

Database Systems I

Notes

By: Nahason Matoke

1.0	OVERVIEW OF DATABASE SYSTEMS	4
1.1	Review Of Traditional Processing And It's Limitations	4
1.2	Evaluation of the DBMS.....	5
1.3	Types Of Database Systems.....	5
1.4	Advantages Of The Database Systems	6
1.5	Disadvantages Of Database Systems	7
	File System Environment.....	8
	Database System Environment	8
1.6	The Database System Environment	9
2.0	DATABASE ARCHITECTURE AND ENVIRONMENT	12
2.1	Abstraction and Data Integration	12
	Data Independence.....	15
2.3	Components Of The DBMS	16
2.4	Typical DBMS Functions	17
2.5	Overall System Structure	19
	Major Components Of Dbms.....	20
	Database Life Cycle (DBLC).....	20
3.0	CONCEPTUAL DATA MODEL.....	22
3.1	Types Of Data Models	22
3.2	The E- R Model (Entity Relationship).....	22
3.2.1	E-R Model Basic Concepts.....	23
3.2.2	Characteristics Of Attributes.....	23
3.2.3	Relationship Sets.....	24
	Types Of Relationships.....	24
	Exercise.....	25
3.3	Entity-Relationship Diagram	25
	Exercise.....	26
3.4	Entity modeling (Diagrammatic representation) relationships	27
	Exercise.....	27
	Exercise.....	28
3.5	DATA NORMALIZATION.....	30
3.5.1	Advantages of Normalisation Approach.....	32
3.5.2	Disadvantages	32
	Exercise.....	32
	Pharmacy drug dispensing card	33
4.0	RELATIONAL DATABASE SYSTEM.....	34
4.1	Relational Data Structure	34
	Properties of Relations.....	34
4.2	Relational Database Language.....	35
	Structured Query Language	35
	Components of SQL	35
4.2.1	Basic Structure Of SQL Statement	35
	FROM	36

Select clause.....	36
From Clause	37
Ordering Display of Tuples	37
4.2.2 Aggregate Functions	38
Null Values	38
4.2.3 Tuple Variables	38
4.2.4 String Operations	39
4.2.5 SQL and Set	40
UNION.....	40
UNION ALL	40
EXCEPT	40
4.4.6 VIEWS	41
4.4.7 Modification Of The Database.....	41
4.2.7 Schema Definition in SQL.....	43

1.0 OVERVIEW OF DATABASE SYSTEMS

- Data management, which focuses on data collection, storage and retrieval, constitutes a core activity for any organisation.
- To generate relevant information efficiently you need quick access to data (raw facts) from which the required information is produced.
- Efficient data management requires the use of a computer database. A database is a shared, integrated computer structure that houses a collection of: -
 - i. End -user data i.e. raw facts of interest to the user.
 - ii. Meta -data i.e. raw facts of interest to the user
 - iii. Meta data i.e. data about data through which the data is integrated. The Meta data provides a description of the data characteristics and the set of relationships that link the data found within the database. The database resembles a very well organized electronic filing cabinet in which powerful software referred to as DBMS helps manage the cabinet's contents.

1.1 Review Of Traditional Processing And It's Limitations

- Consider a saving bank enterprise that keeps information about all customers and savings accounts in permanent system files at the bank.
- The bank will need a number of applications e.g.
 - i. Program to debit or credit an account
 - ii. A program to add a new account
 - iii. A program to find the balance of an account
 - iv. A program to generate monthly statements
 - v. Any new program would be added as per the banks requirements
- Such a typical filing /processing system has the limitation of more and more files and application programs being added to the system at any time. Such a scheme has a number of major disadvantages:
 - i. **Data redundancy and inconsistency** - Since the files and application programs are created by different programmers over a long period of time, the files are likely to have different formats and the programs may be written in several programming languages. Moreover, the same piece of information may be duplicated in several files. This redundancy leads to higher storage and access costs. It may also lead to inconsistency i.e. the various copies of the same data may no longer agree.

- ii. **Difficulty in accessing** - Suppose that one of the bank officers needs to find out the names of all customers who live within the city's 78-phone code. The officer would ask the data processing department to generate such a list. Such a request may not have been anticipated while designing the system originally and the only options available are:-
 - ◆ Extract the data manually
 - ◆ Write the necessary application, therefore do not allow the data to be accessed conveniently and efficiently
- iii. **Data isolation** - Since data is scattered in various files and files may be in different formats, it may be difficult to write new applications programs to retrieve the appropriate data.
- iv. **Concurrent access anomalies** - Interaction of concurrent updates may result in inconsistent data e.g. if 2 customers withdraw funds say 50/= and 100/= from an account at about the same time the result of the concurrent execution may leave the account in an incorrect state.
- v. **Security problems** - Not every user of the database system should be able to access all the data. Since application programs are added to the system in an ad-hoc manner, it is difficult to enforce security constraints.
- vi. **Integrity** - The data value stored in the database must satisfy certain types of consistency constraints e.g. a balance of a bank account may never fall below a prescribed value e.g. 5,000/=. These constraints are enforced in a system by adding appropriate code in the various application programs. However, when new constraints are added there is need to change the other programs to enforce.

Conclusion.

These difficulties among others have prompted the development of DBMS.

1.2 Evaluation of the DBMS

Unlike the file system with many separate and unrelated files, the Database consists of logically related data store in a single data repository. The problems inherent in file systems make using the database system very desirable and therefore, the database represents a change in the way the end user data are stored accessed and arranged.

1.3 Types Of Database Systems

i. Single User database systems

This is a database system that supports one user at a time such that if user A is using the database, users B & C must wait until user A complete his or her database work.

If a single user database runs on a personal computer it's called a desktop database.

ii. Multi-user database

This is a database that supports multiple users at the same time for relatively small number e.g. 50 users in a department the database is referred to as a workgroup database. While one, which supports many departments is called an enterprise database.

iii. Centralized Database system

This is a database system that supports a database located at a single site.

iv. Distributed database system

This is a database system that supports a database distributed across several different sites.

v. Transactional DBMS/Production DBMS

This is a database system that supports immediate response transaction e.g. sale of a product.

vi. Decision Support DBMS

It focuses primarily on the production of information required to make a tactical or strategic decision at middle and high management levels.

1.4 Advantages Of The Database Systems

1. **Centralized Control** - Via the DBA it is possible to enforce centralized management and control of data. This means that necessary modifications, which do not affect other application changes, meet the data independence DBMS requirement.
2. **Reduction of redundancies** - Unnecessary duplication of data is avoided effectively reducing total amount of data required, consequently the reduction of storage space. It also eliminates extra processing necessary to trace the required data in a large mass of data. It also eliminates inconsistencies. Any redundancies that exist in the DBMS are controlled and the system ensures that his multiple copies are consistent.
3. **Shared data** - In a DBMS, sharing of data under its control by a number of application programs and user is possible e.g. backups.
4. **Integrity** - Centralized control can also ensure that adequate checks are incorporated to the DBMS provide data integrity. Data integrity means that the data contained in the database is both accurate and consistent e.g. employee age must be between 28-25 years.

5. **Security** - Only authorized people must access confidential data. The DBA ensures that proper access procedures are followed including proper authentication schemes process that the DBMS and additional checks before permitting access to sensitive data. Different levels of security can be implemented for various types of data or operations.
6. **Conflict Resolution** - The DBA is in a position to resolve conflicting resolve conflicting requirements of various users and applications. It is by choosing the best file structure and access method to get optimum performance for the response. This could be by classifying applications into critical and less critical applications.
7. **Data Independence** - It involves both logical and physical independence logical data independence indicates that the conceptual schemes can be changed without affecting the existing external schemes. Physical data independence indicates that the physical storage structures/devices used for storing the data would be changed without necessitating a change in the conceptual view or any of the external use.

