Difference between File System and DBMS

Basis	File System	DBMS
Structure	The file system is software that manages and organizes the files in a storage medium within a computer.	DBMS is software for managing the database.
Data Redundancy	Redundant data can be present in a file system.	In DBMS there is no redundant data.
Backup and Recovery	It doesn't provide backup and recovery of data if it is lost.	It provides backup and recovery of data even if it is lost.
Query processing	There is no efficient query processing in the file system.	Efficient query processing is there in DBMS.
Consistency	There is less data consistency in the file system.	There is more data consistency because of the process of normalization.
Complexity	It is less complex as compared to DBMS.	It has more complexity in handling as compared to the file system.
Security Constraints	File systems provide less security in comparison to DBMS.	DBMS has more security mechanisms as compared to file systems.
Cost	It is less expensive than DBMS.	It has a comparatively higher cost than a file system.
Data Independence	There is no data independence.	In DBMS data independence exists.
User Access	Only one user can access data at a time	Multiple users can access da time
Meaning	The user has to write procedures for managing databases	The user not required to wri procedures
Sharing	Data is distributed in many files. So, not easy to share data	Due to centralized nature sharing is e
Data Abstraction	It give details of storage and representation of data	It hides the internal details of Database
Integrity Constraints	Integrity Constraints are difficult to implement	Integrity constraints are easy implement
Example	Cobol, C++	Oracle, SQL Server

ADVANTAGES OF DBMS

The advantages of the DBMS are explained below −

- **Redundancy problem can be solved.**

In the File System, duplicate data is created in many places because all the programs have their own files which create data redundancy resulting in wastage of memory. In DBMS, all the files are integrated in a single database. So there is no chance of duplicate data.

For example: A student record in a library or examination can contain duplicate values, but when they are converted into a single database, all the duplicate values are removed.

- **Has a very high security level.**

Data security level is high by protecting your precious data from unauthorized access. Only authorized users should have the grant to access the database with the help of credentials.

- **Presence of Data integrity.**

Data integrity makes unification of so many files into a single file. DBMS allows data integrity which

MS

tiple users can access data
.

ser not required to writ
dures

entralized nature sharing is e

the internal details of
e

constraints are easy t
t

L Server

have their
the files are

but when

nly authorized

grity which

makes it easy to decrease data duplicity Data integration and reduces redundancy as well as data inconsistency

- **Support multiple users.**

DBMS allows multiple users to access the same database at a time without any conflicts

- **Avoidance of inconsistency.**

DBMS controls data redundancy and also controls data consistency Data consistency is nothing but if you want to update data in any files then all the files should not be updated again

In DBMS, data is stored in a single database so data becomes more consistent in comparison to file processing systems

- **Shared data**

Data can be shared between authorized users of the database in DBMS All the users have their own right to access the database Admin has complete access to the database He has a right to assign users to access the database

- **Enforcement of standards**

As DBMS have central control of the database. So, a DBA can ensure that all the applications follow some standards such as format of data, document standards etc These standards help in data migrations or in interchanging the data

- **Any unauthorized access is restricted**

Unauthorized persons are not allowed to access the database because of security credentials

- **Provide backup of data**

Data loss is a big problem for all the organizations In the file system users have to back up the files in regular intervals which lead to waste of time and resources

DBMS solves this problem of taking backup automatically and recovery of the database

Tunability

Tuning means adjusting something to get a better performance Same in the case of DBMS, as it provides tunability to improve performance DBA adjusts databases to get effective results

Disadvantages of DBMS

The disadvantages of DBMS are as follows

- **Complexity**

The provision of the functionality that is expected of a good DBMS makes the DBMS an extremely complex piece of software Database designers, developers, database administrators and end-users must understand this functionality to take full advantage of it

Failure to understand the system can lead to bad design decisions, which leads to a serious consequence for an organization

- ***Size***

The functionality of DBMS makes use of a large piece of software which occupies megabytes of disk space

- **Performance**

Performance may not run as fast as desired

- **Higher impact of a failure**

The centralization of resources increases the vulnerability of the system because all users and applications rely on the availability of DBMS the failure of any component can bring operation to halt

- **Cost of DBMS**

The cost of DBMS varies significantly depending on the environment and functionality provided

