COM 214 DATABASE SYSTEMS

Introduction:

In computerized information system data is the basic resource of the organization. So, proper organization and management for data is required fro organization to run smoothly. Database management system deals the knowledge of how data stored and managed on a computerized information system. In any organization, it requires accurate and reliable data for better decision making, ensuring privacy of data and controlling data efficiently.The examples include deposit and/or withdrawal from a bank,hotel,airline or railway reservation, purchase items from supermarkets in all cases, a database is accessed.

What is data:

Data is the known facts or figures that have implicit meaning. It can also be defined as it is the representation of facts ,concepts or instruction in a formal manner, which is suitable for understanding and processing. Data can be represented in alphabets(A-Z, a-z),in digits(0-9) and using special characters(+,-.#,$, etc) e.g: 25, “ajit” etc.

Information:

Information is the processed data on which decisions and actions are based. Information

can be defined as the organized and classified data to provide meaningful values. Eg: “The age of Ravi is 25”

File:

File is a collection of related data stored in secondary memory.

File Oriented approach:

The traditional file oriented approach to information processing has for each application a separate master file and its own set of personal file. In file oriented approach the program

dependent on the files and files become dependent on the files and files become dependents upon the programs.

Disadvantages of file oriented approach:

1) Data redundancy and inconsistency:

The same information may be written in several files. This redundancy leads to higher storage and access cost. It may lead data inconsistency that is the various copies of the same data may longer agree for example a changed customer address may be reflected in single file but not else where in the system.

2) Difficulty in accessing data :

The conventional file processing system do not allow data to retrieved in a convenient and efficient manner according to user choice.

3) Data isolation :

Because data are scattered in various file and files may be in different formats with new application programs to retrieve the appropriate data is difficult.

4) Integrity Problems:

Developers enforce data validation in the system by adding appropriate code in the various application program. How ever when new constraints are added, it is difficult to change the programs to enforce them.

5) Atomicity:

It is difficult to ensure atomicity in a file processing system when transaction failure occurs due to power failure, networking problems etc. (atomicity: either all operations of the transaction are reflected properly in the database or non are)

6) Concurrent access:

In the file processing system it is not possible to access a same file for transaction at same time

7) Security problems:

There is no security provided in file processing system to secure the data from unauthorized user access

Database:

A database is organized collection of related data of an organization stored in formatted way which is shared by multiple users.

The main feature of data in a database are:

1. It must be well organized

2. it is related

3. It is accessible in a logical order without any difficulty

4. It is stored only once

for example: consider the roll no, name, address of a student stored in a student file. It is collection of related data with an implicit meaning.

Data in the database may be persistent, integrated and shared.

Persistent:

If data is removed from database due to some explicit request from user to remove.

Integrated:

A database can be a collection of data from different files and when any redundancy among those files are removed from database is said to be integrated data.

Sharing Data:

The data stored in the database can be shared by multiple users simultaneously with out affecting the correctness of data.

Why Database:

In order to overcome the limitation of a file system, a new approach was required.

Hence a database approach emerged. A database is a persistent collection of logically related data. The initial attempts were to provide a centralized collection of data. A database has a self describing nature. It contains not only the data sharing and integration of data of an organization in a single database. A small database can be handled manually but for a large database and having multiple users it is difficult to maintain it, In that case a computerized database is useful.

The advantages of database system over traditional, paper based methods of record

keeping are:

 compactness:

No need for large amount of paper files

 speed:

The machine can retrieve and modify the data more faster way then human being

 Less drudgery: Much of the maintenance of files by hand is eliminated

 Accuracy: Accurate,up-to-date information is fetched as per requirement of the

user at any time.

Database Management System (DBMS):

A database management system consists of collection of related data and refers to a set of

programs for defining, creation, maintenance and manipulation of a database.

Function of DBMS:

1. **Defining database schema**: it must give facility for defining the database structure also specifies access rights to authorized users.

2. **Manipulation of the database:** The dbms must have functions like insertion of record into database updation of data, deletion of data, retrieval of data

3. **Sharing of database:**The DBMS must share data items for multiple users by maintaining consistency of data.

4. **Protection of database:** It must protect the database against unauthorized users.

5. **Database recovery:** If for any reason the system fails DBMS must facilitate data base recovery.

Advantages of dbms:

Reduction of redundancies:

Centralized control of data by the DBA avoids unnecessary duplication of data and effectively reduces the total amount of data storage required avoiding duplication in the elimination of the inconsistencies that tend to be present in redundant data files.

Sharing of data:

A database allows the sharing of data under its control by any number of application programs or users.

Data Integrity:

Data integrity means that the data contained in the database is both accurate and consistent. Therefore data values being entered for storage could be checked to ensure that they fall with in a specified range and are of the correct format.

Data Security:

The DBA who has the ultimate responsibility for the data in the dbms can ensure that proper access procedures are followed including proper authentication schemas for access to the DBS and additional check before permitting access to sensitive data.

Conflict resolution:

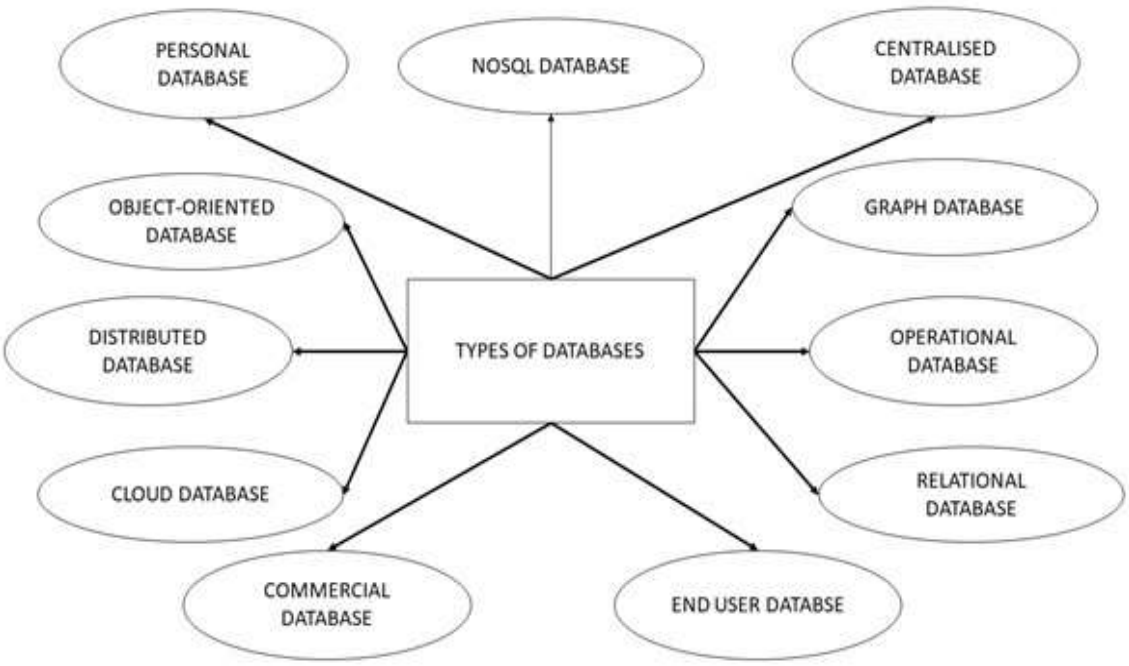
DBA resolve the conflict on requirements of various user and applications. The DBA chooses the best file structure and access method to get optional performance for the application.

Data Independence:

Types of databases.

