UNIT-1

DATA:

Data is a collection of facts, such as numbers, words, measurements, observations or just descriptions of things.

Database:

A database is a collection of related data which represents some aspect of the real world.

DBMS:

Database Management System (DBMS) is software for storing and retrieving users' data while considering appropriate security measures. It consists of a group of programs which manipulate the database. The DBMS accepts the request for data from an application and instructs the operating system to provide the specific data. In large systems, a DBMS helps users and other third-party software to store and retrieve data. It provides an interface between the data and the software application.

Goals of DBMS

Following are the goals of DBMS,

(i) To support system activities.

(ii) To perform monitoring and turning so as to ensure satisfactory performance level.

(iii) To perform backup and recovery activities in order to recover from disaster.

(iv) To carry out security auditing and monitoring that deals with the assignment of access rights and use of access privileges

Disadvantages of DBMS

The disadvantages of DBMS are as follows,

1. High conversion cost
2. High management and installation cost
3. Need of new specialized personnel
4. Need for explicit backup and recovery
5. Security breaches.

Applications of DBMS

Database management systems are used by many individuals either directly or indirectly. Some of the applications ofDBMS are listed below.

- **Railway Reservation System** − The railway reservation system database plays a very important role by keeping record of ticket booking, train's departure time and arrival status and also gives information regarding train late to people through the database.
- **Library Management System** − Now-a-days it's become easy in the Library to track each book and maintain it because of the database. This happens because there are thousands of books in the library. It is very difficult to keep a record of all books in a copy or register. Now DBMS used to maintain all the information related to book issue dates, name of the book, author and availability of the book.
- **Banking** − Banking is one of the main applications of databases. We all know there will be a thousand transactions through banks daily and we are doing this without going to the bank. This is all possible just because of DBMS that manages all the bank transactions.
- **Universities and colleges** − Now-a-days examinations are done online. So, the universities and colleges are maintaining DBMS to store Student's registrations details, results, courses and grade all the information in the database. For example, telecommunications. Without DBMS there is no telecommunication company. DBMS is most useful to these companies to store the call details and monthly postpaid bills.
- **Credit card transactions** − The purchase of items and transactions of credit cards are made possible only by DBMS. A credit card holder has to know the importance of their information that all are secured through DBMS.
- **Finance** − Now-a-days there are lots of things to do with finance like storing sales, holding information and finance statement management etc. these all can be done with database systems.
- **Military** − In military areas the DBMS is playing a vital role. Military keeps records of soldiers and it has so many files that should be kept secure and safe. DBMS provides a high security to military information.
- **Online Shopping** − Now-a-days we all do Online shopping without wasting the time by going shopping with the help of DBMS. The products are added and sold only with the help of DBMS like Purchase information, invoice bills and payment.
- **Human Resource Management** − The management keeps records of each employee's salary, tax and work through DBMS.
- **Manufacturing** − Manufacturing companies make products and sell them on a daily basis. To keep records of all those details DBMS is used.
- **Airline Reservation system** − Just like the railway reservation system, airlines also need DBMS to keep records of flights arrival, departure and delay status.

Purpose of DBMS:

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.

The purpose of DBMS is to transform the following −

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

Drawbacks in File System

There are so many drawbacks in using the file system. These are mentioned below −

- Data redundancy and inconsistency: Different file formats, duplication of information in different files.
- Difficulty in accessing data: To carry out new task we need to write a new program.
- Data Isolation − Different files and formats.
- Integrity problems.
- Atomicity of updates − Failures leave the database in an inconsistent state. For example, the fund transfer from one account to another may be incomplete.
- Concurrent access by multiple users.
- Security problems.

Database system offer so many solutions to all these problems

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.

View of data:

Database is a collection of large volumes of data. A user will not always require complete data, so the responsibility of the database system is to provide only the data, that is required to the user. A user, who wishes to see the record of a student withid '102', may not be provided with all student details. Similarly, a programmer who wishes to enhance the features of database system is not concerned about the data, but need to know how is data stored. Hence, different users need different view of data.

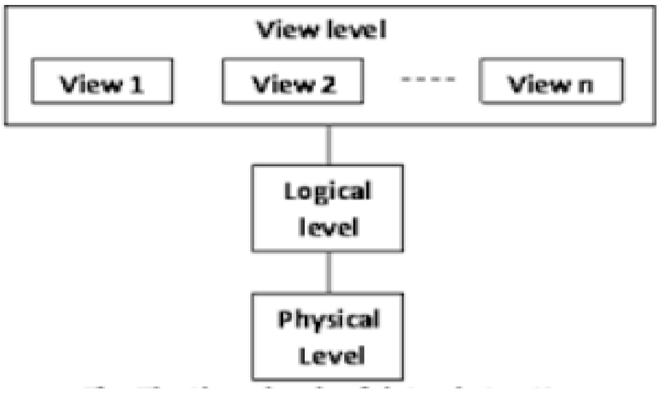
Data Abstraction

In DBMS, the data can be abstracted in three levels (or) in other words there are three levels of abstraction in DBMS.

The goal of the three level abstraction in a DBMS is to separate the user's request and the physical storage of data in database. The data abstracted at each of these levels is described by a word scheme. A scheme means a plan that explains the records and relationships existing between them, at each level. Basically the word scheme, means a systematic plan for achieving some objective, is used in the database literature with the word "schema".

- **Physical level:** The lowest level of abstraction describes how the data are actually 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. The logical level thus describes the entire database in terms of a small number of relatively simple structures. Although implementation of the simple structures at the logical level may involve complex physical-level structures, the user of the logical level does not need to be aware of this complexity. This is referred to as physical data independence. 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. Many users of the database system do not need all this information; instead, they need to access only a part of the database. The view level of abstraction exists to simplify their interaction with the system. The system may provide many views for the same database.

Figure 1.1 shows the relationship among the three levels of abstraction. An analogy to the concept of data types in programming languages may clarify the distinction among levels of abstraction. Many high-level programming.



Instances and Schemas:

- The collection of information stored in the database at a particular moment is called an **instance** of the database.
- The overall design of the database is called the database schema. A database schema corresponds to the variable declarations (along with associated type definitions) in a program.

- Each variable has a particular value at a given instant. The values of the variables in a program at a point in time correspond to an instance of a database schema.
- Database systems have several schemas, partitioned according to the levels of abstraction.
- The physical schema describes the database design at the physical level, while the logical schema describes the database design at the logical level.
- A database may also have several schemas at the view level, sometimes called sub schemas that describe different views of the database.

