**# Database:** Database are structure collection of data that are organized so that it’s easy to manage, access and update information. They are designed to store large amounts of data in a way that allows for efficient retrieval and manipulation, often using specialized software called database management systems. Databases are used in many application, form storing customer information for businesses to organizing scientific data for research.

**# Database Schema**: A database schema is skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relation among them are associated. It formulates all the constraints that are to be applied in the data. A database schema defines its entities and the relationship among them. It contains a descriptive details of the database, which can be depicted by means of schema diagram.

Database schema can be divided into two categories.

1. Physical database schema.
2. Logical database schema.

|  |  |
| --- | --- |
| Distributes Database | Centralized Database |
| Multipled database files are stored at various at various location. | It is made up of a single database file. |
| Multiple people can access and change data at the same time. | Bottlenecks occur when numerous user access the same file at the same time. |
| Files are sent swiftly from the users closest location. | It’s possible that delivering file at the same time. |
| Data can be recovered if one of the sites fails. | In the event of a system breakdown, a single site equals down time. |
| The synchronization of several files from various database is required. | In a single, central system it is easier to update and manage data . |

**# What is database Normalization? Explain about different normal forms with suitable example.**

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.

* There are different types of Normal forms :

|  |  |
| --- | --- |
| **Course** | **Content** |
| Programming Web | Java, C++, PHP, HTMl, ASP |

**1.First normal form (1NF**): First normal form is defined in the definition of relations itself. This rule defines that all the attributes in a relation must have atomic domains. The values in a atomic are indivisible units.

Example= Consider a table for storing student information. Instead of having a single column. For “Phone Number” where multiple phone numbers are stored together(e.g “123-456-7890, 987-654-3210”), you would have separate columns for each phone number.

**2.Second normal form(2NF):**2NF builds on 1NF and ensure that all columns in a table that are not parts of the primary key must depends on the entire primary key, not just a part of it.

Example: Suppose you have a table where “Student ID” and “Course code” together form the composite primary key. Attributes like “Student Name” and “course title” should depend on both “student ID” and “course code” not just one of them.

**3.Third Normal form(3NF):** 3NF builds on 2NFand ensure that there are no transitive dependencies: no non-key column. Example: In a table where “student ID” is the primary key and “Address” is a dependent on “city ”,”city” should be a separate table with its own primary key. This way “Address” depends directly on ”Student ID” and not indirectly through “city “.

**4.Boyce-Code Normal form(BCNF):** BCNF is a stricter version of 3NF, where every determinant (attribute that determines another attribute) is a candidate key. Example : If a table has attribute like {student ID, Course code, Grade} and {student ID, course code} is the candidate ley, Grade should directly depend on this key without any other dependencies.

# **Define Entity set and Relationship set.**

**\*Entity set:** An entity set refers to a collection or group of similar type of entities that share common attributes. In simpler terms, it represent a category or class of object or things that are relevant to the database. Example: In a database for a university, the entity set “student” would include all individual students, each characterized by attributes such as student ID , name, and date of birth.

**\*Relationship set:** A relationship set defines associations or connections between entity sets. It describes how entities form different entity other. Continuing with the university database example a relationship set “Enrolls In” might link the “student” entity set with the “course” entity set, indicating which students are enrolled in which courses.

**Shadow Paging:** The shadow paging approach compares two copies of disc memory to restore data. All transactional statements in a database management system must success or fail in all condition, including power outages, faults, and crashes in order to maintain atomicity. Shadow paging an alternative of log based system and is used to recover the database. During the process of shadow paging two page tables create such as current page table and shadow page table to access the pages of the bisk.

