C.P.PATEL & F.H.SHAH COMMERCE COLLEGE
(MANAGED BY SARDAR PATEL EDUCATION TRUST)
BCA, BBA (ITM) & PGDCA PROGRAMME:

**BBA(ITM) Semester V - Paper Code: UM05CBBI09**

**UNIT 1- Introduction to RDBMS and SQL**

| Sr. No. | Topics |
|---|---|
| 1 | Introduction to RDBMS and SQL |
| 2 | Data models – Hierarchical, Network, Relational: Concepts and Terminology, |
| 3 | E-R Diagram: Entities, Attributes and Types of Relationships; |
| 4 | Introduction to DBMS – Terminology, Advantages, Keys; Normalization (1NF, 2NF, 3NF); |
| 5 | Introduction to SQL- Types of SQL Statements: DDL (Data Definition Language), DML (Data Manipulation Language), DQL (Data Query Language), DCL (Data Control Language), TCL (Transaction Control Language). |

## SQL Data models – Hierarchical, Network, Relational

**DBMS Database Models**

A Database model defines the logical design and structure of a database and defines how data will be stored, accessed and updated in a database management system. While the **Relational Model** is the most widely used database model, there are other models too:

1. Hierarchical Model
2. Network Model
3. Entity-relationship Model
4. Relational Model

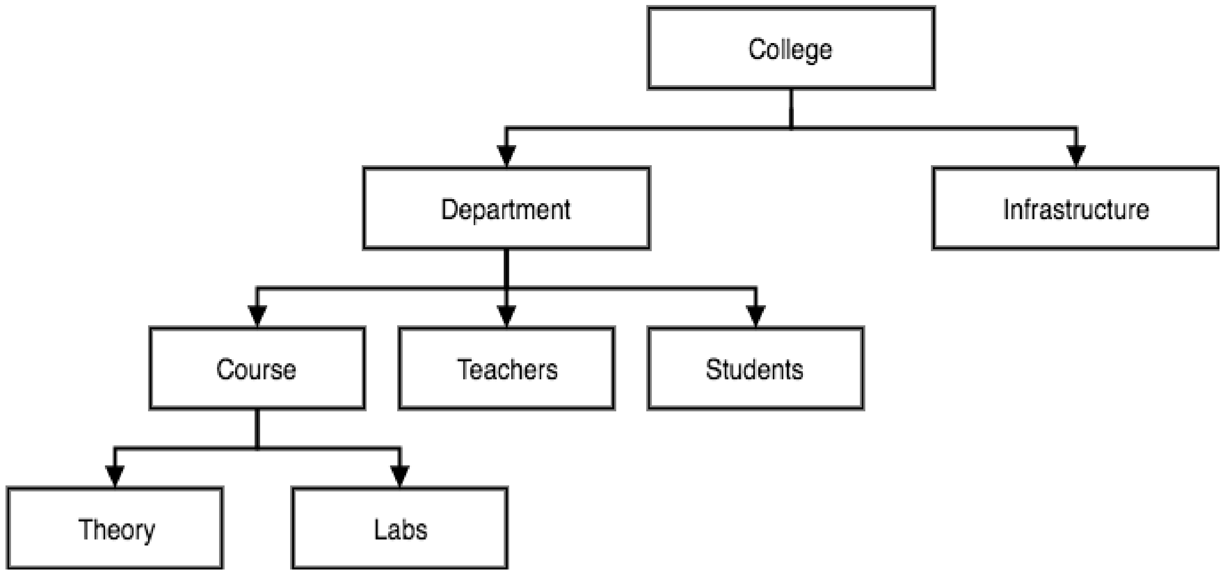
**Hierarchical Model**

This database model organises data into a tree-like-structure, with a single root, to which all the other data is linked. The hierarchy starts from the **Root** data, and expands like a tree, adding child nodes to the parent nodes.

In this model, a child node will only have a single parent node.

This model efficiently describes many real-world relationships like index of a book, recipes etc.

In hierarchical model, data is organised into tree-like structure with one one-to-many relationship between two different types of data, for example, one department can have many courses, many professors and of-course many students.

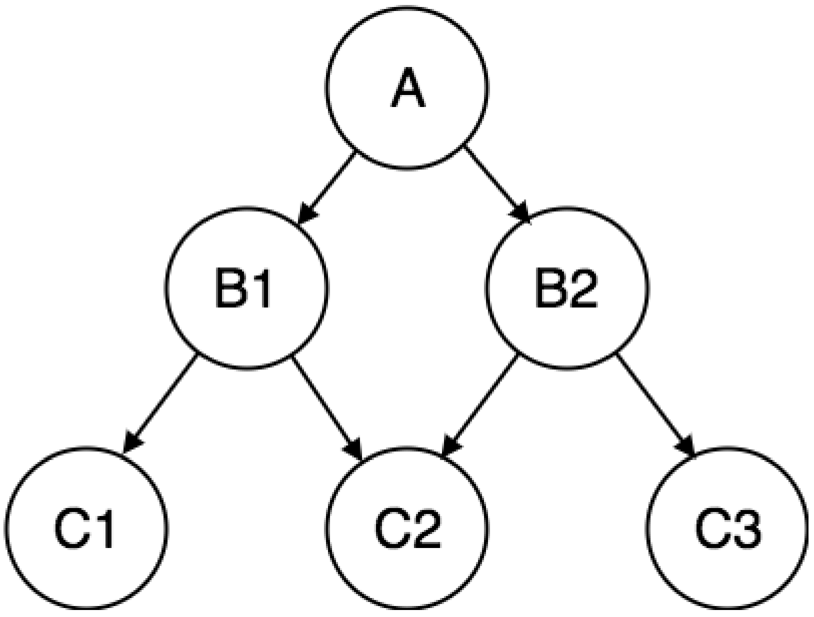


**Network Model**

This is an extension of the Hierarchical model. In this model data is organised more like a graph, and are allowed to have more than one parent node.

