SUBJECT NAME	**:**	**DATABASE MANAGEMENT SYSTEMS**
SUBJECT CODE	**:**	**CS8492**
YEAR/SEM	**:**	**II Year / Fourth Semester**
DEPARTMENT	**:**	**INFORMATION TECHNOLOGY**
AUTHOR	**:**	**R. LAVANYA, M.E., (Ph.D)**
DESGINATION	**:**	**ASSISTANT PROFESSOR**

REGULATIONS - 2017

SYLLABUS

OBJECTIVES:

- To learn the fundamentals of data models and to represent a database system using ER diagrams.
- To study SQL and relational database design.
- To understand the internal storage structures using different file and indexing techniques which will help in physical DB design.
- To understand the fundamental concepts of transaction processing- concurrency control techniques and recovery procedures.
- To have an introductory knowledge about the Storage and Query processing Techniques

UNIT I RELATIONAL DATABASES 10

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 8

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.

UNIT III TRANSACTIONS 9

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

UNIT IV IMPLEMENTATION TECHNIQUES 9

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

UNIT V ADVANCED TOPICS 9

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

Unit –1

RELATIONAL DATABASES

PART-A

1. What is the purpose of Database Management System? (NOV 2014)

- Real world entity
- Relation-based tables
- Isolation of data and application
- Less redundancy
- Consistency
- Query Language
- ACID Properties
- Multiuser and concurrent Access
- Multiple Views
- Security

2.What is Data Definition Language? Give example. (NOV 2016, APR 2018)

- Data Definition Language (DDL) – specifies constructs for schema definition, relation definition, integrity constraints, views and schema modification.

- Data Definition Language (DDL) is a standard for commands that define the different structures in a database. DDL statements create, modify, and remove database objects such as tables, indexes, and users. Common DDL statements are CREATE, ALTER, and DROP.

3. Write the Characteristics that distinguish the Database approach with the File-based Approach? (MAY 2015, Nov 2019)

Database Approach	**File-Based Approach**
A database System contains not only the database itseld but also the descriptions of data structure and constraints (meta-data).	Data definition is a part of application programs
In the database approach, the data structure is stored in the system catalog not in the programs.	The structure of the data files is defined in the application programs so if a user wants to change the structure of a file, all the programs that access that file might need to be changed as well.
A multiuser database system allows multiple user access to the database at the same time. They have concurrency control strategies and integrity checks for several user access.	Same data is managed more than once this leads to redundancy and inconsistency.

4.What are the disadvantages of file processing system? (NOV 2015)

- Data redundancy and inconsistency
- Difficulty in accessing data
- Data isolation
- Marinating Integrity
- Atomicity Problems
- Concurrent Access Anomalies

- Security Problems

5. Differentiate File Processing System with Database Management System. (NOV 2016)

1. A database management system coordinates both the physical and the logical access to the data, whereas a file-processing system coordinates only the physical access.

2. A database management system reduces the amount of data duplication by ensuring that a physical piece of data is available to all programs authorized to have access to it, whereas data written by one program in a file-processing system may not be readable by another program.

3. A database management system is designed to allow flexible access to data (i.e., queries), whereas a file-processing system is designed to allow predetermined access to data (i.e., compiled programs).

4. A database management system is designed to coordinate multiple users accessing the same data at the same time. A file-processing system is usually designed to allow one or more programs to access different data files at the same time. In a file-processing system, a file can be accessed by two programs concurrently only if both programs have read-only access to the file.

5. Redundancy is control in DBMS, but not in file system

6. Unauthorized access is restricted in DBMS but not in file system.

7. DBMS provide back up and recovery. When data is lost in file system then it not recover.

8. DBMS provide multiple user interfaces. Data is isolated in file system,

6. What is a weak entity? Give Example. (NOV 2016, APR 2018)

Weak entity set: entity set that do not have key attribute of their own are called weak entity sets
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.
Examples. The existence of rooms is entirely depended on the existence of a hotel.

7. **Who is a DBA? What are the responsibilities of a DBA? April/May-2011 (OR)**
Mention some responsibilities of a database Administrator? (NOV 2018)
A database administrator (short form DBA) is a person responsible for the design, implementation, maintenance and repair of an organization's database. They are also known by the titles Database Coordinator or Database Programmer, and is closely related to the Database Analyst, Database Modeller, Programmer Analyst, and Systems Manager.
The role includes the development and design of database strategies, monitoring and improving database performance and capacity, and planning for future expansion requirements. They may also plan, co-ordinate and implement security measures to safeguard the database.

8. What is a data model? List the types of data model used. (April/May-2011 , May 2019)
A **database model** is the theoretical foundation of a database and fundamentally determines in which manner data can be stored, organized, and manipulated in a database system. It thereby defines the infrastructure offered by a particular database system. The most popular example of a database model is the relational model.

Types of data model used

□□Hierarchical model
□□Network model
□□Relational model
□□Entity-relationship
□□Object-relational model
□□Object model

9. Define Atomicity in transaction management. (APR 2013)
In database systems, **atomicity** (or atomicness; from Greek a-tomos, undividable) is one of the ACID **transaction** properties. In an atomic **transaction**, a series of database operations either all occur, or nothing occurs.
10. Give an example of one to one and one to many relationship? (APR 2013, NOV 2018)
One-to-One (1-1) relationship is defined as the relationship between two tables where both the tables should be associated with each other based on only one matching row. This relationship can be created using **Primary key-Unique foreign key constraints**.

With One-to-One Relationship in SQL Server, for example, a person can have only one passport.

The One-to-Many relationship is defined as a relationship between two tables where a row from one table can have multiple matching rows in another table. This relationship can be created using **Primary key-Foreign key relationship**.

In the One-to-Many Relationship in SQL Server, for example, a book can have multiple authors.

11. What do you meant by simple and composite attribute? (DEC 2013)
Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.
A composite attribute consists of a group of values from more than one domain. For example, the Address attribute consists of several domains such as house number, street number, city, country, etc.
12. Define query? (DEC 2013)
A query is a statement requesting the retrieval of information. The portion of DML that involves information retrieval is called a query language.

13. Define database management system? List various **applications of DBMS? (MAY 2019)**
Database management system (DBMS) is a collection of interrelated data and a set of programs to access those data.

a) Banking
b) Airlines
c) Universities
d) Credit card transactions
e) Tele communication
f) Finance g) Sales
h) Manufacturing
i) Human resources

14.Differentiate between static and Dynamic SQL? (NOV 2016 , NOV 2015, NOV 2014) (or)
What is Static SQL how it is different from Dynamic SQL? (NOV 2017) (or)
Describe about the static SQL and Dynamic SQL in detail? (Apr 2019)

Static (Embedded) SQL	Dynamic (Interactive) SQL
In static SQL how databases will be accessed is predetermined in the embedded SQL statement.	In dynamic SQL, how database will be accessed is determined at run time
It is more swift and efficient. It is less flexible.	It is less swift and efficient. It is more flexible.
SQL statements are compiled at compile time	SQL statements are compiled at run time
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.
It is generally used for situations where data is distributed uniformly.	It is generally used for situations where data is distributed non-uniformly.

15.Name the Categories of SQL Commands? (JUN 2016)

SQL commands are divided in to the following categories:

1. Data definition language
2. Data manipulation language
3. Data Query language
4. Data control language
5. Data administration statements
6. Transaction control statements

16.Explain "Query Optimization"? State the need of Query Optimization?(JUN 2016, MAY 2015)

Query optimization refers to the process of finding the lowest –cost method of evaluating a given query.

Query Optimization is a crucial and difficult part of the overall query processing. The main objective of query optimization is to minimize the following cost function:

I/O cost + CPU cost + Communication cost.

It generates an optimal evaluation plan (with lowest cost) for the query plan.

17.Why SQL allow duplicate tuples in a table or in a query result? (NOV 2015)

By default query result and table allow duplicate records. so to avoid unnecessary checks during SELECT, INSERT, UPDATE or DELETE SQL allow duplicate tuples in a table or in a query result.

SELECT returns data according to WHERE filter (or all data, if it is omitted). It simply checks WHERE conditions for each rows. If table has duplicates and they comply with WHERE clause, the rows are selected. we can add DISTINCT to avoid duplicates.

Duplicated rows in tables can be restricted with UNIQUE keys.

18. Give a brief description on DCL command? (NOV 2014)

DCL- Data Control Language

Commands:

i)Grant : Gives a privilege to user

ii) Revoke: Takes back privileges granted from user.

19.What are Primary key constraints? (JUN 2013)

The PRIMARY KEY constraint uniquely identifies each record in a database table. Primary keys must contain UNIQUE values, and cannot contain NULL values. A table can have only one primary key, which may consist of single or multiple fields.

20.Write the purpose of trigger?(JUN 2013)

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

In a DBMS, a trigger is a SQL procedure that initiates an action (i.e., fires an action) when an event (INSERT, DELETE or UPDATE) occurs. ... Stored procedures are explicitly executed by invoking a CALL to the procedure while triggers are implicitly executed.

21.State the difference between security and integrity? (NOV 2013)

Data integrity = making sure the data is correct and not corruptData security = making sure only the people who should have access to the data are the only ones who can access the data. also, keeping straight who can read the data and who can write data.

Pure integrity of data refers to the property which determines that data, once stored, has not been altered in an unauthorised way -- either by a person, or by the malfunctioning of hardware.

22.Which operators are called as unary operators and why are they called so? (NOV 2013)

The unary operators are :

Increment: ++x, x++

Decrement: −−x, x−−

Address: &x.

Indirection: *x.

Positive: +x.

Negative: −x.

Ones' complement: ~x.

Logical negation: !x.

A **unary operator**, is an **operator** that takes a single operand in an expression or a statement.

A **unary operator** is an **operator** that takes only one value for its operation.

23**. 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

24. What are the three clauses of SQL expression?

SQL expression consists of three clauses:

Select From where

25. Give the general form of SQL query?

Select A1, A2……., An **From** R1, R2……, Rm **Where** P

26. 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

27. Define tuple variable?

Tuple variables are used for comparing two tuples in the same relation. The tuple variables are defined in the **from** clause by way of the **as** clause.

28. List the string operations supported by SQL?

1) Pattern matching Operation
2) Concatenation
3) Extracting character strings
4) Converting between uppercase and lower case letters.

29. List the set operations of SQL?

1) Union
2) Intersect operation
3) The except operation

30. What is the use of Union and intersection operation?

Union: The result of this operation includes all tuples that are either in r1 or in r2 or in both r1 and r2.Duplicate tuples are automatically eliminated.

Intersection: The result of this relation includes all tuples that are in both r1 and

31. What are aggregate functions? And list the aggregate functions supported by SQL? (NOV 2018)

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

32. 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.

33. 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 setcomparisions, and determine set cardinality.

34. 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. The form of the

create view command is

Create view v as <query expression>

35. What is the use of with clause in SQL?

The **with** clause provides a way of defining a temporary view whose definition is available only to the query in which the **with** clause occurs.

36. List the table modification commands in SQL?

Deletion

Insertion Updates

Update of a view

37. List out the statements associated with a database transaction?

Commit work and Rollback work

38. What is transaction?

Transaction is a unit of program execution that accesses and possibly updated various data items.

39. 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.

40. 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.

41. Mention the two forms of integrity constraints in ER model?

Key declarations
Form of a relationship

42. What are domain constraints?

A domain is a set of values that may be assigned to an attribute .all values that appear in a column of a relation must be taken from the same domain.

43. What are referential integrity constraints?

A value that appears in one relation for a given set of attributes also appears for acertain set of attributes in another relation.

44. What is assertion? Mention the forms available.

An assertion is a predicate expressing a condition that we wish the database always to satisfy.
Domain integrity constraints.
Referential integrity constraints

45. Give the syntax of assertion?

Create assertion <assertion name>**check**<predicate>

4**6.** What is the need for triggers? **List the requirements needed to design a trigger.**

Triggers are useful mechanisms for alerting humans or for starting certain tasks automatically when certain conditions are met.

The requirements are
Specifying when a trigger is to be executed.
Specify the actions to be taken when the trigger executes.

47. 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.

48. List the disadvantages of relational database system

Repetition of data
Inability to represent certain information.

49. Write a SQL statement to find the names and loan numbers of all customers who have a loan at XYZ branch? (NOV 2018)

Select name, loan_no from customers where branch=”XYZ

50. What are the disadvantages of file processing system?

The disadvantages of file processing systems are

a) Data redundancy and inconsistency
b) Difficulty in accessing data
c) Data isolation
d) Integrity problems
e) Atomicity problems
f) Concurrent access anomalies

51. 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 back up and recovery

52. State the levels of data abstraction in DBMS? (NOV 2017)

a) Physical level
b) logical level
c) view level

53. Define instance and schema?

Instance: Collection of data stored in the data base at a particular moment is called an Instance of the database.

Schema: The overall design of the data base is called the data base schema.

54. Define the terms 1) physical schema 2) logical schema 3)Conceptual Schema

Physical schema: The physical schema describes the database design at the physical level, which is the lowest level of abstraction describing how the data are actually stored.

Logical schema: The logical schema describes the database design at the logical level, which describes what data are stored in the database and what relationship exists among the data.

conceptual schema:

The schemas at the view level are called subschemas that describe different views of the database.

55. Define data model?

A data model is a collection of conceptual tools for describing data, data relationships, data semantics and consistency constraints.

56. What is storage manager? What are the components of storage Manager?

A storage manager is a program module that provides the interface between the low level data stored in a database and the application programs and queries submitted to the system.

The storage manager components include

a) Authorization and integrity manager
b) Transaction manager
c) File manager
d) Buffer manager

57. What is the purpose of storage manager? List the data structures implemented by the storage manager.

The storage manager is responsible for the following

a) Interaction with he file manager
b) Translation of DML commands in to low level file system commands
c) Storing, retrieving and updating data in the database

The storage manager implements the following data structure

a) Data files
b) Data dictionary
c) indices

58. What is a data dictionary?

A data dictionary is a data structure which stores meta data about the structure of the database ie. the schema of the database.

59.Define codd's rule.

Dr.E.F.Codd founder of relational db systems,places relational models characteristic in three categories.They are

a)Structural features
b)Integrity features
c)Data manipulation features.

60.Write short notes on relational model and relational algebra?(Nov 2019)

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.

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

61. Define tuple , attribute, relation,tuple variable and Domain?

- **Attributes:** column headers
- **Tuple**: Row

- elation is a subset of a Cartesian product of list domains.
- uple variable is a variable whose domain is the set of all tuples.

.

- or each attribute there is a set of permitted values called the ***domain*** of that attribute.

62. What is a primary key, candidate key,super key and foreign key? (APR 2018)

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

Minimal super keys are called ***candidate keys.***

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

A relation schema r1 derived from an ER schema may include among its attributes the primary key of another relation schema r2.this attribute is called a ***foreign key*** from r1 referencing r2.

63. What is a SELECT and a PROJECT operation?

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

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

64. Write short notes on tuple relational calculus and domain relational calculus

The tuple relational calculation is anon procedural query language. It describes the desired information with out giving a specific procedure for obtaining that information.
A query or expression can be expressed in tuple relational calculus as {t | P (t)}

The domain relational calculus uses domain variables that take on values from an attribute domain rather than values for entire tuple.

65. Write short notes on Schema diagram.

A database schema along with primary key and foreign key dependencies can be depicted pictorially by schema diagram. Each relation appears as a box with attributes listed inside it and the relation name above it.

66. List the disadvantages of relational database system

Repetition of data

Inability to represent certain information.

67. Distinguish between key and super key? (MAY 2017)

Super Key: An attribute or set of attributes that uniquely identifies a tuple within a relation

Candidate key: A super key such that no proper subset is a super key within the relation

Primary key: The candidate key that is selected to identify tuples uniquely within the relation, the candidate keys which are not selected as Primary Key are called "Alternate keys"

68.What are the problems caused by redundancy? (NOV 2017)

The Problems caused due to redundancy are : insertion anamaly, deletion anamaly and updation anamaly.

PART - B

1. **Explain select, project and Cartesian product operations in relational algebra with an example? (NOV 2016, APR 2018)**

Select Operation (σ)

It selects tuples that satisfy the given predicate from a relation.

Notation − $\sigma_p(r)$

Where **σ** stands for selection predicate and **r** stands for relation. *p* is prepositional logic formula which may use connectors like **and, or,** and **not**. These terms may use relational operators like − =, ≠, ≥, < , >, ≤.

For example −

$\sigma_{subject = "database"}(Books)$

Output − Selects tuples from books where subject is 'database'.

$\sigma_{subject = "database" \text{ and } price = "450"}(Books)$

Output − Selects tuples from books where subject is 'database' and 'price' is 450.

$\sigma_{subject = "database" \text{ and } price = "450" \text{ or } year > "2010"}(Books)$

Output − Selects tuples from books where subject is 'database' and 'price' is 450 or those books published after 2010.

Project Operation (∏)

It projects column(s) that satisfy a given predicate.

Notation − $\prod_{A1, A2, An}$ (r)

Where A_1, A_2 , A_n are attribute names of relation **r**.

Duplicate rows are automatically eliminated, as relation is a set.

For example −

$\prod_{subject, author}$ (Books)

Selects and projects columns named as subject and author from the relation Books.

Cartesian Product (X)

Combines information of two different relations into one.

Notation − r X s

Where **r** and **s** are relations and their output will be defined as −

r X s = { q t | q ∈ r and t ∈ s}

$\sigma_{author = 'tutorialspoint'}$(Books X Articles)

Output − Yields a relation, which shows all the books and articles written by tutorialspoint.

2. **Briefly explain about views of data? (JUN 2016)**

Data Abstraction (View) in DBMS:

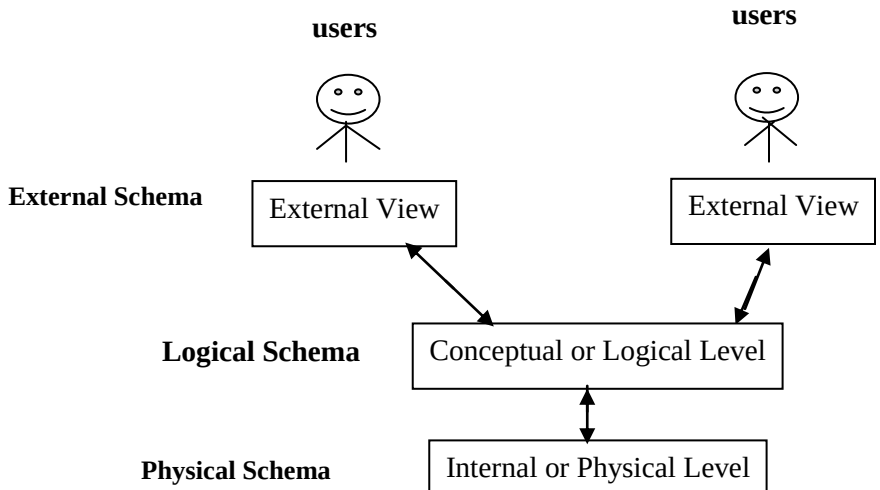
Database contains three levels of abstraction namely,

Internal or Physical level: describes the physical structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.

Conceptual or Logical level: describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations and constraints, data stored in database, and the relationships among the data.

External or View level: includes number of external schemas or user views. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group.

The below figure shows the levels of abstraction.



Mapping:

The transformation of requests and results from one level to another is called as mapping.

Schema

It is the logical structure of the database. The tow types of schema are:

- **Physical schema** represents database design at the physical level
- **Logical schema** represents database design at the logical level

Instance – the actual content of the database at a particular point in time is called the instance of the schema

Data Independence

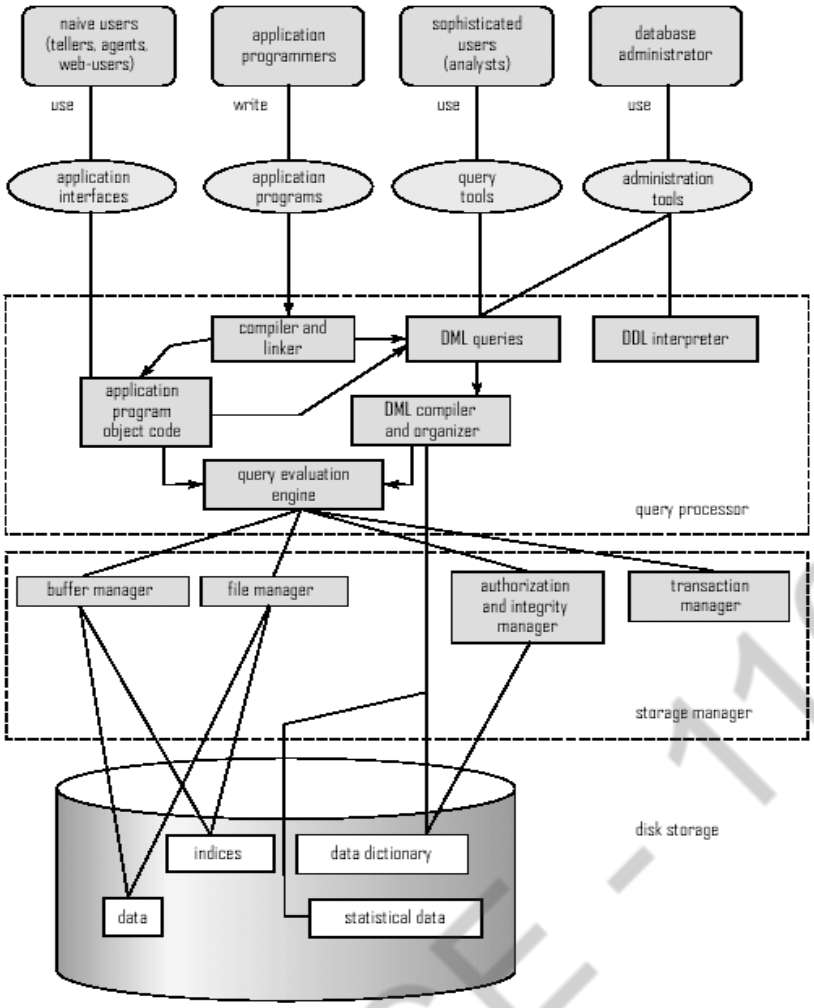
The ability to change a schema at one level without affecting the higher level schemas is called as data independence. There are two types of data independence:

- **Physical Data Independence** – is the ability to modify the physical or internal schema without changing the logical or external schemas.
- **Logical Data Independence** – is the ability to modify logical or conceptual schema without changing the external schemas or application programs.

3. **With the help of a neat block diagram explain the basic architecture of a database management system? (NOV 2015) (or)**
   Briefly explain about Database System Architecture? (JUN 2016) (or)
   Explain the overall architecture of the database system in detail? (MAY 2017) (OR)
   State and explain the architecture of DBMS? (NOV 2017, Nov 2019)

The functional components of a database system can be broadly divided into two Categories namely,

- Storage manager components
- Query Processor components



DATABASE SYSTEM ARCHITECTURE

Database Users and interfaces

There are four different types of database-system users. Different types of interfaces have been designed for the different types of users.

- **Naïve Users** are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously. The typical user interface for naïve users is a forms interface where the user can fill in appropriate fields of the form. Naïve users may also simply read reports generated from the database.

- **Application Programmers** are computer professionals who write application programs. Application Programmers can choose from many tools to develop user interfaces. Rapid Application Development (RAD) tools are tools that enable an application programmer to construct forms and reports without witting a program.

- **Sophisticated Users** interact with the system without writing programs. Instead they form requests in a database query language. They submit such queries to query processor, whose function is to break down DML statements into instructions that the storage manager understands.

- **Specialized Users** are sophisticated users who write specialized database applications that do not fit into the traditional data processing framework.

- **Database Administrator:** The person who has central control over the system is called database administrator (DBA). The functions of DBA include:

- **Schema Definition:** the DBA creates the original database schema by executing a set of data definition statements in the DDL.

- **Storage Structure and access-method definition:** the DBA is also responsible for defining the storage structure and access methods.

- **Schema and physical organization modification:** the DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve performance.

- **Granting of authorization for data access:** By granting different types of authorization, the database administrator can regulate which parts of the database various users can access. The authorization information is kept in a special system structure that the database system consults whenever someone attempts to access the data in the system.

- **Routine Maintenance:** The DBA,
  - Periodically makes the backup of the database, stores either onto tapes or onto remote servers, to prevent loss of data in case of disasters such as flooding.
  - Ensures that enough disk space is available for normal operations, and upgrades disk space as required.
  - Monitors jobs running on the database and ensures that the performance is not degraded by very expensive tasks submitted by some users.

Functional Components

Storage Manager

Storage manager is important because databases typically require a large amount of storage space. A storage manager is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.

The storage manager is responsible for the interaction with the file manager. The storage manager translates the various DML statements into low-level file system commands. Thus, the storage manager is responsible for storing retrieving, and updating data in the database.

The **storage manager components** include:

Authorization and integrity manager

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

- **Transaction Manager**

  It ensures that the database remains in a consistent state despite system failures, and that 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 responsible for fetching data from disk storage into main memory, and decides what data to cache in main memory.

The **storage manger** implements several **data structures** as part of the physical system, implementation:

- **Data files**, which store the database itself.
- **Data Dictionary**, which stores structure of the database called metadata.
- **Indices**, which provide fast access to data items that hold particular values.

Query Processor

The **query processor components** include

- **DDL interpreter**, interprets DDL statements and records the definitions in the data dictionary
- **DML compiler** translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands. The DML compiler also performs query optimization that is it picks the lowest cost evaluation plan from among the alternatives.
- **Query evaluation engine** executes low-level instructions generated by the DML compiler.

4. What are the advantages of having a centralized control of data? Illustrate your answer with suitable example? (NOV 2015)

A **centralised database** (sometimes abbreviated **CDB**) is a database that is located, stored, and maintained in a single location. This location is most often a central computer or database system, for example a desktop or server CPU, or a mainframe computer. **All** of the information stored on the CBS is accessible from a large number of different points, which in turn creates a significant amount of both advantages and disadvantages.

A centralized database consists of a single data server into which all data are stored and from which all data are retrieved. All the data reside at a single location and all applications must retrieve all data from that location.

The centralized approach consists of a central server into which all forecast data are stored. At some predefined time, software on this central server requests data from each of the local data servers scattered throughout the country. These data are received, processed and stored, possibly at lower spatial and temporal resolution than the data from which it was derived.

Typical Examples:

In most cases, a centralised database would be used by an organisation (e.g. a business company) or an institution (e.g. a university.) Users access a centralised database through a computer network which is able to give them access to the central CPU, which in turn maintains to the database itself.

DBMS itself was a centralized DBMS where all the DBMS functionality, application program execution and user interface processing carried out in one machine.

User management system.

Central Documents management system.

ADVANTAGES

Centralised databases hold a substantial amount of advantages against other types of databases. Some of them are listed below:

- Data integrity is maximised and data redundancy is minimised,[6] as the single storing place of all the data also implies that a given set of data only has one primary record. This aids in the maintaining of data as accurate and as consistent as possible and enhances data reliability.
- Generally bigger data security, as the single data storage location implies only a one possible place from which the database can be attacked and sets of data can be stolen or tampered with.
- Better data preservation than other types of databases due to often-included fault-tolerant setup.

- Easier for using by the end-user due to the simplicity of having a single database design.
- Generally easier data portability and database administration.
- More cost effective than other types of database systems as labour, power supply and maintenance costs are all minimised.
- Data kept in the same location is easier to be changed, re-organised, mirrored, or analysed.
- All the information can be accessed at the same time from the same location.[7]
- Updates to any given set of data are immediately received by every end-user.

Decreased Risk: With Centralized data management, all edits and manipulation to core data are housed and stored centrally. This model allows for staunch controls, detailed audit trails, and enables business users to access consistent data.

Data Consistency: When data feeds are managed in a central repository, an organization can achieve consistent data management and distribution throughout its global offices and internal systems.

Data Quality: A data-centric approach enables the establishment of a data standard across an enterprise, allowing organizations to make better business assessments.

Operational Efficiency: When one business unit controls an organization data centrally, the resources previously devoted to data management can be redirected back to core business needs.

Single Point of Entry: By introducing single point of entry for data, this allows changes from data vendors to be implemented once, rather than in multiple instances.

Cost Saving: With data management centralized, costs attributed to vendor relationships are better controlled, minimizing any redundancy in market data contracts and their associated costs.

Factors for Adoption of Centralized Approach:

Data can be organized in single point, by introducing single point of entry for data Database Administrator can implement the data only once instead of in multiple sites.

Data consistency can achieve by introducing data-centric approach.

Centralized database approach is suitable for establishment of data standards across an enterprise.

For batter security purpose Centralized database approach is suitable.

For quick efficient searching Centralized approach is good one.

For controlled access to the database repository.

5.Write Short notes on

i) Data model and its types (Nov 2014)

(or)

Explain the different groups of data models with suitable examples? (Apr 2019)

.DATA MODELS

A Data Model is a collection of tools for describing

- Data
- Data relationships
- Data semantics and
- Data constraints

The **various Data Models** are

- Relational model
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semi-structured data model (XML)

Relational Model

The relational model uses a collection of tables to represent both data and relationships

among the data. Each table has multiple columns, and each column has a unique name. The below table called customer table, shows, for example, that the customer identified by customer-id 100 is named john and lives at 12 anna st. in Chennai and also shows his account number

Customer_id	Customer_name	Cutomer_street	Customer_city	Account_no
100	John	12 anna st	Chennai	A-101
101	Karthik	3 main st	Chennai	A201
103	Lilly	4 north st	Chennai	A-204

The relational model is an example of a **record-based model**. The record-based models are so named because the database is structured in fixed-format records of several types. Each table contains records of particular type. Each record type defines a fixed number of fields, or attributes. The columns of the table correspond to the attributes of the record type.

The relational model is at a lower level abstraction than the E-R model. Database designs are often carried out in the E-R model, and then translated to the relational model.

Entity-Relationship data model

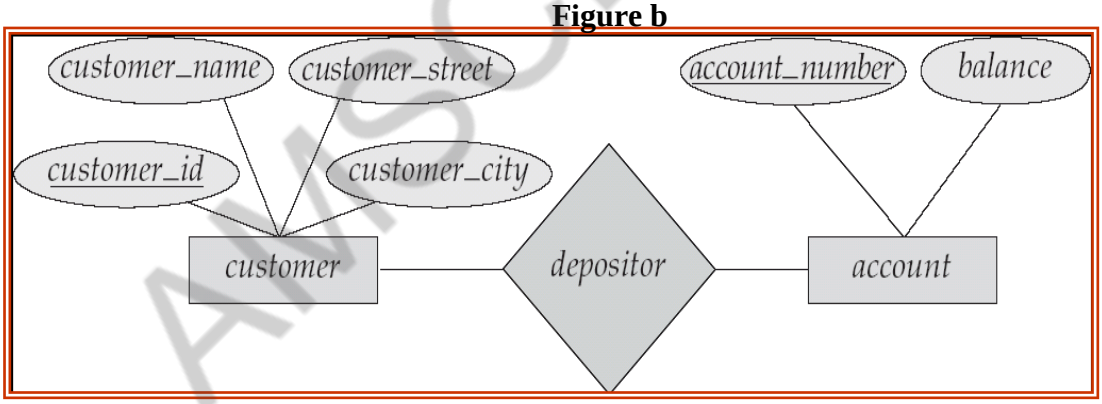
The entity-relationship (E-R) data model, models an enterprise as a collection of *entities* and *relationships.*

Entity: is a "thing" or "object" in the enterprise that is distinguishable from other objects. They are described by a set of *attributes*

Relationship: is an association among several entities

The E-R model was developed to facilitate database design. The E-R model is very useful in mapping the meanings and interactions of real world enterprises onto a conceptual schema. The E-R model is represented diagrammatically by an *entity-relationship diagram* as shown below. In the **figure (b) customer** and **account** represents **entities**, the **ellipses** represent **attributes** and **depositor** represents **relationship** among the entities.

Figure b



Object based data models

Object based data models are categorized into object-oriented data model and object-relational data model. The **object–oriented data model** can be seen as extending the E-R model with notions of encapsulation, methods or functions and object identity. The **object-relational model** extends the relational data model by including object orientation and constructs to deal with added data types.

Semi-structured data model (XML)

Extensible markup Language is defined by the WWW Consortium (W3C). It was originally intended as a document markup language and not a database language. It has the ability to specify new tags, and to create nested tag structures which made XML a great way to exchange **data**, not just documents. XML has become the basis for all new generation data interchange formats. A wide variety of tools is available for parsing, browsing and querying XML documents/data

Hierarchical Model:

Hierarchical database model is one of the oldest models, dating from 1950's. The hierarchical model assumes that a tree structure is the most frequently occurring relationship. The hierarchical model organizes data elements as tabular rows, one for each instance of an entity. Consider a company's organizational structure. At the top we have general manager (GM). Under him he has a several deputy general managers (DGM). Each DGM take care of a couple of departments and each department will have a manger and many employees. When represented in hierarchical model there will be separate rows for representing the GM, each DGM, each Department, each Manager and each Employee. The row position implies a relationship to other rows. A given employee belongs to the department that is the closest above it in the list and the department belongs to the manager immediately above it and so on.

The hierarchical model represents relationships in a lineraized tree. It is possible to locate a set of employees working for say, Manager X by first locating Manager X and then including every employee in the list after X and before the next occurrence of a manager or the end of the list.

Network Model:

The network model replaces the hierarchical tree with a graph, thus, allowing more general connections among the nodes. The main difference of the network model from the hierarchical model is its ability to handle many-to-many relationships. In other words, it allows a record to have more than one parent. Suppose an employee works for two departments. The strict hierarchical arrangement is not possible here and the tree becomes a more generalized graph-a network. Logical proximity fails because it is not possible to place a data item simultaneously in two locations in the list. Although it is possible to handle such situations in a hierarchical model, it becomes more complicated and difficult to comprehend. The network model was evolved to specifically handle non-hierarchical relationships.

In network database terminology, a relationship is a set. Each set is made of at least two types of records: an owner record and a member record.

6.Explain in detail about E. F. Codd's Twelve Rules for Relational Databases?

E. F. Codd's Twelve Rules for Relational Databases

Codd's twelve rules call for a language that can be used to define, manipulate, and query the data in the database, expressed as a string of characters. Some references to the twelve rules include a thirteenth rule – or rule zero:

1. Information Rule: All information in the database should be represented in one and only one way – as values in a table

2. Guaranteed Access Rule: Each and every datum (atomic value) is guaranteed to be logically accessible by resorting to a combination of table name, primary key value, and column name.

3. Systematic Treatment of Null Values: Null values (distinct from empty character string or a string of blank characters and distinct from zero or any other number) are supported in the fully relational DBMS for representing missing information in a systematic way, independent of data type.

4. Dynamic Online Catalog Based on the Relational Model: The database

description is represented at the logical level in the same way as ordinary data, so authorized users can apply the same relational language to its interrogation as they apply to regular data.

5. Comprehensive Data Sublanguage Rule: A relational system may support several languages and various modes of terminal use. However, there must be at least one language whose statements are expressible, per some well-defined syntax, as character strings and whose ability to support all of the following data definition, view definition, data manipulation (interactive and by program), integrity constraints, authorization , transaction boundaries (begin, commit, and rollback).

6. View Updating Rule: All views that are theoretically updateable are also updateable by the system.

7. High-Level Insert, Update, and Delete: The capability of handling a base relation or a derived relation as a single operand applies not only to the retrieval of data, but also to the insertion, update, and deletion of data.

8. Physical Data Independence: Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage representation or access methods.

9. Logical Data Independence: Application programs and terminal activities remain logically

unimpaired when information preserving changes of any kind that theoretically permit

unimpairment are made to the base tables.

10. Integrity Independence: Integrity constraints specific to a particular relational database must be definable in the relational data sublanguage and storable in the catalog, not in the application programs.

11. Distribution Independence: The data manipulation sublanguage of a relational DBMS must enable application programs and terminal activities to remain logically unimpaired whether and whenever data are physically centralized or distributed.

12. Nonsubversion Rule: If a relational system has or supports a low-level (single-record-at-a-time) language, that low-level language cannot be used to subvert or bypass the integrity rules or constraints expressed in the higher-level (multiple-records-at-a-time) relational language.

7.Explain in detail about Relational Algebra? (NOV 2019)

(or)

List the operations of relational algebra and the purpose of each with example? (MAY 2017)

Relational Algebra

Relational algebra is a procedural query language, which takes instances of relations as input and yields instances of relations as output. It uses operators to perform queries. An operator can be either **unary** or **binary**. They accept relations as their input and yield relations as their output. Relational algebra is performed recursively on a relation and intermediate results are also considered relations.

The fundamental operations of relational algebra are as follows −

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

We will discuss all these operations in the following sections.

Select Operation (σ)

It selects tuples that satisfy the given predicate from a relation.

Notation − $\sigma_p(r)$

Where **σ** stands for selection predicate and **r** stands for relation. *p* is prepositional logic formula which may use connectors like **and, or,** and **not**. These terms may use relational operators like − =, ≠, ≥, < , >, ≤.

For example −

$\sigma_{subject = "database"}(Books)$

Output − Selects tuples from books where subject is 'database'.

$\sigma_{subject = "database" \text{ and } price = "450"}(Books)$

Output − Selects tuples from books where subject is 'database' and 'price' is 450.

$\sigma_{subject = "database" \text{ and } price = "450" \text{ or } year > "2010"}(Books)$

Output − Selects tuples from books where subject is 'database' and 'price' is 450 or those books published after 2010.

Project Operation (∏)

It projects column(s) that satisfy a given predicate.

Notation − $\prod_{A1, A2, An}(r)$

Where A_1, A_2 , A_n are attribute names of relation **r**.

Duplicate rows are automatically eliminated, as relation is a set.

For example −

$\prod_{subject, author}(Books)$

Selects and projects columns named as subject and author from the relation Books.

Union Operation (∪)

It performs binary union between two given relations and is defined as −

$$r \cup s = \{ t \mid t \in r \text{ or } t \in s\}$$

Notation − r U s

Where **r** and **s** are either database relations or relation result set (temporary relation).

For a union operation to be valid, the following conditions must hold −

- **r**, and **s** must have the same number of attributes.
- Attribute domains must be compatible.
- Duplicate tuples are automatically eliminated.

$$\prod_{\text{author}} (\text{Books}) \cup \prod_{\text{author}} (\text{Articles})$$

Output − Projects the names of the authors who have either written a book or an article or both.

Set Difference (−)

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

Notation − **r** − **s**

Finds all the tuples that are present in **r** but not in **s**.

$$\prod_{\text{author}} (\text{Books}) - \prod_{\text{author}} (\text{Articles})$$

Output − Provides the name of authors who have written books but not articles.

Cartesian Product (X)

Combines information of two different relations into one.

Notation − r X s

Where **r** and **s** are relations and their output will be defined as −

$$r \text{ X } s = \{ q\ t \mid q \in r \text{ and } t \in s\}$$

$$\sigma_{\text{author = 'tutorialspoint'}}(\text{Books X Articles})$$

Output − Yields a relation, which shows all the books and articles written by tutorialspoint.

Rename Operation (ρ)

The results of relational algebra are also relations but without any name. The rename operation allows us to rename the output relation. 'rename' operation is denoted with small Greek letter **rho** ρ.

Notation − $\rho_{\ x}$ (E)

Where the result of expression **E** is saved with name of **x**.

Additional operations are −

- Set intersection
- Assignment
- Natural join

Relational Calculus

In contrast to Relational Algebra, Relational Calculus is a non-procedural query language, that is, it tells what to do but never explains how to do it.

Relational calculus exists in two forms −

Tuple Relational Calculus (TRC)
Filtering variable ranges over tuples

Notation − {T | Condition}

Returns all tuples T that satisfies a condition.

For example −

```
{ T.name |  Author(T) AND T.article = 'database' }
```

Output − Returns tuples with 'name' from Author who has written article on 'database'.

TRC can be quantified. We can use Existential (∃) and Universal Quantifiers (∀).

For example −

```
{ R| ∃T  ∈ Authors(T.article='database' AND R.name=T.name)}
```

Output − The above query will yield the same result as the previous one.

Domain Relational Calculus (DRC)

In DRC, the filtering variable uses the domain of attributes instead of entire tuple values (as done in TRC, mentioned above).

Notation −

$\{ a_1, a_2, a_3, ..., a_n \mid P (a_1, a_2, a_3, ... ,a_n)\}$

Where a1, a2 are attributes and **P** stands for formulae built by inner attributes.

For example −

```
{< article, page, subject > |  ∈ TutorialsPoint ∧ subject = 'database'}
```

Output − Yields Article, Page, and Subject from the relation TutorialsPoint, where subject is database.

Just like TRC, DRC can also be written using existential and universal quantifiers. DRC also involves relational operators.

The expression power of Tuple Relation Calculus and Domain Relation Calculus is equivalent to Relational Algebra.

8.Explain the various types of join? (APR 2018)

JOIN Operation

- The sequence of cartesian product followed by select is used quite commonly to identify and select related tuples from two relations, a special operation, called **JOIN**. It is denoted by a
- This operation is very important for any relational database with more than a single relation, because it allows us to process relationships among relations.
- The general form of a join operation on two relations $R(A_1, A_2, \ldots, A_n)$ and $S(B_1, B_2, \ldots, B_m)$ is:

  $R \ _{<\text{join condition}>} S$ where R and S can be any relations that result from general *relational algebra expressions.*

Example: Suppose that we want to retrieve the name of the manager of each department. To get the manager's name, we need to combine each DEPARTMENT tuple with the EMPLOYEE tuple whose SSN value matches the MGRSSN value in the department tuple. We do this by using the join operation.

DEPT_MGR ← DEPARTMENT $_{\text{MGRSSN=SSN}}$ EMPLOYEE

DEPT_MGR	DNAME	DNUMBER	MGRSSN	. . .	FNAME	MINIT	LNAME	SSN	. . .
	Research	5	333445555	. . .	Franklin	T	Wong	333445555	. . .
	Administration	4	987654321	. . .	Jennifer	S	Wallace	987654321	. . .
	Headquarters	1	888665555	. . .	James	E	Borg	888665555	. . .

FIGURE 6.6 Result of the JOIN operation DEPT_MGR ← DEPARTMENT $\bowtie_{\text{MGRSSN=SSN}}$ EMPLOYEE.

- **EQUIJOIN Operation**

The most common use of join involves join conditions with equality comparisons only. Such a join, where the only comparison operator used is =, is called an EQUIJOIN. In the result of an EQUIJOIN we always have one or more pairs of attributes (whose names need not be identical) that have *identical values* in every tuple. The JOIN seen in the previous example was EQUIJOIN.

- **NATURAL JOIN Operation**

Because one of each pair of attributes with identical values is superfluous, a new operation called natural join—denoted by *—was created to get rid of the second (superfluous) attribute in an EQUIJOIN condition. The standard definition of natural join requires that the two join attributes, or each pair of corresponding join attributes, have the **same name** in both relations. If this is not the case, a renaming operation is applied first.

