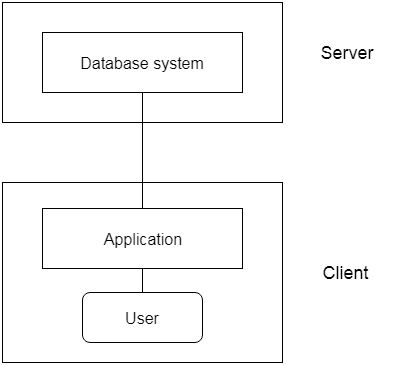
1-Tier Architecture

* In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and uses it.
* Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.
* The 1-Tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response.

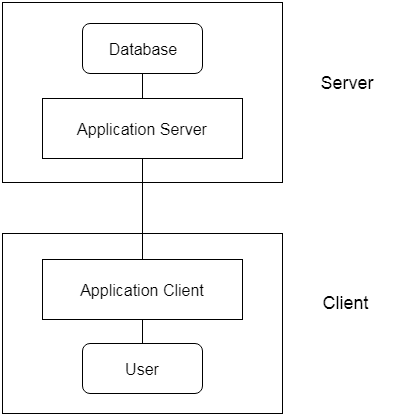
2-Tier Architecture

* The 2-Tier architecture is same as basic client-server. In the two-tier architecture, applications on the client end can directly communicate with the database at the server side. For this interaction, API's like: **ODBC**, **JDBC** are used.
* The user interfaces and application programs are run on the client-side.
* The server side is responsible to provide the functionalities like: query processing and transaction management.
* To communicate with the DBMS, client-side application establishes a connection with the server side.  
  

**Fig: 2-tier Architecture**

3-Tier Architecture

* The 3-Tier architecture contains another layer between the client and server. In this architecture, client can't directly communicate with the server.
* The application on the client-end interacts with an application server which further communicates with the database system.
* End user has no idea about the existence of the database beyond the application server. The database also has no idea about any other user beyond the application.
* The 3-Tier architecture is used in case of large web application.



**Fig: 3-tier Architecture**

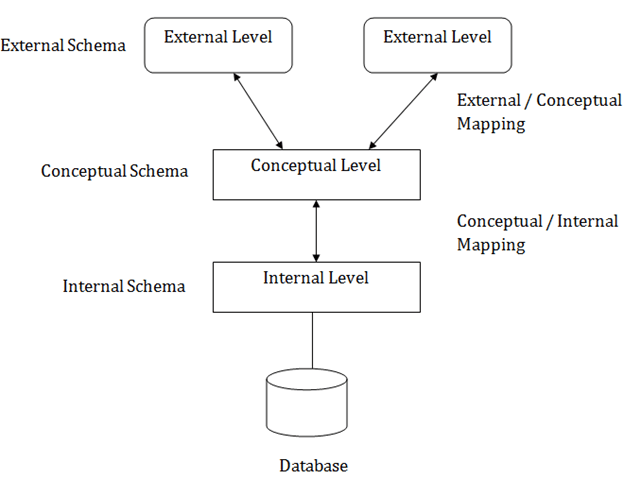
# **Three schema Architecture**

ADVERTISEMENT

ADVERTISEMENT

* The three schema architecture is also called ANSI/SPARC architecture or three-level architecture.
* This framework is used to describe the structure of a specific database system.
* The three schema architecture is also used to separate the user applications and physical database.
* The three schema architecture contains three-levels. It breaks the database down into three different categories.

**The three-schema architecture is as follows:**



**In the above diagram:**

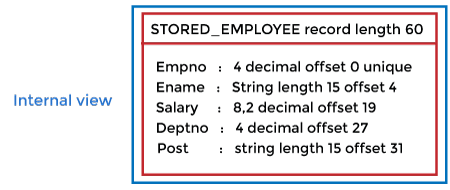
* It shows the DBMS architecture.
* Mapping is used to transform the request and response between various database levels of architecture.
* Mapping is not good for small DBMS because it takes more time.
* In External / Conceptual mapping, it is necessary to transform the request from external level to conceptual schema.
* In Conceptual / Internal mapping, DBMS transform the request from the conceptual to internal level.

