DBMS - UNIT-I

Introduction:

A database-management system (DBMS) is a collection of interrelated data and a set of programs to access those data. The collection of data, usually referred to as the database, contains information relevant to an enterprise. The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.

2.Purpose of Database Systems

The Database Management System (DBMS) is defined as a software system that allows the user to define, create and maintain the database and provide control access to the data.

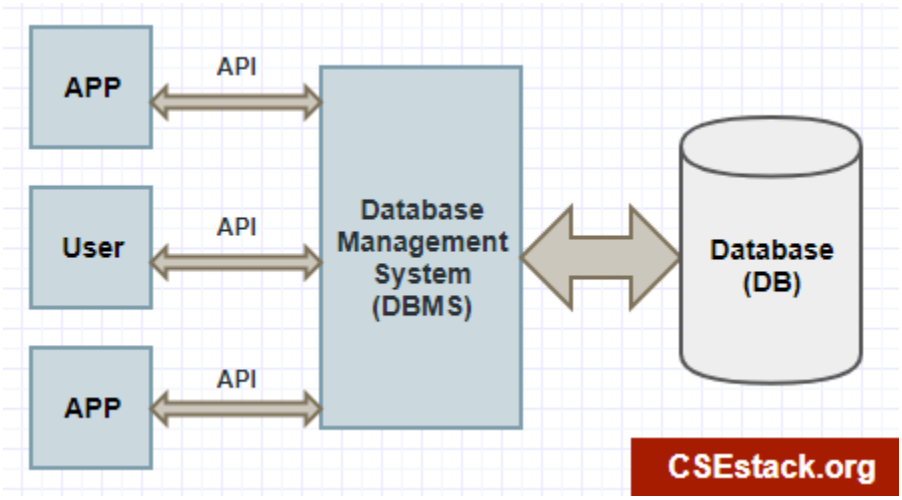
It is a collection of programs used for managing data and simultaneously it supports different types of users to create, manage, retrieve, update and store information.

Purpose

The purpose of DBMS is to transform the following −

- Data into information.
- Information into knowledge.
- Knowledge to the action.

The diagram given below explains the process as to how the transformation of data to information to knowledge to action happens respectively in the DBMS −



Uses of DBMS

The main uses of DBMS are as follows −

- Data independence and efficient access of data.
- Application Development time reduces.
- Security and data integrity.

- Uniform data administration.
- Concurrent access and recovery from crashes.

Applications of DBMS

The different applications of DBMS are as follows −

- **Railway Reservation System** − It is used to keep record of booking of tickets, departure of the train and the status of arrival and give updates to the passengers with the help of a database.
- **Library Management System** − There will be so many numbers of books in the library and it is very hard to keep a record of all the books in a register or a copy. So, DBMS is necessary to keep track of all the book records, issue dates, name of the books, author and maintain the records.
- **Banking** − We are doing a lot of transactions daily without directly going to the banks. The only reason is the usage of databases and it manages all the data of the customers over the database.
- **Educational Institutions** − All the examinations and the data related to the students maintained over the internet with the help of a database management system. It contains registration details of the student, results, grades and courses available. All these works can be done online without visiting an institution.
- **Social Media Websites** − By filling the required details we are able to access social media platforms. Many users daily sign up for social websites such as Facebook, Pinterest and Instagram. All the information related to the users are stored and maintained with the help of DBMS.

Widely used Database-System Applications:

Databases are widely used. Here are some representative applications:

• Enterprise Information

Sales, Accounting, Human resources, Manufacturing, Online retailers, etc.,

• Banking and Finance

Banking, Credit card transactions, Finance.

• Universities.

• Airlines.

• Telecommunication.

Database systems arose in response to early methods of computerized management of commercial data. The file-processing system is supported by a conventional operating system. The system stores permanent records in various files, and it needs different application programs to extract records from, and add records to, the appropriate files. Before database management systems (DBMSs) were introduced, organizations usually stored information in such systems.

Disadvantages of file-processing systems:

- **Data redundancy and inconsistency:**

Since different programmers create the files and application programs over a long period, the various files are likely to have different structures and the programs may be written in several programming languages. Moreover, the same information may be duplicated in several places (files).

