

# **Scribe Notes**

## **Week 1**

**(7,9,11 Jan)**

- **Adwait Thattey (S20170010004)**
- **Aman Kumar (S2017001009)**
- **Aman Gupta (S20170010008)**
- **Thoyiba (S2017001165)**
- **Sanjay Kumar (S2017001139)**

## **Contents**

### **1. Basic Concepts**

- a. Basic Definitions
- b. Big Data
- c. Simplified Database System Environment
- d. Database Facilities
- e. Drawbacks of using file systems
- f. Entity Relationships examples
- g. Main characteristics of database approach
- h. Actors on the scene
- i. Actors behind the scene
- j. Database on mobile phones
- k. Databases used by major companies

## **2. Database System Concepts and Architecture**

- a. Database Models
- b. Categories of database models
- c. Schemas and Instances
- d. ACID
- e. Three Schema Architecture
- f. Data Independence
- g. Database Languages
- h. DBMS Components
- i. Different Tiers of database architecture
- j. Classification of DBMS

## **3. Data modeling using Entity Relationship Diagrams**

- a. ER Diagrams
- b. Types of attributes
- c. Relationship types
- d. Common symbols
- e.

## **4. Terminology**

## **5. Sources**

# Basic Concepts

## Basic Definitions:

- Database: collection of related data.
- Data: Known facts that can be recorded and have implicit meaning.
- Database Management System (DBMS): Collection of programs to store, modify and extract information from a database through various queries.
- Database Schema: It is a skeleton structure that represents the logical view of the entire database. Analogous to variable type.
- Instance: The data which is stored in the database at a particular moment of time is instance of the database. Analogous to value of a variable.
- Query: A type of command that retrieves data from the server.
- Row: A set of related data i.e., A tuple.
- Column: They represent attributes. Each column has a type and a name.
- Transaction: It is a collection of database operation that are treated as a unit. It ends in either a commit or a rollback.

## What is Big Data?

Big Data are high volume, high velocity and/or high variety data sets that are too large or complex for traditional data-processing application software to adequately deal with.

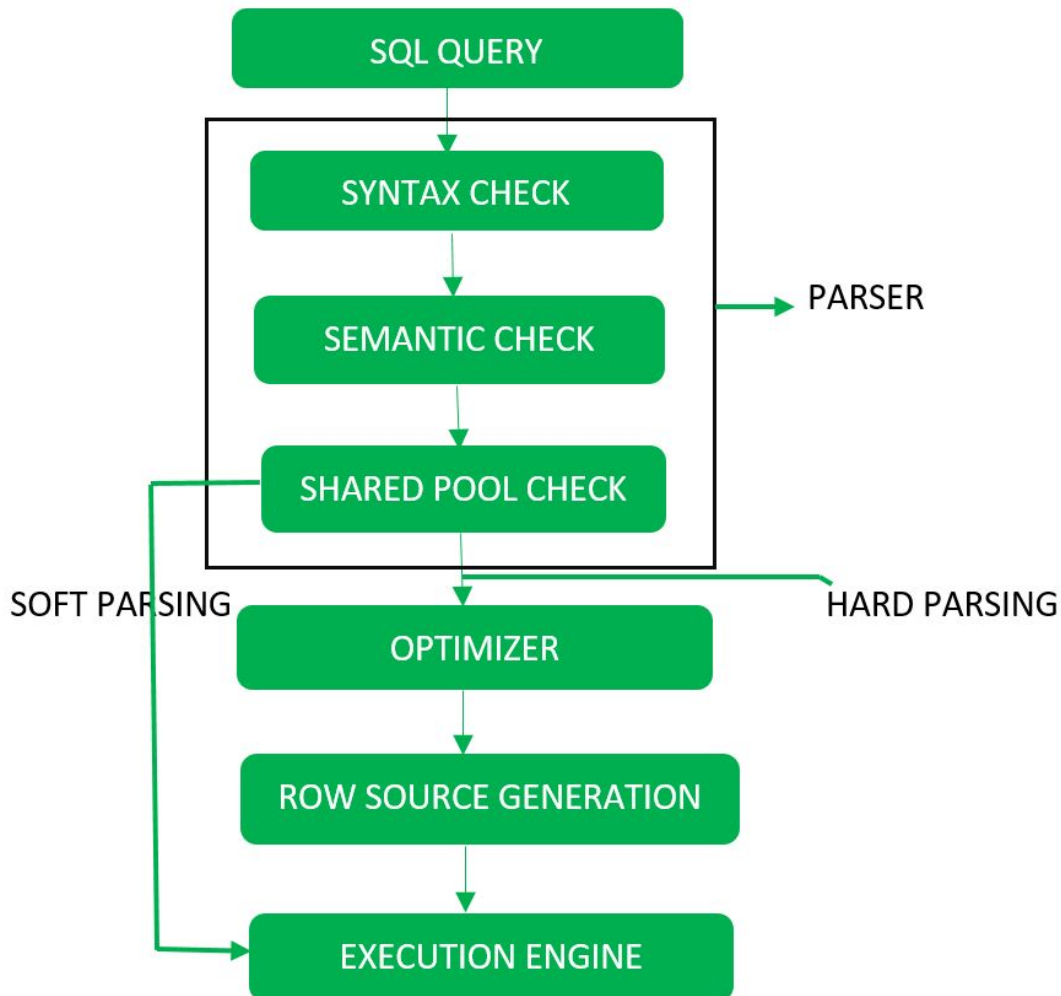
Walmart handles more than 1M customers transactions every hour, which is imported into database estimated to contain more than 2.5 PB of data.

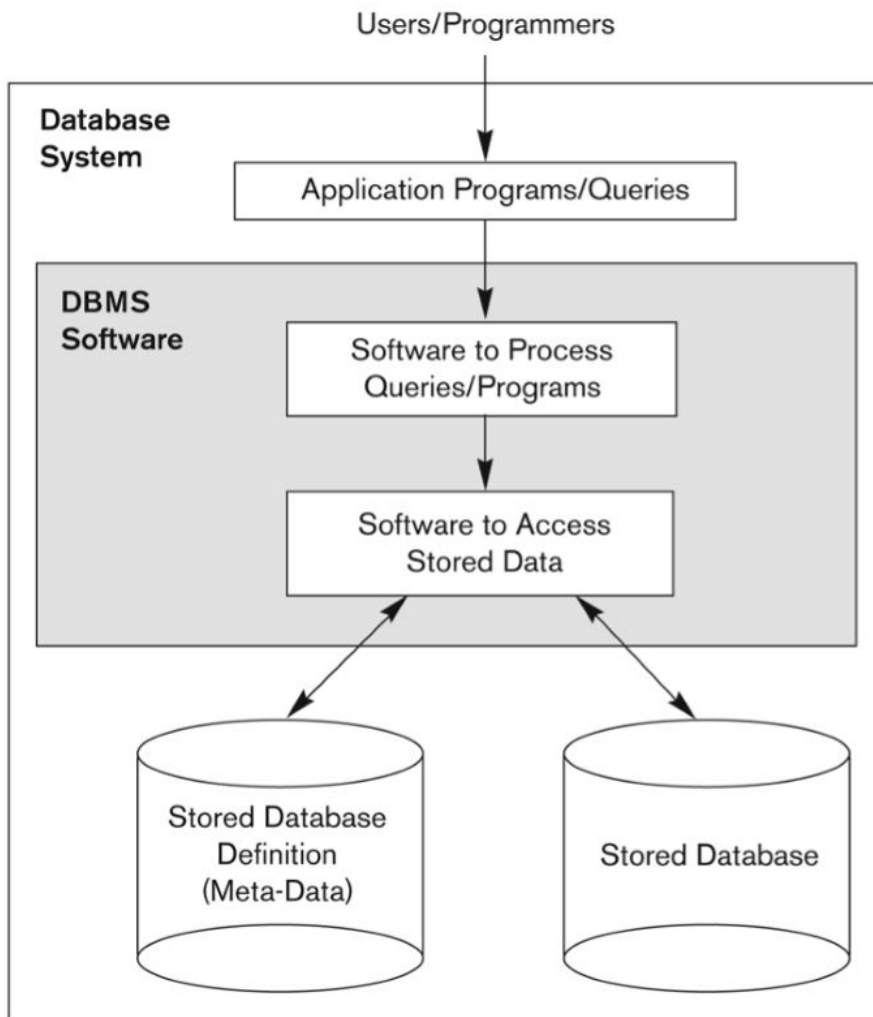
Facebook handles 40B photos from its user database.

Life Cycle of Data: 4"A"s

## Simplified Database System Environment:

An application program accesses the database by sending queries or requests for data to the DBMS. It is passed through Parser for syntax check, Semantic check and Shared Pool Check and then through Optimizer and finally a execution engine runs the query.