## **Objectives of Three schema Architecture**

The main objective of three level architecture is to enable multiple users to access the same data with a personalized view while storing the underlying data only once. Thus it separates the user's view from the physical structure of the database. This separation is desirable for the following reasons:

* Different users need different views of the same data.
* The approach in which a particular user needs to see the data may change over time.
* The users of the database should not worry about the physical implementation and internal workings of the database such as data compression and encryption techniques, hashing, optimization of the internal structures etc.
* All users should be able to access the same data according to their requirements.
* DBA should be able to change the conceptual structure of the database without affecting the user's
* Internal structure of the database should be unaffected by changes to physical aspects of the storage.

### 1. Internal Level

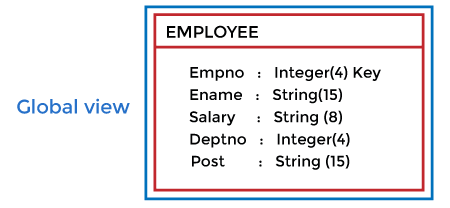


* The internal level has an internal schema which describes the physical storage structure of the database.
* The internal schema is also known as a physical schema.
* It uses the physical data model. It is used to define that how the data will be stored in a block.
* The physical level is used to describe complex low-level data structures in detail.

The internal level is generally is concerned with the following activities:

* Storage space allocations.  
  **For Example:** B-Trees, Hashing etc.
* Access paths.  
  **For Example:** Specification of primary and secondary keys, indexes, pointers and sequencing.
* Data compression and encryption techniques.
* Optimization of internal structures.
* Representation of stored fields.

### 2. Conceptual Level



* The conceptual schema describes the design of a database at the conceptual level. Conceptual level is also known as logical level.
* The conceptual schema describes the structure of the whole database.
* The conceptual level describes what data are to be stored in the database and also describes what relationship exists among those data.
* In the conceptual level, internal details such as an implementation of the data structure are hidden.
* Programmers and database administrators work at this level.

### 3. External Level

DBMS Three schema Architecture

* At the external level, a database contains several schemas that sometimes called as subschema. The subschema is used to describe the different view of the database.
* An external schema is also known as view schema.
* Each view schema describes the database part that a particular user group is interested and hides the remaining database from that user group.
* The view schema describes the end user interaction with database systems.

## **Mapping between Views**

The three levels of DBMS architecture don't exist independently of each other. There must be correspondence between the three levels i.e. how they actually correspond with each other. DBMS is responsible for correspondence between the three types of schema. This correspondence is called Mapping.

**There are basically two types of mapping in the database architecture:**

* Conceptual/ Internal Mapping
* External / Conceptual Mapping

**Conceptual/ Internal Mapping**

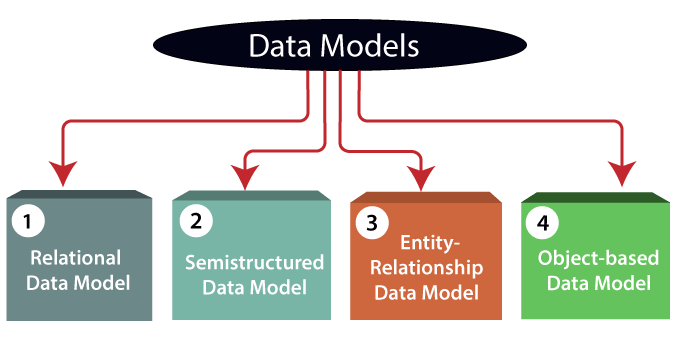
The Conceptual/ Internal Mapping lies between the conceptual level and the internal level. Its role is to define the correspondence between the records and fields of the conceptual level and files and data structures of the internal level.

**External/ Conceptual Mapping**

The external/Conceptual Mapping lies between the external level and the Conceptual level. Its role is to define the correspondence between a particular external and the conceptual view.