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. The data models can be classified into four different categories:

• **Relational Model**: The relational model uses a collection of tables to represent both data and the relationships among those data. Each table has multiple columns, and each column has a unique name. Tables are also known as relations. The relational model is an example of a record-based model.

The relational data model is the most widely used data model, and a vast majority of current database systems are based on the relational model.

• **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 that can be seen as extending the E-R model with notions of encapsulation, methods (functions), and object identity. The object-relational data model combines features of the object-oriented data model and relational data model.

• Semi structured Data Model. The semi structured 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 semi structured data.

Database Languages

The database languages are called sub-languages. This is because they do not have all the facilities of full-fledged programming languages. The sub-language has twoparts,

1. Data Definition Language (DDL)

2. Data Manipulation Language (DML).

1. Data Definition Language (DDL)

Data definition language defines the schema for the database by specifying entities and the relationship among them. It also says some security constraints. The execution of DDL statements results in new tables. These tables are stored in “system catalog” also called data dictionary.

Data dictionary has the information of all the definitions of database schema, low-level data structures, files and records. It also maintains the relationship information among various records.

Data dictionary provides faster access to database by maintaining certain indexes. Data dictionary also contains Meta data that has data about the data stored in database. Data definition language allows the user to specify the storage and access methods by using a special language called data storage and definition language.

2. Data Manipulation Language (DML)

Data manipulation language performs the following operations,

(i) Insertion of new data

(ii) Deletion of stored data

(iii) Retrieval of stored data

(iv) Modifying the data.

Data manipulations are applied at internal, conceptual and external databases. However, the level of complexity involved varies. Complex low-level procedures allow efficient access, but external procedures focus on ease of usage, there by incurring low complexity.

Data manipulation language that involves in retrieval of data is called “query language”. DML are classified based on the retrieval constructs.

(a) Procedural DMLs

Using procedural DML the user can specify what data needs to be retrieved. The user also has to guide as how to retrieve the data.

This means that the user has to express the operations that would retrieve the desired data. DML initiates by evaluating these operations. The result of evaluation is again operated until the desired data is retrieved. The DML statements are embedded into high-level languages to perform selective and iterative operations.

(b) Non-procedural DMLs

Non-procedural languages are also called declarative languages. These languages are easy to learn as the user simply needs to express the desired data. It is up to the database system to choose an optimal procedure to retrieve the data. DBMS translates the DML statements into set of procedures and retrieves the data.

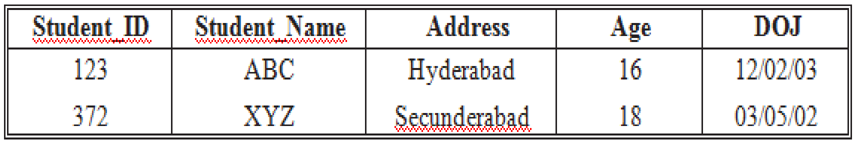
SQL (Structured Query Language) and QBE (Query By Example) are some of examples of non-procedural DML.

Relational Databases

- In Relational database Data is stored in the form of tables. Each table contains information about a particular object which is represented in rows and columns. The column specifies the different attributes of an object and row (also called tuple) specifies actual instance of the object.
- The relation in relational database refers to different tables in the database. It is necessary that, every row in a table should consist of unique value i.e., individual tuple contain atomic value.
- Relational databases are based on Entity Relationship (ER) diagrams that represent the database as a set of entities and their relationship.
- Let us consider a relational database of all students that contain the following relation tables. Student, Course, College.

Student table consists of following fields, Student_ID, Student_Name, Address, Age, DOJ
Course table consists of following fields, Course_ID, Course_Name, Course_duration
College table consists of following fields, College_ID, College_Name, and Address.

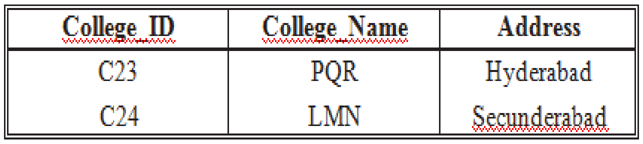
Student



Student_ID	Student_Name	Address	Age	DOJ
123	ABC	Hyderabad	16	12/02/03
372	XYZ	Secunderabad	18	03/05/02

Student Table

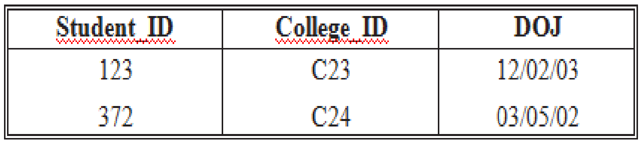
College



College_ID	College_Name	Address
C23	PQR	Hyderabad
C24	LMN	Secunderabad

College Table

Join



Student_ID	College_ID	DOJ
123	C23	12/02/03
372	C24	03/05/02

Join Table

Application Programs

Application programs are the software codes that enable a user to interact with the database.

Database Access for Application Programs

The application programs access the database in two ways,

(i) By using application programming interface

(ii) By extending the host language syntax.

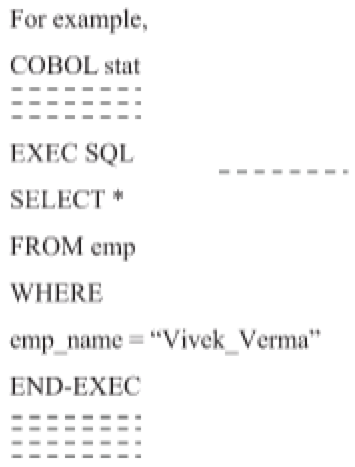
Several application development environments provide a platform for developing applications and include facilities that enable GUI designing and other features that enhance the quality of the software. Some of the application development environments are Sybase, JBuilder.

(i) Application Programming Interface (API)

Application Programming Interface is a set of procedures that sends the DML and DDL to database statements and retrieves the results after processing from the database. The commonly used API include ODBC (Open Database Connectivity), and JDBC (Java Database Connectivity). These APIs provides an interface for application programs written in C and Java respectively.