1.5 Disadvantages Of Database Systems

1. Cost - in terms of:

- The DBMS - software
- Purchasing or developing S/W
- H/W
- Workspace (disks for storage)
- Migration (movement from tradition separate systems to an integrated one)

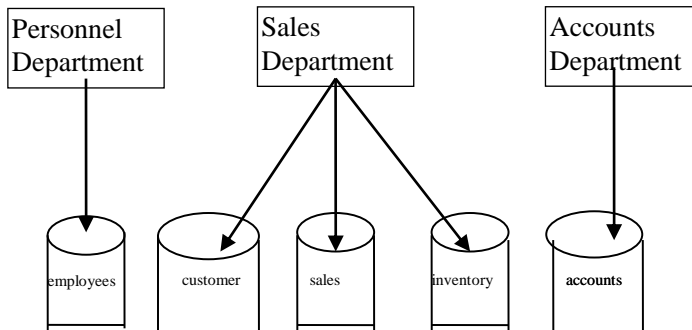
2. Centralization Problems

You would require adequate backup incase of failure

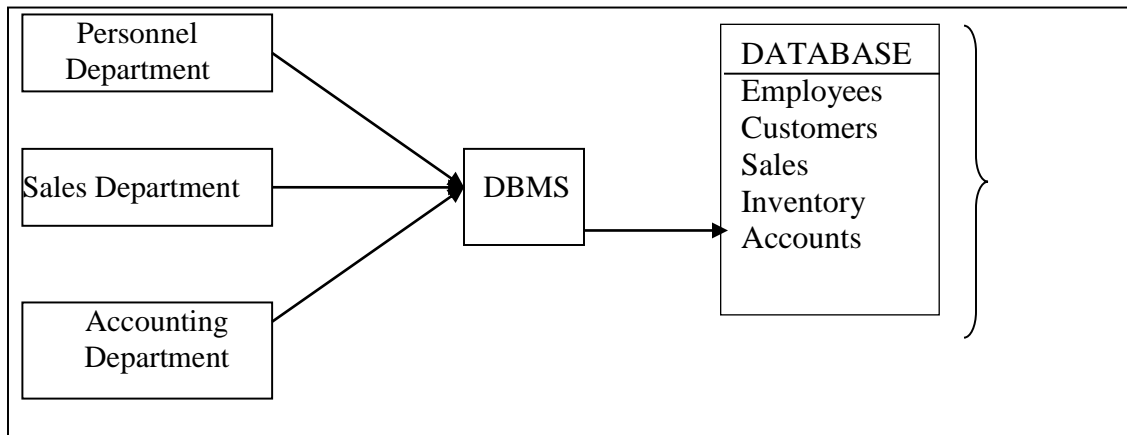
You would require increased severity of security breeches and disruption of operation of the organisation because of downtimes and failures.

3. Complexity of Backup and recovery

File System Environment



-



Database System Environment

The database eliminates most of the file systems' data inconsistencies, anomalies and structural dependency problems.

The current generation of DBMS software stores not only the data structures in a central location but also stores the relationships between the database components

The DBMS also takes care of defining all the required access paths of the required component.

1.6 The Database System Environment

The term database system refers to an organisation of components that define and regulate the collection storage, management and use of data within a database environment.

The database system is composed of 5 major parts i.e.

- | | |
|-------------|---------------|
| a. Hardware | d. Procedures |
| b. Software | e. Data |
| c. People | |

Hardware

This identifies all the systems physical devices e.g. the composition peripherals, storage devices etc.

Software

These are a collection of programs used by the computers within the database system.

- i. O.S - manages all hardware components and makes it possible for all other and software to run on the composition.
- ii. The DBMS - manages the database within the database system e.g. Oracle, DB2, Ms Access etc.
- iii. Applications programs and utilities to access and manipulate data in the DBMS.

People

These are all database systems users:-

1. **Systems administrator** - Oversees the database systems general operations.
2. **Database administrator (DBA)** - Manages the DBMS use and ensures that the database is functioning properly. His functions include:
 - i. Scheme definition - The original database scheme is created by writing a set of definitions, which are translated by DDL compiler to a set of tables that are permanently stored in the data dictionary.
 - ii. Storage structure and Access Methods Definitions - By writing a set of definitions for appropriate storage structures and access methods, which are translated by the data storage and definition language compiler.
 - iii. Scheme and physical organisation modifications - Modification to either the database schema or description of the physical storage organisation are accompanied by writing a set of definitions which are used by either the DDL compiler or the data storage and definition language compiler to generate modification to appropriate internal systems tables e.g. data dictionary.
 - iv. Granting authorization to data access - This is so as to regulate which parts of the database users can access.
 - v. The database manager keeps integrity Constrains in a special system structure whenever an update takes place in the system.
3. **Database designers** - These are the database architects who design the database structure.
4. **Systems Analysts & Programmers (application programmers)** - They design and implement the application programs they design & create the data entry scheme, reports & procedures through which users access and manipulate the databases data.

5. **End users** - These are the people who use the application programs to run the organizations daily operations. They fall in the following classes:

- i. Sophisticated users - These interact with the system without writing programs. They form their requests in a database query language.
- ii. Specialized database applications that do not fit in the traditional data processing framework e.g. CAD Systems, knowledge based & expert systems.
- iii. Application programmers: These interact with the system through the DML & applications.
- iv. Naive – Unsophisticated user who interact with the systems by invoking one of the permanent application programs that have been written previously.

Procedures

- These are instructions and rules that govern the design and use of the database system.
- They enforce standards by which business is conducted within the organisation and with customers.
- They also ensure that there is an organized way to monitor and audit both the data that enter the database and the information that is generated through the use of such data.

DATA

This covers the collection of facts stored in the database and since data is the raw material from which information is generated the determination of what data is to be stored into the database and how the data is to be organized is a vital part of the database designer jobs.

2.0 DATABASE ARCHITECTURE AND ENVIRONMENT

2.1 Abstraction and Data Integration

Abstraction is the simplification mechanism to hide superfluous details of a set of objects. It allows one to concentrate on the properties that are of interest to the application e.g. a car is an abstraction of personal transportation vehicle but does not reveal details about model, year, colour etc.

Vehicle itself is an abstraction that includes the types; car, truck, bus and lorry.

Consider a non- database environment of a number of application programs as shown below:

Application 1 will contain values for the attributes employee Name and Employee. Address and this record can be described in pseudo-code as

```
Type  Employee = record
      Employee.name:string
      Employee.address:string
End
```

Application 2 will have:

```
Type  Employee = record
      Employee.name: String
      Employee.soc_sec_No: Integer
      Employee.Address: String
      Employee. Annual_salary:integer
End
```

In a non-database environment each application is responsible for maintaining the currency of data and a change in data item.

In a database environment, data can be stored in this application and their requirement be integrated by whoever is responsible for centralized control (DBA).

The integrated version would appear as recorded containing attributes required by both applications.