There is also the recurrent annual maintenance cost

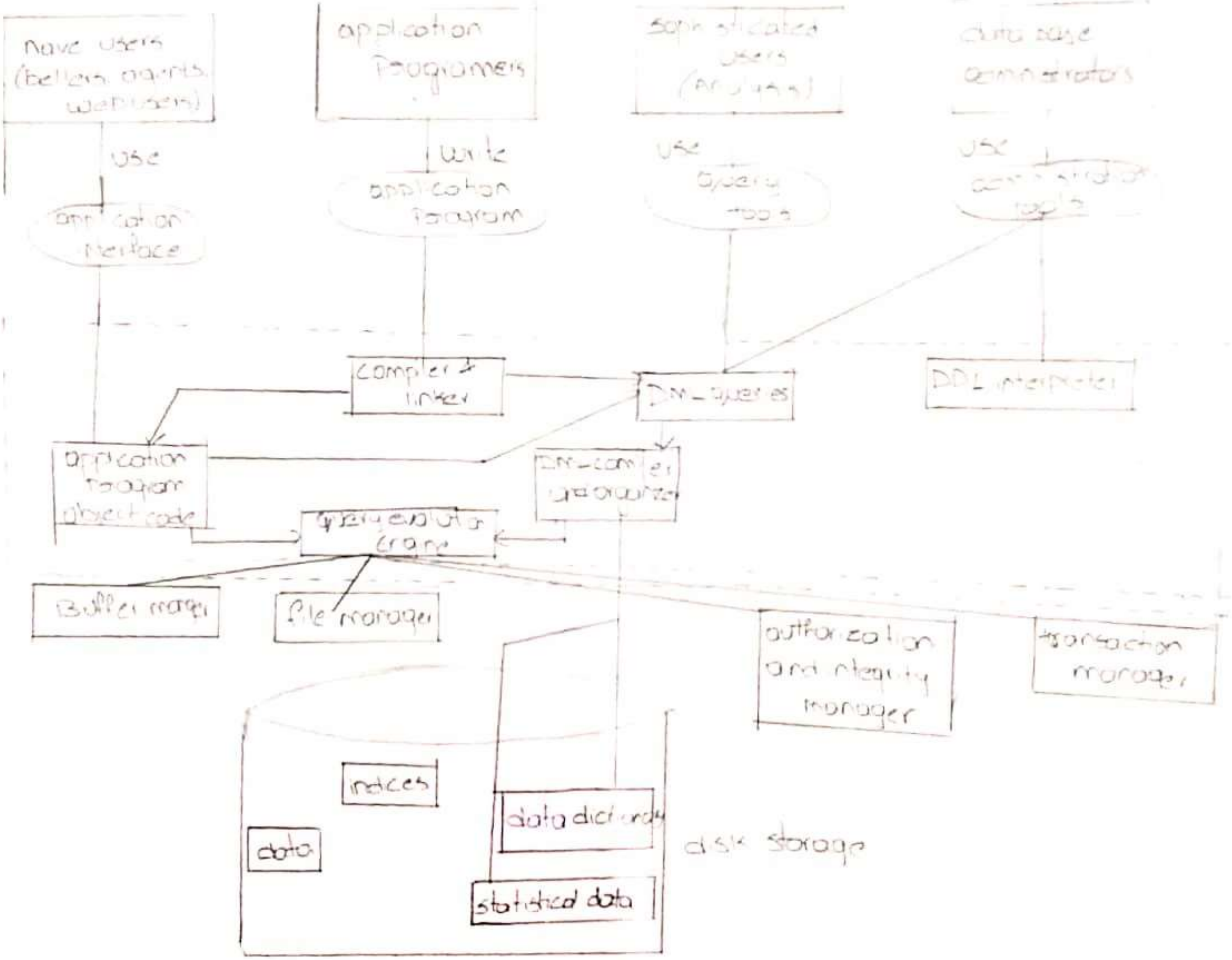
1. DATABASE SYSTEMS APPLICATIONS

A Database management system is a computerized record-keeping system It is a repository or a container for collection of computerized data files The overall purpose of DBMS is to allow the users to define store, retrieve and update the information contained in the database on demand Information can be anything that is of significance to an individual or organization

Databases touch all aspects of our lives. Some of the major areas of application are as follows:

1 Enterprise Information
2 Banking and Finance
3 Universities
4 Airlines
5 Telecommunication

System structure



A primary goal of a database system is to retrieve information from and store new information into the database. People who work with a database can be categorized as database users or database administrators

1.12.1 Database Users and User Interfaces

- **Application programmers**
  - interact with system through DML calls
- **Sophisticated users** 複雜, 多用途
  - Submit query without write program
  - E.g. OLAP (Online analytical processing). data mining tools
- **Specialized users**
  - write specialized database applications that do not fit into the traditional data processing framework
  - E.g. CAD. expert system. complex data type (graphics. audio)
- **Naive users** (end user)
  - invoke one of the permanent application programs that have been written previously
  - E.g. people accessing database over the web. <u>bank tellers</u>. <u>clerical staff</u> 辦事員

9. Database Architecture

A Database Architecture is a representation of DBMS design. It helps to design, develop, implement, and maintain the database management system. A DBMS architecture allows dividing the database system into individual components that can be independently modified, changed, replaced, and altered. It also helps to understand the components of a database.

A Databasestores critical information and helps access data quickly and securely. Therefore, selecting the correct Architecture of DBMS helps in easy and efficient data management.

- Types of DBMS Architecture
- 1-Tier Architecture
- 2-Tier Architecture
- 3-Tier Architecture

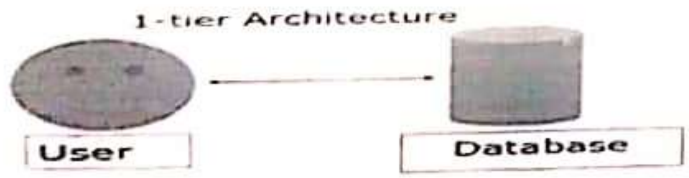
Types of DBMS Architecture

There are mainly three types of DBMS architecture:

- One Tier Architecture (Single Tier Architecture)
- Two Tier Architecture
- Three Tier Architecture

1-Tier Architecture

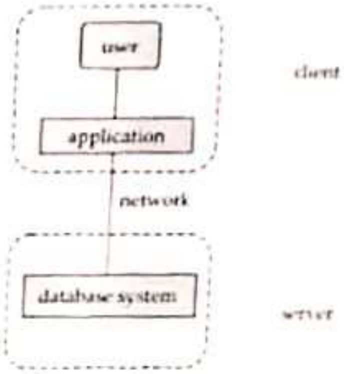
1 Tier Architecture in DBMS is the simplest architecture of Database in which the client, server, and Database all reside on the same machine. A simple one tier architecture example would be anytime you install a Database in your system and access it to practice SQL queries. But such architecture is rarely used in production.



