DBMS
A database management system is a collection of interrelated data and a set of programs to access those data.
The collection of data usually referred to as the database., contains info relevant to an enterprise.
The primary goal of a dbms is to provide a way to store and retrieve database info that is both convenient and efficient.

CHARACTERISTICS OF DBMS

DBMS vs FILE SYSTEM
- Data redundancy and inconsistency
- Difficulty in accessing data
- Data isolation
- Integrity problems
- Atomicity problems Security problems

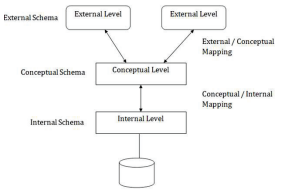
DATABASE USERS
- ***Sophisticated users*** – They interact with the system without writing programs.
- ***Naive Users*** – They are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously.
- ***Application Programmers*** – They are computer professionals who write application programs.
- ***Specialized Users*** – They are sophisticated users who write specialized dB applications that do not fit into the traditional data – processing framework.

STRUCTURED DATA
The info stored in relational db's are known as structures data because it is represented in a strict format.
Eg : tables, excel sheets.

SEMI STRUCTURED DATA
It is info that doesn't reside in a relational db but that has some organizational properties that make it easier to analyze.
Eg : graph, ER diagrams.

UNSTRUCTURED DATA
It is a data which is not organized in a predefined manner.
Eg : html programming, source codes.

THREE SCHEMA ARCHITECTURE
The main goal of the architecture is to separate the user application and the physical database.
Schema can be divided in 3 levels:
1. Internal level
2. Conceptual level
3. External level



Internal Level: It describes the physical structure of the db.
Conceptual level: It hides the details of the physical storage structure and concentrates on describing entities, data types, relationships, etc.
External Level: It describes the part of the db that a user is interested in and hides the rest of the db from the user group.

DATABASE LANGUAGES
1. DDL (Data Definition Language): used to specify the conceptual schema only.

ER MODEL
A popular high level conceptual data model. Conceptual data models use concepts such as entities, attributes and relationships
Entity: An entity represents a real-world object.
Relationship: Relationship among those objects.
Attributes: set of properties for describing the entities.

ENTITY
An entity represents a real-world object. It can be a person, place, job, etc. A collection of entities that have the same attributes is called entity type. Eg : STUDENT Collection of entities of a particular entity type at a point in time is called entity set.

WEAK ENTITY
The entity sets which do not have sufficient attributes to form a primary key are known as weak entity sets.

TYPES OF ATTRIBUTES
1. Simple and composite attributes: composite attributes can be divided into further parts (eg: name) and single attribute cannot be divided into further (eg: age)
2. Single valued and multi valued attributes: Single Valued attribute have a single value for a particular entity (eg: age) and multi valued attribute can have set of values for a particular entity (eg: languages known).
3. Derived and Stored attributes: Derived attributes can be derived from other attributes (eg: age) and stored attributes are attributes from which the values of other attributes are derived (eg: birth date).

RELATIONAL MODEL
- The relational model represents the database as a collection of relations
- In the formal relational model terminology,
  A row is called a tuple
  A column is called an attribute
  The table is called a relation.

DOMAIN
- A set of atomic values allowed for an attribute
- Eg: Student age

RELATION SCHEMA
- Describes a relation Made up of a relation name R and a list of attributes A1, A2, ……An
- Eg: STUDENT (Name, Rollnum, Age) STUDENT (Name: string, Rollnum: integer, Age: integer)

DEGREE OF A RELATION
- Number of attributes in a relation schema
- Eg: STUDENT (Name, Rollnum, Age)- degree-3

CARDINALITY
- Total number of tuples present in a relation

RELATIONAL MODELS CONSTRAINTS
- Constraints are the restrictions or the limitations on data in the database
  1. Inherent model-based constraints or implicit constraints: Constraints that are inherent in the data model
  2. Schema-based constraints or explicit constraints: Constraints that are defined directly in the schemas of the data model
  3. Application-based or semantic constraints or business rules: Constraints that cannot be directly expressed in the schemas of the data model, and hence must be expressed and enforced by the application programs.