The record will appear as:

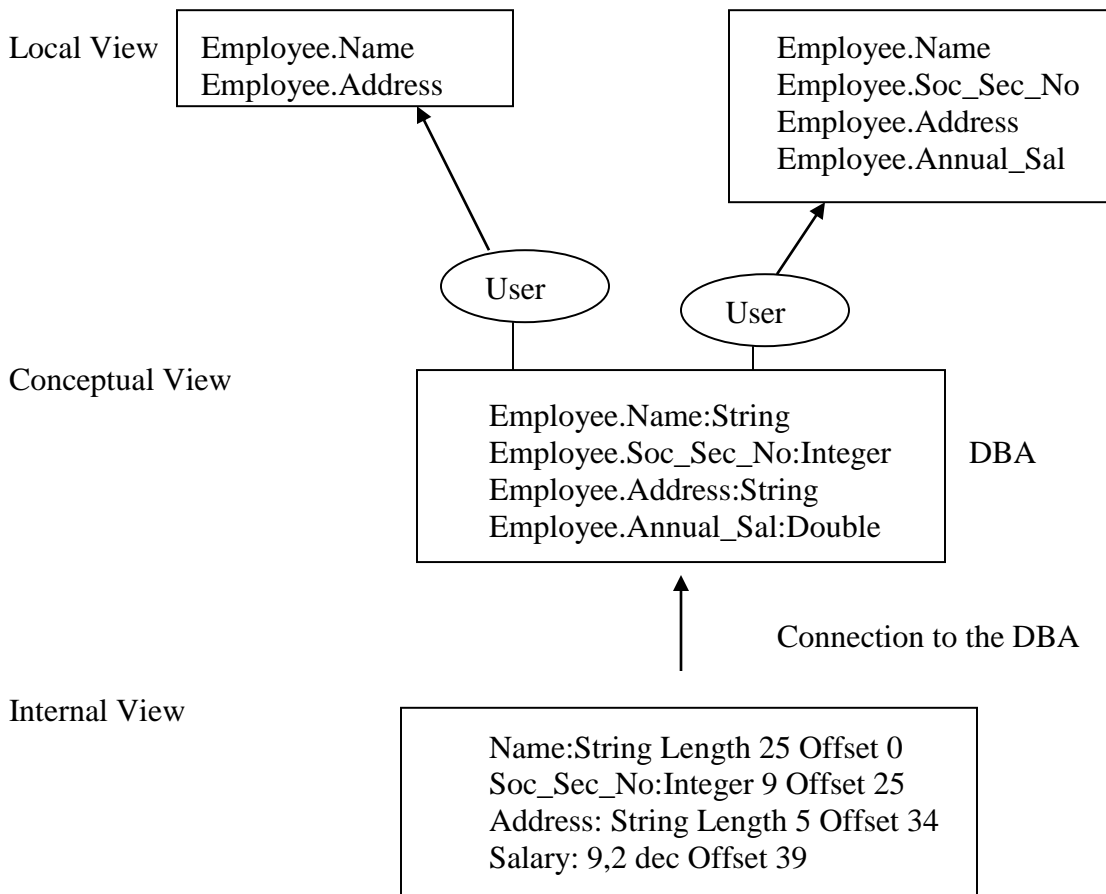
```
Type  Employee = record
      Employee.Name:string
      Employee.soc-sec.no: Integer
      Employee.Address:string
      Employee.Annual_Salary: double
End
```

The views supported are derived from the conceptual record by using appropriate mapping.

The application programs no longer require information about the storage structure; storage device types or access methods. These are absorbed by the DBMS.

There are 3 level abstractions corresponding to 3 views:

- i. The highest level which is seen by the application programs or user called "external or user view"
- ii. A sum total of users view called global view a conceptual view.
- iii. Lower level which is the description of the actual method of storing the data. It is also referred to as the internal view.



The 3 level scheme architecture is called the ANSI/SPARC model (American National Standard Institute/Standards Planning and Requirements Committee.)

It is divided into 3 levels:

- External
- Conceptual
- Internal

The view of each level is described as a scheme, which is an outline or a plan that describes the records and relations existing in the view. It also describes the way in which entities at one level of abstraction can be mapped onto the next level.

External Level (External or User view)

This is at the highest level of database abstraction where only those portions of the database of concern to the user or application programs are included.

Any number of user views may be possible, some of which may be identical.

Each external view is described by means of a scheme called external scheme, which consists of a definition of the logical records and the relationships in the external view.

It also contains the method of devising the objects in the external view from the objects in the conceptual view (entities, attributes and relationships).

Conceptual Or Global View

Contains all database entities and the relationships among them are included and one conceptual view represents the entire database.

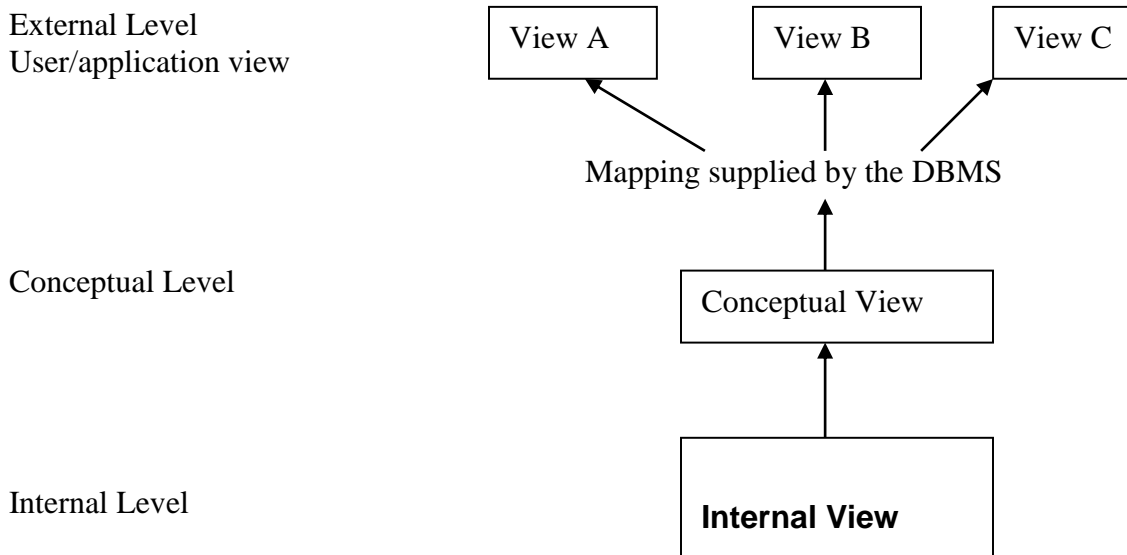
It is defined by the conceptual scheme. Also contains the methods of deriving the objects in the conceptual view from the objects in the internal view.

It is independent of the physical presentation.

Internal View

This is the lowest level of abstraction closest to the physical storage method used.

It indicated how data would be stored and describe the data structures and access methods to be used by the database. The internal schema implements it.



The 3 levels of architecture of a DBMS

Mapping between views

Two mappings are required, one between external and conceptual views and another between the conceptual records to internal ones.

Data Independence

This is the immunity of users/application programs from changes in storage structure and access mechanism.

The 3 levels of abstractions along with the mappings from internal to conceptual and from conceptual to external provide 2 distinct levels of data independence i.e.:

- Logical Data Independence
- Physical Data Independence

(i) Logical Data Independence

This indicates that the conceptual schema can be changed without affecting the existing external schema.

The mapping between the external and conceptual levels would absorb the change.

It also insulates application programs from operations such as combining two records into one or splitting an existing record into 2 or more records. The LDI is achieved by providing the external level or user view database.

The application programs or users see the database as described by the respective external view.

DBMS provided a mapping from this view to the conceptual view.

NB: The view at conceptual level of the database is the sum total of the current and anticipated views of the database.

(ii) Physical Data Independence

This indicates that the physical storage structures or devices used for storing the data can be changed without necessitating a change in the conceptual view or any of the external view. Any change is absorbed by the mapping between the conceptual and internal views.

2.3 Components Of The DBMS

A DBMS is software used to build, maintain and control database systems. It allows a systematic approach to the storage and retrieval of data in a computer.

