Database is collection of data which is related by some aspect. Data is collection of facts and figures which can be processed to produce information. Name of a student, age, class and her subjects can be counted as data for recording purposes.

Mostly data represents recordable facts. Data aids in producing information which is based on facts. For example, if we have data about marks obtained by all students, we can then conclude about toppers and average marks etc.

A database management system stores data, in such a way which is easier to retrieve, manipulate and helps to produce information.

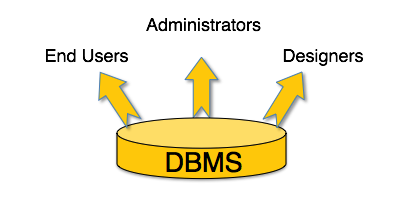
**Characteristics**

Traditionally data was organized in file formats. DBMS was all new concepts then and all the research was done to make it to overcome all the deficiencies in traditional style of data management. Modern DBMS has the following characteristics:

* **Real-world entity**: Modern DBMS are more realistic and uses real world entities to design its architecture. It uses the behavior and attributes too. For example, a school database may use student as entity and their age as their attribute.
* **Relation-based tables**: DBMS allows entities and relations among them to form as tables. This eases the concept of data saving. A user can understand the architecture of database just by looking at table names etc.
* **Isolation of data and application**: A database system is entirely different than its data. Where database is said to active entity, data is said to be passive one on which the database works and organizes. DBMS also stores metadata which is data about data, to ease its own process.
* **Less redundancy**: DBMS follows rules of normalization, which splits a relation when any of its attributes is having redundancy in values. Following normalization, which itself is a mathematically rich and scientific process, make the entire database to contain as less redundancy as possible.
* **Consistency**: DBMS always enjoy the state on consistency where the previous form of data storing applications like file processing does not guarantee this. Consistency is a state where every relation in database remains consistent. There exist methods and techniques, which can detect attempt of leaving database in inconsistent state.
* **Query Language**: DBMS is equipped with query language, which makes it more efficient to retrieve and manipulate data. A user can apply as many and different filtering options, as he or she wants. Traditionally it was not possible where file-processing system was used.
* **ACID Properties**: DBMS follows the concepts for ACID properties, which stands for Atomicity, Consistency, Isolation and Durability. These concepts are applied on transactions, which manipulate data in database. ACID properties maintains database in healthy state in multi-transactional environment and in case of failure.
* **Multiuser and Concurrent Access**: DBMS support multi-user environment and allows them to access and manipulate data in parallel. Though there are restrictions on transactions when they attempt to handle same data item, but users are always unaware of them.
* **Security**: Features like multiple views offers security at some extent where users are unable to access data of other users and departments. DBMS offers methods to impose constraints while entering data into database and retrieving data at later stage. DBMS offers many different levels of security features, which enables multiple users to have different view with different features. For example, a user in sales department cannot see data of purchase department is one thing, additionally how much data of sales department he can see, can also be managed. Because DBMS is not saved on disk as traditional file system it is very hard for a thief to break the code.

**Users**

DBMS is used by various users for various purposes. Some may involve in retrieving data and some may involve in backing it up. Some of them are described as follows:



[*Image: DBMS Users*]

* **Administrators**: A bunch of users maintain the DBMS and are responsible for administrating the database. They are responsible to look after its usage and by whom it should be used. They create users access and apply limitation to maintain isolation and force security. Administrators also look after DBMS resources like system license, software application and tools required and other hardware related maintenance.
* **Designer**: This is the group of people who actually works on designing part of database. The actual database is started with requirement analysis followed by a good designing process. They people keep a close watch on what data should be kept and in what format. They identify and design the whole set of entities, relations, constraints and views.
* **End Users**: This group contains the persons who actually take advantage of database system. End users can be just viewers who pay attention to the logs or market rates or end users can be as sophisticated as business analysts who takes the most of it.

Data model tells how the logical structure of a database is modeled. Data Models are fundamental entities to introduce abstraction in DBMS. Data models define how data is connected to each other and how it will be processed and stored inside the system.

The very first data model could be flat data-models where all the data used to be kept in same plane. Because earlier data models were not so scientific they were prone to introduce lots of duplication and update anomalies.