(ii) Extending Host Language Syntax

Database can also be accessed by extending host language syntax so as to embed DML statements within the host programming language. SQL (Structured Query Language), which is a database sub-language, is embedded in a programming language.



```
For example,
COBOL stat
- - - - - - - -
- - - - - - - -
- - - - - - - -
EXEC SQL          - - - - - - - -
SELECT *
FROM emp
WHERE
emp_name = "Vivek_Verma"
END-EXEC
- - - - - - - -
- - - - - - - -
- - - - - - - -
```

The execution of these SQL embedded statements is different from the normal execution. A special compiler called SQL pre compiler is used to compile the source code.

Database design:

Database design mainly involves the design of the database schema. The design of a complete database application environment that meets the needs of the enterprise being modeled requires attention to a broader set of issues.

For designing a database, there are four phases, namely:

- **Initial Phase:** The Initial Phase of database design is to fully describe the data needs of the potential database users.
- **Conceptual Design phase :** The designer chooses a data model, by applying the concepts of the data model that is chosen and translates these requirements into a conceptual schema of the database. A fully developed conceptual schema indicates the functional needs of the user. Here, the users describe various operations that have to be performed on the data.

  The operations can include updating, searching, retrieving and deleting data.
- **Logical Design Phase :** The database designer plots the high-level conceptual schema onto the implementation of the data model of the database system that has to be used.

  The implementation data model is also called the relational data model.
- **Physical Design phase :** Here, the designer uses the resulting system-specific database schema. In this phase, the physical features of the database are specified.

The following two issues are to be avoided In designing a database schema,

- Redundancy
- Incompleteness

Data storage and querying:

Generally the user interacts with the database management system through an interface. The DBMS does theprocessing and retrieves the data from the database.

Database system is divided into two modules,

1. Storage management
2. Query processing.

Data stored in database may costs more than trillion bytes of data. Main memory cannot accommodate for such large amount of data, hence the data is stored in disk. But for processing data needs to be transferred from disk to the main memory. But this transfer consumes processor time; hence thedata needs to be arranged such that the data transfer rate is nottoo high. This is taken care by storage manager.

The main task of database system is to provide the userwith simplified view of data. This is achieved by hiding the physical level implementation details from the user, the user is provided with only high-level view. But, for faster processing the operations need to be done at physical level. The queries at logical level into optimal sequence of operations at physical level.

1. Storage Management

Storage management is handled by storage manager that is basically a program module. This module acts as an interface between low-level data stored in database and the application programs.

Data is stored in disk using file system provided by operating system. The storage manager provides interaction with file manager and converts the complex DML statements into low-level file system commands. In addition to this, storage manager is responsible for storing, retrieving and modifying data within the database. Storage manager consists of the following key components,

(i) Transaction manger

(ii) File manager

(iii) Buffer manager.

(iv) Integrity manager

(v) Authorization manager.

(i) Transaction Manager: It manages the transactions so as to ensure that data remain in consistent state even after the system failures. It also enables the execution of concurrent transactions without any conflicts.

(ii) File Manager: It manages the process of allocating disk and data structures that are used for representing the information saved on disk.

(iii) Buffer Manager: It handles the transfer of data from disk onto the main memory and decides what data must be kept in main memory.

(iv) Integrity Manager: It verifies whether the integrity constraints defined on the data are satisfied.

(v) Authorization Manager: It checks the authority of users and allows only authorized users to access the data.

Application Programs

The following are the different data structures used by storage manager.

(a) Data Files

These are the files that contain the database.

(b) Data Dictionary

It maintains metadata regarding the different data structures used in database.

(c) Indices

It provides fast access to the required data items.

2. **<u>Query Processor:</u>**

(a) Interpreter of Data Definition Language Statements

DDL statements written by DBA to define the schema are interpreted and stored in the data dictionary.

(b) Compiler of Data Manipulation Language Statements

As any other compiler, DML compiler converts the DML statements in low-level instructions. It also optimizes thequery i.e., perform "query optimization"

(c) Query Evaluation

DML compiler converts DML statements into low level instructions which are evaluated by query evaluation machine.

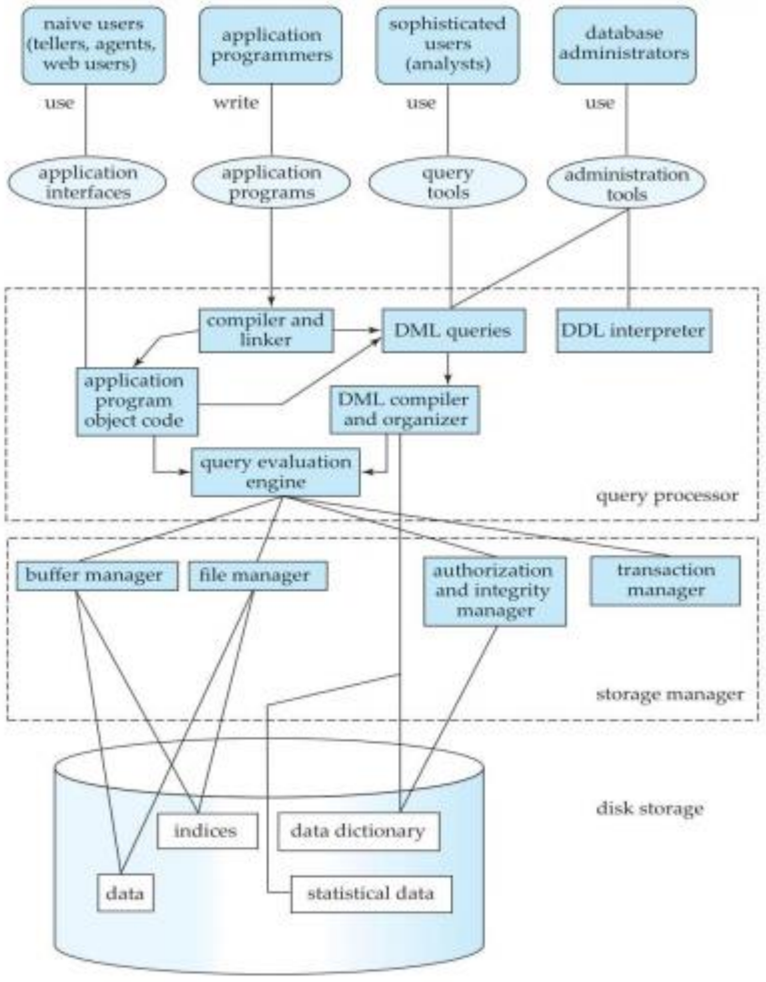


Figure 1.5 System structure.

