PG & REASEARCH DEPARTMENT OF COMPUTER SCIENCE

GOVERNMENT ARTS COLLEGE (GRADE-I) – ARIYALUR

DATABASE SYSTEMS

COURSE CODE: 16SCCCS4

Unit I

DATABASE APPLICATIONS

Applications where we use Database Management Systems are:

Telecom: There is a database to keeps track of the information regarding calls made, network usage, customer details etc. Without the database systems it is hard to maintain that huge amount of data that keeps updating every millisecond.

Industry: Where it is a manufacturing unit, warehouse or distribution centre, each one needs a database to keep the records of ins and outs. For example distribution centre should keep a track of the product units that supplied into the centre as well as the products that got delivered out from the distribution centre on each day; this is where DBMS comes into picture.

Banking System: For storing customer info, tracking day to day credit and debit transactions, generating bank statements etc. All this work has been done with the help of Database management systems.

Sales: To store customer information, production information and invoice details.

Airlines: To travel though airlines, we make early reservations, this reservation information along with flight schedule is stored in database.

Education sector: Database systems are frequently used in schools and colleges to store and retrieve the data regarding student details, staff details, course details, exam details, payroll data, attendance details, fees details etc. There is a hell lot amount of inter-related data that needs to be stored and retrieved in an efficient manner.

Online shopping: You must be aware of the online shopping websites such as Amazon, Flipkart etc. These sites store the product information, your addresses and preferences, credit details and provide you the relevant list of products based on your query. All this involves a Database management system.

PURPOSE OF DATABASE SYSTEMS

The main purpose of database systems is to manage the data. Consider a university that keeps the data of students, teachers, courses, books etc. To manage this data we need to store this data somewhere where we can add new data, delete unused data, update outdated data, retrieve data, to perform these operations on data we need a Database management system that allows us to store the data in such a way so that all these operations can be performed on the data efficiently

VIEW OF DATA IN DBMS

Abstraction is one of the main features of database systems. Hiding irrelevant details from user and providing abstract view of data to users, helps in easy and efficient **user-database** interaction. In the previous tutorial, we discussed the three level of DBMS architecture, The top level of that architecture is “view level”. The view level provides the “**view of data**” to the users and hides the irrelevant details such as data relationship, database schema, constraints, security etc from the user.

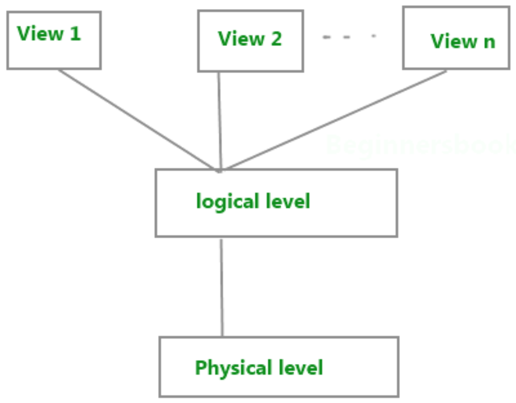
To fully understand the view of data, you must have a basic knowledge of data abstraction and instance & schema. Refer these two tutorials to learn them in detail.

DATA ABSTRACTION

INSTANCE AND SCHEMA

DATA ABSTRACTION IN DBMS

Database systems are made-up of complex data structures. To ease the user interaction with database, the developers hide internal irrelevant details from users. This process of hiding irrelevant details from user is called data abstraction.



Three Levels of data abstraction

We have three levels of abstraction:

PHYSICAL LEVEL:

This is the lowest level of data abstraction. It describes how data is actually stored in database. You can get the complex data structure details at this level.

LOGICAL LEVEL:

This is the middle level of 3-level data abstraction architecture. It describes what data is stored in database.

VIEW LEVEL:

Highest level of data abstraction. This level describes the user interaction with database system.

Example: Let's say we are storing customer information in a customer table. At **physical level** these records can be described as blocks of storage (bytes, gigabytes, terabytes etc.) in memory. These details are often hidden from the programmers.

At the **logical level** these records can be described as fields and attributes along with their data types, their relationship among each other can be logically implemented. The programmers generally work at this level because they are aware of such things about database systems.

At **view level**, user just interact with system with the help of GUI and enter the details at the screen, they are not aware of how the data is stored and what data is stored; such details are hidden from them.

INSTANCE AND SCHEMA IN DBMS

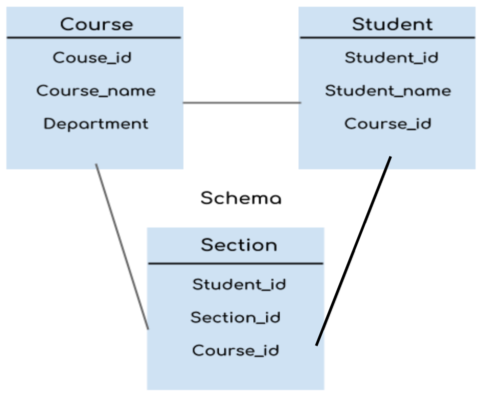
In this guide, we will learn what is an instance and schema in DBMS.

DBMS SCHEMA

DEFINITION OF SCHEMA:

Design of a database is called the schema. Schema is of three types: Physical schema, logical schema and view schema.

