----------SQL Interview Questions----------

1. What is SQL?

SQL stands for Structured Query Language. It is a language used to interact with the database, i.e to create a database, to create a table in the database, to retrieve data or update a table in the database, etc. SQL is an ANSI(American National Standards Institute) standard.

2. What are the different types of SQL commands?

- SELECT: Retrieves data from a database.
- INSERT: Adds new records to a table.
- UPDATE: Modifies existing records in a table.
- DELETE: Removes records from a table.
- CREATE: Creates a new database, table, or view.
- ALTER: Modifies the existing database object structure.
- DROP: Deletes an existing database object.

3. **Explain 5 SQL Subsets / Languages / Interaction commands ?**

1. **DDL** – Data Definition Language
2. **DQL** – Data Query Language
3. **DML** – Data Manipulation Language
4. **DCL** – Data Control Language
5. **TCL** – Transaction Control Language

DDL (Data Definition Language)

Data Definition Language actually consists of the SQL commands that can be used to define the database schema.

Command	Description	Syntax
CREATE	Create database or its objects (table, index, function, views, store procedure, and triggers)	CREATE TABLE table_name (column1 data_type, column2 data_type, ...);
DROP	Delete objects from the database	DROP TABLE table_name;
ALTER	Alter the structure of the database	ALTER TABLE table_name ADD COLUMN column_name data_type;
TRUNCATE	Remove all records from a table, including all spaces allocated for the records are removed	TRUNCATE TABLE table_name;

Command	Description	Syntax
COMMENT	Add comments to the data dictionary	COMMENT 'comment_text' ON TABLE table_name;
RENAME	Rename an object existing in the database	RENAME TABLE old_table_name TO new_table_name;

DQL (Data Query Language)

DQL statements are used for performing queries on the data within schema objects. The purpose of the DQL Command is to get some schema relation based on the query passed to it.

Command	Description	Syntax
SELECT	It is used to retrieve data from the database	SELECT column1, column2, ...FROM table_name WHERE condition;

DML (Data Manipulation Language)

The SQL commands that deal with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

Command	Description	Syntax
INSERT	Insert data into a table	**INSERT** INTO table_name (column1, column2, ...) VALUES (value1, value2, ...);
UPDATE	Update existing data within a table	**UPDATE** table_name SET column1 = value1, column2 = value2 WHERE condition;
DELETE	Delete records from a database table	**DELETE** FROM table_name WHERE condition;
LOCK	Table control concurrency	**LOCK** TABLE table_name IN lock_mode;
CALL	Call a PL/SQL or JAVA subprogram	**CALL** procedure_name(arguments);
EXPLAIN PLAN	Describe the access path to data	**EXPLAIN PLAN** FOR SELECT * FROM table_name;

DCL (Data Control Language)

DCL includes commands such as GRANT and REVOKE which mainly deal with the rights, permissions, and other controls of the database system.

Command	Description	Syntax
GRANT	Assigns new privileges to a user account, allowing access to specific database objects, actions, or functions.	**GRANT** privilege_type [(column_list)] ON [object_type] object_name TO user [WITH GRANT OPTION];
REVOKE	Removes previously granted privileges from a user account, taking away their access to certain database objects or actions.	**REVOKE** [GRANT OPTION FOR] privilege_type [(column_list)] ON [object_type] object_name FROM user [CASCADE];

TCL (Transaction Control Language)

Transactions group a set of tasks into a single execution unit. Each transaction begins with a specific task and ends when all the tasks in the group are successfully completed. If any of the tasks fail, the transaction fails.

Command	Description	Syntax
BEGIN TRANSACTION	Starts a new transaction	BEGIN TRANSACTION [transaction_name];
COMMIT	Saves all changes made during the transaction	COMMIT;
ROLLBACK	Undoes all changes made during the transaction	ROLLBACK;
SAVEPOINT	Creates a savepoint within the current transaction	SAVEPOINT savepoint_name;

4. **What is a primary key in SQL?**

It is a unique identifier for each record in a table. It ensures that each row in the table has a distinct and non-null value in the primary key column. Primary keys enforce data integrity and create relationships between tables.

5. **What is a foreign key?**

It is a field in one table referencing the primary key in another. It establishes a relationship between the two tables, ensuring data consistency and enabling data retrieval across tables.Data Abstraction

6. **Explain the difference between DELETE and TRUNCATE commands.**

The DELETE command is used by professionals to remove particular rows from a table based on a condition, allowing you to selectively delete records. TRUNCATE, on the other hand, removes all rows from a table without specifying conditions. TRUNCATE is faster and uses fewer system resources than DELETE but does not log individual row deletions.

7. What is Unique Key ?

A unique key is a single or combination of fields that ensure all values stores in the column will be unique. It means a column cannot stores duplicate values. This key provides uniqueness for the column or set of columns.

8. **What is the difference between a primary key and a unique key?**

The primary key act as a unique identifier for each record in the table.

The unique key is also a unique identifier for records when the primary key is not present in the table.

We cannot store NULL values in the primary key column.

We can store NULL value in the unique key column, but only one NULL is allowed.

We cannot change or delete the primary key column values.

We can modify the unique key column values.

9. What is a constraint in SQL? Name a few.

A constraint in SQL defines rules or restrictions that apply to data in a table, ensuring data integrity. Common constraints include:

- PRIMARY KEY: Ensures the values' uniqueness in a column.
- FOREIGN KEY: Enforces referential integrity between tables.
- UNIQUE: Ensures the uniqueness of values in a column.
- CHECK: Defines a condition that data must meet to be inserted or updated.
- NOT NULL: Ensures that there are no NULL values in a column.

10. **What are aggregate functions?**

Aggregate functions in SQL perform calculations on a set of values and return a single result.

- SUM: To calculate the sum of values in a column.
- COUNT: To count a column's number of rows or non-null values.
- AVG: To calculate the average of values in a column.
- MIN: To retrieve the minimum value in a column.
- MAX: To retrieve the maximum value in a column.

11. Explain different types of joins with examples.

- INNER JOIN: Gathers rows that have matching values in both tables.
- RIGHT JOIN: Gathers all rows from the right table and any matching rows from the left table.
- LEFT JOIN: Gathers all rows from the left table and any matching rows from the right table.
- FULL JOIN: Gathers all rows when there's a match in either table, including unmatched rows from both tables.

12. **What is a subquery?**

A subquery refers to a query that is embedded within another query, serving the purpose of fetching information that will subsequently be employed as a condition or value within the encompassing outer query.

14. **What is the difference between UNION and UNION ALL?**

UNION merges the outcomes of two or more SELECT statements, removing duplicate rows, whereas UNION ALL merges the results without removing duplicates. While UNION ALL is faster, it may include duplicate rows.

15. What are correlated subqueries?

It is a type of subquery that makes reference to columns from the surrounding outer query. This subquery is executed repeatedly, once

for each row being processed by the outer query, and its execution depends on the outcomes of the outer query.

16. **What is Auto Increment?**

Auto-Increment feature that automatically generates a numerical Primary key value for every new record inserted.

17. What is an ALIAS command?

Aliases are the temporary names given to a table or column for the purpose of a particular SQL query

18. What are Union, minus, and Interact commands?

Set Operations in SQL eliminate duplicate tuples and can be applied only to the relations which are union compatible. Set Operations available in SQL are :

- Set Union
- Set Intersection
- Set Difference

UNION Operation: This operation includes all the tuples which are present in either of the relations. For example: To find all the customers who have a loan or an account or both in a bank.

```
SELECT CustomerName FROM Depositor
UNION
```

```
SELECT CustomerName FROM Borrower ;
```

The union operation automatically eliminates duplicates. If all the duplicates are supposed to be retained, UNION ALL is used in place of UNION.

INTERSECT Operation: This operation includes the tuples which are present in both of the relations. For example: To find the customers who have a loan as well as an account in the bank:

```
SELECT CustomerName FROM Depositor
INTERSECT
SELECT CustomerName FROM Borrower ;
```

The Intersect operation automatically eliminates duplicates. If all the duplicates are supposed to be retained, INTERSECT ALL is used in place of INTERSECT.

EXCEPT for Operation: This operation includes tuples that are present in one relationship but should not be present in another relationship. For example: To find customers who have an account but no loan at the bank:

```
SELECT CustomerName FROM Depositor
EXCEPT
SELECT CustomerName FROM Borrower ;
```

The Except operation automatically eliminates the duplicates. If all the duplicates are supposed to be retained, EXCEPT ALL is used in place of EXCEPT.

19. What is the difference between BETWEEN and IN operators in SQL?

BETWEEN: The BETWEEN operator is used to fetch rows based on a range of values.

```
SELECT * FROM Students
WHERE ROLL_NO BETWEEN 20 AND 30;
```

This query will select all those rows from the table. Students where the value of the field ROLL_NO lies between 20 and 30.

IN: The IN operator is used to check for values contained in specific sets.

```
SELECT * FROM Students
WHERE ROLL_NO IN (20,21,23);
```

This query will select all those rows from the table Students where the value of the field ROLL_NO is either 20 or 21 or 23.

