Introduction to Database Concepts

Introduction

In computerized information system data is the basic resource of the organization. So, proper organization and management for data is required for organization to run smoothly.

Database management system deals the knowledge of how data is stored and managed on a computerized information system. In any organization, it requires accurate and reliable data for better decision making, ensuring privacy of data and controlling data efficiently.

The examples include deposit and/or withdrawal from a bank, hotel, airline or railway reservation, purchase items from supermarkets in all cases, a database is accessed.

What is data?

Data is the known facts or figures that have implicit meaning. It can also be defined as it is the representation of facts, concepts or instruction in a formal manner, which is suitable for understanding and processing. Data can be represented in alphabets (A-Z, a-z), in digits (0-9) and using special characters (+,-.#,$, etc) e.g.: 25, “ajit” etc.

a) Introduction to Data Processing

There are various ways to process a data

- **Stage 0**) Manual Information System
  - Records
  - Files
  - Index Cards
- **Stage 1**) Sequential Information System
  - Tapes
  - Files
  - Slow, on-interactive
- **Stage 2**)File Based Information System
  - Disk
  - Application Program has its own file ==> Data Dependence
  - Data Redundancy
- **Stage 3**) DBMS Information System
  - Generalized Data Management Software
  - Transaction Processing

Example of Manual Information System (Stage 0):

In earlier days people used to maintain registers for maintaining records which we called as Manual Information System
For example customers
AAA
BBB
YYY
WER
WWW, etc

Suppose any change in records of WER, and then it was a problem to find the name and do the changes if the list is long and cannot be maintained in file systems.

Learning from File system issues to find name, details etc a new system, was derived in which data had data arranged in sequential manner

Example of Sequential

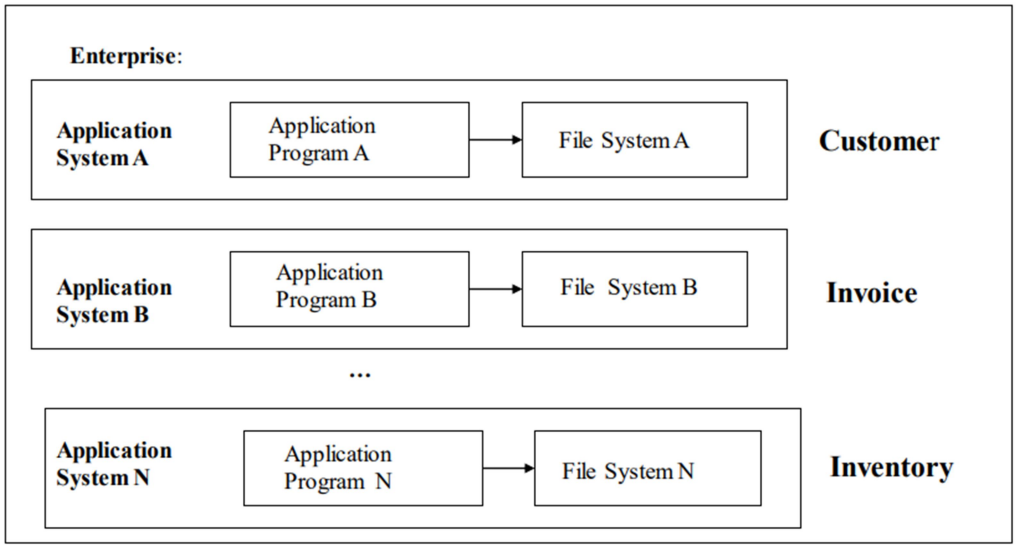
Learning from Manual system, people invented a new way and started storing information in sequence like
All details of customers starting with A on 1 Page/Book
All details of customers starting with B in page in different page
..
…
….
Similarly till all details of customers starting with Z in page in different page

But now this method also had a problem
Let us discuss what problem this method had!!

b) Overview of File Systems

The earliest business computer systems were used to process business records and produce information. They were generally faster and more accurate than equivalent manual systems.

These systems stored groups of records in separate files, and so they were called file processing systems. In a typical file processing systems, each department has its own files, designed specifically for those applications. The department itself working with the data processing staff, sets policies or standards for the format and maintenance of its files.