To apply a natural join on the DNUMBER attributes of DEPARTMENT and DEPT_LOCATIONS, it is sufficient to write: **DEPT_LOCS ← DEPARTMENT * DEPT_LOCATIONS**

(a) PROJ_DEPT

PNAME	PNUMBER	PLOCATION	DNUM	DNAME	MGRSSN	MGRSTARTDATE
ProductX	1	Bellaire	5	Research	333445555	1988-05-22
ProductY	2	Sugarland	5	Research	333445555	1988-05-22
ProductZ	3	Houston	5	Research	333445555	1988-05-22
Computerization	10	Stafford	4	Administration	987654321	1995-01-01
Reorganization	20	Houston	1	Headquarters	888665555	1981-06-19
Newbenefits	30	Stafford	4	Administration	987654321	1995-01-01

(b) DEPT_LOCS

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE	LOCATION
Headquarters	1	888665555	1981-06-19	Houston
Administration	4	987654321	1995-01-01	Stafford
Research	5	333445555	1988-05-22	Bellaire
Research	5	333445555	1988-05-22	Sugarland
Research	5	333445555	1988-05-22	Houston

FIGURE 6.7 Results of two NATURAL JOIN operations. (a) PROJ_DEPT ← PROJECT * DEPT. (b) DEPT_LOCS ← DEPARTMENT * DEPT_LOCATIONS.

The set of operations including **select s, project p , union È, set difference - , and Cartesian product X** is called a complete set because any other relational algebra expression can be expressed by a combination of these five operations.

- For example:

R Ç S = (R È S) – ((R - S) È (S - R))

R $_{\text{<join condition>}}$S = **s** $_{\text{<join condition>}}$ **(R X S)**

The OUTER JOIN Operation

- In NATURAL JOIN tuples without a *matching* (or *related*) tuple are eliminated from the join result. Tuples with null in the join attributes are also eliminated. This amounts to loss of information.
- A set of operations, called outer joins, can be used when we want to keep all the tuples in R, or all those in S, or all those in both relations in the result of the join, regardless of whether or not they have matching tuples in the other relation.
- The left outer join operation keeps every tuple in the *first* or *left* relation R in R S; if no matching tuple is found in S, then the attributes of S in the join result are filled or "padded" with null values.
- A similar operation, right outer join, keeps every tuple in the *second* or right relation S in the result of R S.
- A third operation, full outer join, denoted by keeping all tuples in both the left and the right relations when no matching tuples are found, padding them with null values as needed.

RESULT	FNAME	MINIT	LNAME	DNAME
	John	B	Smith	null
	Franklin	T	Wong	Research
	Alicia	J	Zelaya	null
	Jennifer	S	Wallace	Administration
	Ramesh	K	Narayan	null
	Joyce	A	English	null
	Ahmad	V	Jabbar	null
	James	E	Borg	Headquarters

Additional Relational operations

- **Aggregate Functions and Grouping**
  - A type of request that cannot be expressed in the basic relational algebra is to specify mathematical **aggregate functions** on collections of values from the database.
  - Examples of such functions include retrieving the average or total salary of all employees or the total number of employee tuples. These functions are used in simple statistical queries that summarize information from the database tuples.
  - Common functions applied to collections of numeric values include SUM, AVERAGE, MAXIMUM, and MINIMUM. The COUNT function is used for counting tuples or values.

(a)

R	DNO	NO_OF_EMPLOYEES	AVERAGE_SAL
	5	4	33250
	4	3	31000
	1	1	55000

(b)

DNO	COUNT_SSN	AVERAGE_SALARY
5	4	33250
4	3	31000
1	1	55000

(c)

COUNT_SSN	AVERAGE_SALARY
8	35125

8.Consider the given relation schema . (MAY 2017, NOV 2018)

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

Write the following queries in SQL

(a) Find the name of all employees who have borrowed a book published by McGraw-Hill.
(b) Find the name of all employees who have borrowed all book published by McGraw-Hill.
(c) Find the names of employees who have borrowed more than five different books published by McGraw-Hill.
(d) For each publisher, find the name of employees who have borrowed more than five books of that publisher.

a. select name from employee e, books b, loan l where e.empno = l.empno and l.isbn = b.isbn and b.publisher = 'McGrawHill'

b. select name from employee e join loan l on e.empno=l.empno join (select isbn from books where publisher = 'McGrawHill') x on l.isbn=x.isbn group by e.empno,name having count(*)= (select count(*) from books where publisher='McGrawHill')

c. select name from employee,loan where employee.empno=loan.empno and isbn in (select distinct isbn from books where publisher='McGraw-Hill') group by employee.empno,name having count(isbn) >=5

d. select name from employee,loan,books where employee.empno=loan.empno and books.isbn=loan.isbn group by employee.empno, name,books.publisher having count(loan.isbn) >=5

9.Explain the following with examples:

i)DDL
ii)DML
iii)Embedded SQL (NOV 2014)

i) Data definition language:

The data definition language is used to specify a database schema by a set of definitions.

The storage structure and access methods used by the database system, is specified by a set of statements in a special type of DDL called a data storage and definition language. These statements define the implementation details of the database schemas, which are usually hidden from the users.

The data values stored in the database must satisfy certain constraints. For example, suppose the balance on an account should not fall below $1100. The DDL provides facilities to specify such constraints. The database systems check these constraints every time the database is updated.

The various constraints involved in the database is given below:

Domain Constraints:

A domain of possible values must be associated with every attribute (for example, integer types, character types, date/time types). Declaring an attribute to be of particular domain acts as a constraint on the values it can take. Domain constraints are tested easily by the system whenever a new data item is entered into the database.

Referential Integrity:

The referential integrity is used to ensure that a value that appears in one relation for a given set of attributes also appears for a certain set of attributes in another relation. Database modifications can cause violations of referential integrity. When a referential integrity constraint is violated, the database rejects the action that caused the violation.

Assertion:

An assertion is any condition that the database must always satisfy. When an assertion is created, the system tests it for validity. If the assertion is valid, then any future modification to the database is allowed only if it does not that assertion to be violated.

Authorization:

Authorization specifies what types of access are permitted for users on various data values in the database. The authorizations can be categorized as:

- Read authorization, allows reading, but not modification of existing data.
- Update authorization, allows insertion of new data, but not modification of existing data.
- Delete authorization, allows deletion of data.

The DDL gets as input some instructions and generates some output. The output of the DDL is placed in the ***data dictionary***, which contains ***metadata***- that is data about data. The data dictionary is considered to be special type of table, which can only be accessed and updated by the database system itself. A database system consults the data dictionary before reading or modifying actual data.

ii) Data Manipulation Language:

A data manipulation language (DML) is a language that enables users to access or manipulate data organized by the appropriate data model. The types of access are:

- Retrieval of information stored in the database
- Insertion of new information into the database
- Deletion of information from the database
- Modification of information stored in the database

There are basically two types:

Procedural DMLs require a user to specify what data are needed and how to get

those data.

Declarative DMLs (also referred to as nonprocedural DMLs) require a user to specify what data are needed without specifying how to get those data.

A ***query*** is a statement requesting the retrieval of information. The portion of a DML that involves information retrieval is called a query language.

Embedded SQL:

An embedded SQL program must be processed by a special preprocessor prior to compilation. The preprocessor replaces embedded SQL requests with host-language declarations and procedure calls that allow run-time execution of the database accesses. Then, the resulting program is compiled by the host language compiler. To identify embedded SQL requests to the preprocessor, we use the EXEC SQL statement. It has the form

EXEC SQL <embedded SQL statement> END-EXEC

The exact syntax for embedded SQL requests depends on the language in which SQL is embedded. For instance, a semicolon is used instead of END-EXEC when SQL is embedded in C. The java embedding of SQL called (SQLJ) uses the syntax

#SQL{<embedded SQL statement>}

The statement SQL INCLUDE is placed in the program to identify the place where the preprocessor should insert the special variables used for communication between the program and the database system. Variables of the host language can be used within embedded SQL statements, but they must be preceded by a colon (:) to distinguish them from SQL variables.

Before executing any SQL statements, the program must first connect to the database. This is done using

EXEC SQL **connect to** server **user** user-name END-EXEC

10.Expalin the aggregate functions in SQL with an example(APR 2018)

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

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.

- **Aggregate Functions**
  - A type of request that cannot be expressed in the basic relational algebra is to specify mathematical **aggregate functions** on collections of values from the database.
  - Examples of such functions include retrieving the average or total salary of all employees or the total number of employee tuples. These functions are used in simple statistical queries that summarize information from the database tuples.
  - Common functions applied to collections of numeric values include SUM, AVERAGE, MAXIMUM, and MINIMUM. The COUNT function is used for counting tuples or values.

(a)

R	DNO	NO_OF_EMPLOYEES	AVERAGE_SAL
	5	4	33250
	4	3	31000
	1	1	55000

(b)

DNO	COUNT_SSN	AVERAGE_SALARY
5	4	33250
4	3	31000
1	1	55000

(c)

COUNT_SSN	AVERAGE_SALARY
8	35125

Example:

1. Consider the following SQL query on the EMPLOYEE relation:

SELECT Lname, Fname
FROM EMPLOYEE
WHERE Salary > (**SELECT MAX** (Salary)
FROM EMPLOYEE
WHERE Dno=5);

This query retrieves the names of employees (from any department in the company) who earn a salary that is greater than the *highest salary in department 5*. The query includes a nested subquery and hence would be decomposed into two blocks.

2. **Find the name of all employees who have borrowed all book published by McGraw-Hill.**

Select name from employee e join loan l on e.empno=l.empno join (select isbn from books where publisher = 'McGrawHill') x on l.isbn=x.isbn group by e.empno,name having count(*)= (select count(*) from books where publisher=’McGrawHill’)

11.Describe the six clauses in the syntax of an SQL query, and show what type of constructs can be specified in each of the six clauses. Which of the six clauses are required and which are optional? (NOV 2015)

The six clauses of the SELECT statement

There are six clauses that can be used in an SQL statement. These six clauses are SELECT, FROM, WHERE, GROUP BY, HAVING, and ORDER BY. Clauses must be coded in a specific sequence.

1. SELECT column name(s)*
2. FROM table or views
3. WHERE conditions or predicates are met
4. GROUP BY subsets of rows
5. HAVING a common condition as a group
6. ORDER BY a sorting method

column name(s) are more correctly referred to as elements, because the SELECT statement displays both columns that exist in the table and columns that may be generated by SQL as a result of a query.

Example query

```
SELECT perkey, sum(dollars)
FROM aroma.sales
WHERE perkey < 50
GROUP BY perkey
HAVING sum(dollars) > 8000
ORDER BY perkey;
```

The SELECT clause is where you list the columns you're interested in. The SELECT displays what you put here.

In the example, the perkey column, as well as the sum of the dollar column, are selected.

The FROM clause indicates the table you're getting your information from. You can list more than one table. The number of tables you could list is specific to your operating system. In the example, both columns are selected from the Sales table.

SELECT and FROM are required; the rest of these clauses are optional and serve to filter or limit, aggregate or combine, and control the sort.

The **WHERE** clause is where you indicate a condition. This helps you filter unwanted data from the results. WHERE gives you a subset of the rows in a table. In the example, only rows with a perkey value of less than 50 are selected.

GROUP BY allows you to group your data to achieve more meaningful results. Instead of getting a total sum of the dollar sales for all the rows selected, you can break down sales by perkey to get the daily sales total. This is done in the example by indicating GROUP BY perkey.

HAVING puts a condition on your groups. In the example, only those days that have a total dollar amount greater than 8,000 are returned.

ORDER BY orders your result rows. You can choose to order results by ASC (ascending) or DESC (descending) order. The default is ASC.

The SELECT statement is the most common usage of data manipulation language (DML). Other DML statements (UPDATE, INSERT, and DELETE) and the other two components of
SQL (data definition language and control language) .

12.Explain about SQL Fundamentals? (JUN 2016) (OR)
State and Explain DDL, DML, DCL with suitable examples. (NOV 2017), (NOV 2019)

□□The standard for relational database management systems (RDBMS)
□□SQL-92 and SQL-99 Standards –

Purpose:

□□Specify syntax/semantics for data definition and manipulation
□□Define data structures
□□Enable portability
□□Specify minimal (level 1) and complete (level 2) standards
□□Allow for later growth/enhancement to standard

□□Catalog
□□A set of schemas that constitute the description of a database

□□Schema
□□The structure that contains descriptions of objects created by a user (base tables, views, constraints)

□□Data Definition Language (DDL)
□□Commands that define a database, including creating, altering, and dropping tables and establishing constraints

□□Data Manipulation Language (DML)
□□Commands that maintain and query a database

□□Data Control Language (DCL)
□□Commands that control a database, including administering privileges and committing Data

SQL Database Definition

□□Data Definition Language (DDL)
□□Major CREATE statements:
□□CREATE SCHEMA – defines a portion of the database owned by a particular user
□□CREATE TABLE – defines a table and its columns
□□CREATE VIEW – defines a logical table from one or more views

□□Other CREATE statements:
CHARACTER SET, COLLATION, TRANSLATION,ASSERTION, DOMAIN

□□In SQL, a VIEW is a virtual relation based on the result-set of a SELECT statement.
□□A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database. In some cases, we can modify a view and present the data as if the data were coming from a single table.

□□Syntax:
CREATE VIEW view_name **AS SELECT** column_name(s) **FROM** table_name **WHERE** Condition

SQL – Relations, Tables & Views

□□When we say Relation, it could be a Table or a View. There are three kind of relations:

1. Stored relations □□tables
We sometimes use the term ―base relation‖ or ―base table‖
1. Virtual relations □□views
2. Temporary results

SQL – Create View

□□Example: Create a view with title and year and made by Paramount studio. Movie (title, year, length, inColor, studioName, producerC#)

CREATE VIEW ParamountMovie **AS SELECT** title,year **FROM** Movie **WHERE** studioName =_Paramount‘;

SQL – Querying View

□□A view could be used from inside a query, a stored procedure, or from inside another view. By adding functions, joins, etc., to a view, it allows us to present exactly the data we want to the user.

SELECT title **FROM** ParamountMovie **WHERE** year = _1979‘;

□□Have same result as

SELECT title **FROM** Movie **WHERE** studioName = _Paramount‘ **AND** year = _1979‘;

Example:

Movie (title, year, length, inColor, studioName, producerC#) MovieExec (name, address, cert#, netWorth)

CREATE VIEW MovieProd **AS SELECT** title, name **FROM** Movie, MovieExec **WHERE** producerC# = cert#; **SELECT** name **FROM** MovieProd **WHERE** title = _Gone With the Wind‘;

□□Same result as query from tables

SELECT name **FROM** Movie, MovieExec **WHERE** producerC# = cert# **AND** title = _The War Of the World‘;

Data Types in SQL

□□String types
□□CHAR(n) – fixed-length character data, n characters long Maximum length = 2000 bytes
□□VARCHAR2(n) – variable length character data, maximum 4000 bytes
□□LONG – variable-length character data, up to 4GB. Maximum 1 per table
□□Numeric types
□□NUMBER(p,q) – general purpose numeric data type
□□INTEGER(p) – signed integer, p digits wide
□□FLOAT(p) – floating point in scientific notation with p binary digits precision

□□Date/time type
□□DATE – fixed-length date/time in dd-mm-yy form

13.Explain about Data Definition Language? (JUN 2016)

Data Definition Language:

A data definition language or data description language (DDL) is a syntax similar to a computer programming language for defining data structures, especially database schemas. Many data description languages use a declarative syntax to define fields and data types.

SQL, however, uses a collection of imperative verbs whose effect is to modify the schema of the database by adding,changing, or deleting definitions of tables or other objects. These statements can be freely mixed with other SQL statements, so the DDL is not truly a separate language.

CREATE statements

Create - To make a new database, table, index, or stored procedure.

A CREATE statement in SQL creates an object in a relational database management system (RDBMS). In the SQL 1992 specification, the types of objects that can be created are schemas, tables, views, domains, character sets, collations, translations, and assertions. Many implementations extend the syntax to allow creation of additional objects, such as indexes and user profiles. Some systems (such as PostgreSQL) allow CREATE, and other DDL commands, inside a transaction and thus they may be rolled back.

CREATE TABLE statement

A commonly used CREATE command is the CREATE TABLE command. The typical usage is:

CREATE TABLE [table name] ([column definitions]) [table parameters].

column definitions: A comma-separated list consisting of any of the following
Column definition: [column name] [data type] {NULL | NOT NULL} {column options}
Primary key definition: PRIMARY KEY ([comma separated column list])
Constraints: {CONSTRAINT} [constraint definition]

RDBMS specific functionality For example, the command to create a table named employees with a few sample columns would be:

CREATE TABLE employees (id INTEGER PRIMARY KEY,first_name VARCHAR(50) NULL, last_name VARCHAR(75) NOT NULL,
fname VARCHAR(50) NOT NULL,dateofbirth DATE NULL);

DROP statements

Drop - To destroy an existing database, table, index, or view.

A DROP statement in SQL removes an object from a relational database management system (RDBMS). The types of objects that can be dropped depends on which RDBMS is being used, but most support the dropping of tables, users, and databases. Some systems (such as PostgreSQL) allow DROP and other DDL commands to occur inside of a transaction and

thus be rolled back.

DROP objecttype objectname.

For example, the command to drop a table named employees would be:

DROP employees;

The DROP statement is distinct from the DELETE and TRUNCATE statements, in that DELETE and TRUNCATE do not remove the table itself. For example, a DELETE statement might delete some (or all) data from a table while leaving the table itself in the database, whereas a DROP statement would remove the entire table from the database.

ALTER statements

Alter - To modify an existing database object.
An ALTER statement in SQL changes the properties of an object inside of a relational database management system (RDBMS). The types of objects that can be altered depends on which RDBMS is being used.

The typical usage is:

ALTER objecttype objectname parameters.

For example, the command to add (then remove) a column named bubbles for an existing table named sink would be:

ALTER TABLE sink ADD bubbles INTEGER;
ALTER TABLE sink DROP COLUMN bubbles;

Rename statement

Rename - to rename the table. for example
RENAME TABLE old_name TO new_name;

Referential integrity statements

Finally, another kind of DDL sentence in SQL is one used to define referential integrity relationships, usually implemented as primary key and foreign key tags in some columns of the tables. These two statements can be included inside a CREATE TABLE or an ALTER TABLE sentence.

14.What is embedded SQL? Give an example?(Nov 2016)
(or)
Justify the need of Embedded SQL? (May 2017) (or)
Justify the need of Embedded SQL? Write Embedded dynamic SQL statement in C to retrieve all the student records whose mark is more than 90? (NOV 2017)

SQL provides a powerful declarative query language. Writing queries in SQL is usually much easier than coding the same queries in a general-purpose language. However, a programmer must have

access to a database from a general purpose programming language for at least two reasons:

1. Not all queries can be expressed in SQL, since SQL does not provide the full expressive power of a general purpose language. That is, there exist queries that can be expressed in a language such as C, java, or COBOL that cannot be expressed in SQL. To write such queries, we can embed SQL within a more powerful language.

2. Non declarative actions-such as printing a report, interacting with a user, or sending the results of a query to a graphical user interface cannot be done from within SQL.

The SQL standard defines embeddings of SQL in a variety of programming languages such as Pascal, PL/I, FORTRAN, C, and COBOL. A language to which SQL queries are embedded is referred to as a ***host language***, and the SQL structures permitted in the host language comprise ***embedded SQL.***

Programs written in the host language can use the embedded SQL syntax to access and update data stored in a database. This embedded form of SQL extends the programmer's ability to manipulate the database even further. In embedded SQL, all query processing is performed by the database system, which then makes the result of the query available to the programmer one tuple at a time.

An embedded SQL program must be processed by a special preprocessor prior to compilation. The preprocessor replaces embedded SQL requests with host-language declarations and procedure calls that allow run-time execution of the database accesses. Then, the resulting program is compiled by the host language compiler. To identify embedded SQL requests to the preprocessor, we use the EXEC SQL statement. It has the form

EXEC SQL <embedded SQL statement> END-EXEC

The exact syntax for embedded SQL requests depends on the language in which SQL is embedded. For instance, a semicolon is used instead of END-EXEC when SQL is embedded in C. The java embedding of SQL called (SQLJ) uses the syntax

#SQL{<embedded SQL statement>}

The statement SQL INCLUDE is placed in the program to identify the place where the preprocessor should insert the special variables used for communication between the program and the database system. Variables of the host language can be used within embedded SQL statements, but they must be preceded by a colon (:) to distinguish them from SQL variables.

Before executing any SQL statements, the program must first connect to the database. This is done using

EXEC SQL **connect to** server **user** user-name END-EXEC

Here server identifies the server to which a connection is to be established. Database implementations may require a password to be provided in addition to a user name.

To write a relational query, the declare cursor statement is used. The program must use the open and fetch commands to obtain the result tuples.

Consider the banking schema. Assume that we have a host language variable amount, and that we wish to find the names and cities of residence of customers who have more than amount dollars in any account. The query can be written as follows:

EXEC SQL

```
Declare c cursor for
    select 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 and to save the result in the temporary relation.

EXEC SQL **open** *c* END-EXEC

If the SQL statement results in an error the database system stores an error diagnostic in the SQL communication area (SQLCA) variables, whose declarations are inserted by the SQL include statement.

- 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***

The fetch statement requires one host-language variable for each attribute of the result relation. In the example query *cn* ***holds the*** *customer-name* ***and*** *cc* ***holds the*** *customer-city.*

To obtain all tuples of the result, the program must contain a loop to iterate over tuples. When the program executes an open statement on a cursor, the cursor is set to point to the first tuple of the result. Each time it executes a fetch statement, the cursor is updated to point to the next tuple of the result. When no further tuples remain to be processed, the variable called SQLSTATE in the SQL communication area (SQLCA) gets set to '02000' to indicate no more data is available.

- 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

Updates through Cursors

- Embedded SQL can update tuples fetched by cursor by declaring that the cursor is for update as shown below:

declare c *cursor for*
select * *from* account *where* branch-name = *'Perryridge'*
for update

- *We then iterate through the tuples by performing fetch operations on the cursor, and after fetching each tuple the following code is executed.*
  update account *set* balance = balance + *100 where current of* c

15.Explain in detail about Data Manipulation Language?

DATA MANIPULATION LANGUAGE (DML):

The language used to manipulate data is called DML Manipulation of data includes the insertion of data into the database, modification of data stored in the database, deletion of data from the database and retrieval of data for further process. The commands used are

- ✓ Insert
- ✓ Update
- ✓ Delete

#THE INSERT COMMAND

The INSERT is used to add a single tuple to a relation. The values should be listed *in the same order* in which the corresponding attributes were specified in the CREATE TABLE command. There are three ways to insert a record into the database.

1. **Inserting Data's in specified columns**:

Syntax:

INSERT INTO table_name(col1,col2,…….,coln)
VALUES(val1,val2,……,valn);

2. **Inserting Values:**

Syntax:

INSERT INTO table_name VALUES(val1,val2,……...,valn);

3**. Inserting Multiple Rows**:

Syntax:

INSERT INTO tablename VALUES(&col1,'&col2','&col3',…….,&coln);

***For example,* to add a new tuple to the EMPLOYEE relation.**

INSERT INTO EMPLOYEE **VALUES** ('Richard', 'K', 'Marini', '653298653', '1962-12-30', '98 Oak Forest, Katy, TX', 'M', 37000, '653298653', 4);

#THE DELETE COMMAND

The DELETE command removes tuples from a relation. It includes a WHERE clause, similar to that used in an SQL query, to select the tuples to be deleted. Tuples are explicitly deleted from only one table at a time.

Depending on the number of tuples selected by the condition in the WHERE clause, zero, one, or several tuples can be deleted by a single DELETE command. A missing WHERE clause specifies that all tuples in the relation are to be deleted; however, the table remains in the database as an empty table.

Syntax:

DELETE FROM table_name
WHERE column name=some value;

Example:

DELETE FROM EMPLOYEE **WHERE** Lname='Brown';

#THE UPDATE COMMAND

The **UPDATE** command is used to modify attribute values of one or more selected tuples.An additional **SET** clause in the UPDATE command specifies the attributes to be modified and their new values.

Syntax:

UPDATE table_name
SET column name =new value
WHERE column name=some value;

For example, to change the location and controlling department number of project number 10 to 'Bellaire' and 5, respectively.

UPDATE PROJECT **SET** Plocation = 'Bellaire', Dnum = 5 **WHERE** Pnumber=10;

#THE SELECT STATEMENT

SQL has one basic statement for retrieving information from a database: the **SELECT** statement The basic form of the SELECT statement, sometimes called a **mapping** or a **select-from-where block**, is formed of the three clauses SELECT, FROM, and WHERE and has the following form:

SELECT <attribute list>
FROM <table list>
WHERE <condition>;

where

■ <attribute list> is a list of attribute names whose values are to be retrieved by the query.
■ <table list> is a list of the relation names required to process the query.
■ <condition> is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query

Example:

Retrieve the birth date and address of the employee(s) whose name is 'John B. Smith'.

Q0: SELECT Bdate, Address **FROM** EMPLOYEE **WHERE** Fname='John' **AND** Minit='B' **AND** Lname='Smith';

Bdate	Address
1965-01-09	731Fondren, Houston, TX

16.Explain the various ways in which Select Statement can be used?

The SELECT statement can be used in many ways. They are:

1. Selecting some columns: To select specified number of columns from the table the following command is used.

Syntax:

SELECT column name **FROM** table_name;

2**. Query All Columns**:To select all columns from the table * is used instead of column names.

Syntax:

SELECT * FROM table_name

3. **Select using DISTINCT:**The DISTINCT keyword is used to return only different values (i.e.) this command does not select the duplicate values from the table .

Syntax:

SELECT DISTINCT column name(s) **FROM** table_name;

4. **Select using IN:**If you want to get the rows, which contain certain values, the best way to do it is to use the IN conditional expression.

Syntax:

SELECT column name(s) **FROM** table_name **WHERE**
Column name **IN(**value1,value2,……,value n);

5**. Select using BETWEEN**:BETWEEN can be used to get those items that fall within a range.

Syntax:

SELECT column name **FROM** table_name **WHERE**
Column name **BETWEEN** value1 **AND** value2;

6. **Renaming:**The select statement can be used to rename either a column or the entire table.

***Syntax*:**

Renaming a column: SELECT column name **AS** new name **FROM** table_name;

Renaming a table: SELECT column name **FROM** table_name **AS** newname;

7**. Sorting**: The select statement with the **order by Clause** is used to sort the contents of the table either in ascending or descending order.

Syntax:

SELECT column name **FROM** table_name **WHERE**
Condition ORDER BY column name **ASC/DESC;**

8. **To Select by matching some patterns**:The select statement along with **like clause** I is used to match strings. The **like** condition is used to specify a search pattern in a column.

Syntax:

SELECT column name FROM table_name WHERE
Column name LIKE "% qr-";
% : Matches any sub string.

- : Matches a single character.

9**. SELECT INTO statement:**The SELECT INTO statement is most often used to create backup copies of tables or for archiving records.

Syntax:

SELECT Column_name(s) INTO variable_name(s)
FROM table_name
WHERE condition.

10**. To Select NULL values**:We can use the SELECT statement to select the 'null' values also. For retrieving rows where some of the columns have been defined as NULLs there is a special comparison operator of the form IS[NOT]NULL.

Syntax:

SELECT column name FROM table_name WHERE
Column name IS NULL;

11. **Select using AND, OR, NOT**:We can combine one or more conditions in a SELECT statement using the logical operators AND,OR,NOT.

Syntax:

SELECT column name FROM table_name WHERE
Condition1 LOGICAL OPERATOR condition2;

UNIT – 2

DATABASE DESIGN

PART - A

1.Why 4NF in Normal Form is more desirable than BCNF? (NOV 2014)

Because 4NF minimize the redundancy as well as make storage management. Redundancy is reduced as we normalize it further and this avoids consistency problems.

2.Define functional Dependency? (MAY 2015)

Functional dependency is a relationship that exists when one attribute uniquely determines another attribute.

It R is a relation with attributes X and Y, a functional dependency between the attributes is represented as X->Y, which specifies Y is functionally dependent on X.

Functional Dependecny (FD) is a set of constraints between two attributes in a relation.

3.State the anamolies of 1NF?(NOV 2015)

1NF databases have some problems:

Most notable:

- repetition of data
- to change a department name all tuples of the relation need to be updated since the department name can exist in multiple rows.

4.Explain entity relationship model? (APR 2016)

ER model defines the mapping between the entities in the database

ER model is a graphical representation of real world objects with their attributes and relationship. It makes the system easily understandable. This model is considered as a top down approach for designing a requirement.

5. What is meant by lossless-join decomposition? APRIL/MAY-2011

We claim the above decomposition is lossless. How can we decide whether decomposition is lossless?

1. Let *R* be a relation schema.
2. Let *F* be a set of functional dependencies on *R.*
3. Let R1and R2 form a decomposition of *R.*
4. The decomposition is a lossless-join decomposition of *R* if at least one of the following functional dependencies are in :

a. $R1 \cap R2 \rightarrow R1$

b. $R1 \cap R2 \rightarrow R2$

6. Define Boyce codd normal form? Why BCNF Stricter then 3NF? (Nov 2019)

A relation schema R is in BCNF with respect to a set F of functional dependencies if, for all functional dependencies in F.

BCNF is stricter than 3NF because each and every BCNF is relation to 3NF but every 3NF is not relation to BCNF. 4. BCNF non-transitionally depends on individual candidate key but there is

no such requirement in 3NF.Hence BCNF is stricter than 3NF.

7. What is meant by functional dependencies? What are the uses of functional dependencies?

Consider a relation schema R and α C R and β C R. The functional dependency α -> β holds on relational schema R if in any legal relation r(R), for all pairs of
tuples t1 and t2 in r such that t1 [α] = t2 [α], and also t1 [β] =t2 [β].

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.

8. Explain trivial dependency?

Functional dependency of the form α -> β is trivial if β C α Trivial functional dependencies are satisfied by all the relations.

9. What is meant by normalization of data and Denormalization?

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.

Denormalization: It is the process of attempting to optimize the performance of a database by adding redundant data or by grouping data.

10. Give the properties of decomposition. (MAY 2019)

Lossless-join decomposition
Dependency preservation
Repetition of information

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.00

12.Define Domain / key normal form?

It is a normal form used in database normalization which requires that the database contains no constraints other than domain constraints and key constraints.

13. What are the desirable properties of decomposition? (MAY 2017, Nov 2017)

Lossless join and dependency preserving are the two desirable properties of decomposition.

Lossless join decomposition property:

Let ***R*** be the relational schema with instance ***r*** is decomposed into $R_1, R_2, \ldots, R_n$ with instance $r_1, r_2, \ldots, r_n$.

If $r_1 \bowtie r_2 \bowtie \ldots\ldots \bowtie r_n = r$, then it is called Lossless Join Decomposition.

i.e. if natural joins of all the decompositions gives the original relation, then it is said to be Lossless Join Decomposition.

Dependency preserving Property:

The second property of decomposition is **Dependency Preserving Decomposition**.

If the original table is decomposed into multiple fragments, then somehow, we suppose to get all original FDs from these fragments. In other words, every dependency in original table must be preserved or say, every dependency must be satisfied by at least one decomposed table.

14. What is an entity relationship model?

The entity relationship model is a collection of basic objects called entities and relationship among those objects. An entity is a thing or object in the real world that is distinguishable from other objects.

15. What are attributes? Give examples.

An entity is represented by a set of attributes. Attributes are descriptive properties possessed by each member of an entity set.

Example: possible attributes of customer entity are customer name, customer id, customer street, customer city.

16. What is relationship? What is meant by the degree of relationship set?

A relationship is an association among several entities.

Example: A depositor relationship associates a customer with each account that he/she has.

The degree of relationship type is the number of participating entity types

17. Define the terms Entity set and Relationship set? (MAY 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.

18. Define single valued and multivalued attributes.

Single valued attributes: attributes with a single value for a particular entity are called single valued attributes.

Multivalued attributes: Attributes with a set of value for a particular entity are called multivalued attributes.

19. What are stored and derived attributes?

Stored attributes: The attributes stored in a data base are called stored attributes.

Derived attributes: The attributes that are derived from the stored attributes are called derived attributes.

20. 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.

21. Define the terms

i) Entity type

ii) Entity set

iii) Key attribute
iv) 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.

Entity type: An entity type defines a collection of entities that have the same attributes.
Entity set: The set of all entities of the same type is termed as an entity set.

22. 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.

23. 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

24. Explain the two types of participation constraint.

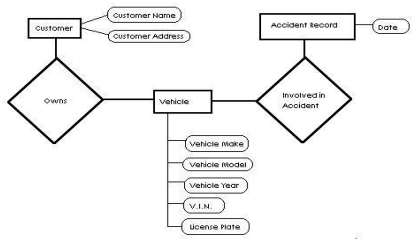
• **Total:** The participation of an entity set E in a relationship set R is said to be **total** if every entity in E participates in at least one relationship in R.
• **Partial:** if only some entities in E participate in relationships in R, the participation of entity set E in relationship R is said to be **partial.**

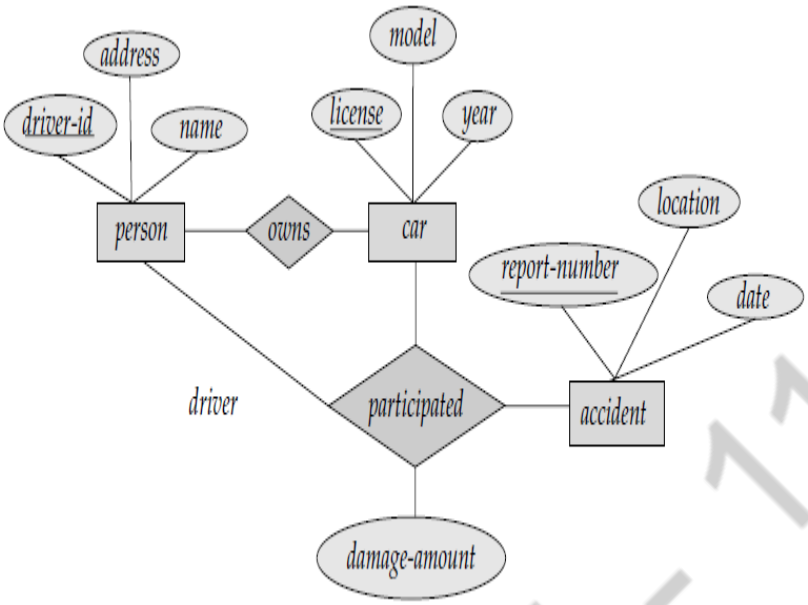
25. What is the significance of "participation role name" in the description of relationship types? (Nov 2019)

The Participation role is the part that every entity participates in a relationship. This role is important to use role name in the depiction of relationship type when the similar entity type participates more than once in a relationship type in various roles. The role names are necessary in recursive relationships.

PART - B

1. **Construct an E-R diagram for a car insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents. Each insurance policy covers one or more cars, and has one or more premium payments associated with it. Each payment is for a particular period of time and has an associated due date, and the date when the payment was received? (NOV 2016, NOV 2018)**





E-R diagram for a Car-insurance company.

appropriate tables for the above ER Diagram :
Car insurance tables:
person (driver-id, name, address)

car (license, year,model)

accident (report-number, date, location)

participated(driver-id, license, report-number, damage-amount)

2.Discuss the correspondence between the ER model construct and the relational model constructs. Show how each ER model construct can be mapped to the relational model. Discuss the option for mapping EER model construct? (MAY 2017), (NOV 2019)

Relational Data Model:

The model uses the concept of a mathematical relation-which looks somewhat like a table of values-as its basic building block, and has its theoretical basis in set theory and first order predicate logic.

The relational model represents the database a collection of relations. Each relation resembles a table of values or, to some extent, a "flat" file of records. When a relation is thought of as a table of values, each row in the table represents a collection of related data values. In

the relation model, each row in the table represents a fact that typically corresponds to a real-world entity or relationship. The table name and column names are used to help in interpreting the meaning of the values in each row. In the formal relational model terminology, a row is called a tuple, a column header is called an attribute, and the table is called a relation. The data type describing the types of values that can appear in each column is represented by domain of possible values.

ER Model:

An entity-relationship model (ERM) is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database, and its requirements in a top-down fashion. Diagrams created by this process are called entity-relationship diagrams, ER diagrams, or ERDs.

The first stage of information system design uses these models during the requirements analysis to describe information needs or the type of information that is to be stored in a database. In the case of the design of an information system that is based on a database, the conceptual data model is, at a later stage (usually called logical design), mapped to a logical data model, such as the relational model; this in turn is mapped to a physical model during physical design. We create a relational schema from an entity-relationship(ER) schema.

In the case of the design of an information system that is based on a database, the conceptual data model is, at a later stage (usually called logical design), mapped to a logical data model, such as the relational model; this in turn is mapped to a physical model during physical design. Sometimes, both of these phases are referred to as "physical design". Key elements of this model are entities, attributes, identifiers and relationships.

Correspondence between ER and Relational Models:

ER Model	**Relational Model**
Entity type	"Entity" relation
1:1 or 1:N relationship type	Foregin key
M:N relationship type	"Relationship" relation and two foreign keys
n ary relationship type	"Relationship" relation and n foreign keys
Simple attributes	Attributes
Composite attributes	Set of simple component attributes
Multivalued attributes	Relation and foreign key
Value set	Domain
Key attribute	Primary key or secondary key

Mapping of regular entity types:

For each regular entity type E in the ER schema, create a relation R that includes all the simple attributes of E. Include only the simple component attributes of a composite attribute. Choose one of the key attributes of E as primary key for R. If the chosen key of E is composite, the set of simple attributes that form it will together the primary key of R.

If multiple keys were identified for E during the conceptual design, the information describing the attributes that form each additional key is kept in order to specify secondary (unique) keys of relation R. Knowledge about keys is also kept for indexing purpose and other types of analyses.

We create the relations EMPLOYEE, DEPARTMENT, and PROJECT in to correspond to the regular entity types EMPLOYEE, DEPARTMENT, and PROJECT. The foreign key and relationship attributes, if any, are not include yet; they will be added during subsequent steps. These, include the attributes SUPERENO and DNO of EMPLOYEE, MGRNO and MGRSTARTDATE of DEPARTMENT, and DNUM of PROJECT. We choose ENO, DNUMBER, and PNUMBER as primary keys for the relations EMPLOYEE, DEPARTMENT, and PROJECT, respectively. Knowledge that DNAME of DEPARTMENT and PNAME of PROJCET are secondary keys is kept for possible use later in the design.

The relation that is created from the mapping of entity types are sometimes called entity relations because each tuple represents an entity instance.

3.Explain in detail about Functional Dependencies? (NOV 2018)
(OR)
Briefly discuss about the Functional Dependency Concepts? (MAY 2018, Nov 2019)

<u>Functional Dependencies Definition</u>

Functional dependency (FD) is a constraint between two sets of attributes from the database.

o A functional dependency is a property of the semantics or meaning of the attributes. In every relation R(A1, A2,…, An) there is a FD called the PK -> A1, A2, …, An Formally the FD is defined as follows

o If X and Y are two sets of attributes, that are subsets of T

For any two tuples t1 and t2 in r , if t1[X]=t2[X], we must also have t1[Y]=t2[Y].

Notation:

o If the values of Y are determined by the values of X, then it is denoted by X -> Y o
Given the value of one attribute, we can determine the value of another attribute

X f.d. Y or X -> y

Example: Consider the following,

Student Number -> Address, Faculty Number -> Department,
Department Code -> Head of Dept

Functional dependencies allow us to express constraints that cannot be expressed using super keys. Consider the schema:

Loan-info-schema = (customer-name, loan-number, branch-name, amount).

We expect this set of functional dependencies to hold:
loan-number amount

loan-number branch-name

but would not expect the following to hold:
loan-number customer-name

Use of Functional Dependencies

We use functional dependencies to:

o test relations to see if they are legal under a given set of functional dependencies.

o If a relation *r* is legal under a set *F* of functional dependencies, we say that *r* satisfies *F*. o specify constraints on the set of legal relations

o We say that *F* holds on *R* if all legal relations on *R* satisfy the set of functional dependencies *F*.

Note: A specific instance of a relation schema may satisfy a functional dependency even if the functional dependency does not hold on all legal instances.

o For example, a specific instance of *Loan-schema* may, by chance, satisfy *loan-number customer-name.*

Example

Employee

SSN	Name	JobType	DeptName
557-78-6587	Lance Smith	Accountant	Salary
214-45-2398	Lance Smith	Engineer	Product

Note: Name is functionally dependent on SSN because an employee‘s name can be uniquely determined from their SSN. Name does not determine SSN, because more than one employee can have the same name.

Keys

- Whereas a key is a set of attributes that uniquely identifies an entire tuple, a functional dependency allows us to express constraints that uniquely identify the values of certain

attributes.

- However, a candidate key is always a determinant, but a determinant doesn‘t need to be a key.

Axioms

- Before we can determine the closure of the relation, Student, we need a set of rules.

- Developed by Armstrong in 1974, there are six rules (axioms) that all possible functional dependencies may be derived from them.

1. ***Reflexivity Rule*** --- If *X* is a set of attributes and *Y* is a subset of *X*, then ***X*** □ ***Y*** holds. each subset of *X* is functionally dependent on *X*.

2. ***Augmentation Rule*** --- If ***X*** □ ***Y*** holds and *W* is a set of attributes, and then ***WX*** □ ***WY***

holds.

3. ***Transitivity Rule*** --- If ***X*** □ ***Y*** and ***Y*** □ ***Z*** holds, then ***X*** □ ***Z*** holds.

Derived Theorems from Axioms

4. ***Union Rule*** --- If ***X*** □ ***Y*** and ***X*** □ ***Z*** holds, then ***X*** □ ***YZ*** holds.

5. ***Decomposition Rule*** --- If ***X*** □ ***YZ*** holds, then so do ***X*** □ ***Y*** and ***X*** □ ***Z***.

6. ***Pseudo transitivity Rule*** --- If ***X*** □ ***Y*** and ***WY*** □ ***Z*** hold then so does ***WX*** □ ***Z***.

Back to the Example

SNo SName CNo CName Addr Instr. Office

Based on the rules provided, the following dependencies can be derived.

(SNo, CNo) □ ***SNo*** (Rule 1) -- subset

(SNo, CNo) □ ***CNo*** (Rule 1)

(SNo, CNo) □ ***(SName, CName)*** (Rule 2) -- augmentation

CNo □ ***office*** (Rule 3) -- transitivity

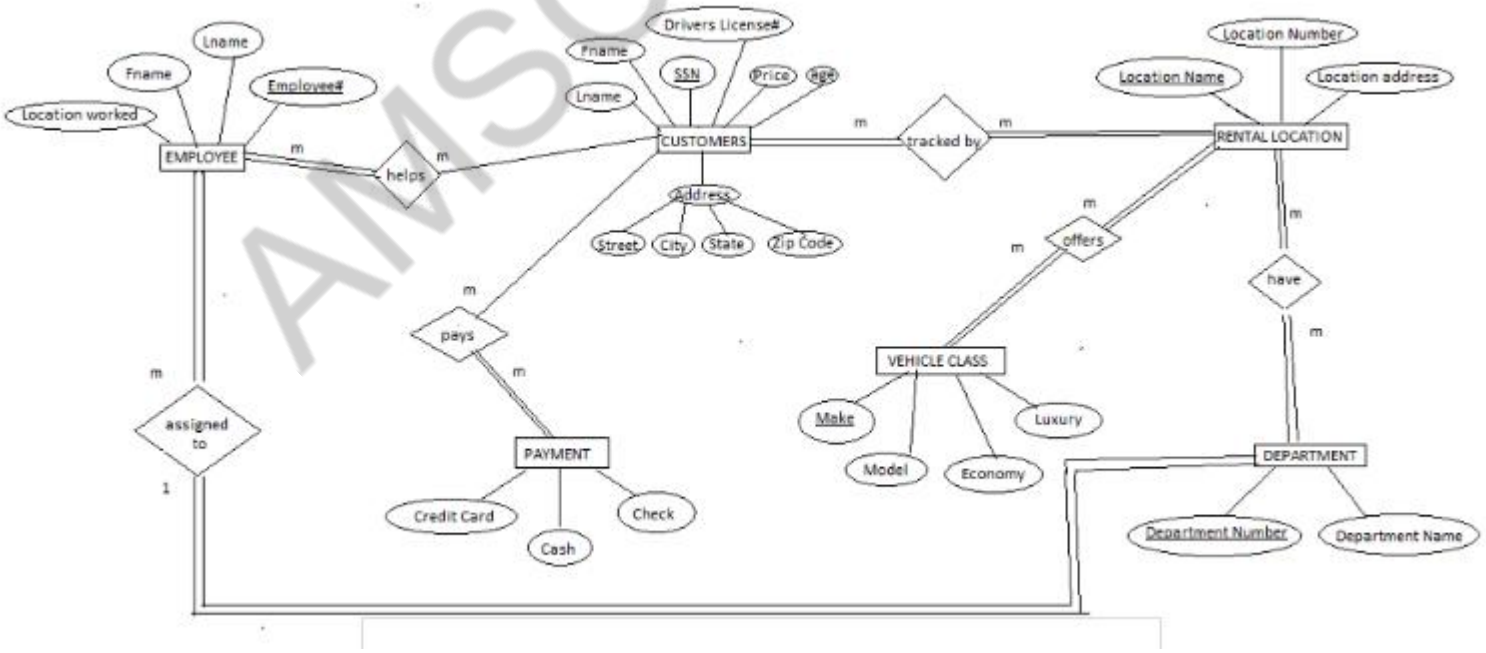
SNo □ ***(SName, address)*** (Union Rule) etc.

Properties of FDs

- X □ Y says redundant X-values always cause the redundancy of Y-values.
- FDs are given by DBAs or DB designers.
- FDs are enforced/guaranteed by DBMS.
- Given an instance r of a relation R, we can only determine that some FD is not satisfied by R, but can not determine if an FD is satisfied by R.

5,A car rental company maintains a database for all vehicles in its current fleet. For all vehicles, it includes the vehicle identification number, license number, manufacturer, model, date of purchase and color. Special data are included for certain types of vehicles.
Trucks: cargo capacity
Sports car: horsepower, renter age requirement
Vans: number of passengers
Off-road vehicle: ground clearance,drivetrain(four or two-wheeler drive)
Construct an ER model for the car rental company database.(NOV 2015)

ER MODEL FOR A CAR RENTAL COMPANY DATABASE



6.State the need for normalization of a Database and Explain the various Normal Forms (1st, 2nd,3rd, BCNF, 4th, 5th and Domain-key) with suitable examples? (JUN 2015, APR 2018)

(OR)

Exemplify multivalue dependency and fourth normal form (4NF) and join dependency and fifth

normal form(5NF) ? (NOV 2019)

(OR)

What is Normalization? Explain in detail about all Normal Forms (MAY 2019, NOV 2014)

Normalization of Database

Database Normalisation is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like Insertion, Update and Deletion Anamolies. It is a multi-step process that puts data into tabular form by removing duplicated data from the relation tables.

Normalization is used for mainly two purpose,

- Eliminating reduntant(useless) data.
- Ensuring data dependencies make sense i.e data is logically stored.

Problem Without Normalization

Without Normalization, it becomes difficult to handle and update the database, without facing data loss. Insertion, Updation and Deletion Anamolies are very frequent if Database is not Normalized. To understand these anomalies let us take an example of **Student** table.

S_id	S_Name	S_Address	Subject_opted
401	Adam	Noida	Bio
402	Alex	Panipat	Maths
403	Stuart	Jammu	Maths
404	Adam	Noida	Physics

- **Updation Anamoly :** To update address of a student who occurs twice or more than twice in a table, we will have to update **S_Address** column in all the rows, else data will become inconsistent.
- **Insertion Anamoly :** Suppose for a new admission, we have a Student id(S_id), name and address of a student but if student has not opted for any subjects yet then we have to insert **NULL** there, leading to Insertion Anamoly.
- **Deletion Anamoly :** If (S_id) 401 has only one subject and temporarily he drops it, when we delete that row, entire student record will be deleted along with it.

Normalization Rule

Normalization rule are divided into following normal form.

1. First Normal Form
2. Second Normal Form
3. Third Normal Form
4. BCNF

First Normal Form (1NF)

As per First Normal Form, no two Rows of data must contain repeating group of information i.e each set of column must have a unique value, such that multiple columns cannot be used to fetch the same row. Each table should be organized into rows, and each row should have a primary key that distinguishes it as unique.

The **Primary key** is usually a single column, but sometimes more than one column can be combined to create a single primary key. For example consider a table which is not in First normal form

Student Table :

Student	Age	Subject
Adam	15	Biology, Maths
Alex	14	Maths
Stuart	17	Maths

In First Normal Form, any row must not have a column in which more than one value is saved, like separated with commas. Rather than that, we must separate such data into multiple rows.

Student Table following 1NF will be :

Student	Age	Subject
Adam	15	Biology
Adam	15	Maths

Alex	14	Maths
Stuart	17	Maths

Using the First Normal Form, data redundancy increases, as there will be many columns with same data in multiple rows but each row as a whole will be unique.

Second Normal Form (2NF)

As per the Second Normal Form there must not be any partial dependency of any column on primary key. It means that for a table that has concatenated primary key, each column in the table that is not part of the primary key must depend upon the entire concatenated key for its existence. If any column depends only on one part of the concatenated key, then the table fails **Second normal form**.

In example of First Normal Form there are two rows for Adam, to include multiple subjects that he has opted for. While this is searchable, and follows First normal form, it is an inefficient use of space. Also in the above Table in First Normal Form, while the candidate key is {**Student**, **Subject**}, **Age** of Student only depends on Student column, which is incorrect as per Second Normal Form. To achieve second normal form, it would be helpful to split out the subjects into an independent table, and match them up using the student names as foreign keys.

New Student Table following 2NF will be :

Student	**Age**
Adam	15
Alex	14
Stuart	17

In Student Table the candidate key will be **Student** column, because all other column i.e **Age** is dependent on it.

New Subject Table introduced for 2NF will be :

Student	**Subject**
Adam	Biology
Adam	Maths

Alex	Maths
Stuart	Maths

In Subject Table the candidate key will be {**Student**, **Subject**} column. Now, both the above tables qualifies for Second Normal Form and will never suffer from Update Anomalies. Although there are a few complex cases in which table in Second Normal Form suffers Update Anomalies, and to handle those scenarios Third Normal Form is there.

Third Normal Form (3NF)

Third Normal form applies that every non-prime attribute of table must be dependent on primary key, or we can say that, there should not be the case that a non-prime attribute is determined by another non-prime attribute. So this *transitive functional dependency* should be removed from the table and also the table must be in **Second Normal form**. For example, consider a table with following fields.

Student_Detail Table :

Student_id	Student_name	DOB	Street	city	State	Zip

In this table Student_id is Primary key, but street, city and state depends upon Zip. The dependency between zip and other fields is called **transitive dependency**. Hence to apply **3NF**, we need to move the street, city and state to new table, with **Zip** as primary key.

New Student_Detail Table :

Student_id	Student_name	DOB	Zip

Address Table :

Zip	Street	City	state

The advantage of removing transtive dependency is,

- Amount of data duplication is reduced.
- Data integrity achieved.

Boyce and 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 anamoly 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:

- R must be in 3rd Normal Form
- and, for each functional dependency (X -> Y), X should be a super Key.

Consider the following relationship : **R (A,B,C,D)**

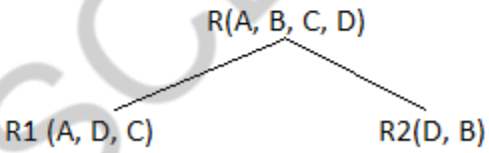
and following dependencies :

A -> BCD
BC -> AD
D -> B

Above relationship is already in 3rd NF. Keys are **A** and **BC**.

Hence, in the functional dependency, **A -> BCD**, A is the super key.
in second relation, **BC -> AD**, BC is also a key.
but in, **D -> B**, D is not a key.

Hence we can break our relationship R into two relationships **R1** and **R2**.



Breaking, table into two tables, one with A, D and C while the other with D and B.

MULTIVALUE DEPENDENCY AND FOURTH NORMAL FORM(4NF):

In the fourth normal form,

- It should meet all the requirement of 3NF
- Attribute of one or more rows in the table should not result in more than one rows of the same table leading to multi-valued dependencies

To understand it clearly, consider a table with Subject, Lecturer who teaches each subject and recommended Books for each subject.

COURSE
SUBJECT
LECTURER
BOOKS

SUBJECT	LECTURER	BOOKS
Mathematics	Alex	Maths Book1
Mathematics	Bosco	Maths Book2
Physics	Rose	Physics Book
Chemistry	Adam	Chemistry Book

If we observe the data in the table above it satisfies 3NF. But LECTURER and BOOKS are two independent entities here. There is no relationship between Lecturer and Books. In the above example, either Alex or Bosco can teach Mathematics. For Mathematics subject , student can refer either 'Maths Book1' or 'Maths Book2'. i.e.;

SUBJECT --> LECTURER

SUBJECT-->BOOKS

This is a multivalued dependency on SUBJECT. If we need to select both lecturer and books recommended for any of the subject, it will show up (lecturer, books) combination, which implies lecturer who recommends which book. This is not correct.