20. What is pattern matching ?

SQL pattern matching provides for pattern search in data if you have no clue as to what that word should be. This kind of SQL query uses wildcards to match a string pattern, rather than writing the exact word. The LIKE operator is used in conjunction with **SQL Wildcards** to fetch the required information.

- **Using the % wildcard to perform a simple search**

The % wildcard matches zero or more characters of any type and can be used to define wildcards both before and after the pattern. Search a student in your database with first name beginning with the letter K:

```
SELECT *
FROM students
WHERE first_name LIKE 'K%'
```

- **Omitting the patterns using the NOT keyword**

Use the NOT keyword to select records that don't match the pattern. This query returns all students whose first name does not begin with K.

```
SELECT *
FROM students
WHERE first_name NOT LIKE 'K%'
```

- **Matching a pattern anywhere using the % wildcard twice**

Search for a student in the database where he/she has a K in his/her first name.

```
SELECT *
FROM students
WHERE first_name LIKE '%Q%'
```

- **Using the _ wildcard to match pattern at a specific position**

The _ wildcard matches exactly one character of any type. It can be used in conjunction with % wildcard. This query fetches all students with letter K at the third position in their first name.

```
SELECT *
FROM students
WHERE first_name LIKE '__K%'
```

- **Matching patterns for a specific length**

The _ wildcard plays an important role as a limitation when it matches exactly one character. It limits the length and position of the matched results. For example -

```
SELECT *   /* Matches first names with three or more letters */
FROM students
WHERE first_name LIKE '___%'
```

```
SELECT *   /* Matches first names with exactly four characters */
FROM students
WHERE first_name LIKE '____'
```

21. Explain Normalization

Normalization represents the way of organizing structured data in the database efficiently. It includes the creation of tables, establishing relationships between them, and defining rules for those relationships. Inconsistency and redundancy can be kept in check based on these rules, hence, adding flexibility to the database.

Denormalization -

Denormalization is the inverse process of normalization, where the normalized schema is converted into a schema that has redundant information. The performance is improved by using redundancy and keeping the redundant data consistent. The reason for performing denormalization is the overheads produced in the query processor by an over-normalized structure.

various forms of Normalization -

Normal Forms are used to eliminate or reduce redundancy in database tables. The different forms are as follows:

- **First Normal Form:**
  A relation is in first normal form if every attribute in that relation is a **single-valued attribute**. If a relation contains a composite or multi-valued attribute, it violates the first normal form. Let's consider the following **students** table. Each student in the table, has a name, his/her address, and the books they issued from the public library –

Students Table

Student	Address	Books Issued	Salutation
Sara	Amanora Park Town 94	Until the Day I Die (Emily Carpenter), Inception (Christopher Nolan)	Ms.
Ansh	62nd Sector A-10	The Alchemist (Paulo Coelho), Inferno (Dan Brown)	Mr.
Sara	24th Street Park Avenue	Beautiful Bad (Annie Ward), Woman 99 (Greer Macallister)	Mrs.

Student	Address	Books Issued	Salutation
Ansh	Windsor Street 777	Dracula (Bram Stoker)	Mr.

As we can observe, the Books Issued field has more than one value per record, and to convert it into 1NF, this has to be resolved into separate individual records for each book issued. Check the following table in 1NF form –

Students Table (1st Normal Form)

Student	Address	Books Issued	Salutation
Sara	Amanora Park Town 94	Until the Day I Die (Emily Carpenter)	Ms.
Sara	Amanora Park Town 94	Inception (Christopher Nolan)	Ms.
Ansh	62nd Sector A-10	The Alchemist (Paulo Coelho)	Mr.
Ansh	62nd Sector A-10	Inferno (Dan Brown)	Mr.
Sara	24th Street Park Avenue	Beautiful Bad (Annie Ward)	Mrs.
Sara	24th Street Park Avenue	Woman 99 (Greer Macallister)	Mrs.
Ansh	Windsor Street 777	Dracula (Bram Stoker)	Mr.

- **Second Normal Form:**

A relation is in second normal form if it satisfies the conditions for the first normal form and does not contain any partial dependency. A relation in 2NF has **no partial dependency**, i.e., it has no non-prime attribute that depends on any proper subset of any candidate key of the table. Often, specifying a single column Primary Key is the solution to the problem. Examples –

Example 1 - Consider the above example. As we can observe, the Students Table in the 1NF form has a candidate key in the form of [Student, Address] that can uniquely identify all records in the table. The field Books Issued (non-prime attribute) depends partially on the Student field. Hence, the table is not in 2NF. To convert it into the 2nd Normal Form, we will partition the tables into two while specifying a new ***Primary Key*** attribute to identify the individual records in the Students table. The ***Foreign Key*** constraint will be set on the other table to ensure referential integrity.

Students Table (2nd Normal Form)

Student_ID	Student	Address	Salutation
1	Sara	Amanora Park Town 94	Ms.
2	Ansh	62nd Sector A-10	Mr.
3	Sara	24th Street Park Avenue	Mrs.
4	Ansh	Windsor Street 777	Mr.

Books Table (2nd Normal Form)

Student_ID	Book Issued
1	Until the Day I Die (Emily Carpenter)
1	Inception (Christopher Nolan)
2	The Alchemist (Paulo Coelho)
2	Inferno (Dan Brown)
3	Beautiful Bad (Annie Ward)
3	Woman 99 (Greer Macallister)
4	Dracula (Bram Stoker)

Example 2 - Consider the following dependencies in relation to R(W,X,Y,Z)

WX -> Y [W **and** X together determine Y]

XY -> Z [X **and** Y together determine Z]

Here, WX is the only candidate key and there is no partial dependency, i.e., any proper subset of WX doesn't determine any non-prime attribute in the relation.

- **Third Normal Form**

A relation is said to be in the third normal form, if it satisfies the conditions for the second normal form and there is **no transitive dependency** between the non-prime attributes, i.e., all non-prime attributes are determined only by the candidate keys of the relation and not by any other non-prime attribute.

Example 1 - Consider the Students Table in the above example. As we can observe, the Students Table in the 2NF form has a single

candidate key Student_ID (primary key) that can uniquely identify all records in the table. The field Salutation (non-prime attribute), however, depends on the Student Field rather than the candidate key. Hence, the table is not in 3NF. To convert it into the 3rd Normal Form, we will once again partition the tables into two while specifying a new ***Foreign Key*** constraint to identify the salutations for individual records in the Students table. The ***Primary Key*** constraint for the same will be set on the Salutations table to identify each record uniquely.

Students Table (3rd Normal Form)

Student_ID	Student	Address	Salutation_ID
1	Sara	Amanora Park Town 94	1
2	Ansh	62nd Sector A-10	2
3	Sara	24th Street Park Avenue	3
4	Ansh	Windsor Street 777	1

Books Table (3rd Normal Form)

Student_ID	Book Issued
1	Until the Day I Die (Emily Carpenter)
1	Inception (Christopher Nolan)
2	The Alchemist (Paulo Coelho)
2	Inferno (Dan Brown)
3	Beautiful Bad (Annie Ward)

Student_ID	Book Issued
3	Woman 99 (Greer Macallister)
4	Dracula (Bram Stoker)

Salutations Table (3rd Normal Form)

Salutation_ID	Salutation
1	Ms.
2	Mr.
3	Mrs.

Example 2 - Consider the following dependencies in relation to R(P,Q,R,S,T)

P -> QR [P together determine C]

RS -> T [B **and** C together determine D]

Q -> S

T -> P

For the above relation to exist in 3NF, all possible candidate keys in the above relation should be {P, RS, QR, T}.

- **Boyce-Codd Normal Form**

A relation is in Boyce-Codd Normal Form if satisfies the conditions for third normal form and for every functional dependency, Left-Hand-Side is super key. In other words, a relation in BCNF has non-trivial functional dependencies in form X –> Y, such that X is always a super key. For example - In the above example, Student_ID serves as the sole unique identifier for the Students Table and Salutation_ID for the

Salutations Table, thus these tables exist in BCNF. The same cannot be said for the Books Table and there can be several books with common Book Names and the same Student_ID.