# **Data Models**

Data Model is the modeling of the data description, data semantics, and consistency constraints of the data. It provides the conceptual tools for describing the design of a database at each level of data abstraction. Therefore, there are following four data models used for understanding the structure of the database:



**1) Relational Data Model:** This type of model designs the data in the form of rows and columns within a table. Thus, a relational model uses tables for representing data and in-between relationships. Tables are also called relations. This model was initially described by Edgar F. Codd, in 1969. The relational data model is the widely used model which is primarily used by commercial data processing applications.

**2) Entity-Relationship Data Model:** An ER model is the logical representation of data as objects and relationships among them. These objects are known as entities, and relationship is an association among these entities. This model was designed by Peter Chen and published in 1976 papers. It was widely used in database designing. A set of attributes describe the entities. For example, student\_name, student\_id describes the 'student' entity. A set of the same type of entities is known as an 'Entity set', and the set of the same type of relationships is known as 'relationship set'.

**3) Object-based Data Model:** An extension of the ER model with notions of functions, encapsulation, and object identity, as well. This model supports a rich type system that includes structured and collection types. Thus, in 1980s, various database systems following the object-oriented approach were developed. Here, the objects are nothing but the data carrying its properties.

**4) Semistructured Data Model:** This type of data model is different from the other three data models (explained above). The semistructured data model allows the data specifications at places where the individual data items of the same type may have different attributes sets. The Extensible Markup Language, also known as XML, is widely used for representing the semistructured data. Although XML was initially designed for including the markup information to the text document, it gains importance because of its application in the exchange of data.

# **Data model Schema and Instance**

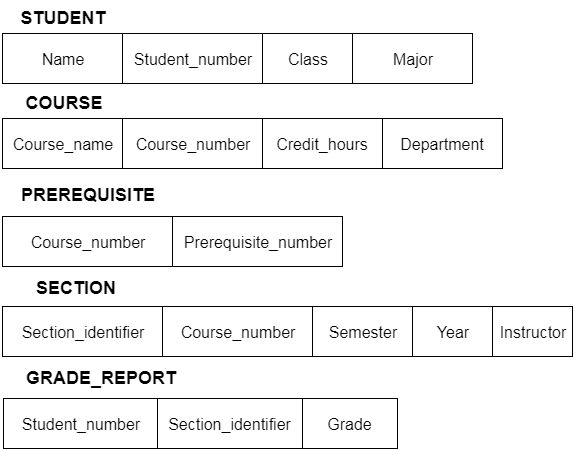
ADVERTISEMENT

ADVERTISEMENT

* The data which is stored in the database at a particular moment of time is called an instance of the database.
* The overall design of a database is called schema.
* A database schema is the skeleton structure of the database. It represents the logical view of the entire database.
* A schema contains schema objects like table, foreign key, primary key, views, columns, data types, stored procedure, etc.
* A database schema can be represented by using the visual diagram. That diagram shows the database objects and relationship with each other.
* A database schema is designed by the database designers to help programmers whose software will interact with the database. The process of database creation is called data modeling.

A schema diagram can display only some aspects of a schema like the name of record type, data type, and constraints. Other aspects can't be specified through the schema diagram. For example, the given figure neither show the data type of each data item nor the relationship among various files.

In the database, actual data changes quite frequently. For example, in the given figure, the database changes whenever we add a new grade or add a student. The data at a particular moment of time is called the instance of the database



# **Data Independence**

ADVERTISEMENT

ADVERTISEMENT

* Data independence can be explained using the three-schema architecture.
* Data independence refers characteristic of being able to modify the schema at one level of the database system without altering the schema at the next higher level.