Transaction Management:

Transaction refers to a series of actions such as read, write, abort and commit, which are performed by DBMS. Transaction management refers to managing the transactions such that the data remain in consistent state even after the execution of the transactions. It also provides optimal concurrentexecution of the transactions.

Log: During the execution of transactions, if some changes aremade to the database then a failure may occur, which is referredto as 'system crash'. In order to avoid this failure, the changesmade to the database must be stored in the form of records byusing a database structure called Log. These records contain the information about all the update activities. Now, to ensure that the stored information is available at the time of recovery froma system crash, a principle called write-ahead logging need tobe ensured.

According to this principle the information regarding any sort of changes made to the database object must initiallybe recorded in the log record. These log records are maintainedas a sequential file and the write operations performed on these records are sequential writes. But, prior to storing the information about the changes into the disk, it is necessary to force each and every log record including the record whose LSNis equal to the page LSN must be stored in the stable storage.

When the database is recovered from a system crash, huge amount of time is consumed. Thus, to reduce the timeon operation called 'check point' is implemented. Check point refers to a synchronization point that indicates the amount of search that is to be made in the log so as to bring the databaseback to the consistent state.

Database architecture:

Database architecture can be seen as a single tier or multi-tier. But logically, database architecture is of two types like: **2-tier architecture** and **3-tier architecture**.

1-Tier Architecture

- In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and uses it.
- Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.
- The 1-Tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response.

2-Tier Architecture

- The 2-Tier architecture is same as basic client-server. In the two-tier architecture, applications on the client end can directly communicate with the database at the server side. For this interaction, API's like: **ODBC**, **JDBC** are used.
- The user interfaces and application programs are run on the client-side.

- The server side is responsible to provide the functionalities like: query processing and transaction management.
- To communicate with the DBMS, client-side application establishes a connection with the server side.

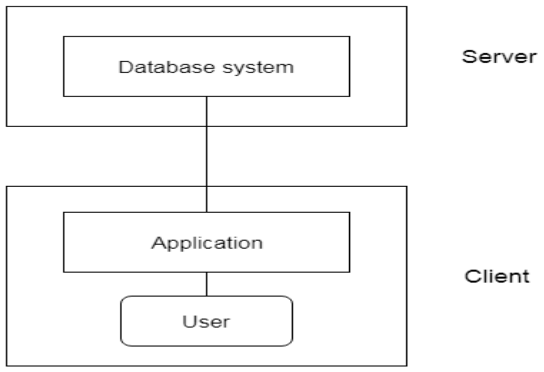


Fig: 2-tier Architecture

3-Tier Architecture

- The 3-Tier architecture contains another layer between the client and server. In this architecture, client can't directly communicate with the server.
- The application on the client-end interacts with an application server which further communicates with the database system.
- End user has no idea about the existence of the database beyond the application server. The database also has no idea about any other user beyond the application.
- The 3-Tier architecture is used in case of large web application.

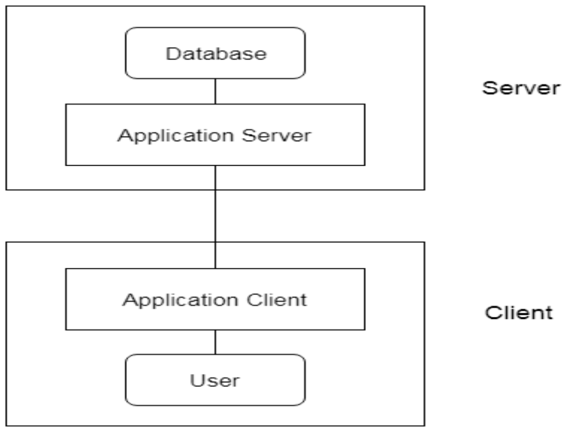


Fig: 3-tier Architecture

Data Mining

Data mining is a process of extracting knowledge fromhuge amount of data. It refers to a way of extracting useful information from an organization's database. The knowledge extracted can include pattern types, association rules and different trends. Data mining is not confined to a particular organization; instead it has techniques to explore the knowledgehidden in any data. The different techniques used for extracting data are artificial intelligence, statistical and mathematical techniques and pattern recognition techniques.

Organizations that make use of data mining techniquesare benefited in their corresponding business area by identifying the significant trends and anomalies that were not possible to be detected by a human analyst. The association shows impor tant knowledge about the database and entities present in the database.

The purpose of data mining is to discover relation thatconnects different database entities.

Reasons for Data Mining

The following are the reasons for using data mining.

1. Knowledge discovery
2. Data visualization
3. Data correction.

1. Knowledge Discovery

The objective of knowledge discovery process is to identify the invisible correlation, patterns and trends available in the database.

2. Data Visualization

The objective of data visualization is to "harmonize" large volume of data so as to find an effective way of displayingdata.

3. Data Correction

This process is used to identify and correct incomplete,erroneous and inconsistent data.

Specialty databases

A specialty database can be defined as an electronic repository which is located in a computer's RAM for storing specialty related data and metadata. This type of databases are used to perform operations data search, data retrieval, data manipulation and data calculation. In order to overcome the restrictions possessed by relational data model, database developers have developed multiple data models. Object based data model and semi-structured data model are the two data models based on specialty database.

Object based data model

Object- oriented data model that can be seen as extending the E-R model with notions of encapsulation, methods (functions), and object identity. Inheritance, object identity, and encapsulation (Information hiding), with methods to provide an interface to objects, are among the key concepts of object-oriented programming that have found applications in data modeling. The object-oriented data model also supports a rich type system, including structured and collection types. In the 1980s, several database systems based on the object-oriented data model were developed.

<u>Semi-structured Data Model:</u>

Semi-structured data models allow data specification only where each data item of similar type contain different set of attributes. An Extensible Markup Language (XML) is widely used to represent semi-structured data.

<u>Database Users and Administrators:</u>

There are four types of database users and they use different user interfaces. They are sophisticated, specialized users, application programmers and naive users.

1. **Sophisticated Users**

Sophisticated users are the users who interact with database system without writing programs. They use certain queries to retrieve the data. These queries are submitted to a query processor who evaluates the query and process it.

2. **Specialized Users**

Specialized users are sophisticated users who use specialized database applications written by them. Example of such applications include knowledge-based systems, expert systems, environment-modeling systems, computer-aided design systems and systems that store complex data such as graphics and audio.