```
SELECT c.LECTURER, c.BOOKS FROM COURSE c WHERE SUBJECT = 'Mathematics';
```

To eliminate this dependency, we divide the table into two as below:

COURSE_LECTURER
SUBJECT
LECTURER

COURSE_BOOKS
SUBJECT
BOOKS

4NF

SUBJECT	LECTURER
Mathematics	Alex
Mathematics	Bosco
Physics	Rose
Chemistry	Adam

SUBJECT	BOOKS
Mathematics	Maths Book1
Mathematics	Maths Book2
Physics	Physics Book
Chemistry	Chemistry Book

Now if we want to know the lecturer names and books recommended for any of the subject, we will fire two independent queries. Hence it removes the multi-valued dependency and confusion around the data. Thus the table is in 4NF.

JOIN DEPENDENCY AND FIFTH NORMAL FORM (5NF):

A database is said to be in 5NF, if and only if,

- It's in 4NF
- If we can decompose table further to eliminate redundancy and anomaly, and when we re-join the decomposed tables by means of candidate keys, we should not be losing the original data or any new record set should not arise. In simple words, joining two or more decomposed table should not lose records nor create new records.

Consider an example of different Subjects taught by different lecturers and the lecturers taking classes for different semesters.

Note: Please consider that Semester 1 has Mathematics, Physics and Chemistry and Semester 2 has only Mathematics in its academic year!!

COURSE
SUBJECT
LECTURER
CLASS

SUBJECT	LECTURER	CLASS
Mathematics	Alex	SEMESTER 1
Mathematics	Rose	SEMESTER 1
Physics	Rose	SEMESTER 1
Physics	Joseph	SEMESTER 2
Chemistry	Adam	SEMESTER 1

In above table, Rose takes both Mathematics and Physics class for Semester 1, but she does not take Physics class for Semester 2. In this case, combination of all these 3 fields is required to identify a valid data. Imagine we want to add a new class - Semester3 but do not know which Subject and who will be taking that subject. We would be simply inserting a new entry with Class as Semester3 and leaving Lecturer and subject as NULL. As we discussed above, it's not a good to have such entries. Moreover, all the three columns together act as a primary key, we cannot leave other two columns blank!

Hence we have to decompose the table in such a way that it satisfies all the rules till 4NF and when join them by using keys, it should yield correct record. Here, we can represent each lecturer's Subject area and their classes in a better way. We can divide above table into three - (SUBJECT, LECTURER), (LECTURER, CLASS), (SUBJECT, CLASS)

SUBJECT	LECTURER
Mathematics	Alex
Mathematics	Rose
Physics	Rose
Physics	Joseph
Chemistry	Adam

5NF

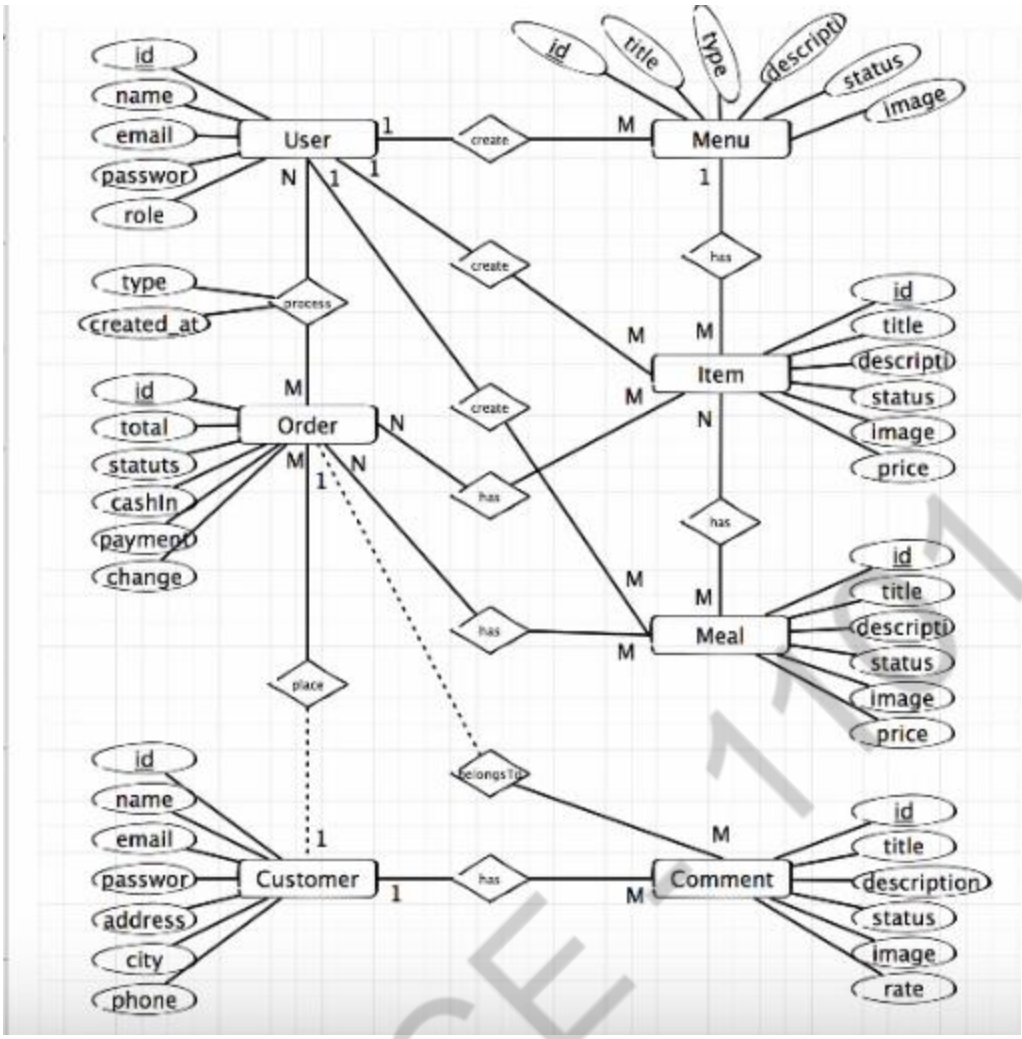
CLASS	LECTURER
SEMESTER 1	Alex
SEMESTER 1	Rose
SEMESTER 1	Rose
SEMESTER 2	Joseph
SEMESTER 1	Adam

CLASS	SUBJECT
SEMESTER 1	Mathematics
SEMESTER 1	Physics
SEMESTER 1	Chemistry
SEMESTER 2	Physics

Now, each of combinations is in three different tables. If we need to identify who is teaching which subject to which semester, we need join the keys of each table and get the result.

For example, who teaches Physics to Semester 1, we would be selecting Physics and Semester1 from table 3 above, join with table1 using Subject to filter out the lecturer names. Then join with table2 using Lecturer to get correct lecturer name. That is we joined key columns of each table to get the correct data. Hence there is no lose or new data - satisfying 5NF condition.

7. Draw E-R diagram for the "Restaurant menu ordering system" that will facilitate the food items ordering and services with in a restaurant. The entire restaurant scenario is detailed as follows. The customer is able to view the food items menu, call the waiter, place orders and obtain the final bill through the computer kept in their table. The waiters through their wireless tablet PC are able to initializw a table for customer, control the table functions to assist customers, orders, send orders to food preparation staff(chef) and finalize the customers bill. The food preparation staffs(chefs), with their touch-display interfaces to the system, are able to view orders sent to the kitchen by waiters. During preparation, they are able to let the waiter know the status of each item and can send notifications when items are completed. The system should have full accountability and logging facilities and should support supervisor actions to account for exceptional circumstances such as meal being refunded or walked out on? (May 2015).



8.Explain first normal form, second normal form, third normal form and BCNF with an example? (NOV 2016, Nov 2019)

1st Normal Form:

The Requirements:

- The requirements to satisfy the 1st NF:

- Each table has a primary key: minimal set of attributes which can uniquely identify a record

- The values in each column of a table are atomic (No multi-value attributes allowed). There are no repeating groups: two columns do not store similar information in the same table

1st Normal Form

Example 1:

Un-normalized Students table:

Student#	**AdvID**	**AdvName**	**AdvRoom**	**Class1**	**Class2**
123	**123A**	**James**	**555**	**102-8**	**104-9**
124	**123B**	**Smith**	**467**	**209-0**	**102-8**

Normalized Students table:

Student#	**AdvID**	**AdvName**	**AdvRoom**	**Class#**
123	123A	James	555	102-8
123	123A	James	555	104-9
124	123B	Smith	467	209-0
124	123B	Smith	467	102-8

Example 2:

Title	**Author1**	**Author2**	**ISBN**	**Subject**	**Pages**	**Publisher**

Database System Concepts	Abraham Silberschatz	Henry F. Korth	0072958863	MySQL, Computers	1168	McGraw-Hill
Operating System Concepts	Abraham Silberschatz	Henry F. Korth	0471694665	Computers	944	McGraw-Hill

Table 1 problems

- This table is not very efficient with storage.
- This design does not protect data integrity.
- Third, this table does not scale well.
- In our Table 1, we have two violations of First Normal Form:
- First, we have more than one author field,
- Second, our subject field contains more than one piece of information. With more than one value in a single field, it would be very difficult to search for all books on a given subject.

Table 2

Title	Author	ISBN	Subject	Pages	Publisher
Database System Concepts	Abraham Silberschatz	0072958863	MySQL	1168	McGraw-Hill
Database System Concepts	Henry F. Korth	0072958863	Computers	1168	McGraw-Hill
Operating System Concepts	Henry F. Korth	0471694665	Computers	944	McGraw-Hill
Operating System Concepts	Abraham Silberschatz	0471694665	Computers	944	McGraw-Hill

- We now have two rows for a single book. Additionally, we would be violating the

- A better solution to our problem would be to separate the data into separate tables- an Author table and a Subject table to store our information, removing that information from the

Book table:

- Each table has a primary key, used for joining tables together when querying the data. A primary key value must be unique with in the table (no two books can have the same ISBN

number), and a primary key is also an index, which speeds up data retrieval based on the primary key.
- Now to define relationships between the tables

Second Normal Form (2NF)

- a form is in 2NF if and only if it is in 1NF and has no attributes which require only part of the key to uniquely identify them

- To do - remove part-key dependencies:

- where a key has more than one attribute, check that each non-key attribute depends on the whole key and not part of the key.

- for each subset of the key which determines an attribute or group of attributes create a new form. Move the *dependant* attributes to the new form..

- Add the part key to new form, making it the primary key.

- Mark the part key as a foreign key in the original form.

- **Result: 2NF forms**

If each attribute A in a relation schema R meets one of the following criteria:

- It must be in first normal form.
- It is not partially dependent on a candidate key.
- Every non-key attribute is fully dependent on each candidate key of the relation.

Second Normal Form (or 2NF) deals with redundancy of data in vertical columns.

Example of Second Normal Form:

Here is a list of attributes in a table that is in First Normal Form:

Department

Project_Name

Employee_Name

Emp_Hire_Date

Project_Manager

Project_Name and Employee_Name are the candidate key for this table. Emp_Hire_Date and Project_Manager are partially depend on the Employee_Name, but not depend on the Project_Name. Therefore, this table will not satisfy the Second Normal Form

In order to satisfy the Second Normal Form, we need to put the Emp_Hire_Date and Project_Manager to other tables. We can put the Emp_Hire_Date to the Employee table and put the Project_Manager to the Project table.

So now we have three tables:

Department	**Project**
Project_Name	Project_ID
Employee_Name	Project_Name
	Project_Manager

Employee

Employee_ID

Employee_Name

Employee_Hire_Date

Now, the Department table will only have the candidate key left.

Third Normal Form

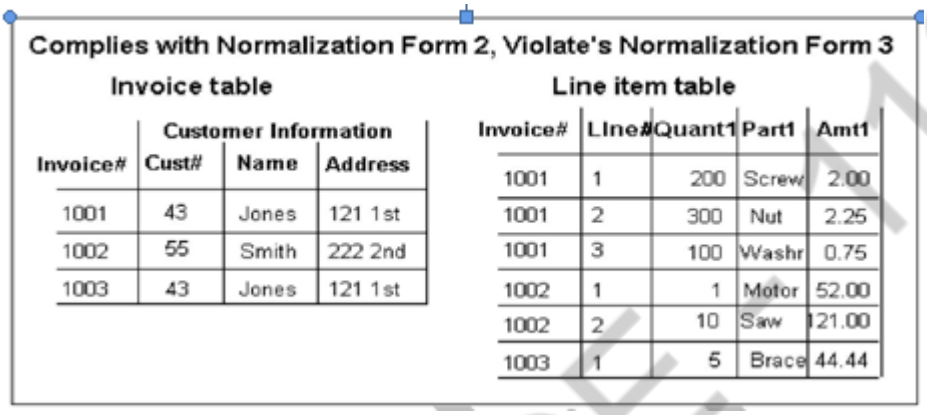
A relation R is in Third Normal Form (3NF) if and only if it is:

- in Second Normal Form.
- Every non-key attribute is non-transitively dependent on the primary key.

An attribute C is transitively dependent on attribute A if there exists an attribute B such that A B and B C, then A C.

Example of Third Normal Form:

Here is the Second Normal Form of the table for the invoice table:



Complies with Normalization Form 2, Violate's Normalization Form 3

Invoice table

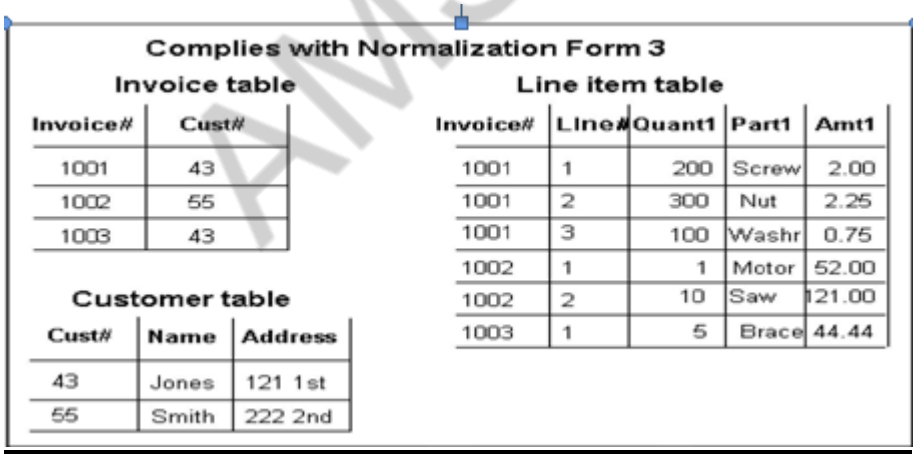
	Customer Information		
Invoice#	Cust#	Name	Address
1001	43	Jones	121 1st
1002	55	Smith	222 2nd
1003	43	Jones	121 1st

Line item table

Invoice#	LIne#	Quant1	Part1	Amt1
1001	1	200	Screw	2.00
1001	2	300	Nut	2.25
1001	3	100	Washr	0.75
1002	1	1	Motor	52.00
1002	2	10	Saw	121.00
1003	1	5	Brace	44.44

It violates the Third Normal Form because there will be redundancy for having multiple invoice number for the same customer. In this example, Jones had both invoice 1001 and 1003.

To solve the problem, we will have to have another table for the Customers



Complies with Normalization Form 3

Invoice table

Invoice#	Cust#
1001	43
1002	55
1003	43

Customer table

Cust#	Name	Address
43	Jones	121 1st
55	Smith	222 2nd

Line item table

Invoice#	LIne#	Quant1	Part1	Amt1
1001	1	200	Screw	2.00
1001	2	300	Nut	2.25
1001	3	100	Washr	0.75
1002	1	1	Motor	52.00
1002	2	10	Saw	121.00
1003	1	5	Brace	44.44

By having Customer table, there will be no transitive relationship between the invoice number and the customer name and address. Also, there will not be redundancy on the customer information

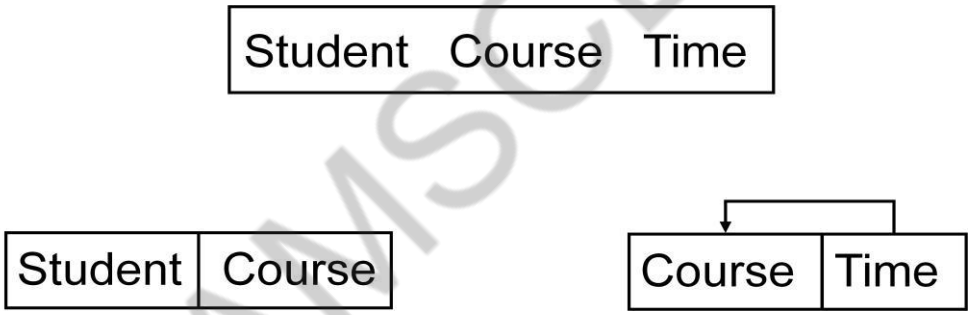
Boyce-Codd Normal Form:

- A relation is in Boyce-Codd normal form (BCNF) if for every FD A □ B either
- B is contained in A (the FD is trivial), or
- A contains a candidate key of the relation,
- In other words: every determinant in a non-trivial dependency is a (super) key.
- The same as 3NF except in 3NF we only worry about non-key Bs
- If there is only one candidate key then 3NF and BCNF are the same

Stream and BCNF

- Stream is not in BCNF as the FD {Time} □ {Course} is non-trivial and {Time} does not contain a candidate key

Conversion to BCNF



Stream has been put into BCNF but we have lost the FD {Student, Course} □ {Time}

Decomposition Properties

- Lossless: Data should not be lost or created when splitting relations up

- Dependency preservation: It is desirable that FDs are preserved when splitting relations up
- Normalisation to 3NF is always lossless and dependency preserving
- Normalisation to BCNF is lossless, but may not preserve all dependencies

Converting to BCNF

1) The determinant, Offering#, becomes part of the key and the dependant attribute T_Code, becomes a non key attribute. So the Dependency diagram is now

   S_Num, Offering# □ T_Code, Review Date

2) There are problems with this structure as T_Code is now dependant on only part of the key. This violates the rules for 2NF, so the table needs to be divided with the partial dependency becoming a new table. The dependencies would then be

   S_Num, Offering# □ T_Code, Review Date

   Offering# □ T_Code.

3) The original table is divided into two new tables. Each is in 3NF and in BCNF.

Student Review:

S_num	Offering#	Review Date
123599	01764	2nd march
123599	01765	12th april
123599	01789	2nd may
246700	01764	3rd march
346700	01765	7th may

OfferingTeacher:

Offering#	T_code#

01764	FIT104
01765	PIT305
01789	PIT107

Note that even relations in BCNF can have anomalies.

Anomalies:

INSERT: We cannot record the city for a supplier_no without also knowing the supplier_name

DELETE: If we delete the row for a given supplier_name, we lose the information that the supplier_no is associated with a given city.

UPDATE: Since supplier_name is a candidate key (unique), there are none.

9. Design and draw an E-R diagram for university database? (Nov 2019)

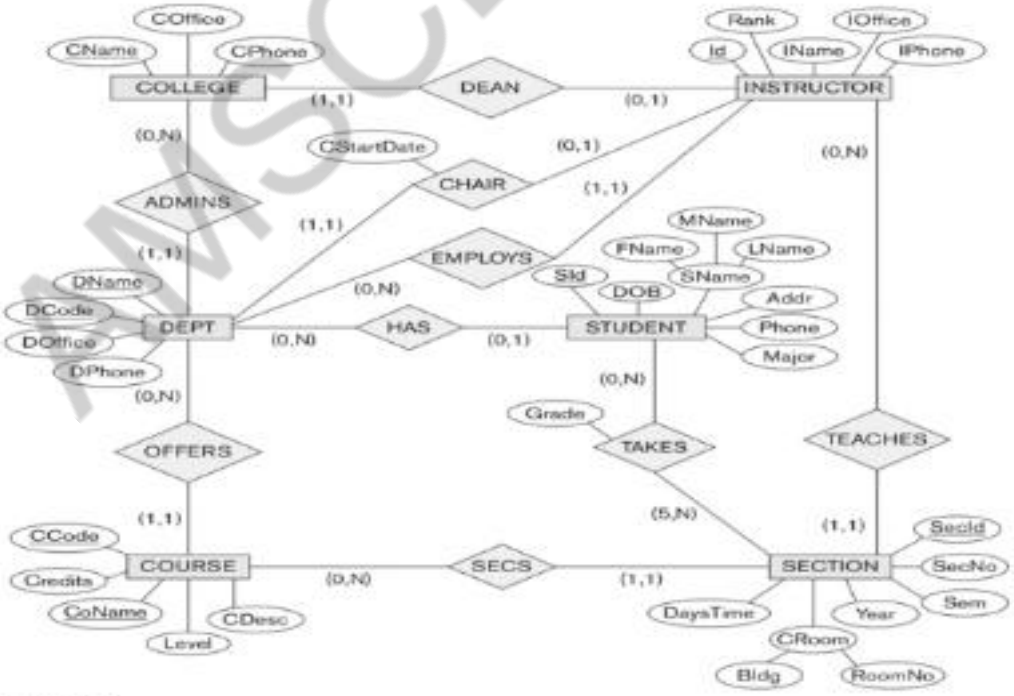


Figure 3.20
An ER diagram for a UNIVERSITY database schema.

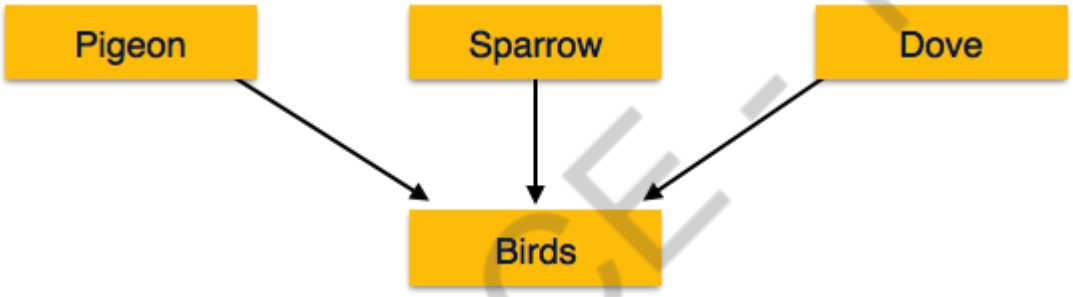
10. Explain with suitable example, the constraints of specialization and generalization in ER data modeling.(Nov 2019).

The ER Model has the power of expressing database entities in a conceptual hierarchical manner. As the hierarchy goes up, it generalizes the view of entities, and as we go deep in the hierarchy, it gives us the detail of every entity included.

Going up in this structure is called **generalization**, where entities are clubbed together to represent a more generalized view. For example, a particular student named Mira can be generalized along with all the students. The entity shall be a student, and further, the student is a person. The reverse is called **specialization** where a person is a student, and that student is Mira.

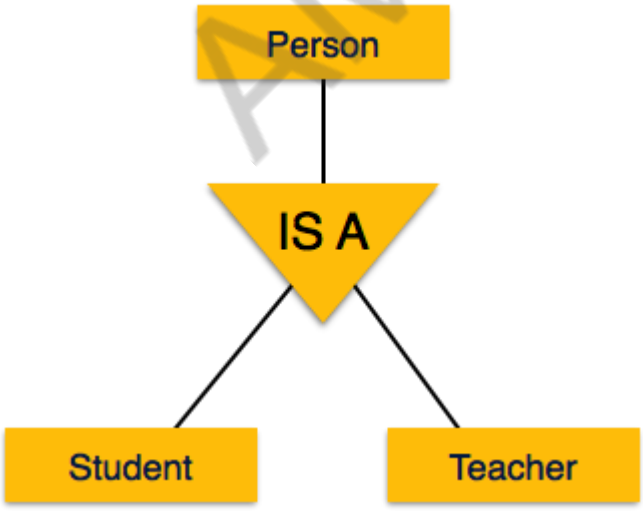
Generalization:

As mentioned above, the process of generalizing entities, where the generalized entities contain the properties of all the generalized entities, is called generalization. In generalization, a number of entities are brought together into one generalized entity based on their similar characteristics. For example, pigeon, house sparrow, crow and dove can all be generalized as Birds.



Specialization:

Specialization is the opposite of generalization. In specialization, a group of entities is divided into sub-groups based on their characteristics. Take a group 'Person' for example. A person has name, date of birth, gender, etc. These properties are common in all persons, human beings. But in a company, persons can be identified as employee, employer, customer, or vendor, based on what role they play in the company.

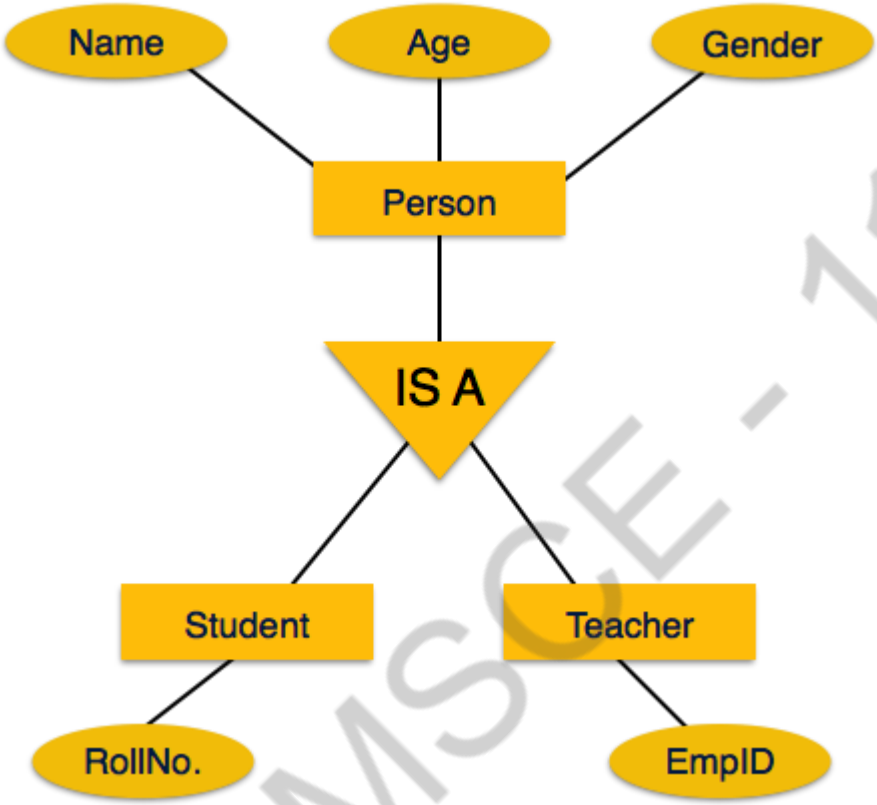


Similarly, in a school database, persons can be specialized as teacher, student, or a staff, based on what role they play in school as entities.

Inheritance :

We use all the above features of ER-Model in order to create classes of objects in object-oriented programming. The details of entities are generally hidden from the user; this process known as **abstraction**.

Inheritance is an important feature of Generalization and Specialization. It allows lower-level entities to inherit the attributes of higher-level entities.



For example, the attributes of a Person class such as name, age, and gender can be inherited by lower-level entities such as Student or Teacher.

UNIT- 3

TRANSACTIONS

PART-A

1. What is transaction?

Collections of operations that form a single logical unit of work are called transactions.

2. What are the two statements regarding transaction?

The two statements regarding transaction of the form:

Begin transaction
End transaction

3. What are the properties of transaction? (Nov'2014, May'2015 & May 2016)
(OR)
What are ACID Properties?(Dec 2017)

The properties of transactions are:

Atomicity: Either all operations of the transaction are properly reflected in the database or none are.

Consistency: Execution of a transaction in isolation preserves the consistency of the database.

Isolation: Although multiple transactions may execute concurrently, each transaction must be unaware of other concurrently executing transactions. Intermediate transaction results must be hidden from other concurrently executed transactions. That is, for every pair of transactions Ti and Tj, it appears to Ti that either Tj finished execution before Ti started, or Tj started execution after Ti finished.

Durability: After a transaction completes successfully, the changes it has made to the database persist,
even if there are system failures.

4. Highlight the role of recovery management component?(Dec 2018)

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

5. 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.

6. What are the states of transaction? (May 2019)

The states of transaction are

Active - Reading and Writing data items, if something wrong happens during reading and writing aborts to Failed.

Partially Committed - All reading and writing operations are done aborts to Failed when rollback occurs or committed when commit occurs.

Committed - Transaction successfully completed and all write operations made permanent in the database.

Failed - Transaction halted and all operations rolled back
Terminated - terminates either commits or failed

7. 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.

8. Give the reasons for allowing concurrency?

The reasons for allowing concurrency is 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.

9. 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.

10. What are the two types of Serializability?

The two types of Serializability is

- Conflict serializability
- View serializability

11. 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.

12. What are the different modes of lock?

The modes of lock are:

Exclusive
Shared

13. Define deadlock?

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

14. Define the phases of two phase locking protocol.

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.

15. 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.

16. 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.

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

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

18. 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.

19. What are the two types of errors?

The two types of errors are:

Logical error

System error

20. What are the storage types?

The storage types are:

Volatile storage

Nonvolatile storage

21. Define blocks?

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

22. 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.

23. What is meant by buffer blocks?

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

24. What is meant by disk buffer?

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

25. What is meant by log-based recovery scheme? (May 2019)

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.

26. 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.

27. 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.

28. Define page.

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

29. Explain 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.

30. Give the drawbacks of shadow-paging technique? (Dec 2018)

The drawbacks of shadow-paging technique are

- Commit Overhead
- Data fragmentation
- Garbage collection

31. 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.

32. Differentiate strict two phase locking protocol and rigorous two phase locking protocol. (May/June 2016)

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.

33. 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.

34. 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.

35. What is Serializability? How is it tested? (May 2014, Nov'2014, Nov'2016, &May 2018)

A (possibly concurrent) schedule is serializable if it is equivalent to a serial schedule. Precedence graph is used to test the serializability.

36. What is meant by Concurrency control? (Dec'2015)

A transaction is a particular execution of the program. When multiple transactions are trying to access the same shareable resource, many problems arise if the access control is not done properly. Mechanisms to which access control can be maintained is called Concurrency control.

37. Give an example for Two phase commit protocol. (Dec'2015)

Client want all or nothing transactions and Transfer either happens or nothing at all.

38. What is meant by garbage collection.(May/June 2016)

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.

39. State the need for concurrency? (Dec 2017)

Following are the purposes of concurrency control:

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

40. What is serializable schedule? (May 2017)

The schedule in which the transactions execute one after the other is called Serial Schedule. It is consistent in nature. For Example: Consider two transactions T1 and T2. All operations in T1 is executed after that all transactions in T2 are executed.

41. What type of lock is needed for insert and delete operations? (May 2017)

The exclusive lock is needed to insert and delete operations.

42. What is the difference between shared lock and exclusive lock? (May 2018)

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

43. What is rigorous two phase locking protocol? (Dec 2013)

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

44. List the four conditions for deadlock.(Dec 2016)

There are four mandatory conditions that must exist simultaneously for a deadlock to occur:

Mutual Exclusion: Some data items must be locked by some transactions in Exclusive Mode. These data items are not accessible to other transactions than the one currently holding it.

Hold and Wait: Some Transactions must be holding Exclusive Locks on some Data Items and at the same time waiting for grant of Exclusive Lock on some other data items, which may be currently locked by other transactions.

No Preemption: The data items locked exclusively by a Transaction cannot be forcibly pre-empted. The Transaction will release the locks on such items only voluntarily, when it has finished with the data items.

Cyclic Wait: There must exist a situation, where in a set of n transactions say $\{T_0, T_1, T_{n-1}\}$ are waiting in a cyclic manner for the data items locked by each other i.e.,

T_0 is waiting for some data item currently locked by T_1

T_1 is waiting for some data item currently locked by T_2

T_2 is waiting for some data item currently locked by T_3

.

.

.

T_{n-2} is waiting for some data item currently locked by T_{n-1}

T_{n-1} is waiting for some data item currently locked by T_0

45.List the responsibilities of a DBMS has whenever a transaction is submitted to the system for execution? (Nov 2019)

Begin the transaction. Execute a set of data manipulations and/or queries. If no errors occur then commit the transaction and end it. If errors occur then roll back the transaction and end it.

46.Brief any two violations that may occur if a transaction executes a lower isolation level than serializable? (Nov 2019)

- Lost updates.
- Dirty read (or uncommitted data).
- Unrepeatable read (or inconsistent retrievals).

PART-B

1. Briefly explain about Two phase commit and three phase commit protocols. (Nov' 2014, May 2015 & May 2016)
(OR)
Explain two phase commit protocol with an example?

Two phase commit protocol

• Assumes fail-stop model – failed sites simply stop working, and do not cause any other harm,
such as sending incorrect messages to other sites.
• Execution of the protocol is initiated by the coordinator after the last step of the transaction has been reached.
• The protocol involves all the local sites at which the transaction executed
• Let *T* be a transaction initiated at site *Si*, and let the transaction coordinator at *Si* be *Ci*.

Phase 1: Obtaining a Decision (prepare)

• Coordinator asks all participants to *prepare* to commit transaction *Ti*.
– Ci adds the records <prepare *T*> to the log and forces log to stable storage
– sends prepare *T* messages to all sites at which *T* executed
• Upon receiving message, transaction manager at site determines if it can commit the transaction
– if not, add a record <no *T*> to the log and send abort *T* message to *Ci*
– if the transaction can be committed, then:
– add the record <ready *T*> to the log
– force *all records* for *T* to stable storage
– send ready *T* message to C*i*

Phase 2: Recording the Decision (commit)

• *T* can be committed of *Ci* received a ready *T* message from all the participating sites: otherwise *T* must be aborted.
• Coordinator adds a decision record, <commit *T*> or <abort *T*>, to the log and forces record onto stable storage. Once the record stable storage it is irrevocable (even if failures occur)
• Coordinator sends a message to each participant informing it of the decision (commit or abort)
• Participants take appropriate action locally.

Possible Failures

Site Failure
Coordinator Failure
Network Partition

Three phase commit protocols.

Assumptions:
– No network partitioning
– At any point, at least one site must be up.

– **At** most *K* sites (participants as well as coordinator) can fail

Phase 1: Obtaining Preliminary Decision: Identical to 2PC Phase 1.

– **Every** site is ready to commit if instructed to do so

– Under 2PC each site is obligated to wait for decision from coordinator

– Under 3PC, knowledge of pre-commit decision can be used to commit despite coordinator failure

Phase 2: Recording the Preliminary Decision

Coordinator adds a decision record (<**abort** *T*> or <**precommit** *T*>) in its log and forces record to stable storage. Coordinator sends a message to each participant informing it of the decision.

Participant records decision in its log

– If abort decision reached then participant aborts locally

– If pre-commit decision reached then participant replies with <**acknowledge** T>

Phase 3: Recording Decision in the Database

Executed only if decision in phase 2 was to pre commit

Coordinator collects acknowledgments. It sends <**commit** *T*> message to the participants as soon as it receives *K* acknowledgments.

Coordinator adds the record <**commit** *T*> in its log and forces record to stable storage.

Coordinator sends a message to each participant to <**commit** *T*>.

Participants take appropriate action locally

Under 3PC, knowledge of pre-commit decision can be used to commit despite coordinator failure

- Avoids blocking problem as long as < K sites fail

Drawbacks:

- higher overheads
- Assumptions may not be satisfied in practice.

2. What is serializability? Explain its types? (Nov/Dec 2015)
(OR)
Explain conflict serializability and view serializability? (May 2018)
(OR)
Discuss in detail about the testing of serializability? (May 2019)

A (possibly concurrent) schedule is serializable, if it is equivalent to a serial schedule. Different forms of schedule equivalence give rise to the notions of:

1. C**onflict serializability**
2. V**iew serializability**

1. Conflict serializability

Instructions l_i and l_j of transactions T_i and T_j respectively, **conflict** if and only if there exists some item Q accessed by both l_i and l_j, and at least one of these instructions wrote Q.

1. l_i = **read**(Q), l_j = **read**(Q). l_i and l_j don't conflict.
2. l_i = **read**(Q), l_j = **write**(Q). They conflict.

3. l_i = **write**(Q), l_j = **read**(Q). They conflict
4. l_i = **write**(Q), l_j = **write**(Q). They conflict

- If a schedule S can be transformed into a schedule S' by a series of swaps of non-conflicting instructions, we say that S and S' are **conflict equivalent**.
- We say that a schedule S is **conflict serializable** if it is conflict equivalent to a serial schedule

2. View serializability

Let S and S' be two schedules with the same set of transactions. S and S' are **view equivalent** if the following three conditions are met, for each data item Q,

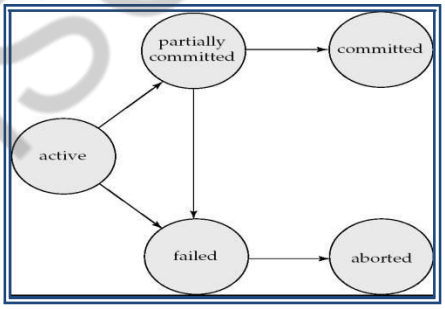
- If in schedule S, transaction T_i reads the initial value of Q, then in schedule S' also transaction T_i must read the initial value of Q.
- If in schedule S transaction T_i executes **read**(Q), and that value was produced by transaction T_j (if any), then in schedule S' also transaction T_i must read the value of Q that was produced by the same **write**(Q) operation of transaction T_j.
- The transaction (if any) that performs the final **write**(Q) operation in schedule S must also perform the final **write**(Q) operation in schedule S'.

3. Brief the states of a transaction with a neat diagram.(Nov 2019)

(OR)

List all possible sequences of states through which a transaction may pass? Explain why each state transition may occur? (May 2018)

Explain with an example the properties that must be satisfied by a transaction? (May 2018, May 2019, Nov 2019)



The states of transaction are

- **Active** - Reading and Writing data items, if something wrong happens during reading and writing aborts to Failed.
- **Partially Committed** - All reading and writing operations are done aborts to Failed when rollback occurs or committed when commit occurs.
- **Committed** - Transaction successfully completed and all write operations made permanent in the database.

- **Failed -** Transaction halted and all operations rolled back
- **Terminated -** terminates either commits or failed

The properties of transactions are:

- **Atomicity:** Either all operations of the transaction are properly reflected in the database or none are.

- **Consistency:** Execution of a transaction in isolation preserves the consistency of the database.

- **Isolation:** Although multiple transactions may execute concurrently, each transaction must be unaware of other concurrently executing transactions. Intermediate transaction results must be hidden from other concurrently executed transactions. That is, for every pair of transactions Ti and Tj, it appears to Ti that either Tj finished execution before Ti started, or Tj started execution after Ti finished.

- **Durability:** After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.

4. Explain the concepts of concurrent execution in Transaction processing system. (Nov/Dec 2014)

The transaction-processing system allows **concurrent execution** of multiple transactions to improve the system performance. In concurrent execution, the database management system controls the execution of two or more transactions in parallel; however, allows only one operation of any transaction to occur at any given time within the system. This is also known as **interleaved execution** of multiple transactions. The database system allows concurrent execution of transactions due to two reasons.

First, a transaction performing read or write operation using I/O devices may not be using the CPU at a particular point of time. Thus, while one transaction is performing I/O operations, the CPU can process another transaction. This is possible because CPU and I/O system in the computer system are capable of operating in parallel. This overlapping of I/O and CPU activities reduces the amount of time for which the disks and processors are idle and, thus, increases the **throughput** of the system (the number of transactions executed in a given amount of time).

5. Why concurrency control is needed? Explain with an example.
(OR)
What is Concurrency control? (May 2018)

Concurrency control is the process of managing simultaneous execution of transactions (such as queries, updates, inserts, deletes and so on) in a multiprocessing database system without having them interfere with one another.

This property of DBMS allows many transactions to access the same database at the same time without interfering with each other.

The primary goal of concurrency is to ensure the atomicity of the execution of transactions

in a multi-user database environment. Concurrency controls mechanisms attempt to interleave (parallel) READ and WRITE operations of multiple transactions so that the interleaved execution yields results that are identical to the results of a serial schedule execution.

Problems of Concurrency Control :

When concurrent transactions are executed in an uncontrolled manner, several problems can occur.

The concurrency control has the following three main problems:

- Lost updates.
- Dirty read (or uncommitted data).
- Unrepeatable read (or inconsistent retrievals).

Lost Update Problem:

A lost update problem occurs when two transactions that access the same database items have their operations in a way that makes the value of some database item incorrect.

In other words, if transactions T1 and T2 both read a record and then update it, the effects of the first update will be overwritten by the second update.

Example:

Consider the situation given in figure that shows operations performed by two transactions, Transaction- A and Transaction- B with respect to time.

Transaction- A	Time	Transaction- B
-----	t0	----
Read X	t1	----
----	t2	Read X
Update X	t3	----
----	t4	Update X
----	t5	----

At time t1, Transactions-A reads value of X.
At time t2, Transactions-B reads value of X.
At time t3, Transactions-A writes value of X on the basis of the value seen at time t1.
At time t4, Transactions-B writes value of X on the basis of the value seen at time t2.

So, update of Transactions-A is lost at time t4, because Transactions-B overwrites it without looking at its current value. Such type of problem is referred as the Update Lost Problem, as update made by one transaction is lost here.

Dirty Read Problem:

A dirty read problem occurs when one transaction updates a database item and then the transaction fails for some reason. The updated database item is accessed by another transaction before it is changed back to the original value. In other words, a transaction T1 updates a record, which is read by the transaction T2.

Then T1 aborts and T2 now has values which have never formed part of the stable database.

Example:

Consider the situation given in figure:

Transaction- A	Time	Transaction- B
----	t0	----
----	t1	Update X
Read X	t2	----
----	t3	Rollback
----	t4	----

At time t1, Transactions-B writes value of X.
At time t2, Transactions-A reads value of X.
At time t3, Transactions-B rollbacks. So, it changes the value of X back to that of prior to t1.
So, Transaction-A now has value which has never become part of the stable database. Such type of problem is referred as the Dirty Read Problem, as one transaction reads a dirty value which has not been committed
.

Inconsistent Retrievals Problem:

Unrepeatable read (or inconsistent retrievals) occurs when a transaction calculates some summary (aggregate) function over a set of data while other transactions are updating the data.

The problem is that the transaction might read some data before they are changed and other data after they are changed, thereby yielding inconsistent results.

In an unrepeatable read, the transaction T1 reads a record and then does some other processing

during which the transaction T2 updates the record. Now, if T1 rereads the record, the new value will be inconsistent with the previous value.

Example:

Consider the situation given in figure that shows two transactions operating on three accounts:

Account-1	**Account-2**	**Account-3**
Balance = 200	**Balance = 250**	**Balance = 150**

Transaction- A	Time	Transaction- B
-----	t0	----
Read Balance of Acc-1 sum <-- 200 Read Balance of Acc-2	t1	----
Sum <-- Sum + 250 = 450	t2	----
----	t3	Read Balance of Acc-3
----	t4	Update Balance of Acc-3 150 --> 150 - 50 --> 100
----	t5	Read Balance of Acc-1
----	t6	Update Balance of Acc-1 200 --> 200 + 50 --> 250
---- Read Balance of Acc-3	t7	COMMIT
Sum <-- Sum + 250 = 450	t8	----

Transaction-A is summing all balances; while, Transaction-B is transferring an amount 50 from Account-3 to Account-1.
Here, the result produced by Transaction-A is 550, which is incorrect. if this result is written in database, database will be in inconsistent state, as actual sum is 600.
Here, Transaction-A has seen an inconsistent state of database, and has performed inconsistent analysis.

6. What is deadlock? How does it occur? How transactions be written to (i) Avoid deadlock (ii) guarantee correct execution. Illustrate with suitable example. (Nov' 2014 Nov'2015 & Nov'2016)

(OR)

Explain the methods used to handle Deadlock? (Dec 2018)

(OR)

When does deadlock occurs? (Dec 2017, Nov 2019)

Deadlock

System is deadlocked if there is a set of transactions such that every transaction in the set is waiting for another transaction in the set.

- Consider the following two transactions:
  T_1: write (A) T_2: write(A)
  write(B) write(B)
- Schedule with deadlock

Avoid deadlock:

Deadlock prevention protocol
Ensure that the system will *never* enter into a deadlock state.

Some prevention strategies:

Approach1

- Require that each transaction locks all its data items before it begins execution either all are locked in one step or none are locked.
- Disadvantages
  - Hard to predict, before transaction begins, what data item need to be locked.
  - Data item utilization may be very low.

Approch2

- Assign a unique timestamp to each transaction.
- These timestamps only to decide whether a transaction should wait or rollback.

Schemes:

1. Wait-die scheme
2. Wound-wait scheme

Wait-die scheme

- non preemptive technique
- When transaction Ti request a data item currently held by Tj, Ti is allowed to wait only if it has a timestamp smaller than that of Tj. otherwise ,Ti rolled back(dies)
- **older transaction may wait for younger one** to release data item. Younger transactions never wait for older ones; they are rolled back instead.

– A transaction may die several times before acquiring needed data item

Example.

- Transaction T_1,T_2,T_3 have time stamps 5,10,15,respectively.
- if T_1 requests a data item held by T_2,then T_1 will wait.

- If T_3 request a data item held by T_2,then T_3 will be rolled back

Wound-wait scheme

- Preemptive technique