Depending upon the usage requirements, there are following types of databases available in the market:

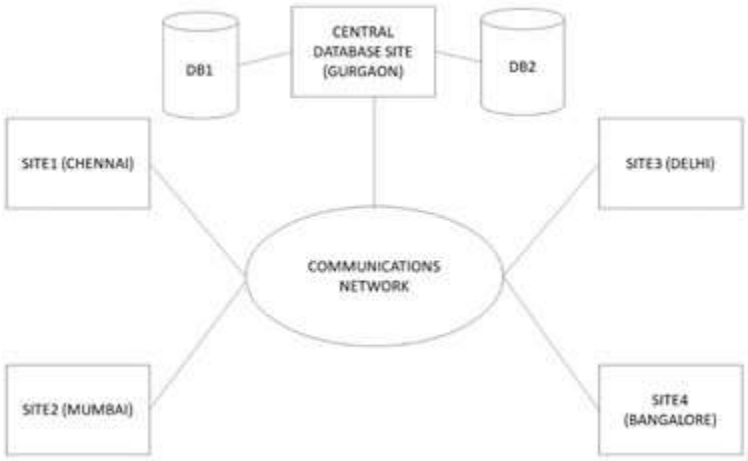
1. Centralised database.
2. Distributed database.
3. Personal database.
4. End-user database.
5. Commercial database.
6. NoSQL database.
7. Operational database.
8. Relational database.
9. Cloud database.
10. Object-oriented database.
11. Graph database.



1. Centralised Database

The information(data) is stored at a centralized location and the users from different locations can access this data. This type of database contains application procedures that help the users to access the data even from a remote location.

Various kinds of authentication procedures are applied for the verification and validation of end users, likewise, a registration number is provided by the application procedures which keeps a track and record of data usage. The local area office handles this thing.

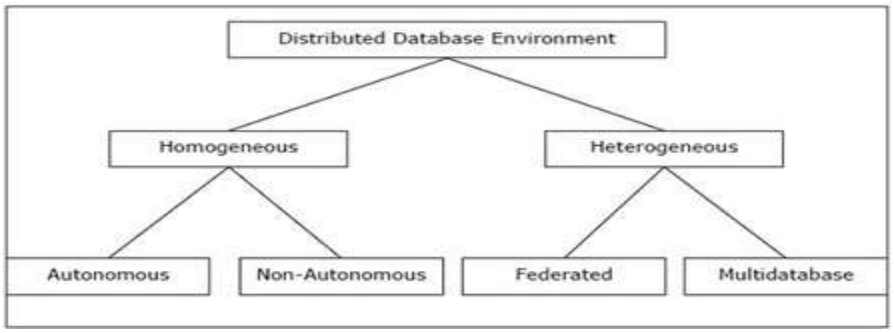


2.Distributed Database

Just opposite of the centralized database concept, the distributed database has contributions from the common database as well as the information captured by local computers also. The data is not at one place and is distributed at various sites of an organization. These sites are connected to each other with the help of communication links which helps them to access the distributed data easily.

You can imagine a distributed database as a one in which various portions of a database are stored in multiple different locations(physical) along with the application procedures which are replicated and distributed among various points in a network.

There are two kinds of distributed database, viz. homogenous and heterogeneous. The databases which have same underlying hardware and run over same operating systems and application procedures are known as homogeneous DDB, for eg. All physical locations in a DDB. Whereas, the operating systems, underlying hardware as well as application procedures can be different at various sites of a DDB which is known as heterogeneous DDB.



3.Personal Database

Data is collected and stored on personal computers which is small and easily manageable. The data is generally used by the same department of an organization and is accessed by a small group of people.

4.End User Database

The end user is usually not concerned about the transaction or operations done at various levels and is only aware of the product which may be a software or an application. Therefore, this is a shared database which is specifically designed for the end user, just like different levels' managers. Summary of whole information is collected in this database.

5.Commercial Database

These are the paid versions of the huge databases designed uniquely for the users who want to access the information for help. These databases are subject specific, and one cannot afford to maintain such a huge information. Access to such databases is provided through commercial links.

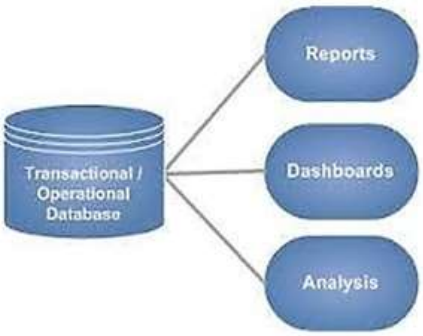
6.NoSQL Database

These are used for large sets of distributed data. There are some big data performance issues which are effectively handled by relational databases, such kind of issues are easily managed by NoSQL databases. There are very efficient in analyzing large size unstructured data that may be stored at multiple virtual servers of the cloud.

7.Operational Database

Information related to operations of an enterprise is stored inside this database. Functional lines like marketing, employee relations, customer service etc. require such kind of databases.

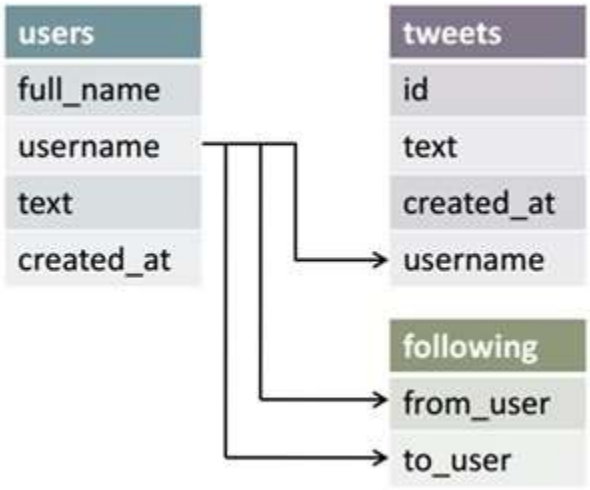
-



8.Relational Databases

These databases are categorized by a set of tables where data gets fit into a pre-defined category. The table consists of rows and columns where the column has an entry for data for a specific category and rows contains instance for that data defined according to the category. The Structured Query Language (SQL) is the standard user and application program interface for a relational database.

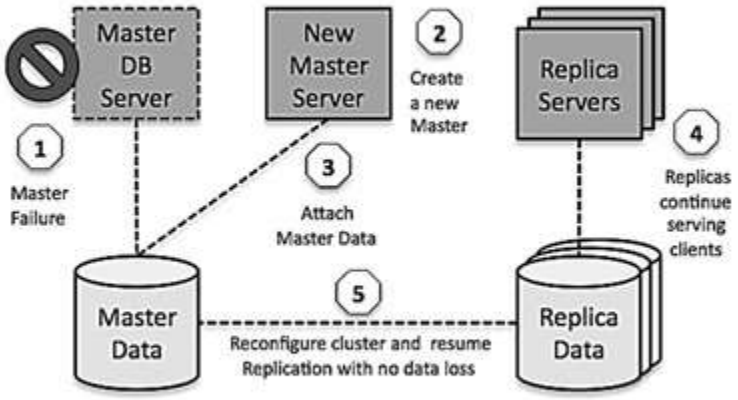
There are various simple operations that can be applied over the table which makes these databases easier to extend, join two databases with a common relation and modify all existing applications.



9.Cloud Databases

Now a day, data has been specifically getting stored over clouds also known as a virtual environment, either in a hybrid cloud, public or private cloud. A cloud database is a database that has been optimized or built for such a virtualized environment. There are various benefits of a cloud database, some of which are the ability to pay for storage capacity and bandwidth on a per-user basis, and they provide scalability on demand, along with high availability.