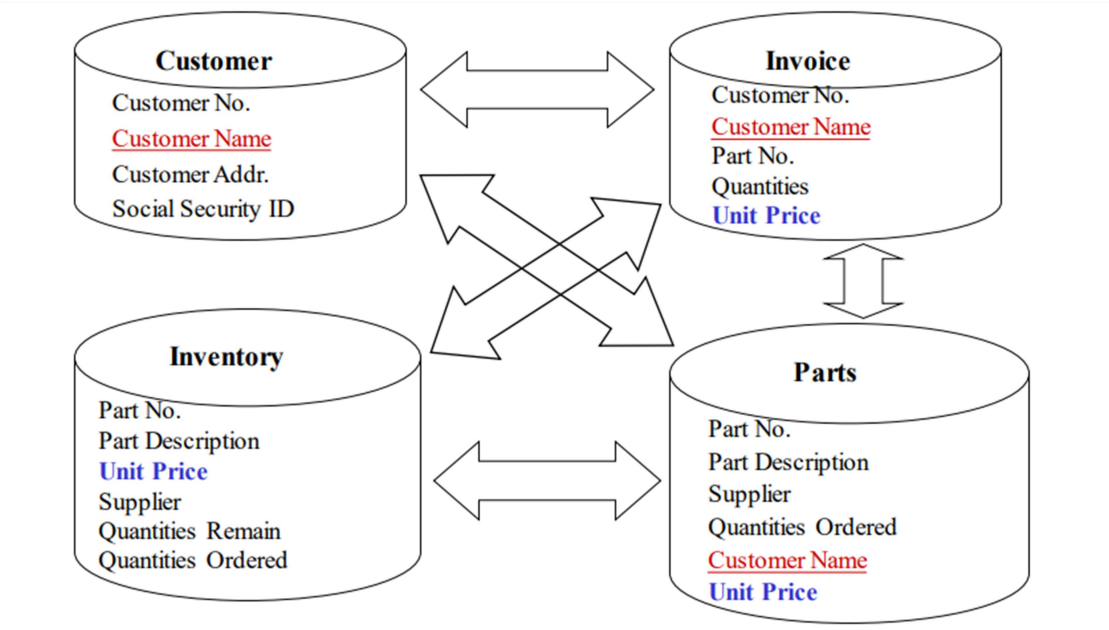
Programs are dependent on the files and vice-versa; that is, when the physical format of the file is changed; the program has also to be changed. Although the traditional file oriented approach to information processing is still widely used, it does have some very important disadvantages.

Consider part of a savings-bank enterprise that keeps information about all customers and savings accounts. One way to keep the information on a computer is to store it in operating system files. To allow users to manipulate the information, the system has a number of application programs that manipulate the files, including

A program to debit or credit an account
A program to add a new account
A program to find the balance of an account
A program to generate monthly statements

System programmers wrote these application programs to meet the needs of the bank. New application programs are added to the system as the need arises.

For example,
Suppose that the savings bank decides to offer checking accounts. As a result, the bank creates new permanent files that contain information about all the checking accounts maintained in the bank, and it may have to write new application programs to deal with situations that do not arise in savings accounts, such as overdrafts. Thus, as time goes by, the system acquires more files and more application programs.



This typical file-processing system is supported by a conventional operating system. The system stores permanent records in various files, and it needs different application programs to extract records from, and add records to, the appropriate files. Before Database management systems (DBMSs) came along, organizations usually stored information in such systems.

c) Drawback of File Systems

Keeping organizational information in a file-processing system has a number of major disadvantages:

- **Data redundancy and inconsistency**.

Since different programmers create the files and application programs over a long period, the various files are likely to have different formats and the programs may be written in several programming languages. Moreover, the same information may be duplicated in several places (files). For example, the address and telephone number of a particular customer may appear in a file that consists of savings-account records and in a file that consists of checking-account records. This redundancy leads to higher storage and access cost. In addition, it may lead to data inconsistency; that is, the various copies of the same data may no longer agree.

For example, a changed customer address may be reflected in savings-account records but not elsewhere in the system.

- **Difficulty in accessing data**

Suppose that one of the bank officers needs to find out the names of all customers who live within a particular postal-code area. The officer asks the data-processing department to generate such a list.

Because the designers of the original system did not anticipate this request, there is no application program on hand to meet it.

There is, however, an application program to generate the list of all customers. The bank officer has now two Choices:

- Either obtains the list of all customers and extract the needed information manually

Or

- Ask a system programmer to write the necessary application program.

Both alternatives are obviously unsatisfactory. Suppose that such a program is written, and After that, several days later, the same officer needs to trim that list to include only those customers who have an account balance of Rupees 10,000 or more.

As expected, a program to generate such a list does not exist. Again, the officer has the preceding two options, neither of which is satisfactory.

The point here is that conventional file-processing environments do not allow needed data to be retrieved in a convenient and efficient manner. More responsive data-retrieval Systems are required for general use.