For example: In the following diagram, we have a schema that shows the relationship between three tables: Course, Student and Section. The diagram only shows the design of the database, it doesn't show the data present in those tables. Schema is only a structural view(design) of a database as shown in the diagram below.



The design of a database at physical level is called **physical schema**, how the data stored in blocks of storage is described at this level.

Design of database at logical level is called **logical schema**, programmers and database administrators work at this level, at this level data can be described as certain types of data records gets stored in data structures, however the internal details such as implementation of data structure is hidden at this level (available at physical level).

Design of database at view level is called **view schema**. This generally describes end user interaction with database systems.

To learn more about these schemas, refer 3 level data abstraction architecture.

DBMS INSTANCE

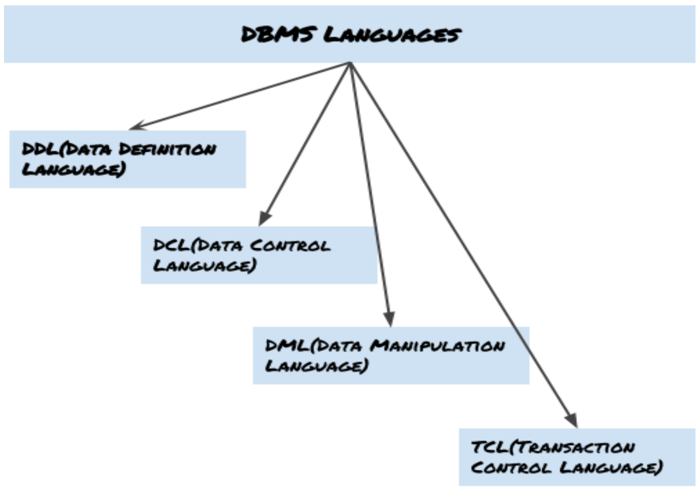
Definition of instance: The data stored in database at a particular moment of time is called instance of database. Database schema defines the variable declarations in tables that belong to a particular database; the value of these variables at a moment of time is called the instance of that database.

For example, lets say we have a single table student in the database, today the table has 100 records, so today the instance of the database has 100 records. Lets say we are going to add another 100 records in this table by tomorrow so the instance of database tomorrow will have 200 records in table. In short, at a particular moment the data stored in database is called the instance, that changes over time when we add or delete data from the database.

DBMS LANGUAGES

Database languages are used to read, update and store data in a database. There are several such languages that can be used for this purpose; one of them is SQL (Structured Query Language).

TYPES OF DBMS LANGUAGES:


DBMS Languages
DDL(Data Definition Language)
DCL(Data Control Language)
DML(Data Manipulation Language)
TCL(Transaction Control Language)

DATA DEFINITION LANGUAGE (DDL)

DDL is used for specifying the database schema. It is used for creating tables, schema, indexes, constraints etc. in database. Lets see the operations that we can perform on database using DDL:

- To create the database instance – CREATE
- To alter the structure of database – **ALTER**
- To drop database instances – DROP
- To delete tables in a database instance – **TRUNCATE**
- To rename database instances – **RENAME**
- To drop objects from database such as tables – **DROP**
- To Comment – **Comment**

All of these commands either defines or update the database schema that’s why they come under Data Definition language.

DATA MANIPULATION LANGUAGE (DML)

DML is used for accessing and manipulating data in a database. The following operations on database comes under DML:

- To read records from table(s) – SELECT
- To insert record(s) into the table(s) – **INSERT**
- Update the data in table(s) – UPDATE
- Delete all the records from the table – DELETE

DATA CONTROL LANGUAGE (DCL)

DCL is used for granting and revoking user access on a database –

- To grant access to user – GRANT
- To revoke access from user – REVOKE

In practical data definition language, data manipulation language and data control languages are not separate language, rather they are the parts of a single database language such as SQL.

TRANSACTION CONTROL LANGUAGE(TCL)

The changes in the database that we made using DML commands are either performed or rollbacked using TCL.

- To persist the changes made by DML commands in database – COMMIT
- To rollback the changes made to the database – ROLLBACK
-

RELATIONAL DATABASE

A relational database (RDB) is a collective set of multiple data sets organized by tables, records and columns. RDBs establish a well-defined relationship between database tables. Tables communicate and share information, which facilitates data searchability, organization and reporting.

RDBs use Structured Query Language (SQL), which is a standard user application that provides an easy programming interface for database interaction.

TECHOPEDIA EXPLAINS RELATIONAL DATABASE (RDB)

RDBs organize data in different ways. Each table is known as a relation, which contains one or more data category columns. Each table record (or row) contains a unique data instance defined for a corresponding column category. One or more data or record characteristics relate to one or many records to form functional dependencies. These are classified as follows:

- One to One: One table record relates to another record in another table.
- One to Many: One table record relates to many records in another table.
- Many to One: More than one table record relates to another table record.
- Many to Many: More than one table record relates to more than one record in another table.

RDB performs "select", "project" and "join" database operations, where select is used for data retrieval, project identifies data attributes, and join combines relations.