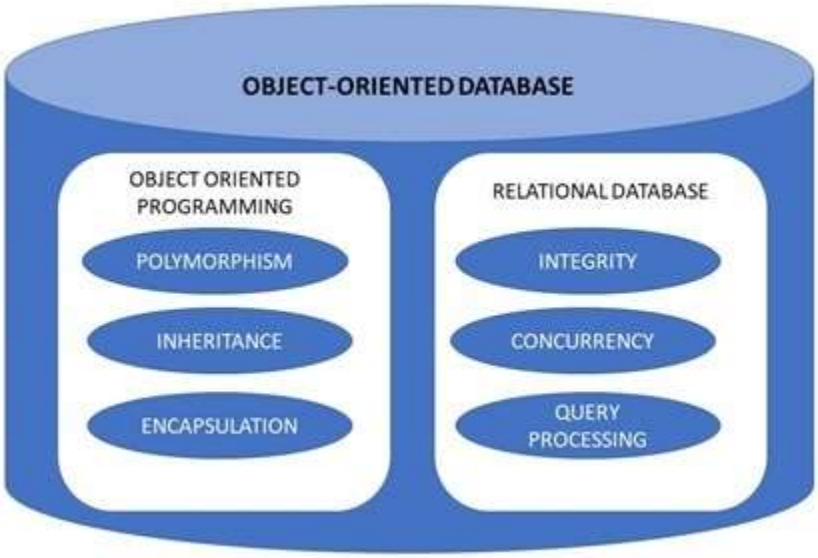
A cloud database also gives enterprises the opportunity to support business applications in a software-as-a-service deployment.



10.Object-Oriented Databases

An object-oriented database is a collection of object-oriented programming and relational database. There are various items which are created using object-oriented programming languages like C++, Java which can be stored in relational databases, but object-oriented databases are well-suited for those items.

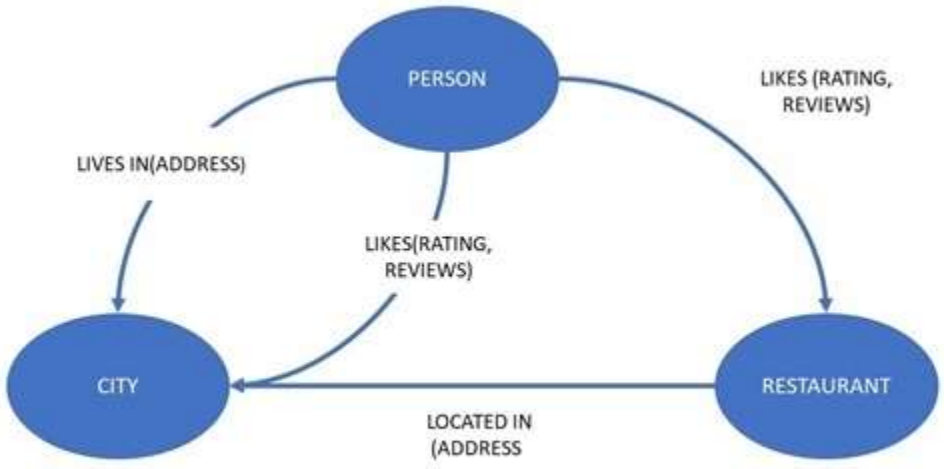
An object-oriented database is organized around objects rather than actions, and data rather than logic. For example, a multimedia record in a relational database can be a definable data object, as opposed to an alphanumeric value.



11.Graph Databases

The graph is a collection of nodes and edges where each node is used to represent an entity and each edge describes the relationship between entities. A graph-oriented database, or graph database, is a type of NoSQL database that uses graph theory to store, map and query relationships.

Graph databases are basically used for analyzing interconnections. For example, companies might use a graph database to mine data about customers from social media.



DATABASE USERS

Naive users :

Users who need not be aware of the presence of the database system or any other system supporting their usage are considered naïve users . A user of an automatic teller machine falls on this category.

Application programmers :

Professional programmers who are responsible for developing application programs or user interfaces utilized by the naïve and online user falls into this category.

Database Administration :

A person who has central control over the system is called database administrator .

The function of DBA are :

1. creation and modification of conceptual Schema

definition

2. Implementation of storage structure and access method.

3. schema and physical organization modifications .

4. granting of authorization for data access.

5. Integrity constraints specification.

6. Execute immediate recovery procedure in case of failures

7. ensure physical security to database

DATABASE MANAGER

Database manager:

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

The responsibilities of database manager are:

1. **Interaction with file manager**: The data is stored on the disk using the file system which is provided by operating system. The database manager translate the the different DML statements into low-level file system commands. so The database manager is responsible for the actual storing,retrieving and updating of data in the database.

2. **Integrity enforcement:**The data values stored in the database must satisfy certain constraints(eg: the age of a person can't be less then zero).These constraints are specified by DBA. Data manager checks the constraints and if it satisfies then it stores the data in the database.

3. **Security enforcement:**Data manager checks the security measures for database from unauthorized users.

4. **Backup and recovery:**Database manager detects the failures occurs due to different causes (like disk failure, power failure,deadlock,s/w error) and restores the database to original state of the database.

5. **Concurrency control:**When several users access the same database file simultaneously, there may be possibilities of data inconsistency. It is responsible of database manager to control the problems occurs for concurrent transactions.

DATABASE LANGUAGE :

1. **Data definition language(DDL) :**

DDL is used to define database objects .The conceptual schema is specified by a set of definitions expressed by this language. It also give some details about how to implement this schema in the physical devices used to store the data. This definition includes all the entity sets and their associated attributes and their relation ships. The result of DDL statements will be a set of tables that are stored in special file called data dictionary.

2. **Data manipulation language(DML) :**

A DML is a language that enables users to access or manipulate data stored in the database. Data manipulation involves retrieval of data from the database, insertion of new data into the database and deletion of data or modification of existing data.

There are basically two types of DML:

- **procedural**: Which requires a user to specify what data is needed and how to get it.

- **non-rocedural:** which requires a user to specify what data is needed with out specifying how

to get it.

3. **Data control language(DCL):**

This language enables user to grant authorization and canceling authorization of database objects.

TRANSACTION

A **transaction** is a *unit* of program execution that accesses and possibly updates various data items.A transaction is the DBMS's abstract view of a user program: a sequence of reads and writes A transaction must see a consistent database .During transaction execution the database may be temporarily inconsistent.A sequence of many actions which are considered to be one atomic unit of work .When the transaction completes successfully (is committed), the database must be consistent .After a transaction commits, the changes it has made to the database persist, even if there are system failures .Multiple transactions can execute in parallel.Two main issues to deal with:

- Failures of various kinds, such as hardware failures and system crashes
- Concurrent execution of multiple transactions

ACID Properties

To preserve the integrity of data the database system transaction mechanism must ensure:

- ***Atomicity*.** Either all operations of the transaction are properly reflected in the database or none are
- ***Consistency*.** Execution of a transaction in isolation preserves the consistency of the database
- ***Isolation*.** Although multiple transactions may execute concurrently, each transaction must be unaware of other concurrently executing transactions. Intermediate transaction results must be hidden from other concurrently executed transactions

That is, for every pair of transactions *Ti* and *Tj*, it appears to *Ti* that either *Tj*, finished execution before *Ti* started, or *Tj* started execution after *Ti* finished

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

ENTITY RELATIONSHIP DIAGRAM

An Entity Relationship Diagram (ERD) is a visual representation of different data using conventions that describe how these data are related to each other.

The entity-relationship data model perceives the real world as consisting of basic objects,

called entities and relationships among these objects. It was developed to facilitate data base design by allowing specification of an enterprise schema which represents the overall logical structure of a data base.

Main features of ER-MODEL:

• Entity relationship model is a high level conceptual model

• It allows us to describe the data involved in a real world enterprise in terms of

objects and their relationships.

• It is widely used to develop an initial design of a database

• It provides a set of useful concepts that make it convenient for a developer to

move from a baseid set of information to a detailed and description of information

that can be easily implemented in a database system

• It describes data as a collection of entities, relationships and attributes.

Basic concepts:

There are three basic elements in an ER Diagram: entity, attribute, relationship. There are more elements which are based on the main elements. They are weak entity, multivalued attribute, derived attribute, weak relationship and recursive relationship. Cardinality and ordinality are two other notations used in ER diagrams to further define relationships.

Entity

An entity can be a person, place, event, or object that is relevant to a given system. For example, a school system may include students, teachers, major courses, subjects, fees, and other items. Entities are represented in ER diagrams by a rectangle and named using singular nouns.