ER MODEL TO RELATIONAL MODEL
1.Create a table for each entity
2.Entity's attribute should become fields of tables with respect to data types
3.Declare primary key

RELATIONAL ALGEBRA
- It is a procedural query language
- It consists of a set of operation that take one or two relation as input and produce a new relation as their result
- It consists of two types of operation
- ***Unary operation:*** They operate on one relation.
  - Eg: Select, Project
- ***Binary operation:*** They operate on pairs of relationships.
  - Eg: Cartesian product

Select operation
- The select operation select tuples that satisfy a given predicate

Project operation
- It projects attributes(columns) that satisfy a given predicate

Cartesian product
- It combines information of two different relations into one relation

SQL
- The SQL (Structured Query Language) is a standard language for storing and managing data in a RDBMS.
- It enables a user to create, read, update and delete relational databases and tables
- SQL is not case sensitive. Generally, keywords of SQL are written in upper case.
- SQL comprises DDL and DML. Using DDL, one can design and modify database schema whereas DML used to store and retrieve data from database

DDL
CREATE
DROP
ALTER
- ***CREATE TABLE*** *tablename(col1 datatype,col2 datatype,....);*
  Eg: CREATE TABLE student(id int primary key,name varchar(20));
- ***ALTER TABLE*** *tablename ADD columnname datatype;*
  Eg: ALTER TABLE student ADD mark int
- ***DROP TABLE****/DATABASE tablename/databasename;*
  Eg: DROP TABLE student;

DML
INSERT
UPDATE
DELETE
- ***INSERT*** *INTO tablename (col1, col2,...) VALUES (value1,value2,..);*
  Eg: INSERT INTO student VALUES (1, ashi);
- ***UPDATE*** *tablename SET col1=val1, col2=val2,.. WHERE condition;*
  Eg:UPDATE student SET name='anu' WHERE id=1
- ***DELETE FROM*** *tablename WHERE condition;*
  Eg:DELETE FROM student WHERE id=1;

PHYSICAL FILES
- Physical files contain the actual data that is stored on the system, and a description of how data is to be presented to or received from a program.
- They contain only one record format, and one or more members.
- Records in database files can be externally or program-described. A physical file can have a keyed sequence access path.
- This means that data is presented to a program in a sequence based on one or more key fields in the file.

LOGICAL FILES
- Logical files do not contain data.
- They contain a description of records found in one or more physical files.
- A logical file is a view or representation of one or more physical files.
- Logical files that contain more than one format are referred to as multi-format logical files.
- If your program processes a logical file which contains more than one record format, you can use a read by record format to set the format you wish to use.

FILE ORGANIZATIONS
- File organization refers to the organization of the data of a file into records, blocks, and access structures; this includes the way records and blocks are placed on the storage medium and interlinked.
- An access method, on the other hand, provides a group of operations that can be applied to a file.
- In general, it is possible to apply several access methods to a file organization.

Spanned and Unspanned Organization
- A block is the unit of data transfer between disk and memory.
- When the block size is larger than the record size, each block will contain numerous records, although some files may have unusually large records that cannot fit in one block.
- Part of the record can be stored on one block and the rest on another.
- A pointer at the end of the first block points to the block containing the remainder of the record.
- This organization is called spanned because records can span more than one block.
- Whenever a record is larger than a block, we must use a spanned organization.
- If records are not allowed to cross block boundaries, the organization is called unspanned.

Allocating File Blocks on Disk
- Contiguous allocation
  - the file blocks are allocated to consecutive disk blocks.
  - This makes reading the whole file very fast using double buffering, but it makes expand x0002 ing the file difficult.
- Linked allocation
  - each file block contains a pointer to the next file block.
  - This makes it easy to expand the file but makes it slow to read the whole file.
- Clusters allocation
  - A combination of the two allocates clusters of consecutive disk blocks, and the clusters are linked.
  - Clusters are sometimes called file segments or extents.
- Indexed allocation
  - where one or more index blocks contain pointers to the actual file blocks