**Different types of SQL join:**

**Inner join:** The INNER JOIN keyword selects all rows from both the tables as long as the condition satisfies. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be same. Syntax: SELECT table1.column1,table1.column2,table2.column1,.. FROM table1 INNER JOIN table2 ON table1.matching\_column = table2.matching\_column;

**Left join:** This join returns all the rows of the table on the left side of the join and matching rows for the table on the right side of join. The rows for which there is no matching row on right side, the result-set will contain *null*. LEFT JOIN is also known as LEFT OUTER JOIN. Syntax: SELECT table1.column1,table1.column2,table2.column1,..FROM table1 LEFT JOIN table2 ON table1.matching\_column = table2.matching\_column;

**Right join:** RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of join. The rows for which there is no matching row on left side, the result-set will contain *null*. RIGHT JOIN is also known as RIGHT OUTER JOIN.

Syntax: SELECT table1.column1,table1.column2,table2.column1,.. FROM table1 RIGHT JOIN table2 ON table1.matching\_column = table2.matching\_column;

**Full Join**: FULL JOIN creates the result-set by combining result of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both the tables. The rows for which there is no matching, the result-set will contain *NULL* values.

**Syntax:** SELECT table1.column1,table1. column2, table2.column1,..

FROM table1 FULL JOIN table2 ON table1.matching\_column = table2.matching\_column;

**What is log based recovery? Explain Undo and Redo Protocols.**

Logs are the sequence of records, that maintain the records of actions performed by a transaction. In a log-Based recovery, log of each transaction is maintained in some stable storage. If any failure occurs, it can be recovered from there to recover the database. All these information will be stored in the order of executions.

**Undo protocols:** The undo protocol in database ensure that if a transaction fails or needs to be cancelled its change can be undone safely .It works by logging all actions a transaction performs. If the transaction aborts these actions are used to revere any changes it made, restoring the database to its previous state. This logging ensure that even after a crash, changes made by committed transactions can recovered accurately. **Redo Protocols:** The Redo protocol in database ensure that change made by committed transactions are durable and not last, especially after a system crash or failure. It works by logging updates to the database. If a transaction commits successfully, these logged actions are used to reapply its change during system recovery , ensuring that all committed changes are preserved and the database remain consistent.

**Short notes;**

**#Types of database users:** Database users are the one who really use and taken the benefits of database. There will be different types of users depending on their need and way of accessing the database.

* Application programmer
* Sophisticated users
* Specialized users
* Natïve users

**\*Native users** : Parametic end users are the unsophisticated who don’t have any DBMS Knowledge but they frequency use the database application in their daily life to get the desired results. For example, Railway’s ticket booking users are native users. **\*Sophisticated users:** Sophisticated users can be engineers, scientists , business analyst, who are familiar with the database. They don’t write the program code but they interact database writing SQL queries directly through the query processor.

**# Primary and Foreign key:** A Primary key is a column or set of columns in a table that uniquely identifies tuples (row) in that table. Example : Student table :

|  |  |  |
| --- | --- | --- |
| Stu\_Id | Stu\_Name | Stu\_Age |
| 101 | Ram | 20 |
| 102 | Sita | 18 |

In the above students table, the stu\_Id column uniquely identifies each row of the table.

**Properties of primary key:**

* A relation can have only one primary key.
* Primary key cannot contain null values.
* Each value in a primary key must be unique .

**# Foreign key:** Foreign keys are the columns of a table that points to the primary key of another table. In the below example, that stu\_Id column in course -enrollment table is a foreign key as it points to the primary key of the student table.

|  |  |
| --- | --- |
| Course\_Id | Stu\_Id |
| C01 | 101 |
| C02 | 102 |

Course-enrollment table:

Student table:

|  |  |  |
| --- | --- | --- |
| Stu\_Id | Stu\_Name | Stu\_Age |
| 101 | Ram | 20 |
| 102 | Sita | 18 |

**# Domain Constraints :** Domain constraints can be violated if an attribute value is not appearing in the corresponding domain or it is not of the appropriate data type. Domain constraints specify that within each tuple, and the value of each attribute must be unique. This is specified as data types which include standard data types integers, real numbers, characters, Booleans, variable length strings, etc.

Example: Create DOMAIN Customer Name CHECK (value not NULL)

The example shown demonstrates creating a domain constraint such that Customer Name is not NULL.