- When transaction T_i requests a data item currently held by T_j,T_i is allowed to wait only if it has a timestamp larger than that of T_j. Otherwise T_j is rolled back
- Older transaction wounds (forces rollback) of younger transaction instead of waiting for it. Younger transactions may wait for older ones.

Example

- Transaction T_1, T2, T3 have time stamps 5, 10, 15 respectively.
- if T_1 requests a data item held by T_2,then the data item will be preempted from T_2,and T_2 will be rolled back.
- If T_3 requests a data item held by T_2, then T_3 will wait.

7. Explain about dead lock recovery algorithm with an example.(Nov 2019)

The common solution is to roll back one or more transactions to break the deadlock.

Three action need to be taken

a. Selection of victim
b. Rollback
c. Starvation

Selection of victim

i. Set of deadlocked transations,must determine which transaction to roll back to break the deadlock.
ii. Consider the factor minimum cost

Rollback

- once we decided that a particular transaction must be rolled back, must determine how far this transaction should be rolled back
- Total rollback
- Partial rollback

Starvation

Ensure that a transaction can be picked as victim only a finite number of times.

8. Illustrate Granularity locking method in concurrency control.

Lock Granularity:

A database is basically represented as a collection of named data items. The size of the data item chosen as the unit of protection by a concurrency control program is called GRANULARITY.

Locking can take place at the following level:

- Database level.
- Table level.
- Page level.
- Row (Tuple) level.
- Attributes (fields) level.

i. Database level Locking:

At database level locking, the entire database is locked. Thus, it prevents the use of any tables in the database by transaction T2 while transaction T1 is being executed. Database level of locking is suitable for batch processes. Being very slow, it is unsuitable for on-line multi-user DBMSs.

ii. Table level Locking :

At table level locking, the entire table is locked. Thus, it prevents the access to any row (tuple) by transaction T2 while transaction T1 is using the table. if a transaction requires access to several tables, each table may be locked. However, two transactions can access the same database as long as they access different tables. Table level locking is less restrictive than database level. Table level locks are not suitable for multi-user DBMS

iii. Page level Locking:

At page level locking, the entire disk-page (or disk-block) is locked. A page has a fixed size such as 4 K, 8 K, 16 K, and 32 K so on. A table can span several pages, and a page can contain several rows (tuples) of one or more tables. Page level of locking is most suitable for multi-user DBMSs.

iv. Row (Tuple) level Locking:

At row level locking, particular row (or tuple) is locked. A lock exists for each row in each table of the database. The DBMS allows concurrent transactions to access different rows of the same table, even if the rows are located on the same page. The row level lock is much less restrictive than database level, table level, or page level locks. The row level locking improves the availability of data. However, the management of row level locking requires high overhead cost.

v. Attributes (fields) level Locking:

At attribute level locking, particular attribute (or field) is locked. Attribute level locking allows concurrent transactions to access the same row, as long as they require the use of different attributes within the row. The attribute level lock yields the most flexible multi-user data access. It requires a high level of computer overhead.

9. Describe Database Recovery concepts.

(OR)

Explain deferred and immediate modification versions of the log based recovery scheme? (May 2019)

Crash Recovery

Though we are living in highly technologically advanced era where hundreds of satellite monitor the earth and at every second billions of people are connected through information technology, failure is expected but not every time acceptable.

DBMS is highly complex system with hundreds of transactions being executed every second. Availability of DBMS depends on its complex architecture and underlying hardware or system software. If it fails or crashes amid transactions being executed, it is expected that the system would follow some sort of algorithm or techniques to recover from crashes or failures.

Failure Classification

To see where the problem has occurred we generalize the failure into various categories, as follows:

TRANSACTION FAILURE

When a transaction is failed to execute or it reaches a point after which it cannot be completed successfully it has to abort. This is called transaction failure. Where only few transaction or process are hurt.

Reason for transaction failure could be:

- **Logical errors:** where a transaction cannot complete because of it has some code error or any internal error condition
- **System errors:** where the database system itself terminates an active transaction because DBMS is not able to execute it or it has to stop because of some system condition. For example, in case of deadlock or resource unavailability systems aborts an active transaction.

SYSTEM CRASH

There are problems, which are external to the system, which may cause the system to stop abruptly and cause the system to crash. For example interruption in power supply, failure of underlying hardware or software failure.
Examples may include operating system errors.

DISK FAILURE:

In early days of technology evolution, it was a common problem where hard disk drives or storage drives used to fail frequently.
Disk failures include formation of bad sectors, unreachability to the disk, disk head crash or any other failure, which destroys all or part of disk storage.

Storage Structure

We have already described storage system here. In brief, the storage structure can be divided in various categories:

- **Volatile storage:** As name suggests, this storage does not survive system crashes and mostly placed very closed to CPU by embedding them onto the chipset itself for examples: main memory, cache memory. They are fast but can store a small amount of information.

- **Nonvolatile storage:** These memories are made to survive system crashes. They are huge in data storage capacity but slower in accessibility. Examples may include, hard disks, magnetic tapes, flash memory, non-volatile (battery backed up) RAM.

Recovery and Atomicity

When a system crashes, it many have several transactions being executed and various files opened for them to modifying data items. As we know that transactions are made of various operations, which are atomic in nature. But according to ACID properties of DBMS, atomicity of transactions as a whole must be maintained that is, either all operations are executed or none.

When DBMS recovers from a crash it should maintain the following:

- It should check the states of all transactions, which were being executed.
- A transaction may be in the middle of some operation; DBMS must ensure the atomicity of transaction in this case.
- It should check whether the transaction can be completed now or needs to be rolled back.
- No transactions would be allowed to left DBMS in inconsistent state.
- There are two types of techniques, which can help DBMS in recovering as well as maintaining the atomicity of transaction:
- Maintaining the logs of each transaction, and writing them onto some stable storage before actually modifying the database.
- Maintaining shadow paging, where are the changes are done on a volatile memory and later the actual database is updated.

Log-Based Recovery

Log is a sequence of records, which maintains the records of actions performed by a transaction. It is important that the logs are written prior to actual modification and stored on a stable storage media, which is failsafe.

Log based recovery works as follows:

- The log file is kept on stable storage media
- When a transaction enters the system and starts execution, it writes a log about it

$<T_n, Start>$

- When the transaction modifies an item X, it write logs as follows:

$<T_n, X, V_1, V_2>$

- It reads Tn has changed the value of X, from V1 to V2.
- When transaction finishes, it logs:

$<T_n, commit>$

Database can be modified using two approaches:

Deferred database modification: All logs are written on to the stable storage and database is updated when transaction commits.

Immediate database modification: Each log follows an actual database modification. That is, database is modified immediately after every operation.

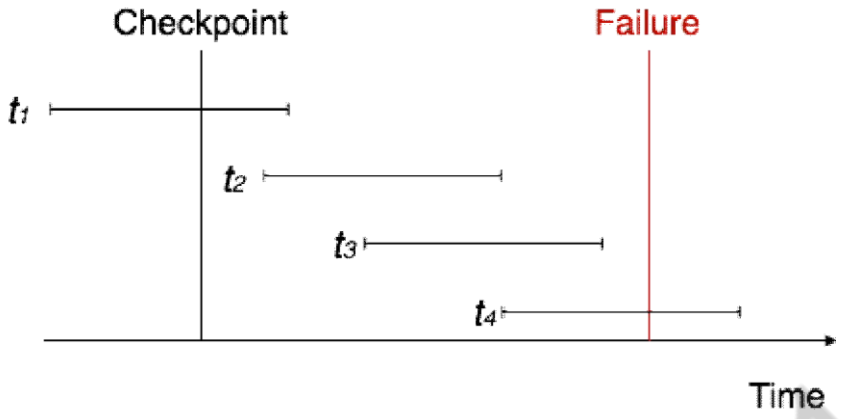
Recovery with concurrent transactions
When more than one transactions are being executed in parallel, the logs are interleaved. At the time of recovery it would become hard for recovery system to backtrack all logs, and then start recovering. To ease this situation most modern DBMS use the concept of 'checkpoints'.

CHECKPOINT

Keeping and maintaining logs in real time and in real environment may fill out all the memory space available in the system. At time passes log file may be too big to be handled at all. Checkpoint is a mechanism where all the previous logs are removed from the system and stored permanently in storage disk. Checkpoint declares a point before which the DBMS was in consistent state and all the transactions were committed.

RECOVERY

When system with concurrent transaction crashes and recovers, it does behave in the following manner:



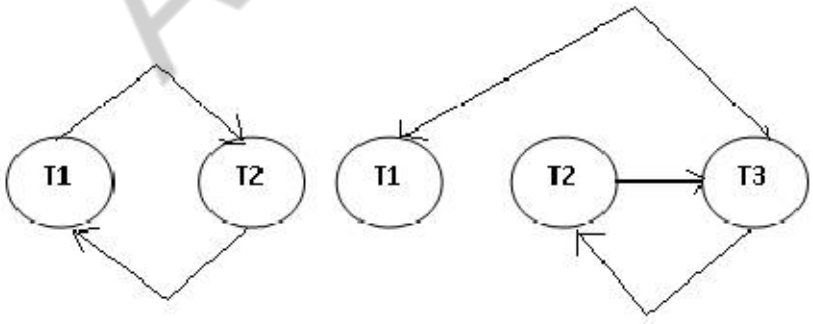
The recovery system reads the logs backwards from the end to the last Checkpoint.

- It maintains two lists, undo-list and redo-list.
- If the recovery system sees a log with <T_n, Start> and <T_n, Commit> or just <T_n, Commit>, it puts the transaction in redo-list.
- If the recovery system sees a log with <T_n, Start> but no commit or abort log found, it puts the transaction in undo-list.
- All transactions in undo-list are then undone and their logs are removed. All transaction in redo-list, their previous logs are removed and then redone again and log saved.

10. Consider the following schedules. The actions are listed in the order they are schedule, and prefixed with transaction name.
S1: T1: R(X), T2: R(x), T1: W(Y), T2: W(Y), T1: R(Y), T2: R(Y)
S2:T3: R(X), T1: R(X), T1: W(Y), T2: R (Z), T2: W (Z), T3: R (Z)
For each of the schedules, answer the following questions:

i. What is the precedence graph for the schedule?
ii. Is the schedule conflict-serializable? If so, what are all the conflict equivalent serial schedules?
iii. Is the schedule view-serializable? If so, what are all the view equivalent serial schedules? (Apr/May 2015)

The Precedence graph for this schedule



1. Serizability (or view) cannot be decided but NOT conflict serizability. It is recoverable and avoid cascading aborts; NOT strict

2. It is serializable, conflict-serializable, and view-serializable regardless which action (commit or abort) follows It is NOT avoid cascading aborts, NOT strict; We cannot decide whether it's recoverable or not, since the abort/commit sequence of these two transactions are not specified.
3. It is the same with 2.
4. Serizability (or view) cannot be decided but NOT conflict serizability. It is NOT avoided cascading aborts, NOT strict; we cannot decide whether it's recoverable or not, since the abort/commit sequence of these transactions are not specified.
5. It is serializable, conflict-serializable, and view-serializable; It is recoverable and avoid cascading aborts; it is NOT strict.
6. It is NOT serializable, NOT view-serializable, and NOT conflict-serializable; it is recoverable and avoids cascading aborts; It is NOT strict.
7. It belongs to all classes
8. It is serializable, NOT view-serializable, NOT conflict-serializable; it is NOT recoverable, therefore NOT avoid cascading aborts, NOT strict.
9. It is serializable, view-serializable, and conflict-serializable; It is NOT recoverable, therefore NOT avoid cascading aborts, NOT strict.
10. It belongs to all above classes.
11. It is NOT serializable and NOT view-serializable, NOT conflict-serializable; it is recoverable, avoid cascading aborts and strict.
12. It is NOT serializable and NOT view-serializable, NOT conflict-serializable; it is recoverable, but NOT avoid cascading aborts, NOT strict.

11. What is concurrency control? How is it implemented in DBMS? Illustrate with a suitable example. (Nov'2015, May 2019)
(OR)
How the timestamps are implemented ? Explain? (Dec 2018)
(OR)
State and explain lock based concurrency control with suitable example?(Dec 2017)

In a multiprogramming environment where multiple transactions can be executed simultaneously, it is highly important to control the concurrency of transactions. We have concurrency control protocols to ensure atomicity, isolation, and serializability of concurrent transactions. Concurrency control protocols can be broadly divided into two categories −

- Lock based protocols
- Time stamp based protocols

Lock-based Protocols

Database systems equipped with lock-based protocols use a mechanism by which any transaction cannot read or write data until it acquires an appropriate lock on it. Locks are of two kinds –

- **Binary Locks** − A lock on a data item can be in two states; it is either locked or unlocked.
- **Shared/exclusive** − This type of locking mechanism differentiates the locks based on their uses. If a lock is acquired on a data item to perform a write operation, it is an

exclusive lock. Allowing more than one transaction to write on the same data item would lead the database into an inconsistent state. Read locks are shared because no data value is being changed.

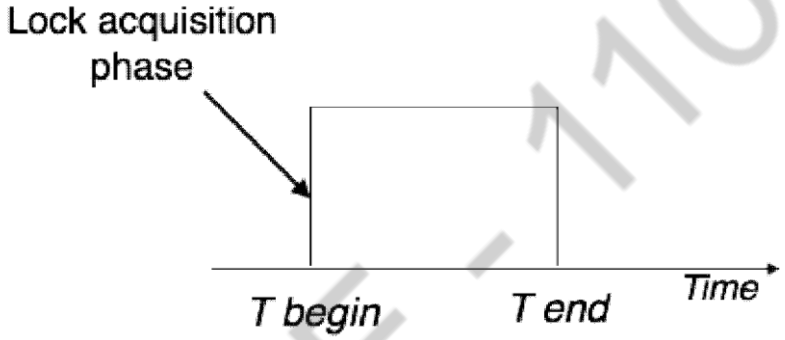
There are four types of lock protocols available −

Simplistic Lock Protocol

Simplistic lock-based protocols allow transactions to obtain a lock on every object before a 'write' operation is performed. Transactions may unlock the data item after completing the ‘write’ operation.

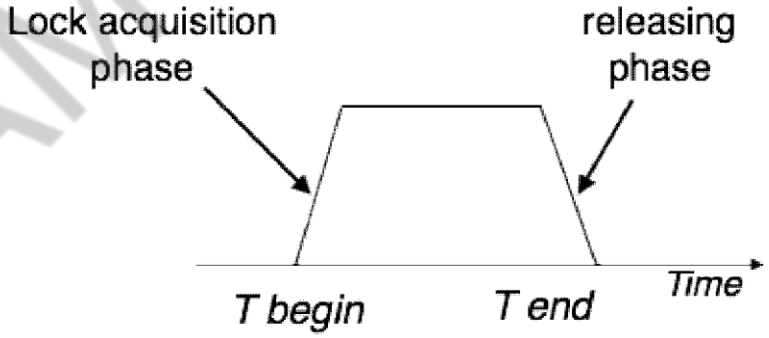
Pre-claiming Lock Protocol

Pre-claiming protocols evaluate their operations and create a list of data items on which they need locks. Before initiating an execution, the transaction requests the system for all the locks it needs beforehand. If all the locks are granted, the transaction executes and releases all the locks when all its operations are over. If all the locks are not granted, the transaction rolls back and waits until all the locks are granted.



Two-Phase Locking 2PL

This locking protocol divides the execution phase of a transaction into three parts. In the first part, when the transaction starts executing, it seeks permission for the locks it requires. The second part is where the transaction acquires all the locks. As soon as the transaction releases its first lock, the third phase starts. In this phase, the transaction cannot demand any new locks; it only releases the acquired locks.

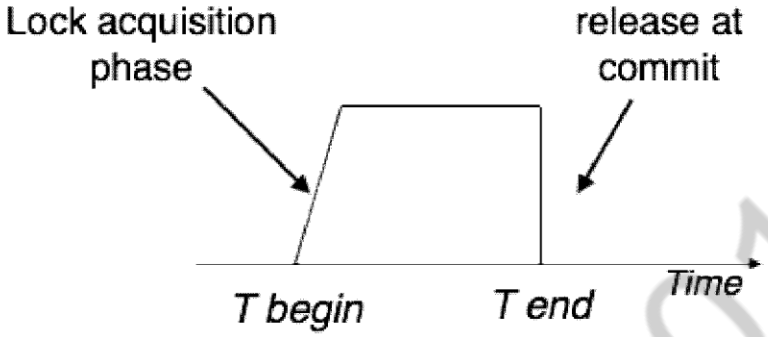


Two-phase locking has two phases, one is **growing**, where all the locks are being acquired by the transaction; and the second phase is shrinking, where the locks held by the transaction are being released.

To claim an exclusive (write) lock, a transaction must first acquire a shared (read) lock and then upgrade it to an exclusive lock.

Strict Two-Phase Locking

The first phase of Strict-2PL is same as 2PL. After acquiring all the locks in the first phase, the transaction continues to execute normally. But in contrast to 2PL, Strict-2PL does not release a lock after using it. Strict-2PL holds all the locks until the commit point and releases all the locks at a time.



Strict-2PL does not have cascading abort as 2PL does.

Timestamp-based Protocols

The most commonly used concurrency protocol is the timestamp based protocol. This protocol uses either system time or logical counter as a timestamp.

Lock-based protocols manage the order between the conflicting pairs among transactions at the time of execution, whereas timestamp-based protocols start working as soon as a transaction is created.

Every transaction has a timestamp associated with it, and the ordering is determined by the age of the transaction. A transaction created at 0002 clock time would be older than all other transactions that come after it. For example, any transaction 'y' entering the system at 0004 is two seconds younger and the priority would be given to the older one.

In addition, every data item is given the latest read and write-timestamp. This lets the system know when the last 'read and write' operation was performed on the data item.

Timestamp Ordering Protocol

The timestamp-ordering protocol ensures serializability among transactions in their conflicting read and write operations. This is the responsibility of the protocol system that the conflicting pair of tasks should be executed according to the timestamp values of the transactions.

- The timestamp of transaction T_i is denoted as $TS(T_i)$.
- Read time-stamp of data-item X is denoted by R-timestamp(X).
- Write time-stamp of data-item X is denoted by W-timestamp(X).

Timestamp ordering protocol works as follows −

- **If a transaction Ti issues a read(X) operation −**
  - If TS(Ti) < W-timestamp(X)
    - Operation rejected.
  - If TS(Ti) >= W-timestamp(X)
    - Operation executed.
  - All data-item timestamps updated.

- **If a transaction Ti issues a write(X) operation −**
  - If TS(Ti) < R-timestamp(X)
    - Operation rejected.
  - If TS(Ti) < W-timestamp(X)
    - Operation rejected and Ti rolled back.
  - Otherwise, operation executed.

Thomas' Write Rule

This rule states if TS(Ti) < W-timestamp(X), then the operation is rejected and T_i is rolled back.

Time-stamp ordering rules can be modified to make the schedule view serializable.

Instead of making T_i rolled back, the 'write' operation itself is ignored.

In a multi-process system, deadlock is an unwanted situation that arises in a shared resource environment, where a process indefinitely waits for a resource that is held by another process.

For example, assume a set of transactions $\{T_0, T_1, T_2, ..., T_n\}$. T_0 needs a resource X to complete its task. Resource X is held by T_1, and T_1 is waiting for a resource Y, which is held by T_2. T_2 is waiting for resource Z, which is held by T_0. Thus, all the processes wait for each other to release resources. In this situation, none of the processes can finish their task. This situation is known as a deadlock.

Deadlocks are not healthy for a system. In case a system is stuck in a deadlock, the transactions involved in the deadlock are either rolled back or restarted.

12. Explain about Locking Protocols. (May 2016 & Nov'2016) (or)
Briefly describe two phase locking in concurrency control techniques. (Nov'2016)
(OR)
Explain the two phase locking protocol with an example?(May 2018, Nov 2019)
(OR)
Differentiate strict two phase locking and rigorous two phase locking protocol with an example? (Dec 2018)

Two-Phase Locking (2PL)

- Two-Phase Locking (2PL) is a concurrency control method which divides the execution phase of a transaction into three parts.
- It ensures conflict serializable schedules.
- If read and write operations introduce the first unlock operation in the transaction, then it is said to be Two-Phase Locking Protocol.

This protocol can be divided into two phases,

1. **In Growing Phase,** a transaction obtains locks, but may not release any lock.
2. **In Shrinking Phase,** a transaction may release locks, but may not obtain any lock.

- Two-Phase Locking does not ensure freedom from deadlocks.

Types of Two – Phase Locking Protocol

Following are the types of two – phase locking protocol:

1. Strict Two – Phase Locking Protocol
2. Rigorous Two – Phase Locking Protocol
3. Conservative Two – Phase Locking Protocol

1. Strict Two-Phase Locking Protocol

- Strict Two-Phase Locking Protocol avoids cascaded rollbacks.
- This protocol not only requires two-phase locking but also all exclusive-locks should be held until the transaction commits or aborts.
- It is not deadlock free.
- It ensures that if data is being modified by one transaction, then other transaction cannot read it until first transaction commits.
- Most of the database systems implement rigorous two – phase locking protocol.

2. Rigorous Two-Phase Locking

- Rigorous Two – Phase Locking Protocol avoids cascading rollbacks.
- This protocol requires that all the share and exclusive locks to be held until the transaction commits.

3. Conservative Two-Phase Locking Protocol

- Conservative Two – Phase Locking Protocol is also called as Static Two – Phase Locking Protocol.
- This protocol is almost free from deadlocks as all required items are listed in advanced.
- It requires locking of all data items to access before the transaction starts.

UNIT-4

IMPLEMENTATION TECHNIQUES

PART-A

1. Give the measures of quality of a disk.

Capacity,Access time,Seek time,Data transfer rate Reliability
Rotational latency time.

2. Compare sequential access devices versus random access devices with an example

Sequential access devices

Must be accessed from the beginning Eg:- tape storage
Access to data is much slower Cheaper than disk

Random access devices

It is possible to read data from any location Eg:-disk storage
Access to data is faster
Expensive when compared with disk

3. What are the types of storage devices?

Primary storage
Secondary storage
Tertiary storage

4. Draw the storage device hierarchy according to their speed and their cost.

Cache , Main memory ,Flash memory, Magnetic disk ,Optical disk ,Magnetic tapes

5. What are called jukebox systems?

Jukebox systems contain a few drives and numerous disks that can be loaded into one of the drives automatically.

6. 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.

7. Define access time.

Access time is the time from when a read or write request is issued to when data transfer begins.

8. 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.

9. Define average seek time.

The average seek time is the average of the seek times, measured over a sequence of random requests.

10. Define rotational latency time.

The time spent waiting for the sector to be accessed to appear under the head is called the rotational latencytime.

11. Define average latency time.

The average latency time of the disk is one-half the time for a full rotation of the disk.

12. 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.

13. 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.

14. 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.

15. What are called journaling file systems?

File systems that support log disks are called journaling file systems.

16. What is the use of RAID?

A variety of disk-organization techniques, collectively called redundant arrays of independent disks are used to improve the performance and reliability.

17. 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 fail ed disk is repaired.

18. What is called mirroring?

The simplest approach to introducing redundancy is to duplicate every disk. This technique is called mirroring or shadowing.

19. 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.

20. 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.

21. 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.

22. 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

23. What are the factors to be taken into account when choosing a RAID level?

o Monetary cost of extra disk storage requirements.

o Performance requirements in terms of number of I/O operations o Performance when a disk has failed.

o Performances during rebuild.

24. Define software and hardware RAID systems. (May 2016)

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.

25. 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.

26. 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.

27. 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.

28. What are the ways in which the variable-length records represented in database systems? (Nov 2018)

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.

29. 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

30. 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.

31. 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.

32. 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 is a single file for each relation.

33. 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.

34. 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.

35. What is known as clustering file organization?

In the clustering file organization, records of several different relations are stored in the same file.

36. What is an index?

An index is a structure that helps to locate desired records of a relation quickly, without examining all records.

37. What are the two types of ordered indices?

Primary index

Secondary index

38. What are the types of indices?

Ordered indices

Hash indices

39. What are the techniques to be evaluated for both ordered indexing and hashing?

Access types
Access time
Insertion time
Deletion time
Space overhead

40. 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.

41. What is a primary index?

A primary index is an index whose search key also defines the sequential order of the file.

42. 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.

43. What are the two types of indices?

Dense index
Sparse index

44. What are called multilevel indices?

Indices with two or more levels are called multilevel indices.

45. 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.

46. 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.

47. 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.

48. What is B-Tree?

A B-tree eliminates the redundant storage of search-key values .It allows search key values to appear only once.

49. 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.

50. Define static hashing and dynamic hashing.

Static hashing

Static hashing uses a hash function in which the set of bucket address is fixed. Such hash functions cannot easily accommodate databases that grow larger over time.

Dynamic hashing

Dynamic hashing allows us to modify the hash fufnction dynamically. Dynamic hashing copes with changes in database size by splitting and coalescing buckets as the database grows and shrinks.

51. Differentiate between static hashing and dynamic hashing. (Nov'2014 , Dec 2014, May 2015 & Dec 2015)

Static Hashing	Dynamic Hashing
In static hashing, when a search-key value is provided, the hash function always computes the same address.	Hash function, in dynamic hashing, is made to produce a large number of values and only a few are used initially.
The number of buckets provided remains unchanged at all times i.e. fixed	Dynamic hashing provides a mechanism in which data buckets are added and removed dynamically and on-demand .i.e. no. of buckets not fixed.
Space and overhead is more	Minimum space and less overhead
As file grows performance decreases	Performance do not degrade as file grows

52. What is a hash index?

A hash index organizes the search keys, with their associated pointers, into a hash file structure.

53. 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

54.Differentiate open hashing and closed hashing (overflow chaining)

Closed hashing

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.

55. 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 known as linear probing.

56. Give an example of a join that is not a simple equi-join for which partitioned parallelism can be used. (Nov/Dec 2015)

$$r \text{ join } (r.A = s.B) \wedge (r.A < s.c)$$

57. What is mean by garbage collection? (May 2016)

Garbage collection (GC) is a dynamic approach to automatic memory management and heap allocation that processes and identifies dead memory blocks and reallocates storage for reuse. The primary purpose of garbage collection is to reduce memory leaks.

GC implementation requires three primary approaches, as follows:

- Mark-and-sweep - In process when memory runs out, the GC locates all accessible memory and then reclaims available memory.
- Reference counting - Allocated objects contain a reference count of the referencing number. When the memory count is zero, the object is garbage and is then destroyed. The freed memory returns to the memory heap.
- Copy collection - There are two memory partitions. If the first partition is full, the GC locates all accessible data structures and copies them to the second partition, compacting memory after GC process and allowing continuous free memory.

58. List out the mechanisms to avoid collision during hashing. (Dec'2016)

The various mechanisms to avoid collision during hashing are

- Closed Hashing (Open addressing)
  - Linear Probing
  - Quadratic Probing
  - Double hashing
- Open Hashing
  - Separate Chaining

59. What are the disadvantages of B Tree over B+ Tree. (Dec'2016)

(OR)

Why is a B+ tree usually preferred as an access structure to a data file?(Nov 2018)

The disadvantages of B Tree over B+ Tree are:

- In a B tree, search keys and data stored in internal or leaf nodes which leads to wastage of memory.
- Here in B tree the search is not that easy as compared to a B+ tree.

- Here, searching becomes difficult in B- tree as data cannot be found in the leaf node.
- In B tree, the leaf node cannot store using linked list.

60. Differentiate between B Tree over B+ Tree.

	B Tree	**B+ Tree**
Storage	In a B tree, search keys and data stored in internal or leaf nodes.	In a B+ tree, data stored only in leaf nodes.
Data	The leaf nodes of the three store pointers to records rather than actual records.	The leaf nodes of the tree stores the actual record rather than pointers to records.
Space	These trees waste space	There trees do not waste space.
Function of leaf nodes	In B tree, the leaf node cannot store using linked list.	In B+ tree, leaf node data are ordered in a sequential linked list.
Searching	Here, searching becomes difficult in B- tree as data cannot be found in the leaf node.	Here, searching of any data in a B+ tree is very easy because all data is found in leaf nodes.
Search accessibility	Here in B tree the search is not that easy as compared to a B+ tree.	Here in B+ tree the searching becomes easy.
Redundant key	They do not store redundant search key.	They store redundant search key.
Applications	They are an older version and are not that advantageous as compared to the B+ trees.	Many database system implementers prefer the structural simplicity of a B+ tree.

61. State the need for Query Optimization? (MAY 2015)

To manage long-running queries
Sharing of system resources
Faster server and application processing
To avoid table locking and data corruption issues
Cost reduction

62. What is called query processing? What are the steps involved in query processing? (APR 2018, NOV 2017)

Query processing refers to the range of activities involved in extracting data from a database.

The basic steps are:
Parsing and translation
optimization
evaluation

63. What is called an evaluation primitive?

A relational algebra operation annotated with instructions on how to evaluate is called an evaluation primitive.

64. What is called a query evaluation plan? How do you measure the cost of query evaluation? A sequence of primitive operations that can be used to evaluate ba query is a query evaluation plan or a query execution plan.

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

65. 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.

66. List out the operations involved in query processing

Selection operation
Join operations. Sorting. Projection
Set operations Aggregation

67. What are called as index scans?

Search algorithms that use an index are referred to as index scans.

68. What is called as external sorting?

Sorting of relations that do not fit into memory is called as external sorting.

69. Explain nested loop join? What is meant by block nested loop join?

Nested loop join consists of a pair of nested for loops. r is the outer relation and s is the inner relation.

Block nested loop join is the variant of the nested loop join where every block of the inner

relation is paired with every block of the outer relation. With in each pair of blocks every tuple in one block is paired with every tuple in the other blocks to generate all pairs of tuples.

70. What is meant by hash join?

In the hash join algorithm a hash function h is used to implement partition tuples of both relations.

71. 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.

72. 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.

73. What is known as fudge factor?

The number of partitions is increased by a small value called the fudge factor,which is usually 20 percent of the number of hash partitions computed

74. What is a query execution plan? (MAY 2017)

A query plan (or query execution plan) is an ordered set of steps used to access data in a SQL relational database management system. This is a specific case of the relational model concept of access plans.

75.What cost components are used most often as the basic for cost function? (MAY 2017)

Cost Components for Query Execution

The cost of executing a query includes the following components:

1. Access cost to secondary storage.
2. Disk storage cost.
.3. Computation cost.
4. Memory usage cost.
5. Communication cost.

76. What are ordered indices? (May 2014, Dec 2017)
(OR)
Define ordered indices with example?

This is type of indexing which is based on sorted indexing values. Various ordered indices are primary indexing, secondary indexing.

77.What are data fragmentation? State the various fragmentation with an example?(Dec 2017)

- Fragmentation
  - The system partitions the relation into several fragment and stores each fragment at different sites
  - Two approaches
    - Horizontal fragmentation
    - Vertical fragmentation

Horizontal fragmentation

Splits the relation by assigning each tuple of r to one or more fragments

relation r is partitioned into a number of subsets, r_1 ,r_2,…..r_n and can be reconstruct the original relation using union of all fragments, that is

$$r = r_1 \cup r_2 \cup \ldots\ldots \cup r_n$$

- Vertical fragmentation
  - Splits the relation by decomposing scheme R of relation and reconstruct the original relation by using natural join of all fragments. that is

$$r = r_1 \quad r_2 \quad \ldots\ldots \quad r_n$$

78.What is a hash function? Give an example?

In a hash file organization we obtain the bucket of a record directly from its search-key value using a hash function.

Hash function h is a function from the set of all search-key values K to the set of all bucket addresses B.

Hash function is used to locate records for access, insertion as well as deletion.

Example:

-There are 10 buckets,

-The hash function returns the sum of the binary representations of the characters modulo 10

- E.g. h(Perryridge) = 5 h(Round Hill) = 3 h(Brighton) = 3

79.Define dense index?(May 2019)

Dense index — Index record appears for every search-key value in the file.

- Secondary indices have to be dense.

80. Mention all the operations of file? (May 2019)

The various operations of file are:

- Read
- Write
- Search
- Update
- Delete

81.How do you represent leaf node of a B+ tree of order P? (Nov 2019)

The structure of the leaf nodes of a B+ tree of order 'b' is as follows: Each leaf node is of the form : $<<K_1, D_1>, <K_2, D_2>, \ldots.., <K_{c-1}, D_{c-1}>, P_{next}>$ Every leaf node has : $K_1 < K_2 < \ldots. < K_{c-1}$, c <= b. Each leaf node has at least \ceil(b/2) values. All leaf nodes are at same level.

82.Which cost components contribute to query execution? (Nov 2019)

- Disk storage cost.
- Computation cost.
- Memory usage cost.
- Communication cost.

PART - B

1.What is meant by semantic query optimization? How does it differ from other query optimization technique? Give example? (MAY 2017)

Semantic Query Optimizer

- Semantic query optimization is the process of determining the set of semantic transformations that results in a semantically equivalent query with a lower execution cost.
- Two queries are **semantically equivalent** if they return the same answer for any database state satisfying a given set of integrity constraints.
- A **semantic transformation** transforms a given query into a semantically equivalent one.
- ODB-QOptimizer determines more specialized classes to be accessed and reduces the number of factors by applying the Integrity Constraint Rules.

Semantic query optimization is the process of transforming a query issued by a user into a different query which, because of the semantics of the application, is guaranteed to yield the correct answer for all states of the database. While this process has been successfully applied in centralised databases, its potential for distributed and heterogeneous systems is enormous, as there is the potential to eliminate inter-site joins which are the single biggest cost factor in query processing. Further justification for its use is provided by the fact that users of heterogeneous databases typically issue queries through high-level languages which may result in very inefficient queries if mapped directly, without consideration of the semantics of the system. Even if this is not the case, users cannot be expected to be familiar with the semantics of the component databases, and may consequently issue queries which are unnecessarily complicated.

A different approach to query optimization, called semantic query optimization, has been suggested. This technique, which may be used in combination with the techniques discussed previously, uses constraints specified on the database schema—such as unique attributes and other more complex constraints—in order to modify one query into another query that is more efficient to execute. We will not discuss this approach in detail but we will illustrate it with a simple example. Consider the SQL query:

```
SELECT      E.Lname, M.Lname

FROM        EMPLOYEE AS E, EMPLOYEE AS M

WHERE       E.Super_ssn=M.Ssn AND E.Salary > M.Salary
```

This query retrieves the names of employees who earn more than their supervisors. Suppose that we had a constraint on the database schema that stated that no employee can earn more than his or her direct

supervisor. If the semantic query optimizer checks for the existence of this constraint, it does not need to execute the query at all because it knows that the result of the query will be empty. This may save considerable time if the constraint checking can be done efficiently. However, searching through many constraints to find those that are applicable to a given query and that may semantically optimize it can also be quite time-consuming. With the inclusion of active rules and additional metadata in database systems, semantic query optimization techniques are being gradually incorporated into the DBMSs.

The term quote semantic query optimization quote (SQO) denotes a methodology whereby queries against databases are optimized using semantic information about the database objects being queried. The result of semantically optimizing a query is another query which is syntactically different to the original, but semantically equivalent and which may be answered more efficiently than the original. SQO is distinctly different from the work performed by the conventional SQL optimizer. The SQL optimizer generates a set of logically equivalent alternative execution paths based ultimately on the rules of relational algebra. However, only a small proportion of the readily available semantic information is utilised by current SQL optimizers. Researchers in SQO agree that SQO can be very effective.

However, after some twenty years of research into SQO, there is still no commercial implementation. In this thesis we argue that we need to quantify the conditions for which SQO is worthwhile. We investigate what these conditions are and apply this knowledge to relational database management systems (RDBMS) with static schemas and infrequently updated data. Any semantic query optimizer requires the ability to reason using the semantic information available, in order to draw conclusions which ultimately facilitate the recasting of the original query into a form which can be answered more efficiently. This reasoning engine is currently not part of any commercial RDBMS implementation.

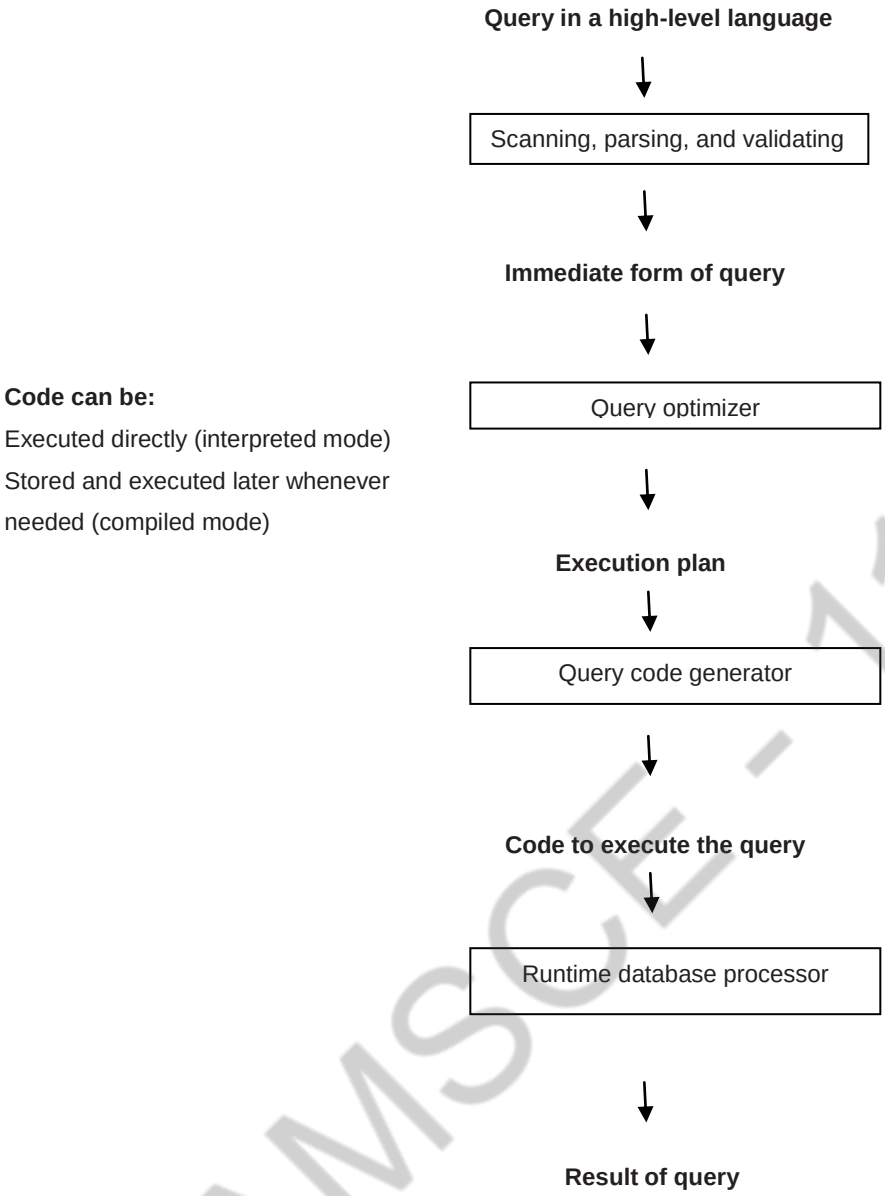
2.**Briefly explain about Query Processing? (Apr 2016, Nov 2019) (or)**

Explain query optimization with an example? (Nov 2016) (or)
What is Query Optimization? Explain the steps in Query Optimization? (APR 2018)

QUERY PROCESSING AND OPTIMIZATION

A query expressed in a high-level query language such as SQL must first be scanned, parsed, and validated.1 The **scanner** identifies the query tokens—such as SQL keywords, attribute names, and relation names—that appear in the text of the query, whereas the **parser** checks the query syntax to determine whether it is formulated according to the syntax rules (rules of grammar) of the query language. The query must also be **validated** by checking that all attribute and relation names are valid and semantically meaningful names in the schema of the particular database being queried. An internal representation of the query is then created, usually as a tree data structure called a **query tree**. It is also possible to represent the query using a graph data structure called a **query graph**. The DBMS must then devise an **execution strategy** or **query plan** for retrieving the results of the query from the database files. 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**

Different steps of processing a high level query



The **query optimizer** module has the task of producing a good execution plan, and the **code generator** generates the code to execute that plan. The **runtime database processor** has the task of running (executing) the query code, whether in compiled or interpreted mode, to produce the query result. If a runtime error results, an error message is generated by the runtime database processor.

TRANSLATING SQL QUERIES INTO RELATIONAL ALGEBRA

An SQL query is first translated into an equivalent extended relational algebra expression represented as a query tree data structure—that is then optimized. Typically, SQL queries are decomposed into *query blocks,* which form the basic units that can be translated into the algebraic operators and optimized. A **query block** contains a

single SELECT-FROM-WHERE expression, as well as GROUP BY and HAVING clauses if these are part of the block. Hence, nested queries within a query are identified as MIN, SUM, and COUNT

Consider the following SQL query on the EMPLOYEE relation:

SELECT Lname, Fname
FROM EMPLOYEE
WHERE Salary > (**SELECT MAX** (Salary)
FROM EMPLOYEE
WHERE Dno=5);

This query retrieves the names of employees (from any department in the company) who earn a salary that is greater than the *highest salary in department 5*. The query includes a nested subquery and hence would be decomposed into two blocks.

The inner block is:

(**SELECT MAX** (Salary)
FROM EMPLOYEE
WHERE Dno=5)

This retrieves the highest salary in department 5.

The outer query block is:

SELECT Lname, Fname
FROM EMPLOYEE
WHERE Salary > c

where c represents the result returned from the inner block. The inner block couldbe translated into the following extended relational algebra expression:

ℑMAX Salary(σDno=5(EMPLOYEE))

and the outer block into the expression:

πLname,Fname(σSalary>c(EMPLOYEE))

The query optimizer would then choose an execution plan for each query block.

3.Discuss about the Join order optimization and Heuristic optimization algorithms? (APR 2015)

Join Order Optimization Algorithm

The join order is fixed if any join logical files are referenced. The join order is also fixed if the OPNQRYF JORDER(*FILE) parameter is specified or the query options file (QAQQINI) FORCE_JOIN_ORDER parameter is *YES.

Otherwise, the following join ordering algorithm is used to determine the order of the tables:

1. Determine an access method for each individual table as candidates for the primary dial.
2. Estimate the number of rows returned for each table based on local row selection.

If the join query with row ordering or group by processing is being processed in one step, then the table with the ordering or grouping columns is the primary table.

3. Determine an access method, cost, and expected number of rows returned for each join combination of candidate tables as primary and first secondary tables.

   The join order combinations estimated for a four table inner join would be:
   1-2 2-1 1-3 3-1 1-4 4-1 2-3 3-2 2-4 4-2 3-4 4-3.
4. Choose the combination with the lowest join cost and number of selected rows or both.
5. Determine the cost, access method, and expected number of rows for each remaining table joined to the previous secondary table.
6. Select an access method for each table that has the lowest cost for that table.
7. Choose the secondary table with the lowest join cost and number of selected rows or both.
8. Repeat steps 4 through 7 until the lowest cost join order is determined.

Heuristic Optimization

Perform selection early (reduces the number of tuples).
Perform projection early (reduces the number of attributes)
Perform most restrictive selection and join operations before other similar operations.
Steps in Typical Heuristic Optimization
1. Deconstruct conjunctive selections into a sequence of single selection operations (Equiv. rule 1).
2. Move selection operations down the query tree for the earliest possible execution (Equiv. rules 2,7a, 7b, 11).
3. Execute first those selection and join operations that will produce the smallest relations (Equiv.rule 6).
4. Replace Cartesian product operations that are followed by a selection condition by join operations (Equiv. rule 4a).
5. Deconstruct and move as far down the tree as possible lists of projection attributes, creating new projections where needed (Equiv. rules 3, 8a, 8b, 12).
6. Identify those subtrees whose operations can be pipelined, and execute them using pipelining.

Structure of Query Optimizers

_ The System R optimizer considers only left-deep join orders.
_ This reduces optimization complexity and generates plans amenable to pipelined evaluation. System R also uses heuristics to push selections and projections down the query tree.
_ For scans using secondary indices, the Sybase optimizer takes into account the probability that the page containing the tuple is in the buffer.
_ Some query optimizers integrate heuristic selection and the generation of alternative access plans.
– System R and Starburst use a hierarchical procedure based on the nested-block concept of SQL: heuristic rewriting followed by cost-based join-order optimization.
– The Oracle7 optimizer supports a heuristic based on available access paths.
_ Even with the use of heuristics, cost-based query optimization imposes a substantial overhead.
This expense is usually more than offset by savings at query-execution time, particularly by

reducing the number of slow disk accesses.

4.Give a detailed description about Query Processing and optimization. Explain the cost estimation of Query Optimization? (Nov 2014) (or)

Explain the catalog information for cost estimation for selection and sorting operation in database? (NOV 2017) (or)
How does a DBMS represent a relational query evaluation plan? (NOV 2018)

BASIC STEPS IN QUERY PROCESSING

1. Parsing and translation
2. Optimization
3. Evaluation

Parsing and translation
Parsing and translation translates the query into its internal form. This is then translated into

$\sigma_{balance<2500}(\Pi_{balance}(account))$ is

$\Pi_{balance}(\sigma_{balance<2500}(account))$ equivalent to

xecutes that plan, and returns the

ıery.

ılent expressions E.g.

$$b_r = \left\lceil \frac{n_r}{f_r} \right\rceil$$

Catalog Information about Indices
fi : average fan-out of internal nodes of index i, for tree-structured indices such as B+-trees.
HTi : number of levels in index i — i.e., the height of i .
– For a balanced tree index (such as a B+-tree) on attribute A of relation r ,
HTi = dlogfi (V (A, r)e.
– For a hash index, HTi is 1.
LBi : number of lowest-level index blocks in i — i.e., the number of blocks at the leaf level of the index.

Measures of Query Cost

• Many possible ways to estimate cost, for instance disk accesses, CPU time, or even Communication overhead in a distributed or parallel system.

Typically disk access is the predominant cost, and is also relatively easy to estimate. Therefore number of block transfers from disk is used as a measure of the actual cost of evaluation. It is assumed that all transfers of blocks have the same cost.

• Costs of algorithms depend on the size of the buffer in main memory, as having more memory reduces need for disk access. Thus memory size should be a parameter while estimating cost; often use worst case estimates.

• We refer to the cost estimate of algorithm A as EA. We do not include cost of writing output to disk.

Selection Operation

File scan – search algorithms that locate and retrieve records that fulfill a selection condition.
Algorithm A1 (linear search):

Scan each file block and test all records to see whether they satisfy the selection condition.
– Cost estimate (number of disk blocks scanned) EA1 = br
– If selection is on a key attribute, EA1 = (br / 2) (stop on finding record)
– Linear search can be applied regardless of
1 selection condition, or
2 ordering of records in the file, or
3 availability of indices

Algorithm A2 (binary search):

Applicable if selection is an equality comparison on the attribute on which file is ordered.
– Assume that the blocks of a relation are stored contiguously.
– Cost estimate (number of disk blocks to be scanned):
[log2(br)] — cost of locating the first tuple by a binary search on the blocks.
SC(A, r) —number of records that will satisfy the selection.
[SC(A, r)/fr] — number of blocks that these records will occupy.
- Equality condition on a key attribute:
SC(A, r) = 1;
estimate reduces to
EA2 = [log2(br)]
Statistical Information for Examples
• faccount = 20 (20 tuples of account fit in one block)
• V (branch-name, account) = 50 (50 branches)
• V (balance, account) = 500 (500 different balance values)
• naccount = 10000 (account has 10,000 tuples)
• Assume the following indices exist on account:
– A primary, B+-tree index for attribute branch-name
– A secondary, B+-tree index for attribute balance

Selection Cost Estimate Example

Number of blocks is baccount = 500:
10, 000 tuples in the relation; each block holds 20 tuples.
Assume account is sorted on branch-name.
– V (branch-name, account) is 50

– 10000/ 50 = 200 tuples of the account relation pertain to Perryridge branch
– 200/ 20 = 10 blocks for these tuples
– A binary search to find the first record would take
dlog2(500)e = 9 block accesses
Total cost of binary search is 9 + 10 ? 1 = 18 block accesses (versus 500 for linear scan)

Since Indices speed query processing why might they not be kept on several search keys? List as many reasons as possible? (NOV 2018)

5.Describe File Organization.

(OR)

How the records are represented and organized in files.Explain wih suitable example? (Dec 2018)

-The database is stored as a collection of ***files***.
- Each file is a sequence of ***records***.
-A **record** is a sequence of fields.
-Classifications of records

i. **Fixed length record**
ii. **Variable length record**

Fixed length record approach:

- assume record size is fixed
- each file has records of one particular type only
- Different files are used for different relations. This case is easiest to implement.

- Simple approach:
  - Record access is simple
  - Modification: do not allow records to cross block boundaries

Variable length record

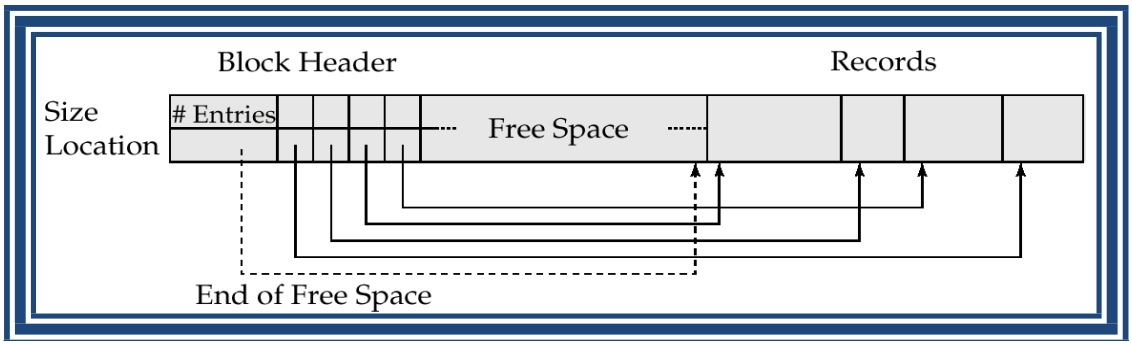
- Deletion of record *I:*
  alternatives*:*
  - move records $i + 1, \ldots, n$ to $i, \ldots, n - 1$
  - do not move records, but link all free records on a *free list*
- Variable-length records arise in database systems in several ways:
  - Storage of multiple record types in a file.
  - Record types that allow variable lengths for one or more fields.
- Byte string representation
  - Attach an *end-of-record* (⊥) control character to the end of each record
  - Difficulty with deletion

0	perryridge	A-102	400	A-201	900	⊥
1	roundhill	A-305	350	⊥		
2	mianus	A-215	700	⊥		

Slotted Page Structure

Slotted page header contains:

- number of record entries
- end of free space in the block
- location and size of each record

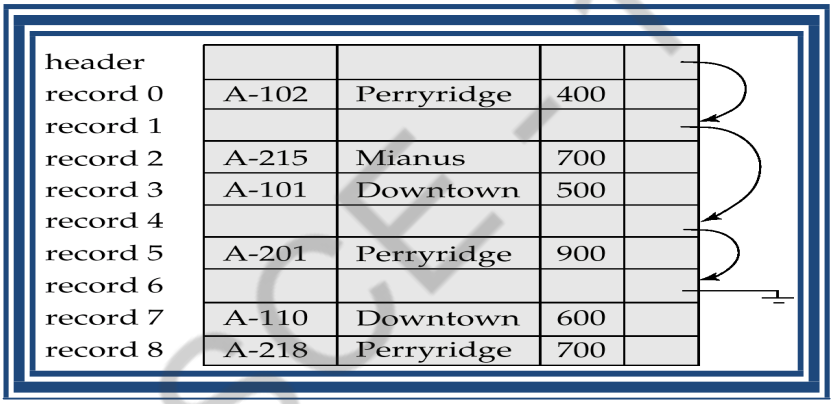


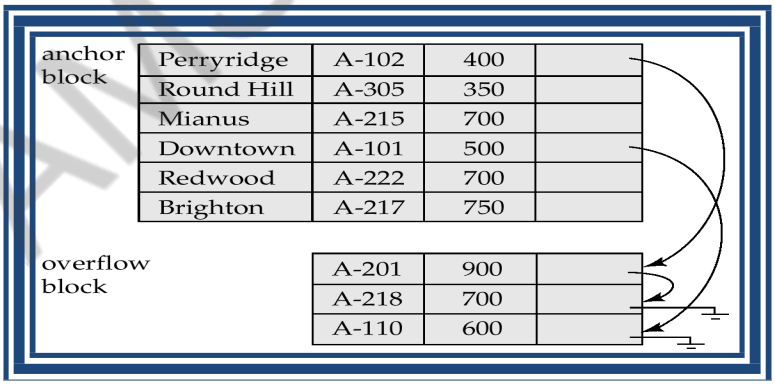
Free Lists

- Store the address of the first deleted record in the file header.
- Use this first record to store the address of the second deleted record, and so on

Pointer method

- A variable-length record is represented by a list of fixed-length records, chained together via pointers.
- Can be used even if the maximum record length is not known





- Disadvantage to pointer structure; space is wasted in all records except the first in a a chain.
- Solution is to allow two kinds of block in file:
  - Anchor block – contains the first records of chain

Overflow block – contains records other than those that are the first records of chains.

6.Define RAID and Briefly Explain RAID techniques. (Nov'2014, May 2015, Nov'2015, May 2016 & Nov'2016)
(OR)
What is RAID? Briefly discuss about RAID? (May 2019, Nov 2019)

RAID: Redundant Arrays of Independent Disks

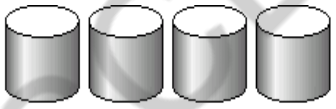
a. disk organization techniques that manage a large numbers of disks, providing a view of a single disk of
   i. high capacity and high speed by using multiple disks in parallel, and
   ii. high reliability by storing data redundantly, so that data can be recovered even if a disk fails

RAID Level 0: striping; non-redundant.

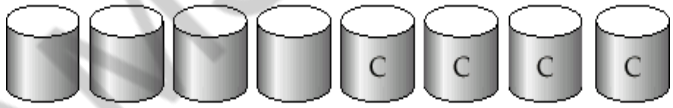
- Used in high-performance applications where data lost is not critical.

RAID Level 1: **Mirroring** (or **shadowing**)

- Duplicate every disk.
- Every write is carried out on both disks
- If one disk in a pair fails, data still available in the other
  - Data loss would occur only if a disk fails, and its mirror disk also fails before the system is repaired
    - Probability of combined event is very small



(a) RAID 0: nonredundant striping



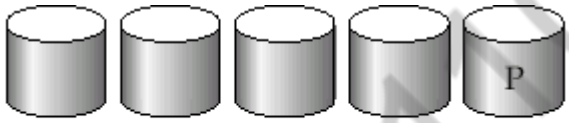
(b) RAID 1: mirrored disks

- **RAID Level 2**: Error-Correcting-Codes (ECC) with bit striping.
- **RAID Level 3**: Bit-Interleaved Parity
  - a single parity bit is enough for error correction, not just detection, since we know which disk has failed
    - When writing data, corresponding parity bits must also be computed and written to a parity bit disk
    - To recover data in a damaged disk, compute XOR of bits from other disks (including parity bit disk)

(c) RAID 2: memory-style error-correcting codes

(d) RAID 3: bit-interleaved parity

- **RAID Level 4:** Block-Interleaved Parity.
  - When writing data block, corresponding block of parity bits must also be computed and written to parity disk
  - To find value of a damaged block, compute XOR of bits from corresponding blocks (including parity block) from other disks.



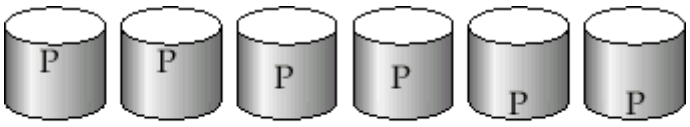
(e) RAID 4: block-interleaved parity

–

- **RAID Level 5:** Block-Interleaved Distributed Parity; partitions data and parity among all *N* + 1 disks, rather than storing data in *N* disks and parity in 1 disk.

(f) RAID 5: block-interleaved distributed parity

- **RAID Level 6**: P+Q Redundancy scheme; similar to Level 5, but stores extra redundant information to guard against multiple disk failures.
  - Better reliability than Level 5 at a higher cost; not used as widely.



(g) RAID 6: P + Q redundancy

7.Explain Secondary storage devices.

Can differentiate storage into:

b. **volatile storage:** loses contents when power is switched off
c. **non-volatile storage**:
   i. Contents persist even when power is switched off.
   ii. Includes secondary and tertiary storage.

primary storage: Fastest media but volatile (cache, main memory).

secondary storage: next level in hierarchy, non-volatile, moderately fast access time. also called **on-line storage**

tertiary storage: lowest level in hierarchy, non-volatile, slow access timealso called **off-line storage**

E.g. magnetic tape, optical storage

Cache – fastest and most costly form of storage; volatile; managed by the computer system hardware.

Main memory:

- fast access
- generally too small
- capacities of up to a few Gigabytes widely used currently
- **Volatile** — contents of main memory are usually lost if a power failure or system crash occurs.

Flash memory

- Data survives power failure
- Data can be written at any location.
- The location can be erased and written to again
- Can support only a limited number of write/erase cycles.
- Reads are fast.
- But writes are slow (few microseconds), erase is slower
- also known as EEPROM (Electrically Erasable Programmable Read-Only Memory)

Magnetic-disk

a. Data is stored on spinning disk, and read/written magnetically
b. Primary medium for the long-term storage of data.
c. Data must be moved from disk to main memory for access, and written back for storage
    i. Much slower access than main memory
d. **direct-access** – possible to read data on disk in any order, unlike magnetic tape
    i. Much larger capacity than main memory/flash memory
    ii. disk failure can destroy data, but is very rare

Read-write head

-Positioned very close to the platter surface

-Reads or writes magnetically.

-Surface of platter divided into circular **tracks**

-Each track is divided into **sectors.**

-A sector is the smallest unit of data that can be read or written.

-Head-disk assemblies

-multiple disk platters on a single spindle (typically 2 to 4)

-ne head per platter, mounted on a common arm.

Cylinder *i* consists of i^{th} track of all the platters

Disk controller – interfaces between the computer system and the disk drive hardware.

- accepts high-level commands to read or write a sector
- initiates actions such as moving the disk arm to the right track and actually reading or writing the data

  - Ensures successful writing by reading back sector after writing it.
- **Optical storage**
  - non-volatile, data is read optically from a spinning disk using a laser
  - CD-ROM and DVD most popular forms
  - Write-one, read-many (WORM) optical disks are available (CD-R and DVD-R)
  - Multiple write versions also available (CD-RW, DVD-RW)
  - Reads and writes are slower than with magnetic disk
- **Tape storage**
  - non-volatile, Used mainly for backup, for storage of infrequently used information, and as an off-line medium for transferring information from one system to another..
- Hold large volumes of data and provide high transfer rates
  - **sequential-access** – much slower than disk
- Very slow access time in comparison to magnetic disks and optical disks
  - very high capacity (300 GB tapes available)
  - storage costs much cheaper than disk, but drives are expensive

8.Explain about Multidimensional and Parallel with an example

MULTIDIMENSIONAL DATABASES:

A multidimensional database (MDB) is a type of database that is optimized for data warehouse and online analytical processing (OLAP) applications. Multidimensional databases are frequently created using input from existing relational databases. Whereas a relational database is typically accessed using a Structured Query Language (SQL) query, a multidimensional database allows a user to ask questions like "How many Aptivas have been sold in Nebraska so far this year?" and similar questions related to summarizing business operations and trends. An OLAP application that accesses data from a multidimensional database is known as a MOLAP (multidimensional OLAP) application.

A multidimensional database - or a multidimensional database management system (MDDBMS) - implies the ability to rapidly process the data in the database so that answers can be generated quickly. A number of vendors provide products that use multidimensional databases. Approaches to how data is stored and the user interface vary.Conceptually, a multidimensional database uses the idea of a data cube to represent the dimensions of data available to a user. For example, "sales" could be viewed in the dimensions of product model, geography, time, or some additional dimension. In this case, "sales" is known as the *measure attribute* of the data cube and the other dimensions are seen as *feature attribute*s. Additionally, a database creator can define hierarchies and levels within a dimension (for example, state and city levels within a regional hierarchy).

PARALLEL DATABASES:

A parallel database is designed to take advantage of such architectures by running multiple instances which "share" a single physical database. In appropriate applications, a parallel server can allow access to a single database by users on multiple machines, with increased performance.

A parallel server processes transactions in parallel by servicing a stream of transactions using multiple CPUs on different nodes, where each CPU processes an entire transaction. Using parallel data manipulation language you can have one transaction being performed by multiple nodes. This is an efficient approach because many applications consist of online insert and update transactions which tend to have short data access requirements. In addition to balancing the workload among CPUs, the parallel database provides for concurrent access to data and protects data integrity.

Key elements of parallel processing:

- Speedup and Scaleup: the Goals of Parallel Processing
- Synchronization: A Critical Success Factor
- Locking
- Messaging

9.Illustrate indexing and hashing techniques with suitable examples. (Nov'2015)

In an **ordered index,** index entries are stored sorted on the search key value.

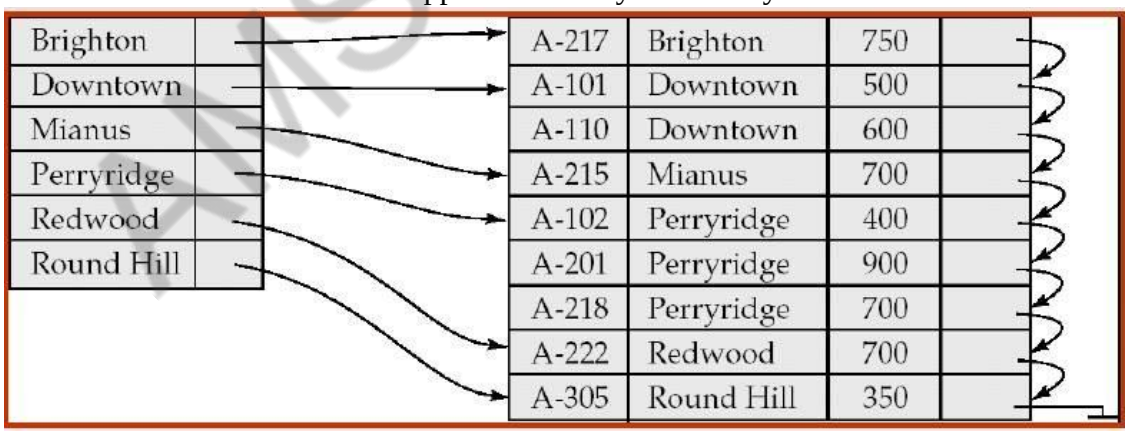
Primary index: in a sequentially ordered file, the index whose search key specifies the sequential order of the file.

Secondary index: an index whose search key specifies an order different from the sequential order of the file.

Types of Ordered Indices

- Dense index
- Sparse index

Dense index — Index record appears for every search-key value in the file.

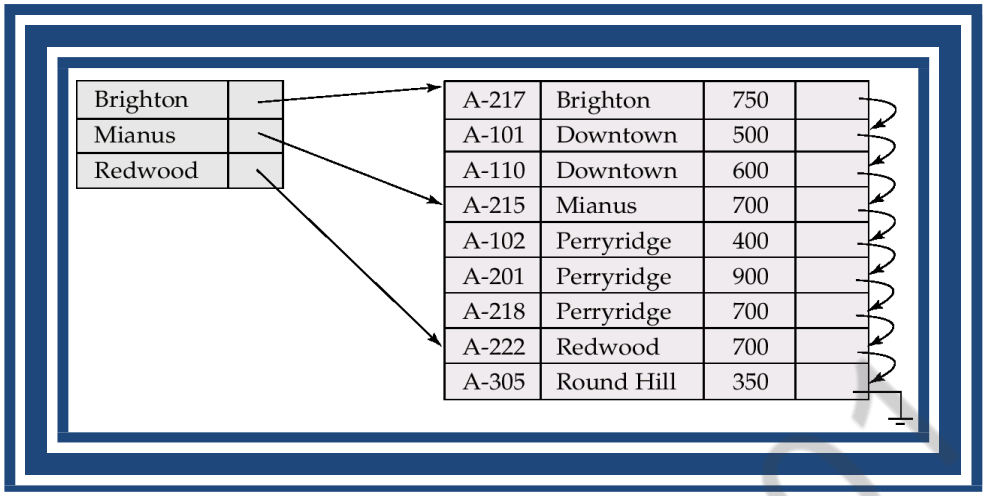


Sparse Index: contains index records for only some search-key values.

- Applicable when records are sequentially ordered on search-key

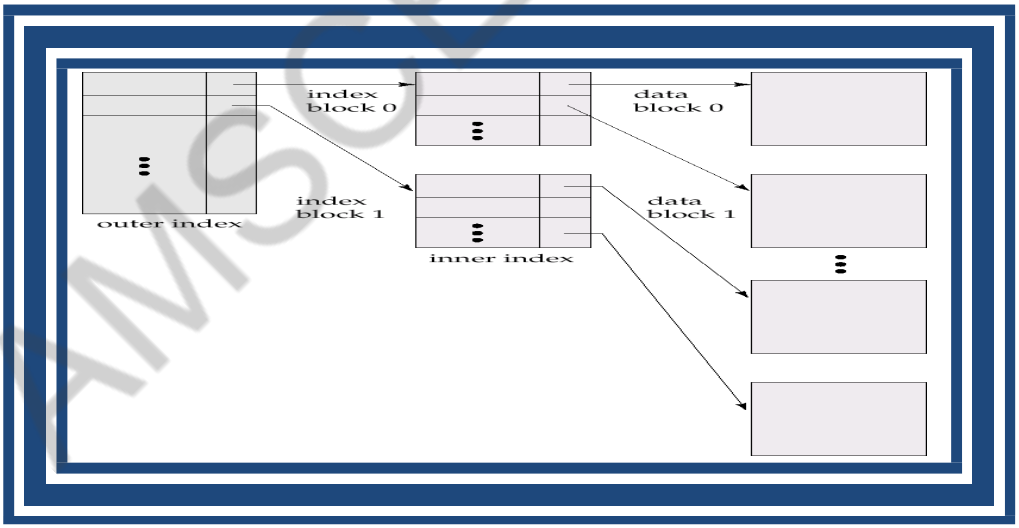
To locate a record with search-key value *K* we:

- Find index record with largest search-key value < *K*
- Search file sequentially starting at the record to which the index record points



Multilevel index

- If primary index does not fit in memory, access becomes expensive.
- To reduce number of disk accesses to index records, treat primary index kept on disk as a sequential file and construct a sparse index on it.
- outer index – a sparse index of primary index
- inner index – the primary index file
- If even outer index is too large to fit in main memory, yet another level of index can be created, and so on.



Secondary Indices

- Index record points to a bucket that contains pointers to all the actual records with that particular searchkey value.
- Secondary indices have to be dense

Hashing:

Static hashing:

A bucket is a unit of storage containing one or more records.

In a hash file organization we obtain the bucket of a record directly from its search-key value using a hash function.

Hash function h is a function from the set of all search-key values K to the set of all bucket addresses B.

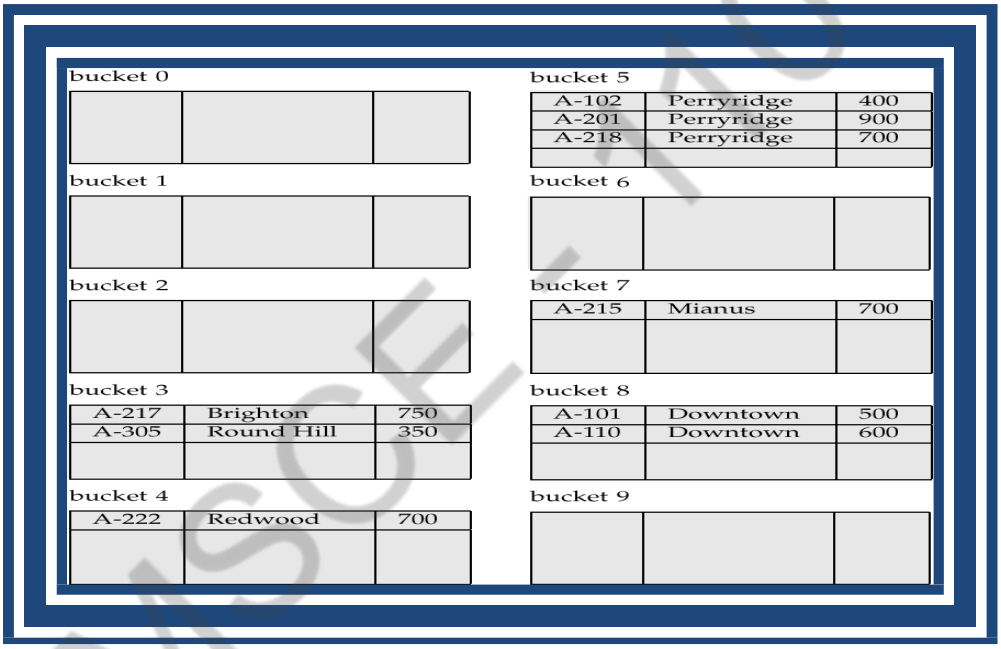
Hash function is used to locate records for access, insertion as well as deletion.

Example:

-There are 10 buckets,

-The hash function returns the sum of the binary representations of the characters modulo 10

– E.g. h(Perryridge) = 5 h(Round Hill) = 3 h(Brighton) = 3

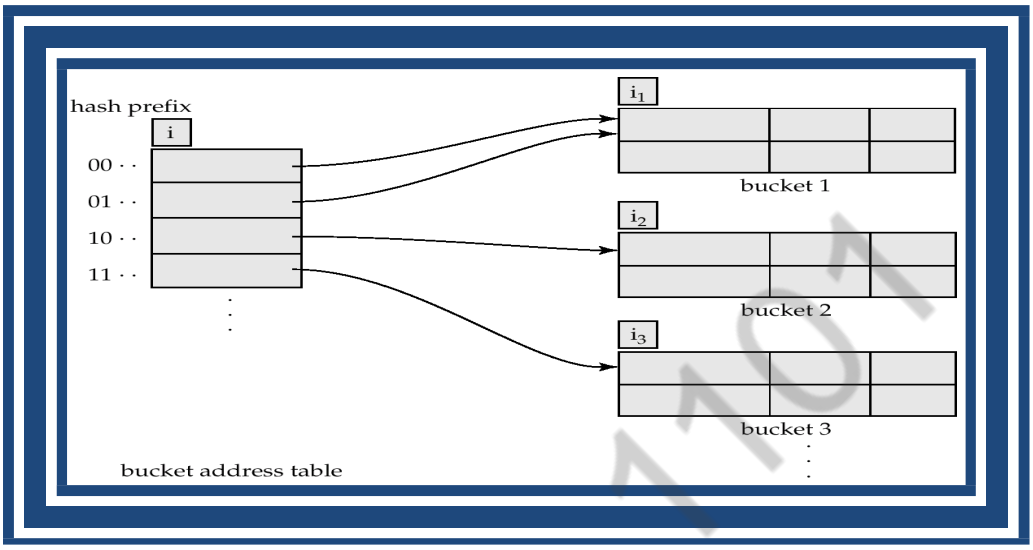


Dynamic hashing:

- Good for database that grows and shrinks in size
- Allows the hash function to be modified dynamically
- Extendable hashing – one form of dynamic hashing
  - Hash function generates values over a large range — typically b-bit integers, with b=32.
  - At any time use only a prefix of the hash function to index into a table of bucket addresses.
  - Let the length of the prefix be i bits, $0 \le i \le 32$.
  - Bucket address table size = $2^{i.}$ Initially i = 0
  - Value of i grows and shrinks as the size of the database grows and shrinks.
  - Multiple entries in the bucket address table may point to a bucket.
  - Thus, actual number of buckets is $< 2^{i}$

– The number of buckets also changes dynamically due to coalescing and splitting of buckets.

General Extendable Hash Structure



In this structure, $i_2 = i_3 = i$, whereas $i_1 = i - 1$ (see next slide for details)

Use of Extendable Hash Structure

- To locate the bucket containing search-key K_j:
  1. Compute $h(K_j) = X$
  2. Use the first i high order bits of X as a displacement into bucket address table, and follow the pointer to appropriate bucket

Updates in Extendable Hash Structure

- To insert a record with search-key value K_j
  - follow same procedure as look-up and locate the bucket, say j.
  - If there is room in the bucket j insert record in the bucket.
  - Overflow buckets used instead in some cases.
- To delete a key value,
  - locate it in its bucket and remove it.
  - The bucket itself can be removed if it becomes empty
  - Coalescing of buckets can be done

Decreasing bucket address table size is also possible

10.Explain about B^+ trees indexing concepts with an example (Nov'2014 & May 2016) (OR)

Explain the B+ tree indexes on multiple keys with a suitable example? (Dec 2017) (OR)

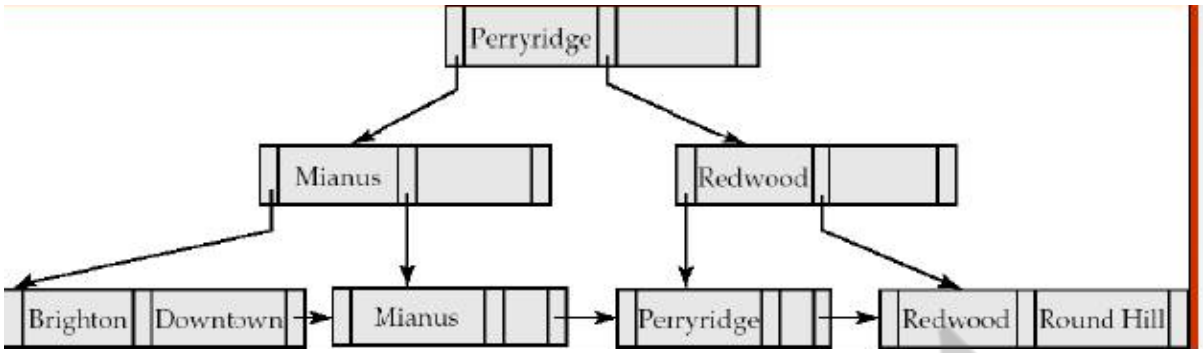
Describe the structure of B+ tree and give the algorithm for search in the B+ tree with example? (Apr 2019)

A B^+-tree is a rooted tree satisfying the following properties:

- All paths from root to leaf are of the same length
- Each node that is not a root or a leaf has between $[n/2]$ and n children.

- Special cases:
    - If the root is not a leaf, it has at least 2 children.

- If the root is a leaf, it can have between 0 and (n–1) values.



B+Tree Node Structure

-Typical node

*Ki are the search-key values

*Pi are pointers to children (for non-leaf nodes) or pointers to records or buckets of records (for leaf nodes).

-The searchkeys in a node are ordered

$K1 < K2 < K3 < . . . < Kn–1$

Properties of leaf node

- For $i = 1, 2, . . ., n–1$, pointer P_i either points to a file record with search-key value K_i, or to a bucket of pointers to file records, each record having search-key value K_i.
- P_n points to next leaf node in search-key order

Non-Leaf Nodes in B^+-Trees

Non leaf nodes form a multi-level sparse index on the leaf nodes. For a non-leaf node with m pointers:All the search-keys in the subtree to which P_1 points are less than K_1.

Disadvantage of indexed-sequential files: performance degrades as file grows, since many overflow blocks get created. Periodic reorganization of entire file is required.

Advantage of B^+-tree index files: automatically reorganizes itself with small, local, changes, in the face of insertions and deletions. Reorganization of entire file is not required to maintain performance.

Disadvantage of B^+-trees: extra insertion and deletion overhead, space overhead.

11.Explain about B trees indexing concepts with an example (Nov'2014)

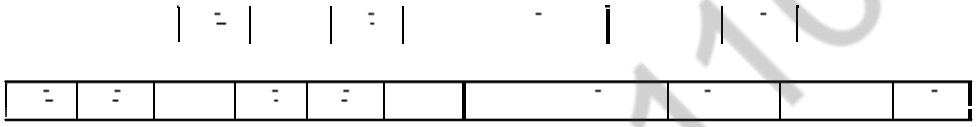
A B-tree is a tree data structure that keeps data sorted and allows searches, insertions, and deletions in logarithmic amortized time. Unlike self-balancing binary search trees, it is optimized for systems that read and write large blocks of data. It is most commonly used in database and file systems.

- Similar to B+-tree, but B-tree allows search-key values to appear only once; eliminates redundant storage of search keys.

- Search keys in nonleaf nodes appear nowhere else in the B-tree; an additional pointer field for each search key in a nonleaf node must be included.

The rules for a B tree are as follows.

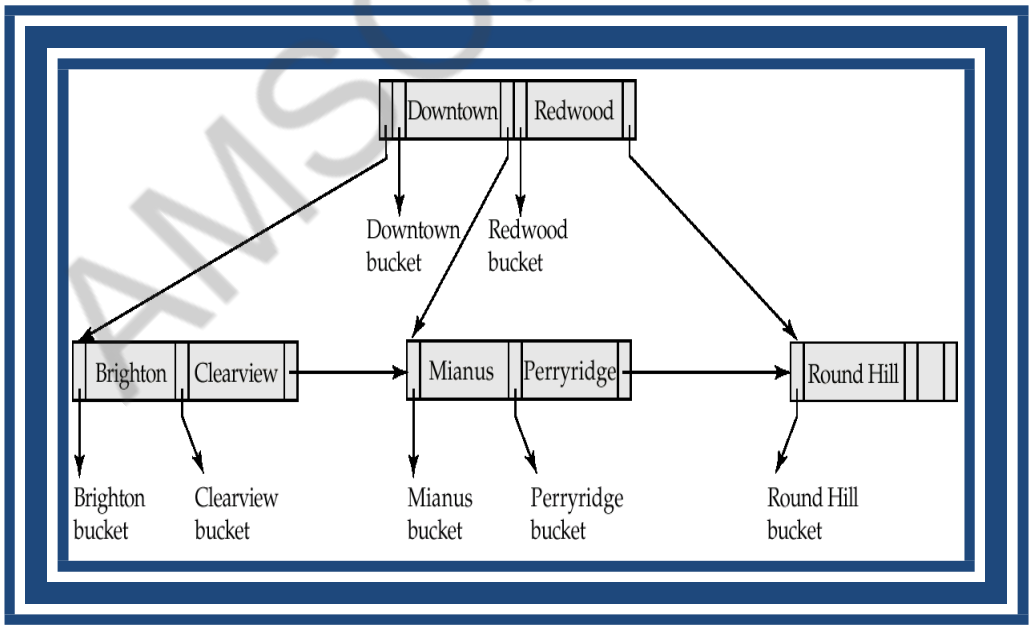
- If the root is not a leaf node, it must have at least two children.
- For a tree of order n, each node except the root and leaf nodes must have between n/2 and n pointers .
- The number of key values contained in a non leaf node is 1 less than the number of pointers.
- The tree must always be balanced ie every path from the root node to



B-tree Structure for a) Leaf node and b) Non-leaf node

Where Ki are the search-key values and Pi are pointers to children
Nonleaf node – pointers B_i are the bucket or file record pointers.

Example for B tree



Advantages of B-Tree indices:

- May use less tree nodes than a corresponding B^+-Tree.

- Sometimes possible to find search-key value before reaching leaf node.

Disadvantages of B-Tree indices:

- Only small fraction of all search-key values are found early
- Non-leaf nodes are larger, so fan-out is reduced. Thus, B-Trees typically have greater depth than corresponding B^+-Tree
- Insertion and deletion more complicated than in B^+-Trees
- Implementation is harder than B^+-Trees.

12.What is Hashing? Explain static hashing and dynamic hashing with an example? (May 2018)

(OR)

Explain the distinction between static hashing and dynamic hashing? Discuss the relative merits of each technique in database applications? (Dec 2017)

Static hashing:

A bucket is a unit of storage containing one or more records.

In a hash file organization we obtain the bucket of a record directly from its search-key value using a hash function.

Hash function h is a function from the set of all search-key values K to the set of all bucket addresses B.

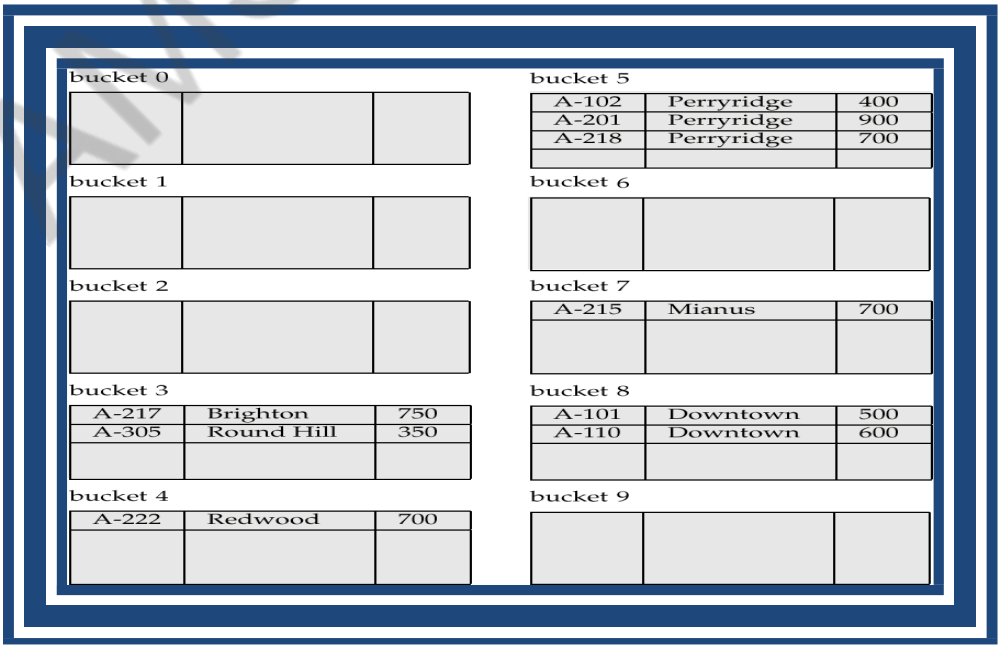
Hash function is used to locate records for access, insertion as well as deletion.

Example:

-There are 10 buckets,

-The hash function returns the sum of the binary representations of the characters modulo 10

– E.g. h(Perryridge) = 5 h(Round Hill) = 3 h(Brighton) = 3

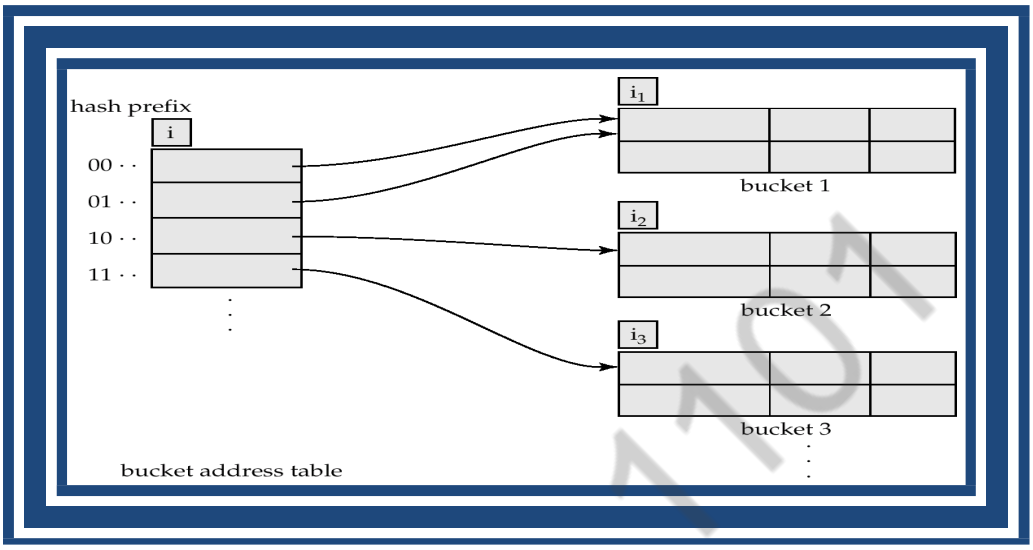


Dynamic hashing:

- Good for database that grows and shrinks in size
- Allows the hash function to be modified dynamically
- Extendable hashing – one form of dynamic hashing
  - Hash function generates values over a large range — typically b-bit integers, with b=32.
  - At any time use only a prefix of the hash function to index into a table of bucket addresses.
  - Let the length of the prefix be i bits, $0 \leq i \leq 32$.
  - Bucket address table size = 2^{i}. Initially i = 0
  - Value of i grows and shrinks as the size of the database grows and shrinks.
  - Multiple entries in the bucket address table may point to a bucket.
  - Thus, actual number of buckets is $< 2^{i}$

– The number of buckets also changes dynamically due to coalescing and splitting of buckets.

General Extendable Hash Structure



In this structure, $i_2 = i_3 = i$, whereas $i_1 = i - 1$ (see next slide for details)

Use of Extendable Hash Structure

- To locate the bucket containing search-key K_j:
  3. Compute $h(K_j) = X$
  4. Use the first i high order bits of X as a displacement into bucket address table, and follow the pointer to appropriate bucket

Updates in Extendable Hash Structure

- To insert a record with search-key value K_j
  - follow same procedure as look-up and locate the bucket, say j.
  - If there is room in the bucket j insert record in the bucket.
  - Overflow buckets used instead in some cases.
- To delete a key value,
  - locate it in its bucket and remove it.
  - The bucket itself can be removed if it becomes empty
  - Coalescing of buckets can be done

Decreasing bucket address table size is also possible

13. Why data dictionary storage is important? (Nov 2019)

A data dictionary contains metadata i.e data about the database. The data dictionary is very important as it contains information such as what is in the database, who is allowed to access it, where is the database physically stored etc. The users of the database normally don't interact with the data dictionary, it is only handled by the database administrators.

The data dictionary in general contains information about the following:

- Names of all the database tables and their schemas.
- Details about all the tables in the database, such as their owners, their security constraints, when they were created etc.
- Physical information about the tables such as where they are stored and how.
- Table constraints such as primary key attributes, foreign key information etc.
- Information about the database views that are visible.

This is a data dictionary describing a table that contains employee details.

Field Name	Data Type	Field Size for display	Description	Example
Employee Number	Integer	10	Unique ID of each employee	1645000001
Name	Text	20	Name of the employee	David Heston
Date of Birth	Date/Time	10	DOB of Employee	08/03/1995
Phone Number	Integer	10	Phone number of employee	6583648648

The different types of data dictionary are:

Active Data Dictionary

If the structure of the database or its specifications change at any point of time, it should be reflected in the data dictionary. This is the responsibility of the database management system in which the data dictionary resides.

So, the data dictionary is automatically updated by the database management system when any changes are made in the database. This is known as an active data dictionary as it is self updating.

Passive Data Dictionary

This is not as useful or easy to handle as an active data dictionary. A passive data dictionary is maintained separately to the database whose contents are stored in the dictionary. That means that if the database is modified the database dictionary is not automatically updated as in the case of Active Data Dictionary.

So, the passive data dictionary has to be manually updated to match the database. This needs careful handling or else the database and data dictionary are out of sync.

14. With simple algorithms explain the computing of Nester-loop join and Block Nested-loop join? (Nov 2019)

Computation of Joins :

When we want to join two tables, say P and Q, each tuple in P has to be compared with each tuple in Q to test if the join condition is satisfied. If the condition is satisfied, the corresponding tuples are concatenated, eliminating duplicate fields and appended to the result relation. Consequently, this is the most expensive operation.

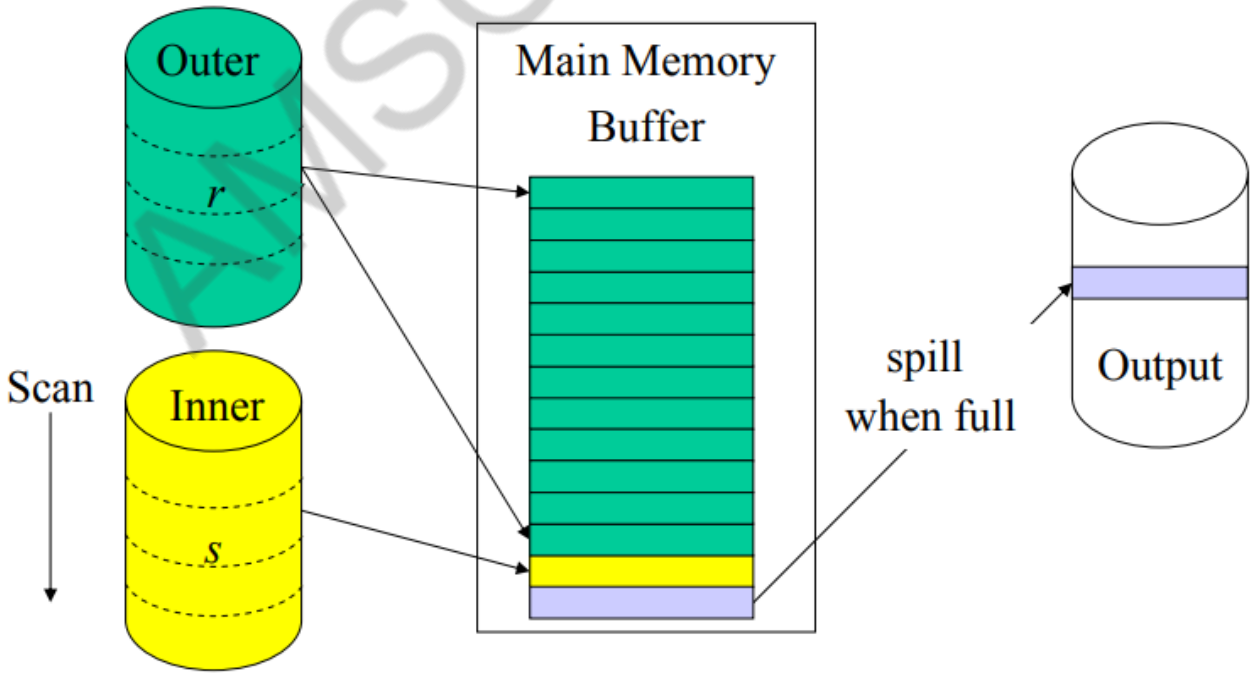
The common approaches for computing joins are −

Nested-loop Approach

This is the conventional join approach. It can be illustrated through the following pseudocode (Tables P and Q, with tuples tuple_p and tuple_q and joining attribute a) −