Most DBMS(s) have several major components, which include the following:

1. **Data Definition Language (DDL)** - These are commands used for creating and altering the structure of the database.
The structures comprise of Field Names, Field sizes, Type of data for each field, File organizational technique. The DDL commands are used to create new objects, alter the structure of existing ones or completely remove objects from the system.
2. **Data Manipulation language (DML)** - This is the user language interface and is used for executing and modifying the contents of the database. These commands allow access and manipulation of data for output. They include commands for adding, inserting, deleting, sorting, displaying, painting etc. These are the most frequently used commands once the database has been created.
3. **Data Control Language (DCL)** - These are commands used to control access to the database in response to DML commands. It acts as an interface between the DML and the OS. It provides security and control to the data.
4. **Query Languages** - A query language is a formalized method of constructing queries in database system. It provides the ways in which the user interrogates the database for data without using conventional programs. For relation database, structures query languages (SQL) has emerged as the standard language. Almost all the DBMS(s) use SQL running on machines ranging from microcomputers to large main frames.
5. **Form Generator** - A form is a screen display version of a paper form, which can be used for both input and output.
6. **Menu Generator** - This is used to generate different types of menus to suit user requirements.
7. **Report Generator** - This is a tool that gives non- specialized users the capability of providing reports from one or more files through easily constructed statements. The reports may be produced either constructed statements. The reports may be produced either on screen or paper. A report generator has the following features:
 - Page headings and footings
 - Page Numbering
 - Sorting
 - Combining data from several files
 - Column headings
 - Totaling and subtotaling
 - Grouping of data

- Reports titling
8. **Business Graphics** - Some DBMS may provide means of generating graphical output e.g. bar charts, pie charts scatter graphics line plots etc. others will allow users to export data into graphics software.
 9. **Application Generators** - This is a type of 4th generation language used to create complete application programs. The user describes what need to be done, the data and files that are to be used and the application generator then translates the description into a program. They are also referred to as rapid application tools.
 10. **Data Dictionary (DD)** - This provides the following facilities:
 - Documentation of data items
 - Provision of Standard definition and names for data items.
 - Data item description.
 - Removal of redundancy in documentation of data item.
 - Documentation of relationships between data items;
 11. **Fourth Generation Languages (4GL'S)** - A 4GL'S is a non-procedural language in which the programs flow and not designed by the programmer but by the 4G software itself. The user requests for the result rather than a detailed procedure to obtain these results.

2.4 Typical DBMS Functions

A DBMS performs several functions that guarantee the integrity and consistency of the data in the database. Most of these functions are transparent to end-users and can be achieved only through the use of a DBMS. They include:

- i. **Data Dictionary Management** - The DBMS enquires that definitions of the data element and their relationships (metadata) be stored in a data dictionary. The DBMS uses the DD to look up the required data component, structures and relationships thus relieving us from having to code such complex relationships in each program. Any changes made in the database structure are automatically recorded in the DD thereby freeing us from having to modify all the programs that access the changed structure. So, the DBMS provides data obstruction and removes structural or data dependency of the system.
- ii. **Data Storage Management** - Creation of complex structure required for data storage is done by DBMS thus relieving us from the difficult task of defining and programming the physical data characteristics. A modern DBMS system provides storage for data and related data entry forms or screen definitions, report definition, data validation rules, procedural code structures to handle video and picture formats etc.

- iii. ***Data Transformation and Presentation*** - Transformation of entered data to conform the data structures that are required to store the data is done by the DBMS relieving us the core issue of making a distinction between the data logical formats and data physical format. By maintaining data independence the DBMS translates logical requests into commands that physically locate and retrieve the requested data. That is the DBMS transform the physically retrieved data to conform to the users logical expectations. This is by providing application programs with software independence and data abstraction.
- iv. ***Security Management*** - The DBMS creates the systems security that enforces users security and data privacy within the database. Security rules determine which users can access database which data item each user can access and which data operations (read, add, delete, modify) the user may perform. This is important in multi user database system where many users can access the database simultaneously.
- v. ***Multi User Access Control*** - The database creates complex structures that allow multi-user access to the structure. In order to provide data integrity and consistency the DBMS uses sophisticated algorithms to ensure that multiple users can access the database concurrently and still guarantee integrity of the database.
- vi. ***Back-up and recovery management*** - To ensure data safety and integrity current DBMS systems provide special utilities that allow the DBA to perform backup and special backup and restore procedures. Recovery management deals with recovery of the database after a failure such as a bad sector in the disk, a power failure etc. Such capability is critical to the preservation of the database integrity.
- vii. ***Data integrity Management*** - The DBMS promotes and enforces integrity rules to eliminate data integrity problems thus minimizing data redundancy and maximizing data consistency. The relationships stored in the Data Dictionary are used to enforce data integrity. Data integrity is especially important in transaction oriented database systems.
- viii. ***Data base Access Language and Application Programming Interfaces*** - The DBMS provides data access via a query language. It contains 2 components, DDL and DML. The DDL defines the structures in which the data are housed and the DML allows end users to extract the data from the database. It also allows data access to programmers via procedural languages such as Cobol, C, Pascal, and Visual Basic etc. It also provides utilities used by the DBA and the Database Designer to create, implement, monitor and maintain the database.

- ix. ***Database Communication interfaces*** - Current generation of DBMS's provide special communication routines designed to allow the database to accept end-user requests within a computer network environment. The DBMS may provide communication functions to access the database through the internet using internet browsers e.g. Netscape or Explorer as the front-ends

2.5 Overall System Structure

A database system is partitioned into modules that deal with each of the responsibilities of the overall systems. The design of the database system must include consideration of the interface between the database system and the O.S. The functional components of a database system include:

- File Manager
- Data base manager
- Query processor
- DML pre-compiler
- DDL compiler

File Manager

This manages the allocation of space in the disk storage and the data structures used to represent information stored on the disk. It deals more on the physical aspects.

Database Manager

Provides the interface between the low level data stored in the database and the application and programs the queries submitted to the system.

Query Processor

This translates statements in a query language into low-level instruction that the DB manager understands. In addition the query processor attempts to transform a user request into more efficient statement, thus finding a good strategy for executing the query.

DML Pre-compiler

This converts the DML statements embedded in an application program to normal procedure calls in the language. The pre-compiler must interact with the query processor order generate the appropriate code.

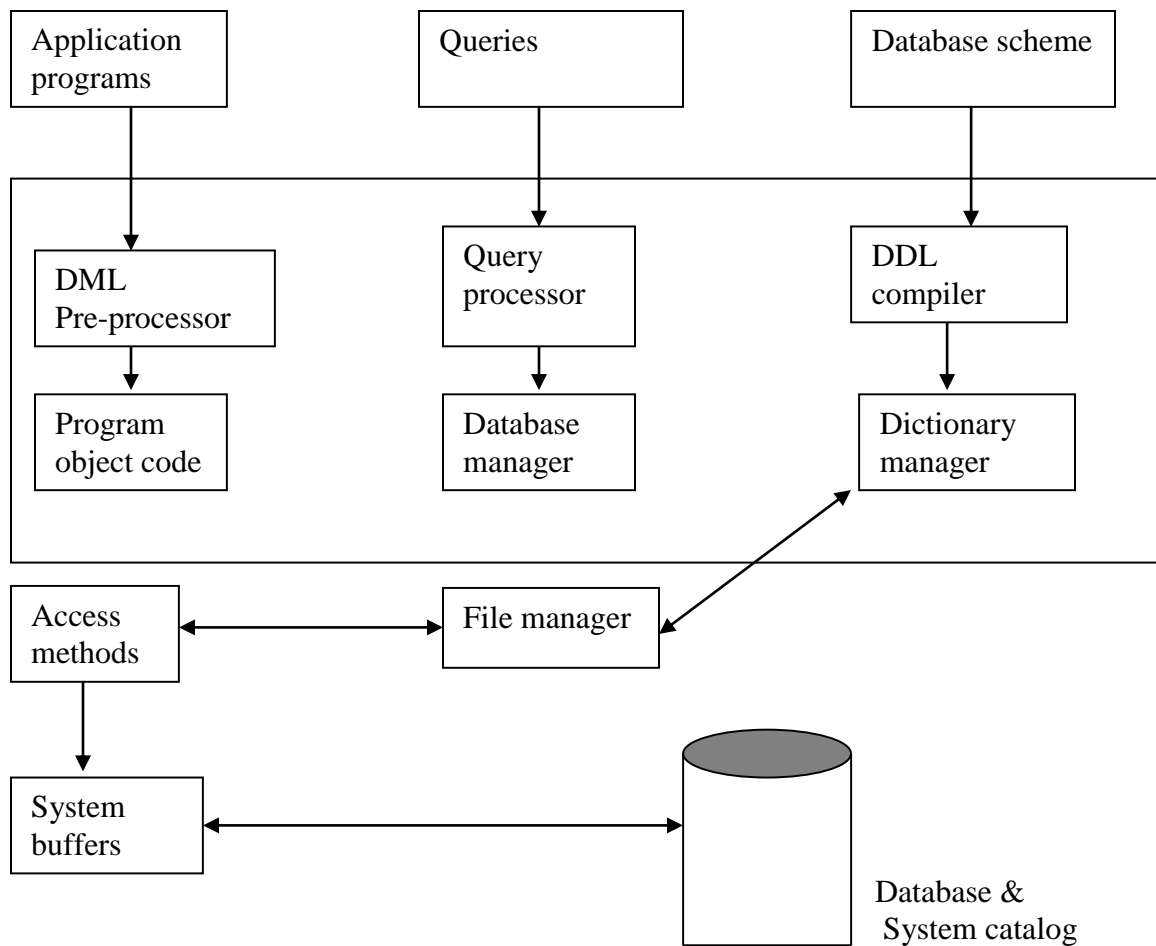
DDL Compiler - This converts DDL statements to a set of table containing metadata.