- **Data isolation**.

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

- **Integrity problems**.

The data values stored in the database must satisfy certain types of consistency constraints. For example, the balance of a bank account may never fall below a prescribed amount (say, 3000 Rs). Developers enforce these constraints in the system by adding appropriate code in the various application programs. However, when new constraints are added, it is difficult to change the programs to enforce them. The problem is compounded when constraints involve several data items from different files.

d) Purpose of database system

In early days database system was made up on the top of file systems with below intention

- To overcome drawback of file system to store data
- To make the information stored securely and made available online without the fear of losing it over time.

Some of the main purposes of database systems are:

Less prone to issues:

Information stored on databases are less prone to disasters as compared to information on paper files. With the advent of cloud computing, now your data is stored and replicated over multiple servers in disparate locations, which would essentially make it invulnerable to any natural or human made disaster.

Scalability:

You can create a custom database for your business and store information in it as your business grows. You can even migrate this information to another bigger database when you feel the current database can't handle your information. You may even change the way information is stored on your database to keep up with the ever changing business needs. Such type of scalability is not possible for manual files.

Data Security:

Another purpose of using a database system is data security. You can put security on your data which can make it unavailable to unauthorized people. There are numerous techniques to do that. Some database systems even provide you multiple layers of security to keep your data off from the hands of an unauthorized person. From a simple password on your excel file to the advanced data encryption techniques available in Oracle and other databases, this is definitely a feature which will convince you to switch over here from manual files.

Computing capabilities:

Modern database systems are not just the places where you just store information. They also offer advanced computing facilities where you use inbuilt calculations and aggregations. Such things come in handy when you need to do data mining or have to generate last minute reports on your data.

Easy to maintain:

Databases are easy to maintain as compared to traditional ways of storing information/manual files. The housekeeping and maintenance cost is almost 5-10% of the same required in keeping and maintaining a manual files system. Faster data access also gives it an edge over traditional systems.

e) Concept of a database

The term **'database'** is defined as any collection of electronic records that can be processed to produce useful information.

The data can be accessed, modified, managed, controlled and organized to perform various data-processing operations. The data is typically indexed across rows, columns and tables that make workload processing and data querying efficient. Different types of databases include: object-oriented, relational, distributed, hierarchical, network, and others.

In enterprise applications, databases involve mission-critical, security-sensitive, and compliance-focused record items that have complicated logical relationships with other datasets and grow exponentially over time as the user base increases. As a result, these organizations require technology solutions to maintain, secure, manage, and process the data stored in databases. This is where Database Management System come into play.

What is 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.
DBMS allows users to create their own databases as per their requirement. The term "DBMS" includes the user of the database and other application programs. It provides an interface between the data and the software application.

History of DBMS

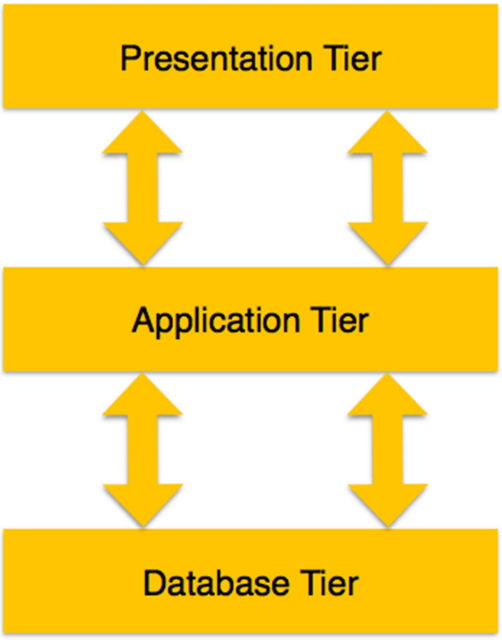
Here, are the important landmarks from the history:

- 1960 - Charles Bachman designed first DBMS system
- 1970 - Codd introduced IBM'S Information Management System (IMS)
- 1976- Peter Chen coined and defined the Entity-relationship model also know as the ER model
- 1980 - Relational Model becomes a widely accepted database component
- 1985- Object-oriented DBMS develops.
- 1990s- Incorporation of object-orientation in relational DBMS.
- 1991- Microsoft ships MS access, a personal DBMS and that displaces all other personal DBMS products.
- 1995: First Internet database applications
- 1997: XML applied to database processing. Many vendors begin to integrate XML into DBMS products.