For example, if a student has a double major (say,music and mathematics) the address and telephone number of that student may appear in a file that consists of student records of students in the Musicdepartment and in a file that consists of student records of students in the Mathematics department.

This redundancy leads to higher storage and access cost. In addition, it may lead to data inconsistency

- **Difficulty in accessing data:**

The point here is that conventional file-processing environments do not allow needed data to be retrieved in a convenient and efficient manner.

Example:

one of the university clerks needs to find out the names of all students who live within a particular postal-code area. The clerk asks the data-processing department to generate such a list. Because the designers of the original system did not anticipate this request,there is no application program on hand to meet it.

- **Data isolation:**

Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.

- **Integrity problems:**

The data values stored in the database must satisfy certain types of consistency constraints.

Example:

Suppose the university maintains an account for each department, and records the balance amount in each account. Suppose also that the university requires that the account balance of a department may never fall below zero. Developers enforce these constraints in the system by adding appropriate code in the various application programs.

- **Atomicity problems:**

A computer system, like any other device, is subject to failure. In many applications, it is crucial that, if a failure occurs, the data be restored to the consistent state that existed prior to the failure.

Example:

A program to transfer $500 from the account balance of department A to the account balance of department B. If a system failure occurs during the execution of the program

- **Concurrent-access anomalies**:

For the sake of overall performance of the system and faster response, many systems allow multiple users to update the data simultaneously.

Example:

suppose a registration program maintains a count of students registered for a course, in order to enforce limits on the number of students registered.

• **Security problems**:

Not every user of the database system should be able to access all the data.

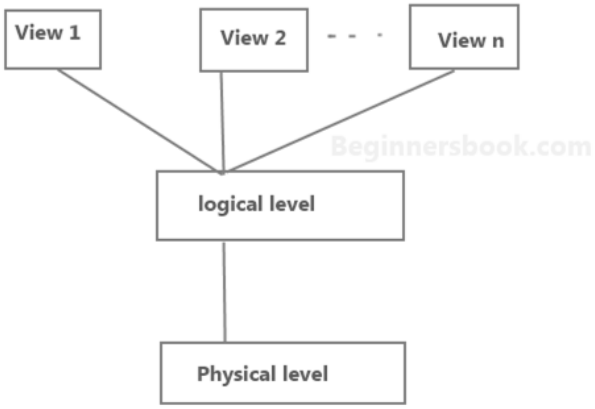
Example:

In a university, payroll personnel need to see only that part of the database that has financial information. They do not need access to information about academic records.

These difficulties, among others, prompted the development of database systems. In what follows, we shall see the concepts and algorithms that enable database systems to solve the problems with file-processing systems.

3.Views of data

A database system is a collection of interrelated data and a set of programs that allow users to access and modify these data. A major purpose of a database system is to provide users with an *abstract* view of the data. That is, the system hides certain details of how the data are stored and maintained.



Three Levels of data abstraction

Data Abstraction

For the system to be usable, it must retrieve data efficiently. The need for efficiency has led designers to use complex data structures to represent data in the database. Since many database-system users are not computer trained, developers hide the complexity from users through several levels of abstraction, to simplify users' interactions with the system:

- **Physical level**. The lowest level of abstraction describes *how* the data are ac tually stored. The physical level describes complex low-level data structures in detail.
- **Logical level**. The next-higher level of abstraction describes *what* data are stored in the database, and what relationships exist among those data. Database administrators, who must decide what information to keep in the database, use the logical level of abstraction.
- **View level**. The highest level of abstraction describes only part of the entire database. Even though the logical level uses simpler structures, complexity remains because of the variety of information stored in a large database.

For example, we may describe a record as follows:[1]

```
type instructor = record
ID : char (5);
name : char (20);
dept name : char (20);
salary : numeric (8,2);
end;
```

This code defines a new record type called *instructor* with four fields. Each field has a name and a type associated with it. A university organization may have several such record types, including

- *department*, with fields *dept name*, *building*, and *budget*
- *course*, with fields *course id*, *title*, *dept name*, and *credits*
- *student*, with fields *ID*, *name*, *dept name*, and *tot cred*

At the physical level, an *instructor*, *department*, or *student* record can be de scribed as a block of consecutive storage locations.