**The Main Principles of DBMS**

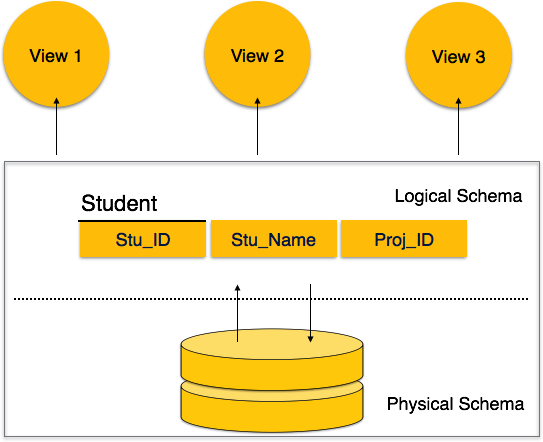
Database systems can be viewed as mediators between human beings who want to use data and physical devices that hold it. Early database management was based on explicit usage of file systems and customized application software. Gradually, principles and mechanisms were developed that insulated database users from the details of the physical implementation and separated database functionalities into physical, logical, and external levels.

The separation of the logical definition of data from its physical implementation is central to the field of databases. One of the major research directions in the field has been the development and study of abstract, human-oriented models and interfaces for specifying the structure of stored data and for manipulating it. These models permit the user to concentrate on a logical representation of data that resembles his or her vision of the reality modeled by the data much more closely than the physical representation.

**Database schema**

Database schema skeleton structure of and it represents the logical view of entire database. It tells about how the data is organized and how relation among them is associated. It formulates all database constraints that would be put on data in relations, which resides in database.

A database schema defines its entities and the relationship among them. Database schema is a descriptive detail of the database, which can be depicted by means of schema diagrams. All these activities are done by database designer to help programmers in order to give some ease of understanding all aspect of database.



[*Image: Database Schemas*]

Database schema can be divided broadly in 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 secondary storage etc.
* **Logical Database Schema:** This defines all logical constraints that need to be applied on data stored. It defines tables, views and integrity constraints etc.

**Database Instance**

It is important that we distinguish these two terms individually. Database schema is the skeleton of database. It is designed when database doesn't exist at all and very hard to do any changes once the database is operational. Database schema does not contain any data or information.

Database instances, is a state of operational database with data at any given time. This is a snapshot of database. Database instances tend to change with time. DBMS ensures that its every instance (state) must be a valid state by keeping up to all validation, constraints and condition that database designers has imposed or it is expected from DBMS itself.

Entity relationship model defines the conceptual view of database. It works around real world entity and association among them. At view level, ER model is considered well for designing databases.

**Entity-Relationship Model**

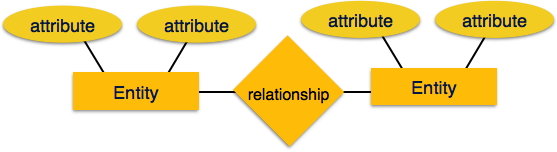
Entity-Relationship model is based on the notion of real world entities and relationship among them. While formulating real-world scenario into database model, ER Model creates entity set, relationship set, general attributes and constraints.

ER Model is best used for the conceptual design of database.

ER Model is based on:

* **Entities** and their *attributes*
* **Relationships** among entities

These concepts are explained below.



[*Image: ER Model*]

* **Entity**

An entity in ER Model is real world entity, which has some properties called ***attributes***. Every attribute is defined by its set of values, called ***domain***.

For example, in a school database, a student is considered as an entity. Student has various attributes like name, age and class etc.

* **Relationship**

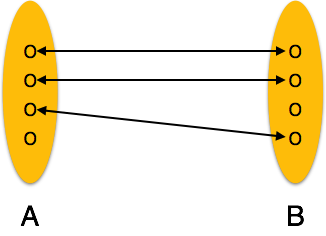
The logical association among entities is called ***relationship***. Relationships are mapped with entities in various ways. Mapping cardinalities define the number of association between two entities.

### Mapping Cardinalities:

**Cardinality** defines the number of entities in one entity set which can be associated to the number of entities of other set via relationship set.