In this database model data is more related as more relationships are established in this database model. Also, as the data is more related, hence accessing the data is also easier and fast. This database model was used to map many-to-many data relationships.

This was the most widely used database model, before Relational Model was introduced.



**Entity-relationship Model**

In this database model, relationships are created by dividing object of interest into entity and its characteristics into attributes.

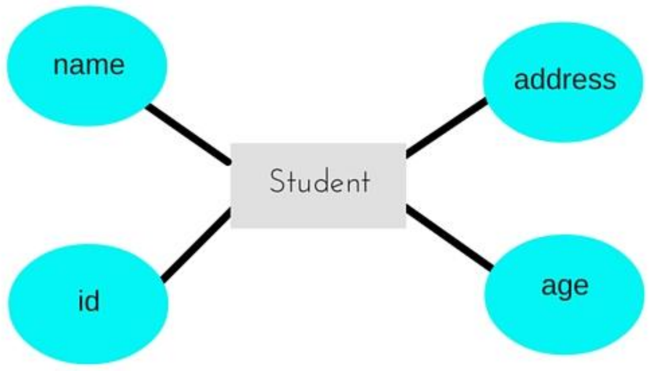
Different entities are related using relationships.

E-R Models are defined to represent the relationships into pictorial form to make it easier for different stakeholders to understand.

This model is good to design a database, which can then be turned into tables in relational model(explained below).

Let's take an example, If we have to design a School Database, then **Student** will be an **entity** with **attributes** name, age, address etc. As **Address** is generally complex, it can be another **entity** with **attributes** street name, pincode, city etc, and there will be a relationship between them.

Relationships can also be of different types. To learn about E-R Diagrams in details, click on the link.



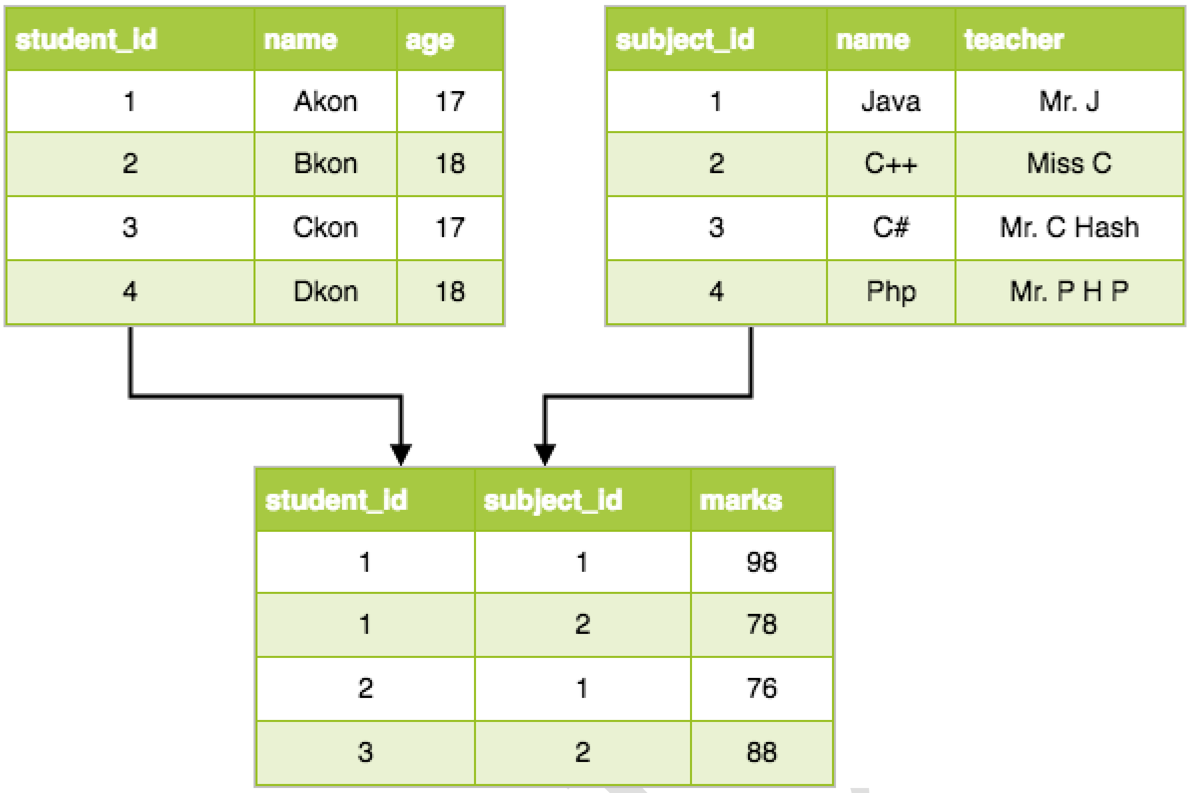
**Relational Model**
In this model, data is organised in two-dimensional **tables** and the relationship is maintained by storing a common field.

This model was introduced by E.F Codd in 1970, and since then it has been the most widely used database model, in fact, we can say the only database model used around the world.