2-Tier Architecture

A **2 Tier Architecture** in DBMS is a Database architecture where the presentation layer runs on a client (PC, Mobile, Tablet, etc.), and data is stored on a server called the

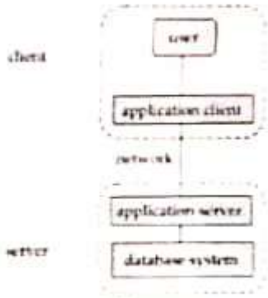
second tier. Two tier architecture provides added security to the DBMS as it is no exposed to the end-user directly. It also provides direct and faster communicatio



(a) Two-tier architecture

3-Tier Architecture

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



(b) Three-tier architecture

Attribute : each column in a table. Attributes are the Properties which define a relation

Eg: Student Roll no, NAME etc

student rollno	Name

Schema : The over all design of the data base is called schema

Eg:
```
Create table student
(
ID      number(5),
Name    Varchar2 (10),
Marks   numeric (3).
);
```

iii, Tuple : It is nothing but a single row of a table which contains single record.

ID	Name	marks
~~Hari~~ 112	Hari	58
~~Ram~~ 113	Ram	78
~~Sam~~ 114	sam	98

→ Tuple.

iv, Domain : Domains are the data type definition that resolve to a primitive datatype or another domains.

~~Relational~~

A domain defines the permitted range of value attributes.

Each attribute has a particular

that is, a domain of values

and semantics of a relation.

for an attribute of an entity.

```
- Eg:  Create table Employee
       (
         ID       number(10),  → domain.
         Ename    varchar2(10),
         phone.   number (10)
       );
```

v, Relational Instances:-

Relational instance is a finite set of tuples in the RDBMS system. Relation instances never have duplicate tuples.

4. DATABASE LANGUAGES

A database system provides a **data-definition language** to specify the database schema and **data-manipulation language** to express database queries and updates. In practice, the data-definition and data-manipulation languages are not two separate languages, instead they simply form parts of a single database language, such as the widely used SQL language.

1.4.1 Data-Manipulation Language

A data-manipulation language (DML) is a language that enables users to access or manipulate data as organized by the appropriate data model. The types of access are:

- Retrieval of information stored in the database
- Insertion of new information into the database
- Deletion of information from the database
- Modification of information stored in the database

There are basically two types:

- Procedural DMLs require a user to specify *what* data are needed and *how* to get those data.
- Declarative DMLs (also referred to as nonprocedural DMLs) require a user to specify *what* data are needed *without* specifying how to get those data.

Declarative DMLs are usually easier to learn and use than are procedural DMLs. However, since a user does not have to specify how to get the data, the database system has to figure out an efficient means of accessing data.

A query is a statement requesting the retrieval of information. The portion of a DML that involves information retrieval is called a query language. Although technically incorrect, it is common practice to use the terms *query language* and *data-manipulation language* synonymously.

1.4.2 Data-Definition Language (DDL)

We specify a database schema by a set of definitions expressed by a special language called a **data-definition language (DDL)**. The DDL is also used to specify additional properties of the data.

We specify the storage structure and access methods used by the database system by a set of statements in a special type of DDL called a data storage and definition language. These statements define the implementation details of the database schemas, which are usually hidden from the users.

The data values stored in the database must satisfy certain consistency constraints. For example, suppose the university requires that the account balance of a department must never be negative. The DDL provides facilities to specify such constraints. The database system checks these constraints every time the database is updated. In general, a constraint can be an arbitrary predicate pertaining to the database. However, arbitrary predicates may be costly to test. Thus, database systems implement integrity constraints that can be tested with minimal overhead:

ccess
es of

to

to

ral
he

of
h
d

ge called a

rties of the

he database
and

- **Domain Constraints**. A domain of possible values must be associated with every attribute (for example, integer types, character types, date/time types). Declaring an attribute to be of a particular domain acts as a constraint on the values that it can take. Domain constraints are the most elementary form of integrity constraint. They are tested easily by the system whenever a new data item is entered into the database.

- **Referential Integrity**. There are cases where we wish to ensure that a value that appears in one relation for a given set of attributes also appears in a certain set of attributes in another relation (referential integrity). For example, the department listed for each course must be one that actually exists. More precisely, the *dept_name* value in a *course* record must appear in the *dept_name* attribute of some record of the *department* relation. Database modifications can cause violations of referential integrity. When a referential-integrity constraint is violated, the normal procedure is to reject the action that caused the violation.

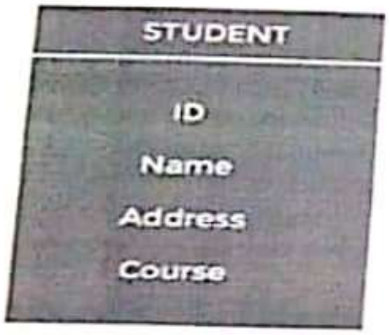
- **Assertions**. An assertion is any condition that the database must always satisfy. Domain constraints and referential-integrity constraints are special forms of assertions. However, there are many constraints that we cannot express by using only these special forms. For example, "Every department must have at least five courses offered every semester" must be expressed as an assertion. When an assertion is created, the system tests it for validity. If the assertion is valid, then any future modification to the database is allowed only if it does not cause that assertion to be violated.

- **Authorization**. We may want to differentiate among the users as far as the type of access they are permitted on various data values in the database. These differentiations are expressed in terms of **authorization**, the most common being: **read authorization**, which allows reading, but not modification, of data; **insert authorization**, which allows insertion of new data, but not modification of existing data; **update authorization**, which allows modification, but not deletion, of data; and **delete authorization**, which allows deletion of data. We may assign the user all, none, or a combination of these types of authorization.

