

Unit 1

Introduction

1.1 Database Management System purpose and applications

Data: Collection of numbers, characters/ Data are values of qualitative or quantitative variables, belonging to a set of items. Talking in terms of computing, data is basically information that can be translated into a particular form for efficient movement and processing.

Example: Name, age, weight, height, etc.

Database: A database is an organized collection of data. The data are typically organized to model relevant aspects of reality in a way that supports processing requiring this information. **Example:** Library, dictionary

The library contains a huge collection of books of different genres, here the library is database and books are the data.

Database Management System: A Database Management System is a collection of interrelated data and a set of programs to access those data. The main purpose of DBMS is to provide a way to store and retrieve database information that is both convenient and efficient. Management of data involves both defining structures for storage of information and providing mechanisms for the manipulation of information. In addition, the database system must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access. **Example:** Oracle, SQL –Server, MySQL, MS Access, Sybase etc.

Database Management Systems allows users to do the following:

- ❖ **Define Data** – Allows the users to create, modify and delete the definitions which define the organization of the database.
- ❖ **Update Data** – Provides access to the users to insert, modify and delete data from the database.
- ❖ **Retrieve Data** – Allows the users to retrieve data from the database based on the requirement.
- ❖ **Administration of users** – Registers the users and monitors their action, enforces data security, maintains data integrity, monitors performance and deals with concurrency control.

Application Areas of Database System

Some of the most common application of database management system is:

- **Airlines and railways:** Airlines and railways use online databases for reservation, and for displaying the schedule information.
- **Banking:** Banks use databases for customer inquiry, accounts, loans, and other transactions.
- **Education:** Schools, colleges and universities use databases for course registration, result, and other information.
- **Telecommunications:** Telecommunication departments use databases to store information about the communication network, telephone numbers, record of calls, for generating monthly bills, etc.
- **Credit card transactions:** Databases are used for keeping track of purchases on credit cards in order to generate monthly statements.
- **E-commerce:** Integration of heterogeneous information sources for business activity such as online shopping, booking of holiday package, consulting a doctor, etc.
- **Health care information systems and electronic patient record:** Databases are used for maintaining the patient health care details.
- **Digital libraries and digital publishing:** Databases are used for management and delivery of large bodies of textual and multimedia data.
- **Finance:** Databases are used for storing information such as sales, purchases of stocks and bonds or data useful for online trading.
- **Sales:** Databases are used to store product, customer and transaction details.
- **Human resources:** Organizations use databases for storing information about their employees, salaries, benefits, taxes, and for generating salary checks.

Objectives of DBMS:

The objectives that the management should keep in mind when they design and organize their data base management systems are:

- Provide for mass storage of relevant data,
- Make access to the data easy for the user,
- Provide prompt response to user requests for data,
- Make the latest modifications to the database available immediately,
- Eliminate redundant data,
- Allow for multiple users to be active at one time,
- Allow for growth in the database system,
- Protect the data from physical harm and unauthorized access.

Needs of DBMS:

- Processing Queries and Object Management
- Controlling redundancy and inconsistency
- Efficient memory management and indexing
- Concurrency control and transaction management
- Access Control and ease in accessing data
- Integrity constraints
- Data security
- Data scalability, expandability and flexibility

1.2 Database System vs. File System

File system is a method of organizing the files with a hard disk or other medium of storage. File system arranges the files and helps in retrieving the files, when required. It is compatible with different file types, such as mp3, doc, txt, mp4, etc and these are also grouped into directories. It also influences the method of writing and reading data to the hard disk. Examples are NTFS or the New Technology File System and EXT, the Extended File System.

DBMS, meanwhile, is the acronym for Database Management System. It is also software used to store and regain user's data, while also maintaining the required security measures. This includes a group of programs that can help to manipulate the database. In bigger systems, DBMS helps the users as well as third party software to store and recover the data. Examples are MySQL, MS SQL Server, and Oracle and so on.