**# What is DBMS? Explain about the application of DBMS.**

Database Management System(DBMS) are software system used to store, retrieve and run queries on data. A DBMS serves as an interface between and end-user and a database allowing users to create, read, update and delete data in the database.

**Application of DBMS:**

**\*Banking:** for customer information, account activities, payment, deposits, loans etc.

**\*Airlines :** for reservation and schedule information:

**\*Universities:** for student information, course registration, college and grade.

**\*Telecommunication:** It help to keep call records, monthly bills maintaining balance etc.

**\*Sales**: use for storing customer, product and sales information.

**\*Manufacturing**: It is used for the management of supply chain and for tracking production of items. Inventories status in warehouses.

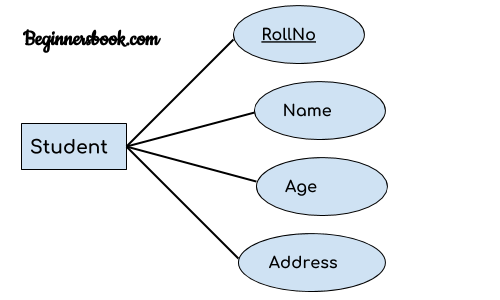
|  |  |
| --- | --- |
| **Disadvantage** | **Advantage** |
| - Management complexity.  - Maintaining currency.  - Frequently upgrade/ replacement cycles.  - Increased costs | **-** Improve the data sharing  **-** Improve the data security.  **-** Better data integration.  **-** Better data integration.  **-** Minimized data inconsistency. |

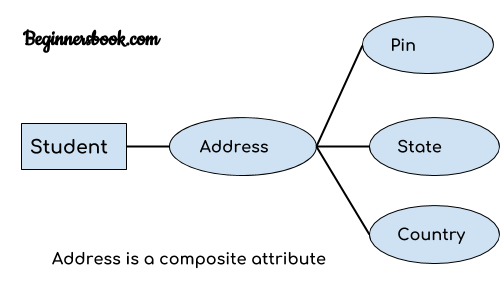
**# What are attribute ? Explain different types of attributes withs example.**

An attribute describes the property of an entity. An attribute is represented as Oval in an ER diagram. There are four types of attributes:

1. Key attribute  
2. Composite attribute  
3. Multivalued attribute  
4. Derived attribute.

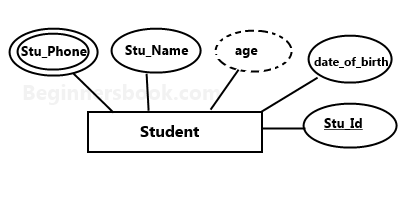
**\* Key** **attribute:** A key attribute can uniquely identify an entity from an entity set. For example, student roll number can uniquely identify a student from a set of students. Key attribute is represented by oval same as other attributes however the text of key attribute is underlined.



***\*Composite attribute:*** An attribute that is a combination of other attributes is known as composite attribute. For example, In student entity, the student address is a composite attribute as an address is composed of other attributes such as pin code, state, country. 

**\*Multivalued attribute:** An attribute that can hold multiple values is known as multivalued attribute. It is represented with **double ovals** in an ER Diagram. For example – A person can have more than one phone numbers so the phone number attribute is multivalued.

**\* Derived attribute:** A derived attribute is one whose value is dynamic and derived from another attribute. It is represented by **dashed oval** in an ER Diagram. For example – Person age is a derived attribute as it changes over time and can be derived from another attribute (Date of birth).



**# What is Data Model? List different data models and explain**

A set of concept to describe the structure of a database and certain constraints that the databases should abbey.

**Types of data model:**

**i)Hierarchical Data model:** - logically represents by an upside down tree.

- Each parent can have many children.

- Each child has only one parent.

- The top layer is perceived as the parent of the segment directly beneath it.

**ii)Network Data model:** - Graphics structure.

- Allow more connection between nodes.

- Ex: A employes work for two department is not possible in hierarchical model but here it is possible.