Weak Entity

A weak entity is an entity that depends on the existence of another entity. In more technical terms it can defined as an entity that cannot be identified by its own attributes. It uses a foreign key combined with its attributed to form the primary key. An entity like order item is a good example for this. The order item will be meaningless without an order so it depends on the existence of order.

Weak Entity Example in ER diagrams

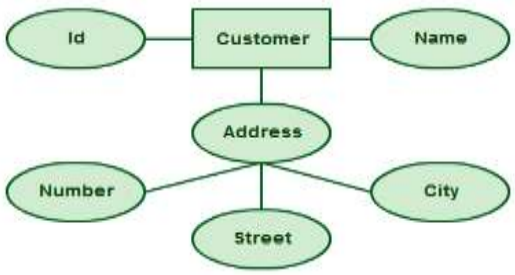
Attribute

Weak Entity Example in ER diagrams

Attribute

An attribute is a property, trait, or characteristic of an entity, relationship, or another attribute. For example, the attribute Inventory Item Name is an attribute of the entity Inventory Item. An entity can have as many attributes as necessary. Meanwhile, attributes can also have their own

specific attributes. For example, the attribute "customer address" can have the attributes number, street, city, and state. These are called composite attributes. Note that some top level ER diagrams do not show attributes for the sake of simplicity. In those that do, however, attributes are represented by oval shapes.



Attributes in ER diagrams, note that an attribute can have its own attributes (composite attribute)

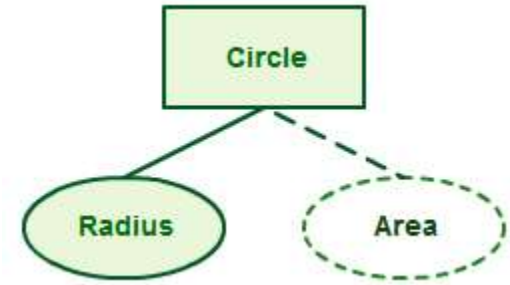
Multivalued Attribute

If an attribute can have more than one value it is called an multivalued attribute. It is important to note that this is different to an attribute having its own attributes. For example a teacher entity can have multiple subject values.

Example of a multivalued attribute

Derived Attribute

An attribute based on another attribute. This is found rarely in ER diagrams. For example for a circle the area can be derived from the radius.



Derived Attribute in ER diagrams

RELATIONSHIP

A relationship describes how entities interact. For example, the entity "carpenter" may be related to the entity "table" by the relationship "builds" or "makes". Relationships are represented by diamond shapes and are labeled using verbs.

Using Relationships in Entity Relationship Diagrams

Recursive Relationship

If the same entity participates more than once in a relationship it is known as a recursive relationship. In the below example an employee can be a supervisor and be supervised, so there is a recursive relationship.

Example of a recursive relationship in ER diagrams

Cardinality and Ordinality

These two further defines relationships between entities by placing the relationship in the context of numbers. In an email system, for example, one account can have multiple contacts. The

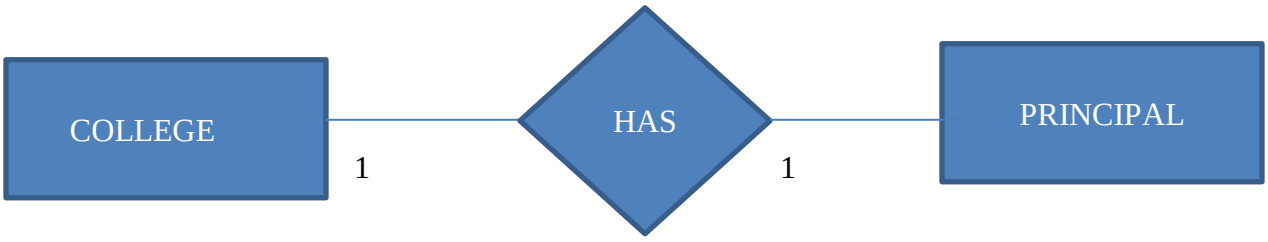
relationship in this case follows a "one to many" model. There are number of notations used to present cardinality in ER diagrams. Chen, UML, Crow's foot, Bachman are some of the popular notations.

one to one:

An entity in A is associated with at most one entity in B, and an entity in B is associated

with at most one entity in A.

Eg: relationship between college and principal

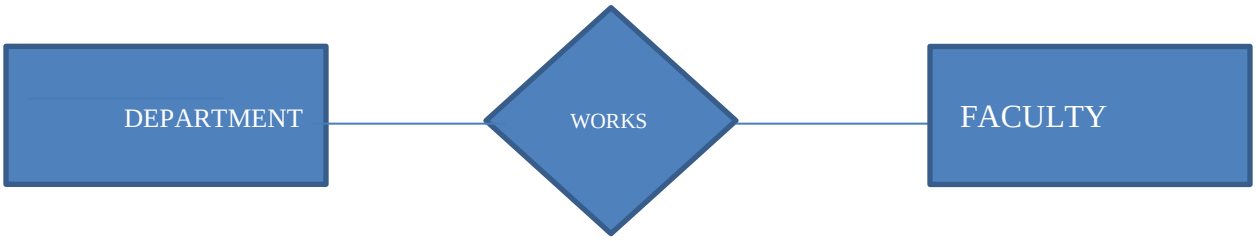


One to many:

An entity in A is associated with any number of entities in B. An entity in B is associated

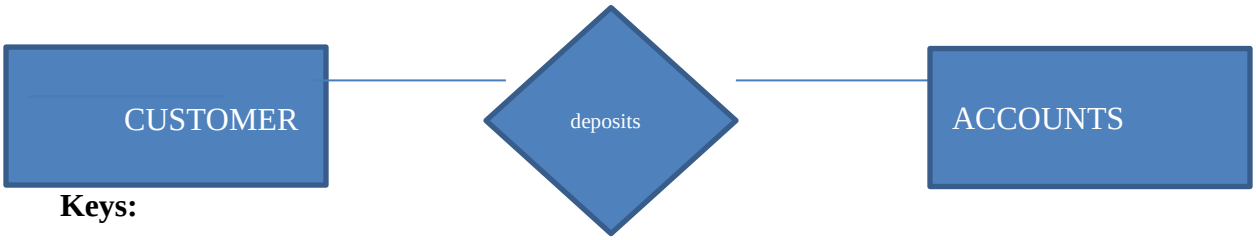
with at the most one entity in A.

Eg: Relationship between department and faculty



Many –to-many:

Entities in A and B are associated with any number of entities from each other.



Keys:

Super key: Super key:

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

For example , customer-id,(cname,customer-id),(cname,telno)

Candidate key:

In a relation R, a candidate key for R is a subset of the set of attributes of R, which have the following properties:

• *Uniqueness:* no two distinct tuples in R have the same values for the candidate key

• *Irreducible:* No proper subset of the candidate key has the uniqueness property that is the candidate key.

Eg: ((cname,telno)

Primary key:

The primary key is the candidate key that is chosen by the database designer as the principal means of identifying entities with in an entity set. The remaining candidate keys if any, are called *alternate key*.