22. What are Entity and Relationships ?

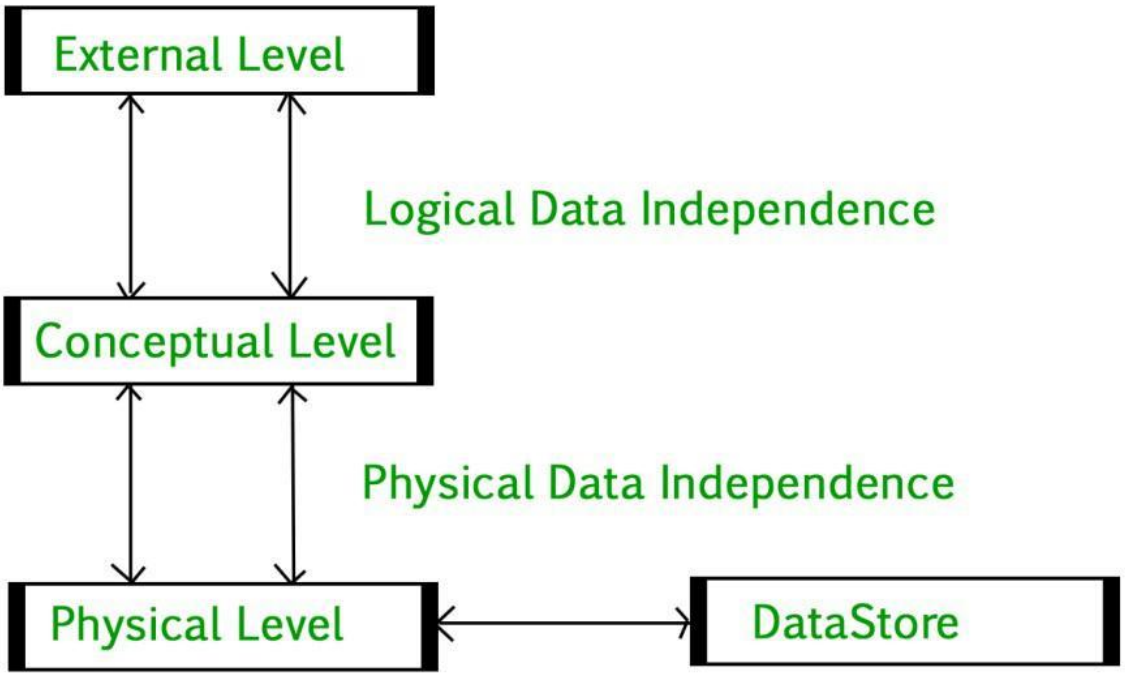
Entity: An entity can be a real-world object, either tangible or intangible, that can be easily identifiable. For example, in a college database, students, professors, workers, departments, and projects can be referred to as entities. Each entity has some associated properties that provide it an identity.

Relationships: Relations or links between entities that have something to do with each other. For example - The employee's table in a company's database can be associated with the salary table in the same database.

23. Explain 3 Tier Architecture ?

The 3-tier architecture is a commonly used architectural approach in Database Management Systems (DBMSs) for the design and development of applications that work with databases. The 3-tier architecture divides an application’s components into three tiers or layers. Each layer has its own set of responsibilities.

DBMS 3-Tier architecture divides the complete system into three inter-related but independent modules as shown below:

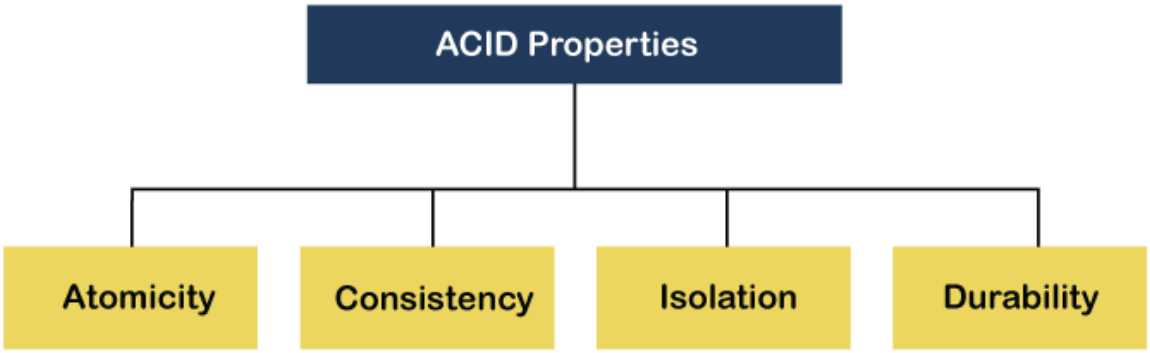


23.Explain ACID?

DBMS is the management of data that should remain integrated when any changes are done in it. It is because if the integrity of the data is affected, whole data will get disturbed and corrupted. Therefore, to maintain the integrity of the data, there are four properties described in the database management system, which are known as the **ACID** properties.

ACID Properties

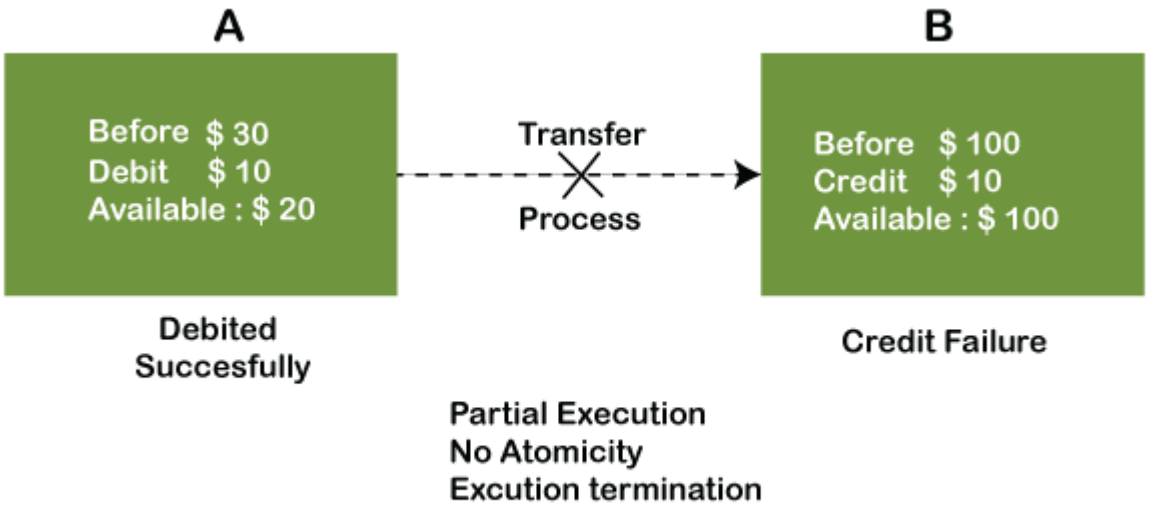
The expansion of the term ACID defines for:



1) Atomicity

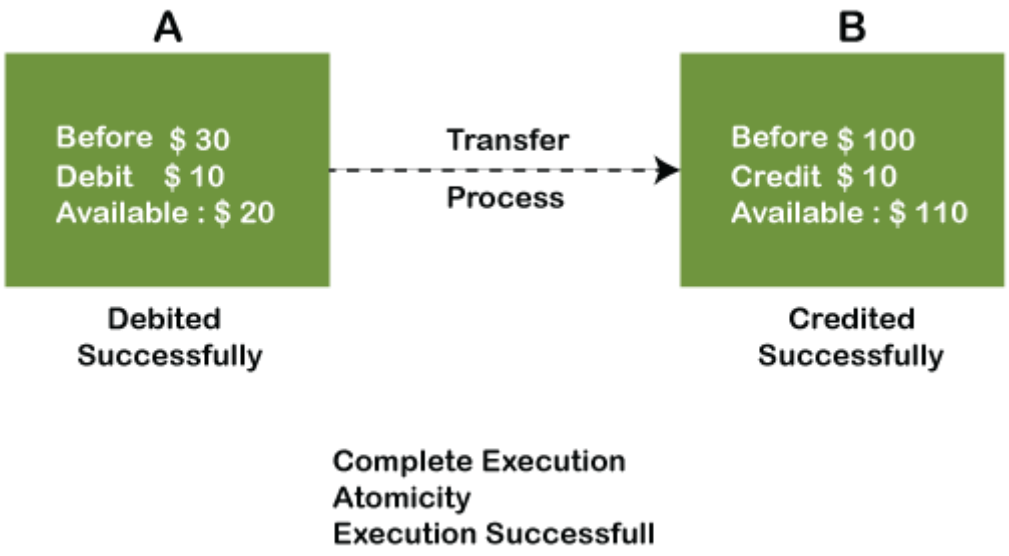
The term atomicity defines that the data remains atomic. It means if any operation is performed on the data, either it should be performed or executed completely or should not be executed at all. It further means that the operation should not break in between or execute partially. In the case of executing operations on the transaction, the operation should be completely executed and not partially.

Example: If Remo has account A having $30 in his account from which he wishes to send $10 to Sheero's account, which is B. In account B, a sum of $ 100 is already present. When $10 will be transferred to account B, the sum will become $110. Now, there will be two operations that will take place. One is the amount of $10 that Remo wants to transfer will be debited from his account A, and the same amount will get credited to account B, i.e., into Sheero's account. Now, what happens - the first operation of debit executes successfully, but the credit operation, however, fails. Thus, in Remo's account A, the value becomes $20, and to that of Sheero's account, it remains $100 as it was previously present.



In the above diagram, it can be seen that after crediting $10, the amount is still $100 in account B. So, it is not an atomic transaction.