OPERATION ON FILES
Typical file operations include:
- ***OPEN:*** Readies the file for access, and associates a pointer that will refer to a current file record at each point in time.
- ***FIND:*** Searches for the first file record that satisfies a certain condition, and makes it the current file record.
- ***FINDNEXT:*** Searches for the next file record (from the current record) that satisfies a certain condition, and makes it the current file record.
- ***READ:*** Reads the current file record into a program variable
- ***INSERT:*** Inserts a new record into the file & makes it the current file record.
- ***DELETE:*** Removes the current file record from the file, usually by marking the record to indicate that it is no longer valid.
- ***MODIFY:*** Changes the values of some fields of the current file record.
- ***CLOSE:*** Terminates access to the file.
- ***REORGANIZE:*** Reorganizes the file records.
- For example, the records marked deleted are physically removed from the file or a new organization of the file records is created.
- ***READ_ORDERED:*** Read the file blocks in order of a specific field of the file.

HASHING TECHNIQUES
- The search condition must be an equality condition on a single field, called the hash field.
- In most cases, the hash field is also a key field of the file, in which case it is called the hash key.

A COLLISION
occurs when the hash field value of a record that is being inserted hashes to an address that already contains a different record.
- In this situation, we must insert the new record in some other position, since its hash address is occupied. The process of finding another position is called ***collision resolution.***

There are numerous ***methods for collision resolution***, including the following:
1. Open addressing: Proceeding from the occupied position specified by the hash address, the program checks the subsequent positions in order until an unused (empty) position is found.
2. Chaining: For this method, various overflow locations are kept, usually by extending the array with a number of overflow positions.
Additionally, a pointer field is added to each record location.
A collision is resolved by placing the new record in an unused overflow location and setting the pointer of the occupied hash address location to the address of that overflow location.
3. Multiple hashing: The program applies a second hash function if the first results in a collision.
If another collision results, the program uses open addressing or applies a third hash function and then uses open addressing if necessary.

TYPES OF ORDERED INDEXES
There are several types of ordered indexes.
- ***Primary index***
  - is specified on the ordering key field of an ordered file of records.
  - An ordering key field is used to physically order the file records on disk, and every record has a unique value for that field.
- ***Clustered Index***
  - If the ordering field is not a key field that is, if numerous records in the file can have the same value for the ordering field another type of index, called a clustering index, can be used.
  - The data file is called a clustered file.
  - Notice that a file can have at most one physical ordering field, so it can have at most one primary index or one clustering index, but not both.
- ***Secondary Index***
  - It can be specified on any non-ordering field of a file.
  - A data file can have several secondary indexes in addition to its primary access method

Primary Indexes
- A primary index is an ordered file whose records are of fixed length with two fields, and it acts like an access structure to efficiently search for and access the data records in a data file.
- The first field is of the same data type as the ordering key field called the primary key of the data file, and the second field is a pointer to a disk block (a block address).
- There is one index entry (or index record) in the index file for each block in the data file.

NORMALIZATION
- Normalization is the process of organizing the data in the database
- It is used to minimize the redundancy from a relation or set of relations
- It is also used to eliminate the insertion anomaly update anomaly and deletion anomaly
- It divides the larger table into the smaller table and links them using relationship

First Normal Form (1NF)
- A relation will be 1NF if it contains an atomic value.
- It states that an attribute of a table cannot hold multiple values.
- It must hold only single-valued attribute.

Second Normal Form (2NF)
- In the 2NF, relational must be in 1NF.
- In the second normal form, no non-prime attribute is dependent on the proper subset of any candidate key of table

Third Normal Form (3NF)
- In 3NF, the relation must be in 2NF
- Transitive functional dependency of non-prime attribute on any super key should be removed

Boyce Codd Normal Form (BCNF)
- BCNF is the advance version of 3NF.

FUNCTIONAL DEPENDENCY
- A functional dependency is an association between two attributes of the same relational database table.
- One of the attributes is called the determinant and the other attribute is called the determined
- If A is the determinant and B is the determined then we say that A functionally determines B and graphically represent this as A -> B.

Lossless join and dependency preserving decomposition
- Decomposition of a relation is done when a relation in relational model is not in appropriate normal form.
- Relation R is decomposed into two or more relations if decomposition is lossless join as well as dependency preserving.