**iii)Relational data model:**  - Data in the form of table.

-Each table = application entity.

-Each roe=instances of the entity.

-SQL serves as an uniform interface for user providing a collection of standard expression for storing and retrieving data.

**\*What is data model ? List different data models and explain**.

> A set of concept to describe the structure of a database and certain constraints that the database should obey.

**Types of data model:** **1.Hierachical Data model: -** logically represented by an upside down tree. **-** Each parent can have many children. **–** Each child has only one parent. **–** The top layer is perceived as the parent of the segment directly beneath it .

**2. Network data model: -** Graph structure. **-** Allow more connection between nodes. **-** EX: A employee work for two department is not possible in hierarchical model but here it’s possible.

**\* What are Integrity constraints? Explain about different domain constraints.**

Integrity constraints are a set of rules. It is used to maintain the quality of information. Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected. Thus, integrity constraint is used to guard against accidental damage to the database.

**Domain constraints:** Domain constraints can be defined as the definition of a valid set of values for an attribute. The data type of domain includes string, character, integer, time, date, currency, etc. The value of the attribute must be available in the corresponding domain.

**\* Explain about domain constraint: 1) Not Null  constraint:** NOT NULL constraint restricts a column from having a NULL value. Once NOT NULL constraint is applied to a column, you cannot pass a null value to that column. It enforces a column to contain a proper value.Eg: CREATE TABLE Student(s\_id int NOT NULL, Name varchar(60), Age int); The above query will declare that the s\_id field of Student table will not take NULL value.

**2) UNIQUE constraint: UNIQUE** constraint ensures that a field or column will only have unique values. A **UNIQUE** constraint field will not have duplicate data. This constraint can be applied at column level or table level. E.g: CREATE TABLE Student(s\_id int NOT NULL UNIQUE, Name varchar(60), Age int); The above query will declare that the **s\_id** field of **Student** table will only have unique values and wont take NULL value.

**3) Primary key constraint:** Primary key constraint uniquely identifies each record in a database. A Primary Key must contain unique value and it must not contain null value. Usually Primary Key is used to index the data inside the table. E.g: CREATE table Student (s\_id int PRIMARY KEY, Name varchar(60) NOT NULL, Age int) The above command will creates a PRIMARY KEY on the s\_id.

**4) foreign key**: In **Customer\_Detail** table, **c\_id** is the primary key which is set as foreign key in **Order\_Detail** table. The value that is entered in **c\_id** which is set as foreign key in **Order\_Detail** table must be present in **Customer\_Detail** table where it is set as primary key. This prevents invalid data to be inserted into **c\_id** columnof **Order\_Detail** table.If you try to insert any incorrect data, DBMS will return error and will not allow you to insert the data. EXAMPLE: CREATE table Order\_Detail( order\_id int PRIMARY KEY, order\_name varchar(60) NOT NULL, c\_id int FOREIGN KEY REFERENCES Customer\_Detail(c\_id) );

**\* What is Query processing? Explain different steps involved in query processing.**

**Query processing** is the method a database uses to interpret and execute a request for data. When you ask a database for information, several steps happen to make sure you get the correct data efficiently. Here’s a simple breakdown of the steps involved:

**Steps in Query Processing**

**\*Parsing:** The database checks if your query is written correctly (like proper grammar in a sentence). It creates a structure (called a parse tree) that represents the query. **\*** **Translation:** The parse tree is converted into a logical plan that outlines what the database needs to do to get your data. **\*** **Optimization:** The database looks for the best way to execute the query. It considers different methods and chooses the most efficient one, like figuring out the quickest route to a destination. **\* Execution Plan Generation:** The optimized plan is turned into a detailed execution plan, which lays out the exact steps to retrieve the data. **\* Execution:** The database carries out the execution plan, accessing the data and performing necessary operations (like filtering or joining tables). \* **Result Formatting:** The final data is organized and formatted so you can easily understand it, and then it’s sent back to you. **Example**: SELECT name FROM employees WHERE department = 'Sales';

**\* What is transaction in database ?**

A transaction is an action, or series of actions that are being performed by a single user or application program, which reads or updates the contents of the database. A transaction can be defined as a logical unit of work on the database. This may be an entire program, a piece of a program or a single command and it may engage in any number of operations on the database.