The basic structure of data in the relational model is tables. All the information related to a particular type is stored in rows of that table.

Hence, tables are also known as **relations** in relational model.

In the coming tutorials we will learn how to design tables, normalize them to reduce data redundancy and how to use Structured Query language to access data from tables.



| student_id | name | age |
|---|---|---|
| 1 | Akon | 17 |
| 2 | Bkon | 18 |
| 3 | Ckon | 17 |
| 4 | Dkon | 18 |

| subject_id | name | teacher |
|---|---|---|
| 1 | Java | Mr. J |
| 2 | C++ | Miss C |
| 3 | C# | Mr. C Hash |
| 4 | Php | Mr. P H P |

| student_id | subject_id | marks |
|---|---|---|
| 1 | 1 | 98 |
| 1 | 2 | 78 |
| 2 | 1 | 76 |
| 3 | 2 | 88 |

## Entity Relationship Diagram (ER Diagram)

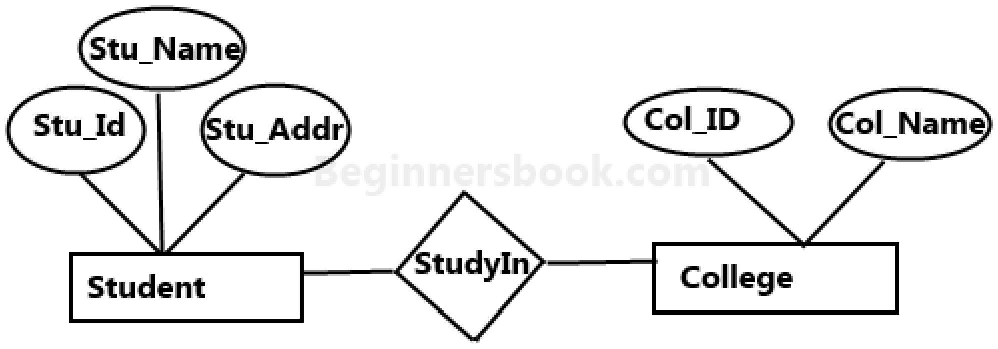
An **Entity–relationship model (ER model)** describes the structure of a database with the help of a diagram, which is known as **Entity Relationship Diagram (ER Diagram)**. An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

### What is an Entity Relationship Diagram (ER Diagram)?

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

### A simple ER Diagram:

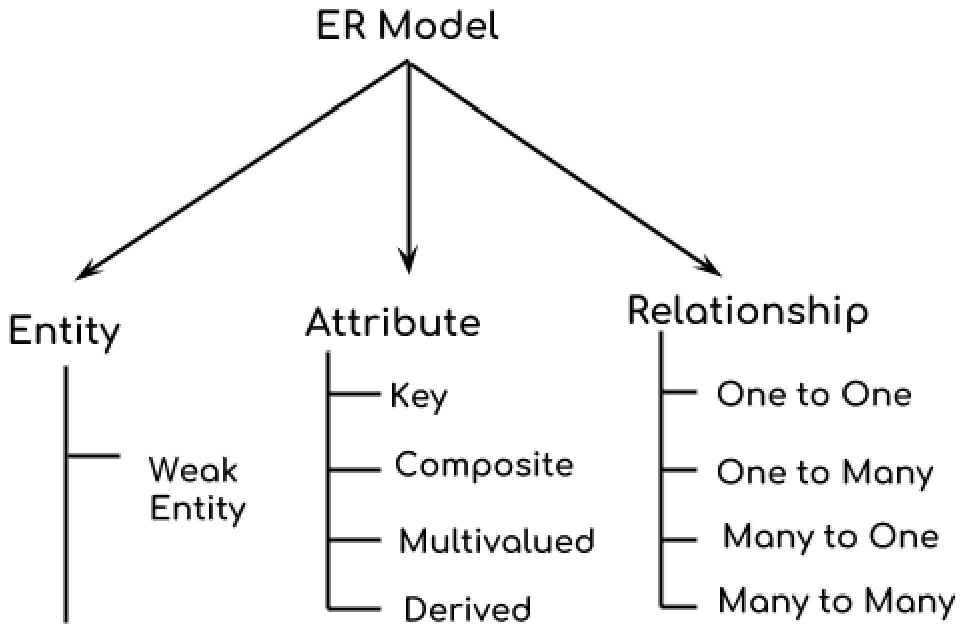
In the following diagram we have two entities Student and College and their relationship. The relationship between Student and College is many to one as a college can have many students however a student cannot study in multiple colleges at the same time. Student entity has attributes such as Stu_Id, Stu_Name & Stu_Addr and College entity has attributes such as Col_ID & Col_Name.



Sample E-R Diagram

Here are the geometric shapes and their meaning in an E-R Diagram. We will discuss these terms in detail in the next section(Components of a ER Diagram) of this guide so don't worry too much about these terms now, just go through them once.