```
For each tuple_p in P
For each tuple_q in Q
If tuple_p.a = tuple_q.a Then
  Concatenate tuple_p and tuple_q and append to Result
End If
Next tuple_q
Next tuple-p
```

- Basically, for each block of the outer table (r), scan the entire inner table (s). „
- Requires quadratic time, O(n2) „
- Improved when buffer is used.



Example of Nested Loop Join:

$$Customer \bowtie_{C.CustomerID = CO.EmpId} CheckedOut$$

- Parameters

$r_{CheckedOut} = 40.000$ $\quad$ $r_{Customer} = 200$

$b_{CheckedOut} = 2.000$ $\quad$ $b_{Customer} = 10$

$n_B = 6$ (size of main memory buffer)

- Algorithm:

repeat: read $(n_B - 2)$ blocks from outer relation

repeat: read 1 block from inner relation

compare tuples

- Cost: $b_{outer} + (\lceil b_{outer} / (n_B - 2) \rceil) \times b_{inner}$
- *CheckedOut* as outer: $2.000 + \lceil 2.000/4 \rceil \times 10 = 7.000$
- *Customer* as outer: $10 + \lceil 10/4 \rceil \times 2.000 = 6.010$

Sort-merge Approach

In this approach, the two tables are individually sorted based upon the joining attribute and then the sorted tables are merged. External sorting techniques are adopted since the number of records is very high and cannot be accommodated in the memory. Once the individual tables are sorted, one page each of the sorted tables are brought to the memory, merged based upon the joining attribute and the joined tuples are written out.

Hash-join Approach

This approach comprises of two phases: partitioning phase and probing phase. In partitioning phase, the tables P and Q are broken into two sets of disjoint partitions. A common hash function is decided upon. This hash function is used to assign tuples to partitions. In the probing phase, tuples in a partition of P are compared with the tuples of corresponding partition of Q. If they match, then they are written out.

UNIT- 5

ADVANCED TOPICS

PART-A

1. Define Distributed Database Management Systems.(Dec 2016, May 2018)

A distributed database management system consists of loosely coupled sites(computer) that share no physical components and each site is associated with a database system.

2. What are various fragmentations? State Various fragmentations with example?(Dec 2017)

There are two types of fragmentation. They are Horizontal Fragmentation and Vertical Fragmentation.

Horizontal fragmentation :

Splits the relation by assigning each tuple of r to one or more fragments

relation r is partitioned into a number of subsets, r_1 ,r_2,…..r_n and can be reconstruct the original relation using union of all fragments, that is

$$r = r_1 \cup r_2 \cup \ldots\ldots \cup r_n$$

Vertical fragmentation :

- Splits the relation by decomposing scheme R of relation and reconstruct the original relation by using natural join of all fragments. that is

$r = r_1$ r_2 …… r_n

3. Give an example of two phase commit protocol? (Dec 2015)

Client want all or nothing transactions and Transfer either happens or nothing at all.

4. How does the concept of an object in the object oriented model differ from the concept of an entity in an entity relationship model? (Dec 2016)

- An entity is simply a collection of variables or data items.
- An object is an encapsulation of data as well as the methods(code) to operate on the data
- The data member of an object are directly visible only to its methods.
- The outside world can gain access to the object's data only by passing pre-defined messages to it, and these messages are implemented by the methods.

5. State the functions of XML Schema? (Dec2017)

The XML Schema are used to represent the structure of XML document.
The goal or purpose of XML Schema is to define the building blocks of an XML document.

These can be used as an alternative to XML DTD.

6. Define XQuery?

It is similar to SQL to database.
It is used to Query XML database.
XQuery is supported by all major databases
Main task is to get information out of XML databases.

7. Differentiate Database Management System and Information Retrieval System?

Database Management System	**Information Retrieval System**
It makes use of structured data	It makes use of unstructured data
It is schema driven	There is no fixed schema in IR system
Relational database model is used	Free Form query model is used
Result based on exact matching	Result based on appropriate matching

8. What are the advantages of distributed databases? (May 2008)

- There is fast data processing as several sites participate in request processing.
- Reliability and Availability of the system is high
- It Possess reduced operating cost
- It is easier to expand the system by adding more sites
- It has improved sharing ability and local autonomy

9. List out the reasons for development of distributed databases? (May 2006)

Following are the reasons for development of distributed databases-

- To control the data present at geographically different sites
- To obtain highly available and reliable data processing systems.

10. What are two approaches to store a relation in the distributed databases? (May 2004)

Replication: System maintains multiple copies of data, stored in different sites, for fast retrieval and fault tolerance.
Fragmentation: Relation is partitioned into several fragments stored in distinct sites.

11. Differentiate between Homogeneous and Heterogeneous Schema?

Homogeneous Schema	Heterogeneous Schema
It shares global schema	It has different schemas
Each site provides part of its autonomy in terms of right to change or software	Each site maintains its own right to change the schema or software.
Due to same schema, there is no problem in query processing.	Due to different schemas, there are lot of problems in query processing.

12.What are the advantages of fragmentation?

- It allows parallel processing on fragments of a relation
- It allows a relation to be split so that tuples are located where they are most frequently accessed.

13.What is meant by object-oriented data model?

The object-oriented paradigm is based on encapsulation of data and code related to an object in to a single unit, whose contents are not visible to the outside world.

14.What is the major advantage of object-oriented programming paradigm?

The ability to modify the definition of an object without affecting the rest of the system is the major advantage of object-oriented programming paradigm.

15.What are the methods used in object-oriented programming paradigm?

*read-only
*update

16.What is the main difference between read-only and update methods?

A read-only method does not affect the values of a variable in an object, whereas an update method may change the values of the variables.

17.What is the use of keyword ISA?

The use of keyword ISA is to indicate that a class is a specialization of another class.

18.Differentiate sub-class and super-class?

The specialization of a class is called subclasses.eg: employee is a subclass of person and teller is a subclass of employee.Conversely, employee is a super class of teller, and person is a super class of employee.

19.What is substitutability?

Any method of a class-say A can equally well be invoked with any object belonging to any subclasses B of A. This characteristic leads to code reuse, since the messages, methods, and functions do not have to be written again for objects of class B.

20.What is multiple inheritance?

Multiple inheritance permits a class to inherit variables and methods from multiple super

classes.

21.What is DAG?

The class-subclass relationship is represented by a directed acyclic graph.eg: employees can be temporary or permanenet.we may create subclasses temporary and permanenet, of the class employee.

22.What is disadvantage of multiple inheritance?

There is potential ambiguity if the same variable or method can be inherited from more than one superclass.eg: student class may have a variable dept identifying a student's department, and the teacher class may correspondingly have a variable dept identifying a teacher's department.

23.What is object identity?

An object retains its identity even if some or all the values of variables or definitions of methods change overtime.

24.What are the several forms of identity?

*Value
*Name
*Built-in

25.What is a value?

A data value is used for identity. This form of identity is used in relational systems.eg: The primary key value of a tuple identifies the tuple.

26.What is a Name?

A user-supplied name is used for identity. This form of identity is used for files in file systems. The user gives each file a name that uniquely identifies it, regardless of its contents.

27.What is a Built-in?

A notation of identity is built-into the data model or programming language and no user-supplied identifier is required. This form of identity is used in objectoriented systems.

28.What is meant by object identifiers?

Object-oriented systems use an object identifier to identify objects. Object identifiers are unique: that is each object has a single identifier, and no two objects have the same identifier.

29.What are composite objects?

Objects that contain other objects are called complex objects or composite objects.

30.What is object containment?

References between objects can be used to model different real-world concepts.

31.Why containment is important in oosystems?

Containment is an important concept in oosystems because it allows different users

to view data at different granularities.

32.Define object-relational systems?

Systems that provide object-oriented extensions to relational systems are called object-relational systems.

33.How persistent programming languages differ from traditional programming languages?

Database languages differ from traditional programming languages in that they directly

manipulate data that are persistent-that is, data that continue to exist even after the program terminated. Relation in a database and tuples in a relation are examples of persistent data. In contrast, the only persistent data that traditional programming languages directly manipulate are files.

34.What is nested relational model?

The nested relational model is an extension of relational model in which domains may be either atomic or relation valued.

35.List some instances of collection types?

*sets
*arrays
*multisets

36.How to create values of structured type?

Constructor functions are used to create values of structured types. A function with the same name as a structured type is a constructor function for the structured type.

37.Define XML Database.

An XML database is a data persistence software system that allows data to be stored in XML format. These data can then be queried, exported and serialized into the desired format. XML databases are usually associated with document-oriented databases.

38.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.

39.What is a heterogeneous distributed database?

In a heterogeneous distributed database, different sites may use different schemas, and different dbms s/w.The sites may not be aware of one another, and they may provide only limited facilities for cooperation in transaction processing.

40.Define statistical database.

A statistical database is a database used for statistical analysis purposes. It is an OLAP (online analytical processing), instead of OLTP (online transaction processing) system. Modern decision, and classical statistical databases are often closer to the relational model than the multidimensional model commonly used in OLAP systems today.

41.What is data acquisition?

Every data warehouse has a source for acquiring data. The vast majority of the data warehouse data is derived from the organizational operational data. Desired data is extracted, filtered, translated and integrated into the storage environment.

42.What are the two approaches to store relations in distributed database?

*Replication *Fragmentation

43.Define Database privilege?

It is a right to execute a particular type of SQL statement or to access another users object

Eg. Right to connect to database.
Right to create a table
Right to select rows from another users table
Right to execute another users stored procedure.

44.What is threat?

A threat is any situation, event or personnel that will adversely affect the database security and smooth and efficient functioning of organization.

A threat may be caused by a situation or event involving a person, action or circumstance that is likely to bring harm to organization

45.Define Database security.

Database security concerns the use of a broad range of information security controls to protect databases (potentially including the data, the database applications or stored functions, the database systems, the database servers and the associated network links) against compromises of their confidentiality, integrity and availability.

46.Mention the users of database.

Database administrators
Security officers
Network administrators
Application developers
Application administrators
DB users

47.What does database security refer to?

Database security refers to the protection from unauthorized access and malicious destruction or alteration.

48.List some security violations (or) name any forms of malicious acess.

Unauthorized Reading of data
Unauthorized modification of data
Unauthorized destruction of data.

49.List the types of authorization.

Read authorization
Write authorization
Update authorization
Drop authorization

50.What is authorization graph?

Passing of authorization from one user to another can be represented by an authorization graph.

51.List out various user authorization to modify the database *Sch*ema.

Index Authorization
Resource authorization
Alteration authorization
Drop authorization

52.What are audit trails?

An audit trail is a log of all changes to the database along with information such as which user performed the change and when the change was performed.

53.What is Benefits of Database Encryption?

Ensure guaranteed access to encrypted data by authorized users by automating storage and back-up for mission critical master encryption keys.Simplify data privacy compliance obligations and reporting activities through the use of a security-certified encryption and key management to enforce critical best practices and other standards of due care.

54.Name the various privileges in SQL?

Delete
Select
Insert
update

55.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.

56.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 loopholes.

57.What does authentication refer? List some authentication techniques.

Authentication refers to the task of verifying the identity of a person.

authentication techniques:

Challenge response scheme

Digital Signatures
Non repudiation

58.Define Crawling and indexing the web. (Nov' 2014)

Web Crawling is the process of search engines combing through web pages in order to properly index them. These "web crawlers" systematically crawl pages and look at the keywords contained on the page, the kind of content, all the links on the page, and then returns that information to the search engine's server for indexing. Then they follow all the hyperlinks on the website to get to other websites. When a search engine user enters a query, the search engine will go to its index and return the most relevant search results based on the keywords in the search term. Web crawling is an automated process and provides quick, up to date data.

59.Define Relevance Ranking. (Nov'2014)

A system in which the search engine tries to determine the theme of a site that a link is coming from.

60. Write about the four types (Star, Snowflake, Galaxy and Fact Constellation) of Data warehouse schemas. (May'2015)

1. Star schema: The **star schema** architecture is the simplest **data warehouse schema**. It is called a **star schema** because the diagram resembles a **star**, with points radiating from a center. The center of the **star** consists of fact table and the points of the **star** are the dimension tables.

2. Fact constellation: is a measure of online analytical processing, which a collection of multiple fact tables is sharing dimension tables, viewed as a collection of stars. This is an improvement over **Star schema**.

3. Snowflake schema: In computing, a **snowflake schema** is a logical arrangement of tables in a multidimensional database such that the entity relationship diagram resembles a**snowflake** shape. The **snowflake schema** is represented by centralized fact tables which are connected to multiple dimensions..

4. Galaxy Schema : It is the combination of both star schema and snowflake schema**.**

61. Define Threats and Risks (May 2015, Dec 2017)

Threats: Accidental loss and corruption, and from deliberate unauthorized attempts to access or alter that data. Loss of integrity, Loss of availability and Loss of confidentiality.

Risks: Unauthorized or unintended activity or misuse by authorized database users, database administrators, or network/systems managers, or by unauthorized users or hackers.

53. Define Association Rule Mining. (May 2015)

Association rule mining is discovering frequent patterns, associations and correlations among items which are meaningful to the users and can generate strong rules on the basis of these frequent patterns, which helps in decision support system.

62. List the types of privileges used in database access control. (Nov'2015)

Types of privileges used in database access control are Read, Write and Update.

63. Can we have more than one constructor in a class? If yes, explain the need for such a situation. (Nov/Dec 2015)

Yes, default constructor and constructor with parameter.

64. Explain Data Classification. (May/June 2016)

Classification is a data mining function that assigns items in a collection to target categories or classes. The goal of classification is to accurately predict the target class for each case in the data. For example, a classification model could be used to identify loan applicants as low, medium, or high credit risks.

65. Mention the Advantages of Data Warehousing. (May 2016)

Data warehouses tend to have a very high query success as they have complete control over the four main areas of data management systems.

a. Clean data
b. Indexes: multiple types
c. Query processing: multiple options
d. Security: data and access
e. Easy report creation.
f. Enhanced access to data and information.

66. Mention the Disadvantages of Data Warehousing.

There are considerable disadvantages involved in moving data from multiple, often highly disparate, data sources to one data warehouse that translate into long implementation time, high cost, lack of flexibility, dated information, and limited capabilities.

a. Preparation may be time consuming.
b. Compatibility with existing systems.
c. Security issues.
d. Long initial implementation time and associated high cost
e. Limited flexibility of use and types of users
f. Difficult to accommodate changes in data types and ranges, data source schema, indexes and queries

67. Mention the Disadvantages of Data Warehousing.

A **distributed database** (DDB) is a collection of multiple, logically interrelated **databases distributed**over a computer network. A **distributed database management system** (D–**DBMS**) is the software that manages the DDB and provides an access mechanism that makes this **distribution** transparent to the users.

68. How does the concept of an object in object oriented model differ from ER model in differ from the concept of an entity relationship model. (Nov'2016)

The object-oriented paradigm is based on encapsulation of data and code related to an object in to a single unit, whose contents are not visible to the outside world. The main terms are i) class ii) object iii) association and iv) attributes.

ERD stands for Entity Relationship Diagram. It works as an important component of a conceptual data model. ERD is often used to graphically represent the logical structure of a database. The main terms are i) entity ii) instance of an entity iii) relationship and iv) attributes.

69.How are transactions performed in object oriented database? (Nov 2018)

Database management systems must support atomic transactions. Object-oriented databases are geared toward engineering and design applications. Nested transactions provide a more direct support of project development for these applications.

70.How spatial databases are more helpful then active databases? (Nov 2018)

A spatial database is a database that is optimized for storing and querying data that represents objects defined in a geometric space. Most spatial databases allow the representation of simple geometric objects. Such databases can be useful for websites that wish to identify the locations of their visitors for customization.

The spatial database has proper ways to manage and store spatial objects and spatial relationships. Therefore, access a spatial database to process a spatial query is more efficient.

71. Mention few features of Multimedia Databases? (May 2019)

A Multimedia system has four basic characteristics:

- Multimedia systems must be computer controlled.
- Multimedia systems are integrated.
- The information they handle must be represented digitally.
- The interface to the final presentation of media is usually interactive.

72. Compare sequential access devices versus random access devices with an example? (May 2019)

Sequential Access to a data file means that the computer system reads or writes information to the file sequentially, starting from the beginning of the file and proceeding step by step. **Random Access** to a file means that the computer system can read or write information anywhere in the data file. This type of operation is also called "Direct Access".

Sequential drive stores files and data in a specific order, while a random access drive puts them all over the place.

Example:

The old fashioned tape drive is a sequential drive. Although tape drives are no longer used in modern PCs, some companies still use them to create durable backup archives.

Meanwhile, a modern disk drive can be programmed to store data either sequentially or through random access. CDs and DVDs can also use both methods. A music CD, for example, is sequential, but one with a database on it might use random access in order to fit more data onto the disk.

73. List information types of documents necessary for relevance ranking of documents in IR? (Nov 2019)

Relevant document means a document that— Based on 8 documents 8. relevant document means any invoice, account, drawing, plan, technical specification or other document relating to the approved operation.

74.What one could understand from allocation schema? (Nov 2019)

The allocation schema is a description of where the data(fragments)are to be located, taking account of any replication.

PART –B

1.Briefly explain about Two phase commit and three phase commit protocols. (Nov' 2014, May 2015 & May 2016)

(OR)

Explain two phase commit protocol with an example?(Nov 2017)

Two phase commit protocol

• Assumes fail-stop model – failed sites simply stop working, and do not cause any other harm,
such as sending incorrect messages to other sites.
• Execution of the protocol is initiated by the coordinator after the last step of the transaction has been reached.
• The protocol involves all the local sites at which the transaction executed
• Let *T* be a transaction initiated at site *Si*, and let the transaction coordinator at *Si* be *Ci*.

Phase 1: Obtaining a Decision (prepare)

• Coordinator asks all participants to *prepare* to commit transaction *Ti*.
  – Ci adds the records <prepare *T*> to the log and forces log to stable storage
  – sends prepare *T* messages to all sites at which *T* executed
• Upon receiving message, transaction manager at site determines if it can commit the transaction
  – if not, add a record <no *T*> to the log and send abort *T* message to *Ci*
  – if the transaction can be committed, then:
  – add the record <ready *T*> to the log
  – force *all records* for *T* to stable storage
  – send ready *T* message to C*i*

Phase 2: Recording the Decision (commit)

• *T* can be committed of *Ci* received a ready *T* message from all the participating sites: otherwise *T* must be aborted.
• Coordinator adds a decision record, <commit *T*> or <abort *T*>, to the log and forces record onto stable storage. Once the record stable storage it is irrevocable (even if failures occur)
• Coordinator sends a message to each participant informing it of the decision (commit or

abort)
• Participants take appropriate action locally.

Possible Failures

Site Failure
Coordinator Failure
Network Partition

Three phase commit protocols.

Assumptions:

– **No** network partitioning
– **At** any point, at least one site must be up.

– **At** most *K* sites (participants as well as coordinator) can fail

Phase 1: Obtaining Preliminary Decision: Identical to 2PC Phase 1.

– **Every** site is ready to commit if instructed to do so

– Under 2PC each site is obligated to wait for decision from coordinator

– Under 3PC, knowledge of pre-commit decision can be used to commit despite coordinator failure

Phase 2: Recording the Preliminary Decision

Coordinator adds a decision record (<**abort** *T*> or <**precommit** *T*>) in its log and forces record to stable storage. Coordinator sends a message to each participant informing it of the decision.

Participant records decision in its log

– If abort decision reached then participant aborts locally

– If pre-commit decision reached then participant replies with <**acknowledge** T>

Phase 3: Recording Decision in the Database

Executed only if decision in phase 2 was to pre commit

Coordinator collects acknowledgments. It sends <**commit** *T*> message to the participants as soon as it receives *K* acknowledgments.

Coordinator adds the record <**commit** *T*> in its log and forces record to stable storage.

Coordinator sends a message to each participant to <**commit** *T*>.

Participants take appropriate action locally

Under 3PC, knowledge of pre-commit decision can be used to commit despite coordinator failure

- Avoids blocking problem as long as < K sites fail

Drawbacks:

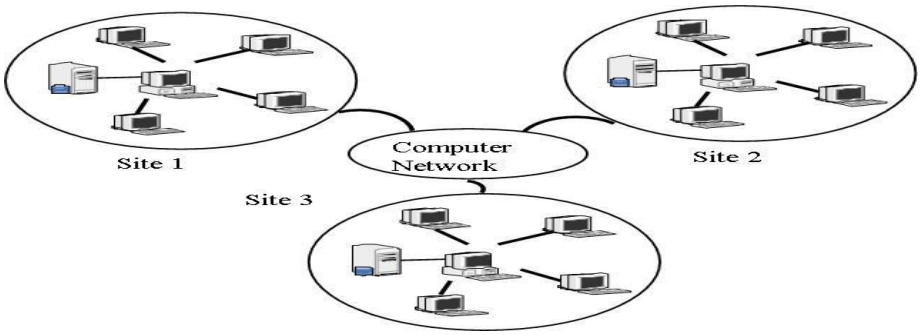
- higher overheads
- Assumptions may not be satisfied in practice.

2.Discuss in detail about Distributed Databases? (May 2019, Nov 2019)
(OR)
Explain with diagrammatic illustration the architecture of distributed database management system? (May 2018)

In distributed database system data reside in several location where as centralized database system the data reside in single location

Classification

- Homogenous distributed DB
- Heterogeneous distributed DB



- Homogenous distributed DB
  - All sites have identical database management software, are aware of one another.
  - Agree to cooperate in processing users' request.
- Heterogeneous distributed DB
  - different sites may use different schemas and different DBMS software.
  - The sites may not be aware of one another
  - Provide only limited facilities for cooperation in transaction processing.

- Consider a relation r, there are two approaches to store this relation in the distributed DB.
  - Replication
  - Fragmentation
- Replication
  - The system maintains several identical replicas(copies) of the relation at different site.
  - Full replication- copy is stored in every site in the system.
- Advantages and disadvantages
  - Availability
  - Increased parallelism

  Increased overhead update
- Fragmentation
  - The system partitions the relation into several fragment and stores each fragment at different sites
  - Two approaches
    - Horizontal fragmentation
    - Vertical fragmentation

Horizontal fragmentation

Splits the relation by assigning each tuple of r to one or more fragments

relation r is partitioned into a number of subsets, r_1 ,r_2,…..r_n and can be reconstruct the original relation using union of all fragments, that is

$$r = r_1 \cup r_2 \cup \ldots\ldots \cup r_n$$

- Vertical fragmentation
  - Splits the relation by decomposing scheme R of relation and reconstruct the original relation by using natural join of all fragments. that is

$$r = r_1 \quad r_2 \quad \ldots\ldots \quad r_n$$

3.Write short notes on Spatial Databases. (Nov'2014, Nov'2015 , Nov'2016 &

Nov 2017 & May 2019)

Spatial databases:

Spatial databases that keeps track of objects in an multidimensional space.

eg:- * Cartographic databases

* Meteorological databases

Cartographic databases:

That stores maps include two dimensional spatial descriptions of their objects from countries and states to rivers,cities,roads,seas and so on.

Applications of these databases:environmental,emergency and battle management.

Metrological databases:

For weather information, are three dimentional,since temperatures and other meteorological information are related to three dimensional spatial points.

Spatial characteristics:

The two dimensional geometric concepts such as pointers ,lines and lines segments,circles,polygon,arcs are used to spatial characteristics of objects.

Spatial operations:

Spatial operations are needed to operate on objects.

eg: To compute the distance between two objects.

Spatial boolean conditions:

To check whether two objects spatially overlap.

Emergergency management applications:

Static spatial characteristics:

Some of these objects generally have static spatial charatceristics,such as streets and highway, water pumps, police stations and hospitals.

Dynamic spatial characteristics that change over time, such as police vehicles, ambulances or fire trucks.

Categories of spatial queries:

1. Range query
2. Nearest neighbor query
3. Spatial joins or overlays

1. Range query:

Finds the objects of a particular type that are within a given spatial area or within a particular distance from a given location.

eg:find all schools within the Bombay city area.

2.Nearest neighbor query or adjacency:

It finds an object of a particular type that is closer to a given location.

eg:find the school that is closest to your house.

3.Spatial joins or overlays:

The objects are two types based on some spatial condition, such as the objects intersecting or overlapping spatially or being within a certain distance of one another .

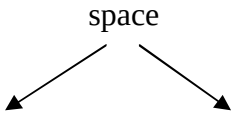
eg:- Find aall cities that fall on a major highway or finds all homes that are within two miles of a lake.

Spatial storage structures:

1. R-trees
2. quad trees

1.R-tree:

R-trees can easily answer queries, such as find all objects in a given area by limiting the tree search to those subtrees whose rectangles intersect with the area given in the query.



subspace 1 subspace 2

area area

subspace :- rectangular sub spaced with needed objects.

2.Quad tree:

Quad tree generally divide each space subspace into equally sized areas .Each subspaces used to identify the positions of various objects.

4.Compare and contrast object oriented and XML databases? (Dec 2018)

Object oriented Databases:

An object-oriented database is a database that subscribes to a model with information represented by objects. Object-oriented databases are a niche offering in the relational database management system (RDBMS) field and are not as successful or well-known as mainstream database engines.

As the name implies, the main feature of object-oriented databases is allowing the definition of objects, which are different from normal database objects. Objects, in an object-oriented database, reference the ability to develop a product, then define and name it. The object can then be referenced, or called later, as a unit without having to go into its complexities. This is very similar to objects used in object-oriented programming.

A real-life parallel to objects is a car engine. It is composed of several parts: the main cylinder block, the exhaust system, intake manifold and so on. Each of these is a standalone component; but when machined and bolted into one object, they are now collectively referred to as an engine. Similarly, when programming one can define several components, such as a vertical line intersecting a perpendicular horizontal line while both lines have a graded measurement. This object can then be collectively labeled a graph. When utilizing the ability to plot components, there is no need to first define a graph; but rather the instance of the created graph can be called.

Examples of object-oriented database engines include db4o, Smalltalk and Cache.

XML Databases:

An XML database is a database that stores data in XML format. This type of database is suited for businesses with data in XML format and for situations where XML storage is a practical way to archive data, metadata and other digital resources.

This data can be queried, transformed, exported and returned to a calling system. XML databases are a flavor of document-oriented databases which are in turn a category of NoSQL database.

There are a number of reasons to directly specify data in XML or other document formats such as JSON. For XML in particular, they include:

- An enterprise may have a lot of XML in an existing standard format

- Data may need to be exposed or ingested as XML, so using another format such as relational forces double-modeling of the data
- XML is very well suited to sparse data, deeply nested data and mixed content (such as text with embedded markup tags)
- XML is human readable whereas relational tables require expertise to access
- Metadata is often available as XML
- Semantic web data is available as XML

XML enabled databases typically offer one or more of the following approaches to storing XML within the traditional relational structure:

1. XML is stored into a CLOB (Character large object)
2. XML is `shredded` into a series of Tables based on a Schema

5.Explain about Document Type Definition and XML Schema? (Dec 2016 & Dec 2018)

(OR)

Brief on the methods to store XML Documents?(NOV 2019)

DTD:

The XML Document Type Declaration, commonly known as DTD, is a way to describe XML language precisely. DTDs check vocabulary and validity of the structure of XML documents against grammatical rules of appropriate XML language.

An XML DTD can be either specified inside the document, or it can be kept in a separate document and then liked separately.

Syntax:

Basic syntax of a DTD is as follows −

```
<!DOCTYPE element DTD identifier
[
  declaration1
  declaration2
  ........
]>
```

In the above syntax,

- The DTD starts with <!DOCTYPE delimiter.
- An element tells the parser to parse the document from the specified root element.
- DTD identifier is an identifier for the document type definition, which may be the path to a file on the system or URL to a file on the internet. If the DTD is pointing to external path, it is called External Subset.
- The square brackets [ ] enclose an optional list of entity declarations called Internal Subset.

Internal DTD:

A DTD is referred to as an internal DTD if elements are declared within the XML files. To refer it as internal DTD, standalone attribute in XML declaration must be set to yes. This means, the declaration works independent of an external source.

Syntax:

Following is the syntax of internal DTD −

<!DOCTYPE root-element [element-declarations]>

where root-element is the name of root element and element-declarations is where you declare the elements.

Example:

Following is a simple example of internal DTD −

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "yes" ?>
<!DOCTYPE address [
  <!ELEMENT address (name,company,phone)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT company (#PCDATA)>
  <!ELEMENT phone (#PCDATA)>
]>

<address>
  <name>Tanmay Patil</name>
  <company>TutorialsPoint</company>
  <phone>(011) 123-4567</phone>
</address>
```

Let us go through the above code −

Start Declaration − Begin the XML declaration with the following statement.