FILE SYSTEM	DBMS
Used to manage and organise the files stored in the hard disk of the computer	A software to store and retrieve the user's data
Redundant data is present	No presence of redundant data
Query processing is not so efficient	Query processing is efficient
Data consistency is low	Due to the process of normalisation, the data consistency is high

Less complex, does not support complicated transactions	More complexity in managing the data, easier to implement complicated transactions
Less security	Supports more security mechanisms
Less expensive in comparison to DBMS	Higher cost than the File system
Does not support crash recovery	Crash recovery mechanism is highly supported

1.3 View of Data

- **Data Abstraction (Physical, logical and view level)**
- **Data Independence**

Data Abstraction

Database systems comprise of complex data-structures. In order to make the system efficient in terms of retrieval of data, and reduce complexity in terms of usability of users, developers use abstraction i.e. hides irrelevant details from the users. This approach simplifies database design.

A major purpose of a database system is to provide users with an abstract view of the data i.e. the system hides certain details of how the data are stored and maintained. It gives architecture to separate the user applications and the physical database. It defines in the following three levels:

- ❖ **Physical level:** This is the lowest level of data abstraction. It describes how data is actually stored in database. You can get the complex data structure details at this level. Actually, it is decided by developers or database application programmers how to store the data in the database. For example, customer's information is stored in tables and data is stored in the form of blocks of storage such as bytes, gigabytes etc.
- ❖ **Logical level:** Logical levels hide the detail of physical storage and describe about the entities data and constraints. It is the middle level and it is used by the application programmer it describes what data are stored in the database and what relationship among those data. All logical tasks

are performed at this level. It tries to describe the entire or whole data because it describes what tables to be created and what are the links among those tables that are created. It is less complex than the physical level. Logical level is used by developers or database administrators (DBA). So, overall, the logical level contains tables (fields and attributes) and relationships among table attributes.

- ❖ **View level:** The highest level of abstraction describes only part of the entire 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. View level can be used by all users (all levels' users). This level is the least complex and easy to understand. For example, a user can interact with a system using GUI that is view level and can enter details at GUI or screen and the user does not know how data is stored and what data is stored, this detail is hidden from the user.