Finally, at the view level, computer users see a set of application programs that hide details of the data types.

4.Data Models

Underlying the structure of a database is the **data model**: a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints. A data model provides a way to describe the design of a database at the physical, logical, and view levels. There are a number of different data models that we shall cover in the text. The data models can be classified into four different categories:

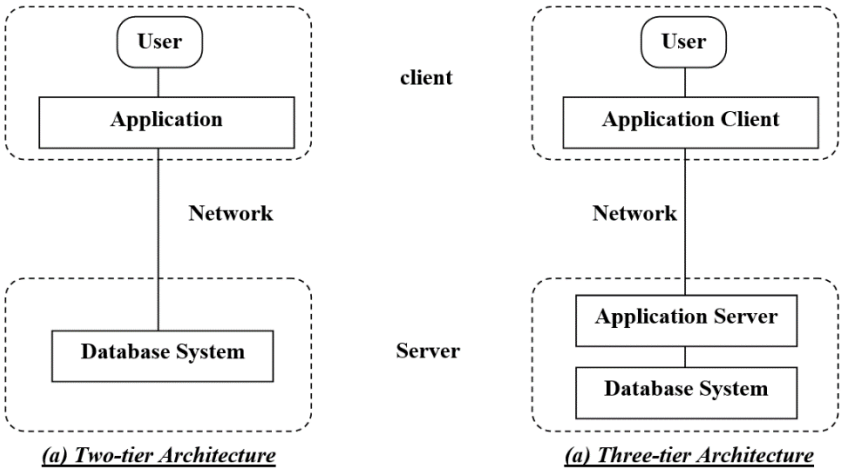
- **Relational Model**. The relational model uses a collection of tables to repre sent both data and the relationships among those data. Each table has mul tiple columns, and each column has a unique name. Tables are also known as **relations.**
- **Entity-Relationship Model**. The entity-relationship (E-R) data model uses a collection of basic objects, called *entities*, and *relationships* among these objects. An entity is a "thing" or "object" in the real world that is distinguishable from other objects.
- **Object-Based Data Model**. Object-oriented programming (especially in Java, C++, or C#) has become the dominant software-development methodology. This led to the development of an object-oriented data model.
- **Semistructured Data Model**. The semistructured data model permits the specification of data where individual data items of the same type may have different sets of attributes. The **Extensible Markup Language (XML)** is widely used to represent semistructured data.

Historically, the **network data model** and the **hierarchical data model** pre ceded the relational data model. These models were tied closely to the underlying implementation, and complicated the task of modeling data. As a result they are used little now, except in old database code that is still in service in some places. They are outlined online in Appendices D and E for interested readers.

5.Database Architecture

The architecture of a database system is greatly influenced by the underlying 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.Most users of a database system today are not present at the site of the database system, but connect to it through a network. We can therefore differentiate between client machines, on which remote database users work, and server machines, on which the database system runs.



(a) Two-tier Architecture *(a) Three-tier Architecture*

Database applications are usually partitioned into two or three parts. In a **two-tier architecture,** the application resides at the client machine,where it invokes database system functionality at the server machine query language statements. Application program interface standards like ODBC and JDBC are used for interaction between the client and the server.

In contrast, in a **three-tier architecture,** the client machine acts as merely a front end and does not contain any direct database calls. Instead,

the client end communicates with an application server, usually through a forms interface.