RDBs have many other advantages, including:

- Easy extendability, as new data may be added without modifying existing records. This is also known as scalability.
- New technology performance, power and flexibility with multiple data requirement capabilities.
- Data security, which is critical when data sharing is based on privacy. For example, management may share certain data privileges and access and block employees from other data, such as confidential salary or benefit information.

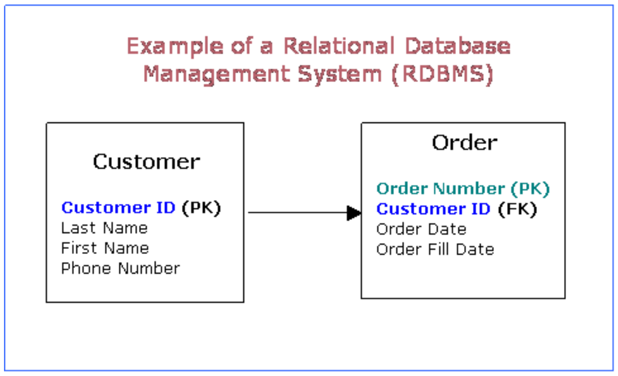
DATABASE DESIGN

A relational DBMS stores information in a set of "tables", each of which has a unique identifier or "primary key". The tables are then related to one another using "foreign keys".

A foreign key is simply the primary key in a different table. Diagrammatically, a foreign key is depicted as a line with an arrow at one end.

In the example below, "Customer ID" is the primary key (PK) in one table and the foreign key (FK) in another. The arrow represents a one-to-many relationship between the two tables. The relationship indicates that one customer can have one or more orders.

A given order, however, can be initiated by one and only one customer.



Example of a Relational Database

OBJECT BASED AND SEMISTRUCTURED DATABASES

Object DATABASE OR Object oriented database management system is a database in which the information is represented in form of object as used in object-oriented programming. It is different from rational database. This type of database is used when there is complex data or/and multiple data relationships. It have a many-to-many object relationship. It should not be used when there are few join tables and there are large volume of simple transaction data.

It works well with the following application:

Multimedia Application.
CAS Application

FEATURES OF OBJECT ORIENTED DATABASE:

- It support transactions.
- It supply querying in bulk data.
- Concurrent Access
- Security

EXAMPLE OF OBJECT DATABASE

In **Semi-Structured Database** the data are in the form of structured data that edoes not conform with the formal structure of data models associated with rational databases or other form of data. Therefore, it is also known as self-describing structure.

TYPES OF SEMI-STRUCTURED DATABASE:

- XML semi-structured database
- JSON (JavaScript Object Notation)semi-structured database

Advantages of Semi-Structured Database

- It can show the information of data source that is not constrained by schema.
- It is used to view structured data as semi-structured data.
- The data transfer format may be portable.

DATA STORAGE AND QUERYING

A database system is partitioned into modules that deal with each of the responsibilities of the overall system. The functional components of a database system can be broadly divided into the storage manager and the query processor components.

The storage manager is important because databases typically require a large amount of storage space. Corporate databases range in size from hundreds of gigabytes to, for the largest databases, terabytes of data. A gigabyte is approximately 1000 megabytes (actually 1024) (1 billion bytes), and a terabyte is 1 million megabytes (1 trillion bytes). Since the main memory of computers cannot store this much information, the information is stored on disks. Data are moved between disk storage and main memory as needed. Since the movement of data to and from disk is slow relative to the speed of the central processing unit, it is imperative that the database system structure the data so as to minimize the need to move data between disk and main memory.

The query processor is important because it helps the database system to simplify and facilitate access to data. The query processor allows database users to obtain good performance while being able to work at the view level and not be burdened with understanding the physical-level details of the implementation of he system. It is the job of the database system to translate updates and queries written in a nonprocedural language, at the logical level, into an efficient sequence of operations at the physical level.

1. STORAGE MANAGER

The storage manager is the component of a database system that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system. The storage manager is responsible for the interaction with the file manager. The raw data are stored on the disk using the file system provided by the operating system. The storage manager translates the various DML statements into low-level file-system commands. Thus, the storage manager is responsible for storing, retrieving, and updating data in the database.

The storage manager components include:

- **Authorization and integrity manager**, which tests for the satisfaction of integrity constraints and checks the authority of users to access data.
- **Transaction manager**, which ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicting.
- **File manager**, which manages the allocation of space on disk storage and the data structures used to represent information stored on disk.
- **Buffer manager**,which is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory. The buffer manager is a critical part of the database system, since it enables the database to handle data sizes that are much larger than the size of main memory.

The storage manager implements several data structures as part of the physical system implementation:

- **Data files**, which store the database itself.
- **Data dictionary**, which stores metadata about the structure of the database, in particular the schema of the database.
- **Indices**, which can provide fast access to data items. Like the index in this textbook, a database index provides pointers to those data items that hold a particular value. For example, we could use an index to find the instructor record with a particular ID, or all instructor records with a particular name. Hashing is an alternative to indexing that is faster in some but not all cases.