**Figure 1.1**  
A simplified database system environment.

## List of Database Facilitates:

- Create
- Insert
- Manipulate
  - o Retrieval
  - o Modification

- o Accessing
- Concurrent Access
- Protection & Security
  - o System Protection: Hardware and Software malfunction/crashes
  - o Security Protection: Unauthorized / malicious access
- Presentation and Visualization
- Maintenance

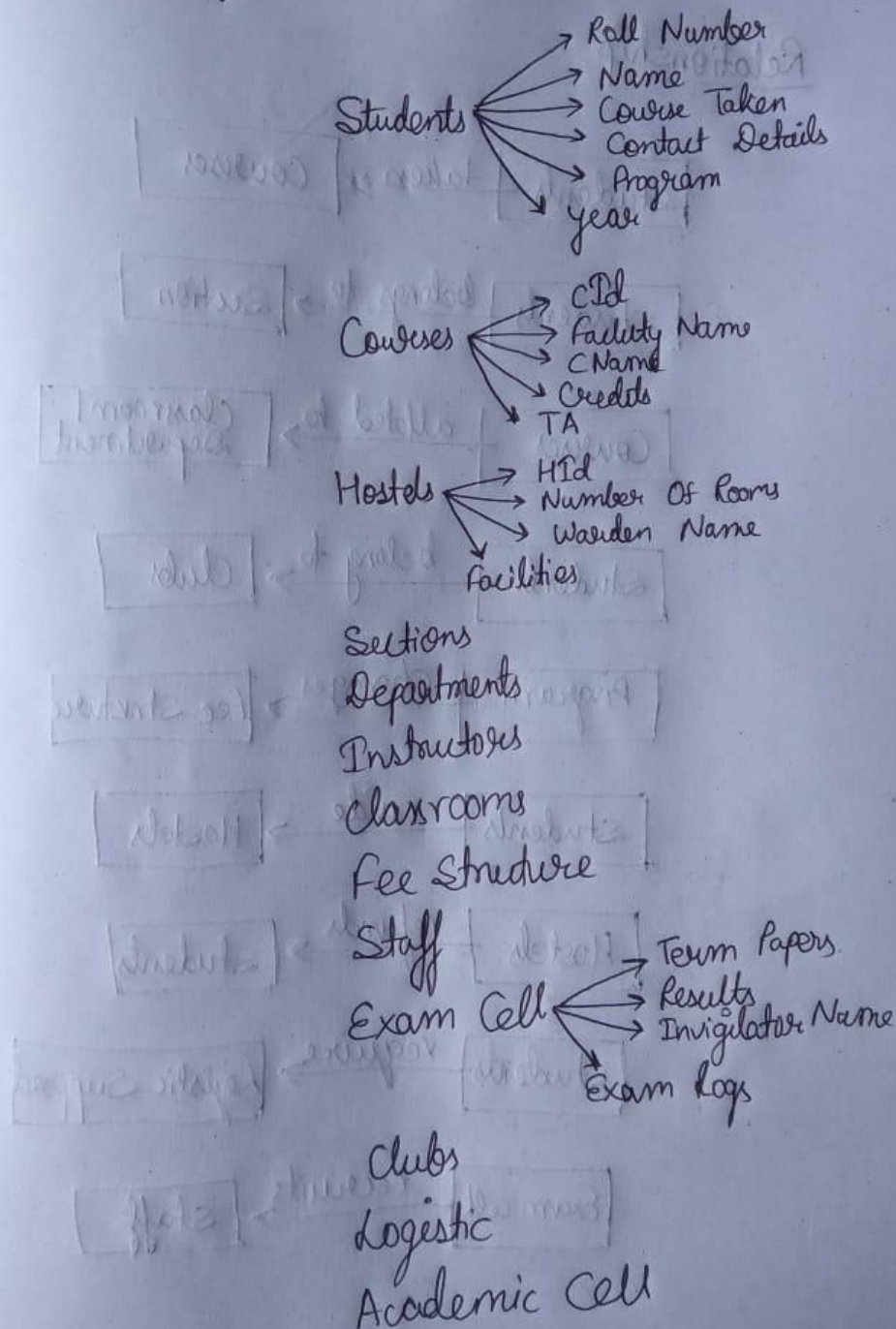
## **Drawbacks of using file systems to store data**

- Data redundancy and inconsistency: Data redundancy is a condition that occurs when the same piece of data exists in multiple places in the database whereas data inconsistency is a condition that occurs when the same data exists in different formats in multiple tables.
- Difficulty in accessing data
- Data isolation
- Integrity problems: Integrity constraints e.g. (account balance > 0)
- Atomicity of updates: It is one of the ACID (atomicity, consistency, isolation, durability). It means that any database transaction either happens as a whole or doesn't happen at all.
- Concurrent access by multiple users: Uncontrolled concurrent accesses can lead to inconsistencies. Examples: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time.
- Security problems

Examples of Entity and Relationship:

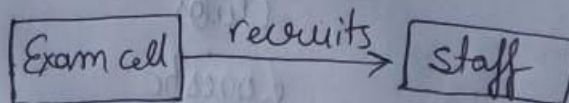
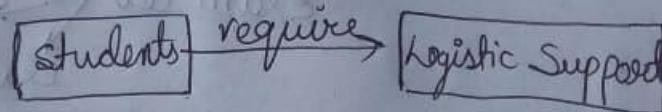
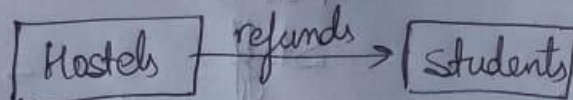
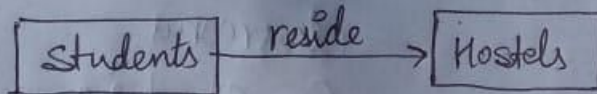
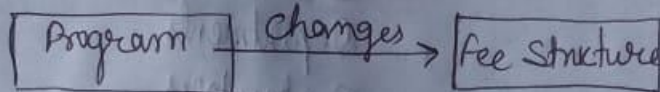
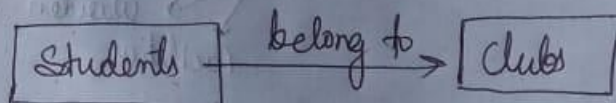
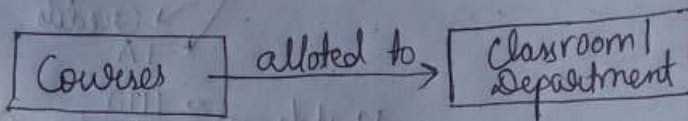
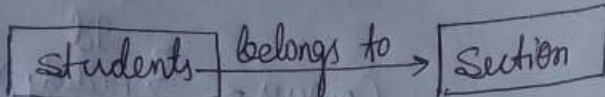
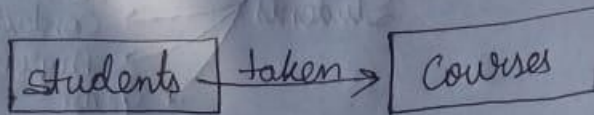
## University:

### University Database:-



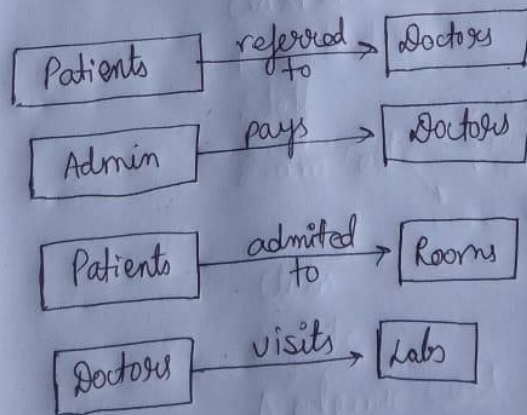
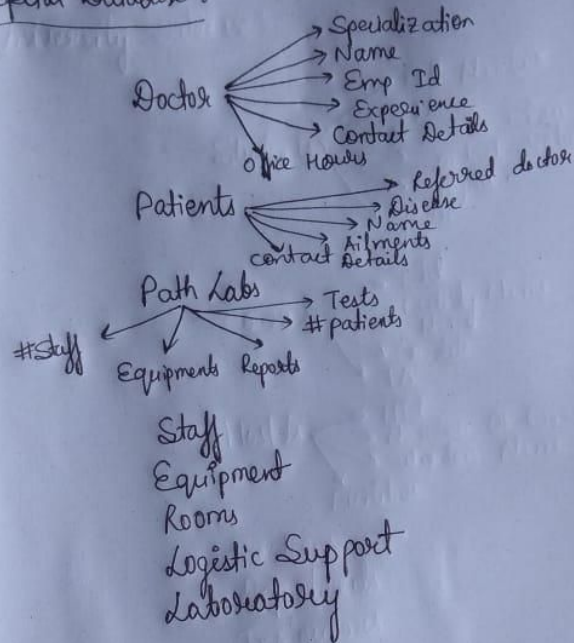


## Relationship



# Hospital

Hospital Database :-



## **Main characteristics of database approach:**

- Self-describing nature of a database system: Database system not only contains the database itself but also complete definition of the description of the database structure and constraints. This definition is stored in the database which contains the information of each file, the type and storage format of each data. The information stored in the database is called **meta data**.
- Insulation between programs and data: In traditional file processing any changes made to the file may lead to the change in the all application programs that accesses that file.
- Data Abstraction: A data model is used to hide storage details and present the users with a conceptual view of the database.
- Support of multiple views of the data: A database usually have many users and each user may require a different perspective 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.
- Sharing of data and multi-user transaction processing: Database must allow multiple users to access the database at the same time.

**Database Users:** In large organizations many people are involved in the design, use and manipulation of the database.

**Actors on the Scene:** People whose job involves day to day use of database.

- **Database administrators:** The DBA is responsible for authorizing access to the database, coordinating and monitoring its use, and acquiring software and hardware resources as required.
- **Database designers:** Database designers are responsible for identifying the data, choosing appropriate structure to represent and store this data. Database designers typically interact with each potential group of users and develop views of the database that meet the data and the processing requirements of these groups.
- **End-users:** These people jobs is to access the database and updating the reports. Some categories:
  - Casual end user occasionally accesses the database.
  - Naïve or parametric end users constantly query and update the database using standard type of queries and updates.
  - Sophisticated end users include engineers, scientists, business analyst and others who thoroughly familiarize themselves with the facilities of the database to implement their own application to meet their complex requirements.
  - Stand alone maintain personal databases by using ready-made program packages.
- **System Analysts and Application Developers:** They make specifications for varied users and implement this as a program.