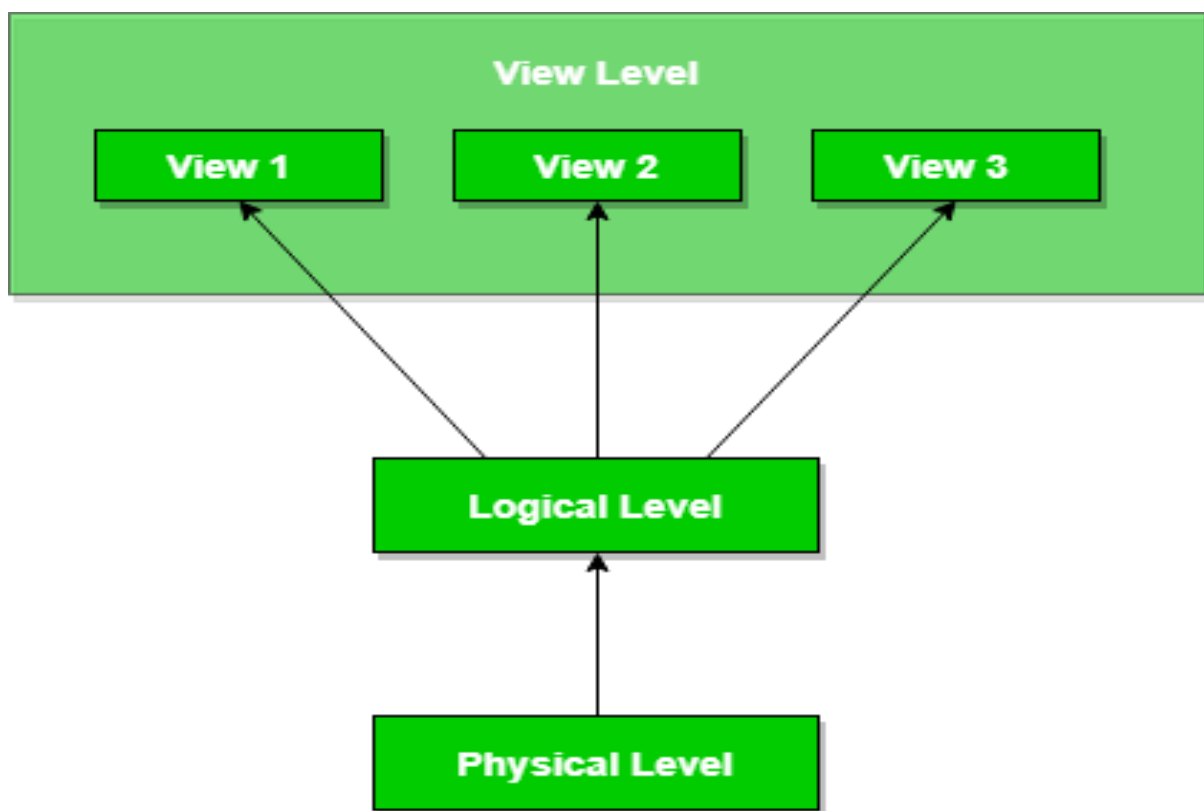


Fig: Three levels of Data Abstraction

Data Independence

The main purpose of data abstraction is achieving data independence in order to save time and cost required when the database is modified or altered. Data independency can be defined as the capacity to change the schema at one level of a database system without having to change schema at the next higher level.

Both data abstraction and data independence hide implementation details from users. This allows users to focus on the general structure rather than low-level implementation.

If the database changes and expands over time, it is very important that the changes in one level should not affect the data at other levels of the database. This would save time and cost required when changing the database.

There are mainly two types of Data Independence:

- Physical Data Independence
- Logical Data Independence

Physical Data Independence

Physical Data Independence means changing the physical level without affecting the logical level or conceptual level. Using this property, we can change the storage device of the database without affecting the logical schema.

The changes in the physical level may include changes using the following –

- A new storage device like magnetic tape, hard disk, etc.
- A new data structure for storage.
- A different data access method or using an alternative files organization technique.
- Changing the location of the database.

Logical Data Independence

Logical view of data is the user view of the data. It presents data in the form that can be accessed by the end users. Logical Data Independence says that users should be able to manipulate the Logical View of data without any information of its physical storage. Software or the computer program is used to manipulate the logical view of the data.

The data independence provides the database in simple structure. It is based on application domain entities to provide the functional requirement. It provides abstraction of system functional requirements. Static structure for the logical view is defined in the class object diagrams. Users cannot manipulate the logical structure of the database.

The changes in the logical level may include –

- Change the data definition.
- Adding, deleting, or updating any new attribute, entity or relationship in the database.

1.4 Instances and Schemas

Databases change over time as information is inserted and deleted. 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. Schemas are changed infrequently, if at all.

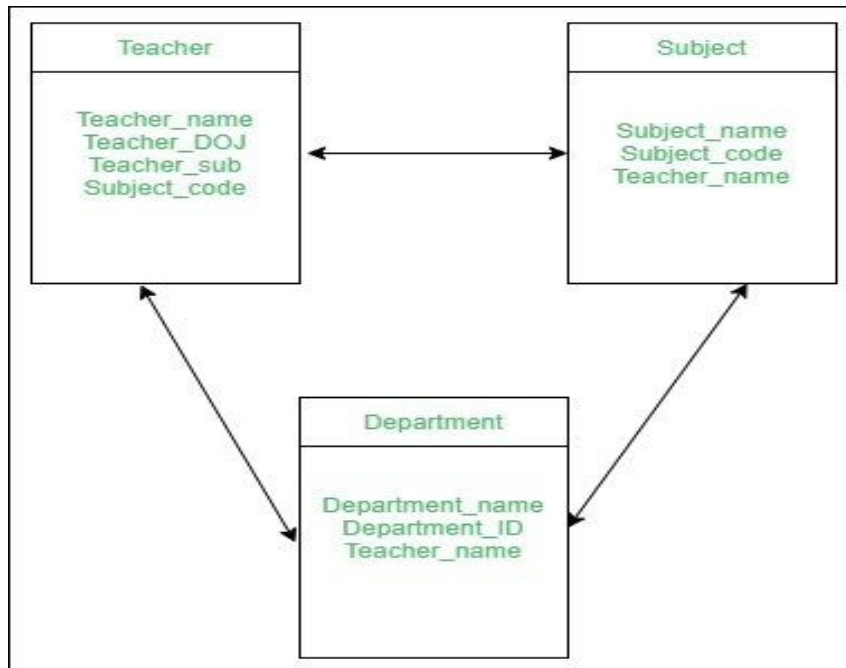
The concept of database schemas and instances can be understood by analogy to a program written in a programming language. A database schema corresponds to the variable declarations (along with associated type definitions) in a program. Each variable has a particular value at a given instant. The values 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 subschemas that describe different views of the database. Of these, the logical schema is by far the most important, in terms of its effect on application programs, since programmers construct applications by using the logical schema.