3. Keys

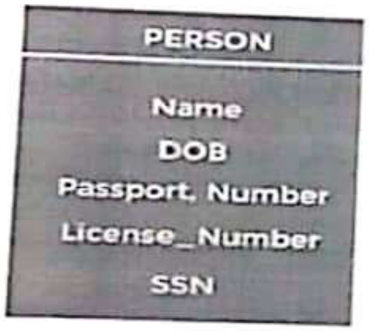
- Keys play an important role in the relational database.
- It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.

For example, ID is used as a key in the Student table because it is unique for each student. In the PERSON table, passport_number, license_number, SSN are keys since they are unique for each person.



STUDENT

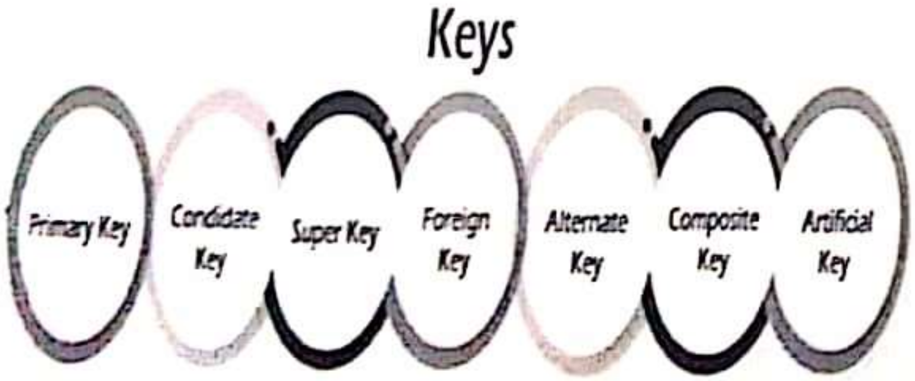
ID
Name
Address
Course



PERSON

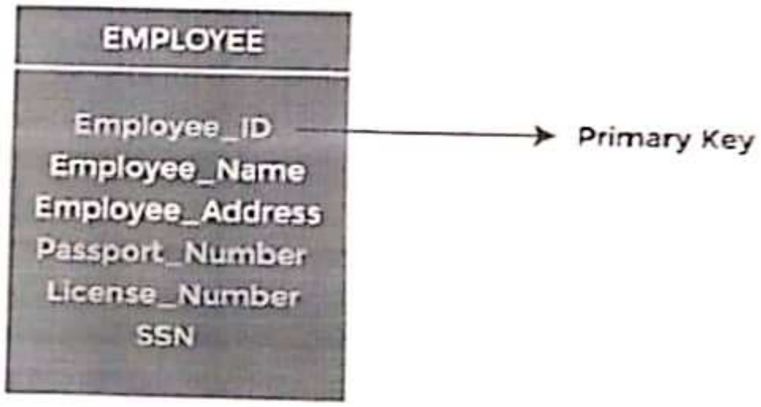
Name
DOB
Passport. Number
License_Number
SSN

Types of keys:



1. Primary key

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



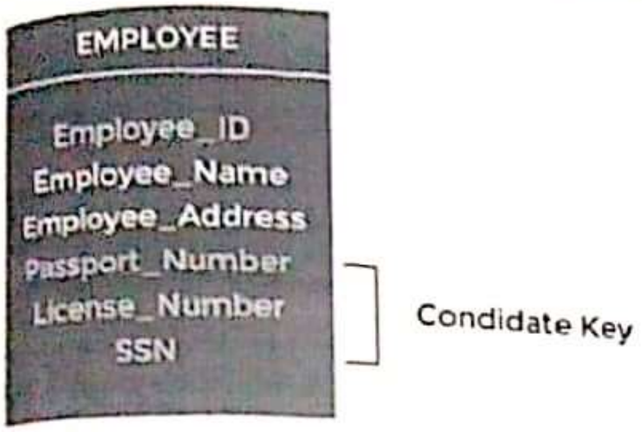
EMPLOYEE

Employee_ID
Employee_Name
Employee_Address
Passport_Number
License_Number
SSN

2. Candidate key

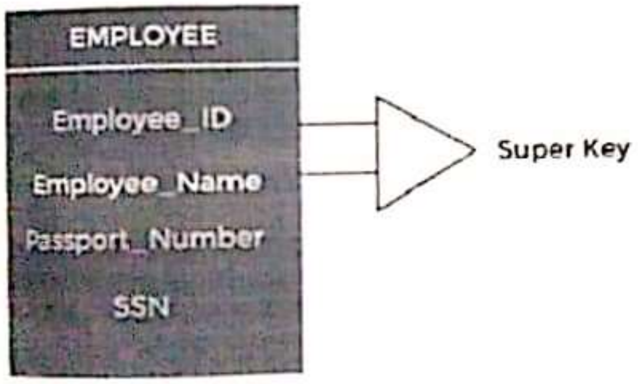
- A candidate key is an attribute or set of attributes that can uniquely identify a tupl
- Except for the primary key, the remaining attributes are considered a candida candidate keys are as strong as the primary key.

For example: In the EMPLOYEE table, id is best suited for the primary key. The rest of the 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.



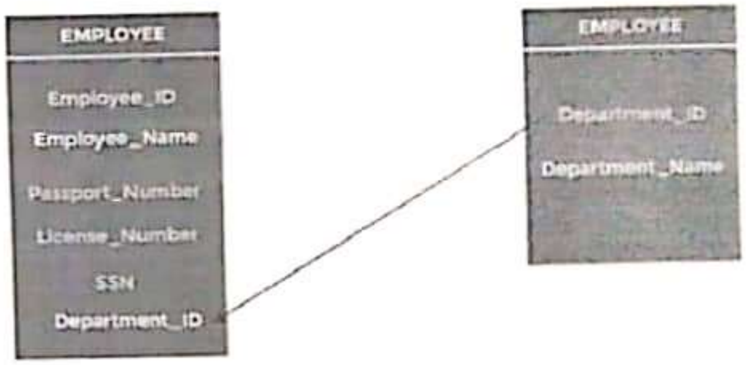
For example: In the above EMPLOYEE table, for(EMPLOEE_ID, EMPLOYEE_NAME), the name of two employees can be the same, but their EMPLYEE_ID can't be the same. Hence, this combination can also be a key.