**Rectangle**: Represents Entity sets.
**Ellipses**: Attributes
**Diamonds**: Relationship Set
**Lines**: They link attributes to Entity Sets and Entity sets to Relationship Set
**Double Ellipses:** Multivalued Attributes
**Dashed Ellipses**: Derived Attributes
**Double Rectangles**: Weak Entity Sets
**Double Lines**: Total participation of an entity in a relationship set
**Components of a ER Diagram**

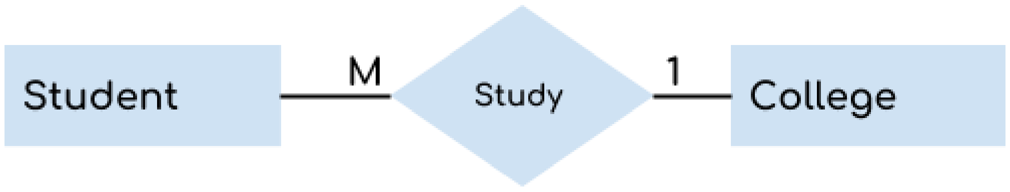


As shown in the above diagram, an ER diagram has three main components:
1. Entity
2. Attribute
3. Relationship

**1. Entity**

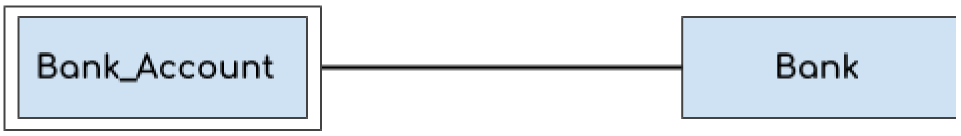
An entity is an object or component of data. An entity is represented as rectangle in an ER diagram.

For example: In the following ER diagram we have two entities Student and College and these two entities have many to one relationship as many students study in a single college. We will read more about relationships later, for now focus on entities.



**Weak Entity:**

An entity that cannot be uniquely identified by its own attributes and relies on the relationship with other entity is called weak entity. The weak entity is represented by a double rectangle. For example – a bank account cannot be uniquely identified without knowing the bank to which the account belongs, so bank account is a weak entity.



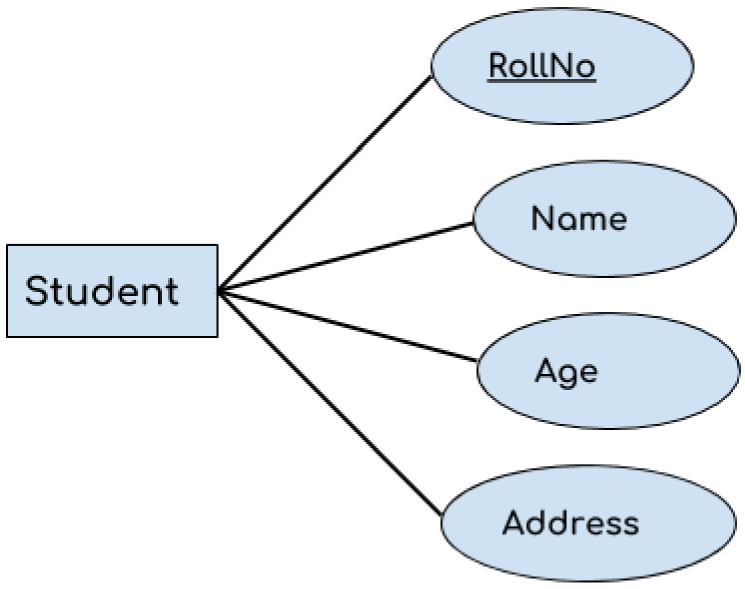
**2. Attribute**

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

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

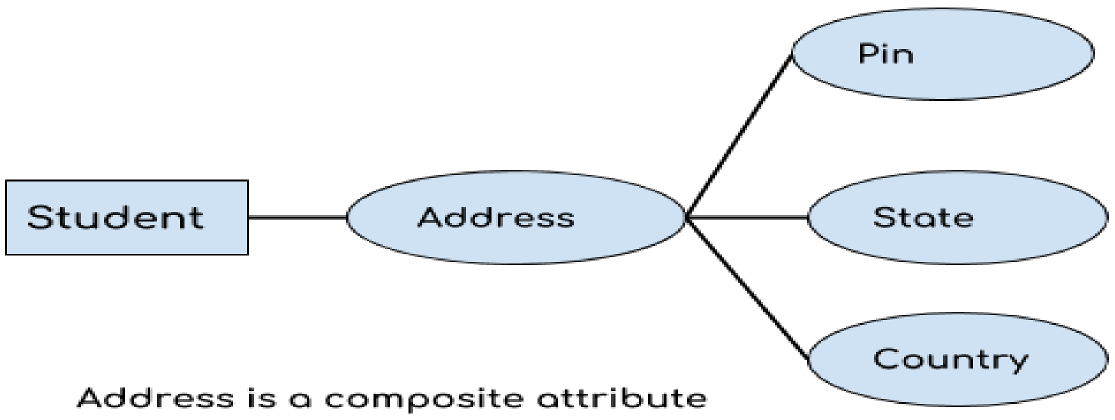
***1. Key attribute:***

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



**2. Composite attribute:**

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



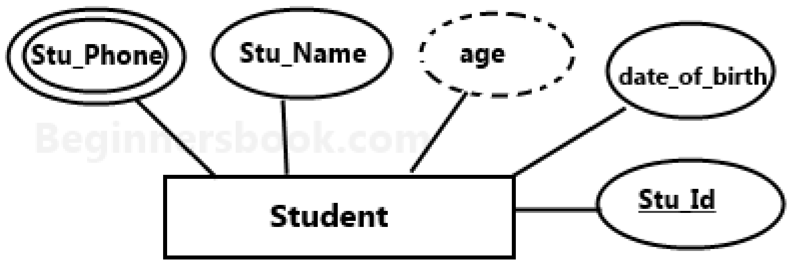
**3. Multivalued attribute:**

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

***4.* Derived attribute:**

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

**E-R diagram with multivalued and derived attributes**:

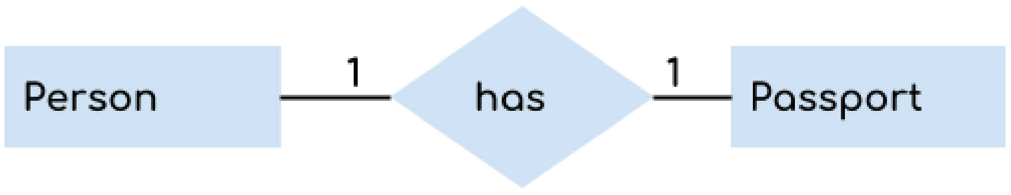


**3. Relationship**

A relationship is represented by diamond shape in ER diagram, it shows the relationship among entities. There are four types of relationships:
1. One to One
2. One to Many
3. Many to One
4. Many to Many