The physical schema is hidden beneath the logical schema, and can usually be changed easily without affecting application programs. Application programs are said to exhibit physical data independence if they do not depend on the physical schema, and thus need not be rewritten if the physical schema changes.

We have a schema that shows the relationship between three tables: **Teacher**, **Subject** and **Department**. The diagram only shows the design of the database, it doesn't show the data present in those tables. Schema is only a structural view(design) of a database as shown in the diagram.



❖ **Physical Schema:**

- design of a database at physical level
- how the data stored in blocks of storage is described

❖ **Logical Schema:**

- design of database at logical level
- programmers and database administrators work at this level
- at this level data can be described as certain types of data records gets stored in data structures

❖ **View Schema:**

- design of database at view level
- generally describes end user interaction with database systems

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

For example, let's say we have a single table student in the database, today the table has 100 records, and so today the instance of the database has 100 records. Let's say we are going to add another 100 records in this table by tomorrow so the instance of database tomorrow will have 200 records in table.

In short, at a particular moment the data stored in database is called the instance that changes over time when we add or delete data from the database.

Difference between Schema and Instance :

SCHEMA	INSTANCE
It is the overall description of the database.	It is the collection of information stored in a database at a particular moment.
Schema is same for whole database.	Data in instances can be changed using addition, deletion, updation.
Does not change Frequently.	Changes Frequently.
Defines the basic structure of the database i.e how the data will be stored in the database.	It is the set of Information stored at a particular time.

1.5 Database Languages (DDL, DML and DCL)

Database languages, also known as query languages or data query languages, are a classification of programming languages that developers use to define and access databases, which are collections of organized data that users can access electronically. These languages allow users to complete tasks such as controlling access to data, defining and updating data and searching for information within the database management system (DBMS).

a. Data definition language (DDL)

Data definition language (DDL) creates the framework of the database by specifying the database schema, which is the structure that represents the organization of data. Its common uses include the creation and alteration of tables, files, indexes and columns within the database. This language also allows users to rename or drop the existing database or its components. Here's a list of DDL statements:

- ❖ CREATE – is used to create the database or its objects (like table, index, function, views, store procedure and triggers).
- ❖ DROP – is used to delete objects from the database.
- ❖ ALTER–is used to alter the structure of the database.
- ❖ TRUNCATE–is used to remove all records from a table, including all spaces allocated for the records are removed.
- ❖ COMMENT –is used to add comments to the data dictionary.
- ❖ RENAME –is used to rename an object existing in the database.

Example:

Create table account (account-number char (10), balance integer);

b. Data manipulation language (DML)

Data manipulation language (DML) provides operations that handle user requests, offering a way to access and manipulate the data that users store within a database. Its common functions include inserting, updating and retrieving data from the database. Here's a list of DML statements:

- ❖ INSERT – is used to insert data into a table.
- ❖ SELECT – is used to retrieve data from a table.
- ❖ UPDATE – is used to update existing data within a table.
- ❖ DELETE – is used to delete records from a database table.