Major Components Of Dbms

Programmers

Users

DBA



Database Life Cycle (DBLC)

1. The Database Initial Study

- Examine the current system operation.
- Try to establish how and why the current system fails.
- Define the problems and constraints
- Define the objectives
- Define scope and boundaries

2. Database Design

- This involves the conceptual design, selection of database, management system software.
- Creation of the logical design
- Creation of the physical design

3. Implementation

- This involves installation of the DBMS
- Creation of the database
- Loading or conversion of data

4. Testing and evaluation

The activities involve:

- Testing the database
- Tune the database
- Evaluate the database application programs
- Provide the required information flow

5. Operation

Once the database has passed the evaluation stage it is considered to be operational, the database, its management, its users and its application programs constitute a complete I.S. The beginning of the operational phase starts the process of system evaluation.

6. Maintenance and Evaluation

It involves the following:

- Preventive Maintenance
- Corrective maintenance
- Adaptive maintenance
- Assignment and maintenance of access permission to new and old user
- Generation of database access statistics to improve the efficiency and usefulness of audits and to monitor system persons.
- Periodic security based on the system generated statistics
- Periodic (monthly, quarterly or yearly) system using summaries for internal billing or budgeting purposes.

3.0 CONCEPTUAL DATA MODEL

A database model is a collection of logical constructs used to represent the data structure and relationships found within the database.

3.1 Types Of Data Models

1. Object Based Logical Models

They are used in describing data at the conceptual and view levels. They provide fairly flexible structuring capabilities and allow data constraints to be specified explicitly.

They include:

- E - R Model
- Object Oriented Model
- Binary Model
- Semantic Data Models
- Info-logical Data Model
- Function

2. Record Based Logical Models

These are models used in describing data at the conceptual and view levels. They are used to specify the overall logical structure of the database and to provide a higher-level description implementation. It is hard to understand.

3. Physical Data Models

These are models that are used to describe data at the lowest level. They are very few in number and the two widely known ones are:

- i. Unifying model
- ii. Frame memory model

NB: Like the E-R model, the object-oriented model is based on a collection of object where an object contains values stored in instance variables with the object.

3.2 The E- R Model (Entity Relationship)

It is based on a perception over a real world, which consists of a collection of basic objects called entities and relationships among this objects. An entity is an object that is distinguished from other objects via a specific set of attributes.

3.2.1 E-R Model Basic Concepts

The model employs the following components:

- Entity sets
- Relationship sets
- Attributes

1. Entity sets

An entity is a thing or object in the real world that is distinguishable from all other objects. It may be concrete e.g. a person or a book or it may be abstract e.g. a loan, holiday a concept etc. An entity set is a set of entities of the same type that share the same properties or attitudes e.g. a set of all persons who are customers of a bank.

2. Relationship sets

An association between two or more entities is called a relationship.

3. Attributes

They are descriptive properties or characteristics possessed by each member of an entity set.

3.2.2 Characteristics Of Attributes

1. Simple and Composite attributes - e.g. a customer name or first name, middle name, last name. Composite attributes are necessary if a user wishes to refer to entire attribute on some occasions and to only a component of the attributes on other occasions.
2. Single valued and Multi valued Attribute - The social security number or ID number can only have a single value at any instance and therefore its said to be single valued. An attribute like dependant name can take several values ranging from 0-n thus it is said to be multi valued.
3. Null Attributes - A null value is used when an entity does not have a value for an attribute e.g. dependent name.
4. Calculated attribute - The value for this type of attribute can be derived from the values of other related attributes or entities e.g.
 - i. Employment length value can be derived from the value for the start date and the current date.
 - ii. Loans held can be a count of the number of loans a customer has.

3.2.3 Relationship Sets

A relationship is an association amongst several entities while a relationship set is a set of relationships of the same tuple. It is a mathematical relation on $n > 2$ possible non-distinct entity sets e.g. consider 2 entity sets, loan and branch. A relationship set loan, branch can be defined to denote association between a bank loan and the branch in which that loan is obtained.

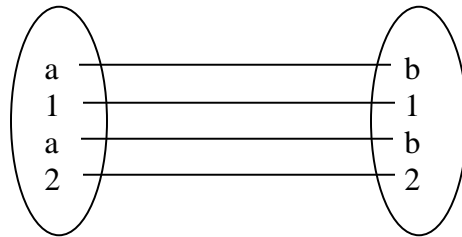
Example

Consider 2 entity sets Customer and loan.

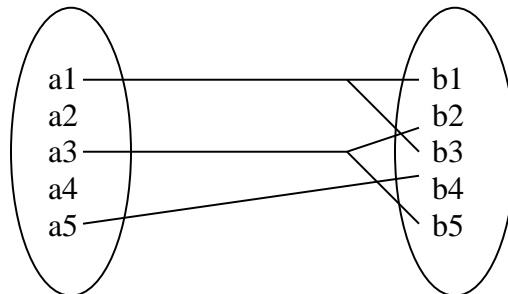
A relationship set - A borrower can be defined to denote the association between customers and the bank loans that the customers have.

Types Of Relationships

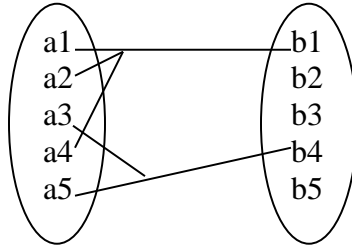
- i. One to one relationship (**1:1**) - An entity in **A** is associated with utmost one entity in **B** while an entity in **B** is associated with at utmost one entity in **A**.



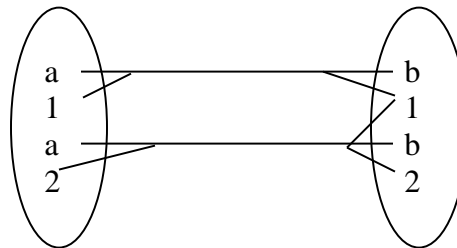
- ii. One to Many relationship (**1:M**) - An entity in **A** is associated with any number of entities in **B** while an entity in **B** can be associated with at most one entity in **A**.



