

2. ***Dynamic SQL***
- *Definition*
- *Phases*
- *Example*

EMBEDDED SQL:

- A language in which **SQL queries are embedded** is referred to as a **host** language, and the **SQL structures** permitted in the **host language comprise embedded SQL**.
- **EXEC SQL** statement is used to **identify embedded SQL** request to the preprocessor

EXEC SQL <embedded SQL statement > END_EXEC

- The high level language which supports embedding SQL with in it is also known as **host language**.
- **Phases:**
  - ✓ Open
  - ✓ Fetch
  - ✓ Close
- **Example query**
  - From within a host language, find the names and cities of customers with more than the variable amount dollars in some account
  - Specify the query in SQL and declare a cursor for it

EXEC SQL

declare *c* ***cursor for***

select*depositor.customer_name, customer_city*

from *depositor, customer, account*

where*depositor.customer_name = customer.customer_name*

and *depositor account_number = account.account_number*

and*account.balance> :amount*

END_EXEC

- The **open** statement causes the query to be evaluated

*EXEC SQL **open** c END_EXEC*

- The **fetch** statement causes the values of one tuple in the query result to be placed on host language variables.

*EXEC SQL **fetch** c into :cn, :cc END_EXEC*

  - Repeated calls to fetch get successive tuples in the query result
- The **close** statement causes the database system to delete the temporary
  - relation that holds the result of the query.

*EXEC SQL **close** c END_EXEC*

Features OfEmbedded 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 language is quite easy.

DYNAMIC SQL

- Dynamic SQL is a programming technique which Allows programs to construct and submit SQL queries at run time.
- Dynamic SQL statements are not embedded in the source program but stored as strings of characters that are manipulated during a program's runtime.
- 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.
- **Phases:**
  - ✓ Prepare
  - ✓ Execute

- **Example**

char sqlprog = "update account*
*set balance = balance * 1.05*
where account_number = ?"

*EXEC SQL **prepare** dynprog from :sqlprog;*
Char account [10] = "A-101";
*EXEC SQL **execute** dynprog using :account;*

- The dynamic SQL program contains a ?, which is a place holder for a value that is provided when the SQL program is executed.

S.NO.	STATIC (EMBEDDED) SQL	DYNAMIC (INTERACTIVE) SQL
1.	In static SQL how database will be accessed is predetermined in the Embedded SQL statement.	In dynamic SQL, how database will be accessed is determined at run time.
2.	It is more swift and efficient.	It is less swift and efficient.
3.	SQL statements are compiled at Compile time.	SQL statements are compiled at Run time.
4.	Parsing; validation, optimization, and generation of application plan are Done at compile time.	Parsing, validation, optimization, and generation of application plan are done at run time.
5.	It is generally used for situations Where data is distributed uniformly.	It is generally used for situations Where data is distributed non-uniformly.
6.	EXECUTE IMMEDIATE, EXECUTE and PREPARE statements are not used.	EXECUTE IMMEDIATE, EXECUTE and PREPARE statements are used.
7.	It is less flexible.	It is more flexible.

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.

1. What is meant by normalization of data? (Or) What is normalization? (MAY 2010, 2014)

- It is a process of analyzing the given relation schemas based on their Functional Dependencies (FDs) and primary key to achieve the properties
- Minimizing redundancy
- Minimizing insertion, deletion and updating anomalies.

2. Define canonical cover or Minimal Cover.

A canonical cover Fc for F is a set of dependencies such that F logically implies all dependencies in FC and Fc logically implies all dependencies in F. Fc must have the following properties

3. List the properties of canonical cover.

Fc must have the following properties.

- No functional dependency in Fc contains an extraneous attribute.
- Each left side of a functional dependency in Fc is unique.

4. Explain the desirable properties of decomposition. APR-2019

- Lossless-join or non loss decomposition
- Dependency preservation
- Repetition of information

5. What is first normal form?

- It is the process of eliminating duplicate forms from same data and creating the separate tables for each group of related data and also to identify each row with a unique column or set of columns.
- The domain of attribute must include only atomic (simple, indivisible) values.

6. What is meant by functional dependencies? (or) Write a note on functional dependencies.(MAY 2010)

- Functional Dependency is when one attribute determines another attribute in a DBMS system.
- Functional Dependency plays a vital role to find the difference between **good and bad database design.**
- A functional dependency is **denoted by an arrow →**
- The functional dependency of X on Y is represented by X →Y and X is a determinant set and Y is a dependent set.

7. What are the uses of functional dependencies?

- To test relations to see whether they are legal under a given set of functional dependencies.
- To specify constraints on the set of legal relations.
- Functional Dependency avoids data redundancy.
- It helps to maintain the quality of data in the database.
- It helps to defined meanings and constraints of databases.
- It helps to identify bad designs.
- It helps to find the facts regarding the database design.

8. Explain trivial dependency.

- The Trivial dependency is a set of attributes which are called a trivial if the set of attributes are included in that attribute.
- So, X -> Y is a trivial functional dependency if Y is a subset of X.

9. What are axioms?

- Axioms or rules of inference provide a simpler technique for reasoning about functional dependencies on a relational database.

10. What is meant by computing the closure set of functional dependency?

- The closure of F denoted by **F+.**
- It is the set of functional dependencies logically implied by a given set F.
- The closure set of functional dependency can be computed using basic three rules which are also called as **Armstrong Axioms**.
- These are as follows: **Reflexivity, Augmentation, and Transitivity**.

11. What is 2NF?

- A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on primary key (i.e) All attributes are fully dependent on primary key.
- A relation is in 2NF if it contains no repeating groups and no partial key functional dependencies.

12. What is a Prime attribute?

- An attribute, which is a part of the candidate key is known as a prime attribute.
- **Example:**

  {rollno} is a candidate key and hence rollno is a prime attribute and name, city, phoneno are non prime attributes.

13. What is a non Prime attribute?

- An attribute, which is not a part of candidate key is known as a **non Prime attribute**.**(or)**
- Attributes of the relation **which does not exist in any of the possible candidate keys of the relation**, such attributes are called non prime attributes.
- Non prime attributes also called as **Non Key attributes**.
- **Example:**

{rollno} is a candidate key and hence rollno is a prime attribute and name, city, phoneno are non prime attributes.

14. What is partial functional dependency?

- Partial dependency means that a non prime attribute is functionally dependent on part of a candidate key.
- A functional dependency X->Y is a partial dependency
  - ✓ if Y is functionally dependent on X and
  - ✓ Y can be determined by any proper subset of X.

15. What is 3NF? (DEC 2010)

A relation is in 3NF if ,

- It contains no repeating groups
- It is in the Second Normal form.(i.e. it does not have partial functional dependency.
- It doesn't have transitive dependency.

16. What is transitive dependency?

- A functional dependency is said to be transitive if it is indirectly formed by two functional dependencies.
- If **a → b** & **b → c**, then according to axiom of transitivity, **a → c**. This is a **transitive functional dependency.**

Example:

X -> Z is **a transitive dependency** if the following three functional dependencies hold true:

- X->Y
- Y does not ->X
- Y->Z

17. What is BCNF?

- BCNF (**Boyce Codd Normal Form**) is the advanced version of 3NF.
- A table is in BCNF if every functional dependency X->Y, X is the super key of the table.
- For BCNF, the table should be in 3NF, and for every FD. LHS is super key.

18. What is multivalued dependency?

- Multivalued dependency occurs when two attributes in a table are independent of each other but, both depend on a third attribute.
- A multivalued dependency consists of at least two attributes that are dependent on a third attribute that's why it always requires at least three attributes.
- For example: Consider a bike manufacture company, which produces two colors (Black and white) in each model every year.

19. What is trivial multivalued dependency?

- An multivalued dependency X→Y in r is called as trivial multivalued dependency if
  - y is a subset of X
  - XUY=R.

20. What is non trivial multivalued dependency?

- A multivalued functional dependency that satisfies the condition y is a subset of X**,** XUY=R.is called as non trivial functional dependency.

21. What is 4NF?

- A relation schema R is in 4NF with a set of functional dependencies F if for every non-trivial multivalued dependency X→Y in f+, X is a super key of R.
- A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
- For a dependency A → B, if for a single value of A, multiple values of B exists, then the relation will be a multi-valued dependency.

22. What is Trivial Join Dependency?

A join dependency specified on a relation schema R is a trivial join dependency if one of schema R in join dependency is equal to R

23. What is 5NF?

- A relation schema R is in 5NF , if and only of every nontrivial join dependency that is satisfied by R is implied by the candidate keys of R , if and only if each of A,B,…..Z is a super key of R.
- 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)**

24. What is a super key?

- It is a set of one or more attributes, that taken collectively allows us identify uniquely a tuple in the relation.
- Super Key is a superset of Candidate key
- Example: Reg_no of students entity ie a superkey.

25. What are the ways to choose the functional dependency?

- To test relations to see whether they are legal under a given set of functional dependencies.
- If a relation r is said to be legal under a g set of fune dependencies F, use say that r satisfies F.
- To satisfy constraints on set of legal relations
- If a relation on schema R satisfies the set of functional dependency F then it means that F holds on R.

26. What is a closure of functional dependency?

Given a relation schema R a functional dependency f on R is logically implied by a set of functional dependencies F on R, if every relation instance r(R) that satisfies F also satisfies f.

27. What are the Armstrong Axioms available?

The Armstrong axioms consist of the following rules

Reflexivity rule:

If α is set of attributes and β c α, then α->β holds**.**

Augmentation rule:

If α->β holds, and γ is a set of attributes, then γα -> γβ holds**.**

Transitivity rule:

If α->β, β->γ holds, then α->γ holds addition rules.

Union rules:

If α->β, α->γ, α->βγ holds.

Decomposition rule:

If α->βγ , holds then α->β , α->γ holds**.**

Pseudo transitivity rules:

If α->β , and γβ->δ , holds αγ-.δ holds

28. What is a closure of attribute set?

- It is the set of all attributes functionally determined by α under a set F of functional dependencies; the closure of α under F.
- The input is a set F of functional dependencies and set α of attributes.

30. What is an extraneous attribute?

An attribute of function dependency is said to be extraneous, if we remove it without changing the closure of set of functional dependency

31. What are the goals of relational database design?

To generate set of relational schemas that allows us to store information with unnecessary redundancy and allow us to retrieve information easily.

32. What are the pitfalls of relational database design?

- Repetition of information
- Inability to store certain data

33. What is Reflexivity rule?

If α is set of attributes and β c α, then α->β holds**.**

34. What is Augmentation rule?

If α->β holds, and γ is a set of attributes, then γα -> γβ holds**.**

35. What is Transitivity rule?

If α->β, β->γ holds, then α->γ holds addition rules.

36. What is Union rule?

If α->β, α->γ, α->βγ holds.

37. What is Decomposition rule?

If α->βγ , holds then α->β , α->γ holds**.**

38. What is a Pseudo transitivity rule?

If α->β , and γβ->δ , holds αγ-.δ holds

39. Define irreducible sets of dependencies. (NOV/DEC 2010)

- A functional depending sets S is irreduciable if the set has the following three properties:
  - Each right set of a functional dependency of S contains only one attribute.
  - Each left set of a functional dependency of S is irreducible. It means reducing one attribute from left set will change the content of S (S will lose some information).
  - Reducing any functional dependency will change the content of S.
- Sets of Functional Dependencies(FD) with these properties are also called ***canonical*** or ***minimal.***

41. What is meant by lossless-join decomposition or loss less join ?(APL/MAY 2011)

- Decomposition must be lossless. It means that the information should not get lost from the relation that is decomposed.
- It gives a guarantee that the join will result in the same relation as it was decomposed.

42. What is the need for Normalization? (MAY/JUNE 2013)

- Minimizing redundancy
- Minimizing insertion, deletion and updating anomalies.

43. Define the terms i) Entity set ii) Relationship set. (Apr-2019)

Entity set: The set of all entities of the same type is termed as an entity set.

Relationship set: The set of all relationships of the same type is termed as a Relationship set.

44. Define weak and strong entity sets.

- **Weak entity set:** entity set that do not have key attribute of their own are called weak entity sets.
- **Strong entity set:** Entity set that has a primary key is termed a strong entity set

45. What does the cardinality ratio specify?

- Mapping cardinalities or cardinality ratios express the number of entities to which another entity can be associated.
- Mapping cardinalities must be one of the following:
  - One to one
  - One to many
  - Many to one
  - Many to many

46. What is a weak entity? Give Example. Nov/Dec 2016

In a relational database, a weak entity is an entity that cannot be uniquely identified by its attributes alone; therefore, it must use a foreign key in conjunction with its attributes to create a primary key. The foreign key is typically a primary key of an entity it is related to.

47 What are the desirable properties of decomposition? Apr/May 2017

- Decomposition is the process of breaking down in parts or elements.
- It replaces a relation with a collection of smaller relations.
- It breaks the table into multiple tables in a database.

- If there is no proper decomposition of the relation, then it may lead to problems like loss of information.
- **The properties of Decomposition,**
  1. Lossless Decomposition
  2. Dependency Preservation
  3. Lack of Data Redundancy

48. Distinguish between key and super key. Apr/May 2017

1. **Super key:**
   - It is a set of one or more attributes, that taken collectively allows us identify uniquely a tuple in the relation.
   - Super Key is a superset of Candidate key.
   - Example: Reg_no of students entity ie a superkey.
2. **Primary key:**
   - It is a candidate key which acts as a principal means of identifying tuples within a relation.

49. Difference between strong entity SET and weak entity SET.

Strong Entity Set	Weak Entity Set
it has its own primary key.	It does not save sufficient attributes to form a primary Key on its own.
It is represented by a rectangle.	It is represented by a double rectangle.
It contains a primary key represented by an underline.	It contains a Partial Key or discriminator represented by a dashed underline.
The member of strong entity set is called as dominant entity set.	The member of weak entity set is called as subordinate entity set.
The Primary Key is one of its 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.
The relationship between two strong entity set is represent by a diamond symbol.	The relationship between one strong and a weak entity set is represented by a double diamond sign. It is known as identifying relationship.
The line connecting strong entity set with the relationship is single	The line connecting weak entity set with the identifying relationship is double.
Total participation in the relationship may or may not exist.	Total participation in the identifying relationship always exists.

50. What is the significance of "participation role name" in the description of relationship types?

- Each entity type that participates in a relationship type plays a particular role in the relationship.
- It necessary to use role names in the description of relationship types, in some cases the same entity type participates more than once in relationship type in different roles.
- In such cases the role names become essential for distinguishing the meaning of each participation. Such relationship types are called recursive relationships.

51.'Boyce-Codd normal form is found to be stricter than third normal form'. Justify the statement.

Boyce-Codd normal form (BCNF) was proposed as a simpler form of 3NF, but it was found to be stricter than 3NF. That is, every relation in BCNF is also in 3NF; however, a relation in 3NF is not necessarily in BCNF.

PART-B

1. Explain the Entity Relationship Model.(APRIL/MAY 2010) (nov/dec 2010) (May/June 2012)

- *Introduction*
- *Basic Concepts of ER Model*
  - *Entity*
  - *Entity Sets*
  - *Relationship Set*
  - *Attributes*
- *Constraints*
  - *Mapping cardinalities*
  - *Participation constraints*
  - *Keys*
- *Entity Relationship Diagram –Notations*
- *Extended E-R Features*
  - *Specialization*
  - *Generalization*
  - *Constraints on generalization*
  - *Attribute inheritance*

Introduction

- The E-R model is **a <u>high-level data model</u>** it distinguish between basic object called **<u>entities and relationship among those object</u>.**

Basic Concepts of ER Model

- Entity
- Entity sets
- Relationship sets
- Attributes

Entity

- **Entity:** It is a **<u>thing or object in the real world</u>** that is distinguishable from other objects.

- An **entity has a set of properties** and the values for some set of properties may **uniquely identify an entity**.
- **Ex:** Person is an entity and person_id– uniquely identifies that person.

Entity Sets:

- **Entity set:** An entity set is a **set of entity of same type** that share **same properties** or attributes.
- **Ex: Table 2.1**Student - represents the set of all students in the university.

Roll no	S_name
101	John
102	Peter
301	Saran
405	Michael

Table 2.1: Student

Types of Entity Set

1. Strong Entity Set

- An entity set that has a **primary key** is called strong entity set

2. Weak Entity Set

- An entity set **that does not have a primary key** is called weak entity set

3. Identifying or Owner Entity set

- **Weak entity set** must be **associated with another entity set** called identifying or owner entityset.

Relationship Set

- **Relationship**: **Association** among several entities.
- **Relationship set:** A relationship set is a of **relationships of same type**.

Attributes

- An attribute of an entity set is a function that **maps from the entity set into a domain**.
- For each attributes, there is a set of permitted value set of that attribute.

- **Types** of attributes:
  - ✓Simple and composite
  - ✓Single value and multivalued
  - ✓Derived
  - ✓Descriptive

Simple & composite attributes:

1. **Simple attributes:**

- Attributes **that cannot be divided into subparts** are called simple attributes
- Ex: S.no is a simple attributes.

2. **Composite attribute:**

- Attribute that can be **divided into subparts** is called as composite attributes.
- Ex: name -first _name, last_name

Single valued & multi valued attribute:

1. **Single valued attribute**:

- The attribute that have **single value** for a particular entity is called single valued attribute.
- Ex: student_id – refers to only one student.

2. **Multivalued attributes**:

- The attribute that has a **set of values** for a specific entity is called multivalued attributes.
- Ex: phone_number.

Derived attributes:

- The value of this type of attribute can be **derived from the values** of other related attributes or entities.

Example: DOB - base or stored attribute

↓

Age - derived attribute

Descriptive attributes:

- The **attribute** present in the **relationship** is called as **descriptive attribute.**

Example: 1 Figure 2.1: ER diagram with descriptive attribute

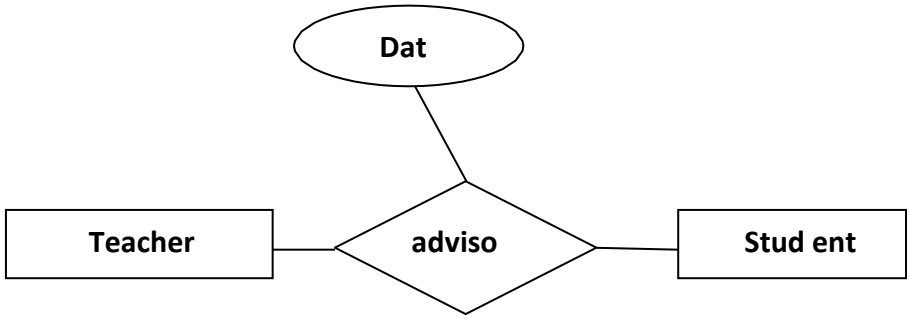


Figure 2.1: ER diagram with descriptive attribute

NULL value

- The attribute takes a NULL value, when an entity does not have a value for it.

Ex : ER diagram with composite, multivalued and derived attributes

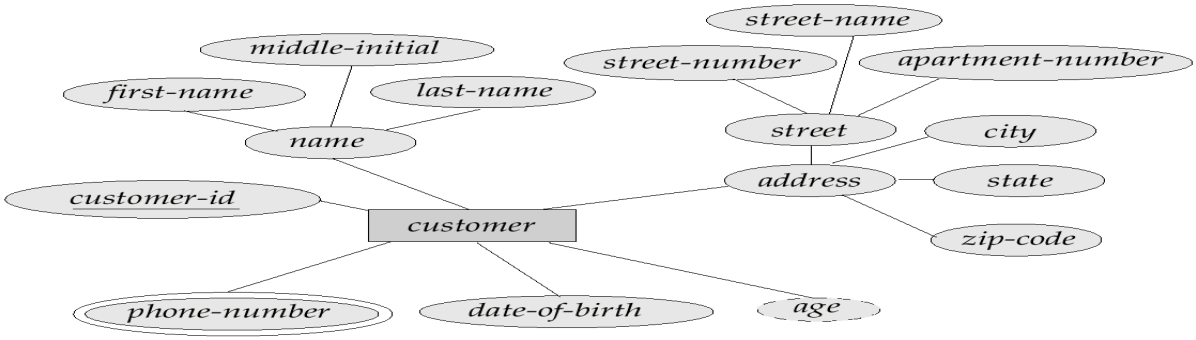


Figure 2.2: ER diagram notation

In the above ER diagram **Figure 2.2**

C_name& address - Composite attribute

Street - Component attribute

Phone-number - Multi-valued attribute

Age - Derived attribute

CONSTRAINTS

- Mapping cardinalities
- Participation constraints
- keys

Mapping cardinalities:

- A mapping cardinality is **a data constraint** that specifies **how many entities an entity can be related** to in a relationship set.
- Types of mapping cardinalities are
  - ✓ one to one
  - ✓ one to many
  - ✓ many to one
  - ✓ many to many
- Consider a binary relationship set R on entity sets A and B.
- There are four possible mapping cardinalities in this case:

1. One-To-One (MAY/JUNE 2013)

- An entity in A is related to **at most one entity** in B, and an entity in B is related to at most one entity in A.
- **Example: Figure 2.3** there is one project manager who manages only one project.

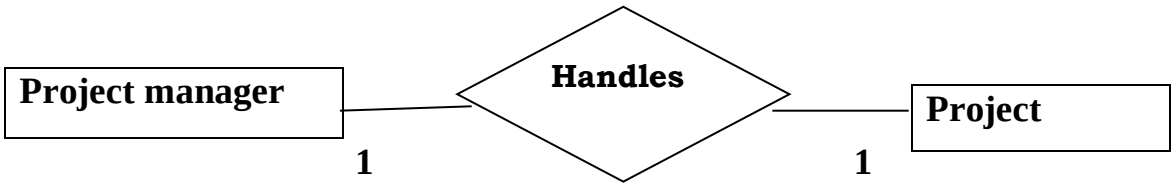


Figure 2.3: One-To-One relation

2.One-To-Many

- An entity in A is related to **any number of entities** in B, but an entity in B is related to at most one entity in A.

- **Example: Figure 2.4** one customer places order at a time.

Figure 2.4: One-To-Many Relation

3. Many-To-One

An entity in A is related to at most one entity in B, but an entity in B is related to any number of entities in A.

Example: Figure 2.5 Many Student take a Computer Science Course.

Figure 2.5: Many-To-One Relation

4. Many-To-Many

- An entity in A is related to any number of entities in B, but an entity in B is related to any number of entities in A.
- **Example: Figure 2.6** Many teachers can teach many students

Figure 2.6 Many-To-Many Relation

Participation Constraints

1. Total Participation:

- The participation of an entity set E in a relationship set R is said to be total, if **every entity in E participates in at least one relationship** in R
- **Example : Figure 2.6** for total participation events

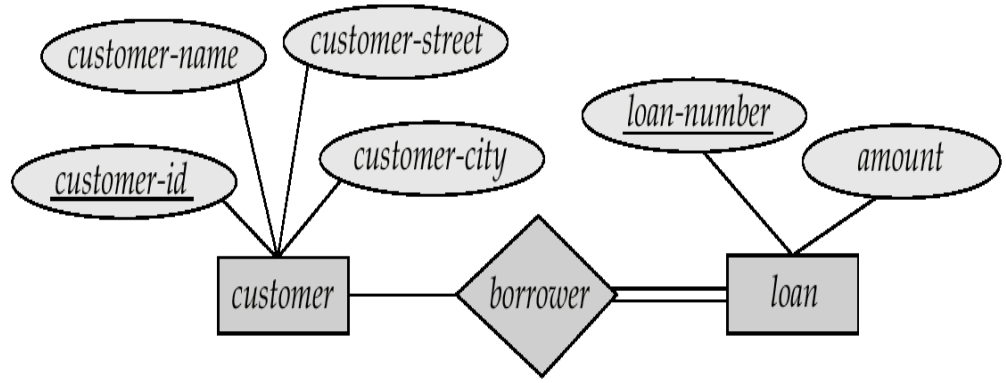


Figure 2.6 Total Participation

- Double Line indicate that from loan to borrower, each loan must have atleast one associated customer

2. Partial Participation:

- If only some entities in E participate in relationship in R ,then the participation of **entity E in relationship R is said to be partial** **Figure 2.6**

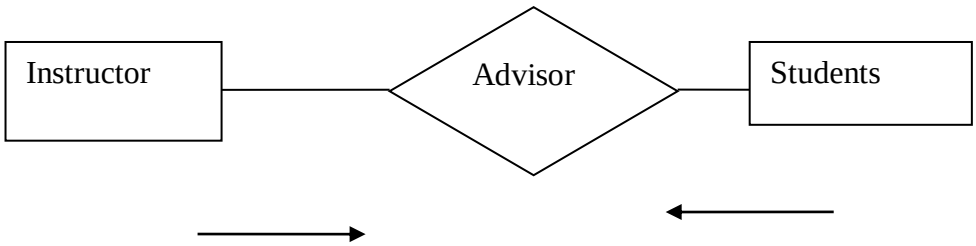


Figure 2.6 Partial Participation

ENTITY RELATIONSHIP DIAGRAM - SYMBOLS

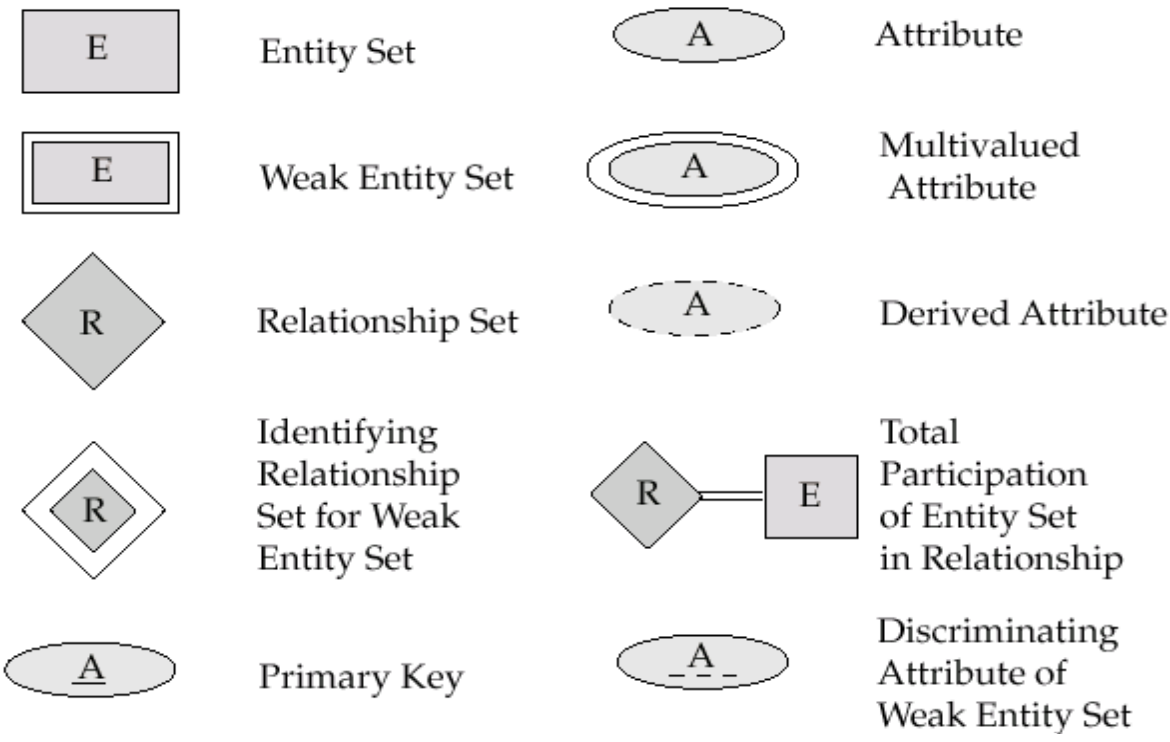


Figure2.7: ENTITY RELATIONSHIP DIAGRAM - SYMBOLS

EX 1: ER Diagram corresponding to Customer & Loan

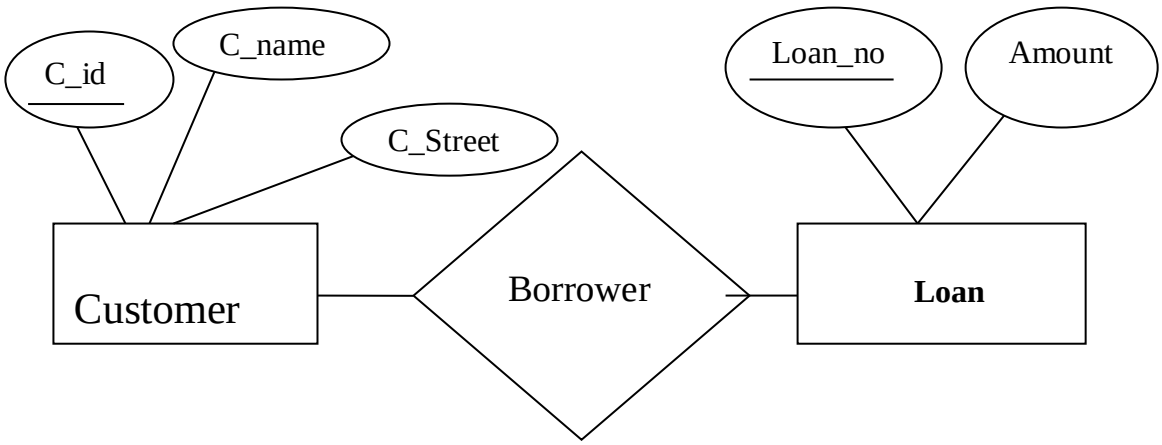


Figure2.7 ER Diagram corresponding to Customer & Loan

2. EXPAIN IN DETAIL ABOUT ENHANCED E-R MODLEL.

- Specialization
- Generalization
- Attribute inheritance
- Aggregation

Specialization

- The process of **designating sub groupings** within an entity set is called specialization.
- This is a **top down process.**
- The symbol used for specialization/generalization is

- Ex: Specializations of person allow us to distinguish among persons according to whether they are employee or customers.
- Specialization is defined by a triangle component **IS A** .it stands for customer is a person.
- IS A can also referred as a **super class-subclass** relationship is shown in **Figure2.8**

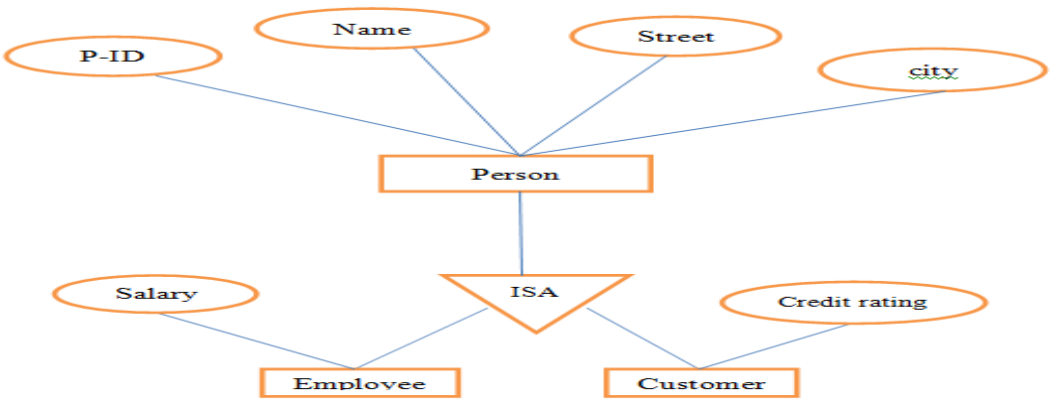


Figure2.8: Specialization & Generalization

Generalization:

- The process of making super class from subclasses is called **Generalization.**
- It is a containment relationship that exists between a **higher level entity set and one or more lower level entity set.** To create a generalization, the attribute must be given in common name.
- This is a **Bottom up process**.
- **Super class:** An Entity type that represents a general concept at high level is called Super class.
- **Sub Class:** An Entity type that represents a specific concept at lower levels is called sub class.

Constraints on generalization

Constraint1:

- Condition defined
- User defined

Condition defined

- In lower level entity sets, membership is evaluated on the basis of whether or not an entity **satisfies an explicit condition or predicate**.
- Entities that satisfy account type= ―saving‖, belong to low level entity saving account.
- Entities that satisfy account type =‖checking‖ belongs to lowlevel entity set checking account.
- Since the lower level entities are evaluated on the basics of same attribute it is called as attribute defined.

User defined:

- Not constrained by a membership condition.
- Database user assign entities to a given entity set.

Constraint 2:

- Disjoint
- Overlapping

Disjoint:

- A disjointness constraint requires that an entity belong to no more one low level entity set.
- **Ex:** Account entities satisfy only one condition either a saving account or checking account, but cannot be both.

Overlapping:

- The same entity may belong to more than one lower level entity set within a single generalization.
- **Ex:** Generalization applied to customer & employee leads to higher level entity set person.

Constraint 3:

Total generalization or specialization:

- Each higher level entity set must belong to an lower level entity set.

Partial generalization or specialization:

- Some higher level entity set may not belong to any lower level entity set.

Attribute Inheritance:

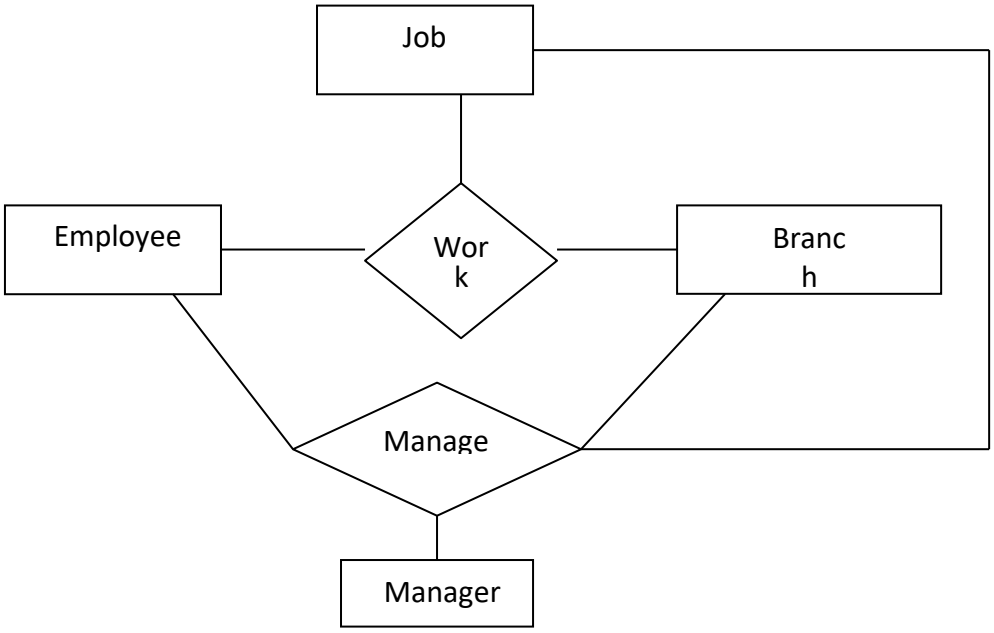
- The attributes of higher level entity are said to be inherited by lower level entity set.
- **Ex:** Customer & employee inherit the attributer of a person.

Aggregation

- Aggregation is a process when relation between two entities is treated as a single entity.
- The limitation of ER model is that it cannot express relationships among relationship. One alternative for representing the relationship is to create a quaternary relationship
- Aggregation is an abstraction through which relationship are treated as higher level entities.

Example: Figure2.9 ER diagram with redundant relationship

Figure2.9: ER diagram with redundant relationship



EXAMPLE: Figure2.10 ER diagram with Aggregation

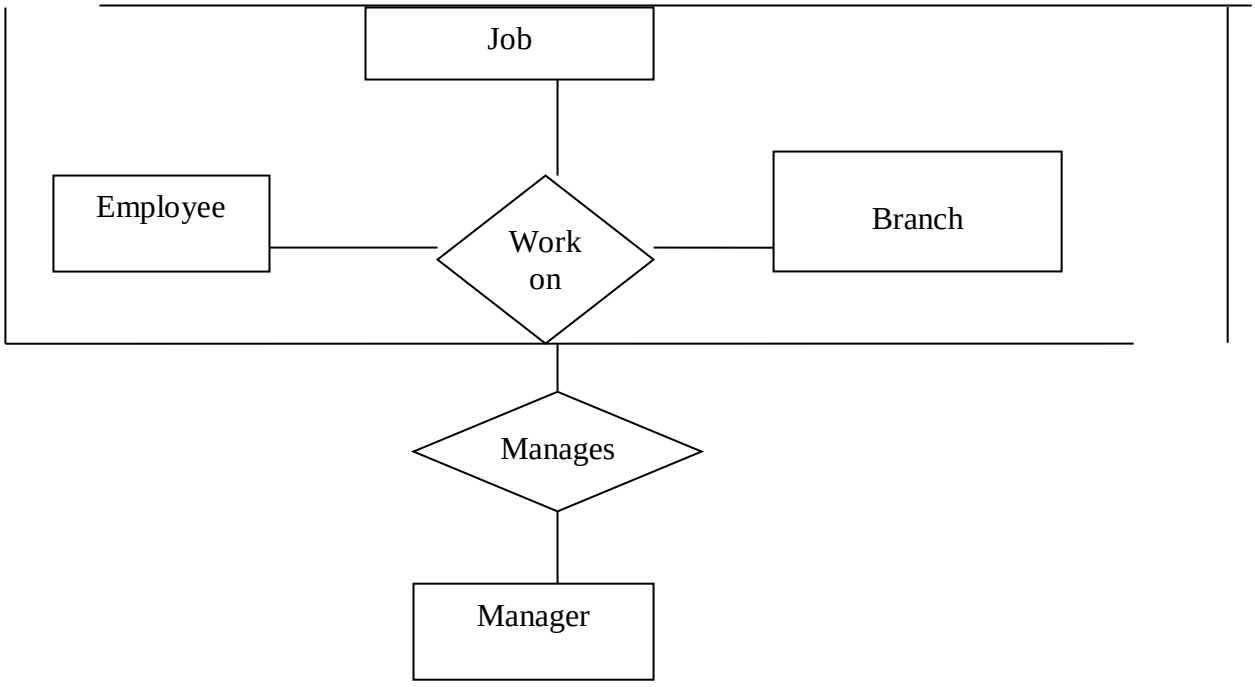


Figure2.10: ER diagram with Aggregation

3. Construct an ER diagram for hospital with a set of patients and a set of Medical doctors. Associated with each patient a log of the various test and examinations conducted.

- Patients are treated in a single ward by the doctors assigned to them. Usually each patient will be assigned a single doctor, but in rare cases they will have two.
- Healthcare assistants also attend to the patients; a number of these are associated with each ward.
- Initially the system will be concerned solely with drug treatment. Each patient is required to take a variety of drugs a certain number of times per day and for varying lengths of time.
- The system must record details concerning patient treatment and staff payment. Some staff is paid part time and doctors and care assistants work varying amounts of overtime at varying rates (subject to grade).
- The system will also need to track what treatments are required for which patients and when and it should be capable of calculating the cost of treatment per week for each patient (though it is currently unclear to what use this information will be put).

How do we start an ERD?

- **Define Entities:** these are usually nouns used in descriptions of the system, in the discussion of business rules, or in documentation; identified in the narrative (see highlighted items above).
- **Define Relationships:** these are usually verbs used in descriptions of the system or in discussion of the business rules (entity _____ entity); identified in the narrative (see highlighted items above).
- **Add attributes to the relations:** these are determined by the queries, and may also suggest new entities, e.g. grade; or they may suggest the need for keys or identifiers.

What questions can we ask?

- Which doctors work in which wards?
- How much will be spent in a ward in a given week?
- How much will a patient cost to treat?
- How much does a doctor cost per week?
- Which assistants can a patient expect to see?
- Which drugs are being used?
- Add cardinality to the relations
- This flexibility allows us to consider a variety of questions such as:
- Which beds are free?
- Which assistants work for Dr. X?
- What is the least expensive prescription?
- How many doctors are there in the hospital?
- Which patients are family related?

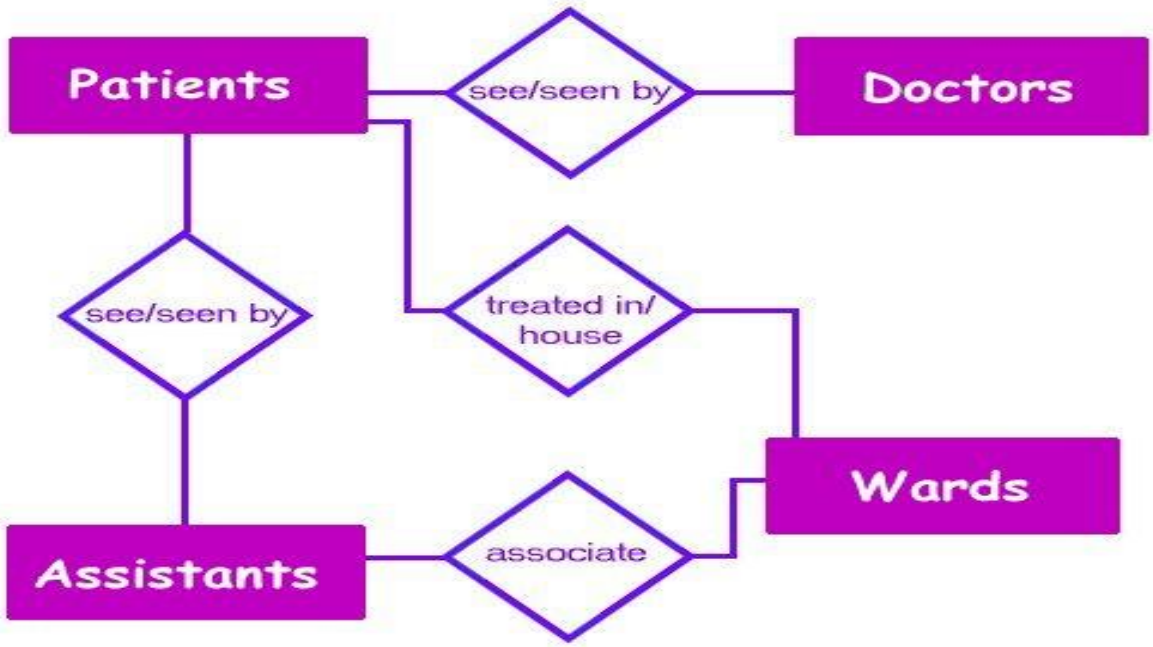


Figure2.11:Examples of ER diagrams

a. Draw an ER diagram describing theatre.

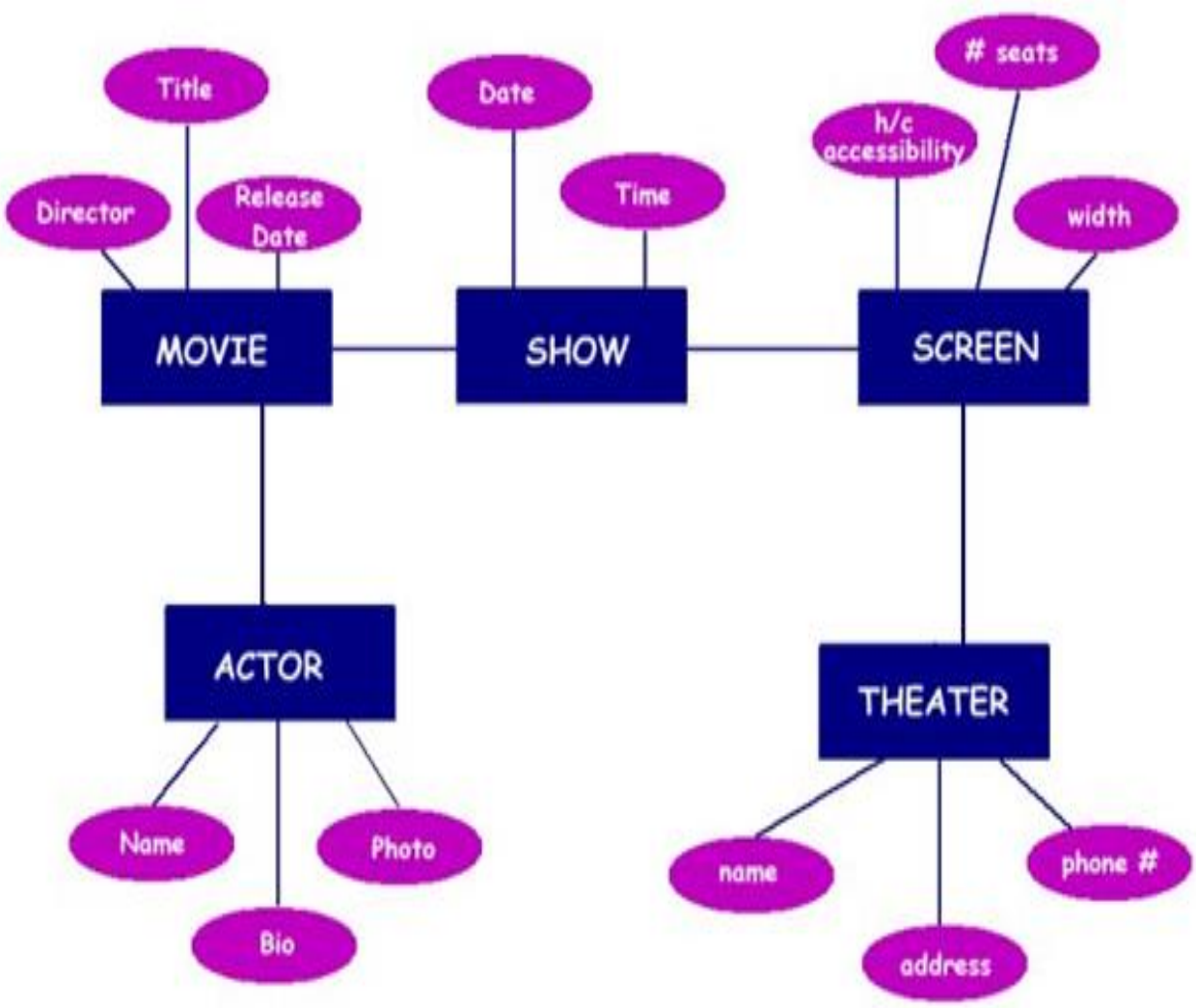


Figure2.12: ER diagram describing theatre

b.Construct an ER diagram for a mechanic shop that repairs a car.