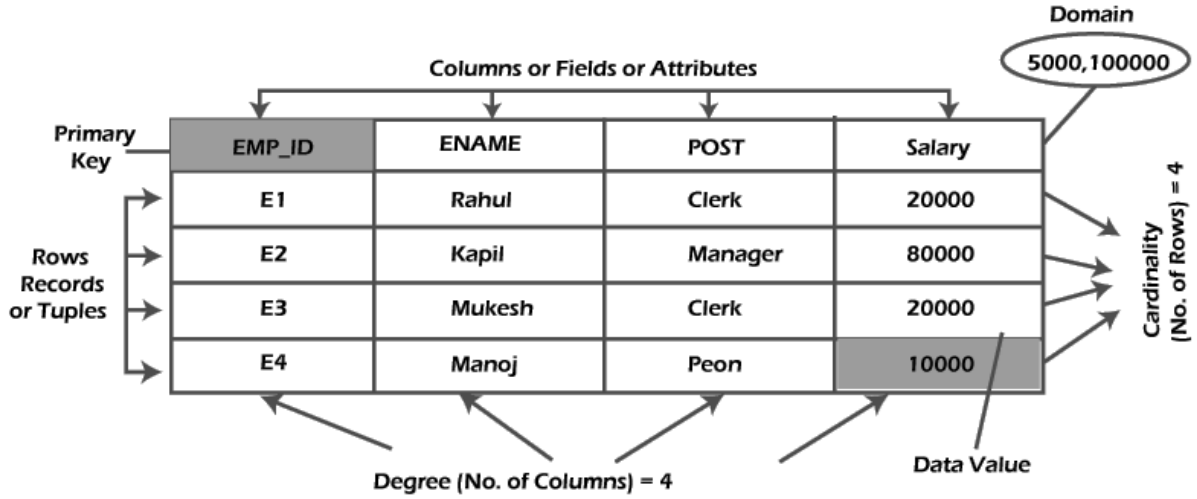
The application server in turn communicates with a database system to access data. The business logic of the application, which says what actions to carry out under what conditions, is embedded in the application server, instead of being distributed across multiple clients. Three-tier applications are more appropriate for large applications, and for applications that run on the World Wide Web.

RELATIONAL DATABASES

A data model is a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints. In this part, we focus on the relational model. The relational model is today the primary data model for commercial data processing applications.

Relational Model

A relational database consists of a collection of tables, each of which is assigned a unique name. For example, consider the instructortable of , which stores information about instructors. The table has four column headers: ID, name, dept name, and salary.



What is table/Relation?

Everything in a relational database is stored in the form of relations. The RDBMS database uses tables to store data. A table is a collection of related data entries and contains rows and columns to store data. Each table represents some real-world objects such as person, place, or event about

which information is collected. The organized collection of data into a relational table is known as the logical view of the database.

ID	Name	AGE	COURSE
1	Ajeet	24	B.Tech
2	Aryan	20	C.A
3	Mahesh	21	BCA
4	Ratan	22	MCA
5	Vimal	26	BSC

Properties of a Relation:

- Each relation has a unique name by which it is identified in the database.
- Relation does not contain duplicate tuples.
- The tuples of a relation have no specific order.
- All attributes in a relation are atomic, i.e., each cell of a relation contains exactly one value.

A table is the simplest example of data stored in RDBMS.

Let's see the example of the student table.

What is a row or record?

A row of a table is also called a record or tuple. It contains the specific information of each entry in the table. It is a horizontal entity in the table. For example, The above table contains 5 records.

Properties of a row:

- No two tuples are identical to each other in all their entries.
- All tuples of the relation have the same format and the same number of entries.
- The order of the tuple is irrelevant. They are identified by their content, not by their position.

Let's see one record/row in the table.

ID	Name	AGE	COURSE
1	Ajeet	24	B.Tech

What is a column/attribute?

A column is a vertical entity in the table which contains all information associated with a specific field in a table. For example, "name" is a column in the above table which contains all information about a student's name.

Properties of an Attribute:

- Every attribute of a relation must have a name.
- Null values are permitted for the attributes.
- Default values can be specified for an attribute automatically inserted if no other value is specified for an attribute.
- Attributes that uniquely identify each tuple of a relation are the primary key.

Name
Ajeet
Aryan
Mahesh
Ratan
Vimal

What is data item/Cells?

The smallest unit of data in the table is the individual data item. It is stored at the intersection of tuples and attributes.

Properties of data items:

- Data items are atomic.
- The data items for an attribute should be drawn from the same domain.

In the below example, the data item in the student table consists of Ajeet, 24 and Btech, etc.

ID	Name	AGE	COURSE
1	Ajeet	24	B.Tech

Degree:

The total number of attributes that comprise a relation is known as the degree of the table.