**Actors behind the scene:** People who work to maintain the database environment.

- **System Designers and Implementers:** They implement the DBMS modules and interfaces as a software package.
- **Tool Developers:** They design and implement tools that is the software packages that facilitates database modeling and design, database system design, and improve performance.
- **Operators and Maintenance Personnel:** They are responsible for maintaining actual hardware and software environment of the database system.

### **How database is stored in mobile phones?**

SQLite is available on every android device. Using an SQLite database in Android does not require any database setup or administration. We only have to define SQL statements for creating and updating the database.

### **How social networking sites store their database?**

**Facebook:**

**MySQL:** wall posts, user information and timeline.

**Memcached:** A memory caching system to speed up dynamic database driven websites by caching data and objects in RAM.

**Haystack:** Implements HTTP based photo server which stores photos in a generic object store.

**Twitter:**

- Unique IDs for each tweet are generated by Snowflake.
- FlockDB is used for ID to ID mapping.
- Gizzard is Twitter's distributed data storage framework built on the top of MySQL.

**Instagram:** PostgreSQL and Cassandra

**Google:** Big Table

**Yahoo:** PostgreSQL

**P.T.O->**

## **2. Database System Concepts and Architecture**

### **Database Models**

Data models are a collection of concepts that can be used to describe the structure of a database.

The structures typically include:

#### **Elements and their data types:**

These specify the list of elements(fields) that will reside in the table and data types define the types of values that each field is allowed to take.

#### **Relationships:**

A relationship is where two or more tables are linked together because they contain related data. This enables users to run queries for related data across multiple tables.

#### **Constraints:**

Constraints are the rules enforced on the data columns of a table. These are used to limit the type of data that can go into a table. This ensures the accuracy, reliability and integrity of the data in the database.

Constraints are of 2 types:

#### **Domain constraints:**

Constraints on fields, restrictions on values that each field can have.

## **Referential Integrity Constraints:**

Constraints on Foreign Keys, CASCADE behaviour etc.

## **Operations:**

The models also include basic operations specifying insertions, retrivals, updates etc.

Some database designers may also choose to specify dynamic behaviour of database which contain user-defined operations.

These help in [data-abstraction](#), which is one of the fundamental characteristics of a database.

# **Categories of Data Models**

Data models can be typically divided into 4 categories

- **High Level (Conceptual) Data Models**
- **Low Level (Physical) Data Models**
- **Representational (Implementation) Data Models**
- **Self Describing Data Models**

## **High Level Models:**

These provide concepts that are close to the way most users perceive data.

These use concepts such as entities, attributes, relationships etc.

An entity represents a real-world object or concept.

An attribute represents some property of interest that further describes an entity.



A relationship among represents an association or dependencies among 2 or more entities.

These models typically contain information about **WHAT** the database contains. They are mostly used by business persons/ designers who have knowledge about the functional requirements of the project.

### **Low Level Models:**

These describe **HOW** the data is going to be stored on the file system. These are used by developers who have made the database software. These define, how the queries/commands fired by the user will interact with the database. They also deal with concepts like [access path](#) and [indexing](#) to make the database more efficient.

### **Representational Data Models:**

This models falls between the conceptual and physical category. They are used to provide concepts that can be well understood by the end user but that are not too far removed from the way data is organized in computer storage.

These are the ones that are most frequently used.

Representational data models hide many details of data storage on disk but can be implemented on a computer system directly.

These include the currently popular, Relational Data Model as well as old legacy data models like the network and hierarchical models.

### **Self Describing Data Models:**

The data storage in systems based on these models combines the description of the data with the data values themselves.

These have becoming increasingly popular after the data sharing formats like **JSON** and **XML** started becoming widely used.

Examples include NOSQL.

## **Schemas and Instances**

The description of a database is called the **database schema**, which is specified during database design stage. It contains the entities, data types, constraints etc. A database schema is not expected to change frequently. In fact, in some situations, it may become very difficult to alter the schema in post-production stage.

A schema is represented through a **schema diagram**

PTO->

Example of schema diagram:

**STUDENT**

Name	Student_number	Class	Major
------	----------------	-------	-------

**COURSE**

Course_name	Course_number	Credit_hours	Department
-------------	---------------	--------------	------------

**PREREQUISITE**

Course_number	Prerequisite_number
---------------	---------------------

**SECTION**

Section_identifier	Course_number	Semester	Year	Instructor
--------------------	---------------	----------	------	------------

**GRADE\_REPORT**

Student_number	Section_identifier	Grade
----------------	--------------------	-------

Each object in a schema is called a **schema construct**. (STUDENT, COURSE etc in above diagram)

A schema diagram displays only some aspects of a schema, such as the names of record types and data items, and some types of constraints. Other aspects (like relationships, data types) may or may not be displayed. These are used for a very high level view.

The data in the database at a particular moment in time is called a **database state** or **snapshot**. It is sometimes also called an **current set of occurrences** or an **instances**.

The DBMS is also partly responsible for ensuring that every state of the database is a **valid state**—that is, a state that satisfies the structure and constraints specified in the schema. Typically, ACID architecture is used to guarantee validity.

Sometimes the DBMS also stores the descriptions of the schema constructs and constraints called the **meta-data** in the DBMS catalog so that DBMS software can refer to the schema whenever it needs to.

If a change needs to be introduced in the schema, during the production, it is called as **schema evolution**. DBMS include various operations for schema evolution that can be applied while the db is operational so that changes can be applied without affecting the existing data.

The database schema changes infrequently whereas database state changes every time database is updated.

Schema is also called intension

State is also called extension

**PTO->**

# ACID

**ACID** is a set of properties of database transactions intended to guarantee validity even in the event of errors, power failures, etc.

A sequence of database operations that satisfies the ACID properties is called a transaction.

The properties are:

**A**tomicity

**C**onsistency

**I**solation

**D**urability

## **Atomicity:**

An atomic transaction is a series of database operations such that either *all* occur, or *nothing* occurs. A guarantee of atomicity prevents updates to the database occurring only partially, which can cause greater problems than rejecting the whole series outright.

## **Consistency:**

Consistency refers to the requirement that any given database transaction must change affected data only in allowed ways. Any data written to the database must be valid according to all defined rules, including constraints, [cascades](#), [triggers](#), etc. This guarantees that any programming errors in application level code cannot result in the violation of any defined database constraints.

**Isolation:**

Isolation determines how transaction integrity is visible to other users and systems. Isolation is typically defined at database level as a property that defines how/when the changes made by one operation become visible to other.

**Durability:**

Durability guarantees that transactions that have committed will survive permanently. Many DBMSs implement durability by writing transactions into a transaction log that can be reprocessed to recreate the system state right before any later failure.

**Methods to guarantee ACID properties:****Rollbacks:**

A rollback is an operation which returns the database to some previous state. They are typically related to **Atomicity**. Atomicity prevents updates to be committed in a partial state, so a rollback is performed when a transaction is unable to complete, preserving the atomicity. Rollbacks are also important for database integrity, because they mean that the database can be restored to a clean copy even after erroneous operations are performed.

The rollback feature is usually implemented with a [transaction log](#)

**Locks:**

**Locking** is the technique of preventing simultaneous access to data in a database. If a process tries to read a database entry while another process is writing to it, it may happen that old (incorrect) details will be read giving wrong results. Or in a worse scenario, if 2 processes attempt to write to a particular entry, it may lead to inconsistency. This is prevented by '**locking**' the record. When one process is performing an operation on a record,

other processes are not allowed to interact with it and must wait until previous operation is completed.

## **Three Schema Architecture**

Three-schema architecture, was proposed to help achieve and visualize the following characteristics:

- use of a catalog to store the database description (schema)
- insulation of programs and data
- support of multiple user views

Part of the usage of the three-schema architecture is to look at how the design maintenance differs from the core system maintenance. In this architecture, schemas are defined at the following three levels:

### **1. Internal Schema:**

Internal schema at the internal level to describe physical storage structures and access paths. The internal schema uses a physical data model.