THE QUERY PROCESSOR

The query processor components include:

- **DDL interpreter**,which interprets DDL statements and records the definitions in the data dictionary.
- **DML compiler**,which translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands.
- A query can usually be translated into any of a number of alternative evaluation plans that all give the same result. The DMLcompiler also performs **query optimization**; that is, it picks the lowest cost evaluation plan from among the alternatives
- **Query evaluation engine**, which executes low-level instructions generated by the DML compiler.

DATABASE USERS AND ADMINISTRATORS

DATABASE USERS

Database users are the one who really use and take the benefits of database. There will be different types of users depending on their need and way of accessing the database.

- **Application Programmers -** They are the developers who interact with the database by means of DML queries. These DML queries are written in the application programs like C, C++, JAVA, Pascal etc. These queries are converted into object code to communicate with the database. For example, writing a C program to generate the report of employees who are working in particular department will involve a query to fetch the data from database. It will include a embedded SQL query in the C Program.
- **Sophisticated Users -** They are database developers, who write SQL queries to select/insert/delete/update data. They do not use any application or programs to request the database. They directly interact with the database by means of query language like SQL. These users will be scientists, engineers, analysts who thoroughly study SQL and DBMS to apply the

concepts in their requirement. In short, we can say this category includes designers and developers of DBMS and SQL.

- **Specialized Users -** These are also sophisticated users, but they write special database application programs. They are the developers who develop the complex programs to the requirement.
- **Stand-alone Users -** These users will have stand –alone database for their personal use. These kinds of database will have readymade database packages which will have menus and graphical interfaces.
- **Native Users -** these are the users who use the existing application to interact with the database. For example, online library system, ticket booking systems, ATMs etc which has existing application and users use them to interact with the database to fulfill their requests.

DATABASE ADMINISTRATORS

The life cycle of database starts from designing, implementing to administration of it. A database for any kind of requirement needs to be designed perfectly so that it should work without any issues. Once all the design is complete, it needs to be installed. Once this step is complete, users start using the database. The database grows as the data grows in the database. When the database becomes huge, its performance comes down. Also accessing the data from the database becomes challenge. There will be unused memory in database, making the memory inevitably huge. These administration and maintenance of database is taken care by database Administrator – DBA. A DBA has many responsibilities. A good performing database is in the hands of DBA.

- **Installing and upgrading the DBMS Servers: -** DBA is responsible for installing a new DBMS server for the new projects. He is also responsible for upgrading these servers as there are new versions comes in the market or requirement. If there is any failure in upgradation of the existing servers, he should be able revert the new changes back to the older version, thus maintaining the DBMS working. He is also responsible for updating the service packs/ hot fixes/ patches to the DBMS servers.
- **Design and implementation: -** Designing the database and implementing is also DBA's responsibility. He should be able to decide proper memory management, file organizations, error handling, log maintenance etc for the database.
- **Performance tuning: -** Since database is huge and it will have lots of tables, data, constraints and indices, there will be variations in the performance from time to time. Also, because of some designing issues or data growth, the database will not work as expected. It is responsibility of the DBA to tune the database performance. He is responsible to make sure all the queries and programs works in fraction of seconds.
- **Migrate database servers: -** Sometimes, users using oracle would like to shift to SQL server or Netezza. It is the responsibility of DBA to make sure that migration happens without any failure, and there is no data loss.
- **Backup and Recovery: -** Proper backup and recovery programs needs to be developed by DBA and has to be maintained him. This is one of the main responsibilities of DBA. Data/objects should be backed up regularly so that if there is any crash, it should be recovered without much effort and data loss.
- **Security: -** DBA is responsible for creating various database users and roles, and giving them different levels of access rights.
- **Documentation: -** DBA should be properly documenting all his activities so that if he quits or any new DBA comes in, he should be able to understand the database without any effort. He should basically maintain all his installation, backup, recovery, security methods. He should keep various reports about database performance.

TYPES OF DBA

There are different kinds of DBA depending on the responsibility that he owns.

- **Administrative DBA -** This DBA is mainly concerned with installing, and maintaining DBMS servers. His prime tasks are installing, backups, recovery, security, replications, memory management, configurations and tuning. He is mainly responsible for all administrative tasks of a database.
- **Development DBA -** He is responsible for creating queries and procedure for the requirement. Basically his task is similar to any database developer.
- **Database Architect -** Database architect is responsible for creating and maintaining the users, roles, access rights, tables, views, constraints and indexes. He is mainly responsible for designing the structure of the database depending on the requirement. These structures will be used by developers and development DBA to code.
- **Data Warehouse DBA -**DBA should be able to maintain the data and procedures from various sources in the datawarehouse. These sources can be files, COBOL, or any other programs. Here data and programs will be from different sources. A good DBA should be able to keep the performance and function levels from these sources at same pace to make the datawarehouse to work.
- **Application DBA -**He acts like a bridge between the application program and the database. He makes sure all the application program is optimized to interact with the database. He ensures all the activities from installing, upgrading, and patching, maintaining, backup, recovery to executing the records works without any issues.
- **OLAP DBA -** He is responsible for installing and maintaining the database in OLAP systems. He maintains only OLAP databases.