For example, the student table has 4 attributes, and its degree is 4.

ID	Name	AGE	COURSE
1	Ajeet	24	B.Tech
2	Aryan	20	C.A
3	Mahesh	21	BCA
4	Ratan	22	MCA
5	Vimal	26	BSC

Cardinality:

The total number of tuples at any one time in a relation is known as the table's cardinality. The relation whose cardinality is 0 is called an empty table.

For example, the student table has 5 rows, and its cardinality is 5.

ID	Name	AGE	COURSE
1	Ajeet	24	B.Tech
2	Aryan	20	C.A

3	Mahesh	21	BCA
4	Ratan	22	MCA
5	Vimal	26	BSC

Domain:

The domain refers to the possible values each attribute can contain. It can be specified using standard data types such as integers, floating numbers, etc. **For example**, An attribute entitled Marital_Status may be limited to married or unmarried values.

NULL Values

The NULL value of the table specifies that the field has been left blank during record creation. It is different from the value filled with zero or a field that contains space.

Data Integrity

There are the following categories of data integrity exist with each RDBMS:

Entity integrity: It specifies that there should be no duplicate rows in a table.

Domain integrity: It enforces valid entries for a given column by restricting the type, the format, or the range of values.

Referential integrity specifies that rows cannot be deleted, which are used by other records.

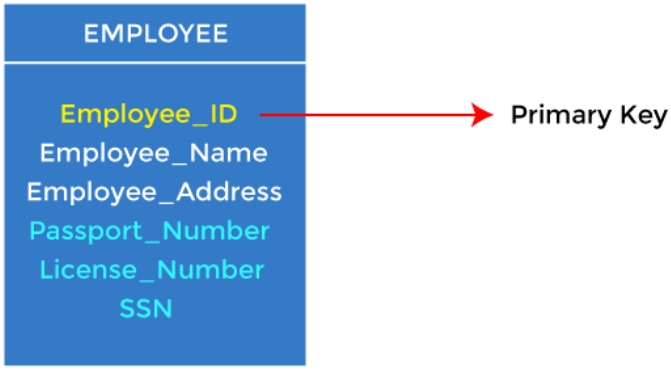
User-defined integrity: It enforces some specific business rules defined by users. These rules are different from the entity, domain, or referential integrity.

Keys

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

1. Primary key

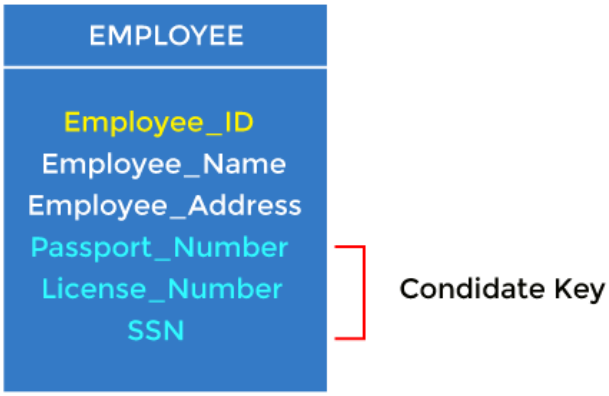
- It is the first key used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys, as we saw in the PERSON table. The key which is most suitable from those lists becomes a primary key.
- In the EMPLOYEE table, ID can be the primary key since it is unique for each employee. In the EMPLOYEE table, we can even select License_Number and Passport_Number as primary keys since they are also unique.
- For each entity, the primary key selection is based on requirements and developers.



2. Candidate key

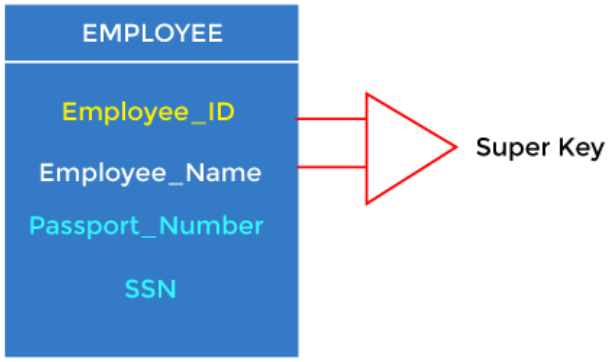
- A candidate key is an attribute or set of attributes that can uniquely identify a tuple.
- Except for the primary key, the remaining attributes are considered a candidate key. The candidate keys are as strong as the primary key.