**Properties of Transaction (ACID): \*Atomicity:** A transaction is a single unit of operation. You either execute it entirely or do not execute it at all. There cannot be partial execution. \***Consistency:** Once the transaction is executed, it should move from one consistent state to another. \***Isolation:**Transaction should be executed in isolation from other transactions (no Locks). During concurrent transaction execution, intermediate transaction results from simultaneously executed transactions should not be made available to each other. \***Durability:** **·**After successful completion of a transaction, the changes in the database should persist. Even in the case of system failures.

**\*Explain strict 2pl with an example.** Strict 2PL works similarly to the basic 2PL protocol at first. It gradually obtains locks as needed, but the locks are only released after a commit. A commit in database terms is the command used to tell the database to execute the queries now instead of rolling back. ***Example:*** **# Strict 2PL example def strict\_2pl(transaction\_a, transaction\_b): # Begin transaction A transaction\_a.begin() # Transaction A acquires lock on resource X transaction\_a.lock(resource\_x) # Transaction A performs operation on resource X transaction\_a.operate(resource\_x) # Commit transaction A transaction\_a.commit() # Begin transaction B transaction\_b.begin() # Transaction B acquires lock on resource X transaction\_b.lock(resource\_x) # Transaction B performs operation on resource X transaction\_b.operate(resource\_x)# Commit transaction B transaction\_b.commit()**

**\*What is database security ? Why do we need database security ? Explain the levels of database security.**

->Database security is the process, tools and controls that secure and protect database against and intentional threats.

->The objective of database security is to secure sensitive data and maintain the confidentiality, availability and integrity of the database. **the main levels of database security** **1. Physical Security** : Protects the hardware and facilities where the database is stored. Involves security measures like surveillance cameras, access control systems, and environmental controls (temperature, humidity). **2. Network Security :** Protects the database from unauthorized access over a network. Includes firewalls, intrusion detection systems, and secure network protocols (like VPNs). **3. Database Access Control** : Ensures that only authorized users can access the database. Uses authentication (verifying user identities) and authorization (defining user permissions). Roles and privileges are assigned based on user needs (e.g., read, write, execute). **4. Data Encryption** : Protects data by converting it into a coded format. Data can be encrypted at rest (stored data) and in transit (data being transmitted over networks). Ensures that even if data is intercepted, it remains unreadable without the decryption key. **5. Audit and Monitoring**: Tracks access and modifications to the database. Logs activities to identify any unauthorized access or anomalies. Regular audits help ensure compliance with security policies and regulations. **6. Backup and Recovery**: Protects data from loss due to hardware failures, data corruption, or disasters. Regular backups ensure that data can be restored to a specific point in time. A recovery plan outlines procedures for restoring data and services after an incident. **7. Application Security**: Ensures that applications accessing the database are secure. Involves validating input data to prevent SQL injection and other attacks. Regularly updating and patching applications to fix vulnerabilities

**\*Explain Transaction states in detail.** Here are the main transaction states: **1. New (or Created) :** The transaction is initiated but not yet started.It is waiting for execution. **Example:** A user starts a new transaction in a database application. **2. Active (or Running):**  The transaction is currently executing.It performs operations such as reading or writing data. **Example:** A user is in the process of updating records. **3. Partially Committed:** The transaction has executed all its operations but has not yet been committed. It’s waiting for a commit command. **Example:** A transaction has finished its operations, but the user hasn’t confirmed to save the changes yet. **4. Failed:** The transaction cannot proceed due to an error (e.g., system crash, constraint violation).Changes made by the transaction are not permanent. **Example:** A transaction tries to insert a record that violates a unique constraint. **5. Aborted :** The transaction is rolled back, and all operations are undone. The database state is restored to what it was before the transaction started. **Example:** A user decides to cancel a transaction after encountering an error. **6. Committed:** The transaction has been successfully completed and all changes are permanently saved to the database. It confirms that the transaction's operations are final. **Example:** A transaction that updates records has been confirmed by the user, making all changes permanent.