TRANSACTION MANAGEMENT IN DBMS

A **transaction** is a set of logically related operations. For example, you are transferring money from your bank account to your friend's account, the set of operations would be like this:

SIMPLE TRANSACTION EXAMPLE

1. Read your account balance
2. Deduct the amount from your balance
3. Write the remaining balance to your account
4. Read your friend's account balance
5. Add the amount to his account balance
6. Write the new updated balance to his account

This whole set of operations can be called a transaction. Although I have shown you read, write and update operations in the above example but the transaction can have operations like read, write, insert, update, delete.

In DBMS, we write the above 6 steps transaction like this:
Lets say your account is A and your friend's account is B, you are transferring 10000 from A to B, the steps of the transaction are:

```
1. R(A);
2. A = A - 10000;
3. W(A);
4. R(B);
5. B = B + 10000;
6. W(B);
```

In the above transaction **R** refers to the **Read operation** and **W** refers to the **write operation**.

TRANSACTION FAILURE IN BETWEEN THE OPERATIONS

Now that we understand what is transaction, we should understand what are the problems associated with it.

The main problem that can happen during a transaction is that the transaction can fail before finishing the all the operations in the set. This can happen due to power failure, system crash etc. This is a serious problem that can leave database in an inconsistent state. Assume that transaction fail after third operation (see the example above) then the amount would be deducted from your account but your friend will not receive it.

To solve this problem, we have the following two operations

Commit: If all the operations in a transaction are completed successfully then commit those changes to the database permanently.

Rollback: If any of the operation fails then rollback all the changes done by previous operations.

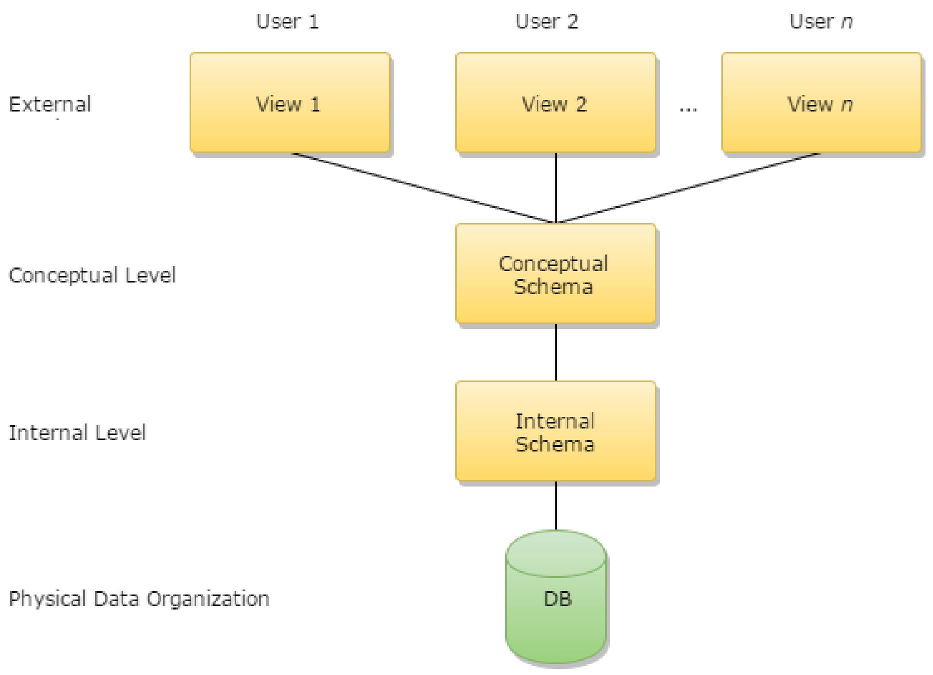
DATABASE ARCHITECTURE

Table of Contents

THREE-LEVEL ANSI-SPARC ARCHITECTURE

An early proposal for a standard terminology and general architecture for database systems was produced in 1971 by the DBTG (Data Base Task Group) appointed by the Conference on Data Systems and Languages (CODASYL, 1971). The DBTG recognized the need for a two-level approach with a system view called the schema and user views called sub-schemas.

Here is the figure showing the ANSI_SPARC Architecture of the database system:



ANSI_SPARC Architecture of the database system

The levels form a three-level architecture that includes an external, a conceptual, and an internal level. The way users recognize the data is called the external level. The way the DBMS and the operating system distinguish the data is the internal level, where the data is stored using the data structures and file. The conceptual level offers both the mapping and the desired independence between the external and internal levels.

WHAT IS DATABASE ARCHITECTURE?