<?xml version = "1.0" encoding = "UTF-8" standalone = "yes" ?>

DTD − Immediately after the XML header, the document type declaration follows, commonly referred to as the DOCTYPE −

<!DOCTYPE address [

The DOCTYPE declaration has an exclamation mark (!) at the start of the element name. The DOCTYPE informs the parser that a DTD is associated with this XML document.

DTD Body − The DOCTYPE declaration is followed by body of the DTD, where you declare elements, attributes, entities, and notations.

```
<!ELEMENT address (name,company,phone)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT phone_no (#PCDATA)>
```

Several elements are declared here that make up the vocabulary of the <name> document. <!ELEMENT name (#PCDATA)> defines the element name to be of type "#PCDATA". Here #PCDATA means parse-able text data.

End Declaration − Finally, the declaration section of the DTD is closed using a closing bracket and a closing angle bracket (]>). This effectively ends the definition, and thereafter, the XML document follows immediately.

Rules:

- The document type declaration must appear at the start of the document (preceded only by the XML header) − it is not permitted anywhere else within the document.
- Similar to the DOCTYPE declaration, the element declarations must start with an exclamation mark.
- The Name in the document type declaration must match the element type of the root element.

External DTD:

In external DTD elements are declared outside the XML file. They are accessed by specifying the system attributes which may be either the legal .dtd file or a valid URL. To refer it as external DTD, standalone attribute in the XML declaration must be set as no. This means, declaration includes information from the external source.

Syntax:

Following is the syntax for external DTD −

<!DOCTYPE root-element SYSTEM "file-name">

where file-name is the file with .dtd extension.

Example:

The following example shows external DTD usage −

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "no" ?>
<!DOCTYPE address SYSTEM "address.dtd">
<address>
   <name>Tanmay Patil</name>
   <company>TutorialsPoint</company>
   <phone>(011) 123-4567</phone>
</address>
```

The content of the DTD file address.dtd is as shown −

```
<!ELEMENT address (name,company,phone)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
```

Types

You can refer to an external DTD by using either system identifiers or public identifiers.

System Identifiers

A system identifier enables you to specify the location of an external file containing DTD declarations. Syntax is as follows −

<!DOCTYPE name SYSTEM "address.dtd" [...]>

As you can see, it contains keyword SYSTEM and a URI reference pointing to the location of the document.

Public Identifiers:

Public identifiers provide a mechanism to locate DTD resources and is written as follows −

<!DOCTYPE name PUBLIC "-//Beginning XML//DTD Address Example//EN">

As you can see, it begins with keyword PUBLIC, followed by a specialized identifier. Public identifiers are used to identify an entry in a catalog. Public identifiers can follow any format, however, a commonly used format is called Formal Public Identifiers, or FPIs.

XML SCHEMA

XML Schema is commonly known as XML Schema Definition (XSD). It is used to describe and validate the structure and the content of XML data. XML schema defines the elements, attributes and data types. Schema element supports Namespaces. It is similar to a database schema that describes the data in a database.

Syntax:

We need to declare a schema in your XML document as follows −

Example:

The following example shows how to use schema −

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">
  <xs:element name = "contact">
    <xs:complexType>
      <xs:sequence>
        <xs:element name = "name" type = "xs:string" />
        <xs:element name = "company" type = "xs:string" />
        <xs:element name = "phone" type = "xs:int" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

The basic idea behind XML Schemas is that they describe the legitimate format that an XML document can take.

Elements:

XML - Elements are the building blocks of XML document. An element can be defined within an XSD as follows −

```
<xs:element name = "x" type = "y"/>
```

Definition Types

You can define XML schema elements in the following ways −

Simple Type:

Simple type element is used only in the context of the text. Some of the predefined simple types are: xs:integer, xs:boolean, xs:string, xs:date. For example −

```
<xs:element name = "phone_number" type = "xs:int" />
```

Complex Type:

A complex type is a container for other element definitions. This allows you to specify which child elements an element can contain and to provide some structure within your XML documents. For example −

```
<xs:element name = "Address">
  <xs:complexType>
    <xs:sequence>
      <xs:element name = "name" type = "xs:string" />
      <xs:element name = "company" type = "xs:string" />
      <xs:element name = "phone" type = "xs:int" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

In the above example, Address element consists of child elements. This is a container for other <xs:element> definitions, that allows to build a simple hierarchy of elements in the XML document.

Global Types:

With the global type, you can define a single type in your document, which can be used by all other references. For example, suppose you want to generalize the person and company for different addresses of the company. In such case, you can define a general type as follows −

```
<xs:element name = "AddressType">
  <xs:complexType>
    <xs:sequence>
      <xs:element name = "name" type = "xs:string" />
      <xs:element name = "company" type = "xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Now let us use this type in our example as follows −

```
<xs:element name = "Address1">
  <xs:complexType>
    <xs:sequence>
      <xs:element name = "address" type = "AddressType" />
      <xs:element name = "phone1" type = "xs:int" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name = "Address2">
  <xs:complexType>
    <xs:sequence>
      <xs:element name = "address" type = "AddressType" />
      <xs:element name = "phone2" type = "xs:int" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Instead of having to define the name and the company twice (once for Address1 and once for Address2), we now have a single definition. This makes maintenance simpler, i.e., if you decide to add "Postcode" elements to the address, you need to add them at just one place.

Attributes:

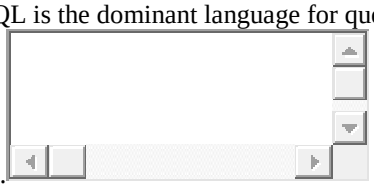
Attributes in XSD provide extra information within an element. Attributes have name and type property as shown below −

```
<xs:attribute name = "x" type = "y"/>
```

6.Give XML representation of Bank Management System? (Dec 2018)

Compared to storage of data in a database, the XML representation may be inefficient, since tag names are repeated throughout the document. However, in spite of this disadvantage, an XML representation has significant advantages when it is used to exchange data, for example, as part of a message:

- First, the presence of the tags makes the message self-documenting; that is, a schema need not be consulted to understand the meaning of the text. We can readily read the fragment above, for example.
- Second, the format of the document is not rigid. For example, if some sender adds additional information, such as a tag last-accessed noting the last date on which an account was accessed, the recipient of the XML data may simply ignore the tag. The ability to recognize and ignore unexpected tags allows the format of the data to evolve over time, without invalidating existing applications.
- Finally, since the XML format is widely accepted, a wide variety of tools are available to assist in its processing, including browser software and database tools.

Just as SQL is the dominant language for querying relational data, XML is becoming the dominant format for data exchange. 

```
<bank>
<account>
<account-number> A-101 </account-number>
<branch-name> Downtown </branch-name>
<balance> 500 </balance>
</account>
<account>
<account-number> A-102 </account-number>
<branch-name> Perryridge </branch-name>
<balance> 400 </balance>
</account>
<account>
<account-number> A-201 </account-number>
<branch-name> Brighton </branch-name>
<balance> 900 </balance>
</account>
<customer>
<customer-name> Johnson </customer-name>
<customer-street> Alma </customer-street>
<customer-city> Palo Alto </customer-city>
</customer>
<customer>
<customer-name> Hayes </customer-name>
<customer-street> Main </customer-street>
<customer-city> Harrison </customer-city>
</customer>
<depositor>
<account-number> A-101 </account-number>
<customer-name> Johnson </customer-name>
</depositor>
<depositor>
<account-number> A-201 </account-number>
<customer-name> Johnson </customer-name>
</depositor>
<depositor>
<account-number> A-102 </account-number>
<customer-name> Hayes </customer-name>
</depositor>
</bank>
```

7.Explain the necessary characteristics a system must satisfy to be considered as an object oriented database management system? (May 2018)

An object-oriented database management system (**OODBMS**) is a database management system that supports the creation and modeling of data as objects. **OODBMS** also includes support for classes of objects and the inheritance of class properties, and incorporates methods, subclasses and their objects.

The main features of object-oriented database management are:

- Capabilities & Scalability.
- Standard Persistence APIs
- Performance and Speed.
- Ease of Use and Learning.
- Object Model & Data Types.
- Primary Keys.
- Query Support.
- Multi User Support.

Advantages of OODBMS:

Extensibility. OODBMSs allow new data types to be built from existing types. The ability to factor out common properties of several classes and form them into a superclass that can be shared with subclasses can greatly reduce redundancy within system is regarded as one of the main **advantages of object orientation**.

OODBMS allow the integration of **databases**, operating systems, spreadsheets, languages, AI systems, word processors and other **objects** or applications. They enable the referential sharing of products and applications, which **is** accomplished through inheritance and **object** identity.

An object-oriented database system must satisfy two criteria: it should be a DBMS, and it should be an object-oriented system, i.e., to the extent possible, it should be consistent with the current crop of object-oriented programming languages.

The first criterion translates into five features: persistence, secondary storage management, concurrency, recovery and an ad hoc query facility.

The second one translates into eight features: complex objects, object identity, encapsulation, types or classes, inheritance, overriding combined with late binding, extensibility and computational completeness.

8.Write short notes on Multimedia databases. (Nov'2015 , Nov 2017 & May 2018)

MULTIMEDIA CONCEPTS:

Multimedia databases that allow users to store and query different types of multimedia information.It includes

- images :- pictures,drawings
- videoclips :- movies,newsreels,home video
- audio clips :- song,phone msg,speeches
- documents :- book ,articles

Content based retrieval:

The multimedia source is being retrieval based on its containing certain objects or activities.

Identifying the contents of multimedia sources is difficult and time –consuming task.

There are two type of approaches:

- Automatic analysis
- Manual identification

Automatic analysis:-

To identify certain mathematical characteristics of their contents. This approach depends on the type of multimedia source(image,text,video,audio).

Manual identification :

This approach can be applied to all the different multimedia sources, but it requires a manual preprocessing phase where a person has to scan each multimedia source to identify and catalog the object and activities it contains so that they can be used to index these sources.

Characteristics of type of multimedia sources:-

Image:

An image is typically stored either in raw form as a set of pixels or cell values or

uncompressed form to save space.

The image shape descriptor describes the geometric shape of the raw image, which is a rectangle of cells of certain width and height.

Image -> represented by an m by n grid of cells.

Types:-

- black and white image pixel----------> one bit
- grey scale or color image---------→pixel is multiple bits

Mathematical transforms:-

Used to reduce the number of cells stored but still maintain the main image characteristics.

- DFT ----> Discrete Fourier Transform
- DCT -- → Discrete Cosine Transform

Techniques for searching the images from databases:

Two main techniques are used to search images.

1.Distance function approach

2.Transformation approach

1. Distance function approach:-

Distance function to compare the given image with the stored images and their segments.

Indexes can be created to group together stored images that are close in the distance metric so as to limit the search space.

2. Transformation approach:-

Measures the image similarity by having a small number of transformations that can transform one image cell to match the other image.Transform include rotations,transformations and scaling.

video source:-

A video source is represented as a sequence of frames where each frame is a still image .

video-→ video segments -> sequence of contiguous frame that include

same objects / activities.

Each video segment can be used to index the segments an indexing technique called frame segment trees has been proposed for video indexing.

The index includes both objects [persons,houses,cars] and activities [driving,talking].

Text/documents source:

Text/document source is basically the full text of some article ,book or magazine

These sources are typically indexed by identifying the keywords that appear in the text and their relative frequencies.

Techniques have been developed to reduce the number of keywords to those that are relevant to the collection.

1. Singular value decomposition:-

Which is based on matrix transformation ,can be used for reducing the number of keywords.

2. Telescoping vector trees:- or TV trees,can be used to group similar documents.

- Audiosources:-

Includes stored recorded messages ,such as speeches ,class presentations,even surveillance recording of phase messages or conversions by law enforcement.

Discrete transforms can be used to identify the main chaacteristics of certain person's voice.

Audio characteristics include loudness,intensity,pitch and clarity.

9. Explain about Object Oriented Databases concepts? (May 2018)

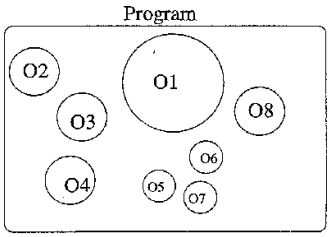
OODB = Object Orientation + Database Capabilities

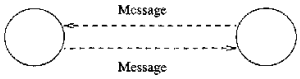
Object-Oriented Database Features:
-persistence
- support of transactions
- simple querying of bulk data
- concurrent access
- resilience
- security

Integration and Sharing
-Seamless integration of operating systems, databases, languages, spreadsheets, word processors, AI expert system shells.
-Sharing of data, information, software components, products, computing environments.
-Referential sharing:
Multiple applications, products, or objects share common sub-objects.

Object-oriented databases allows referential sharing through the support of object identity and inheritance.

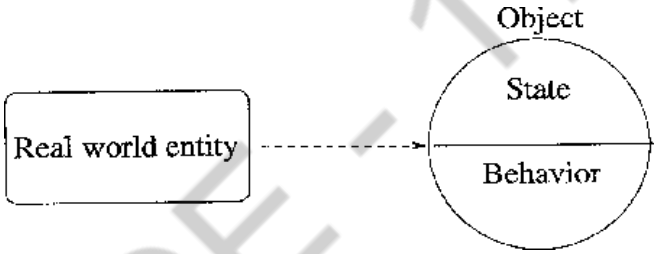
The object-oriented paradigm is illustrated below:





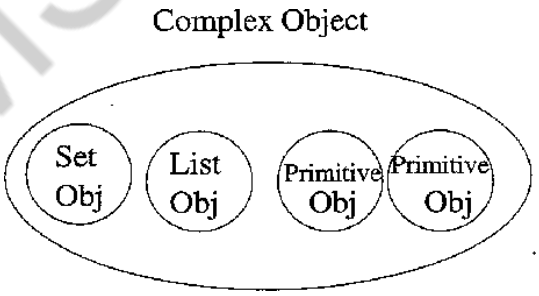
Objects and Identity

The following figure shows object with state and behavior. The state is represented by the values of the object's attributes, and the behavior is defined by the methods acting on the state of the object. There is a unique object identifier OID to identify the object.



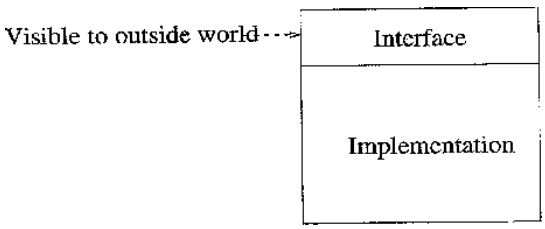
Complex Objects

Complex objects are built by applying constructors to simpler objects including: sets, lists and tuples. An example is illustrated below:



Encapsulation

Encapsulation is derived from the notion of Abstract Data Type (ADT). It is motivated by the need to make a clear distinction between the specification and the implementation of an operation. It reinforces modularity and provides a form of logical data independence.

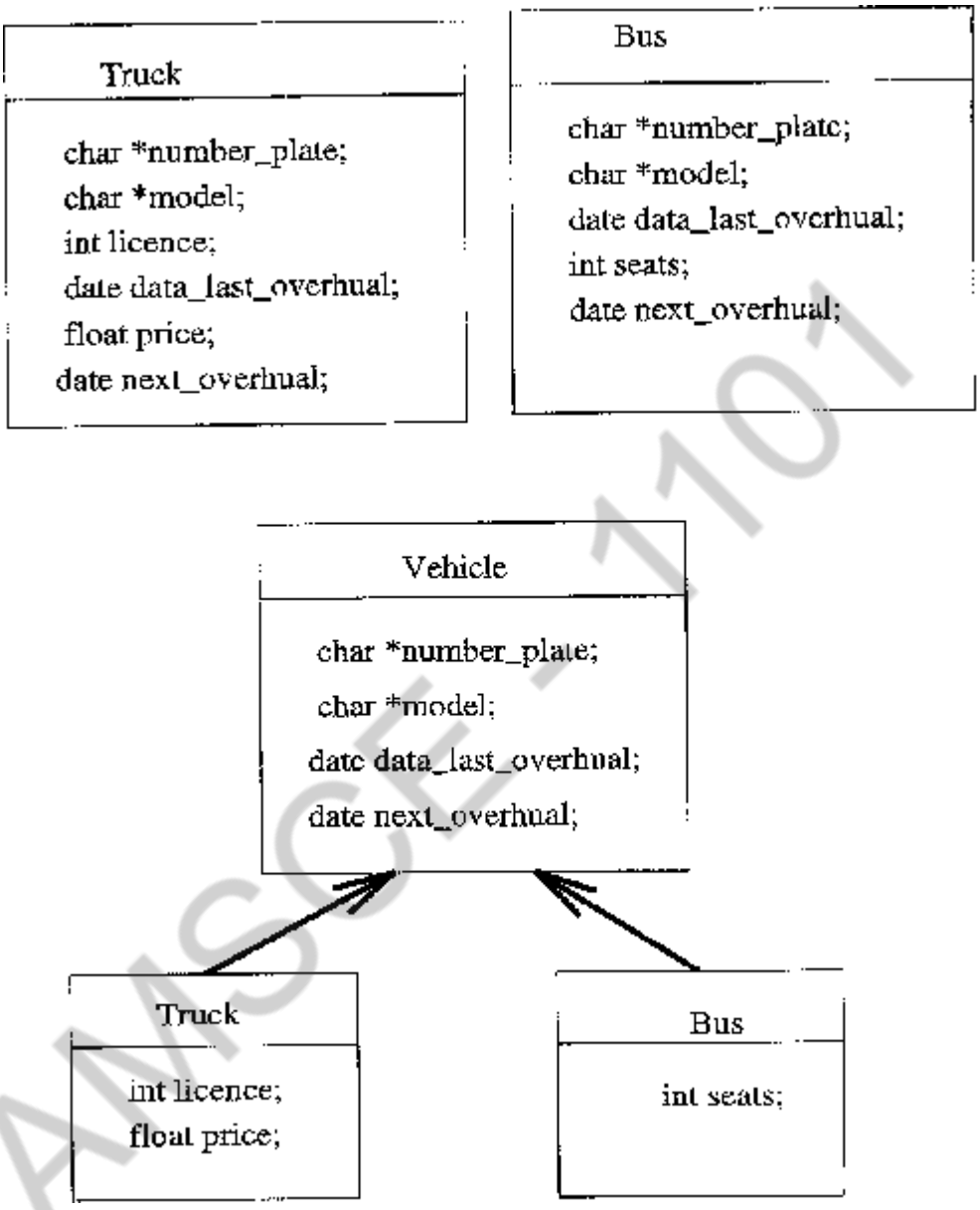


Class

A class object is an object which acts as a template.
It specifies:
A structure that is the set of attributes of the instances
A set of operations
A set of methods which implement the operations
Instantiation means generating objects, Ex. 'new' operation in C++
Persistence of objects: Two approaches
An implicit characteristic of all objects
An orthogonal characteristic - insert the object into a persistent collection of objects

Inheritance

A mechanism of reusability, the most powerful concept of OO programming



Association

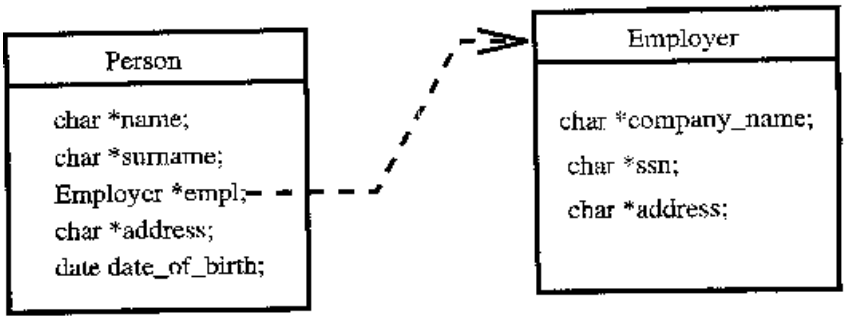
Association is a link between entities in an application

In OODB, associations are represented by means of references between objects

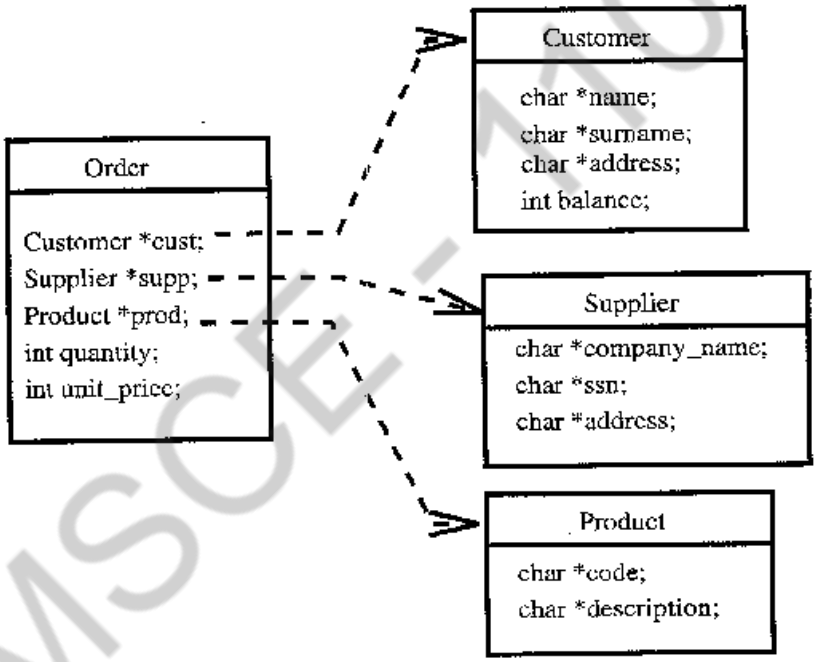
a representation of a binary association

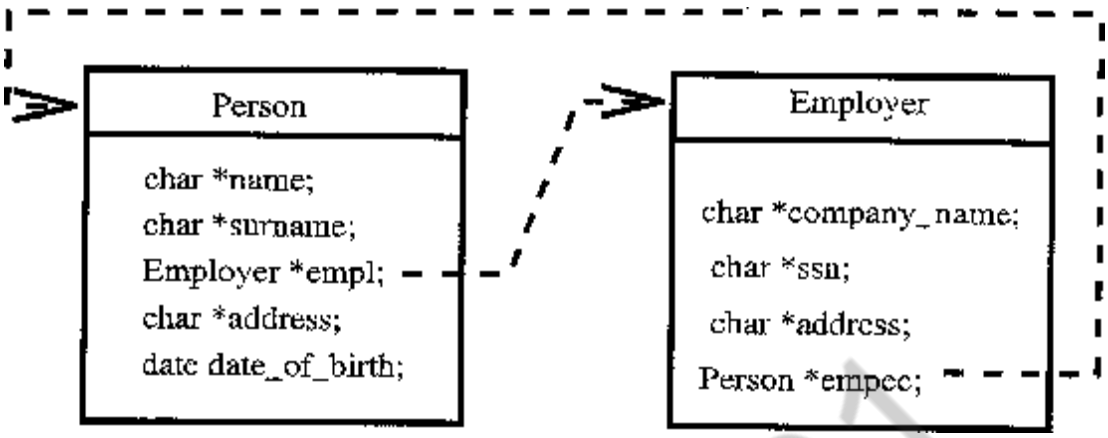
a representation of a ternary association

reverse reference



Representation of a binary association





ADVANTAGES OF OODB

-An integrated repository of information that is shared by multiple users, multiple products, multiple applications on multiple platforms.

- It also solves the following problems:

1. The semantic gap: The real world and the Conceptual model is very similar.

2. Impedance mismatch: Programming languages and database systems must be interfaced to solve application problems. But the language style, data structures, of a programming language (such as C) and the DBMS (such as Oracle) are different. The OODB supports general purpose programming in the OODB framework.

3. New application requirements: Especially in OA, CAD, CAM, CASE, object-orientation is the most natural and most convenient.

10. Explain in detail about XML Databases?

XML database

An **XML database** is a data persistence software system that allows data to be stored in XML format. These data can then be queried, exported and serialized into the desired format. XML databases are usually associated with document-oriented databases.

Two major classes of XML database exist:[1]

1. **XML-enabled**: these may either map XML to traditional database structures (such as a relational database[2]), accepting XML as input and rendering XML as output, or more recently support native XML types within the traditional database. This term implies that the database processes the XML itself (as opposed to relying on middleware).
2. **Native XML** (NXD): the internal model of such databases depends on XML and uses XML documents as the fundamental unit of storage, which are, however, not necessarily stored in the form of text files.

11.Explain about Distributed Databases and their characteristics, functions and advantages and disadvantages. (May'2016)

Distributed Database: A logically interrelated collection of shared data and their description, physically distributed over a computer network.

Distributed Processing: A centralized database, which may be accessed from different computer systems, over an underlying network.

Replicated DBMS: A DDBMS that keeps and controls replicate data, such as Relations, in multiple databases.

Distributed DBMS (DDBMS) consists of a collection of sites, each of which maintains a local db system. So it is a network of computers interconnected by a data communication system so that the physical db is distributed on at least two of the system's components:

Each site on the network is able to process local Transactions (i.e. - access data only in that single site)
Each site may participate in the execution of Global Transactions (i.e. - access data in several sites) which requires communication among the sites.
Note 1: The above can be thought of: Local Applications & Global Applications
Note 2: This scheme is transparent to users.

Homogeneous DDBMS: This is the case when the application programs are independent of how the db is distributed; i.e. if the distribution of the physical data can be altered without having to make alterations to the application programs. Here, all sites use the same DBMS product - same schemata and same data dictionaries.

Heterogeneous DDBMS: This is the case when the application programs are dependent on the physical location of the stored data; i.e. application programs must be altered if data is moved from one site to another. Here, there are different kinds of DBMSs (i.e. Hierarchical, Network, Relational, Object., etc.), with different underlying data models.

Characteristics of a DDBMS

A DDBMS developed by a single vendor may contain:
- Data independence
- Concurrency Control
- Replication facilities
- Recovery facilities
- Coordinated Data Dictionary
- Authorization System
- Shared Manipulation Language

Also:
- Transaction Manager (TM)

• Data Manager (DM)
• Transaction Coordinator (TC)

NOTE: a Distributed Data Processing System is a system where the application programs run on distributed computers which are linked together by a data transmission network.

Advantages of DDBMSs

- More accurately reflects organizational structure
- Shareability and Local Autonomy (enforces global and local policies)
- Availability and Reliability (failed central db vs failed node)
- Performance (process/data migration and speed)
- Economics
- Modular growth
- Integration (with older systems)

Disadvantages of DDBMSs

- Complexity (Replication overhead, etc)
- Maintenance Costs (of sites)
- Security (Network Security)
- Integrity Control (More complex)
- Lack of Standards
- Lack of Experience and Misconceptions
- Database Design more complex

12.Write short notes on Mobile databases. (Nov'2014 & Nov'2015)

Mobile Databases

Recent advances in portable and wireless technology led to mobile computing, a new dimension in data communication and processing.

Portable computing devices coupled with wireless communications allow clients to access data from virtually anywhere and at any time.

There are a number of hardware and software problems that must be resolved before the capabilities of mobile computing can be fully utilized.

Some of the software problems – which may involve data management, transaction management, and database recovery – have their origins in distributed database systems. In mobile computing, the problems are more difficult, mainly:

The limited and intermittent connectivity afforded by wireless communications.

The limited life of the power supply(battery).

The changing topology of the network.

In addition, mobile computing introduces new architectural possibilities and challenges.

Mobile Computing Architecture

The general architecture of a mobile platform is illustrated in Fig.

It is distributed architecture where a number of computers generally referred to as Fixed Hosts and Base Stations are interconnected through a high-speed wired network.

Fixed hosts are general purpose computers configured to manage mobile units.

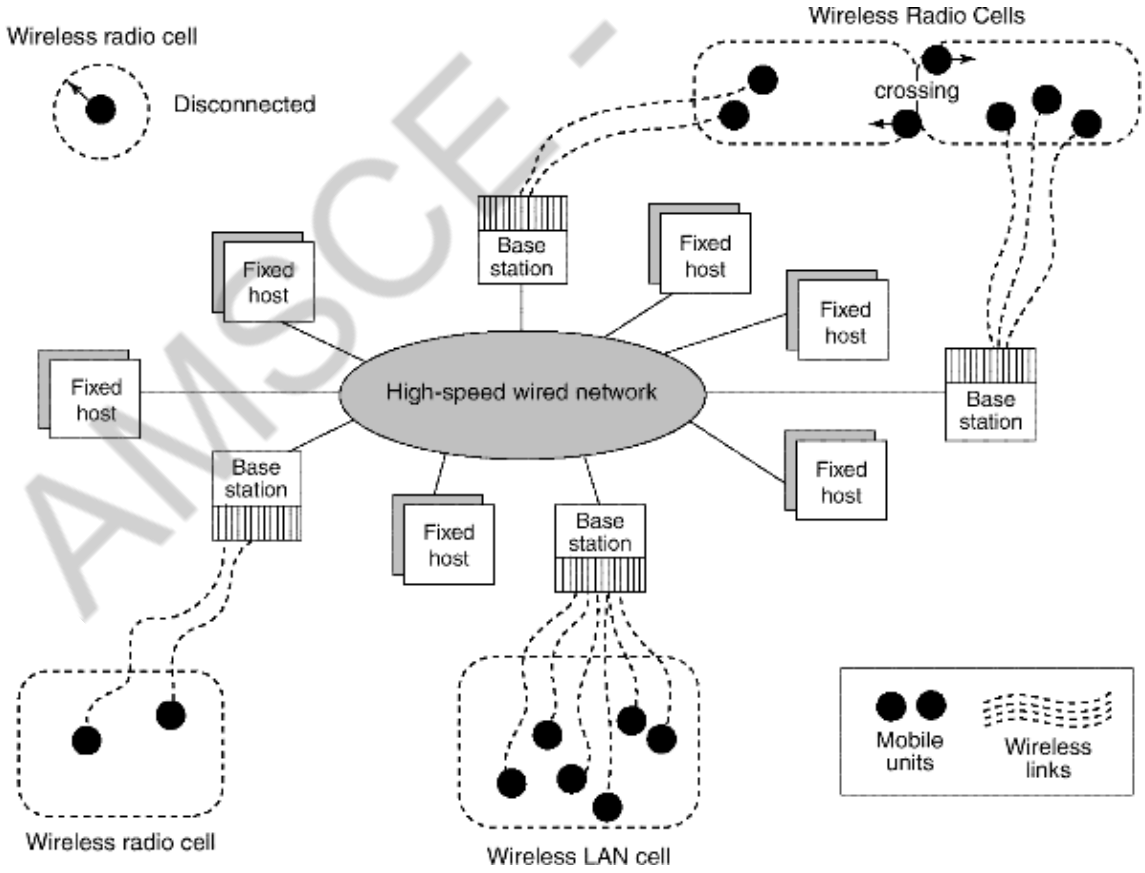
Base stations function as gateways to the fixed network for the Mobile Units.

Wireless Communications –

The wireless medium have bandwidth significantly lower than those of a wired network.

The current generation of wireless technology has data rates range from the tens to hundreds of kilobits per second (2G cellular telephony) to tens of megabits per second (wireless Ethernet, popularly known as WiFi).

Modern (wired) Ethernet, by comparison, provides data rates on the order of hundreds of megabits per second.



Architecture of a Mobile Computing

The other characteristics distinguish wireless connectivity options:

interference, locality of access, range, support for packet switching, seamless roaming throughout a geographical region.

Some wireless networks, such as WiFi and Bluetooth, use unlicensed areas of the frequency spectrum, which may cause interference with other appliances, such as cordless telephones.

Modern wireless networks can transfer data in units called packets, that are used in wired networks in order to conserve bandwidth.

Client/Network Relationships –

Mobile units can move freely in a **geographic mobility domain**, an area that is circumscribed by wireless network coverage.

To manage entire mobility domain is divided into one or more smaller domains, called **cells**, each of which is supported by at least one base station.

Mobile units be unrestricted throughout the cells of domain, while maintaining information **access contiguity**.

The communication architecture described earlier is designed to give the mobile unit the impression that it is attached to a fixed network, emulating a traditional client-server architecture.

Wireless communications, however, make other architectures possible. One alternative is a mobile ad-hoc network (**MANET**)

In a **MANET**, co-located mobile units do not need to communicate via a fixed network, but instead, form their own using cost-effective technologies such as Bluetooth.

In a **MANET**, mobile units are responsible for routing their own data, effectively acting as base stations as well as clients.

Moreover, they must be robust enough to handle changes in the network topology, such as the arrival or departure of other mobile units.

MANET applications can be considered as peer-to-peer, meaning that a mobile unit is simultaneously a client and a server.

Transaction processing and data consistency control become more difficult since there is no central control in this architecture.

Resource discovery and data routing by mobile units make computing in a MANET even more complicated.

Sample MANET applications are multi-user games, shared whiteboard, distributed calendars, and battle information sharing.

Characteristics of Mobile Environments

The characteristics of mobile computing include:

Communication latency

Intermittent connectivity

Limited battery life

Changing client location

The server may not be able to reach a client

A client may be unreachable because it is dozing – in an energy-conserving state in which many subsystems are shut down – or because it is out of range of a base station.

In either case, neither client nor server can reach the other, and modifications must be made to the architecture in order to compensate for this case.

Proxies for unreachable components are added to the architecture.

For a client (and symmetrically for a server), the proxy can cache updates intended for the server.

Mobile computing poses challenges for servers as well as clients.

The latency involved in wireless communication makes scalability a problem.

Since latency due to wireless communications increases the time to νservice each client request, the server can handle fewer clients.

One way servers relieve this problem is by broadcasting data whenever possible.

A server can simply broadcast data periodically.

Broadcast also reduces the load on the server, as clients do not have to maintain active connections to it.

Client mobility also poses many data management challenges.

Servers must keep track of client locations in order to efficiently route messages to them.

Client data should be stored in the network location that minimizes the traffic necessary to access it.

The act of moving between cells must be transparent to the client.

The server must be able to gracefully divert the shipment of data from one base to another, without the client noticing.

Client mobility also allows new applications that are location-based.

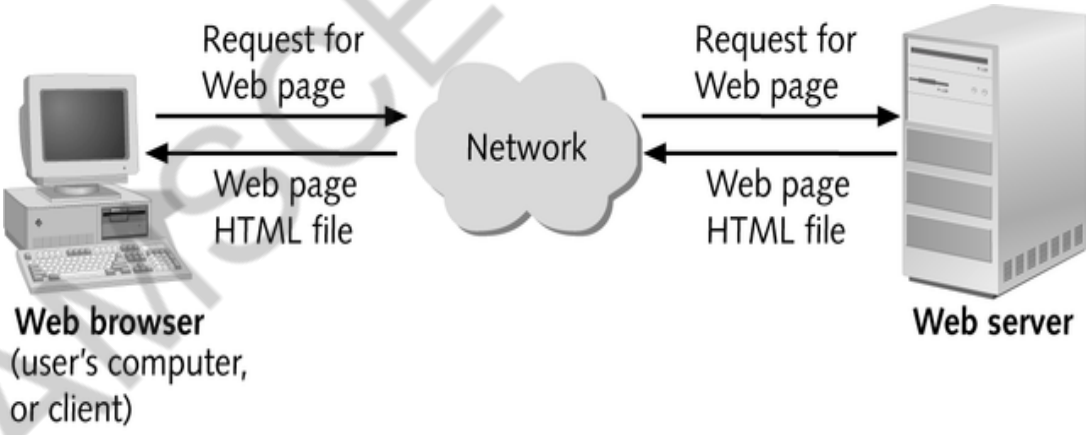
13.Write short notes on Web databases. (Nov'2015)

WEB DATABASES:

Web database is the online **database** that can be only accessed using the **web** form-based interface.

Web Basics

- The Web consists of computers on the Internet connected to each other in a specific way
- The Web has a client/server architecture
- Web browsers
  - Also called **browsers**
  - Programs used to connect client-side computers to the Internet
- Web servers
  - Run special Web server software
  - Listener
    - Component included in Web server software
    - Monitors for messages sent to it from client browsers



- Web page
  - Usually a file with an .htm or .html extension that contains Hypertext Markup Language (HTML) tags and text
  - HTML
    - Document layout language (not a programming language)
    - Defines structure and appearance of Web pages
    - Allows Web pages to embed hypertext links to other Web pages

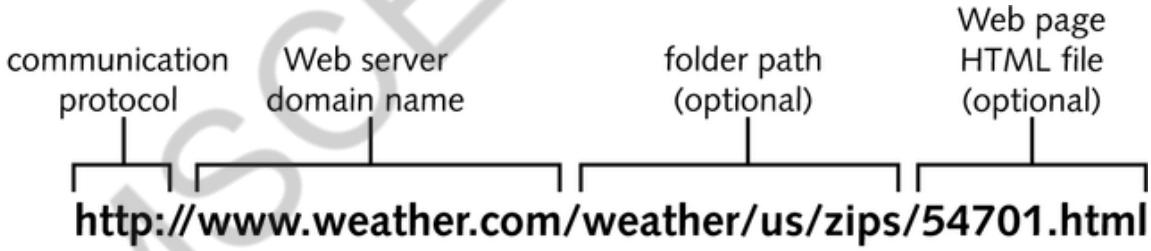
Communication protocols

- Agreements between sender and receiver regarding how data are sent and interpreted
- Internet is built on two network protocols:

        - Transmission Control Protocol (TCP)
        - Internet Protocol (IP)
    - Interaction b/w Browser and Server is governed by the HTTP protocol (Request/Response Tx)
        - HTTP is stateless!
            - (Will discuss in more detail when processing forms)

Web Address

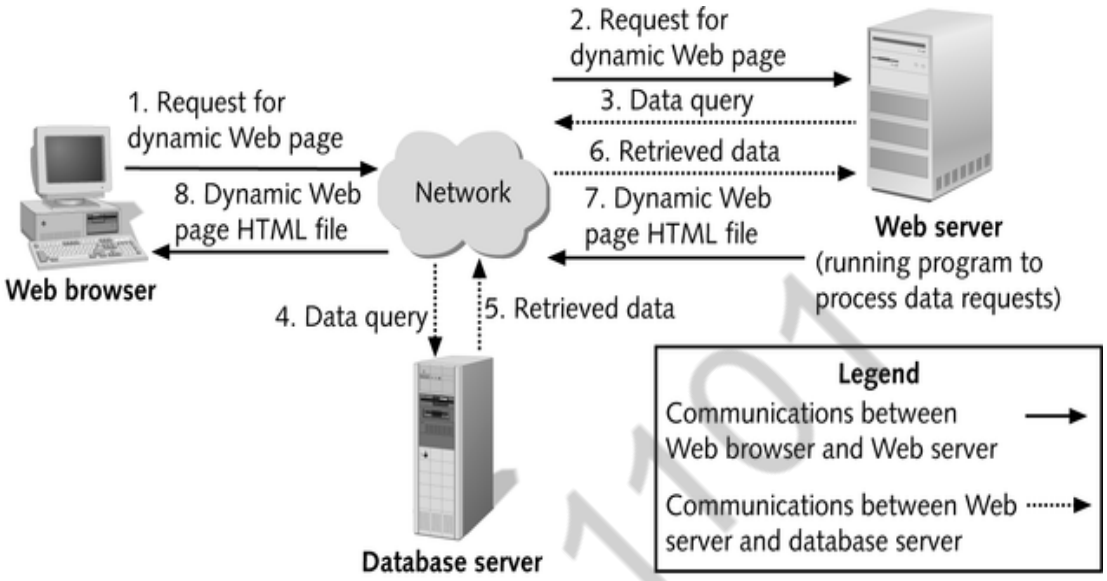
- Packets
    - Data that can be routed independently through Internet
- Domain name
    - Represents an IP address
- A **domain names server** maintains tables with domain names matched to their IP addresses
- Internet Service Providers (ISPs)
    - Provide commercial Internet access
- Hypertext Transfer Protocol
    - Communication protocol used on the Web
- Web address
    - Also called **Uniform Resource Locator (URL)**
- Internet URLs
    - Specify a Web server or domain name
    - Specify communication protocol as first part of URL
- File URL
    - HTML file stored on user's hard drive



Dynamic Web Pages

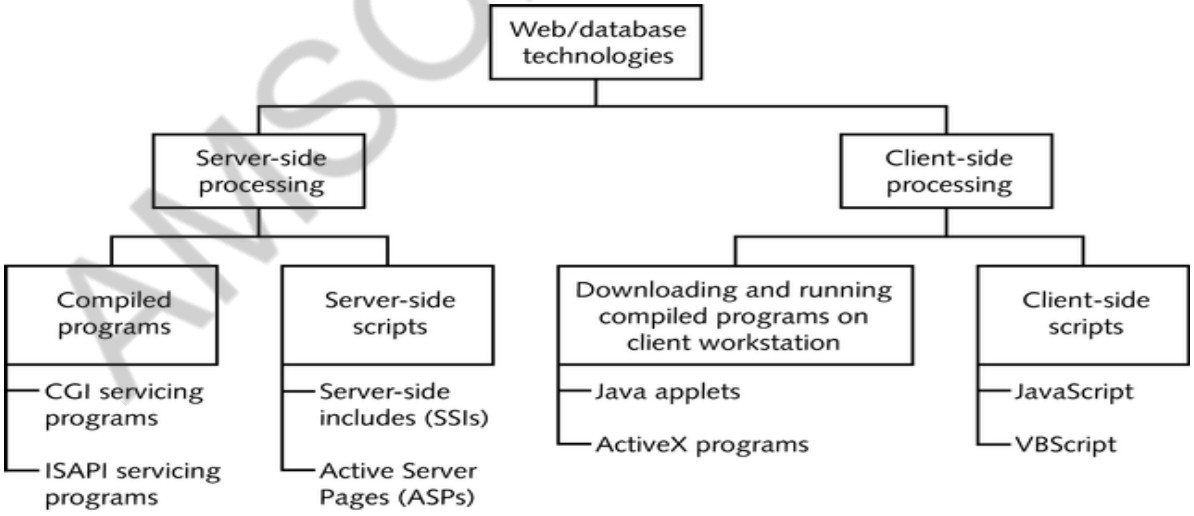
- Static Web page
    - Page content established at the time page is created
    - Useful for displaying data that doesn't change often, and for navigating between HTML Web page files
- Dynamic Web page
    - Also called an interactive Web page
    - Page content varies according to user requests or inputs

Database-driven Web site Architecture



Database-driven Web site architecture for action query

Server-side and client-side Web database technologies



- In **server-side processing**, the Web server:
  - Receives the dynamic Web page request
  - Performs all of the processing necessary to create the dynamic Web page
  - Sends the finished Web page to the client for display in the client's browser

- **Client-side processing**
  - Some processing is done on the client workstation, either to form the request for the dynamic Web page or to create or display the dynamic Web page
  - Eg Javascript code to validate user input.
  - Often needs to be "executed" by the Browser.

14.Explain about Threats and risks in database security.