3. **Application Programmers**

Application programmers are computer professionals and they write application programs. They use several tools to develop user interfaces, one such tool is Rapid Application Development (RAD). This tool requires little programming effort to create forms and reports.

4. **Naive Users**

Naive users are unsophisticated users. They do not know anything about database system. They simply navigate through the entire provided user interface to get the data. They invoke already written and tested applications programs through GUIs without being aware of how they are written and executed. These transactions are called 'canned transactions'.

For example, if we want to know the list of the courses offered by a university of Australia the student goes through the link provided. He does not need to know how his request is processed. The student here is a naive user. Forms are the basic users interface for naive users. To retrieve any data they just fill in the form and submit it. In addition, the reports generated from the database are simply read by naive users.

<u>Database Administrator:</u>

As the database is being used among multiple users, there should have some control over the data and programs that provide access to data. Database Administrator (DBA) is the one who provides control over the data and program designed to make data accessible.

Database Administrator is responsible for performing the following functions,

1. **Database Schema Definition**

   DBA creates database schema (conceptual, internal) by invoking data definition statements in the DDL.

- **Conceptual Schema**

  DBA identifies the entities important for the organization and the data to be stored in the entities; this is called conceptual database design. The DBA executes conceptual DDL statements to design conceptual schema.

- **Internal Schema**

  DBA after identifying the data to be stored, decides how to store the data. The physical storage is defined by using internal DDL statements. DBA then provides conceptual/internal mapping.

2. **Database Schema, Physical Structure Modification**

   DBA perform changes on the schema in order to accommodate all the changing requirements. DBA also modifies the physical structure so as to enhance the performance.

3. **Users Authorization**

   It is the responsibility of DBA to control the flow of data, so as to prevent it from being accessed by unauthorized users. DBA assigns level of authorities to every user, and this information is stored in a system structure. Each time a user makes a request for accessing the data, the authority information is checked so as to verify whether the requested user is authorized to view the requested part of the database or not.

4. **Users Interaction**

   DBA communicates with users so as to ensure that the data is readily available to the user and defines desired external schemas using external DDL. He/She then provides external/ conceptual mapping.

5. **Database Maintenance**

   The different activities performed while carrying out maintenance function include,

6. **Creating Regular Backup**

   The data may be lost or corrupted due to the human error or hardware failure. This cannot be avoided and hence the data must be loaded in some other place such that it can be reloaded after recovering from failure. This is known as backup storage.

7. **Monitoring Performance**

   The DBA makes some changes like tuning and reorganizing so as to enhance the performance.

8. **Ensuring Availability of Free Space**

DBA check, if available free space is sufficient forprocessing and upgrades the disk if the space is not sufficient

<u>Structure of Relational Database:</u>

A relational database is a collection of tables. It comprises,

1. A collection of tables each of which has a unique name.
2. Each table consists of multiple rows.
3. Each row is a set of values that by definition are relatedto each other is some way, these values. Confirm to theattributes or columns of the tables.

The basic data structure of the relational model is the table where information about a particular entity is representedin rows and columns.

A "Relation" in a relational database refers to a table inthe database. It is a set of types.

The concept of database table is closely related to the pure mathematical concept of relation, from which term relational data model originates.

Each attribute of a table has a set of permitted value forthat attribute; this set of permitted values is called the domainof that attribute

Example

Consider a bank account table with attributes account number, branch name and balance.

Account_number	Branch_name	balance
A-101	DOWNTOWN	500
A-102	PERRYRIDGE	400
A-201	BRIGHTON	900
A-215	MIANUS	700
A-217	BRIGHTON	750
A-222	REDWOOD	700
A-305	ROUND HILL	350

Let us assume $D1$ represents a set of all account numbers, $D2$ represent a set of all branch names and $D3$ represents a setof all balances.

Each bank_account row consist of a 3-tuple (V_1, V_2, V_3). V_1 is an account number in domain $D1$, $V2$ is a branch name in
domain $D2$, $V3$ is a balance in domain $D3$.

Each instance of a table *r* is a subset of the Cartesian product $D_1 \times D_2 \times D_n$. This definition is identical to the pure mathematical definition of a relation. The main difference is that in a database we assign names to the table's attributes whereas mathematical relations only label their attributes by their integer indices. 1,. , n.

Thus, the terms relation and tuple can be used instead of the terms table and row respectively.

A relation is a set of tuples, a tuple that belongs to a relation and is written as tER. Two relations are same as long as they have same attributes $V_1,.....V_n$ with domains D_2 ,. D_n
and contain same values and same tuples regardless of the order
in which these tuples appear.

The domains *D* of all attributes of a relation should be atomic that is the elements of *D* should not be breakable into sub-compounds. An example of an atomic domain is the set of integers. Whereas the set of all sets of integers is a non-atomic domain.

<u>Database schema and database Instance:</u>

The database schemas and instances can be better understood by analogy to a program written in a programming language. Database schema corresponds to the variable declarations in the program. Every variable has a certain value at a given instant. These values of the variables in a program at a particular time correspond to an instance of a database schema. The relational database schema refers to the relation schema collection.

Example

A college database contains the relations such as Students, Professors, HODs, Teachers, Examiners, and Guides etc.

Description of the relation schema includes the,

(i) Specification of relations name

(ii) Field's name

(iii) Field's domain.

Syntax

R(f1: D1, f2 : D2, f3 : D3, , fi : Di)

Example

Consider a relation schema of the customers in a Bank database as,

Customers (Cid : Integer, Cname : String, Accno : Integer, Bname String, Amt : Real) Where,

Customers = Name of the relation

Cid, Cname, Accno, Bname, Amt = Name of the fields.

Integer, String, Real = Name of the Domain

On the other hand, a set of records or tuples having exactly the same number of fields as the relation schema is called a relation instance.

Number of fields in each tuple = Number of fields in a relation schema

The instance 'C2' of the customer's relation mentioned above can be illustrated as,

	Cid	Cname	Accno	Bname	Amt
Tuples	100	John	41	SBI	10,000
	101	Peter	52	ICICI	20,000
	102	Alexander	64	HDFC	30,000
	103	Elizabeth	52	AXIS	45,000
	104	Joe	78	RBI	50,000

Keys:

Key Constraints

A key constraint can be defined as a statement that consists of minimal subset of attributes that uniquely determine a recordin a table.