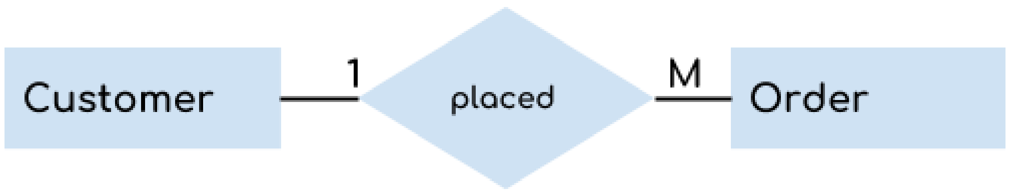
***1. One to One Relationship***

When a single instance of an entity is associated with a single instance of another entity then it is called one to one relationship. For example, a person has only one passport and a passport is given to one person.



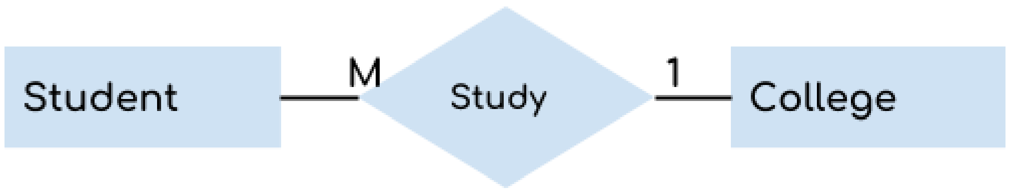
***2. One to Many Relationship***

When a single instance of an entity is associated with more than one instances of another entity then it is called one to many relationship. For example – a customer can place many orders but a order cannot be placed by many customers.



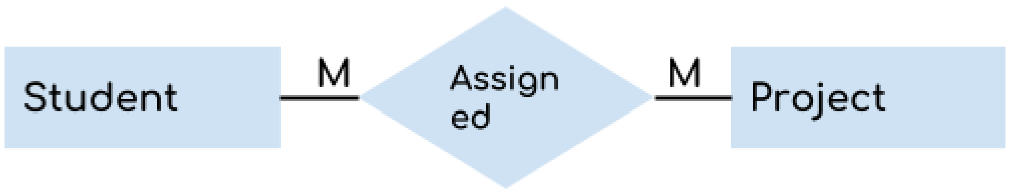
***3. Many to One Relationship***

When more than one instances of an entity is associated with a single instance of another entity then it is called many to one relationship. For example – many students can study in a single college but a student cannot study in many colleges at the same time.



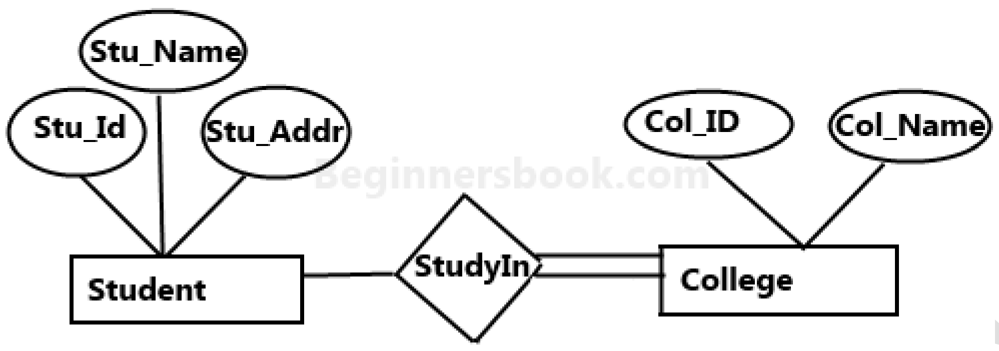
***4. Many to Many Relationship***

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



**Total Participation of an Entity set**

A Total participation of an entity set represents that each entity in entity set must have at least one relationship in a relationship set. For example: In the below diagram each college must have at-least one associated Student.



**E-R Digram with total participation of College entity set in StudyIn relationship Set - This indicates that each college must have atleast one associated Student.**

## Introduction to DBMS and RDBMS

➢ **WHAT IS DATABASE**

A database can be defined as a collection of coherent, meaningful data. The phrase collection of coherent data needs to have a point of reference to be understood. A simple point of reference would be the example of a postal address, this would generally contain:

- A building name,
- A flat number in the building
- A road name
- An area name
- A state name
- A pincode
- A country name (This is optional, but necessary if the letter is destined to a foreign country)

To project on this further, multiple addresses kept together in one place, such as an address book, could be termed as a coherent collection of data. Thus the address book is a database and the postal addresses in the book, is the data that fills the database.