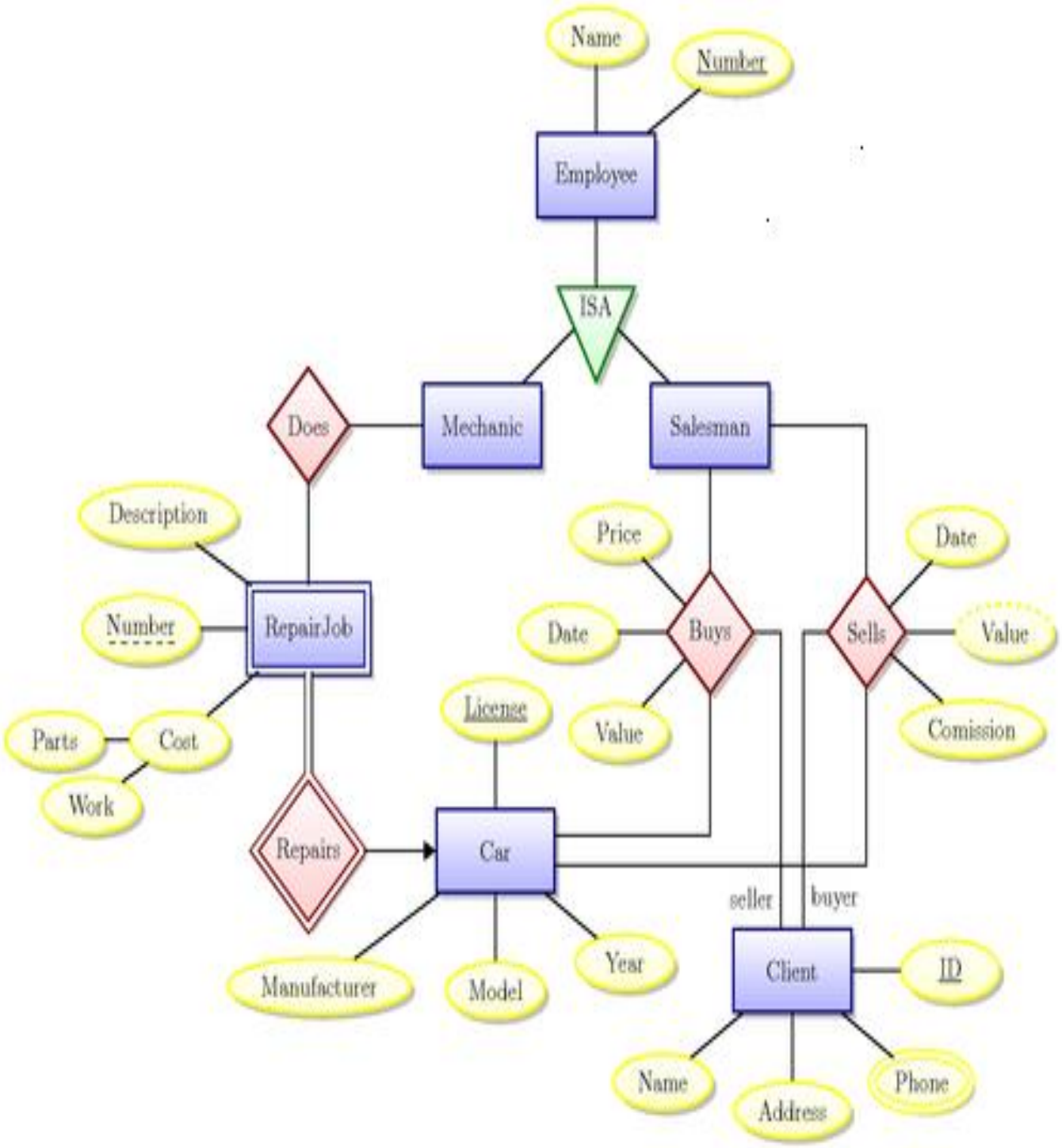


Figure2.13: ER diagram for a mechanic shop that repairs a car.

c. Draw an ER diagram for creating and delivery that project

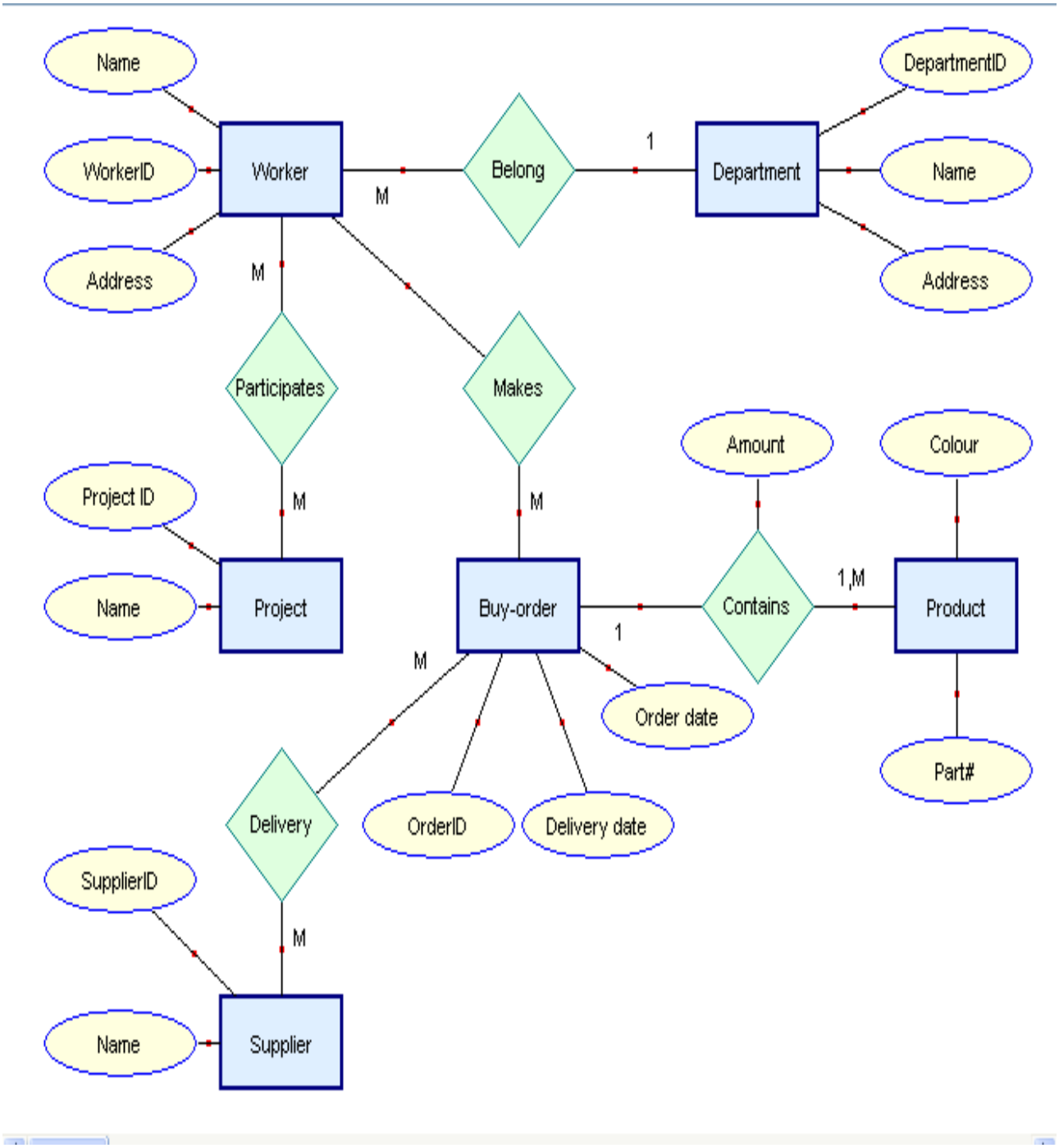


Figure2.14: ER diagram for creating and delivery that project

d. Construct an ER diagram for purchase an item using credit card.

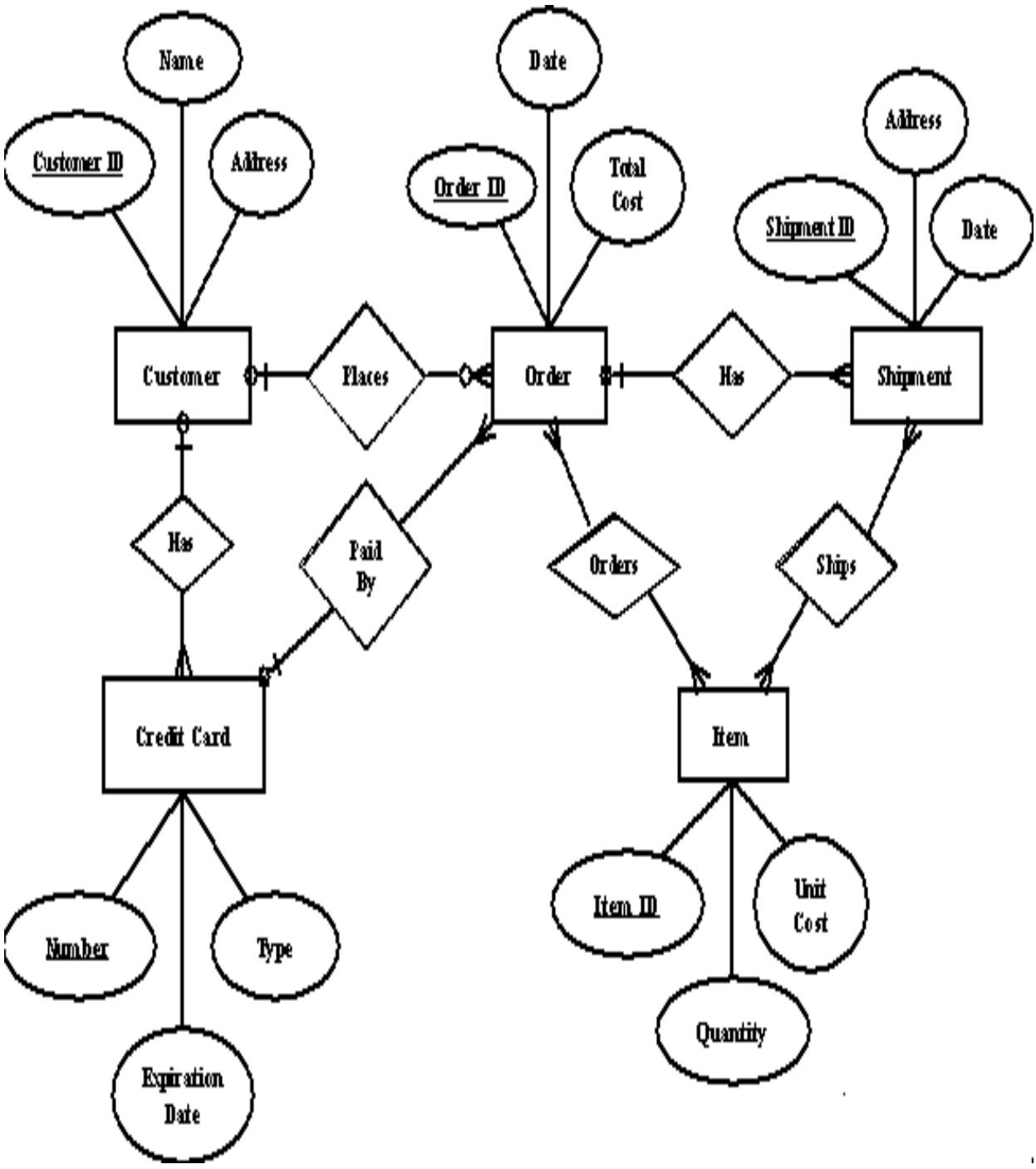


Figure2.15: ER diagram for purchase an item using credit card.

e.. Entity Relationship Diagram and schematic for Books.

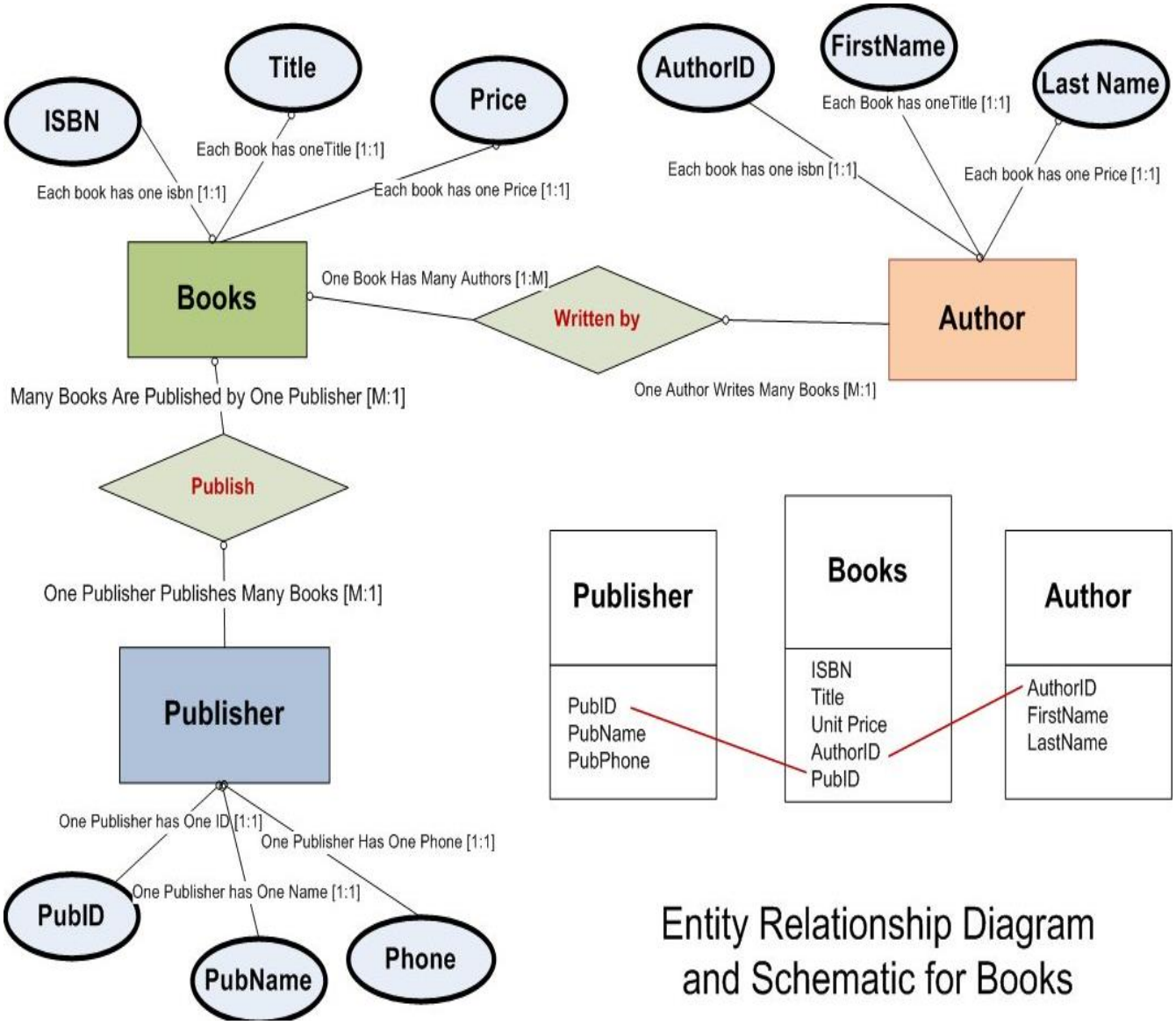


Figure2.15: Entity Relationship Diagram and schematic for Books

4. Explain the concept of functional dependency in detail.

Define a functional dependency. List and discuss the six inference rules for functional dependencies. Give relevant examples. DEC 2011

What are the pitfalls in relational database design? With a suitable example, explain the role of functional dependency in the process of normalization.

MAY 2011

Explain the concept of functional dependency in detail?(DEC 2011))(APR 2019)

Functional dependencies:

- Functional Dependency is when one attribute determines another attribute in a DBMS system.
- Functional Dependency plays a vital role to find the difference between good and bad database design.
- A functional dependency is denoted by an arrow →
- The functional dependency of X on Y is represented by X →Y and X is a determinant set and Y is a dependent set.
- Given a relation *R*, a set of attributes *X* in *R* is said to **functionally determine** another attribute *Y*, also in *R*, (written $X \rightarrow Y$) if and only if each *X* value is associated with exactly one *Y* value.

Rules of Functional Dependencies

Below given are the three most important rules for Functional Dependency:

- **Reflexive rule**

  If X is a set of attributes and Y is subset of X, then X holds a value of Y.

- **Augmentation rule**

  When x -> y holds, and c is attribute set, then ac -> bc also holds. That is adding attributes which do not change the basic dependencies.

- **Transitivity rule**

This rule is very much similar to the transitive rule in algebra if x -> y holds and y -> z holds, then x -> z also holds. X -> y is called as functionally that determines y.

Types of Functional Dependencies

- Multivalued dependency
- Trivial functional dependency
- Non-trivial functional dependency
- Transitive dependency

Multivalued dependency

- Multivalued dependency occurs when two attributes in a table are independent of each other but, both depend on a third attribute.
- A multivalued dependency consists of at least two attributes that are dependent on a third attribute that's why it always requires at least three attributes.
- **Example: Table2.1**Suppose there is a bike manufacturer company which produces two colors(white and black) of each model every year.

BIKE_MODEL	MANUF_YEAR	COLOR
M2011	2008	White
M2001	2008	Black
M3001	2013	White
M3001	2013	Black
M4006	2017	White
M4006	2017	Black

Table2.1: bike manufacturer company

- These two columns can be called as multivalued dependent on BIKE_MODEL.
- The representation of these dependencies is shown below:
    1. BIKE_MODEL → → MANUF_YEAR
    2. BIKE_MODEL → → COLOR

Trivial functional dependency

- The Trivial dependency is a set of attributes which are called a trivial if the set of attributes are included in that attribute.
- So, X -> Y is a trivial functional dependency if Y is a subset of X.
- For example:

Emp_id	Emp_name
AS555	Harry
AS811	George
AS999	Kevin

Table2.2: formatized table

Consider this table with two columns Emp_id and Emp_name.

{Emp_id, Emp_name} -> Emp_id is a trivial functional dependency as Emp_id is a subset of {Emp_id,Emp_name}.

Non trivial functional dependency

- Functional dependency which also known as a nontrivial dependency occurs when A->B holds true where B is not a subset of A.
- In a relationship, if attribute B is not a subset of attribute A, then it is considered as a non-trivial dependency.

Company	CEO	Age
Microsoft	Satya Nadella	51

Google	Sundar Pichai	46
Apple	Tim Cook	57

Table2.3: Non trivial functional dependency

Example:

(Company} -> {CEO} (if we know the Company, we knows the CEO name) But CEO is not a subset of Company, and hence it's non-trivial functional dependency.

Transitive dependency:

A transitive is a type of functional dependency which happens when t is indirectly formed by two functional dependencies.

Example:

Company	**CEO**	**Age**
Microsoft	Satya Nadella	51
Google	Sundar Pichai	46
Alibaba	Jack Ma	54

Table2.4: Transitive dependency

{Company} -> {CEO} (if we know the compay, we know its CEO's name)

{CEO } -> {Age} If we know the CEO, we know the Age

Therefore according to the rule of rule of transitive dependency:

{ Company} -> {Age} should hold, that makes sense because if we know the company name, we can know his age.

Extraneous attributes

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.

Canonical cover:

- A canonical cover of a set of functional dependencies F such that ALL the following properties are satisfied:
- F logically implies all dependencies in F_C.
- F_C logically implies all dependencies in F.
- No functional dependency in F_C contains an extraneous attribute.
- Each left side of a functional dependency in F_C is unique. That is, there are no two dependencies and in such that.

5. Explain in detail about non loss decomposition. (Nov/Dec 2017)

Decomposition

- Decomposition is the process of breaking down in parts or elements.
- It replaces a relation with a collection of smaller relations.
- It breaks the table into multiple tables in a database.
- It should always be lossless, because it confirms that the information in the original relation can be accurately reconstructed based on the decomposed relations.
- If there is no proper decomposition of the relation, then it may lead to problems like loss of information.

The properties of Decomposition,

1. Lossless Decomposition
2. Dependency Preservation
3. Lack of Data Redundancy

1. Nonloss decomposition (or) Lossless joins

- Decomposition must be lossless. It means that the information should not get lost from the relation that is decomposed.
- It gives a guarantee that the join will result in the same relation as it was decomposed.

To check for lossless join decomposition using FD set, following conditions must hold:

1. 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.

   Att(R1) U Att(R2) = Att(R)

2. Intersection of Attributes of R1 and R2 must not be NULL.

   Att(R1) ∩ Att(R2) ≠ Φ

3. Common attribute must be a key for at least one relation (R1 or R2)

   Att(R1) ∩ Att(R2) -> Att(R1) or Att(R1) ∩ Att(R2) -> Att(R2)

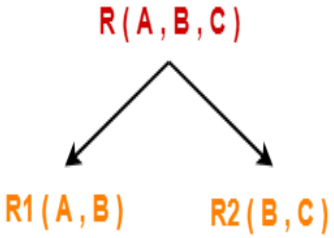
Example-

Consider the following relation R(A , B , C)-

A	B	C
1	2	1
2	5	3
3	3	3

Table2.5:R(A , B , C)

Consider this relation is decomposed into two sub relations R_1(A , B) and R_2(B , C)-



The two sub relations are-

A	B
1	2
2	5
3	3

Table2.6:R_1(A , B)

B	C
2	1
5	3
3	3

Table2.7:R_2(B , C)

Now, let us check whether this decomposition is lossless or not.

For lossless decomposition, we must have-

$R_1 \bowtie R_2 = R$

Now, if we perform the natural join ($\bowtie$) of the sub relations R_1 and R_2 , we get-

A	B	C
1	2	1
2	5	3
3	3	3

Table2.8: $R_1 \bowtie R_2 = R$

- This relation is same as the original relation R. Thus, we conclude that the above decomposition is lossless join decomposition.

2. Dependency Preservation

- Dependency is an important constraint on the database.
- Every dependency must be satisfied by at least one decomposed table.
- If {A → B} holds, then two sets are functional dependent. And, it becomes more useful for checking the dependency easily if both sets in a same relation.
- This decomposition property can only be done by maintaining the functional dependency.
- In this property, it allows to check the updates without computing the natural join of the database structure.

3. Lack of Data Redundancy

- Lack of Data Redundancy is also known as **a Repetition of Information.**
- The proper decomposition should not suffer from any data redundancy.
- The careless decomposition may cause a problem with the data.
- The lack of data redundancy property may be achieved by Normalization process.

6. What is Normalization? Explain the properties that decomposition should posses. (MAY 2007)(APR-2019)

Normalization:

- Normalization of data is a process of **analyzing the given relation schema** based on their **functional dependencies and primary keys** to achieve the desirable properties of
  - ✓ Minimizing redundancy
  - ✓ Minimizing the insertion, deletion & update anomalies.
- This can be achieved by **decomposition**.
- Decomposition should posses 2 desirable properties
  - Lossless join (or) non additive join
  - Functional dependency preservation.

Prime Attribute:

Any attribute of relation schema R,I s called as prime attribute, if it is a member of some candidate key of R.

Non prime Attribute:

Any attribute of relation schema R, is called as non-prime attribute, if it is not a member of any candidate key.

Candidate key:

- If a relation schema has more than one key, then each key is called as a candidate key.
- The data in the database can be considered to be in one of a number of normal forms.
    1. 1st Normal Form
    2. 2nd Normal Form
    3. 3rd Normal Form
    4. Boyce/Codd Normal Form
    5. 4th Normal Form
    6. 5th Normal Form

7. Explain First Normal Form (1 NF) with a suitable Example? (MAY/JUNE 2012)

First Normal Form

- It is the process of eliminating duplicate forms from same data and creating the separate tables for each group of related data and also to identify each row with a unique column or set of columns.
- The table has repeating groups; it is called an unnormalized table.
- Relational databases require that each row only has a single value per attribute, and so a repeating group in a row is not allowed.
- A relation is in first normal form if it meets the **definition of a relation**:
    1. Each attribute (column) value must be a single value only.

2. All values for a given attribute (column) must be of the same type.
3. Each attribute (column) name must be unique.
4. No two tuples (rows) in a relation can be identical.

- **Example:**

Course	Content
Programming	Java, c++
Web	HTML, PHP, ASP

Table2.9: course table

We re-arrange the relation (table) as below, to convert it to First Normal Form.

Course	Content
Programming	Java
Programming	c++
Web	HTML
Web	PHP
Web	ASP

Table2.9: After First Normal Form

Summary: 1NF

- A relation is in 1NF if it contains no repeating groups
- To convert an unnormalised relation to 1NF either
- Flatten the table and change the primary key or
- Decompose the relation into smaller relations, one for the repeating groups and one for the non-repeating groups.
- Remember to put the primary key from the original relation into both new relations.
- This option is liable to give the best results.

8. Explain Second Normal Form (2 NF) with a suitable Example.(MAY/JUNE 2012)

Second Normal Form

- A relation is in 2NF if and only if, it is in 1NF and every non-key attribute is fully functionally dependent on the whole key (primary key). (i.e) All attributes are fully dependent on primary key.
- A relation is in 2NF if it contains no repeating groups and no partial key functional dependencies.
- To convert a relation with partial functional dependencies to 2NF. Create a set of new relations:
  - One relation for the attributes is fully dependent upon the key.
  - One relation for each part of the key that has partially dependent attributes.
- A relation is in second normal form (2NF) if its non-prime attributes are fully functionally dependent on primary *key*.
- A relation is in second normal form if it is free from partial-key dependencies

  Prime attribute − an attribute, which is a part of the prime-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.
- **Types** of functional dependency,
  - Full functional dependency
  - Partial functional dependency
- **Full functional dependency:**
  - A functional dependency P → Q is fully functional dependency if removal of any attribute A from P means that the dependency does not hold any more.
  - **Example:**

If AD → C, is fully functional dependency, then we cannot remove A or D. i.e. C is fully functional dependent on AD. If we are able to remove A or D, then it is not fully functional dependency.

- **Partial functional dependency:**
  - A functional dependency P → Q is partial functional dependency if removal of any attribute A from P means that the dependency still holds.

Example: TEACHER table

TEACHER_ID	SUBJECT	TEACHER_AGE
25	Chemistry	30
25	Biology	30
47	English	35
83	Math	38
83	Computer	38

Table2.10:TEACHER table

- In the given table, non-prime attribute TEACHER_AGE is dependent on TEACHER_ID which is a proper subset of a candidate key. That's why it violates the rule for 2NF.
- To convert the given table into 2NF, we decompose it into two tables:

TEACHER_DETAIL table:

TEACHER_ID	TEACHER_AGE
25	30
47	35
83	38

Table2.11: TEACHER_DETAIL

TEACHER_SUBJECT table:

TEACHER_ID	SUBJECT
25	Chemistry
25	Biology
47	English
83	Math
83	Computer

Table2.12: TEACHER_SUBJECT

Summary: 2NF

- A relation is in 2NF if it contains no repeating groups and no partial key functional dependencies.
- Rule: A relation in 1NF with a single key field must be in 2NF
- To convert a relation with partial functional dependencies to 2NF. Create a set of new relation.
- One relation for the attributes that are fully dependent upon the key
- One relation for each part of the key that has partially dependent attributes.

9 Explain Third Normal Form (3 NF) with a suitable Example.(MAY/JUNE 2012)

Third Normal Form

- The third normal form let us first discuss the concept 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 -> Z is a transitive dependency if the following functional dependencies hold true:

X->Y

Y->Z

Concept of Super key and Candidate Key

- **Super key :** A super key is a set or one of more columns (attributes) to unique identify rows in a table.
- **Candidate key:** The minimal set of attribute which can uniquely identify a tuple known as candidate key.
- For example consider following table

Reg ID	RollNo	Sname
101	1	AAA
102	2	BBB
103	3	CCC
104	4	DDD

Table2.13: Sample data

Super keys

- {RegID}
- {RegID, RollNo}
- {RegID,Sname}
- {RollNo,Sname}
- {RegID, RollNo,Sname}

Candidate Keys

- {RegID}
- {RollNo}

Third Normal Form

- A table is said to be in the Third Normal Form when,

  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 3NF can be defined as :

- A table is in 3NF if it is in 2NF and for each functional dependency x-> y at least one of the following conditions holds:

  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	Rajastan
5	EEE	55555	Mumbai	Maharashtra

Table2.14: Student _details

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-c-sname

sid-e-zipcode

zipcode->cityname

cityname->state

The above denotes the transitive dependency. Hence above table is not in 3NF. We can convert it into 3NF as follows:

Student

sid	sname	zip code
1	AAA	11111
2	BBB	22222
3	CCC	33333
4	DDD	44444
5	EEE	55555

Table2.15: Student

ZIP

zipcode	cityname	state
11111	Pune	Maharashtra
22222	Surat	Gujarat
33333	Chennai	Tamilnadu
44444	Jaipur	Rajasthan
55555	Mumbai	Maharashtra

Table2.16: ZIP

Summary: 3NF

- A relation is in 3NF if it contains no repeating groups, no partial functional dependencies, and no transitive functional dependencies.
- T converts a relation with transitive functional dependencies to 3NF, remove the attributes involved in the transitive dependency and put them in a new relation.
- Rule: A relation in 2NF with only one non-key attribute must be in 3NF
- In a normalized relation a non-key field must provide a fact about the key, the whole key and nothing but the key.
- Relations in 3NF are sufficient for most practical database design problems. However, 3NF does not guarantee that all anomalies have been removed.

10. Explain Boyce/Codd Normal Form (BCNF) with a suitable Example. (MAY/JUNE 2012)

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.
- For a table to be in BCNF, following conditions must be satisfied:

i) R must be in 3rd Nohnal Form

ii) For each functional dependency (X ~ 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-

sid	course	Teacher
1	C	Ankita
1	Java	Poonam
2	C	Ankita
3	C++	Supriya
4	C	Archana

Table2.17:Enrollment Table

From above table following observations can be made:

- One student can enroll for multiple courses. For example student with sidel 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 car taught by both the teachers namely - Ankita and Archana.
- The candidate key for above table can be (sid,course), because using these

columns we can find

- The above table holds following dependencies
    - (sid,course)->Teacher
    - Teacher->course
- The above table is not in BCNF because of the dependency teacher-s-course. that the teacher is not a superkey or in other words, teacher is a non p attribute and course is a prime attribute and non-prime attribute derives thep attribute.
- To convert the above table to BCNF we must decompose above table into Stu and Course tables

sid	Teacher
1	Ankita
1	Poonam
2	Ankita
3	Supriya
4	Archana

Table2.18: Student

Teacher	course
Ankita	C
Poonam	Java
Ankita	C
Supriya	C++
Archana	C

Table2.19: Course

Now the table is in BCNF

11. Explain the concept of Multivalued dependency and Fourth Normal Form(4NF). (MAY/JUNE 2012)

Concept of Multivalued Dependencies

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

1. For a dependency A –7 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 →→
- If there exists a multivalued dependency then the table is not in 4th normal form.
- For example: Consider following table for information about student

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

Table2.20: Student

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

Table2.21: Student (splitup)

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

sid →>Course

sid →> 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 cont ' multivalued dependency.

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

Table2.22: Student Table

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

key:sid,course)

sid	Course
1	C
1	C++
2	Java

Table2.23:Student_course table

Key : (sid,skill)

sid	Skill
1	English
1	German
2	English
2	French

Table2.24: Student_skill table

Thus the table in in 4th normal form.

12. Explain Join Dependency and Fifth Normal Form (5 NF). (MAY/JUNE 2013)

Join Dependency and Fifth Normal Form (5NF)

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	Icecrearn
Sunil	Britania	Biscuits

Table2.25:Company details

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

The above table has multivalued dependency as

Seller→>{Company, 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
Rupali	Godrej
Sharda	Dabur
Sunil	Amul
Sunil	Britania

Table2.26: Seller _Company

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

Table2.27: Seller_product

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

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

Table2.28: Final table after 4th normal form

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

Seller	Company
Rupali	Godrej
Sharda	Dabur
Sunil	Amul
Sunil	Britania

Table2.29: Seller , Company

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

Table2.30: Seller_Product

Company	Product
Godrej	Cinthol
Dabur	Honey
Dabur	HairOil
Dabur	RoseWater
Amul	Icecream
Britania	Biscuit

Table2.31: Company_Product

Thus the table in in 5th normal form.

13. Discuss in detail the steps involved in the ER to relational mapping in the process of relational database design.

- ER Model, when conceptualized into diagrams, gives a good overview of entity-relationship, which is easier to understand.
- ER diagrams can be mapped to relational schema, that is, it is possible to create relational schema using ER diagram.
- It cannot import all the ER constraints into relational model, but an approximate schema can be generated.
- There are several processes and algorithms available to convert ER Diagrams into Relational Schema.
- ER diagrams mainly comprise of −
  - ✓ Entity and its attributes
  - ✓ Relationship, which is association among entities.

Mapping Entity

An entity is a real-world object with some attributes.

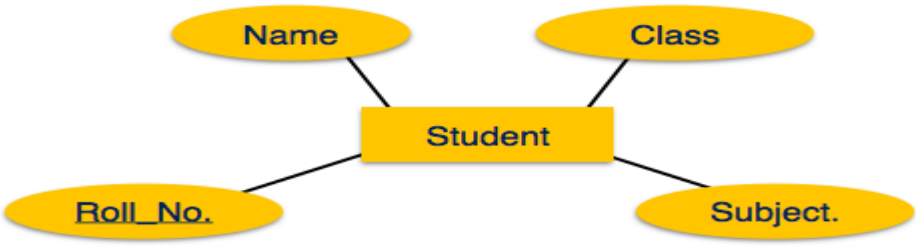


Figure2.16: Mapping Entity

Mapping Process (Algorithm)

- Create table for each entity.
- Entity's attributes should become fields of tables with their respective data types.
- Declare primary key.

Mapping Relationship

A relationship is an association among entities.

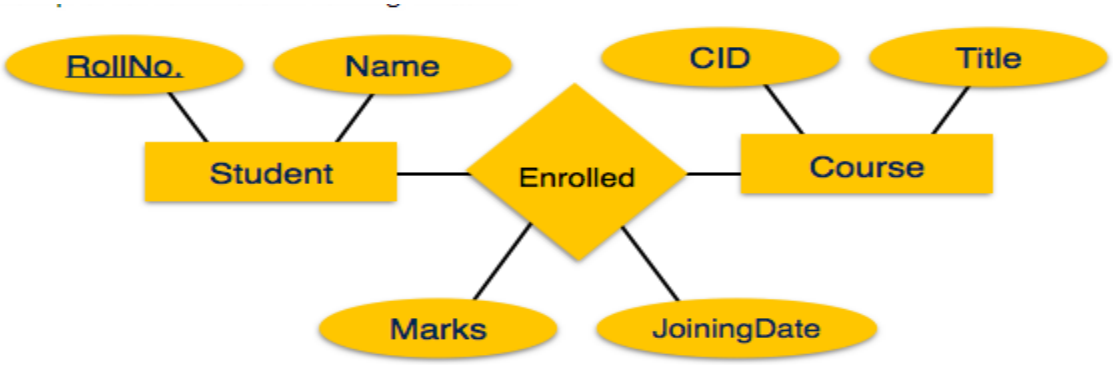


Figure2.17: Mapping Relationship

Mapping Process

- Create table for a relationship.
- Add the primary keys of all participating Entities as fields of table with their respective data types.
- If relationship has any attribute, add each attribute as field of table.
- Declare a primary key composing all the primary keys of participating entities.
- Declare all foreign key constraints.

Mapping Weak Entity Sets

A weak entity set is one which does not have any primary key associated with it.

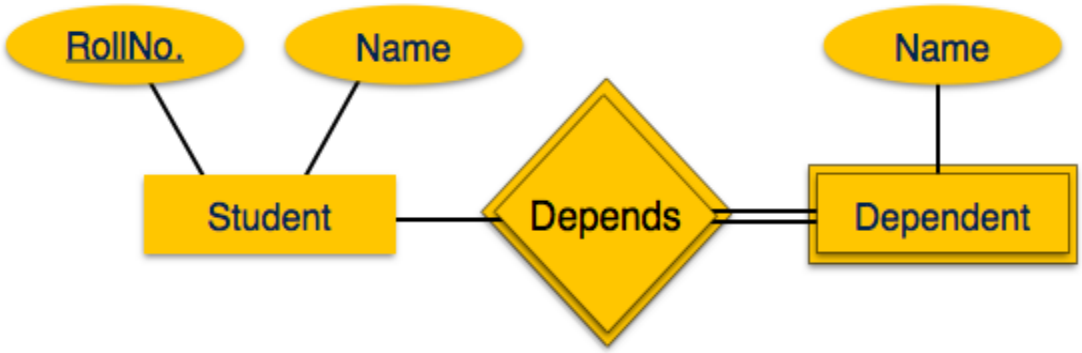


Figure2.18: Mapping Weak Entity Sets

Mapping Process

- Create table for weak entity set.
- Add all its attributes to table as field.
- Add the primary key of identifying entity set.
- Declare all foreign key constraints.

Mapping Hierarchical Entities

ER specialization or generalization comes in the form of hierarchical entity sets.

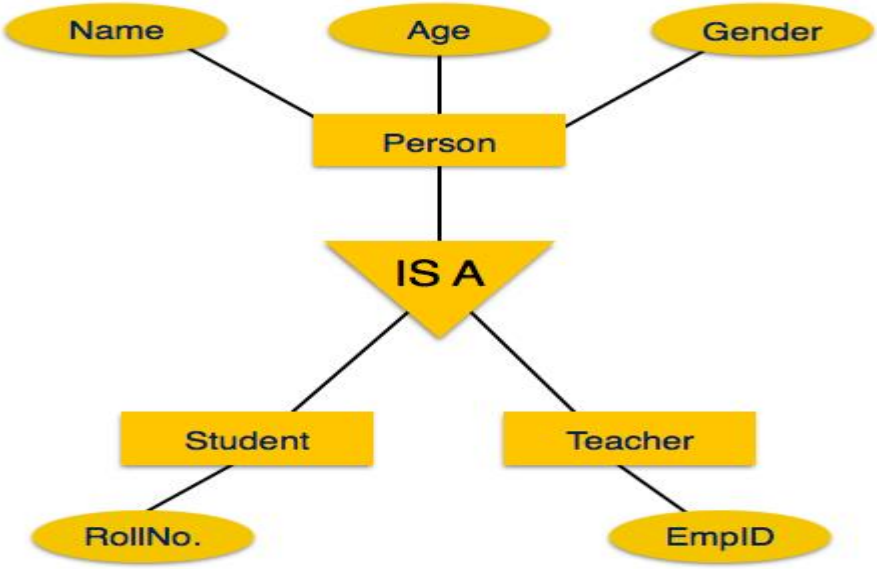


Figure2.19: Mapping Hierarchical Entities

Mapping Process

- Create tables for all higher-level entities.
- Create tables for lower-level entities.
- Add primary keys of higher-level entities in the table of lower-level entities.
- In lower-level tables, add all other attributes of lower-level entities.
- Declare primary key of higher-level table and the primary key for lower-level table.
- Declare foreign key constraints.

UNIT -III – TRANSACTIONS

Transaction Concepts – ACID Properties – Schedules – Serializability – Transaction support in SQL –Need for Concurrency – Concurrency control –Two Phase Locking- Timestamp – Multiversion – Validation and Snapshot isolation– Multiple Granularity locking – Deadlock Handling – Recovery Concepts – Recovery based on deferred and immediate update – Shadow paging – ARIES Algorithm.

PART – A

1. What are the ACID properties? (Nov/Dec 2020 & Apr/May 2021)

A - **A**tomicity
C - **C**onsistency
I -**I**solation
D - **D**urability
It is a set of properties that guarantee database transactions are processed reliably.

2. What are two pitfalls of lock-based protocols? (APRIL/MAY-2011)

- Deadlock
- Starvation

3. What is transaction? (APRIL/MAY-2010)

- Collections of operations that form a single **logical unit of work** are called **transactions.**
- A program contains statements of the form **begin transaction** and **end transaction**.
- The transaction consists of all operations executed between the begin transaction and end transaction

4. What are the two statements regarding transaction?

The two statements regarding transaction of the form:
- Begin transaction
- End transaction

5. What are the properties of transaction?

Atomicity

- Consistency
- Isolation
- Durability

6. What is recovery management component?

Ensuring durability is the responsibility of a software component of the base system called the recovery management component.

7. When is a transaction rolled back?

- Any changes that the aborted transaction made to the database must be undone.
- Once the changes caused by an aborted transaction have been undone, then the transaction has been rolled back.

8. What are the states of transaction? (Apr/May 2019)

The states of transaction are

- Active
- Partially committed
- Failed
- Aborted
- Committed
- Terminated

9. List out the statements associated with a database transaction.

- Commit work
- Rollback work

10. What is a shadow copy scheme?

- It is simple, but efficient, scheme called the shadow copy schemes.
- It is based on making copies of the database called shadow copies that one transaction is active at a time.
- The scheme also assumes that the database is simply a file on disk.

11. Give the reasons for allowing concurrency.

The reasons for allowing concurrency are if the transactions run serially, a short transaction may have to wait for a preceding long transaction to complete, which can lead to unpredictable delays in running a transaction. So concurrent execution reduces the unpredictable delays in running transactions.

12. What is average response time?

The average response time is that the average time for a transaction to be completed after it has been submitted.

13. What are the two types of serializability?

The two types of serializability are

- Conflict serializability
- View serializability

14. Define lock.

Lock is the most common used to implement the requirement is to allow a transaction to access a data item only if it is currently holding a lock on that item.

15. What are the different modes of lock?

The modes of lock are:

- Shared
- Exclusive

16. Define deadlock.

Neither of the transaction can ever proceed with its normal execution. This situation is called deadlock.

17. Define the phases of two-phase locking protocol. (NOV/DEC 2011) (MAY/JUNE 2013)

- Growing phase: a transaction may obtain locks but not release any lock.
- Shrinking phase: a transaction may release locks but may not obtain any new locks.

18. Define upgrade and downgrade.

- It provides a mechanism for conversion from shared lock to exclusive lock is known as upgrade.
- It provides a mechanism for conversion from exclusive lock to shared lock is known as

downgrade.

19. What is a database graph?

The partial ordering implies that the set D may now be viewed as a directed acyclic graph, called a database graph.

20. What are the two methods for dealing deadlock problem?

The two methods for dealing deadlock problem are deadlock detection and deadlock recovery.

21. What is a recovery scheme?

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.

22. What are the two types of errors?

The two types of errors are:
- Logical error
- System error

23. What are the storage types?

The storage types are:
- Volatile storage
- Nonvolatile storage

24. Define blocks.

The database system resides permanently on nonvolatile storage, and is partitioned into fixed-length storage units called blocks.

25. What is meant by Physical blocks?

The input and output operations are done in block units. The blocks residing on the disk are referred to as physical blocks.

26. What is meant by buffer blocks?

The blocks residing temporarily in main memory are referred to as buffer blocks.

27. What is meant by disk buffer?

The area of memory where blocks reside temporarily is called the disk buffer.

28. What is meant by log-based recovery?

The most widely used structures for recording database modifications is the log. The log is a sequence of log records, recording all the update activities in the database. There are several

types of log records.

29. What are uncommitted modifications?

The immediate-modification technique allows database modifications to be output to the database while the transaction is still in the active state. Data modifications written by active transactions are called uncommitted modifications.

30. Define shadow paging.

An alternative to log-based crash recovery technique is shadow paging. This technique needs fewer disk accesses than do the log-based methods.

31. Define page.

The database is partitioned into some number of fixed-length blocks, which are referred to as pages.

32. What is current page table and shadow page table?

The key idea behind the shadow paging technique is to maintain two-page tables during the life of the transaction: the current page table and the shadow page table. Both the page tables are identical when the transaction starts.

