KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed University Established Under Section 3 of UGC Act 1956)
Coimbatore - 641021.
(For the candidates admitted from 2018 onwards)
DEPARTMENT OF COMMERCE COMPUTER APPLICATION

DATABASE MANAGEMENT SYSTEM [18CCU302]

UNIT I INTRODUCTION TO DBMS

File Systems Organization - Sequential, Pointer, Indexed, Direct - Purpose of Database System- Database System Terminologies-Database characteristics- Data models – Types of data models – Components of DBMS- Relational Algebra. LOGICAL DATABASE DESIGN: Relational DBMS - Codd's Rule - Entity-Relationship model - Extended ER Normalization – Functional Dependencies, Anomaly- 1NF to 5NF- Domain Key Normal Form – Denormalization

UNIT II SQL & QUERY OPTIMIZATION

SQL Standards - Data types - Database Objects- DDL-DML-DCL-TCL-Embedded SQL-Static Vs Dynamic SQL - QUERY OPTIMIZATION: Query Processing and Optimization - Heuristics and Cost Estimates in Query Optimization.

UNIT III TRANSACTION PROCESSING AND CONCURRENCY CONTROL

Introduction-Properties of Transaction- Serializability- Concurrency Control – Locking Mechanisms- Two Phase Commit Protocol-Dead lock.

UNIT IV TRENDS IN DATABASE TECHNOLOGY

Overview of Physical Storage Media – Magnetic Disks – RAID – Tertiary storage – File Organization – Organization of Records in Files – Indexing and Hashing –Ordered Indices – B+ tree Index Files – B tree Index Files – Static Hashing – Dynamic Hashing - Introduction to Distributed Databases- Client server technology- Multidimensional and Parallel databases- Spatial and multimedia databases- Mobile and web databases- Data Warehouse-Mining- Data marts.

UNIT V ADVANCED TOPICS

DATABASE SECURITY: Data Classification-Threats and risks – Database access Control – Types of Privileges –Cryptography- Statistical Databases.- Distributed Databases-Architecture-Transaction Processing-Data Warehousing and Mining-Classification-Association rules-Clustering-Information Retrieval- Relevance ranking-Crawling and Indexing the Web- Object Oriented Databases-XML Databases.

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University Established Under Section 3 of UGC Act 1956)

Coimbatore – 641 021.

LECTURE PLAN

DEPARTMENT Of COMMERCE WITH COMPUTER APPLICATION

STAFF NAME: Ms.R.Lydia Priyadharsini/Mr.Jothish
SEMESTER:III
SUBJECT NAME: DATABASE MANAGEMENT SYSTEM
SUB.CODE: 18CCU302 CLASS: II B.Com CA

Unit - I			
S. No.	**Lecture Duration (Hr)**	**Topics to be covered**	**Support Material**
1.	1	File System Organization ➢ Sequential ➢ Pointer ➢ Indexed ➢ Direct	T1: Pg.No.: 2-4
2.	1	➢ Purpose of Database System ➢ Database system Terminologies ➢ Database Characteristics	T1: Pg.No.:16-18
3.	1	➢ Data Models ➢ Types of Data Models ➢ Components of DBMS	T1: Pg.No.: 23-28
4.	1	➢ Relational Algebra	T1: Pg.No.: 30-34
5.	1	Logical Database Design : ➢ Relational DBMS ➢ Codd's Rule	T1: Pg.No.:34-36
6.	1	Entity Relationship Model	T1: Pg.No.:36-37
7.	1	Extended ER Normalization	T1: Pg.No.:34-36
8.	1	Functional Dependencies Anomaly	T1: Pg.No.:36-37
9.	1	1 NF to 5 NF	T1: Pg.No.: 23-28
10.	1	➢ Domain Key Normal form ➢ Denormalization	T1: Pg.No.: 63-64

11.	1	Recapitulation And Important Questions Discussion	
Total Number of Hours Planned for Unit – I			**11**

Unit - II			
S. No.	**Lecture Duration (Hr)**	**Topics to be covered**	**Support Material**
1	1	SQL standards Datatypes	T1: Pg.No.: 64-71
2	1	Database Objects	T1:Pg.No.:71 w1,w2
3	1	➢ DDL ➢ DML ➢ DCL ➢ TCL	T1: Pg.No.:71-86, W2
4	1	Embedded SQL-Static Vs Dynamic SQL	W2
5		QUERY OPTIMIZATION: Query Processing and Optimization	T1: Pg.No.:90-94
6	1	Heuristics and Cost Estimates in Query Optimization.	T1: Pg.No.:90-94
7	1	Recapitulation And Important Questions Discussion	
Total Number of Hours Planned for Unit – II			**7**
Unit - III			
S. No.	**Lecture Duration (Hr)**	**Topics to be covered**	**Support Material**
1.	1	➢ Introduction ➢ Properties of Transaction	T1: Pg.No.:111-113
2	1	Serializability	W1
3		Concurrency Control	T1: Pg.No.:125-129
4	1	Locking Mechanisms	T1: Pg.No.:131-136
5	1	Two Phase Commit Protocol	T1: Pg.No.:136-146
6	1	Dead lock.	T1: Pg.No.:140-145

7	1	Recapitulation And Important Questions Discussion	T1: Pg.No.:145-148
Total Number of Hours Planned for Unit – III			**7**

Unit – IV			
S. No.	**Lecture Duration (Hr)**	**Topics to be covered**	**Support Material**
1.	1	➢ Overview of Physical Storage Media ➢ Magnetic Disks	T1: Pg.No.:215
2	1	➢ RAID ➢ Tertiary storage	T1: Pg.No.:215-217
3	1	➢ File Organization ➢ Organization of Records in Files	W2
4	1	➢ Indexing and Hashing ➢ Ordered Indices	W1
5	1	➢ B+ tree Index Files ➢ B tree Index Files	T1:Pg.No.: T1:228-231, w1
6	1	➢ Static Hashing ➢ Dynamic Hashing	W1
7	1	➢ Introduction to Distributed Databases	W2
8	1	➢ Client server technology ➢ Multidimensional and Parallel databases	W1
9	1	➢ Spatial and multimedia databases	T1: Pg.No.:215
9	1	Mobile and web databases	T1: Pg.No.:215-217
10	1	Data Warehouse-Mining- Data marts.	W2
11	1	Recapitulation And Important Questions Discussion	W1
Total Number of Hours Planned for Unit – IV			**11**

Unit – V			
S. No.	**Lecture Duration (Hr)**	**Topics to be covered**	**Supporrt Material**
1	1	➢ DATABASE SECURITY: ➢ Data Classification ➢ Threats and risks	R2:Pg.No.: 3-15
2	1	➢ Database access Control ➢ Types of Privileges	R2:Pg.No.: 184-186
3	1	➢ Cryptography	W3
4	1	➢ Statistical Databases ➢ Distributed Databases	R2:Pg.No.: 227-235
5	1	➢ Architecture ➢ Transaction Processing	W3,W2 ,W4
6	1	➢ Data Warehousing and Mining	W4
7	1	➢ Classification ➢ Association rules ➢ Clustering	R2:Pg.No.: 227-235
8	1	➢ Information Retrieval ➢ Relevance ranking	W3,W2 ,W4
9	1	➢ Crawling and Indexing the Web	W4
10	1	➢ Object Oriented Databases ➢ XML Databases.	W4
11	1	Recapitulation And Important Questions Discussion	
12	1	Discussion of Previous Year ESE Question Papers	
Total Number of Hours Planned for Unit – V			**12**

SUGGESTED READINGS

1. Rajiv Chopra (2016), "Database Management Systems (DBMS)", 5TH Edition, S.Chand, New Delhi.
2. Nilesh Shah, (2015),"*Database Systems Using Oracle***"**, 2nd Edition, Pearson Education, New Delhi.
3. Raghu Ramakrishnan, Johannes Gehrke, (2014), "*Database Management Systems*", 3rd Edition, McGraw Hill Education (India) Edition, New Delhi.
4. Abraham Silberschatz, Henry F. Korth and S. Sudharshan (2011), "*Database System Concepts*", Sixth Edition, Tata Mc Graw Hill.
5. G.K.Gupta, (2011), "*Database Management Systems*", Tata Mc Graw Hill, New Delhi.

KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BCom CA COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 18CCU302 UNIT: I BATCH-2018-21

UNIT-I

SYLLABUS

UNIT I - INTRODUCTION TO DBMS

File Systems Organization - Sequential, Pointer, Indexed, Direct - Purpose of Database System- Database System Terminologies-Database characteristics- Data models – Types of data models – Components of DBMS- Relational Algebra. LOGICAL DATABASE DESIGN: Relational DBMS - Codd's Rule - Entity-Relationship model - Extended ER Normalization – Functional Dependencies, Anomaly- 1NF to 5NF- Domain Key Normal Form – Denormalization

1. -Sequential, Pointer, Indexed, Direct

A file is a collection or set (ordered or unordered) of data elements stored on storage media. A system software module is responsible for managing (reading, processing, deleting, etc.) a file.

Logical Structure of a file

- **Field**: Smallest (fixed) indivisible logical unit of a file. A field holds a part of some data value. **Record**: A set of logically related fields. Cardinality of this set may be fixed or variable, i.e., a record size may be fixed or variable. A file, therefore, is a collection of logically related records.

A Field

A Record

SSN	Name	Age	Phone #

A File

SSN	Name	Age	Phone #

A record can be of fixed or variable size. Most commonly fixed size records are used. The file structure has been illustrated here in a tabular form but note that a file is not just a table its records can be organized in many different ways each way defines a unique organization.

Operations on file Create: Create a new file.

- **Write**: Write to a file
- **Read**: Read a part of a file
- **Rewind**: Go to the beginning of the file
- **Delete**: Remove a file from the system.
- **Close**: Close a file (at the end of manipulation).
- **Open**: Open a file for processing.
- **Modify**: Modify a field value.

There are basically three categories of file organizations (a) Sequential organization, (b) organization using Index, and (c) Random organization. We begin with sequential category. Files of these categories are called sequential files.

Sequential file

In this organization records are written consecutively when the file is created. Records in a sequential file can be stored in two ways. Each way identifies a file organization.

- **Pile file**: Records are placed one after another as they arrive (no sorting of any kind).
- **Sorted file**: Records are placed in ascending or descending values of the primary key.

File Reorganization:

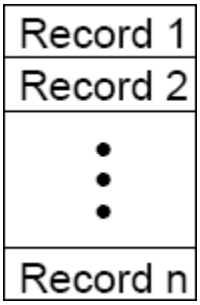
In file reorganization all records, which are marked to be deleted are deleted and all inserted records are moved to their correct place (sorting). File reorganization steps are:

- read the entire file (all blocks) in RAM.
- remove all the deleted records.
- write all the modified blocks at a different place on the disk.

Inserting a record: To insert a record, it is placed at the end of the file. No need to sort (ascending or descending order) the file. However, the file may have duplicate records.

Deleting or modifying a record: This will require fetching the block containing the record, finding the record in the block and just marking it deleted, and then writing the modified block to the disk.

Sorted Sequential File: In a sorted file, first the record is inserted at the end of the file and then moved to its correct location (ascending or descending). Records are stored in order of the values of the key field.



A sequential file organization

A sequential file usually has an overflow area. This area is to avoid sorting the file after every deletion, insertion and/or modification. All records, which were added after the file was first populated, go to this overflow area. The overflow area is not itself sorted it is a pile file with fixed size record. At some convenient time the entire file is sorted where records from the overflow area go to their correct position in the main file.

Retrieval: Records are retrieved in a consecutive order. The order of record storage determines order of retrieval. During retrieval several required operations (partial result output etc.) can be performed simultaneously.

Insert, delete and modify (Update): Sequential files are usually not updated in place. Each operation regenerates a new version of the file. The new version contains the up-to-date information and the old version is usually saved for recovery. The new version becomes the

old

version and the next update to the file uses this old version to generate next new version. The intensity or frequency of use of a sequential file is defined by a parameter called ―*Hit ratio*‖, which defines is defined as follows:

$$Hit\ ratio = \frac{No.of\ records\ accessed\ \ for\ responding\ to\ a\ query}{Total\ number\ of\ records\ in\ the\ file}.$$

Desirable: high hit ratio value. This means a larger number of records are accessed to respond to a query. Interactive transactions have very low hit ratio.

Advantages of sequential file
• Good for batch transactions.
• Simple to implement
• Good for report generation, statistical computation and inventory control.

Disadvantages
• Not good for interactive transactions
• High overheads in file processing for simple queries.

Index File Organization
Index organization tries to reduce access time. It may not reduce the storage requirement of a file. Some important terms of this organization are.

Index: An index is similar to a pointer with an additional property. This additional property allows an index to identify the record it is pointing to. For example, an index to a record of employees points and identifies an employee record. This means that an index has some semantics associated with it. Formally, an index maps the *key space* to the *record space*. Index for a file may be created on the *primary* or *secondary* keys.

Index types: Primary, Non dense, and Dense.
Primary Index: An ordered file of index record. This file contains index records which are of fixed length. Each index record has two fields:
• one field holds the primary key of the data file record.
• the other holds pointer to disk block where the data file is stored.
Nondense index: No. of entries in the index file << no. of records in the data file.
Dense index: No. of entries in the index file = no. of records in the data file.
Example: How record search time is reduced by indexing.

Index Sequential File (ISAM - Index Sequential Access Method)
Introduced by IBM in early 1960s. This file organization closely related to the physical characteristics of the storage media. It has two parts: (a) Index Part and (b) Data Part. Index Part: This part stores pointers to the actual record location on the disk. It may have several levels and each level represents some physical layout such as sector, cylinder, of the storage media. Data Part: It holds actual data records and is made up of two distinct areas: Prime area and Overflow area. The prime area holds records of the file. The overflow area holds records when the prime area overflows.

Direct File Indexing provides the most powerful and flexible tools for organizing files. However, (a) indexes take up space and time to keep them organized and (b) the amount of I/O increases with the size of index. This problem can be minimized by direct file organization where the address of the desired record can be found directly (no need of indexing or sequential search). Such file are created using some *hashing* function so they are called *hashing organization* or *hashed files*.

Hashing Two types of hashing: (a) Static and (b) Dynamic.

- **Static**: The address space size is predefined and does not grow or shrink with file.
- **Dynamic**: The address space size can grow and shrink with file.

2. Purpose of Database System:

Database management systems were developed to handle the following difficulties of typical file-processing systems supported by conventional operating systems.

- Data redundancy and Consistency
- Self Describing Nature (of Database System)
- Data isolation or Abstraction
- Integrity
- Atomicity
- Concurrent Access or sharing of Data
- Security
- Support for multiple views of Data

(i) Data redundancy and Consistency:

In file System, each user maintains separate files and programs to manipulate these files because each requires some data not available from other user‘s files. This redundancy in defining and storage of data results in

- wasted storage space,
- redundant efforts to maintain common update,
- higher storage and access cost and
- Leads to inconsistency of data (ie.,) various copies of same data may not agree.

In Database approach, a single repository of data is maintained that is maintained that is defined once and then accessed by various users. Thus redundancy is controlled and it is consistent.

***(ii)* Self Describing Nature of Database System**:

In File System, the structure of the data file is embedded in the access programs. A database system contains not only the database itself but also a complete definition or description of database structure and constraints. This definition is stored in System catalog which contains information such as structure of each file, type and storage format of each data item and various constraints on the data. Information stored in the catalog is called Meta-Data. DBMS is not written for specific applications, hence it must refer to catalog to know structure of file etc., and hence it can work equally well with any number of database applications.

***(iii)* Data Isolation or Abstraction:**

Conventional File processing systems do not allow data to be retrieved in convenient and efficient manner. More responsive data retrieval systems are required for general use. The structure of the data file is embedded in the access programs. So any changes to the structure of a file may require changing all programs that access this file. Because data are scattered

in various files and files may be in different formats, writing new application programs to retrieve

appropriate data is difficult. But the DBMS access programs do not require such changes in most cases. The structure of data files is stored in DBMS catalog separately from the access programs. This property is known as program data independence. Operations are separately specified and can be changed without affecting the interface. User application programs can operate on data by invoking these operations regardless of how they are implemented. This property is known as program operation independence. Program data independence and program operation independence are together known as data independence.

***(iv)* Enforcing Integrity Constraints**:

The data values must satisfy certain types of consistency constraints. In File System, Developers enforce constraints by adding appropriate code in application program. When new constraints are added, it is difficult to change the programs to enforce them.

In data base system, DBMS provide capabilities for defining and enforcing constraints. The constraints are maintained in system catalog. Therefore application programs work independently with addition or modification of constraints. Hence integrity problems are avoided.

***(v)* Atomicity:**

A Computer system is subjected to failure. If failure occurs, the data has to be restored to the consistent state that existed prior to failure. The transactions must be atomic – it must happen in entirety or not at all. It is difficult to ensure atomicity in File processing System. In DB approach, the DBMS ensures atomicity using the Transaction manager inbuilt in it. DBMS supports online transaction processing and recovery techniques to maintain atomicity.

***(vi)* Concurrent Access or sharing of Data:**

When multiple users update the data simultaneously, it may result in inconsistent data. The system must maintain supervision which is difficult because data may be accessed by many different application programs that may have not been coordinated previously. The database (DBMS) include concurrency control software to ensure that several programs /users trying to update the same data do so in controlled manner, so that the result of update is correct. (vii) Security: Every user of the database system should not be able to access the data. But since the application programs are added to the system in an adhoc manner, enforcing such security constraints is difficult in file system. DBMS provide security and authorization subsystem, which the DBA uses to create accounts and to specify account restrictions.

(viii) Support for multiple views of Data: Database approach support multiple views of data. A database has many users each of whom may require a different view of the database. View may be a subset of database or virtual data retrieved from database which is not explicitly stored. DBMS provide multiple views of the data or DB.Different application programs are to be written for different views of data.

3. Database System Terminologies:

a) File system:

A file system is a method for storing and organizing computer files and the data they contain to make it easy to find and access them. It is controlled by Operating system.

b) Data:

Data are Known facts that can be recorded and that have implicit meaning

c) Database: Database is a collection of related data with some inherent meaning. It is designed built and populated with data for specific purpose. It represents some aspects of real world.

d) Database Management System: It is a collection of programs that enables to create and maintain database. It is a general purpose Software system that facilitates the process of defining, constructing and manipulating data bases for various applications.

Defining involves specifying data types, structures and constraints for the data to be stored in the database.

Constructing is the process of storing the database itself on some storage medium that is controlled by DBMS.

Manipulating includes functions such as querying the database to retrieve specific data, updating database to reflect changes and generating reports from the data. **Eg.** Oracle, Ms-access, Sybase, Informix, Foxpro.

STUDENT	Name	StudentNumber	Class	Major
	Smith	17	1	CS
	Brown	8	2	CS

COURSE	CourseName	CourseNumber	CreditHours	Department
	Intro to Computer Science	CS1310	4	CS
	Data Structures	CS3320	4	CS
	Discrete Mathematics	MATH2410	3	MATH
	Database	CS3380	3	CS

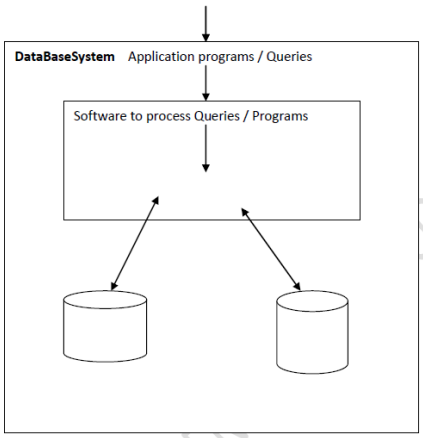
SECTION	SectionIdentifier	CourseNumber	Semester	Year	Instructor
	85	MATH2410	Fall	98	King
	92	CS1310	Fall	98	Anderson
	102	CS3320	Spring	99	Knuth
	112	MATH2410	Fall	98	Chang
	119	CS1310	Fall	98	Anderson
	135	CS3380	Fall	98	Stone

GRADE_REPORT	StudentNumber	SectionIdentifier	Grade
	17	112	B
	17	119	C
	8	85	A
	8	92	A
	8	102	B
	8	135	A

Example for a database - A University Database

e) Database System: Database and DBMS together known as Database system.

User / programmers



Database System Environment

f) Applications of Database System:

- Banking – Customer information, accounts, loans and transactions
- Airlines – Reservation and schedule information

- Universities – Student, course, registration and grades
- Credit and Transactions – purchases on cards and report
- Telecommunications – Records of calls, report, balances etc.,
- Finance – Holdings, sales, stocks and bond purchase.
- Sales – customer, product, purchase.
- Manufacturing – Inventory items, orders, purchase.
- Human Resources-Employee details, salary, tax etc.,

4. Database Characteristics:

- Self-Describing Nature of a Database System
- Insulation between Programs and Data, and Data Abstraction
- Support of Multiple Views of the Data
- Sharing of Data and Multiuser Transaction Processing

Self-Describing Nature of a Database System

A fundamental characteristic of the database approach is that the database system contains not only the database itself but also a complete definition or description of the database structure and constraints. This definition is stored in the system catalog, which contains information such as the structure of each file, the type and storage format of each data item, and various constraints on the data. The information stored in the catalog is called meta-data, and it describes the structure of the primary database. The catalog is used by the DBMS software and also by database users who need information about the database structure. A general purpose DBMS software package is not written for a specific database application, and hence it must refer to the catalog to know the structure of the files in a specific database, such as the type and format of data it will access. The DBMS software must work equally well with *any number of database applications*—for example, a university database, a banking database, or a company database—as long as the database definition is stored in the catalog. DBMS software can access diverse databases by extracting the database definitions from the catalog and then using these definitions. Whenever a request is made to access, say, the Name of a STUDENT record, the DBMS software refers to the catalog to determine the structure of the STUDENT file and the position and size of the Name data item within a STUDENT record.

Insulation between Programs and Data, and Data Abstraction

The structure of data files is stored in the DBMS catalog separately from the access programs. We call this property *program-data independence*. In a DBMS environment, we just need to change the description of STUDENT records in the catalog to reflect the inclusion of the new data item Birthdate; no programs are changed. The next time a DBMS program refers to the catalog, the new structure of STUDENT records will be accessed and used. An operation (also called a *function*) is specified in two parts. The *interface* (or *signature*) of an operation includes the operation name and the data types of its arguments (or parameters). The *implementation* (or *method*) of the operation is specified separately and can be changed without affecting the interface. User application programs can operate on the data by invoking these operations through their names and arguments, regardless of how the operations are implemented. This may be termed *program-operation* independence. The characteristic that allows program-data independence and program-operation independence is called *data abstraction*. A DBMS provides users with a conceptual representation of data that does not include many of the details of how the data is stored or how the operations are

implemented. The data model uses logical concepts, such as objects, their properties, and their interrelationships, that may be easier for

most users to understand than computer storage concepts. Hence, the data model *hides* storage and implementation details that are not of interest to most database users.

Support of Multiple Views of the Data

A database typically has many users, each of whom may require a different perspective or view of the database. A *view* may be a subset of the database or it may contain virtual data that is derived from the database files but is not explicitly stored. Some users may not need to be aware of whether the data they refer to is stored or derived. A multiuser DBMS whose users have a variety of applications must provide facilities for defining multiple views. For example, one user of the database may be interested only in the transcript of each student; the view for this user is displayed. A second user, who is interested only in checking that students have taken all the prerequisites of each course they register for, may require the different view.

Sharing of Data and Multiuser Transaction Processing

A multiuser DBMS must allow multiple users to access the database at the same time. This is essential if data for multiple applications is to be integrated and maintained in a single database. The DBMS must include concurrency control software to ensure that several users trying to update the same data do so in a controlled manner so that the result of the updates is correct. For example, when several reservation clerks try to assign a seat on an airline flight, the DBMS should ensure that each seat can be accessed by only one clerk at a time for assignment to a passenger. These types of applications are generally called *on-line transaction processing* (OLTP) applications. A fundamental role of multiuser DBMS software is to ensure that concurrent transactions operate correctly.

5. Data Models

Definitions:

a) Data Model: Data model is a collection of concepts that can be used to describe the structure of a database (ie., Data, Relationships, Data types and constraints).

b) Schema: Complete definition and description of the database is known as database schema. Each object in the schema is known as schema construct. It is known as Intension.

c) Data Base State: The data in the database at a particular moment in time is called a database state or snapshot. It is known as the extension of database schema.

DBMS restores the description of the schema constructs and constraints (meta data) in DBMS Catalog.

Types of Data model:

- High level or Conceptual: Close to users
- Low level or Physical: how data is stored in computer
- Representational or Implementational: Concepts understood by users and not too far from the way data is organized Eg. Network, Hierarchical Model.

DBMS Architecture:

- Three schema architecture

- Achieve the database characteristics

Three Schema Architecture: Separates the user applications and physical database. Schemas can be defined in three levels:

(i) Internal Level:

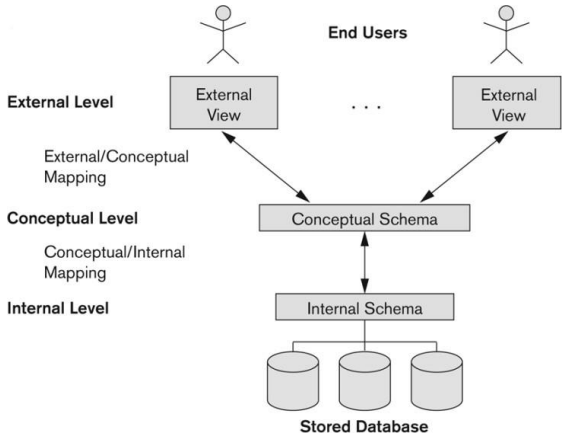
- It has an internal schema which describes physical storage structure of the database.
- How the data are actually stored
- Uses physical model
- Describes the complete details of data storage and access paths for the database.

(ii) Conceptual Level:

- It has an conceptual schema which describes the structure of whole database
- What data are stored and what relationships exist among data.
- Uses high level or implementational data model.
- Hides the details of physical storage structures and describes datatypes, relationships, operations and constraints.

(iii) External or View Level:

- Includes a number of external schemas or views.
- Each external schema describes the part of the database and hides the rest.
- Uses high level or implementational data model.



DBMS architecture

- **Data Independence:** The capacity to change the schema at one level of database system without having to change the schema at the next higher level.
- **Logical Data independence:** The capacity to change the conceptual schema without having to change the External schema.
- **Physical Data Independence:** The capacity to change the physical schema without having to change the Conceptual schema.

Data base Application Architecture:

- Client : Machine in which remote database users work
- Server: machine in which database system runs.

Two-tier Architecture:

- Application is partitioned into component that resides at the client machine, which invokes database system functionality at server machine through query language statements.

- Eg. API – ODBC & JDBC used for Interaction.

Three tier Architecture:

- The client merely acts as a front end and does not contain any direct database calls.
- Client end communicates with application server usually through form interface.
- Application server (has Business logic) communicates with database system.

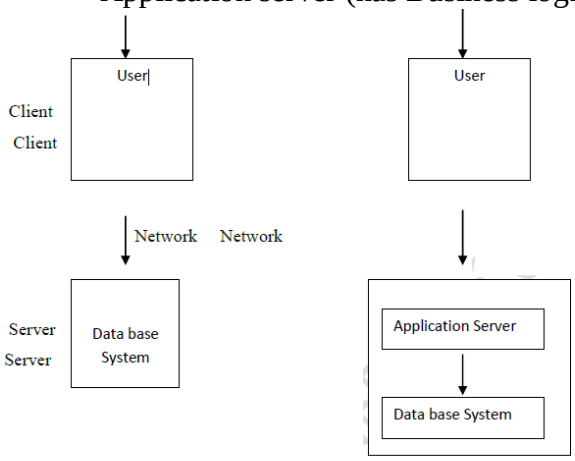
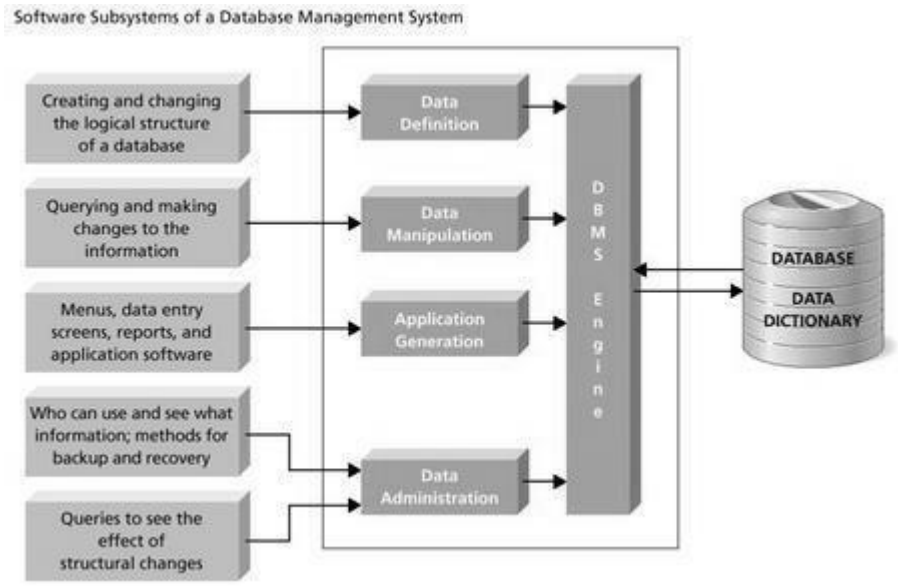


Fig 1.3 Two – Tier and Three Tier Architecture

Components of DBMS

DBMSs are the technology tools that directly support managing organizational data. With a DBMS you can create a database including its logical structure and constraints, you can manipulate the data and information it contains, or you can directly create a simple database application or reporting tool. Business applications, which perform the higher level tasks of managing business processes, interact with end users and other applications and, to store and manage data, rely on and directly operate their own underlying database through a standard programming interface like ODBC. The following diagram illustrates the five components of a DBMS.



Database Engine:

The Database Engine is the core service for storing, processing, and securing data. The Database Engine provides controlled access and rapid transaction processing to meet the requirements of the most demanding data consuming applications within your enterprise. Use the Database Engine to create relational databases for online transaction processing or online analytical processing data. This includes creating tables for storing data, and database objects such as indexes, views, and stored procedures for viewing, managing, and securing data. You can use SQL Server Management Studio to manage the database objects, and SQL Server Profiler for capturing server events.

Data dictionary:

A data dictionary is a reserved space within a database which is used to store information about the database itself. A data dictionary is a set of table and views which can only be read and never altered. Most data dictionaries contain different information about the data used in the enterprise. In terms of the database representation of the data, the data table defines all schema objects including views, tables, clusters, indexes, sequences, synonyms, procedures, packages, functions, triggers and many more. This will ensure that all these things follow one standard defined in the dictionary.

The data dictionary also defines how much space has been allocated for and / or currently in used by all the schema objects. A data dictionary is used when finding information about users, objects, and schema and storage structures. Every time a data definition language (DDL) statement is issued, the data dictionary becomes modified. A data dictionary may contain information such as:

- Database design information
- Stored SQL procedures
- User permissions
- User statistics
- Database process information
- Database growth statistics
- Database performance statistics

Query Processor:

A relational database consists of many parts, but at its heart are two major components: the storage engine and the query processor. The storage engine writes data to and reads data from the disk. It manages records, controls concurrency, and maintains log files.The query processor accepts SQL syntax, selects a plan for executing the syntax, and then executes the chosen plan. The user or program interacts with the query processor, and the query processor in turn interacts with the storage engine. The query processor isolates the user from the details of execution: The user specifies the result, and the query processor determines how this result is obtained. The query processor components include

- DDL interpreter
- DML compiler
- Query evaluation engine

Report writer/ generator:

Report writer is a program, usually part of a database management system that extracts information from one or more files and presents the information in a specified format. Most report writers allow you to select records that meet certain conditions and to display selected fields in rows and columns. You can also format data into pie charts, bar charts, and other

diagrams. Once you have created a format for a report, you can save the format specifications in a file and continue reusing it for new data.