Lossless Join Decomposition
- If the information is not lost from the relation that is decomposed, then the decomposition will be lossless.
- i.e., the relation is said to be lossless decomposition if natural joins of all the decomposition give the original relation.
- If we decompose a relation R into relations R1 and R2
  1. Decomposition is lossy if R1 ⨝ R2 is not R
  2. Decomposition is lossless if R1 ⨝ R2 is equal to R

THE DEPENDENCY PRESERVATION PROPERTY
- The dependency preservation property, which ensures that each functional dependency is represented in some individual relation resulting after decomposition
- In the dependency preservation, at least one decomposed table must satisfy every dependency.

TRANSACTION
Transactions group a set of tasks into a single execution unit.
Each transaction begins with a specific task and ends when all the tasks in the group successfully complete.
If any of the tasks fail, the transaction fails.
Therefore, a transaction has only two results: success or failure.
Incomplete steps result in the failure of the transaction.

ACID PROPERTIES
A – Atomicity: the entire transaction takes place at once or doesn't happen at all
C – Consistency: The dB must be consistent before and after the transaction
I – Isolation: Multiple Transactions occur independently without interference
D – Durability: The changes of a successful transaction occurs even if the system failure occurs

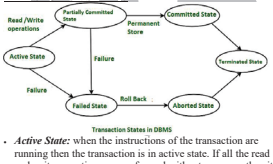
CONCURRENCY CONTROL
Concurrency Control in Database Management System is a procedure of managing simultaneous operations without conflicting with each other

Concurrency Control Protocols
- Lock-Based Protocols
- Two Phase Locking Protocol
- Timestamp-Based Protocols
- Validation-Based Protocols

LOCK-BASED PROTOCOLS
- Lock Based Protocols in DBMS is a mechanism in which a transaction cannot Read or Write the data until it acquires an appropriate lock.
- Lock based protocols help to eliminate the concurrency problem in DBMS for simultaneous transactions by locking or isolating a particular transaction to a single user.
- All lock requests are made to the concurrency-control manager. Transactions proceed only once the lock request is granted.

TRANSACTION MODEL
Transactions access data using two operations:
- ***read(X):*** which transfers the data item X from the database to a variable, also called X, in a buffer in main memory belonging to the transaction that executed the read operation.
- ***write(X):*** which transfers the value in the variable X in the main-memory buffer of the transaction that executed the write to the data item X in the database.

TRANSACTION STATES



Transaction States in DBMS

- ***Active State:*** when the instructions of the transaction are running then the transaction is in active state. If all the read and write operations are performed without any error then it goes to the partially committed state; if any instruction fails, it goes to the failed state.
- ***Partially committee:*** After Completion of all the read and write operation the changes are made in main memory or local buffer. If the changes are made permanent on the database, then the state will change to committed state and in case of failure it will go to the failed state.

SCHEDULE
- A series of operation from one transaction to another transaction is known as schedule.
- It is used to preserve the order of the operation in each of the individual transaction.
- The serial schedule is a type of schedule where one transaction is executed completely before starting another transaction. Schedule is divided into 3. Serial, non-serial, serializable schedules

Serial Schedule
- When the first transaction completes its cycle, then the next transaction is executed.

Non-Serial Schedules
- If interleaving of operations is allowed, then there will be non-serial schedule.
- It contains many possible orders in which the system can execute the individual operations of the transactions

Serializable schedule
- The serializability of schedules is used to find non-serial schedules that allow the transaction to execute concurrently without interfering with one another.
- It identifies which schedules are correct when executions of the transaction have interleaving of their operations.
- A non-serial schedule will be serializable if its result is equal to the result of its transactions executed serially.