The below image shows that both debit and credit operations are done successfully. Thus the transaction is atomic.



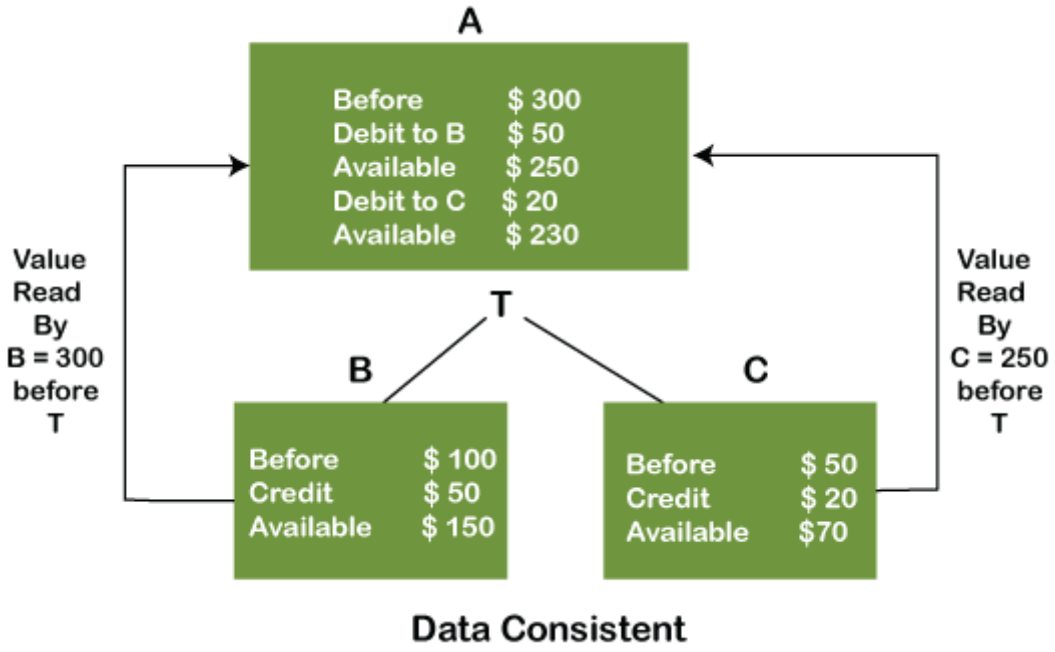
Thus, when the amount loses atomicity, then in the bank systems, this becomes a huge issue, and so the atomicity is the main focus in the bank systems.

2) Consistency

The word **consistency** means that the value should remain preserved always. In DBMS, the integrity of the data should be maintained, which means if a change in the database is made, it should remain preserved always. In the case of transactions, the integrity of the data is very essential so that the database remains consistent before and after the transaction. The data should always be correct.

Example:

Advertisement



Data Consistent

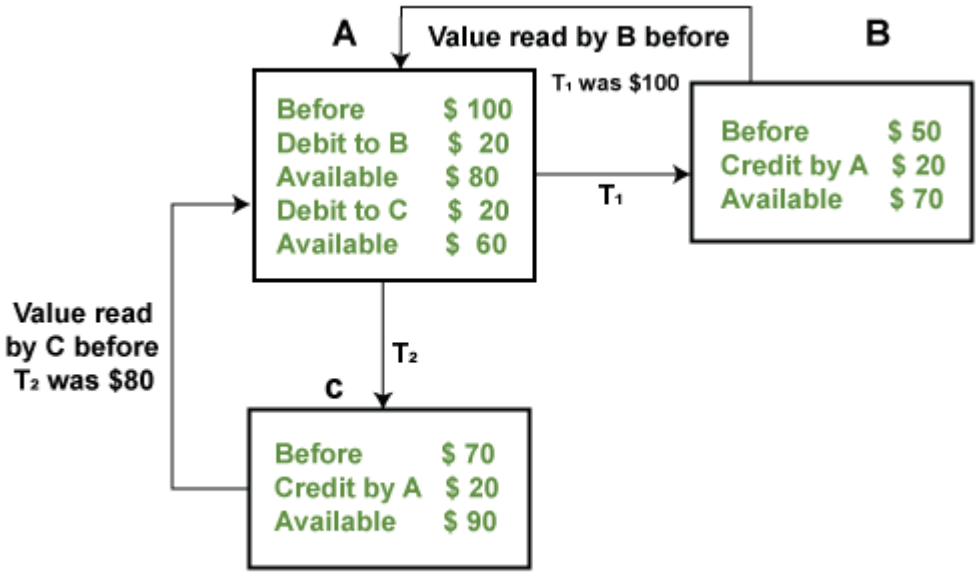
In the above figure, there are three accounts, A, B, and C, where A is making a transaction T one by one to both B & C. There are two operations that take place, i.e., Debit and Credit. Account A firstly debits $50 to account B, and the amount in account A is read $300 by B before the transaction. After the successful transaction T, the available amount in B becomes $150. Now, A debits $20 to account C, and that time, the value read by C is $250 (that is correct as a debit of $50 has been successfully done to B). The debit and credit operation from account A to C has been done successfully. We can see that the

transaction is done successfully, and the value is also read correctly. Thus, the data is consistent. In case the value read by B and C is $300, which means that data is inconsistent because when the debit operation executes, it will not be consistent.

3) Isolation

The term 'isolation' means separation. In DBMS, Isolation is the property of a database where no data should affect the other one and may occur concurrently. In short, the operation on one database should begin when the operation on the first database gets complete. It means if two operations are being performed on two different databases, they may not affect the value of one another. In the case of transactions, when two or more transactions occur simultaneously, the consistency should remain maintained. Any changes that occur in any particular transaction will not be seen by other transactions until the change is not committed in the memory.

Example: If two operations are concurrently running on two different accounts, then the value of both accounts should not get affected. The value should remain persistent. As you can see in the below diagram, account A is making T1 and T2 transactions to account B and C, but both are executing independently without affecting each other. It is known as Isolation.



Isolation - Independent execution of T_1 & T_2 by A

4) Durability

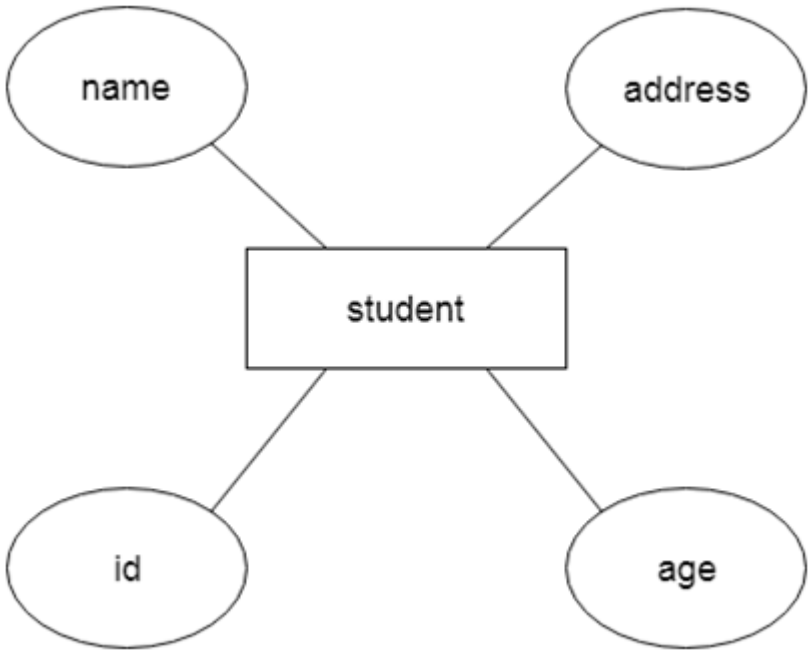
Durability ensures the permanency of something. In DBMS, the term durability ensures that the data after the successful execution of the operation becomes permanent in the database. The durability of the data should be so perfect that even if the system fails or leads to a crash, the database still survives. However, if gets lost, it becomes the responsibility of the recovery manager for ensuring the durability of the database. For committing the values, the COMMIT command must be used every time we make changes.

Therefore, the ACID property of DBMS plays a vital role in maintaining the consistency and availability of data in the database.