The current page table may be changed when a transaction performs a write operation.

33. What are the drawbacks of shadow-paging technique?

- Commit Overhead
- Data fragmentation
- Garbage collection

34. Define garbage collection.

Garbage may be created also as a side effect of crashes. Periodically, it is necessary to find all the garbage pages and to add them to the list of free pages. This process is called garbage collection.

35. What is strict two-phase locking protocol and rigorous two-phase locking protocol? (MAY/JUNE 2012)

- In **strict two-phase locking protocol,** all exclusive mode locks taken by a transaction is held until that transaction commits.
- **Rigorous two-phase locking protocol** requires that all locks be held until the transaction

commits.

36. How the time stamps are implemented?

• Use the value of the system clock as the time stamp. That is a transaction's time stamp is equal to the value of the clock when the transaction enters the system.

• Use a logical counter that is incremented after a new timestamp has been assigned; that is the time stamp is equal to the value of the counter.

37. What are the time stamps associated with each data item?

• W-timestamp (Q) denotes the largest time stamp if any transaction that executed WRITE (Q) successfully.

• R-timestamp (Q) denotes the largest time stamp if any transaction that executed READ (Q) successfully.

38. What are the facilities available in SQL for Database recovery? (MAY/JUNE 2010)

- SQL support for transaction, and hence for transaction-based recovery.
- Most executable "SQL statements are guaranteed to be atomic.
- SQL provides direct analogs of BEGIN TRANSACTION, COMMIT, and ROLLBACK called START TRANSACTION, COMMIT WORK, and ROLLBACK WORK, here is the syntax for START TRANSACTION.

39. What are the benefits and disadvantages does strict two-phase locking provide?

Benefits:

▪ Transaction only read values of committed transaction
▪ No cascaded aborts.

Disadvantages:

▪ Limited concurrency
▪ Deadlocks

40. List the two commonly used Concurrency Control techniques.

●Two phased Locking

●Time stamp-based ordering

41. List the SQL statements used for transaction control.

●Commit, Roll back, Savepoint.

42. What is shadow paging?

●An alternative to log-based crash recovery technique is shadow paging.

●This technique needs fewer disk accesses than do the log-based methods. Here is divided into pages that can be stored anywhere in the disk. Inorder to identify the location of a give page we use page table.

43. List the properties that must be satisfied by the transaction.

● Atomicity

●Consistency reservation

●Isolation

●Durability or permanency

44. What are the types of transparencies that a distributed database must support?

1. Location transparency
2. Fragmentation transparency
3. Replication transparency

45. What are the three kinds of intent locks?

●Intent Exclusive

●Intent shared

●Shared-Intent Exclusive

PART – B

1. Explain in detail about transactions. (Nov/Dec 2021)

Collections of operations (processes) that form a single logical unit of work are called **transactions.** (or) Transaction is an executing program that forms a logical unit of database processing.

- A transaction includes one or more database access operations. These can include insertion, deletion, modification or retrieval operations.
- The database operation can be embedded within an application or can be specified by high level query language.
- These operations executed between the **begin transaction** and **end transaction.**

Transactions access data using two operations:

- ✓ **read(*X*)-** which transfers the data item *X* from the database to a local buffer belonging to the transaction that executed the read operation.
- ✓ **write(*X*)-** which transfers the data item *X* from the local buffer of the transaction that executed the write back to the database.

In a real database system, the write operation does not automatically result in the immediate update of the data on the disk;

- The write operation may be temporarily stored in memory and executed on the disk later. For now, however, we shall assume that the write operation updates the database immediately.
- A transaction must be in one of the following states is shown in Fig 3.1.
  - ✓ Active
  - ✓ Partially committed
  - ✓ Failed
  - ✓ Aborted
  - ✓ Committed

Active

- ✓ In this state, the transaction is being executed.
- ✓ This is the initial state of every transaction.

Partially Committed

When a transaction executes its final operation, it is said to be in a partially committed state.

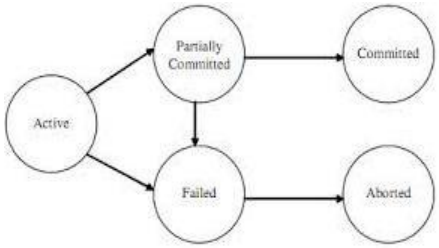


Fig.3.1 Transaction states

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 cans no longer proceed further.

Aborted

If any of the checks fails and the transaction has reached a failed state, then the recovery manager rolls back all its write operations on the database to bring the database back to its original state where it was prior to the execution of the transaction. Transactions in this state are called **aborted.**

The database recovery module can select one of the two operations after a transaction aborts.

- ✓ Re-start the transaction
- ✓ Kill the transaction

Committed

If a transaction executes all its operations successfully, it is said to be committed.

Example:

Let T_i be a transaction that transfer money from account A (5000) to account B. The transaction can be defined as

T_i:
read (A)
A: = A – 5000(withdraw from A)
write (A); (update A)
read (B) (deposit B)
B: = B + 5000
write (B) (update B)

2. Explain in detail about properties of transaction (ACID Properties). (Nov/Dec 2020 & Apr/May 2021)

ACID property:

A - Atomicity
C - Consistency
I - Isolation
D - Durability

It is a set of properties that guarantee **database transactions** are **processed reliably.**

Example:

Ti-read(A);
A:=A-50;
write(A);
read(B);
B:=B+50;
Write(B);

ATOMICITY:

Either all operation of the transaction **is reflected properly** in the database, **or none are**.

- This **all-or-none** properly referred to as atomicity.
- Atomicity is handled by **Transaction Management** Component.

Example:

Consider before execution of transaction Ti, A=1000 and B=2000.

Suppose during the execution of transaction Ti , a failure occurs after the write(A) operation but before write (B) operation the values of A & B reflected in the database are A=950, B=2000 is an inconsistent state.

- Ensure that such inconsistent are not visible in database.
- To ensure atomicity, the database keeps track of the old values of any data. This information is written to file called log.
- If the transaction does not complete its execution, the database system restores the old values from the log.
- **Responsibility**
- Ensuring the atomicity is the responsibility of the database system particularly

recovery system.

CONSISTENCY:

Execution of the transaction in **isolation** (i.e.) with no other transaction executing in **parallel** preserves the consistency.

Example:

The consistency requirement is the sum of A and B be unchanged by the execution of transaction.

- If the database is consistent before the execution of transaction, the database remains consistent after the execution of transaction.

Responsibility:

- Responsibility is the **application programmer** who codes the transaction

ISOLATION:

Even though multiple transaction may execute concurrently, the system guarantees that for every pair of transaction Ti and Tj it appears to Ti that either Tj finished execution before Ti started execution or after Ti finished.

- Each **transaction in unaware of other transaction** executing concurrently in the system.

Example:

Consider the database is temporarily in consistent while the transaction to transfer funds from A to B is executing with the deducted total written toB.

- A way to avoid this problem is to execute the transaction serially (i.e.) one after another.
- **Responsibility:**
- Ensuring Isolation property is the responsibility of **concurrent control component**.

DURABILITY:

After a transaction completes successfully, the **changes made to the database persists, even if there are system failures.**

- Durability ensures that Updates carried out by the transaction have been written to disk before the transactions completes.
  - **Responsibility:**
- Ensuring Durability is the responsibility of a component called **the recovery management**.

3. Explain in detail about Schedule and its types with example.

The objective of concurrency control protocol is to schedule transactions in such a way as to avoid any interference between them.

- **Schedule** is a sequence of the operation by a set of concurrent transactions that preserves the order of the operations in each of the individual transactions.
- A schedule for a set of transactions must consist of all instructions of those transactions.
- It Must preserve the order in which the instructions appear in each individual transaction.
- **Serial schedule** is a schedule where the operation of each transaction is executed consecutively without any interleaved operation from other transactions.
- A Schedule S is said to be serializable if it is equivalent to **serial schedule.**
- In a serial schedule, the transaction is performed in serial order (i.e.,) if T1 and T2 are transaction, serial order would be T1 followed by T2 or T2 followed by T1.
- **Non serial schedule** is a schedule where the operations from a set of concurrent transaction are interleaved.

Serializable schedule (Nov/Dec 2021):

The serializability of schedules is used to find non-serial schedules that allow the transaction to execute concurrently without interfering with one another.

- It identifies which schedules are correct when executions of the transaction have interleaving of their operations.
- A non-serial schedule will be serializable if its result is equal to the result of its transactions executed serially.

Example: 1

Non serial schedule S1:

T1	T2
read(x)	
write(x)	
	read(x)
	write(x)
read(y)	
write(y)	
	read(y)
	write(y)

Non serial schedule S2:

T1	T2
read(x)	
write(x)	
	read(x)
read(y)	
	write(x)
write(y)	
	read(y)
	write(y)

Serial schedule S3:

T1	T2
read(x)	
write(x)	
read(y)	
write(y)	
	read(x)
	write(x)
	read(y)
	write(y)

- If a set of transaction executed concurrently, the schedule is correct if it produces the same results as some serial execution such a schedule is called **serializable**.
- Schedule S3 is a serial schedule and, since S1 and S2 are equivalent to S3, S1 and S2 are serializable schedule.

4. Brief explanation about serializability and its types. (Apr/May 2019)

Serializability is the classical **concurrency scheme.**

- It ensures that a schedule for executing concurrent transactions is equivalent to one that executes the transactions serially in some order.
- Serializable schedule if a schedule is equivalent to some serial schedule then that schedule is called **Serializable schedule**.
- **The object of serializability** is to find on serial schedules that allow transaction to execute concurrently without interfering with one another, and there by produce a database state that could be produced by a serial execution.

Example: 1

Non serial schedule S1

T1	T2
read(x)	
write(x)	
	read(x)
	write(x)
read(y)	
write(y)	
	read(y)
	write(y)

Non serial schedule S2

T1	T2
read(x)	
write(x)	
	read(x)
read(y)	
	write(x)
write(y)	
	read(y)
	write(y)

Serial schedule S3

T1	T2
read(x)	
write(x)	
read(y)	
write(y)	
	read(x)
	write(x)
	read(y)
	write(y)

If a set of transaction executed concurrently, the schedule is correct if it produces the same results as some serial execution such a schedule is called **serializable**,

- Schedule S3 is a serial schedule and, since S1 and S2 are equivalent to S3, S1 and S2 are serializable schedule.
- Types of serializability:
    1. Conflict serializable.
    2. View serializable

Conflict serializability:

A schedule S is conflict serializable if it is conflict equivalent to a serial schedule.

Conflict instruction:

Instructions Ii and Ij of transactions Ti and T_j respectively are conflict, if and only if there exist some item Q accessed by Ii and Ij and at least one of these instructions have write Q.

Cases to consider:

- I_i = read(Q), I_j=read(Q)
  Order of Ii and Ij does not matter, since the same value of Q is read.
- I_i= read(Q) , I_j = write(Q)
  If Ii comes before I_j then T_i does not read the value of Q written by T_j. If Ij comes before I_i, then T_i reads the value of Q that is written by T_j. Conflict occurs and it results in inconsistency problem.
- I_i =write(Q), I_j=read(Q)
  Conflict occurs and results in uncommitted dependency problem.
- I_i=write(Q), Ij= write(Q)
- Since both instructions are write operation, the order of these instructions does not affect either T_i or T_j. However, the value obtained by the next read(Q) instruction of S is affected.

Example conflicting instructions S1

T1	T2
read(x)	
write(x)	
	read(x)
	write(x)
read(y)	
write(y)	
	read(y)
	write(y)

The write (x) instruction of T1 conflicts with the read (x) instruction of T2.

- However, the write (x) instruction of T2 does not conflict with the read (y) instruction of T1, because the two instructions access different data items.
- If a schedule S can be transformed into a schedule S' by a series of swaps of non-conflicting instruction, then S and S' are conflict equivalent.
- Swap the read (y) instruction of T1 with the read (x) instruction of T2.
- Swap the write (y) instruction of T1 with the write (x) instruction of T2.
- Swap the write (y) instruction of T1 with the read (x) instruction of T2.

The final result of these swaps is a serial **schedule S2**

T1	T2
read(x)	
write(x)	
read(y)	
write(y)	
	read(x)
	write(x)
	read(y)
	write(y)

The concept of **conflict equivalence** leads to the concept of conflict serializability.

- A schedule S is conflict serializable, if it is conflict equivalents to a serial schedule.
- Thus, schedule S2 is conflict serializable, since it is conflict equivalent to the serial schedule S1.

View serializability:

A schedule S is view serializable if it is view equivalent to a serial schedule.The schedule S and S' are said to be view equivalent if the following conditions met:

1. For each data item x, if transactions T1 reads the initial value of x in schedule S, then transaction T1 must, in schedule S', also read the initial value of x.
2. For each data item x, if transaction T1 executes read (x) in schedule S, and if that value was produced by a write(x) operation executed by transaction T2, then the read(x) operation of transaction T1 must, in schedule S, also read the value of x that was produced by the same write (x) operation of transaction T2.
3. For each data item, x, the transaction that performs the final write(x) operation in schedule S must perform the final write(x) operation in schedule S'.

Example: Schedule1

T1	T2
read(x) write(x) read(y) write(y)	 read(x) write(x) read(y) write(y)

Schedule2

T1	T2
read(x) write(x) read(y) write(y)	 read(x) write(x) read(y) write(y)

- Schedule 1 is view equivalent to schedule2, because the value of A and B read by transaction T2 were produced by T1 in both schedules.
- The concept of the view equivalent leads to the concept of view serializability.
- A schedule S is **view serializable** if it is view equivalent to a serial schedule.

Test for serializability:

Serializability is tested by the use of precedence graph G(V, E) where V -> vertex and E -> Edge.

The set of all edges Ti->Tj exists if and only if these three conditions must hold

- Ti executes read Q before Tj executes write Q
- Ti executes write Q before Tj executes read Q
- Ti executes write Q before Tj executes write Q

If the precedence graph has no cycles it is conflict serializable is shown in fig 3.2

If the precedence graph has cycles it is not conflict serializable is shown in fig 3.3

Example 1:

Fig. 3.2 Conflict Serializable

Read(A)

A: =A-50
Write(A) Read(B)
B: =B-50
Write(B)

Read(A) Temp=A*0.1
A:=A-temp Write(A)
Read(B)
B:=B+temp
write(B)

Example 2:

Fig. 3.3 Non-Conflict Serializable

T1	T2
Read(A)	
A:=A-50	
	Read(A)
	Temp=A*0.1
	A:=A-temp
	Write(A)
	Read(B)
Write(A)	
Read(B)	
B:=B+50	
Write(B)	
	B:=B+temp
	Write(B)

5. Briefly explain about concurrency and its problems. (APRIL/MAY 2010)

CONCURRENCY:

Concurrency allows many transactions to access the same database at the same time. Process of managing simultaneous execution of transactions in a shared database, to ensure the serializability of transactions, is known as **concurrency control.**

- Process of managing simultaneous operations on the database without having them interferes with one another.
- Prevents interference when two or more users are accessing database simultaneously and at least one is updating data.
- Although two transactions may be correct in themselves, interleaving of operations may produce an incorrect result.

Need for Concurrency:

A major objective in developing a database is to enable many users to access shared data concurrently.

- Concurrent access is relatively easy if all users are only reading data, as there is no way that they can interfere with one another.

- However, when two or more users are accessing the database simultaneously and at least one is updating data, there may be interference that can result in inconsistencies.
- Simultaneous execution of transactions over a shared database can create several data integrity and consistency problems (or) In concurrency transaction, it faces **three Concurrency Problems:**
  - ✓ The lost update problem
  - ✓ The uncommitted dependency problem (or) temporary update problem
  - ✓ The inconsistent analysis problem (or) incorrect summary problem

The 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. Successfully completed update is overridden by another user.

The transaction details shown in fig 3.4.

- ❖ Transaction A retrieves some tuple t at time t1;
- ❖ Transaction B retrieves that same tuple t at time t2;
- ❖ Transaction A updates the tuple at time t3 and transaction B updates the same tuple at time t4.
- ❖ So, Transaction A's update is lost at time t4, because transaction B overwrites it without even looking at it.

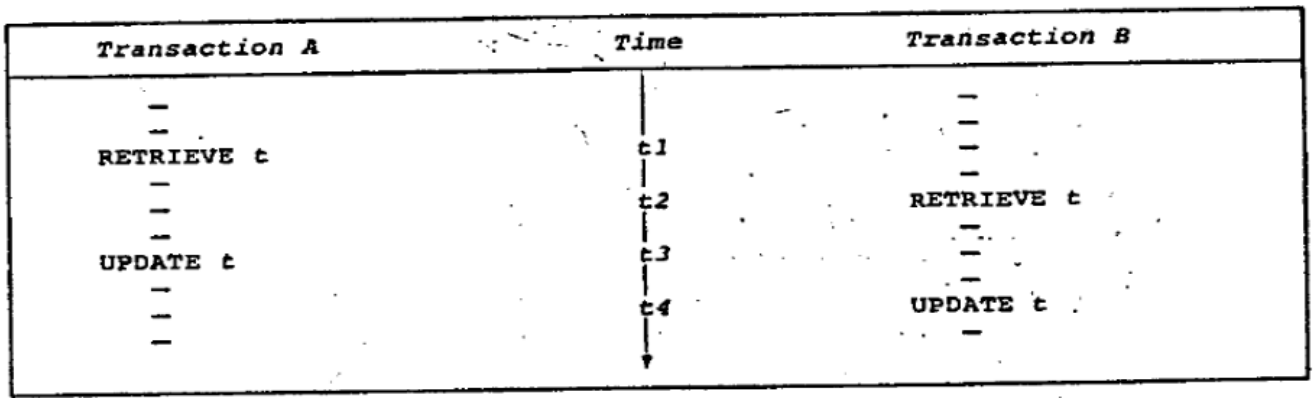


Fig 3.4 Transaction A loses an update at time t4

The Uncommitted Dependency problem (or) temporary update problem:-

This problem occurs when one transaction updates a database item and then the transaction fails for some reason. The updated item is accessed by another transaction before it is changed back to its original value.

- Occurs when one transaction can see intermediate results of another transaction before it has committed.
- The uncommitted dependency problem arises if one transaction is allowed to retrieve and update a tuple that has been updated by another transaction but not yet committed by that other transaction.
- There is possibility that it never will be committed but will be rolled back, so first transaction will have seen some data that no longer exists.

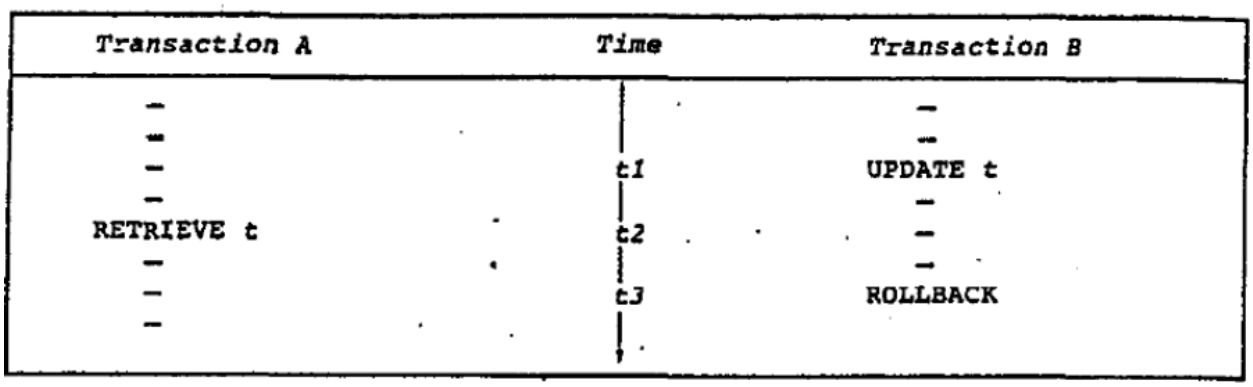


Fig 3.5: Transaction A becomes dependent on an uncommitted change at time t2

In the above Figure 3.5 shows

- ❖ Transaction A sees an uncommitted update, also called an uncommitted change at time t2. That update is then undo at time t3.
- ❖ Transaction A is therefore operating on a false assumption, that tuple t has the value seen at time t2, whereas in fact it has whatever value it had prior to time t1.
- ❖ Transaction A might produce an incorrect result.
- ❖ The rollback of transaction B might be due to no fault of B. it might for example the system crash.

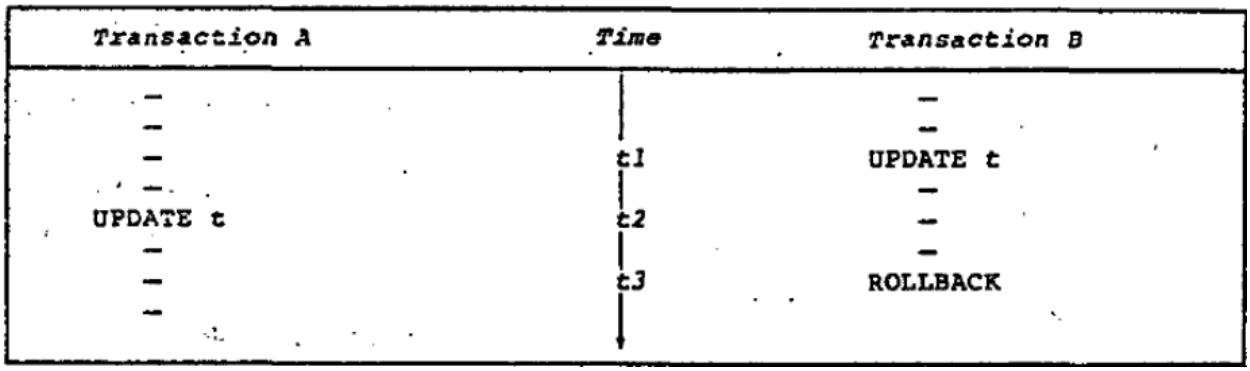


Fig 3.6: Transaction A updates uncommitted change at time t2, and loses that update at time t3.

In the above Figure 3.6 shows

- The transaction A become dependent on an uncommitted change at time t2, but it actually loses an update at time t3, because rollback at time t3 because the rollback at time t3 causes tuple t to be restored to its value prior to time t1.
- This is another version of the last update problem.

The inconsistent Analysis problem (or) incorrect summary problem:

If one transaction is calculating an aggregate summary function on a number of records while other transactions are updating some of these records, the aggregate function may calculate some values before they are updated and others after they are updated. The transaction details showed in fig 3.7.

- It occurs when transaction reads several values but second transaction updates some of them during execution of first.
- The transaction A and B operating on account (ACC) tuples. Transaction A is summing account balances. Transaction B is transferring an amount 10 from account 3 to account 1.
- The result produced by A, 110 is incorrect. If A were to go on to write that result back into the database, it would actually leave the database in an inconsistent state.
- A has seen an inconsistent state of the database and has therefore performed an inconsistent analysis. Since B commits all its updates before A sees ACC3.

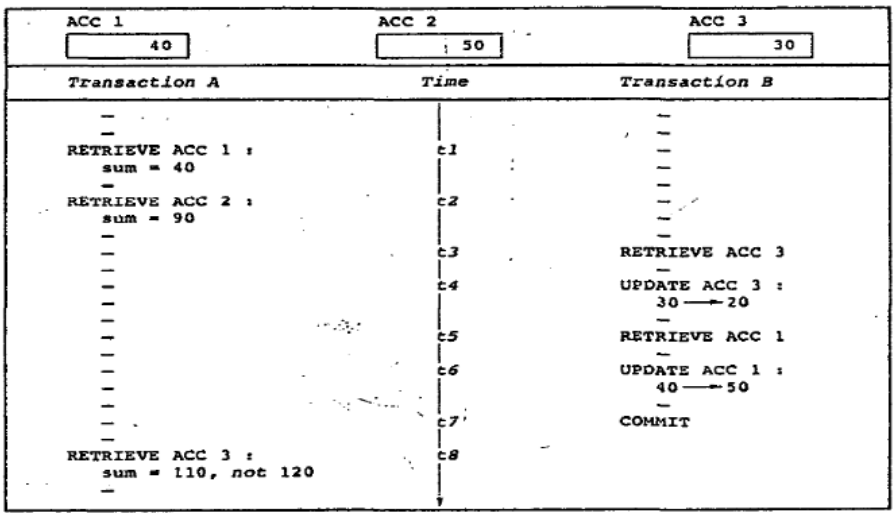


Fig 3.7: Transaction A performs an inconsistent analysis

A Closer Look:

The operations in concurrency point of view are database retrievals and database updates. The transaction is consisting of sequence of such operations (BEGIN TRANSACTION and COMMIT or ROLLBACK).

A and B are concurrent transaction. The Problems can occur if A and B want to read or write the same database object, say tuple t.

There are four possibilities:

- **RR**- A and B want read same tuple t. Read cannot interface with each other so there is no problem in this case.
- **RW**- A read t and then B wants to write t. If B is allowed to perform its write, inconsistent analysis is affected by RW conflict.

The B does perform its write and a then reads t again, it will see a value difference from what it saw before, state of affairs referred to as non-repeatable read, it causes RW conflicts.

- **WR**: A writes t and then B wants to read it. If B is allowed to perform its read then the uncommitted dependency problem can arise. The uncommitted dependencies are caused by WR conflicts.
- **WW**: A writes t and B wants to write t. If B is allowed to perform its write, then the lost problem can arise, the lost updates are caused by WW conflicts. Example: B write, if it is allowed, is said to be dirty write.

6. Explain in detail about locking protocols with an example. (Nov/Dec 2019)

Locking Protocol:

In concurrency system, such kind of control mechanism is clearly needed to ensure that concurrent transactions do not interfere with each other. This concurrency control mechanism called locking.

- Locking is a procedure used to control concurrent access to data when one transaction is accessing the database, a lock may deny access to other transaction to prevent incorrect results.
- One way to ensure isolation is to require that data items be accessed in a mutually exclusive manner.
- 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.
- Each transaction in the system follows a set of rules, called a locking protocol, indicating when a transaction may lock and unlock each of the data items.
- Locking protocols restrict the number of possible schedules. The set of all such schedules is a proper subset of all possible serializable schedules.

Data items can be locked in two modes:

1) **Shared Mode**
   - If a transaction Ti has obtained a shared-mode lock on item Q, then Ti **can read**, **but cannot write.**
   - It is denoted by **S.**
2) **Exclusive**
   - If a transaction Ti has obtained an exclusive-mode lock on item Q, then Ti can **both read and write Q.**
   - It is denoted by **X**

A transaction requests **a shared lock** on data item by executing the **lock-S(Q)** instruction. Similarly, a transaction requests an exclusive lock through the **lock-X(Q)** instruction. A transaction can unlock a data item Q by the **unlock(Q)** instruction.

- To access a data item, transaction Ti must first lock that item. If the data item is already locked by another transaction in an incompatible **mode,** the concurrency-control manager will not grant the lock until all incompatible locks held by other transactions have been released.
- Thus, Ti is made to wait until all incompatible locks held by other transactions have been released.

Given a set of lock modes, we can define a compatibility function on them as follows:

- Let A and B represent arbitrary lock modes. Suppose that a transaction Ti requests a lock of mode Aon item Q on which transaction Tj(Ti=Tj) currently holds a lock of mode B.
- If transaction Ti can be granted a lock on Q immediately, in spite of the presence of the mode Block, then we say mode A is compatible with mode B. Such a function can be represented conveniently by a matrix.
- The compatibility relation between the two modes of locking appears in the matrix Figure 3.8. An element comp(A, B) of the matrix has the value true if and only if mode is compatible with mode B.

	S	X
S	True	False
X	False	False

Fig 3.8 Locking Matrix

Consider again the banking example. Let A and B be two accounts that are accessed by transactions T1 and T2.

```
T1: lock-X(B);
    read(B);
    B: =B −50;
    write(B);
    unlock(B);
    lock-X(A);
    read(A);
    A: =A + 50;
```

```
                write(A);
                unlock(A).
T2: lock-S(A);
                read(A);
                unlock(A);
                lock-S(B);
                read(B);
                unlock(B);
                display(A+B).
```

- Transaction T1transfers $50 from account B to account A.
- Transaction T2displays the total amount of money in accounts A and B—that is, the sum A+ B.
- Suppose that the values of accounts A and B are $100 and $200, respectively. If these two transactions are executed serially, either in the orderT1, T2or the orderT2, T1, then transaction T2will display the value $300.
- If, however, these transactions are executed concurrently, then schedule 1. In this case, transaction T2 displays $250, which is incorrect. The reason for this mistake is that the transaction T1unlocked data item B too early, as a result of which T2saw an inconsistent state.

Suppose now that unlocking is delayed to the end of the transaction. Trans-action T3corresponds to T1 with unlocking delayed. Transaction T4corresponds to T2with unlocking delayed.

Transaction T3(transaction T1with unlocking delayed).

```
T3: lock-X(B);
        read(B);
        B: =B −50;
        write(B);
        lock-X(A);
        read(A);
        A: =A + 50;
        write(A);
        unlock(B);
        unlock(A).
```

Transaction T4(transaction T2with unlocking delayed).

T4: lock-S(A);
read(A);
lock-S(B);
read(B);
display(A+B);
unlock(A);
unlock(B).

Consider the partial schedule of T3 and T4. Since T3 is holding an exclusive-mode lock on T4 is requesting a shared-mode lock on B, T4is waiting for T3to unlock B. Similarly, since T4 is holding a shared-mode lock on A and T3 is requesting an exclusive-mode lock on A, T3 is waiting for T4 to unlock A. Thus, we have arrived at a state where neither of these transaction scan ever proceeds with its normal execution. This situation is called **deadlock.**

7. Explain Two phase locking protocol& it types in detail.

The protocol that ensures serializability is called two phase locking protocol. **Two-phase locking protocol** needs that each transaction issue lock and unlock requests in two phases:

- ✓ **Growing phase**. A transaction may obtain locks, but may not release any lock.
- ✓ **Shrinking phase**. A transaction may release locks, but may not obtain any new locks.

Initially, a transaction is in the growing phase. The transaction acquires locks as needed. Once the transaction releases a lock, it enters the shrinking phase, and it can issue no more lock requests.

Example: T3: lock-X(B);
Read(B);
B=B-50;
Write(B);
Lock-X(A);
Read(A);
A=A+50;
Write(A);
Unlock(B); Unlock(A)

Advantages

- The two-phase locking protocol ensures conflict serializability.
- In any transaction the point in the schedule where the transaction has obtained its final lock is called the lock point of the transactions.

Disadvantages

- Two-phase locking does not ensure freedom from deadlock.
- Cascading rollback may occur under two-phase locking.

Types of two-phase locking:

- Strict two-phase locking.
- Rigorous two-phase locking.

Strict two-phase locking

Strict two-phase locking is a **locking method** used in concurrent systems.

Cascading schedule: In this schedule one transaction is dependent on another transaction. So if one has to rollback then the other has to rollback.

cascading rollbacks can be avoided by a modification of two-phase locking called **the strict two-phase locking protocol**.

This two-phase locking protocol requires not only that locking be two phases (Growing phase and Shrinking phase), but also that all exclusive-mode locks taken by a transaction be held until that transaction commits. This is called **strict two-phase locking protocol**.

The rules of **Strict 2PL are:**

- ✓ If a transaction T wants to read/write an object, it must request a shared/exclusive on the object.
- ✓ All the exclusive locks held by transaction T are released when T commits (and not before)

Rigorous two-phase locking

Another variant of two-phase locking is the **rigorous two-phase locking protocol**, which requires that all locks be held until the transaction commit.

If lock conversion is allowed then two type of conversion lock is available

- ✓ **Upgrading locks:** It provides a mechanism for conversion from shared lock (read lock) to exclusive lock (write lock) is known as upgrading lock.
- ✓ **Downgrading lock:** It provides a mechanism for conversion from exclusive lock (write lock) to shared lock (read lock) is known as downgrade.

8) Explain about Timestamp Ordering Algorithm for Concurrency Control:

A **timestamp** is a unique identifier created by the DBMS to identify a transaction. Typically, timestamp values are assigned in the order in which the transactions are submitted to the system, so a timestamp can be thought of as the *transaction start time*. We will refer to the timestamp of transaction *T* as **TS(*T*)**.

The Timestamp Ordering Algorithm for Concurrency Control:

A schedule in which the transactions participate is then serializable, and the *only equivalent serial schedule permitted* has the transactions in order of their timestamp values. This is called **timestamp ordering (TO).**

The algorithm associates with each database item ***X* two timestamp (TS) values**:

1. **read_TS(*X*).** The **read timestamp** of item *X* is the largest timestamp among all the timestamps of transactions that have successfully read item *X*—that is, read_TS(*X*) = TS(*T*), where *T* is the *youngest* transaction that has read *X* successfully.

2. **write_TS(*X*).** The **write timestamp** of item *X* is the largest of all the timestamps of transactions that have successfully written item *X*—that is, write_TS(*X*) = TS(*T*), where *T* is the *youngest* transaction that has written *X* successfully. Based on the algorithm, *T* will also be the last transaction to write item *X*.

- read_item(*X*) or a write_item(*X*) operation, the **basic TO** algorithm compares the timestamp of *T* with read_TS(*X*) and write_TS(*X*) to ensure that the timestamp order of transaction execution is not violated. If this order is violated, then transaction *T* is aborted and resubmitted to the system as a new transaction with a *new timestamp*.

- If *T* is aborted and rolled back, any transaction *T*1 that may have used a value written by *T* must also be rolled back. Similarly, any transaction *T*2 that may have used a value written by *T*1 must also be rolled back, and so on. This effect is known as **cascading rollback.**

The concurrency control algorithm must check whether conflicting operations violate the **timestamp ordering in the following two cases**:

1. Whenever a transaction *T* issues a write_item(*X*) operation, the following check is performed:

a. If read_TS(*X*) > TS(*T*) or if write_TS(*X*) > TS(*T*), then abort and roll back *T* and reject the operation. This should be done because some *younger* transaction with a timestamp greater than TS(*T*)—and hence *after T* in the timestamp ordering—has already read or written the value of item *X* before *T* had a chance to write *X*, thus violating the timestamp ordering.

b. If the condition in part (a) does not occur, then execute the write_item(*X*) operation of *T* and set write_TS(*X*) to TS(*T*).

2. Whenever a transaction *T* issues a read_item(*X*) operation, the following check is performed:

a. If write_TS(*X*) > TS(*T*), then abort and roll back *T* and reject the operation.This should be done because some younger transaction with timestamp greater than TS(*T*)—and hence *after T* in the timestamp ordering—has already written the value of item *X* before *T* had a chance to read *X*.

b. If write_TS(*X*) ≤ TS(*T*), then execute the read_item(*X*) operation of *T* and set read_TS(*X*) to the *larger* of TS(*T*) and the current read_TS(*X*).

Strict Timestamp Ordering (TO). A variation of basic TO called **strict TO** ensure that the schedules are both **strict** (for easy recoverability) and (conflict) serializable. In this variation, a transaction *T* issues a read_item(*X*) or write_item(*X*) such that TS(*T*) > write_TS(*X*) has its read or write operation *delayed* until the transaction *T* that *wrote* the value of *X* (hence TS(*T*) = write_TS(*X*)) has committed or aborted.

9. Explain about Multiversion Concurrency Control Techniques.

These protocols for concurrency control keep copies of the old values of a data item when the item is updated (written); they are known as **multiversion concurrency control** because several versions (values) of an item are kept by the system.

- When a transaction requests to read an item, the *appropriate* version is chosen to maintain the serializability of the currently executing schedule.
- One reason for keeping multiple versions is that some read operations that would be rejected in other techniques can still be accepted by reading an *older version* of the item to maintain serializability.
- When a transaction writes an item, it writes a *new version* and the old version(s) of the item is retained. Some multiversion concurrency control algorithms use the concept of view serializability rather than conflict serializability.

Multiversion Technique Based on Timestamp Ordering

In this method, several versions *X*1, *X*2, … , *Xk* of each data item *X* are maintained. For *each version*, the value of version *Xi* and the following two timestamps associated with version *Xi* are kept:

1.read_TS(*Xi*). The **read timestamp** of *Xi* is the largest of all the timestamps of transactions that have successfully read version *Xi*.

2. write_TS(*Xi*). The **write timestamp** of *Xi* is the timestamp of the transaction that wrote the value of version *Xi*.

Multiversion Two-Phase Locking Using Certify Locks

In this multiple-mode locking scheme, there are *three locking modes* for an item— read, write, and *certify*—instead of just the two modes (read, write).

- Hence, the state of LOCK(*X*) for an item *X* can be one of read-locked, write-locked, certify-locked, or unlocked. In the standard locking scheme, with only read and write locks a write lock is an exclusive lock. We can describe the relationship between read and write locks in the standard scheme is shown in fig 3.9.

(a)

	Read	Write
Read	Yes	No
Write	No	No

(b)

	Read	Write	Certify
Read	Yes	Yes	No
Write	Yes	No	No
Certify	No	No	No

Fig 3.9 Lock compatibility tables.
(a) Lock compatibility table for read/write locking scheme.
(b) Lock compatibility table for read/write/certify locking scheme.

Validation-Based (Optimistic) Concurrency Control:

In this scheme, updates in the transaction are *not* applied directly to the database items on disk until the transaction reaches its end and is *validated*.

- During transaction execution, all updates are applied to *local copies* of the data items that are kept for the transaction. At the end of transaction execution, a **validation phase** checks whether any of the transaction's updates violate serializability.

- Certain information needed by the validation phase must be kept by the system. If serializability is not violated, the transaction is committed and the database is updated from the local copies; otherwise, the transaction is aborted and then restarted later.

There are **three phases for this concurrency control protocol:**

1. **Read phase.** A transaction can read values of committed data items from the database. However, updates are applied only to local copies (versions) of the data items kept in the transaction workspace.

2. **Validation phase.** Checking is performed to ensure that serializability will not be violated if the transaction updates are applied to the database.

3. **Write phase.** If the validation phase is successful, the transaction updates are applied to the database; otherwise, the updates are discarded and the transaction is restarted.

The optimistic protocol we describe uses transaction timestamps and also requires that the write_sets and read_sets of the transactions be kept by the system. Additionally, *start* and *end* times for the three phases need to be kept for each transaction.

- The write_set of a transaction is the set of items it writes, and the read_set is the set of items it reads.
- In the validation phase for transaction *Ti*, the protocol checks that *Ti* does not interfere with any recently committed transactions or with any other concurrent transactions that have started their validation phase.

The validation phase for *Ti* checks that, for *each* such transaction *Tj* that is either recently committed or is in its validation phase, *one* of the following conditions holds:

1. Transaction *Tj* completes its write phase before *Ti* starts its read phase.
2. *Ti* starts its write phase after *Tj* completes its write phase, and the read_set of *Ti* has no items in common with the write_set of *Tj*.
3. Both the read_set and write_set of *Ti* have no items in common with the write_set of *Tj*, and *Tj* completes its read phase before *Ti* completes its read phase.

When validating transaction *Ti* against each one of the transactions *Tj*, the first condition is checked first since (1) is the simplest condition to check.

- Only if condition 1 is false is condition 2 checked, and only if (2) is false is condition 3—the most complex to evaluate—checked.
- If any one of these three conditions holds with each transaction *Tj*, there is no interference and *Ti* is validated successfully.
- If *none* of these three conditions holds for any one *Tj*, the validation of transaction *Ti* fails (because *Ti* and *Tj* may violate serializability) and so *Ti* is aborted and restarted later because interference with *Tj may* have occurred.

Concurrency Control Based on Snapshot Isolation:

The basic definition of **snapshot isolation** is that a transaction sees the data items that it reads based on the committed values of the items in the *database snapshot* (or database state) when the transaction starts.

- Snapshot isolation will ensure that the phantom record problem does not occur, since the database transaction, or, in some cases, the database statement, will only see the records that were committed in the database at the time the transaction started.

- Any insertions, deletions, or updates that occur after the transaction starts will not be seen by the transaction.
- In addition, snapshot isolation does not allow the problems of dirty read and non repeatable read to occur.
- In this scheme, read operations do not require read locks to be applied to the items, thus reducing the overhead associated with two-phase locking.
- However, write operations do require write locks. Thus, for transactions that have many reads, the performance is much better than 2PL.

Temporary version store (tempstore):

When writes do occur, the system will have to keep track of older versions of the updated items in a **temporary version store**, with the timestamps of when the version was created.

- This is necessary so that a transaction that started before the item was written can still read the value (version) of the item that was in the database snapshot when the transaction started.
- To keep track of versions, items that have been updated will have pointers to a list of recent versions of the item in the *tempstore,* so that the correct item can be read for each transaction.
- The tempstore items will be removed when no longer needed, so a method to decide when to remove unneeded versions will be needed.
- Variations of this method have been used in several commercial and open source DBMSs, including Oracle and PostGRES.
- Variations of snapshot isolation (SI) techniques, known as **serializable snapshot isolation (SSI)**, have been proposed and implemented in some of the DBMSs that use SI as their primary concurrency control method.

10. Explain about Granularity of Data Items and Multiple Granularity Locking.

All concurrency control techniques assume that the database is formed of a number of named data items. A database item could be chosen to be one of the following:

- A database record
- A field value of a database record
- A disk block
- A whole file
    - The whole database

.**Granularity Level Considerations for Locking**

The size of data items is often called the **data item granularity**. *Fine granularity* refers to small item sizes, whereas *coarse granularity* refers to large item sizes. Several tradeoffs must be considered in choosing the data item size.

- First, notice that the larger the data item size is, the lower the degree of concurrency permitted. For example, if the data item size is a disk block, a transaction *T* that needs to lock a single record *B* must lock the whole disk block *X* that contains *B* because a lock is associated with the whole data item (block).

- Now, if another transaction *S* wants to lock a different record *C* that happens to reside in the same disk block *X* in a conflicting lock mode, it is forced to wait.

- If the data item size was a single record instead of a disk block, transaction *S* would be able to proceed, because it would be locking a different data item (record).

- On the other hand, the smaller the data item size is, the more the number of items in the database. Because every item is associated with a lock, the system will have a larger number of active locks to be handled by the lock manager.

- More lock and unlock operations will be performed, causing a higher overhead. In addition, more storage space will be required for the lock table.

- For timestamps, storage is required for the read_TS and write_TS for each data item, and there will be similar overhead for handling a large number of items.

Multiple Granularity Level Locking

Since the best granularity size depends on the given transaction, it seems appropriate that a database system should support multiple levels of granularity, where the granularity level can be adjusted dynamically for various mixes of transactions.

Figure 3.10 shows a simple granularity hierarchy with a database containing two files, each file containing several disk pages, and each page containing several records.

- This can be used to illustrate a **multiple granularity level** 2PL protocol, with shared/exclusive locking modes, where a lock can be requested at any level.

- However, additional types of locks will be needed to support such a protocol efficiently.

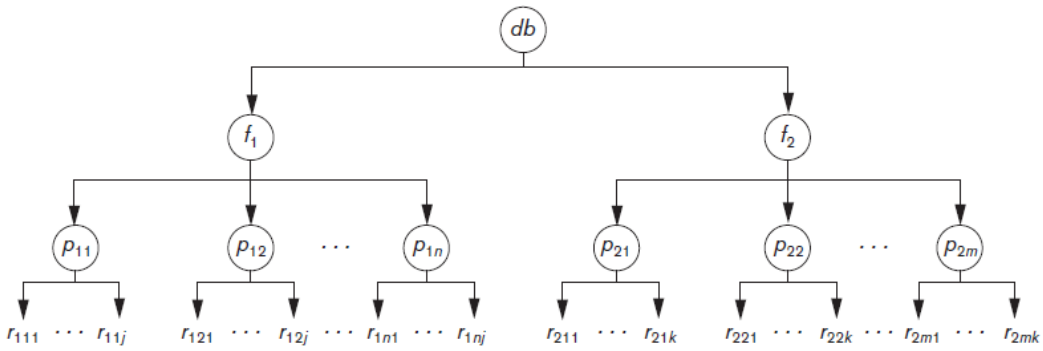


Fig.3.10. A granularity hierarchy for illustrating multiple granularity level locking.

Consider the following scenario, which refers to the example in Figure 3.10.

- Suppose transaction *T*1 wants to update *all the records* in file *f*1, and *T*1 requests and is granted an exclusive lock for *f*1. Then all of *f*1's pages (*p*11 through *p*1*n*)—and the records contained on those pages—are locked in exclusive mode.

- This is beneficial for *T*1 because setting a single file-level lock is more efficient than setting *n* pagelevel locks or having to lock each record individually.

- Now suppose another transaction *T*2 only wants to read record *r*1*nj* from page *p*1*n* of file *f*1; then *T*2 would request a shared record-level lock on *r*1*nj*.

- However, the database system (that is, the transaction manager or, more specifically, the lock manager) must verify the compatibility of the requested lock with already held locks.

- One way to verify this is to traverse the tree from the leaf *r*1*nj* to *p*1*n* to *f*1 to *db*. If at any time a conflicting lock is held on any of those items, then the lock request for *r*1*nj* is denied and *T*2 is blocked and must wait. This traversal would be fairly efficient.

