**DBMS:-**

Database management system is software that is used to manage the database.

**What is Database**

The database is a collection of inter-related data which is used to retrieve, insert and delete the data efficiently. It is also used to organize the data in the form of a table, schema, views, and reports, etc.

**Task Performed by DBMS ;**

* **Data Definition:** It is used for creation, modification, and removal of definition that defines the organization of data in the database.
* **Data Updation:** It is used for the insertion, modification, and deletion of the actual data in the database.
* **Data Retrieval:** It is used to retrieve the data from the database which can be used by applications for various purposes.
* **User Administration:** It is used for registering and monitoring users, maintain data integrity, enforcing data security, dealing with concurrency control, monitoring performance and recovering information corrupted by unexpected failure.

**DBMS ARCHITECTURE**

### 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.

### 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.

### 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.

# Three schema Architecture

* 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.

### Internal Level (physical schema)

The internal level has an internal schema which describes the physical storage structure of the database.

### 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 level describes what data are to be stored in the database and also describes what relationship exists among those data.

Implementation of datastructures are hidden.

### External Level (view schema)

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.

The view schema describes the end user interaction with database systems.

# DBMS - Data Models

Data models define how the logical structure of a database is modeled.

## Entity-Relationship Model

**Entity** − An entity in an ER Model is a real-world entity having properties called **attributes**. Every **attribute** is defined by its set of values called **domain**.

**Relationship** − The logical association among entities is called ***relationship***.

# ER model:

1. ER model -> Entity-Relationship model
2. High-level data model.
3. Used to define the data elements and relationship for a specified system.
4. It develops a conceptual design for the database
5. The database structure is portrayed as a diagram called an entity-relationship diagram.

## Component of ER Diagram

1. **Entity**
2. **Attributes**
3. **Relation**

**Entity:**

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

->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.

**Attributes:**

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

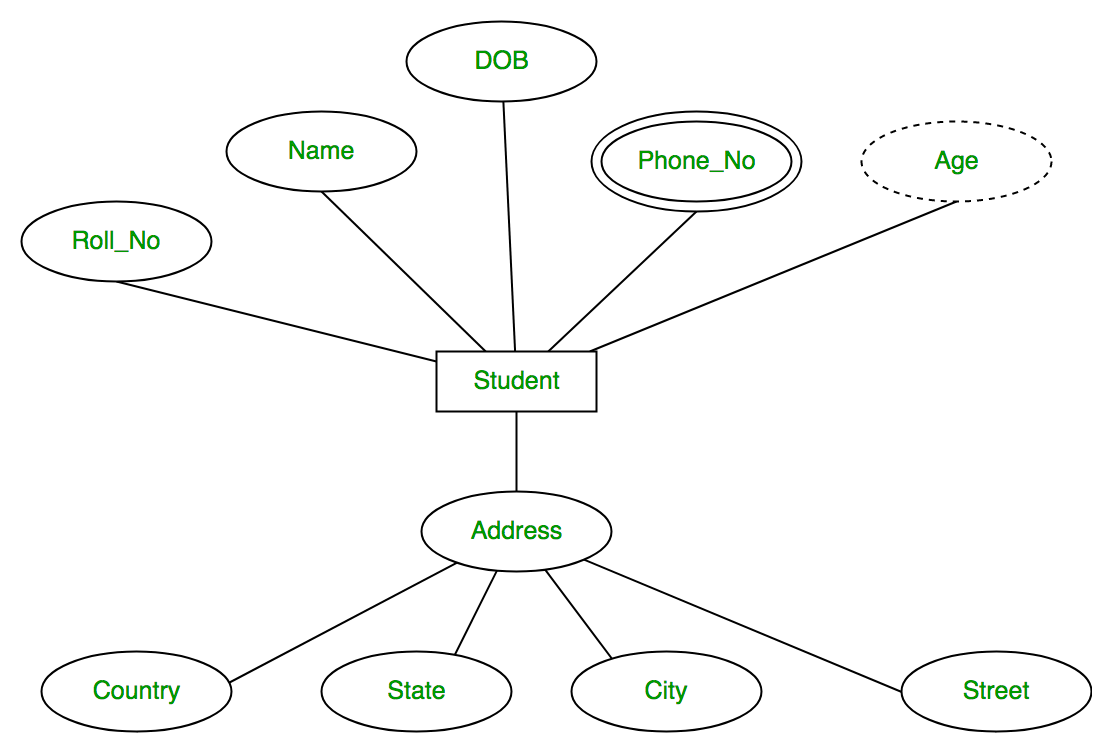
Eclipse is used to represent an attribute.

**Types:**

**->key**

**->composite**

**->derived**

**->multivalued** 

**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.

Eg:id

**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.

**Eg:name-🡪first\_name,mid\_name,l\_name**

**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.

Eg:DOB-🡪age

**MUTLIVALUED ATTRIBUTE:**

An attribute can have more than one value.

The double oval is used to represent multivalued attribute.

Eg:phone\_no

**3.RELATIONSHIP:**

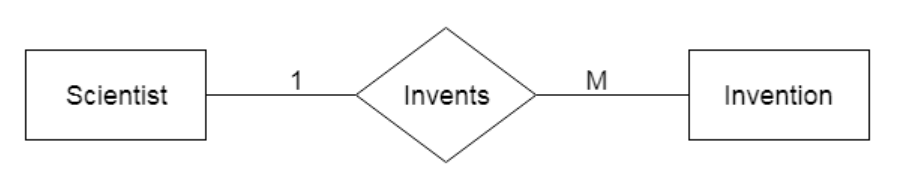
A relationship is used to describe the relation between entities.