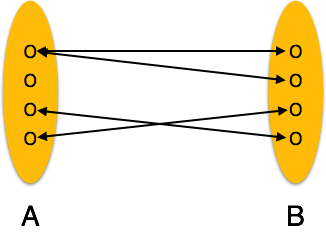
Mapping cardinalities:

* + one to one
  + one to many
  + many to one
  + many to many
* **One-to-one:** one entity from entity set A can be associated with at most one entity of entity set B and vice versa.



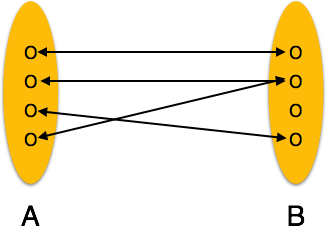
[*Image: One-to-one relation*]

* **One-to-many:** One entity from entity set A can be associated with more than one entities of entity set B but from entity set B one entity can be associated with at most one entity.



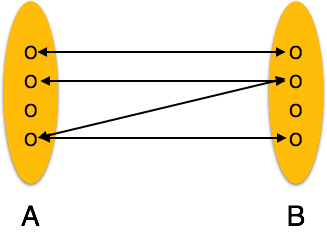
[*Image: One-to-many relation*]

* **Many-to-one:** More than one entities from entity set A can be associated with at most one entity of entity set B but one entity from entity set B can be associated with more than one entity from entity set A.



[*Image: Many-to-one relation*]

* **Many-to-many:** one entity from A can be associated with more than one entity from B and vice versa.



Every object like entity, attributes of an entity, relationship set, and attributes of relationship set can be represented by tools of ER diagram.

## Entity

A real-world thing either animate or inanimate that can be easily identifiable and distinguishable. For example, in a school database, student, teachers, class and course offered can be considered as entities. All entities have some attributes or properties that give them their identity.

An entity set is a collection of similar types of entities. Entity set may contain entities with attribute sharing similar values. For example, Students set may contain all the student of a school; likewise Teachers set may contain all the teachers of school from all faculties.

## Attributes

Entities are represented by means of their properties, called attributes. All attributes have values. For example, a student entity may have name, class, age as attributes.

There exist a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.

### Types of attributes:

* **Simple attribute:**

Simple attributes are atomic values, which cannot be divided further. For example, student's phone-number is an atomic value of 10 digits.

* **Composite attribute:**

Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first\_name and last\_name.

* **Derived attribute:**

Derived attributes are attributes, which do not exist physical in the database, but there values are derived from other attributes presented in the database. For example, average\_salary in a department should be saved in database instead it can be derived. For another example, age can be derived from data\_of\_birth.

* **Single-valued attribute:**

Single valued attributes contain on single value. For example: Social\_Security\_Number.

* **Multi-value attribute:**

Multi-value attribute may contain more than one values. For example, a person can have more than one phone numbers, email\_addresses etc.

These attribute types can come together in a way like:

* simple single-valued attributes
* simple multi-valued attributes
* composite single-valued attributes
* composite multi-valued attributes

### Entity-set and Keys

Key is an attribute or collection of attributes that uniquely identifies an entity among entity set.

For example, roll\_number of a student makes her/him identifiable among students.

* **Super Key:** Set of attributes (one or more) that collectively identifies an entity in an entity set.
* **Candidate Key:** Minimal super key is called candidate key that is, supers keys for which no proper subset are a superkey. An entity set may have more than one candidate key.
* **Primary Key:** This is one of the candidate key chosen by the database designer to uniquely identify the entity set.

## Relationship

The association among entities is called relationship. For example, employee entity has relation works\_at with department. Another example is for student who enrolls in some course. Here, Works\_at and Enrolls are called relationship.

### Degree of relationship

The number of participating entities in a relationship defines the degree of the relationship.