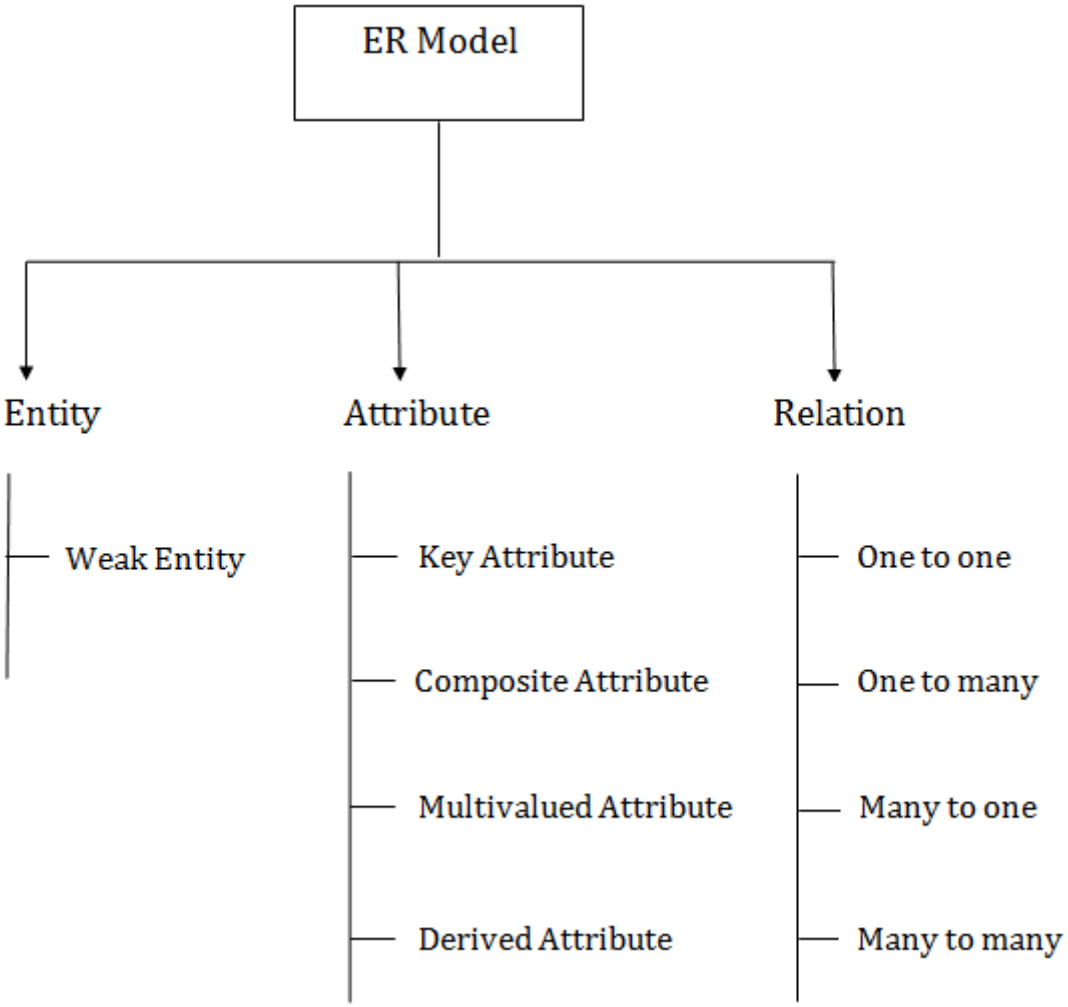
24. Explain ER (Entity Relationship) Diagram in DBMS

- ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

For example, Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc. The address can be another entity with attributes like city, street name, pin code, etc and there will be a relationship between them.



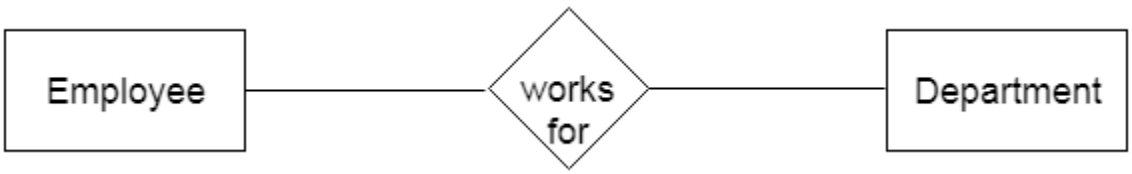
Component of ER Diagram



1. Entity:

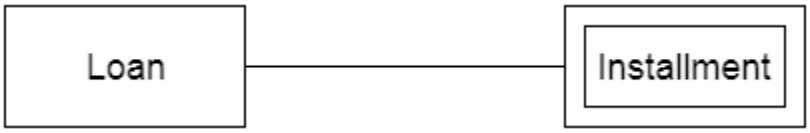
An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



a. Weak Entity

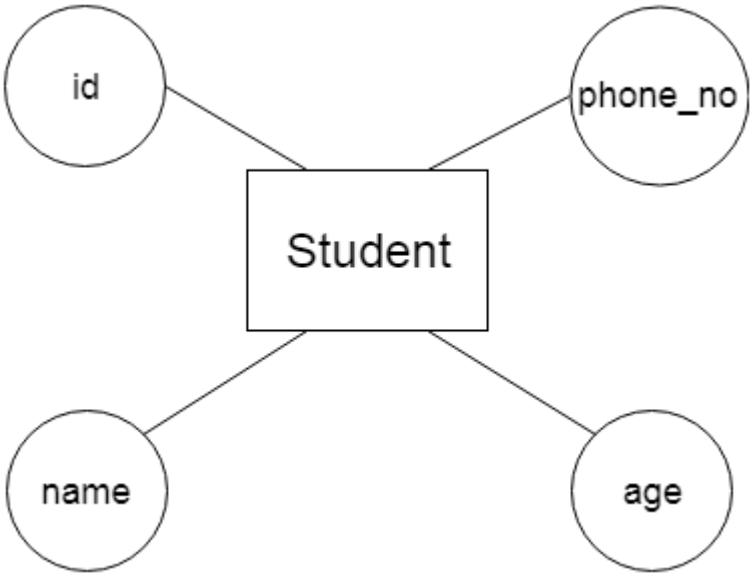
An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



2. Attribute

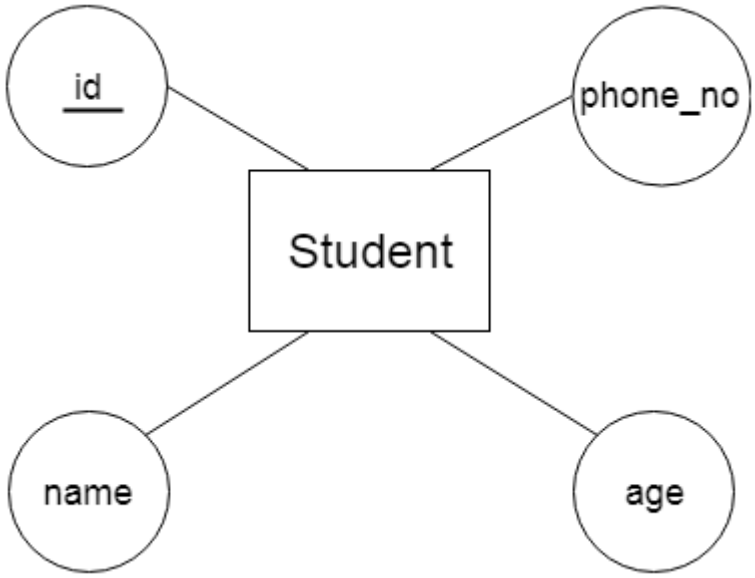
The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

For example, id, age, contact number, name, etc. can be attributes of a student.



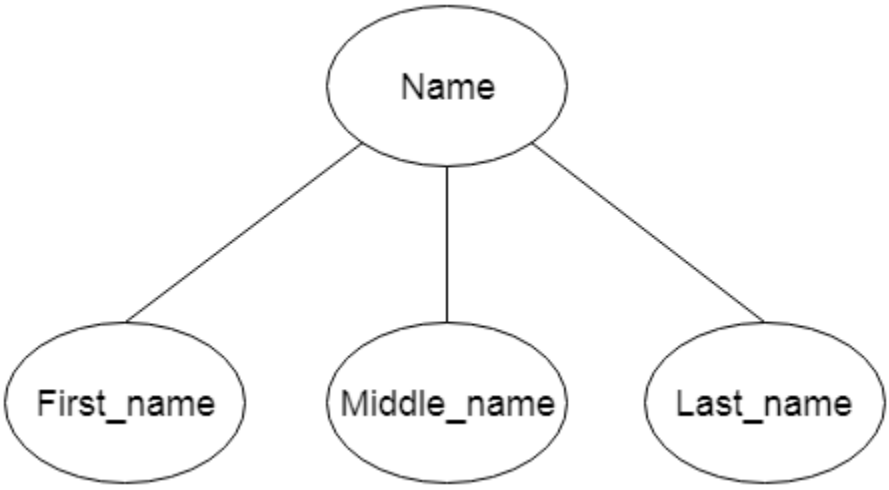
a. Key Attribute

The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.



b. Composite Attribute

An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



c. Multivalued Attribute

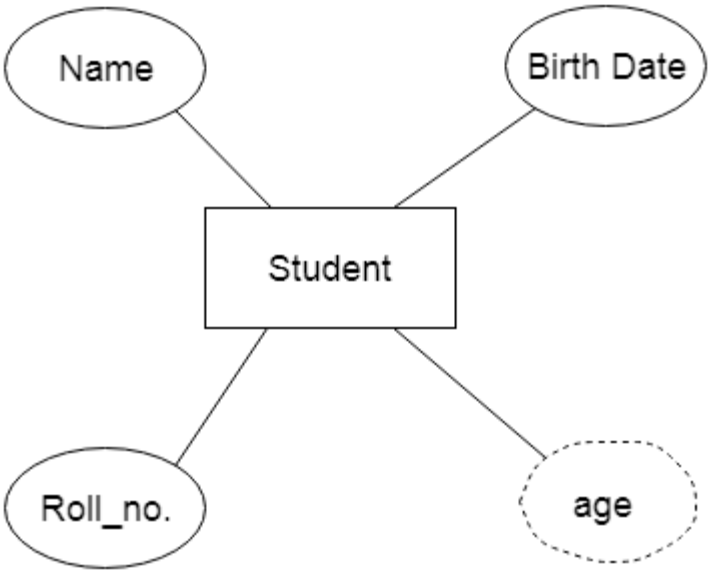
An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

For example, a student can have more than one phone number.

d. Derived Attribute

An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

For example, A person's age changes over time and can be derived from another attribute like Date of birth.