Candidate key:

A candidate key is a collection of fields/columns/attributes that uniquely identifies a tuple. For example, in "Customer" relation the attribute "Cid" isa key which uniquely defines a tuple in a relation. No two rows in a relation "Customer" can have the same "Cid" value. The setof attributes that form a candidate key need not be all keys. The attributes may be treated as candidates to be taken as key. For example, the set (Cid, Cname) is a candidate key which means either Cid or Cname can be taken as key but not both, each of them independently and uniquely identifies a particular row. The alternate keys are the candidate keys that are not taken as keys.

Composite Key

Composite key consist of more than one attribute that uniquely identifies a tuple in a relation. All the attributes that form a set of keys and all of them taken together determines a unique row in a table. For example, the set (Cid, Accno) is a composite key which maintains the uniqueness of each row. Both Cid, Accno are taken as keys.

Super Key

A super key is a combination of a candidate key and a composite key i.e., super key is a set of attributes or a single attribute that uniquely identifies a tuple in a relation. For example, consider the super key,

{Cid, Accno, Cname}

Here, all the three attributes taken together can identify a particular record or a combination of any two attributes can identify a particular record or any one of the attribute can identify a particular record.

Primary Key

Only a single attribute can uniquely identify a particular record. More specifically, it can be defined as the candidate key which has been selected as key to identify unique records. For example, “Cid” attribute in “Customer” relation can be treated as a primary key.

Defining Key Constraints

In SQL, one can eliminate the insertion of duplicate data by using a UNIQUE constraint. This constraint helps the user to insert unique values for the columns which have been declared as UNIQUE, forming a candidate key any one of the columns among them can be declared as primary key by using primary key constraints.

Example

Consider the creation of Employee” table,

CREATE TABLE Employee (Ename CHAR(30),Eid INTEGER, Bdate DATE, Address CHAR(30), DNo INTEGER,Age INTEGER, Phone_number NTEGER, UNIQUE (Eid, DNo),CONSTRAINT key PRIMARY KEY (Eid));

This example shows the creation of employee table with attributes Ename, eid, Bdate, Address, Dept Number (DNo), Age, Phone_Number. UNIQUE key constraint is used on columns Eid and DNO which ensures that the values inserted in these columns are unique. The last line of declaration defines a primary key constraint. The syntax used for defining constraint is,

CONSTRAINT Constraint name PRIMARY KEY (key) i.e., CONSTRAINT key PRIMARY KEY (Eid).

This line declares Eid as primary key for the “employee” relation. If the user inserts repeated values for “Eid” then error occurs and constraint name is returned indicating violation of constraint.

Foreign Key Constraints

Foreign key serves as a primary key in another table. Basically, foreign keys provides a link between two tables and are used to maintain data consistency. A foreign key is a set of attributes or single attribute in one relation that forms candidate key in other relation. For example, consider another relation “DEPARTMENT” with attributes (Dname, DNo, Manager_id).

Here, (DNo, Manager_id) is the Candidate key. DNo refers to the dept number to which a particular employee belongs manager_id refers to the manager who is responsible for controlling the department. And therefore, the two tables employees and departments are related to each other by DNo and Manager_id.

With the concept of foreign key, tuples can be inserted and deleted by ensuring that foreign key constraint is not violated.

Further, a foreign key can be used as a reference for some relations. In this case, it is called as self-referential key

Defining Foreign Key Constraints in SQL

Syntax for applying foreign key is as follows,

```
CREATE TABLE Department(Dname CHAR(30), DNo INTEGER, Manager_id
INTEGER, PRIMARY KEY (DNo, Manager_id),FOREIGN KEY (Manager_id)
REFERENCES Employee)
```

The statement FOREIGN KEY (Manager_id) REFERENCES Employee - means that the foreign key “Manager_id” uses primary id “Eid” of employee relation as a reference. Every tuple with Manager_id must match a tuple in employee relation.

Schema Diagrams:

Schema diagrams are used for illustrating a database schema along with primary key and foreign key dependencies. A schema can be defined as a complete description of database.

The specifications for database schema are provided during the database design and this schema does not change frequently. Schema refinement is a process of refining the schema so as to solve the problems caused by redundantly storing the information.

Redundancy in its simplest terms can be defined as duplication of data, which is main cause for all the problems which exists in database. One way to eliminate the redundant data is to make use of decomposition, which is the process of dividing longer relation into smaller relations.

This division into smaller schemes is based on the functional and other dependencies which are specified by the database designer.

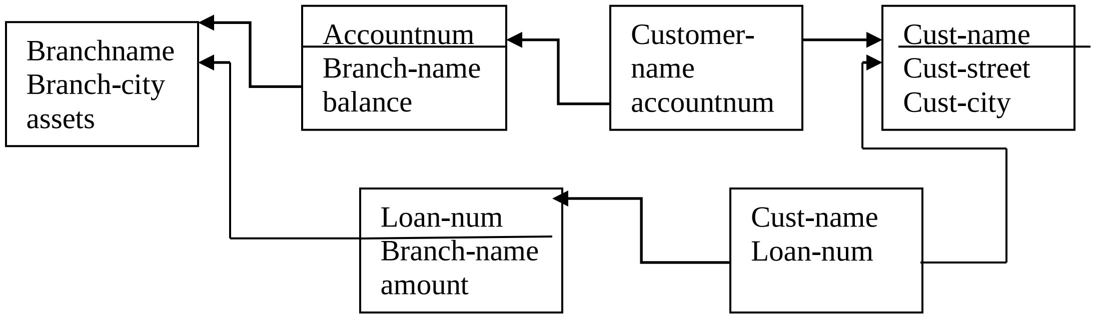


Fig: schema diagram for banking enterprise

A database schema, along with primary key and foreign key dependencies, can be depicted by schema diagrams. Figure shows the schema diagram for banking system. Each relation appears as a box, with the relation name at the top in blue, and the attributes listed inside the box. Primary key attributes are shown underlined. Foreign key dependencies appear as arrows from the foreign key attributes of the referencing relation to the primary key of the referenced relation

<u>Relational Query Languages:</u>

A query language refers to a language wherein the user requests and retrieves data from a database by sending queries. Typically, this type of language possess higher ranking than that of a standard programming language. A relational query language is broadly classified into two types,

(i) Procedural language.

(ii) Non-procedural language.

(i) Procedural Language

In this type of language, user instructs the system about set of operations to be performed in a sequential manner. It provides complete details about what operations must be performed and how they must be carried.