For example: In the EMPLOYEE table, id is best suited for the primary key. The rest of the attributes, like SSN, Passport_Number, License_Number, etc., are considered a candidate key.



3. Super Key

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

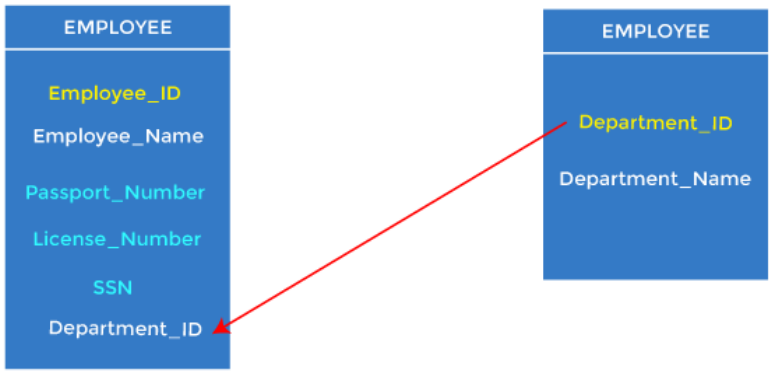


For example: In the above EMPLOYEE table, for(EMPLOEE_ID, EMPLOYEE_NAME), the name of two employees can be the same, but their EMPLYEE_ID can't be the same. Hence, this combination can also be a key.

The super key would be EMPLOYEE-ID (EMPLOYEE_ID, EMPLOYEE-NAME), etc.

4. Foreign key

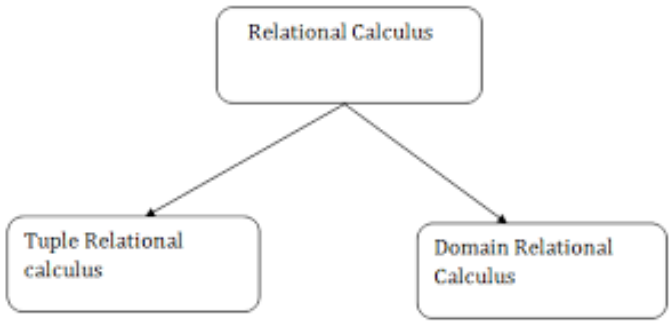
- Foreign keys are the column of the table used to point to the primary key of another table.
- Every employee works in a specific department in a company, and employee and department are two different entities. So we can't store the department's information in the employee table. That's why we link these two tables through the primary key of one table.
- We add the primary key of the DEPARTMENT table, Department_Id, as a new attribute in the EMPLOYEE table.
- In the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.



The Domain Relational Calculus

A second form of relational calculus, called domain relational calculus, uses domain variables that take on values from an attributes domain, rather than values for an entire tuple.

Relational calculus is a non-procedural query language.



1. Tuple Relational Calculus (TRC)

It is a non-procedural query language which is based on finding a number of tuple variables also known as range variable for which predicate holds true. It describes the desired information without giving a specific procedure for obtaining that information. The tuple relational calculus is specified to select the tuples in a relation. In TRC, filtering variable uses the tuples of a relation. The result of the relation can have one or more tuples.

Notation:

A Query in the tuple relational calculus is expressed as following notation

1. {T | P (T)} or {T | Condition (T)}

Where

T is the resulting tuples

P(T) is the condition used to fetch T.

For example:

1. { T.name | Author(T) AND T.article = 'database' }

Output: This query selects the tuples from the AUTHOR relation. It returns a tuple with 'name' from Author who has written an article on 'database'.

TRC (tuple relation calculus) can be quantified. In TRC, we can use Existential (∃) and Universal Quantifiers (∀).

For example:

1. { R| ∃T ∈ Authors(T.article='database' AND R.name=T.name)}

Output: This query will yield the same result as the previous one.