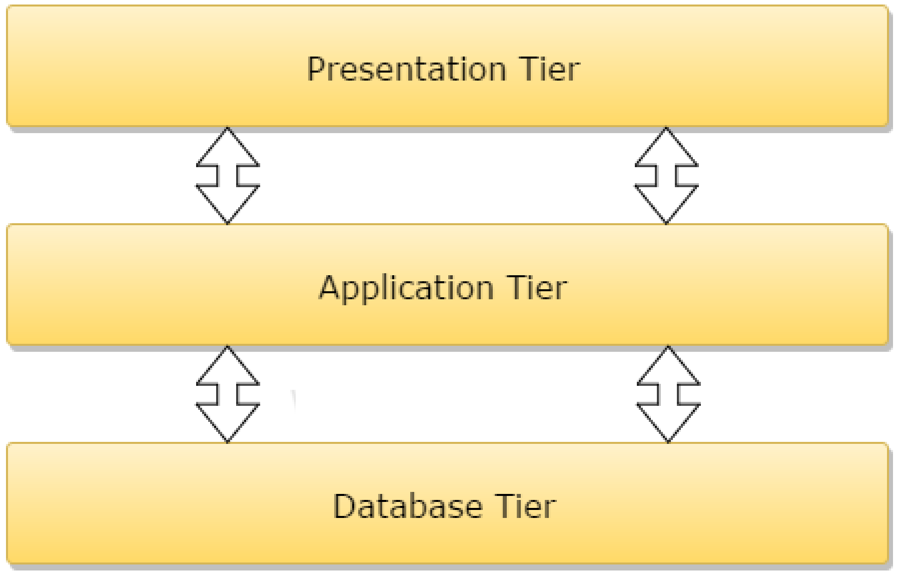
A DBMS architecture is depending on its design and can be of the following types:

- Centralized
- Decentralized
- Hierarchical

DBMS architecture can be seen as either a single tier or multi-tier. An architecture having n-tier splits the entire system into related but independent N modules that can be independently customized, changed, altered, or replaced.

The architecture of a database system is very much influenced by the primary computer system on which the database system runs. Database systems can be centralized, or client-server, where one server machine executes work on behalf of multiple client machines. Database systems can also be designed to exploit parallel computer architectures. Distributed databases span multiple geographically separated machines.

THE THREE-TIER ARCHITECTURE



The Three Tier Architecture

A 3-tier application is an application program that is structured into three major parts; each of them is distributed to a different place or places in a network. These three divisions are as follows:

- The workstation or presentation layer
- The business or application logic layer
- The database and programming related to managing layer

HISTORY OF DATABASE MANAGEMENT

A Database Management System allows a person to organize, store, and retrieve data from a computer. It is a way of communicating with a computer's "stored memory." In the very early years of computers, "punch cards" were used for input, output, and data storage. Punch cards offered a fast way to enter data, and to retrieve it. Herman Hollerith is given credit for adapting the punch cards used for weaving looms to act as the memory for a mechanical tabulating machine, in 1890. Much later, databases came along.

Databases (or DBs) have played a very important part in the recent evolution of computers. The first computer programs were developed in the early 1950s, and focused almost completely on coding languages and algorithms. At the time, computers were basically giant calculators and data (names, phone numbers) was considered the leftovers of processing information. Computers were just starting to become commercially available, and when business people started using them for real-world purposes, this leftover data suddenly became important.

Enter the Database Management System (DBMS). A database, as a collection of information, can be organized so a Database Management System can access and pull specific information. In 1960, Charles W. Bachman designed the Integrated Database System, the "first" DBMS. IBM, not wanting to be left out, created a database system of their own, known as IMS. Both database systems are described as the forerunners of navigational databases.

By the mid-1960s, as computers developed speed and flexibility, and started becoming popular, many kinds of general use database systems became available. As a result, customers demanded a standard be developed, in turn leading to Bachman forming the Database Task Group. This group took responsibility for the design and standardization of a language called Common Business Oriented Language (COBOL). The Database Task Group presented this standard in 1971, which also came to be known as the "CODASYL approach."

The CODASYL approach was a very complicated system and required substantial training. It depended on a "manual" navigation technique using a linked data set, which formed a large network. Searching for records could be accomplished by one of three techniques:

- Using the primary key (also known as the CALC key)
- Moving relationships (also called sets) to one record from another
- Scanning all records in sequential order

Eventually, the CODASYL approach lost its popularity as simpler, easier-to-work-with systems came on the market.

Edgar Codd worked for IBM in the development of hard disk systems, and he was not happy with the lack of a search engine in the CODASYL approach, and the IMS model. He wrote a series of papers, in 1970, outlining novel ways to construct databases. His ideas eventually evolved into a paper titled, A RELATIONAL MODEL OF DATA FOR LARGE SHARED DATA BANKS, which described new method for storing data and processing large databases. Records would not be stored in a free-form list of linked records, as in CODASYL navigational model, but instead used a "table with fixed-length records."

IBM had invested heavily in the IMS model, and wasn't terribly interested in Codd's ideas. Fortunately, some people who didn't work for IBM "were" interested. In 1973, Michael Stonebraker and Eugene Wong (both then at UC Berkeley) made the decision to research relational database systems. The project was called INGRES (INTERACTIVE GRAPHICS AND RETRIEVAL SYSTEM), and successfully demonstrated a relational model could be efficient and practical. INGRES worked with a query language known as QUEL, in turn, pressuring IBM to develop SQL in 1974, which was more advanced (SQL became ANSI and OSI standards in 1986 1nd 1987). SQL quickly replaced QUEL as the more functional query language.

RDBM Systems were an efficient way to store and process structured data. Then, processing speeds got faster, and "unstructured" data (art, photographs, music, etc.) became much more common place. Unstructured data is both non-relational and schema-less, and Relational Database Management Systems simply were not designed to handle this kind of data.

ER MODEL

The ER model defines the conceptual view of a database. It works around real-world entities and the associations among them. At view level, the ER model is considered a good option for designing databases.