Diamond or rhombus is used to represent the relationship.

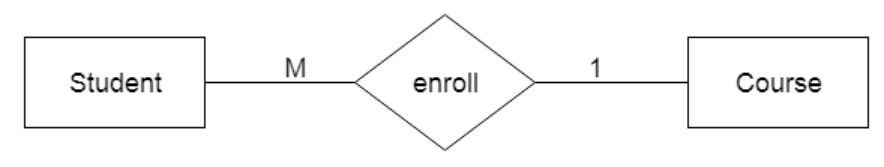
**->ONE TO ONE:**



**->ONE TO MANY:**



**->MANY TO ONE:**



**->MANY TO MANY:**



# **Notation of ER diagram:**

# 

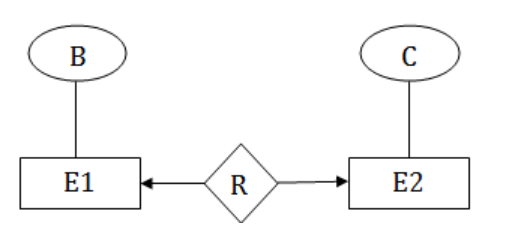
# **Mapping Constraint:**

# **->describe the relationship sets that involve more than two entity sets.**

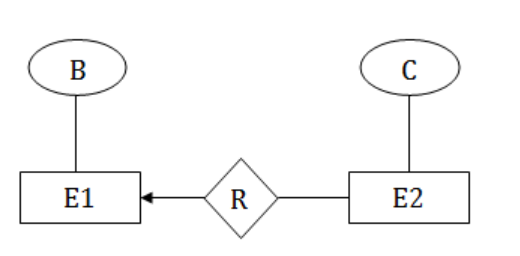
# **There are four possible mapping cardinalities in binary relationship:**

1. One to one (1:1)
2. One to many (1:M)
3. Many to one (M:1)
4. Many to many (M:M)

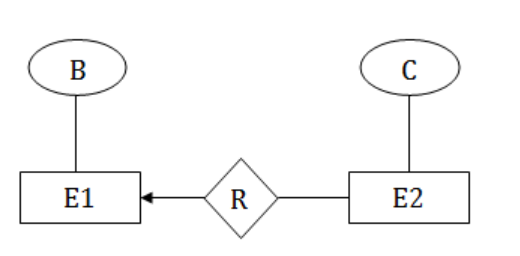
## One-to-one



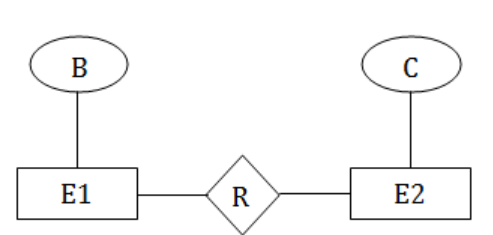
One to many (1:M):



Many to one (M:1):



Many to many (M:M)



# **Keys:**

# ->It is used to uniquely identify any record or row of data from the table

# ->It is used to establish and identify relationships between tables

# TYPES:

->primary

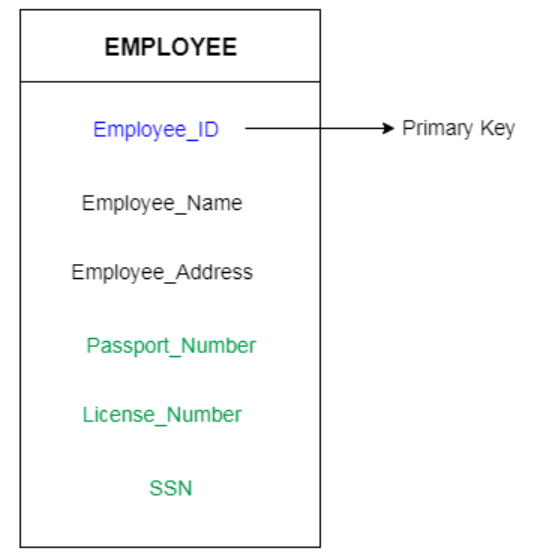
->super

->foreign

->candidate

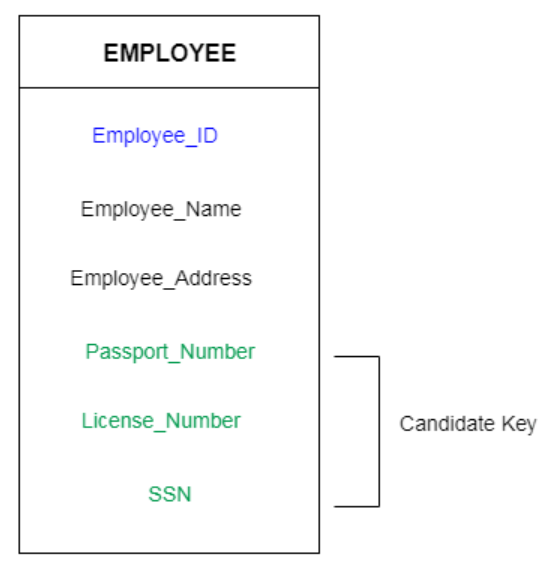
PRIMARY KEY:

It is the first key which is used to identify one and only one instance of an entity uniquely.