* Binary = degree 2
* Ternary = degree 3
* n-ary = degree n

## Entity

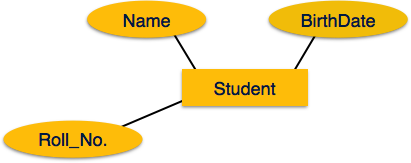
Entities are represented by means of rectangles. Rectangles are named with the entity set they represent.

http://www.tutorialspoint.com/dbms/images/entities.png

[*Image: Entities in a school database*]

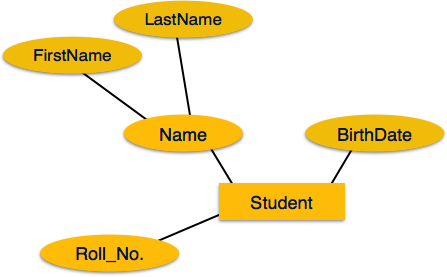
## Attributes

Attributes are properties of entities. Attributes are represented by means of eclipses. Every eclipse represents one attribute and is directly connected to its entity (rectangle).



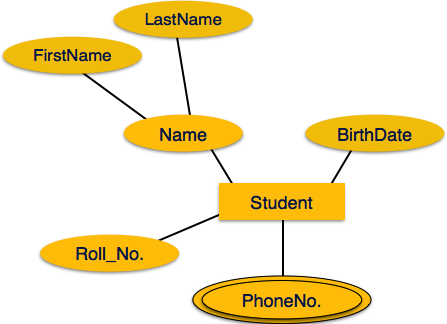
[*Image: Simple Attributes*]

If the attributes are ***composite***, they are further divided in a tree like structure. Every node is then connected to its attribute. That is composite attributes are represented by eclipses that are connected with an eclipse.



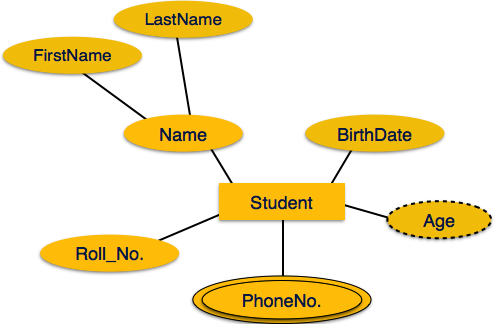
[*Image: Composite Attributes*]

***Multivalued*** attributes are depicted by double eclipse.



[*Image: Multivalued Attributes*]

***Derived*** attributes are depicted by dashed eclipse.



[*Image: Derived Attributes*]

## Relationship

Relationships are represented by diamond shaped box. Name of the relationship is written in the diamond-box. All entities (rectangles), participating in relationship, are connected to it by a line.

### Binary relationship and cardinality

A relationship where two entities are participating is called a ***binary relationship***. Cardinality is the number of instance of an entity from a relation that can be associated with the relation.

* **One-to-one**

When only one instance of entity is associated with the relationship, it is marked as '1'. This image below reflects that only 1 instance of each entity should be associated with the relationship. It depicts one-to-one relationship



[*Image: One-to-one*]

* **One-to-many**

When more than one instance of entity is associated with the relationship, it is marked as 'N'. This image below reflects that only 1 instance of entity on the left and more than one instance of entity on the right can be associated with the relationship. It depicts one-to-many relationship



[*Image: One-to-many*]

* **Many-to-one**

When more than one instance of entity is associated with the relationship, it is marked as 'N'. This image below reflects that more than one instance of entity on the left and only one instance of entity on the right can be associated with the relationship. It depicts many-to-one relationship



[*Image: Many-to-one*]

* **Many-to-many**

This image below reflects that more than one instance of entity on the left and more than one instance of entity on the right can be associated with the relationship. It depicts many-to-many relationship



[*Image: Many-to-many*]

### Participation Constraints

* **Total Participation:** Each entity in the entity is involved in the relationship. Total participation is represented by double lines.
* **Partial participation:** Not all entities are involved in the relation ship. Partial participation is represented by single line.

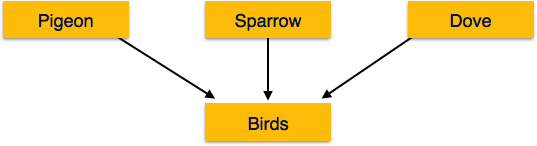


ER Model has the power of expressing database entities in conceptual hierarchical manner such that, as the hierarchical goes up it generalize the view of entities and as we go deep in the hierarchy it gives us detail of every entity included.