Characteristics of Database Management System

- Provides security and removes redundancy
- Self-describing nature of a database system
- Insulation between programs and data abstraction
- Support of multiple views of the data
- Sharing of data and multiuser transaction processing
- DBMS allows entities and relations among them to form tables.
- It follows the ACID concept (Atomicity, Consistency, Isolation, and Durability).
- DBMS supports multi-user environment that allows users to access and manipulate data in parallel.

Database Architecture



The design of a DBMS depends on its architecture. It can be centralized or decentralized or hierarchical. The architecture of a DBMS can be seen as either single tier or multi-tier. An n-tier architecture divides the whole system into related but independent **n** modules, which can be independently modified, altered, changed, or replaced.

In 1-tier architecture, the DBMS is the only entity where the user directly sits on the DBMS and uses it. Any changes done here will directly be done on the DBMS itself. It does not provide handy tools for end-users. Database designers and programmers normally prefer to use single-tier architecture.

If the architecture of DBMS is 2-tier, then it must have an application through which the DBMS can be accessed. Programmers use 2-tier architecture where they access the DBMS by means of an application. Here the application tier is entirely independent of the database in terms of operation, design, and programming.

3-tier Architecture

A 3-tier architecture separates its tiers from each other based on the complexity of the users and how they use the data present in the database. It is the most widely used architecture to design a DBMS.

- **Database (Data) Tier** − At this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.
- **Application (Middle) Tier** − At this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End-users are unaware of any existence of the database beyond the application. At the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.
- **User (Presentation) Tier** − End-users operate on this tier and they know nothing about any existence of the database beyond this layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.

Multiple-tier database architecture is highly modifiable, as almost all its components are independent and can be changed independently.

f) Database system vs file system

DBMS	File Management System
Multi-user access	It does not support multi-user access
Design to fulfil the need for small and large businesses	It is only limited to smaller DBMS system.
Remove redundancy and Integrity	Redundancy and Integrity issues
Expensive. But in the long term Total Cost of Ownership is cheap	It's cheaper
Easy to implement complicated transactions	No support for complicated transactions

g) View of data

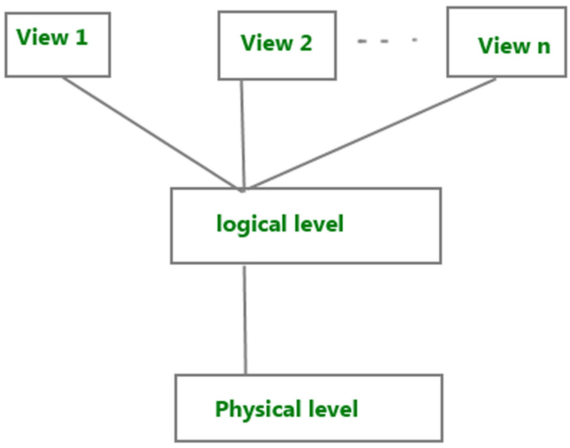
Abstraction is one of the main features of database systems. Hiding irrelevant details from user and providing abstract view of data to users, helps in easy and efficient user-database interaction.

We discussed the three levels of DBMS architecture earlier, The top level of that architecture is "view level". The view level provides the "view of data" to the users and hides the irrelevant details such as data relationship, database schema, constraints, security etc from the user.

To fully understand the view of data, basic knowledge of **data abstraction** and **instance & schema** is required.

Data Abstraction

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



Three Levels of data abstraction

We have three levels of abstraction:

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

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

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

Example:

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

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

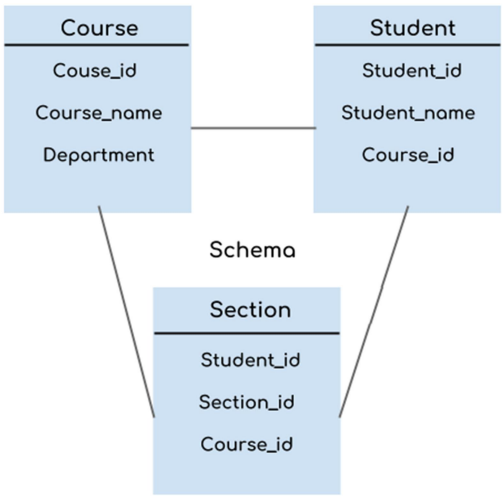
At view level, user just interact with system with the help of GUI and enter the details at the screen, they are not aware of how the data is stored and what data is stored; such details are hidden from them.