**CANDIDATE KEY:**

A candidate key is an attribute or set of an attribute which can uniquely identify a tuple

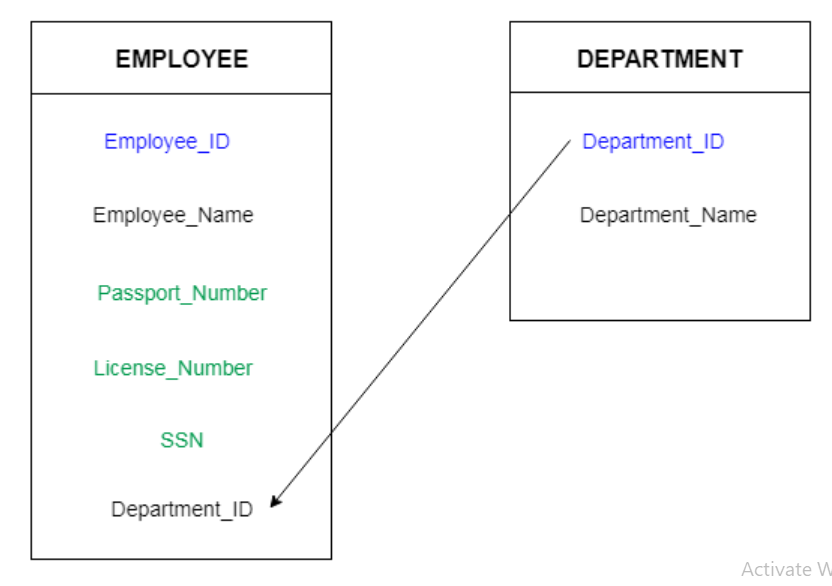


**SUPER KEY:**

Super key is a set of an attribute which can uniquely identify a tuple. Super key is a superset of a candidate key.

**FOREIGN KEY:**

* Foreign keys are the column of the table which is used to point to the primary key of another table.
* the **foreign key** is defined in a second table, but it refers to the primary **key** or a unique **key** in the first table



# **Generalization:**

**->**Generalization is like a bottom-up approach in which two or more entities of lower level combine to form a higher level entity if they have some attributes in common.

->entities of lower lever combine toform higher level entity

->higher level entity also combine with entities of lower level to form further higher level entity.

# 

[**next →**](https://www.javatpoint.com/dbms-aggregation)[**← prev**](https://www.javatpoint.com/dbms-generalization)

# **Specialization**

# **->top-down approach**

# **->opposite to generalization**

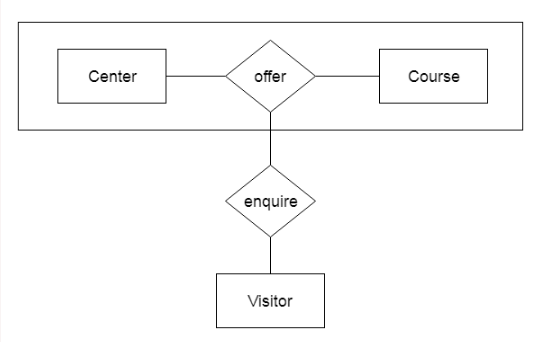
# **->one higher level entity can be broken down into two lower level entities.**

# 

# **Aggregation:**

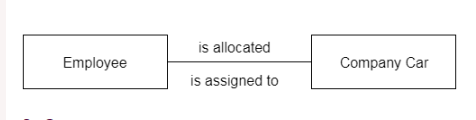
# **In aggregation, the relation between two entities is treated as a single entity.**

# **In aggregation, relationship with its corresponding entities is aggregated into a higher level entity.**

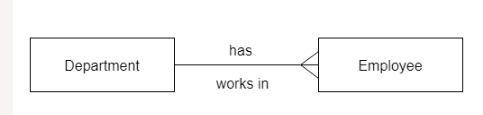


# **Relationship of higher degree**

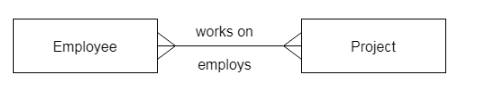
1. One-to-one (1:1)



1. One-to-many (1:M)



1. Many-to-many (M:N)



## Database Schema

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated.

A database schema can be divided broadly into two categories −

* **Physical Database Schema** − This schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.
* **Logical Database Schema** − This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.

# DBMS - Normalization

## Functional Dependency

Functional dependency (FD) is a set of constraints between two attributes in a relation. Functional dependency says that if two tuples have same values for attributes A1, A2,..., An, then those two tuples must have to have same values for attributes B1, B2, ..., Bn.

Functional dependency is represented by an arrow sign (→) that is, X→Y, where X functionally determines Y. The left-hand side attributes determine the values of attributes on the right-hand side.

## Armstrong's Axioms

Set of rules for functional dependency

**1. Reflexive Rule (IR1)**

In the reflexive rule, if Y is a subset of X, then X determines Y.

If X ⊇ Y then X  →    Y

**2. Augmentation Rule (IR2)**

The augmentation is also called as a partial dependency. In augmentation, if X determines Y, then XZ determines YZ for any Z.

If X    →  Y then XZ   →   YZ

**3. Transitive Rule (IR3)**

In the transitive rule, if X determines Y and Y determine Z, then X must also determine Z.

If X   →   Y and Y  →  Z then X  →   Z

**4. Union Rule (IR4)**

Union rule says, if X determines Y and X determines Z, then X must also determine Y and Z.

If X    →  Y and X   →  Z then X  →    YZ

**Proof:**

1. X → Y (given)  
2. X → Z (given)  
3. X → XY (using IR2 on 1 by augmentation with X. Where XX = X)  
4. XY → YZ (using IR2 on 2 by augmentation with Y)  
5. X → YZ (using IR3 on 3 and 4)

**5. Decomposition Rule (IR5)**

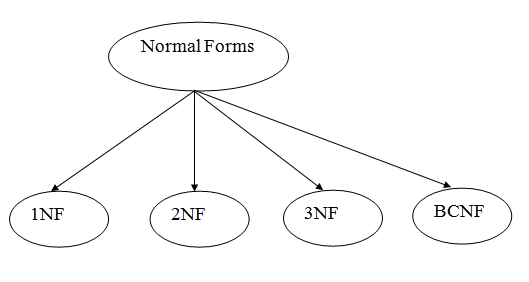
Decomposition rule is also known as project rule. It is the reverse of union rule.