- To make multiple granularity level locking practical, additional types of locks, called **intention locks**, are needed.

- The idea behind intention locks is for a transaction to indicate, along the path from the root to the desired node, what type of lock (shared or exclusive) it will require from one of the node's descendants.

There are three types of intention locks:

1. Intention-shared (IS) indicates that one or more shared locks will be requested on some descendant node(s).
2. Intention-exclusive (IX) indicates that one or more exclusive locks will be requested on some descendant node(s).

3. Shared-intention-exclusive (SIX) indicates that the current node is locked in shared mode but that one or more exclusive locks will be requested on some descendant node(s)

	IS	IX	S	SIX	X
IS	Yes	Yes	Yes	Yes	No
IX	Yes	Yes	No	No	No
S	Yes	No	Yes	No	No
SIX	Yes	No	No	No	No
X	No	No	No	No	No

Fig 3.11 Lock compatibility matrix for multiple granularity locking

The compatibility table of the three intention locks, and the actual shared and exclusive locks, is shown in Figure 3.11.

The **multiple granularity locking (MGL)** protocol consists of the following rules:

1. The lock compatibility must be adhered to.
2. The root of the tree must be locked first, in any mode.
3. A node *N* can be locked by a transaction *T* in S or IS mode only if the parent node *N* is already locked by transaction *T* in either IS or IX mode.
4. A node *N* can be locked by a transaction *T* in X, IX, or SIX mode only if the parent of node *N* is already locked by transaction *T* in either IX or SIX mode.
5. A transaction *T* can lock a node only if it has not unlocked any node (to enforce the 2PL protocol).
6. A transaction *T* can unlock a node, *N*, only if none of the children of node *N* are currently locked by *T*.

- Rule 1 simply states that conflicting locks cannot be granted. Rules 2, 3, and 4 state the conditions when a transaction may lock a given node in any of the lock modes.
- Rules 5 and 6 of the MGL protocol enforce 2PL rules to produce serializable schedules.

Basically, the locking *starts from the root* and goes down the tree until the node that needs to be locked is encountered, whereas unlocking *starts from the locked node* and goes up the tree until the root itself is unlocked.

consider the following **three transactions**:

1. *T*1 wants to update record *r*111 and record *r*211.
2. *T*2 wants to update all records on page *p*12.
3. *T*3 wants to read record *r*11*j* and the entire *f*2 file.

T_1	T_2	T_3
IX(*db*)		
IX(f_1)		
	IX(*db*)	
		IS(*db*)
		IS(f_1)
		IS(p_{11})
IX(p_{11})		
X(r_{111})		
	IX(f_1)	
	X(p_{12})	
		S(r_{11j})
IX(f_2)		
IX(p_{21})		
X(p_{211})		
unlock(r_{211})		
unlock(p_{21})		
unlock(f_2)		
		S(f_2)
	unlock(p_{12})	
	unlock(f_1)	
	unlock(*db*)	
unlock(r_{111})		
unlock(p_{11})		
unlock(f_1)		
unlock(*db*)		
		unlock(r_{11j})
		unlock(p_{11})
		unlock(f_1)
		unlock(f_2)
		unlock(*db*)

Fig.3.12 Lock operations to illustrate a serializable schedule.

Figure 3.12 shows a possible serializable schedule for these three transactions. Only the lock and unlock operations are shown.

- The notation <lock_type>(<item>) is used to display the locking operations in the schedule. The multiple granularity level protocol is especially suited when processing a mix of transactions that include (1) short transactions that access only a few items (records or fields) and (2) long transactions that access entire files.

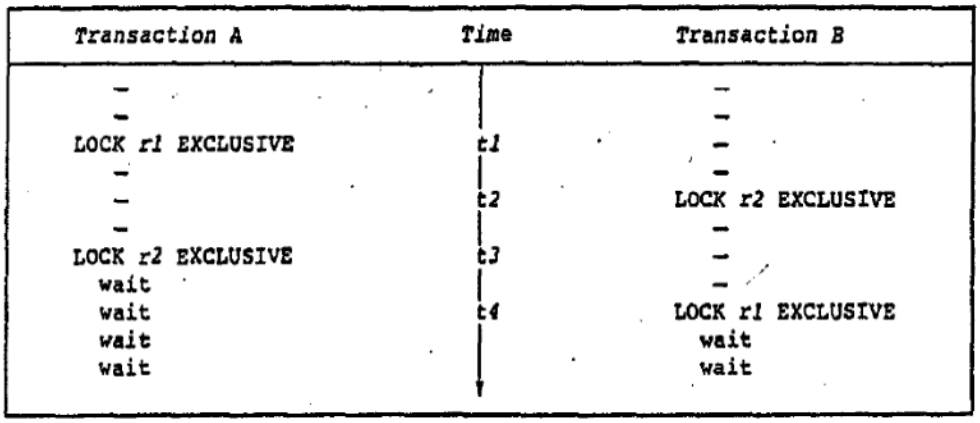
- In this environment, less transaction blocking and less locking overhead are incurred by such a protocol when compared to a single-level granularity locking approach.

11.Explain about deadlock, deadlock prevention, detection and recovery. (Nov/Dec 2020 & Apr/May 2021)

Deadlock

Deadlock is a situation in which two or more transactions are in a simultaneous wait state, each of them waiting for one of the others to release a lock before it can proceed.

- A system is in a deadlock state if there exists a set of transactions such that **every transaction in the set is waiting for another transaction in the set.**
- Here exists a set of waiting transactions {T0, T1.....Tn} such that T0 is **waiting for a data item** that T1 **holds** and **T1 is waiting for a data item** and that T2 **holds** and Tn-1 is waiting for a data item that Tn holds and Tn is waiting for a data item that T0 holds.
- The below figure 3.13 shows a deadlock involving two transactions, but deadlocks involving three, four or more transactions are also possible.
- However, the deadlock never does involve more than two transactions in practice.



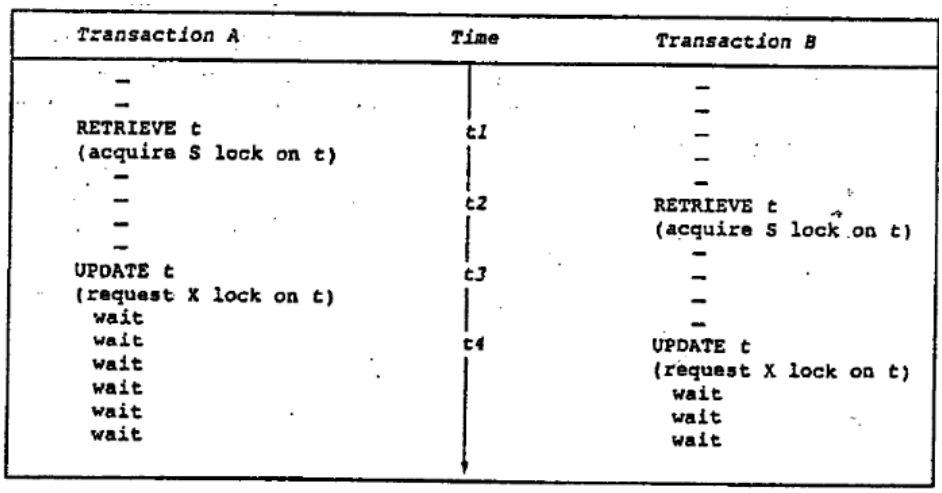


Fig 3.13 An example of deadlock

If a deadlock occurs, the system detects it and breaks it. Breaking the deadlock involves choosing the deadlocked transactions as the victim (injured) and rolling it back, thereby releasing its locks and allowing some other transaction to proceed.

The victim has failed and been rolled back through no fault of its own. Some systems automatically restart such a transaction from beginning.

Deadlock Handling:

There are three general techniques for handling deadlock:

- ✓ Timeouts
- ✓ Deadlock prevention
- ✓ Deadlock detection and recovery

Timeouts:

A transaction that requests a lock will wait for only a system defined period of time. If the lock has not been granted within this period, the lock request times out. In this case, the DBMS assumes the transaction may be deadlocked, even though it may not be, and it aborts and automatically restarts the transaction.

Deadlock prevention:

Two approaches for deadlock prevention

1) Ensures that **no cyclic waits can occur** by ordering the requests for locks, or requiring all locks to be acquired together.
2) Performs **transaction rollback** instead of waiting for a lock

Scheme 1:

1. Requires **each transaction locks all its data items** before it begins execution.
2. Either all are locked in one step or none are locked.

Disadvantage of Scheme 1:

1. It is often **hard to predict**, before the transaction begins, what data items need to be locked.
2. Data item utilization may be very low, since many of the data items may be locked but **unused for a long time.**

Scheme 2:

Impose an **ordering of all data items** and to require that a transaction lock data items only in a **sequence consistent** with the ordering.

- The approach to prevent deadlocks is to use **preemption and transaction rollbacks.**
- In preemption, when a transaction T2 requests a lock that transaction T1 holds, the lock granted to T1 may be preempted by rolling back of T1 and granting of lock to T2. We assign a unique timestamp to each transaction.

Two different deadlock prevention schemes using timestamps include

- Wait-die
- Wound-wait

Wait-die:

It is **non-preemptive technique**.

When a transaction Ti requests a data item currently held by Tj , Ti is allowed to wait only if it has a timestamp **smaller than** that of tj (ie Ti is **older than**Tj). Otherwise, Ti is rolled back.

Example:

- Suppose that transaction T22, T23, T24 have timestamps 5, 10 and 15 respectively.
- If T22 requests a data item held by t23, then T22 will wait.
- If T24 requests a data item held by T23, then T24 will be rolled back.

Wound-wait:

- It is a **preemptive technique.**
- When transaction Ti requests a data item currently held by Tj, Ti is allowed to wait only if it has time a time stamp **larger than** that of Tj (ie) (Ti is younger than Tj). Otherwise **Tj is rolled back.**(Tj is wounded by TI)

Example:

- If T22 requests a data item held T23, then data item will be preempted from T23 and T23 will be rolled back.
- If T24 requests data item held by T23, then T23 will wait.

Table 3.1 shows the Difference between Wait -Die and Wound - Wait

Wait-die	Wound-wait
An older transaction must wait for a younger one to release its data item.	An older transaction never waits for a younger transaction.
If a transaction Ti disk is rolled back then Ti may reuse the same sequence of requests when it is restarted	When Ti is restarted and requests the data now being held by Tj, Ti waits. There may be fewer roll backs.

Table 3.1-Difference between Wait -Die and Wound - Wait

Deadlock detection and Recovery:

To recover from the deadlock, the system must

- **Maintain information** about the **current allocation of data items** to transactions.
- **Provide an algorithm** that uses this information to **determine** whether the system has entered to a deadlock state.
- **Recover from deadlock** when the detection algorithm determines that a deadlock exists.

Deadlock detection:

- It can be described precisely in terms of a directed graph called a **wait for graph**.
- Graph consists of a pair G (V, E) where V is a set of vertices and E is set of edges. Each element in set E of edges is an ordered pair Ti->Tj.
- If Ti->Tj is in E, then there is a directed edge from transaction Ti to implying that transaction Ti is waiting for transaction Tj to release a data item that it needs.
- When transaction Ti requests a data item currently being held by transaction Tj, then edge Ti->Tj is inserted in the wait for graph.
- This edge is removed only when transaction Tj is no longer holding a data item needed by transaction Ti.
- A deadlock exists in the system if and only if the wait-for graph contains a cycle.
- Each transaction involved in the cycle is said to be deadlocked.

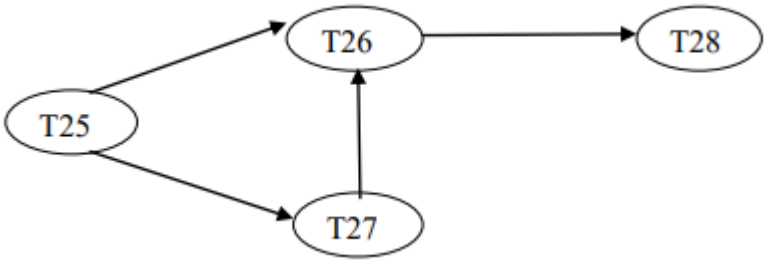


Fig. 3.14 – Transactions without deadlock

- Transaction T25 is waiting for transaction T26 and T27.
- Transaction T27 is waiting for T26.
- Transaction T26 is waiting for T25.

Since the graph has no cycle, the system is not in deadlock state is shown in above fig 3.14.

Suppose the transaction T28 is requesting an item held by T24.

- The edge T28->T27 is added to wait for graph, resulting in new system state.
- The graph contains the cycle. T26 T28 T27 T26
- Simplifying that transactions T26, T27, T28 are all deadlocked is shown in fig 3.15.

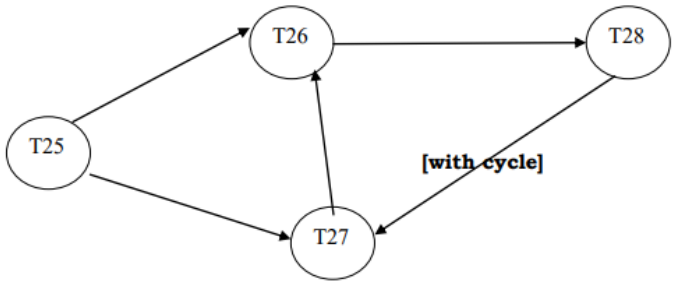


Fig. 3.15 – Transactions with deadlock

If deadlock occur frequently, then the detection algorithm should be invoked more frequently than usual.

2. Deadlock recovery:

When **a detection algorithm determines that a deadlock** exists, the system must **recover from deadlock.**

Actions need to be taken:

• **Select a victim:**

- Given a set of deadlocked transaction determine which transaction to roll back to break the deadlock.
- We should roll back the transactions that will incur the maximum cost.

Many factors determine the **cost of rollback**.

- Transaction execution time.
- No. of data items used by the transaction.
- More no. of data items needed by the transaction to complete.
- No. of transactions involved in rollback.

Starvation: If the transaction **never completes its designated task**, thus there is Starvation.

12. Explain in detail about recovery and its types. (Nov/Dec 2019)

RECOVERY:

DBMS is responsible for making sure that either

- **All operations in transaction** are **completed successfully** and the changes are recorded permanently in the database.
- The DBMS **must not permit** some operations of a transaction T to be applied to the DB while others of T are not.

Failures Type Generally classified as Transaction Failure, System Failure and Media Failure

TRANSACTION RECOVERY:

Reasons for a transaction fail in the middle of execution:

A **computer failure** (System crash) – media failures

A **transaction or system error**: logical program error

- ✓ Some operation in transaction may cause it to fail, such as integer overflow or divide by zero.
- ✓ May occur because of erroneous parameter values
- ✓ Logical programming
- ✓ User may interrupt during execution

Load error or exception conditions detected by the transaction :no data for the transaction

- ✓ During transaction execution, conditions may occur that necessitate cancellation of the transaction
- ✓ Ex. Data for the transaction may not found
- ✓ Exception should be programmed (not be consider failure)

Concurrency control enforcement: by concurrency control method

- ✓ The concurrency control method may decide to abort the transaction,
- ✓ Because of violent serializability
- ✓ Because several transactions are in state of deadlock

Disk failure, physical problems and catastrophe: ex. Power failure, fire, overwrite disk

Transaction states and additional Operation:

For **recovery purpose**, the system needs to keep track of when the transaction

- ✓ starts,
- ✓ terminates and
- ✓ commits or aborts.

The **recovery manager** keeps track of the followings

- ✓ **Begin_transaction**: mark the beginning of transaction execute
- ✓ **Read or write**: specified operations on the database item that executes as part of transaction
- ✓ **End_transaction**: specifies that operations have ended and marks the end of execution (Necessary to check)
  - The change can be **committed**
  - Or whether the transaction has to **aborted**
- ✓ **Commit_Transaction:** successful end (will not undo)
- ✓ **Rollback:** unsuccessful end (undone)

- State of Transaction
  - ✓ **Active**, the initial state; the transaction stays in this state while it is executing.
  - ✓ **Partially committed**, after the final statement has been executed
  - ✓ **Failed**, after the discovery that normal execution can no longer proceed.
  - ✓ **Aborted**, after the transaction has been rolled backed and the database has been restored to its state prior to the start of transaction.
  - ✓ **Committed**, after successful completion is shown in fig 3.16.

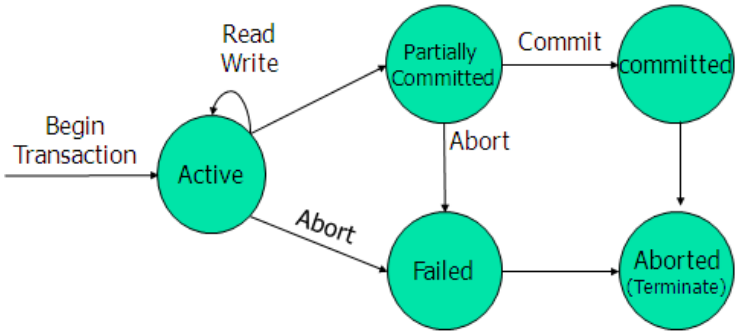


Fig 3.16 State of a transaction

1. The transaction has **committed** only if it has entered the **committed state**.
2. The transaction has **aborted** only if it has entered the **aborted state**.
3. The transaction is said to have **terminated** if has either **committed or aborted**.

The System Log:

- ✓ The system maintain log by keep track of all transactions that affect the database.
- ✓ Log is kept on Disk.
- ✓ Effected only by disk or catastrophic failure
- ✓ Keep Log records

Log records:

T is transaction ID

- ✓ (Start_transaction, T) start transaction
- ✓ (Write_item, T, X, old_value, new_value) transaction write/update item x
- ✓ (Read_item, T, X) transaction read item X
- ✓ (Commit, T) complete operation of T
- ✓ (Abort, T) terminate transaction T

- ✓ Log file keep track
- ✓ System fail occur
- ✓ Restart system, system will recovery
  - Redo transaction that already commit
  - Undo no commit transaction

SYSTEM RECOVERY:

- The system must be prepared to recover from **local failures** such as overflow exception and also from **global failure** such as power outage.
- A local failure affects only the transaction which the failure has occurred, but global failure affects all the transactions in progress at the time of the failure.

The global failure falls into two broad categories.

- **System failures (e.g. power outage):**which affect all transactions currently in progress but do not physically damage the database. A system failure is sometimes called a soft crash.
- **Media failures (e.g., head crash on the disk),** which do cause damage to the database or some portion; affect at least those transactions currently using that portion. A media failure is sometimes called hard crash.

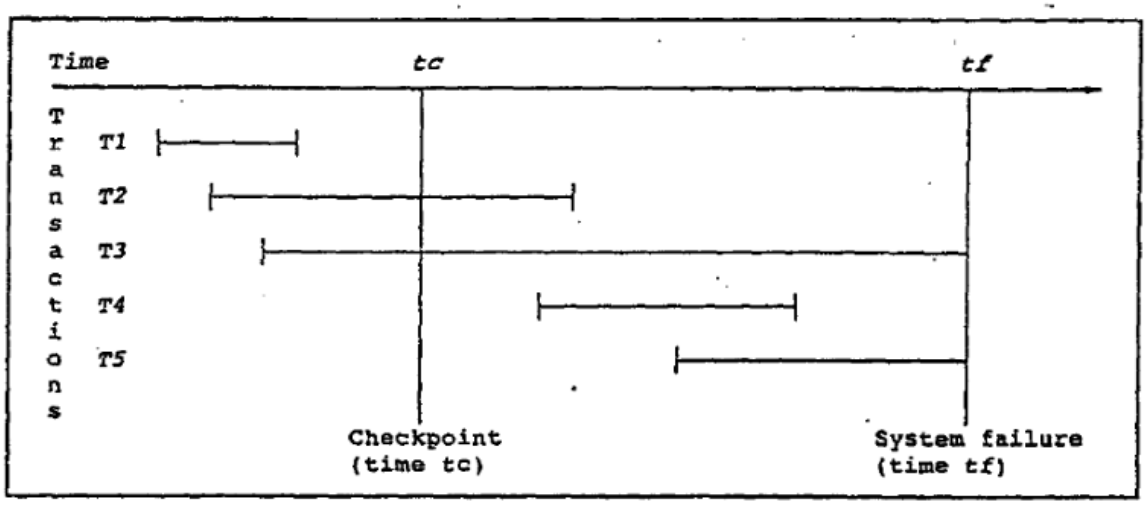


Fig 3.17 Five transaction categories

- The key point regarding system failure is that the contents of main memory are lost.
- The particular state of any transaction that was in progress at the time of the failure is therefore no longer known;

- A transaction not successfully completed and so much is undone – i.e., rolled back – when the system restarts.
- A transaction successfully completed and so much is redo certain transactions at restart time.
- Example: The Five transaction is shown in fig 3.17.
  - Transactions of T1 completed prior to time tc.
  - Transaction of T2 started prior to time tc and completed after time tc and before tf.
  - Transaction of T3 also started prior to time tc but did not complete by time tf.
  - Transaction of T4 started after time tc and completed before tf.
  - Finally transaction of T5 also started after time tc but did not complete by time tf.

It should be clear that when the system is restarted. Transactions of types T3 and T5 must be undone, and transactions of types T2 and T4 must be redone. Transactions T1 do not enter into restart process at all because their updates were forced to the database at time tc as part of the did point process.

At restart time, the system first goes through the following procedure

1. Start with two lists of transactions, the undo and Redo list
2. Set the undo list equal to the list of all transactions given in the most the checkpoint and redo list to empty
3. If BEGIN TRANSACTION log record is found for transaction T,addT the UNDO list.
4. If a COMMIT log record is found for transaction T, move T from the list to the REDO.
5. When the end of the log is reached, the undo and redo list identify. Transaction T3 and T5 must be undo. Transaction T2 and T4 are redo

- The Restoring database to correct state by redoing work is sometimes called forward recovery.
- The restoring database to correct state by undoing work is sometimes called backward recovery.

Finally, all such recovery is complete, and then the system is ready to accept new work.

MEDIA RECOVERY:

A media failure is a failure such as a disk head crash or a disk controller failure, in which some portion of the database has been physically destroyed.

- Recovery from such a failure basically involves reloading the database from a backup copy and then using the log to redo all transactions that completed since that backup copy was taken.
- There is no need to undo transactions that were still in progress at the time of the failure, since by definition all updates of such transactions have been undone anyway.

13. Explain about Recovery Techniques Based on Immediate Update.

When a transaction issues an update command, the database on disk can be updated *immediately*, without any need to wait for the transaction to reach its commit point. Notice that it is *not a requirement* that every update be applied immediately to disk; it is just possible that some updates are applied to disk *before the transaction commits*.

(a)

T_1
read_item(*A*)
read_item(*D*)
write_item(*D*)

T_2
read_item(*B*)
write_item(*B*)
read_item(*D*)
write_item(*D*)

T_3
read_item(*A*)
write_item(*A*)
read_item(*C*)
write_item(*C*)

T_4
read_item(*B*)
write_item(*B*)
read_item(*A*)
write_item(*A*)

(b)

[start_transaction,T_1]
[write_item, T_1, *D*, 20]
[commit, T_1]
[checkpoint]
[start_transaction, T_4]
[write_item, T_4, *B*, 15]
[write_item, T_4, *A*, 20]
[commit, T_4]
[start_transaction, T_2]
[write_item, T_2, *B*, 12]
[start_transaction, T_3]
[write_item, T_3, *A*, 30]
[write_item,T_2, *D*, 25] ← System crash

T_2 and T_3 are ignored because they did not reach their commit points.

T_4 is redone because its commit point is after the last system checkpoint.

Figure 3.18 An example of recovery using deferred update with concurrent transactions.
(a) The READ and WRITE operations of four transactions.
(b) System log at the point of crash.

- Provisions must be made for *undoing* the effect of update operations that have been applied to the database by a *failed transaction.* This is accomplished by rolling back the transaction and undoing the effect of the transaction's write_item operations is shown in fig 3.18.

- Therefore, the **UNDO-type log entries**, which include the **old value** of the item, must be stored in the log. Because UNDO can be needed during recovery, these methods follow a **steal strategy** for deciding when updated main memory buffers can be written back to disk.

Theoretically, we can distinguish **two main categories of immediate update algorithms**:

1. If the recovery technique ensures that all updates of a transaction are recorded in the database on disk *before the transaction commits*, there is never a need to REDO any operations of committed transactions. This is called the **UNDO/NO-REDO recovery algorithm**.

In this method, all updates by a transaction must be recorded on disk *before the transaction commits*, so that REDO is never needed. Hence, this method must utilize the **steal/force strategy** for deciding when updated main memory buffers are written back to disk.

2. If the transaction is allowed to commit before all its changes are written to the database, we have the most general case, known as the **UNDO/REDO recovery algorithm**. In this case, the **steal/no-force strategy** is applied.

This is also the most complex technique, but the most commonly used in practice.

- When concurrent execution is permitted, the recovery process again depends on the protocols used for concurrency control.

- The procedure RIU_M (Recovery using Immediate Updates for a Multiuser environment) outlines a recovery algorithm for concurrent transactions with immediate update (UNDO/REDO recovery).

- Assume that the log includes checkpoints and that the concurrency control protocol produces *strict schedules*—as, for example, the strict two-phase locking protocol does.

- That a strict schedule does not allow a transaction to read or write an item unless the transaction that wrote the item has committed. However, deadlocks can occur in strict two-phase locking, thus requiring abort and UNDO of transactions.

Procedure RIU_M (UNDO/REDO with checkpoints).

1. Use two lists of transactions maintained by the system: the committed transactions since the last checkpoint and the active transactions.
2. Undo all the write_item operations of the *active* (uncommitted) transactions, using the UNDO procedure. The operations should be undone in the reverse of the order in which they were written into the log.
3. Redo all the write_item operations of the *committed* transactions from the log, in the order in which they were written into the log, using the REDO procedure defined earlier.

The UNDO procedure is defined as follows:

Procedure UNDO (WRITE_OP).

- Undoing a write_item operation write_op consists of examining its log entry [write_item, *T*, *X*, old_value, new_value] and setting the value of item *X* in the database to old_value, which is the before image (BFIM).

- Undoing a number of write_item operations from one or more transactions from the log must proceed in the *reverse order* from the order in which the operations were written in the log.

In NO-UNDO/REDO procedure, step 3 is more efficiently done by starting from the *end of the log* and redoing only *the last update of each item X.*Whenever an item is redone, it is added to a list of redone items and is not redone again.

14) Explain about Shadow Paging.

Shadow paging considers the database to be made up of a number of fixed size disk pages (or disk blocks)—say, *n*—for recovery purposes. A **directory** with *n* entries is constructed, where the *i*th entry points to the *i*th database page on disk.

- The directory is kept in main memory if it is not too large, and all references—reads or writes—to database pages on disk go through it.

- When a transaction begins executing, the **current directory**—whose entries point to the most recent or current database pages on disk—is copied into a **shadow directory**.

- The shadow directory is then saved on disk while the current directory is used by the transaction. During transaction execution, the shadow directory is *never* modified.

- When a write_item operation is performed, a new copy of the modified database page is created, but the old copy of that page is *not overwritten.* Instead, the new page is written elsewhere on some previously unused disk block.

- The current directory entry is modified to point to the new disk block, whereas the shadow directory is not modified and continues to point to the old unmodified disk block.

Figure 3.19 illustrates the concepts of shadow and current directories. For pages updated by the transaction, two versions are kept. The old version is referenced by the shadow directory and the new version by the current directory.

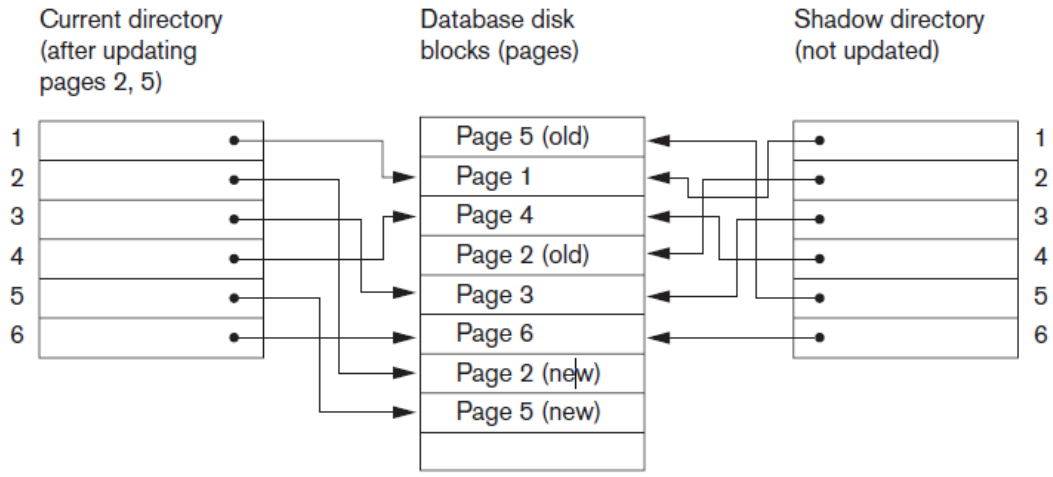


Fig 3.19 An example of shadow paging

The modified database pages are discarded from the current directory. The state of the database before transaction execution is available through the shadow directory, and that state is recovered by reinstating the shadow directory.

- The database thus is returned to its state prior to the transaction that was executing when the crash occurred, and any modified pages are discarded. Committing a transaction corresponds
- to discarding the previous shadow directory.

- Since recovery involves neither undoing nor redoing data items, this technique can be categorized as a NO-UNDO/NO-REDO technique for recovery.

- In a multiuser environment with concurrent transactions, logs and checkpoints must be incorporated into the shadow paging technique. One disadvantage of shadow paging is that the updated database pages change location on disk.

- This makes it difficult to keep related database pages close together on disk without complex storage management strategies. Furthermore, if the directory is large, the overhead of writing shadow directories to disk as transactions commit is significant. A further complication is how to handle **garbage collection** when a transaction commits.

- The old pages referenced by the shadow directory that have been updated must be released and added to a list of free pages for future use.

- These pages are no longer needed after the transaction commits. Another issue is that the operation to migrate between current and shadow directories must be implemented as an atomic operation.

15) Explain about The ARIES Recovery Algorithm.

The ARIES algorithm as an example of a recovery algorithm used in database systems. It is used in many relational database-related products of IBM.

ARIES uses a steal/no-force approach for writing, and it is based on three concepts:

1. The write-ahead logging, repeating history during redo, and logging changes during undo.
2. The second concept, **repeating history**, means that ARIES will retrace all actions of the database system prior to the crash to reconstruct the database state *when the crash occurred.* Transactions that were uncommitted at the time of the crash (active transactions) are undone.
3.The third concept, **logging during undo**, will prevent ARIES from repeating the completed undo operations if a failure occurs during recovery, which causes a restart of the recovery process.

The ARIES recovery procedure consists of three main steps:

- Analysis
- REDO and
- UNDO.

The **Analysis step** identifies the dirty (updated) pages in the buffer and the set of transactions active at the time of the crash. The appropriate point in the log where the REDO operation should start is also determined.

The **REDO phase** actually reapplies updates from the log to the database. Generally, the REDO operation is applied only to committed transactions. However, this is not the case in ARIES.
The actual buffers may be lost during a crash, since they are in main memory. Additional tables stored in the log during checkpointing (Dirty Page Table, Transaction Table) allow ARIES to identify this information.

Finally, during the **UNDO phase**, the log is scanned backward and the operations of transactions that were active at the time of the crash are undone in reverse order. The information needed for ARIES to accomplish its recovery procedure includes the log, the Transaction Table, and the Dirty Page Table. Additionally, checkpointing is used. These tables are maintained by the transaction manager and written to the log during checkpointing.

- In ARIES, every log record has an associated **log sequence number (LSN)** that is monotonically increasing and indicates the address of the log record on disk. Each LSN corresponds to a *specific change* (action) of some transaction.

A log record is written for any of the following actions:

- Updating a page (write), committing a transaction (commit), aborting a transaction (abort), undoing an update (undo), and ending a transaction (end).

In addition to the log, two tables are needed for efficient recovery:
The **Transaction Table** and the **Dirty Page Table**, which are maintained by the transaction manager.

- The **Transaction Table** contains an entry for *each active transaction,* with information such as the transaction ID, transaction status, and the LSN of the most recent log record for the transaction.
- The **Dirty Page Table** contains an entry for each dirty page in the DBMS cache, which includes the page ID and the LSN corresponding to the earliest update to that page.

Checkpointing in ARIES consists of the following:

Writing a begin_checkpoint record to the log, writing an end_checkpoint record to the log, and writing the LSN of the begin_checkpoint record to a special file.

- This special file is accessed during recovery to locate the last checkpoint information.
- With the end_checkpoint record, the contents of both the Transaction Table and Dirty Page Table are appended to the end of the log.

(a)

Lsn	Last_lsn	Tran_id	Type	Page_id	Other_information
1	0	T_1	update	C	...
2	0	T_2	update	B	...
3	1	T_1	commit		...
4	begin checkpoint				
5	end checkpoint				
6	0	T_3	update	A	...
7	2	T_2	update	C	...
8	7	T_2	commit		...

(b)

TRANSACTION TABLE

Transaction_id	Last_lsn	Status
T_1	3	commit
T_2	2	in progress

DIRTY PAGE TABLE

Page_id	Lsn
C	1
B	2

(c)

TRANSACTION TABLE

Transaction_id	Last_lsn	Status
T_1	3	commit
T_2	8	commit
T_3	6	in progress

DIRTY PAGE TABLE

Page_id	Lsn
C	7
B	2
A	6

Figure 3.20 An example of recovery in ARIES.

(a) The log at point of crash.

(b) The Transaction and Dirty Page Tables at time of checkpoint.

(c) The Transaction and Dirty Page Tables after the analysis phase.

Consider the recovery example shown in Figure 3.20. **There are three transactions**:

- *T*1, *T*2, and *T*3. *T*1 updates page *C*, *T*2 updates pages *B* and *C*, and *T*3 updates page A. Figure 3.20(a) shows the partial contents of the log, and Figure 3.20(b) shows the contents of the Transaction Table and Dirty Page Table.

- Now, suppose that a crash occurs at this point. Since a checkpoint has occurred, the address of the associated begin_checkpoint record is retrieved, which is location 4. The analysis phase starts from location 4 until it reaches the end.

- The end_checkpoint record contains the Transaction Table and Dirty Page Table in Figure 3.20(b), and the analysis phase will further reconstruct these tables. When the analysis phase encounters log record 6,a new entry for transaction *T*3 is made in the Transaction Table and a new entry for page A is made in the Dirty Page Table.

- After log record 8 is analyzed, the status of transaction *T*2 is changed to committed in the Transaction Table.

- Figure 3.20(c) shows the two tables after the analysis phase.

16. Explain about Transaction Support in SQL.

A single SQL statement is always considered to be atomic—either it completes execution without an error or it fails and leaves the database unchanged.

- With SQL, there is no explicit Begin_Transaction statement. Transaction initiation is done implicitly when particular SQL statements are encountered.
- However, every transaction must have an explicit end statement, which is either a COMMIT or a ROLLBACK.
- Every transaction has certain characteristics attributed to it. These characteristics are specified by a SET TRANSACTION statement in SQL.

The characteristics are the ***access mode*, the *diagnostic area size*, and the *isolation level***.

- The **access mode** can be specified as READ ONLY or READ WRITE. The default is READ WRITE, unless the isolation level of READ UNCOMMITTED is specified in which case READ ONLY is assumed. A mode of READ WRITE allows select, update, insert, delete, and create commands to be executed. A mode of READ ONLY, as the name implies, is simply for data retrieval.

- The **diagnostic area size** option, DIAGNOSTIC SIZE *n*, specifies an integer value *n*, which indicates the number of conditions that can be held simultaneously in the diagnostic area. These conditions supply feedback information (errors or exceptions) to the user or program on the *n* most recently executed SQL statement.

- The **isolation level** option is specified using the statement ISOLATION LEVEL <isolation>, where the value for <isolation> can be READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, or SERIALIZABLE.15 The default isolation level is SERIALIZABLE, although some systems use READ COMMITTED as their default. The use of the term SERIALIZABLE here is based on not allowing violations that cause **dirty read, unrepeatable read, and phantoms.**

IN SERIALIZABLE, then one or more of the following three violations may occur:

1. **Dirty read.** A transaction *T*1 may read the update of a transaction *T*2, which has not yet committed. If *T*2 fails and is aborted, then *T*1 would have read a value that does not exist and is incorrect.
2. **Nonrepeatable read.** A transaction *T*1 may read a given value from a table. If another transaction *T*2 later updates that value and *T*1 reads that value again, *T*1 will see a different value.
3. **Phantoms.** A transaction *T*1 may read a set of rows from a table, perhaps based on some condition specified in the SQL WHERE-clause. Now suppose that a transaction *T*2 inserts a new row *r* that also satisfies the WHERE-clause condition used in *T*1, into the table used by *T*1.

- The record *r* is called a **phantom record** because it was not there when *T*1 starts but is there when *T*1 ends. *T*1 may or may not see the phantom, a row that previously did not exist. If the equivalent serial order is *T*1 followed by *T*2, then the record *r* should not be seen;

- But if it is *T*2 followed by *T*1, then the phantom record should be in the result given to *T*1. If the system cannot ensure the correct behavior, then it does not deal with the phantom record problem.

	Type of Violation		
Isolation Level	Dirty Read	Nonrepeatable Read	Phantom
READ UNCOMMITTED	Yes	Yes	Yes
READ COMMITTED	No	Yes	Yes
REPEATABLE READ	No	No	Yes
SERIALIZABLE	No	No	No

Fig 3.21. Possible Violations Based on Isolation Levels as Defined in SQL

Fig 3.21 summarizes the possible violations for the different isolation levels. An entry of *Yes* indicates that a violation is possible and an entry of *No* indicates that it is not possible. READ UNCOMMITTED is the most forgiving, and SERIALIZABLE is the most restrictive in that it avoids all three of the problems mentioned above.

A sample SQL transaction might look like the following:

```
EXEC SQL WHENEVER SQLERROR GOTO UNDO;
EXEC SQL SET TRANSACTION
READ WRITE
DIAGNOSTIC SIZE 5
ISOLATION LEVEL SERIALIZABLE;
EXEC SQL INSERT INTO EMPLOYEE (Fname, Lname, Ssn, Dno, Salary)
VALUES ('Robert', 'Smith', '991004321', 2, 35000);
EXEC SQL UPDATE EMPLOYEE
SET Salary = Salary * 1.1 WHERE Dno = 2;
EXEC SQL COMMIT;
GOTO THE_END;
UNDO: EXEC SQL ROLLBACK;
THE_END: ... ;
```

The above transaction consists of first inserting a new row in the EMPLOYEE table and then updating the salary of all employees who work in department 2. If an error occurs on any of the SQL statements, the entire transaction is rolled back. This implies that any updated salary (by this transaction) would be restored to its previous value and that the newly inserted row would be removed.

MAILAM ENGINEERING COLLEGE

MAILAM (PO), Villupuram (Dt.) Pin: 604 304

(Approved by AICTE New Delhi, Affiliated to Anna University Chennai & Accredited by National Board of Accreditation (NBA) New Delhi)

DEPARTMENT OF INFORMATION TECHNOLOGY

STAFF NAME: Mrs.C.RAMYA, AP/ IT, Ms.M.SUBATHRA, AP/IT
CLASS / SEM: II / IV
SUBJECT CODE / NAME: CS3492 / DATABASE MANAGEMENT SYSTEMS

SYLLABUS

UNIT III - TRANSACTIONS

Transaction Concepts – ACID Properties – Schedules – Serialisability – Transaction support in SQL – Need for Concurrency – Concurrency control –Two Phase Locking- Timestamp – Multi version – Validation and Snapshot isolation– Multiple Granularity locking – Deadlock Handling – Recovery Concepts – Recovery based on deferred and immediate update – Shadow paging – ARIES Algorithm

UNIT IV - IMPLEMENTATION TECHNIQUES

RAID – File Organization – Organization of Records in Files – Data dictionary Storage – Column Oriented Storage– Indexing and Hashing –Ordered Indices – B+ tree Index Files – B tree Index Files – Static Hashing – Dynamic Hashing – Query Processing Overview – Algorithms for Selection, Sorting and join operations – Query optimization using Heuristics - Cost Estimation.

UNIT V - ADVANCED TOPICS

Distributed Databases: Architecture, Data Storage, Transaction Processing, Query processing and optimization – NOSQL Databases: Introduction – CAP Theorem – Document Based systems – Key value Stores – Column Based Systems – Graph Databases. Database Security: Security issues – Access control based on privileges – Role Based access control – SQL Injection – Statistical Database security – Flow control – Encryption and Public Key infrastructures – Challenges

TEXT BOOKS:

1. Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Seventh Edition, McGraw Hill, 2020.
2. Ramez Elmasri, Shamkant B. Navathe, "Fundamentals of Database Systems", Seventh Edition, Pearson Education, 2017

REFERENCES:

1. C.J.Date, A.Kannan, S.Swamynathan, "An Introduction to Database Systems", Eighth Edition, Pearson Education, 2006.

PREPARED BY

Mrs. C.Ramya, AP/IT
Ms.M.Subathra, AP/IT

II Batch Co-Ordinator
Mrs. G.Vasanthi, AP/IT

VERIFIED BY
Dr.S.Kalaivany,
Prof & HOD/IT

PRINCIPAL

UNIT IV

IMPLEMENTATION TECHNIQUES : RAID - File Organization - Organization of Records in Files-Data Dictionary Storage-Column Oriented Storage - 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.

2-Marks

1. What are the advantages and disadvantages of indexed sequential file? MAY-2011

The **advantage** of ordering records in a sequential file according to a key is that you can then search the file more quickly. If you know the key value that you want, you can use one of the relatively fast searches.

The **disadvantage** is that when you insert, you need to rewrite at least everything after the insertion point, which makes inserts very expensive unless they are doneat the end of the file.

2. What is database tuning? MAY-2011

Database tuning describes a group of activities used to optimize and homogenize the performance of a database. It usually overlaps with query tuning, but refers to design of the database files, selection of the database management system (DBMS), operating system and CPU the DBMS runs on.

3. Give the measures of quality of a disk.

- Capacity
- Access time
- Seek time
- Data transfer rate
- Reliability
- Rotational latency time.

4. Compare sequential access devices versus random access devices with an example.

Sequential access devices	*Random access devices*
Must be accessed from the beginning	It is possible to read data from any location
Ex: tape storage	Ex: tape storage
Access to data is much slower	Access to data is faster
Cheaper than disk	Expensive when compared with disk

5. What are the types of storage devices?

- Primary storage
- Secondary storage
- Tertiary storage

6. State the storage device hierarchy.(Nov/Dec-20)

- Cache
- Main memory
- Flash memory
- Magnetic disk
- Optical disk
- Magnetic tapes

7. What are called jukebox systems?

Jukebox systems contain a few d rives and numerous disks that can be loaded into one of the drives automatically.

8. What is called remapping of bad sectors?

If the controller detects that a sector is damaged when the disk is initially formatted, or when an attempt is made to write the sector, it can logically map the sector to a different physical location.

9. Define access time.

Access time is the time from when a read or write request is issued to when data transfer begins.

10. Define seek time.

The time for repositioning the arm is called the seek time and it increases with the distance that the arm is called the seek time.

11. Define average seek time.

The average seek time is the average of the seek times, measured over a sequence of random requests.

12. Define rotational latency time.

The time spent waiting for the sector to be accessed to appear under the head is called the rotational latency time.

13. Define average latency time.

The average latency time of the disk is one-half the time for a full rotation of the disk.

14. What is meant by data-transfer rate?

The data-transfer rate is the rate at which data can be retrieved from or stored to the disk.

15. What is meant by mean time to failure?

The mean time to failure is the amount of time that the system could run continuously without failure.

16. What are a block and a block number?

A block is a contiguous sequence of sectors from a single track of one platter. Each request specifies the address on the disk to be referenced. That address is in the form of a block number.

17. What are called journaling file systems?

File systems that support log disks are called journaling file systems.

18. What is the use of RAID? (MAY/June 2013)

A variety of disk-organization techniques, collectively called redundant arrays of independent disks are used to improve the performance and reliability.

19. Explain how reliability can be improved through redundancy.

- The simplest approach to introducing redundancy is to duplicate every disk. This technique is called mirroring or shadowing. A logical disk then consists of two physical disks, and write is carried out on both the disk.
    - If one of the disks fails the data can be read from the other. Data will be lost if the second disk fails before the first failed disk is repaired.

20. What is called mirroring?

The simplest approach to introducing redundancy is to duplicate every disk. This technique is called mirroring or shadowing.

21. What is called mean time to repair?

The mean time to failure is the time it takes to replace a failed disk and to restore the data on it.

22. What is called bit-level striping?

Data striping consists of splitting the bits of each byte across multiple disks. This is called bit-level striping.

23. What is called block-level striping?

Block level striping stripes blocks across multiple disks. It treats the array of disks as a large disk, and gives blocks logical numbers.