7. Relational Algebra:

Relational algebra is a procedural query language, which consists of basic set of relational model operations. These operations enable the user to specify basic retrieval requests. It takes one or more relations as input and result of retrieval is a relation. Relational Algebra operations are divided into two groups

(i) Set operations
(ii) Fundamental operations (Relation based operations)

Relational Algebra Notations:

Operation	Keyword	Symbol
Set Operators (Binary)		
Union	UNION	∪
Intersection	INTERSECTION	∩
Cartesian product	X	×
Set Difference	MINUS	-

Operation	Keyword	Symbol
Other Operators		
Group or Aggregate	AGGREGATE	ℑ

Operation	Keyword	Symbol
Relation Based Operators		
Selection	SELECT	σ
Projection	PROJECT	π
Renaming	RENAME	ρ
Assignment	<-	←
Binary Relation Based Operators		
Join	JOIN	⋈
Left outer join	LEFT OUTER JOIN	⟕
Right outer join	RIGHT OUTER JOIN	⟖
Full outer join	FULL OUTER JOIN	⟗
Division	DIVIDE	%

Relational Algebra

A **query language** is a language in which user requests information from the database. it

can be categorized as either **procedural** or **nonprocedural.** In a procedural language the user instructs the system to do a sequence of operations on database to compute the desired result. In nonprocedural language the user describes the desired information without giving a specific procedure for obtaining that information.
The relational algebra is a procedural query language. It consists of a set of operations that take one or two relations as input and produces a new relation as output.

Fundamental Operations

- SELECT
- PROJECT
- UNION

- SET DIFFERENCE
- CARTESIAN

PRODUCT

- RENAME

Select and project operations are unary operation as they operate on a single relation. Union, set difference, Cartesian product and rename operations are binary operations as they operate on pairs of relations.

Other Operations

- SET INTERSECTION
- NATURAL JOIN
- DIVISION
- ASSIGNMENT

The select operation: - to identify a set of tuples which is a part of a relation and to extract only these tuples out. The select operation selects tuples that satisfy a given predicate or condition. It is a unary operation defined on a single relation.

- It is denoted as **σ**.

Consider the following table "Book" :- Code:

```
+--------+--------+-----------------+
| Acc-no | Yr-pub | title           |
+--------+--------+-----------------+
| 734216 | 1982   | Algorithm design |
| 237235 | 1995   | Database systems |
| 631523 | 1992   | Compiler design  |
| 543211 | 1991   | Programming      |
| 376112 | 1992   | Machine design   |
+--------+--------+-----------------+
```

Example1:- Select from the relation "Book" all the books whose year of publication is 1992.
Code:

```
σ Yr-pub=1992(Book)
```

Example2:- Select from the relation "Book" all the books whose Acc-no is greater than equal to 56782.
Code:

```
σ  Acc-no>=56782(Book)
```

The project operation: - returns its argument relation with certain attributes left out. It is a unary operation defined on a single relation

It is denoted as **Π**.

Example:- List all the Title and Acc-no of the "Book" relation. Code:

```
Π Acc-no, Title (Book)
```

The union operation: - is used when we need some attributes that appear in either or both of the two relations.

- It is denoted as **U**.

Example:
Borrower (customer-name, loan-number) Depositor (customer-name, account-number)
Customer (customer-name, street-number, customer-city)

List all the customers who have either an account or a loan or both
Code:

```
Π customer-name (Borrower) U Π customer-name (Depositor)
```

The set difference operation: - finds tuples in one relation but not in other.

- It is denoted as –

Example:
Find the names of all customers who have an account but not a loan. Code:

```
Π customer-name (Depositor) - Π customer-name (Borrower)
```

The Cartesian product operation: - allows combining information from two relations.

- It is denoted as **r X s** where r and s are relations.

Consider the following relation or table "r" :- Code:

```
+--------+-------+
| A    | B    |
+--------+-------+
|  a   |  1   |
|  b   |  2   |
|  a   |  2   |
+--------+-------+
```

Consider another relation or table "s" :-

Code:

B	**C**
3	1a
2	2b

Therefore, rXs
gives:- Code:

r.A	r.B	s.B	s.C

If relation r has n1 tuples and relation s has n2 tuples then **r X s** has n1*n2 tuples.

Example:
Borrower (customer-name, loan-number)
Loan (loan-number, branch-name, city, amount)

List the names of all customers who have a loan in "Perryridge"
branch Code:

```
Π customer-name (σ Borrower.loan-number=Loan.loan-number (σ branch-name="Perryridge" (Borrower X Loan)))
```

The rename operation: - used to rename. It is denoted as **ρ**.

E **:** relational algebra expression

ρ x (E): returns the result of expression E under the name x.
ρ x (A1, A2, A3… An) (E): returns the result of expression E under the name x with attributes renamed to A1, A2, A3… An.
The set intersection operation: - finds tuples in both the relations.

- It is denoted as ∩. Example:

Borrower (customer-name, loan-number) Depositor (customer-name, account-number)
Customer (customer-name, street-number, customer-city)

List all the customers who have both a loan and an

account. Code:

Π customer-name (Borrower) ∩ Π customer-name (Depositor)

The natural join operation: -
it is a binary operation and a combination of certain selections and a Cartesian product into one operation.

- It is denoted as |X| .
- It is associative.

It forms a Cartesian product of its two arguments.
Then performs a selection forcing equality on those attributes those appear in both the relations. And finally removes duplicates attributes.

r(R): r is a relation with attributes R. s(S): s is a relation with attributes S.

If R ∩ S = Φ i.e. they have no attributes in common then **r |X| s = r X s**

The division / quotient operation: -

- It is denoted as ÷.

Letr(R) and s(S) be relations

r ÷ s: - the result consists of the restrictions of tuples in r to the attribute names unique to R, i.e. in the Header of r but not in the Header of s, for which it holds that all their combinations with tuples in s are present in r.
Example:
Relation or table "r":- Code:

```
+--------+-------+
| A    | B    |
+--------+-------+




+--------+-------+
```

Relation or table "s":- Code:

```
+------+
| B  |
+------+
| 2  |
| 3  |
+------+
```

Therefore, r ÷ s

Code:

```
+------+
| A |
+------+
| b  |
| a  |
| p  |
+------+
```

7. Logical Database Design: Relational DBMS - Codd's Rule

A **relational database management system (RDBMS)** is a database management system (DBMS) that is based on the relational model as introduced by *E. F. Codd*. Most popular commercial and open source databases currently in use are based on the relational model.
A short definition of an RDBMS may be a DBMS in which data is stored in the form of tables and the relationship among the data is also stored in the form of tables.

E.F. Codd, the famous mathematician has introduced 12 rules for the relational model for databases commonly known as **Codd's rules**. The rules mainly define what is required for a DBMS for it to be considered *relational*, i.e., an RDBMS. There is also one more rule i.e. Rule00 which specifies the relational model should use the relational way to manage the database. The rules and their description are as follows:-

Rule 0: Foundation Rule A *relational database management system* should be capable of using its *relational* facilities (exclusively) to *manage* the *database*.

Rule 1: Information Rule All information in the database is to be represented in one and only one way. This is achieved by values in column positions within rows of tables.

Rule 2: Guaranteed Access Rule All data must be accessible with no ambiguity, that is, 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.

Rule 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.

Rule 4: Dynamic On-line 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. The authorized users can access the database structure by using common language i.e. SQL.

Rule 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 is comprehensible:

- data definition
- view definition
- data manipulation (interactive and by program)
- integrity constraints
- authorization
- Transaction boundaries (begin, commit, and rollback).

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

Rule 7: High-level Insert, Update, and Delete The system is able to insert, update and delete operations fully. It can also perform the operations on multiple rows simultaneously.

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

Rule 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.

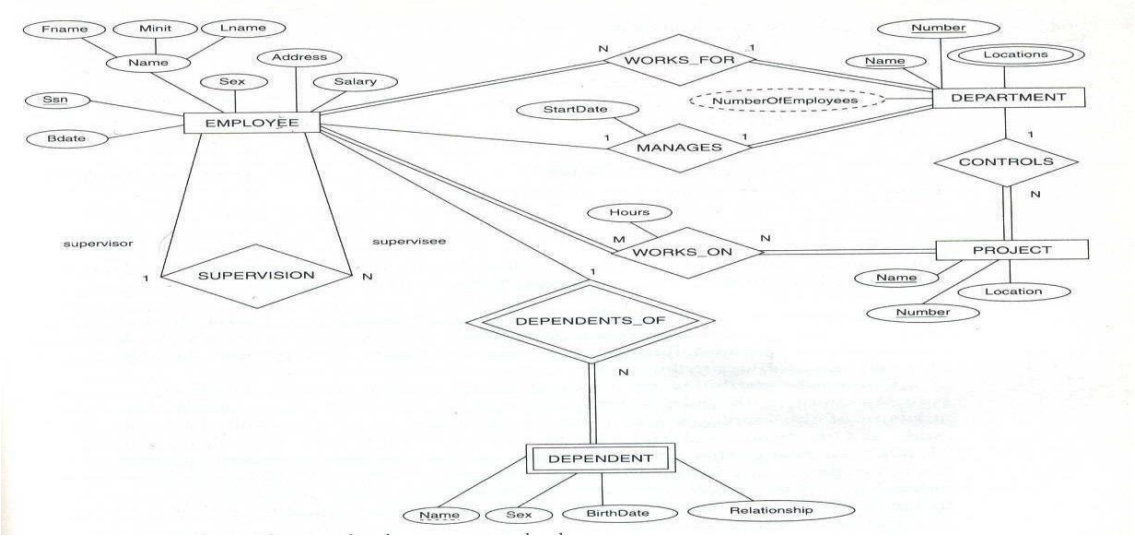
Rule 10: Integrity Independence Integrity constraints specific to a particular relational database must be definable in the relational data sublanguage and storable in the catalog, not in the application programs.

Rule 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.

Rule 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.

ENTITY-RELATIONSHIP MODEL

- The basic object that the ER model represents is an entity, which is a "thing" in the real world with an independent existence.
- An entity may be an object with a physical existence—a particular person, car, house, or employee—or it may be an object with a conceptual existence—a company, a job, or a university course.
- Each entity has attributes—the particular properties that describe it.
- For example, an employee entity may be described by the employee‘s name, age, address, salary, and job.
- A particular entity will have a value for each of its attributes.
- The attribute values that describe each entity become a major part of the data stored in the database.



Types of attributes:

- Simple versus Composite - Composite attributes can be divided into smaller subparts, which represent more basic attributes with independent meanings.

- Single-Valued versus Multivalued - Most attributes have a single value for a particular entity; such attributes are called single-valued. A multivalued attribute may have lower and upper bounds on the number of values allowed for each individual entity.

- Stored versus Derived - In some cases two (or more) attribute values are related.

Entity Types, Entity Sets, Keys, and Value Sets

Entity Types and Entity Sets

- A database usually contains groups of entities that are similar. For example, a company employing hundreds of employees may want to store similar information concerning each of the employees. These employee entities share the same attributes, but each entity has its own value(s) for each attribute.
- An entity type defines a collection (or set) of entities that have the same attributes. Each entity type in the database is described by its name and attributes.
- Figure shows two entity types, named EMPLOYEE and COMPANY, and a list of attributes for each. A few individual entities of each type are also illustrated, along with the values of their attributes.
- The collection of all entities of a particular entity type in the database at any point in time is called an entity set; the entity set is usually referred to using the same name as the entity type. For example, EMPLOYEE refers to both a type of entity as well as the current set of all employee entities in the database.
- An entity type is represented in ER diagrams as a rectangular box enclosing the entity type name.
- Attribute names are enclosed in ovals and are attached to their entity type by straight lines.
- Composite attributes are attached to their component attributes by straight lines.
- Multivalued attributes are displayed in double ovals.

- An entity type describes the schema or intension for a set of entities that share the same structure.
- The collection of entities of a particular entity type are grouped into an entity set, which is also called the extension of the entity type.

Key Attributes of an Entity Type

- An important constraint on the entities of an entity type is the key or uniqueness constraint on attributes.
- An entity type usually has an attribute whose values are distinct for each individual entity in the collection. Such an attribute is called a key attribute, and its values can be used to identify each entity uniquely.
- For example, the Name attribute is a key of the COMPANY entity type in Figure, because no two companies are allowed to have the same name. For the PERSON entity type, a typical key attribute is SocialSecurityNumber.
- Sometimes, several attributes together form a key, meaning that the combination of the attribute values must be distinct for each entity. If a set of attributes possesses this property, we can define a composite attribute that becomes a key attribute of the entity type.
- In ER diagrammatic notation, each key attribute has its name underlined inside the oval,

Value Sets (Domains) of Attributes

- Each simple attribute of an entity type is associated with a value set (or domain of values), which specifies the set of values that may be assigned to that attribute for each individual entity. If the range of ages allowed for employees is between 16 and 70, we can specify the value set of the Age attribute of EMPLOYEE to be the set of integer numbers between 16 and 70.

Extended Entity-Relationship (EER) Model

The extended entity-relationship (EER) model is a language for definition of structuring (and functionality) of database or information systems. It uses inductive development of structuring.

Basic attributes are assigned to base data types. *Complex attributes* can be constructed by applying constructors such as tuple, list or set constructors to attributes that have already been constructed.

Entity types conceptualize structuring of things of reality through attributes.

Relationship types associate types that have already been constructed into an association type. The types may be restricted by integrity constraints and by specification of identification of objects defined on the corresponding type. Typical integrity constraint of the extended entity- relationship model is participation, lookup and general cardinality constraints.

Entity, cluster and relationship classes contain a finite set of objects defined on these types. The types of an EER schema are typically depicted by an EER diagram.

The extended entity-relationship model is mainly used as a language for conceptualization of the structure of information systems applications. Conceptualization of database or information systems aims in a representation of the logical and physical structure of an information system in a given database management system (or for a database paradigm), so that it contains all the information required by the user and required for the efficient behavior of

the whole information system for all users. Furthermore, conceptualization may target to specify the database application processes and the user interaction. Description of structuring is currently the main use of the extended ER model. The diagram representation of EER schemata uses rectangles and diamonds for the entity and relationship types.

Functional Dependencies. Anomaly-1NF to 5NF

Functional Dependencies

A functional dependency is a constraint between two sets of attributes from the database.
Let R be the relational schema R={A1,A2,…An}. A functional dependency denoted by X □ Y Between two sets of attributes X and Y that are subset of R specifies a constraint on the possible tuples that can forma a relation state r of R.
X□ Y --- X functionally determines Y (or) there is a functional dependency from X to Y (or) Y is functionally dependent on X.
X □ L.H.S of F.D.
Y □ R.H.S of F.D.

Definition 1: X functionally determines Y in a relational schema R iff whenever two tuples of r(R) agree on their X value , they must necessarily agree on their Y value. F.D. is a property of semantics or meaning odf attributes.

Definition 2 :
For any two tuples t1 and t2 in r if t1[x]=t2[x], we must have t1[y]=t2[y], ie., values of Y component of a tuple in r depend on and determined by the values of X component.

Normal Forms:

Normalization was proposed by Codd (1972)

- It takes the relational schema through a series tests whether it satisfies a certain normal form.
- It proceeds in Top down fashion Design by analysis.
- Codd has defined I, II., III NF and a stronger definition of 3NF known as BCNF.(Boyce Codd Normal Form)
- All these normal forms are based on f.ds among attributes of a relation.

Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly. Let's discuss about anomalies first then we will discuss normal forms with examples.

Anomalies in DBMS

There are three types of anomalies that occur when the database is not normalized. These are – Insertion, update and deletion anomaly. Let's take an example to understand this.

Example: Suppose a manufacturing company stores the employee details in a table named employee that has four attributes: emp_id for storing employee's id, emp_name for storing employee's name, emp_address for storing employee's address and emp_dept for storing the department details in which the employee works. At some point of time the table looks like this:

emp_id	emp_name	emp_address	emp_dept
101	Rick	Delhi	D001
101	Rick	Delhi	D002
123	Maggie	Agra	D890
166	Glenn	Chennai	D900
166	Glenn	Chennai	D004

The above table is not normalized. We will see the problems that we face when a table is not normalized.

Update anomaly: In the above table we have two rows for employee Rick as he belongs to two departments of the company. If we want to update the address of Rick then we have to update the same in two rows or the data will become inconsistent.

Insert anomaly: Suppose a new employee joins the company, who is under training and currently not assigned to any department then we would not be able to insert the data into the table if emp_dept field doesn't allow nulls.

Delete anomaly: Suppose, if at a point of time the company closes the department D890 then deleting the rows that are having emp_dept as D890 would also delete the information of employee Maggie since she is assigned only to this department.
To overcome these anomalies we need to normalize the data. In the next section we will discuss about normalization.

Normalization
Here are the most commonly used normal forms:
- First normal form(1NF)
- Second normal form(2NF)
- Third normal form(3NF)
- Boyce & Codd normal form (BCNF)

First normal form (1NF)
As per the rule of first normal form, an attribute (column) of a table cannot hold multiple values. It should hold only atomic values.

Example: Suppose a company wants to store the names and contact details of its employees. It creates a table that looks like this:

emp_id	emp_name	emp_address	emp_mobile
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212 9900012222

103	Ron	Chennai	7778881212
104	Lester	Bangalore	9990000123 8123450987

Two employees (Jon & Lester) are having two mobile numbers so the company stored them in the same field as you can see in the table above.

This table is **not in 1NF** as the rule says "each attribute of a table must have atomic (single) values", the emp_mobile values for employees Jon & Lester violates that rule.
To make the table complies with 1NF we should have the data like this:

emp_id	emp_name	emp_address	emp_mobile
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212
102	Jon	Kanpur	9900012222
103	Ron	Chennai	7778881212
104	Lester	Bangalore	9990000123
104	Lester	Bangalore	8123450987

Second normal form (2NF)
A table is said to be in 2NF if both the following conditions hold:
- Table is in 1NF (First normal form)
- No non-prime attribute is dependent on the proper subset of any candidate key of table. An attribute that is not part of any candidate key is known as non-prime attribute.

Example: Suppose a school wants to store the data of teachers and the subjects they teach. They create a table that looks like this: Since a teacher can teach more than one subjects, the table can have multiple rows for a same teacher.

teacher_id	subject	teacher_age
111	Maths	38
111	Physics	38
222	Biology	38
333	Physics	40
333	Chemistry	40

Candidate Keys: {teacher_id, subject}
Non prime attribute: teacher_age
The table is in 1 NF because each attribute has atomic values. However, it is not in 2NF because non prime attribute teacher_age is dependent on teacher_id alone which is a proper subset of candidate key. This violates the rule for 2NF as the rule says "**no** non-prime attribute is dependent on the proper subset of any candidate key of the table".
To make the table complies with 2NF we can break it in two tables like this:

teacher_details table:

teacher_id	teacher_age
111	38
222	38
333	40

teacher_subject table:

teacher_id	subject
111	Maths
111	Physics
222	Biology
333	Physics
333	Chemistry

Now the tables comply with Second normal form (2NF).

Third Normal form (3NF)

A table design is said to be in 3NF if both the following conditions hold:

- Table must be in 2NF
- **Transitive functional dependency** of non-prime attribute on any super key should be removed.

An attribute that is not part of any **candidate key** is known as non-prime attribute.

In other words 3NF can be explained like this: A table is in 3NF if it is in 2NF and for each functional dependency X-> Y at least one of the following conditions hold:

- X is a **super key** of table
- Y is a prime attribute of table

An attribute that is a part of one of the candidate keys is known as prime attribute.

Example: Suppose a company wants to store the complete address of each employee, they create a table named employee_details that looks like this:

emp_id	emp_name	emp_zip	emp_state	emp_city	emp_district
1001	John	282005	UP	Agra	Dayal Bagh
1002	Ajeet	222008	TN	Chennai	M-City
1006	Lora	282007	TN	Chennai	Urrapakkam

1101	Lilly	29 20 08	UK	Pauri	Bhagwan
1201	Steve	22 29 99	MP	Gwalior	Ratan

Super keys: {emp_id}, {emp_id, emp_name}, {emp_id, emp_name, emp_zip}…so on
Candidate Keys: {emp_id}

Non-prime attributes: all attributes except emp_id are non-prime as they are not part of any candidate keys.

Here, emp_state, emp_city & emp_district dependent on emp_zip. And, emp_zip is dependent on emp_id that makes non-prime attributes (emp_state, emp_city & emp_district) transitively dependent on super key (emp_id). This violates the rule of 3NF.

To make this table complies with 3NF we have to break the table into two tables to remove the transitive dependency:

employee table:

emp_id	emp_name	emp_zip
1001	John	282005
1002	Ajeet	222008
1006	Lora	282007
1101	Lilly	292008
1201	Steve	222999

employee_zip table:

emp_zip	emp_state	emp_city	emp_district
282005	UP	Agra	Dayal Bagh
222008	TN	Chennai	M-City
282007	TN	Chennai	Urrapakkam
292008	UK	Pauri	Bhagwan
222999	MP	Gwalior	Ratan

Boyce Codd normal form (BCNF)

It is an advance version of 3NF that's why it is also referred as 3.5NF. BCNF is stricter than 3NF. A table complies with BCNF if it is in 3NF and for every **functional dependency**X->Y, X should be the super key of the table.

Example: Suppose there is a company wherein employees work in **more than one department**. They store the data like this:

emp_id	emp_nationality	emp_dept	dept_type	dept_no_of_emp
1001	Austrian	Production and planning	D001	200

1001	Austrian	stores	D001	250
1002	American	design and technical support	D134	100
1002	American	Purchasing department	D134	600

Functional dependencies in the table above:
emp_id -> emp_nationality
emp_dept -> {dept_type, dept_no_of_emp}

Candidate key: {emp_id, emp_dept}
The table is not in BCNF as neither emp_id nor emp_dept alone are keys.
To make the table comply with BCNF we can break the table in three tables like this:

emp_nationality table:

emp_id	emp_nationality
1001	Austrian
1002	American

emp_dept table:

emp_dept	dept_type	dept_no_of_emp
Production and planning	D001	200
stores	D001	250
design and technical support	D134	100
Purchasing department	D134	600

emp_dept_mapping table:

emp_id	emp_dept
1001	Production and planning
1001	stores
1002	design and technical support
1002	Purchasing department

Functional dependencies:
emp_id -> emp_nationality
emp_dept -> {dept_type, dept_no_of_emp}

Candidate keys:
For first table: emp_id
For second table: emp_dept
For third table: {emp_id, emp_dept}
This is now in BCNF as in both the functional dependencies left side part is a key.

Domain-Key Normal Form (DKNF)

There is no hard and fast rule about defining normal forms only up to 5NF. Historically, the process of normalization and the process of discovering undesirable dependencies was carried through 5NF as a meaningful design activity, but it has been possible to define stricter normal forms that take into account additional types of dependencies and constraints. The idea behind domain-key normal form (DKNF) is to specify (theoretically, at least) the "ultimate normal form" that takes into account all possible types of dependencies and constraints. A relation is said to be in DKNF if all constraints and dependencies that should hold on the relation can be enforced simply by enforcing the domain constraints and key constraints on the relation. For a relation in DKNF, it becomes very straightforward to enforce all database constraints by simply checking that each attribute value in a tuple is of the appropriate domain and that every key constraint is enforced.

Denormalization:

The ultimate goal during normalization is to separate the logically related attributes into tables to minimize redundancy, and thereby avoid the update anomalies that lead to an extra processing overhead to maintain consistency in the database.

Denormalization doesn‘t mean not normalizing. It‘s a step in the process.

First we normalize, then we realize that we now have hundreds or thousands of small tables and that the performance cost of all those joins will be prohibitive, and then carefully apply some denormalization techniques so that the application will return results before the werewolves have overrun the city.

Since normalization is about reducing redundancy, denormalizing is about deliberately adding redundancy to improve performance. Before beginning, consider whether or not it‘s necessary.Data Normalization, Denormalization, and the Forces of Darkness / Hollingsworth / p18

- Is the system‘s performance unacceptable with fully normalized data? Mock up a client and do some testing.
- If the performance is unacceptable, will denormalizing make it acceptable? Figure out where your bottlenecks are.
- If you denormalize to clear those bottlenecks, will the system and its data still be reliable? Unless the answers to all three are ―yes,‖ denormalization should be avoided. Unfortunately for me, the Council of Light has some pretty old hardware and can‘t or won‘t upgrade to newer and more efficient SQL software. I‘d better get to work.

KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BCom CA COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 18CCU302 BATCH-2018-21

UNIT-II

SYLLABUS

UNIT II SQL & QUERY OPTIMIZATION

SQL Standards - Data types - Database Objects- DDL-DML-DCL-TCL-Embedded SQL-Static Vs Dynamic SQL - QUERY OPTIMIZATION: Query Processing and Optimization - Heuristics and Cost Estimates in Query Optimization

SQL Standards

The SQL language may be considered one of the major reasons for the commercial success of relational databases. Because it became a standard for relational databases, users were less concerned about migrating their database applications from other types of database systems. The advantage of having such a standard is that users may write statements in a database application program that can access data stored in two or more relational DBMSs without having to change the database sublanguage (SQL) if both relational DBMSs support standard SQL. The name **SQL** is presently expanded as Structured Query Language. Originally, SQL was called SEQUEL (Structured English Query Language).

SQL is now the standard language for commercial relational DBMSs. A joint effort by the American National Standards Institute (ANSI) and the International Standards Organization (ISO) has led to a standard version of SQL (ANSI 1986), called SQL-86 or SQL1. A revised and much expanded standard called SQL-92 (also referred to as SQL2) was subsequently developed. The next standard that is well-recognized is SQL:1999, which started out as SQL3. Two later updates to the standard are SQL:2003 and SQL:2006, which added XML features among other updates to the language. Another update in 2008 incorporated more object database features in SQL. We will try to cover the latest version of SQL as much as possible. SQL is a comprehensive database language: It has statements for data definitions, queries, and updates. Hence, it is both a DDL *and* a DML.

Data types

The basic **data types** available for attributes include numeric, character string, bit string, Boolean, date, and time.

- **Numeric** data types include integer numbers of various sizes (INTEGER or INT, and SMALLINT) and floating-point (real) numbers of various precision (FLOAT or REAL, and DOUBLE PRECISION).
- **Character-string** data types are either fixed length—CHAR(*n*) or CHARACTER(*n*), where *n* is the number of characters—or varying length—VARCHAR(*n*) or CHAR VARYING(*n*) or CHARACTER VARYING(*n*), where *n* is the maximum number of characters.
- **Bit-string** data types are either of fixed length *n*—BIT(*n*)—or varying length—

BIT VARYING(*n*), where *n* is the maximum number of bits.

- A **Boolean** data type has the traditional values of TRUE or FALSE.
- The **DATE** data type has ten positions, and its components are YEAR, MONTH, and DAY in the form YYYY-MM-DD.
- A **timestamp** data type (TIMESTAMP) includes the DATE and TIME fields, plus a minimum of six positions for decimal fractions of seconds and an optional WITH TIME ZONE qualifier.
- Another data type related to DATE, TIME, and TIMESTAMP is the INTERVAL data type.

Database Objects

A database object in a relational database is a data structure used to either store or reference data. The most common object that people interact with is the table. Other objects are indexes, stored procedures, sequences, views and many more.

When a database object is created, a new object type cannot be created because all the various object types created are restricted by the very nature, or source code, of the relational database model being used, such as Oracle, SQL Server or Access. What is being created is instances of the objects, such as a new table, an index on that table or a view on the same table.

Most of the major database engines offer the same set of major database object types:

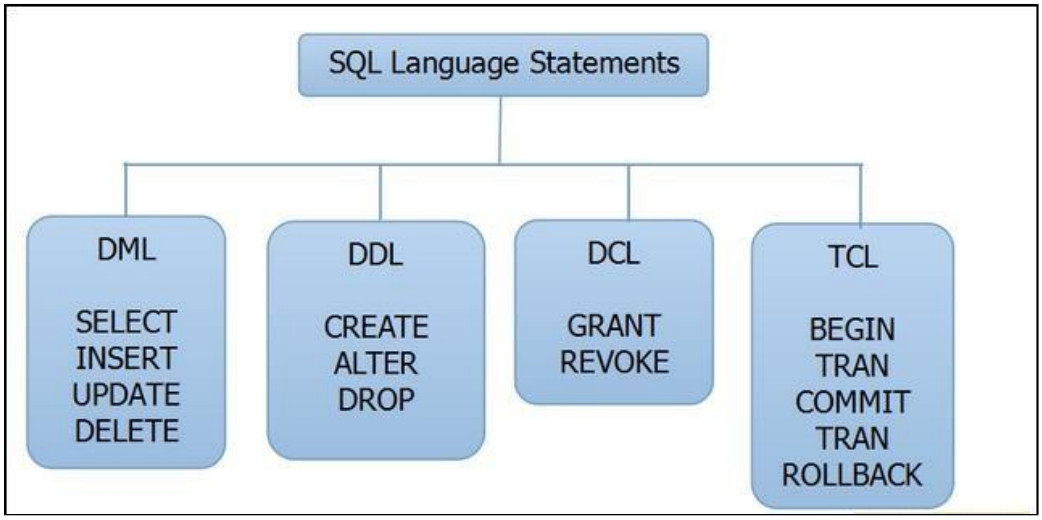
- Tables
- Indexes
- Sequences
- Views
- Synonyms

DDL-DML-DCL-TCL

SQL language is divided into four types of primary language statements: DML, DDL, DCL and TCL. Using these statements, we can define the structure of a database by creating and altering database objects, and we can manipulate data in a table through updates or deletions. We also can control which user can read/write data or manage transactions to create a single unit of work.

The four main categories of SQL statements are as follows:

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



DML (Data Manipulation Language)

DML statements affect records in a table. These are basic operations we perform on data such as selecting a few records from a table, inserting new records, deleting unnecessary records, and updating/modifying existing records.

DML statements include the following: **SELECT** – select records from a table **INSERT** – insert new records

UPDATE – update/Modify existing records

DELETE – delete existing records

DDL (Data Definition Language)

DDL statements are used to alter/modify a database or table structure and schema. These statements handle the design and storage of database objects.

CREATE – create a new Table, database, schema **ALTER** – alter existing table, column description **DROP** – delete existing objects from database

DCL (Data Control Language)

DCL statements control the level of access that users have on database objects.

GRANT – allows users to read/write on certain database objects

REVOKE – keeps users from read/write permission on database objects

TCL (Transaction Control Language)

TCL statements allow you to control and manage transactions to maintain the integrity of data within SQL statements.

BEGIN Transaction – opens a transaction

COMMIT Transaction – commits a transaction

ROLLBACK Transaction – ROLLBACK a transaction in case of any error

Embedded SQL

The technique is for the SQL statements where that can be embedded in a general- purpose programming language. The two languages:C and Java. The examples used with the C language, known as **embedded SQL.** In this embedded approach, the programming language is called the **host language**. Most SQL statements—including data or constraint definitions, queries, updates, or view definitions—can be embedded in a host language program.

SQL provides a powerful declarative query language. However, access to a database from a general-purpose programming language is required because,

- SQL is not as powerful as a general-purpose programming language. There are queries that cannot be expressed in SQL, but can be programmed in C, Fortran, Pascal, Cobol, etc.
- Nondeclarative actions -- such as printing a report, interacting with a user, or sending the result to a GUI -- cannot be done from within SQL.

The SQL standard defines embedding of SQL as *embedded SQL* and the language in which SQL queries are embedded is referred as *host language*.

The result of the query is made available to the program one tuple (record) at a time. To identify embedded SQL requests to the preprocessor, we use EXEC SQL statement: EXEC SQL embedded SQL statement END-EXEC.

Note: A semi-colon is used instead of END-EXEC when SQL is embedded in C or Pascal. Embedded SQL statements: **declare cursor**, **open**, and **fetch** statements.

EXEC SQL

declare *c* **cursor for select** *cname, ccity* **from** *deposit, customer*

where *deposit.cname = customer.cname* **and** *deposit.balance > :amount*

END-EXEC

where *amount* is a host-language variable.