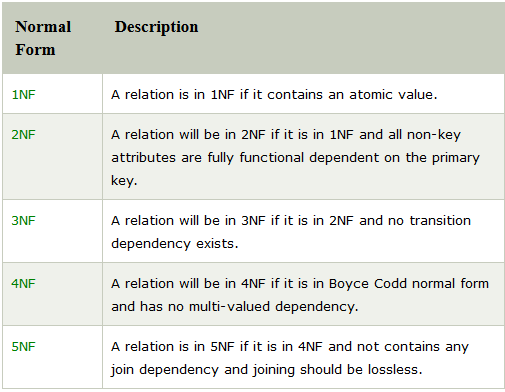
This Rule says, if X determines Y and Z, then X determines Y and X determines Z separately.

If X   →   YZ then X   →   Y and X  →    Z

# Normalization

* Normalization is the process of organizing the data in the database.
* Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.
* Normalization divides the larger table into the smaller table and links them using relationship.

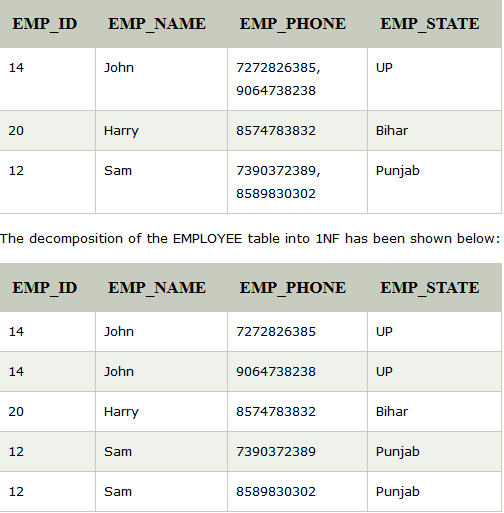




# First Normal Form (1NF)

 A relation will be 1NF if it contains an atomic value.

 It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.

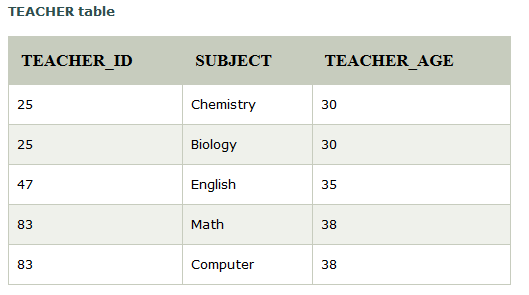


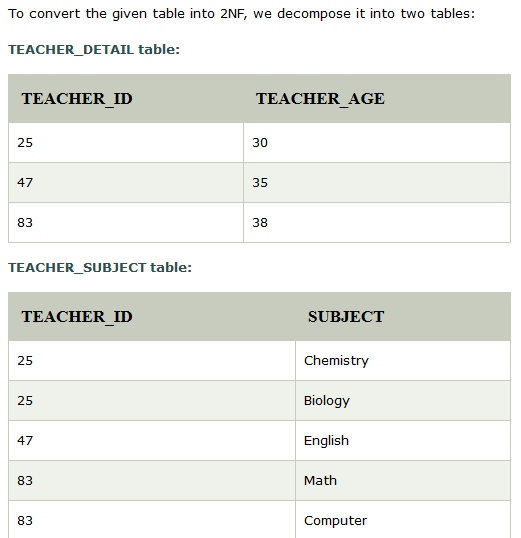
# Second Normal Form (2NF)

 In the 2NF, relational must be in 1NF.

 In the second normal form, all non-key attributes are fully functional dependent on the primary key

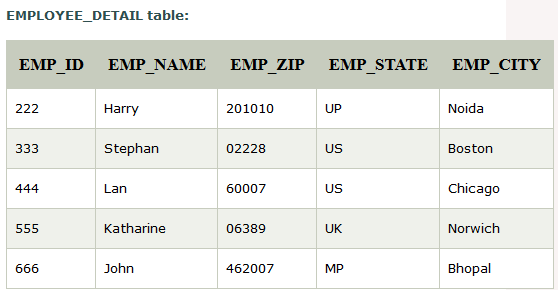
**Example:** Let's assume, a school can store the data of teachers and the subjects they teach. In a school, a teacher can teach more than one subject.





# Third Normal Form (3NF)

* A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.
* 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.