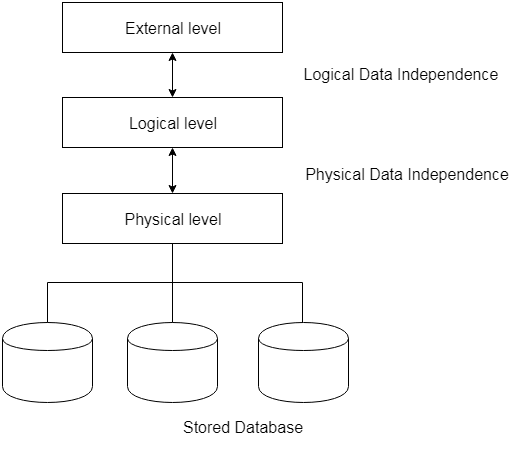
There are two types of data independence:

## **1. Logical Data Independence**

* Logical data independence refers characteristic of being able to change the conceptual schema without having to change the external schema.
* Logical data independence is used to separate the external level from the conceptual view.
* If we do any changes in the conceptual view of the data, then the user view of the data would not be affected.
* Logical data independence occurs at the user interface level.

## **2. Physical Data Independence**

* Physical data independence can be defined as the capacity to change the internal schema without having to change the conceptual schema.
* If we do any changes in the storage size of the database system server, then the Conceptual structure of the database will not be affected.
* Physical data independence is used to separate conceptual levels from the internal levels.
* Physical data independence occurs at the logical interface level.



**Fig: Data Ind**

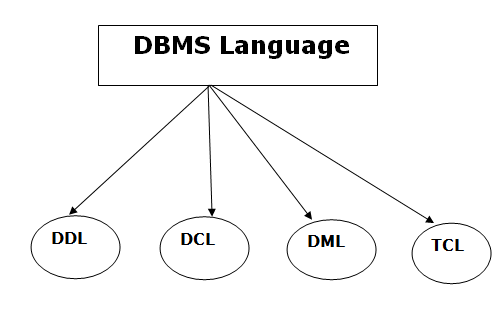
# **Database Languages in DBMS**

ADVERTISEMENT

ADVERTISEMENT

* A DBMS has appropriate languages and interfaces to express database queries and updates.
* Database languages can be used to read, store and update the data in the database.

## **Types of Database Languages**



## **1. Data Definition Language (DDL)**

* **DDL** stands for **D**ata **D**efinition **L**anguage. It is used to define database structure or pattern.
* It is used to create schema, tables, indexes, constraints, etc. in the database.
* Using the DDL statements, you can create the skeleton of the database.
* Data definition language is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.

Here are some tasks that come under DDL:

* **Create:** It is used to create objects in the database.
* **Alter:** It is used to alter the structure of the database.
* **Drop:** It is used to delete objects from the database.
* **Truncate:** It is used to remove all records from a table.
* **Rename:** It is used to rename an object.
* **Comment:** It is used to comment on the data dictionary.

These commands are used to update the database schema that's why they come under Data definition language.

## **2. Data Manipulation Language (DML)**

**DML** stands for **D**ata **M**anipulation **L**anguage. It is used for accessing and manipulating data in a database. It handles user requests.

Here are some tasks that come under DML:

ADVERTISEMENT

* **Select:** It is used to retrieve data from a database.
* **Insert:** It is used to insert data into a table.
* **Update:** It is used to update existing data within a table.
* **Delete:** It is used to delete all records from a table.
* **Merge:** It performs UPSERT operation, i.e., insert or update operations.
* **Call:** It is used to call a structured query language or a Java subprogram.
* **Explain Plan:** It has the parameter of explaining data.
* **Lock Table:** It controls concurrency.

## **3. Data Control Language (DCL)**

* **DCL** stands for **D**ata **C**ontrol **L**anguage. It is used to retrieve the stored or saved data.
* The DCL execution is transactional. It also has rollback parameters.

(But in Oracle database, the execution of data control language does not have the feature of rolling back.)

Here are some tasks that come under DCL:

* **Grant:** It is used to give user access privileges to a database.
* **Revoke:** It is used to take back permissions from the user.

There are the following operations which have the authorization of Revoke:

CONNECT, INSERT, USAGE, EXECUTE, DELETE, UPDATE and SELECT.

## **4. Transaction Control Language (TCL)**

TCL is used to run the changes made by the DML statement. TCL can be grouped into a logical transaction.