**\*\*Introduction of SQL:** Structure Query Language(SQL) is a database query language used for storing and managing data in Relational DBMS. SQL was the first commercial language introduced for E.F Codd's **Relational** model of database. Today almost all RDBMS(MySql, Oracle, Infomix, Sybase, MS Access) use **SQL** as the standard database query language. SQL is used to perform all types of data operations in RDBMS. **Some key point about SQL:**

* Database management
* Data querying
* Data manipulation
* Schema definition
* Access control
* Transaction

**\*\* Functional Dependency:**

**Definition**: A functional dependency (FD) is a relationship between two attributes in a database, denoted as **X → Y**. It indicates that if two tuples (rows) have the same value for attribute(s) X, they must also have the same value for attribute(s) Y. **Key Concepts**:

**Determinant**: The attribute(s) on the left side (X) that determines the value of the attribute(s) on the right side (Y). **Dependent Attribute**: The attribute(s) on the right side (Y) that depends on the determinant.**Examples**: **Student Table**: **StudentID → Name**: StudentID uniquely determines the Name. **StudentID → Major**: StudentID uniquely determines the Major. **Properties**

1. **Reflexivity**: If Y is a subset of X, then X → Y holds.
2. **Augmentation**: If X → Y, then XZ → YZ for any Z.
3. **Transitivity**: If X → Y and Y → Z, then X → Z.

**Types**

* **Trivial FD**: X → Y is trivial if Y is a subset of X.
* **Non-Trivial FD**: X → Y is non-trivial if Y is not a subset of X.
* **Partial Dependency**: Non-key attribute depends on part of a composite key.
* **Transitive Dependency**: X → Z via Y (if X → Y and Y → Z).

**\*\*Concept of Recovery :** Database recovery refers to the process of restoring a database to a consistent state after a failure or corruption. It involves bringing the database back to a point where data is accurate, transactions are committed, and system integrity is maintained. Several techniques are employed to achieve effective database recovery, each tailored to address specific types of failures.

Types of database recovery:

1) Backup and Restore.

2)Transaction log processing.

3)Checkpoint mechanisms.

4)Shadow paging.

5)Database Replication

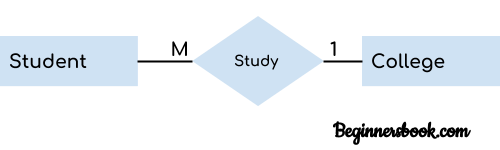
**\*Define terms data and information.**

**Data:** Data refers to information, facts or figure that are collected, stored and analyzed for reference or analysis. It can be in various form such as numbers, text, images or multimedia.

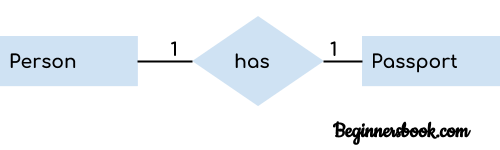
**Information:** Information refers to processed or organized data that has context, relevance and purpose. It is meaningful data that provides insights , knowledge or instruction to someone who receives it.

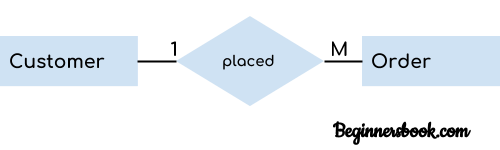
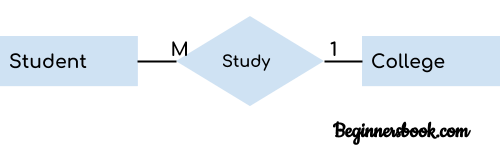
**\*Explain ER-diagrams. What are entity, attributes and keys ? Describe different types of relationship**.