- iii. Many to one relationship (**M:1**) - An entity in **A** is associated with at most one entity in **B** and an entity in **B** can be associated with a number of entities in **A**.



- iv. Many to many (**M:N**) - An entity in **A** is associated with at least one entity in **B** and an entity in **B** can be associated with a number of entities in **A**.



Existence Dependencies

If the existence of an entity X depends on the existence of entity Y, then X is said to be existence dependent on Y. If Y is deleted, so is X. Y is said to be the dominant entity and X is said to be subordinate entity.

Exercise.

Differentiate between super key, primary candidates and candidate keys.

3.3 Entity-Relationship Diagram

Components of E-R diagram

- (i) Rectangles: - They represent entity sets.
- (ii) Ellipses: - represent attributes
- (iii) Diamond: - represents relationship sets
- (iv) Lines - Link attributes to entities and entity sets to relationship sets
- (v) Double ellipses: - represent multi-value attributes
- (vi) Dashed ellipses: - denote derived attributes
- (vii) Double lines: - indicate total participation of an entity in relationship sets.

Exercise.

Draw an E-R diagram that shows the hospital environment, theatres, patients (in and out-patients) doctors, nurses, wards and ward beds.

Weak Entity Set

This is an entity set that does not have sufficient attributes to form a primary e.g. an entity set payments comprising of the attributes payment number, payment date and payment amount. Although each payment entity is distinct, payment for different loan e.g. may share the same payment number thus this entity set does not have a primary key.

Strong Entity Set

This is an entity set that has a primary key. For weak entity set to be meaningful it must be part of a one to many relationships.

Specialisation

An entity set may include sub-groupings of entities that are distinct in some way from other entities in the set. This is called specialization of the entity set e.g. the entity bank account could have different types e.g.

- Credit account
- Checking account
- Savings account - interest rate
- Checking account - overdraft amount

Under checking account you could have type:

- i. Standard check account
- ii. Gold checking account
- iii. Senior checking account

For the standard it may be divided by number count of checks gold minimum balance and an interest payment.

Senior checking account - age limit

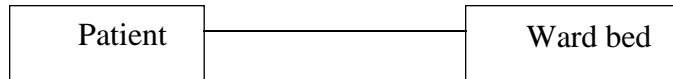
A specialised entity set may be specialised by one or more distinguishing features.

Aggregation

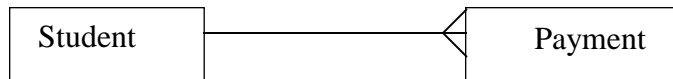
This is abstraction through which relationships are treated as higher-level entities e.g. the relationship set borrower and the entity sets customer and loan can be treated as a higher set called borrower as a whole.

3.4 Entity modeling (Diagrammatic representation) relationships

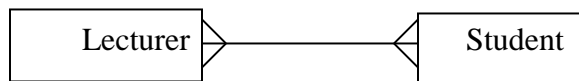
i. One to one relationship



ii. One to many relationship



iii. Many to many relationship



NB: Whenever the degree of a relationship is many to many we must decompose the relationship to one-to-one or one-to-many. The decomposition process will create a new entity.

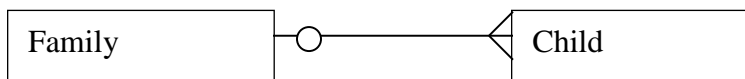
Exercise

A company consists of a number of departments each having a number of employees each department has a manager who must be on a monthly payroll, other employees are either on a monthly or weekly payroll and are members of the sports club if they so wish. Construct an entity - relationship diagram depicting the scenario.

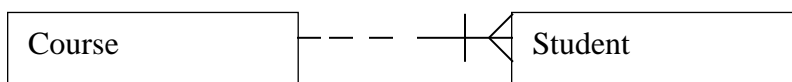
Mandatory and Optional

Optional relationships are shown by either, use of a small circle drawn along the line or a dotted line while mandatory relationships are shown by use of either a bar drawn across the line or a continuous line.

Optional



Mandatory



Representing Attributes

Although E-R diagrams describe many of the features of the logical model, they do not show the attributes associated with each entity, this additional information is represented conveniently in form of a table.

Exercise

Consider the entity relationship Student_Course that defines a course undertaken by many students.

Generate a sample tabular representation of the above assuming key attributes are course-code and stud-no respectively.

A HOSPITAL DATABASE SYSTEM.

A hospital wishes to maintain a database to assist the administration of its wards and operating theatres, and to maintain information relating to its patients, surgeons and nurses.

Information in relation to patients is captured on admission and patients are assigned to a ward. A nurse is assigned to a ward. Nurses are identified by their staff numbers and their names, address, phone numbers and grades are also recorded. Each ward has a unique number and is dedicated to a type of patient (e.g. pediatric, maternity, etc)

A patient may have a number of operations. The information to be recorded about an operation include the type of operation, the patient, the surgeons involved , date, time and location.

Only one surgeon may perform an operation, any other surgeons present being considered as assisting at the operation. Surgeons come under the direction of senior surgeons, called consultants, who may also perform or assist at operations. Information recorded about a surgeon includes name, address and phone number.

An operation can be performed in only one theatre but a given theatre may be the location of many operations.

A nurse may or may not be assigned to a theatre and he/she cannot be assigned to more than one theatre. A theatre may have many nurses assigned to it.

Required.

- ❖ Design and develop a database system for the above application. This should include:
- ❖ Logical data model.
- ❖ Forms for data entry.

❖ Integrity and security features.

❖ Reports including the following:

- i.) List of all operations scheduled for the following week.
- ii.) List of all in-patients and their ailments.
- iii.) Details of bed occupancy/availability.
- iv.) Summary list of all patients for specified doctors.
- v.) Theatre occupancy/availability

3.5 DATA NORMALIZATION

Normalisation is the process of applying a number of rules to the tables, which have been identified in order to simplify. The aim is to highlight dependencies between the various data items so that we can reduce these dependencies.

The rules applied are referred to as: -

- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)

1NF

A table or relation is said to be in first normal form, if and only if it contains no repeating groups i.e. it has no repeated values for particular attributes in a simple record. If there are repeating groups and attributes they should be isolated to form a new entity.

2NF


A table is said to be in 2NF if and only if it is in 1NF and every non-key attribute is fully dependent on the key attribute. Attributes not fully dependent should be isolated to form a new entity.

3NF

A table is said to be in 3NF if and only if it is 2nd NF and every non-key attribute is not dependent on any other non-key attribute. All non-key attributes that are dependent on other non-key attributes, should be isolated to form a new entity

Example: An invoice

Invoice No. _____		Date _____		
Customer _____		Delivery to _____		
Address _____				

				
Product Code	Description	Quantity	Price	Amount
Thank you			Amount	

Un-normalised data.