Here are some tasks that come under TCL:

* **Commit:** It is used to save the transaction on the database.
* **Rollback:** It is used to restore the database to original since the last Commit.

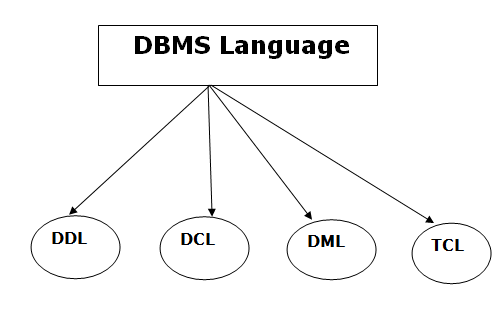
# **Database Languages in DBMS**

ADVERTISEMENT

ADVERTISEMENT

* A DBMS has appropriate languages and interfaces to express database queries and updates.
* Database languages can be used to read, store and update the data in the database.

## **Types of Database Languages**



## **1. Data Definition Language (DDL)**

* **DDL** stands for **D**ata **D**efinition **L**anguage. It is used to define database structure or pattern.
* It is used to create schema, tables, indexes, constraints, etc. in the database.
* Using the DDL statements, you can create the skeleton of the database.
* Data definition language is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.

Here are some tasks that come under DDL:

* **Create:** It is used to create objects in the database.
* **Alter:** It is used to alter the structure of the database.
* **Drop:** It is used to delete objects from the database.
* **Truncate:** It is used to remove all records from a table.
* **Rename:** It is used to rename an object.
* **Comment:** It is used to comment on the data dictionary.

These commands are used to update the database schema that's why they come under Data definition language.

## **2. Data Manipulation Language (DML)**

**DML** stands for **D**ata **M**anipulation **L**anguage. It is used for accessing and manipulating data in a database. It handles user requests.

Here are some tasks that come under DML:

ADVERTISEMENT

* **Select:** It is used to retrieve data from a database.
* **Insert:** It is used to insert data into a table.
* **Update:** It is used to update existing data within a table.
* **Delete:** It is used to delete all records from a table.
* **Merge:** It performs UPSERT operation, i.e., insert or update operations.
* **Call:** It is used to call a structured query language or a Java subprogram.
* **Explain Plan:** It has the parameter of explaining data.
* **Lock Table:** It controls concurrency.

## **3. Data Control Language (DCL)**

* **DCL** stands for **D**ata **C**ontrol **L**anguage. It is used to retrieve the stored or saved data.
* The DCL execution is transactional. It also has rollback parameters.

(But in Oracle database, the execution of data control language does not have the feature of rolling back.)

Here are some tasks that come under DCL:

* **Grant:** It is used to give user access privileges to a database.
* **Revoke:** It is used to take back permissions from the user.

There are the following operations which have the authorization of Revoke:

CONNECT, INSERT, USAGE, EXECUTE, DELETE, UPDATE and SELECT.

## **4. Transaction Control Language (TCL)**

TCL is used to run the changes made by the DML statement. TCL can be grouped into a logical transaction.

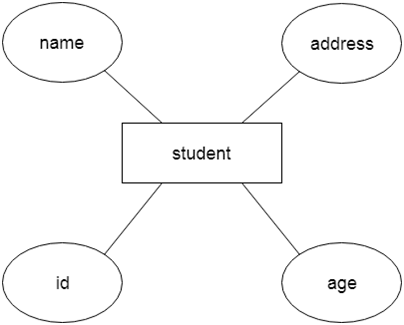
Here are some tasks that come under TCL:

* **Commit:** It is used to save the transaction on the database.

# **Rollback:** It is used to restore the database to original since the last Commit. **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**

PlayNext

Unmute

Current TimeÂ 0:00

/

DurationÂ 11:26

Loaded: 0.58%

Â

Fullscreen

Backward Skip 10sPlay VideoForward Skip 10s

Anomalies in Table | Anomalies in DBMS

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.