EXEC SQL **open** *c*

END-EXEC

This statement causes the DB system to execute the query and to save the results within a temporary relation.

A series of **fetch** statement are executed to make tuples of the results available to the program. EXEC SQL **fetch** *c* **into** *:cn, :cc* END-EXEC

The program can then manipulate the variable *cn* and *cc* using the features of the host programming language.

A single **fetch** request returns only one tuple. We need to use a **while** loop (or equivalent) to process each tuple of the result until no further tuples (when a variable in the SQLCA is set).

We need to use **close** statement to tell the DB system to delete the temporary relation that held the result of the query.

EXEC SQL **close** *c* END-EXEC

Embedded SQL can execute any valid **update**, **insert**, or **delete** statements.

Examples:

0) int loop ;
1) EXEC SQL BEGIN DECLARE SECTION ;
2) varchar dname [16], fname [16], lname [16], address [31] ;
3) char ssn [10], bdate [11], sex [2], minit [2] ;
4) float salary, raise ;
5) int dno, dnumber ;

```
6) int SQLCODE ; char SQLSTATE [6] ;
7) EXEC SQL END DECLARE SECTION ;

//Program Segment E1: Program segment E1, a C program segment with embedded SQL
0) loop = 1 ;
1) while (loop) {
2) prompt("Enter a Social Security Number: ", ssn) ;
3) EXEC SQL
4) select Fname, Minit, Lname, Address, Salary
5) into :fname, :minit, :lname, :address, :salary
6) from EMPLOYEE where Ssn = :ssn ;
7) if (SQLCODE == 0) printf(fname, minit, lname, address, salary)
8) else printf("Social Security Number does not exist: ", ssn) ;
9) prompt("More Social Security Numbers (enter 1 for Yes, 0 for No): ", loop) ;
10) }
```

Program segment E2, a C program segment that uses cursors with embedded SQL for update purposes.

```
//Program Segment E2:
0) prompt("Enter the Department Name: ", dname) ;
1) EXEC SQL
2) select Dnumber into :dnumber
3) from DEPARTMENT where Dname = :dname ;
4) EXEC SQL DECLARE EMP CURSOR FOR
5) select Ssn, Fname, Minit, Lname, Salary
6) from EMPLOYEE where Dno = :dnumber
7) FOR UPDATE OF Salary ;
8) EXEC SQL OPEN EMP ;
9) EXEC SQL FETCH from EMP into :ssn, :fname, :minit, :lname, :salary ;
10) while (SQLCODE == 0) {
11) printf("Employee name is:", Fname, Minit, Lname) ;
12) prompt("Enter the raise amount: ", raise) ;
13) EXEC SQL
14) update EMPLOYEE
15) set Salary = Salary + :raise
16) where CURRENT OF EMP ;
17) EXEC SQL FETCH from EMP into :ssn, :fname, :minit, :lname, :salary ;
18) }
19) EXEC SQL CLOSE EMP ;
```

Static Vs Dynamic SQL

Static SQL is SQL statements in an application that do not change at runtime and, therefore, can be hard-coded into the application. **Dynamic SQL** is SQL statements that are constructed at runtime; for example, the application may allow users to enter their own queries. Thus, the SQL statements cannot be hard-coded into the application.

Static SQL provides performance advantages over dynamic SQL because static

SQL is pre-processed, which means the statements are parsed, validated, and optimized only once.

If you are using a standards-based API, such as ODBC, to develop your application, static SQL is probably not an option for you. However, you can achieve a similar level of performance by using either statement pooling or stored procedures.

S.No.	Static (embedded) SQL	Dynamic (interactive) SQL
1.	In static SQL how database will be accessed is predetermined in the embedded SQL statement.	In dynamic SQL, how database will be accessed is determined at run time.
2.	It is more swift and efficient.	It is less swift and efficient.
3.	SQL statements are compiled at compile time.	SQL statements are compiled at run time.
4.	Parsing, validation, optimization, and generation of application plan are done at compile time.	Parsing, validation, optimization, and generation of application plan are done at run time.
5.	It is generally used for situations where data is distributed uniformly.	It is generally used for situations where data is distributed non-uniformly.
6.	EXECUTE IMMEDIATE, EXECUTE and PREPARE statements are not used.	EXECUTE IMMEDIATE, EXECUTE and PREPARE statements are used.
7.	It is less flexible.	It is more flexible.

QUERY OPTIMIZATION: Query Processing and Optimization

A query expressed in a high-level query language such as SQL must first be scanned, parsed, and validated. 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**. ***Introduction to Query Processing***

- In databases that provide low-level access routines such as IMS or flat file databases, the programmer must write code to perform the queries.
- With higher level database query languages such as SQL and QUEL, a special component of the DBMS called the Query Processor takes care of arranging the underlying access routines to satisfy a given query.
- Thus queries can be specified in terms of the required results rather than in terms of how to achieve those results.

A query is processed in four general steps:

1. Scanning and Parsing
2. Query Optimization or planning the execution strategy
3. Query Code Generator (interpreted or compiled)
4. Execution in the runtime database processor

1. Scanning and Parsing

- When a query is first submitted (via an applications program), it must be scanned and parsed to determine if the query consists of appropriate syntax.
- **Scanning** is the process of converting the query text into a tokenized representation.
- The tokenized representation is more compact and is suitable for processing by the parser.
- This representation may be in a tree form.
- The **Parser** checks the tokenized representation for correct syntax.
- In this stage, checks are made to determine if columns and tables identified in the query exist in the database and if the query has been formed correctly with the appropriate keywords and structure.
- If the query passes the parsing checks, then it is passed on to the Query Optimizer.

2. Query Optimization or Planning the Execution Strategy

- For any given query, there may be a number of different ways to execute it.
- Each operation in the query (SELECT, JOIN, etc.) can be implemented using one or more different *Access Routines*.
- For example, an access routine that employs an index to retrieve some rows would be more efficient that an access routine that performs a full table scan.
- The goal of the **query optimizer** is to find a *reasonably efficient* strategy for executing the query (not quite what the name implies) using the access routines.
- Optimization typically takes one of two forms: *Heuristic Optimization* or *Cost Based Optimization*
- In **Heuristic Optimization**, the query execution is refined based on *heuristic rules* for reordering the individual operations.
- With **Cost Based Optimization**, the overall cost of executing the query is systematically reduced by estimating the costs of executing several different execution plans.

3. Query Code Generator (interpreted or compiled)

- Once the query optimizer has determined the execution plan (the specific ordering of access routines), the code generator writes out the actual access routines to be executed.
- With an interactive session, the query code is interpreted and passed directly to the runtime database processor for execution.
- It is also possible to *compile* the access routines and store them for later execution.

4. Execution in the runtime database processor

- At this point, the query has been scanned, parsed, planned and (possibly) compiled.
- The runtime database processor then executes the access routines against the database.
- The results are returned to the application that made the query in the first place.
- Any runtime errors are also returned.

Query optimization is a function of many relational database management systems.

The **query optimizer** attempts to determine the most efficient way to execute a given query by considering the possible query plans.

Generally, the query optimizer cannot be accessed directly by users: once queries are submitted to database server, and parsed by the parser, they are then passed to the query optimizer where optimization occurs. However, some database engines allow guiding the query optimizer with hints.

A query is a request for information from a database. Queries results are generated by accessing relevant database data and manipulating it in a way that yields the requested information.

Heuristic Query Optimization

Oracle calls this Rule Based optimization.

A query can be represented as a tree data structure. Operations are at the interior nodes and data items (tables, columns) are at the leaves.

The query is evaluated in a *depth-first* pattern.

Consider this query from the Elmasri/Navathe textbook:

```
SELECT PNUMBER, DNUM, LNAME
FROM PROJECT, DEPARTMENT,
EMPLOYEE WHERE
        DNUM=DNUMBER and
MGREMPID=EMPID and
        PLOCATION = 'Stafford';
```

Or, in relational algebra:

$$\pi_{\text{pnumber, dnum, lname}} \left(\sigma_{\text{plocation = 'Stafford'}} \left(\sigma_{\text{mgrssn=ssn}} \left(\sigma_{\text{dnum=dnumber}} \right. \right. \right.$$

on the following schema:

EMPLOYEE TABLE

FNAME	MI	LNAME	EMPID	BDATE	ADDRESS	S	SALARY	SUPERMPID	DNO
JOHN	B	SMITH	123456789	09-JAN-55	731 FONDREN, HOUSTON, TX	M	30000	333445555	5
FRANKLIN	T	WONG	333445555	08-DEC-45	638 VOSS,HOUSTON TX	M	40000	888665555	5
ALICIA	J	ZELAYA	999887777	19-JUL-58	3321 CASTLE, SPRING,TX	F	25000	987654321	4

JENNIFER	S	WALLACE	987654321	20-JUN-31	291 BERRY, TX	BELLAIRE,	F	43000	88866555 5	4
RAMESH	K	NARAYAN	666884444	15-SEP-52	975 FIRE HUMBLE, TX	OAK,	M	38000	33344555 5	5
JOYCE	A	ENGLISH	453453453	31-JUL-62	5631 RICE, TX	HOUSTON,	F	25000	33344555 5	5
AHMAD	V	JABBAR	987987987	29-MAR-59	980 DALLAS, TX	HOUSTON,	M	25000	98765432 1	4
JAMES	E	BORG	888665555	10-NOV-27	450 STONE, TX	HOUSTON,	M	55000		1

DEPARTMENT TABLE:

DNAME	DNUMBER	MGREMPID	MGRSTARTD
HEADQUARTERS	1	888665555	19-JUN-71
ADMINISTRATION	4	987654321	01-JAN-85
RESEARCH	5	333445555	22-MAY-78

PROJECT TABLE

PNAME	PNUMBER	PLOCATION	DNUM
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerizatn.	10	Stafford	4
Reorganization	20	Houston	1
NewBenefits	30	Stafford	4

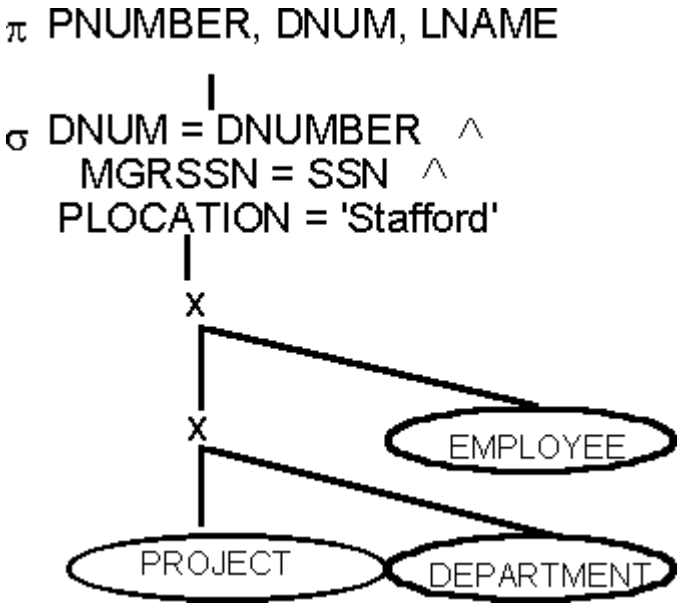
- An overall rule for heuristic query optimization is to perform as many select and project operations as possible before doing any joins.
- There are a number of transformation rules that can be used to transform a query:
    1. Cascading selections. A list of conjunctive conditions can be broken up into separate individual conditions.
    2. Commutativity of the selection operation.
    3. Cascading projections. All but the last projection can be ignored.
    4. Commuting selection and projection. If a selection condition only involves attributes contained in a projection clause, the two can be commuted.
    5. Commutativity of Join and Cross Product.
    6. Commuting selection with Join.

7. Commuting projection with Join.
8. Commutativity of set operations. Union and Intersection are commutative.
9. Associativity of Union, Intersection, Join and Cross Product.
10. Commuting selection with set operations.
11. Commuting projection with set operations.
12. Logical transformation of selection conditions. For example, using DeMorgan's law, etc.
13. Combine Selection and Cartesian product to form Joins.

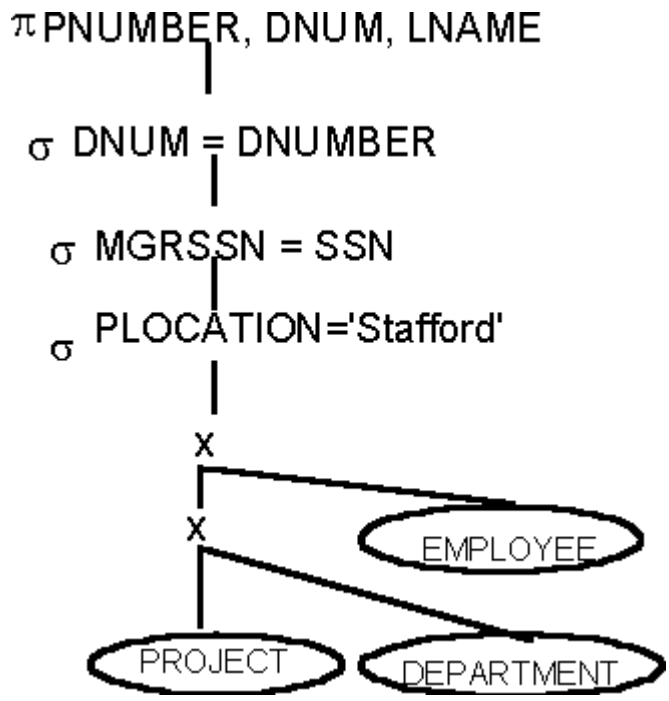
- These transformations can be used in various combinations to optimize queries. Some general steps follow:
    1. Using rule 1, break up conjunctive selection conditions and chain them together.
    2. Using the commutativity rules, move the selection operations as far down the tree as possible.
    3. Using the associativity rules, rearrange the leaf nodes so that the most restrictive selection conditions are executed first. For example, an equality condition is likely more restrictive than an inequality condition (range query).
    4. Combine cartesian product operations with associated selection conditions to form a single Join operation.
    5. Using the commutativity of Projection rules, move the projection operations down the tree to reduce the sizes of intermediate result sets.
    6. Finally, identify subtrees that can be executed using a single efficient access method.

Example of Heuristic Query Optimization

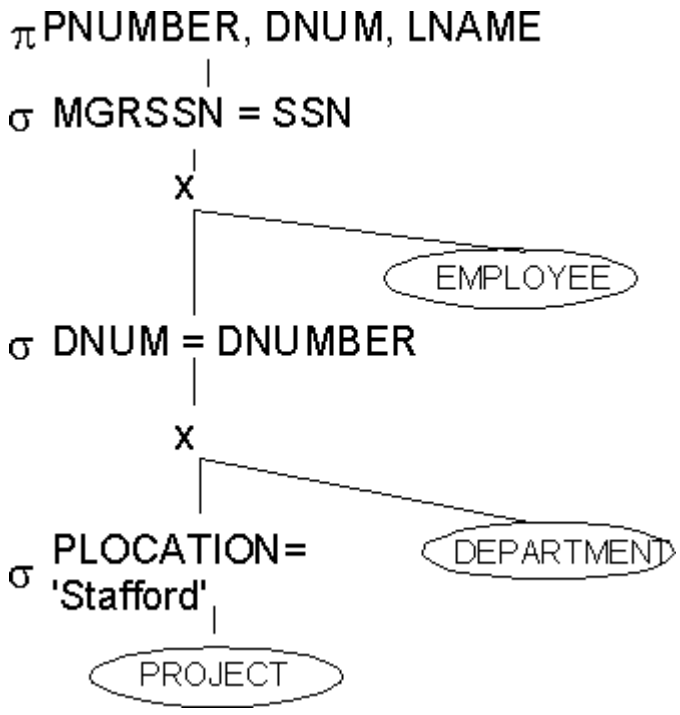
1. Original Query Tree



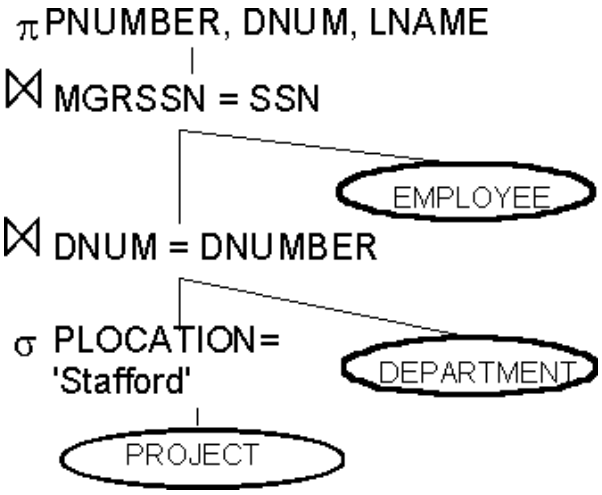
2. Use Rule 1 to Break up Cascading Selections



3. Commute Selection with Cross Product



4. Combine Cross Product and Selection to form Joins



Cost Estimates in Query Optimization

A query optimizer does not depend solely on heuristic rules; it also estimates and compares the costs of executing a query using different execution strategies and algorithms, and it then chooses the strategy with the *lowest cost estimate.* For this approach to work, accurate *cost estimates* are required so that different strategies can be compared fairly and realistically.

The cost of executing a query includes the following components:

- Access cost to secondary storage.
- Disk storage cost.
- Computation cost.
- Memory usage cost.

- Communication cost.

Example to Illustrate Cost-Based Query Optimization

We will consider query Q2 and its query tree to illustrate cost-based query optimization:

Q2: **SELECT** Pnumber, Dnum, Lname, Address, Bdate
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE Dnum=Dnumber **AND** Mgr_ssn=Ssn **AND** Plocation='Stafford';

The first cost-based optimization to consider is join ordering. As previously mentioned, we assume the optimizer considers only left-deep trees, so the potential join orders—without CARTESIAN PRODUCT—are:

1. PROJECT ⋈ DEPARTMENT ⋈ EMPLOYEE
2. DEPARTMENT ⋈ PROJECT ⋈ EMPLOYEE
3. DEPARTMENT ⋈ EMPLOYEE ⋈ PROJECT
4. EMPLOYEE ⋈ DEPARTMENT ⋈ PROJECT

KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BCom CA COURSE NAME: DATABASE MANAGEMENT SYSTEM

COURSE CODE: 18CCU302 BATCH-2018-21

UNIT-III

SYLLABUS

UNIT III TRANSACTION PROCESSING AND CONCURRENCY CONTROL

Introduction-Properties of Transaction- Serializability- Concurrency Control– Locking Mechanisms- Two Phase Commit Protocol-Dead lock.

Introduction

The concept of transaction provides a mechanism for describing logical units of database processing. Transaction processing systems are systems with large databases and hundreds of concurrent users executing database transactions. Examples of such systems include airline reservations, banking, credit card processing, online retail purchasing, stock markets, supermarket checkouts, and many other applications. These systems require high availability and fast response time for hundreds of concurrent users. A transaction is typically implemented by a computer program, which includes database commands such as retrievals, insertions, deletions, and updates.

Single-User versus Multiuser Systems

One criterion for classifying a database system is according to the number of users who can use the system concurrently. A DBMS is single-user if at most one user at a time can use the system, and it is multiuser if many users can use the system—and hence access the database—concurrently. Single-user DBMSs are mostly restricted to personal computer systems; most other DBMSs are multiuser. For example, an airline reservations system. Database systems used in banks, insurance agencies, stock exchanges, supermarkets, and many other applications are multiuser systems. Multiple users can access databases—and use computer systems—simultaneously because of the concept of multiprogramming, which allows the operating system of the computer to execute multiple programs—or processes—at the same time. A single central processing unit (CPU) can only execute at most one process at a time. However, multiprogramming operating systems execute some commands from one process, then suspend that process and execute some commands from the next process, and so on.

Transactions, Database Items, Read and Write Operations, and DBMS Buffers

A transaction is an executing program that forms a logical unit of database processing. A transaction includes one or more database access operations—these can include insertion, deletion, modification, or retrieval operations. One way of specifying the transaction boundaries is by specifying explicit begin transaction and end transaction statements in an application program. If the database operations in a transaction do not update the database but only retrieve data, the transaction is called a read-only transaction; otherwise it is known as a read-write transaction. A database is basically represented as a collection of *named data items.* The size of a data item is called its granularity.

A data item can be a *database record*, but it can also be a larger unit such as a whole *disk block*, or even a smaller unit such as an individual *field (attribute) value* of some record in the database. The DBMS will maintain in the database cache a number of data buffers in main memory. Each buffer typically holds the contents of one database disk block, which contains some of the database items being processed. **Concurrency control and recovery mechanisms** are mainly concerned with the database commands in a transaction. Several problems can occur when concurrent transactions execute in an uncontrolled manner.

- The Lost Update Problem.
- The Temporary Update (or Dirty Read) Problem
- The Incorrect Summary Problem.
- The Unrepeatable Read Problem

If a transaction fails after executing some of its operations but before executing all of them, the operations already executed must be undone and have no lasting effect.

Types of Failures. Failures are generally classified as transaction, system, and media failures. There are several possible reasons for a transaction to fail in the middle of execution:

- A computer failure (system crash).
- A transaction or system error/
- Local errors or exception conditions detected by the transaction.
- Concurrency control enforcement
- Disk failure
- Physical problems and catastrophes

Properties of Transaction

Transactions should possess several properties, often called the **ACID** properties; they should be enforced by the concurrency control and recovery methods of the DBMS. The following are the ACID properties:

■ **Atomicity.** A transaction is an atomic unit of processing; it should either be performed in its entirety or not performed at all.

■ **Consistency preservation.** A transaction should be consistency preserving, meaning that if it is completely executed from beginning to end without interference from other transactions, it should take the database from one consistent state to another.

■ **Isolation.** A transaction should appear as though it is being executed in isolation from other transactions, even though many transactions are executing concurrently. That is, the execution of a transaction should not be interfered with by any other transactions executing concurrently.

■ **Durability or permanency.** The changes applied to the database by a committed transaction must persist in the database. These changes must not be lost because of any failure.

Serializability

When multiple transactions are being executed by the operating system in a multiprogramming environment, there are possibilities that instructions of one transactions are interleaved with some other transaction.

- **Schedule** − A chronological execution sequence of a transaction is called a schedule. A schedule can have many transactions in it, each comprising of a number of instructions/tasks.
- **Serial Schedule** − It is a schedule in which transactions are aligned in such a way that one transaction is executed first. When the first transaction completes its cycle, then the next transaction is executed. Transactions are ordered one after the other. This type of schedule is called a serial schedule, as transactions are executed in a serial manner.

In a multi-transaction environment, serial schedules are considered as a benchmark. The execution sequence of an instruction in a transaction cannot be changed, but two transactions can have their instructions executed in a random fashion. This execution does no harm if two transactions are mutually independent and working on different segments of data; but in case these two transactions are working on the same data, then the results may vary. This ever-varying result may bring the database to an inconsistent state.

To resolve this problem, we allow parallel execution of a transaction schedule, if its transactions are either serializable or have some equivalence relation among them.

Equivalence Schedules

An equivalence schedule can be of the following types −

Result Equivalence

If two schedules produce the same result after execution, they are said to be result equivalent. They may yield the same result for some value and different results for another set of values. That's why this equivalence is not generally considered significant.

View Equivalence

Two schedules would be view equivalence if the transactions in both the schedules perform similar actions in a similar manner.

For example −

- If T reads the initial data in S1, then it also reads the initial data in S2.
- If T reads the value written by J in S1, then it also reads the value written by J in S2.
- If T performs the final write on the data value in S1, then it also performs the final write on the data value in S2.

Conflict Equivalence

Two schedules would be conflicting if they have the following properties −

- Both belong to separate transactions.
- Both accesses the same data item.
- At least one of them is "write" operation.

Two schedules having multiple transactions with conflicting operations are said to be conflict equivalent if and only if −

- Both the schedules contain the same set of Transactions.
- The order of conflicting pairs of operation is maintained in both the schedules.

Note − View equivalent schedules are view serializable and conflict equivalent schedules are conflict serializable. All conflict serializable schedules are view serializable too.

Concurrency Control

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

A **lock** is a variable associated with a data item that describes the status of the item with respect to possible operations that can be applied to it. Generally, there is one lock for each data item in the database. Locks are used as a means of synchronizing the access by concurrent transactions to the database items.

Types of Locks and System Lock Tables

Several types of locks are used in concurrency control.

- Binary locks, which are simple, but are also too restrictive for database concurrency control purposes
- Shared/exclusive locks—also known as read/write locks—which provide more general locking capabilities and are used in practical database locking schemes
- Certify lock, - show how it can be used to improve performance of locking protocols.

Binary Locks. A **binary lock** can have two **states** or **values:** locked and unlocked (or 1 and 0, for simplicity). Two operations, lock_item and unlock_item, are used with binary locking.

```
lock_item(X):
B:  if LOCK(X) = 0              (* item is unlocked *)
        then LOCK(X) ←1         (* lock the item *)
    else
        begin
        wait (until LOCK(X) = 0
            and the lock manager wakes up the transaction);
        go to B
        end;
unlock_item(X):
    LOCK(X) ← 0;                (* unlock the item *)
    if any transactions are waiting
        then wakeup one of the waiting transactions;
```

If the simple binary locking scheme described here is used, every transaction must obey the following rules:

1. A transaction *T* must issue the operation lock_item(*X*) before any read_item(*X*) or write_item(*X*) operations are performed in *T*.
2. A transaction *T*must issue the operation unlock_item(*X*) after all read_item(*X*) and write_item(*X*) operations are completed in *T*.
3. A transaction *T* will not issue a lock_item(*X*) operation if it already holds the lock on item *X*.1
4. A transaction *T* will not issue an unlock_item(*X*) operation unless it already holds the lock on item *X*.

Shared/Exclusive (or Read/Write) Locks. The preceding binary locking scheme is too restrictive for database items because at most, one transaction can hold a lock on a given item. We should allow several transactions to access the same item X if they all access X for reading purposes only. a different type of lock called a multiple-mode lock is used.

In this scheme—called shared/exclusive or read/write locks—there are three locking operations: _lock(X), write_lock(X), and unlock(X).

A read-locked item is also called share-locked because other transactions are allowed to read the item, whereas a write- read locked item is called exclusive-locked because a single transaction exclusively holds the lock on the item. Each record in the lock table will have four fields: <Data_item_name, LOCK, No_of_reads, Locking_transaction(s)>. The value (state) of LOCK is either read-locked or write-locked, suitably coded (if we assume no records are kept in the lock table for unlocked items).

When we use the shared/exclusive locking scheme, the system must enforce the following rules:
1. A transaction T must issue the operation read_lock(X) or write_lock(X) before any read_item(X) operation is performed in T.

2. A transaction T must issue the operation write_lock(X) before any write_item(X) operation is performed in T.

```
read_lock(X):
B:  if LOCK(X) = "unlocked"
        then begin LOCK(X) ← "read-locked";
                no_of_reads(X) ← 1
                end
    else if LOCK(X) = "read-locked"
        then no_of_reads(X) ← no_of_reads(X) + 1
    else begin
            wait (until LOCK(X) = "unlocked"
                and the lock manager wakes up the transaction);
            go to B
            end;
write_lock(X):
B:  if LOCK(X) = "unlocked"
        then LOCK(X) ← "write-locked"
    else begin
            wait (until LOCK(X) = "unlocked"
                and the lock manager wakes up the transaction);
            go to B
            end;
unlock (X):
    if LOCK(X) = "write-locked"
        then begin LOCK(X) ← "unlocked";
                wakeup one of the waiting transactions, if any
                end
    else it LOCK(X) = "read-locked"
        then begin
                no_of_reads(X) ← no_of_reads(X) –1;
                if no_of_reads(X) = 0
                    then begin LOCK(X) = "unlocked";
                            wakeup one of the waiting transactions, if any
                            end
                end;
```

3. A transaction T must issue the operation unlock(X) after all read_item(X) and write_item(X) operations are completed in T.3
4. A transaction T will not issue a read_lock(X) operation if it already holds a read (shared) lock or a write (exclusive) lock on item X. This rule may be relaxed, as we discuss shortly.
5. A transaction T will not issue a write_lock(X) operation if it already holds a read (shared) lock or write (exclusive) lock on item X. This rule may also be relaxed, as we discuss shortly.
6. A transaction T will not issue an unlock(X) operation unless it already holds a read (shared) lock or a write (exclusive) lock on item X.

Conversion of Locks. Sometimes it is desirable to relax conditions 4 and 5 in the preceding list in order to allow lock conversion; that is, a transaction that already holds a lock on item X is allowed under certain conditions to convert the lock from one locked state to another. For example, it is possible for a transaction T to issue a read_lock(X) and then later to upgrade the lock by issuing a write_lock(X) operation. If T is the only transaction holding a read lock on X at the time it issues the write_lock(X) operation, the lock can be upgraded; otherwise, the transaction must wait. It is also possible for a transaction T to issue a write_lock(X) and then later to downgrade the lock by issuing a read_lock(X) operation.

Two Phase Commit Protocol

A transaction is said to follow the two-phase locking protocol if all locking operations (read_lock, write_lock) precede the first unlock operation in the transaction. Such a transaction can be divided into two phases: an expanding or growing (first) phase, during which new locks on items can be acquired but none can be released; and a shrinking (second) phase, during which existing locks can be released but no new locks can be acquired. If lock conversion is allowed, then upgrading of locks (from read-locked to write-locked) must be done during the expanding phase, and downgrading of locks (from write-locked to read-locked) must be done in the shrinking phase. Hence, a read_lock(X) operation that downgrades an already held write lock on X can appear only in the shrinking phase.

It can be proved that, if every transaction in a schedule follows the two-phase locking protocol, the schedule is guaranteed to be serializable, obviating the need to test for serializability of schedules. The locking protocol, by enforcing two-phase locking rules, also enforces serializability. Although the two-phase locking protocol guarantees serializability (that is, every schedule that is permitted is serializable), it does not permit all possible serializable schedules (that is, some serializable schedules will be prohibited by the protocol). Basic, Conservative, Strict, and Rigorous Two-Phase Locking. There are a number of variations of two-phase locking (2PL). The technique just described is known as basic 2PL. A variation known as conservative 2PL (or static 2PL) requires a transaction to lock all the items it accesses before the transaction begins execution, by predeclaring its read-set and write-set.

Deadlock

Deadlock occurs when each transaction T in a set of two or more transactions is waiting for some item that is locked by some other transaction T_ in the set. Hence, each transaction in the set is in a waiting queue, waiting for one of the other transactions in the set to release the lock on an item. But because the other transaction is also waiting, it will never release the lock.

Deadlock Prevention Protocols. One way to prevent deadlock is to use a deadlock prevention protocol. One deadlock prevention protocol, which is used in conservative two-phase locking, requires that every transaction lock all the items it needs in advance (which is generally not a practical assumption). A second protocol, which also limits concurrency, involves ordering all the items in the database and making sure that a transaction that needs several items will lock them according to that order.

A number of other deadlock prevention schemes have been proposed that make a decision about what to do with a transaction involved in a possible deadlock situation. Some of these techniques use the concept of **transaction timestamp** TS(*T*), which is a unique identifier assigned to each transaction. The rules followed by these schemes are:

- **Wait-die.** If TS(*Ti*) < TS(*Tj*), then (*Ti* older than *Tj*) *Ti* is allowed to wait; otherwise (*Ti* younger than *Tj*) abort *Ti* (*Ti dies*) and restart it later *with the same timestamp.*
- **Wound-wait.** If TS(*Ti*) < TS(*Tj*), then (*Ti* older than *Tj*) abort *Tj* (*Ti wounds Tj*) and restart it later *with the same timestamp;* otherwise (*Ti* younger than *Tj*) *Ti* is allowed to wait.

Deadlock Detection. A second, more practical approach to dealing with deadlock is **deadlock detection**, where the system checks if a state of deadlock actually exists. This solution is attractive if we know there will be little interference among the transactions—that is, if different transactions will rarely access the same items at the same time. A simple way to detect a state of deadlock is for the system to construct and maintain a **wait-for graph**. One node is created in the wait-for graph for each transaction that is currently executing.