Going up in this structure is called generalization, where entities are clubbed together to represent a more generalized view. For example, a particular student named, Mira can be generalized along with all the students, the entity shall be student, and further a student is person. The reverse is called specialization where a person is student, and that student is Mira.

## Generalization

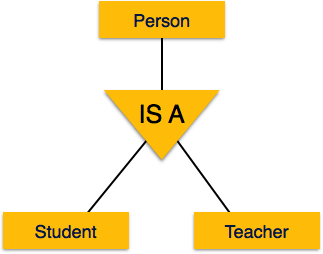
As mentioned above, the process of generalizing entities, where the generalized entities contain the properties of all the generalized entities is called Generalization. In generalization, a number of entities are brought together into one generalized entity based on their similar characteristics. For an example, pigeon, house sparrow, crow and dove all can be generalized as Birds.



[*Image: Generalization*]

## Specialization

Specialization is a process, which is opposite to generalization, as mentioned above. In specialization, a group of entities is divided into sub-groups based on their characteristics. Take a group Person for example. A person has name, date of birth, gender etc. These properties are common in all persons, human beings. But in a company, a person can be identified as employee, employer, customer or vendor based on what role do they play in company.



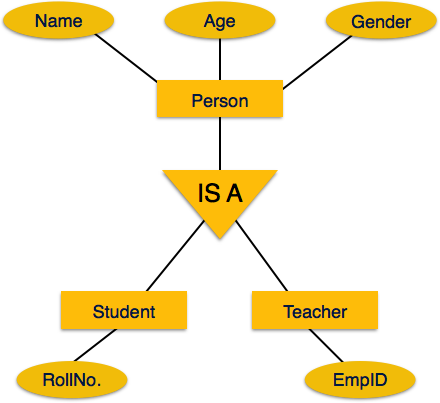
[*Image: Specialization*]

Similarly, in a school database, a person can be specialized as teacher, student or staff; based on what role do they play in school as entities.

## Inheritance

We use all above features of ER-Model, in order to create classes of objects in object oriented programming. This makes it easier for the programmer to concentrate on what she is programming. Details of entities are generally hidden from the user, this process known as abstraction.

One of the important features of Generalization and Specialization, is inheritance, that is, the attributes of higher-level entities are inherited by the lower level entities.



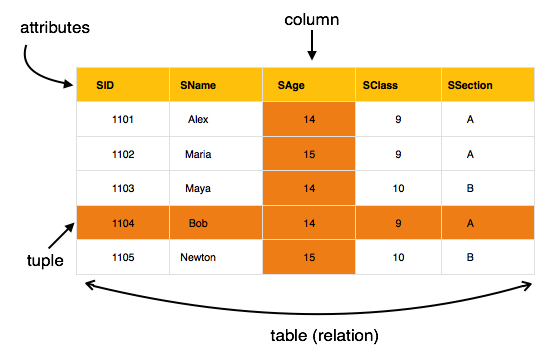
[*Image: Inheritance*]

For example, attributes of a person like name, age, and gender can be inherited by lower level entities like student and teacher etc.

Relational data model is the primary data model, which is used widely around the world for data storage and processing. This model is simple and have all the properties and capabilities required to process data with storage efficiency.

**Relational Model**

The most popular data model in DBMS is Relational Model. It is more scientific model then others.



[*Image: Table in relational Model*]

The main highlights of this model are:

* Data is stored in tables called relations.
* Relations can be normalized.
* In normalized relations, values saved are values.
* Each row in relation contains unique value
* Each column in relation contains values from a same domain.

## Concepts

**Tables:** In relation data model, relations are saved in the format of Tables. This format stores the relation among entities. A table has rows and columns, where rows represent records and columns represents the attributes.

**Tuple:** A single row of a table, which contains a single record for that relation is called a tuple.

**Relation instance:** A finite set of tuples in the relational database system represents relation instance. Relation instances do not have duplicate tuples.

**Relation schema:** This describes the relation name (table name), attributes and their names.

**Relation key:** Each row has one or more attributes which can identify the row in the relation (table) uniquely, is called the relation key.