24. What are the two main goals of parallelism?

Load –balance multiple small accesses, so that the throughput of such accesses increases. Parallelize large accesses so that the response time of large accesses is reduced

25. What are the factors to be taken into account when choosing a RAID level?

- Monetary cost of extra disk storage requirements.

- Performance requirements in terms of number of I/O operations
- Performance when a disk has failed.
- Performances during rebuild.

26. What is meant by software and hardware RAID systems?

RAID can be implemented with no change at the hardware level, using only software modification. Such RAID implementations are called software RAID systems and the systems with special hardware support are called hardware RAID systems.

27. Define hot swapping.

Hot swapping permits the removal of faulty disks and replaces it by new ones without turning power off. Hot swapping reduces the mean time to repair.

28. Which level of RAID is best? Why?

RAID level 1 is the RAID level of choice for many applications with moderate storage requirements and high I/O requirements. RAID 1 follows mirroring and provides best write performance.

29. Distinguish between fixed length records and variable length records.

Fixed length records

Every record has the same fields and field lengths are fixed.

Variable length records

File records are of same type but one or more of the fields are of varying size.

30. What are the ways in which the variable-length records arise in database Systems?

- Storage of multiple record types in a file.
- Record types that allow variable lengths for one or more fields.
- Record types that allow repeating fields.

31. Explain the use of variable length records.

- They are used for Storing of multiple record types in a file.

- Used for storing records that has varying lengths for one or more fields.
- Used for storing records that allow repeating fields

32. What is the use of a slotted-page structure and what is the information present in the header?

The slotted-page structure is used for organizing records within a single block. The header contains the following information.

- The number of record entries in the header.
- The end of free space
- An array whose entries contain the location and size of each record.

33. What are the two types of blocks in the fixed -length representation?

Define them.

- Anchor block: Contains the first record of a chain.
- Overflow block: Contains the records other than those that are the first record of a chain.

34. What is known as heap file organization?

In the heap file organization, any record can be placed anywhere in the file where there is space for the record. There is no ordering of records. There isa single file for relation**.**

35. What is known as sequential file organization?

In the sequential file organization, the records are stored in sequential order, according to the value of a "search key" of each record.

36. What is hashing file organization?

In the hashing file organization, a hash function is computed on some attribute of each record. The result of the hash function specifies in which block of the file the record should be placed.

37. What is known as clustering file organization?

In the clustering file organization, records of several different relations are stored in the same file.

38. What is an index?

An index is a structure that helps to locate desired records of a relation quickly, without examining all records.

39. What are the two types of ordered indices? DEC-2009

- Primary index
- Secondary index
- Clustered index

40. What are the types of indices?

- Ordered indices – Search keys are stored in sorted order.
- Hash indices – Search keys are distributed uniformly across buckets using hash function.

41. What are the techniques to be evaluated for both ordered indexing and hashing? (Nov/Dec-20)

- Access types
- Access time
- Insertion time
- Deletion time
- Space overhead

42. What is known as a search key?

An attribute or set of attributes used to look up records in a file is called a search key.

43. What is a primary index?

A primary index is an index whose search key also defines the sequential order of the file.

44. What are called index-sequential files?

The files that are ordered sequentially with a primary index on the search key are called index-sequential files.

45. What are the two types of indices?

- Dense index
- Sparse index

46. What are called multilevel indices?

Indices with two or more levels are called multilevel indices.

47. What are called secondary indices?

Indices whose search key specifies an order different from sequential order of the file are called secondary indices. The pointers in secondary index do not point directly to the file. Instead each points to a bucket that contains pointers to the file.

48. What are the disadvantages of index sequential files?

The main disadvantage of the index sequential file organization is that performance degrades as the file grows. This degradation is remedied by reorganization of the file.

49. What is a B+ Tree index?

A B+ Tree index takes the form of a balanced tree in which every path from the root of the root of the root of the tree to a leaf of the tree is of the same length.

50. What is B-Tree?

A B-tree eliminates the redundant storage of search-key values .It allows search key values to appear only once.

51. What is hashing?

Hashing allows us to find the address of a data item directly by computing a hash function on the search key value of the desired record.

52. How do you create index in SQL?

We create index by he create index command.

Create index <index name> on <relation name> (<attribute list>)

53. Distinguish between static hashing and dynamic hashing?

Static hashing

Static hashing uses a hash function in which the set of bucket adders is fixed. Such hash functions cannot easily accommodate databases that grow larger over time.

Dynamic hashing

Dynamic hashing allows us to modify the hash function dynamically. Dynamic hashing copes with changes in database size by splitting and coalescing buckets as the database grows and shrinks.

54. What is a hash index?

A hash index organizes the search keys, with their associated pointers, into a hash file structure.

55. What can be done to reduce the occurrences of bucket overflows in a hash file organization?

To reduce bucket overflow the number of bucket is chosen to be (nr/fr)*(1+d).

We handle bucket overflow by using

- Overflow chaining (closed hashing)
- Open hashing

56. Differentiate open hashing and closed hashing.

Closed hashing (overflow chaining)

If a record must be inserted in to a bucket b, and b is already full, the system provides an overflow bucket for b, and inserts the record in to the overflow bucket. If the overflow bucket is also full, the system provides another overflow bucket, and so on. All the overflow buckets of a given buckets are chained together in a linked list, overflow handling using linked list is known as closed hashing.

Open hashing

The set of buckets is fixed, and there are no overflow chains. Instead, if a bucket is full, the system inserts records in some other bucket in the initial set of buckets.

57. What is linear probing?

Linear probing is a type of open hashing. If a bucket is full the system inserts records in to the next bucket that has space. This is knownas linear probing.

58. What is called query processing?

Query processing refers to the range of activities involved inextracting data from a database.

59. What are the steps involved in query processing?

The basic steps are:

- Parsing and translation
- Optimization
- Evaluation

60. What is called an evaluation primitive?

A relational algebra operation annotated with instructions on how to evaluate is called an evaluation primitive.

61. Define query optimization.

Query optimization refers to the process of finding g the lowest –cost method of evaluating a given query.

62. What is called a query -execution engine?

The query execution engine takes a query evaluation plan, executes that plan, and returns the answers to the query.

63. How do you measure the cost of query evaluation?

The cost of a query evaluation is measured in terms of a number of different resources including disk accesses, CPU time to execute a query, and in a distributed database system the cost of communication

64. List out the operations involved in query processing.

- Selection operation
- Join operations.
- Sorting.
- Projection
- Set operations
- Aggregation

65. What are called as index scans?

Search algorithms that use an index are referred to as index scans.

66. What is called as external sorting?

Sorting of relations that do not fit into memory is called as external sorting.

67. Explain nested loop join.

Nested loop join consists of a pair of nested for loops.

Example: r| | s

T s r is the outer relation and s is the inner relation.

68. What is meant by block nested loop join?

Block nested loop join is the variant of the nested loop join where every block of the inner relation is paired with ever y block of the outer relation. Within each pair of blocks ever y tuple in one block is paired with every tuple in the other blocks to generate all pairs of tuples.

69. What is meant by hash join?

In the hash join algorithm a hash function h is used to implement partition tuples of both relations.

70. What is called as recursive partitioning?

The system repeats the splitting of the input until each partition of the build input fits in the memory. Such partitioning is called recursive partitioning.

71. What is called as an N-way merge?

The merge operation is a generalization of the two-way merge used by the standard in-memory sort-merge algorithm. It merges N runs, so it is called an N-way merge.

72. What is known as fudge factor?

The number of partitions is increased by a small value called thefudge factor, which is usually 20 percent of the number of hash partitions computed.

73. What are ordered indices? DEC-2011

Ordered Indices is a types of index in which search keys are stored in sorted order.

74. Distinguish between sparse index and dense index. DEC-2011,Apr/may 2019

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.

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.

75. What is the difference between primary index and secondary index? MAY-2010

Primary index:

- In a sequentially ordered file, the index whose search key specifies the sequential order of the file. Also called clustering index.
- The search key of a primary index is usually but not necessarily the primary key. An index structure that is defined on the ordering field

Secondary index:

- An index whose search key specifies an order different from the sequential order of the file. Also called non-clustering index.

76. Define the terms fragmentation and replication, in terms of where data is stored.

Fragmentation:

The system partitions the relation into several fragments and stores each fragments at a different site.

Replication:

The system maintains several identical copies of the relation and storeseach replica at a different site.

77. What are structured data types? What are collection types, in particular? DEC-2008

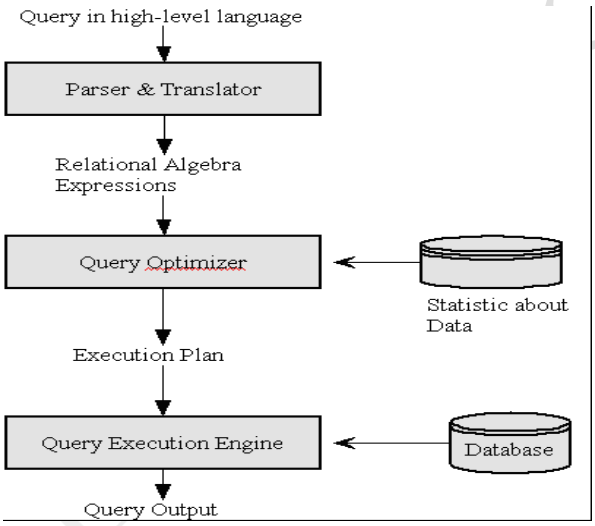
Sets are instances of collection types. Other instances of collection types include arrays and multi sets.

80. What is the drawback of flash memory? DEC-2010

Disadvantages:

- Flash memory cells have a limited number of write and erase cycles before failing.
- Most flash drives do not have a write-protection mechanism
- Smaller size devices, such as flash drives make them easier to lose
- Currently costs a lot more per gigabyte than traditional hard drives for large storage capacities.
- May require a special version of a program to run on a flash-based drive to protect from prematurely wearing out the drive.

81. Give a diagrammatic representation to indicate the basic steps in query processing. DEC-2002



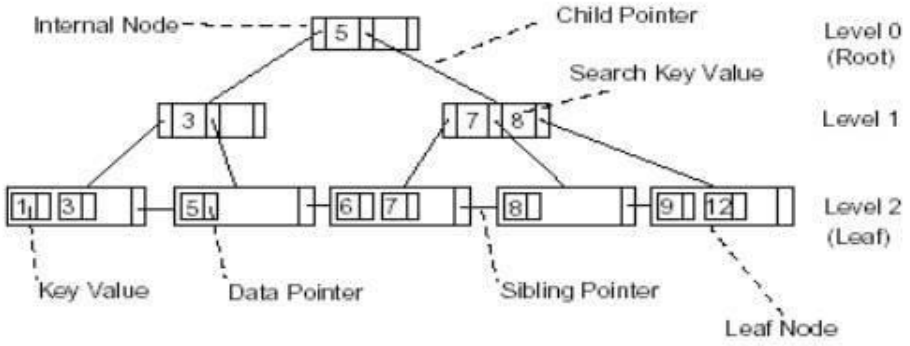
82. How to choose the best evaluation plan for a query? Explain. MAY-2003

The optimizer should choose the access path that retrieves the fewest records in the most efficient way by estimating the different cost and choosing the methods with the least estimated cost.

83. Draw the structure of a typical node of a B+ tree and explain briefly.

B+ tree considers all the keys in nodes except the leaves as dummies. All keys are duplicated in the leaves. This has the advantage that is all the leaves are linked together sequentially, the entire tree may be scanned without visiting the higher nodes at all.

B+ Tree Structure



- A B + -Tree consists of one or more blocks of data, called *node*s, linked together by pointers. The B + -Tree is a tree structure. The tree has a single node at the top, called the *root nod*e. The root node points to two or more blocks , called *child node*s. Each child nodes points to further child nodes and so on.

85. What can be done to reduce the occurrences of bucket overflows in a hash file organization? DEC-2007

To reduce the occurrence of overflows

1. Choose the hash function more carefully, and make better estimation of the relation size.

2. If the estimated size of the relation is n, and the number of records per block

is f_r, allocate $(n/f_r)*(1+d)$ buckets instead of (n_r/f_r) buckets.

Here d is a fudge factor, typically around 0.2. Some space is wasted.

86. How does a B tree differ from B+ tree? Why is a B+ tree usually preferred as an access structure to a data file? DEC-2008

A B tree of order m (maximum number of children for each node) is a tree which satisfies:

1. Every node has at most m children.
2. Every node (except root) has at least m/2 children.
3. The root has at least two children if it is not a leaf node.
4. All leaves appear in the same level, and carry information.
5. A non-leaf node with k children contains k−1 keys.

In a B+ tree, in contrast to a B-tree, all records are stored at the leaf level of the tree, only keys are stored in interior nodes.

- Advantage of B+ trees over B trees is they allow you to in pack more pointers to other nodes by removing pointers to data, thus increasing the fanout and thus decreasing the depth of the tree.
- In B tree we have to traverse the whole tree structure to get result while it is easier to search in B+ tree as the leaves form a linked list.
- Since the internal nodes in B+ tree contains only pointers(index) to the keyvalues it very useful in implementing Indexed Sequential files.
- We can traverse easily in B+ tree using the indices to reach the required key.

87. Describe flash memory. MAY-2010

- Flash memory is an electronic non-volatilecomputer storage device that can be electrically erased and reprogrammed.
- Flash memory was developed from EEPROM (electrically erasable programmable read-only memory). There are two main types of flash memory, which are named after the NAND and NOR logic gates.

88. Mention all the operation of files(Apr/may 2019)

Various file operation are 1. Creation of file 2. Insertion of data 3. Deletion of data 4.Searching desired data from a file

89. Which cost Components Contribute to Query Execution?

The cost of query evaluation can be measured in terms of a number of different resources,

- ✓ disk accesses
- ✓ CPU time to execute a query
- ✓ cost of communication
- ✓ number of block transfers from disk

PART-B

1. Briefly explain RAID and RAID levels.

- RAID stands for **Redundant array of independent disks.**
- It is a way of **storing the same data in different places** on multiple hard disks to protect data in the case of a drive failure.
- RAID storage uses multiple disks in order to provide fault tolerance, to improve overall performance, and to increase storage capacity in a system.
- RAID allows you to store the same data redundantly (in multiple paces) in a balanced way to improve overall performance.

Synopsis:

1. RAID 0 - Striped disk array without fault tolerance
2. RAID 1: (Mirroring and duplexing)
3. RAID 2: (Error-correcting coding)
4. RAID 3: (Bit-interleaved parity)
5. RAID 4: (Dedicated parity drive)
6. RAID 5: (Block interleaved distributed parity)
7. RAID 6: (Independent data disks with double parity)

RAID 0: (Striped disk array without fault tolerance)

- It also known as a **stripe set or striped volume splits** ("stripes") data evenly across two or more disks, without parity information (equivalence)as in Fig 4.1.
- RAID 0 provides **no fault tolerance or redundancy**, the failure of one drive will cause the **entire array to fail**; as a result of having data striped across all disks, the failure will result in total data loss.
- RAID 0 is normally used to **increase performance.**

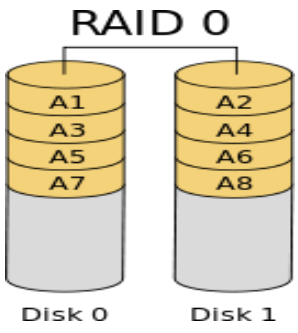


Fig 4.1 RAID 0

RAID 1: (Mirroring and duplexing)

- RAID 1 consists of an exact copy (or mirror) of a set of data on two or more disks; a classic RAID 1 contains two disks.
- This configuration offers striping (**splits**) or spanning (Both sides) and no parity information of disk space across multiple disks.
- Data is mirrored on all disks belonging to the array, and the array can only be as big as the smallest member disk.

RAID 2: (Error-correcting coding)

- It consists of **bit-level striping** with dedicated Hamming-code as in Fig 4.2.
- This configuration **uses striping across disks**, with some disks **storing error checking and correcting information**.
- All disk **spindle rotation is synchronized** and data is striped such that each sequential bit is on a different drive.
- **Hamming code offered little advantage** so it is rarely implemented and not currently used.

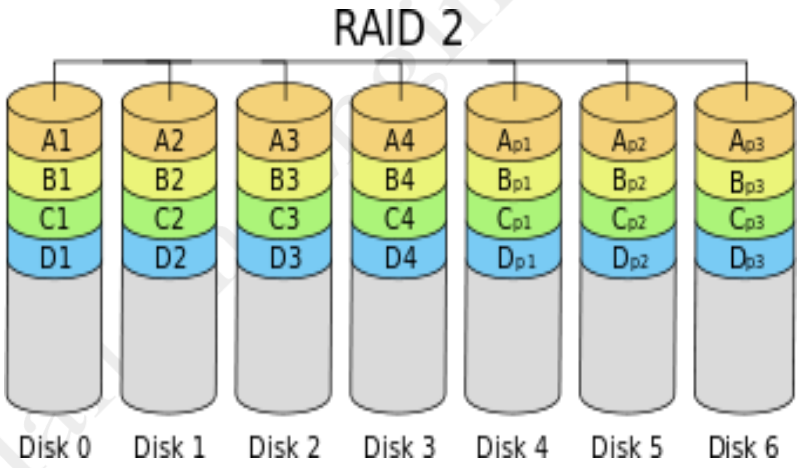


Fig 4.2 RAID 2

RAID 3: (Bit-interleaved parity)

- It provides byte-level striping with a dedicated parity disk as in Fig 4.3.
- This is cannot service or process on multiple requests simultaneously or concurrently.
- Data are spread across all disks in array and it will reside in the same location.
- If we are going to perform any I/O operation, then every disk usually requires synchronized spindles.

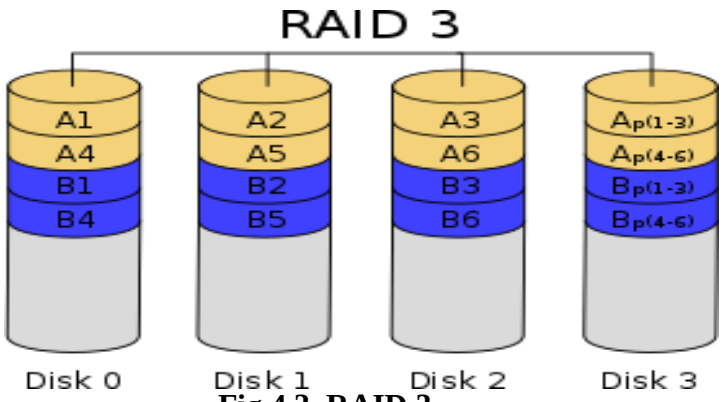


Fig 4.3 RAID 3

RAID 4: (Dedicated parity drive)

- It consists of **block-level striping** with a dedicated parity disk as in Fig 4.4.
- It provides a good performance of random reads, while the performance of random writes is low due to the need to write all parity data to a single disk.
- In this one, read/write operation does not have to spread across all data drives. As a result, more I/O operations can be executed in parallel.
- This level that provided no significant advantages so it is rarely implemented and not currently used.

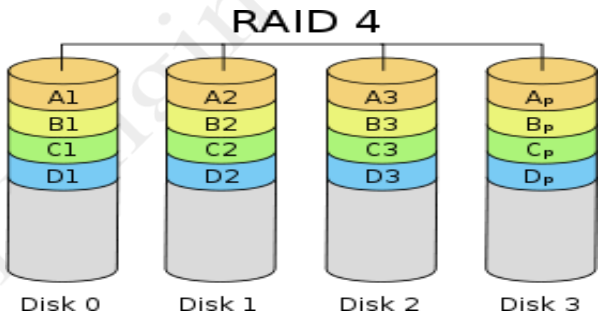


Fig 4.4 RAID 4

RAID 5: (Block interleaved distributed parity)

- It consists of block-level striping with distributed parity as in Fig 4.5.
- It requires that all drives but one be present to operate.
- This results in excellent performanceand good fault tolerance
- RAID 5 requires at least three disks.
- Level 5 is one of the most popular implementations of RAID.

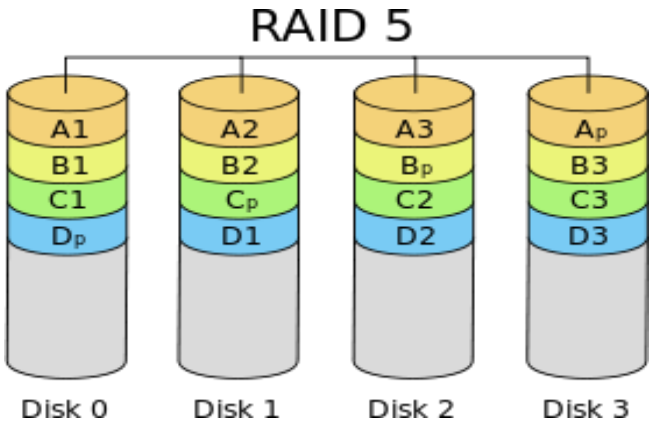


Fig 4.5 RAID 5

RAID 6: (Independent data disks with double parity)

- This is simply extends RAID 5 by adding another parity block; thus, it uses block - level striping with two parity blocks distributed across all member disks as in Fig 4.6.
- Double parity provides fault tolerance up to two failed drives.
- Any form of RAID that can continue to execute read and write requests to all of a RAID array's virtual disks in the presence of any two concurrent disk failures.

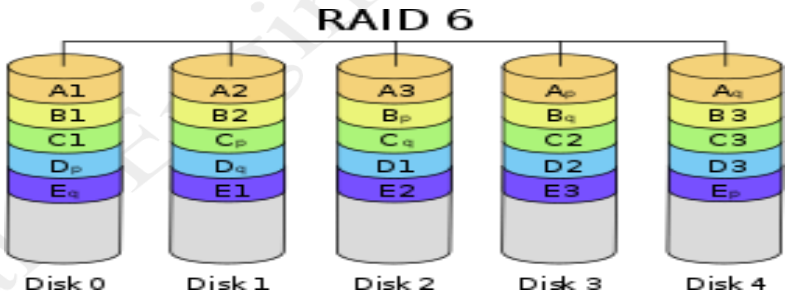


Fig 4.6 RAID 6

2. Explain in details about with file system organization and their type with suitable diagram.

- A file is a collection or **set of data elements** (ordered or unordered) stored on storage media like computer and other devices.
- A system software or computer program (Text Editor) is **responsible for managing a file** (reading, processing, deleting, etc.).

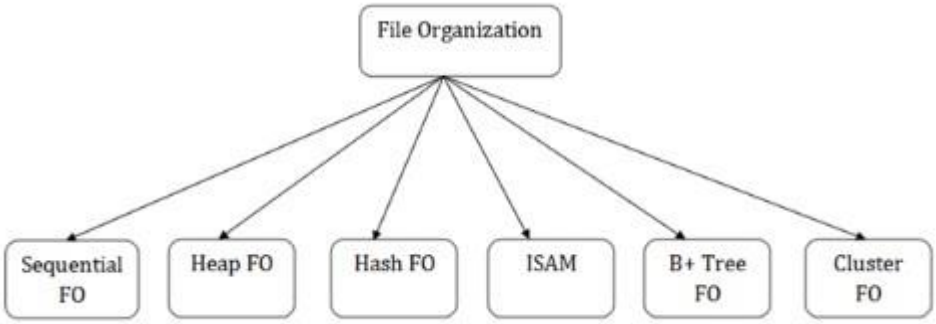


Fig 4.7 Types of file organization

Types of file organization as in fig 4.7:

1) Sequential file organization:

- Records in sequential files can be **read or written only sequentially**.

A-217	Brighton	750	
A-101	Downtown	500	
A-110	Downtown	600	
A-215	Mianus	700	
A-102	Perryridge	400	
A-201	Perryridge	900	
A-218	Perryridge	700	
A-222	Redwood	700	
A-305	Round Hill	350	

- Records in a sequential file **can be stored in two ways**.
  - **i. Pile File method**
  - **ii. Sorted File method**

i. Pile File Method:

- In this method the record can be store in a sequence, i.e., one after another. Here, the record will be inserted in the order in which they are inserted into tables as in Fig 4.8.

- In case of updating or deleting of any record, the record will be searched in the memory blocks. When it is found, then it will be marked for deleting, and the new record is inserted.

Fig 4.8 Example of Insertion of the new record:

Suppose we have four records R1, R3 and so on up to R9 and R8 in a sequence. Hence, records are nothing but a row in the table. Suppose we want to insert a new record R2in the sequence, then it will be placed at the end of the file as in Fig 4.9. Here, records are nothing but a row in any table.

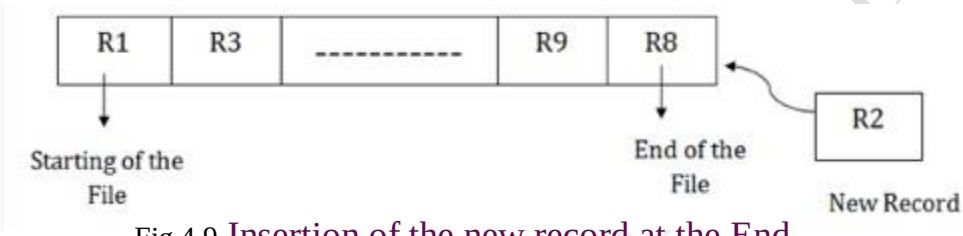


Fig 4.9 Insertion of the new record at the End

ii. Sorted File Method:

- In this method, the new record is always inserted at the file's end, and then it will sort the sequence in ascending or descending order as in Fig 4.10.
- Sorting of records is based on any primary key or any other key.
- In the case of modification of any record, it will update the record and then sort the file, and lastly, the updated record is placed in the right place.

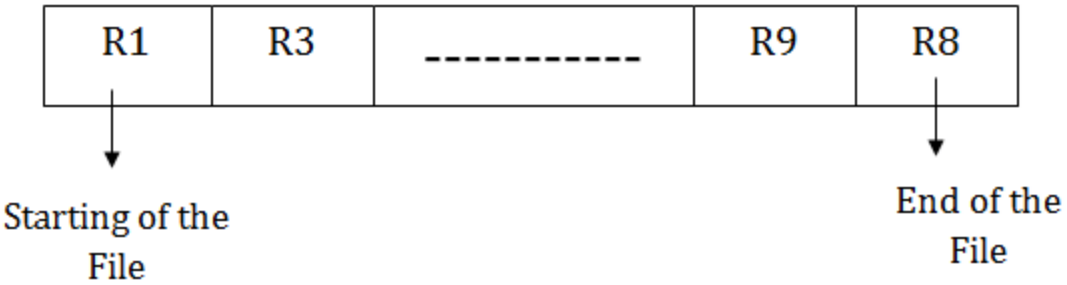


Fig 4.10 Example of Insertion of the new record:

Suppose there is a preexisting sorted sequence of four records R1, R3 and so on up to R6 and R7. Suppose a new record R2 has to be inserted in the sequence, then it will be inserted at the end of the file, and then it will sort the sequence as in Fig 4.11.

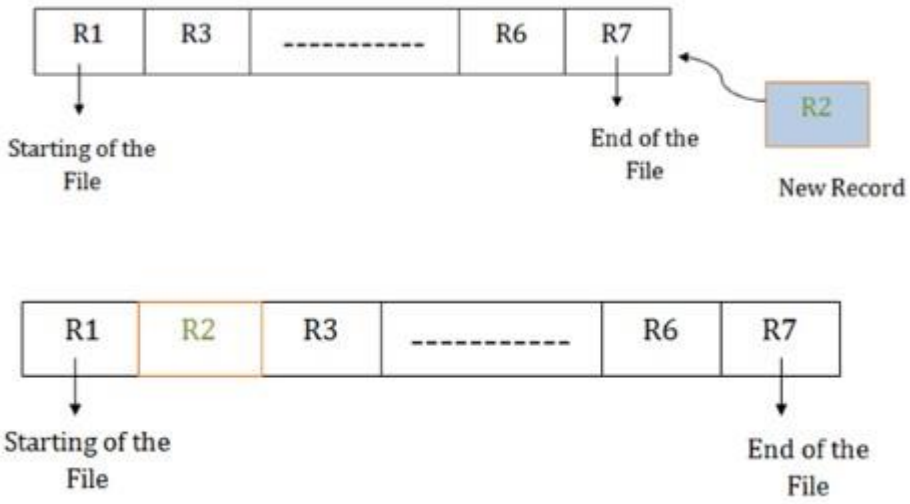


Fig 4.11 Insertion of the new record at the End

Advantage:

- **Fast and efficient** when dealing with large volume of data.
- Good for group or batch transactions.
- **Simple** to implement
- Good for report generation, statistical computation and inventory control.

Disadvantages:

- All new transaction **should sorted** for a sequential file processing
- **Not good** for interactive transactions

2) Heap file organization

- In heap file organization, the records are inserted at the file's end. When the records are inserted, it doesn't require the sorting and ordering of records.
- When the data block is full, the new record is stored in some other block.
- This new data block need not to be the very next data block, but it can select any data block in the memory to store new records.
- The heap file is also known as an unordered file.
- In the file, every record has a unique id, and every page in a file is of the same size as in below Fig 4.12.

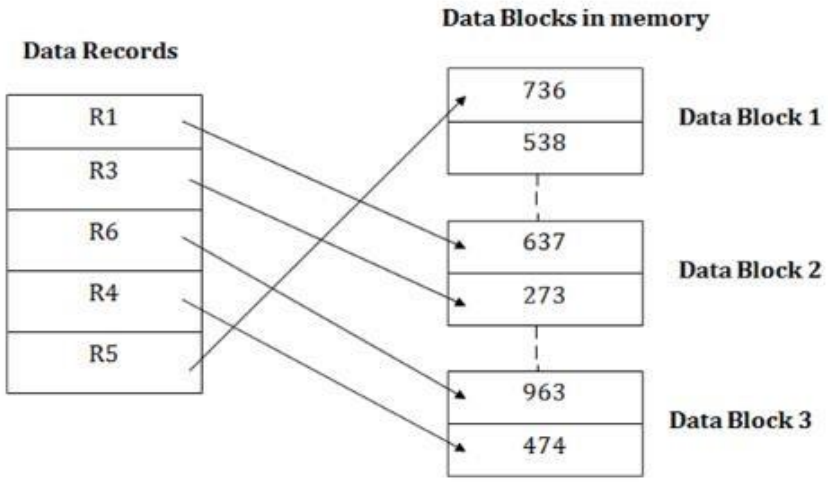


Fig 4.12 Heap file organization

Examples of Insertion of a new record

Suppose we have five records R1, R3, R6, R4 and R5 in a heap and suppose we want to insert a new record R2 in a heap. If the data block 3 is full then it will be inserted in any of the database selected by the DBMS as in fig 4.13, let's say data block 1.

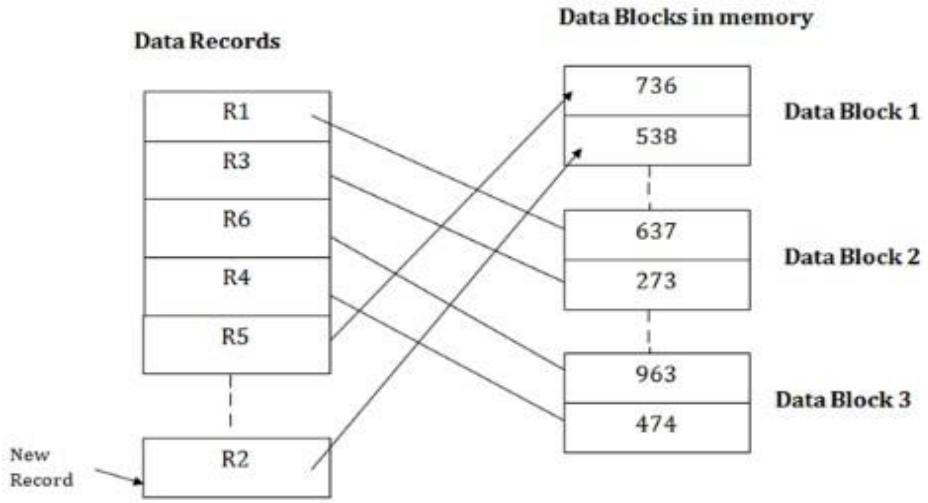


FIG 4.13 Examples of Insertion of a new record

If we want to search, update or delete the data in heap file organization, then we need to traverse the data from staring of the file till we get the requested record.

If the database is very large then searching, updating or deleting of record will be time-consuming because there is no sorting or ordering of records. In the heap file organization, we need to check all the data until we get the requested record.

3. Hash File Organization

- **Hashing** is an efficient technique to directly search the location of desired data on the disk without using index structure.

- 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 data bucket as in Fig 4.14.

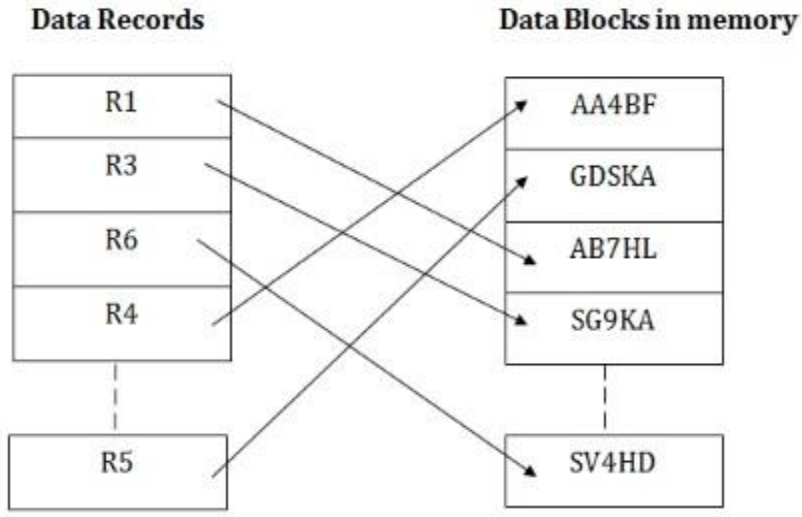


Fig 4.14 Hash File Organisation

When a record has to be received using the hash key columns, then the address is generated, and the whole record is retrieved using that address.

In the same way, when a new record has to be inserted, then the address is generated using the hash key and record is directly inserted as in Fig 4.15. The same process is applied in the case ofdelete and update.

In this method, there is no effort for searching and sorting the entire file. In this method, each record will be stored randomly in the memory.

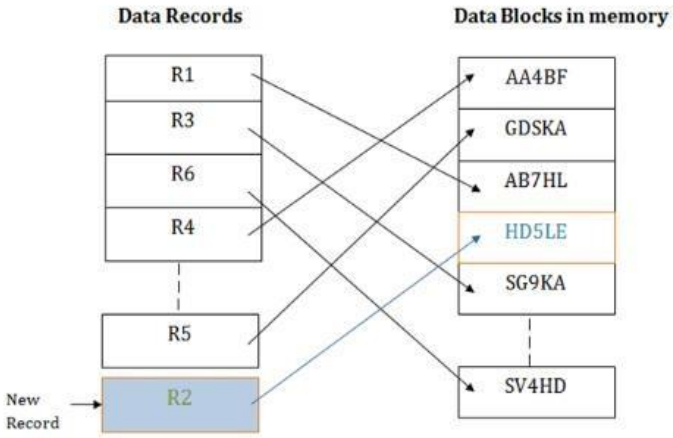


Fig 4.15 New record insertion.

4. Indexed sequential access method (ISAM)

- In this method, records are stored in the file using the primary key.

- An index value is generated for each primary key and mapped with the record. This index contains the address of the record in the file Fig 4.16.

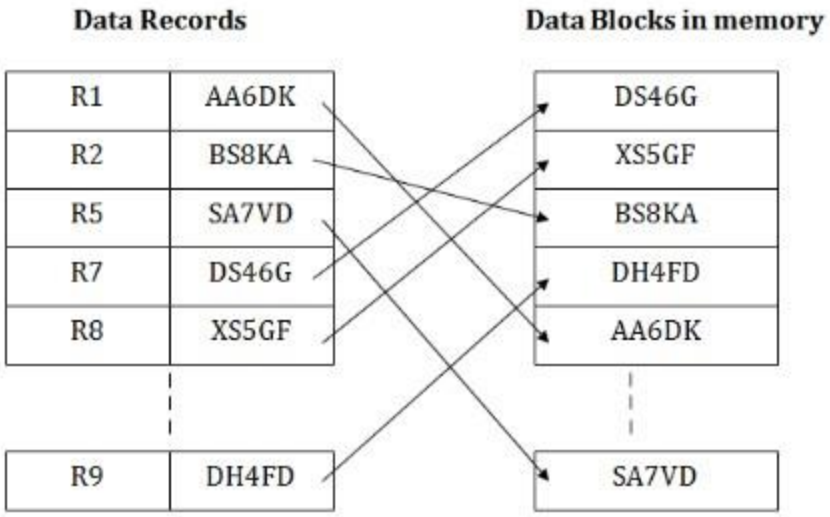


Fig 4.15 Indexed sequential access method

If any record has to be retrieved based on its index value, then the address of the data block is fetched and the record is retrieved from the memory.

5. B+ File Organization

- B+ tree file organization is the advanced method of an indexed sequential access method. It uses a tree-like structure to store records in File.
- It uses the same concept of key-index where the primary key is used to sort the records. For each primary key, the value of the index is generated and mapped with the record.
- The B+ tree is similar to a binary search tree (BST), but it can have more than two children.
- In this method, all the records are stored only at the leaf node. Intermediate nodes act as a pointer to the leaf nodes. They do not contain any records.

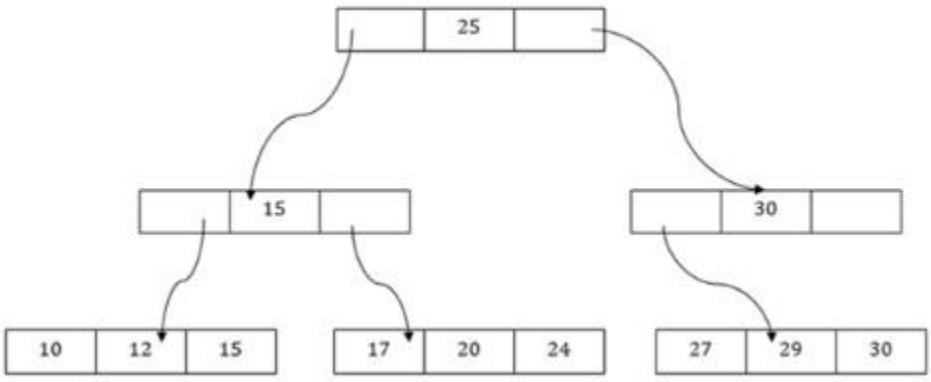


Fig 4.16 B+ tree

The above Fig 4.16 B+ tree shows that:

- There is one root node of the tree, i.e., 25.
- There is an intermediary layer with nodes. They do not store the actual record. They have only pointers to the leaf node.

- The nodes to the left of the root node contain the prior value of the root and nodes to the right contain next value of the root, i.e., 15 and 30 respectively.
- There is only one leaf node which has only values, i.e., 10, 12, 17, 20, 24, 27 and 29.
- Searching for any record is easier as all the leaf nodes are balanced.
- In this method, searching any record can be traversed through the single path and accessed easily.

6. Cluster file organization

- When the two or more records are stored in the same file, it is known as clusters.
- These files will have two or more tables in the same data block, and key attributes which are used to map these tables together are stored only once.
- This method reduces the cost of searching for various records in different files.
- In the given example, we are retrieving the record for only particular departments. This method can't be used to retrieve the record for the entire department.

Cluster Key

DEP_ID	DEP_NAME	EMP_ID	EMP_NAME	ADDRESS
10	Math	7	Amy	Bihar
11	English	4	Amelia	Meerut
12	Java	2	Robert	Gujarat
12		8	Sonoo	UP
13	Physics	6	Jackson	Delhi
14	Civil	1	John	Delhi
14		5	Kristen	Noida
15	Chemistry	3	David	Mumbai

Table 1 :Retrieving the record

In this method, we can directly insert, update or delete any record. Data is sorted based on the key with whichsearching is done. Cluster key is a type of key with which joining of the table is performed as in Table 1.

Types of Cluster file organization:

Cluster file organization is of two types:

1. Indexed Clusters:

- In indexed cluster, records are grouped based on the cluster key and storedtogether.
- The above EMPLOYEE and DEPARTMENT relationship is an example of an indexed cluster. Here, all the records are grouped based on the cluster key-DEP_ID and all the records are grouped.

2. Hash Clusters:

3. It is similar to the indexed cluster. In hash cluster, instead of storing the records based on the cluster key, we generate the value of the hash key for the cluster key and store the records with the same hash key value.

3. Explain the concept of Data Dictionary and column oriented storage in DBMS.

Data dictionary is mini database management system that manages the metadata.

- Data dictionaries are helpful to the database administrators in management of database.
- The general structure of data dictionary is as shown in the Table 2.
- The data in the database dictionary is maintained by several programs and generates the reports if required.
- The dictionary is integrated with the database systems in which the data is controlled by the data dictionaries and is made available to the DBMS software.

Following type of information is maintained by data dictionary.

a) Description of the schema of the database.
b) Detailed information about the physical database design
c) Description of database users, their roles and their access rights.
d) Description of database transactions.
e) The description of the relationship between database transactions and data items referenced by them.
f) Information about the usage statistics. That means, how many times the queries are raised to the database, how many transactions are made by the DBMS.

- For example – Consider a Student database, in which various fields are RollNo, FirstName, LastName, and CourseID. The data dictionary for this database maintains the information about this database. The data dictionary contains the column names, Data types of each field and the description of each column of the database.

Database

RollNo	FirstName	LastName	CourseID
101	AAA	ZZZ	CS111
102	BBB	YYY	ETC121
103	CCC	WWW	Mech923

Data dictionary

Column	Data type	Description
RollNo	int	Primary key of table
FirstName	varchar(20)	First name of the student
LastName	Varchar(20)	Last name of the student
CourseID	Varchar(20)	The ID of the course taken by the student. This ID is present in course table.

Table 2: Representation of data dictionary

Active and passive data dictionaries:

Active data dictionary

- Active Data dictionary is managed automatically by the database management system.
- They are consistent with current structure.
- In the active data dictionary, when any modification or changes is executed by the DBMS, then this dictionary it also gets modified by the DBMS automatically.
- Most of the active data dictionaries are derived from system catalogue.

Passive data dictionary

- Passive data dictionary is used only for documentation purpose.
- Passive dictionary is self-contained application and set of files used for documenting the data processing environment.
- The purpose of maintaining or modification of the database is manual.
- It is managed by the users of the database systems.

Difference between active and passive data dictionary

Sr. No,	Active data dictionary	Passive data dictionary
1.	The database management system automatically maintains the active data dictionary	The passive data dictionary is modified whenever the structure of database gets changed.
2.	It is very consistent	It is not every consistent
3.	The database management systems automatically manage this dictionary	These users are responsible for manually managing this dictionary.
4.	It does not require separate database	It requires separate database for working with dictionary.

Column Oriented Storage:

- Column oriented storage which is also called as columnar database management system that stores data in columns rather than rows.
- The advantage of column-oriented storage is to retrieve the result of query **efficiently.**
- It also improves disk I/O performances.
- **Example-**

Row - oriented

ID	Name	Subject	Marks
1	AAA	English	87
2	BBB	Maths	95
3	CCC	History	83

Column - oriented

ID	Name
1.	AAA
2.	BBB
3.	CCC

ID	Subject	Marks
1.	English	87
2.	Maths	95
3.	History	83

- As column oriented database store data by columns instead or rows, it column more than data in smaller amount of memory.
- The data retrievel is done column by column only the columns the need required are retrieved. This makes it possible for efficient retravel of data large amount of data can be handled.
- It is mainly used in data analytics, business intelligence and data warehouse.

Applications of columnar Database

- ✓ Queries that involve only a few columns
- ✓ Aggregation queries against vast amounts of data
- ✓ Column-wise compression

Advantages of Columnar Database:

- ✓ Related to big data comes into play then the column-oriented database have greater attention in such case.
- ✓ The data in the columnar database has a highly compressible nature.
- ✓ Efficiency and speed.
- ✓ Self-indexing: It uses less disk space than a relational database management system containing the same data.

Limitation of columnar Database:

1. For loading incremental data,traditional database are more relevant as compared to column-oriented databases.
2. For online transaction processing(OLTP)applications,Row oriented databases are more appropriate than columnar databases

4. Discuss in details about indexing and ordered indices.(Nov/Dec-20)

- Indexing is used to optimize the performance of a database by minimizing the number of disk accesses required when a query is processed.
- The index is a type of data structure. It is used to locate and access the data in a database table quickly.

Index structure:

Indexes can be created using some database columns.

Search key	Data Reference

- The first column of the database is the **search key** that contains a copy of the primary key or candidate key of the table.
- The second column of the database is the **data reference**. It contains a set of pointers holding the address of the disk block where the value of the particular key can be found.