Timeouts. Another simple scheme to deal with deadlock is the use of **timeouts**. This method is practical because of its low overhead and simplicity. In this method, if a transaction waits for a

period longer than a system-defined timeout period, the system assumes that the transaction may be deadlocked and aborts it—regardless of whether a deadlock actually exists or not.

Starvation. Another problem that may occur when we use locking is **starvation**, which occurs when a transaction cannot proceed for an indefinite period of time while other transactions in the system continue normally. This may occur if the waiting scheme for locked items is unfair, giving priority to some transactions over others. One solution for starvation is to have a fair waiting scheme, such as using a **first-come-first-served** queue.

KARPAGAM ACADEMY OF HIGHER EDUCATION

CLASS: II BCom CA **COURSENAME:DATABASEMANAGEMENT SYSTEM**

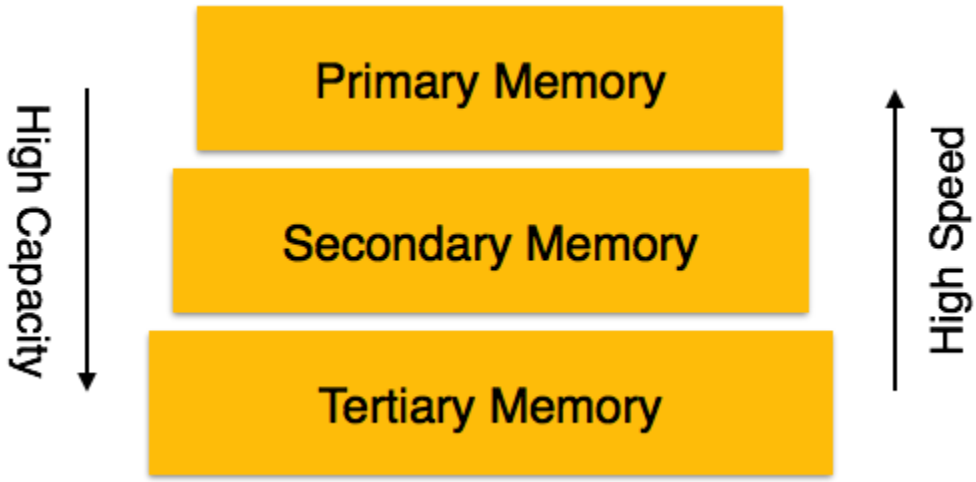
COURSE CODE: 18CCU302 **BATCH :2018-21**

UNIT-IV

SYLLABUS

TRENDS IN DATABASE TECHNOLOGY Overview of Physical Storage Media – Magnetic Disks – RAID – Tertiary storage – File Organization – Organization of Records in Files – Indexing and Hashing –Ordered Indices – B+ tree Index Files – B tree Index Files – Static Hashing – Dynamic Hashing - Introduction to Distributed Databases- Client server technology- Multidimensional and Parallel databasesSpatial and multimedia databases- Mobile and web

Databases are stored in file formats, which contain records. At physical level, the actual data is stored in electromagnetic format on some device. These storage devices can be broadly categorized into three types −



- **Primary Storage** − The memory storage that is directly accessible to the CPU comes under this category. CPU's internal memory (registers), fast memory (cache), and main memory (RAM) are directly accessible to the CPU, as they are all placed on the motherboard or CPU chipset. This storage is typically very small, ultra-fast, and volatile. Primary storage requires continuous power supply in order to maintain its state. In case of a power failure, all its data is lost.
- **Secondary Storage** − Secondary storage devices are used to store data for future use or as backup. Secondary storage includes memory devices that are not a part of the CPU chipset or motherboard, for example, magnetic disks, optical disks (DVD, CD, etc.), hard disks, flash drives, and magnetic tapes.
- **Tertiary Storage** − Tertiary storage is used to store huge volumes of data. Since such storage devices are external to the computer system, they are the slowest in speed. These storage

devices are mostly used to take the back up of an entire system. Optical disks and magnetic tapes are widely used as tertiary storage.

Memory Hierarchy

A computer system has a well-defined hierarchy of memory. A CPU has direct access to it main memory as well as its inbuilt registers. The access time of the main memory is obviously less than the CPU speed. To minimize this speed mismatch, cache memory is introduced. Cache memory provides the fastest access time and it contains data that is most frequently accessed by the CPU.

The memory with the fastest access is the costliest one. Larger storage devices offer slow speed and they are less expensive, however they can store huge volumes of data as compared to CPU registers or cache memory.

Magnetic Disks

Hard disk drives are the most common secondary storage devices in present computer systems. These are called magnetic disks because they use the concept of magnetization to store information. Hard disks consist of metal disks coated with magnetizable material. These disks are placed vertically on a spindle. A read/write head moves in between the disks and is used to magnetize or de-magnetize the spot under it. A magnetized spot can be recognized as 0 (zero) or 1 (one).

Hard disks are formatted in a well-defined order to store data efficiently. A hard disk plate has many concentric circles on it, called **tracks**. Every track is further divided into **sectors**. A sector on a hard disk typically stores 512 bytes of data.

Redundant Array of Independent Disks

RAID or **R**edundant **A**rray of **I**ndependent **D**isks, is a technology to connect multiple secondary storage devices and use them as a single storage media.

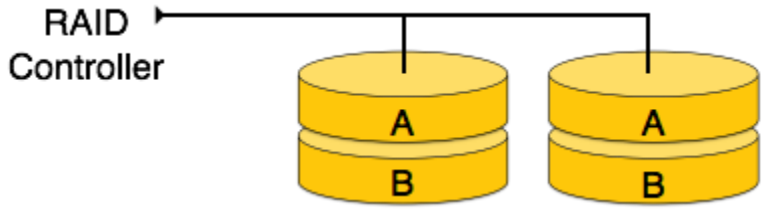
RAID consists of an array of disks in which multiple disks are connected together to achieve different goals. RAID levels define the use of disk arrays.

RAID 0

In this level, a striped array of disks is implemented. The data is broken down into blocks and the blocks are distributed among disks. Each disk receives a block of data to write/read in parallel. It enhances the speed and performance of the storage device. There is no parity and backup in Level 0.

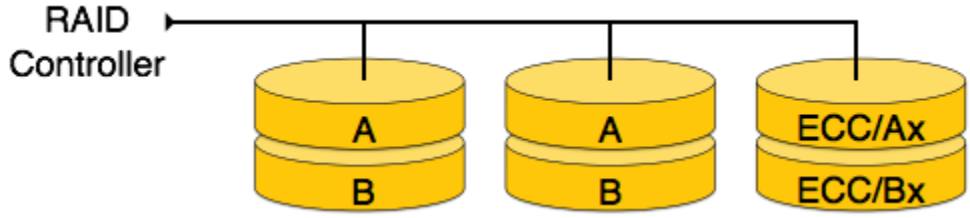
RAID 1

RAID 1 uses mirroring techniques. When data is sent to a RAID controller, it sends a copy of data to all the disks in the array. RAID level 1 is also called **mirroring** and provides 100% redundancy in case of a failure.



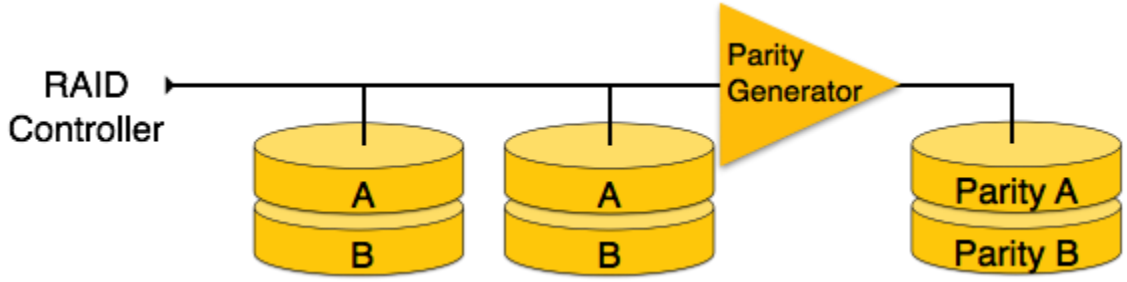
RAID 2

RAID 2 records Error Correction Code using Hamming distance for its data, striped on different disks. Like level 0, each data bit in a word is recorded on a separate disk and ECC codes of the data words are stored on a different set disks. Due to its complex structure and high cost, RAID 2 is not commercially available.



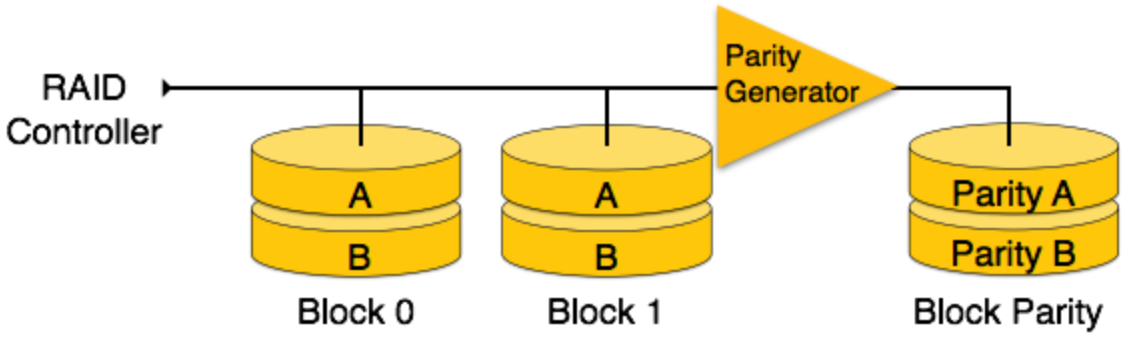
RAID 3

RAID 3 stripes the data onto multiple disks. The parity bit generated for data word is stored on a different disk. This technique makes it to overcome single disk failures.



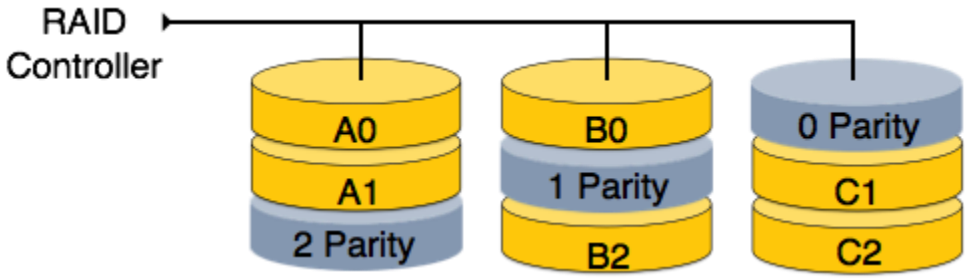
RAID 4

In this level, an entire block of data is written onto data disks and then the parity is generated and stored on a different disk. Note that level 3 uses byte-level striping, whereas level 4 uses block-level striping. Both level 3 and level 4 require at least three disks to implement RAID.



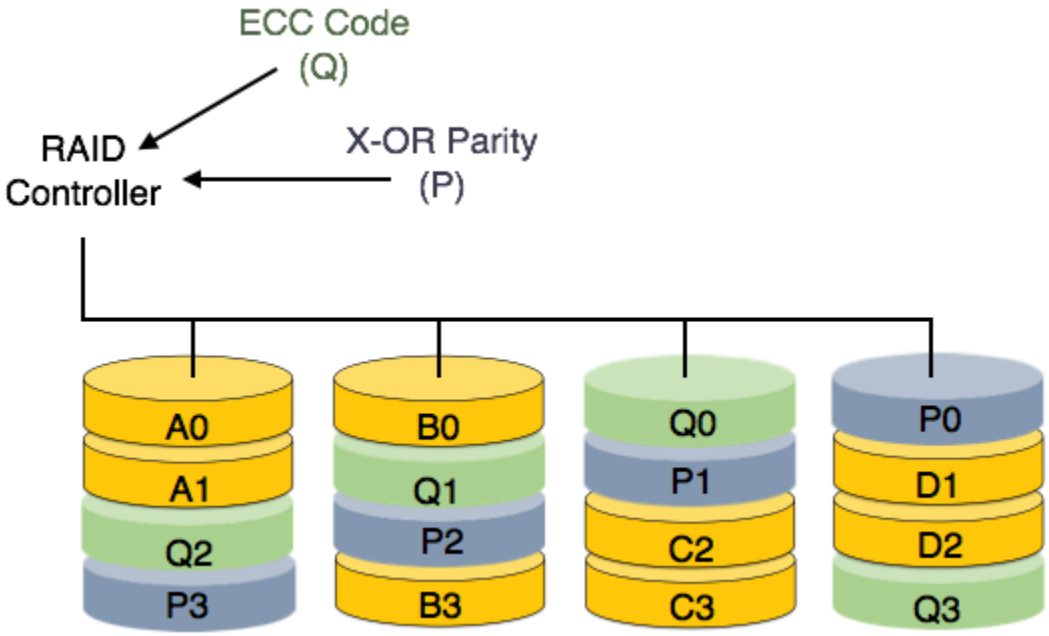
RAID 5

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



RAID 6

RAID 6 is an extension of level 5. In this level, two independent parities are generated and stored in distributed fashion among multiple disks. Two parities provide additional fault tolerance. This level requires at least four disk drives to implement RAID.

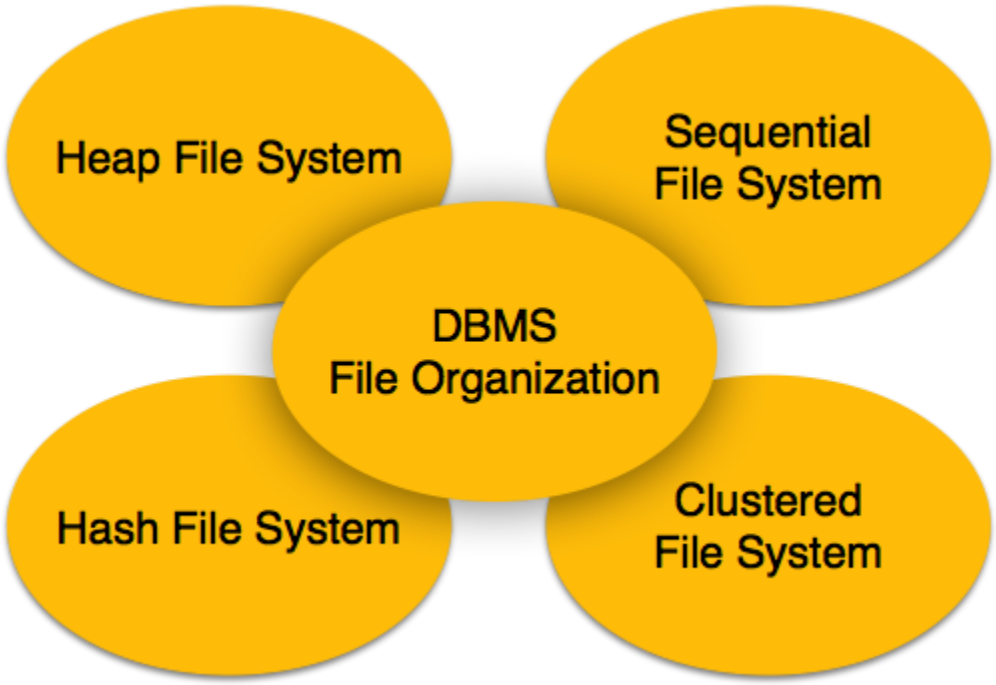


FILE ORGANISATION :

Relative data and information is stored collectively in file formats. A file is a sequence of records stored in binary format. A disk drive is formatted into several blocks that can store records. File records are mapped onto those disk blocks.

File Organization

File Organization defines how file records are mapped onto disk blocks. We have four types of File Organization to organize file records −



Heap File Organization

When a file is created using Heap File Organization, the Operating System allocates memory area to that file without any further accounting details. File records can be placed anywhere in that memory area. It is the responsibility of the software to manage the records. Heap File does not support any ordering, sequencing, or indexing on its own.

Sequential File Organization

Every file record contains a data field (attribute) to uniquely identify that record. In sequential file organization, records are placed in the file in some sequential order based on the unique key field or search key. Practically, it is not possible to store all the records sequentially in physical form.

Hash File Organization

Hash File Organization uses Hash function computation on some fields of the records. The output of the hash function determines the location of disk block where the records are to be placed.

Clustered File Organization

Clustered file organization is not considered good for large databases. In this mechanism, related records from one or more relations are kept in the same disk block, that is, the ordering of records is not based on primary key or search key.

File Operations

Operations on database files can be broadly classified into two categories −

- **Update Operations**
- **Retrieval Operations**

Update operations change the data values by insertion, deletion, or update. Retrieval operations, on the other hand, do not alter the data but retrieve them after optional conditional filtering. In both types of operations, selection plays a significant role. Other than creation and deletion of a file, there could be several operations, which can be done on files.

- **Open** − A file can be opened in one of the two modes, **read mode** or **write mode**. In read mode, the operating system does not allow anyone to alter data. In other words, data is read only. Files opened in read mode can be shared among several entities. Write mode allows data modification. Files opened in write mode can be read but cannot be shared.
- **Locate** − Every file has a file pointer, which tells the current position where the data is to be read or written. This pointer can be adjusted accordingly. Using find (seek) operation, it can be moved forward or backward.
- **Read** − By default, when files are opened in read mode, the file pointer points to the beginning of the file. There are options where the user can tell the operating system where to locate the file pointer at the time of opening a file. The very next data to the file pointer is read.
- **Write** − User can select to open a file in write mode, which enables them to edit its contents. It can be deletion, insertion, or modification. The file pointer can be located at the time of opening or can be dynamically changed if the operating system allows to do so.
- **Close** − This is the most important operation from the operating system's point of view. When a request to close a file is generated, the operating system
    - removes all the locks (if in shared mode),
    - saves the data (if altered) to the secondary storage media, and
    - releases all the buffers and file handlers associated with the file.

The organization of data inside a file plays a major role here. The process to locate the file pointer to a desired record inside a file various based on whether the records are arranged sequentially or clustered.

HASHING

For a huge database structure, it can be almost next to impossible to search all the index values through all its level and then reach the destination data block to retrieve the desired data. Hashing is an effective technique to calculate the direct location of a data record on the disk without using index structure.

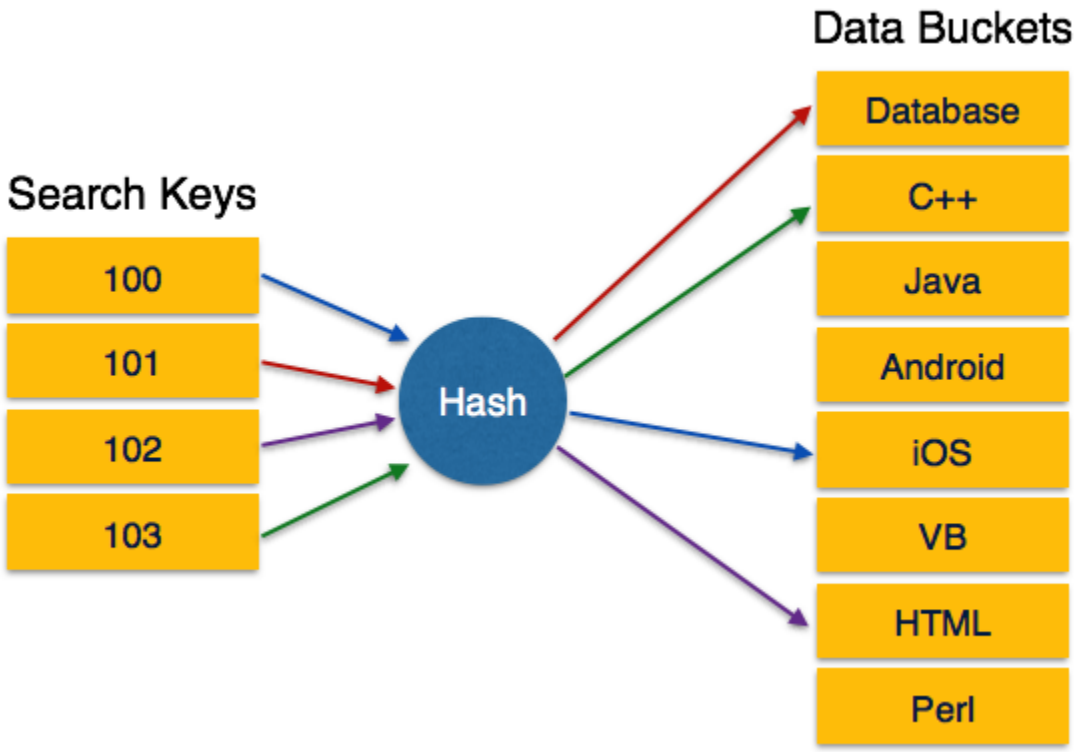
Hashing uses hash functions with search keys as parameters to generate the address of a data record.

Hash Organization

- **Bucket** − A hash file stores data in bucket format. Bucket is considered a unit of storage. A bucket typically stores one complete disk block, which in turn can store one or more records.
- **Hash Function** − A hash function, **h,** is a mapping function that maps all the set of search-keys **K** to the address where actual records are placed. It is a function from search keys to bucket addresses.

Static Hashing

In static hashing, when a search-key value is provided, the hash function always computes the same address. For example, if mod-4 hash function is used, then it shall generate only 5 values. The output address shall always be same for that function. The number of buckets provided remains unchanged at all times.



Operation

- **Insertion** − When a record is required to be entered using static hash, the hash function **h** computes the bucket address for search key **K**, where the record will be stored.

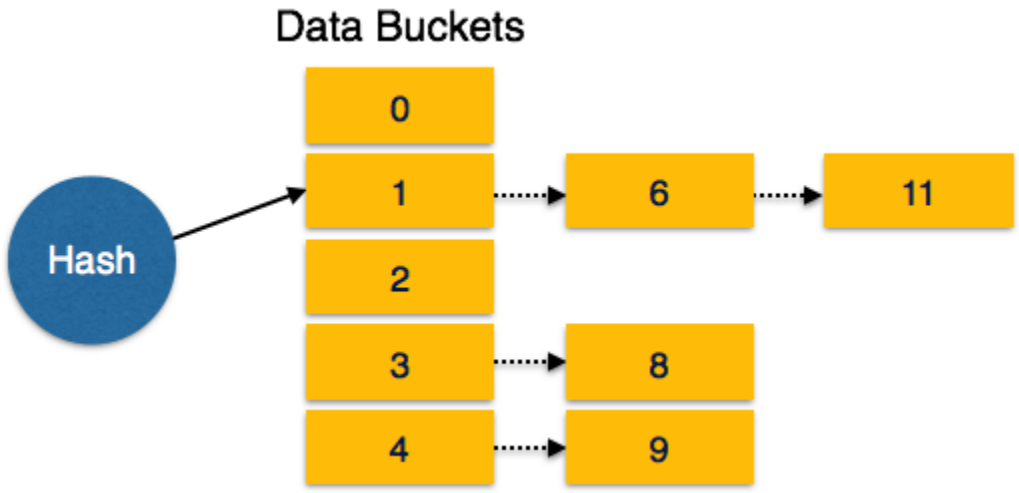
  Bucket address = h(K)

- **Search** − When a record needs to be retrieved, the same hash function can be used to retrieve the address of the bucket where the data is stored.
- **Delete** − This is simply a search followed by a deletion operation.

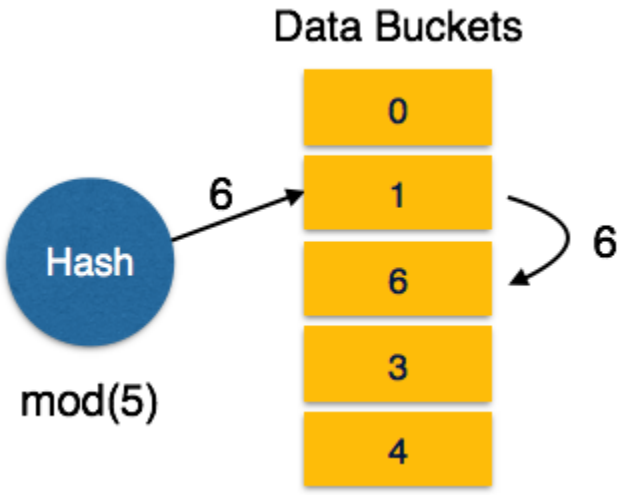
Bucket Overflow

The condition of bucket-overflow is known as **collision**. This is a fatal state for any static hash function. In this case, overflow chaining can be used.

- **Overflow Chaining** − When buckets are full, a new bucket is allocated for the same hash result and is linked after the previous one. This mechanism is called **Closed Hashing**.



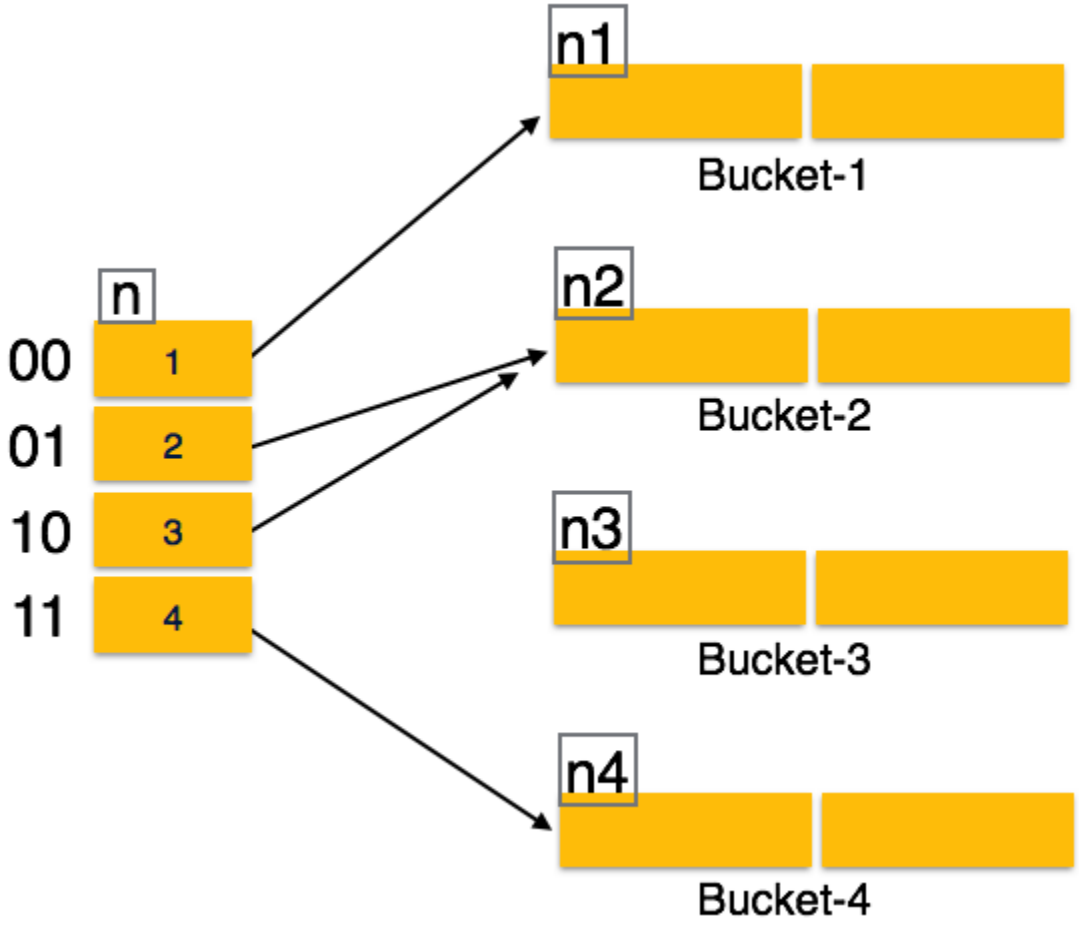
- **Linear Probing** − When a hash function generates an address at which data is already stored, the next free bucket is allocated to it. This mechanism is called **Open Hashing**.



Dynamic Hashing

The problem with static hashing is that it does not expand or shrink dynamically as the size of the database grows or shrinks. Dynamic hashing provides a mechanism in which data buckets are added and removed dynamically and on-demand. Dynamic hashing is also known as **extended hashing**.

Hash function, in dynamic hashing, is made to produce a large number of values and only a few are used initially.



Organization

The prefix of an entire hash value is taken as a hash index. Only a portion of the hash value is used for computing bucket addresses. Every hash index has a depth value to signify how many bits are used for computing a hash function. These bits can address 2n buckets. When all these bits are consumed − that is, when all the buckets are full − then the depth value is increased linearly and twice the buckets are allocated.

Operation

- **Querying** − Look at the depth value of the hash index and use those bits to compute the bucket address.
- **Update** − Perform a query as above and update the data.
- **Deletion** − Perform a query to locate the desired data and delete the same.
- **Insertion** − Compute the address of the bucket
    - If the bucket is already full.

        - Add more buckets.
        - Add additional bits to the hash value.
        - Re-compute the hash function.
    - Else
        - Add data to the bucket,
    - If all the buckets are full, perform the remedies of static hashing.

Hashing is not favorable when the data is organized in some ordering and the queries require a range of data. When data is discrete and random, hash performs the best.

Hashing algorithms have high complexity than indexing. All hash operations are done in constant time

INDEXING :

We know that data is stored in the form of records. Every record has a key field, which helps it to be recognized uniquely.

Indexing is a data structure technique to efficiently retrieve records from the database files based on some attributes on which the indexing has been done. Indexing in database systems is similar to what we see in books.

Indexing is defined based on its indexing attributes. Indexing can be of the following types −

- **Primary Index** − Primary index is defined on an ordered data file. The data file is ordered on a **key field**. The key field is generally the primary key of the relation.
- **Secondary Index** − Secondary index may be generated from a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values.
- **Clustering Index** − Clustering index is defined on an ordered data file. The data file is ordered on a non-key field.

Ordered Indexing is of two types −

- Dense Index
- Sparse Index

Dense Index

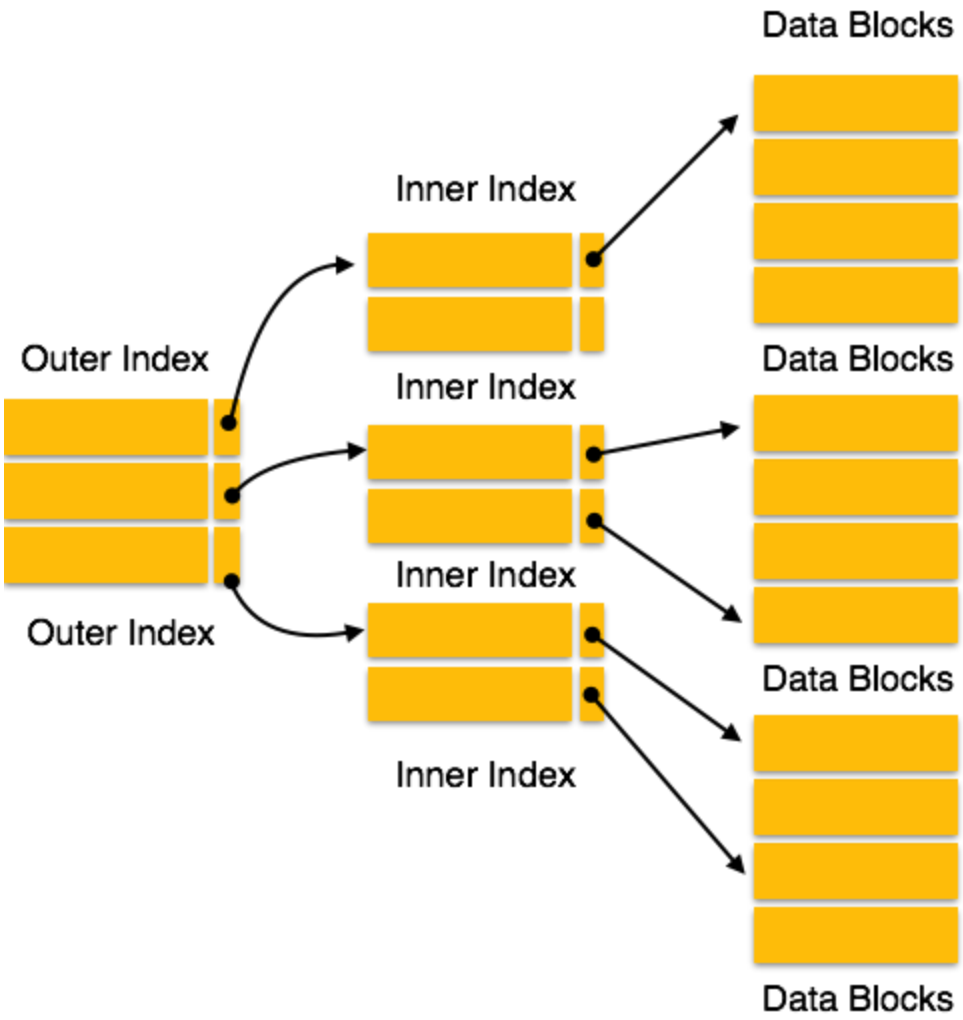
In dense index, there is an index record for every search key value in the database. This makes searching faster but requires more space to store index records itself. Index records contain search key value and a pointer to the actual record on the disk.