**Attribute domain:** Every attribute has some pre-defined value scope, known as attribute domain.

## Constraints

Every relation has some conditions that must hold for it to be a valid relation. These conditions are called Relational Integrity Constraints. There are three main integrity constraints.

* Key Constraints
* Domain constraints
* Referential integrity constraints

### Key Constraints:

There must be at least one minimal subset of attributes in the relation, which can identify a tuple uniquely. This minimal subset of attributes is called **key** for that relation. If there are more than one such minimal subsets, these are called ***candidate keys***.

Key constraints forces that:

* in a relation with a key attribute, no two tuples can have identical value for key attributes.
* key attribute can not have NULL values.

Key constrains are also referred to as Entity Constraints.

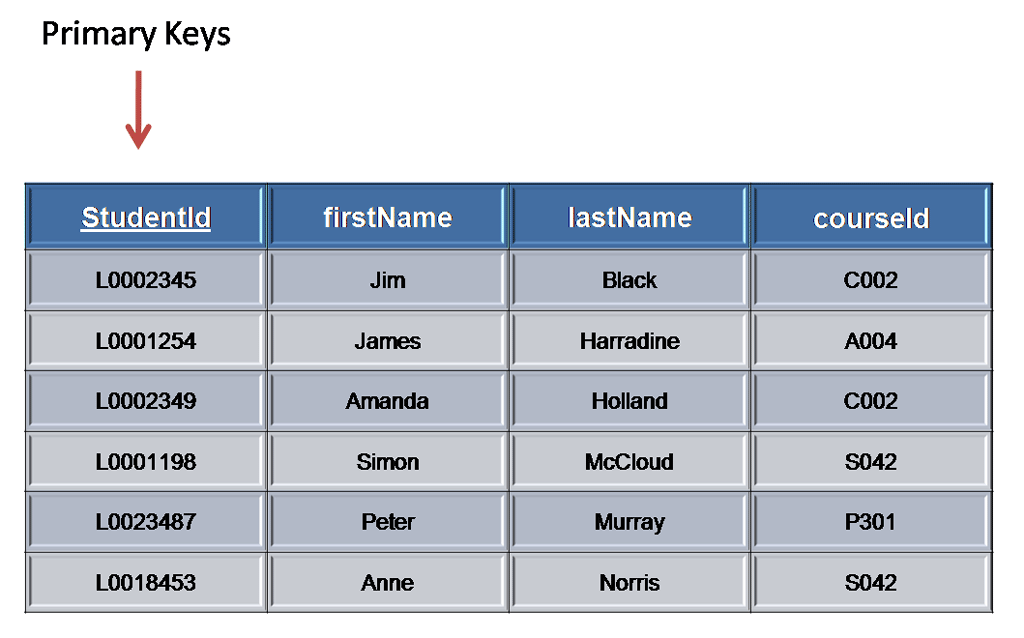
**Types of Keys**

**Primary Key**

A primary key is a candidate key that is most appropriate to be the main reference key for the table. As its name suggests, it is the primary key of reference for the table and is used throughout the database to help establish relationships with other

tables. As with any candidate key the primary key must contain unique values, must never be null and uniquely identify each record in the table.

As an example, a student id might be a primary key in a student table, a department code in a table of all departments in an organisation. This module has the code DH3D 35 that is no doubt used in a database somewhere to identify RDBMS as a unit in a table of modules. In the table below we have selected the candidate key student\_id to be our most appropriate primary key