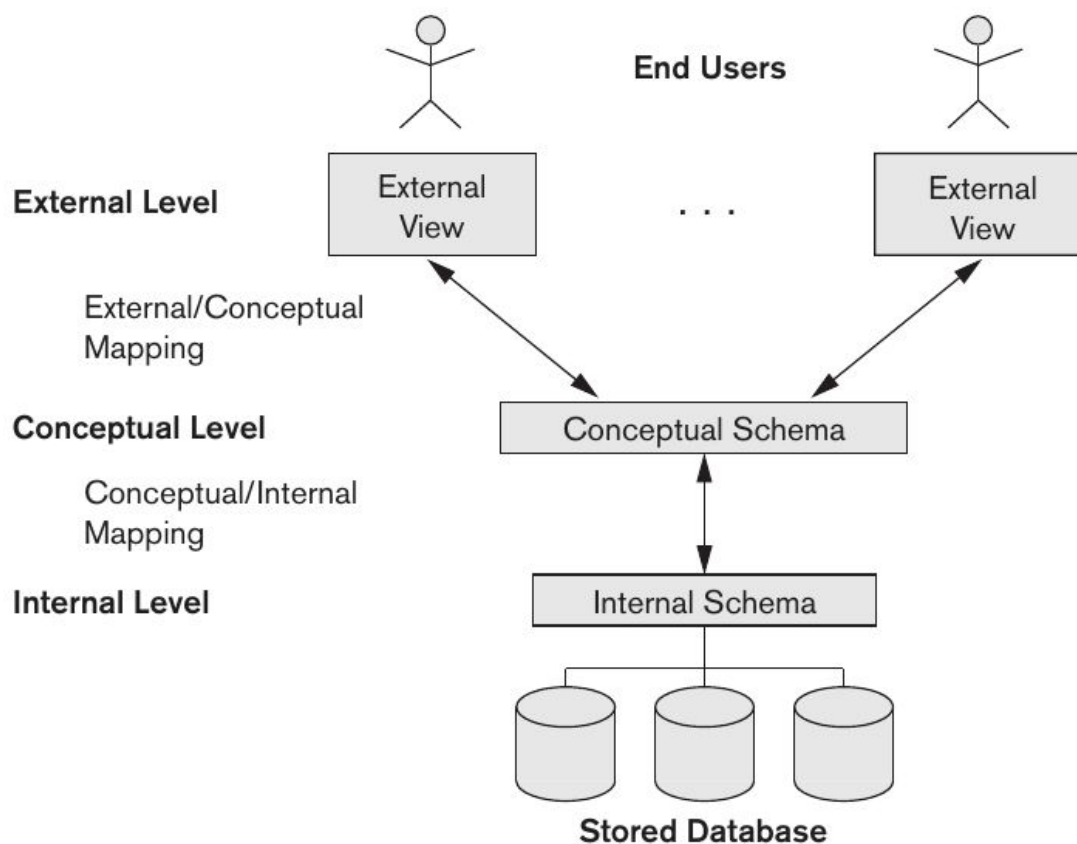
### **2. Conceptual Schema:**

The conceptual level has a conceptual schema, which describes the structure of the whole database for a community of users. It hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints. It's implementation is often based on a conceptual schema design in a high-level data model.

### 3. External Schema:

The view level includes a number of external schemas. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group.

A diagram showing 3 schema architecture:



In the three-schema architecture, each user group refers to its own schema. The DBMS transforms a request on an external schema into a request against the conceptual schema, and then into a request on the internal schema for processing over the stored database.



The processes of transforming requests and results between levels are called **mappings**.

Many DBMSs support the three-schema architecture only to some extent and do not separate the 3 levels completely. In many cases, the internal schema is handled by DBMS whereas the user specifies only conceptual or external schema and the task to map the conceptual/external schema to internal schema is left to the application program (example: django)

## **Data Independence**

**Data independence**, is defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level.

Using this , if a change is made in the internal schema, the application program can still work as before as they refer only to external schemas. Only the mappings between conceptual and internal schema will have to be changed.

There are 2 types of Database Independence:

### **Logical data independence:**

It is defined as the capacity to change the conceptual schema without having to change external schemas or application programs. Only mappings need to be changed. When the conceptual schema is modified, the application programs that refer to the external schema must still work as before.

Changes to constraints can be applied directly to the conceptual schema without affecting the external schemas or application programs.

**Physical data independence:**

It is the capacity to change the internal schema without having to change the conceptual schema. If the same data as before remains in the database, we do not need to change the conceptual schema. This is much easier to achieve than logical independence.

Usually physical data independence exists in most databases whereas logical independence is a bit more difficult to achieve and may be implemented only to an extent.

## **Database Languages**

A DBMS provides appropriate languages and interfaces for each category of users.

Every DBMS supports the following languages

**DDL (Data Definition Language):**

This is the language mostly used by Data Administrators. It is used to define schemas (both internal and conceptual). All DBMS have a DDL compiler which processes the DDL statements, makes the schema constructs and stores them in the DBMS catalog. The DDL generates a set of table templates that are stored in a data dictionary. The data dictionary typically contains schema, constraints and authorizations.

Set of operations on DDL:

- CREATE
- ALTER
- DROP

- TRUNCATE
- RENAME
- COMMENT

### **SDL( Storage Definition Language):**

This language is used to specify the internal schema. Some Databases choose to define internal schemas with SDL and conceptual with the DDL. However most modern databases define both internal and conceptual with DDL and SDL is not used.

In most modern relational DBMSs, there is no specific language that performs the role of SDL. Instead, the internal schema is specified by a combination of functions, parameters, and specifications related to storage of files. These permit the DBA staff to control indexing choices and mapping of data to storage.

### **View Definition Language (VDL):**

This is used to specify user views and their mappings to the conceptual schema. In most relational databases, SQL provides the role of VDL.

### **Data Manipulation Language (DML):**

This is used to perform typical manipulations to data including retrieval, insertion, deletion, and modification of the data, etc. Also known as query language. There are mainly 2 types of DML.

1. **High Level (non-procedural) (set-at-a-time) DML** : It specifies complex database operations concisely. Most DBMSs allow high-level DML statements either to be entered interactively from a

display monitor or terminal or to be embedded in a general-purpose programming language. It is mainly concerned with WHAT things are to be done and not HOW they are done.

- 2. Low level (procedural) (record-at-a-time) DML :** It must be embedded in a general-purpose programming language. This type of DML typically retrieves individual records or objects from the database and processes each separately. They usually process only 1 record at a time. These are concerned with both WHAT and HOW things are done.

**Set of operations on DML:**

- INSERT
- DELETE
- MODIFY
- UPDATE
- CALL (for procedural calls)
- LOCK TABLE

**Data Control Language (DCL):**

This is used to provide authorization to the database thereby giving only controlled access to the db. This makes sure that only an authorized person can make major changes to the database.

**Set of commands / operations:**

- GRANT
- REVOKE

**Transaction Control Language:**

Mainly focuses on maintaining ACID properties.

### Commands/ Operations:

- COMMIT
- SAVE POINT
- ROLLBACK

## DBMS Components

### 1. Stored Data Manager:

DBMS and its catalog is stored on a disk. Access to the disk is controlled by the Operating System. A high level Stored Data manager or a storage manager controls access to DBMS information that is stored on disk, whether it is part of the database or the catalog.

Tasks of storage manager:

- Providing interaction with OS for read/write mechanisms
- Organizing file structure
- Managing access to tables
- Providing efficient ways store, retrieve, update the data
- Indexing and hashing the tables to provide faster access

Some storage managers include another component **buffer management module** to schedule disk read/write

**P.T.O->**

## **2. Query Processing:**

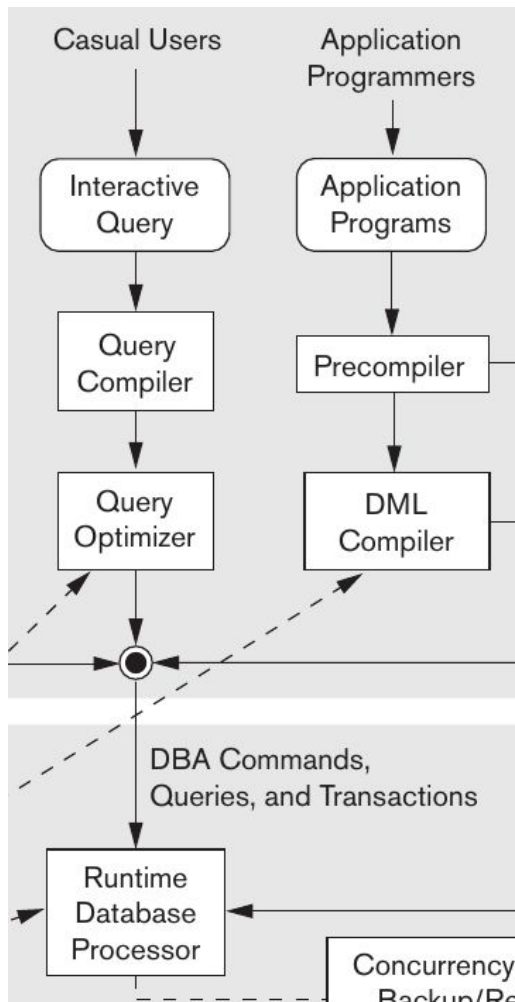
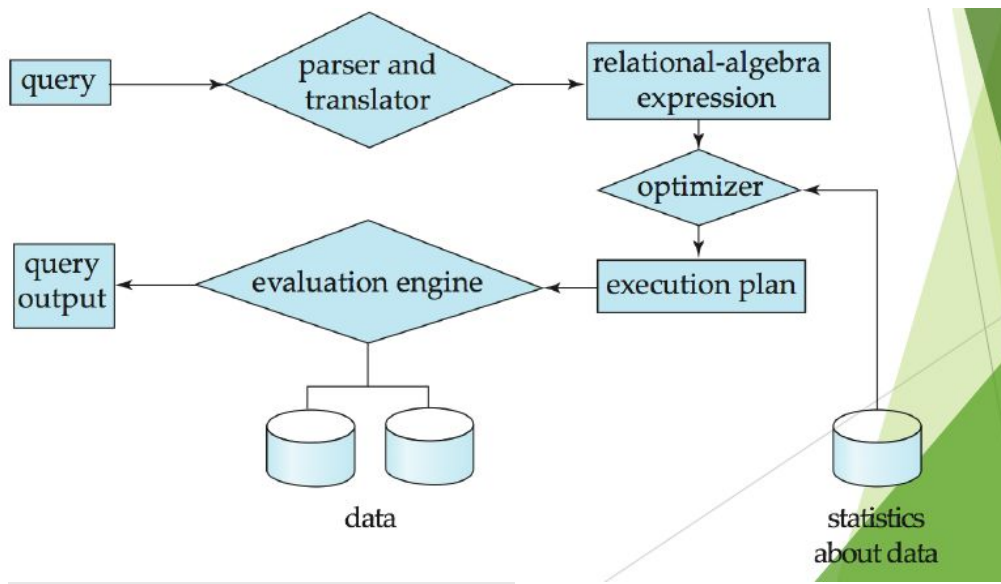
This module deals with taking user queries and operate on tables.