Instance and schema in DBMS

DBMS Schema

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

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



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

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

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

h) Data models

A Database model defines the logical design and structure of a database and defines how data will be stored, accessed and updated in a database management system. While the Relational Model is the most widely used database model, there are other models too:

- Hierarchical Model
- Network Model
- Entity-relationship Model
- Relational Model

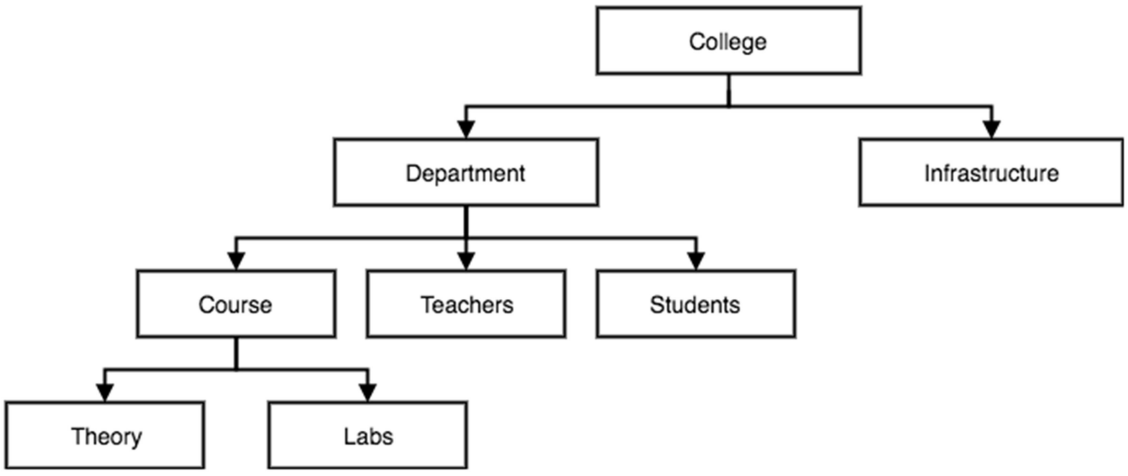
Hierarchical Model

This database model organises data into a tree-like-structure, with a single root, to which all the other data is linked. The heirarchy starts from the Root data, and expands like a tree, adding child nodes to the parent nodes.

In this model, a child node will only have a single parent node.

This model efficiently describes many real-world relationships like index of a book, recipes etc.

In hierarchical model, data is organised into tree-like structure with one one-to-many relationship between two different types of data, for example, one department can have many courses, many professors and of-course many students.

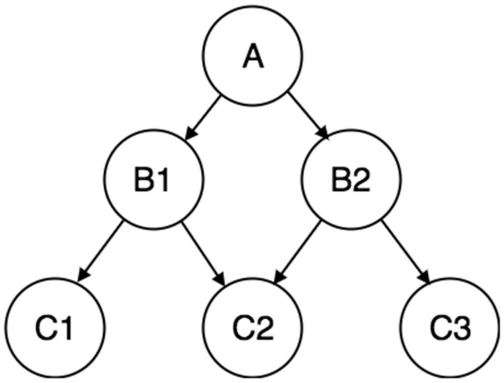


Network Model

This is an extension of the Hierarchical model. In this model data is organised more like a graph, and are allowed to have more than one parent node.

In this database model data is more related as more relationships are established in this database model. Also, as the data is more related, hence accessing the data is also easier and fast. This database model was used to map many-to-many data relationships.

This was the most widely used database model, before Relational Model was introduced.



Entity-relationship Model

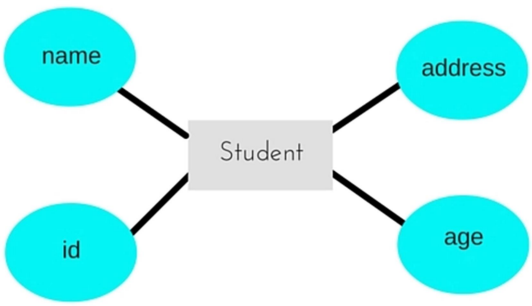
In this database model, relationships are created by dividing object of interest into entity and its characteristics into attributes.

Different entities are related using relationships.

E-R Models are defined to represent the relationships into pictorial form to make it easier for different stakeholders to understand.

This model is good to design a database, which can then be turned into tables in relational model(explained below).

Let's take an example, If we have to design a School Database, then Student will be an entity with attributes name, age, address etc. As Address is generally complex, it can be another entity with attributes street name, pincode, city etc, and there will be a relationship between them.