➢ **WHAT IS DATABASE MANAGEMENT SYSTEMS (DBMS)**

To be able to successfully design and maintain databases we have to do the following:

1. Identify which part of the world' s data is of interest to us
2. Identify what specific objects in that part of the world's data are of interest
3. Identify a relationship between the objects

Hence, the objects, there attributes and the relationship between them (that are of interest to us) are stored in the database that is designed, built and populated with data for a specific purpose.

Software houses took up the cha1lenge of designing a system that would help in managing data in such a database. These systems were called Database Management Systems (DBMS).

DBMS is a system that allows inserting, updating, deleting and processing of data. Some of the DBMS developed by software houses were Oracle, Ingress, Sybase, Dbase 3+, Fox base, FoxPro, MS Access, Database, Data flex, Advanced Revelation, and so on.

**ADVANTAGES OF DBMS:-**

1. The amount of data redundancy in stored data can be reduced.
2. No more data inconsistencies.
3. Stored data can be shared by a single or multiple users.
4. Standards can be set and followed.
5. Data integrity can be maintained. Data integrity refers to the problem of ensuring that database contains only accurate data.
6. Security of data can be simply implemented.
7. Data independence can be achieved, i.e. data and programs that manipulate the data are two different entities.

**DISADVANTAGE OF DBMS:-**

1. Cost of Hardware / Software and Migration.
2. Problems associated with centralization.
3. Complexity of Backup and Recovery

**DBMS V/S RDBMS**

| **DBMS** | **RDBMS** |
|---|---|
| In DBMS relationship between two tables or files are maintained programmatically | In RDBMS relationship between two tables or files can<br>be specified at the time of table creation |
| DBMS does not support Client/Server Architecture | Most of the RDBMS supports Client/Server Architecture |
| DBMS does not supports Distributed databases | Most of the RDBMS supports Distributed databases |
| In DBMS there is no security of data | In RDBMS there are multiple levels of security<br>1. Logging in at O/S level<br>2. Command level<br>(i.e. at RDBMS level)<br>3. Object Level |
| Each table is given an extension in DBMS | Many tables are grouped in one database in RDBMS |
| DBMS may satisfy less than 7 to 8 rules of Dr. E F Codd | RDBMS usually satisfy more than 7 to 8 rules of Dr. E F Codd |

**Naming Conventions**

| **DBMS** | **RDBMS** |
|---|---|
| Field | Column, Attributes |
| Record | Row, Tuple, Entity |
| File | Table, Relation, Entity Class |

## Keys:-

➢ **PRIMARY KEY CONCEPTS:**

A primary key is one or more column(s) in a table that is used to identify each row in the table.

A multiple column primary key is called a **composite primary key.**

- No primary key value can appear in more than one row in the table.
- No column that is part of the primary key can contain a null.

CREATE TABLE dept

(deptno  NUMBER(2) CONSTRAINTS pk_dept PRIMARY KEY,

dname   VARCHAR2(9), loc VARCHAR2(10) )

➢ **UNIQUE KEY CONCEPTS:**

A unique key is similar to a primary key, except that the purpose of unique key is to ensure that information in the column for each row is unique. A table may have many unique keys.

CREATE TABLE dept

(deptno  NUMBER(2) UNIQUE,

dname   VARCHAR2(9), loc VARCHAR2(10) );

➢ **FOREIGN KEY CONCEPTS:**

Foreign key represents relationship between tables. A foreign key is a column (or a group of columns) whose values are derived from the primary key of the same or some other table.

The existence of a foreign key implies that the table with the foreign key is related to the primary key table from which the foreign key is derived. A foreign key must have a corresponding primary key value in the primary key table.

Example:

Consider the *stud* as a primary key table:

CREATE TABLE stud

(s_no NUMBER(3) PRIMARY KEY,

s_name   VARCHAR2(9), address VARCHAR2(15))

The foreign key table:

CREATE TABLE mark

(s_no NUMBER(3) REFERENCES stud(s_no),

mark1 NUMBER(3), mark2 NUMBER(3))

**NORMALIZATION**

Normalization is a process that helps analysts or database designers to design table structures for an application. The focus of normalization is to attempt to reduce redundant table data to the very minimum.

Through the normalization process, the collection of data in a single table is replaced, by the same data being distributed over multiple tables with a specific relationship being setup between the tables. By this process RDBMS schema designers try their best to reduce table data to the very minimum.

Normalization is carried out for the following reasons:
1. To structure the data between tables so that data maintenance is simplified
2. To allow data retrieval at optimal speed
3. To simplify data maintenance through updates, inserts and deletes
4. To reduce the need to restructure tables as new application requirements arise
5. To improve the quality of design for an application by rationalization of table data

Normalization is a technique that:
1. Decomposes data into two-dimensional tables
2. Eliminates any relationships in which table data does fully depend upon the primary key of a record
3. Eliminates any relationship that contains transitive dependencies

A description of the three forms of Normalization is as mentioned below.

**First Normal Form**

When a table is decomposed into two-dimensional tables with all repeating groups of data eliminated, the table data is said to be in its first normal form.

The repetitive portion of data belonging to the record is termed as repeating groups. To understand the application of normalization to table data the following table structure will be taken as an example:

***Table: EmpProj***

| **Field** | **Key** | **Type** |
|---|---|---|
| Project Number | -- | |
| Project Name | -- | |
| Employee Number | -- | 1-n |
| Employee Name | -- | 1-n |
| Rate Category | -- | 1-n |
| Hurly Rate | -- | 1-n |

1-n indicates that there are many occurrences of this field - it is a repeating group.