3. Relationship

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

Types of relationship are as follows:

a. One-to-One Relationship

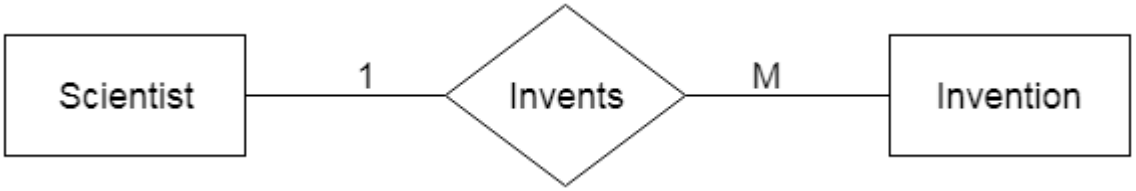
When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

For example, A female can marry to one male, and a male can marry to one female.

b. One-to-many relationship

When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

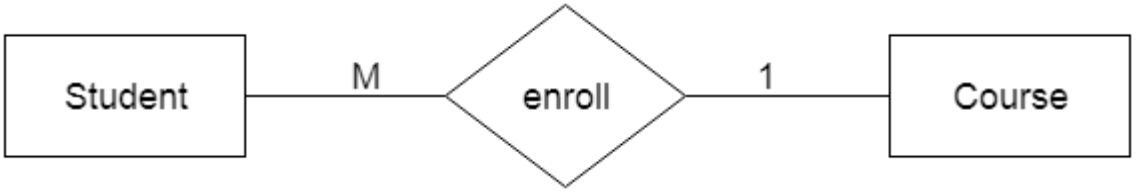
For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.



c. Many-to-one relationship

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

For example, Student enrolls for only one course, but a course can have many students.



d. Many-to-many relationship

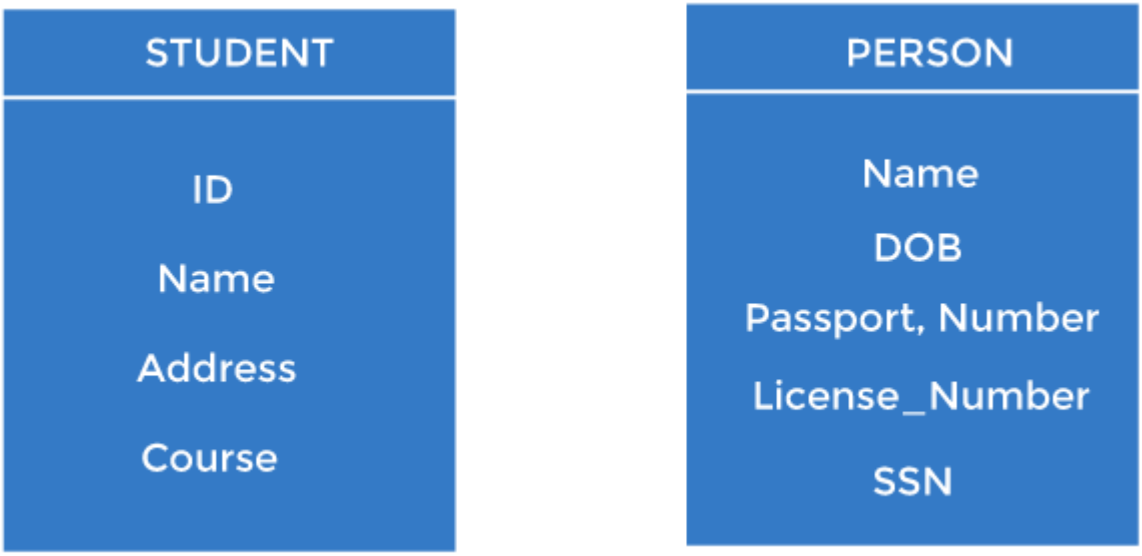
When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

For example, Employee can assign by many projects and project can have many employees.

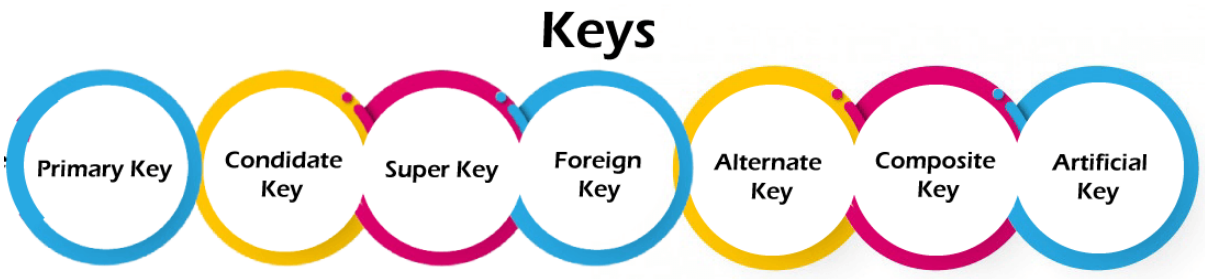
25. Explain Keys?

- Keys play an important role in the relational database.
- It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.

For example, ID is used as a key in the Student table because it is unique for each student. In the PERSON table, passport_number, license_number, SSN are keys since they are unique for each person.

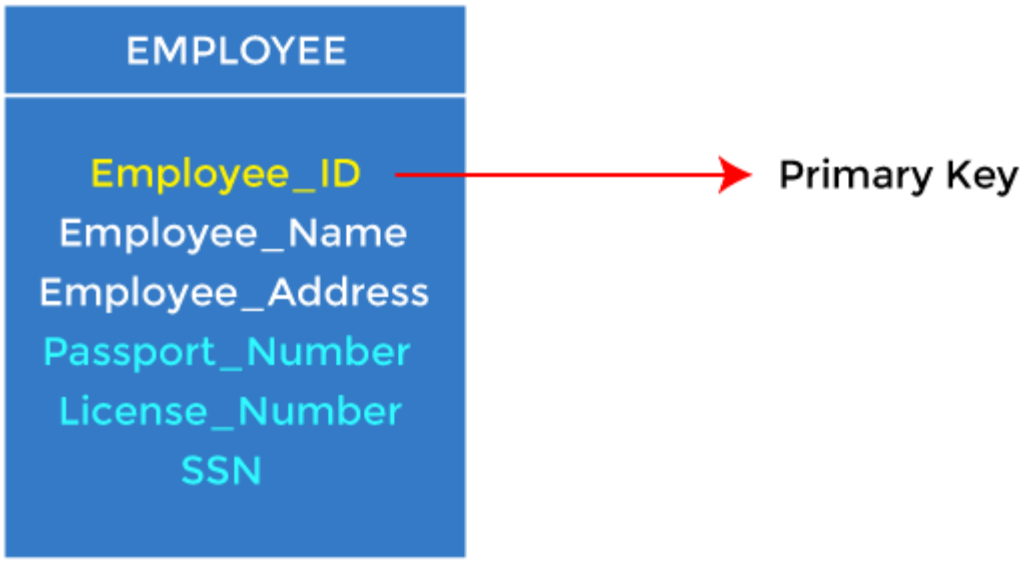


Types of keys:



1. Primary key

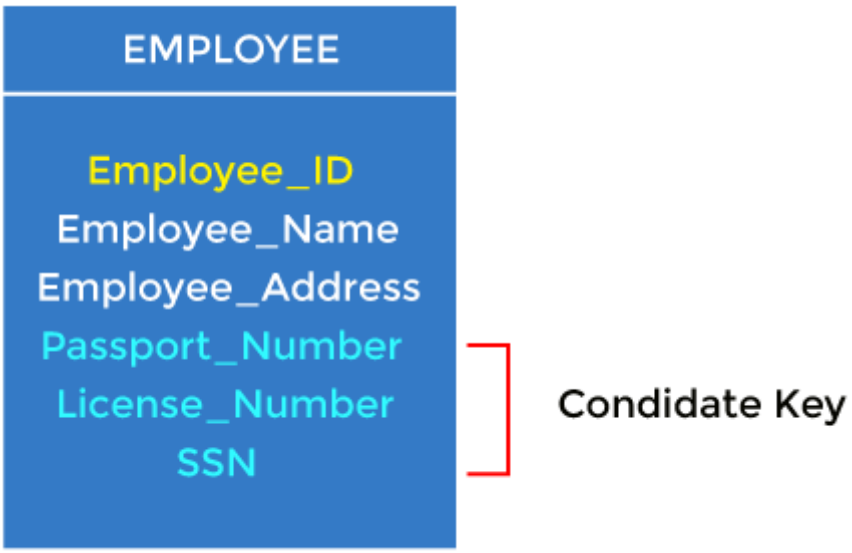
- It is the first key used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys, as we saw in the PERSON table. The key which is most suitable from those lists becomes a primary key.
- In the EMPLOYEE table, ID can be the primary key since it is unique for each employee. In the EMPLOYEE table, we can even select License_Number and Passport_Number as primary keys since they are also unique.
- For each entity, the primary key selection is based on requirements and developers.