Tips on How to Draw ER Diagrams

- Identify all the relevant entities in a given system and determine the relationships among these entities.
- An entity should appear only once in a particular diagram.
- Provide a precise and appropriate name for each entity, attribute, and relationship in the diagram. Terms that are simple and familiar always beats vague, technical-sounding words. In naming entities, remember to use singular nouns. However, adjectives may be used to distinguish entities belonging to the same class (part-time employee and full time employee, for example). Meanwhile attribute names must be meaningful, unique, system-independent, and easily understandable.
- Remove vague, redundant or unnecessary relationships between entities.
- Never connect a relationship to another relationship.
- Make effective use of colors. You can use colors to classify similar entities or to highlight key areas in your diagrams

ER Diagram Templates

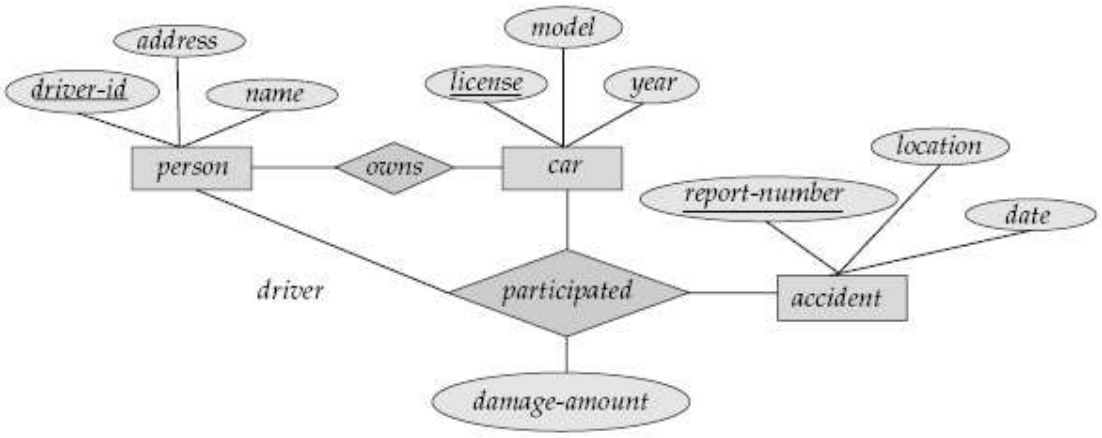


Figure 2.1 E-R diagram for a Car-insurance company.

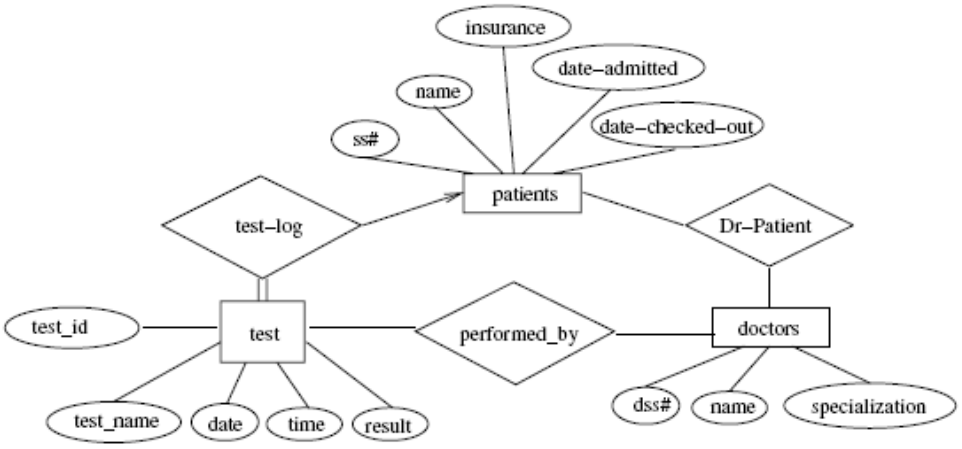


Figure 2.2 E-R diagram for a hospital.

A university registrar's office maintains data about the following entities: (a) courses, including number, title, credits, syllabus, and prerequisites; (b) course offerings, including course number, year, semester, section number, instructor(s), timings, and classroom; (c) students, including student-id, name, and program; and (d) instructors, including identification number, name, department, and title. Further, the enrollment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modeled.

Construct an E-R diagram for the registrar's office. Document all assumptions that you make about the mapping constraints.

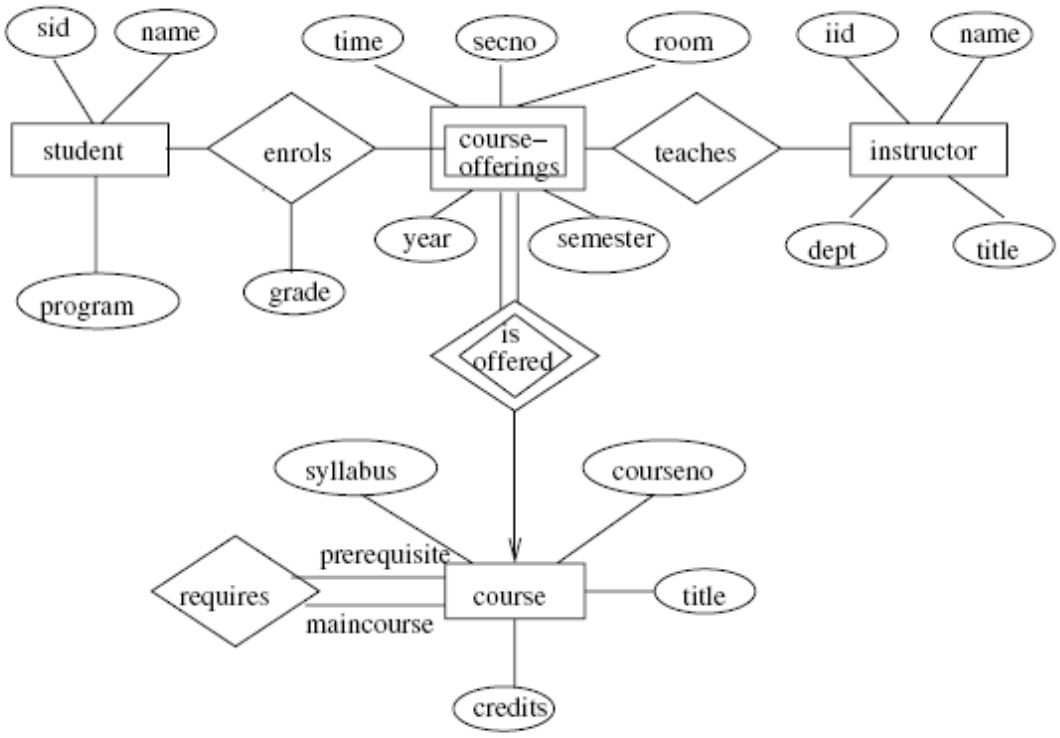


Figure 2.3 E-R diagram for a university.

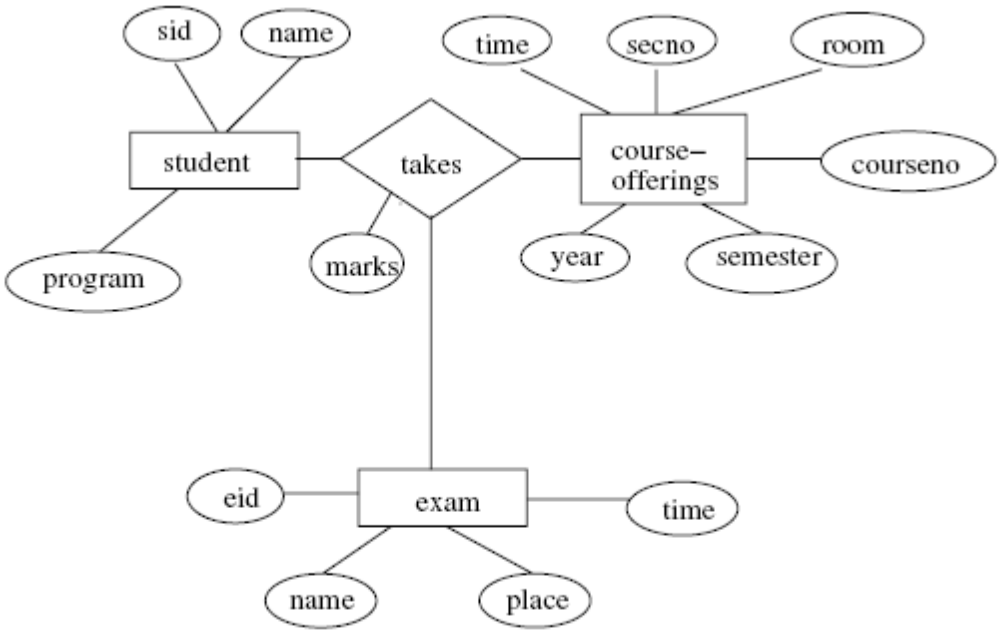


Figure 2.4 E-R diagram for marks database.