Data held in the above table structure

| **Project Number** | **Project Name** | **Employee Number** | **Employee Name** | **Rate Category** | **Hourly Rate** |
|---|---|---|---|---|---|
| P001 | My SQL & Linux | E001 | Sharanam | A | 7000 |

| | | | | | |
|---|---|---|---|---|---|
| P001 | My SL & Linux | E002 | Vaishali | B | 6500 |
| P001 | My SQL & Linux | E006 | Hansel | C | 4500 |
| P002 | Office & Linux | E001 | Sharanam | A | 7000 |
| P002 | Office & Linux | E007 | Chhaya | B | 4000 |

In the above data there are a few problems:

- The Project Name in the second record is misspelled. This can be solved, by removing duplicates. Do this using normalization
- Data is repeated and thus occupies more space

A table is in 1st normal form if:

- There are no repeating groups
- All the key attributes are defined
- All attributes are dependent on a primary key

So far there are no keys, and there are repeating groups. So remove the repeating groups, and define the primary key.

To convert a table to its First Normal Form:

1. The un-normalized data in the first table is the entire table

2. A key that will uniquely identify each record should be assigned to the table. This key has to be unique because it should be capable of identifying any specific row from the table for extracting information for use. This key is called the table's primary key.

This table is now in 1st normal form.

***Table: EmpProj***

| **Field** | **Key** |
|---|---|
| Project Number | Primary Key |
| Project Name | -- |
| Employee Number | Primary Key |
| Employee Name | -- |
| Rate Category | -- |
| Hurly Rate | -- |

**Second Normal Form**

A table is said to be in its second normal form when each record in the table is in the first normal form and each column in the record is fully dependent on its primary key.

A table is in 2nd normal form if:

- It's in 1st normal form
- It includes no partial dependencies (where an attribute is dependent on only a part of a primary key)

The steps to convert a table to its Second Normal Form:

1. Find and remove fields that are related to the only part of the key
2. Group the removed items in another table

3. Assign the new table with the key i.e. part of a whole composite key
Going through all the fields reveals the following:

- Project name is only dependent on Project number
- Employee name, Rate category and Hourly rate are dependent only on Employee number

To convert the table into the second normal form remove and place these fields in a separate table, with the key being that part of the original key they are dependent on.

This leads to the following 3 tables:

***Table: EmpProj***

| **Field** | **Key** |
|---|---|
| Project Number | Primary Key |
| Employee Number | Primary Key |

**Table: Proj**

| **Field** | **Key** |
|---|---|
| Project Number | Primary Key |
| Project Name | -- |

**Table: Emp**

| **Field** | **Key** |
|---|---|
| Employee Number | Primary Key |
| Employee Name | -- |
| Rate Category | -- |
| Hurly Rate | -- |

The table is now in 2nd normal form, but not yet in its 3rd normal form

**Third Normal Form**

Table data is said to be in third normal format when all transitive dependencies are removed from this data".

The table is in 3rd normal form if:

- It's in 2nd normal form
- It contains no transitive dependencies (where a non-key attribute is dependent on another non-key attribute)

A general case of transitive dependencies is as follows:
A, B, C are three columns in table.
If C is related to B
If B is related to A
Then C is indirectly related to A
This is when transitive dependency exists.
To convert such data to its third normal form remove this transitive dependency by splitting each relation in two separate relations. This means that data in columns A, B, C must be placed in three separate tables, which are linked using a foreign key

Going through all the fields reveals the following:

- Employee table is the only one with more than one non-key attribute
- Employee name is not dependent on either Rate category or Hourly rate
- Hourly rate is dependent on Rate category

To convert the table into the third normal form remove and place these fields in a separate table, with the attribute it was dependent on as key, as follows:

This leads to the following 4 tables:

***Table: EmpProj***

| **Field** | **Key** |
|---|---|
| Project Number | Primary Key |
| Employee Number | Primary Key |

**Table: Proj**

| **Field** | **Key** |
|---|---|
| Project Number | Primary Key |
| Project Name | -- |

**Table: Emp**

| **Field** | **Key** |
|---|---|
| Employee Number | Primary Key |
| Employee Name | -- |
| Rate Category | -- |

**Table: Rate**

| **Field** | **Key** |
|---|---|
| Rate Category | -- |
| Hurly Rate | -- |

These tables are all now in their 3rd normal form, and ready to be implemented. There are other normal forms such as Boyce-Codd normal form (BCNF), 4th and 5th normal form, but these are very rarely used for business applications. In most cases, tables that are in there 3rd normal form are already conforming to these types of table formats anyway.

**Circumstances under which Normalization can be avoided**

There are situations when normalization can be avoided. This can be seen in the following table structure:

**Table: Customer**

| **Field** | **Key** |
|---|---|
| Customer Number | Primary Key |
| Customer Name | -- |
| Address | -- |
| Pincode | -- |
| City | -- |

Going through all the fields reveals the following:

- No repeating groups exists
- A primary key is defined

This proves that it is at least in 1st normal form

- There is only one key so partial dependencies do not exists

This proves that it is at least in 2nd normal form

- There are transitive dependencies

It seems that like City might be determined by Pincode, which' is usually the case in most parts of the world, but there is no need to remove City, and place it in a separate table, with Pincode as the key.
Although this table is not technically in its 3rd normal format, removing this information is not worth it. Creating more tables increases the load slightly, slowing processing down. This is often counteracted by the reduction in table sizes, and redundant data. But in this case, where the City would almost always be referenced as part of the address, it isn't worth it. Normalization is just a helpful process that usually results in the most efficient table structure, and not a rule for database design.