ENTITY

An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity.

An entity set is a collection of similar types of entities. An entity set may contain entities with attribute sharing similar values. For example, a Students set may contain all the students of a school; likewise a Teachers set may contain all the teachers of a school from all faculties. Entity sets need not be disjoint.

ATTRIBUTES

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

There exists 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, a 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 the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, average_salary in a department should not be saved directly in the database, instead it can be derived. For another example, age can be derived from data_of_birth.
- **Single-value attribute** − Single-value attributes contain single value. For example − Social_Security_Number.
- **Multi-value attribute** − Multi-value attributes may contain more than one values. For example, a person can have more than one phone number, email_address, 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, the roll_number of a student makes him/her identifiable among students.

- **Super Key** − A set of attributes (one or more) that collectively identifies an entity in an entity set.
- **Candidate Key** − A minimal super key is called a candidate key. An entity set may have more than one candidate key.
- **Primary Key** − A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

RELATIONSHIP

The association among entities is called a relationship. For example, an employee **works_at** a department, a student **enrolls** in a course. Here, Works_at and Enrolls are called relationships.

RELATIONSHIP SET

A set of relationships of similar type is called a relationship set. Like entities, a relationship too can have attributes. These attributes are called **descriptive attributes**.

DEGREE OF RELATIONSHIP

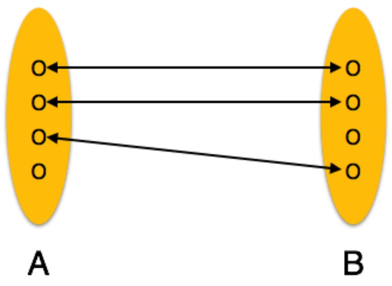
The number of participating entities in a relationship defines the degree of the relationship.

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

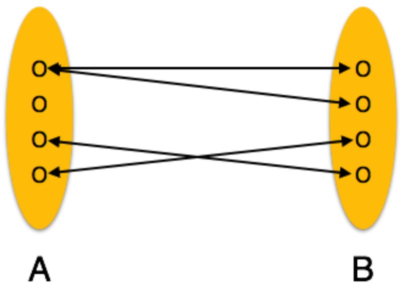
MAPPING CARDINALITIES

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

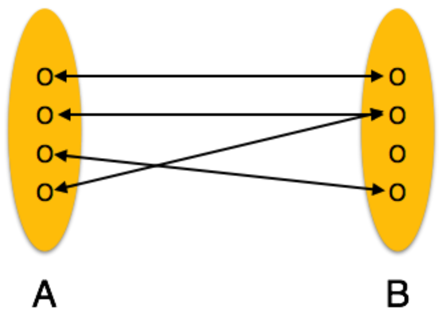
- **One-to-one** − One entity from entity set A can be associated with at most one entity of entity set B and vice versa.



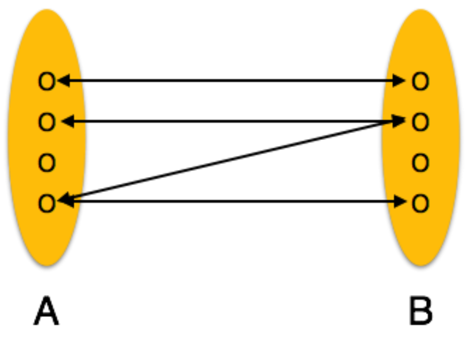
- **One-to-many** − One entity from entity set A can be associated with more than one entities of entity set B however an entity from entity set B, can be associated with at most one entity.



- **Many-to-one** − More than one entities from entity set A can be associated with at most one entity of entity set B, however an entity from entity set B can be associated with more than one entity from entity set A.



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



ENTITY RELATIONSHIP DIAGRAM – ER DIAGRAM

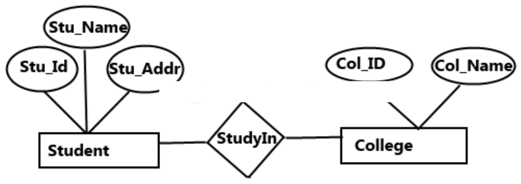
An **Entity–relationship model (ER model)** describes the structure of a database with the help of a diagram, which is known as **Entity Relationship Diagram (ER Diagram)**. An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model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

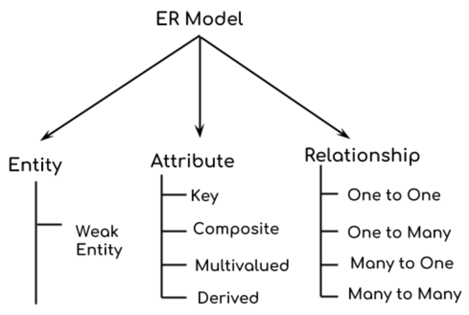


Sample E-R Diagram

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

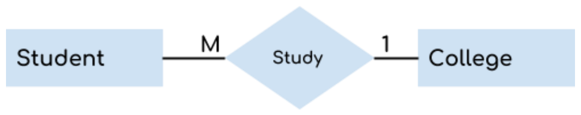


Components of ER Diagram

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

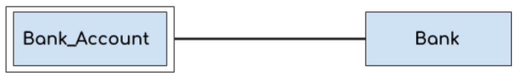
ENTITY

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.