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.

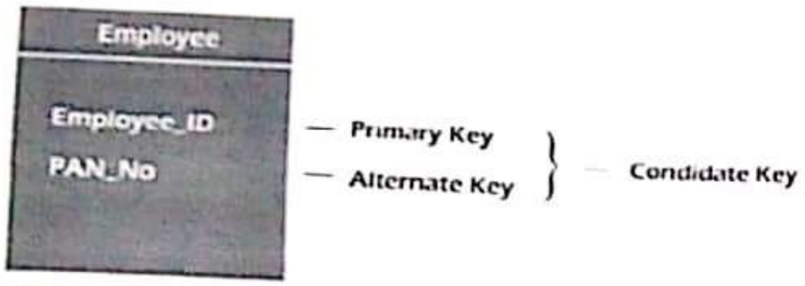
- Every employee works in a specific department in a company, and employee and department are two different entities. So we can't store the department's information in the employee table. That's why we link these two tables through the primary key of one table.
- We add the primary key of the DEPARTMENT 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.



5 Alternate key

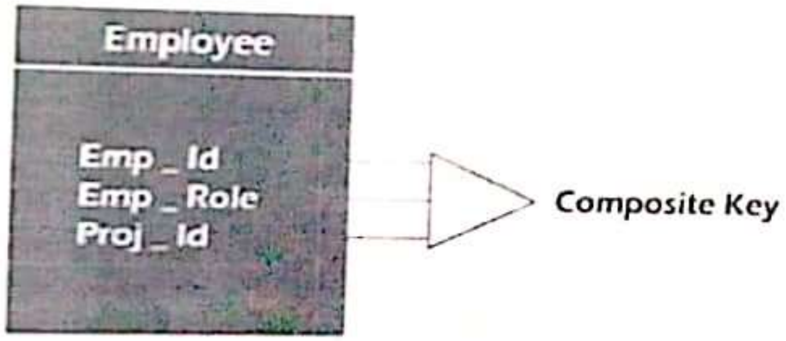
There may be one or more attributes or a combination of attributes that uniquely identify each tuple in a relation. These attributes or combinations of the attributes are called the candidate keys. One key is chosen as the primary key from these candidate keys, and the remaining candidate key, if it exists, is termed the alternate key. **In other words,** the total number of the alternate keys is the total number of candidate keys minus the primary key. The alternate key may or may not exist. If there is only one candidate key in a relation, it does not have an alternate key.

For example, employee relation has two attributes, Employee_Id and PAN_No, that act as candidate keys. In this relation, Employee_Id is chosen as the primary key, so the other candidate key, PAN_No, acts as the Alternate key.

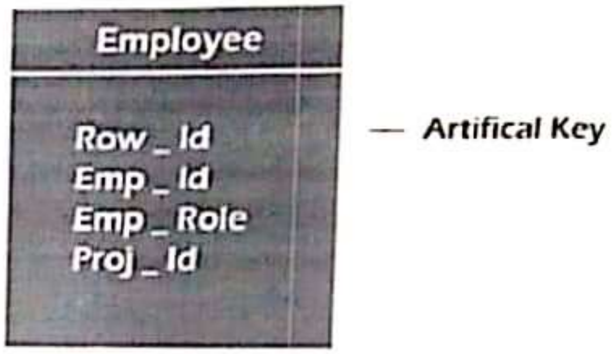


6. Composite key

Whenever a primary key consists of more than one attribute, it is known as a composite key. This key is also known as Concatenated Key



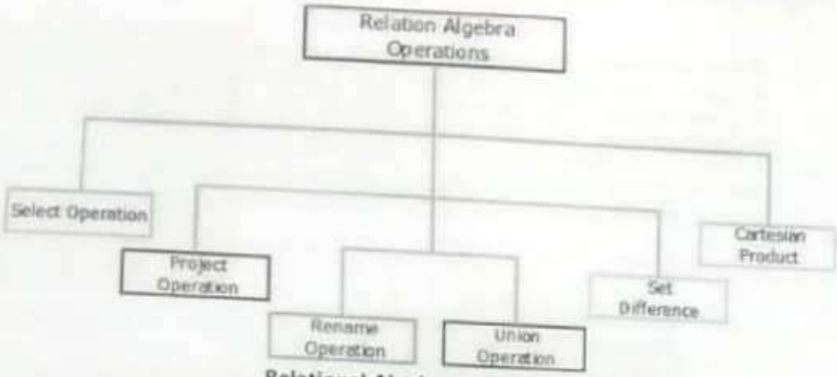
For example, in employee relations, we assume that an employee may be assigned multiple roles, and an employee may work on multiple projects simultaneously. So the primary key will be composed of all three attributes, namely Emp_ID, Emp_role, and Proj_ID in combination. So these attributes act as a composite key since the primary key comprises more than one attribute



7. Artificial key

The key created using arbitrarily assigned data are known as artificial keys. These keys are c when a primary key is large and complex and has no relationship with many other relations. Th values of the artificial keys are usually numbered in a serial order

For example, the primary key, which is composed of Emp_ID, Emp_role, and Proj_ID, is la employee relations. So it would be better to add a new virtual attribute to identify each tuple relation uniquely



Relational Algebra : Operations

RELATIONAL OPERATIONS

1.The Select Operation : This operation is used to fetch rows from given table or relation on the basis of given conditions, it is denoted by "Sigma(σ)".