CHECK-POINTING
- The methodology utilized for removing all previous transaction logs and storing them in permanent storage is called a Checkpoint.
- A checkpoint is used for recovery if there is an unexpected shutdown in the database.
- Checkpoints work on some intervals and write all dirty pages (modified pages) from logs relay to data file from i.e from a buffer to physical disk.
- It is also known as the hardening of dirty pages

SCHEMA BASED CONSTRAINTS

1. DOMAIN CONSTRAINTS
- Must be atomic value
- Performs data type check

2. KEY CONSTRAINTS
- An attribute that can uniquely identify each tuple in a relation is called a key
- A super-key specifies that no two tuples can have the same value
- Every relation has atleast one super-key - set of all attributes.
  - Eg: {RollNo, Email, {RollNo, Name},.....}
- Candidate key is a set of attributes that uniquely identify the tuples in a relation
  - Eg: RollNo and Email

3. CONSTRAINTS OF NULL VALUES
- Specifies whether null values are permitted or not (NOT NULL)
  - Eg:Name

4. ENTITY INTEGRITY CONSTRAINTS
- Specify that no primary key value can be null

RELATIONSHIP

It is the association among 2 or more entities.
Eg: Teacher teaches student
The number of entity types that participate in a relationship is called degree of relationship.

Unary relationship: exist when there is an association with only one entity.
Binary relationship: two entities.
Ternary relationship: three entities.

CONSTRAINTS

Mapping cardinalities or cardinality ratios:
Max number of relationship instances that an entity can participate in.

Participation Constraints:
Specifies whether existence of an entity depends on its being related to another entity.

2. SDL (Storage Definition Language): used to specify the internal schema.
3. VDL (View Definition Language): To specify user views and their mapping to the conceptual schema.
4. DML (Data Manipulation Language): for manipulation of data in the db.

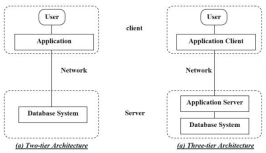
Types of DML :

High level (or non-procedural DML): specify what data are needed without specifying how to get those data.
Low level (or procedural DML): specify what data are needed and how get those data.

DATABASE ARCHITECTURES

1. Centralized architecture:
- they run on a single computer system and do not interact with other.
- Eg: keyboard, mouse connected in a PC

2. Client-Server systems:
- ***Client:*** user machine that provides user interface capabilities and local processing;
- ***Server:*** provides services to client machines



(a) Two-tier Architecture *(a) Three-tier Architecture*

DATA MODELS

A collection of conceptual diagrams that can be used to describe the structure of a database.

Categories of Data Models:
- ***High level or conceptual data model:***
  - provide concepts that are close to the way many users can perceive data.
  - Eg: Entity Relationship model.
- ***Low level or physical data model:***
  - provide concepts that describe the details of how data is stored on the computer storage media.
- ***Representational or implementation data models:***
  - which provide concepts that may be easily understood by end users but that are not too far removed from the way data is organized in computer storage.
  - Eg : network and hierarchical models

ENTITY RELATIONSHIP MODEL

A popular high – level conceptual model Conceptual data models that use concepts such as entities, attributes, and relationships

Entity: represents a real-world object
Relationship: relationship over objects.
Attributes: set of properties for describing the entities.

NETWORK MODEL

It is representational model. Data in the network model are represented by collection of records and the relationship among the data are represented by link.

SCHEMA
- The overall design of the database system is called schema Layout or blueprint of a db.
- The diagrammatic representation of schema is called schema diagram.

FILE HEADERS
- A file header or file descriptor contains information about a file that is needed by the system programs that access the file records.
- The header includes information to determine the disk addresses of the file blocks as well as to record format descriptions, which may include field lengths and the order of fields within a record for fixed-length unspanned records and field type codes, separator characters, and record type codes for variable-length records.
- To search for a record on disk, one or more blocks are copied into main memory buffers.
- Programs then search for the desired record or records within the buffers, using the information in the file header.
- If the address of the block that contains the desired record is not known, the search programs must do a linear search through the file blocks.
- Each file block is copied into a buffer and searched until the record is located or all the file blocks have been searched unsuccessfully.
- This can be very time-consuming for a large file.
- The goal of a good file organization is to locate the block that contains a desired record with a minimal number of block transfers.

Primary file organizations
- Primary file organizations determine how the file records are physically placed on the disk, and hence how the records can be accessed.
  1. Heap file (or unordered file) places the records on disk in no particular order by appending new records at the end of the file.
  2. Sorted file (or sequential file) keeps the records ordered by the value of a particular field (called the sort key).
  3. Hashed file uses a hash function applied to a particular field (called the hash key) to determine a record's placement on disk.
  4. Other primary file organizations, such as B-trees, use tree structures.