Indexing Methods:

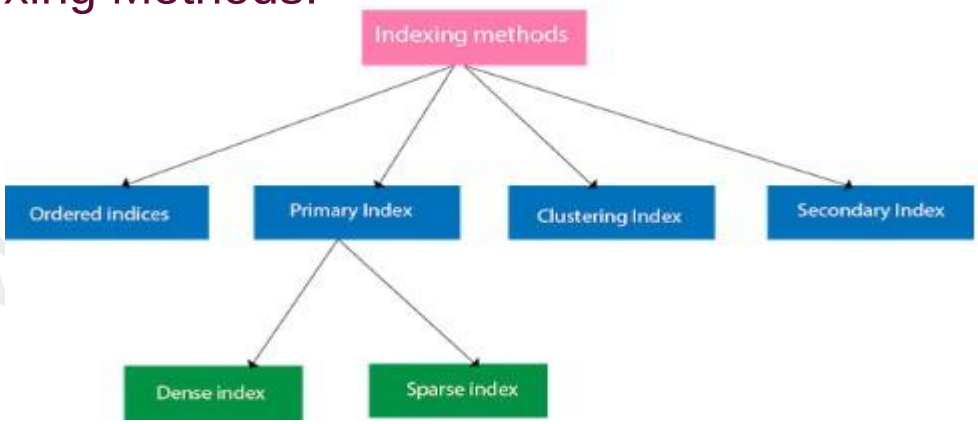


Fig 4.17 Indexing methods

1. Ordered indices

From the Fig 4.17,The indices are usually sorted to make searching faster. The indices which are sorted are known as ordered indices.

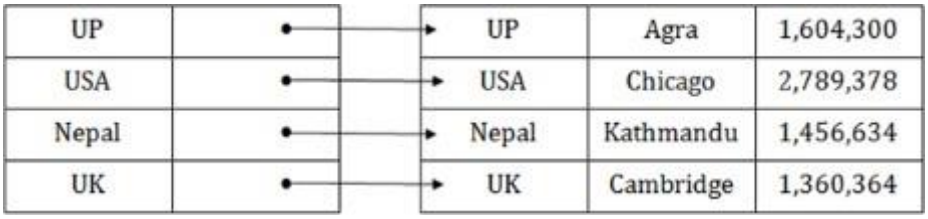
- **Example**: Suppose we have an employee table with thousands of record and each of which is 10 bytes long. If their IDs start with 1, 2, 3. and so on and we have to search student with ID-543.
- In the case of a database with no index, we have to search the disk block from starting till it reaches 543. The DBMS will read the record after reading 543*10=5430 bytes.
- In the case of an index, we will search using indexes and the DBMS will read the record after reading 542*2= 1084 bytes which are very less compared to the previous case.

2. Primary Index

- If the index is created on the basis of the primary key of the table, then it is known as primary indexing.
- These primary keys are unique to each record and contain 1:1 relation between the records.
- As primary keys are stored in sorted order, the performance of the searching operation is quite efficient.
- The primary index can be classified into two types:
    i. Dense index
    ii. Sparse index.

i. Dense index

- The dense index contains an index record for every search key value in the data file. It makes searching faster.
- The number of records in the index table is same as the number of records in the main table.
- It needs more space to store index record itself.
- The index records have the search key and a pointer to the actual record on the diskas in Table 2.



UP		UP	Agra	1,604,300
USA		USA	Chicago	2,789,378
Nepal		Nepal	Kathmandu	1,456,634
UK		UK	Cambridge	1,360,364

Table 2:Dense Index

Sparse index

- In the data file, index record appears only for a few items. Each item points to a block as in Table 3.

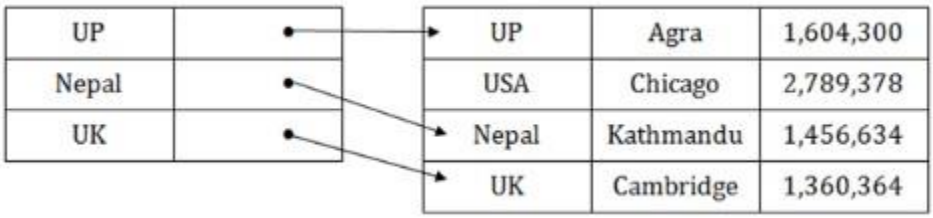


Table 3:Dense Index

3. Clustering Index

- A clustered index can be defined as an ordered data file. Sometimes the index is created on non-primary key columns which may not be unique for each record.
- In this case, to identify the record faster, we will group two or more columns to get the unique value and create index out of them. This method is called a clustering index.
- The records which have similar characteristics are grouped, and indexes are created for these group.

Example1: Suppose a company contains several employees in each department. Suppose we use a clustering index, where all employees which belong to the same Dept_ID are considered within a single cluster, and index pointers point to the cluster as a whole. Here Dept_Id is a non-unique key as in Fig 4.18.

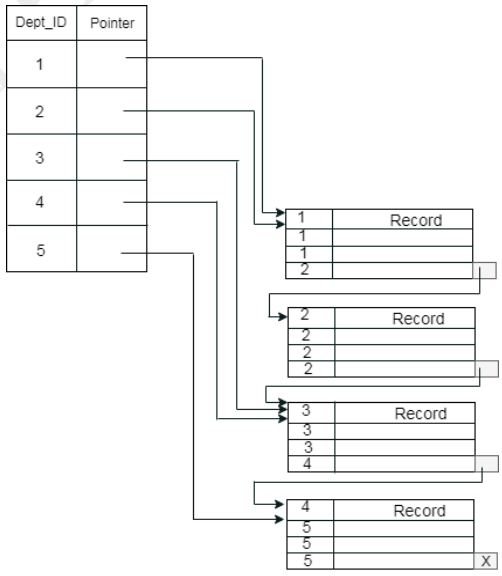


Fig 4.18**:Clustering Index**

- In this, instead of pointing to each record in the main table, the index points to the records in the main table in a gap.
- The previous schema is little confusing because one disk block is shared by records which belong to the different cluster.
- If we use separate disk block for separate clusters, then it is called better technique as in Fig 4.19.

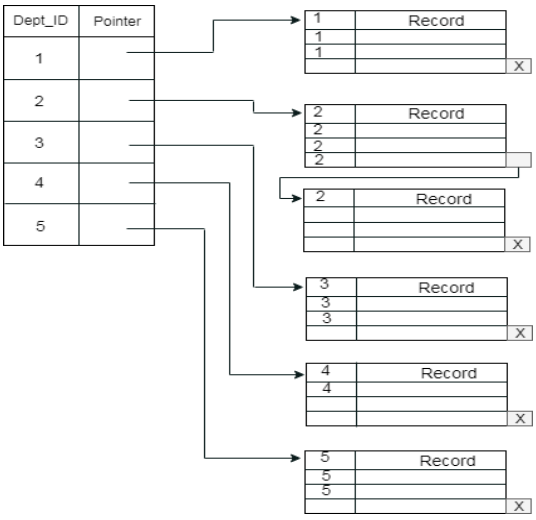


Fig 4.19:Separate Cluster

4. Secondary Index(multilevel Index)

- Index records **include search-key values and data pointers**.
- Multilevel index is stored on the **disk along with the actual database files**.
- As the size of the database grows, so does the size of the indices.
- Multi-level Index helps in **breaking down the index into several smaller indices** as in Fig 4.20
- It can be saved in a single disk block, which **can easily be accommodated anywhere** in the main memory.
- The mapping of the first level is stored in the primary memory, so that address fetch is faster.
- The mapping of the second level and actual data are stored in the secondary memory (hard disk).

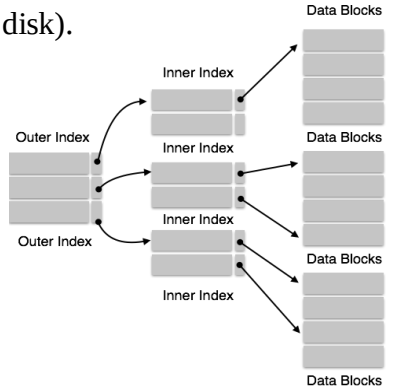


Fig 4.20:Secondary Index

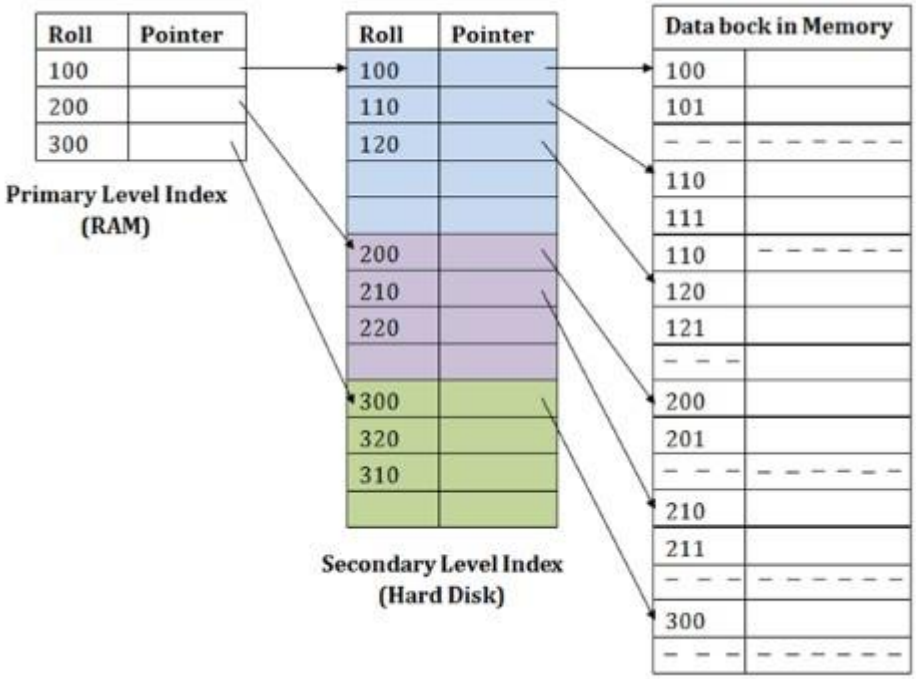


Table 4: Secondary Index

For example:

- If you want to find the record of roll 111 in the diagram, then it will search the highest entry which is smaller than or equal to 111 in the first level index. It will get 100 at this level.
- Then in the second index level, again it does max (111) <= 111 and gets 110. Now using the address 110, it goes to the data block and starts searching each record till it gets 111 as in Table 4.
- This is how a search is performed in this method. Inserting, updating or deleting is also done in the same manner.

5.Discuss in details about hashing. Compare static hashing and dynamic hashing.

- Hashing is an efficient technique to directly search the location of desired data on the disk without using index structure.
- 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 data bucket.**

Hash File Organization:

- **Data bucket –** Data buckets are the memory locations where the records are stored. These buckets are also considered as Unit of Storage.
- **Hash Function –** Hash function is a simple mathematical mapping function that maps all the set of search keys to actual record address.

  Generally, hash function uses primary key to generate the hash index – address of the data block..
- **Hash Index-**The prefix of an entire hash value is taken as a hash index. Every hash index has a depth value to signify how many bits are used for computing a hash function.

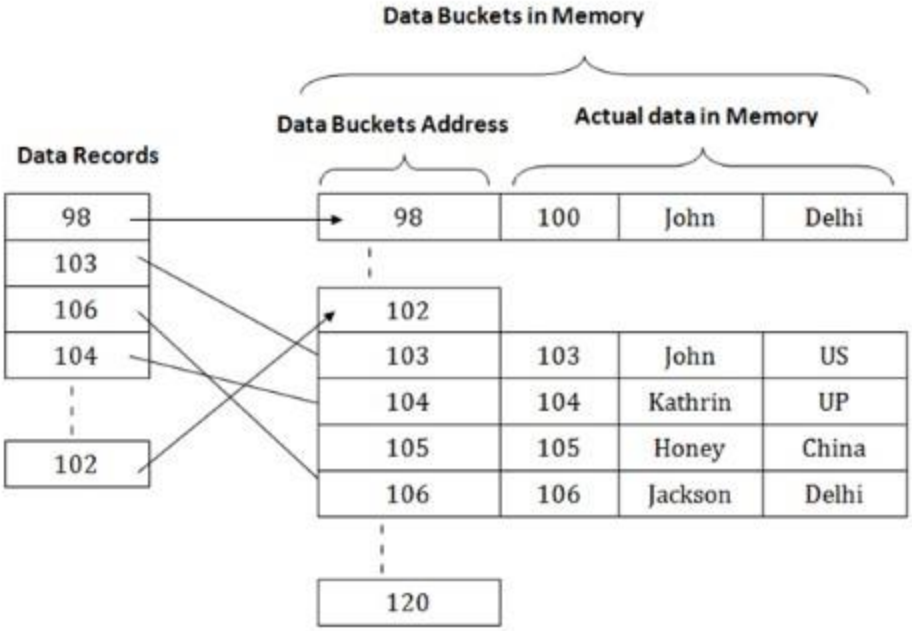


Fig 4.21(a) Hash File Organization

- The above Fig 4.21(a) shows data block addresses same as primary key value.
- This hash function can also be a simple mathematical function like exponential, mod, cos, sin, etc.
- Suppose we have mod (5) hash function to determine the address of the data block.
- In this case, it applies mod (5) hash function on the primary keys and generates 3, 3, 1, 4 and 2 respectively, and records are stored in those data block addresses as in Fig 4.22(b)

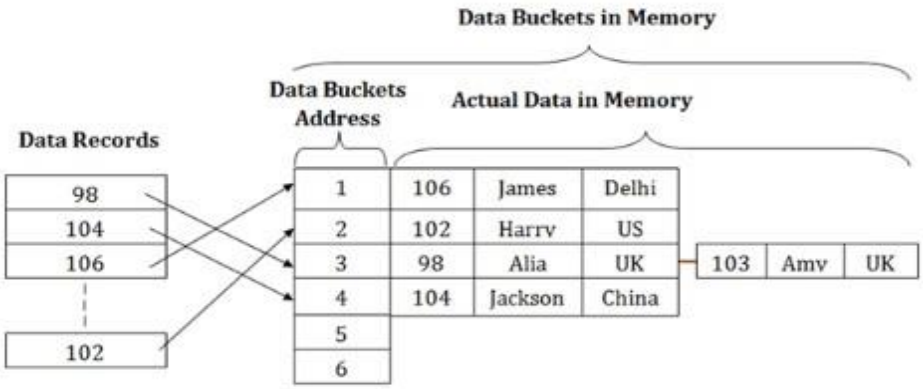
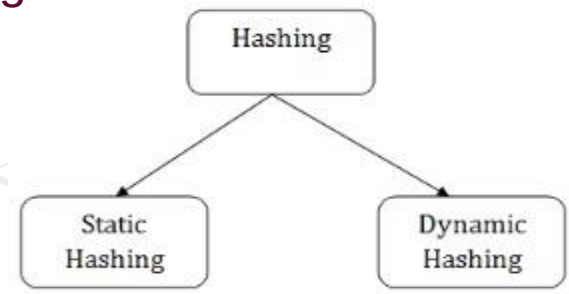


Fig 4.22 (b) Hash File Organization

Types of Hashing:



1. Static Hashing

➢ In static hashing, when a search-key value is provided, the hash function always computes the same address.

Operations:

Insertion – When a new record is inserted into the table, The hash function h generate a bucket address for the new record based on it hash key K. Bucket address = h(K)

- **Searching –** When a record needs to be searched, The same hash function is used to retrieve the bucket address for the record.
- **Updation –** The data record that needs to be updated is first searched using hash function, and then the data record is updated.
- **Deletion –** If we want to delete a record, using the hash function we will first fetch the record which is supposed to be deleted.
- Then we will remove the records for that address in memory.

If we want to insert some new record into the file but the address of a data bucket generated by the hash function is not empty, or data already exists in that address. This situation in the static hashing is known as **bucket overflow**. This is a critical situation in this method.

To overcome this situation, there are various methods. Some commonly used methods are as follows:

i. Open Hashing

ii. Close Hashing

i. Open Hashing

When a hash function generates an address at which data is already stored, then the next bucket will be allocated to it. This mechanism is called as **Linear Probing**.

For example: suppose R3 is a new address which needs to be inserted, the hash function generates address as 112 for R3. But the generated address is already full. So the system searches next available data bucket, 113 and assigns R3 to it as in fig 4.23.

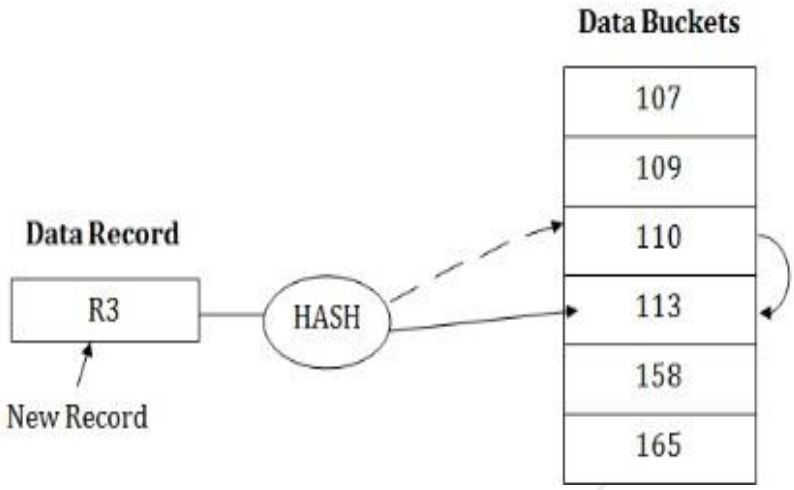


Fig 4.23 open Hashing

ii. Close Hashing

- When buckets are full, then a new data bucket is allocated for the same hash result and islinked after the previous one. This mechanism is known as **Overflow chaining**.

For example: Suppose R3 is a new address which needs to be inserted into the table, thehash function generates address as 110 for it. But this bucket is full to store the new data. In this case, a new bucket is inserted at the end of 110 buckets and is linked to it as in Fig 4.23.

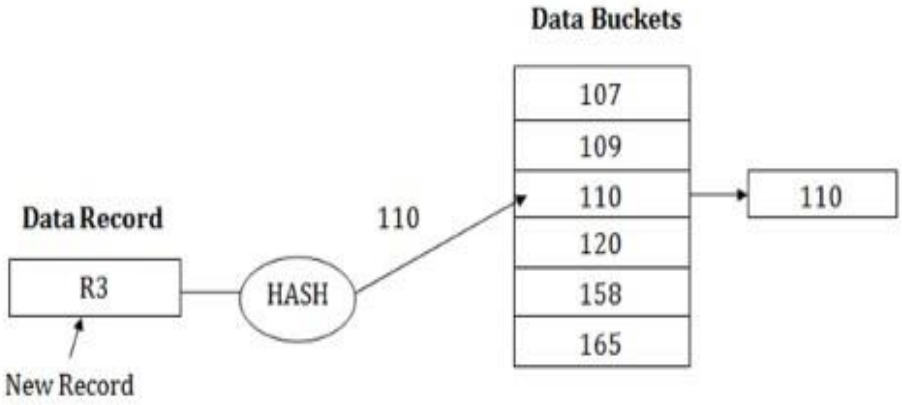


Fig 4.23 Close Hashing

2. Dynamic Hashing.

- 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.
- Dynamic hashing is also known as extended hashing.
- Hash function, in dynamic hashing, is made to produce a large number of values and only a few are used initially

Overflow Chaining – When buckets are full, a new bucket is allocated for the same hash result and is linked after the previous one.as in Fig 4.24

- This mechanism is called **Closed Hashing**.

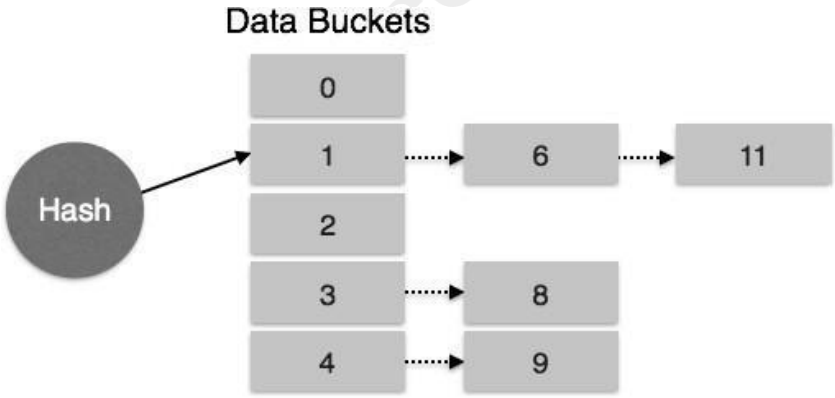


Fig 4.24 Overflow Chaining

- **Linear Probing** – When a hash function generates an address at which data is already stored, the next free bucket is allocated to it as in Fig 4.25.
- This mechanism is called Open Hashing.

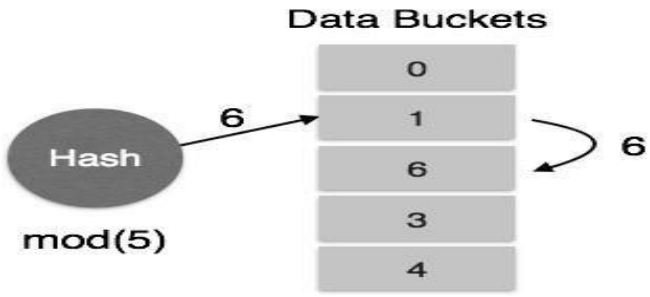


Fig 4.25 **Linear Probing**

Dynamic Hashing

- 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.
- Dynamic hashing is also known as extended hashing.
- Hash function, in dynamic hashing, is made to produce a large number of values and only a few are used initially

How to search a key

- First, calculate the hash address of the key.
- Check how many bits are used in the directory, and these bits are called as i.
- Take the least significant i bits of the hash address. This gives an index of the directory.
- Now using the index, go to the directory and find bucket address where the record might be.

How to insert a new record

- Firstly, you have to follow the same procedure for retrieval, ending up in some bucket.
- If there is still space in that bucket, then place the record in it.
- If the bucket is full, then we will split the bucket and redistribute the records.

For example:

Consider the following grouping of keys into buckets, depending on the prefix of their hash address:

Key	Hash address
1	11010
2	00000
3	11110
4	00000
5	01001
6	10101
7	10111

The last two bits of 2 and 4 are 00. So it will go into bucket B0. The last two bits of 5 and 6 are 01, so it will go into bucket B1. The last two bits of 1 and 3 are 10, so it will go into bucket B2. The last two bits of 7 are 11, so it will go into B3 as in Fig 4.26

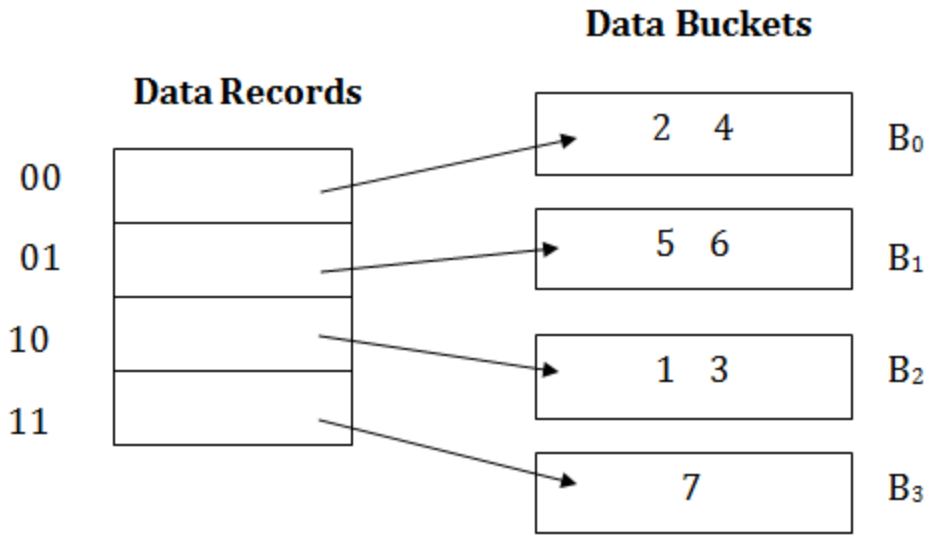


Fig 4.26 Insert a new record

Insert key 9 with hash address 10001 into the above structure:

- Since key 9 has hash address 10001, it must go into the first bucket. But bucket B1 is full, so it will get split as in fig 4.27.
- The splitting will separate 5, 9 from 6 since last three bits of 5, 9 are 001, so it will go into bucket B1, and the last three bits of 6 are 101, so it will go into bucket B5.
- Keys 2 and 4 are still in B0. The record in B0 pointed by the 000 and 100 entry because last two bits of both the entry are 00.

- Keys 1 and 3 are still in B2. The record in B2 pointed by the 010 and 110 entry because last two bits of both the entry are 10.
- Key 7 are still in B3. The record in B3 pointed by the 111 and 011 entry because last two bits of both the entry are 11.

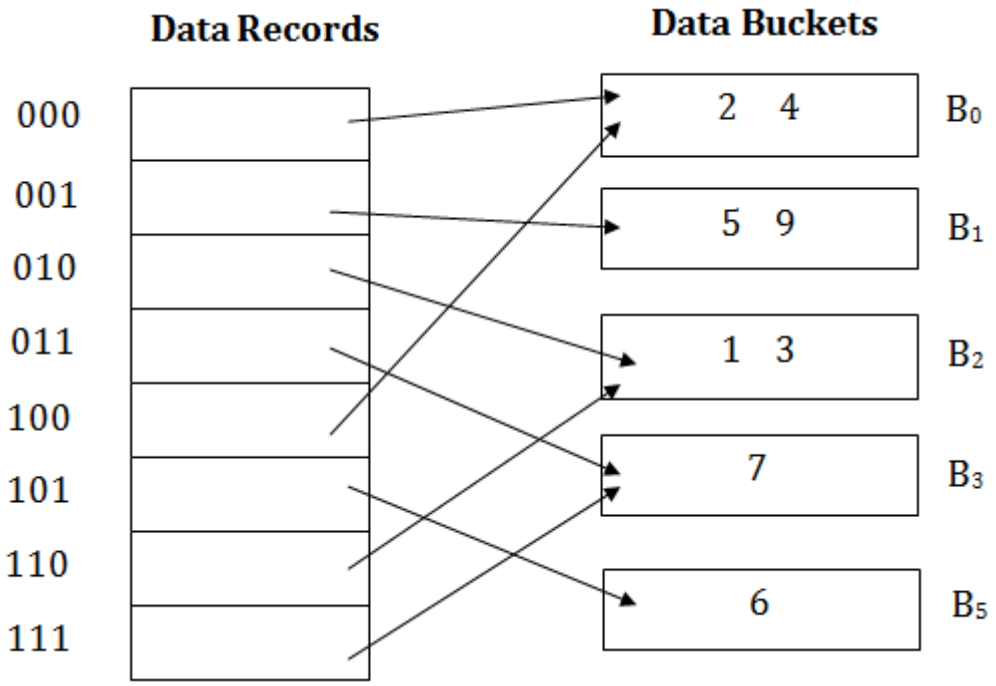


Fig 4.27 Insert key 9 with hash address 10001

Advantages of dynamic hashing

- In this method, the performance does not decrease as the data grows in the system. It simply increases the size of memory to accommodate the data.
- In this method, memory is well utilized as it grows and shrinks with the data. There will not be any unused memory lying.
- This method is good for the dynamic database where data grows and shrinks frequently.

Disadvantages of dynamic hashing

- In this method, if the data size increases then the bucket size is also increased. These addresses of data will be maintained in the bucket address table.
- This is because the data address will keep changing as buckets grow and shrink.
- If there is a huge increase in data, maintaining the bucket address table becomes tedious.
- In this case, the bucket overflow situation will also occur. But it might take little time to reach this situation than static hashing.

STATIC HASHING	DYNAMIC HASHING
Numbers of buckets are fixed.	Numbers of Buckets are not fixed.
As the file grows, performance decreases.	Performance do not degrade as the
Space overhead is more.	Minimum space lies overhead.
Here we do not use Bucket Address table.	Bucket address table is used.
No complex implementation.	Implementation is complex.
Chaining used is overflow chaining.	Overflow chaining is not used

6. Briefly explain about B+ tree index and B Tree file with example.

- B+ Tree is **an advanced method** of Indexed Sequential Access Method (ISAM) file organization.
- It uses the same **concept of key-index**, but in a tree like structure.
- B+ tree is **similar to binary search tree**, but it can have more than two leaf nodes.
- It stores all the **records only at the leaf node**.
- Intermediary nodes will have pointers to the leaf nodes.
- They do not contain any data/records.

The main goal of B+ tree is:

- Sorted Intermediary and leaf nodes
- Fast traversal and Quick Search
- No overflow pages

Searching a record in B+ Tree

- Suppose we want to search 65 in the below Fig 4.28 B+ tree structure.

- First we will fetch for the intermediary node which will direct to the leaf node that can contain record for 65.
- So we find branch between 50 and 75 nodes in the intermediary node.
- Then we will be redirected to the third leaf node at the end.
- Here DBMS will perform sequential search to find 65.

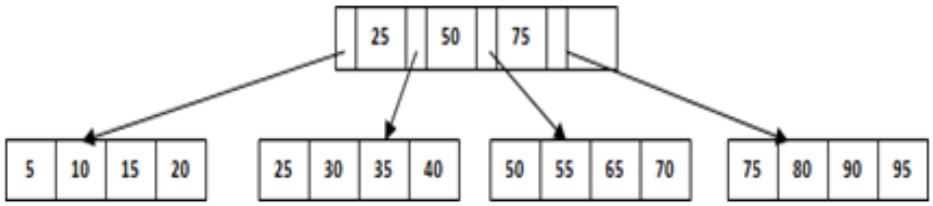


Fig 4.28 Searching a record in B+ Tree

Insertion in B+ tree:

- Suppose we have to insert a record 60 in the following example.

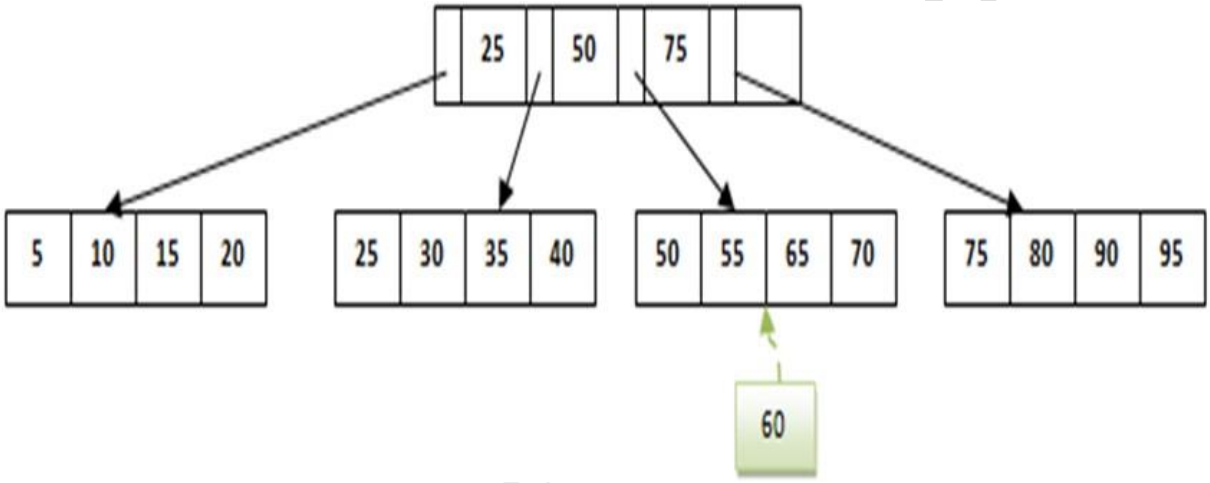


Fig 4.28 Inserting a record in B+ Tree

- As in Fig 4.28 ,It will go to 3rd leaf node after 55.
- Since it is a balanced tree and that leaf node is already full, we cannot insert the record there.

- But it should be inserted there without affecting the fill factor, balance and order.So the only option here is to split the leaf node.

- The 3rd leaf node should have values (50, 55, 60, 65, 70) and its current root node is 50. We will split the 3rd leaf node as (50, 55) and (60, 65, 70) as in Fig 4.29

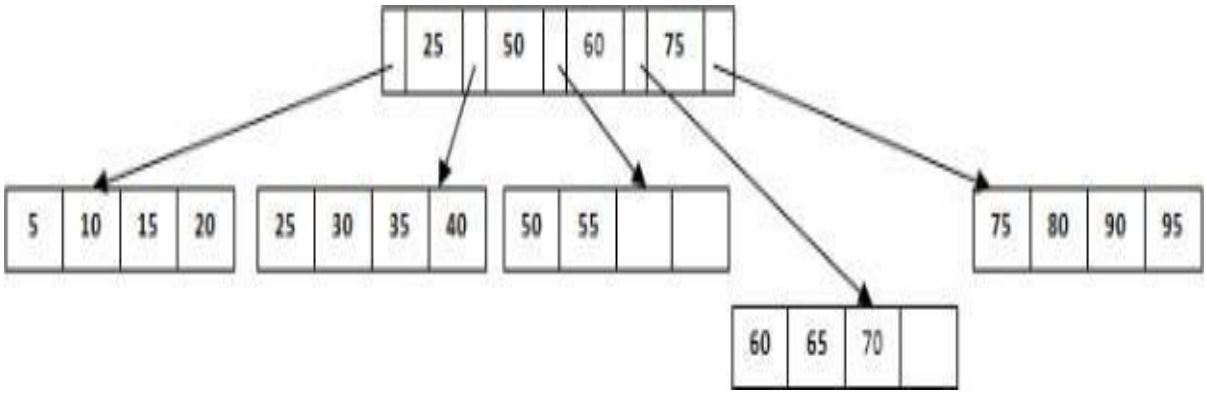


Fig 4.29 Inserting a record in B+ Tree (Leaf node is Full)

Delete in B+ tree:

- Suppose we have to delete 60 from the above example.
- What will happen in this case?
- We have to remove 60 from 4th leaf node as well as from the intermediary node too.
- So we need to modify it have a balanced tree.
- After deleting 60 from above B+ tree and re-arranging nodes, it will appear as below Fig 4.30.

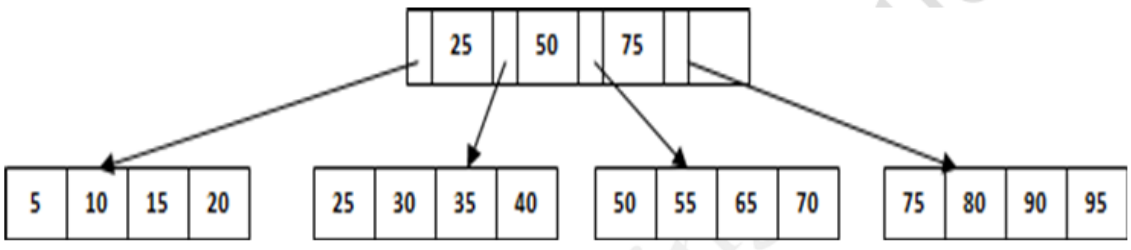


Fig 4.30 Deleting a record in B+ Tree

B TREE INDEX FILES

- B tree index file is similar to B+ tree index files, but it uses binary searchconcepts.
- In this method, each root will branch to only two nodes and each intermediary node will also have the data.
- And leaf node will have lowest level of data.
- However, in this method also, records will be sorted.
- Since all intermediary nodes also have records, it reduces the traversing till leaf node for the data.
- A simple B tree can be represented as below Fig 4.31

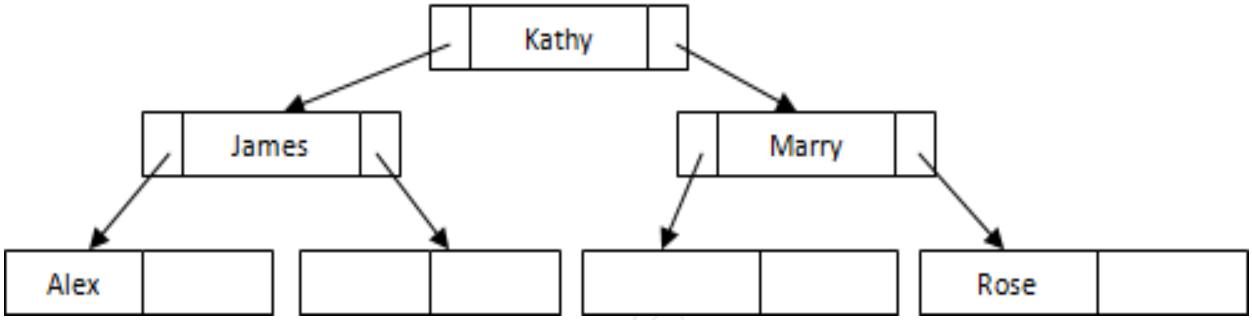


Fig 4.31 B Tree

See the difference between this tree structure and B+ tree for the same example above. Here there is no pointers leaf node.

- All the records are stored in all the nodes.
- If we need to insert any record, it will be done as B+ tree index files, but it will make sure that each node will branch only to two nodes.
- If there is not enough space in any of the node, it will split the node and store the records as in Fig 4.32 a,b.

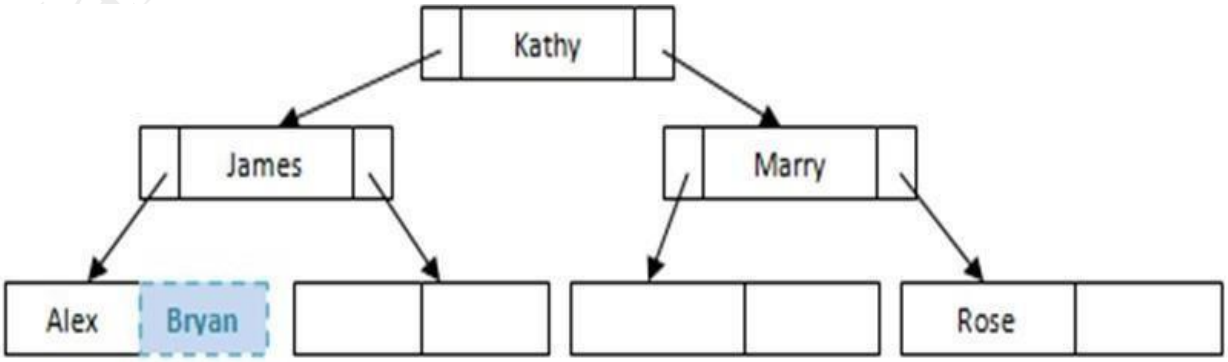


Fig 4.32(a) Inserting in B Tree

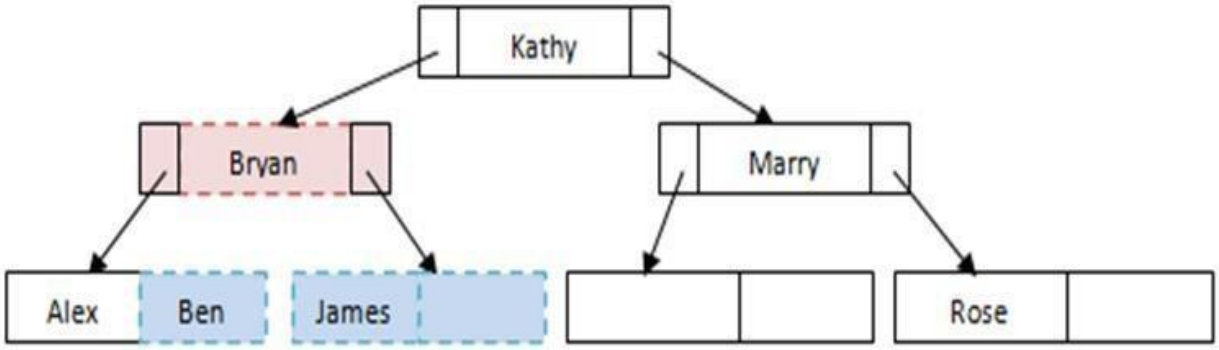


Fig 4.32(b) Inserting in B Tree

7. Explain query processing and query optimization with an example.

- Query processing describes how SQL queries are typically **translated into relational algebra** queries and then how it is optimized or **chooses better and optimal execution strategy**.
- Query Processing would mean the entire process or activity which involves query translation into low level instructions.

Goal:

a. To find an efficient Query Execution plan.

b. Minimize the cost significantly.

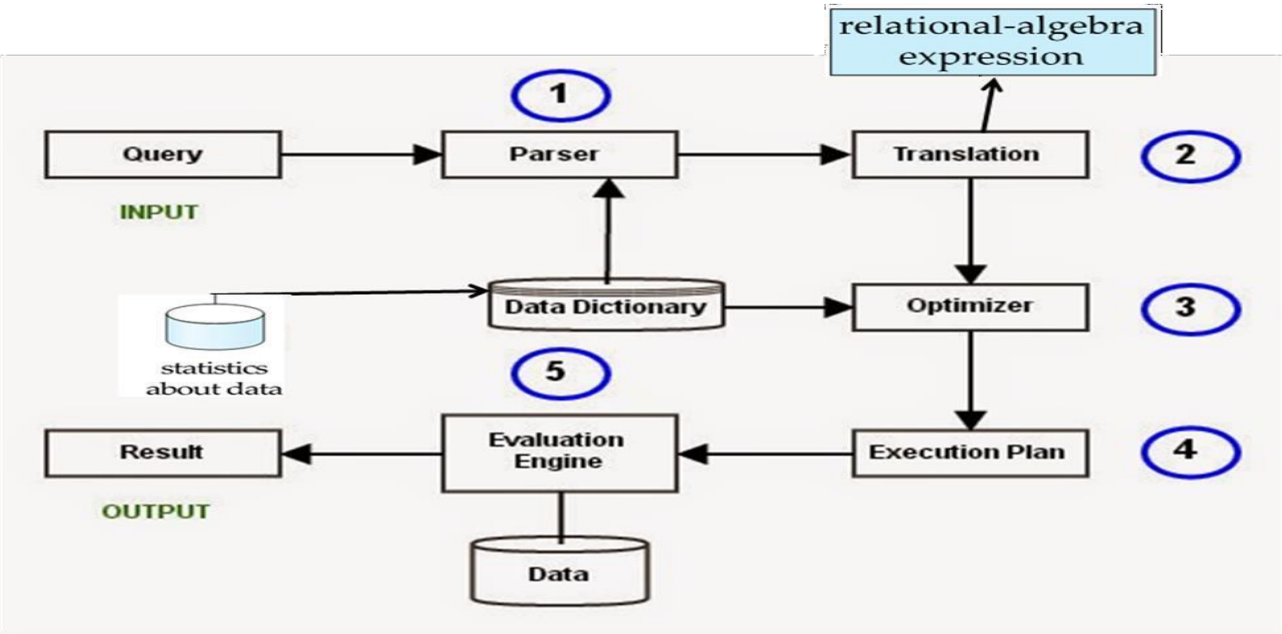
c. Minimize execution time.

Example:

Select student name whose CGPA is greater then 9 in the following Student table

S_id	Name	CGPA	Age
1001	Siva	8.5	21
1002	Kumar	9.2	21
1003	Guna	7.8	22
1004	Sekar	9.5	20
1005	Murgan	7.7	22

The major steps involved in query processing are depicted in the figure 4.33 below



SQL QUERY:

Select Name from Student where CGPA>9;

Parsing:

- In this step, the parser of the query processor module **checks the syntax of the query, the user's privileges** to execute the query, the table names and attribute names, etc.
- The **correct table name, attribute names** and the privilege of the users can be taken from the system catalog (data dictionary).

Translation:

- If we have written a valid query, then it is converted from high level language **SQL** to low level instruction in **Relational Algebra**.
- For example our SQL query can be converted into a relational algebra equivalent as follows:
  - $\pi_{(Name)}(\sigma_{(CGPA>9)} Student)$

(OR)

  - $\sigma_{(CGPA>9)}(\pi_{(Name)} Student)$

Optimizer:

- Optimizer **uses the statistical data** stored as part of data dictionary.
- The statistical data are information about the size of the table, the length of records, the indexes created on the table, etc.
- Optimizer also **checks for the conditions and conditional attributes** which are parts of the query.

Execution:

- A query can be **expressed in many ways**.
- The query processor module, at this stage, using the information collected in steps to **find different relational algebra expressions that are equivalent** and return the result of the one which we have written already.

- Optimizer **uses the statistical data** stored as part of data dictionary.
- The statistical data are information about the size of the table, the length of records, the indexes created on the table, etc.
- Optimizer also **checks for the conditions and conditional attributes** which are parts of the query.

Evaluation:

- Even if **there are many execution plans constructed** through statistical data, though **they return same result** (obvious), they differ in terms of Time consumption to execute the query, or the Space required executing the query.
- Hence, it is mandatory choose one plan which obviously **consumes less cost**. **Output:**
- The final result show to the user.