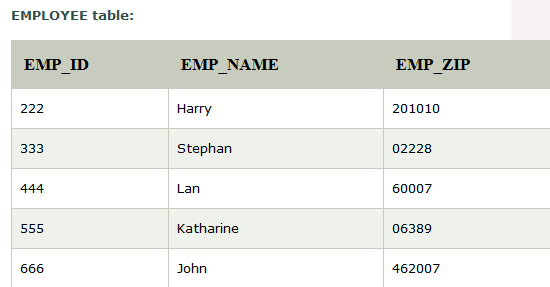


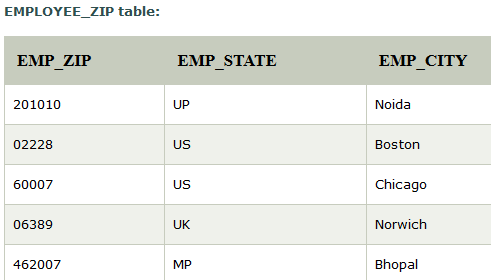
**Super key in the table above:**

{EMP\_ID}, {EMP\_ID, EMP\_NAME}, {EMP\_ID, EMP\_NAME, EMP\_ZIP}....so on

**Candidate key:** {EMP\_ID}

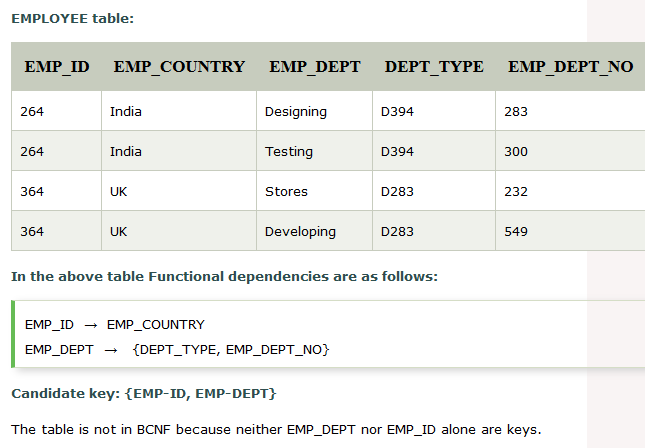
Here, EMP\_STATE & EMP\_CITY dependent on EMP\_ZIP and EMP\_ZIP dependent on EMP\_ID. The non-prime attributes (EMP\_STATE, EMP\_CITY) transitively dependent on super key(EMP\_ID). It violates the rule of third normal form.

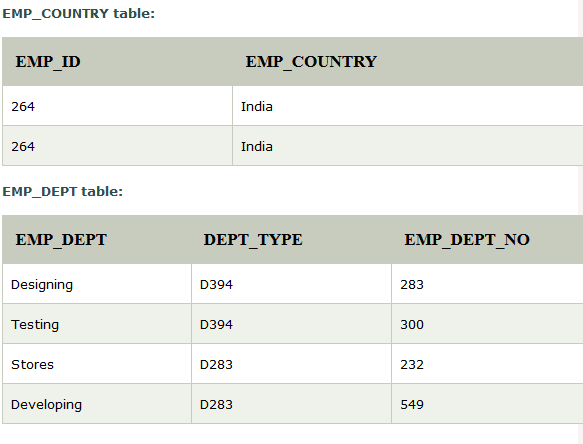


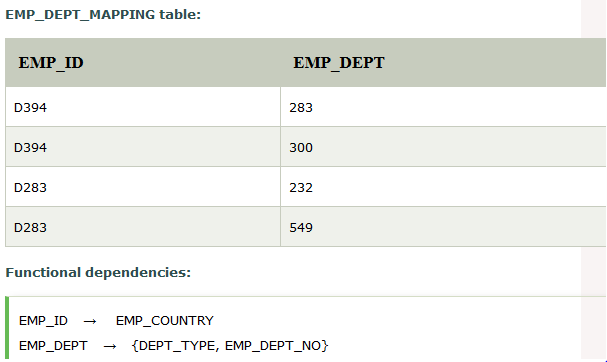


# Boyce Codd normal form (BCNF)

* BCNF is the advance version of 3NF. It is stricter than 3NF.
* A table is in BCNF if every functional dependency X → Y, X is the super key of the table.
* For BCNF, the table should be in 3NF, and for every FD, LHS is super key.







**Candidate keys:**

**For the first table:** EMP\_ID   
**For the second table:** EMP\_DEPT  
**For the third table:** {EMP\_ID, EMP\_DEPT}

# Keys

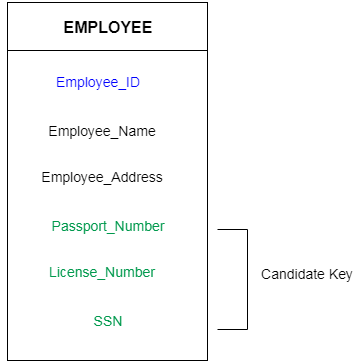
* Keys play an important role in the relational database.
* It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.

### 1. Primary key

* It is the first key which is used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys as we saw in PERSON table. The key which is most suitable from those lists become a primary key.

### 2. Candidate key

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



### 3. Super Key

Super key is a set of an attribute which can uniquely identify a tuple. Super key is a superset of a candidate key.

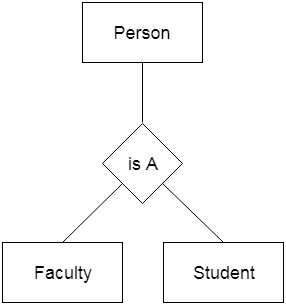
**For example:** In the above EMPLOYEE table, for(EMPLOEE\_ID, EMPLOYEE\_NAME) the name of two employees can be the same, but their EMPLYEE\_ID can't be the same. Hence, this combination can also be a key.

### 4. Foreign key

* Foreign keys are the column of the table which is used to point to the primary key of another table.

# Generalization

* Generalization is like a bottom-up approach in which two or more entities of lower level combine to form a higher level entity if they have some attributes in common.
* Generalization is more like subclass and superclass system, but the only difference is the approach. Generalization uses the bottom-up approach.



**DBMS Transaction:**

1. The transaction is a set of logically related operation. It contains a group of tasks.
2. A transaction is an action or series of actions. It is performed by a single user to perform operations for accessing the contents of the database.

**Operations:**

* + Read – Read the data from DB and store in buffer
  + Write – write the data from buffer and store in DB
  + Commit – Save the transaction on DB
  + Rollback – restore the DB to original since the last commit

**Properties:**

* + Atomicity
  + Consistency
  + Isolation
  + Durability

**Atomicity:**

* It states that all operations of the transaction take place at once if not, the transaction is aborted.
* There is no midway, i.e., the transaction cannot occur partially. Each transaction is treated as one unit and either run to completion or is not executed at all.