Secondary organization
- A secondary organization or auxiliary access structure allows efficient access to file records based on alternate fields than those that have been used for the primary file organization

Files, Fixed-Length Records, and Variable-Length Records
- A file is a sequence of records.
- In many cases, all records in a file are of the same record type.
- If every record in the file has exactly the same size (in bytes), the file is said to be made up of fixed-length records.
- If different records in the file have different sizes, the file is said to be made up of variable-length records.

JOIN

It is used to combine two tables based on a specified common field between them EQUI-JOIN, NATURAL JOIN

EQUI-JOIN
- It performs a JOIN against equality or matching column(s) values of the associated tables
- An equal sign(=) is used as comparison operator in the where clause to refer equality
  - *SELECT columnlist FROM table1,table2 WHERE table1.columnname = table2.columnname;*
- You may also perform EQUI-JOIN by using JOIN keyword followed by ON keyword
  - *SELECT * FROM table1 JOIN table2{ON (join condition)};*

NATURAL JOIN
- It is a type of EQUI-JOIN and is structured in such a way that, columns with the same name of associated tables will appear once only
  - *SELECT * FROM table1 NATURAL JOIN table2;*

ARMSTRONG'S AXIOMS
- Armstrong's Axiom is a mathematical notation used to find the functional dependencies in a database.
- Conceived by William W. Armstrong
- It is a list of axioms or inference rules that can be implemented on any relational database.
- It is denoted by the symbol F+.

CLOSURE OF FUNCTIONAL DEPENDENCY
- The Closure of Functional Dependency means the complete set of all possible attributes that can be functionally derived from given functional dependency
- If "F" is a functional dependency then closure of functional dependency can be denoted using "{F}+
- There are three steps to calculate closure of functional dependency
  - ***Step-1:*** Add the attributes which are present on Left-Hand Side in the original functional dependency.
  - ***Step-2:*** Now, add the attributes present on the Right-Hand Side of the functional dependency.
  - ***Step-3:*** With the help of attributes present on Right-Hand Side, check the other attributes that can be derived from the other given functional dependencies.
  - ***Step-4:*** Repeat this process until all the possible attributes which can be derived are added in the closure.

Equivalence of Functional Dependencies (FD)
- Two different sets of functional dependencies for a given relation may or may not be equivalent.
  - If FD1 can be derived from FD2, we can say that FD2 - FD1.
  - If FD2 can be derived from FD1, we can say that FD1 - FD2.
  - If above two cases are true, FD1=FD2

- Each index entry has the value of the primary key field for the first record in a block and a pointer to that block as its two field values.

Clustering Indexes
- If file records are physically ordered on a non key field which does not have a distinct value for each record that field is called the clustering field and the data file is called a clustered file.
- We can create a different type of index, called a clustering index.

External Hashing for Disk Files

Hashing for disk files is called external hashing. To suit the characteristics of disk storage, the target address space is made of buckets, each of which holds multiple records. A bucket is either one disk block or a cluster of contiguous disk blocks. The hashing function maps a key into a relative bucket number, rather than assigning an absolute block address to the bucket. A table maintained in the file header converts the bucket number into the corresponding disk block address.

Dynamic and Extendible Hashing Techniques

Hashing techniques are adapted to allow the dynamic growth and shrinking of the number of file records. These techniques include the following: dynamic hashing, extendible hashing, and linear hashing. Both dynamic and extendible hashing use the binary representation of the hash value h(K) in order to access a directory. In dynamic hashing the directory is a binary tree. In extendible hashing the directory is an array of size 2d where d is called the global depth.

- The idea behind hashing is to provide a function h, called a hash function or randomizing function, which is applied to the hash field value of a record and yields the address of the disk block in which the record is stored.
- A search for the record within the block can be carried out in a main memory buffer.
- For most records, we need only a single-block access to retrieve that record.