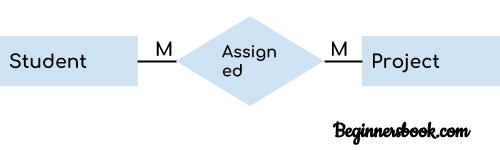
An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database.

**# Entity:** An entity is an object or component of data. An entity is represented as rectangle in an ER diagram. For example: In the following ER diagram we have two entities Student and College and these two entities have many to one relationship as many students study in a single college. 

**Relationship:** A relationship is represented by diamond shape in ER diagram, it shows the relationship among entities. There are four types of relationships:  
1. One to One  
2. One to Many  
3. Many to One  
4. Many to Many **1)One to one relationship:** When a single instance of an entity is associated with a single instance of another entity then it is called one to one relationship. For example, a person has only one passport and a passport is given to one person**.**

****

**2) One to many relationship**: When a single instance of an entity is associated with more than one instances of another entity then it is called one to many relationship. For example – a customer can place many orders but a order cannot be placed by many customers. **3) Many to one relationship:** When more than one instances of an entity is associated with a single instance of another entity then it is called many to one relationship. For example – many students can study in a single college but a student cannot study in many colleges at the same time

**4) Many to many relationship**: When more than one instances of an entity is associated with more than one instances of another entity then it is called many to many relationship. For example, a can be assigned to many projects and a project can be assigned to many students. 

**\* Why Normalization is needed in database.**

Normalization is needed in database to minimize redundancy and dependency by organizing data into tables and defining relationships between them. This process ensure data integrity and efficiency in data storage and retrieval. In simple terms, normalization helps in:

* Reducing Redundancy.
* Avoiding update Anomalies
* Improving Data Integrity.

**# Relational mapping:** A relational mapping transforms any objects data members type to a corresponding relational dtabase (SQL) data source representation in any supported relational database. Relational mapping let you map on object model into a relational data model. Relational mapping transforms object data members to relational database fielda. Use them to map simple data including primitives JdK classes and large object values.

**# Data Recovery:** Database system like any others computer system are subject to failures but the data stored in them must be availabel as and when requried. When a database fails it must possess the fscilities for fast recovery. It must also have atoimicity i.e. either transactions are complete successfully and committed or the transaction should have no effect on database.

Types of data recovery:

-Rollback/ Undo Recovery

-Commit/ Redo Recovery

**\* Explain Time-stamp ordering Techniques.**

The time-stamp ordering protocol is used to order the transactions based on their time stamps . The order of transaction is nothing but the ascending order of the transaction creation. The priority of the older transactions is higher that’s why it executes first.

Ordering rules:

**Read (x) operation:**

-If W-TS(X) > TS(Ti) then the operation is rejected .

-If W-TS(X) >=TS(Ti) then the operation is executed.

-Timestamp of all the data items are updated.

**Write (X) operations:**

**-**If TS(Ti) < R- TS(x) then the operation is rejected.

-If TS(Ti) < W-TS(x) then the operation is rejected and Ti is rolled back otherwise the operation is executed.

**# Referential Integrity:** Referential Integrity is a realtional database concept , which states that table relationships must always be consistant. In order words any foregin key fields must agree with athe primary key that is referenced by the foregin key. Primary key fields changes must be applied to all foregin keys, or not all. The same restriction also applies to foregin keys in that any updates must be propagated to the primary parent key .

**\*Define relatoional databse. Explain different types of key.**

A relational database ia a type of database that stores and organized data in tables with rows and columns. Each tables represents a specific entity or the table represents a unique instance or record of that entity.