Name
Kumar
Sekar

QUERY PROCESSING AND OPTIMIZATION

- A query typically has many **possible execution strategies**, and the **process of choosing a suitable one** for processing a query is known as query optimization.
- Query optimization is the **process of selecting the most efficient query**-evaluationplan from among the many strategies.
- Amongst all equivalent evaluation plans choose the one with lowest cost.
  - Cost is estimated **using statistical information** from thedatabase catalog
    - e.g. number of tuples in each relation, size of tuples, etc
- Query optimization is process of describing the following.
  - How to measure query costs
  - Algorithms for evaluating relational algebra operations
  - How to combine algorithms for individual operations in order to evaluate acomplete expression.
- Cost is generally measured as total execution time for answering query
  - Many factors contribute to time cost
    - disk accesses, CPU, or even network communication

8.Explain Algorithms for Selection,Sorting and Join operations in detail.

Algorithm for Selection 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.
- The various algorithms used for SELECT Operation based in file scan

1.Algorithm Al : Linear Search .

Scan each file block and test all records to see whether they satisfy the selection condition

Cost = b_r block transfers + 1 seek

Where, br denotes number of blocks containing records from relation r

If selection is on a key attribute, can stop on finding record

Cost = (br/2) block transfers + 1 seek

2.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 = $[\log_2(b_r)x\ (t_r + t_s)$ 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.

Algorithm for Sorting Operation

External sorting

- In external sorting, the data stored on secondary memory is part by part loaded [in]to main memory, sorting can be done over there.
- The sorted data can be then stored in the intermediate files. Finally these [int]ermediate files can be merged repeatedly to get sorted data.
- Thus huge amount of data can be sorted using this technique.
- The external merge sort is a technique in which the data is loaded in intermediate files. Each intermediate file is sorted independently and then combined or merged to get the sorted data. For example : Consider that there are 10,000 records that has to be sorted. Clearly we need to apply external sorting method. Suppose main memory has a capacity to store 500 records in blocks, with each block size Of 100 records.The sorted 5 blocks (i.e. 500 records) are stored in intermediate file. This process will be repeated 20 times to get all the records sorted in chunks.
- In the second step, we start merging a pair of intermediate files in the main memory to get output file.

Algorithm 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:

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 Θ
  if Θ 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.

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 tx10 s

Let, r is called the outer relation and s the inner relation of the join.

```
For each block Br of r do
For each block Bs of s do
For each tuple tr in Br do begin
        For each tuple ts in Bs do begin
        test pair(tr,ts) to see if they satisfy the join condition o
        if o is satisfied, then add(tr,ts) to the result.
        End
    End
  End
End
```

Worst case estimate: br x bs + br block transfers + 2 x br seeks

Each block in the inner relation s is read once for each block in the outer relation.

Best case : br + bs block transfers + 2 seeks

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, br + bs block transfers + [br [bbl + [bs/bbl seeks + the cost of sorting, if relations are unsorted.

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, ..., n}, where A denotes the attributes of r and s used in the join.

Cost of hash join is :

3(br+ bs) +4 n block transfers +2([br/bb]+bs/bb]) seeks

If the entire build input can be kept in main memory no partitioning is required Cost estimate goes down to br + bs.

UNIT V

ADVANCED TOPICS

Distributed Databases: Architecture, Data Storage, Transaction Processing, Query processing and optimization – NOSQL Databases: Introduction – CAP Theorem – Document Based systems – Key value Stores – Column Based Systems – Graph Databases. Database Security: Security issues – Access control based on privileges – Role Based access control – SQL Injection – Statistical Database security – Flow control – Encryption and Public Key infrastructures – Challenges

PART – A

1. What is DDB?

Distributed database (DDB) as a collection of multiple logically interrelated databases distributed over a computer network, and a distributed data base management system (DDBMS) as a software system that manages a distributed database while making the distribution transparent to the user.

2. What are the conditions should be satisfied, a database to be called distributed?

- Connection of database nodes over a computer network
- Logical interrelation of the connected databases
- Absence of homogeneity constraint among connected nodes

3. List the advantages of DDB.

- Improved ease and flexibility of application development
- Increased reliability and availability
- Improved performance
- Easier expansion

4. What is a homogeneous distributed database?

In homogeneous distributed databases, all sites have identical database management system software, are aware of one another, and agree to cooperate in processing user's requests.

5. What is a heterogeneous distributed database?

In heterogeneous distributed database, different sites may use different schemas, and different database software. The sites may not be aware of one another, and they may provide only limited facilities for cooperation in transaction processing.

6. What are the two approaches to store relations in distributed database?

- Replication
- fragmenting

7. What are the two different schemes for fragmenting a relation?

- Horizontal
- Vertical

8. What is horizontal fragmentation?

Horizontal fragmentation splits the relation by assuming each tuple of r to one or more fragments.

9. What is vertical fragmentation?

Vertical fragmentation splits the relation by decomposing the scheme R of relation r.

10. What are the various forms of data transparency?

- Fragmentation transparency
- Replication transparency
- Location transparency

11. What is NoSQL and its uses?

- NoSQL, also referred to as "not only SQL", "non-SQL", is an approach to database design that enables the storage and querying of data outside the traditional structures found in relational databases.
- NoSQL databases use a variety of data models for accessing and managing data. These types of databases are optimized specifically for applications that require large data volume, low latency, and flexible data models, which are achieved by relaxing some of the data consistency restrictions of other databases.

12. What are the types of NoSQL databases?

- Document databases.
- Key-value stores.
- Column-oriented databases.
- Graph databases
- Hybrid systems
- Object databases
- XML databases

13. What are the characteristics of NOSQL database system?

- Scalability
- Availability
- Replication

- Eventual consistency
- Replication model
- Sharding of files
- High performance data access

14. Define CAP theorem.

The CAP theorem, originally introduced as the CAP principle, can be used to explain some of the competing requirements in a distributed system with replication. It is a tool used to make system designers aware of the trade-offs while designing networked shared-data systems.

The three letters in CAP refer to three desirable properties of distributed systems with replicated data:

- **consistency** (among replicated copies)
- **availability** (of the system for read and write operations)
- **partition tolerance** (in the face of the nodes in the system being partitioned by a network fault).

15. What is meant by document-based NoSQL system?

- Document databases are considered to be non-relational (or NoSQL) databases. Instead of storing data in fixed rows and columns, document databases use flexible documents.
- Document databases are the most popular alternative to tabular, relational databases.

16. What does key-value store?

- A key-value database stores data as a collection of key-value pairs in which a key serves as a unique identifier.
- Both keys and values can be anything, ranging from simple objects to complex compound objects.

17. What is meant by a column based database?

While a relational database is optimized for storing rows of data, typically for transactional applications, a columnar database is optimized for fast retrieval of columns of data, typically in analytical applications.

18. Is graph database SQL or NoSQL?

A graph database is a NoSQL database that stores data as a network graph. What differentiates graph databases from other options is that they document and prioritize the relationships between data.

19. What is mean by Database Security?

Security of databases refers to the array of controls, tools, and procedures designed to ensure and safeguard confidentiality, integrity, and accessibility. It will concentrate on confidentiality because it's a component that is most at risk in data security breaches.

20. List out various threats to database.

- Loss of integrity
- Loss of availability
- Loss of confidentiality

21. What are the control measures to ensure database security?

- Access control
- Flow control
- Inference control
- Data encryption

22. What is the role of access control?

- Access control is a fundamental component of data security that dictates who's allowed to access and use company information and resources.
- Through authentication and authorization, access control policies make sure users are who they say they are and that they have appropriate access to company data.

23. Which security issue access control based on granting and revoking privileges?

- In a discretionary access control system, each resource is assigned a unique owner account, which is responsible for managing access to that resource.
- The owner has the ability to grant or revoke access privileges, such as reading or modifying data, to other users or groups of users.

24. What is SQL injection in DBMS?

- SQL injection is a code injection technique that might destroy your database. SQL injection is one of the most common web hacking techniques. SQL injection is the placement of malicious code in SQL statements, via web page input.

25. What are the methods in SQL injection?

- SQL manipulation
- Code injection
- Function call injection

26. What is meant by statistical database security?

- Statistical database security focuses on the protection of sensitive individual values stored in so-called statistical databases and used for statistical purposes, as well as retrieving summaries of values based on categories.
- They do not allow the retrieval of individual information.

27. What is flow control in database?

Flow control is the management of data flow between computers or devices or between nodes in a network so that the data can be handled at an efficient pace.

28. What is PKI?

PKI (or Public Key Infrastructure) is the framework of encryption and cyber security that protects communications between the server (your website) and the client (the users).

29. List out the components of PKI.

- A digital certificate also called a public key certificate
- Private Key tokens
- Registration authority
- Certification authority
- CMS or Certification management system

30. What are the challenges in using public key encryption?

The challenges associated with public key cryptography are:

- It has been susceptible to attacks through spoofed or compromised certification authorities.
- Public key Encryption is vulnerable to Brute-force attack.
- This algorithm also fails when the user lost his private key, then the Public key Encryption becomes the most vulnerable algorithm.
- Public Key Encryption also is weak towards man in the middle attack.

31. What are the challenges in securing a database?

- Data quality
- Insider threats
- Confinement
- SQL injection
- Malware

32. What are the types of encryption?

1. **Symmetric Encryption**– Data is encrypted using a key and the decryption is also done using the same key.
2. **Asymmetric Encryption-** It uses public and private keys to encrypt and decrypt data. One key in the pair which can be shared with everyone is called the public key. The other key in the pair which is kept secret and is only known by the owner is called the private key. Either of the keys can be used to encrypt a message; the opposite key from the one used to encrypt the message is used for decryption.

33. Define Digital Signature.

- A Digital Signature (DS) is an authentication technique based on public key cryptography used in e-commerce applications.
- It associates a unique mark to an individual within the body of his message. This helps others to authenticate valid senders of messages.
- Typically, a user's digital signature varies from message to message in order to provide security against counterfeiting.

34. Define Digital Certificate.

- Digital certificate is issued by a trusted third party which proves sender's identity to the receiver and receiver's identity to the sender.
- A digital certificate is a certificate issued by a Certificate Authority (CA) to verify the identity of the certificate holder.
- The CA issues an encrypted digital certificate containing the applicant's public key and a variety of other identification information.
- Digital certificate is used to attach public key with a particular individual or an entity.

35. What are the impacts of SQL Injection?

- The intruder can retrieve all the user-data present in the database, such as user details, credit card information, and social security numbers, and can also gain access to protected areas like the administrator portal.
- It is also possible to delete the user data from the tables.
- These days all the online shopping applications, bank transactions use back-end database servers.
- If the intruder can exploit SQL injection, the entire server is compromised.

36. What is mean by Database Survivability?

- Database systems need to operate and continue their functions, even with reduced capabilities, despite disruptive events such as information warfare attacks.
- A DBMS, in addition to making every effort to prevent an attack and detecting one in the event of occurrence, should be able to do the following:
  - Confinement
  - Damage assessment
  - Reconfiguration
  - Repair
  - Fault treatment

37. What is RBAC?

Role-based access control (RBAC), also known as role-based security, is a mechanism that restricts system access. It involves setting permissions and privileges to enable access to authorized users.

38. Define Discretionary Access Control (DAC) in database.

- DAC is identity-based access control. DAC mechanisms will be controlled by user identification such as username and password.
- DAC is discretionary because the owners can transfer objects or any authenticated information to other users. In simple words, the owner can determine the access privileges.

39. What are the attributes of DAC?

a. Users can transfer their object ownership to another user.
b. The access type of other users can be determined by the user.
c. Authorization failure can restrict the user access after several failed attempts.
d. Unauthorized users will be blind to object characteristics called file size, directory path, and file name.

40. Define Mandatory access control (MAC) in database.

- Mandatory Access Control (MAC) is is a set of security policies constrained according to system classification, configuration and authentication.
- MAC policy management and settings are established in one secure network and limited to system administrators.
- MAC defines and ensures a centralized enforcement of confidential security policy parameters.

41. What are the attributes of MAC?

a. MAC policies can help to reduce system errors.
b. It has tighter security because only the administrator can access or alter controls.
c. MAC has an enforced operating system that can label and delineate incoming application data.
d. Maintenance will be difficult because only the administrator can have access to the database.

42. State the difference between DAC and MAC.

DAC	MAC
DAC stands for Discretionary Access Control.	MAC stands for Mandatory Access Control.
DAC is easier to implement.	MAC is difficult to implement.
DAC is less secure to use.	MAC is more secure to use.
DAC has complete trust in users.	MAC has trust only in administrators.
Information flow is impossible to control.	Information flow can be easily controlled.
DAC is supported by commercial DBMSs.	MAC is not supported by commercial DBMSs.

PART-B

1. Explain the concept of distributed database Management system.Discuss in detail about the Distributed databases? APR-2019

A **distributed database** is a collection of multiple interconnected databases, which are spread physically across various locations thatcommunicate via a computer network.

Features

- Databases in the collection are logically interrelated with each other. Often they represent a single logical database.
- Data is physically stored across multiple sites. Data in each site can bemanaged by a DBMS independent of the other sites.
- The processors in the sites are connected via a network. They do nothave any multiprocessor configuration.
- A distributed database is not a loosely connected file system.
- A distributed database incorporates transaction processing, but it is not synonymous with a transaction processing system.

Distributed Database Management System

A distributed database management system (DDBMS) is a centralized software system that manages a distributed database in a manner as if it were all stored in a single location.

Features

- It is used to create, retrieve, update and delete distributed databases.
- It synchronizes the database periodically and provides access mechanisms by the virtue of which the distribution becomes transparent to the users.
- It ensures that the data modified at any site is universally updated.
- It is used in application areas where large volumes of data are processed and accessed by numerous users simultaneously.
- It is designed for heterogeneous database platforms.
- It maintains confidentiality and data integrity of the databases.

Factors Encouraging DDBMS

The following factors encourage moving over to DDBMS −

- **Distributed Nature of Organizational Units** − Most organizations in the current times are subdivided into multiple units that are physically distributed over the globe. Each unit requires its own set of local data. Thus, the overall database of the organization becomes distributed.
- **Need for Sharing of Data** − The multiple organizational units often need to communicate with each other and share their data and resources. This demands common databases or replicated databases that should be used in a synchronized manner.

- **Support for Both OLTP and OLAP** − Online Transaction Processing (OLTP) and Online Analytical Processing (OLAP) work upon diversified systems which may have common data. Distributed database systems aid both these processing by providing synchronized data.
- **Database Recovery** − One of the common techniques used in DDBMS is replication of data across different sites.s Replication of data automatically helps in data recovery if database in any site is damaged. Users can access data from other sites while the damaged site is being reconstructed. Thus, database failure may become almost inconspicuousto users.
- **Support for Multiple Application Software** − Most organizations use a variety of application software each with its specific database support. DDBMS provides a uniform functionality for using the same data among different platforms.

Advantages of Distributed Databases

Following are the advantages of distributed databases over centralized databases.

- **Modular Development** − If the system needs to be expanded to newlocations or new units, in centralized database systems, the action requires substantial efforts and disruption in the existing functioning. However, in distributed databases, the work simply requires adding new computers and local data to the new site and finally connecting them tothe distributed system, with no interruption in current functions.
- **More Reliable** − In case of database failures, the total system ofcentralized databases comes to a halt. However, in distributed systems, when a component fails, the functioning of the system continues may be at a reduced performance. Hence DDBMS is more reliable.
- **Better Response** − If data is distributed in an efficient manner, then user requests can be met from local data itself, thus providing faster response. On the other hand, in centralized systems, all queries have to pass through the central computer for processing, which increases the response time.
- **Lower Communication Cost** − In distributed database systems, if data is located locally where it is mostly used, then the communication costs for data manipulation can be minimized. This is not feasible in centralized systems.

2. Explain in detailed about types of distributed databases.

Distributed databases can be broadly classified into

- Homogeneous distributed database
- Heterogeneous distributed

Homogeneous Distributed Databases

In a homogeneous distributed database, all the sites use identical DBMS and operating systems. Its properties are –

- The sites use very similar software.
- The sites use identical DBMS or DBMS from the same vendor.

- Each site is aware of all other sites and cooperates with other sites to process user requests.
- The database is accessed through a single interface as if it is a singledatabase.

Types of Homogeneous Distributed Database

There are two types of homogeneous distributed database −

- **Autonomous** − Each database is independent that functions on its own. They are integrated by a controlling application and use message passing to share data updates.
- **Non-autonomous** − Data is distributed across the homogeneousnodes and a central or master DBMS co-ordinates data updates across the sites.

Heterogeneous Distributed Databases

In a heterogeneous distributed database, different sites have different operating systems, DBMS products and data models. Its properties are −

- Different sites use dissimilar schemas and software.
- The system may be composed of a variety of DBMSs like relational, network, hierarchical or object oriented.
- Query processing is complex due to dissimilar schemas.
- Transaction processing is complex due to dissimilar software.
- A site may not be aware of other sites and so there is limited co- operation in processing user requests.

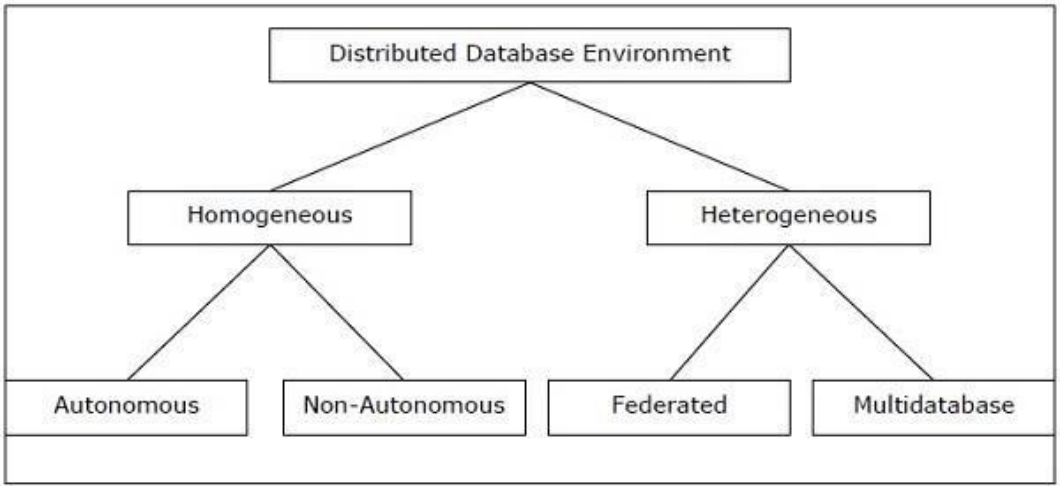


Fig. 5.1 Types of Distributed Databases

Types of Heterogeneous Distributed Databases

- **Federated** – The heterogeneous database systems are independent in nature and integrated together so that they function as a single database system.
- **Un-federated** – The database systems employ a central coordinating module through which the databases are accessed.

3. Explain briefly about the concept of Distributed DBMS Architectures APR-2019

DDBMS architectures are generally developed depending on threeparameters −

- **Distribution** − It states the physical distribution of data across the different sites.
- **Autonomy** − It indicates the distribution of control of the database system and the degree to which each constituent DBMS can operate independently.
- **Heterogeneity** − It refers to the uniformity or dissimilarity of the data models, system components and databases.

Architectural Models

Some of the common architectural models are −

- ❖ Client - Server Architecture for DDBMS
- ❖ Peer - to - Peer Architecture for DDBMS
- ❖ Multi - DBMS Architecture

Client - Server Architecture for DDBMS

This is a two-level architecture where the functionality is divided into servers and clients. The server functions primarily encompass data management, query processing, optimization and transaction management. Client functions include mainly user interface. However, they have some functions like consistency checking and transaction management.

The two different client - server architecture are −

- Single Server Multiple Client
- Multiple Server Multiple Client (shown in the following diagram)

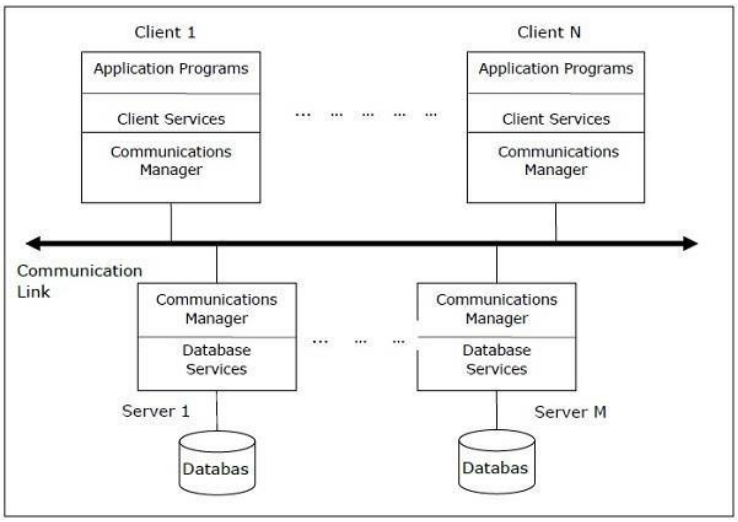


Fig. 5.2 Multiple Server Multiple Client architecture

Peer-to-Peer Architecture for DDBMS

In these systems, each peer acts both as a client and a server for imparting database services. The peers share their resource with other peers and co-ordinate their activities.

This architecture generally has four levels of schemas –

- **Global Conceptual Schema** – Depicts the global logical view of data.
- **Local Conceptual Schema** – Depicts logical data organization at eachsite.
- **Local Internal Schema** – Depicts physical data organization at eachsite.
- **External Schema** – Depicts user view of data.

Multi - DBMS Architectures

This is an integrated database system formed by a collection of two ormore autonomous database systems.

Multi-DBMS can be expressed through six levels of schemas –

- **Multi-database View Level** – Depicts multiple user views comprising ofsubsets of the integrated distributed database.
- **Multi-database Conceptual Level** – Depicts integrated multi-databasethat comprises of global logical multi-database structure definitions.
- **Multi-database Internal Level** – Depicts the data distribution acrossdifferent sites and multi-database to local data mapping.
- **Local database View Level** – Depicts public view of local data.
- **Local database Conceptual Level** – Depicts local data organization ateach site.
- **Local database Internal Level** – Depicts physical data organization ateach site.

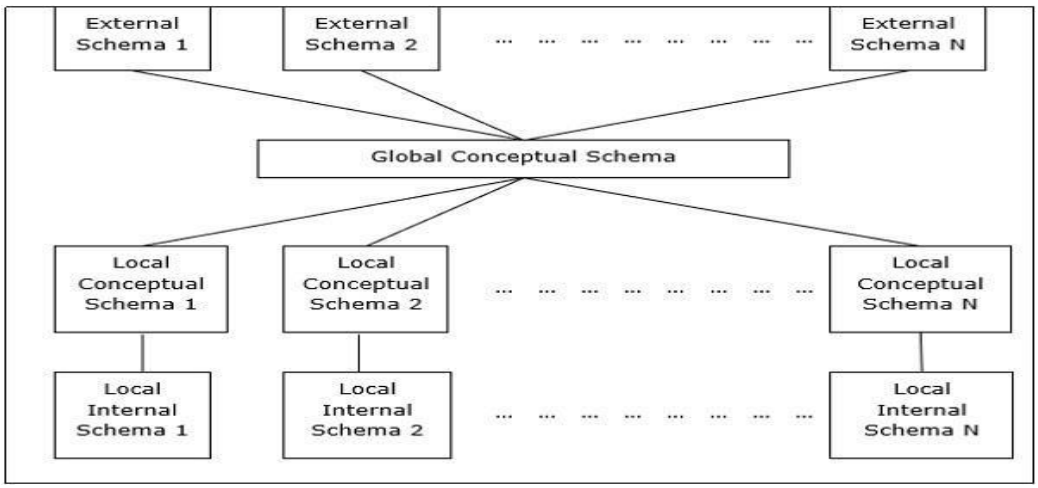


Fig. 5.3 Global Conceptual Schema

4. Write briefly about the Distributed Data Storage or Illustrate the approaches to store relations in distributed Database? (Nov/Dec 2019) (Nov/Dec-20)

There are 2 ways in which data can be stored on different sites. These are:

- Replication
- Fragmentation

Replication

Data replication is the process in which the data is copied at multiple locations (Different computers or servers) to improve the availability of data.

Goals of data replication

Data replication is done with an aim to:

- Increase the availability of data.
- Speed up the query evaluation.

Types of data replication

There are two types of data Replication :

1. **Synchronous Replication:**
   In synchronous replication, the replica will be modified immediately after some changes are made in the relation table. So there is no difference between original data and replica.
2. **Asynchronous replication:**
   In asynchronous replication, the replica will be modified after commit is fired on to the database.

Replication Schemes

The three replication schemes are as follows:

1. **Full Replication**
   In full replication scheme, the database is available to almost everylocation or user in communication network.

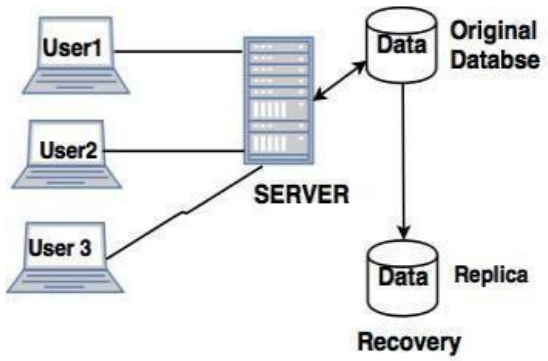


Fig. 5.4 Full Replication process in Distributed database

Advantages of full replication

- High availability of data, as database is available to almost everylocation.
- Faster execution of queries.

Disadvantages of full replication

- Concurrency control is difficult to achieve in full replication.
- Update operation is slower.

2. No Replication

No replication means, each fragment is stored exactly at one location.

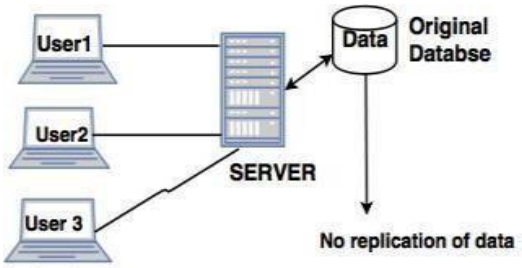


Fig. 5.5 No Replication in Distributed database

Advantages of no replication

- Concurrency can be minimized.
- Easy recovery of data.

Disadvantages of no replication

- Poor availability of data.
- Slows down the query execution process, as multiple clients are accessing the same server.

3. Partial replication

Partial replication means only some fragments are replicated from thedatabase.

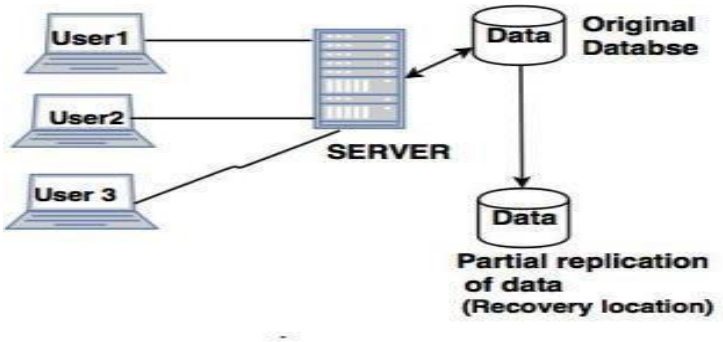


Fig. 5.6 Partial Replication in Distributed database

Advantages of Data Replication

- **Reliability** − In case of failure of any site, the database system continuesto work since a copy is available at another site(s).
- **Reduction in Network Load** − Since local copies of data are available, query processing can be done with reduced network usage, particularly during prime hours. Data updating can be done at non-prime hours.
- **Quicker Response** − Availability of local copies of data ensures quick query processing and consequently quick response time.
- **Simpler Transactions** − Transactions require less number of joins of tables located at different sites and minimal coordination across the network.

Disadvantages of Data Replication

- **Increased Storage Requirements** − Maintaining multiple copies of data is associated with increased storage costs. The storage space required is in multiples of the storage required for a centralized system.
- **Increased Cost and Complexity of Data Updating** − Each time a data item is updated, the update needs to be reflected in all the copies of the data at the different sites. This requires complex synchronization techniques and protocols.
- **Undesirable Application – Database coupling** − If complex update mechanisms are not used, removing data inconsistency requires complex co-ordination at application level. This results in undesirable application database coupling.

1. Fragmentation

The process of dividing the database into a smaller multiple parts is called as **fragmentation.** These fragments may be stored at different locations. The data fragmentation process should be carried out in such a way that thereconstruction of original database from the fragments is possible.

Types of data Fragmentation

There are three types of data fragmentation:

1. Horizontal data fragmentation –splitting by rows

Horizontal fragmentation divides a relation(table) horizontally into the group of rows to create subsets of tables.

For example, in the student schema, if the details of all students of Computer Science Course needs to be maintained at the School of Computer Science, then the designer will horizontally fragment the database as follows –

```
CREATE COMP_STD AS SELECT * FROM STUDENT WHERE COURSE
="Computer Science";
```

2. Vertical Fragmentation

Vertical fragmentation divides a relation(table) vertically into groups of columns to create subsets of tables.

For example, let us consider that a University database keeps records of all registered students in a Student table having the following schema.

STUDENT

Regd_No	Name	Course	Address	Semester	Fees	Marks

Now, the fees details are maintained in the accounts section. In this case, the designer will fragment the database as follows −

```
CREATE TABLE STD_FEES AS SELECT Regd_No,Fees FROM STUDENT;
```

3. Hybrid Fragmentation

Hybrid fragmentation can be achieved by performing horizontal and vertical partition together. Mixed fragmentation is group of rows and columns in relation.

Advantages of Fragmentation

- Since data is stored close to the site of usage, efficiency of the database system is increased.
- Local query optimization techniques are sufficient for most queries since data is locally available.
- Since irrelevant data is not available at the sites, security and privacy of the database system can be maintained.

Disadvantages of Fragmentation

- When data from different fragments are required, the access speeds may be very high.
- In case of recursive fragmentations, the job of reconstruction will need expensive techniques.
- Lack of back-up copies of data in different sites may render the database ineffective in case of failure of a site.

5. What is a transaction in distributed DBMS and explain about different states and properties of transaction?

A program that includes a collection of database operations which are executed as a logical unit of processing the data is known as a transaction.

In a transaction one or more of the data operations are performed such as insert, update, delete or retrieve. All this process is automated and if performed, is performed in completion or is not at all performed.

The transaction that does not include any updating of data but only involve in retrieving the data is known as read-only transaction.

A high level operation is divided into many low level operations. For instance, the operation of data updation is divided into three different low level operations -

- **read_item()** – The data items from the storage till the main memory is read.
- **modify_item()** – The value of the item in the main memory is changed
- **write_item()** – The modified value from the main memory is written to storage.

The access of the database is only restricted to the operations, read_item() and write_item(). Therefore, the basic database operations for any transaction include read and write.

What are the different Transaction Operations?

The operations that can be performed in a transaction at low level are:

- **begin_transaction** – The start of the transaction execution is specified by this marker.
- **read_item or write_item** – As a part of transaction, the operations of main memory are interleaved with the operations of the database.

- **end_transaction** – The end of the transaction is specified by this marker.
- **commit** – The successful completion of the transaction in its entirety is specified by this signal.
- **rollback** – The failure of the transaction and the status that the temporary changes are undone in the database is specified by this signal. Once the transaction is committed, it cannot be rolled back.

Different States of Transaction

There are set of five different states, which a transaction needs to go through. They are active, partially committed, committed, failed and aborted.

- **Active** − The initial state where the transaction enters is the active state. The transaction remains in this state while it is executing read, write or other operations.
- **Partially Committed** − The transaction enters this state after the last statement of the transaction has been executed.
- **Committed** − The transaction enters this state after successful completion of the transaction and system checks have issued commit signal.
- **Failed** − The transaction goes from partially committed state or active state to failed state when it is discovered that normal execution can no longer proceed or system checks fail.
- **Aborted** − This is the state after the transaction has been rolled back after failure and the database has been restored to its state that was before the transaction began.

The diagram demonstrates the different states of the transaction and also the low level transaction operations that may lead to states being changed.

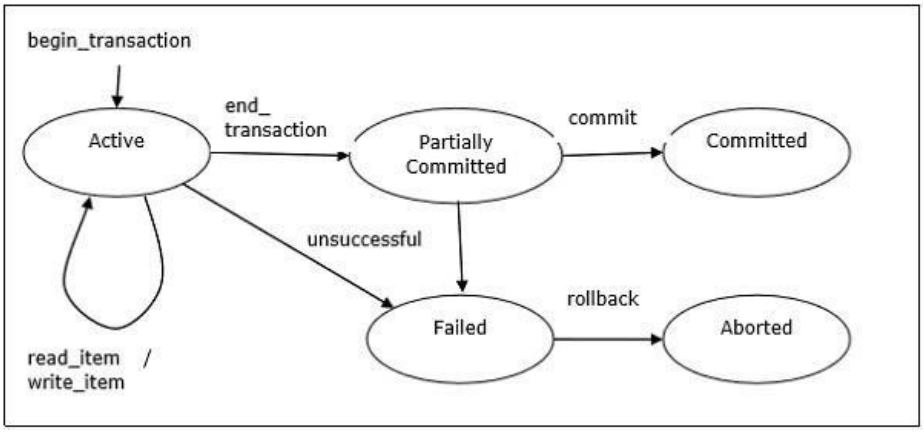


Fig. 5.7 different states of the transaction

Desirable properties of Transactions

The properties such as Atomicity, Consistency, Isolation and Durability, known as ACID need to be maintained by any of the transaction.

Any transaction must maintain the ACID properties, viz. Atomicity, Consistency, Isolation, and Durability.

- **Atomicity** − This property states that a transaction is an atomic unit of processing, that is, either it is performed in its entirety or not performed at all. No partial update should exist.
- **Consistency** − A transaction should take the database from one consistent state to another consistent state. It should not adversely affect any data item in the database.

- **Isolation** − A transaction should be executed as if it is the only one in the system. There should not be any interference from the other concurrent transactions that are simultaneously running.
- **Durability** − If a committed transaction brings about a change, that change should be durable in the database and not lost in case of any failure.

Schedules in Distributed DBMS

In a system with a number of simultaneous transactions, a **schedule** is the total order of execution of operations. Given a schedule S comprising of n transactions, say T1, T2, T3……..Tn; for any transaction Ti, the operations in Ti must execute as laid down in the schedule S.

Types of Schedules

There are two types of schedules −

- **Serial Schedules** − In a serial schedule, at any point of time, only one transaction is active, i.e. there is no overlapping of transactions. This is depicted in the following graph –

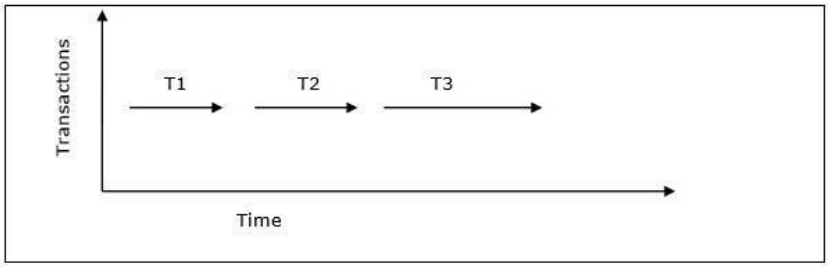


Fig. 5.8 Serial Schedule

- **Parallel Schedules** – In parallel schedules, more than one transactions are active simultaneously, i.e. the transactions contain operations that overlap at time. This is depicted in the following graph

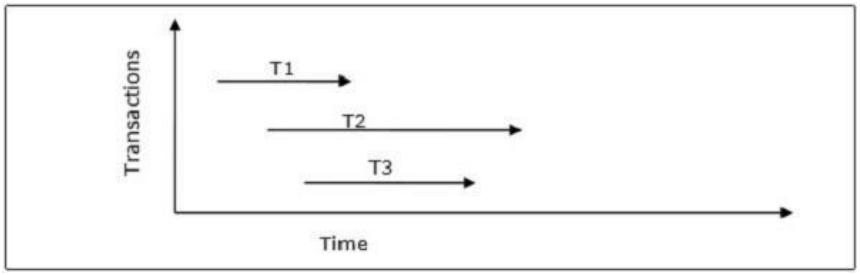


Fig. 5.9 Parallel Schedule

Conflicts in Schedules

In a schedule comprising of multiple transactions, a **conflict** occurs when two active transactions perform non-compatible operations.

Two operations are said to be in conflict, when all of the following three conditions exists simultaneously: The two operations are parts of different transactions.

- Both the operations access the same data item.
- At least one of the operations is a write_item() operation, i.e. it tries to modify the data item.

6. Explain in detail about Query processing in database with example.

Query Processing is a translation of high-level queries into low-level expression. It is the activity performed in extracting data from the database. Query processing is a step wise process that can be used at the physical level of the file system, query optimization and actual execution of the query to get the result. In query processing, it takes various steps for fetching the data from the database.

The steps involved are:

i. Parsing and translation
ii. Optimization
iii. Evaluation

Steps in Query Processing

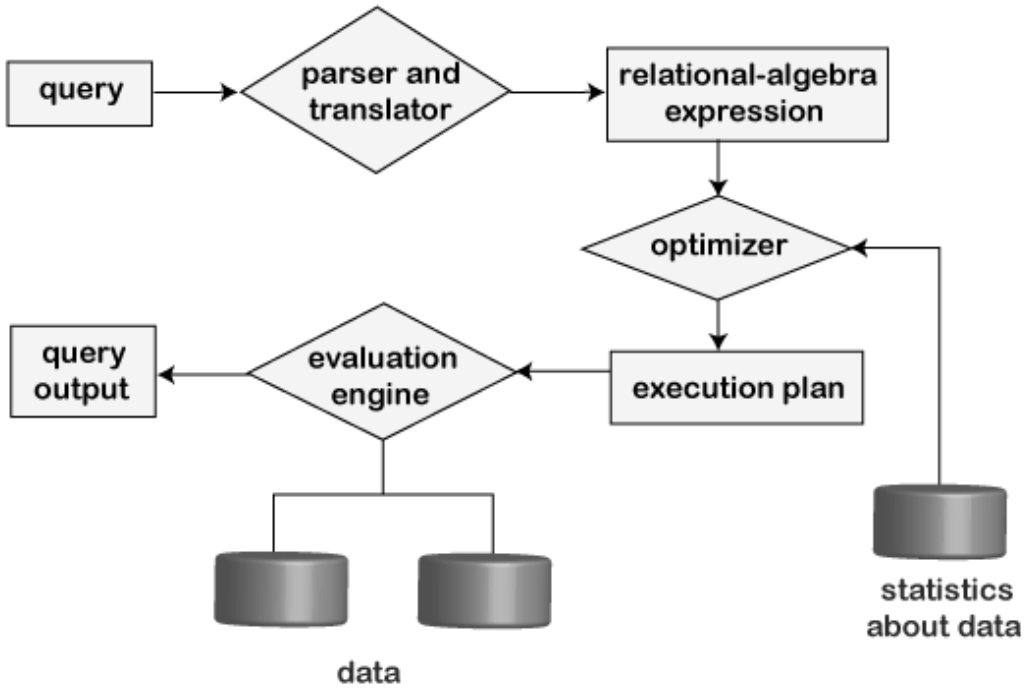


Fig. 5.10 Query processing in Database

Parsing and Translation

- The scanning, parsing, and validating module produces an internal representation of the query. The query optimizer module devises an execution plan which is the execution strategy to retrieve the result of the query from the database files.
- A query typically has many possible execution strategies differing in performance, and the process of choosing a reasonably efficient one is known as query optimization.
- The code generator generates the code to execute the plan. The runtime database processor runs the generated code to produce the query result. Relational algebra is well suited for the internal representation of a query.
- The translation process in query processing is similar to the parser of a query. When a user executes any query, for generating the internal form of the query, the parser in the system checks the syntax of the query, verifies the name of the relation in the database, the tuple, and finally the required attribute value.

- The parser creates a tree of the query, known as 'parse-tree.' Further, translate it into the form of relational algebra. With this, it evenly replaces all the use of the views when used in the query.
- It is done in the following steps:

Step-1:

Parser:

During parse call, the database performs the following checks- Syntax check, Semantic check and Shared pool check, after converting the query into relational algebra.

1. Syntax check – concludes SQL syntactic validity.

Example: SELECT * FORM employee

Here error of wrong spelling of FROM is given by this check.

2. Semantic check – determines whether the statement is meaningful or not.

Example: query contains a table name which does not exist is checked by this check.

3. Shared Pool check – Every query possess a hash code during its execution. So, this check determines existence of written hash code in shared pool if code exists in shared pool then database will not take additional steps for optimization and execution.

Hard Parse and Soft Parse:

- If there is a fresh query and its hash code does not exist in shared pool then that query has to pass through from the additional steps known as hard parsing otherwise if hash code exists then query does not passes through additional steps.
- It just passes directly to execution engine (refer detailed diagram).
- This is known as soft parsing.
- Hard Parse includes following steps – Optimizer and Row source generation.

Step-2:

Optimizer:

- During optimization stage, database must perform a hard parse at least for one unique DML statement and perform optimization during this parse.
- This database never optimizes DDL unless it includes a DML component such as sub-query that require optimization.
- It is a process in which multiple query execution plan for satisfying a query are examined and most efficient query plan is satisfied for execution.
- Database catalog stores the execution plans and then optimizer passes the lowest cost plan for execution.

Step-3:

Execution Engine:

- Finally runs the query and display the required result.
- Suppose a user executes a query. As we have learned that there are various methods of extracting the data from the database.
- In SQL, a user wants to fetch the records of the employees whose salary is greater than or equal to 10000.

- For doing this, the following query is undertaken:

SELECT EMP_NAME FROM EMPLOYEE WHERE SALARY>10000;

Thus, to make the system understand the user query, it needs to be translated in the form of relational algebra. We can bring this query in the relational algebra form as:

- o **σsalary>10000 (πEmp_Name(Employee))**
- o **πEmp_Name(σsalary>10000 (Employee))**

After translating the given query, we can execute each relational algebra operation by using different algorithms. So, in this way, a query processing begins its working.

Evaluation

- For this, with addition to the relational algebra translation, it is required to annotate the translated relational algebra expression with the instructions used for specifying and evaluating each operation. Thus, after translating the user query, the system executes a query evaluation plan.

Query Evaluation Plan

- In order to fully evaluate a query, the system needs to construct a query evaluation plan.
- A query evaluation plan defines a sequence of primitive operations used for evaluating a query. The query evaluation plan is also referred to as **the query execution plan**.
- A **query execution engine** is responsible for generating the output of the given query. It takes the query execution plan, executes it, and finally makes the output for the user query.

Optimization

- The cost of the query evaluation can vary for different types of queries. Although the system is responsible for constructing the evaluation plan, the user does need not to write their query efficiently.
- Usually, a database system generates an efficient query evaluation plan, which minimizes its cost. This type of task performed by the database system and is known as Query Optimization.
- For optimizing a query, the query optimizer should have an estimated cost analysis of each operation. It is because the overall operation cost depends on the memory allocations to several operations, execution costs, and so on.
- Finally, after selecting an evaluation plan, the system evaluates the query and produces the output of the query.

Example:

SELECT LNAME, FNAME FROM EMPLOYEE WHERE SALARY > (SELECT MAX (SALARY) FROM EMPLOYEE WHERE DNO=5);

The inner block

(SELECT MAX (SALARY) FROM EMPLOYEE WHERE DNO=5)

- ➢ Translated in: Π MAX SALARY (σDNO=5(EMPLOYEE))

The Outer block

SELECT LNAME, FNAME FROM EMPLOYEE WHERE SALARY > C

- Translated in: Π LNAZME, FNAME (σSALARY>C (EMPLOYEE))

(C represents the result returned from the inner block.)

- The query optimizer would then choose an execution plan for each block.
- The inner block needs to be evaluated only once. (Uncorrelated nested query).
- It is much harder to optimize the more complex correlated nested queries.

7. Explain about NOSQL database. Discuss how it evolved and describe about its characteristics, categories, advantages & disadvantages with an example.

- NoSQL is a type of database management system (DBMS) that is designed to handle and store large volumes of unstructured and semi-structured data.
- Unlike traditional relational databases that use tables with pre-defined schemas to store data, NoSQL databases use flexible data models that can adapt to changes in data structures and are capable of scaling horizontally to handle growing amounts of data.
- The term **NoSQL** originally referred to "**non-SQL**" or "**non-relational**" databases, but the term has since evolved to mean "**not only SQL**," as NoSQL databases have expanded to include a wide range of different database architectures and data models.
- It provides a mechanism for storage and retrieval of data other than tabular relations model used in relational databases.
- NoSQL database doesn't use tables for storing data. It is generally used to store big data and real-time web applications.

History behind the evolution of NoSQL Databases:

- In the early 1970, Flat File Systems are used. Data were stored in flat files and the biggest problems with flat files are each company implement their own flat files and there are no standards.
- It is very difficult to store data in the files, retrieve data from files because there is no standard way to store data.
- Then the relational database was created by **E.F. Codd** and these databases answered the question of having no standard way to store data.
- But later relational database also get a problem that it could not handle big data, due to this problem there was a need of database which can handle every types of problems then NoSQL database was developed.

Characteristics of NoSQL databases:

1. **Dynamic schema:** NoSQL databases do not have a fixed schema and can accommodate changing data structures without the need for migrations or schema alterations.
2. **Horizontal scalability:** NoSQL databases are designed to scale out by adding more nodes to a database cluster, making them well-suited for handling large amounts of data and high levels of traffic.