Internal Hashing

For internal files, hashing is typically implemented as a hash table through the use of an array of records.
Suppose that the array index range is from 0 to M – 1, ; then we have M slots whose addresses correspond to the array indexes. We choose a hash function that transforms the hash field value into an integer between 0 and M − 1.
One common hash function is the h(K) = K mod M function, which returns the remainder of an integer hash field value K after division by M; this value is then used for the record address

Other hashing functions available are

1. Folding
- involves applying an arithmetic function such as addition or a logical function such as exclusive or to different portions of the hash field value to calculate the hash address (for example, with an address space from 0 to 999 to store 1,000 keys, a 6-digit key 235469 may be folded and stored at the address: (235+964) mod 1000 = 199).

2. picking some digits of the hash field value
- for instance, the third, fifth, and eighth digits—to form the hash address (for example, storing 1,000 employees with Social Security numbers of 10 digits into a hash file with 1000positions would give the social security number 301-67-8923 a hash value of 172 by this hash function)

- It speeds up data recovery process.

NO SQL

NoSQL ("not only SQL") databases are non-tabular databases and store data differently than relational tables.
- NoSQL databases come in a variety of types based on their data model.
- The main types are document, key-value, wide-column, and graph.
- They provide flexible schemas and scale easily with large amounts of data and high user loads.

Key-value dB
- A key-value database (or key-value store) uses a simple key-value method to store data.
- These databases contain a simple string (the key) that is always unique and an arbitrary large data field (the value).
- They are easy to design and implement.

DOCUMENT DB
- Built around JSON-like documents, document databases are both natural and flexible for developers to work with.
- They promise higher developer productivity, and faster evolution with application needs.
- As a class of non-relational, sometimes called NoSQL database, the document data model has become the most popular alternative to tabular, relational databases.

Column-family databases

Column-family databases store data in column families as rows that have many columns associated with a row key.

GRAPH DB

Graph databases store schema-free objects (vertices or nodes) where arbitrary data can be stored (properties) and relations between the objects (edges).
- Edges typically have a direction going from one object to another or multiple objects.
- Vertices and edges form a network of data points which is called a "graph"

- ***Failed State:*** when any instruction of the transaction fails it goes to the failed state.
- ***Aborted State:*** After having the type of failure and transaction goes from failed state to aborted state.
- ***Committed State:*** It is the state when the changes are made permanent on the Data Base and the transaction is complete and therefore terminated in the terminated state.
- ***Terminated State:*** The transaction comes from the "committed state" goes to this state, then the system is consistent and ready for new transaction and the old Transaction is terminated.

SYSTEM LOG
- Log is a sequence of records, which maintains the records of actions performed by a transaction.
- It is important that the logs are written prior to the actual modification and stored on a stable storage media, which is failsafe. Log-based recovery works as follows o
- The log file is kept on a stable storage media.
- When a transaction enters the system and starts execution, it writes a log about it.
- <Tn, Start>
- When the transaction modifies an item X, it writes logs as <Tn, X, V1, V2>. It reads Tn has changed the value of X, from V1 to V2.
- When the transaction finishes, it logs <Tn, commit>

TWO PHASE LOCKING PROTOCOL (2 PL Protocol)

It is a method of concurrency control in DBMS that ensures serializability by applying a lock to the transaction data which blocks other transactions to access the same data simultaneously

Growing Phase: In this phase transaction may obtain locks but may not release any locks.
Shrinking Phase: In this phase, a transaction may release locks but not obtain any new lock

TIMESTAMP-BASED PROTOCOLS
- It is an algorithm which uses the System Time or Logical Counter as a timestamp to serialize the execution of concurrent transactions.
- It ensures that every conflicting read and write operations are executed in a timestamp order.
- The older transaction is always given priority in this method.
- It uses system time to determine the time stamp of the transaction.
- This is the most commonly used concurrency protocol.

VALIDATION BASED PROTOCOL
- It is also called Optimistic Concurrency Control Technique.
- It is called optimistic because of the assumption it makes, i.e., very less interference occurs, therefore, there is no need for checking while the transaction is executed.
- Until the transaction end is reached updates in the transaction are not applied directly to the database.
- All updates are applied to local copies of data items kept for the transaction.
- At the end of transaction execution, while execution of the transaction, a validation phase checks whether any of transaction updates violate serializability.
- If there is no violation of serializability the transaction is committed and the database is updated.