Example: Relational Algebra

(ii) Non-Procedural Language

In this type of language, the user provides information only about what operations must be performed. In other words, user specifies only about the data that must be retrieved but does not provide procedure for retrieving the data.

Example: Tuple relational calculus, domain relational calculus.

In practice, query languages involve both procedural and non-procedural approaches.

Relational Operations:

Procedural relational query language supports various operations which can be employed either in single relation or multiple relation. Since, the result of relational query is always a relation; the relational operation can also be applied to the results of queries and set of relations provided.

Relational Algebra

Relational algebra is a query language, which is used to manipulate the data in the relational data model. The queries in relational algebra are formed using set of operators that accepts at most two relation instances as arguments and then returns a new relation instance as the output. This property of composing the queries using operators helps to easily create a complex query.

The different operations of relational algebra include,

1. Select
2. Project
3. Set operations
4. Rename
5. Joins
6. Division.

Select Operation (σ)

It selects tuples that satisfy the given predicate from a relation.

Notation − $\sigma_p(r)$

Where **σ** stands for selection predicate and **r** stands for relation. *p* is prepositional logic formula which may use connectors like **and, or,** and **not**. These terms may use relational operators like − =, ≠, ≥, < , >, ≤.

BRANCH_NAME	LOAN_NO	AMOUNT
Downtown	L-17	1000
Redwood	L-23	2000
Perryride	L-15	1500
Downtown	L-14	1500
Mianus	L-13	500
Roundhill	L-11	900
Perryride	L-16	1300

Input: σ BRANCH_NAME="perryride" (LOAN)

Output:

BRANCH_NAME	LOAN_NO	AMOUNT
Perryride	L-15	1500

Perryride	L-16	1300

2. Project Operation:

- This operation shows the list of those attributes that we wish to appear in the result. Rest of the attributes is eliminated from the table.
- It is denoted by ∏.

1. Notation: ∏ A1, A2, An (r)

Where A1, **A2**, **A3** is used as an attribute name of relation **r**.

Example: CUSTOMER RELATION

NAME	STREET	CITY
Jones	Main	Harrison
Smith	North	Rye
Hays	Main	Harrison
Curry	North	Rye
Johnson	Alma	Brooklyn
Brooks	Senator	Brooklyn

Input:

1. ∏ NAME, CITY (CUSTOMER)

Output:

NAME	CITY
Jones	Harrison
Smith	Rye
Hays	Harrison
Curry	Rye
Johnson	Brooklyn
Brooks	Brooklyn

3. Set Operations

The following are the different standard set operations defined in relational algebra.

(a) Union ($\cup$)

When applied on two relations, it returns all the rows which are either present in first relation or second relation or in both the relations. It does not return the rows which has the same tuple values. All the rows returned by this operator are unique. One of the constraint of union operator is that both the relations must be "union-compatible" i.e., both the relations must have the same number of columns and also these columns must have the same domain (type of data being stored in each column).

$\cup$ If A and B are two relations and are union compatible then union of these two relations is expressed as, A $\cup$ B Returns uniquely all the tuples of A and B.

Example

Consider a simple Student1 and Student2 relation which represent juniors and seniors.

Student (S1)

Roll No	Class	Name
2101	VIII	Ravi
2102	VII	Kumar
2103	VI	Sheena

Student (S2)

Roll No	Class	Name
3101	IX	Sodir
3102	X	Dhani
3103	XI	Rahul

S1 ∪ S2

Roll No	Class	Name
2101	VIII	Ravi
2102	VII	Kumar
2103	VI	Sheena
3101	IX	Sodir
3102	X	Dhani
3103	XI	Rahul

(b) **Intersection (∩)**

When applied on two relations, A and B, (A∩B it returns all the rows which are common to both the relations. Most importantly, this operator can be applied to only those relations that are union compatible. For example if intersection operator (∩) is applied to S1 and S2 (above relations) then the result will be an empty relation as nothing is common between these relations.

S1∩ S2 = ~ empty relation.

If intersection operator () is applied on EMPLOYEE relation and Projection 2 – the resulting relation will be similar toprojection 2 relation as it has only two tuples in common.

(c) **Set-Difference (Minus '–')**

This operator is also called as "Minus" operator. If A and B are two relations and are union compatible than (A – B) willreturn all the rows which are in A but not in B.

(d) **Cross-Product (*)**

This operator returns all the tuples of relation A plus all the tuples of relation B. Cross product is denoted as,

A × B – returns the Cartesian product of two relations. The resulting relation will be bigger in size than the two relations.

A		B		
Column 1	**Column 2**	**Column A**	**Column 1**	**Column B**
1	a	a1	1	b1
2	b	a2	2	b2
3	c	a3	3	b3

A * B

Column 1	**Column 2**	**Column A**	**Column 1**	**Column B**
1	a	a1	1	b1
1	b	a1	2	b2
1	c	a1	3	b3
2	a	a2	1	b1
2	b	a2	2	b2
2	c	a2	3	b3
3	a	a3	1	b1
3	b	a3	2	b2
3	a	a3	3	b3

<u>4. Renaming (ρ)</u>

This operator is used to rename the relation or attributes or both. The syntax for this operator is,

ρ(STUDENT1, STUDENT)

<u>5. Joins</u>

Joins are used to combine the information of two relations which have at least one field in common. It is one of the most important operator of RDBMS. In simple words, a join operation can be defined as a Cartesian product followed by selection or projection operators. The different forms of join operators are,

(a) Conditional join (Meta join □)

(b) Equi-join

(c) Natural join

(d) Outer joins.

(a) Conditional Join (Meta Join⋈)

This join returns a relation that includes a set of rows from the Cartesian product of two relations A and B such that eachrow satisfies a given condition C.

This can be denoted as A⋈ B

i.e., join A and B based on some condition 'C'. This join operation is same as the Cartesian product performed on tworelations followed by a selection operator. Thus,

$A \bowtie_C B$

The application of conditional join on relations EMPLOYEE and DEPT_LOCATION results into a new relation consisting of 11 fields.

Example

⋈EMPLOYEE DEPT_LOCATION DNo >7

This statement means,

The relation EMPLOYEE and DEPT_LOCATION are joined based on the condition DNo>7. The resulting relation will include all the tuples from EMPLOYEE and dept_LOCATION where DNo.>7.