Relational Model

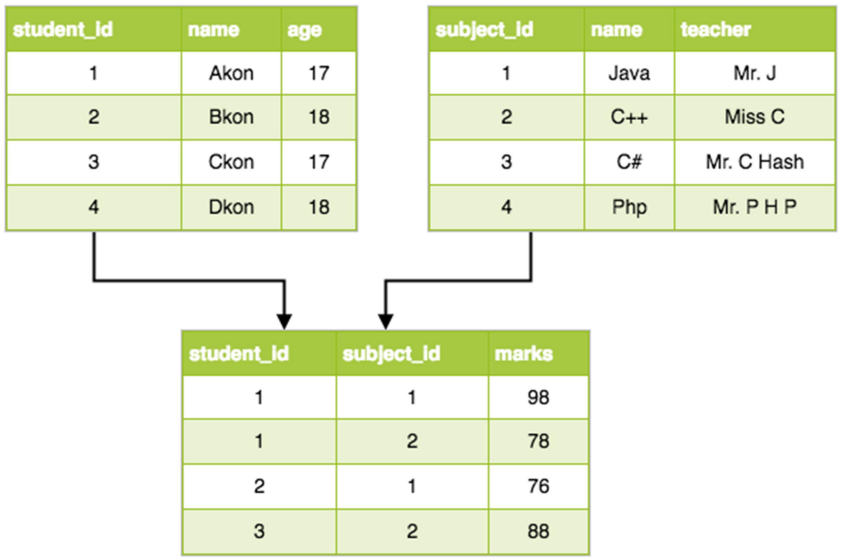
In this model, data is organised in two-dimensional tables and the relationship is maintained by storing a common field.

This model was introduced by E.F Codd in 1970, and since then it has been the most widely used database model, infact, we can say the only database model used around the world.

The basic structure of data in the relational model is tables. All the information related to a particular type is stored in rows of that table.

Hence, tables are also known as relations in relational model.

We will learn how to design tables, normalize them to reduce data redundancy and how to use Structured Query language to access data from tables.



student_id	name	age
1	Akon	17
2	Bkon	18
3	Ckon	17
4	Dkon	18

subject_id	name	teacher
1	Java	Mr. J
2	C++	Miss C
3	C#	Mr. C Hash
4	Php	Mr. P H P

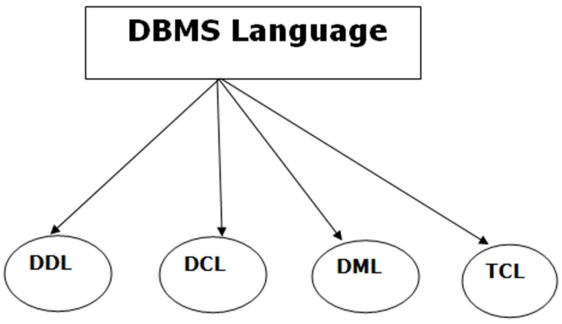
student_id	subject_id	marks
1	1	98
1	2	78
2	1	76
3	2	88

5555

i) Database Language

A DBMS has appropriate languages and interfaces to express database queries and updates.

Database languages can be used to read, store and update the data in the database.



- **Data Definition Language**
  - **DDL** stands for **D**ata **D**efinition **L**anguage. It is used to define database structure or pattern.
  - It is used to create schema, tables, indexes, constraints, etc. in the database.
  - Using the DDL statements, you can create the skeleton of the database.
  - Data definition language is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.

Here are some tasks that come under DDL:

- **Create:** It is used to create objects in the database.
- **Alter:** It is used to alter the structure of the database.
- **Drop:** It is used to delete objects from the database.
- **Truncate:** It is used to remove all records from a table.
- **Rename:** It is used to rename an object.
- **Comment:** It is used to comment on the data dictionary.

These commands are used to update the database schema that's why they come under Data definition language.

- **Data Manipulation Language**

DML stands for **D**ata **M**anipulation **L**anguage. It is used for accessing and manipulating data in a database. It handles user requests.

Here are some tasks that come under DML:

- **Select:** It is used to retrieve data from a database.
- **Insert:** It is used to insert data into a table.
- **Update:** It is used to update existing data within a table.
- **Delete:** It is used to delete all records from a table.
- **Merge:** It performs UPSERT operation, i.e., insert or update operations.
- **Call:** It is used to call a structured query language or a Java subprogram.
- **Explain Plan:** It has the parameter of explaining data.
- **Lock Table:** It controls concurrency.

- **Data Control Language**
  - **DCL** stands for **D**ata **C**ontrol **L**anguage. It is used to retrieve the stored or saved data.
  - The DCL execution is transactional. It also has rollback parameters.