Sparse Index

In sparse index, index records are not created for every search key. An index record here contains a search key and an actual pointer to the data on the disk. To search a record, we first proceed by index record and reach at the actual location of the data. If the data we are looking for is not where we directly reach by following the index, then the system starts sequential search until the desired data is found.

Multilevel Index

Index records comprise search-key values and data pointers. Multilevel index is stored on the disk along with the actual database files. As the size of the database grows, so does the size of the indices. There is an immense need to keep the index records in the main memory so as to speed up the search operations. If single-level index is used, then a large size index cannot be kept in memory which leads to multiple disk accesses.



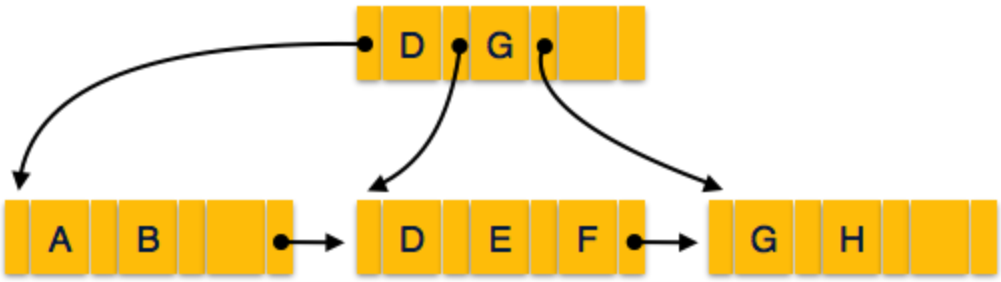
Multi-level Index helps in breaking down the index into several smaller indices in order to make the outermost level so small that it can be saved in a single disk block, which can easily be accommodated anywhere in the main memory.

B^+ Tree

A B^+ tree is a balanced binary search tree that follows a multi-level index format. The leaf nodes of a B^+ tree denote actual data pointers. B^+ tree ensures that all leaf nodes remain at the same height, thus balanced. Additionally, the leaf nodes are linked using a link list; therefore, a B^+ tree can support random access as well as sequential access.

Structure of B^+ Tree

Every leaf node is at equal distance from the root node. A B^+ tree is of the order **n** where **n** is fixed for every B^+ tree.



Internal nodes −

- Internal (non-leaf) nodes contain at least ⌈n/2⌉ pointers, except the root node.
- At most, an internal node can contain **n** pointers.

Leaf nodes −

- Leaf nodes contain at least ⌈n/2⌉ record pointers and ⌈n/2⌉ key values.
- At most, a leaf node can contain **n** record pointers and **n** key values.
- Every leaf node contains one block pointer **P** to point to next leaf node and forms a linked list.

B^+ Tree Insertion

- B^+ trees are filled from bottom and each entry is done at the leaf node.
- If a leaf node overflows −
  - Split node into two parts.
  - Partition at $\mathbf{i = \lfloor (m+1)/2 \rfloor}$.
  - First **i** entries are stored in one node.
  - Rest of the entries (i+1 onwards) are moved to a new node.
  - $\mathbf{i^{th}}$ key is duplicated at the parent of the leaf.
- If a non-leaf node overflows −
  - Split node into two parts.
  - Partition the node at $\mathbf{i = \lceil (m+1)/2 \rceil}$.
  - Entries up to **i** are kept in one node.
  - Rest of the entries are moved to a new node.

B^+ Tree Deletion

- B^+ tree entries are deleted at the leaf nodes.
- The target entry is searched and deleted.
  - If it is an internal node, delete and replace with the entry from the left position.
- After deletion, underflow is tested,

    - If underflow occurs, distribute the entries from the nodes left to it.
- If distribution is not possible from left, then
    - Distribute from the nodes right to it.
- If distribution is not possible from left or from right, then
    - Merge the node with left and right to it.

INTRODUCTION DISTRIBUTED DATABASES:

This chapter introduces the concept of DDBMS. In a distributed database, there are a number of databases that may be geographically distributed all over the world. A distributed DBMS manages the distributed database in a manner so that it appears as one single database to users. In the later part of the chapter, we go on to study the factors that lead to distributed databases, its advantages and disadvantages.

A **distributed database** is a collection of multiple interconnected databases, which are spread physically across various locations that communicate via a computer network.

Features

- Databases in the collection are logically interrelated with each other. Often they represent a single logical database.
- Data is physically stored across multiple sites. Data in each site can be managed by a DBMS independent of the other sites.
- The processors in the sites are connected via a network. They do not have any multiprocessor configuration.
- A distributed database is not a loosely connected file system.
- A distributed database incorporates transaction processing, but it is not synonymous with a transaction processing system.

Distributed Database Management System

A distributed database management system (DDBMS) is a centralized software system that manages a distributed database in a manner as if it were all stored in a single location.

Features

- It is used to create, retrieve, update and delete distributed databases.
- It synchronizes the database periodically and provides access mechanisms by the virtue of which the distribution becomes transparent to the users.
- It ensures that the data modified at any site is universally updated.
- It is used in application areas where large volumes of data are processed and accessed by numerous users simultaneously.

- It is designed for heterogeneous database platforms.
- It maintains confidentiality and data integrity of the databases.

Factors Encouraging DDBMS

The following factors encourage moving over to DDBMS −

- **Distributed Nature of Organizational Units** − Most organizations in the current times are subdivided into multiple units that are physically distributed over the globe. Each unit requires its own set of local data. Thus, the overall database of the organization becomes distributed.
- **Need for Sharing of Data** − The multiple organizational units often need to communicate with each other and share their data and resources. This demands common databases or replicated databases that should be used in a synchronized manner.
- **Support for Both OLTP and OLAP** − Online Transaction Processing (OLTP) and Online Analytical Processing (OLAP) work upon diversified systems which may have common data. Distributed database systems aid both these processing by providing synchronized data.
- **Database Recovery** − One of the common techniques used in DDBMS is replication of data across different sites. Replication of data automatically helps in data recovery if database in any site is damaged. Users can access data from other sites while the damaged site is being reconstructed. Thus, database failure may become almost inconspicuous to users.
- **Support for Multiple Application Software** − Most organizations use a variety of application software each with its specific database support. DDBMS provides a uniform functionality for using the same data among different platforms.

Advantages of Distributed Databases

Following are the advantages of distributed databases over centralized databases.

Modular Development − If the system needs to be expanded to new locations or new units, in centralized database systems, the action requires substantial efforts and disruption in the existing functioning. However, in distributed databases, the work simply requires adding new computers and local data to the new site and finally connecting them to the distributed system, with no interruption in current functions.

More Reliable − In case of database failures, the total system of centralized databases comes to a halt. However, in distributed systems, when a component fails, the functioning of the system continues may be at a reduced performance. Hence DDBMS is more reliable.

Better Response − If data is distributed in an efficient manner, then user requests can be met from local data itself, thus providing faster response. On the other hand, in centralized systems, all queries have to pass through the central computer for processing, which increases the response time.

Lower Communication Cost − In distributed database systems, if data is located locally where it is mostly used, then the communication costs for data manipulation can be minimized. This is not feasible in centralized systems.

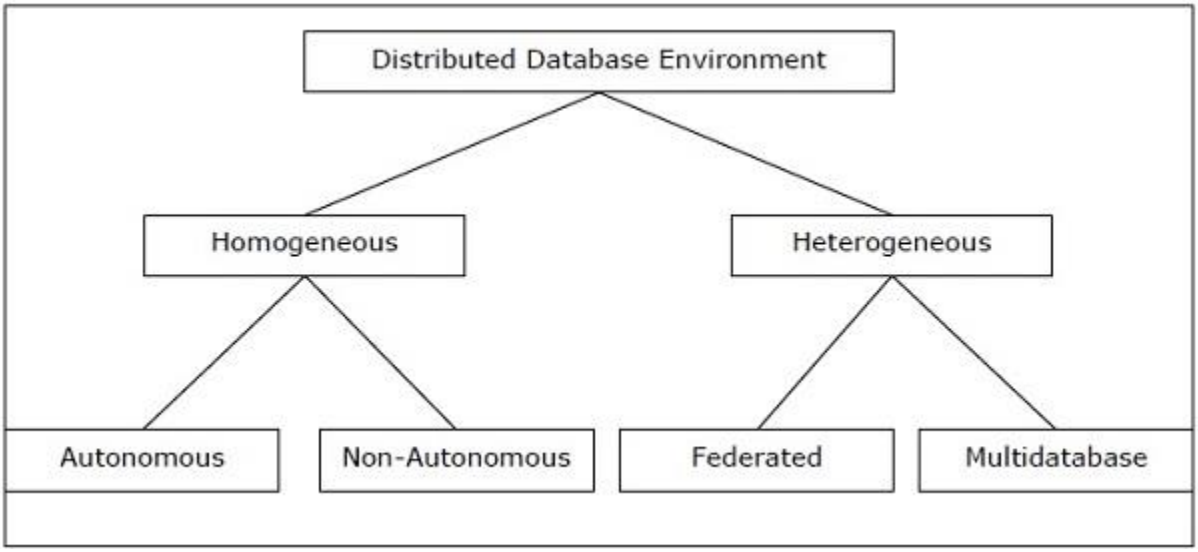
Adversities of Distributed Databases

Following are some of the adversities associated with distributed databases.

- **Need for complex and expensive software** − DDBMS demands complex and often expensive software to provide data transparency and co-ordination across the several sites.
- **Processing overhead** − Even simple operations may require a large number of communications and additional calculations to provide uniformity in data across the sites.
- **Data integrity** − The need for updating data in multiple sites pose problems of data integrity.
- **Overheads for improper data distribution** − Responsiveness of queries is largely dependent upon proper data distribution. Improper data distribution often leads to very slow response to user requests.

Types of Distributed Databases

Distributed databases can be broadly classified into homogeneous and heterogeneous distributed database environments, each with further sub-divisions, as shown in the following illustration.



Homogeneous Distributed Databases

In a homogeneous distributed database, all the sites use identical DBMS and operating systems. Its properties are −

- The sites use very similar software.
- The sites use identical DBMS or DBMS from the same vendor.
- Each site is aware of all other sites and cooperates with other sites to process user requests.
- The database is accessed through a single interface as if it is a single database.

Types of Homogeneous Distributed Database

There are two types of homogeneous distributed database −

- **Autonomous** − Each database is independent that functions on its own. They are integrated by a controlling application and use message passing to share data updates.
- **Non-autonomous** − Data is distributed across the homogeneous nodes and a central or master DBMS co-ordinates data updates across the sites.

Heterogeneous Distributed Databases

In a heterogeneous distributed database, different sites have different operating systems, DBMS products and data models. Its properties are −

- Different sites use dissimilar schemas and software.
- The system may be composed of a variety of DBMSs like relational, network, hierarchical or object oriented.
- Query processing is complex due to dissimilar schemas.
- Transaction processing is complex due to dissimilar software.
- A site may not be aware of other sites and so there is limited co-operation in processing user requests.

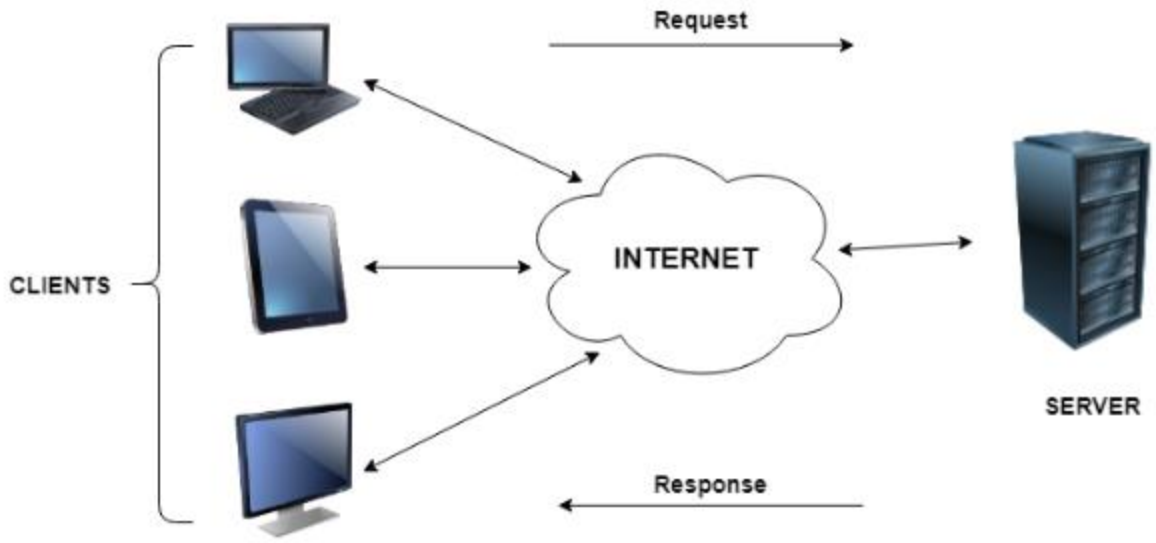
Types of Heterogeneous Distributed Databases

- **Federated** − The heterogeneous database systems are independent in nature and integrated together so that they function as a single database system.
- **Un-federated** − The database systems employ a central coordinating module through which the databases are accessed.

CLIENT SERVER TECHNOLOGY

In client server computing, the clients requests a resource and the server provides that resource. A server may serve multiple clients at the same time while a client is in contact with only one server. Both the client and server usually communicate via a computer network but sometimes they may reside in the same system.

An illustration of the client server system is given as follows:



Characteristics of Client Server Computing

The salient points for client server computing are as follows:

- The client server computing works with a system of request and response. The client sends a request to the server and the server responds with the desired information.
- The client and server should follow a common communication protocol so they can easily interact with each other. All the communication protocols are available at the application layer.
- A server can only accommodate a limited number of client requests at a time. So it uses a system based to priority to respond to the requests.
- Denial of Service attacks hindera servers ability to respond to authentic client requests by inundating it with false requests.
- An example of a client server computing system is a web server. It returns the web pages to the clients that requested them.

Difference between Client Server Computing and Peer to Peer Computing

The major differences between client server computing and peer to peer computing are as follows:

- In client server computing, a server is a central node that services many client nodes. On the other hand, in a peer to peer system, the nodes collectively use their resources and communicate with each other.
- In client server computing the server is the one that communicates with the other nodes. In peer to peer to computing, all the nodes are equal and share data with each other directly.
- Client Server computing is believed to be a subcategory of the peer to peer computing.

Advantages of Client Server Computing

The different advantages of client server computing are:

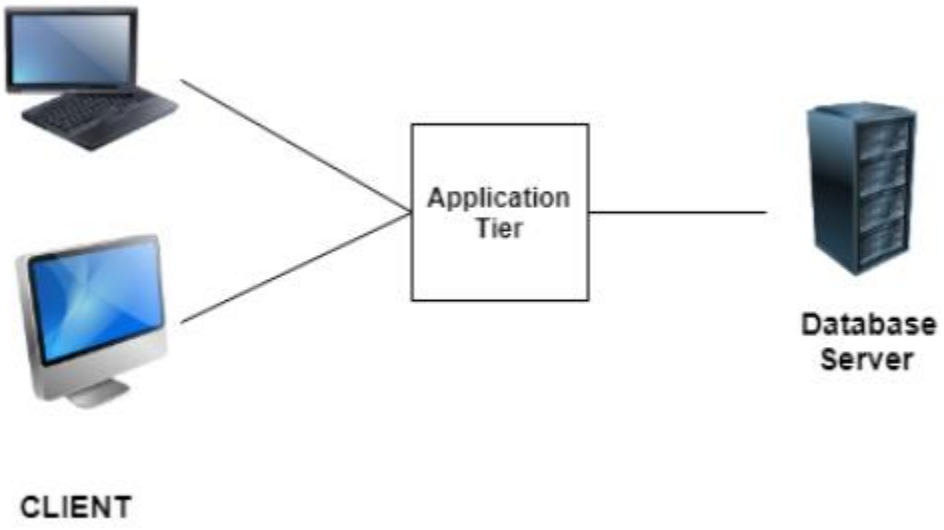
- All the required data is concentrated in a single place i.e. the server. So it is easy to protect the data and provide authorisation and authentication.
- The server need not be located physically close to the clients. Yet the data can be accessed efficiently.
- It is easy to replace, upgrade or relocate the nodes in the client server model because all the nodes are independent and request data only from the server.
- All the nodes i.e clients and server may not be build on similar platforms yet they can easily facilitate the transfer of data.

Disadvantages of Client Server Computing

The different disadvantages of client server computing are:

- If all the clients simultaneously request data from the server, it may get overloaded. This may lead to congestion in the network.
- If the server fails for any reason, then none of the requests of the clients can be fulfilled. This leads of failure of the client server network.
- The cost of setting and maintaining a client server model are quite high.

follows:



Advantages of Three - Tier Client/Server Structure

Some of the advantages of the three-tier client/server structure are:

- The three tier structure provides much better service and fast performance.
- The structure can be scaled according to requirements without any problem.
- Data security is much improved in the three tier structure.

Disadvantages of Three - Tier Client/Server Structure

A major disadvantage of the three-tier client/server structure is:

- Three - tier client/server structure is quite complex due to advanced features.

MULTIDIMENSIONAL DATABASES :

A database management system (DBMS) organized around groups of records that share a common field value. Multidimensional databases are often generated from relational databases. Whereas relational databases make it easy to work with individual records, multidimensional databases are designed for analyzing large groups of records.

The term *OLAP (On-Line Analytical Processing)* has become almost synonymous with multidimensional databases, whereas *OLTP (On-Line Transaction Processing)* generally refers to relational DBMSs.

Multidimensional databases are used mostly for OLAP (online analytical processing) and data warehousing. They can be used to show multiple dimensions of data to users .

A multidimensional database is created from multiple relational databases. While relational databases allow users to access data in the form of queries, the multidimensional databases allow users to ask analytical questions related to business or market trends.

The multidimensional databases uses MOLAP (multidimensional online analytical processing) to access its data. They allow the users to quickly get answers to their requests by generating and analysing the data rather quickly.

The data in multidimensional databases is stored in a data cube format. This means that data can be seen and understood from many dimensions and perspectives.

The revenue costs for a company can be understood and analyzed on the basis of various factors like the company products, the geographical locations of the company offices, time to develop a product, promotions done etc.

Advantages of Multidimensional Databases

Some advantages of multidimensional databases are:

Increased performance

The performance of the multidimensional databases is much superior to that of normal databases such as relational database.

Easy maintenance

The multidimensional database is easy to handle and maintain

Better data presentation

The data in a multidimensional database is multi faceted and contains many different factors. Hence, the data presentation is far superior to conventional databases.

Disadvantages of Multidimensional Databases

One of the disadvantage of multidimensional databases are that it is quite complex and it takes professionals to truly understand and analyse the data in the database.

PARALLEL DATABASES :

Introduction to Parallel Databases

Companies need to handle huge amount of data with high data transfer rate. The client server and centralized system is not much efficient. The need to improve the efficiency gave birth to the concept of Parallel Databases.

Parallel database system improves performance of data processing using multiple resources in parallel, like multiple CPU and disks are used parallely.

It also performs many parallelization operations like, data loading and query processing.

Goals of Parallel Databases

The concept of Parallel Database was built with a goal to:

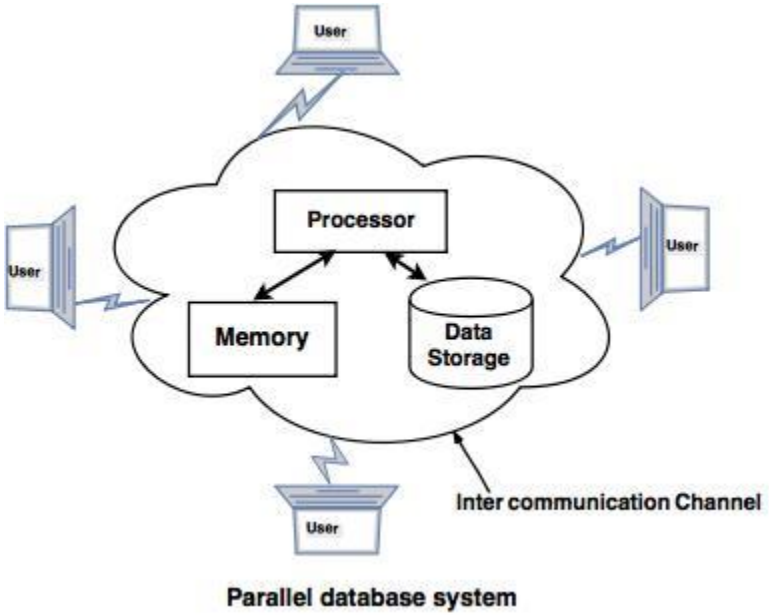
Improve performance:
The performance of the system can be improved by connecting multiple CPU and disks in parallel. Many small processors can also be connected in parallel.

Improve availability of data:
Data can be copied to multiple locations to improve the availability of data.
For example: if a module contains a relation (table in database) which is unavailable then it is important to make it available from another module.

Improve reliability:
Reliability of system is improved with completeness, accuracy and availability of data.

Provide distributed access of data:
Companies having many branches in multiple cities can access data with the help of parallel database

system.



Parallel database system

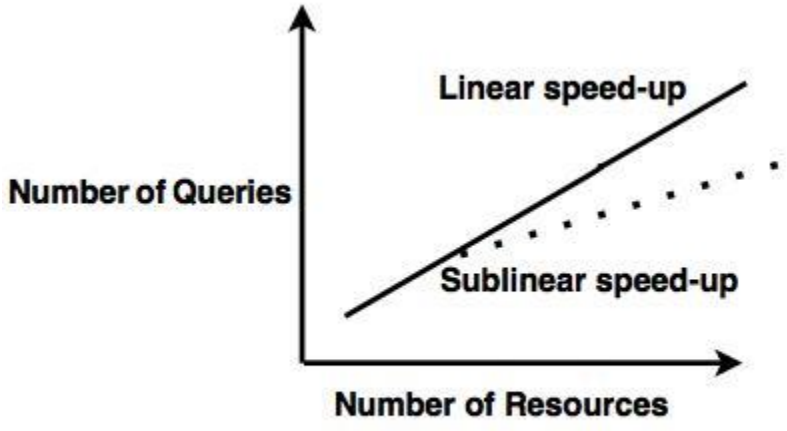
Parameters for Parallel Databases

Some parameters to judge the performance of Parallel Databases are:
1. Response time: It is the time taken to complete a single task for given time.

2. Speed up in Parallel database:

- Speed up is the process of increasing degree of (resources) parallelism to complete a running task in less time.
- The time required for running task is inversely proportional to number of resources.

Formula:
Speed up = T_S / T_L
Where,
T_S = Time required to execute task of size Q
T_L = Time required to execute task of size N*Q



Speed-up in Parallel Databases

- Linear speed-up is N (Number of resources).
- Speed-up is sub-linear if speed-up is less than N.

3. Scale up in Parallel database:

Scale-up is the ability to keep performance constant, when number of process and resources increases proportionally.

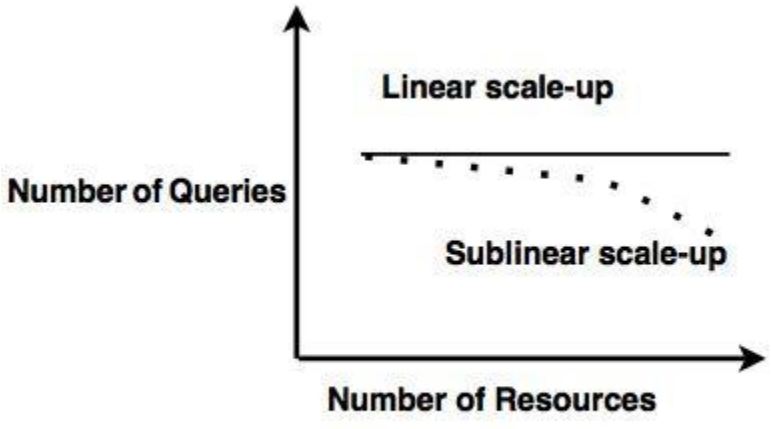
Formula:

Let Q be the Task and Q_N the task where N is greater than Q

T_S = Execution time of task Q on smaller machine M_S

T_L = Execution time of task Q on smaller machine M_L

Scale Up = T_S / T_L



Scale-up in Parallel Databases

SPATIAL DATABASE:

Spatial data is associated with geographic locations such as cities,towns etc. A spatial database is optimized to store and query data representing objects. These are the objects which are defined in a geometric space.

Characteristics of Spatial Database

A spatial database system has the following characteristics

1. It is a database system
2. It offers spatial data types (SDTs) in its data model and query language.
3. It supports spatial data types in its implementation, providing at least spatial indexing and efficient algorithms for spatial join.

Example

A road map is a visualization of geographic information. A road map is a 2-dimensional object which contains points, lines, and polygons that can represent cities, roads, and political boundaries such as states or provinces.

In general, spatial data can be of two types:

1. Vector data: This data is represented as discrete points, lines and polygons
2. Rastor data: This data is represented as a matrix of square cells.

MUTIMEDIA DATABASE:

The multimedia databases are used to store multimedia data such as images, animation, audio, video along with text. This data is stored in the form of multiple file types like .txt(text), .jpg(images), .swf(videos), .mp3(audio) etc.

Contents of the Multimedia Database

The multimedia database stored the multimedia data and information related to it. This is given in detail as follows:

Media data

This is the multimedia data that is stored in the database such as images, videos, audios, animation etc.

Media format data

The Media format data contains the formatting information related to the media data such as sampling rate, frame rate, encoding scheme etc.

Media keyword data

This contains the keyword data related to the media in the database. For an image the keyword data can be date and time of the image, description of the image etc.

Media feature data

Th Media feature data describes the features of the media data. For an image, feature data can be colours of the image, textures in the image etc.

Challenges of Multimedia Database

There are many challenges to implement a multimedia database. Some of these are:

1. Multimedia databases contains data in a large type of formats such as .txt(text), .jpg(images), .swf(videos), .mp3(audio) etc. It is difficult to convert one type of data format to another.
2. The multimedia database requires a large size as the multimedia data is quite large and needs to be stored successfully in the database.
3. It takes a lot of time to process multimedia data so multimedia database is slow.

UNIT V ADVANCED TOPICS

DATABASE SECURITY: Data Classification-Threats and risks – Database access Control – Types of Privileges –Cryptography- Statistical Databases.- Distributed Databases-Architecture-Transaction Processing-Data Warehousing and Mining-Classification-Association rules-Clustering-Information Retrieval- Relevance ranking-Crawling and Indexing the Web- Object Oriented Databases-XML Databases.

DATA CLASSIFICATION :

Data classification is broadly defined as the process of organizing data by relevant categories so that it may be used and protected more efficiently. On a basic level, the **classification process** makes data easier to locate and retrieve. Data classification is of particular importance when it comes to risk management, compliance, and data security.

Data classification involves tagging data to make it easily searchable and trackable. It also eliminates multiple duplications of data, which can reduce storage and backup costs while speeding up the search process.

Here are **three main types of data classification** that are considered industry standards:

- **Content**-based classification inspects and interprets files looking for sensitive information
- **Context**-based classification looks at application, location, or creator among other variables as indirect indicators of sensitive information
- **User**-based classification depends on a manual, end-user selection of each document. User-based classification relies on user knowledge and discretion at creation, edit, review, or dissemination to flag sensitive documents.

Content-, context-, and user-based approaches can be both right or wrong depending on the business need and data type.

AN EXAMPLE OF DATA CLASSIFICATION

An organization may classify data as Restricted, Private or Public. In this instance, public data represents the least-sensitive data with the lowest security requirements, while restricted data is in the highest security classification and represents the most sensitive data. This type of data classification is often the starting point for many enterprises, followed by additional identification and tagging procedures that label data based on its relevance to the enterprise, quality, and other classifications. **The most successful data classification processes** employ follow-up processes and frameworks to keep sensitive data where it belongs.

THE DATA CLASSIFICATION PROCESS

Data classification can be a complex and cumbersome process. Automated systems can help streamline the process, but an enterprise must determine the categories and criteria that will be used to classify data, understand and define its objectives, outline the roles and responsibilities of employees in maintaining proper **data classification protocols**, and implement security standards that correspond with data categories and tags. When done correctly, this process will provide employees and third parties involved in the storage, transmission, or retrieval of data with an operational framework.

Database security is the technique that protects and secures the database against intentional or accidental threats. Security concerns will be relevant not only to the data resides in an organization's database: the breaking of security may harm other parts of the system which may ultimately affect the database structure. Consequently, database security includes hardware part, software part, human resource, and data. To

efficiently do the uses of security needs appropriate controls, which are distinct in a specific mission and purpose for the system. The requirement for getting proper security while often having been neglected or overlooked in the past days; is now more and more thoroughly checked by the different organizations.

We consider database security about the following situations:

- Theft and fraudulent.
- Loss of confidentiality or secrecy.
- Loss of data privacy.
- Loss of data integrity.
- Loss of availability of data.

These listed circumstances mostly signify the areas in which the organization should focus on reducing the risk that is the chance of incurring loss or damage to data within a database. In some conditions, these areas are directly related such that an activity that leads to a loss in one area may also lead to a loss in another since all of the data within an organization is interconnected.

What is a Threat?

Any situation or event, whether intentionally or incidentally, can cause damage which can reflect an adverse effect on the database structure and consequently the organization. A threat may occur by a situation or event involving a person, or the action or situations that is probably to bring harm to an organization and its database.

The degree that an organization undergoes as a result of a threat's following which depends upon some aspects, such as the existence of countermeasures and contingency plans. Let us take an example where you have a hardware failure occurs corrupting secondary storage; all processing activity must cease until the problem is resolved.

Types Of Security and Integrity Threats : A database security program should include the regular review of permissions granted to individually owned accounts and accounts used by automated processes. The accounts used by automated processes should have appropriate controls around password storage such as sufficient encryption and access Controls to reduce the risk of compromise.

In addition to features required in the DBMS for Security and integrity, Some extra features have to be supported by the operating System.

In today’s internet environment, data may be accessed by eavesdroppers, wiretappers, and other illegal users. To prevent this type of threat, data transmitted over public communication channels should be in a encrypted form.