2. Candidate key

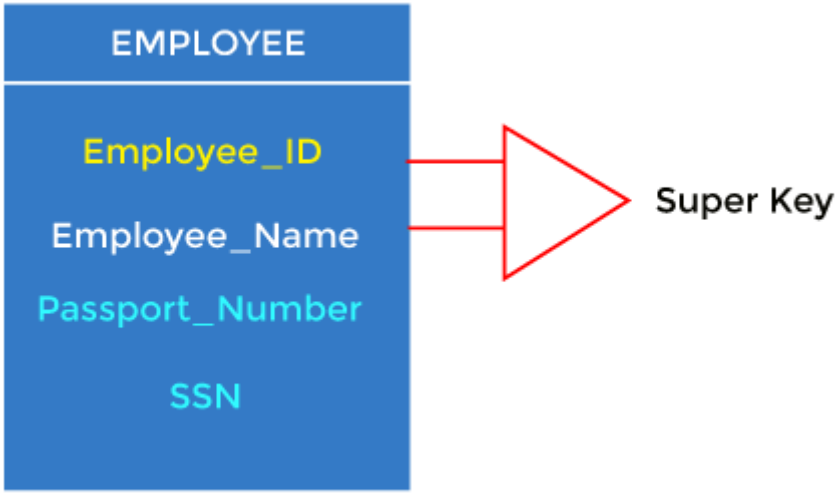
- A candidate key is an attribute or set of attributes that can uniquely identify a tuple.
- Except for the primary key, the remaining attributes are considered a candidate key. The candidate keys are as strong as the primary key.

For example: In the EMPLOYEE table, id is best suited for the primary key. The rest of the attributes, like SSN, Passport_Number, License_Number, etc., are considered a candidate key.



3. Super Key

Super key is an attribute set that can uniquely identify a tuple. A super key is a superset of a candidate key.



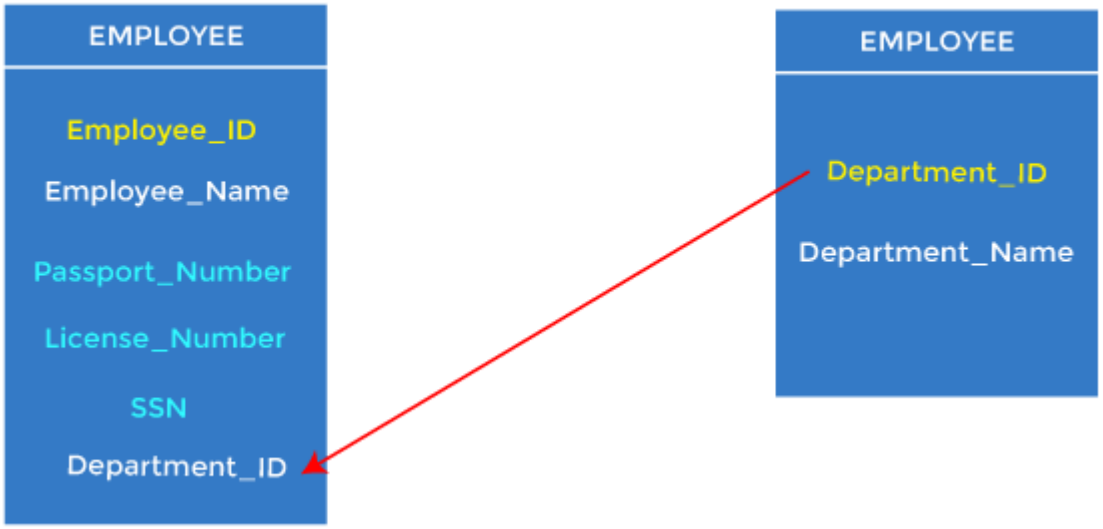
For example: In the above EMPLOYEE table, for(EMPLOEE_ID, EMPLOYEE_NAME), the name of two employees can be the same, but their EMPLYEE_ID can't be the same. Hence, this combination can also be a key.

The super key would be EMPLOYEE-ID (EMPLOYEE_ID, EMPLOYEE-NAME), etc.

4. Foreign key

- Foreign keys are the column of the table used to point to the primary key of another table.
- Every employee works in a specific department in a company, and employee and department are two different entities. So we can't store the department's information in the employee table. That's why we link these two tables through the primary key of one table.
- We add the primary key of the DEPARTMENT table, Department_Id, as a new attribute in the EMPLOYEE table.

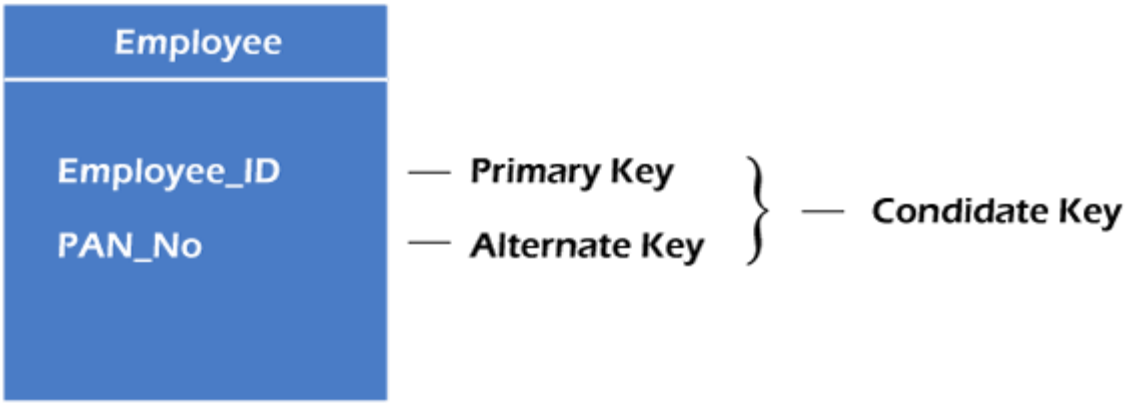
- In the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.



5. Alternate key

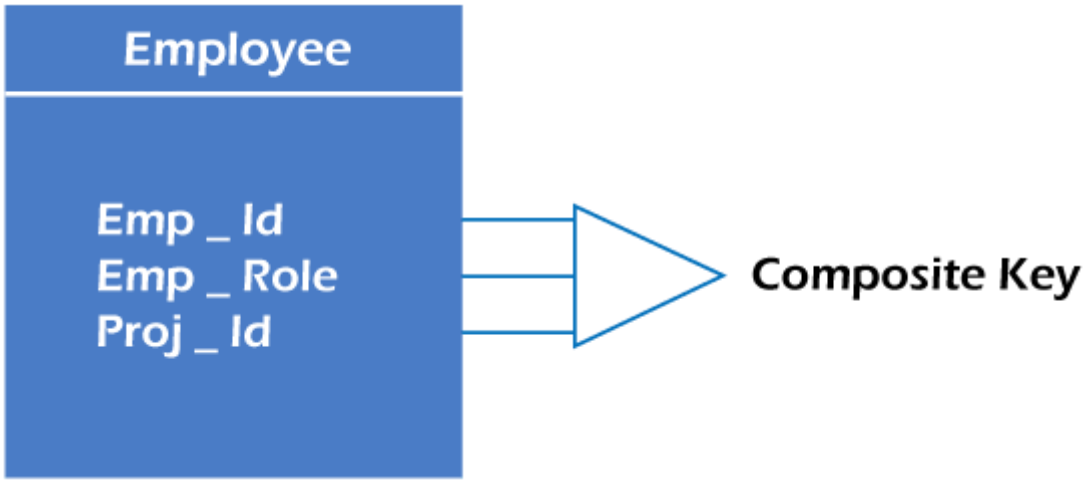
There may be one or more attributes or a combination of attributes that uniquely identify each tuple in a relation. These attributes or combinations of the attributes are called the candidate keys. One key is chosen as the primary key from these candidate keys, and the remaining candidate key, if it exists, is termed the alternate key. In other words, the total number of the alternate keys is the total number of candidate keys minus the primary key. The alternate key may or may not exist. If there is only one candidate key in a relation, it does not have an alternate key.