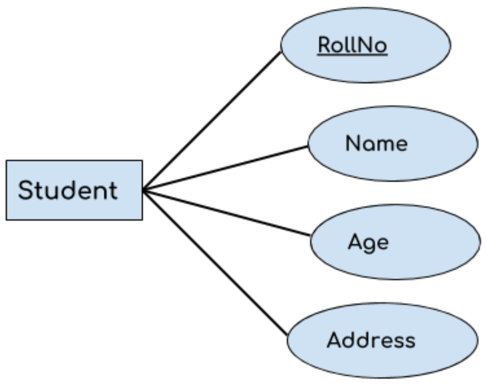
2. ATTRIBUTE

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

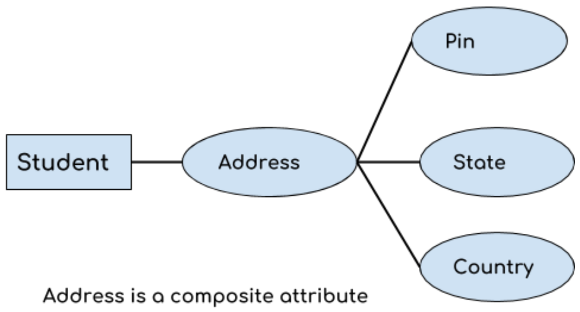
1. KEY ATTRIBUTE:

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.



Address is a composite attribute

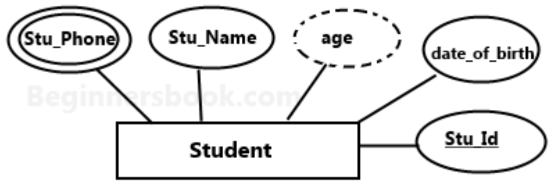
3. MULTIVALUED ATTRIBUTE:

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:

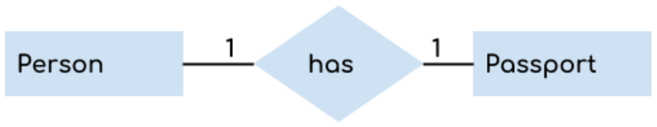


RELATIONSHIP

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

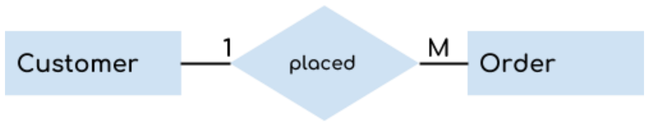
1. ONE TO ONE RELATIONSHIP

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.



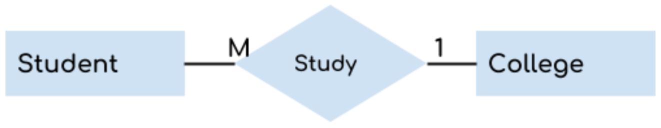
2. ONE TO MANY RELATIONSHIP

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.



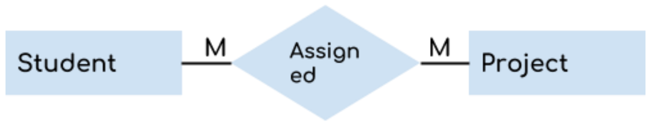
3. MANY TO ONE RELATIONSHIP

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.



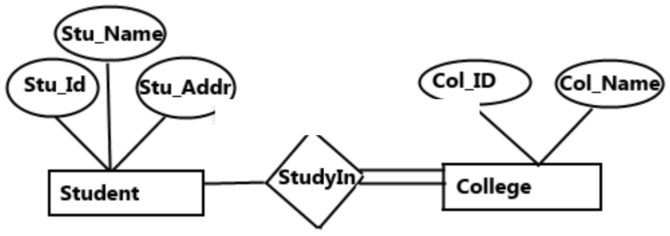
4. MANY TO MANY RELATIONSHIP

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.

EXTENDED E-R FEATURES

- **Specialization** – The process of designating to sub grouping within an entity set is called specialization. In above figure, the “person” is distinguish in to whether they are “employee” or “customer”.

  Formally in above figure specialization is depicted by a triangle component labelled (is a), means the *customer* is a *person*.

  Sometime this ISA (is a) referred as a superclass-subclass relationship. This is also used to emphasize on to creating the distinct lower level entity sets.

- **Generalization** – generalization is relationship that exist between higher level entity set and one or more lower level entity sets. Generalization synthesizes these entity sets into single entity set.

- **Higher level and lower level entity sets** – This property is created by specialization and generalization. The attributes of higher level entity sets are inherited by lower level entity sets.

  For example: In above figure “customers” and “employee” inherits the attributes of “person”.

- **Attribute inheritance**: When given entity set is involved as a lower entity set in only one “ISA” (is a) relationship, it is referred as a *single attribute inheritance*. If lower entity set is involved in more than one ISA (is a) relationship, it is referred as a *multi attribute inheritance.*

- **Aggregation**: there is a one limitation with E-R model that it cannot express relationships among relationships. So aggregation is an abstraction through which relationship is treated as *higher level entities*.