## INTRODUCTION TO STRUCTURED QUERY LANGUAGE (SQL)

Structured Query Language (SQL) is a language that provides an interface to relational database systems. SQL was developed by IBM in the 1970s for use in System R, and is a de facto standard, as well as an ISO and ANSI standard. SQL is often pronounced SEQUEL.

In common usage SQL also encompasses DML (Data Manipulation Language), for Inserts, Updates, Deletes and DDL (Data Definition Language), used for creating and 111odifying tables and other database structures.

### ➢ What is SQL?

SQL (Structural Query Language) is a set of commands that lets you access a relational database. SQL is a standard interface for many relational databases. It has a simple command structure for data definition, access, and manipulation. SQL is set-oriented and non-procedural. When you use SQL you specify what to be done, not how to do it.

### ➢ Features of SQL:

1. SQL can be used by a range of users, including those with little or no programming experience.
2. It is a non-procedural language.
3. It reduces the amount of time required for creating and maintaining systems.
4. It is an English-like language.

### ➢ Rules for SQL

1. SQL starts with a verb {i.e. a SQL action word).
   Example: SELECT statements. This verb may have additional adjectives. Example: SELECT … FROM
2. Each verb is followed by number of clauses.
   Example: FROM, WHERE, HAVING
3. A space separates clauses. Example: DROP TABLE EMP;
4. A comma (,) separates parameters (without a clause)
5. A ';' is used to end SQL statements
6. Statements may be split across lines but keywords may not
7. Lexical units such as identifiers, operator names, literals are separated by one or more spaces or other delimiters that will not be confused with the lexical unit
8. Reserved words cannot be used as identifiers unless enclosed with double quotes. Reserved words are: ALL, DROP, USER, ORDER, DELETE, SELECT, INITIAL, ROWLABEL,

VIEW, EXISTS, UNIUE, INTEGER, AS, ASC, INTO, ALTER, ROWID, INSERT, VALUES, VARCHAR2 etc…

9. Identifiers can contain up to 30 characters and must start with an alphabetic character
10. Character and date literals must be enclosed within single quotes
11. Numeric literals can be represented by simple values such as 0.32, -34, 01991, and so on, Scientific notation as 2E5 meaning 2x1O to the power of 5 = 200,000.
12. Comments may be enclosed between /* and */ symbols and may be multi line. Single line comments may be prefixed with a- symbol

➢ **SQL Delimiters**

Delimiters are symbols or compound symbols, which have a special meaning within SQL and PL/SQL statements,

| | | | |
|---|---|---|---|
| + | Addition | “ | Quote Identifier |
| - | Subtraction | : | Host Variable |
| * | Multiplication | ** | Exponential |
| / | Division | <> != ^= | Relational |
| =>< | Relational | <= >= | Relational |
| () | Expression or list | := | Assignment |
| ; | Terminator | => | Association |
| % | Attribute indicator | \|\| | Concatenation |
| , | Item Separator | << | Label |
| . | Component Selector | ?? | Label |
| @ | Remote access indicator | -- | Comment (single line) |
| ‘ | Character string delimiter | /* */ | Comment (multi line) |

➢ **Components of SQL**

1. DDL (Data Definition Language)

It is a set of SQL commands used to create, modify and delete database structure but not data. These commands are normally not used by a general user, who should be accessing the database via an application. They are normally used by the DBA to a limited extent, a database designer or application developer. These statements are immediate i.e. they are not susceptible to ROLLBACK commands. It should also be noted that if several DML statements for example update's are executed then issuing any DDL command would COMMIT all the updates as every DDL c6mmand implicitly issues a COMMIT command to the database. Anybody using DDL must have the CREA TE object privilege and a Table space area in which to create objects.

2. DML (Data Manipulation Language

It is the area of SQL that allows changing data within the Database.

3. DCL (Data Control Language)

It is the component of SQL statement that control access to data and to the database. Occasionally DCL statements are grouped with DML statements.

4. DQL (Data Query Language)

It is the component of SQL statement that allows getting data from the database and imposing ordering upon it. In includes the SELECT statement. This command is heart of SQL. It allows getting the data out of the database perform operations with it. When SELECT is fired against a table or tables the result is compiled into a further temporary table, which displayed or perhaps received by the program i.e. a front-end.

- **Examples of DDL, DML and DCL commands**

**DDL: Data Definition Language Statements**

Examples:

CREATE - To create objects in the database
ALTER - alters the structure of the database
DROP - Delete objects from the database
TRUNCATE - Remove all records from a table, including all spaces allocated for the records are removed
COMMENT - Add comments to the data dictionary
GRANT - Gives user's access privileges to database
REVOKE - Withdraw access privileges given with the GRANT command

**DML: Data Manipulation Language statements**

Examples:

INSERT - Insert data into a table
UPDATE - Updates existing data within a table
DELETE - Deletes all records from a table, the space for the records remain
CALL - Call a PL/SQL or Java subprogram
EXPLAINS PLAN - Explain access path to data
LOCK - TABLE Control concurrency

**DCL: Data Control Language statements**

Examples:

COMMIT - Save work done
SAVEPOINT - Identify a point in a transaction to which you can later roll back
ROLLBACK - Restore database to original since the last COMMIT
SET TRANSACTION - Change transaction options like what rollback segment to use
GRANT / REVOKE - Grant or take back permissions to or from the oracle users

**DQL: Data Query Language statement**

Examples:

SELECT - Retrieve data from the a database