For example, employee relation has two attributes, Employee_Id and PAN_No, that act as candidate keys. In this relation, Employee_Id is chosen as the primary key, so the other candidate key, PAN_No, acts as the Alternate key.

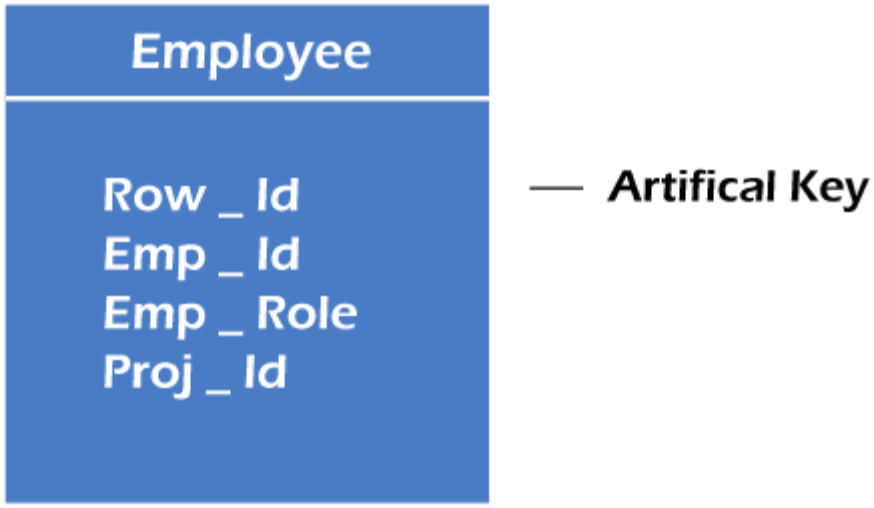


6. Composite key

Whenever a primary key consists of more than one attribute, it is known as a composite key. This key is also known as Concatenated Key.



For example, in employee relations, we assume that an employee may be assigned multiple roles, and an employee may work on multiple projects simultaneously. So the primary key will be composed of all three attributes, namely Emp_ID, Emp_role, and Proj_ID in combination. So these attributes act as a composite key since the primary key comprises more than one attribute.



7. Artificial key

The key created using arbitrarily assigned data are known as artificial keys. These keys are created when a primary key is large and complex and has no relationship with many other relations. The data values of the artificial keys are usually numbered in a serial order.

For example, the primary key, which is composed of Emp_ID, Emp_role, and Proj_ID, is large in employee relations. So it would be better to add a new virtual attribute to identify each tuple in the relation uniquely.

26. **Explain character-manipulation functions? Explains its different types in SQL.**

Character-manipulation functions are used to change, extract, and alter the character string. When one or more characters and words are passed into the function, the function will perform its operation on those input strings and return the result.

The following are the character manipulation functions in SQL:

A) CONCAT: This function is used to join two or more values together. It always appends the second string into the end of the first string. For example:

Input: SELECT CONCAT ('Information-', 'technology') FROM DUAL;

Output: Information-technology

B) SUBSTR: It is used to return the portion of the string from a specified start point to an endpoint. For example:

Input: SELECT SUBSTR ('Database Management System', 9, 11) FROM DUAL;

Output: Management

C) LENGTH: This function returns the string's length in numerical value, including the blank spaces. For example:

Input: SELECT LENGTH ('Hello Javatpoint') FROM DUAL;

Output: 16

D) INSTR: This function finds the exact numeric position of a specified character or word in a given string. For example:

Input: SELECT INSTR ('Hello Javatpoint', 'Javatpoint');

Output: 7

E) LPAD: It returns the padding of the left-side character value for right-justified value. For example:

Input: SELECT LPAD ('200', 6,'*');

Output: ***200

F) RPAD: It returns the padding of the right-side character value for left-justified value. For example:

Input: SELECT RPAD ('200', 6,'*');

Output: 200***

G) TRIM: This function is used to remove all the defined characters from the beginning, end, or both. It also trimmed extra spaces. For example:

Input: SELECT TRIM ('A' FROM 'ABCDCBA');

Output: BCDCB

H) REPLACE: This function is used to replace all occurrences of a word or portion of the string (substring) with the other specified string value. For example:

Input: SELECT REPLACE ('It is the best coffee at the famous coffee shop.', 'coffee', 'tea');

Output: It is the best tea at the famous tea shop.

27. What is the difference between the RANK() and DENSE_RANK() functions?

The **RANK** function determines the rank for each row within your ordered partition in the result set. If the two rows are assigned the same rank, then the next number in the ranking will be its previous rank plus a number of duplicate numbers. For example, if we have three records at rank 4, the next rank listed would be ranked 7.

The **DENSE_RANK** function assigns a unique rank for each row within a partition as per the specified column value without any gaps. It always specifies ranking in consecutive order. If the two rows are assigned the same rank, this function will assign it with the same rank, and the next rank being the next sequential number. For example, if we have 3 records at rank 4, the next rank listed would be ranked 5.

28. What is a trigger?

The trigger is a statement that a system executes automatically when there is any modification to the database. In a trigger, we first specify when the trigger is to be executed and then the action to be performed when the trigger executes. Triggers are used to specify certain integrity constraints and referential constraints that cannot be specified using the constraint mechanism of SQL.

29. What is the difference between SQL DELETE and SQL TRUNCATE commands?

SQL DELETE	SQL TRUNCATE
The DELETE statement removes rows one at a time and records an entry in the transaction log for each deleted row.	TRUNCATE TABLE removes the data by deallocating the data pages used to store the table data and records only the page deallocations in the transaction log.
DELETE command is slower than the identityTRUNCATE command.	While the TRUNCATE command is faster than the DELETE command.
To use Delete you need DELETE permission on the table.	To use Truncate on a table we need at least ALTER permission on the table.

SQL DELETE	SQL TRUNCATE
The identity of the column retains the identity after using DELETE Statement on the table.	The identity of the column is reset to its seed value if the table contains an identity column.
The delete can be used with indexed views.	Truncate cannot be used with indexed views.

30. Why do we use Commit and Rollback commands?

COMMIT	ROLLBACK
COMMIT permanently saves the changes made by the current transaction.	**ROLLBACK undo the changes made by the current transaction.**
The transaction can not undo changes after COMMIT execution.	**Transaction reaches its previous state after ROLLBACK.**
When the transaction is successful, COMMIT is applied.	**When the transaction is aborted, ROLLBACK occurs.**

31. Are NULL values the same as zero or a blank space?

In SQL, zero or blank space can be compared with another zero or blank space. whereas one null may not be equal to another null. null means data might not be provided or there is no data.

32. What is an Index? Explain its different types.

A database index is a data structure that provides a quick lookup of data in a column or columns of a table. It enhances the speed of operations accessing data from a database table at the cost of additional writes and memory to maintain the index data structure.

CREATE INDEX index_name /* Create Index */

ON table_name (column_1, column_2);

DROP INDEX index_name; /* Drop Index */

There are different types of indexes that can be created for different purposes:

- **Unique and Non-Unique Index:**

Unique indexes are indexes that help maintain data integrity by ensuring that no two rows of data in a table have identical key values. Once a unique index has been defined for a table, uniqueness is enforced whenever keys are added or changed within the index.

CREATE UNIQUE INDEX myIndex

ON students (enroll_no);

Non-unique indexes, on the other hand, are not used to enforce constraints on the tables with which they are associated. Instead, non-unique indexes are used solely to improve query performance by maintaining a sorted order of data values that are used frequently.

- **Clustered and Non-Clustered Index:**

Clustered indexes are indexes whose order of the rows in the database corresponds to the order of the rows in the index. This is why only one clustered index can exist in a given table, whereas, multiple non-clustered indexes can exist in the table.

The only difference between clustered and non-clustered indexes is that the database manager attempts to keep the data in the database in the same order as the corresponding keys appear in the clustered index.

Clustering indexes can improve the performance of most query operations because they provide a linear-access path to data stored in the database.

Write a SQL statement to create a UNIQUE INDEX "my_index" on "my_table" for fields "column_1" & "column_2".

33. What is a Subquery? What are its types?

A subquery is a query within another query, also known as a nested query or inner query. It is used to restrict or enhance the data to be queried by the main query, thus restricting or enhancing the output of the main query respectively. For example, here we fetch the contact information for students who have enrolled for the maths

```
SELECT name, email, mob, address
FROM myDb.contacts
WHERE roll_no IN (
 SELECT roll_no
 FROM myDb.students
 WHERE subject = 'Maths');
```

There are two types of subquery **- Correlated and Non-Correlated.**

- **A correlated subquery** cannot be considered as an independent query, but it can refer to the column in a table listed in the FROM of the main query.
- **A non-correlated subquery** can be considered as an independent query and the output of the subquery is substituted in the main query.

Reference :