Invoice (Invoice no., Date, Customer, Cust_address, Deliv_To, Product code, Quantity, Unit Price, amount, Invoice amount)

INF (Identify and separate repeating groups to form a new entity)

INVOICE (Invoice number, date, customer address, Deliv_ address, Invoice_Amount)

PRODUCT (Product code, invoice number, product description, Quantity, Unit price, amount)

2NF (identify and separate non-key attributes not fully dependent on key attribute)

INVOICE (Invoice-no, date, customer address, del.address, invoice total)

PRODUCT (prod-code, prod-description, unit price)

INVOICE PRODUCT(Prod_Code, Invoice_No, Quantity, Amount)

3NF (Identify non-key attributes dependent on other non-key attributes)

INVOICE (Invoice-no, Customer_Number, Date, invoice total)

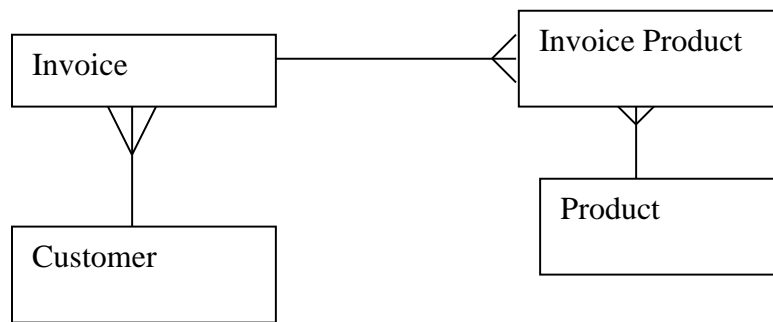
PRODUCT (Prod-code, prod-description, unit price)

INVOICE PRODUCT(Prod_Code, Invoice_No, Quantity, Amount)

CUSTOMER (Customer_Number, Customer_Name, customer_Address, del.address)

NB: Whenever there is no composite key the table is in 3NF

Corresponding ERD



3.5.1 Advantages of Normalisation Approach

1. It is a formal technique with each stage of normalisation process elimination a particular type of undesirable dependency as well as each stage of normalisation eliminating a certain type of redundancy.
2. It highlights constraints and dependencies in the data and helps in understanding the nature of the data.
3. The 3NF produces well-designed databases, which provide a high degree of independence.

3.5.2 Disadvantages

1. It depends a thorough understanding of the entities and their relationships.
2. It's a complex process particularly if the entities are many.

Exercise

- A customer account details in a bank are stored in a table that has the following structure, normalise this data to 3NF. Customer (branch -no, account no, address, postcode, tel)

- ♦ A hospital drug dispensing record requires that, for each patient, the pharmacy must record the following information.

Pharmacy drug dispensing card

Patient No Surname Sex

.....

Date of Birth Address

Ward No Ward name Date

.....

Name of company paying..... Company address

.....

Date	Drug code	Drug name	Quantity	Unit price	Amount

Total

Paid

Balance

- Explain what you understand by data normalization stating each of the three normal forms.
- Perform data normalization for the table to 3NF. Showing clearly the results of each stage.

4.0 RELATIONAL DATABASE SYSTEM

Motivation

1. To shield programmers and users from the structural complexities of the database.
2. For conceptual simplicity

4.1 Relational Data Structure

Relation: A relation corresponds to a table

Tuple: Corresponds to a row of a table

Domain: Is a pool of values from which one or 2 values draw their actual values e.g. the Town Domain is a set of all legal town names. A relation on domains $D_1, D_2, D_3, \dots, D_n$ (not necessarily all distinct) consists of a heading and a body, the heading consists of a fixed head of attributes a_1, a_2, a_3, \dots such that each attribute a_i corresponds to exactly one of the underlying domain D_i . The body consists of a time varying set of tuples where each tuple in turn consists of a set of attribute value pairs (a_i, r_i)

Properties of Relations

1. There is no duplicate tuples – The body of a relation is a mathematical set, which by definition does not include duplicate elements.
2. Tuples are unordered - Sets are unordered
3. Attributes are unordered - The heading of a relation is a set that is unordered.
4. All simple attributes values are atomic meaning that relations do not contain repeating groups (normalized)

Primary Keys

These are special type of more general construct candidate keys. A candidate key is a unique identifier and each relation has at least one candidate key. For a given relation, one of the candidate keys is chosen to be the primary key and the rest are called alternate keys.

Let r be a relation with attributes a_1, a_2, a_n . The set of attributes $K = (A_i, A_j, \dots, A_K)$ of R is said to be a candidate key of R . If it satisfies the following 2 time independent properties:

- i. Uniqueness - At any given time, no 2 distinct types of R have the same values for A_i, A_j, \dots, A_K .
- ii. Minimality - None of A_i, A_j, \dots, A_K can be discarded from K without destroying the uniqueness property.

4.2 Relational Database Language

Structured Query Language

Components of SQL

- i. Data Definition Language (DDL) - DDL provides commands for defining relation schemes, deletion relation, creating indices and modifying relation schemers
- ii. Interactive Data Manipulation Language (DML) - DML includes a query language based on both relational calculus. It includes commands to insert tuples into, delete tuples from and modify tuples in the database.
- iii. Embedded DML - This is designed for use within general purpose programming languages such as PL/1, Cobol, Pascal, Fortran and C.
- iv. View Definition - The SQL DDL includes commands for specifying access rights to relations and view.
- v. Integrity - The SQL DDL includes commands for specifying integrity constraints that the data stored in the database must satisfy. Updates that violate integrity constraints as disallowed.
- vi. Transaction Control - SQL includes commands for specifying the beginning and ending of transactions. Several implementations also allow explicit locking of data for concurrency control.

4.2.1 Basic Structure Of SQL Statement

Basic structure of an SQL expression consists of 3 clauses;

- i. SELECT
- ii. FROM
- iii. WHERE

SELECT

This corresponds o a projection operation of the relational algebra. Its used to list the attributed desired in the result of a query.

FROM

This corresponds to a Cartesian product operation of the relational algebra. It lists the relations to be scanned in the evaluation of the expression

WHERE

Corresponds to the predicate of the relational algebra. It consist of a predicate involving attributes of the relations that appear in the **FROM** clause.

A typical SQL query will be of the form:

SELECT

A₁, A₂, A₃,A_n

FROM

R₁, R₂, R₃,R_n

WHERE

P

A_i represents an attribute; each r a relation and P is a predicate.

Select clause

Examples (i) **SELECT** Branch name
FROM Loan

(ii) **SELECT DISTINCT** Branch-name
FROM Loan

The symbol * can be used to denote all attributes of a given relation

(iii) **SELECT** *
FROM Loan

STUDENT

<u>Code</u>	<u>Stud.id</u>	<u>Name</u>
IMIS	001	Charles
BIT	002	Mary
BIT	003	Maina
CIT	004	Judy

COURSE

<u>Code</u>	<u>Title</u>
IMIS	Info. Systems
BIT	Bachelor of IT
CIT	Cert in IT
DIT	Dip in IT

Select Stud-Id, Name, Code, Title

From Student, Course

Where Student.Code = Course.Code

The select clause can also contain arithmetical expressions involving operations +, -, *, and operating on constants or attributes of tables e.g.

SELECT Branch_name, Loan_number, Amount*100
FROM loan

Where Clause

Specifies a condition that has to be met. SQL uses the logical connectives AND, OR and NOT in the where clause. It also uses operands of logical connectives <, <=, >, >=, = and <>. It also includes a BETWEEN operations e.g.

(i) **Select** loan_number
From loan

(ii) **Select** loan_number
From loan

Where branch_name = "River Road" and Amount **Between** 10,000 **And** 15,000.

From Clause

This specifies the source (relations), which is a Cartesian product. The SQL uses the notion relation-name. Attribute-name to avoid ambiguity in case where an attribute appears in the schemer of more that one relation e.g.

Example

```
Select Customer_name, borrower. loan number  
From borrower, loan  
Where borrower.loan_number = loan.loan_number  
AND branch_name= "Moi Avenue"
```

This will return the name of the customer the loan-number is the customer loan no. appears in Moi Avenue.