Syntax :σ [illegible](Relation Name)

Here, "σ" is the select operation symbol. R is the relation from which the data needs to be fetched on the basis of conditions. Also, relational operators such as =, <, > etc. can also be used along. Let's look at the example to get a clear picture of this.

For example :Consider the table of relation R(Roll No, Name, Age, Marks). If we want to select the name and age of student, then it can be done by:

Query Used :σ [illegible](Student_Details)

Student_Details

Roll No	Name	Age	Marks
1	Anoop	22	30
2	Anurag	23	32
3	Ganesh	21	31

Query Output

Name	Age
Anoop	22
Anurag	23

Relational Algebra : Select Operation

2.The Project Operation : This operation is also used to fetch all the rows/tuples/data according to the requested attribute. It means, using project operation one can simply fetch all the tuples corresponding to a single attribute or multiple attributes. It does not supports any conditions as select operation and is denoted using "Pie(π)".

Syntax :π [illegible](Relation Name)

For example :Consider the table of relation R(Roll No, Name, Age, Marks). If we want to project the marks column, then it can be done by :

Query Used :π [illegible](Student_Details)

Student_Details

Roll No	Name	Age	Marks
1	Anoop	22	30
2	Anurag	23	32
3	Ganesh	21	31

Query Output

Marks
30
32
31

Relational Algebra : Project Operation

3.The Rename Operation : When operations like project and select are performed to fet results, these results requires renaming. They can be renamed using the rename operati which is denoted using Greek letter "Rho(ρ)".

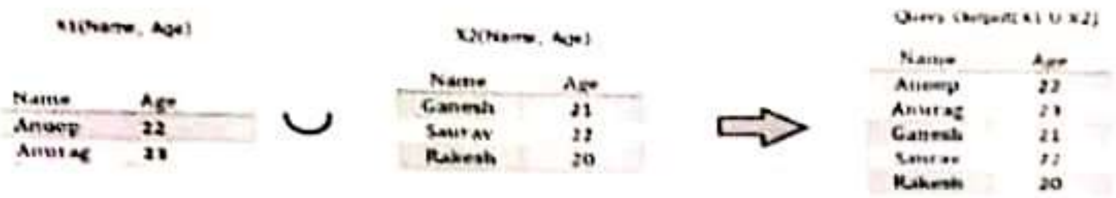
Syntax :ρ [illegible](New Relation)

4.The Union Operation : In order to fetch data from two relations to generate new relati with combined capabilities, union operations can be used. The union operation fetches data from both tables and projects it accordingly. It is denoted through "Union Symbol(Also, two things need to keep in mind while applying union operation are :

- Both the relations compulsory to have same number of attributes.
- Both the relations compulsory to have same domain for attributes.

Syntax : X_1 U X_2, where X_1& X_2 are two different relations satisfying the above two condit

For example :Consider the two tables with relations X_1(Name, Age) and X_2(Name, Age). If wish to apply the union operation, then it can be done by :



X1(Name, Age)

Name	Age
Anoop	22
Anurag	23

∪

X2(Name, Age)

Name	Age
Ganesh	21
Saurav	22
Rakesh	20

⇨

Query Output(X1 ∪ X2)

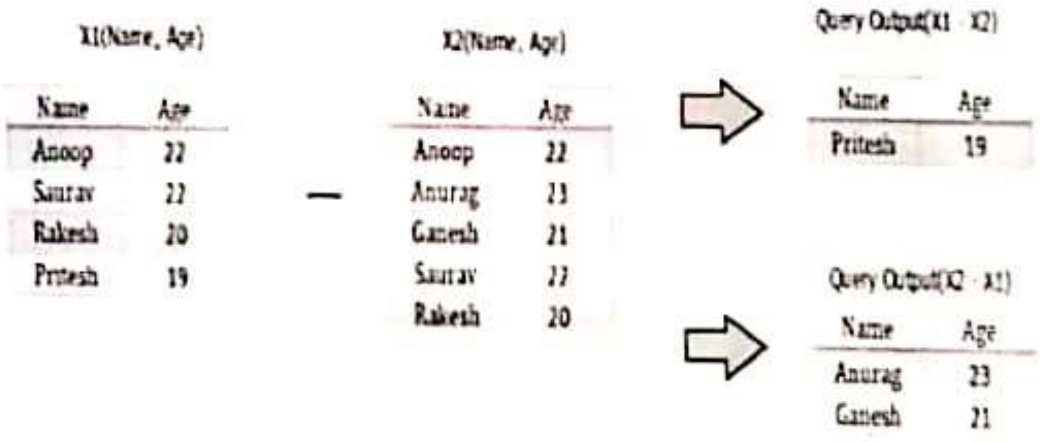
Name	Age
Anoop	22
Anurag	23
Ganesh	21
Saurav	22
Rakesh	20

Relational Algebra : Union Operation

5.The Set Difference Operations : In order to fetch the data which is not present in any one of the relation, set difference operation is used. The set difference operation is denoted by "Minus(-)".

Syntax :$X_1 - X_2$ or $X_2 - X_1$, where X_1 & X_2 are two different relations having some attributes.
Note :$X_1 - X_2 \neq X_2 - X_1$ (Not Commutative)

For example :Consider the two tables with relations X_1(Name, Age) and X_2(Name, Age). If we wish to apply the set difference operation, then it can be done by :



X1(Name, Age)

Name	Age
Anoop	22
Saurav	22
Rakesh	20
Pritesh	19

—

X2(Name, Age)

Name	Age
Anoop	22
Anurag	23
Ganesh	21
Saurav	22
Rakesh	20

⇨

Query Output(X1 - X2)

Name	Age
Pritesh	19

⇨

Query Output(X2 - X1)

Name	Age
Anurag	23
Ganesh	21

Relational Algebra : Set Difference Operation

6.Cartesian Product : The Cartesian product operation will generate the possible combinations among the tuples from the relations resulting in table containing all the data. It combines the information of two or more relations in one single relation. Cartesian product is different from union operation and is denoted by "Cross(X)".