ADVERTISEMENT

ADVERTISEMENT



### 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.



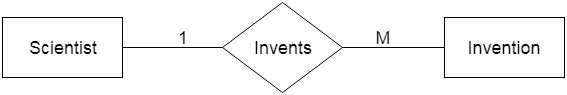
ADVERTISEMENT

ADVERTISEMENT

**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.

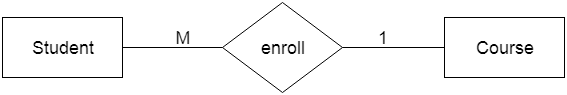
**For example,** Scientist can invent many inventions, but the invention is done by the only specific scientist.



**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.

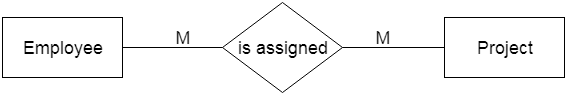
**For example,** Student enrolls for only one course, but a course can have many students.



**d. Many-to-many relationship**

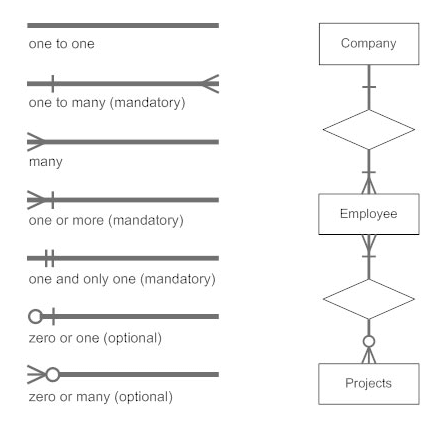
When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

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



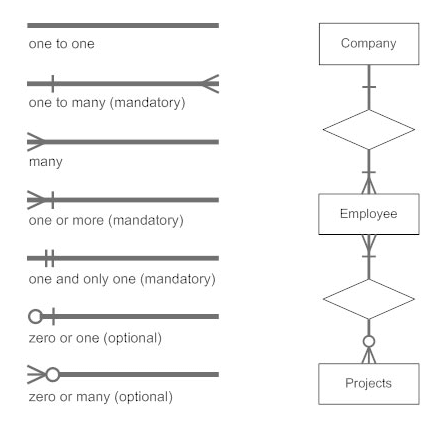
# **Notation of ER diagram**

Database can be represented using the notations. In ER diagram, many notations are used to express the cardinality. These notations are as follows:



**Fig: Notations of ER diagram**

# **Notation of ER diagram**

Database can be represented using the notations. In ER diagram, many notations are used to express the cardinality. These notations are as follows:  


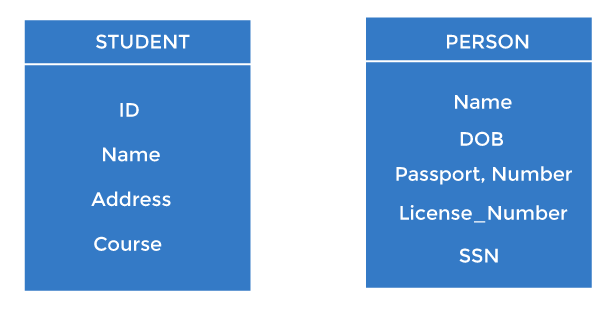
# **Fig: Notations of ER diagramKeys**

ADVERTISEMENT

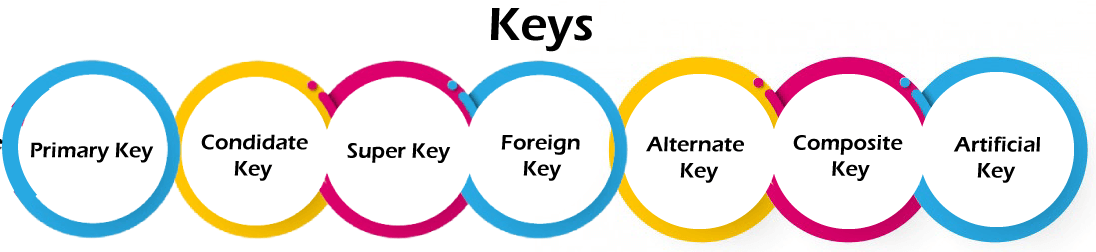
ADVERTISEMENT

* 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.