**Operations:** Abort – all the changes are not visible.

Commit – all the changes are visible.

**Consistency:**

* The integrity constraints are maintained so that the database is consistent before and after the transaction.
* The execution of a transaction will leave a database in either its prior stable state or a new stable state.
* The consistent property of database states that every transaction sees a consistent database instance.
* The transaction is used to transform the database from one consistent state to another consistent state.

EG: The total amount must be maintained before or after the transaction.

**Isolation:**

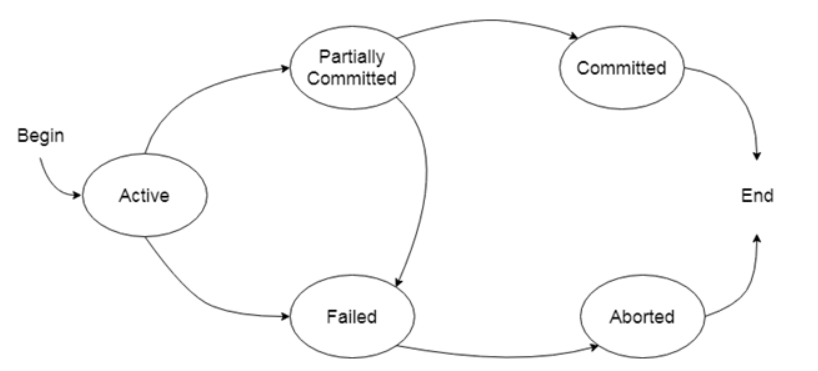
* It shows that the data which is used at the time of execution of a transaction cannot be used by the second transaction until the first one is completed.
* **The** **concurrency control subsystem of the DBMS enforced the isolation property**.

EG: In isolation, if the transaction T1 is being executed and using the data item X, then that data item can't be accessed by any other transaction T2 until the transaction T1 ends.

**Durability:**

* The durability property is used to **indicate the performance of the database's consistent state**. It states that the transaction made the permanent changes.
* They cannot be lost by the erroneous operation of a faulty transaction or by the system failure. **When a transaction is completed, then the database reaches a state known as the consistent state**. That consistent state cannot be lost, even in the event of a system's failure.
* **The** **recovery subsystem of the DBMS has the responsibility of Durability property**.

**States of Transaction:**



**Active state:**

* The active state is the first state of every transaction. In this state, the transaction is being executed.

EX: Insertion or deletion or updating a record is done here. But all the records are still not saved to the database.

**Partially committed:**

* In the partially committed state, a transaction executes its final operation, but the data is still not saved to the database.

EX: total mark calculation , a final display of the total marks step is executed in this state.

**Committed:**

* A transaction is said to be in a committed state if it executes all its operations successfully. In this state, all the effects are now permanently saved on the database system.

**Failed state:**

* If any of the checks made by the database recovery system fails, then the transaction is said to be in the failed state.

EX:total mark calculation, if the database is not able to fire a query to fetch the marks, then the transaction will fail to execute.

**Aborted:**

* If any of the checks fail and the transaction has reached a failed state then the database recovery system will make sure that the database is in its previous consistent state. If not then it will abort or roll back the transaction to bring the database into a consistent state.
* If the transaction fails in the middle of the transaction then before executing the transaction, all the executed transactions are rolled back to its consistent state.

After aborting the transaction, the database recovery module will select one of the two **Operations:**

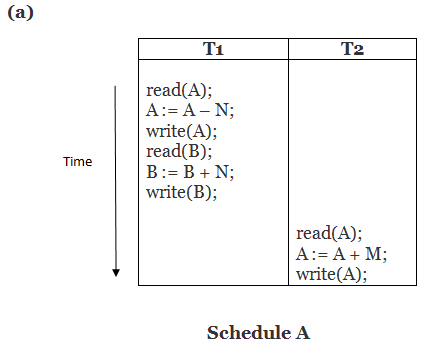
1. Re-start the transaction
2. Kill the transaction

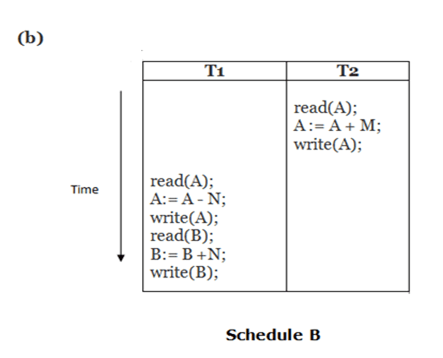
**Schedule:**

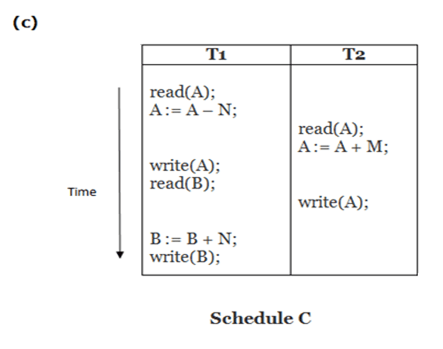
* A series of operation from one transaction to another transaction is known as schedule.
* It is used to preserve the order of the operation in each of the individual transaction.