Syntax :$A_1 - A_2$, where A_1 & A_2 are two different relations having some attributes.
For example :Consider the two tables with relations A_1(Name, Roll No)and A_2(Name, Roll No). If we wish to apply the Cartesian product operation, then it can be done by :

4. SCHEMA DIAGRAMS

A database schema, along with primary key and foreign key dependencies, can be depicted by schema diagrams. Figure 2.8 shows the schema diagram for our university organization. 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.

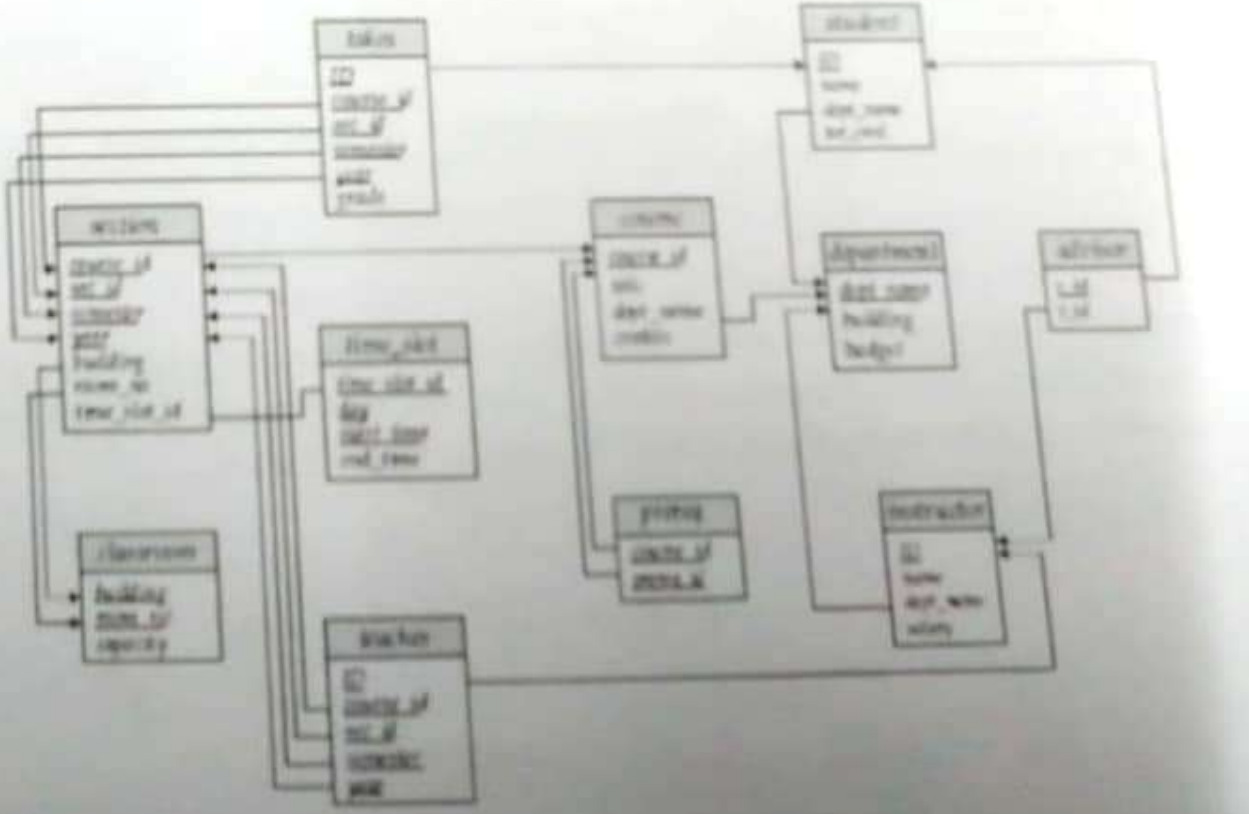


Figure 2.8 Schema diagram for the university database.

Referential integrity constraints other than foreign key constraints are not shown explicitly in schema diagrams. We will study a different diagrammatic representation called the entity-relationship diagram later, in Chapter 7. Entity-relationship diagrams let us represent several kinds of constraints, including independence if they do not depend on the physical schema, and thus need not be rewritten if the physical schema changes.

1.3.3 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 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. Record-based models are so named because the database is structured in fixed-format records of several types. Each table contains records of a particular type. Each record type defines a fixed number of fields, or attributes. The columns of the table correspond to the attributes of the record type.

- 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. The entity-relationship model is widely used in database design, and Chapter 7 explores it in detail.
- 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. Chapter 22 examines the object-relational 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. This is in contrast to the data models mentioned earlier, where every data item of a particular type must have the same set of attributes. The Extensible Markup Language (XML) is widely used to represent semistructured data. Chapter 23 covers it.

Historically, the network data model and the hierarchical data model preceded 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.

4. DATABASE LANGUAGES

A database system provides a **data-definition language** to specify the database schema and **data-manipulation language** to express database queries and updates. In practice, the dat definition and data-manipulation languages are not two separate languages; instead they simpl form parts of a single database language, such as the widely used SQL language.