In conjunction with a Sound database Security program, an appropriate disaster recovery program should exist to ensure that service is not interrupted during a security incident or any other incident that results in an outage of the primary database environment. An example is that of replication for the primary databases to sites located in different geographical regions. Security and integrity threats can be classified into following two categories:

1. Accidental Security and integrity threats
2. Intentional or Malicious Security and integrity threats

Accidental Security and integrity Threats

- **Unauthorized Access**: A user can get access to a portion of the database not normally accessible to that user due to a system error or an error on the part of another user.
- **Failures**: Failures of various types during normal operation, for example, transaction processing or storage media loss. Proper recovery procedures are normally used to recover from failures occurring during transaction processing.
- **Concurrent usage anomalies** : Proper Synchronization mechanisms are used to avoid data inconsistencies due to concurrent usage.
- **System error** : A dial-in user may be assigned the identity of another dial-in user who was disconnected accidentally or who hung up without going through a log-off procedure.
- **Improper authorization**: The authorizer can accidentally give improper authorization to a user, which could lead to a database security and/or integrity violations.
- **Hardware failures**: For example, memory protection hardware that fails could lead to software errors and culminate in a database security and/or integrity violation.

Malicious or intentional Security and integrity Threats

- A Computer System operator or System programmer can intentionally bypass the normal security and integrity mechanisms, alter or destroy the data in the database, or make unauthorized copies of sensitive data.
- An authorized user can get access to a secure terminal or the password of an authorized user and Compromise the database. Such users could also destroy the database files.
- Authorized users could pass on sensitive information under influence or for personal gain.
- System and application programmers could bypass normal security in their programs by directly accessing database files and making changes and Copies for illegal use.
- An unauthorized person could get access to the Computer System, physically or by using a communication channel and compromise the database.

Access controls authenticate and authorize individuals to access the information they are allowed to see and use.

What is access control?

Access control is a method of guaranteeing that users are who they say they are and that they have the appropriate access to company data.

Authentication is a technique used to verify that someone is who they claim to be. Authentication isn't sufficient by itself to protect data, Crowley notes. What's needed is an additional layer, authorization, which determines whether a user should be allowed to access the data or make the transaction they're attempting.

4 Types of access control

Organizations must determine the appropriate **access control model** to adopt based on the type and sensitivity of data they're processing, says Wagner. Older access models include discretionary access control (DAC) and mandatory access control (MAC), role based access control (RBAC) is the most common model today, and the most recent model is known as attribute based access control (ABAC).

Discretionary access control (DAC)

With DAC models, the data owner decides on access. DAC is a means of assigning access rights based on rules that users specify.

Mandatory access control (MAC)

MAC was developed using a nondiscretionary model, in which people are granted access based on an information clearance. MAC is a policy in which access rights are assigned based on regulations from a central authority.

Role Based Access Control (RBAC)

RBAC grants access based on a user's role and implements key security principles, such as "least privilege" and "separation of privilege." Thus, someone attempting to access information can only access data that's deemed necessary for their role.

Attribute Based Access Control (ABAC)

In ABAC, each resource and user are assigned a series of attributes, Wagner explains. "In this dynamic method, a comparative assessment of the user's attributes, including time of day, position and location, are used to make a decision on access to a resource."

It's imperative for organizations to decide which model is most appropriate for them based on data sensitivity and operational requirements for data access. In particular, organizations that process personally identifiable information (PII) or other sensitive information types, including Health Insurance Portability and Accountability Act (HIPAA) or Controlled Unclassified Information (CUI) data, must make access control a core capability in their security architecture, Wagner advises.

Access control solutions

A number of technologies can support the various access control models. In some cases, multiple technologies may need to work in concert to achieve the desired level of access control, Wagner says.

"The reality of data spread across cloud service providers and SaaS applications and connected to the traditional network perimeter dictate the need to orchestrate a secure solution," he notes. "There are multiple vendors providing privilege access and identity management solutions that can be integrated into a traditional Active Directory construct from Microsoft. Multifactor authentication can be a component to further enhance security."

Privileges

Privileges: Privileges defines the access rights provided to a user on a database object. There are two types of privileges.

1) System privileges - This allows the user to CREATE, ALTER, or DROP database objects.
2) Object privileges - This allows the user to EXECUTE, SELECT, INSERT, UPDATE, or DELETE data from database objects to which the privileges apply.

Few CREATE system privileges are listed below:

System Privileges	Description
CREATE object	allows users to create the specified object in their own schema.
CREATE ANY object	allows users to create the specified object in any schema.

The above rules also apply for ALTER and DROP system privileges.

Few of the object privileges are listed below:

Object Privileges	Description
INSERT	allows users to insert rows into a table.
SELECT	allows users to select data from a database object.
UPDATE	allows user to update data in a table.
EXECUTE	allows user to execute a stored procedure or a function.

Roles: Roles are a collection of privileges or access rights. When there are many users in a database it becomes difficult to grant or revoke privileges to users. Therefore, if you define roles, you can grant or revoke privileges to users, thereby automatically granting or revoking privileges. You can either create Roles or use the system roles pre-defined by oracle.

Some of the privileges granted to the system roles are as given below:

System Role	Privileges Granted to the Role
CONNECT	CREATE TABLE, CREATE VIEW, CREATE SYNONYM, CREATE SEQUENCE, CREATE SESSION etc.
RESOURCE	CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER etc. The primary usage of the RESOURCE role is to restrict access to database objects.
DBA	ALL SYSTEM PRIVILEGES

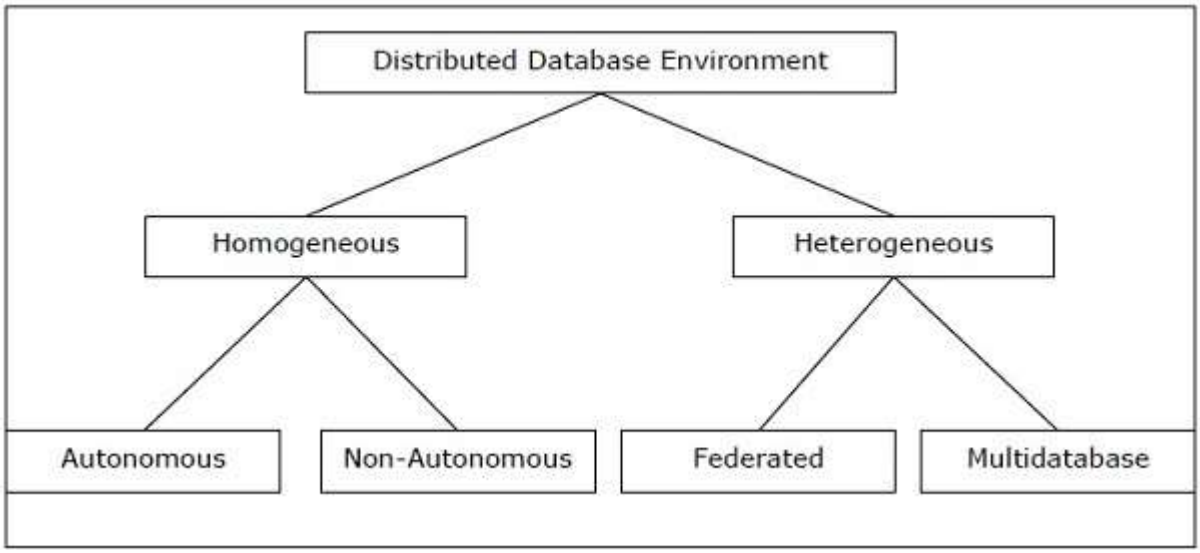
DISTRIBUTED DATABASES

A **distributed database** is a collection of multiple interconnected databases, which are spread physically across various locations that communicate via a computer network.

Distributed databases can be classified into homogeneous and heterogeneous databases having further divisions. The next section of this chapter discusses the distributed architectures namely client – server, peer – to – peer and multi – DBMS. Finally, the different design alternatives like replication and fragmentation are introduced.

Types of Distributed Databases

Distributed databases can be broadly classified into homogeneous and heterogeneous distributed database environments, each with further sub-divisions, as shown in the following illustration.



Homogeneous Distributed Databases

In a homogeneous distributed database, all the sites use identical DBMS and operating systems. Its properties are −

- The sites use very similar software.
- The sites use identical DBMS or DBMS from the same vendor.
- Each site is aware of all other sites and cooperates with other sites to process user requests.
- The database is accessed through a single interface as if it is a single database.

Types of Homogeneous Distributed Database

There are two types of homogeneous distributed database −

- **Autonomous** − Each database is independent that functions on its own. They are integrated by a controlling application and use message passing to share data updates.
- **Non-autonomous** − Data is distributed across the homogeneous nodes and a central or master DBMS co-ordinates data updates across the sites.

Heterogeneous Distributed Databases

In a heterogeneous distributed database, different sites have different operating systems, DBMS products and data models. Its properties are −

- Different sites use dissimilar schemas and software.
- The system may be composed of a variety of DBMSs like relational, network, hierarchical or object oriented.
- Query processing is complex due to dissimilar schemas.

- Transaction processing is complex due to dissimilar software.
- A site may not be aware of other sites and so there is limited co-operation in processing user requests.

Types of Heterogeneous Distributed Databases

- **Federated** − The heterogeneous database systems are independent in nature and integrated together so that they function as a single database system.
- **Un-federated** − The database systems employ a central coordinating module through which the databases are accessed.

Distributed DBMS Architectures

DDBMS architectures are generally developed depending on three parameters −

- **Distribution** − It states the physical distribution of data across the different sites.
- **Autonomy** − It indicates the distribution of control of the database system and the degree to which each constituent DBMS can operate independently.
- **Heterogeneity** − It refers to the uniformity or dissimilarity of the data models, system components and databases.

Architectural Models

Some of the common architectural models are −

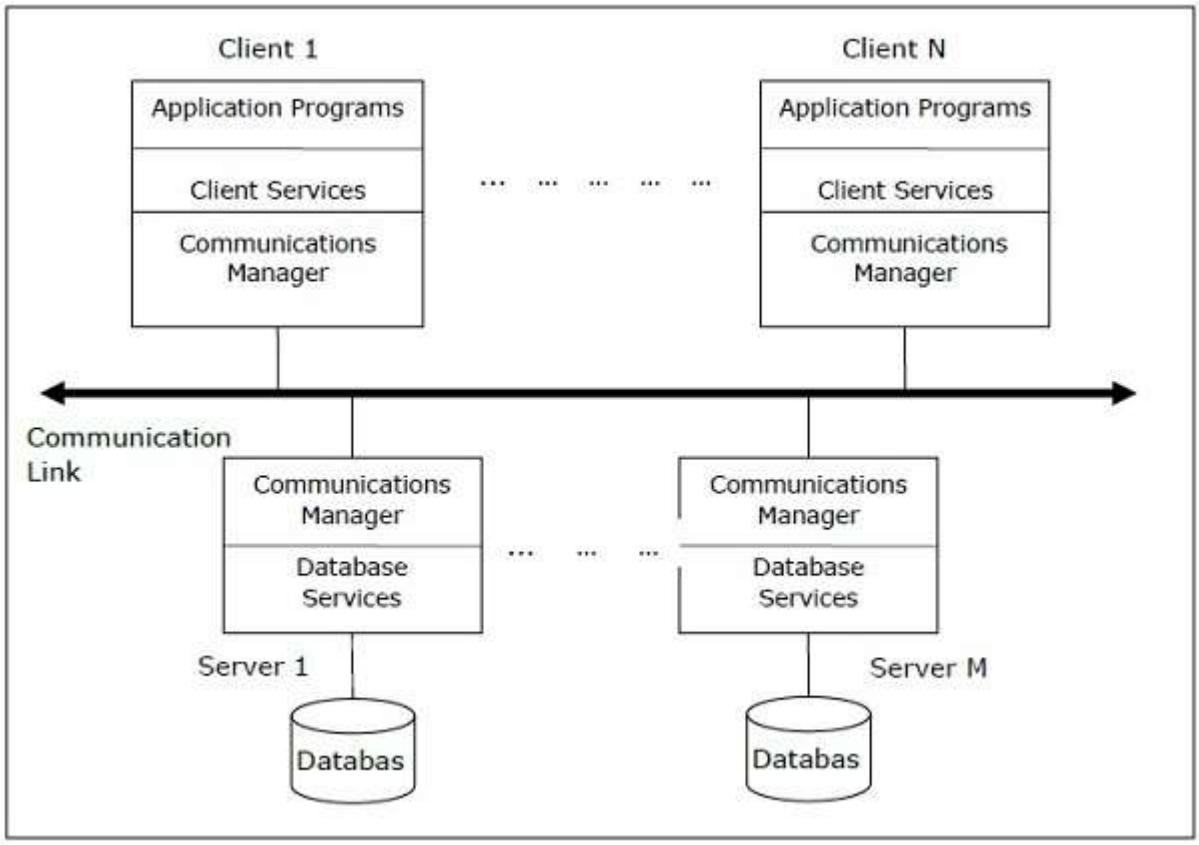
- Client - Server Architecture for DDBMS
- Peer - to - Peer Architecture for DDBMS
- Multi - DBMS Architecture

Client - Server Architecture for DDBMS

This is a two-level architecture where the functionality is divided into servers and clients. The server functions primarily encompass data management, query processing, optimization and transaction management. Client functions include mainly user interface. However, they have some functions like consistency checking and transaction management.

The two different client - server architecture are −

- Single Server Multiple Client
- Multiple Server Multiple Client (shown in the following diagram)

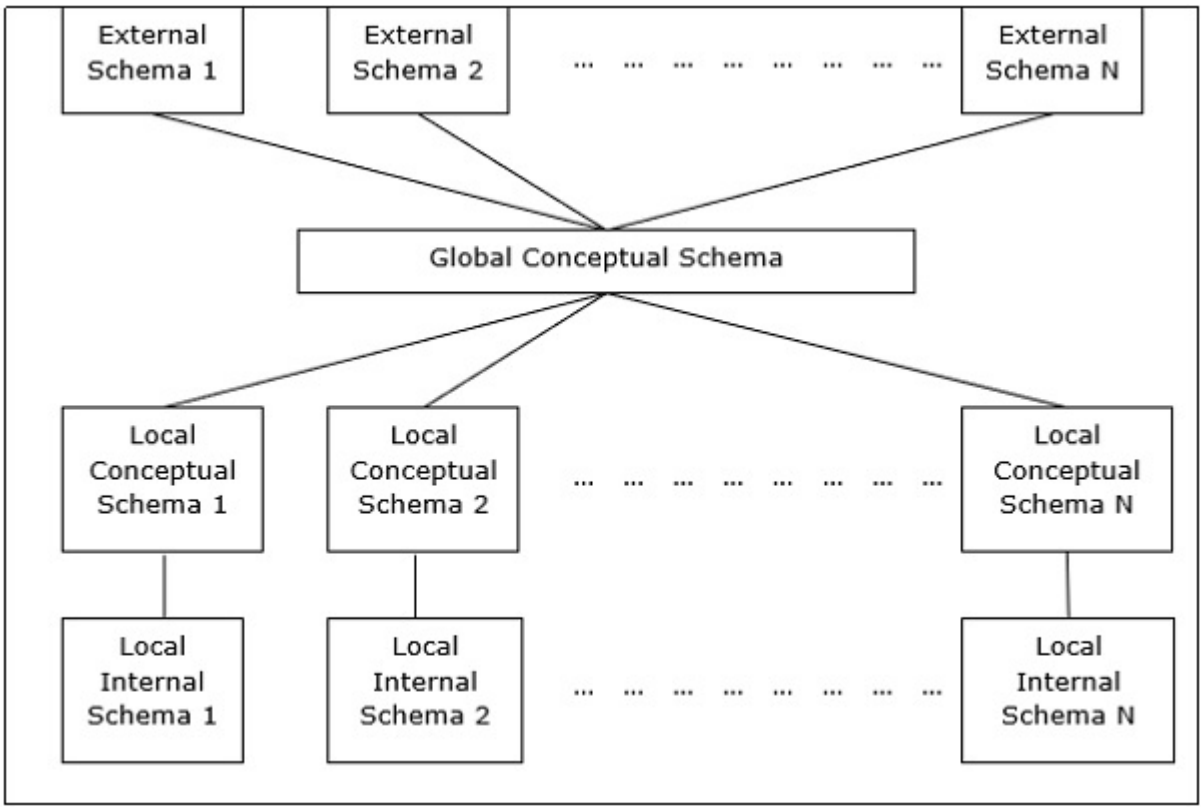


Peer- to-Peer Architecture for DDBMS

In these systems, each peer acts both as a client and a server for imparting database services. The peers share their resource with other peers and co-ordinate their activities.

This architecture generally has four levels of schemas −

- **Global Conceptual Schema** − Depicts the global logical view of data.
- **Local Conceptual Schema** − Depicts logical data organization at each site.
- **Local Internal Schema** − Depicts physical data organization at each site.
- **External Schema** − Depicts user view of data.



TRANSACTIOM PROCESSING

A transaction can be defined as a group of tasks. A single task is the minimum processing unit which cannot be divided further.

Let's take an example of a simple transaction. Suppose a bank employee transfers Rs 500 from A's account to B's account. This very simple and small transaction involves several low-level tasks.

A's Account

Open_Account(A)
Old_Balance = A.balance
New_Balance = Old_Balance - 500
A.balance = New_Balance
Close_Account(A)

B's Account

Open_Account(B)
Old_Balance = B.balance
New_Balance = Old_Balance + 500
B.balance = New_Balance
Close_Account(B)

ACID Properties

A transaction is a very small unit of a program and it may contain several lowlevel tasks. A transaction in a database system must maintain **A**tomicity, **C**onsistency, **I**solation, and **D**urability − commonly known as ACID properties − in order to ensure accuracy, completeness, and data integrity.

- **Atomicity** − This property states that a transaction must be treated as an atomic unit, that is, either all of its operations are executed or none. There must be no state in a database where a transaction is

left partially completed. States should be defined either before the execution of the transaction or after the execution/abortion/failure of the transaction.

- **Consistency** − The database must remain in a consistent state after any transaction. No transaction should have any adverse effect on the data residing in the database. If the database was in a consistent state before the execution of a transaction, it must remain consistent after the execution of the transaction as well.
- **Durability** − The database should be durable enough to hold all its latest updates even if the system fails or restarts. If a transaction updates a chunk of data in a database and commits, then the database will hold the modified data. If a transaction commits but the system fails before the data could be written on to the disk, then that data will be updated once the system springs back into action.
- **Isolation** − In a database system where more than one transaction are being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system. No transaction will affect the existence of any other transaction.

Serializability

When multiple transactions are being executed by the operating system in a multiprogramming environment, there are possibilities that instructions of one transactions are interleaved with some other transaction.

- **Schedule** − A chronological execution sequence of a transaction is called a schedule. A schedule can have many transactions in it, each comprising of a number of instructions/tasks.
- **Serial Schedule** − It is a schedule in which transactions are aligned in such a way that one transaction is executed first. When the first transaction completes its cycle, then the next transaction is executed. Transactions are ordered one after the other. This type of schedule is called a serial schedule, as transactions are executed in a serial manner.

In a multi-transaction environment, serial schedules are considered as a benchmark. The execution sequence of an instruction in a transaction cannot be changed, but two transactions can have their instructions executed in a random fashion. This execution does no harm if two transactions are mutually independent and working on different segments of data; but in case these two transactions are working on the same data, then the results may vary. This ever-varying result may bring the database to an inconsistent state.

To resolve this problem, we allow parallel execution of a transaction schedule, if its transactions are either serializable or have some equivalence relation among them.

Equivalence Schedules

An equivalence schedule can be of the following types −

Result Equivalence

If two schedules produce the same result after execution, they are said to be result equivalent. They may yield the same result for some value and different results for another set of values. That's why this equivalence is not generally considered significant.

View Equivalence

Two schedules would be view equivalence if the transactions in both the schedules perform similar actions in a similar manner.

For example −

- If T reads the initial data in S1, then it also reads the initial data in S2.
- If T reads the value written by J in S1, then it also reads the value written by J in S2.
- If T performs the final write on the data value in S1, then it also performs the final write on the data value in S2.

Conflict Equivalence

Two schedules would be conflicting if they have the following properties −

- Both belong to separate transactions.
- Both accesses the same data item.
- At least one of them is "write" operation.

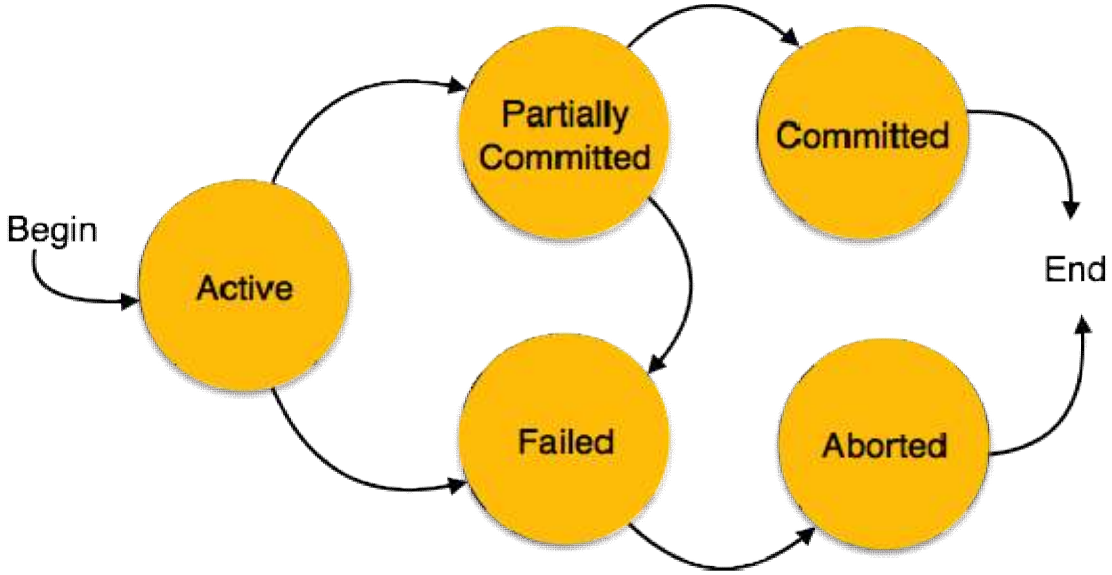
Two schedules having multiple transactions with conflicting operations are said to be conflict equivalent if and only if −

- Both the schedules contain the same set of Transactions.
- The order of conflicting pairs of operation is maintained in both the schedules.

Note − View equivalent schedules are view serializable and conflict equivalent schedules are conflict serializable. All conflict serializable schedules are view serializable too.

States of Transactions

A transaction in a database can be in one of the following states −



- **Active** − In this state, the transaction is being executed. This is the initial state of every transaction.
- **Partially Committed** − When a transaction executes its final operation, it is said to be in a partially committed state.
- **Failed** − A transaction is said to be in a failed state if any of the checks made by the database recovery system fails. A failed transaction can no longer proceed further.
- **Aborted** − If any of the checks fails and the transaction has reached a failed state, then the recovery manager rolls back all its write operations on the database to bring the database back to its original state where it was prior to the execution of the transaction. Transactions in this state are called aborted. The database recovery module can select one of the two operations after a transaction aborts −
    - Re-start the transaction
    - Kill the transaction
- **Committed** − If a transaction executes all its operations successfully, it is said to be committed. All its effects are now permanently established on the database system.

Characteristics of Spatial Database

A spatial database system has the following characteristics

1. It is a database system
2. It offers spatial data types (SDTs) in its data model and query language.
3. It supports spatial data types in its implementation, providing at least spatial indexing and efficient algorithms for spatial join.

Example

A road map is a visualization of geographic information. A road map is a 2-dimensional object which contains points, lines, and polygons that can represent cities, roads, and political boundaries such as states or provinces.

In general, spatial data can be of two types:

1. Vector data: This data is represented as discrete points, lines and polygons
2. Rastor data: This data is represented as a matrix of square cells.

A **parallel database** system seeks to improve performance through parallelization of various operations, such as loading data, building indexes and evaluating queries.[1] Although data may be stored in a distributed fashion, the distribution is governed solely by performance considerations. Parallel databases improve processing and input/output speeds by using multiple CPUs and disks in parallel. Centralized and client–server database systems are not powerful enough to handle such applications. In parallel processing, many operations are performed simultaneously, as opposed to serial processing, in which the computational steps are performed sequentially. Parallel databases can be roughly divided into two groups, the first group of architecture is the multiprocessor architecture, the alternatives of which are the following:

Shared memory architecture

Where multiple processors share the main memory (RAM) space but each processor has its own disk (HDD). If many processes run simultaneously, the speed is reduced, the same as a computer when many parallel tasks run and the computer slows down.

Shared disk architecture

Where each node has its own main memory, but all nodes share mass storage, usually a storage area network. In practice, each node usually also has multiple processors.

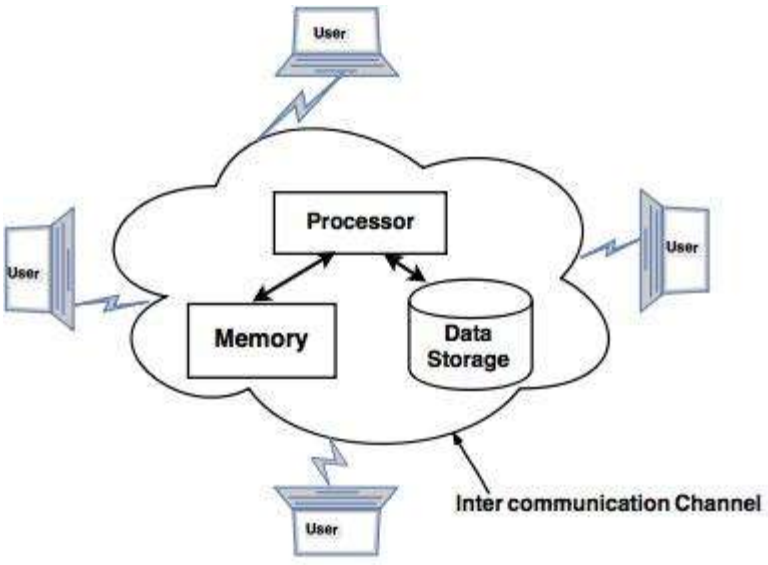
Shared nothing architecture

Where each node has its own mass storage as well as main memory.

The other architecture group is called hybrid architecture, which includes:

- Non-Uniform Memory Architecture (NUMA), which involves the non-uniform memory access.
- Cluster (shared nothing + shared disk: SAN/NAS), which is formed by a group of connected computers.

in this switches or hubs are used to connect different computers its most cheapest way and simplest way only simple topologies are used to connect different computers . much smarter if switches are implemented.



Parallel database system

Types of parallelism

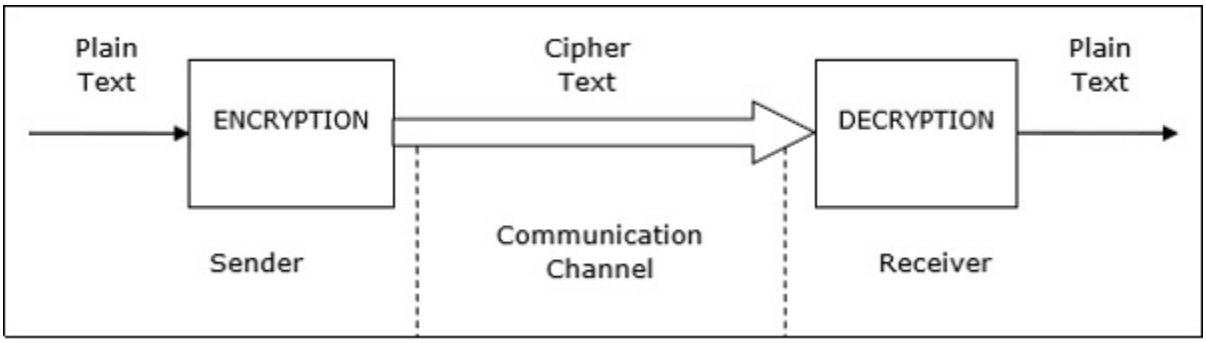
- **Interquery parallelism** : It is possible to process a number of transactions in parallel with each other. Improves Throughput.
- **Independent parallelism** – Execution of each operation individually in different processors only if they can be executed independent of each other. For example, if we need to join four tables, then two can be joined at one processor and the other two can be joined at another processor. Final join can be done later.
- **Pipe-lined parallelism** – Execution of different operations in pipe-lined fashion. For example, if we need to join three tables, one processor may join two tables and send the result set records as and when they are produced to the other processor. In the other processor the third table can be joined with the incoming records and the final result can be produced.
- **Intraoperation parallelism** – Execution of single complex or large operations in parallel in multiple processors. For example, ORDER BY clause of a query that tries to execute on millions of records can be parallelized on multiple processors.

What is Cryptography?

Cryptography is the science of encoding information before sending via unreliable communication paths so that only an authorized receiver can decode and use it.

The coded message is called **cipher text** and the original message is called **plain text**. The process of converting plain text to cipher text by the sender is called encoding or **encryption**. The process of converting cipher text to plain text by the receiver is called decoding or **decryption**.

The entire procedure of communicating using cryptography can be illustrated through the following diagram −



Conventional Encryption Methods

In conventional cryptography, the encryption and decryption is done using the same secret key. Here, the sender encrypts the message with an encryption algorithm using a copy of the secret key. The encrypted message is then send over public communication channels. On receiving the encrypted message, the receiver decrypts it with a corresponding decryption algorithm using the same secret key.

Security in conventional cryptography depends on two factors −

- A sound algorithm which is known to all.
- A randomly generated, preferably long secret key known only by the sender and the receiver.

The most famous conventional cryptography algorithm is **Data Encryption Standard** or **DES**.

The advantage of this method is its easy applicability. However, the greatest problem of conventional cryptography is sharing the secret key between the communicating parties. The ways to send the key are cumbersome and highly susceptible to eavesdropping.

Public Key Cryptography

In contrast to conventional cryptography, public key cryptography uses two different keys, referred to as public key and the private key. Each user generates the pair of public key and private key. The user then puts the public key in an accessible place. When a sender wants to sends a message, he encrypts it using the public key of the receiver. On receiving the encrypted message, the receiver decrypts it using his private key. Since the private key is not known to anyone but the receiver, no other person who receives the message can decrypt it.

The most popular public key cryptography algorithms are **RSA** algorithm and **Diffie– Hellman** algorithm. This method is very secure to send private messages. However, the problem is, it involves a lot of computations and so proves to be inefficient for long messages.

The solution is to use a combination of conventional and public key cryptography. The secret key is encrypted using public key cryptography before sharing between the communicating parties. Then, the message is send using conventional cryptography with the aid of the shared secret key.Digital Signatures

A Digital Signature (DS) is an authentication technique based on public key cryptography used in e-commerce applications. It associates a unique mark to an individual within the body of his message. This helps others to authenticate valid senders of messages.

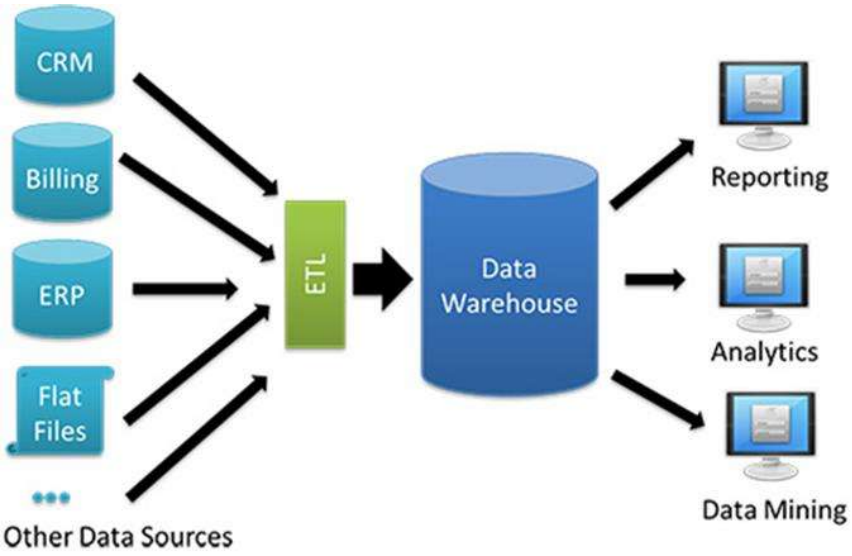
Typically, a user's digital signature varies from message to message in order to provide security against counterfeiting. The method is as follows −

- The sender takes a message, calculates the message digest of the message and signs it digest with a private key.
- The sender then appends the signed digest along with the plaintext message.
- The message is sent over communication channel.
- The receiver removes the appended signed digest and verifies the digest using the corresponding public key.
- The receiver then takes the plaintext message and runs it through the same message digest algorithm.
- If the results of step 4 and step 5 match, then the receiver knows that the message has integrity and authentic.