2. Domain Relational Calculus (DRC)

The second form of relation is known as Domain relational calculus. In domain relational calculus, filtering variable uses the domain of attributes. Domain relational calculus uses the same operators as tuple calculus. It uses logical connectives ∧ (and), ∨ (or) and ¬ (not). It uses Existential (∃) and Universal Quantifiers (∀) to bind the variable. The QBE or Query by example is a query language related to domain relational calculus.

Notation:

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

Where

a1, **a2** are attributes
P stands for formula built by inner attributes

For example:

1. {< article, page, subject > | ∈ javatpoint ∧ subject = 'database'}

Output: This query will yield the article, page, and subject from the relational javatpoint, where the subject is a database.

SQL

- SQL stands for Structured Query Language. It is used for storing and managing data in relational database management system (RDMS).
- It is a standard language for Relational Database System. It enables a user to create, read, update and delete relational databases and tables.
- All the RDBMS like MySQL, Informix, Oracle, MS Access and SQL Server use SQL as their standard database language.
- SQL allows users to query the database in a number of ways, using English-like statements.

Rules:

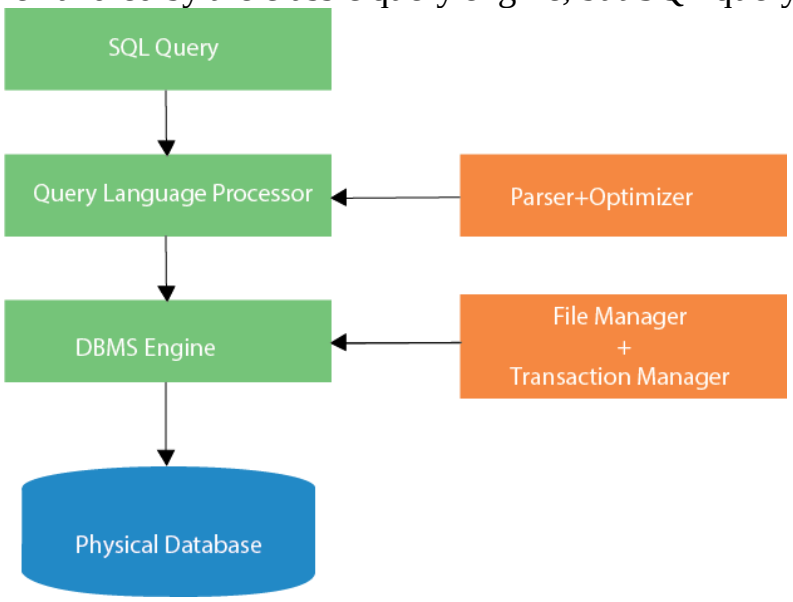
SQL follows the following rules:

- Structure query language is not case sensitive. Generally, keywords of SQL are written in uppercase.

- Statements of SQL are dependent on text lines. We can use a single SQL statement on one or multiple text line.
- Using the SQL statements, you can perform most of the actions in a database.
- SQL depends on tuple relational calculus and relational algebra.

SQL process:

- When an SQL command is executing for any RDBMS, then the system figure out the best way to carry out the request and the SQL engine determines that how to interpret the task.
- In the process, various components are included. These components can be optimization Engine, Query engine, Query dispatcher, classic, etc.
- All the non-SQL queries are handled by the classic query engine, but SQL query engine won't handle logical files.



Characteristics of SQL

- SQL is easy to learn.
- SQL is used to access data from relational database management systems.
- SQL can execute queries against the database.
- SQL is used to describe the data.
- SQL is used to define the data in the database and manipulate it when needed.
- SQL is used to create and drop the database and table.
- SQL is used to create a view, stored procedure, function in a database.
- SQL allows users to set permissions on tables, procedures, and views.

Advantages of SQL

- There are the following advantages of SQL:
- High speed
- Using the SQL queries, the user can quickly and efficiently retrieve a large amount of records from a database.
- No coding needed
- Well defined standards
- Long established are used by the SQL databases that are being used by ISO and ANSI.
- Portability
- SQL can be used in laptop, PCs, server and even some mobile phones.
- Interactive language
- Multiple data view
- Using the SQL language, the users can make different views of the database structure.