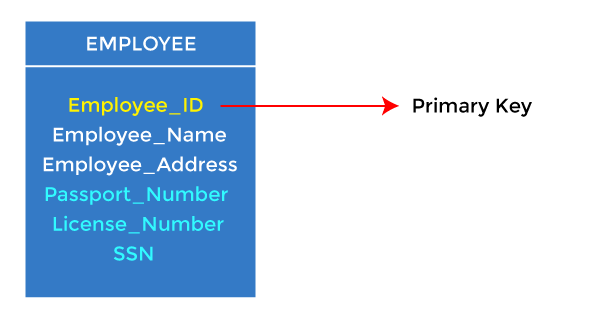


## **Types of keys:**



### 1. Primary key

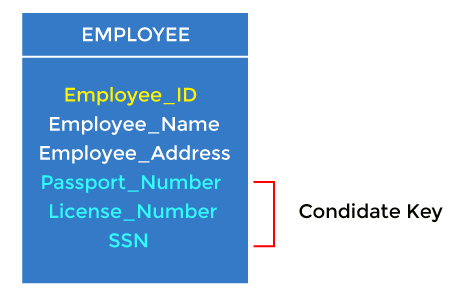
* It is the first key used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys, as we saw in the PERSON table. The key which is most suitable from those lists becomes a primary key.
* In the EMPLOYEE table, ID can be the primary key since it is unique for 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

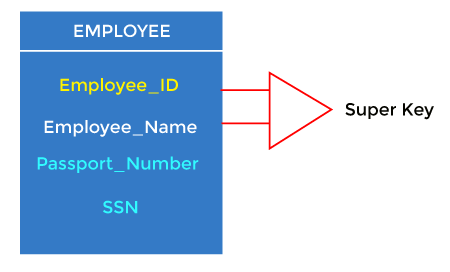
* A candidate key is an attribute or set of attributes that can uniquely identify a tuple.
* Except for the primary key, the remaining attributes are considered a candidate key. The candidate keys are as strong as the primary key.

**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

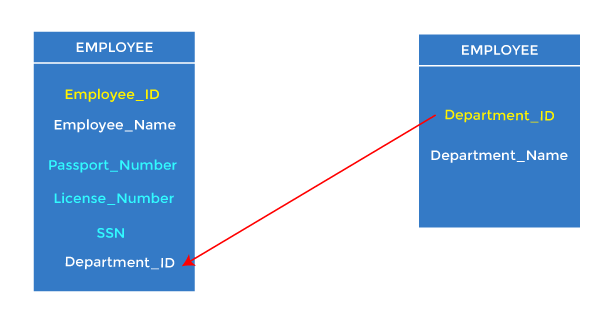
Super key is an attribute set that can uniquely identify a tuple. A super key is a superset of a candidate key.



**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

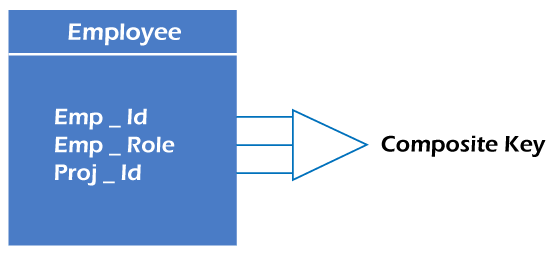
There may be one or more attributes or a combination of attributes that uniquely identify each tuple in a relation. These attributes or combinations of the attributes are called the candidate keys. One key is chosen as the primary key from these candidate keys, and the remaining candidate key, if it 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.

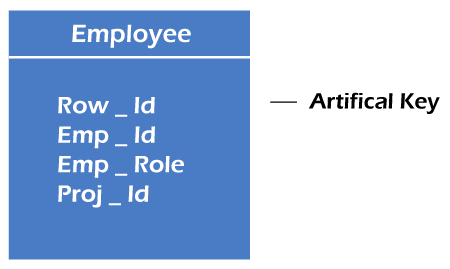


6. Composite key

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.