3. **Document-based:** Some NoSQL databases, such as MongoDB, use a document-based data model, where data is stored in semi-structured format, such as JSON or BSON.
4. **Key-value-based:** Other NoSQL databases, such as Redis, use a key-value data model, where data is stored as a collection of key-value pairs.
5. **Column-based:** Some NoSQL databases, such as Cassandra, use a column-based data model, where data is organized into columns instead of rows.
6. **Distributed and high availability:** NoSQL databases are often designed to be highly available and to automatically handle node failures and data replication across multiple nodes in a database cluster.
7. **Flexibility:** NoSQL databases allow developers to store and retrieve data in a flexible and dynamic manner, with support for multiple data types and changing data structures.
8. **Performance:** NoSQL databases are optimized for high performance and can handle a high volume of reads and writes, making them suitable for big data and real-time applications.

The main types of NoSQL database:

- Document-based databases
- Key-value stores
- Column-based databases
- Graph-based databases

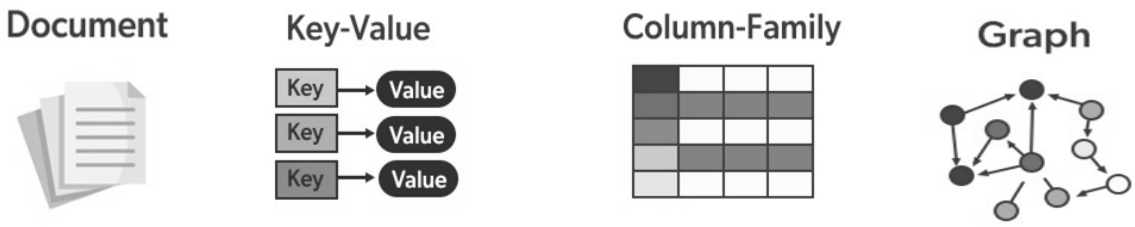


Fig. 5.11 Categories of NoSQL database

➢ **Document-Based Database:**

- The document-based database is a nonrelational database. Instead of storing the data in rows and columns (tables), it uses the documents to store the data in the database. A document database stores data in JSON, BSON, or XML documents.
- Documents can be stored and retrieved in a form that is much closer to the data objects used in applications which means less translation is required to use these data in the applications. In the Document database, the particular elements can be accessed by using the index value that is assigned for faster querying.
- Collections are the group of documents that store documents that have similar contents. Not all the documents are in any collection as they require a similar schema because document databases have a flexible schema.
- Document-oriented NoSQL database solutions include MongoDB, CouchDB, Riak, Amazon SimpleDB, and Lotus Notes.

Key features of documents database:

- Flexible schema: Documents in the database has a flexible schema. It means the documents in the database need not be the same schema.
- Faster creation and maintenance: the creation of documents is easy and minimal maintenance is required once we create the document.
- No foreign keys: There is no dynamic relationship between two documents so documents can be independent of one another. So, there is no requirement for a foreign key in a document database.
- Open formats: To build a document we use XML, JSON, and others.

➢ **Key-Value Stores:**

- A key-value store is a nonrelational database. The simplest form of a NoSQL database is a key-value store. Every data element in the database is stored in key-value pairs.
- The data can be retrieved by using a unique key allotted to each element in the database. The values can be simple data types like strings and numbers or complex objects.
- A key-value store is like a relational database with only two columns which is the key and the value.

Example: Key-value NoSQL solutions include Dynamo, Redis, Riak, Tokyo Cabinet/Tyrant, Voldemort, Amazon SimpleDB, and Oracle BDB.

Key features of the key-value store:

- Simplicity.
- Scalability.
- Speed.

➢ **Column based Databases:**

- A column-oriented database is a non-relational database that stores the data in columns instead of rows.
- That means when we want to run analytics on a small number of columns, you can read those columns directly without consuming memory with the unwanted data.
- Columnar databases are designed to read data more efficiently and retrieve the data with greater speed.
- A columnar database is used to store a large amount of data.

Examples of column-based NoSQL databases include Cassandra, HBase, and Hypertable.

Key features of columnar oriented database:

- Scalability.
- Compression.
- Very responsive.

➢ **Graph-Based databases:**

- Graph-based databases focus on the relationship between the elements.

- It stores the data in the form of nodes in the database. The connections between the nodes are called links or relationships.
- Graph-based NoSQL database solutions include Neo4J, Infinite Graph, and FlockDB.

Key features of graph database:

- In a graph-based database, it is easy to identify the relationship between the data by using the links.
- The Query's output is real-time results.
- The speed depends upon the number of relationships among the database elements.

Advantages of NoSQL database:

- High scalability
- Availability
- Flexibility
- Performance
- Cost-effectiveness

Disadvantages of NoSQL database:

- Lack of standardization
- Lack of ACID compliance
- Narrow focus
- Management challenges
- No GUI
- Backup
- Large document size
- Lack of Support for complex queries

8. Define CAP theorem. Discuss in detail about CAP theorem in NoSQL databases.

- The CAP theorem, originally introduced as the CAP principle, can be used to explain some of the competing requirements in a distributed system with replication. It is a tool used to make system designers aware of the trade-offs while designing networked shared-data systems.
- The three letters in CAP refer to three desirable properties of distributed systems with replicated data:
  - **Consistency** (among replicated copies),
  - **Availability** (of the system for read and write operations)
  - **Partition tolerance** (in the face of the nodes in the system being partitioned by a network fault).

The CAP theorem states that it is not possible to guarantee all three of the desirable properties – consistency, availability, and partition tolerance at the same time in a distributed system with data replication.

The theorem states that networked shared-data systems can only strongly support two of the following three properties:

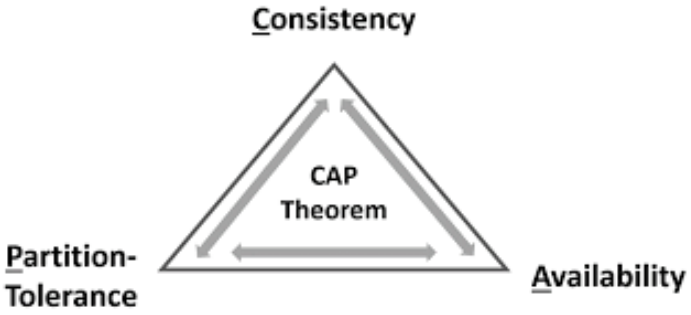


Fig. 5.12 Triangle properties of CAP theorem

Consistency:

- Consistency means that the nodes will have the same copies of a replicated data item visible for various transactions.
- A guarantee that every node in a distributed cluster returns the same, most recent and a successful write. Consistency refers to every client having the same view of the data.
- There are various types of consistency models. Consistency in CAP refers to sequential consistency, a very strong form of consistency.

Availability:

- Availability means that each read or write request for a data item will either be processed successfully or will receive a message that the operation cannot be completed.
- Every non-failing node returns a response for all the read and write requests in a reasonable amount of time.
- The key word here is “every”. In simple terms, every node (on either side of a network partition) must be able to respond in a reasonable amount of time.

Partition Tolerance:

- Partition tolerance means that the system can continue operating even if the network connecting the nodes has a fault that results in two or more partitions, where the nodes in each partition can only communicate among each other.
- That means, the system continues to function and upholds its consistency guarantees in spite of network partitions.
- Network partitions are a fact of life. Distributed systems guaranteeing partition tolerance can gracefully recover from partitions once the partition heals.
- The CAP theorem states that distributed databases can have at most two of the three properties: consistency, availability, and partition tolerance.
- As a result, database systems prioritize only two properties at a time.

The following figure represents which database systems prioritize specific properties at a given time:

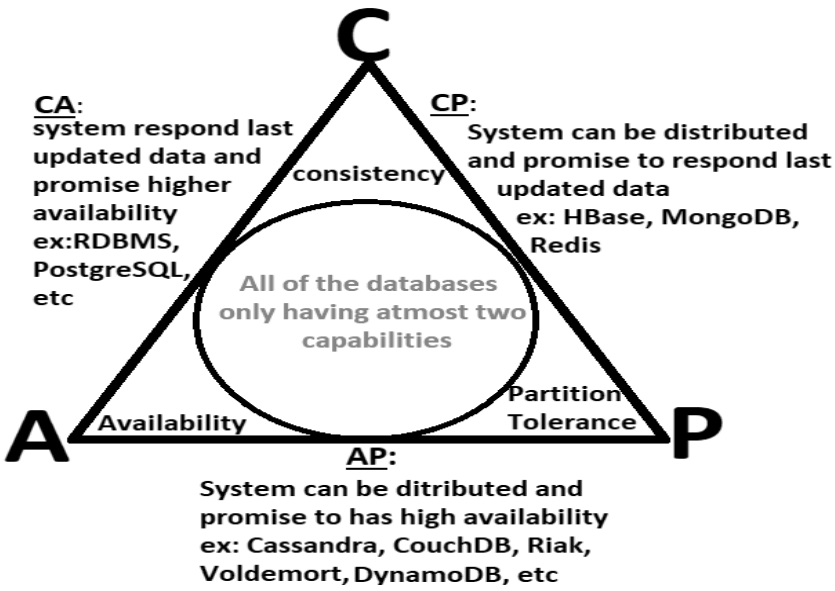


Fig. 5.13 Priorities of database properties

- **CA(Consistency and Availability) -** The system prioritizes availability over consistency and can respond with possibly stale data.
  Example databases: Cassandra, CouchDB, Riak, Voldemort.

- **AP(Availability and Partition Tolerance) -** The system prioritizes availability over consistency and can respond with possibly stale data. The system can be distributed across multiple nodes and is designed to operate reliably even in the face of network partitions.
  Example databases: Amazon DynamoDB, Google Cloud Spanner.

- **CP(Consistency and Partition Tolerance) -** The system prioritizes consistency over availability and responds with the latest updated data. The system can be distributed across multiple nodes and is designed to operate reliably even in the face of network partitions.
  Example databases: Apache HBase, MongoDB, Redis.

9. Explain briefly any two examples of NoSQL database.

i) MongoDB ii) Oracle NoSQL

✓ **MongoDB**

- MongoDB is one of the foremost open-source NoSQL systems. It is a document-oriented database that uses dynamic schemas to store JSON-like documents.
- This database solution features a flexible data model, enabling users to store unstructured data.
- Users can also access full indexing support and replication through intuitive API.
- MongoDB's popularity among developers includes its flexible data model and intuitive API.

Why is MongoDB used?

- **Storage.** MongoDB can store large structured and unstructured data volumes and is scalable vertically and horizontally. Indexes are used to improve search performance. Searches are also done by field, range and expression queries.
- **Data integration**. This integrates data for applications, including for hybrid and multi-cloud applications.
- **Complex data structures descriptions**. Document databases enable the embedding of documents to describe nested structures (a structure within a structure) and can tolerate variations in data.
- **Load balancing.** MongoDB can be used to run over multiple servers.

Key features of MongoDB includes:

- **Ad-hoc queries:** Support for range, field, and regular-expression queries that are capable of returning complete documents, particular fields from inside documents, and even random result samples.
- **Replication:** High availability is achieved through replica sets, including multiple data copies. The primary replica handles writes, and any replica can serve read requests. In case of primary replica failure, a secondary replica takes over as the primary replica.
- **Indexing:** Support for different index types, including single field, multikey (array), compound (multiple fields), geospatial, hashed, and text. Document fields can be indexed using both primary and secondary indices.

Advantages of using MongoDB:

- Schema-less
- Document oriented
- Scalability
- Third-party support
- Aggregation

Disadvantages in MongoDB:

- Continuity
- Write limits
- Data consistency
- Security

✓ **Oracle NoSQL Database**

This proprietary NoSQL database supports key-value and JSON table data models and is built to operate either on-premise or over the cloud. Developers leverage Oracle NoSQL Database Cloud Service to create applications using column, document, and key-value data models.

The characteristics of Oracle NoSQL Database include:

- ACID transactions, comprehensive security, low pay-per-use pricing, and serverless scaling.
- Delivery of predictable, within-milliseconds response times and high availability through robust data replication.

- Support for both provisioned and on-demand capacity modes.
- Simplified access through a user-friendly application programming interface (API).
- Wide support for data models for various business requirements.
- End-to-end compatibility with on-premise Oracle NoSQL Database.

Key features of Oracle Database:

- Scalability and Performance
- Manageability Features
- Database Backup and Recovery
- High Availability
- Business Intelligence
- Content Management
- Security Features
- Data Integrity and Triggers
- Information Integration

Advantages of Oracle NoSQL database:

- Data-model flexibility. Unlike RDBMS solutions, Oracle NoSQL does not restrict you to a predefined set of data types.
- Ability to Handle an Increased Amount of Traffic. As Oracle NoSQL can process queries much quicker than Oracle Database, Oracle NoSQL is able to respond to a lot more queries in the same amount of time.
- Data-model simplicity. In SQL-oriented databases, there is a learning curve in learning the relationship between databases, tables, rows, and keys.

Disadvantages of Oracle NoSQL database:

- Fewer analytical functions to choose from. When compared to Oracle Database, there is significant difference in the amount of built-in analytical functions.
- Eventual data consistency. It is not guaranteed that a write or delete query will be immediately visible for subsequent queries.
- Data redundancy. As there are no mechanisms that insure data integrity, users are more likely to have redundant data across their documents.

10. Explain about document based NoSQL database in detail.

Document Database Model:

- A Document Data Model is a lot different than other data models because it stores data in JSON, BSON, or XML documents.
- In this data model, we can move documents under one document and apart from this, any particular elements can be indexed to run queries faster.
- JSON is a native language that is often used to store and query data too.

- Often documents are stored and retrieved in such a way that it becomes close to the data objects which are used in many applications which means very less translations are required to use data in applications.
- So in the document data model, each document has a key-value pair below is an example for the same.

Example:

```
{
"Name" : "Yashodhra",
"Address" : "Near Patel Nagar",
"Email" : "yahoo123@yahoo.com",
"Contact" : "12345"
}
```

Working of Document Data Model:

This is a data model which works as a semi-structured data model in which the records and data associated with them are stored in a single document which means this data model is not completely unstructured. The main thing is that data here is stored in a document.

Features:

- **Document Type Model:** As we all know data is stored in documents rather than tables or graphs, so it becomes easy to map things in many programming languages.
- **Flexible Schema:** Overall schema is very much flexible to support this statement one must know that not all documents in a collection need to have the same fields.
- **Distributed and Resilient:** Document data models are very much dispersed which is the reason behind horizontal scaling and distribution of data.
- **Manageable Query Language:** These data models are the ones in which query language allows the developers to perform CRUD (Create Read Update Destroy) operations on the data model.

Examples of Document Data Models:

- Amazon DocumentDB
- MongoDB
- Cosmos DB
- ArangoDB
- Couchbase Server
- CouchDB

Advantages:

- **Schema-less:** These are very good in retaining existing data at massive volumes because there are absolutely no restrictions in the format and the structure of data storage.
- **Faster creation of document and maintenance:** It is very simple to create a document and apart from this maintenance requires is almost nothing.

- **Open formats:** It has a very simple build process that uses XML, JSON, and its other forms.
- **Built-in versioning:** It has built-in versioning which means as the documents grow in size there might be a chance they can grow in complexity. Versioning decreases conflicts.

Disadvantages:

- **Weak Atomicity:** It lacks in supporting multi-document ACID transactions. A change in the document data model involving two collections will require us to run two separate queries i.e. one for each collection. This is where it breaks atomicity requirements.
- **Consistency Check Limitations:** One can search the collections and documents that are not connected to an author collection but doing this might create a problem in the performance of database performance.
- **Security:** Nowadays many web applications lack security which in turn results in the leakage of sensitive data. So it becomes a point of concern, one must pay attention to web app vulnerabilities.

Applications of Document Data Model:

- Content Management
- Book Database
- Catalog
- Analytics Platform

11. Explain about key-value store in NoSQL database.

Key-Value database model

- A key-value data model or database is also referred to as a key-value store. It is a non-relational type of database.
- In this, an associative array is used as a basic database in which an individual key is linked with just one value in a collection. For the values, keys are special identifiers. Any kind of entity can be valued.
- The collection of key-value pairs stored on separate records is called key-value databases and they do not have an already defined structure.

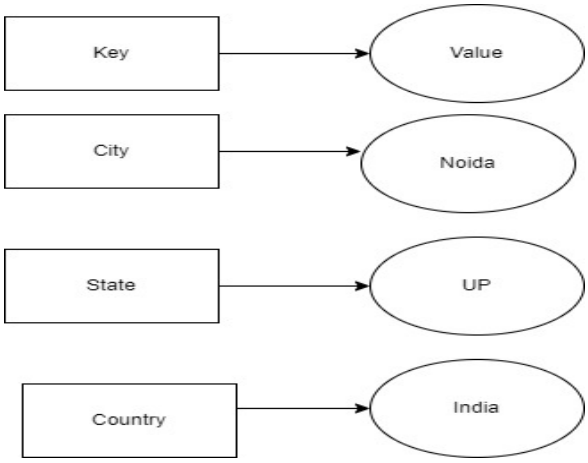


Fig. 5.14 Key-Value model

Working of key-value databases:

A number of easy strings or even a complicated entity are referred to as a value that is associated with a key by a key-value database, which is utilized to monitor the entity.

Like in many programming paradigms, a key-value database resembles a map object or array, or dictionary, however, which is put away in a tenacious manner and controlled by a DBMS.

An efficient and compact structure of the index is used by the key-value store to have the option to rapidly and dependably find value using its key.

For example, Redis is a key-value store used to tracklists, maps, heaps, and primitive types (which are simple data structures) in a constant database.

Redis can uncover a very basic point of interaction to query and manipulate value types, just by supporting a predetermined number of value types, and when arranged, is prepared to do high throughput.

When to use a key-value database:

- User session attributes in an online app like finance or gaming, which is referred to as real-time random data access.
- Caching mechanism for repeatedly accessing data or key-based design.
- The application is developed on queries that are based on keys.

Features:

- One of the most un-complex kinds of NoSQL data models.
- For storing, getting, and removing data, key-value databases utilize simple functions.
- Querying language is not present in key-value databases.
- Built-in redundancy makes this database more reliable.

Advantages:

- It is very easy to use. Due to the simplicity of the database, data can accept any kind, or even different kinds when required.
- Its response time is fast due to its simplicity, given that the remaining environment near it is very much constructed and improved.
- Key-value store databases are scalable vertically as well as horizontally.
- Built-in redundancy makes this database more reliable.

Disadvantages:

- As querying language is not present in key-value databases, transportation of queries from one database to a different database cannot be done.
- The key-value store database is not refined. You cannot query the database without a key.

Some examples of key-value databases:

- Couchbase
- Amazon DynamoDB
- Riak
- Aerospike
- Berkeley DB

12. Explain in detail about Column-based database with example.

Column based Data Model of NoSQL:

Basically, the relational database stores data in rows and also reads the data row by row, column store is organized as a set of columns.

So if someone wants to run analytics on a small number of columns, one can read those columns directly without consuming memory with the unwanted data.

Columns are somehow of the same type and gain from more efficient compression, which makes reads faster than before.

Examples of Columnar Data Model: Cassandra and Apache Hadoop Hbase.

Working of Columnar Data Model:

In Columnar Data Model instead of organizing information into rows, it does in columns. This makes them function the same way that tables work in relational databases.

This type of data model is much more flexible obviously because it is a type of NoSQL database. Columnar Data Model uses the concept of keyspace, which is like a schema in relational models.

The below example will help in understanding the Columnar data model:

Row-Oriented Table:

S.No.	Name	Course	Branch	ID
01.	Tanmay	B-Tech	Computer	2
02.	Abhishek	B-Tech	Electronics	5
03.	Samriddha	B-Tech	IT	7
04.	Aditi	B-Tech	E & TC	8

Column – Oriented Table:

S.No.	Name	ID
01.	Tanmay	2
02.	Abhishek	5
03.	Samriddha	7
04.	Aditi	8

S.No.	Course	ID
01.	B-Tech	2
02.	B-Tech	5
03.	B-Tech	7
04.	B-Tech	8

S.No.	Branch	ID
01.	Computer	2
02.	Electronics	5
03.	IT	7
04.	E & TC	8

Advantages of Columnar Data Model:

- **Well structured:** Since these data models are good at compression so these are very structured or well organized in terms of storage.
- **Scalability:** It can be spread across large clusters of machines, even numbering in thousands.

- **Flexibility:** A large amount of flexibility as it is not necessary for the columns to look like each other, which means one can add new and different columns without disrupting the whole database.
- **Aggregation queries are fast:** The most important thing is aggregation queries are quite fast because a majority of the information is stored in a column. An example would be Adding up the total number of students enrolled in one year.
- **Load Times:** Since one can easily load a row table in a few seconds so load times are nearly excellent.

Disadvantages of Columnar Data Model:

- **Designing indexing Schema:** To design an effective and working schema is too difficult and very time-consuming.
- **Suboptimal data loading:** incremental data loading is suboptimal and must be avoided, but this might not be an issue for some users.
- **Security vulnerabilities:** If security is one of the priorities then it must be known that the Columnar data model lacks inbuilt security features in this case, one must look into relational databases.
- **Online Transaction Processing (OLTP):** Online Transaction Processing (OLTP) applications are also not compatible with columnar data models because of the way data is stored.

Applications of Columnar Data Model:

- Columnar Data Model is very much used in various Blogging Platforms.
- It is used in Content management systems like WordPress, Joomla, etc.
- It is used in Systems that maintain counters.
- It is used in Systems that require heavy write requests.
- It is used in Services that have expiring usage.

13. Explain in detail about Graph-based database with example.

Graph based database model

- Graph Based Data Model in NoSQL is a type of Data Model which tries to focus on building the relationship between data elements.
- As the name suggests Graph-Based Data Model, each element here is stored as a node, and the association between these elements is often known as Links.
- Association is stored directly as these are the first-class elements of the data model. These data models give us a conceptual view of the data.
- These are the data models which are based on topographical network structure.
- Obviously, in graph theory, we have terms like Nodes, edges, and properties, which means here in the Graph-Based data model.
- **Nodes:** These are the instances of data that represent objects which is to be tracked.
- **Edges:** As we already know edges represent relationships between nodes.
- **Properties:** It represents information associated with nodes.

The picture given below represents Nodes with properties from relationships represented by edges.

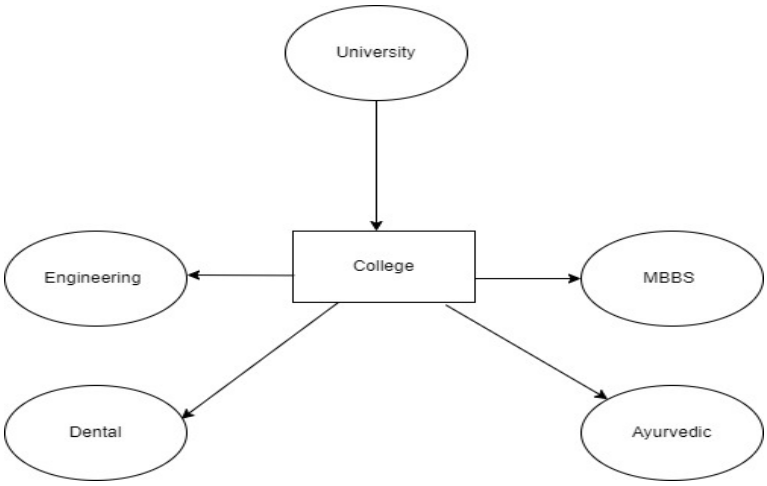


Fig. 5.15 Relationship among Nodes, Edges and Properties

Working of Graph Database Model:

- In these data models, the nodes which are connected together are connected physically and the physical connection among them is also taken as a piece of data.
- Connecting data in this way becomes easy to query a relationship.
- This data model reads the relationship from storage directly instead of calculating and querying the connection steps.
- Like many different NoSQL databases these data models don't have any schema as it is important because schema makes the model well and good and easy to edit.

Examples of Graph Database Models:

- ✓ **JanusGraph:** These are very helpful in big data analytics. It is a scalable graph database system open source too. JanusGraph has different features like:
  - Storage
  - Support for transactions
  - Searching options
- ✓ **Neo4j:** It stands for Network Exploration and Optimization 4 Java. As the name suggests this graph database is written in Java with native graph storage and processing. Neo4j has different features like:
  - Scalable
  - Higher Availability
  - Query Language
- ✓ **DGraph:** It is an open-source distributed graph database system designed with scalability. Its features are:
  - Query Language
  - open-source system

Advantages of Graph Data Model:

- **Structure:** The structures are very agile and workable too.
- **Explicit Representation:** The portrayal of relationships between entities is explicit.
- **Real-time O/P Results:** Query gives us real-time output results.

Disadvantages of Graph Data Model:

- **No standard query language:** Since the language depends on the platform that is used so there is no certain standard query language.
- **Unprofessional Graphs:** Graphs are very unprofessional for transactional-based systems.
- **Small User Base:** The user base is small which makes it very difficult to get support when running into a system.

Applications of Graph Data Model:

- Graph data models are very much used in fraud detection which itself is very much useful and important.
- It is used in Digital asset management which provides a scalable database model to keep track of digital assets.
- It is used in Network management which alerts a network administrator about problems in a network.
- It is used in Context-aware services by giving traffic updates and many more.
- It is used in Real-Time Recommendation Engines which provide a better user experience.

14. Describe about Database Security with common threats and counter measures in detail.

- Security of databases refers to the array of controls, tools, and procedures designed to ensure and safeguard confidentiality, integrity, and accessibility.
- It is a technique for protecting and securing a database from intentional or accidental threats.
- Database security encompasses hardware parts, software parts, human resources, and data.
- To use the security efficiently, appropriate controls are required, which are separated into a specific goal and purpose for the system.

Database security is more prevalent in the following scenarios:

- Theft and fraudulent.
- Loss of Data privacy.
- Loss of Data integrity.
- Loss of confidentiality or secrecy
- Loss of availability of data.

Why Database Security is Important?

Security is an important concern in database management because the information stored in a database is a very valuable and, at times, quite sensitive commodity. As a result, data in a database management system must be protected from abuse and illegal access and updates.

The factors are:

- Intellectual property that is compromised
- The damage to our brand's reputation
- The concept of business continuity
- Penalties or fines to be paid for not complying
- Costs for repairing breaches and notifying consumers about them

Common Threats and Challenges for database security:

- **Insider Dangers**

An insider threat can be an attack on security from any three sources having an access privilege to the database.

  - A malicious insider who wants to cause harm
  - An insider who is negligent and makes mistakes that expose the database to attack. vulnerable to attacks
  - An infiltrator is an outsider who acquires credentials by using a method like phishing or accessing the database of credential information in the database itself.

Insider dangers are among the most frequent sources of security breaches to databases. They often occur as a consequence of the inability of employees to have access to privileged user credentials.

- **Human Error**

The unintentional mistakes, weak passwords or sharing passwords, and other negligent or uninformed behaviors of users remain the root causes of almost half (49 percent) of all data security breaches.

- **Software Vulnerabilities**

Hackers earn their money by identifying and exploiting vulnerabilities in software such as databases management software. The major database software companies and open-source databases management platforms release regular security patches to fix these weaknesses. However, failing to implement the patches on time could increase the risk of being hacked.

- **SQL/NoSQL Injection Attacks**

A specific threat to databases is the infusing of untrue SQL as well as other non-SQL string attacks in queries for databases delivered by web-based apps and HTTP headers. Companies that do not follow the safe coding practices for web applications and conduct regular vulnerability tests are susceptible to attacks using these.

- **Buffer Overflow**

Buffer overflow happens when a program seeks to copy more data into the memory block with a certain length than it can accommodate. The attackers may make use of the extra data, which is stored in adjacent memory addresses, to establish a basis for they can begin attacks.

- **DDoS (DoS/DDoS) Attacks**

In a denial-of-service (DoS) attack in which the attacker overwhelms the targeted server -- in this case, the database server with such a large volume of requests that the server is unable to meet no longer legitimate requests made by actual users. In most cases, the server is unstable or even fails to function.

- **Malware**

Malware is software designed to exploit vulnerabilities or cause harm to databases. Malware can be accessed via any device that connects to the databases network.

- **Attacks on Backups**

Companies that do not protect backup data using the same rigorous controls employed to protect databases themselves are at risk of cyber attacks on backups.

Control Measures for the Security of Data in Databases:

❖ **Authentication**

- Authentication is the process of confirming whether a user logs in only with the rights granted to him to undertake database operations. A certain user can only log in up to his privilege level, but he cannot access any other sensitive data.
- The ability to access sensitive data is restricted by the use of authentication. For example, a mobile phone performs authentication by requesting a PIN, fingerprint, or by face recognition. Similarly, a computer verifies a username by requesting the appropriate password.

❖ **Access Control**

- Database access control is a means of restricting access to sensitive company data to only those people (database users) who are authorized to access such data and permitting access to unauthorized persons. It is a key security concept that reduces risk to the business or organization.
- Physical and logical access control are the two types of access control. Access to campuses, buildings, rooms, and physical IT assets is restricted through physical access control. Connections to computer networks, system files, and data are restricted through logical access control.
- The most well-known Database Access Control examples are:
  - Discretionary Access Control (DAC)
  - Mandatory Access Control (MAC)
  - Role-Based Access Control (RBAC)
  - Attribute-Based Access Control (ABAC)

❖ **Inference Control**

- Inference control in databases, also known as Statistical Disclosure Control (SDC), is a discipline that aims to secure data so that it can be published without disclosing sensitive information associated with specific individuals among those to whom the data corresponds.
- It prevents the user from completing any inference channel. This strategy prevents sensitive information from indirect disclosure.
- There are two kinds of inferences:
  i. Identity disclosure
  ii. Attribute disclosure

❖ **Flow Control**

- Distributed systems involve a large amount of data flow from one site to another as well as within a site.
- Flow control prohibits data from being transferred in such a way that unauthorized agents cannot access it. A flow policy specifies the channels through which data can flow. It also defines security classes for data as well as transactions.

❖ **Applying Statistical Method**

- Statistical database security focuses on the protection of sensitive individual values stored in so-called statistical databases and used for statistical purposes, as well as retrieving summaries of

values based on categories. They do not allow the retrieval of individual information.

- It provides access to the database to obtain statistical information about the number of employees in the company but not to obtain detailed confidential/personal information about a specific individual employee.
- The techniques used to prevent statistical database compromise are classified into two types: noise addition, in which all data and/or statistics are available but are only approximate rather than exact, and restriction, in which the system only delivers statistics and/or data that are deemed safe.

❖ **Encryption**

- Data encryption protects data confidentiality by converting it to encoded information known as cipher text, which can only be decoded with a unique decryption key generated either during or before encryption.
- Data encryption can be used during data storage or transfer, and it is usually used in conjunction with authentication services to ensure that keys are only given to or used by authorized users.
- Data is more accessible and desirable to attackers than ever before, increasing the need for security. Additionally, many firms must comply with data protection regulations, many of which specifically require the use of encryption.

15. Explain in detail about Discretionary access control privileges and Role based access control mechanism.

Discretionary Access Control

- Discretionary Access Control is a mechanism that allows the owner of a resource to control who has access to that resource and what actions they can perform on it.
- The term "discretionary" implies that the owner has the discretion to decide who is granted access and what privileges they are granted.
- In a discretionary access control system, each resource is assigned a unique owner account, which is responsible for managing access to that resource.
- The owner has the ability to grant or revoke access privileges, such as reading or modifying data, to other users or groups of users.

Types of Discretionary Privileges:

There are two levels for assigning privileges to use the database system:

- **The Account Level** - At this level, the DBA specifies the particular privileges that each account holds independently of the relations in the database.
- **The Relation (or Table) Level** - At this level, the DBA can control the privilege to access each individual relation or view in the database.

Account Level Privileges apply to the capabilities provided to the account itself and can include:

- The CREATE SCHEMA or CREATE TABLE privilege, to create a schema or base relation.
- The CREATE VIEW privilege.
- The ALTER privilege, to apply schema changes such as adding or removing attributes from relations.

- The DROP privilege, to delete relations or views.
- The MODIFY privilege, to insert, delete, or update tuples.
- The SELECT privilege, to retrieve information from the database by using a SELECT query.

These privileges give the account holder varying degrees of control over the database and its contents. The DBA must carefully consider the level of access they grant to each account to ensure the security and protection of sensitive information.

Relation Level Privileges are the second level of privileges applies to the relation level, whether they are base relations or virtual (view) relations.

- It is the second level of privileges which is applied to the relation level. This includes tables or relations and virtual relations known as views.
- A user who has created a database object such as a table or a view will get all privileges on that object.
- This user is the holder of owner account which is created for each relation. In this level, an owner account is created for each relation and this account will also have right to pass the privileges to other users by GRANTING privileges to their accounts.
- The granting and revoking of discretionary privileges can be done by using a model known as access matrix model. This model specifies rights of each subject for each object.

Access matrix Model

subject	file1	file2	file3
Mary	read	write	
Sashi		read	write
Rahul	write	read	append

Here, Mary has only read privilege on file1 she can't modify that file1. So, we can represent the privileges of each subject on each object. The matrix M consists of rows which resembles subjects like users, accounts and the columns resembles objects like relations, views. Each position M(i,j) in the matrix represents the types of privileges like read, write, update that subject i holds on object j.

Role-based access control (RBAC)

- Role-based access control (RBAC), also known as role-based security, is a mechanism that restricts system access.
- It involves setting permissions and privileges to enable access to authorized users.
- Role-based access control is generally used in conjunction with the principle of least privilege, where the roles represented will only involve the least level of access needed to done the necessary job service or requirements.

- Most large organizations use role-based access control to provide their employees with varying levels of access based on their roles and responsibilities.
- An organization assigns a role-based access control role to every employee; the role determines which permissions the system grants to the user.
- For example, you can designate whether a user is an administrator, a student, or an end-user, and limit access to specific resources or tasks. An organization may let some individuals create or modify files while providing others with viewing permission only.

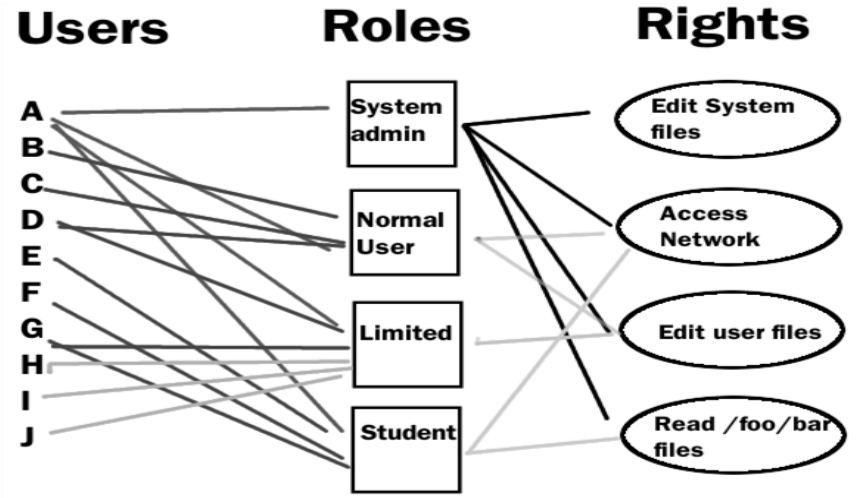


Fig. 5.16 Specific roles and rights assigned to specific users

The **RBAC methodology** is based on a group of **three primary rules** that govern access to secured systems:

- ✓ **Role Assignment** − Each transaction or operation can only be carried out if the user has assumed the suitable role. An operation is represented as some action taken with respect to a system or network object that is secured by RBAC. Roles can be assigned by an independent party or selected by the user attempting to implement the action.
- ✓ **Role Authorization** − The objective of role authorization is to provide that users can only consider a role for which they have been given the suitable authorization. When a user consider a role, they should do so with authorization from an administrator.
- ✓ **Transaction Authorization** − An operation can only be done if the user trying to complete the transaction possesses the suitable role.

The **benefits of RBAC** include the possibility to:

- Create a systematic, repeatable assignment of permissions
- Audit user privileges and correct identified issues
- Add, remove or change roles, as well as implement them across API calls
- Reduce potential errors when assigning user permissions
- Reduce third-party risk and fourth-party risk by providing third-party vendors and suppliers with pre-defined roles
- Decrease the risk of data breaches and data leakage by restricting access to sensitive information

16. Discuss about SQL injection in detail with an example.

SQL injection

SQL injection is a technique used to extract user data by injecting web page inputs as statements through SQL commands. Basically, malicious users can use these instructions to manipulate the application's web server.

1. SQL injection is a code injection technique that can compromise your database.
2. SQL injection is one of the most common web hacking techniques.
3. SQL injection is the injection of malicious code into SQL statements via web page input.

Types of SQL injection attacks

SQL injections can do more harm other than passing the login algorithms. Some of the SQL injection attacks include:

- Updating, deleting, and inserting the data: An attack can modify the cookies to poison a web application's database query.
- It is executing commands on the server that can download and install malicious programs such as Trojans.
- We are exporting valuable data such as credit card details, email, and passwords to the attacker's remote server.
- Getting user login details: It is the simplest form of SQL injection. Web application typically accepts user input through a form, and the front end passes the user input to the back end database for processing.

Example of SQL Injection

Suppose we have an application based on student records. Any student can view only his or her own records by entering a unique and private student ID.

Suppose we have a field like the one below:

Student id: The student enters the following in the input field: **12222345 or 1=1**.

Query:

```
SELECT * from STUDENT where
STUDENT-ID == 12222345 or 1 = 1
```

Now, this **1=1** will return all records for which this holds true. So basically, all the student data is compromised. Now the malicious user can also delete the student records in a similar fashion. Consider the following SQL query.

Query:

```
SELECT * from USER where
USERNAME = “” and PASSWORD=””
```

Now the malicious can use the '=' operator in a clever manner to retrieve private and secure user information. So instead of the above-mentioned query the following query when executed retrieves protected data, not intended to be shown to users.

Query:

```
Select * from User where
(Username = “” or 1=1) AND
(Password=”” or 1=1).
```

Since **1=1** always holds true, user data is compromised.

Impact of SQL Injection

- The intruder can retrieve all the user-data present in the database, such as user details, credit card information, and social security numbers, and can also gain access to protected areas like the administrator portal.
- It is also possible to delete the user data from the tables.
- These days all the online shopping applications, bank transactions use back-end database servers.
- If the intruder can exploit SQL injection, the entire server is compromised.

Preventing SQL Injection

- User Authentication: Validating input from the user by pre-defining length, type of input, of the input field and authenticating the user.
- Restricting access privileges of users and defining how much amount of data any outsider can access from the database. Basically, users should not be granted permission to access everything in the database.
- Do not use system administrator accounts.

17. What is data encryption? Explain in detail about the encryption methods and public key infrastructure in detail.

- Cryptography is the science of encoding information before sending via unreliable communication paths so that only an authorized receiver can decode and use it.
- The coded message is called **cipher text** and the original message is called **plain text**.
- The process of converting plain text to cipher text by the sender is called encoding or **encryption**.
- The process of converting cipher text to plain text by the receiver is called decoding or **decryption**.

The entire procedure of communicating using cryptography can be illustrated through the following diagram –

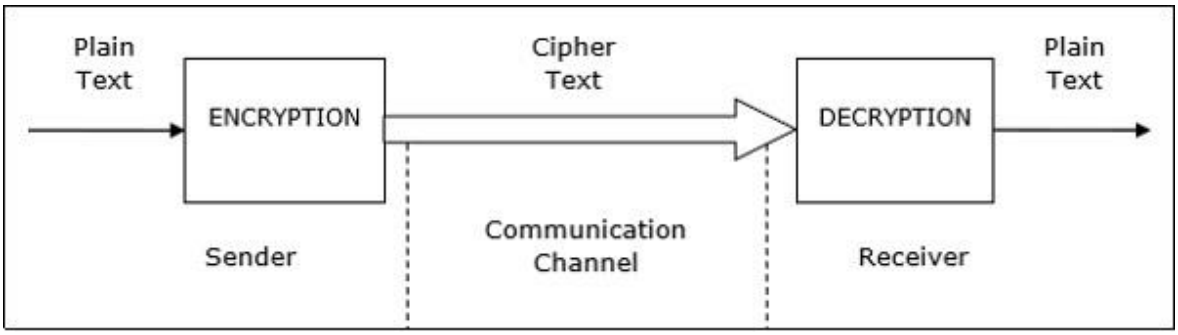


Fig. 5.17 Communication using cryptography

Types of Encryption

1. Symmetric Encryption– Data is encrypted using a key and the decryption is also done using the same key.

2. Asymmetric Encryption-Asymmetric Cryptography is also known as public-key cryptography. It uses public and private keys to encrypt and decrypt data. One key in the pair which can be shared with everyone is called the public key. The other key in the pair which is kept secret and is only known by the owner is called the private key. Either of the keys can be used to encrypt a message; the opposite key from the one used to encrypt the message is used for decryption.

- **Public key**– Key which is known to everyone. Ex-public key of A is 7, this information is known to everyone.
  Private key– Key which is only known to the person who's private key it is.
- **Authentication**-Authentication is any process by which a system verifies the identity of a user who wishes to access it.
- **Non- repudiation**– Non-repudiation means to ensure that a transferred message has been sent and received by the parties claiming to have sent and received the message. Non-repudiation is a way to guarantee that the sender of a message cannot later deny having sent the message and that the recipient cannot deny having received the message.
- **Integrity**– to ensure that the message was not altered during the transmission.
- **Message digest** -The representation of text in the form of a single string of digits, created using a formula called a one way hash function. Encrypting a message digest with a private key creates a digital signature which is an electronic means of authentication.

Public Key Cryptography

- In contrast to conventional cryptography, public key cryptography uses two different keys, referred to as public key and the private key.
- Each user generates the pair of public key and private key. The user then puts the public key in an accessible place.
- When a sender wants to sends a message, he encrypts it using the public key of the receiver. On receiving the encrypted message, the receiver decrypts it using his private key. Since the private key is not known to anyone but the receiver, no other person who receives the message can decrypt it.

- The most popular public key cryptography algorithms are **RSA** algorithm and **Diffie–Hellman** algorithm.
- This method is very secure to send private messages. However, the problem is, it involves a lot of computations and so proves to be inefficient for long messages.
- The solution is to use a combination of conventional and public key cryptography. The secret key is encrypted using public key cryptography before sharing between the communicating parties.
- Then, the message is send using conventional cryptography with the aid of the shared secret key.

Digital Signatures

- A Digital Signature (DS) is an authentication technique based on public key cryptography used in e-commerce applications.
- It associates a unique mark to an individual within the body of his message. This helps others to authenticate valid senders of messages.
- Typically, a user's digital signature varies from message to message in order to provide security against counterfeiting.

Digital Certificate

- Digital certificate is issued by a trusted third party which proves sender's identity to the receiver and receiver's identity to the sender.
- A digital certificate is a certificate issued by a Certificate Authority (CA) to verify the identity of the certificate holder.
- The CA issues an encrypted digital certificate containing the applicant's public key and a variety of other identification information.
- Digital certificate is used to attach public key with a particular individual or an entity.

18. Discuss in detail about the challenges in database security.

Database security is an essential aspect of database management systems (DBMS) as it involves protecting the confidentiality, integrity, and availability of the data stored in the database. A DBMS in addition to making every effort to prevent an attack and detecting one in the event of the occurrence should be able to do the following:

- **Confident:** We should take immediate action to eliminate the attacker's access to the system and to isolate or contain the problem to prevent further spread.
- **Damage assessment:** Determine the extent of the problem, including failed function and corrupted data.
- **Recover:** Recover corrupted or lost data and repair or reinstall failed function to reestablish a normal level of operation.
- **Reconfiguration:** Reconfigure to allow the operation to continue in a degraded mode while recovery proceeds.
- **Fault treatment:** To the extent possible, identify the weakness exploited in the attack and takes steps to prevent a recurrence.

The challenges of database security in DBMS include:

- **Authentication and Authorization:** One of the biggest challenges of database security is ensuring that only authorized users can access the database. The DBMS must authenticate users and grant them appropriate access rights based on their roles and responsibilities.
- **Encryption:** Data encryption is an effective way to protect sensitive data in transit and at rest. However, it can also be a challenge to implement and manage encryption keys and ensure that encrypted data is not compromised.
- **Access Control:** Access control involves regulating the access to data within the database. It can be challenging to implement access control mechanisms that allow authorized users to access the data they need while preventing unauthorized users from accessing it.
- **Auditing and Logging:** DBMS must maintain an audit trail of all activities in the database. This includes monitoring who accesses the database, what data is accessed, and when it is accessed. This can be a challenge to implement and manage, especially in large databases.
- **Database Design:** The design of the database can also impact security. A poorly designed database can lead to security vulnerabilities, such as SQL injection attacks, which can compromise the confidentiality, integrity, and availability of data.
- **Malicious attacks:** Cyber attacks such as hacking, malware, and phishing pose a significant threat to the security of databases. DBMS must have robust security measures in place to prevent and detect such attacks.
- **Physical Security:** Physical security of the database is also important, as unauthorized physical access to the server can lead to data breaches.