Example:

```
select customer.customer_name from customer where customer.customer-id = 192;
```

c. Data control language (DCL)

Data control language (DCL) controls access to the data that users store within a database. Essentially, this language controls the rights and permissions of the database system. It allows users to grant or revoke privileges to the database. Here's a list of DCL statements:

- ❖ GRANT: Gives a user access to the database
- ❖ REVOKE: Removes a user's access to the database

d. Transaction control language (TCL)

Transaction control language (TCL) manages the transactions within a database. Transactions group a set of related tasks into a single, executable task. All the tasks must succeed in order for the transaction to work. Here's a list of TCL statements:

- ❖ COMMIT– commits a Transaction.
- ❖ ROLLBACK– rolls back a transaction in case of any error occurs.
- ❖ SAVEPOINT– sets a savepoint within a transaction.

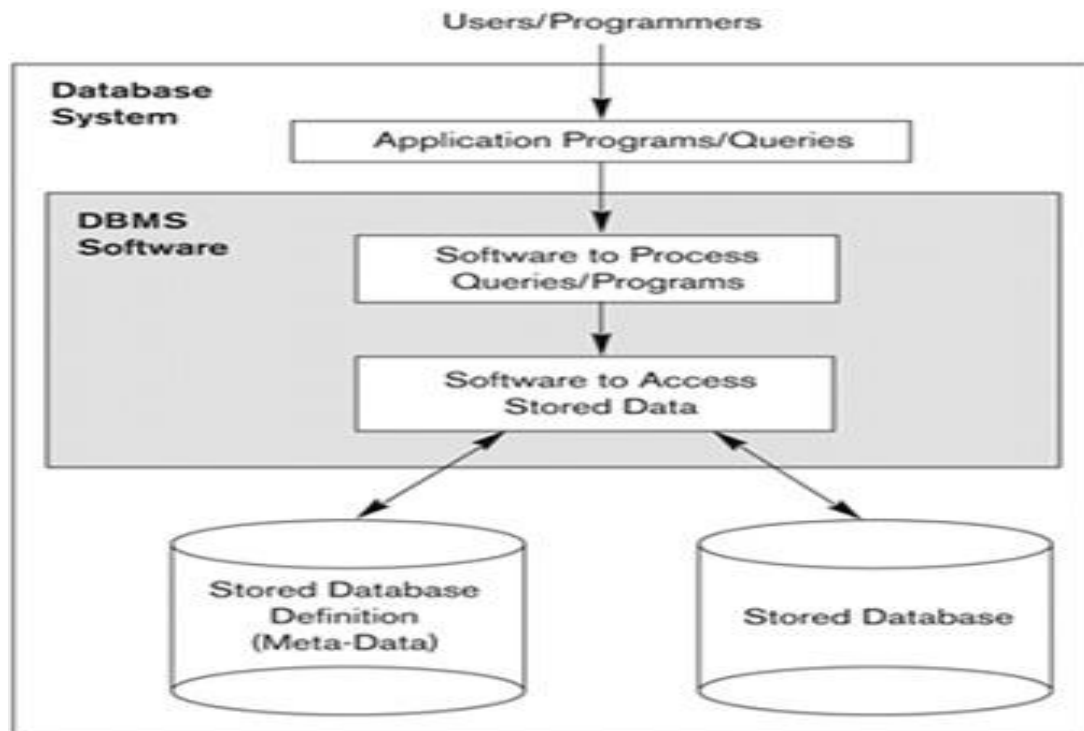
1.6 Database and Application Architecture

- **Database System Architecture**
- **Database Application Architecture (two-tier and three-tier)**

Database System Architecture

DBMS (database management system) is a collection of programs that enables users to create and maintain database. The DBMS is a common purpose software system that facilitates the process of constructing, defining, manipulating and sharing databases among various users as well as applications. Defining a database state the database involves specifying the constraints, data types and structures of the data to be stored in the database. The descriptive information is as well stored in the database in the form database catalogue or dictionary- it is called meta-data.

Manipulating the data comprises the querying the database to retrieve the specific data. An application program accesses the database through transferring the queries or requests for data to DBMS. The significant function provided by the DBMS includes protecting the database and maintain the database.



Database Application Architecture (two-tier and three-tier)

DBMS Architecture

- The DBMS design depends upon its architecture. The basic client/server architecture is used to deal with a large number of PCs, web servers, database servers and other components that are connected with networks.
- The client/server architecture consists of many PCs and a workstation which are connected via the network.
- DBMS architecture depends upon how users are connected to the database to get their request done.