Benefits of ER diagrams

ER diagrams constitute a very useful framework for creating and manipulating databases. First, ER diagrams are easy to understand and do not require a person to undergo extensive training to be able to work with it efficiently and accurately. This means that designers can use ER diagrams to easily communicate with developers, customers, and end users, regardless of their IT proficiency. Second, ER diagrams are readily translatable into relational tables which can be used to quickly build databases. In addition, ER diagrams can directly be used by database developers as the blueprint for implementing data in specific software applications. Lastly, ER diagrams may be applied in other contexts such as describing the different relationships and operations within an organization.

Logical data and physical data :

Logical data are the data for the table created by user in primary memory.

Physical data refers to the data stored in the secondary memory.

Schema and sub-schema

A schema is a logical data base description and is drawn as a chart of the types of data that are used . It gives the names of the entities and attributes and specify the relationships between them.A database schema includes such information as :

 Characteristics of data items such as entities and attributes .

 Logical structures and relationships among these data items .

 Format for storage representation.

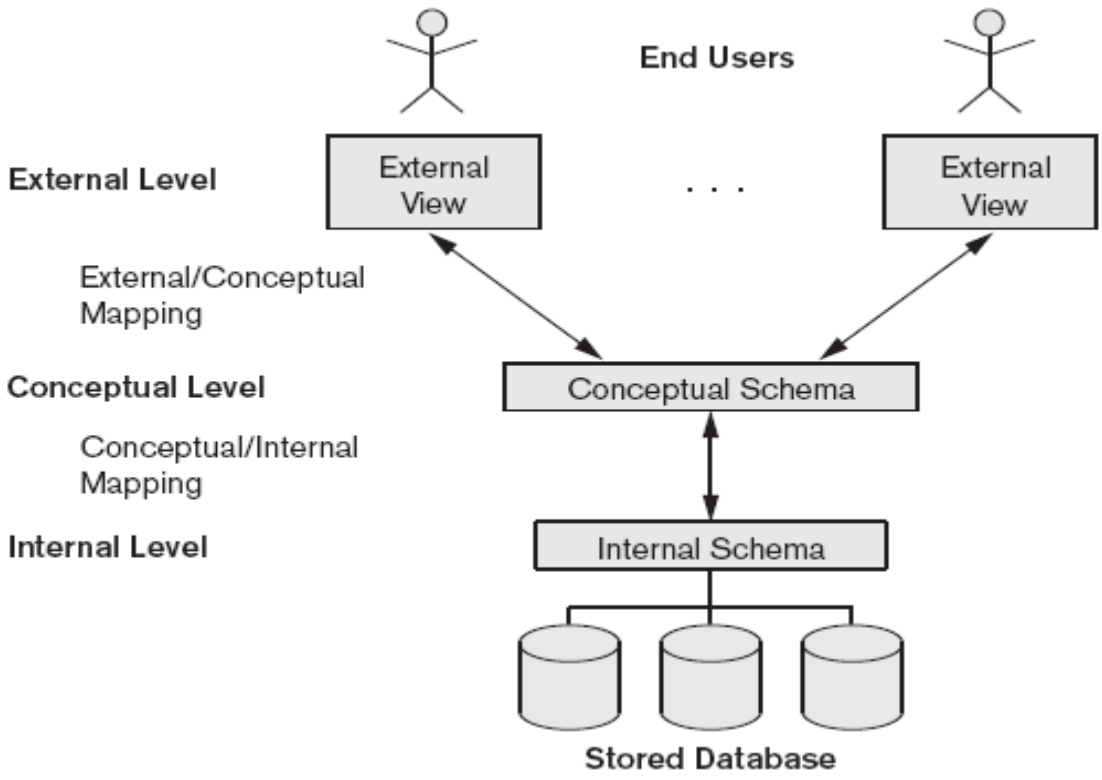
 Integrity parameters such as physical authorization and back up policies.

A *subschema* is derived schema derived from existing schema as per the user

requirement. There may be more then one subschema create for a single conceptual

schema.

THREE-SCHEMA ARCHITECTURE



1.Internal level -Describes physical storage structure of the database

2.**Conceptual** level -Describes structure of the whole DB for the complete community of users

3.**External** or **view** level -Describes part of the DB of interest to a particular user group

Database Instance

It is important that we distinguish these two terms individually. Database schema is the skeleton of database. It is designed when the database doesn't exist at all. Once the database is operational, it is very difficult to make any changes to it. A database schema does not contain any data or information.

A database instance is a state of operational database with data at any given time. It contains a snapshot of the database. Database instances tend to change with time. A DBMS ensures that its

every instance (state) is in a valid state, by diligently following all the validations, constraints, and conditions that the database designers have imposed.

DATA INDEPENDENCE

A database system normally contains a lot of data in addition to users' data. For example, it stores data about data, known as metadata, to locate and retrieve data easily. It is rather difficult to modify or update a set of metadata once it is stored in the database. But as a DBMS expands, it needs to change over time to satisfy the requirements of the users. If the entire data is dependent, it would become a tedious and highly complex job.

DATABASE /SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

Database/Software Development Life Cycle consists of details steps and activities which describes how to design, develop, maintain, replace, alter, enhance, test or even launch a software.

SDLC consist of seven stages Namely:

1.Planning/Investigation

This is the first phase in the systems development process. It identifies whether or not there is the need for a new system to achieve a business"s strategic objectives. This is a preliminary plan for a company"s business initiative to acquire the resources to build on an infrastructure to modify or improve a service. The company might be trying to meet or exceed expectations for their employees, customers and stakeholders too. The purpose of this step is to find out the scope of the problem and determine solutions. Resources, costs, time, benefits and other items should be considered at this stage.

2.Feasibility study

Feasibility is defined as the practical extent to which a project can be performed successfully. To evaluate feasibility, a feasibility study is performed, which determines whether the solution considered to accomplish the requirements is practical and workable in the software. Information such as resource availability, cost estimation for software development, benefits of the software to the organization after it is developed and cost to be incurred on its maintenance are considered

during the feasibility study. The objective of the feasibility study is to establish the reasons for developing the software that is acceptable to users, adaptable to change and conformable to established standards. Various other objectives of feasibility study are listed below.

- To analyze whether the software will meet organizational requirements
- To determine whether the software can be implemented using the current technology and within the specified budget and schedule
- To determine whether the software can be integrated with other existing software.

TYPES OF FEASIBILITY

Various types of feasibility that are commonly considered include technical feasibility, operational feasibility, and economic feasibility.

1.TECHNICAL FEASIBILITY STUDY

Technical feasibility assesses the current resources (such as hardware and software) and technology, which are required to accomplish user requirements in the software within the allocated time and budget. For this, the software development team ascertains whether the current resources and technology can be upgraded or added in the software to accomplish specified user requirements. Technical feasibility also performs the following tasks.

- Analyzes the technical skills and capabilities of the software development team members
- Determines whether the relevant technology is stable and established
- Ascertains that the technology chosen for software development has a large number of users so that they can be consulted when problems arise or improvements are required.

2.OPERATIONAL FEASIBILITY STUDY

Operational feasibility assesses the extent to which the required software performs a series of steps to solve business problems and user requirements. This feasibility is dependent on human resources (software development team) and involves visualizing whether the software will operate after it is developed and be operative once it is installed. Operational feasibility also performs the following tasks.

- Determines whether the problems anticipated in user requirements are of high priority

- Determines whether the solution suggested by the software development team is acceptable
- Analyzes whether users will adapt to a new software
- Determines whether the organization is satisfied by the alternative solutions proposed by the software development team.