1.4.1 Data-Manipulation Language

A data-manipulation language (DML) is a language that enables users to access or manipulate data as organized by the appropriate data model. The types of access are:

- Retrieval of information stored in the database
- Insertion of new information into the database
- Deletion of information from the database
- Modification of information stored in the database

There are basically two types:

- Procedural DMLs require a user to specify *what* data are needed and *how* to get those data.
- Declarative DMLs (also referred to as nonprocedural DMLs) require a user to specify *what* data are needed *without* specifying how to get those data.

Declarative DMLs are usually easier to learn and use than are procedural DMLs. However, since a user does not have to specify how to get the data, the database system has to figure out an efficient means of accessing data.

A query is a statement requesting the retrieval of information. The portion of a DML that involves information retrieval is called a query language. Although technically incorrect, it is common practice to use the terms *query language* and *data-manipulation language* synonymously.

1.4.2 Data-Definition Language (DDL)

We specify a database schema by a set of definitions expressed by a special language called a **data-definition language (DDL)**. The DDL is also used to specify additional properties of the data.

We specify the storage structure and access methods used by the database system by a set of statements in a special type of DDL called a data storage and definition language. These statements define the implementation details of the database schemas, which are usually hidden from the users.

The data values stored in the database must satisfy certain consistency constraints. For example, suppose the university requires that the account balance of a department must never be negative. The DDL provides facilities to specify such constraints. The database system checks these constraints every time the database is updated. In general, a constraint can be an arbitrary predicate pertaining to the database. However, arbitrary predicates may be costly to test. Thus, database systems implement integrity constraints that can be tested with minimal overhead:

The I
(staten
dictio
consid
systen
readin

- Domain Constraints. A domain of possible values must be associated with every attribute (for example, integer types, character types, date/time types). Declaring an attribute to be of a particular domain acts as a constraint on the values that it can take. Domain constraints are the most elementary form of integrity constraint. They are tested easily by the system whenever a new data item is entered into the database.

- Referential Integrity. There are cases where we wish to ensure that a value that appears in one relation for a given set of attributes also appears in a certain set of attributes in another relation (referential integrity). For example, the department listed for each course must be one that actually exists. More precisely, the *dept_name* value in a *course* record must appear in the *dept_name* attribute of some record of the *department* relation. Database modifications can cause violations of referential integrity. When a referential-integrity constraint is violated, the normal procedure is to reject the action that caused the violation.

- Assertions. An assertion is any condition that the database must always satisfy. Domain constraints and referential-integrity constraints are special forms of assertions. However, there are many constraints that we cannot express by using only these special forms. For example, "Every department must have at least five courses offered every semester" must be expressed as an assertion. When an assertion is created, the system tests it for validity. If the assertion is valid, then any future modification to the database is allowed only if it does not cause that assertion to be violated.

- Authorization. We may want to differentiate among the users as far as the type of access they are permitted on various data values in the database. These differentiations are expressed in terms of authorization, the most common being: read authorization, which allows reading, but not modification, of data; insert authorization, which allows insertion of new data, but not modification of existing data; update authorization, which allows modification, but not deletion, of data; and delete authorization, which allows deletion of data. We may assign the user all, none, or a combination of these types of authorization.

Tabular Rep.

9. Database Architecture

A **Database Architecture** is a representation of DBMS design. It helps to design, develop, implement, and maintain the database management system. A DBMS architecture allows dividing the database system into individual components that can be independently modified, changed, replaced, and altered. It also helps to understand the components of a database.

ADatabasestores critical information and helps access data quickly and securely. Therefore, selecting the correct Architecture of DBMS helps in easy and efficient data management.

- Types of DBMS Architecture
- 1-Tier Architecture
- 2-Tier Architecture
- 3-Tier Architecture

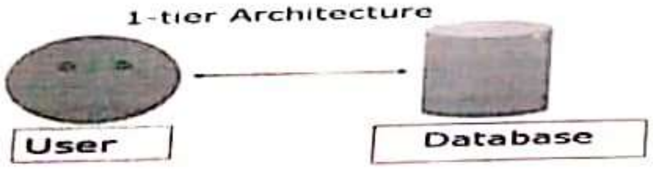
Types of DBMS Architecture

There are mainly three types of DBMS architecture:

- One Tier Architecture (Single Tier Architecture)
- Two Tier Architecture
- Three Tier Architecture

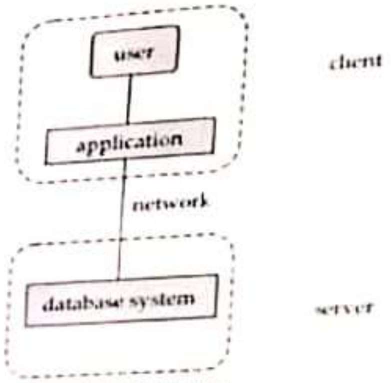
1-Tier Architecture

1 Tier Architecture in DBMS is the simplest architecture of Database in which the client, server, and Database all reside on the same machine. A simple one tier architecture example would be anytime you install a Database in your system and access it to practice SQL queries. But such architecture is rarely used in production.



2-Tier Architecture

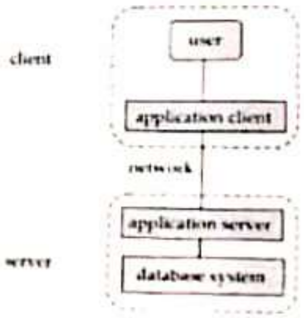
A 2 Tier Architecture in DBMS is a Database architecture where the presentation layer runs on a client (PC, Mobile, Tablet, etc.), and data is stored on a server called the second tier. Two tier architecture provides added security to the DBMS as it is not exposed to the end-user directly. It also provides direct and faster communication



(a) Two-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.



(b) Three-tier architecture