Types of DBMS Architecture

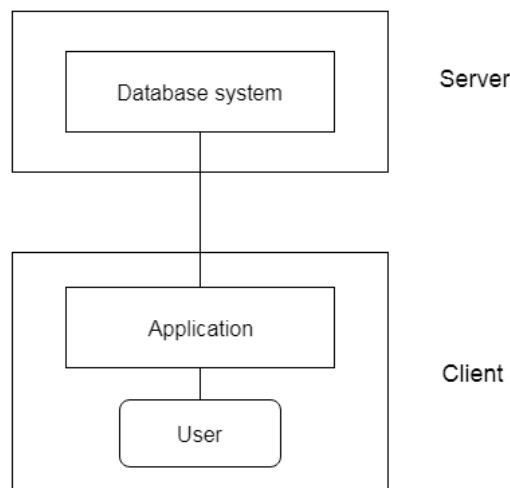
Database architecture can be seen as a single tier or multi-tier. But logically, database architecture is of two types like: **2-tier architecture** and **3-tier architecture**.

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.

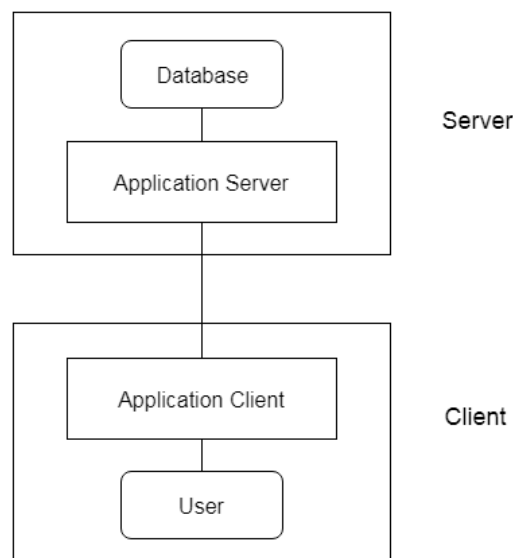
2-Tier Architecture

- 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.



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.



1. Presentation layer (your PC, Tablet, Mobile, etc.)
2. Application layer (server)
3. Database Server

What is three-tier architecture?

Three-tier architecture is a well-established software application architecture that organizes applications into three logical and physical computing tiers: the presentation tier, or user interface; the application tier, where data is processed; and the data tier, where the data associated with the application is stored and managed.

The chief benefit of three-tier architecture is that because each tier runs on its own infrastructure, each tier can be developed simultaneously by a separate development team, and can be updated or scaled as needed without impacting the other tiers.

For decades three-tier architecture was the prevailing architecture for client-server applications. Today, most three-tier applications are targets for [modernization](#), using [cloud-native](#) technologies such as [containers](#) and [microservices](#), and for [migration](#) to the cloud.

The three tiers in detail

Presentation tier

The presentation tier is the user interface and communication layer of the application, where the end user interacts with the application. Its main purpose is to display information to and collect information from the user. This top-level tier can run on a web browser, as desktop application, or a graphical user interface (GUI), for example. Web presentation tiers are usually developed using HTML, CSS and JavaScript. Desktop applications can be written in a variety of languages depending on the platform.

Application tier

The application tier, also known as the logic tier or middle tier, is the heart of the application. In this tier, information collected in the presentation tier is processed - sometimes against other information in the data tier - using business logic, a specific set of business rules. The application tier can also add, delete or modify data in the data tier.

The application tier is typically developed using Python, Java, Perl, PHP or Ruby, and communicates with the data tier using API calls.

Data tier

The data tier, sometimes called database tier, data access tier or back-end, is where the information processed by the application is stored and managed. This can be a relational database management system such as PostgreSQL, MySQL, MariaDB, Oracle, DB2, Informix or Microsoft SQL Server, or in a NoSQL Database server such as Cassandra, CouchDB or MongoDB.

In a three-tier application, all communication goes through the application tier. The presentation tier and the data tier cannot communicate directly with one another.