This is mainly meant for casual users and application programs which make use of interactive queries.

The main components are:

- Parsing and Translation
- Optimization
- Evaluation

**PTO->**



The queries inserted by casual users and application programs go to a query compiler.

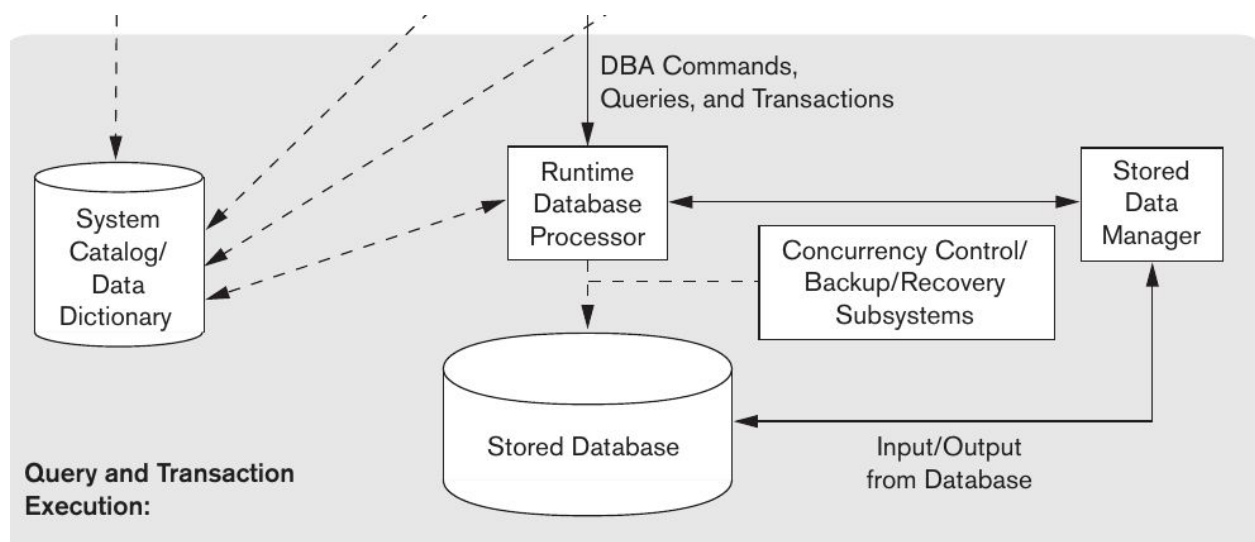
The query compiler compiles them into an internal form. This internal query is subjected to query optimization.

the query optimizer is concerned with the rearrangement and possible reordering of operations, elimination of redundancies, and use of efficient search algorithms during execution.

It consults the system catalog for statistical and other physical information about the stored data and generates executable code that performs the necessary operations for the query and makes calls on the runtime processor.

Application programs are subject to a pre-compiler. The precompiler extracts DML commands from an application program written in a host programming language. These commands are sent to the DML compiler for compilation into object code for database access. The rest of the program is sent to the host language compiler.

The runtime database processor executes the privileged commands. It works with system catalog and stored data manager. The runtime database processor handles other aspects of data transfer, such as management of buffers in the main memory.



## Transaction Management

A transaction is a collection of operations that performs a single logical function in a database application. A transaction is an atomic unit.

Transaction Management deals with preserving the [ACID](#) properties even in worst scenarios like power failure, system failure, etc.



It has mainly 3 parts:

- Transaction management component

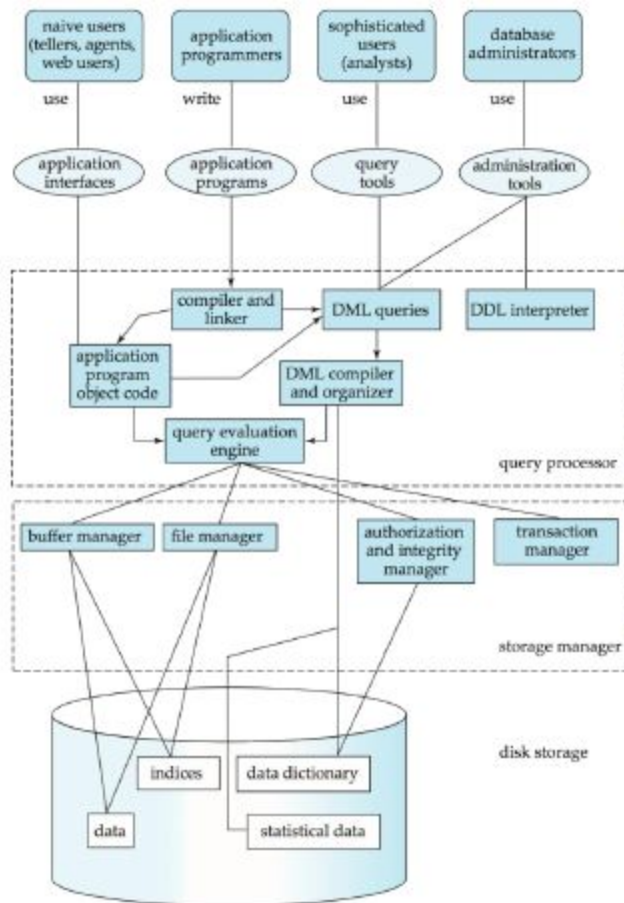
It ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures. This part is mainly responsible for running TCL operations like COMMIT, SAVE POINT, ROLLBACK etc

- Concurrency Control Manager

It ensures that data is consistent when multiple users/processes are operating on the same row/table of data at the same time. It mainly deals with locking of tables/rows to prevent any 2 processes from accessing data at a single moment.

**PTO->**

# Database users and Architecture



Different users use different ways to access the database. Mostly administrators are the only ones who use DDL queries, that don't go via query evaluation engine. Sophisticated and administrator users use interactive queries to operate on database, whereas the other 2 category of users will mostly use application interfaces.

## Different Tiers of database architecture

- 1 tier
- 2 tier
- 3 or n tier architecture

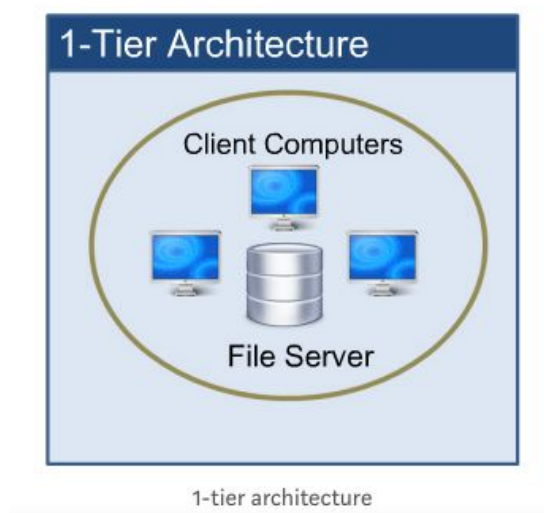
## Single Tier Architecture:

This is the simplest form of Database Architecture. Here all the Client, Server, and Database all reside on the same machine.

An example would be a localhost server running on a machine

They are useful for storing some information locally but in an organized way.

These are rarely used in a production



## 2 Tier Architecture

The two-tier is based on Client Server architecture. The direct communication takes place between client and server. There is no intermediate between client and server. The user interface programs and application programs can run on the client side. When DBMS access is required, the program establishes a connection to the DBMS (which is on the server side); once the connection is created, the client program can communicate with the DBMS.

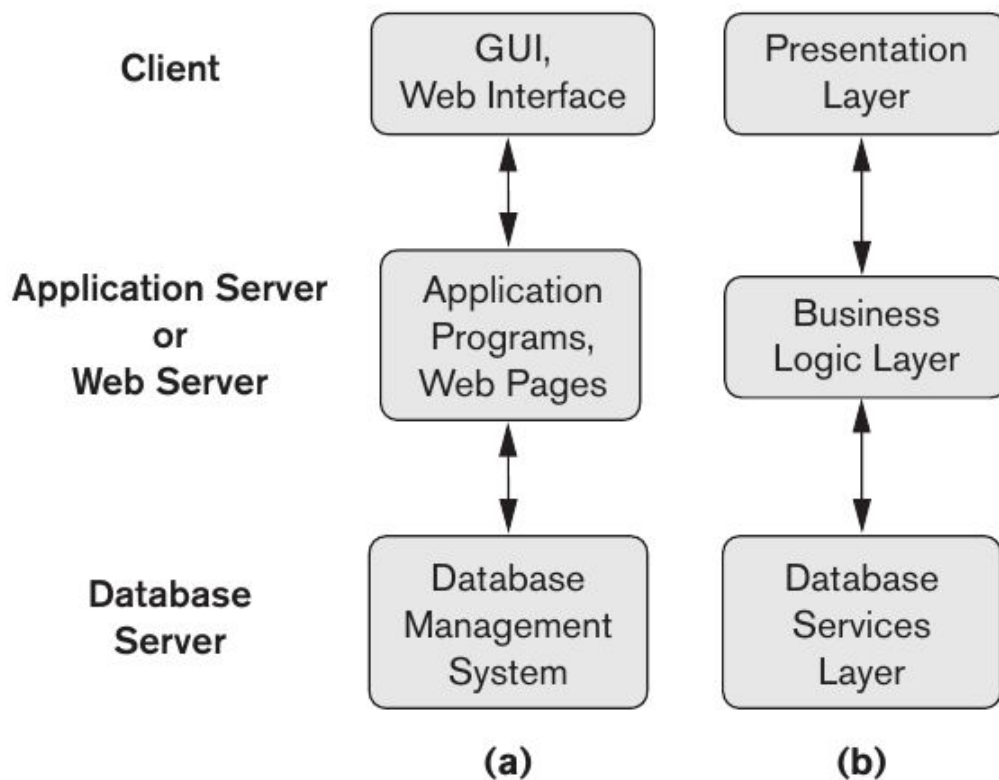
**Open Database Connectivity (ODBC)** provides an API using which client side interacts with server side.