**Key:** key are the essential elements of any relational database . It identifies each tuple in a relation uniquely. Keys are also used to establish the relationship among the tables in a schema. \*Different types of keys: **Super key:** A single attribute or a set of attributes that can uniquely identify all attribyutes of a praticular relation is called super key. A super key is the most general type of key. Example : employe table.

|  |  |  |
| --- | --- | --- |
| **Emp\_id** | **Emp\_no.** | **Emp\_name** |
| **E01** | **3001** | **Hari** |
| **E02** | **3002** | **Sita** |

**Candidate key:** A candidate key is a super key that contains no extra attribute. Candidate keys are selected form the set of super keys, the only thing we take care while selecting candidate key is : It should not have any redundant attributes. Example : employe tables.

|  |  |  |
| --- | --- | --- |
| **Emp\_id** | **Emp\_no.** | **Emp\_name** |
| **E01** | **3001** | **Hari** |
| **E02** | **3002** | **Sita** |

**Primary key:** A primary key is a column or set of column on a table that uniquely identifiers tuples in that tables. Example ; student tables:

|  |  |  |
| --- | --- | --- |
| **std\_id** | **Std\_name** | **Std\_age** |
| **01** | **Ram** | **19** |
| **02** | **Hari** | **30** |

**Foreign key:** Foreign keys are the columns of a table that points to the primary key of another table. For example: In the below example the stu\_id column in course\_enrollment table is a foreign key as it points to the primary key of the student table. course\_enrollment ::

|  |  |
| --- | --- |
| **Course\_id** | **Stu\_id** |
| **C01** | **01** |
| **C02** | **02** |

Student tables:

|  |  |  |
| --- | --- | --- |
| **std\_id** | **Std\_name** | **Std\_age** |
| **01** | **Ram** | **19** |
| **02** | **Hari** | **30** |

**\*Different between Authentication and Authorization.**

|  |  |
| --- | --- |
| **Authentication** | **Authorization** |
| It determines whether users are who they are claiming to be. | It determines the access that should be given or defined to an employee / user |
| It is done before authorization | It is done after the user successfully authenticate themselves. |
| The data is moved through data tokens. | The data is moved through access token |
| Authentication is visible to the user. | Authorization is not visible to the user. |
| It process is changeable by the user. | It is not changeable by the user. |
| e.x. Users in a social networking site are required to authenticate them selves before they are allowed access. | e.x. After a user authenticates successfully the networking site will determines where they are allowed access. |

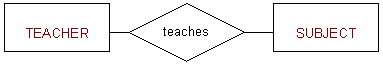
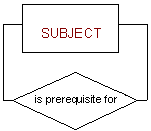
***\* Explain types of Authorization*. ->There are several types of Authorization: *1)Discretionary Access Control (DAC)*:** In DAC, the owner of a resource decide who can access it and what permission they have. For examples in a files system, the owner of a file can set permission( read, write, execute) for specific user or groups. ***2)Mandatary Access Control (MAC):*** MAC is more rigid than DAC and is often used in environments where security is critical. Access decisions are based on security labels assigned to both subject and object. These label define who can access what based on strict rules set by a system administrator. ***3)Role-Based Access Control(RBAC):*** RBACassigns permissions to roles rather than individual users. Users are then assigned to roles based on their job function or responsibilities. ***4)Attribute- Based Access Control(ABAC):*** ABAC uses attributes of users resources and the environment to make access decisions. For examples access might be granted based on a users job title, location, time of day or any other attribute that can be defined in policies.

***\*Entity -relationship model***

In this database model, relationships are created by dividing object of interest into entity and its characteristics into attributes. Different entities are related using relationships. E-R Models are defined to represent the relationships into pictorial form to make it easier for different stakeholders to understand. This model is good to design a database, which can then be turned into tables in relational model(explained below).

***\*Define Degree of relationship. Explain about the relationship in ER model.***

The *degree* of a relationship is the number of entity types that participate in the relationship. The three most common relationships in ER models are *Binary, Unary*and*Ternary.*

**Binary relationship :**A **binary relationship**is when two entities participate, and is the most common relationship degree. **Unary relationship**: A unary relationship is when both participants in the relationship are the same entity 

***Ternary Relationship***: A ternary relationship is when three entities participate in the relationship. 