Data Warehousing

Data warehousing is a collection of tools and techniques using which more knowledge can be driven out from a large amount of data. This helps with the decision-making process and improving information resources.

Data warehouse is basically a database of unique data structures that allows relatively quick and easy performance of complex queries over a large amount of data. It is created from multiple heterogeneous sources.



Characteristics of Data Warehousing

1. *Integrated*
2. *Time variant*
3. *Non-volatile*

The purpose of Data warehouse is to support the decision making process. It makes information easily accessible as we can generate reports from the data warehouse. It usually contains historical data derived from transactional data but can also include data from other sources. Data warehouse is always kept separated from transactional data.

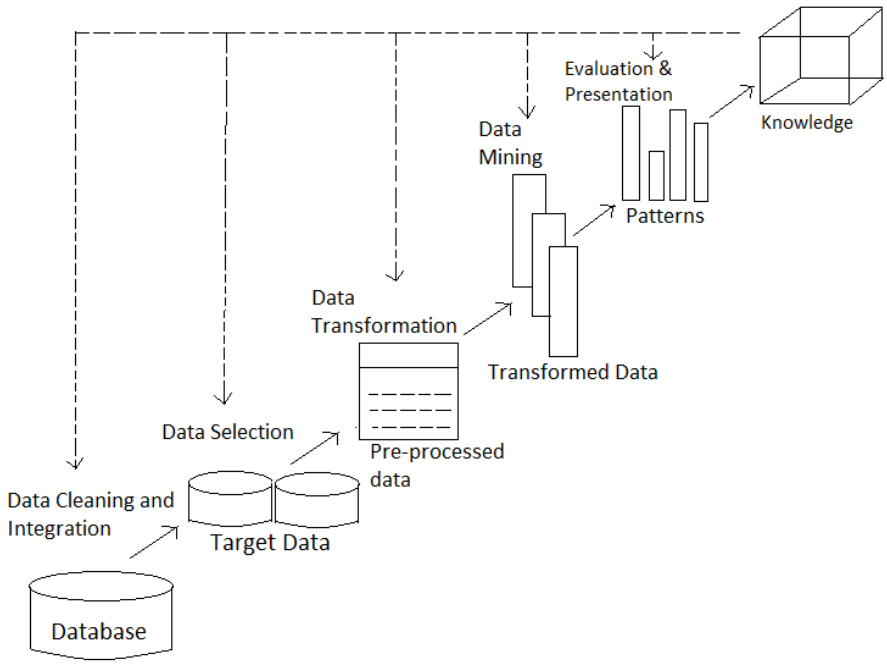
We have multiple data sources on which we apply ETL processes in which we Extract data from data source, then transform it according to some rules and then load the data into the desired destination, thus creating a data warehouse.

Data Mining

Data mining refers to extracting knowledge from large amounts of data. The data sources can include databases, data warehouse, web etc.

Knowledge discovery is an iterative sequence:

1. Data cleaning – Remove inconsistent data.
2. Data integration – Combining multiple data sources into one.
3. Data selection – Select only relevant data to be analysed.
4. Data transformation – Data is transformed into appropriate form for mining.
5. Data mining – methods to extract data patterns.
6. Pattern evaluation – identify interesting patterns in the data.
7. Knowledge representation- visualization and knowledge representation techniques are used.]



What kind of data that can be mined?

1. ***Database Data***
2. ***Data Warehouse***
3. ***Transactional Data***

Scope of Data mining

1. Automated Prediction of trends and behaviours: Data mining automates the process of finding the predictive information in large databases. For example : Consider a marketing company. In this company, data mining uses the past promotional mailing to identify the targets to maximize the return.

2. Automated discovery of previously unknown patterns: Data mining sweeps through the database and identifies previously hidden patterns. For example: In a retail store data mining will go through the entire database and find the pattern for the items which are usually brought together

CLASSIFICATION :

There are two forms of data analysis that can be used for extracting models describing important classes or to predict future data trends. These two forms are as follows −

- Classification
- Prediction

Classification models predict categorical class labels; and prediction models predict continuous valued functions. For example, we can build a classification model to categorize bank loan applications as either safe or risky, or a prediction model to predict the expenditures in dollars of potential customers on computer equipment given their income and occupation.

What is classification?

Following are the examples of cases where the data analysis task is Classification −

- A bank loan officer wants to analyze the data in order to know which customer (loan applicant) are risky or which are safe.
- A marketing manager at a company needs to analyze a customer with a given profile, who will buy a new computer.

In both of the above examples, a model or classifier is constructed to predict the categorical labels. These labels are risky or safe for loan application data and yes or no for marketing data.

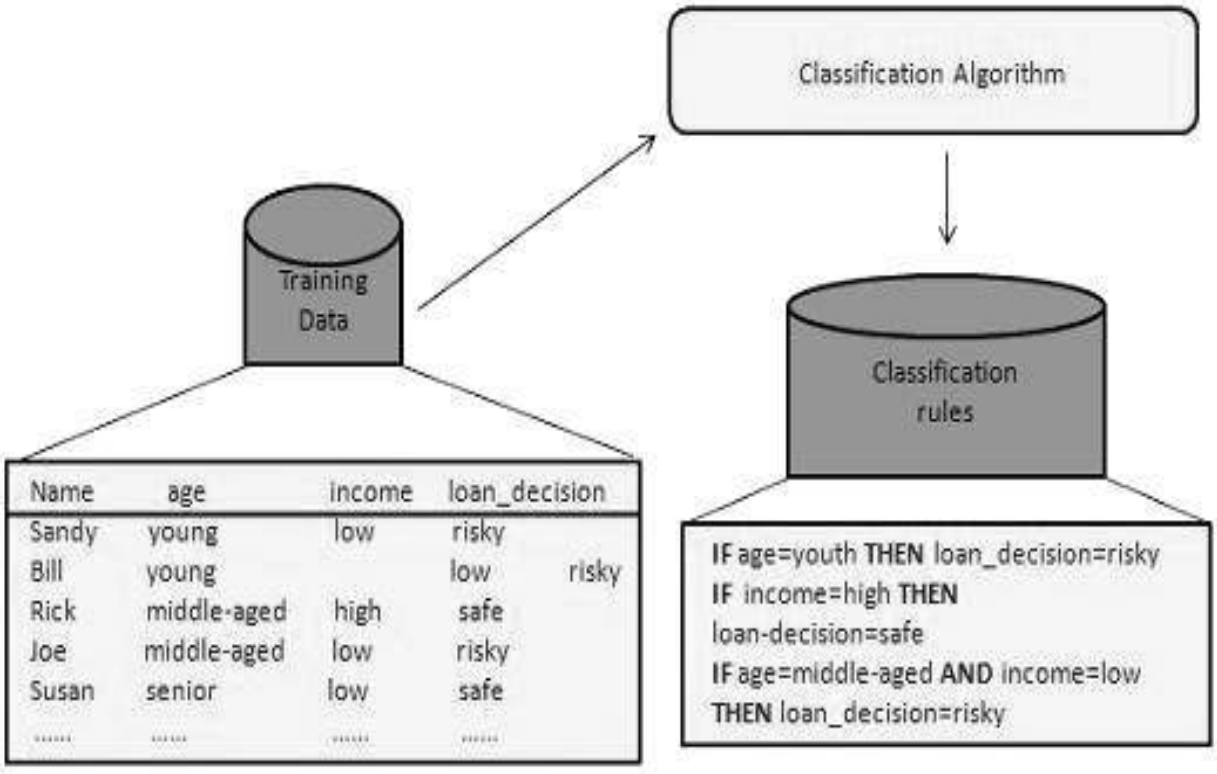
How Does Classification Works?

With the help of the bank loan application that we have discussed above, let us understand the working of classification. The Data Classification process includes two steps −

- Building the Classifier or Model
- Using Classifier for Classification

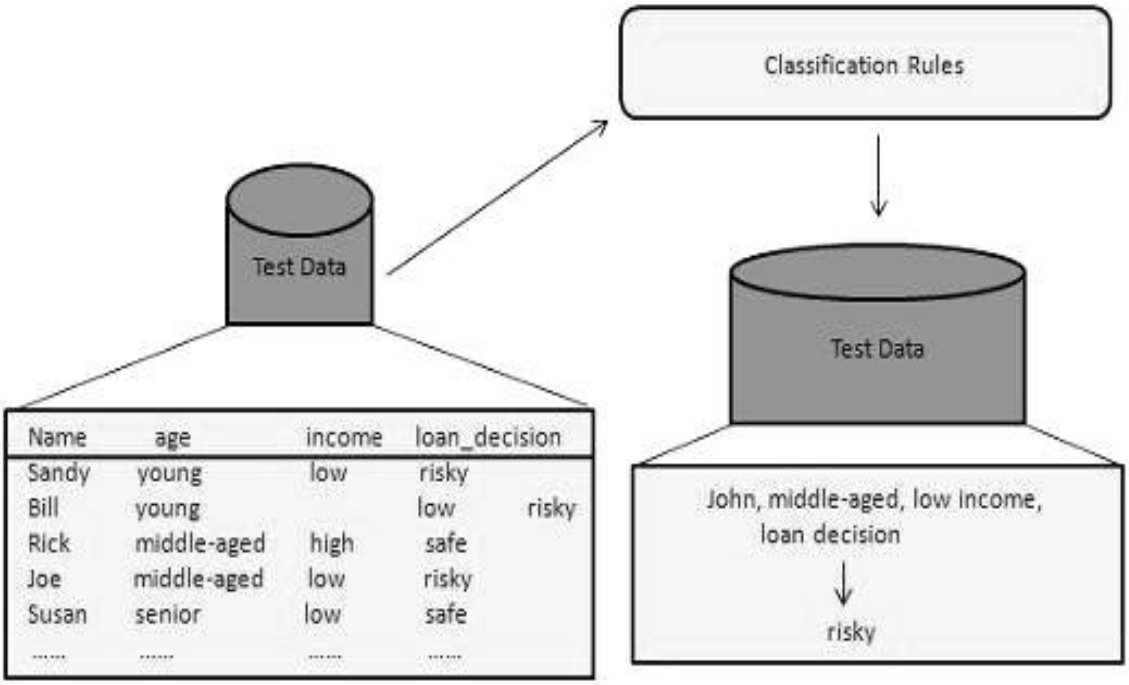
Building the Classifier or Model

- This step is the learning step or the learning phase.
- In this step the classification algorithms build the classifier.
- The classifier is built from the training set made up of database tuples and their associated class labels.
- Each tuple that constitutes the training set is referred to as a category or class. These tuples can also be referred to as sample, object or data points.



. Using Classifier for Classification

In this step, the classifier is used for classification. Here the test data is used to estimate the accuracy of classification rules. The classification rules can be applied to the new data tuples if the accuracy is considered acceptable.



- Association rules search for interesting relationships among the items in a given data set by examining transactions.
  For example - Frequently purchased item- sets can be easily found out by examining the shop cart. This helps in advertising or for placement of goods in the store.
- Association rules have general form:
  $I_1 \rightarrow$ (where $I_1 \cap I_2 = 0$)
- The rule can be read as, Given that someone has purchased the items from the set **I_1, then** they are likely to also buy the items in the set **I_2.**

Large Item-sets

- It is a set of single items, from transactions.
- If some items occur together, then they can form an association rule.

Support:

- It is the percentage of the transactions in which the items appear.
- If **A => B**
- **Support (A → B) = # _ tuples containing both A and B / total #_tuples**
- The support for an association X => Y is the percentage of transactions in the database that contains **X U Y.**

Confidence:

- The confidence or strength for an association rule **A => B** is the ratio of the number of transactions that contain **A.**
- Consider a rule **A => B, which** is a measure of ratio of the number of tuples containing both A and B to the number of tuples containing **'A'.**

Confidence (A → B) = #_tuples containing both A and B / #_tuples_containing A

CLUSTERING

- It is a data mining technique used to place the data elements into their related groups.
- Clustering is the process of partitioning the data (or objects) into the same class, The data in one class is more similar to each other than to those in other cluster.
- The process of partitioning data objects into subclasses is called as cluster.
- A cluster consists of data object with high inter similarity and low intra similarity.
- The quality of cluster depends on the method used.
- Clustering is also called as data segmentation, because it partitions large data sets into groups according to their similarity

Clustering can be helpful in many fields, such as:

1. Marketing:

Clustering helps to find group of customers with similar behavior from a given data set customer record.

2. Biology:

Classification of plants and animal according to their features.

3. Library:

Clustering is very useful in book ordering.

Types of clustering

Clustering methods can be classified into the following categories:

1. Partitioning

In this approach, several partitions are created and then evaluated based on given criteria.

2. Hierarchical method

In this method, the set of data objects are decomposed (multilevel) hierarchically by using certain criteria.

3. Density-based method

This method is based on density (density reachability and density connectivity).

4. Grid-based methods

This approach is based on multi-resolution grid data structure.

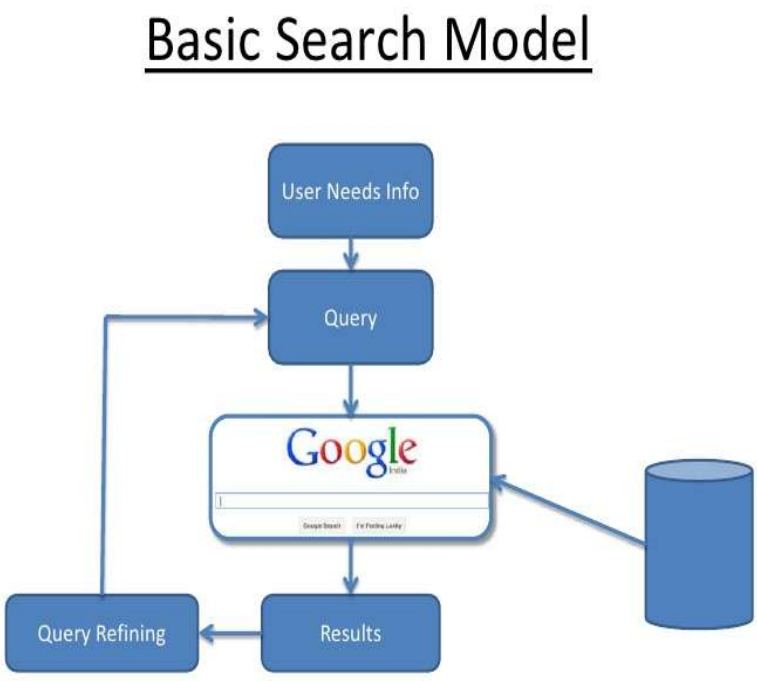
Classification vs Clustering

Classification	Clustering
It is supervised learning.	It is unsupervised learning.
Classification contains previously categorized training set.	In clustering, the characteristics of similarity of data is not known.
Decision tree is used to partition and segment record.	There are a variety of algorithms for clustering, which generally share the same property of interactively assigning records to a cluster.

INFORMATION RETRIEVAL

Information retrieval is the activity of obtaining information resources relevant to an information need from a collection of information resources. An information retrieval process begins when a user enters a query into the system. Queries are formal statements of information needs. User queries are matched against the database information. Depending on the application the data objects may be, for example, text documents, images, audio, mind maps or videos. Most IR systems compute a numeric score on how well each object in the database matches the query, and rank the objects according to this value. The top ranking objects are then shown to the user. The process may then be iterated if the user wishes to refine the query.

Every online database, every search engine, everything that is searched online is based in some way or another on principles developed in IR ◦ IR is at the heart of searching used in systems

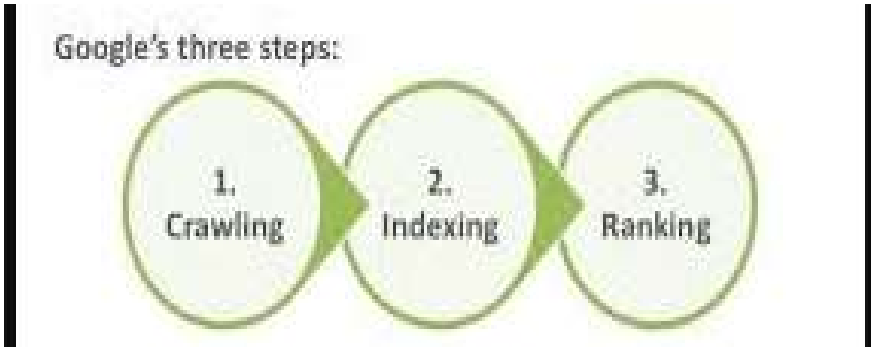


Automated information retrieval systems are used to reduce what has been called information overload. An IR system is a software system that provides access to books, journals and other documents; stores and manages those documents. Web search engines are the most visible IR applications.

Crawling and Indexing

1. CRAWLING AND INDEXING

2. In the simplest terms, you could think of searching the web as looking in a very large book with an impressive index telling you exactly where everything is located. When we perform a Google search, Google programs check their index to determine the most relevant search results. There are three process in delivering search results.



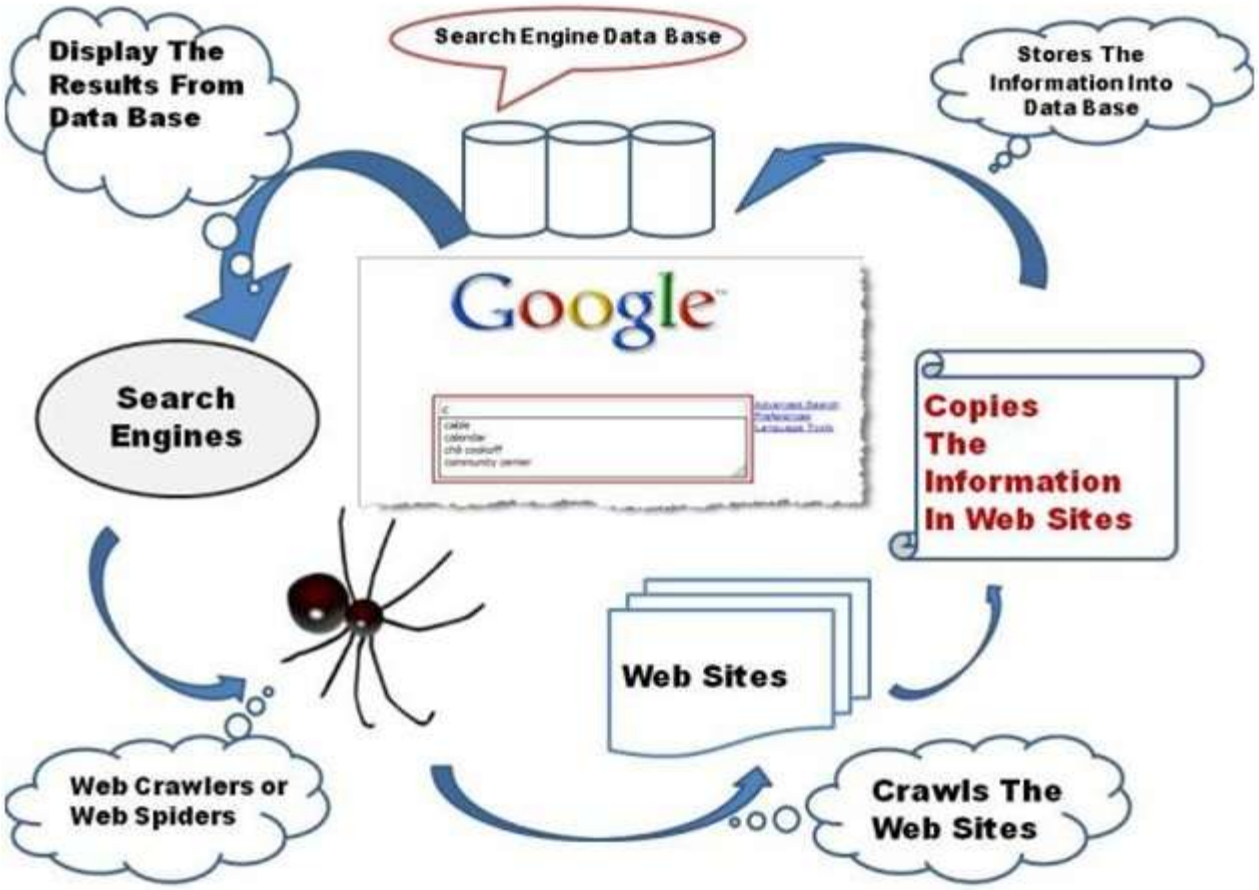
1. Crawling – Does Google know about your site and find it.
2. Indexing – Can Google index your site.
3. Serving – Does the site have good and useful content that is relevant to search users.

CRAWLING Crawling is the process of fetching all the web pages linked with a website. This task is performed by a software called crawler. The crawlers are also known as spiders or bots , they visit website and send information to their respective parent websites.

INDEXING Indexing is the process of creating index for all the fetched web pages and keeping them into a huge database from where it can later be retrieved. An index is another name for the database used by a search engines. It contains information on all the websites the search engine was able to find. If a website is not in a search engine's index, users will not be able to find it using that search engine. Search engines regularly update their indexes.

5. Difference between Indexing and Crawling Crawling is the process through which indexing is done. Google crawlers crawls through the WebPages and index the pages. When search engine crawlers visit any link is called crawling and when crawlers save or index that links in search engine database is called indexing.

6. Crawling means to visit the link by Search engines and indexing means to put the page contents in Database (after analysis) and make them available in search results when a request is made. Crawling means the search engine robot crawl or fetch the web pages while Indexing means search engine robot crawl the web pages, saved the information and it appear in the search engine.


Display The Results From Data Base
Search Engine Data Base
Stores The Information Into Data Base
Google
Search Engines
Copies The Information In Web Sites
Web Sites
Web Crawlers or Web Spiders
Crawls The Web Sites

OBJECT ORIENTED DATABASES

The **ODBMS** which is an abbreviation for **object oriented database management system**, is the data model in which data is stored in form of objects, which are instances of classes. These classes and objects together makes an object oriented data model.

Components of Object Oriented Data Model:

The OODBMS is based on three major components, namely: Object structure, Object classes, and Object identity. These are explained as following below.

1. Object Structure:

The structure of an object refers to the properties that an object is made up of. These properties of an object are referred to as an attribute. Thus, an object is a real world world entity with certain attributes that makes up the object structure. Also an object encapsulates the data code into a single unit which in turn provides data abstraction by hiding the implementation details from the user.

The object structure is further composed of three types of components: Messages, Methods, and Variables. These are explained as following below.

1. **Messages –**
   A message provides an interface or acts as a communication medium between an object and the outside world. A message can be of two types:
   - **Read-only message:** If the invoked method does not change the value of a variable, then the invoking message is said to be a read-only message.
   - **Update message:** If the invoked method changes the value of a variable, then the invoking message is said to be an update message.
2. **Methods –**
   When a message is passed then the body of code that is executed is known as a method. Every time when a method is executed, it returns a value as output. A method can be of two types:
   - **Read-only method:** When the value of a variable is not affected by a method, then it is known as read-only method.
   - **Update-method:** When the value of a variable changes by a method, then it is known as an update method.
3. **Variables –**
   It stores the data of an object. The data stored in the variables makes the object distinguishable from one another.

2. Object Classes:

An object which is a real world entity is an instance of a class. Hence first we need to define a class and then the objects are made which differ in the values they store but share the same class definition. The objects in turn corresponds to various messages and variables stored in it.

XML Database :

XML Database is used to store huge amount of information in the XML format. As the use of XML is increasing in every field, it is required to have a secured place to store the XML documents. The data stored in the database can be queried using **XQuery**, serialized, and exported into a desired format.

XML Database Types

There are two major types of XML databases −

- XML- enabled
- Native XML (NXD)

XML - Enabled Database

XML enabled database is nothing but the extension provided for the conversion of XML document. This is a relational database, where data is stored in tables consisting of rows and columns. The tables contain set of records, which in turn consist of fields.

Native XML Database

Native XML database is based on the container rather than table format. It can store large amount of XML document and data. Native XML database is queried by the **XPath**-expressions.

Native XML database has an advantage over the XML-enabled database. It is highly capable to store, query and maintain the XML document than XML-enabled database.

Example

Following example demonstrates XML database −

```
<?xml version = "1.0"?>
<contact-info>
   <contact1>
      <name>Tanmay Patil</name>
      <company>TutorialsPoint</company>
      <phone>(011) 123-4567</phone>
   </contact1>

   <contact2>
      <name>Manisha Patil</name>
      <company>TutorialsPoint</company>
      <phone>(011) 789-4567</phone>
   </contact2>
</contact-info>
```

Here, a table of contacts is created that holds the records of contacts (contact1 and contact2), which in turn consists of three entities − *name, company* and *phone*.

SUBJECT NAME: DATABASE MANAGEMENT SYSTEM SEMESTER: III					
SUBJECT CODE:18CCU302					
UNIT I					
QUESTION	**OPTION 1**	**OPTION 2**	**OPTION 3**	**OPTION 4**	**ANSWER**
The data stored in the system is partitioned into one or more	Database	Data record	Data field	Stored field	Database
Smallest unit of data representation is	Bit	Field	Record	File	Bit
The database system involves _____ major components	2	4	5	3	4
The data is stored in _____ and _____	Concept & Physical	Shared & Concept	Integrated & Shared	Stored	Integrated & Shared
A(n)_____ data refers to information entering the system from the outside world.	Output	Input	Process	Information	Input
_____ does not come under ACID property.	Atomicity	Compactness	Isolation	Durability	Compactness
DML stands for	Data Manipulation Language	Data Modular Language	Data Multiplication Language	Data Minimal Language	Data Manipulation Language
A(n)_____ data refers to messages and reports.	Output	Input	Process	Both input and output	Output
The_____ consists of secondary storage volumes.	Software	Hardware	Firmware	System	Hardware
Which of the following is not a file organization method?	Sequential Organization	Indexed-sequential organization	Indirect organization	direct Organization	Indirect organization
Views are useful for _____ unwanted information, and for collecting together information from more than one relation into a single view	Hiding	Deleting	Highlighting	All of the above	Hiding
A _____ is a collection of stored operational data.	Field	Database	Memory	Record	Database
The_____ links the basic entities together.	Relationship	Sequence	Selection	Projection	Relationship
The singular form of Data is	Datum	Datus	Dates	datas	Datum

A_____must have the freedom to change the Storage structure.	DBA	DML	DDA	TCL	DBA
A stored_____is the smallest collection of associated stored fields.	Record	Data	File	Field	Record
A stored_____is the smallest named unit of data stored in the database.	Record	Data	File	Field	Field
_____ has the highest level of data abstraction	Physical level	Logical level	view level	data level	view level
With regard to expressive power of the formal relational query languages, which of the following statement is true?	relational algebra is more powerful than relational calculus	relational algebra has same power as relational calculus	relational algebra has same power as safe relational calculus	None of the above	relational algebra has same power as safe relational calculus
A stored _____is the collection of occurrences of one type of stored record.	Record	Data	File	Field	File
Data items are fragmented, replicated and propagated in _____	DBMS	RDBMS	DDBMS	DML	DDBMS
A _____field may be stored in internal arithmetic form or a character string.	Char	Integer	Numeric	Float	Numeric
_____ is an object based data model	Hierarchial	Entity relationship	Relational	Physical	Entity relationship
A _____string field may be stored in any of several character codes.	Char	Integer	Numeric	Float	Char
Highest level of normal form is _____	1NF	5NF	BCNF	4NF	5NF
Relations produced from an E-R model will always be in _____	1NF	3NF	BCNF	4NF	3NF
The _____ in a numeric field may change from inches to centimeters.	Char	Integer	Numeric	Units	Units
The_____ may be desirable to represent data in storage by coded values.	Data coding	Char	Integer	Data byte	Data coding
The_____ constructing an occurrence of the logical field from the corresponding stored field occurrence and presenting it to the application.	Data materialization	Data independence	Data dependence	Record	Data materialization
A _____file may be physically implemented in storage in a wide variety of ways.	Accessed	Related	Stored	Record	Stored
The architecture is divided into _____levels.	4	2	5	3	3
The___level is the one closest to physical storage.	Internal	External	Logical	Conceptual	Internal
The_____ level is the one closest to the user.	Internal	External	Logical	Conceptual	External
The___level is a level of indirection.	Internal	External	Logical	Conceptual	Conceptual
The_____is a subset of the total language that is concerned with database objects and	DSL	DML	DDL	DCL	DSL

operations.					
The data sublanguage as being embedded in a _____language.	Source	Host	Designation	Destination	Host
The_____provides for the definition or description of database objects	DSL	DML	DDL	TCL	DDL
A collection of rows and columns in DBMS is known as _____	Tuple	Record	Table	Attribute	Table
All the keys in DBMS is collectively called as _____	Primary Key	Candidate Key	Foreign Key	Super key	Candidate Key
If R(ABCDE) and {ABàC, CàD} then the Candidate Key is _____	(ABC)	(BCE)	(ABE)	(ABCDE)	(ABE)
The overall logical design of the database is _____	Instance	Schema	Both a and b	None of the above	Schema
A_____supports the manipulation or processing of such objects	DSL	DML	DDL	TCL	DML
A_____record is not necessarily the same as a stored record.	Internal	Extern al	Logical	Conceptual	External
A _____record is not necessarily the same as either an external record, on the one hand, or a stored record on the other.	Internal	Extern al	Logical	Conceptual	Conceptual
The_____database that is not stored in it's entirely at a single physical location.	Shared	Source	Distributed	Designation	Distributed
The component responsible for this internal /physical conversion is called a	Access method	Record	Fiëld method	Interface	Access method
A_____key values uniquely identify the records in the file.	Foreign	Candidate	Primary	Superkey	Primary
The_____is not stored in its entity at a single location but rather is spread across a network of computers.	Network Database	Distributed Database	Database	Central database	Distributed Database
The physical record is the interface between the_____method and the_____database.	Access & Internal	Access & Physical	Access & External	Access & Physical	
The indexed file can be accessed through direct and _____access	Indirect	Sequential access	Direct & Indirect	Direct	Indirect
The _____set provides fast direct access to the data	Sequence	Group	Index	Record	Index
The conceptual schema can sometimes be called as	Logical schema	Physical schema	External schema	Internal schema	Logical schema
An each stored file is sequenced by the access method on its_____	Primary Key	Foreign Key	Candidate Key	Super key	Primary Key
The purpose of index is to provide an _____path to the file.	Access	Root	Unaccessed	Direct	Access
	Relation	Tuple	Attribute	record	Relation

Tables are connected in RDBMS only by means of _____					

	UNIT II					
S.NO	**QUESTION**	**OPTION 1**	**OPTION 2**	**OPTION 3**	**OPTION 4**	**ANSWER**
1	SQL stands for _____.	Structured Query Language	Simple Query Language	Standard Query Language	Statement Query Language	Structured Query Language
2	SQL is a _____ language.	Low Level	High Level	Medium Level	Object Oriented	High Level
3	The research paper which laid down abstract principles of database management was published by _____	E.F Codd	D.Chamberlain	Chris Date	Steve	E.F Codd
4	Who developed Structured English Query language?	E.F Codd	D.Chamberlain	Chris Date	Steve	D.Chamberlain
5	Whose research paper triggered to develop relational languages ?	E.F Codd	D.Chamberlain	Chris Date	Steve	E.F Codd
6	To remove a relation from an SQL database, we use the _____ command.	Delete	Purge	Remove	Drop table	Drop table
7	The basic data type char(n) is a _____ length character string	Fixed	Equal	Variable	Empty	Fixed
8	The basic data type Varchar(n) is a _____ length character string	Fixed	Equal	Variable	Empty	Variable
9	Which one of the following provides the ability to query information from the database and to insert tuples into, delete tuples from, and modify tuples in the database?	DML(Data Manipulation Langauge)	DDL(Data Definition Langauge)	Relational Schema	Query	DML(Data Manipulation Langauge)
10	Which one of the following is used to define the structure of the relation, deleting relations and relating schemas?	DML(Data Manipulation Langauge)	DDL(Data Definition Langauge)	Relational Schema	Query	DDL(Data Definition Langauge)
11	The _____ data type represents the varying length string .	VARCHAR	NUMERIC	INT	CHAR	VARCHAR