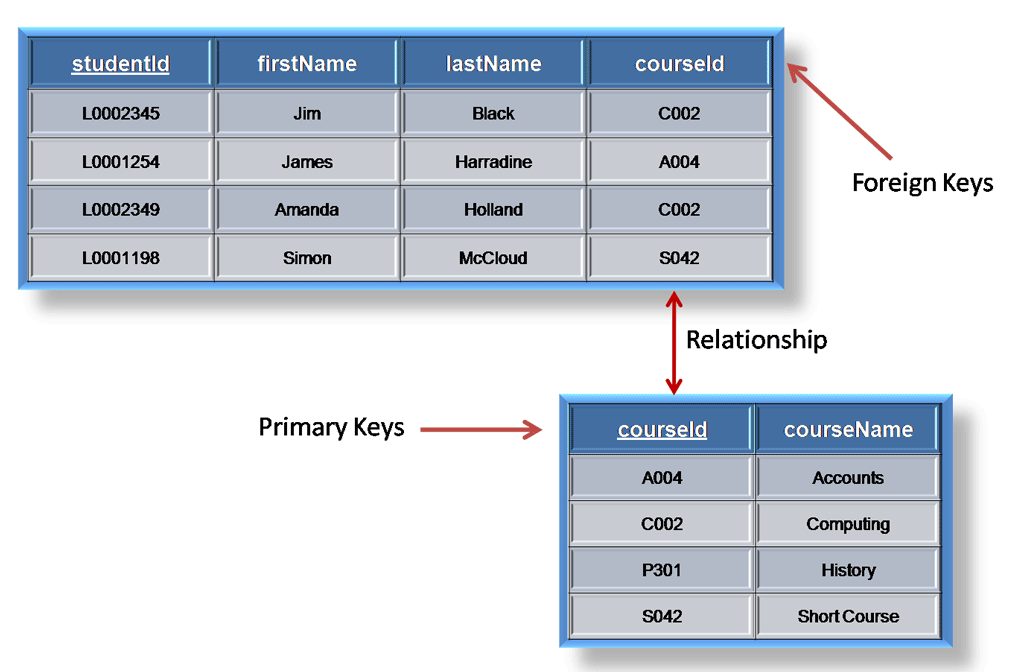


Primary keys are mandatory for every table each record must have a value for its primary key. When choosing a primary key from the pool of candidate keys always choose a single simple key over a composite key.

**Foreign Key**

A foreign key is generally a primary key from one table that appears as a field in another where the first table has a relationship to the second. In other words, if we had a table A with a primary key X that linked to a table B where X was a field in B, then X would be a foreign key in B.

An example might be a student table that contains the course\_id the student is attending. Another table lists the courses on offer with course\_id being the primary key. The 2 tables are linked through course\_id and as such course\_id would be a foreign key in the student table.



### Domain constraints

Attributes have specific values in real-world scenario. For example, age can only be positive integer. The same constraints has been tried to employ on the attributes of a relation. Every attribute is bound to have a specific range of values. For example, age can not be less than zero and telephone number cannot be outside 0-9.

### Referential integrity constraints

This integrity constraints works on the concept of Foreign Key. A key attribute of a relation can be referred in other relation, where it is called ***foreign key***.

Referential integrity constraint states that if a relation refers to a key attribute of a different or same relation, that key element must exists.

ER Model when conceptualized into diagrams gives a good overview of entity-relationship, which is easier to understand. ER diagrams can be mapped to Relational schema that is, it is possible to create relational schema using ER diagram. Though we cannot import all the ER constraints into Relational model but an approximate schema can be generated.

There are more than one processes and algorithms available to convert ER Diagrams into Relational Schema. Some of them are automated and some of them are manual process. We may focus here on the mapping diagram contents to relational basics.

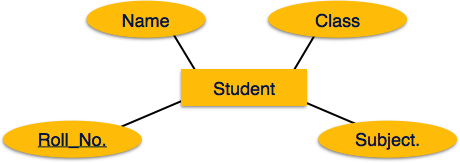
ER Diagrams mainly comprised of:

* Entity and its attributes
* Relationship, which is association among entities.

**Mapping Entity**

An entity is a real world object with some attributes.

Mapping Process (Algorithm):



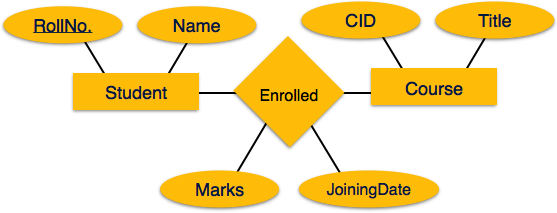
[*Image: Mapping Entity*]

* Create table for each entity
* Entity's attributes should become fields of tables with their respective data types.
* Declare primary key

**Mapping relationship**

A relationship is association among entities.