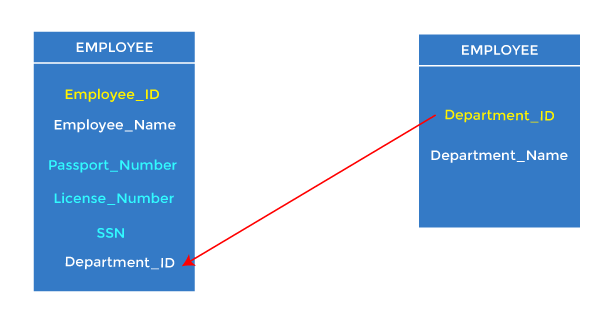
ADVERTISEMENT

**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.

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

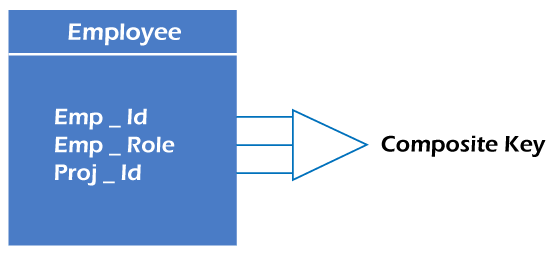
There may be one or more attributes or a combination of attributes that uniquely identify each tuple in a relation. These attributes or combinations of the attributes are called the candidate keys. One key is chosen as the primary key from these candidate keys, and the remaining candidate key, if it 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.

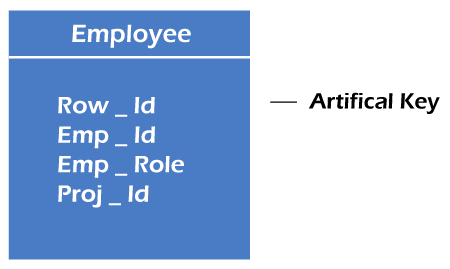


6. Composite key

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.

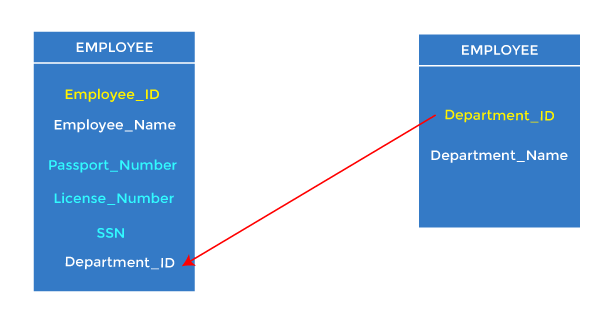
ADVERTISEMENT

**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.

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

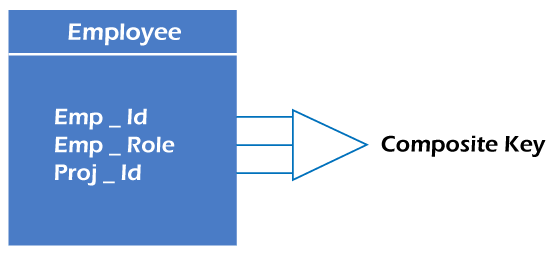
There may be one or more attributes or a combination of attributes that uniquely identify each tuple in a relation. These attributes or combinations of the attributes are called the candidate keys. One key is chosen as the primary key from these candidate keys, and the remaining candidate key, if it 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.

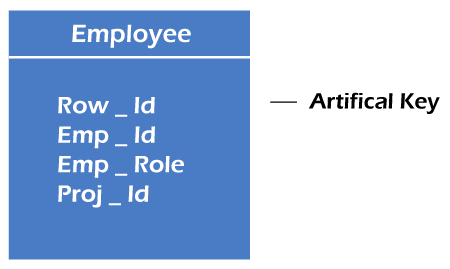


6. Composite key

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.

ADVERTISEMENT

**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.