12	Updates that violate _____ are disallowed.	Integrity constraints	Transaction control	Authorization	DDL constraints	Integrity constraints
13	The union operation is represented by	∩	U	-	*	U
14	The intersection operator is used to get the _____ tuples.	Different	Common	Repeating	All	Common
15	The union operation automatically _____ unlike the select clause.	Adds tuples	Eliminates unique tuples	Adds common tuples	Eliminates duplicate	Eliminates duplicate
16	If we want to retain all duplicates, we must write _____ in place of union.	Union all	Union some	Intersect all	Intersect some	Union all
17	The number of attributes in relation is called as its	Cardinality	Degree	Tuples	Entity	Degree
18	_____ clause is an additional filter that is applied to the result.	Select	Group-by	Having	Order by	Having
19	_____ joins are SQL server default	inner	outer	equi	right	inner
20	The _____ is essentially used to search for patterns in target string.	Like Predicate	Null Predicate	In Predicate	Out Predicate	Like Predicate
21	Which among the following is a transaction control statement?	Stop audit	Rollback	Insert	Start Audit	Rollback
22	In ACID property D stands for _____	Database	Durability	Deal	Development	Durability
23	Which of the following creates a virtual relation for storing the query?	Function	View	Procedure	None	View
24	Materialised views make sure that	View definition is kept stable	View definition is kept up-to-date	View definition is verified for error	View is deleted after specified time	View definition is kept up-to-date
25	Which of the following is used at the end of the view to reject the tuples which do not satisfy the condition in where clause?	With	Check	With check	All	With check
26	Aggregate functions can be	Where	Having	Group By	sum	Having

	used in the select list or the_____clause of a select statement or subquery					
27	In some languages such as Cobol, semicolon is replaced with	EXEC	START-EXEC	END - EXEC	SQL-EXEC	END - EXEC
28	Which of the following is way to build dynamic sql statements ?	Writing a query with parameters	Using sp_executesql	Using EXEC	All of the Mentioned	All of the Mentioned
29	_____ operators are used to compare one expression with another.	Comparision	Logical	Set	Arithmetic	Comparision
30	The Programming language in which the SQL Statements are included is called as_____language.	Host	Command	Procedure	Source	Host
31	Dynamic SQL Statements in SQL Server can be easily built using	Cursor	Stored procedure	Function	All of the Mentioned	Stored procedure
32	Process of finding a good strategy for processing a query is called	Query optimization	Query processing	Query management	Query cost	Query optimization
33	Advanced optimization techniques includes optimization of updates, multiquery optimization and	Top-K optimization	Join minimization	Parametric query optimization	All of the Above	All of the Above
34	Optimizer that explores space of all query-evaluation plans is called	Cost-based	Plan-based	Estimate-based	Count-based	Cost-based
35	Every Embedded SQL Statement has to be started with the _____ statement.	END-EXEC	SQLCODE	SQLSTATE	EXEC SQL	EXEC SQL
36	An option in view maintenance, in which only affected part of view is modified is known as	Immediate view maintenance	Deferred view maintenance	Data-view maintenance	Incremental view maintenance	Incremental view maintenance
37	Process of replacement of a nested query by a query with a	Correlation	Decorrelation	Materialization	Non-materialization	Decorrelation

	join is called					
38	_____checks the tokenized representation for correct syntax	Scanner	Parser	Compiler	Query optimizer	Parser
39	Term that optimizes subexpressions shared by different expressions in a program is called	Multiple subexpression elimination	Common subexpression elimination	Parametric subexpression elimination	Shared subexpression elimination	Common subexpression elimination
40	Each relational-algebra expression represents a particular sequence of	Parameters	Differentials	Operations	Routines	Operations
41	Operations of theta-join are	Distributive	Conjunctive	Commutative	Associative	Commutative
42	Histograms can be computed using	Population	Average of data	Sampling	Hard data	Sampling
43	The task of keeping a view up to date with the underlying data is called as	View handling	View maintainance	View Mnagement	Data maintenance	View maintainance
44	Modern database systems provide more direct support for	Incremental view maintenance	Data-view maintenance	Deferred view maintenance	Immediate view maintenance	Incremental view maintenance
45	Two expressions generate same set of tuples on every legal database instance, relational expressions are said to be	Dependant	Independent	Equivalent	Non-equivalent	Equivalent
46	Materialized views are used to speed up	Query optimization	Query processing	Query management	Query cost	Query processing
47	Which of the following is used to access the database server at the time of executing the program and get the data from the server accordingly?	Embedded SQL	Dynamic SQL	SQL declarations	SQL data analysis	Dynamic SQL
48	Which of the following invokes functions in sql?	Prepared Statements	Connection statement	Callable statements	All of the mentioned	Callable statements
49	Which of the following function is used to find the column count of the particular resultset?	getMetaData()	Metadata()	getColumn()	get Count()	getMetaData()
50	Which of the following is used	EXEC SQL	EXEC SQL	EXEC SQL	EXEC SQL <embedded	EXEC SQL <embedded SQL

	as the embedded SQL in COBOL?	<embedded SQL statement >;	<embedded SQL statement > END-EXEC	<embedded SQL statement >	SQL statement > END EXEC;	statement > END-EXEC
51	Which of the following is used to distinguish the variables in SQL from the host language variables?	.	–	:	,	:
52	_____attempts to determine the most efficient way to execute a given query	Scanner	Parser	Compiler	Query optimizer	Query optimizer
53	A _____ is a request for information from a database	Code	Rule	Language	Query	Query
54	QBE stands for	Query By Example	Query By Endowment	Query By Entity	Query By Entry	Query By Example
55	A rule based Optimization is a _____	Heuristic Optimization	Cost Optimization	Select Optimization	Default Optimization	Heuristic Optimization
56	_____ invloves number of memory buffers needed for query execution	Storage Cost	Computation Cost	Memory Cost	Communication Cost	Memory Cost
57	_____ involves the cost for performing in memory operations on the data buffers	Storage Cost	Computation Cost	Memory Cost	Communication Cost	Computation Cost
58	_____ involves the cost for storing any intermediate files	Storage Cost	Computation Cost	Memory Cost	Communication Cost	Storage Cost
59	_____ involves the cost of commuicating the query from the source	Storage Cost	Computation Cost	Memory Cost	Communication Cost	Communication Cost
60	_____ are used to get accurate estimates of the distribution of column data	Histograms	Graphs	Line Charts	Pie Charts	Histograms

S. N0	QUESTIONS	OPTION 1	OPTION 2	OPTION 3	OPTION 4	ANSWER
1	A transaction is delimited by statements (or function calls) of the form _____	Begin transaction and end transaction	Start transaction and stop transaction	Get transaction and post transaction	Read transaction and write transaction	Begin transaction and end transaction
2	Identify the characteristics of transactions	Atomicity	Isolation	Durability	All of the mentioned	All of the mentioned
3	Which of the following has “all-or-none” property?	Atomicity	Isolation	Durability	All of the mentioned	Atomicity
4	The database system must take special actions to ensure that transactions operate properly without interference from concurrently executing database statements. This property is referred to as	Atomicity	Isolation	Durability	All of the mentioned	Isolation
5	The property of a transaction that persists all the crashes is	Atomicity	Isolation	Durability	All of the mentioned	Durability
6	_____state that only valid data will be written to the database.	Consistency	Atomicity	Durability	Isolation	Consistency
7	Transaction processing is associated with everything below except	Producing detail summary or exception reports	Recording a business activity	Confirming an action or triggering a response	Maintaining a data	Confirming an action or triggering a response
8	The Oracle RDBMS uses the ____ statement to declare a new transaction start and its properties.	BEGIN	SET TRANSACTION	BEGIN TRANSACTION	COMMIT	BEGIN TRANSACTION
9	means that the data used during the execution of a transaction cannot be used by a second transaction until the first one is completed.	Consistency	Atomicity	Durability	Isolation	C
10	Locks placed by command are called _____ .	implicit locks	explicit locks	exclusive locks	shared locks	shared locks
11	Which of the following locks the item from change but not from read?	Implicit lock	Explicit lock	Exclusive lock	Shared lock	Shared lock
12	The advantage of optimistic locking is that:	the lock is obtained only after the transaction has processed.	the lock is obtained before the transaction has processed.	the lock never needs to be obtained.	transactions that are best suited are those with a lot of activity.	the lock is obtained only after the transaction has processed.
13	The ability of DBMS to manage various transactions is _____	Transaction Management	Transaction Control	Concurrency Control	Concurrency management	Transaction Management
14	Either all operations or no operations are done.This Property is	Atomicity	Consistency	Isolation	Durability	Atomicity

	called _____					
15	A _____ is an executing Program that forms a logical unit of database Processing	Query	Transaction	Command	Statement	Transaction
16	Execution of transaction in Isolation so that the database may remain correct is, _____	Atomicity	Consistency	Isolation	Durability	Consistency
17	Each Transaction is unaware of another transaction.This Property is called _____	Atomicity	Consistency	Isolation	Durability	Isolation
18	_____locking uses lock_item and unlock_item.	binary	read	write	fetch	binary
19	A chronological execution sequence of a transaction is called _____.	Schedule	Serial	Allotment	Concurreny	Schedule
20	The Property of Permannent Change occuring to the database is _____	Atomicity	Consistency	Isolation	Durability	Durability
21	_____is the intial State of transaction	Active	Partially Commited	Commmited	Failed	Active
22	_____is the process of managing simultaneous operations	Cocurrency Control	Transaction	Data Access	data control	Cocurrency Control
23	_____ lock has only two states.	Shared	Binary	Exclusive	RW	Binary
24	In _____,the system checks if the state of deadlock actually exists or not.	Deadlock Detection	Deadlock Control	Deadlock Management	Deadlock Prevention	Deadlock Detection
25	The order of execution is given importance in _____Schedule	Serilizable	Non Serializable	Time	Deadlock	Serilizable
26	A _____ locked item is also called as share-locked item.	read	write	data	fetch	read
27	The Read lock is also called as _____	Shared lock	Exclusive lock	Binary Lock	Two state lock	Shared lock
28	The Write lock is also called as _____	Shared lock	Exclusive lock	Binary Lock	Two state lock	Exclusive lock
29	In ACID property D stands for _____	Development	Durability	Database	Deal	Durability
30	Two schedules are in _____ equivalence when the transcation in both the schedules perform similar actions in a similar manner.	result	conflict	view	selection	view
31	If two schedules produce the same result after execution,then they said to be in _____.	result	conflict	view	selection	result
32	In _____,a list of conjuctive conditions can be broken upto several individual conditions.	Cascading selection	Cascading projection	Commutativity	Associativity	Cascading selection
33	A system is in a _____ state if there exists a set of transactions such that every transaction in the set is waiting for another transaction in the set.	Idle	Waiting	Deadlock	Ready	Deadlock
34	The deadlock state can be changed back to stable state by using	Commit	Rollback	Savepoint	Deadlock	Rollback

	_____ statement.					
35	When transaction Ti requests a data item currently held by Tj, Ti is allowed to wait only if it has a timestamp smaller than that of Tj (that is, Ti is older than Tj. Otherwise, Ti is rolled back (die). This is	Wait-die	Wait-wound	Wound-wait	Wait	Wait-die
36	The situation where the lock waits only for a specified amount of time for another lock to be released is	Lock timeout	Wait-wound	Timeout	Wait	Lock timeout
37	The deadlock in a set of a transaction can be determined by	Read-only graph	Wait graph	Wait-for graph	All of the mentioned	Read-only graph
38	A deadlock exists in the system if and only if the wait-for graph contains a _____	Cycle	Direction	Bi-direction	Rotation	Cycle
39	This validation scheme is called the _____ scheme since transactions execute optimistically, assuming they will be able to finish execution and validate at the end	Validation protocol	Validation-based protocol	Timestamp protocol	Optimistic concurrency-control	Validation protocol
40	_____ rollback requires the system to maintain additional information about the state of all the running transactions.	Total	Partial	Time	Commit	Partial
41	In a granularity hierarchy the highest level represents the	Entire database	Area	File	Record	Entire database
42	In a database the file is contained in _____	Entire database	Two area	One area	more than one area	One area
43	_____ denotes the largest timestamp of any transaction that executed write(Q. successfully.	W-timestamp(Q)	R-timestamp(Q)	RW-timestamp(Q)	WR-timestamp(Q)	W-timestamp(Q)
44	The _____ requires that each transaction Ti executes in two or three different phases in its lifetime, depending on whether it is a read-only or an update transaction.	Validation protocol	Validation-based protocol	Timestamp protocol	Optimistic concurrency-control	Validation protocol
45	A protocol that ensures system will never enter a deadlock state is called	Deadlock detection	Deadlock elimination	Deadlock prevention	Deadlock recovery	Deadlock prevention
46	An approach named Lock timeouts is used for	Deadlock detection	Deadlock elimination	Deadlock prevention	Deadlock recovery	Deadlock prevention
47	Deadlock prevention scheme that requires each transaction to locks all its data items before it begins	Initialization	Evaluation	Execution	Processing	Execution
48	Deadlock prevention scheme named wound-wait is a	Non-linear preemptive technique	Linear preemptive technique	Nonpreemptive technique	Preemptive technique	Preemptive technique
49	For dealing with deadlock problem, principal methods are of	2 types	3 types	4 types	5 types	2 types
50	Rigorous two-phase locking protocol permits releasing all locks at the	Beginning of transaction	During execution of transaction	Never in the life-time of transaction	End of transaction	End of transaction

51	System must deal with deadlocks that are not prevented by using schemes of	Validation	Deadlock detection	Deadlock recovery	Both A and B	Both A and B
52	A scheme that creates a new version of a data item for each transaction is defined by	Concurrency control scheme	Multiversion concurrency control scheme	Timestamp concurrency control scheme	Wound concurrency control scheme	Multiversion concurrency control scheme
53	Which of the following is not a recovery technique?	Deferred update	Immediate update	Two-phase commit	Recovery management	Two-phase commit
54	Checkpoints are a part of	Recovery measures	Security measures	Concurrency measures	Authorization measures	Recovery measures
55	..... deals with soft errors, such as power failures.	system recovery	media recovery	database recovery	failure recovery	failure recovery
56	Rollback of transactions is normally used to :	recover from transaction failure	update the transaction	retrieve old records	repeat a transaction	recover from transaction failure
57	A transaction performs on the isolation level of the Read Uncommitted if :	A dirty read occurs	Non-repeatable read occurs	Phantom reads occurs	All of the above	All of the above
58	Which commands are used to control which users have which privilege over which objects ?	QUE and QUIST	GRANT and REVOKE	CASECADE and MVD	None of the above	GRANT and REVOKE
59	When a deadlock is detected the recovery is normally accomplished by :	rollback of transaction	locking of data	consistency checking	none of these	rollback of transaction
60	A directed graph which represents the deadlock is called :	cycle graph	deadlock graph	wait-for graph	none of the above	wait-for graph

S.NO	QUESTIONS	OPTION 1	OPTION 2	OPTION 3	OPTION 4	ANSWERS
1	_____is a method of accessing and maintaining data	B+Tree indexing	Static hashing	Dynamic hashing	Multiple hashing	B+Tree indexing
2	A _____is a database that consists of two or more files located in different sites either on the same network or on entirely different networks.	distributed database	spatial database	parallel database	multimedia database	distributed database
3	_____ stores data in a distributed fashion, the distribution is governed solely by performance considerations.	distributed database	spatial database	parallel database	multimedia database	parallel database
4	_____ is a system used for reporting and data analysis	Data warehouse	Report warehouse	Analysis warehouse	System warehouse	Data warehouse
5	_____only focus on a single subject and only draw data from a handful of data sources.	data marts	Internet	Repository	Data Source	data marts
6	_____ works from remote corners of different locations of any particular area including homes, clients' premises, or simply while routing to remote locations.	Mobile database	Office network	Distributed database	Parallel database	Mobile database
7	_____ is a process used by companies to turn raw data into useful information.	Data mining	Process mining	Raw mining	Extraction mining	Data mining
8	_____is a means for separating the functions of an application into two or more distinct parts.	Client server technology	Application technology	Distinct technology	Function technology	Client server technology
9	_____ include one or more primary media data types such as text, images, graphic objects animation sequences, audio and video.	Multimedia database	Internet database	Html database	Primary database	Multimedia database
10	_____ is optimized for storing and querying data that represents objects defined in a geometric space.	Multimedia database	spatial database	Internet database	object database	spatial database
11	_____allows users to perform lookups on a finalized dictionary set	Static hashing	Dynamic hashing	B+ tree hashing	Look up hashing	Look up hashing
12	Index entries are stored sorted on the. search key value here	Ordered index	Hashed index	Tree index	Sequential index	Ordered index
13	Storing the files in certain order is called_____.	file Organization	database	table organization	ordering	file Organization
14	_____ is used for writing/reading of data to/from a magnetic ribbon.	Magnetic disk	Magnetic tape	Magnetic frames	Magnetic Ribbon	Magnetic tape
15	_____ is the fastest and most costly form of storage.	Cache	Flash memory	Main memory	NAND flash	Cache
16	_____is also known as extended hashing.	Static hashing	Dynamic hashing	B+ tree hashing	B tree hashing	Dynamic hashing
17	The method of access which uses key transformation is known as _____ .	Direct	Hash	Random	Sequential	Hash
18	An indexing operation	Sorts a file	sorts file	establishes	both a and b	establishes

		using a single key	using two keys	an index for a file		an index for a file
19	_____ refers to any type of application or presentation that involves more than one type of media, such as text, graphics, video, animation, and soun	An executable file	Desktop publishing	Multimedia	Hypertext	Multimedia
20	One of the disadvantages of multimedia is:	Cost	Adaptability	Usability	Relativity	Cost
21	_____ is a subject-oriented, integrated, time-variant, nonvolatile collection of data in support of management decisions.	Data Mining.	Data Warehousing.	Web Mining.	Text Mining.	Data Warehousing.
22	The data Warehouse is_____.	read only.	write only.	read write only.	none.	read only.
23	The important aspect of the data warehouse environment is that data found within the data warehouse is_____.	subject-oriented	time-variant.	integrate	All of the above.	All of the above.
24	What are the leaf nodes in a B+ tree?	The topmost nodes	The bottommost nodes	The nodes in between the top and bottom nodes	None of the mentioned	The bottommost nodes
25	A typical _____ program obtains a remote reference to one or more remote objects on a server and then invokes methods on them.	Server	Client	Thread	Concurrent	Client
26	The _____ layer, which provides the interface that client and server application objects use to interact with each other.	Increasing	Count	Bit	Stub/skeleton	Stub/skeleton
27	An object acting as a gateway for the client side.	skeleton	stub	remote	server	stub
28	Floppy disks are examples of :	primary storage	secondary storage	tertiary storage	none of the mentioned	tertiary storage
29	Storage media that is operated directly from computer's central processing unit is considered as	primary storage	secondary storage	tertiary storage	all of above	primary storage
30	The goal of _____ is to look at as few blocks as possible to find the matching records(s).	Indexing	Partitioning	Joining	None of above	Indexing
31	Which of the following is not a type of optical disk?	DVD	CD	WORM	Winchester	Winchester
32	The _____ is the fastest and most costly form of storage, which is relatively small; its use is managed by the computer system hardware.	Cache	Disk	Main memory	Flash memory	Cache
33	Which of the following stores several gigabytes of data but usually lost when power failure?	Disk	Main memory	Flash memory	Secondary Memory	Main memory
34	Which is the cheapest memory device in terms of costs/ bit?	Semiconductor memory	Magnetic disks	Compact disks	Magnetic tapes	Compact disks

35	Tape storage is referred to as _____ storage.	Direct-access	Random-access	Sequential-access	All of the mentioned	Sequential-access
36	A _____ is the smallest unit of information that can be read from or written to the disk.	Track	Spindle	Sector	Platter	Sector
37	Which level of RAID refers to disk mirroring with block striping?	RAID level 1	RAID level 2	RAID level 0	RAID level 3	RAID level 1
38	Optical disk technology uses	Helical scanning	DAT	A laser beam	RAID	RAID
39	With multiple disks, we can improve the transfer rate as well by _____ data across multiple disks.	Striping	Dividing	Mirroring	Dividing	Striping
40	_____ is popular for applications such as storage of log files in a database system since it offers the best write performance.	RAID level 1	RAID level 2	RAID level 0	RAID level 3	RAID level 1
41	Tertiary storage is built with :	a lot of money	unremovable media	removable media	secondary storage	removable media
42	Which one of the following is not a secondary storage?	magnetic disks	magnetic tapes	RAM	cache	RAM
43	In the client / server model, the database:	Is downloaded to the client upon request	Is shared by both the client and server	Resides on the client side	Resides on the server side	Resides on the server side
44	The database design that consists of multiple tables that are linked together through matching data stored in each table is called	Hierarchical database	Network database	Object oriented database	Relational database	Relational database
45	An _____ consists of a search-key value and pointers to one or more records with that value as their search-key value.	Index entry	Index hash	Index cluster	Index map	Index entry
46	In B+ tree the node which points to another node is called	Leaf node	External node	Final node	Internal node	Internal node
47	The physical location of a record is determined by a mathematical formula that transforms a file key into a record location is	B-Tree	Hashed File	Indexed File	Sequential File	Hashed File
48	The index consists of	a list of keys	pointers to the master list	both a and b	all of above	both a and b
49	The files used for speedy disk search by providing the specialized structures of data are classified as	indexes	glossaries	content specification	listing documents	indexes
50	A(n) _____ can be used to preserve the integrity of a document or a message	message digest	message summary	encrypted message	None of above	encrypted message
51	The goal of data mining includes _____	to explain some	to confirm that data	to analyse data for	to explain some	

		observed event or condition	exists	expected relationship	observed event or condition	
52	A data warehouse can be	updated by end users	containing numerous naming conventions and formats	organized around important subject areas	contaning only current data	organized around important subject areas
53	A Program that accepts requests from aweb browser and sends back resultsin form of HTML documents is known as	web host	web server	web interface	web application	web server
54	Communication between application program and a database server,takes place through	ODBC	JDBC	HTTP	ODBC or JDBC	ODBC or JDBC
55	Standard that defines communication between web server and application programs is known to be	HTTP	CGI	TTP	ATP	CGI
56	Languages designed to be executed on client's web browser are called	Client side modelling languages	Client side assembly languages	Client side scripting languages	Client side unified languages	Client side scripting languages
57	URL Stands for _____	Uniform Resource Locator	Universal Resource Locator	Universal Research Locator	Uniform Research Locator	Uniform Resource Locator
58	_____is a database that can be connected to by a mobile computing device over a mobile network	Distributed database	Mobile Database	Web Database	None of above	Mobile Database
59	_____database is portable and physically separate from the corporate database server	Distributed database	Mobile Database	Web Database	None of above	Mobile Database
60	_____ database gives you the ability to build your own database storage	Distributed database	Mobile Database	Web Database	None of above	Web Database

S.NO	QUESTIONS	OPTION 1	OPTION 2	OPTION 3	OPTION 4	ANSWERS
1	Database system which supports majority of concurrent users is classified as _____	multi-function system	multi transaction system	client and disk server system	multiuser system	multiuser system
2	_____inspects and interprets files looking for sensitive information	Context-based classification	User-based classification	Data Based	Content based classification	Content based classification
3	A _____ is permission to access a named object in a prescribed manner.	Right	Ownership	Query	Priviledge	Priviledge
4	The database administrator who authorizes all the new users, modifies the database and takes grants privilege is	Administrator	Operator of operating system	Super user	All of the mentioned	All of the mentioned
5	If we wish to grant a privilege and to allow the recipient to pass the privilege on to other users, we append the _____ clause to the appropriate grant comman	With grant	Grant user	Grant pass privilege	Revoke	With grant
6	A _____is a database used for statistical analysis purposes	Analytical database	OLAP	Scientific Database	Statistical database	Statistical database
7	Which of the reasons will force you to use XML data model in SQL Server ?	Your data represents containment hierarchy	Order is inherent in your data	All of the Mentioned	Your data is sparse or you do not know the structure of the data	All of the Mentioned
8	Which of the following is not a XML storage option ?	Mapping between XML and relational storage	Small object storage	None of the Mentioned	Native storage as XML data type	Small object storage
9	What is hybrid model in SQL Server ?	Combination of relational and non relational data type columns	Using XML with views	Using XML with triggers	Combination of relational and XML data type columns	Combination of relational and XML data type columns
10	Which of the following part of the XML data stored in an XML column is very important for locking ?	Degree of Structure	Hierarchy	Granularity	None of the mentioned	Granularity
11	ODL supports which of the following types of association relationships?	Unary and Binary	Unary, Binary and Ternary	Ternary	Unary	Unary and Binary
12	It is the process of creating index for the fetched pages	Processing	Sequencing	Fetching	Indexing	Indexing
13	_____ is the process by which indexingis done	Linking	Crawling	Listing	Indexing	Crawling
14	Datawarehouse is	Read write	Write only	Read only	None of the these	Read only

15	_____ predicts future trends and behaviour allowing managers to take proactive decisions	Data mining	Data marts	Data decision	Data Warehouse	Data mining
16	What is hybrid model in SQL Server ?	Combination of relational and XML data type columns	Combination of relational and non relational data type columns	Using XML with views	Using XML with triggers	Combination of relational and XML data type columns
17	Which of the following part of the XML data stored in an XML column is very important for locking ?	Granularity	Degree of Structure	Hierarchy	Degree of Storage	Granularity
18	Which of the following feature of SQL Server was used before XML technology for semi structured data?	Stored Procedure	Dynamic management views	In memory database	Dynamic data views	Stored Procedure
19	Authorization on a relation allows a user to update any tuple in relation, is known to be	select authorization	grant authorization	define authorization	update authorization	update authorization
20	Privileges that are granted implicitly to all current and future users, are called as	Private	Natural	Unnatural	Public	Public
21	_____ is the practice and precautions taken to protect valuable information from unauthorised access, recording, disclosure or destruction.	Network Security	Database Security	Information Security	Physical Security	Information Security
22	From the options below, which of them is not a threat to information security?	Disaster	Eavesdropping	Information leakage	Unchanged default password	Unchanged default password
23	_____ platforms are used for safety and protection of information in the clou	Cloud workload protection platforms	Cloud security protocols	AWS	One Drive	Cloud workload protection platforms
24	Which of the following information security technology is used for avoiding browser-based hacking?	Anti-malware in browsers	Remote browser access	Adware remover in browsers	Incognito mode in a browser	Remote browser access
25	Compromising confidential information comes under _____	Bug	Threat	Vulnerability	Attack	Threat
26	Possible threat to any information cannot be _____	reduced	transferred	protected	ignored	ignored
27	Each of types of authorizations is called a/an	Grant access	Privilege	Assignment	Access	Privilege
28	Privileges are granted over some specified parts of a database, such as a	Relation Or view	Schema	Environment	Query statement	Relation Or view
29	In cryptography, what is cipher?	algorithm for performing encryption and	encrypted message	both algorithm for performing	decrypted message	algorithm for performing encryption and

		decryption		encryption and decryption and encrypted message		decryption
30	In asymmetric key cryptography, the private key is kept by	sender	receiver	sender and receiver	all the connected devices to the network	receiver
31	In cryptography, the order of the letters in a message is rearranged by	transpositional ciphers	substitution ciphers	both transpositional ciphers and substitution ciphers	transaction ciphers	transpositional ciphers
32	A distributed database has which of the following advantages over a centralized database?	Software cost	Software complexity	Slow response	Modular growth	Modular growth
33	Memory address refers to the successive memory words and the machine is called as _____	word addressable	byte addressable	bit addressable	Terra byte addressable	word addressable
34	The _____ layer, which provides a high-level view of data and actions on dat	Business logic	Presentation	User interaction	Data access	Business logic
35	The _____ layer, which provides the interface between the business-logic layer and the underlying database.	Business logic	Presentation	User interaction	Data access	Data access
36	Which of the following is finally produced by Hierarchical Clustering ?	final estimate of cluster centroids	tree showing how close things are to each other	assignment of each point to clusters	assignment of data	tree showing how close things are to each other
37	Which of the following clustering requires merging approach ?	Partitional	Hierarchical	Naive Bayes	None of the Mentioned point to clusters	Hierarchical
38	Which method of analysis does not classify variables as dependent or independent?	regression analysis	discriminant analysis	analysis of variance	cluster analysis	cluster analysis
39	The most important part of _____ is selecting the variables on which clustering is base	interpreting and profiling clusters	selecting a clustering procedure	assessing the validity of clustering	formulating the clustering problem	formulating the clustering problem
40	The most commonly used measure of similarity is the _____ or its square.	Euclidean distance	city-block distance	Chebychev's distance	Manhattan distance	Euclidean distance

41	Websites fetched by crawler are indexed and kept in huge database, this process is called as _____.	Indexing	Optimizing	Crawling	Processing	Indexing
42	_____ command can be used to modify a column in a table	alter	update	set	create	alter
43	Grant and revoke are ……. statements.	DDL	TCL	DCL	DML	DCL
44	Which of the following does authentication aim to accomplish?	Restrict what operations/data the user can access	Determine if the user is an attacker	Flag the user if he/she misbehaves	Determine who the user is	Determine who the user is
45	Which of the following does authorization aim to accomplish?	Restrict what operations/data the user can access	Determine if the user is an attacker	Flag the user if he/she misbehaves	Determine who the user is	Restrict what operations/data the user can access
46	In ordered indices the file containing the records is sequentially ordered, a _____ is an index whose search key also defines the sequential order of the file.	Clustered index	Structured index	Unstructured index	Nonclustered index	Clustered index
47	An _____ consists of a search-key value and pointers to one or more records with that value as their search-key value.	Index entry	Index hash	Index cluster	Index map	Index entry
48	In a _____ clustering index, the index record contains the search-key value and a pointer to the first data record with that search-key value and the rest of the records will be in the sequential pointers.	Dense	Sparse	Straight	Continuous	Dense
49	In a _____ index, an index entry appears for only some of the search-key values.	Dense	Sparse	Straight	Continuous	Dense
50	In case the indices values are larger, index is created for these values of the index. This is called	Pointed index	Sequential index	Multiple index	Multilevel index	Multilevel index
51	The search results are generally presented in a line of results often referred to as	Tag List	Search engine results pagse	Search Engine Pages	Category List	Search engine results pagse
52	Search engines are able to search _____type of information	Videos	documents	images	all of the mentioned	all of the mentioned
53	Arrange the search engines by their year of development : 1.Bing,2.Yahoo,3.Ask,4.Google	2 4 1 3	2 1 3 4	2 1 4 3	2 4 3 1	2 4 3 1
54	Web search engines stores information about many web pages by a _____	web indexer	web organizer	web router	web crawler	web crawler
55	Web crawler is also called as _____	link directory	search optimizer	web spider	web manager	web spider
56	HTML stands for _____	Hyper Text Markup Language	Hybrid Text Markup Language	Hyper Text Manipulation Language	None of the above	Hyper Text Markup Language

57	ISP Stands for _____	International service Provider	Internet Service Provider	Internet Server Provider	None of the above	Internet Service Provider
58	WWW is the short form for _____	World Wide Web	Word Wide Web	Word Wild Web	Word Web Wide	World Wide Web
59	HTTP stands for _____	Hybrid Tech Transfer Protocol	Hyper Text Transparent Protocol	Hyper Text Transfer Protocol	Hyper Test Transfer Protocol	Hyper Text Transfer Protocol
60	Grouping a set of objevts in such a way that objects in the same group are more similar is called _____	Grouping	Ordering	Clustering	Aggregating	Clustering