    (But in Oracle database, the execution of data control language does not have the feature of rolling back.)

Here are some tasks that come under DCL:

- **Grant:** It is used to give user access privileges to a database.
- **Revoke:** It is used to take back permissions from the user.

There are the following operations which have the authorization of Revoke:

CONNECT, INSERT, USAGE, EXECUTE, DELETE, UPDATE and SELECT.

- **Transaction Control Language**

TCL is used to run the changes made by the DML statement. TCL can be grouped into a logical transaction.

Here are some tasks that come under TCL:

- **Commit:** It is used to save the transaction on the database.
- **Rollback:** It is used to restore the database to original since the last Commit.

j) Database users and administrator

Database Users:

Users are differentiated by the way they expect to interact with the system:

- ***Application programmers*:**
  - Application programmers are computer professionals who write application programs. Application programmers can choose from many tools to develop user interfaces.
  - Rapid application development (RAD) tools are tools that enable an application programmer to construct forms and reports without writing a program.
- ***Sophisticated users*:**
  - Sophisticated users interact with the system without writing programs. Instead, they form their requests in a database query language.
  - They submit each such query to a query processor, whose function is to break down DML statements into instructions that the storage manager understands.
- ***Specialized users* :**
  - Specialized users are sophisticated users who write specialized database applications that do not fit into the traditional data-processing framework.

  - Among these applications are computer-aided design systems, knowledge base and expert systems, systems that store data with complex data types (for example, graphics data and audio data), and environment-modeling systems.
- ***Naïve users*** **:**
  - Naive users are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously.
  - For example, a bank teller who needs to transfer $50 from account A to account B invokes a program called transfer. This program asks the teller for the amount of money to be transferred, the account from which the money is to be transferred, and the account to which the money is to be transferred.

Database Administrator:

- Coordinates all the activities of the database system. The database administrator has a good understanding of the enterprise's information resources and needs.
- Database administrator's duties include:
  - **Schema definition:** The DBA creates the original database schema by executing a set of data definition statements in the DDL.
  - **Storage structure and access method definition.**
  - **Schema and physical organization modification:** The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve performance.
  - **Granting user authority to access the database:** By granting different types of authorization, the database administrator can regulate which parts of the database various users can access.
  - **Specifying integrity constraints.**
  - **Monitoring performance and responding to changes in requirements.**

k) Transaction Management

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

Simple Transaction Example

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

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

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

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

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

Transaction failure in between the operations

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

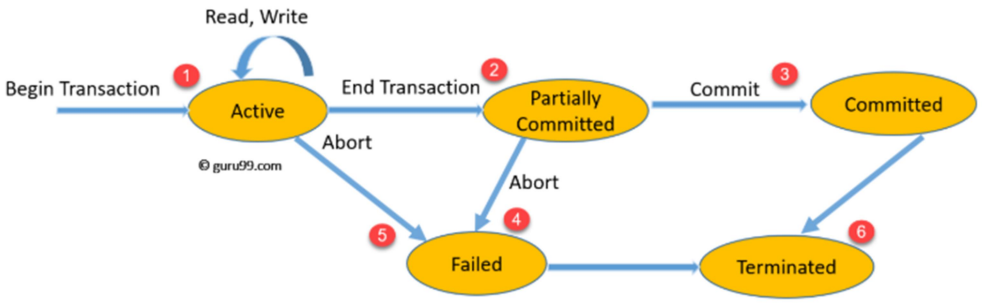
The main problem that can happen during a transaction is that the transaction can fail before finishing the all the operations in the set. This can happen due to power failure, system crash etc. This is a serious problem that can leave database in an inconsistent state. Assume that transaction fail after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

To solve this problem, we have the following two operations

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

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

Even though these operations can help us avoiding several issues that may arise during transaction but they are not sufficient when two transactions are running concurrently. To handle those problems we need to understand database ACID Properties



State Transition Diagram for a Database Transaction

What are ACID Properties?

ACID PROPERTIES are used for maintaining the integrity of database during transaction processing. ACID stands for **A**tomicity, **C**onsistency, **I**solation, and **D**urability.

- **Atomicity:** A transaction is a single unit of operation. You either execute it entirely or do not execute it at all. There cannot be partial execution.
- **Consistency:** Once the transaction is executed, it should move from one consistent state to another.
- **Isolation:** Transaction should be executed in isolation from other transactions (no Locks). During concurrent transaction execution, intermediate transaction results from simultaneously executed transactions should not be made available to each other. (Level 0,1,2,3)
- **Durability:** · After successful completion of a transaction, the changes in the database should persist. Even in the case of system failures.