Advantages:

1. Simplicity
2. Seamless compatibility with existing systems

However these are very rarely used now a days. 3 tier architecture is becoming more popular for web applications.

## Three Tier Architecture



The client layer interacts with an application server/ web server etc. The web server then connects to the database and performs operations based on the requirements of the client . This helps in separating the application logic from the database logic.

The bottom layer includes all data management services. The middle layer can also act as a Web server, which retrieves query results from the database server and formats them into dynamic.

Web pages that are viewed by the Web browser at the client side. The client machine is typically a PC or mobile device connected to the Web.

### **Advantages:**

1. It gives the ability to update the technology stack of one tier, without impacting other areas of the application.
2. It allows for different development teams to each work on their own areas of expertise. Developers are more likely to have deep competency in one area, like coding the front end of an application, instead of working on the full stack.
3. Easier to scale the application up and out.
4. Adds reliability and more independence of the underlying servers or services.

### **N tier architecture**

N tier architecture involves dividing any one tier from the 3 tier architecture into multiple tiers. Usually this is done to the database layer. In large systems, data is usually split into several databases and

another layer is required which looks where the required data is located and then guides the application layer to it. Sometimes the business logic layer is divided into multiple layers. This is widely used in the field of distributed computing.

## **Classification of Database management Systems**

Database can be classified based on several criterias:

1. The data model that is used
2. Number of users
3. Data distribution

**Based on the data model:**

- 1. Relational Data Model**
- 2. Object Data Model**
- 3. Big Data Systems**
- 4. Hierarchical model**
- 5. Network data model**

The 4th and 5th are old legacy models that are no longer used today.

**PTO->**

## **Relational Data Model**

The relational data model represents a database as a collection of tables, where each table can be stored as a separate file.

Most relational databases use the high-level query language called SQL and support a limited form of user views.

The tables in the database are connected to each other via 'relations'

There are typically 3 types of relations:

### **1. One to One Relation (1:1)**

Ex: A user table connected to profile table

### **2. One to Many (or Many to One) relation (1:N)**

Ex: A company table connected to many items that it manufactures

### **3. Many to Many relation (N:N)**

Ex: A teacher table connected to a student table

## **Object Data Model:**

The object data model defines a database in terms of objects, their properties, and their operations.

Objects with the same structure and behavior belong to a class, and classes are organized into hierarchies (or acyclic graphs).

The operations of each class are specified in terms of predefined procedures called methods.

Many databases follow a hybrid between relational and object model. They are called **object-relational** or **extended relational** systems

## **Big data systems:**

These systems include various data models within them namely:

**Key-value data model:** associates a unique key with each value (which can be a record or object) and provides very fast access to a value given its key.

**Document Data Model:** It is based on JSON. Data is stored in the form of documents.

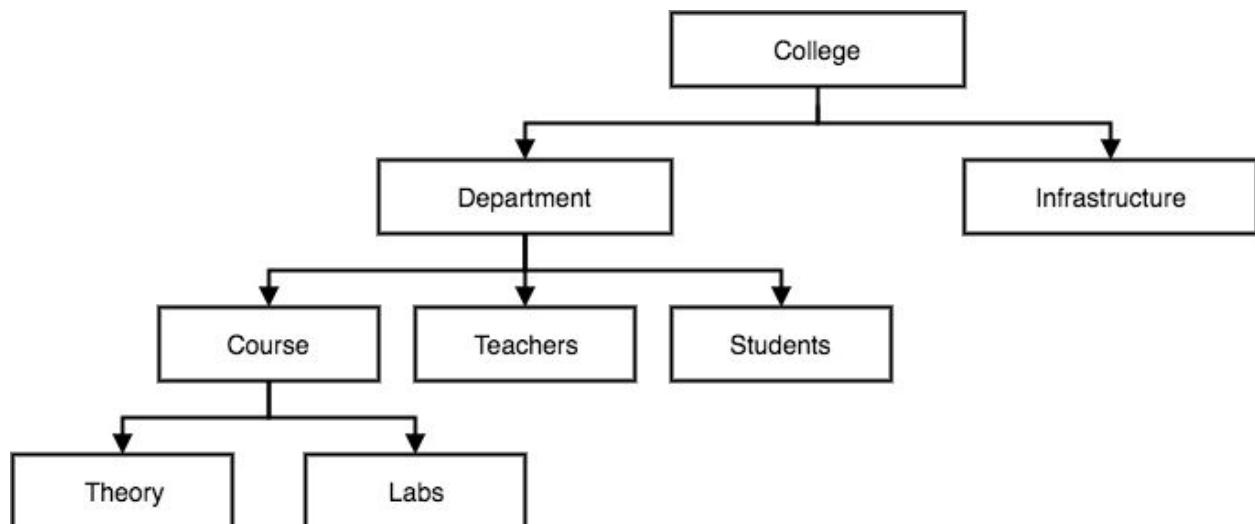


**Graph data model:** stores objects as graph nodes and relationships among objects as directed graph edges.

**Column based data model:** it stores the columns of rows clustered on disk pages for fast access and allow multiple versions of the data.

### **Hierarchical Model:**

Data is organized into tree like structure with each record having one parent record and many children.

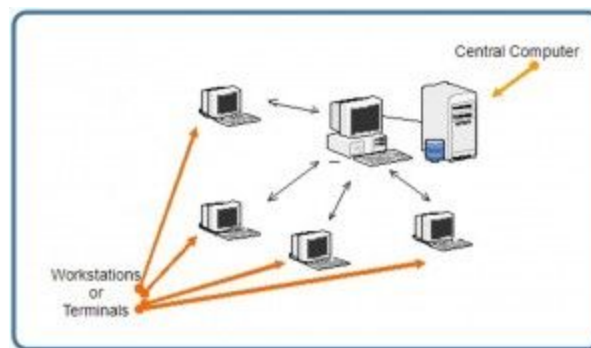


### **Classification based on Distribution of data:**

1. Centralized
2. Client-server
3. Parallel (multi-processor)
4. Distributed

## Centralized database:

With a centralized database system, the DBMS and database are stored at a single site that is used by several other systems



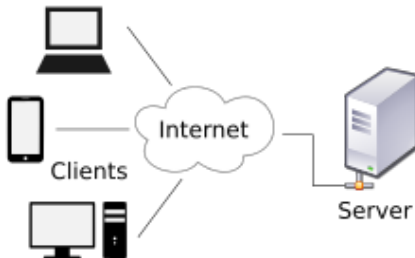
## Client-Server Database:

Client–server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients.

PTO->

A server host runs one or more server programs which share their resources with clients.

A client does not share any of its resources, but requests a server's content or service function.

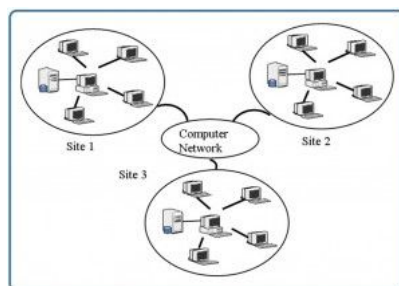


### **Parallel Database:**

Parallel databases improve processing and input/output speeds by using multiple CPUs and disks in parallel. In parallel processing, many operations are performed simultaneously, as opposed to serial processing, in which the computational steps are performed sequentially.

### **Distributed Database:**

In a distributed database system, the actual database and the DBMS software are distributed from various sites that are connected by a computer network,



# Entity Relationship Models

**Entity:** An entity is a thing or object in the real world with an independent existence. Its existence might be either physical or conceptual.

## ER Diagrams:

It is the design or blueprint of a database.