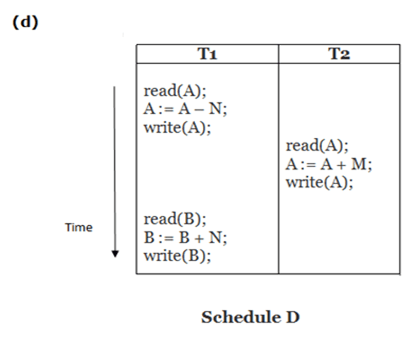
**Types:**

* + Serial - The serial schedule is a type of schedule where one transaction is executed completely before starting another transaction.
  + Non-Serial - If interleaving of operations is allowed, then there will be non-serial schedule.
  + Serializable - The serializability of schedules is used to find non-serial schedules that allow the transaction to execute concurrently without interfering with one another.









A and B are serial

C and D are non-serial

**Failure Classification:**

* Transaction - reason: Logical error, Syntax error.
* System Crash - due to power failure
* Disk Failure - occurs where hard-disk drives or storage drives used to fail frequently

**Relational Model concept**

Relational model can represent as a table with columns and rows. Each row is known as a tuple.

Each table of the column has a name or attribute.

Domain: It contains a set of atomic values that an attribute can take.

Attribute: It contains the name of a column in a particular table. Each attribute Ai must have a domain, dom(Ai)

Relational instance: In the relational database system, the relational instance is represented by a finite set of tuples.

Relation instances do not have duplicate tuples.

Relational schema: A relational schema contains the name of the relation and name of all columns or attributes.

Relational key: In the relational key, each row has one or more attributes. It can identify the row in the relation uniquely.

Properties of Relations

Name of the relation is distinct from all other relations.

Each relation cell contains exactly one atomic (single) value

Each attribute contains a distinct name

tuple has no duplicate value

Relational Algebra

Relational algebra is a procedural query language. It gives a step by step process to obtain the result of the query. It uses

operators to perform queries.

1. Select Operation:

The select operation selects tuples that satisfy a given predicate.

It is denoted by sigma (s).

Notation: s p(r) ex:s BRANCH\_NAME="perryride" (LOAN)

2. Project Operation:

This operation shows the list of those attributes that we wish to appear in the result. Rest of the attributes are eliminated from the table.

It is denoted by ?.

Notation: ? A1, A2, An (r) ex:? NAME, CITY (CUSTOMER)

3. Union Operation:

Suppose there are two tuples R and S. The union operation contains all the tuples that are either in R or S or both in R & S.

It eliminates the duplicate tuples. It is denoted by ?.

Notation: R ? S

A union operation must hold the following condition:

R and S must have the attribute of the same number.

Duplicate tuples are eliminated automatically.

ex:? CUSTOMER\_NAME (BORROW) ? ? CUSTOMER\_NAME (DEPOSITOR)

4. Set Intersection:

Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in both R & S.

It is denoted by intersection n.

Notation: R n S

ex:? CUSTOMER\_NAME (BORROW) n ? CUSTOMER\_NAME (DEPOSITOR)

5. Set Difference:

Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in R but not in S.

It is denoted by intersection minus (-).

Notation: R - S

ex:? CUSTOMER\_NAME (BORROW) - ? CUSTOMER\_NAME (DEPOSITOR)

6. Cartesian product

The Cartesian product is used to combine each row in one table with each row in the other table. It is also known as a cross product.

It is denoted by X.

Notation: E X D

ex:EMPLOYEE X DEPARTMENT

7. Rename Operation:

The rename operation is used to rename the output relation. It is denoted by rho (?).

Example: We can use the rename operator to rename STUDENT relation to STUDENT1.

?(STUDENT1, STUDENT)

Join Operations:

A Join operation combines related tuples from different relations, if and only if a given join condition is satisfied. It is denoted by ?.

ex:Operation: (EMPLOYEE ? SALARY)

**TYPES**

1. Natural Join:

A natural join is the set of tuples of all combinations in R and S that are equal on their common attribute names.

It is denoted by ?.

ex:?EMP\_NAME, SALARY (EMPLOYEE ? SALARY)

2. Outer Join:

The outer join operation is an extension of the join operation. It is used to deal with missing information.

ex:(EMPLOYEE ? FACT\_WORKERS)

An outer join is basically of three types:

Left outer join

Right outer join

Full outer join

a. Left outer join:

Left outer join contains the set of tuples of all combinations in R and S that are equal on their common attribute names.

In the left outer join, tuples in R have no matching tuples in S.

It is denoted by ?. ex:EMPLOYEE ? FACT\_WORKERS

b. Right outer join:

Right outer join contains the set of tuples of all combinations in R and S that are equal on their common attribute names.

In right outer join, tuples in S have no matching tuples in R.

It is denoted by ?.ex:EMPLOYEE ? FACT\_WORKERS

c. Full outer join:

Full outer join is like a left or right join except that it contains all rows from both tables.

In full outer join, tuples in R that have no matching tuples in S and tuples in S that have no matching tuples in R in their common attribute name.

It is denoted by ?.ex:EMPLOYEE ? FACT\_WORKERS

3. Equi join:

It is also known as an inner join. It is the most common join. It is based on matched data as per the equality condition. The equi join uses the comparison operator(=).

ex:CUSTOMER ? PRODUCT

Integrity Constraints

Integrity constraints are a set of rules. It is used to maintain the quality of information.

Types of Integrity Constraint

1. Domain constraints

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.

2. Entity integrity constraints

The entity integrity constraint states that primary key value can't be null.

3. Referential Integrity Constraints

A referential integrity constraint is specified between two tables.

In the Referential integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key

in Table 1 must be null or be available in Table 2.

4. Key constraints

Keys are the entity set that is used to identify an entity within its entity set uniquely.

Relational Calculus

Relational calculus is a non-procedural query language

The relational calculus tells what to do but never explains how to do.