(OR)

Present an Overview of Database Security. (May 2018)

Database security concerns the use of a broad range of information security controls to protect databases (potentially including the data, the database applications or stored functions, the **database systems**, the **database servers** and the associated network links) against compromises of their **confidentiality, integrity and availability**. It involves various types or **categories of control**s, such as technical, procedural/administrative and physical. ***Database security*** is a specialist topic within the broader realms of computer security, information security and risk management.

Types of Security

Database security is a broad area that addresses many issues, including the following:

- Various legal and ethical issues regarding the right to access certain information
- Policy issues at the governmental, institutional, or corporate level as to what kinds of information should not be made publicly available—for example, credit ratings and personal medical records.
- System-related issues such as the *system levels* at which various security functions should be enforced—for example, whether a security function should be handled at the physical hardware level, the operating system level, or the DBMS level.
- The need in some organizations to identify multiple *security levels* and to categorize the data and users based on these classifications—for example, top secret, secret, confidential, and unclassified.

Threats to databases

- Loss of **integrity**
- Loss of **availability**
- Loss of **confidentiality**

To protect databases against these types of threats four kinds of countermeasures can be implemented:

- **Access control:** user accounts, passwords
- **Inference control:** when statistical databases are used
- **Flow control:** to avoid sensitive data reaching unauthorized users
- **Encryption**

Threats

- Excessive and Unused Privileges
- Privilege Abuse
- SQL Injection
- Malware
- Weak Audit Trail
- Storage Media Exposure

- Exploitation of Vulnerabilities
- Misconfigured Databases
- Unmanaged Sensitive Data
- Denial of Service

Security risk to Database System:

Security risks to database systems include, for example:

- Unauthorized or unintended activity or misuse by authorized database users, database administrators, or network/systems managers, or by unauthorized users or hackers (e.g. inappropriate access to sensitive data, metadata or functions within databases, or inappropriate changes to the database programs, structures or security configurations);
- Malware infections causing incidents such as unauthorized access, leakage or disclosure of personal or proprietary data, deletion of or damage to the data or programs, interruption or denial of authorized access to the database, attacks on other systems and the unanticipated failure of database services;
- Overloads, performance constraints and capacity issues resulting in the inability of authorized users to use databases as intended;
- Physical damage to database servers caused by computer room fires or floods, overheating, lightning, accidental liquid spills, static discharge, electronic breakdowns/equipment failures and obsolescence;
- Design flaws and programming bugs in databases and the associated programs and systems, creating various security vulnerabilities (e.g. unauthorized privilege escalation), data loss/corruption, performance degradation etc.;
- Data corruption and/or loss caused by the entry of invalid data or commands, mistakes in database or system administration processes, sabotage/criminal damage.

15.Explain in detail about Object Relational Features? (Dec 2016)

An object-relational database is a database management system similar to a relational database, but with an object-oriented database model: objects, classes and inheritance are directly supported in database schemas as well as within the query language. The popular RDBMS on the market today is Oracle. For the main features of Object-relational database systems are listed below:

User Data Types & Complex Objects:

Complex data creation in most SQL ORDBMSs is based on preliminary schema definition via the user-defined type (UDT). Object-Relational Database Managerment Systems (ORDBMSs) allow users to define datatypes, functions and operators. As a result, the functionality of the ORDBMSs increases along with their performance.

```
1.  create or replace type point as object(
2.    x number,
3.    y number
4.  )
5.  create or replace type side as object(
6.  a point,
7.  b point,
8.  member function get_length return integer
9.  );
```

Inheritance:

In all ORDBMS, inheritance of user defined types is supported to derive new sub-types or sub-classes which would therefore inherit all attributes and methods from the parent type.

```
1.  create or replace type eshape_tp under shape_tp(
2.  OVERRIDING MEMBER FUNCTION get_area return integer
3.  )
```

Method Encapsulation:

Methods are procedures or functions can be encapsulated within the defined object so that applications can use to perform operations on the attributes of the object type. Methods are optional. They define the behavior of objects of that type.

```
1.  create or replace type side as object(
2.  a point,
3.  b point,
4.  member function get_length return integer
5.  );
6.
7.  create or replace type body side as
8.  member function get_length return integer is
9.  begin
10. return sqrt( ((self.a.x-self.b.x)*(self.a.x-self.b.x)) + ((self.a.y-self.b.y)*(self.a.y-self.b.y)) );
11. end;
12. end;
```

16. Explain in detail about XQuery? (May 2017)

XQuery is a query-based language to retrieve data stored in the form of XML. XQuery is to XML what SQL is to a database.

XQuery is a functional language that is used to retrieve information stored in XML format. XQuery can be used on XML documents, relational databases containing data in XML formats, or XML Databases. XQuery 3.0 is a W3C recommendation from April 8, 2014.

The definition of XQuery as given by its official documentation is as follows −

XQuery is a standardized language for combining documents, databases, Web pages and almost anything else. It is very widely implemented. It is powerful and easy to learn. XQuery is replacing proprietary middleware languages and Web Application development languages. XQuery is replacing complex Java or C++ programs with a few lines of code. XQuery is simpler to work with and easier to maintain than many other alternatives.

Characteristics:

- **Functional Language** − XQuery is a language to retrieve/querying XML based data.
- **Analogous to SQL** − XQuery is to XML what SQL is to databases.
- **XPath based** − XQuery uses XPath expressions to navigate through XML documents.

- **Universally accepted** − XQuery is supported by all major databases.
- **W3C Standard** − XQuery is a W3C standard.

Benefits of XQuery:

- Using XQuery, both hierarchical and tabular data can be retrieved.
- XQuery can be used to query tree and graphical structures.
- XQuery can be directly used to query webpages.
- XQuery can be directly used to build webpages.
- XQuery can be used to transform xml documents.
- XQuery is ideal for XML-based databases and object-based databases. Object databases are much more flexible and powerful than purely tabular databases.

XPATH:

- XQuery is XPath compliant. It uses XPath expressions to restrict the search results on XML collections.
- *XPath Examples:*
- We will use the books.xml file and apply XQuery to it.
- books.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<books>

  <book category="JAVA">
    <title lang="en">Learn Java in 24 Hours</title>
    <author>Robert</author>
    <year>2005</year>
    <price>30.00</price>
  </book>

  <book category="DOTNET">
    <title lang="en">Learn .Net in 24 hours</title>
    <author>Peter</author>
    <year>2011</year>
    <price>40.50</price>
  </book>

  <book category="XML">
    <title lang="en">Learn XQuery in 24 hours</title>
    <author>Robert</author>
    <author>Peter</author>
    <year>2013</year>
    <price>50.00</price>
  </book>
```

```
<book category="XML">
  <title lang="en">Learn XPath in 24 hours</title>
  <author>Jay Ban</author>
  <year>2010</year>
  <price>16.50</price>
</book>

</books>
```

- We have given here three versions of an XQuery statement that fulfil the same objective of displaying the book titles having a price value greater than 30.

- XQuery – Version 1

```
(: read the entire xml document :)
let $books := doc("books.xml")

for $x in $books/books/book
where $x/price > 30
return $x/title
```

- Output

```
<title lang="en">Learn .Net in 24 hours</title>
<title lang="en">Learn XQuery in 24 hours</title>
```

- XQuery – Version 2

```
(: read all books :)
let $books := doc("books.xml")/books/book

for $x in $books
where $x/price > 30
return $x/title
```

- Output

```
<title lang="en">Learn .Net in 24 hours</title>
<title lang="en">Learn XQuery in 24 hours</title>
```

- XQuery – Version 3

```
(: read books with price > 30 :)
let $books := doc("books.xml")/books/book[price > 30]

for $x in $books
return $x/title
```

- Output
- <title lang="en">Learn .Net in 24 hours</title>
- <title lang="en">Learn XQuery in 24 hours</title>

17.Explain about Types of Privileges in database language.

A privilege is a right to execute a particular type of SQL statement or to access another user's object. Some examples of privileges include the right to:

- Connect to the database
- Create a table
- Select rows from another user's table
- Execute another user's stored procedure

You grant privileges to users so these users can accomplish tasks required for their jobs. You should grant a privilege only to a user who requires that privilege to accomplish the in necessary work. Excessive granting of unnecessary privileges can compromise security.

A user can receive a privilege in two different ways:

- You can grant privileges to users explicitly. For example, you can explicitly grant to user SCOTT the privilege to insert records into the employees table.
- You can also grant privileges to a role (a named group of privileges), and then grant the role to one or more users. For example, you can grant the privileges to select, insert, update, and delete records from the employees table to the role named clerk, which in turn you can grant to users scott and brian.

Types:

- System Privileges
- Schema Object Privileges
- Table Privileges

- View Privileges
- Procedure Privileges

- Type Privileges
  In some cases it is desirable to grant a privilege to a user temporarily. For example,
- The owner of a relation may want to grant the **SELECT** privilege to a user for a specific task and then revoke that privilege once the task is completed.
- Hence, a mechanism for **revoking** privileges is needed. In SQL, a **REVOKE** command is included for the purpose of **canceling privileges**.
  Suppose that the DBA creates four accounts
- A1, A2, A3, A4 and wants only A1 to be able to create base relations. Then the DBA must issue the following GRANT command in SQL
- **GRANT** CREATETAB TO A1;
- In SQL the same effect can be accomplished by having the DBA issue a **CREATE SCHEMA** command as follows:
- **CREATE SCHAMA** EXAMPLE **AUTHORIZATION** A1;
  - User account A1 can create tables under the schema called **EXAMPLE**.
  - Suppose that A1 **creates** the two base relations **EMPLOYEE** and **DEPARTMENT**
  - A1 is then **owner** of these two relations and hence all the relation privileges on each of them.
  - Suppose that A1 wants to grant A2 the privilege to insert and delete tuples in both of these relations, but A1 does not want A2 to be able to propagate these privileges to additional accounts:

- **GRANT INSERT, DELETE ON** EMPLOYEE, DEPARTMENT **TO A2;**
  - Suppose that A1 decides to revoke the SELECT privilege on the EMPLOYEE relation from A3; A1 can issue:

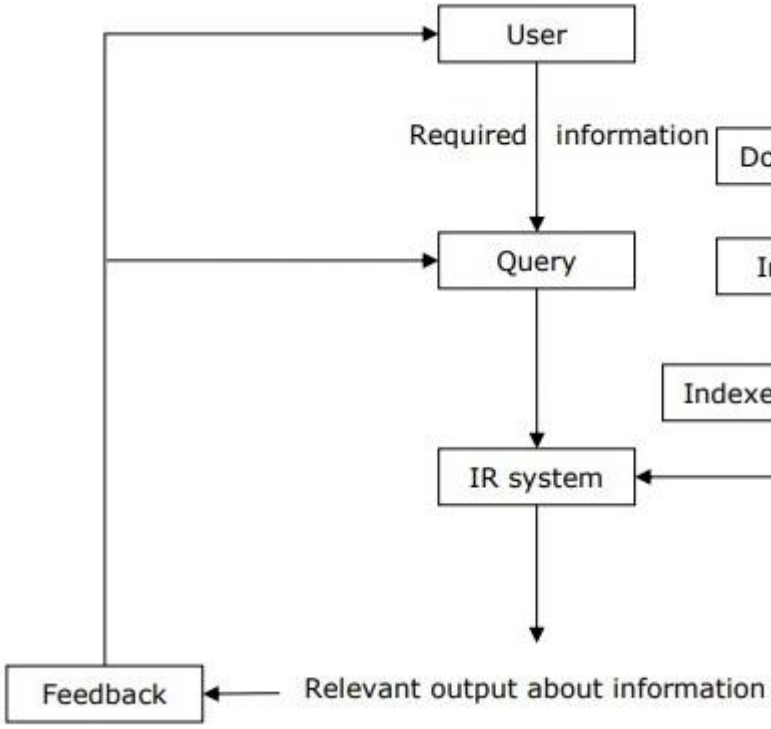
  REVOKE SELECT ON EMPLOYEE **FROM** A3**;**

18.Explain in detail about Information Retrieval ? (Nov 2014, Nov 2019)

Information retrieval:

Information retrieval (IR) may be defined as a software program that deals with the organization, storage, retrieval and evaluation of information from document repositories particularly textual information. The system assists users in finding the information they require but it does not explicitly return the answers of the questions. It informs the existence and location of documents that might consist of the required information. The documents that satisfy user's requirement are called relevant documents. A perfect IR system will retrieve only relevant documents.

With the help of the following diagram, we can understand the process of information retrieval (IR) −



It is clear from the above diagram that a user who needs information will have to formulate a request in the form of query in natural language. Then the IR system will respond by retrieving the relevant output, in the form of documents, about the required information.

Classical Problem in Information Retrieval (IR) System

The main goal of IR research is to develop a model for retrieving information from the repositories of documents. Here, we are going to discuss a classical problem, named **ad-hoc retrieval problem**, related to the IR system.

In ad-hoc retrieval, the user must enter a query in natural language that describes the required information. Then the IR system will return the required documents related to the desired information. For example, suppose we are searching something on the Internet and it gives some exact pages that are relevant as per our requirement but there can be some non-relevant pages too. This is due to the ad-hoc retrieval problem.

Aspects of Ad-hoc Retrieval

Followings are some aspects of ad-hoc retrieval that are addressed in IR research −

- How users with the help of relevance feedback can improve original formulation of a query?
- How to implement database merging, i.e., how results from different text databases can be merged into one result set?
- How to handle partly corrupted data? Which models are appropriate for the same?

Information Retrieval (IR) Model

Mathematically, models are used in many scientific areas having objective to understand some phenomenon in the real world. A model of information retrieval predicts and explains what a user will find

in relevance to the given query. IR model is basically a pattern that defines the above-mentioned aspects of retrieval procedure and consists of the following −

- A model for documents.
- A model for queries.
- A matching function that compares queries to documents.

Mathematically, a retrieval model consists of −

D − Representation for documents.

R − Representation for queries.

F − The modeling framework for D, Q along with relationship between them.

R (q,di) − A similarity function which orders the documents with respect to the query. It is also called ranking.

Types of Information Retrieval (IR) Model

An information model (IR) model can be classified into the following three models −

Classical IR Model

It is the simplest and easy to implement IR model. This model is based on mathematical knowledge that was easily recognized and understood as well. Boolean, Vector and Probabilistic are the three classical IR models.

Non-Classical IR Model

It is completely opposite to classical IR model. Such kind of IR models are based on principles other than similarity, probability, Boolean operations. Information logic model, situation theory model and interaction models are the examples of non-classical IR model.

Alternative IR Model

It is the enhancement of classical IR model making use of some specific techniques from some other fields. Cluster model, fuzzy model and latent semantic indexing (LSI) models are the example of alternative IR model.

Design features of Information retrieval (IR) systems

Let us now learn about the design features of IR systems −

Inverted Index

The primary data structure of most of the IR systems is in the form of inverted index. We can define an inverted index as a data structure that list, for every word, all documents that contain it and frequency of the occurrences in document. It makes it easy to search for 'hits' of a query word.

Stop Word Elimination

Stop words are those high frequency words that are deemed unlikely to be useful for searching. They have less semantic weights. All such kind of words are in a list called stop list. For example, articles "a", "an",

"the" and prepositions like "in", "of", "for", "at" etc. are the examples of stop words. The size of the inverted index can be significantly reduced by stop list. As per Zipf's law, a stop list covering a few dozen words reduces the size of inverted index by almost half. On the other hand, sometimes the elimination of stop word may cause elimination of the term that is useful for searching. For example, if we eliminate the alphabet "A" from "Vitamin A" then it would have no significance.

Stemming

Stemming, the simplified form of morphological analysis, is the heuristic process of extracting the base form of words by chopping off the ends of words. For example, the words laughing, laughs, laughed would be stemmed to the root word laugh.

In our subsequent sections, we will discuss about some important and useful IR models.

The Boolean Model

It is the oldest information retrieval (IR) model. The model is based on set theory and the Boolean algebra, where documents are sets of terms and queries are Boolean expressions on terms. The Boolean model can be defined as −

- **D** − A set of words, i.e., the indexing terms present in a document. Here, each term is either present (1) or absent (0).
- **Q** − A Boolean expression, where terms are the index terms and operators are logical products − AND, logical sum − OR and logical difference − NOT
- **F** − Boolean algebra over sets of terms as well as over sets of documents

  If we talk about the relevance feedback, then in Boolean IR model the Relevance prediction can be defined as follows −
- **R** − A document is predicted as relevant to the query expression if and only if it satisfies the query expression as −

$((text \vee information) \wedge rerieval \wedge \tilde{} \; theory)$

We can explain this model by a query term as an unambiguous definition of a set of documents.

For example, the query term ***"economic"*** defines the set of documents that are indexed with the term ***"economic"***.

Now, what would be the result after combining terms with Boolean AND Operator? It will define a document set that is smaller than or equal to the document sets of any of the single terms. For example, the query with terms ***"social"*** and ***"economic"*** will produce the documents set of documents that are indexed with both the terms. In other words, document set with the intersection of both the sets.

Now, what would be the result after combining terms with Boolean OR operator? It will define a document set that is bigger than or equal to the document sets of any of the single terms. For example, the query with terms ***"social"*** or ***"economic"*** will produce the documents set of documents that are indexed with either the term ***"social"*** or ***"economic"***. In other words, document set with the union of both the sets.

Advantages of the Boolean Mode

The advantages of the Boolean model are as follows −

- The simplest model, which is based on sets.

- Easy to understand and implement.
- It only retrieves exact matches
- It gives the user, a sense of control over the system.

Disadvantages of the Boolean Model

The disadvantages of the Boolean model are as follows −

- The model's similarity function is Boolean. Hence, there would be no partial matches. This can be annoying for the users.
- In this model, the Boolean operator usage has much more influence than a critical word.
- The query language is expressive, but it is complicated too.
- No ranking for retrieved documents.

Vector Space Model

Due to the above disadvantages of the Boolean model, Gerard Salton and his colleagues suggested a model, which is based on Luhn's similarity criterion. The similarity criterion formulated by Luhn states, "the more two representations agreed in given elements and their distribution, the higher would be the probability of their representing similar information."

Consider the following important points to understand more about the Vector Space Model −

- The index representations (documents) and the queries are considered as vectors embedded in a high dimensional Euclidean space.
- The similarity measure of a document vector to a query vector is usually the cosine of the angle between them.

Cosine Similarity Measure Formula

Cosine is a normalized dot product, which can be calculated with the help of the following formula –

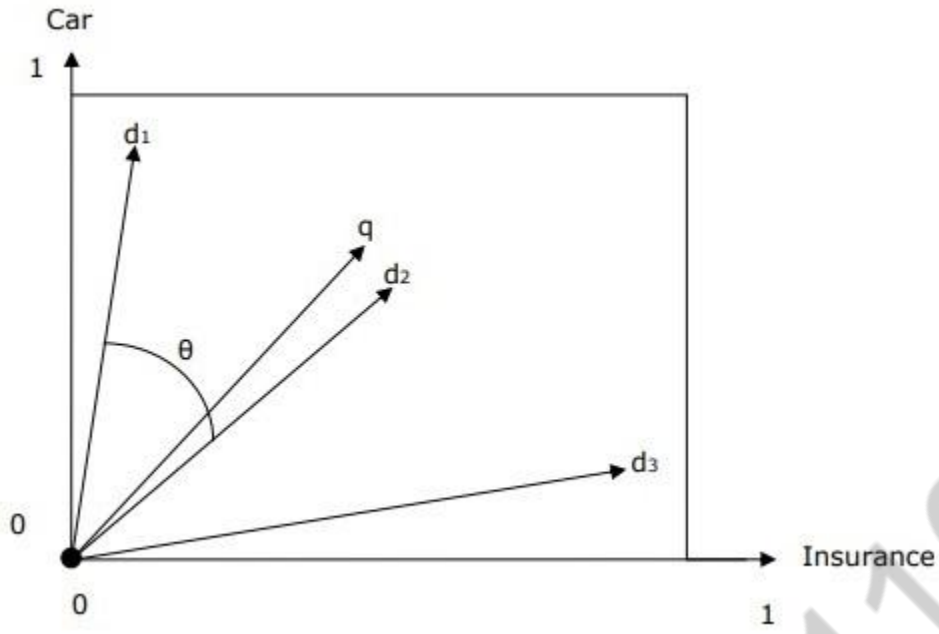
$$Score(\vec{d}\,\vec{q}) = \frac{\sum_{k=1}^{m} d_k \cdot q_k}{\sqrt{\sum_{k=1}^{m}(d_k)^2} \cdot \sqrt{\sum_{k=1}^{m} m(q_k)^2}}$$

$$Score(\vec{d}\,\vec{q}) = 1 \; when \; d = q$$

$$Score(\vec{d}\,\vec{q}) = 0 \; when \; d \; and \; q \; share \; no \; items$$

Vector Space Representation with Query and Document

The query and documents are represented by a two-dimensional vector space. The terms are ***car*** and ***insurance***. There is one query and three documents in the vector space.



The top ranked document in response to the terms car and insurance will be the document $\mathbf{d_2}$ because the angle between **q** and $\mathbf{d_2}$ is the smallest. The reason behind this is that both the concepts car and insurance are salient in d_2 and hence have the high weights. On the other side, $\mathbf{d_1}$ and $\mathbf{d_3}$ also mention both the terms but in each case, one of them is not a centrally important term in the document.

Term Weighting

Term weighting means the weights on the terms in vector space. Higher the weight of the term, greater would be the impact of the term on cosine. More weights should be assigned to the more important terms in the model. Now the question that arises here is how can we model this.

One way to do this is to count the words in a document as its term weight. However, do you think it would be effective method?

Another method, which is more effective, is to use **term frequency ($\mathbf{tf_{ij}}$), document frequency ($\mathbf{df_i}$)** and **collection frequency ($\mathbf{cf_i}$)**.

Term Frequency (tf_{ij})

It may be defined as the number of occurrences of $\mathbf{w_i}$ in $\mathbf{d_j}$. The information that is captured by term frequency is how salient a word is within the given document or in other words we can say that the higher the term frequency the more that word is a good description of the content of that document.

Document Frequency (df_i)

It may be defined as the total number of documents in the collection in which w_i occurs. It is an indicator of informativeness. Semantically focused words will occur several times in the document unlike the semantically unfocused words.

Collection Frequency (cf_i)

It may be defined as the total number of occurrences of $\mathbf{w_i}$ in the collection.

Mathematically,

$$df_i \leq cf_i \; and \; \sum_j tf_{ij} = cf_i$$

Forms of Document Frequency Weighting

Let us now learn about the different forms of document frequency weighting. The forms are described below −

Term Frequency Factor

This is also classified as the term frequency factor, which means that if a term ***t*** appears often in a document then a query containing ***t*** should retrieve that document. We can combine word's **term frequency (tf_{ij})** and **document frequency (df_i)** into a single weight as follows −

$$weight\,(i,j) = \begin{cases} (1 + log(tf_{ij}))log\frac{N}{df_i} & if\; tf_{i,j} \geq 1 \\ 0 & if\; tf_{i,j} = 0 \end{cases}$$

Here N is the total number of documents.

Inverse Document Frequency (idf)

This is another form of document frequency weighting and often called idf weighting or inverse document frequency weighting. The important point of idf weighting is that the term's scarcity across the collection is a measure of its importance and importance is inversely proportional to frequency of occurrence.

Mathematically,

$$idf_t = log\left(1 + \frac{N}{n_t}\right)$$

$$idf_t = log\left(\frac{N - n_t}{n_t}\right)$$

Here,

N = documents in the collection

n_t = documents containing term *t*

User Query Improvement

The primary goal of any information retrieval system must be accuracy − to produce relevant documents as per the user's requirement. However, the question that arises here is how can we improve the output by improving user's query formation style. Certainly, the output of any IR system is dependent on the user's

query and a well-formatted query will produce more accurate results. The user can improve his/her query with the help of ***relevance feedback***, an important aspect of any IR model.

Relevance Feedback

Relevance feedback takes the output that is initially returned from the given query. This initial output can be used to gather user information and to know whether that output is relevant to perform a new query or not. The feedbacks can be classified as follows −

Explicit Feedback

It may be defined as the feedback that is obtained from the assessors of relevance. These assessors will also indicate the relevance of a document retrieved from the query. In order to improve query retrieval performance, the relevance feedback information needs to be interpolated with the original query.

Assessors or other users of the system may indicate the relevance explicitly by using the following relevance systems −

- **Binary relevance system** − This relevance feedback system indicates that a document is either relevant (1) or irrelevant (0) for a given query.
- **Graded relevance system** − The graded relevance feedback system indicates the relevance of a document, for a given query, on the basis of grading by using numbers, letters or descriptions. The description can be like "not relevant", "somewhat relevant", "very relevant" or "relevant".

Implicit Feedback

It is the feedback that is inferred from user behavior. The behavior includes the duration of time user spent viewing a document, which document is selected for viewing and which is not, page browsing and scrolling actions, etc. One of the best examples of implicit feedback is ***dwell time***, which is a measure of how much time a user spends viewing the page linked to in a search result.

Pseudo Feedback

It is also called Blind feedback. It provides a method for automatic local analysis. The manual part of relevance feedback is automated with the help of Pseudo relevance feedback so that the user gets improved retrieval performance without an extended interaction. The main advantage of this feedback system is that it does not require assessors like in explicit relevance feedback system.

Consider the following steps to implement this feedback −

- **Step 1** − First, the result returned by initial query must be taken as relevant result. The range of relevant result must be in top 10-50 results.
- **Step 2** − Now, select the top 20-30 terms from the documents using for instance term frequency(tf)-inverse document frequency(idf) weight.
- **Step 3** − Add these terms to the query and match the returned documents. Then return the most relevant documents.

19.Explain types of database security and database security issues.(May 2016)

Database security issues

- Types of Security
  - Legal and ethical issues
  - Policy issues

- System-related issues
- The need to identify multiple security levels

- Threats to databases
  - Loss of **integrity**
  - Loss of **availability**
  - Loss of **confidentiality**
- To protect databases against these types of threats four kinds of countermeasures can be implemented:

Access control

- The security mechanism of a DBMS must include provisions for restricting access to the database as a whole
- This function is called **access control** and is handled by creating user accounts and passwords to control login process by the DBMS.

Inference control

- The security problem associated with databases is that of controlling the access to a **statistical database**, which is used to provide statistical information or summaries of values based on various criteria.
- The countermeasures to **statistical database security** problem is called **inference control measures**.

Flow control

- Another security is that of **flow control**, which prevents information from flowing in such a way that it reaches unauthorized users.

Encryption

- A final security issue is **data encryption**, which is used to protect sensitive data (such as credit card numbers) that is being transmitted via some type communication network.
- The data is **encoded** using some **encoding algorithm**.

Types of database security

- A DBMS typically includes a database security and authorization subsystem that is responsible for ensuring the security portions of a database against unauthorized access.

Types of database security mechanisms:

- Discretionary Access Control
- Mandatory Access Control
- Role based Access Control.

Discretionary Access Control

The typical method of enforcing **discretionary access control** in a database system is based on the **granting** and **revoking privileges**

Types of Discretionary Privileges

The **account level**:

□At this level, the DBA specifies the particular privileges that each account holds independently of the relations in the database.

□The privileges at the **account level** 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 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.

□The **relation level** (or **table level**):

□At this level, the DBA can control the privilege to access each individual relation or view in the database.

□This includes **base relations** and virtual (**view**) relations.

□To control the granting and revoking of relation privileges, each relation R in a database is assigned and **owner account**, which is typically the account that was used when the relation was

created in the first place.

□The owner of a relation is given all privileges on that relation.

□Suppose that the DBA creates four accounts

□A1, A2, A3, A4

□And wants only A1 to be able to create base relations. Then the DBA must issue the following

GRANT command in SQL

GRANT CREATETAB TO A1;

□In SQL2 the same effect can be accomplished by having the DBA issue a **CREATE SCHEMA**

command as follows:

CREATE SCHAMA EXAMPLE **AUTHORIZATION** A1;

□Suppose that A1 wants to allow A3 to retrieve information from either of the two tables and also

to be able to propagate the SELECT privilege to other accounts.

□A1 can issue the command:

GRANT SELECT ON EMPLOYEE, DEPARTMENT

TO A3 **WITH GRANT OPTION;**

Mandatory Access Control and Role-Based Access Control for Multilevel Security

□The discretionary access control techniques of granting and revoking privileges on relations have

traditionally been the main security mechanism for relational database systems.

□This is an all-or-nothing method:

□A user either has or does not have a certain privilege.

□In many applications, and additional security policy is needed that classifies data and users based
on security classes.
□This approach as mandatory access control would typically be combined with the discretionary access control mechanisms.
□Typical security classes are top secret (TS), secret (S), confidential (C), and unclassified (U),
where TS is the highest level and U the lowest: TS ≥ S ≥ C ≥ U
□The commonly used model for multilevel security, known as the Bell-LaPadula model, classifies
each subject (user, account, program) and object (relation, tuple, column, view, operation) into
one of the security classifications, T, S, C, or U:
□Clearance (classification) of a subject S as class(S) and to the classification of an object O as class (O).
□Two restrictions are enforced on data access based on the subject/object classifications:
□Simple security property: A subject S is not allowed read access to an object O unless class(S) ≥ class (O).
□A subject S is not allowed to write an object O unless class(S) ≤ class(O). This known as the star property (or * property).

Role-based access control

□**I**ts basic notion is that permissions are associated with roles, and users are assigned to appropriate
roles.
□Roles can be created using the CREATE ROLE and DESTROY ROLE commands.
□The GRANT and REVOKE commands discussed under DAC can then be used to assign and revoke privileges from roles.

20.Discuss about the Access control mechanism and Cryptography methods to secure the Database. (Nov'2014 & May 2015)

Access control mechanism

It's all about user's access to the database.

Types

Discretionary Access Control
Mandatory Access Control
Role based Access Control.

Discretionary Access Control

The typical method of enforcing **discretionary access control** in a database system is based on the **granting** and **revoking privileges**

Types of Discretionary Privileges

The **account level**:
□At this level, the DBA specifies the particular privileges that each account holds independently of the relations in the database.

□The privileges at the **account level** 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 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.

The **relation level** (or **table level**):
□At this level, the DBA can control the privilege to access each individual relation or view in the database.
□This includes **base relations** and virtual (**view**) relations.
□To control the granting and revoking of relation privileges, each relation R in a database is assigned and **owner account**, which is typically the account that was used when the relation was created in the first place.
□The owner of a relation is given all privileges on that relation.
□Suppose that the DBA creates four accounts A1, A2, A3, A4 and wants only A1 to be able to create base relations. Then the DBA must issue the following

GRANT command in SQL
GRANT CREATETAB TO A1;
□In SQL2 the same effect can be accomplished by having the DBA issue a **CREATE SCHEMA** command as follows:

CREATE SCHAMA EXAMPLE **AUTHORIZATION** A1;
□Suppose that A1 wants to allow A3 to retrieve information from either of the two tables and also to be able to propagate the SELECT privilege to other accounts.
□A1 can issue the command:

GRANT SELECT ON EMPLOYEE, DEPARTMENT
TO A3 **WITH GRANT OPTION;**

Mandatory Access Control and Role-Based Access Control for Multilevel Security
□The discretionary access control techniques of granting and revoking privileges on relations have traditionally been the main security mechanism for relational database systems.
□This is an all-or-nothing method:
□A user either has or does not have a certain privilege.
□In many applications, and additional security policy is needed that classifies data and users based on security classes.
□This approach as mandatory access control would typically be combined with the discretionary access control mechanisms.

□Typical security classes are top secret (TS), secret (S), confidential (C), and unclassified (U), where TS is the highest level and U the lowest: TS ≥ S ≥ C ≥ U
□The commonly used model for multilevel security, known as the Bell-LaPadula model, classifies each subject (user, account, program) and object (relation, tuple, column, view, operation) into one of the security classifications, T, S, C, or U:
□Clearance (classification) of a subject S as class(S) and to the classification of an object O as class (O).
□Two restrictions are enforced on data access based on the subject/object classifications:
□Simple security property: A subject S is not allowed read access to an object O unless class(S) ≥ class (O).
□A subject S is not allowed to write an object O unless class(S) ≤ class(O). This known as the star property (or * property).

Role-based access control
□**I**ts basic notion is that permissions are associated with roles, and users are assigned to appropriate roles.
□Roles can be created using the CREATE ROLE and DESTROY ROLE commands.
□The GRANT and REVOKE commands discussed under DAC can then be used to assign and revoke privileges from roles.

Cryptography method to secure the database
□Encryption is a means of maintaining secure data in an insecure environment.
□Encryption consists of applying an encryption algorithm to data using some pre specified encryption key.
□The resulting data has to be decrypted using a decryption key to recover the original data.

Data and Advanced Encryption Standards (DES)
□DES can provide end-to-end encryption on the channel between the sender A and receiver B.
□DES algorithm is a careful and complex combination of two of the fundamental building blocksof encryption:
□Substitution and permutation (transposition).
□The DES algorithm derives its strength from repeated application of these two techniques for a total of 16 cycles.
□Plaintext (the original form of the message) is encrypted as blocks of 64 bits.

Public Key Encryption
□Public key algorithms are based on mathematical functions rather than operations on bit patterns.
□They also involve the use of two separate keys
□In contrast to conventional encryption, which uses only one key.
□The use of two keys can have profound consequences in the areas of confidentiality, key distribution, and authentication.
□The two keys used for public key encryption are referred to as the public key and the private key.

The essential steps are as follows:

□Each user generates a pair of keys to be used for the encryption and decryption of messages.

□Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private (private key).

□If a sender wishes to send a private message to a receiver, the sender encrypts the message using the receiver's public key.

□When the receiver receives the message, he or she decrypts it using the receiver's private key.

□No other recipient can decrypt the message because only the receiver knows his or her private key.

Digital Signatures:

□A digital signature is an example of using encryption techniques to provide authentication services in e-commerce applications.

□A digital signature is a means of associating a mark unique to an individual with a body of text.

□The mark should be unforgettable, meaning that others should be able to check that the signature does come from the originator.

□A digital signature consists of a string of symbols.

□Signature must be different for each use.

□This can be achieved by making each digital signature a function of the message that it is signing, together with a time stamp.

□Public key techniques are the means creating digital signatures

21.Suppose an Object Oriented database had an object A, which references object B, which in turn references object C. Assume all objects are on disk initially? Suppose a program first dereferences A, then dereferences B by following the reference from A, and then finally dereferences C. Show the objects that are represented in memory after each dereference, along with their state. (Nov'2015)

- For tuples of relations = "row types."
- For columns of relations = "types."
  - But row types can also be used as column types.

References :

- Row types can have *references*.
- If *T* is a row type, then REF(*T*) is the type of a reference to a *T* object.

```
CREATE ROW TYPE A (
   name CHAR(20) UNIQUE,
   addr CHAR(20) );

CREATE ROW TYPE B(
   name CHAR(20) UNIQUE,
   manf CHAR(20) );
```

```
CREATE ROW TYPE C (
   a REF(A),
   b REF(B),
   price FLOAT);
```

- Row-type declarations do not create tables.
  - They are used in place of element lists in CREATE TABLE statements.
- Example
  - CREATE TABLE a1 OF TYPE A
  - CREATE TABLE b1 OF TYPE B
  - CREATE TABLE c1 OF TYPE B

Dereferencing:

- A -> B = the B attribute of the object referred to by reference A.
- Example
  - Find the B served by Joe.

```
SELECT b -> name
FROM c1
WHERE c -> name = 'Joe';
```

22.Describe the GRANT functions and explain how it relates to security. What types of privileges may be granted? How rights could be revoked? (Nov/Dec 2015)

Granting and **revoking privileges**

- 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 level** (or **table level**):

- At this level, the DBA can control the privilege to access each individual relation or view in the database.

Propagation of Privileges using the GRANT OPTION

- Whenever the owner A of a relation R grants a privilege on R to another account B, privilege can be given to B with or without the GRANT OPTION.
- If the GRANT OPTION is given, this means that B can also grant that privilege on R to other accounts.
  - Suppose that B is given the GRANT OPTION by A and that B then grants the privilege on R to a third account C, also with GRANT OPTION. In this way, privileges on R can propagate to other accounts without the knowledge of the owner of R.
  - If the owner account A now revokes the privilege granted to B, all the privileges that B propagated based on that privilege should automatically be revoked by the system.

- Suppose that the DBA creates four accounts
  - A1, A2, A3, A4
- and wants only A1 to be able to create base relations. Then the DBA must issue the following GRANT command in SQL

  GRANT CREATETAB TO A1;
- In SQL2 the same effect can be accomplished by having the DBA issue a **CREATE SCHEMA** command as follows:

  CREATE SCHAMA EXAMPLE **AUTHORIZATION** A1;
- User account A1 can create tables under the schema called **EXAMPLE**.
- Suppose that A1 **creates** the two base relations **EMPLOYEE** and **DEPARTMENT**
  - A1 is then **owner** of these two relations and hence all the relation privileges on each of them.
- Suppose that A1 wants to grant A2 the privilege to insert and delete tuples in both of these relations, but A1 does not want A2 to be able to propagate these privileges to additional accounts:

  GRANT INSERT, DELETE ON EMPLOYEE, DEPARTMENT **TO A2;**
- Suppose that A1 wants to allow A3 to retrieve information from either of the two tables and also to be able to propagate the SELECT privilege to other accounts.
- A1 can issue the command:

  GRANT SELECT ON EMPLOYEE, DEPARTMENT **TO** A3 **WITH GRANT OPTION;**
- Suppose that A1 decides to revoke the SELECT privilege on the EMPLOYEE relation from A3; A1 can issue.

23.Illustrate of usage of OQL, the DMG's query language? (Nov 2019)

Object Query Language:

Object Query Language (OQL) is a query language standard for object-oriented databases modeled after SQL. OQL was developed by the Object Data Management Group (ODMG). Because of its overall complexity nobody has ever fully implemented the complete OQL. OQL has influenced the design of some of the newer query languages like JDOQL and EJB QL, but they can't be considered as different flavors of OQL.

The following rules apply to OQL statements:

- All complete statements must be terminated by a semi-colon.
- A list of entries in OQL is usually separated by commas but not terminated by a comma(,).
- Strings of text are enclosed by matching quotation marks.

Simple query:

The following example illustrates how one might retrieve the database:

```
SELECT pc.cpuspeed
FROM PCs pc
WHERE pc.ram > 64;
```

Query with grouping and aggregation:

The following example illustrates how one might retrieve the average amount:

```
SELECT manufacturer, AVG(SELECT part.pc.ram FROM partition part)
FROM PCs pc
GROUP BY manufacturer: pc.manufacturer;
```

Note the use of the keyword `partition`, as opposed to aggregation in traditional SQL

Data Mining Query Language (DMQL):

The Data Mining Query Language (DMQL) was proposed by Han, Fu, Wang, et al. for the DBMiner data mining system. The Data Mining Query Language is actually based on the Structured Query Language (SQL). Data Mining Query Languages can be designed to support ad hoc and interactive data mining. This DMQL provides commands for specifying primitives. The DMQL can work with databases and data warehouses as well. DMQL can be used to define data mining tasks. Particularly we examine how to define data warehouses and data marts in DMQL.

Syntax for Task-Relevant Data Specification

Here is the syntax of DMQL for specifying task-relevant data −

use database database_name

or

use data warehouse data_warehouse_name
in relevance to att_or_dim_list
from relation(s)/cube(s) [where condition]
order by order_list
group by grouping_list

Syntax for Specifying the Kind of Knowledge

Here we will discuss the syntax for Characterization, Discrimination, Association, Classification, and Prediction.

Characterization

The syntax for characterization is −

mine characteristics [as pattern_name]
analyze {measure(s) }

The analyze clause, specifies aggregate measures, such as count, sum, or count%.

For example −

Description describing customer purchasing habits.
mine characteristics as customerPurchasing
analyze count%

Discrimination

The syntax for Discrimination is −

mine comparison [as {pattern_name]}
For {target_class } where {t arget_condition }
{versus {contrast_class_i }
where {contrast_condition_i}}
analyze {measure(s) }

For example, a user may define big spenders as customers who purchase items that cost $100 or more on an average; and budget spenders as customers who purchase items at less than $100 on an average. The mining of discriminant descriptions for customers from each of these categories can be specified in the DMQL as −

mine comparison as purchaseGroups
for bigSpenders where avg(I.price) ≥$100
versus budgetSpenders where avg(I.price)< $100
analyze count

Association

The syntax for Association is−

mine associations [as {pattern_name}]
{matching {metapattern} }

For Example −

mine associations as buyingHabits
matching P(X:customer,W) ^ Q(X,Y) ≥ buys(X,Z)

where X is key of customer relation; P and Q are predicate variables; and W, Y, and Z are object variables.

Classification

The syntax for Classification is −

mine classification [as pattern_name]
analyze classifying_attribute_or_dimension

For example, to mine patterns, classifying customer credit rating where the classes are determined by the attribute credit_rating, and mine classification is determined as classifyCustomerCreditRating.

analyze credit_rating

Prediction

The syntax for prediction is −

mine prediction [as pattern_name]
analyze prediction_attribute_or_dimension
{set {attribute_or_dimension_i= value_i}}

Syntax for Concept Hierarchy Specification

To specify concept hierarchies, use the following syntax −

use hierarchy <hierarchy> for <attribute_or_dimension>

We use different syntaxes to define different types of hierarchies such as−

-schema hierarchies
define hierarchy time_hierarchy on date as [date,month quarter,year]

-set-grouping hierarchies
define hierarchy age_hierarchy for age on customer as
level1: {young, middle_aged, senior} < level0: all
level2: {20, ..., 39} < level1: young
level3: {40, ..., 59} < level1: middle_aged
level4: {60, ..., 89} < level1: senior

-operation-derived hierarchies
define hierarchy age_hierarchy for age on customer as
{age_category(1), ..., age_category(5)}

:= cluster(default, age, 5) < all(age)

-rule-based hierarchies
define hierarchy profit_margin_hierarchy on item as
level_1: low_profit_margin < level_0: all

if (price - cost)< $50
level_1: medium-profit_margin < level_0: all

if ((price - cost) > $50) and ((price - cost) ≤ $250))
level_1: high_profit_margin < level_0: all

Syntax for Interestingness Measures Specification

Interestingness measures and thresholds can be specified by the user with the statement −

with <interest_measure_name> threshold = threshold_value

For Example −

with support threshold = 0.05
with confidence threshold = 0.7

Syntax for Pattern Presentation and Visualization Specification

We have a syntax, which allows users to specify the display of discovered patterns in one or more forms.

display as <result_form>

For Example −

display as table

Full Specification of DMQL

As a market manager of a company, you would like to characterize the buying habits of customers who can purchase items priced at no less than $100; with respect to the customer's age, type of item purchased, and the place where the item was purchased. You would like to know the percentage of customers having that characteristic. In particular, you are only interested in purchases made in Canada, and paid with an American Express credit card. You would like to view the resulting descriptions in the form of a table.

use database AllElectronics_db
use hierarchy location_hierarchy for B.address
mine characteristics as customerPurchasing
analyze count%
in relevance to C.age,I.type,I.place_made
from customer C, item I, purchase P, items_sold S, branch B
where I.item_ID = S.item_ID and P.cust_ID = C.cust_ID and
P.method_paid = "AmEx" and B.address = "Canada" and I.price ≥ 100
with noise threshold = 5%
display as table

Data Mining Languages Standardization

Standardizing the Data Mining Languages will serve the following purposes −

- Helps systematic development of data mining solutions.
- Improves interoperability among multiple data mining systems and functions.
- Promotes education and rapid learning.
- Promotes the use of data mining systems in industry and society.