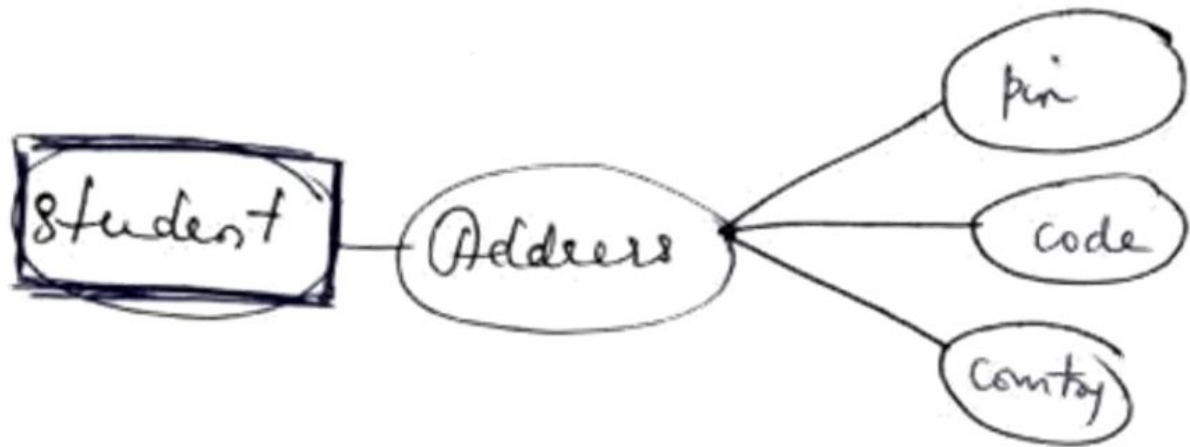
Components of ER diagram:

- **Entity set**
- **Attribute set**
- **Relationship set**

Types of **Attributes**:

- Single valued: Attributes that can have single value at a particular instance of time.
- Multi valued: Attributes that can have more than one value at a particular instance of time. Eg., phone numbers of a person, qualification of a person.

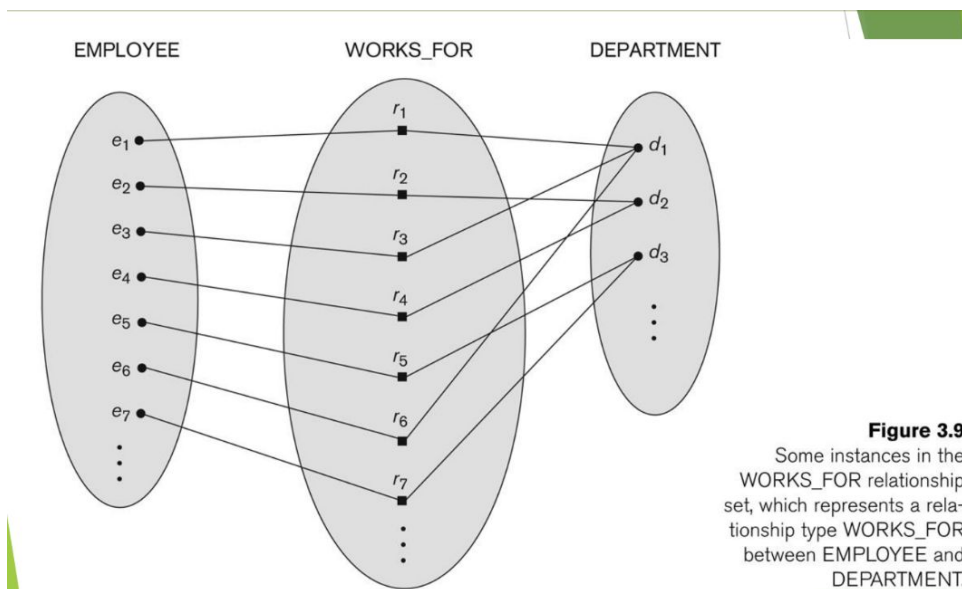
- Composite: It is combination of more than one attribute.  
Example:



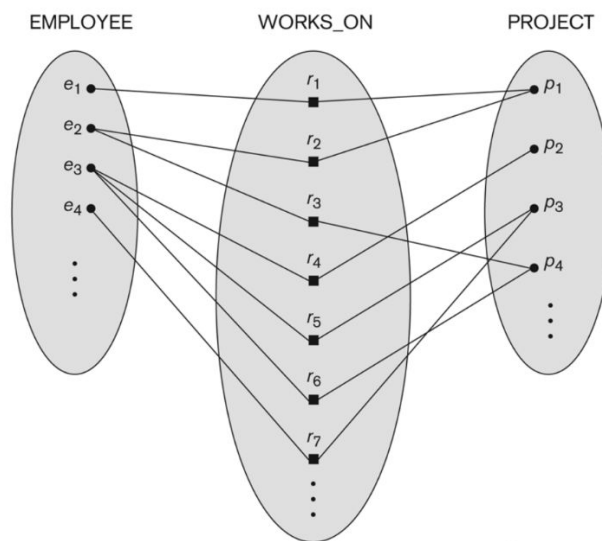
- Derived Attributes: Its value is dynamic and is derived from another attributes. Example: age of a person from DOB.

## Relationship Types:

### Many-to-one(N:1) relationship:



## Many-to-many(M:N) relationship:



**Figure 3.13**  
An M:N relationship,  
WORKS\_ON.

**Cardinality:** Cardinality defines the number of attributes in one entity set which can be associated with the number of attributes of other set via relationship set. They can be one-to-one, one-to-many, many-to-one, many-to-many.












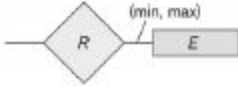
E.g. – (1:n) one-to-many means one instance of entity A is related to n instance of entity B.

**PTO->**



**PTO->**

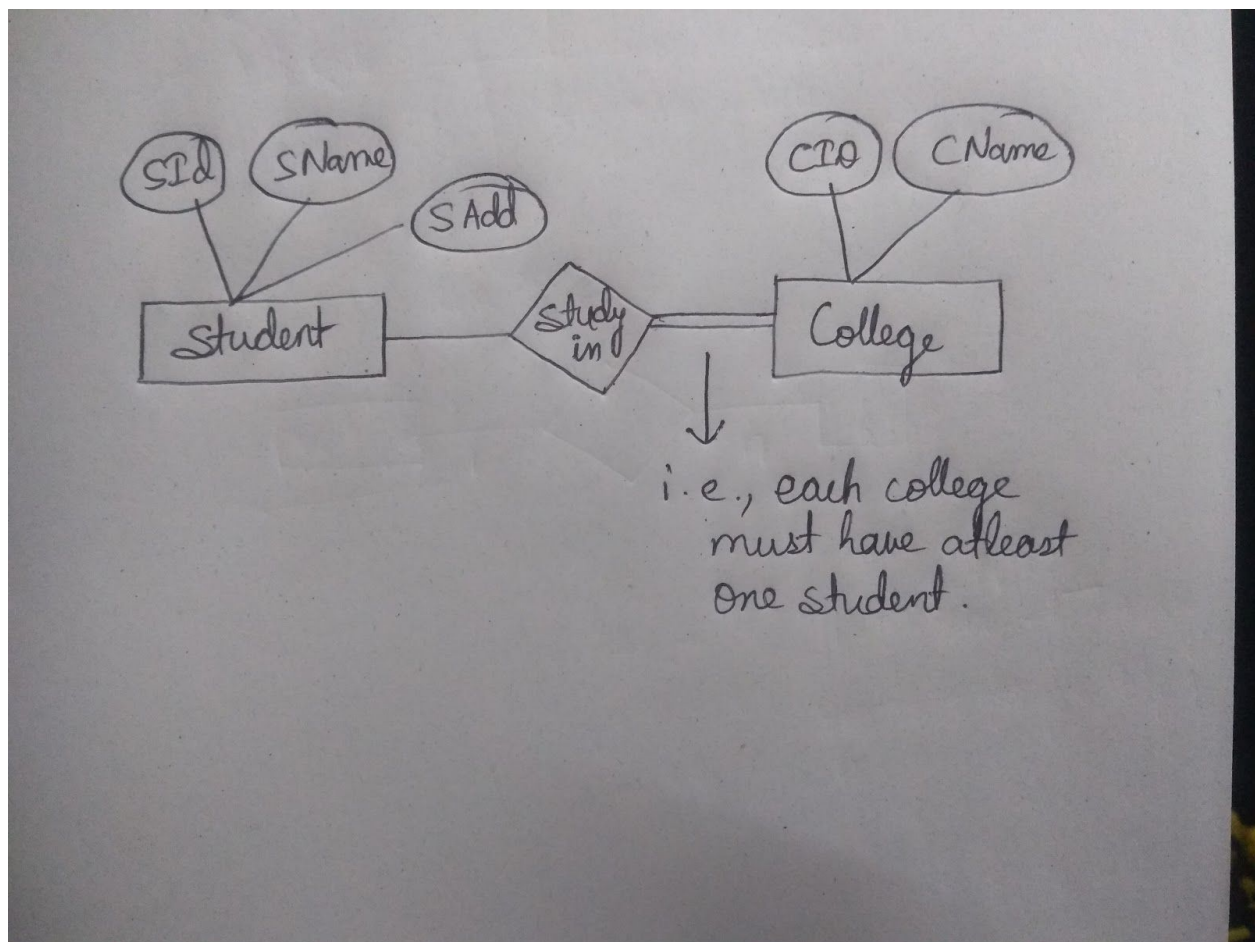
## Common symbols used in ER diagram:

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute
	Total Participation of $E_2$ in $R$
	Cardinality Ratio 1: N for $E_1:E_2$ in $R$
	Structural Constraint (min, max) on Participation of $E$ in $R$