Projection 7

Ename	**Eid**	**Bdate**	**Address**	**Sex**	**($)Sal**	**DNo**	**Phone number**	**Age (years)**	**DNo**	**DLocation**
Brown	12345264	28-7-1968	Chicago	M	40.000	8	773210192	28	8	Detroit
John	12345261	10-8-1965	New Jersey	M	25000	9	773213218	41	9	Chicago
Brown	12345264	28-7-1968	Chicago	M	40000	8	773271872	51	8	Detroit
Bill	12345265	25-7-1955	Detroit	F	35000	8	773271842	51	9	Chicago
Bill	12345265	25-7-1955	Detroit	F	35000	8	773271842	52	9	Chicago
Jill	12345266	04-4-1965	New York	F	42000	8	773291828	41	8	Detroit
Jill	12345266	04-4-1965	New York	F	42000	8	773291628	41	9	Chicago
Donald	12345267	02-8-1768	Detroit	M	20000	9	773423145	28	8	Detroit
Donald	12345267	62-8-1768	Detroit	M	20000	9	773423175	28	9	Chicago
John	12345261	10-8-1965	New Jersey	M	25000	8	773213218	41	8	Detroit

(b) Equi-Join

Equi-join is same as conditional join, the only difference is that, equijoin uses equity '=' operator to join the two relations.

For example one may join DEPARTMENT and DEPT_LOCATION relation with the condition that,

DNo1.Department = DNo2.Dept_location. Where DNo1 and DNo2 are two instances of respective relations. This conditionjoins the tuples where DNo1 = DNo2. The degree of resulting relation will be the sum of degrees of two relation minus the number of fields they have is common. More precisely,

Example

The resulting relation DEPARTMENT$_{\text{DNo1.DEPARTMENT}}$ ⋈= $_{\text{DNo2.DEPT\_LOCATION}}$ DEPT_LOCATION will containall the fields of DEPARTMENT AND DEPT_LOCATION and the common fields will be included only once.

DName	DNo	Manage_id	DLocation
Administration	7	2431	New York
Research	8	3731	Detroit
Head Quarters	9	4341	Chicago

(c) Natural Join

This is default join operation i.e., no condition is specified. Natural join is equivalent to Cartesian product. If two relations have a common field then the application of natural join is equivalent to equi join and if no field is common then the natural join is Cartesian product of the two relations.

This operation can be denoted as,,

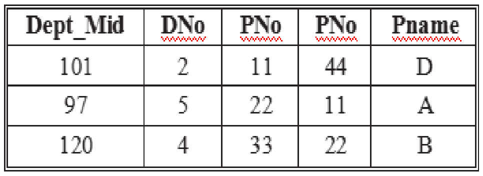
A ⋈ B where A and B are two relations.

If natural join operation is applied on DEPARTMENT and DEPT_LOCATION then the result will be same as Projection8 as they have only DNo field in common.

(d) Outer Joins

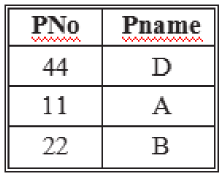
Outer joins are the special variant of "join" operation that are dependent on NULL values. Generally, a 'join' operation performs the cross product of two tables and apply certain join condition. Then it select those rows from the cross product that satisfies the given condition. But with outer joins, DBMS allows the user to select those rows which are common (satisfies the given) and even those rows that does not satisfies the given condition.

Example



Dept_Mid	DNo	PNo	PNo	Pname
101	2	11	44	D
97	5	22	11	A
120	4	33	22	B

Department D1



PNo	Pname
44	D
11	A
22	B

Project P1

If join operation is performed on these two tables,

SELECT *.D1, * .P1

FROM Department D1, Project P1WHERE D1.PNo = P1.PNo;

The result of this statement is as follows,

Dept_Mid	DNo	PNo	PNo	Pname
101	2	11	11	A
97	5	22	22	B

This table shows the simple join operation of two tables where only those rows are selected that satisfied the condition.

However, in order to include rows that does not satisfy the condition, the concept of OUTER JOINS can be used.

There are three types of outer joins namely,

(i) Left outer join

(ii) Right outer join

(iii) Full outer join.

(i) Left Outer Join

Left outer join list all those rows which are common to both the tables along with the unmatched rows of the second table are,

Example

SELECT * .D1, * .P1

FROM Department D1, LEFT OUTER JOIN The result of this statement is as follows,

Dept_Mid	DNo	PNo	PNo	Pname
101	2	11	11	A
97	5	22	22	B
120	4	33	NULL	NULL

So, the left outer join resulted in a relation that have common rows from both the tables and also the row which does not have match in the other table. The values of the attributes corresponding to second table are NULL values.

(ii) Right Outer Join

Right outer join is same as the left outer join but the only difference is the unmatched rows of second table (specified on the right hand side) are listed along with the common rows of both the tables.

SELECT *.D1, *.P1

FROM Department D1

RIGHT OUTER JOIN Project P1WHERE D1.PNo = P1.PNo

The values of attributes for the first table are declared as NULL.

Dept_Mid	**DNo**	**PNo**	**PNo**	**Pname**
NULL	NULL	NULL	44	D
101	2	11	11	A
97	5	22	22	C

(iii) <u>Full Outer Join</u>

This is same as the right outer join and left outer join but only difference is unmatched rows of both tables are listed alongwith the common rows of the tables.

SELECT *.D1, * .P1

FROM Department D1, FULL OUTER JOIN Project P1

WHERE D1.PNo = P1.PNo;
The following table shows the result,

Dept_Mid	**DNo**	**PNo**	**PNo**	**Pname**
101	2	11	11	A
97	5	22	22	B
120	4	33	NULL	NULL
NULL	NULL	NULL	44	D

In this relation all the matched and unmatched columns of both the table are displayed, the values for the unmatched attributes are entered as NULL.

6. Division

Consider two relations *P* and *Q* such that *P* contains two attributes ‘*a*’ and ‘*b*’ and *Q* contains only one attribute ‘*b*’. Theattribute ‘*b*’ in *Q* has similar domain as that of *P*. Thus the division operation represented as *P*/*Q*, can be defined as the set of all values of ‘*a*’ wherein each value of ‘*b*’ in *Q* has a tuple <*a*, *b*> in *P*.

Assume that for a set of ‘*b*’ values in *P* there exists a corresponding set of ‘*a*’ values in *P*. Thus, the value ‘*a*’ is said to
occur in the result of *P*/*Q* if the set of values of ‘*b*’ in *P* has all the values of ‘*b*’ in *Q*.