SQL provides a mechanism for renaming both relations and attributes by use of the As clause it is of the form

Old_name **AS** New_name. e.g.

```
Select distinct Customer_name, Borrower. Loan_number AS loan_Id  
From Borrower, loan  
Where Borrower. Loan_number = loan.loan_number  
AND Branch_name = "Koinange Street"
```

Ordering Display of Tuples

The "order by" clause case the tuples in the result for a query to appear in sorted order e.g.

```
Select distinct Customer - name  
From borrower, loan  
Where borrower.loan_number = loan.loan_number  
And Branch name = "University way"  
Order by customer_name
```

By default the order by clause lists items in ascending order. To specify the sort order use 'desc' for descending order or 'asc' for ascending e.g.

```
Select *  
From loan  
Order by amount desc, loan-number desc
```

4.2.2 Aggregate Functions

These are functions that take a collection (set or multi-set) of values as input and return a single value. These are

Average: **Avg**
Minimum: **Min**
Maximum: **Max**
Total: **Sum**
Count: **Count**

The input to sum and average must be a collection of numbers but the other operators can operate on collection of non-numeric data-types e.g. strings

Example

- (i) **SELECT** Branch name, Avg(balance)
 FROM Account
 GROUP BY Branch -name
- (ii) **SELECT** Branch_name, count (**distinct** customer_name)
 FROM Depositor, account
 WHERE Depositor, account-number = account - number
 GROUP BY Branch name
- (ii) **SELECT** Branch_name, Avg(balance)
 FROM Account
 GROUP BY Branch_name
 HAVING Average (balance) > 1200

Null Values

Null values indicate absence of information about the value of an attribute. e.g.

SELECT loan-number
FROM loan
WHERE Amount is **Null**

Assignment: look into Inner Join and Outer Join

4.2.3 Tuple Variables

- A tuple variable in SQL must be associated with a particular relation
They are defined in the FROM clause via the use of the AS clause. e.g.

SELECT DISTINCT Customer_name, T.loan_number
FROM Borrower **AS** T, loan **AS** S
WHERE T.loan_number = S.Loan_number

Query to find the names of all branches that have assets greater than at least one branch located in Brooklyn would be.

```
SELECT Distinct T.Branch_name  
FROM Branch AS T, Branch AS S  
WHERE T.assets > S.assets AND S.Branch_city = "BROOKLYN"
```

When expressions of the form relation_name.Attribute_name are written, the relation name is an implicitly defined tuple variable.

4.2.4 String Operations

- Most commonly used operation on strings is pattern matching using "LIKE".
- Two characters are used
 - Percent (%) - matches any sub-string
 - Underscore (-) - matches any character
- Patterns are case sensitive i.e. uppercase do not match lower case characters.

Examples

- (i) "Mary %" matches any string beginning with "Mary"
- (ii) "%ry" Matches any string containing "ry" as a sub-string e.g. very, mary, ary etc.
- (iii) "- - -" Matches any string of exactly three characters.
- (iv) "- - %" Matches any string of at least 3 characters.

The query to find customer names for all customers whose addresses include the sub-string "main" would be:-

```
SELECT Customer-name  
FROM Customer  
WHERE Customer -street LIKE "%main %"
```

For patterns to include special pattern characters (i.e. % and _) SQL allows the specification of an escape character. The escape character is placed immediately before a special pattern character to indicate the special pattern. Character is to be treated like a normal character. The key work **ESCAPE** is used.

Examples.

- **LIKE** "ab\%cd%" **ESCAPE** "\" - matches all strings beginning with "ab%cd"
- **LIKE** "ab\\cd%" **ESCAPE** "\" - matches all strings beginning with "ab\cd"

Mismatches.

SQL allows the search for mismatches using the **NOT LIKE** comparison operator Set Operations.

4.2.5 SQL and Set

SQL operations **Union**, **Intersect** and **Except** operate on relations and correspond to the relational operations \cup , \cap and $-$,

(i) Union

To find all customers having a loan, an account or both at the bank

```
(SELECT Customer_name FROM depositor)
UNION
(SELECT Customer_name
FROM Borrower)
```

To indicate duplicates

```
(SELECT Customer_name FROM Depositor)
UNION ALL
(SELECT Customer_name
FROM Borrower)
```

(ii) The Intersection

To find customers who have both a loan and an account at the bank

```
(SELECT Distinct Customer_name
FROM Depositor)
INTERSECT
(SELECT Distinct Customer_name
FROM Borrower)
```

To include duplicates we use “intersect all”

(iii) The Exception

To find customers who have an account but no loan at the bank we write

```
(SELECT Distinct Customer_name FROM Depositor)
EXCEPT
(SELECT Customer_name
FROM Borrower)
```

To include duplicate we use “Except all”

Null Values

- The keyword is used in the predicate test.

Example

```
SELECT Loan_number  
FROM Loan  
WHERE Amount is NULL
```

- To test for the absence of a null value we use the predicate “IS NOT NULL”

4.4.6 VIEWS

Use **CREATE VIEW** command

Syntax

```
CREATE VIEW V AS <query expression>  
Where query expression is a legal query expression.
```

Example

```
CREATE VIEW Customer AS  
(SELECT Branch_name, Customer_name  
FROM Depositor.account)  
WHERE Depositor.Account_number, Account.account_number
```

The names of the attribute of a view can be specified as

```
CREATE VIEW Branch_total_loan(branch-name, total(loan)  
AS  
SELECT Branch_name, SUM (amount)  
FROM loan  
GROUP BY Branch_name
```

NB: A create view clause creates a view definition in the database which stays there until a command **DROP View** (view name) is executed.

4.4.7 Modification Of The Database

Involves **Add**, **REMOVE** or **CHANGE** of information in the database.

(i) Deletion

```
DELETE FROM r  
WHERE P
```

- P represents the predicate, r represent the relation.
- The statement first finds all tuples t in r which P(t) is true & then deletes them from r
- Where clause can be omitted in which case all tuples in P are deleted.

Example

DELETE FROM Loan

- Deletes all tuples from the loan relation.

To delete all loans with loan amounts between 1300 & 1500

DELETE FROM loan

WHERE amount **BETWEEN** 1300 **AND** 1500

To delete all accounts at city square branch

DELETE FROM account

WHERE Branch-name = "City Square"

(ii) Insertion

To insert data into a relation:-

- Specify a tuple to be inserted or
- Write a query whose result is a set of tuples to be inserted

Tuples to be inserted must be in the correct arity.

Example

INSERT INTO Account

VALUES ("City Square", "Account", 6000)

or

INSERT INTO Account (branch-name, account-number, balance)

VALUES ("City Square", "Account", 6000)

(iii) Updates

To change a value in a tuple without changing all values the **UPDATE** statement can be used.

Examples

(i) **UPDATE** Account

SET Balance = Balance * 1.05

(ii) **UPDATE** Account

SET Balance = Balance * 1.06

WHERE balance > 10,000

Update Of A View

- A modification is permitted through a view only if the view in question is defined in terms of one relation of the actual relational database i.e. of a logical level db

Example

```
CREATE VIEW Branch_loan AS  
SELECT Branch_name, loan_number  
FROM loan  
INSERT INTO Branch_loan  
VALUES ("Moi Avenue", "Accoo8")
```

4.2.7 Schema Definition in SQL

Syntax

```
CREATE TABLE r(A1D1, A2D2, -----, AnDn,  
                [Integrity Constraints],  
                .....  
                .....  
                .....  
                [Integrity - constraints])
```

Examples

- (i) **CREATE TABLE** Customer
 (Customer_name **CHAR**(20) **NOT NULL**,
 Customer_street **CHAR**(30),
 Customer_city **CHAR**(30),
 PRIMARY KEY (customer_name))
- (ii) **CREATE TABLE** Branch
 (Branch_name **CHAR** (15) **NOT NULL**,
 Branch_city **CHAR** (30),
 Assets Integer,
 PRIMARY KEY (Branch_name)
 Check (assets >= 0))
- (iii) **CREATE TABLE** Depositor
 (customer_name, **CHAR**(20) **NOT NULL**,
 Account_name **CHAR**(20) **NOT NULL**,
 PRIMARY KEY (Customer_name, Account_number))

The create table commands includes other integrity constraints.

- Primary key - includes a list of the attributes that constitute the primary key
- Unique - includes a list of the attributes that constitute a candidate key
- Foreign key - includes both a list of the attributes that constitute the foreign key & the name of the relation referenced by the foreign key.