Types of Relational calculus:

1. Tuple Relational Calculus (TRC)

The tuple relational calculus is specified to select the tuples in a relation. In TRC, filtering variable uses the tuples of a relation.

The result of the relation can have one or more tuples.

notation:T | P (T)} or {T | Condition (T)}

2. Domain Relational Calculus (DRC)

The second form of relation is known as Domain relational calculus. In domain relational calculus, filtering variable uses the domain of attributes.

Domain relational calculus uses the same operators as tuple calculus. It uses logical connectives ? (and), ? (or) and ? (not).

It uses Existential (?) and Universal Quantifiers (?) to bind the variable.

Notation:{ a1, a2, a3, ..., an | P (a1, a2, a3, ... ,an)}

**Raid**

RAID refers to redundancy array of the independent disk. It is a technology which is used to connect multiple secondary storage devices for increased performance, data redundancy or both. It gives you the ability to survive one or more drive failure depending upon the RAID level used.

There are 7 levels of RAID schemes

RAID 0

* RAID level 0 provides data stripping, i.e., a data can place across multiple disks. It is based on stripping that means if one disk fails then all data in the array is lost.
* This level doesn't provide fault tolerance but increases the system performance.

Pros of RAID 0:

* In this level, throughput is increased because multiple data requests probably not on the same disk.
* This level full utilizes the disk space and provides high performance.
* It requires minimum 2 drives.

Cons of RAID 0:

* It doesn't contain any error detection mechanism.
* The RAID 0 is not a true RAID because it is not fault-tolerance.
* In this level, failure of either disk results in complete data loss in respective array.

## RAID 1

This level is called mirroring of data as it copies the data from drive 1 to drive 2. It provides 100% redundancy in case of a failure.

Pros of RAID 1:

* The main advantage of RAID 1 is fault tolerance. In this level, if one disk fails, then the other automatically takes over.
* In this level, the array will function even if any one of the drives fails.

Cons of RAID 1:

* In this level, one extra drive is required per drive for mirroring, so the expense is higher.

## RAID 2

* RAID 2 consists of bit-level striping using hamming code parity. In this level, each data bit in a word is recorded on a separate disk and ECC code of data words is stored on different set disks.
* Due to its high cost and complex structure, this level is not commercially used. This same performance can be achieved by RAID 3 at a lower cost.

Pros of RAID 2:

* This level uses one designated drive to store parity.
* It uses the hamming code for error detection.

Cons of RAID 2:

* It requires an additional drive for error detection

## RAID 3

* RAID 3 consists of byte-level striping with dedicated parity. In this level, the parity information is stored for each disk section and written to a dedicated parity drive.
* In case of drive failure, the parity drive is accessed, and data is reconstructed from the remaining devices. Once the failed drive is replaced, the missing data can be restored on the new drive.
* In this level, data can be transferred in bulk. Thus high-speed data transmission is possible.

Pros of RAID 3:

* In this level, data is regenerated using parity drive.
* It contains high data transfer rates.
* In this level, data is accessed in parallel.

Cons of RAID 3:

* It required an additional drive for parity.
* It gives a slow performance for operating on small sized files.

## RAID 4

* RAID 4 consists of block-level stripping with a parity disk. Instead of duplicating data, the RAID 4 adopts a parity-based approach.
* This level allows recovery of at most 1 disk failure due to the way parity works. In this level, if more than one disk fails, then there is no way to recover the data.
* Level 3 and level 4 both are required at least three disks to implement RAID.
* In this figure, we can observe one disk dedicated to parity.
* In this level, parity can be calculated using an XOR function. If the data bits are 0,0,0,1 then the parity bits is XOR(0,1,0,0) = 1. If the parity bits are 0,0,1,1 then the parity bit is XOR(0,0,1,1)= 0. That means, even number of one results in parity 0 and an odd number of one results in parity 1.

Pros of RAID 5:

* This level is cost effective and provides high performance.
* In this level, parity is distributed across the disks in an array.
* It is used to make the random write performance better.

Cons of RAID 5:

* In this level, disk failure recovery takes longer time as parity has to be calculated from all available drives.
* This level cannot survive in concurrent drive failure.

## RAID 6

* This level is an extension of RAID 5. It contains block-level stripping with 2 parity bits.
* In RAID 6, you can survive 2 concurrent disk failures. Suppose you are using RAID 5, and RAID 1. When your disks fail, you need to replace the failed disk because if simultaneously another disk fails then you won't be able to recover any of the data, so in this case RAID 6 plays its part where you can survive two concurrent disk failures before you run out of options.

Pros of RAID 6:

* This level performs RAID 0 to strip data and RAID 1 to mirror. In this level, stripping is performed before mirroring.
* In this level, drives required should be multiple of 2.

Cons of RAID 6:

* It is not utilized 100% disk capability as half is used for mirroring.
* It contains very limited scalability.

## RAID level 10 – combining RAID 1 & RAID 0

It is possible to combine the advantages (and disadvantages) of RAID 0 and RAID 1 in one single system. This is a nested or hybrid RAID configuration. It provides security by mirroring all data on secondary drives while using striping across each set of drives to speed up data transfers.

Advantages

* If something goes wrong with one of the disks in a RAID 10 configuration, the rebuild time is very fast since all that is needed is copying all the data from the surviving mirror to a new drive. This can take as little as 30 minutes for drives of  1 TB.

Disadvantages

* Half of the storage capacity goes to mirroring, so compared to large RAID 5  or RAID 6 arrays, this is an expensive way to have redundancy.

<https://www.studytonight.com/dbms/joining-in-sql.php> - joins

<http://www.complexsql.com/complex-sql-queries-examples-with-answers/> - sql queries