Mapping process (Algorithm):



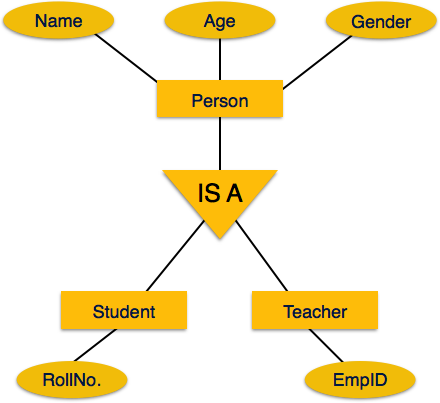
[*Image: Mapping relationship*]

* Create table for a relationship
* Add the primary keys of all participating entities as fields of table with their respective data types.
* If relationship has any attribute, add each attribute as field of table.
* Declare a primary key composing all the primary keys of participating entities.
* Declare all foreign key constraints.

**Mapping hierarchical entities**

ER specialization or generalization comes in the form of hierarchical entity sets.

Mapping process (Algorithm):



[*Image: Mapping hierarchical entities*]

* Create tables for all higher level entities
* Create tables for lower level entities
* Add primary keys of higher level entities in the table of lower level entities
* In lower level tables, add all other attributes of lower entities.
* Declare primary key of higher level table the primary key for lower level table
* Declare foreign key constraints.

SQL is a programming language for Relational Databases. It is designed over relational algebra and tuple relational calculus. SQL comes as a package with all major distributions of RDBMS.

SQL comprises both data definition and data manipulation languages. Using the data definition properties of SQL, one can design and modify database schema whereas data manipulation properties allows SQL to store and retrieve data from database.

## Data definition Language

SQL uses the following set of commands to define database schema:

### CREATE

Creates new databases, tables and views from RDBMS

For example:

Create database tutorial;

Create table article;

Create view for\_students;

### DROP

Drop commands deletes views, tables and databases from RDBMS

Drop object\_type object\_name;

Drop database tutorial;

Drop table article;

Drop view for\_students;

### ALTER

Modifies database schema.

Alter object\_type object\_name parameters;

for example:

Alter table article add subject varchar;

This command adds an attribute in relation article with name subject of string type.

## Data Manipulation Language

SQL is equipped with data manipulation language. DML modifies the database instance by inserting, updating and deleting its data. DML is responsible for all data modification in databases. SQL contains the following set of command in DML section:

* SELECT/FROM/WHERE
* INSERT INTO/VALUES
* UPDATE/SET/WHERE
* DELETE FROM/WHERE

These basic constructs allows database programmers and users to enter data and information into the database and retrieve efficiently using a number of filter options.

### SELECT/FROM/WHERE

* **SELECT**

This is one of the fundamental query command of SQL. It is similar to projection operation of relational algebra. It selects the attributes based on the condition described by WHERE clause.

* **FROM**

This clause takes a relation name as an argument from which attributes are to be selected/projected. In case more than one relation names are given this clause corresponds to cartesian product.

* **WHERE**

This clause defines predicate or conditions which must match in order to qualify the attributes to be projected.

For example:

Select author\_name

From book\_author

Where age > 50;

This command will project names of author’s from book\_author relation whose age is greater than 50.

### INSERT INTO/VALUES

This command is used for inserting values into rows of table (relation).

Syntax is

INSERT INTO table (column1 [, column2, column3 ... ]) VALUES (value1 [, value2, value3 ... ])

Or

INSERT INTO table VALUES (value1, [value2, ... ])

For Example:

INSERT INTO tutorial (Author, Subject) VALUES ("anonymous", "computers");

### UPDATE/SET/WHERE

This command is used for updating or modifying values of columns of table (relation).

Syntax is

UPDATE table\_name SET column\_name = value [, column\_name = value ...] [WHERE condition]

For example:

UPDATE tutorial SET Author="webmaster" WHERE Author="anonymous";

### DELETE/FROM/WHERE

This command is used for removing one or more rows from table (relation).

Syntax is

DELETE FROM table\_name [WHERE condition];

For example:

DELETE FROM tutorial

WHERE Author="unknown";