Example of ACID

Transaction 1: Begin X=X+50, Y = Y-50 END
Transaction 2: Begin X=1.1*X, Y=1.1*Y END

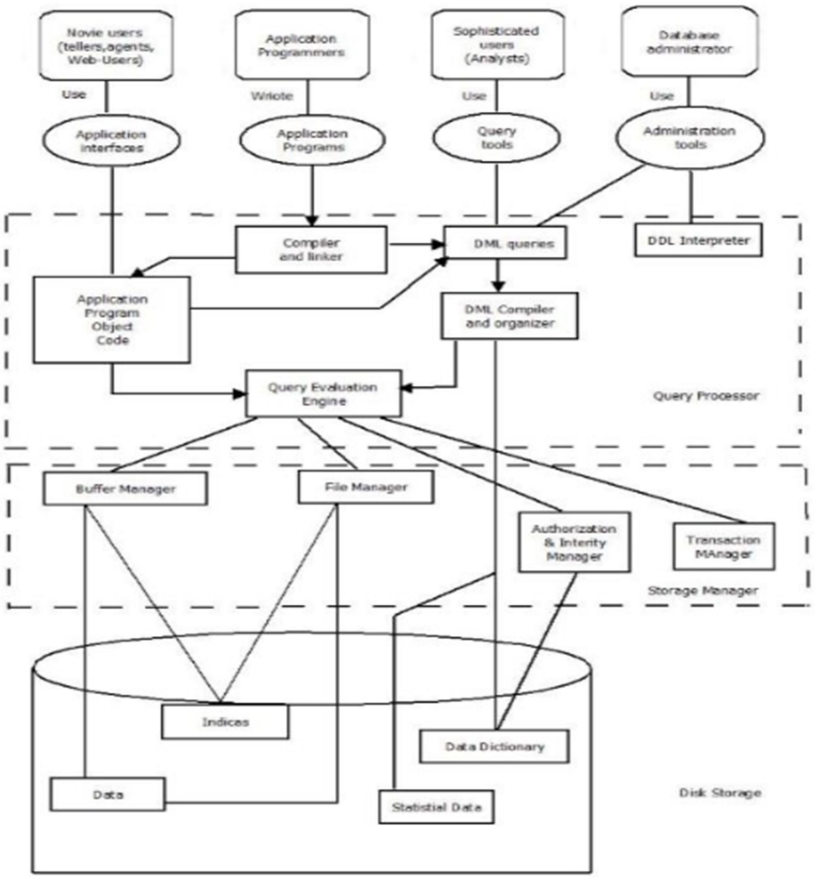
Transaction 1 is transferring Rupees 50 from account X to account Y.

Transaction 2 is crediting each account with a 10% interest payment.

If both transactions are submitted together, there is no guarantee that the Transaction 1 will execute before Transaction 2 or vice versa. Irrespective of the order, the result must be as if the transactions take place serially one after the other.

I) Database System Structure

A database system is partitioned into modules that deal with each of the responsibilities of the overall system. The functional components of a database system can be broadly divided into the **storage manager** and the **query processor** components. The storage manager is important because databases typically require a large amount of storage space. The query processor is important because it helps the database system simplify and facilitate access to data.



It is the job of the database system to translate updates and queries written in a nonprocedural language, at the logical level, into an efficient sequence of operations at the physical level.

Query Processor

The query processor components include

- **DDL interpreter,** which interprets DDL statements and records the definitions in the data dictionary.

- **DML compiler,** which translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands.

A query can usually be translated into any of a number of alternative evaluation plans that all give the same result. The DML compiler also performs **query optimization**, that is, it picks the lowest cost evaluation plan from among the alternatives.

· **Query evaluation engine,** which executes low-level instructions generated by the DML compiler.

Storage Manager

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

The storage manager components include:

· **Authorization and integrity manager**, which tests for the satisfaction of integrity constraints and checks the authority of users to access data.

· **Transaction manager**, which ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicting.

· **File manager**, which manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

· **Buffer manager**, which is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory. The buffer manager is a critical part of the database system, since it enables the database to handle data sizes that are much larger than the size of main memory.

Transaction Manager

A **transaction** is a collection of operations that performs a single logical function in a database application. Each transaction is a unit of both atomicity and consistency. Thus, we require that transactions do not violate any database-consistency constraints. That is, if the database was consistent when a transaction started, the database must be consistent when the transaction successfully terminates. **Transaction - manager** ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.