3.ECONOMIC FEASIBILITY STUDY

Economic feasibility determines whether the required software is capable of generating financial gains for an organization. It involves the cost incurred on the software development team, estimated cost of hardware and software, cost of performing feasibility study, and so on. For this, it is essential to consider expenses made on purchases (such as hardware purchase) and activities required to carry out software development. In addition, it is necessary to consider the benefits that can be achieved by developing the software. Software is said to be economically feasible if it focuses on the issues listed below.

- Cost incurred on software development to produce long-term gains for an organization
- Cost required to conduct full software investigation (such as requirements elicitation and requirements analysis)
- Cost of hardware, software, development team, and training.

3. SYSTEMS ANALYSIS

The analysis phase is where businesses will work on the source of their problem or the need for a change. In the event of a problem, possible solutions are submitted and analyzed to identify the best fit for the ultimate goal(s) of the project. This is where teams consider the functional requirements of the project or solution. It is also where system analysis takes place—or analyzing the needs of the end users to ensure the new system can meet their expectations. Systems analysis is vital in determining what a business"s needs are, as well as how they can be met, who will be responsible for individual pieces of the project, and what sort of timeline should be expected. There are several tools businesses can use that are specific to the second phase. They include:

- CASE (Computer Aided Systems/Software Engineering)
- Requirements gathering

- Structured analysis

4.SYSTEM DESIGN

The design phase describes, in detail, the necessary specifications, features and operations that will satisfy the functional requirements of the proposed system which will be in place. This is the step for end users to discuss and determine their specific business information needs for the proposed system. It"s during this phase that they will consider the essential components (hardware and/or software) structure (networking capabilities), processing and procedures for the system to accomplish its objectives.

5.INTERGRATION AND TESTING

The fifth phase involves systems integration and system testing (of programs and procedures)—normally carried out by a Quality Assurance (QA) professional—to determine if the proposed design meets the initial set of business goals. Testing may be repeated, specifically to check for errors, bugs and interoperability. This testing will be performed until the end user finds it acceptable. Another part of this phase is verification and validation, both of which will help ensure the program"s successful completion.

TYPES OF TESTING

- *Unit testing*
- *Integration testing*
- *System testing*

Unit testing- is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc

Intergration testing- is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.

System Testing- is a level of software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements.

6.SYSTEM IMPLEMENTATION

The sixth phase is when the majority of the code for the program is written. Additionally, this phase involves the actual installation of the newly-developed system. This step puts the project into production by moving the data and components from the old system and placing them in the new system via a direct changeover. While this can be a risky (and complicated) move, the changeover typically happens during off-peak hours, thus minimizing the risk. Both system analysts and end-users should now see the realization of the project that has implemented changes.

Activities in Implementation stage include:

- *Training of end users on the system*
- *Changeover*
- *System documentation*

Types of changeover

Direct changeover:- Direct changeover, also referred to as immediate replacement, tends to be the least favorite of the changeover techniques. In a direct changeover, the entire system is replaced in an instant. Basically, as soon as the new system is powered up, the old system is shut down. This type of changeover carries the most risk because, if something goes wrong, reverting back to the old system usually is impossible. Using the direct changeover technique tends to work best in situations where a system failure isn't critical enough to result in a disaster for the company.

Parallel change over- In a parallel changeover, the new system runs simultaneously with the old for a given period of time. Of all the techniques, this tends to be the most popular, mainly because it carries the lowest risk. If something goes wrong at any point, the entire system can be reverted back to its original state. A primary disadvantage in running two systems at the same time is higher costs. The parallel changeover process also can be quite time-consuming.

Phased Changeover -The phased changeover technique is considered a compromise between parallel and direct changeovers. In a phased changeover, the new system is implemented one stage at a time. As an example, consider a company working toward installing a new financial

system. Implementing the new system one department at a time, the company converts accounts receivable, accounts payable, payroll, and so on. Advantages to phased changeovers are their low cost and isolated errors. The main disadvantage is the process takes a long time to complete because phases need to be implemented separately.

Pilot Changeover-With a pilot changeover, the new system is tried out at a test site before launching it company-wide. For example, a bank may first test the system at one of its branches. This branch is referred to as the pilot, or beta, site for the program. Since parallel changeovers tend to be expensive, using the pilot changeover technique allows companies to run the new system next to their old but on a much smaller scale. This makes the pilot changeover method much more cost-effective. After the things are worked out of the system at the test site, companies usually opt to use the direct changeover technique to launch the system company-wide.

7.**REVIEW AND MAINTENANCE**

The seventh and final phase involves maintenance and regular required updates. This step is when end users can fine-tune the system, if they wish, to boost performance, add new capabilities or meet additional user requirements.

Types of Maintenance

i. Corrective
ii. Adaptive
iii. Perfective
iv. Preventive

Corrective maintenance -Corrective Maintenance most commonly referred to as "bugs," is the most typical change associated with maintenance work. Corrective changes address errors and faults in your software that could affect various areas of your software; design, logic or code. Most commonly, these changes are sprung by bug reports created by users. It is important to note that sometimes problem reports submitted by users are actually enhancements of the system not bugs.

Adaptive Maintenance -Adaptive Maintenance is triggered by changes in the environment your software lives in. An adaptive change can be triggered by changes to the operating system, hardware, software dependencies and even organizational business rules and policies. These modifications to the environment can trigger changes within other parts of your software. For example, updating the server, compilers, etc or modifications to shipping carriers and payment processors can affect functionality in your software.

Perfective Maintenance -Perfective Maintenance refers to the evolution of requirements and features in your existing system. As your software gets exposed to users they will think of different ways to expand the system or suggest new features that they would like to see as part of the software, which in turn can become future enhancements to the system. Perfective changes also includes removing features from a system that are *not effective and functional to the end goal of the system*.

Preventive Maintenance -Preventive Maintenance refer to changes made to increase the understanding and maintainability of your software in the long run. Preventive Maintenance are focused in decreasing the deterioration of your software in the long run. Restructuring, optimizing code and updating documentation are common preventive changes. Executing preventive changes reduces the amount of unpredictable effects a software can have in the long term and helps it become scalable, stable, understandable and maintainable**.**

NORMALIZATIONS

Normalization is a database design technique which organizes tables in a manner that reduces redundancy and dependency of data.It divides larger tables to smaller tables and links them using relationships.

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

Normalization is used for mainly two purposes,

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

Normalization Rule

Normalization rules are divided into the following normal forms:

1. **First Normal Form**
2. **Second Normal Form**
3. **Third Normal Form**

FIRST NORMAL FORM 1NF

For a table to be in the First Normal Form, it should follow the following 4 rules:

1. It should only have single(atomic) valued attributes/columns.
2. Values stored in a column should be of the same domain
3. All the columns in a table should have unique names.
4. And the order in which data is stored, does not matter.

Although all the rules are self explanatory still let's take an example where we will create a table to store student data which will have student's roll no., their name and the name of subjects they have opted for.

Here is our table, with some sample data added to it.

roll_no	name	subject
101	Akon	OS, CN

103	Ckon	Java
102	Bkon	C, C++

Our table already satisfies 3 rules out of the 4 rules, as all our column names are unique, we have stored data in the order we wanted to and we have not inter-mixed different type of data in columns.

But out of the 3 different students in our table, 2 have opted for more than 1 subject. And we have stored the subject names in a single column. But as per the 1st Normal form each column must contain atomic value.

How to solve this Problem?

It's very simple, because all we have to do is break the values into atomic values.

Here is our updated table and it now satisfies the First Normal Form.

roll_no	**name**	**subject**
101	Akon	OS
101	Akon	CN
103	Ckon	Java
102	Bkon	C
102	Bkon	C++

By doing so, although a few values are getting repeated but values for the subject column are now atomic for each record/row.Using the First Normal Form, data redundancy increases, as there will be many columns with same data in multiple rows but each row as a whole will be unique.

SECOND NORMAL FORM (2NF)

Second Normal Form (2NF)

For a table to be in the Second Normal Form,

1. It should be in the First Normal form.
2. And, it should not have Partial Dependency.

What is Dependency?