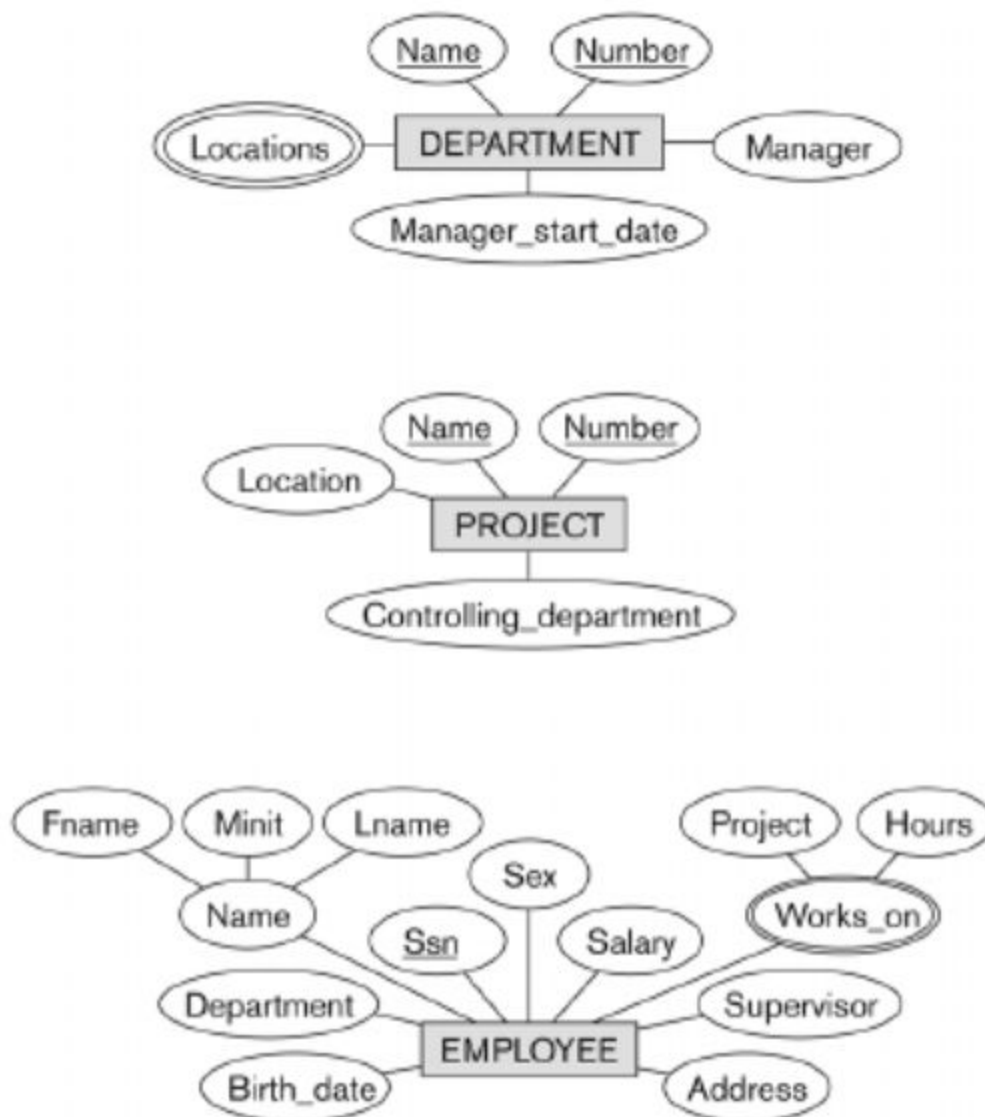
## Total participation of an entity set:

Represents each entity in an entity set must have at least one relationship in relationship set.

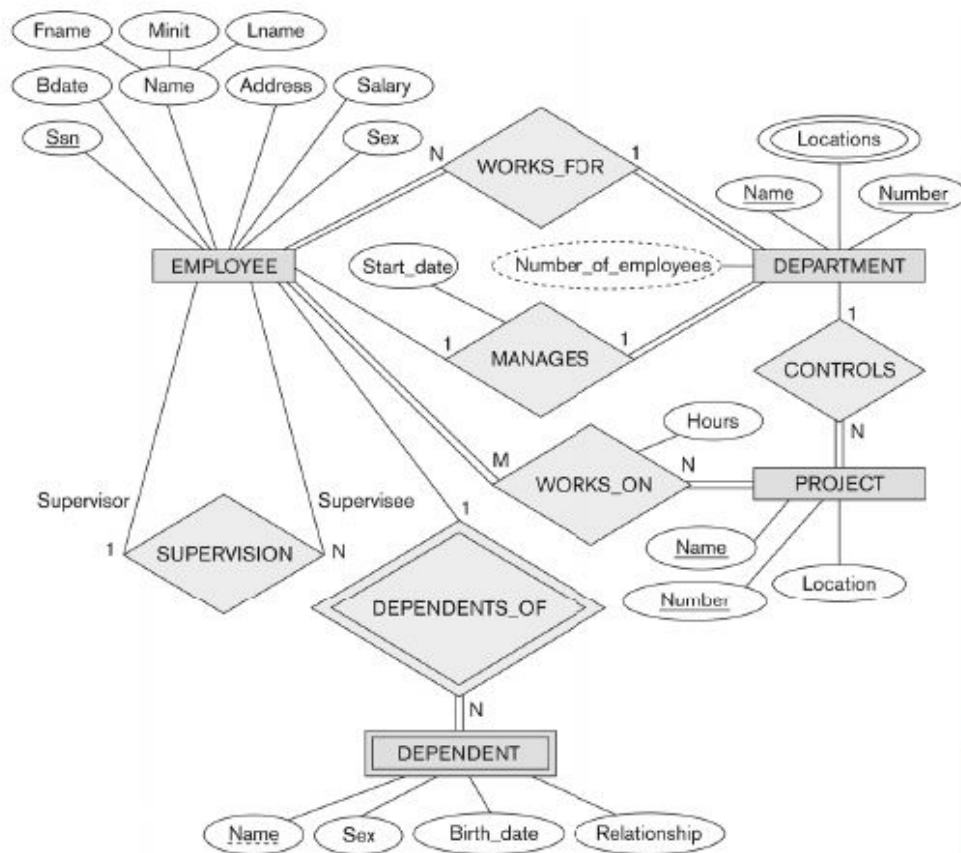


Example:

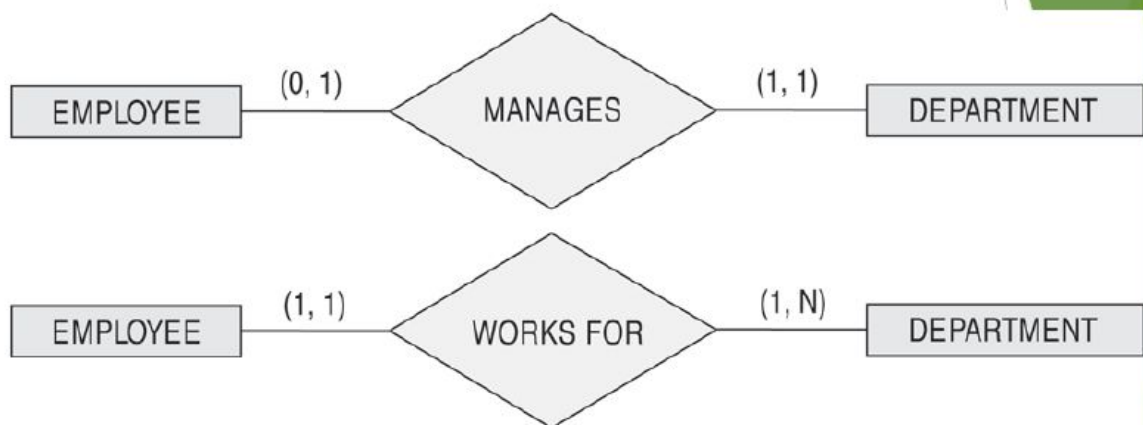
## Entities and attributes ER diagram



Full fledged diagram for the database schema using  
(min, max) notation.



## **(min, max) notation for relationship constraints:**



In first image one employee can be a manager of at most one department while in second image one employee can be a head of more than one department.

## **When not to use database:**

- When no DBMS may suffice
- Main inhibitors (costs) of using a DBMS:
- When a DBMS may be unnecessary
- When a DBMS may be infeasible

# Terminology

**Data-abstraction:** Data abstraction generally refers to the suppression of details of data organization and storage, and the highlighting of the essential features for an improved understanding of data.

**Access Path:** Access Path is a search structure that makes the search for particular database records efficient.

**Index:** An index is an example of an access path that allows direct access to data using an index term or a keyword. It may be organized in a linear, hierarchical (tree-structured), or some other fashion.

**Cascades:** Cascades in relational database model. It is used to recursively delete all the records in the hierarchy should one be deleted. For example a person's details may be spread across 3 tables. Cascades can be used to delete all the records of the person in each table should one of them be deleted.

**Triggers:** A database trigger is procedural code that is automatically executed in response to certain events on a particular table or view in a database. The trigger is mostly used for maintaining the integrity of the information on the database. Triggers can also be used to log historical data.

**Transaction Logs:** A transaction log is a history of actions executed by DBMS used to guarantee [ACID](#) properties over crashes or hardware failures. If, after a start, the database is found in an inconsistent state or not been shut down properly, the database management system reviews the database logs for uncommitted transactions and rolls back the changes made by these transactions.

## Sources

### Books:

1. Fundamentals of Database Systems (7th edition) by Elmasri Navathe
2. Database System Concepts (6th edition) by Abraham Silberschatz, Henry F. Korth, S. Sudarshan
3. Database Design , 2nd Edition by Adrienne Watt and Watt, Adrienne

### Web:

1. [www.opentextbc.ca](http://www.opentextbc.ca)
2. [www.techspirited.com](http://www.techspirited.com)
3. [www.access.redhat.com](http://www.access.redhat.com)
4. [www.en.wikipedia.com](http://www.en.wikipedia.com)
5. [www.geeksforgeeks.com](http://www.geeksforgeeks.com)