Let's take an example of a Student table with columns student_id, name, reg_no(registration number), branch and address(student's home address).

student_id	Name	reg_no	branch	address

In this table, student_id is the primary key and will be unique for every row, hence we can use student_id to fetch any row of data from this table

Even for a case, where student names are same, if we know the student_id we can easily fetch the correct record.

student_id	Name	reg_no	branch	address
10	Akon	07-WY	CSE	Kerala
11	Akon	08-WY	IT	Gujarat

Hence we can say a Primary Key for a table is the column or a group of columns(composite key) which can uniquely identify each record in the table.

I can ask from branch name of student with student_id 10, and I can get it. Similarly, if I ask for name of student with student_id 10 or 11, I will get it. So all I need is student_id and every other column depends on it, or can be fetched using it.

This is Dependency and we also call it Functional Dependency.

What is Partial Dependency?

Now that we know what dependency is, we are in a better state to understand what partial dependency is.

For a simple table like Student, a single column like student_id can uniquely identfy all the records in a table.

But this is not true all the time. So now let's extend our example to see if more than 1 column together can act as a primary key.

Let's create another table for Subject, which will have subject_id and subject_name fields and subject_id will be the primary key.

subject_id	subject_name
1	Java
2	C++
3	Php

Now we have a Student table with student information and another table Subject for storing subject information.

Let's create another table Score, to store the marks obtained by students in the respective subjects. We will also be saving name of the teacher who teaches that subject along with marks.

score_id	student_id	subject_id	marks	teacher
1	10	1	70	Java Teacher
2	10	2	75	C++ Teacher
3	11	1	80	Java Teacher

In the score table we are saving the student_id to know which student's marks are these and subject_id to know for which subject the marks are for.

Together, student_id + subject_id forms a Candidate Key(learn about Database Keys) for this table, which can be the Primary key.

Confused, How this combination can be a primary key?

See, if I ask you to get me marks of student with student_id 10, can you get it from this table? No, because you don't know for which subject. And if I give you subject_id, you would not know for which student. Hence we need student_id + subject_id to uniquely identify any row.
But where is Partial Dependency?
Now if you look at the Score table, we have a column names teacher which is only dependent on the subject, for Java it's Java Teacher and for C++ it's C++ Teacher & so on.
Now as we just discussed that the primary key for this table is a composition of two columns which is student_id & subject_id but the teacher's name only depends on subject, hence the subject_id, and has nothing to do with student_id.
This is Partial Dependency, where an attribute in a table depends on only a part of the primary key and not on the whole key.

How to remove Partial Dependency?

There can be many different solutions for this, but out objective is to remove teacher's name from Score table.
The simplest solution is to remove columns teacher from Score table and add it to the Subject table. Hence, the Subject table will become:

subject_id	subject_name	teacher
1	Java	Java Teacher
2	C++	C++ Teacher
3	Php	Php Teacher

And our Score table is now in the second normal form, with no partial dependency.

score_id	student_id	subject_id	marks
1	10	1	70
2	10	2	75

3	11	1	80

NB

- For a table to be in the Second Normal form, it should be in the First Normal form and it should not have Partial Dependency.
- Partial Dependency exists, when for a composite primary key, any attribute in the table depends only on a part of the primary key and not on the complete primary key.
- To remove Partial dependency, we can divide the table, remove the attribute which is causing partial dependency, and move it to some other table where it fits in well.

Third Normal Form (3NF)

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

1. It is in the Second Normal form.
2. And, it doesn't have Transitive Dependency.

So let's use the same example, where we have 3 tables, **Student**, **Subject** and **Score**.

Student Table

student_id	name	reg_no	branch	address
10	Akon	07-WY	CSE	Kerala
11	Akon	08-WY	IT	Gujarat
12	Bkon	09-WY	IT	Rajasthan

Subject Table

subject_id	subject_name	teacher
1	Java	Java Teacher
2	C++	C++ Teacher
3	Php	Php Teacher

Score Table

score_id	student_id	subject_id	marks
1	10	1	70
2	10	2	75
3	11	1	80

In the Score table, we need to store some more information, which is the exam name and total marks, so let's add 2 more columns to the Score table.

score_id	student_id	subject_id	marks	exam_name	total_marks

Requirements for Third Normal Form

For a table to be in the third normal form,

1. It should be in the Second Normal form.
2. And it should not have Transitive Dependency.

What is Transitive Dependency?

With exam_name and total_marks added to our Score table, it saves more data now. Primary key for our Score table is a composite key, which means it's made up of two attributes or columns → student_id + subject_id.

Our new column exam_name depends on both student and subject. For example, a mechanical engineering student will have Workshop exam but a computer science student won't. And for some subjects you have Prctical exams and for some you don't. So we can say that exam_name is dependent on both student_id and subject_id.

And what about our second new column total_marks? Does it depend on our Score table's primary key?

Well, the column total_marks depends on exam_name as with exam type the total score changes. For example, practicals are of less marks while theory exams are of more marks.

But, exam_name is just another column in the score table. It is not a primary key or even a part of the primary key, and total_marks depends on it.

This is Transitive Dependency. When a non-prime attribute depends on other non-prime attributes rather than depending upon the prime attributes or primary key.

How to remove Transitive Dependency?

Again the solution is very simple. Take out the columns exam_name and total_marks from Score table and put them in an Exam table and use the exam_id wherever required.

Score Table: In 3rd Normal Form

score_id	student_id	subject_id	marks	exam_id

The new Exam table

exam_id	exam_name	total_marks
1	Workshop	200
2	Mains	70
3	Practicals	30

Advantage of removing Transitive Dependency

The advantage of removing transitive dependency is,

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

DATA ANOMALIES

Anomalies are problems that can occur in poorly planned, un-normalised databases where all the **data** is stored in one table (a flat-file database)

If a database design is not perfect, it may contain anomalies, which are like a bad dream for any database administrator. Managing a database with anomalies is next to impossible.

 Update anomalies: If data items are scattered and are not linked to each other properly, then it could lead to strange situations. For example, when we try to update one data item having its copies scattered over several places, a few instances get updated properly while a few others are left with old values. Such instances leave the database in an inconsistent state.

 Deletion anomalies: We tried to delete a record, but parts of it was left undeleted because of unawareness, the data is also saved somewhere else.

 Insert anomalies: We tried to insert data in a record that does not exist at all. Normalization is a method to remove all these anomalies and bring the database.

REVISION QUESTIONS

1. *Keys*. What is the difference between a candidate key, a primary key and a composite key? What considerations might influence the choice of a primary key?
2. Give a brief description of the following
   i. DBMS
   ii. Data Dictionary (DD)
   iii. Data Manipulation Language(DML)
3. Outline the main responsibilities of the Database Administrator ,End-users and Application Developers
4. Define and hence explain the use of
   i. Data mining
   ii. Data integrity
5. Many organization decides to use a database rather than a file management system for their applications. Explain the difference between a file management system and a database
6. Carefully distinguish between each of the following pairs of terms

(i) Field and record
(ii) Transaction file and back-up file
(iii) Object- oriented database model and distributed database model
(iv) Atomicity and consistency

7. A business has a file of customer orders which consists of records containing the fields ;

a) Customer reference number
b) Date of order
c) Customer name and address
d) Delivery town

e) Product code
f) Quantity

(i) Define the term **key field** for a record and identify the key field for the file of customer orders above.
(ii) What is a sort key(secondary key) by reference to the above file explain how processing the file using a sort key can produce useful information for the business

8. List and Explain the Advantages and Disadvantages of Database Management sytems.
9. Discuss the following process as expressed in database/system development life cycle (SDLC)

(i) Information systems planning
(ii) System initialization and identification
(iii) Feasibility study
(iv) System investigation
(v) System analysis
(vi) System implementation
(vii) System maintenance and documentations

10. Describe each of the following components of a database
    - Fields
    - Tables
    - Records
    - Attribute
    - triggers
    - Entity
    - stored procedures