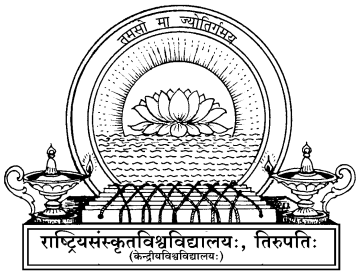
Computer Application

Sastri/ B.A 3rd YEAR
Course/Paper.4

PART – A

Data Base Management Systems



CENTER OF DISTANCE & ONLINE EDUCATION

(Formerly Directorate of Distance Education)

NATIONAL SANSKRIT UNIVERSITY :: TIRUPATI-517 507 (A.P)

(Erstwhile Rashtriya Sanskrit Vidyapeetha, Tirupati)

Unit – I

Basic concepts of DBMS

Unit – I Basic concepts

Structure

Chapter – 1

1.0. Objectives

The main objective of this unit is to make clear about what are data and DBMS.

1.1. Introduction

This chapter deals with DBMS, schemas at different levels.

1.2. Basic concepts

1.2.1. Database: A database is a collection of data. That may sound overly simplistic but it pretty much sums up what any database is.

A database could be as simple as a text file with a list of names. Or it could be as complex as a large, relational database management system, complete with in-built tools to help you maintain the data.

1.2.2. DBMS: DBMS stands for Data Base Management System. DBMS is a collection of interrelated data and set of programs to access those data. The primary goal of DBMS is to provide an environment that is both convenient and efficient to use in retrieving and storing database information.

The following are the advantages DBMS over traditional file processing system:

1. **Controlled data Redundancy**: Data redundancy means information appears in different files in different formats. This leads to repetitive of information and waste storage space. Files that represent same kind of information may become inconsistent. In database approach 'views' of different item is stored in only one place and does not permit any inconsistency. Thus database controls redundancy by saving the storage space and not allowing repetitive of information.
2. **Integrity can be maintained**: the data values stored in the database must satisfy certain types of integrity constraints, for example the balance of bank account should not fall below a prescribed amount say Rs.300/-. DBMS enforces these constraints through various application programs, thus DBMS improves data integrity which is accurate consistent and up to date.
3. **Providing backup and recovery**: because a computer system is an electronic device and it may subject to failure. It is crucial to ensure that once a failure has occurred and has been removed the data are restored to the consistent state that existed prior to the failure. DBMS provides backup and recovery occur in the system. For example if the computer fails during update operation and when it recovers backup and recovery system ensure that the database is in prior state(i.e., before execution of update operation).
4. **Maintaining security**: every user of database system is not allowed to access database. The database administrator (DBA) defines authorized users to access the database. DBA specifies different security rules in accessing database for different users.
5. **Sharing of data**: Sharing of data means the new applications can share the existing data from the current database. This reduces additional storage space.

6. **Enforcement of standards**: Standardization in representing the data is desirable in migration or interchange of data between the systems. DBA ensures that all applicable standards are observed in data representation.
7. **providing multiple user interfaces:** DBMS provides various types of user interfaces for different users depending on technical knowledge/ these include query languages for casual users, programming language interfaces for application programmers, forms and commands for parametric users and menu driven interfaces for ordinary users.
8. **Flexibility**: It may be necessary to change the structure of database due to changes in requirements. For example a user needs the information that is currently not available in the database DBMS adds new file to the database or extend the data fields in existing file without affecting so stored data and application programs.

S.Q.A.1. Define DBMS with suitable example.

S.Q.A.1. What is consistency constraint?

1.2.3. Data independence

Data independence is defined as the capacity to change the schema at one level of database system without affecting the schema in the next higher level.There are two types of data independence

a. **Logical Data Independence:** It is the ability to change the conceptual schema without having to change external schemas or application programs. The change in conceptual schema can expand the dataset by adding new data item or can reduce the structure of database by removing a data item. Changes to constraints can also be applied to the constraints can also be applied to the conceptual schema without effecting the external schemas. DBMS supports logical data independence by changing the view definition and mappings.

b. **Physical Data Independence:** It is the ability to change the internal schema without affecting conceptual or external schemas. Changes to the internal schema are necessary because of reorganization of physical files. It provides new access paths to the database in improving retrieval of information.

Summary

1. DBMS is a collection of interrelated data and set of programs to access those data. A dbms based on relational model is rdbms.
2. Data independence is defined as the capacity to change the schema at one level without effecting other two levels. This is of two types logical data independence and physical data independence.

I. Answer the following questions (10 marks)

1. Write short notes on DBMS.
2. Write short notes on DBA.
3. Write short notes data independence.

II. Answer the following questions (5 marks)

1. Define DBMS.
2. What is data independence?

III. Answer the following questions (1 mark)

1. A database is collection of _____(data)
2. _____ is a collection of interrelated data and set of programs to access those data. (DBMS).
3. _____ means information appears in different files in different formats.(Data redundancy)
4. The ability to change the conceptual schema without having to change external schemas or application programs is called _____(logical data independence)
5. The ability to change the internal schema without affecting conceptual or external schemas is called _____ (physical data independence)

Chapter – II Database System

Structure

2.0. Objective

2.1. Introduction

2.2. Architecture of Data base system

2.3. Data base administrator

2.4. Database languages

- 2.4.1. DDL
- 2.4.2. DML

2.0. Objective

The aim of this chapter is to clearly know about database system, administrator and languages.

2.1. Introduction

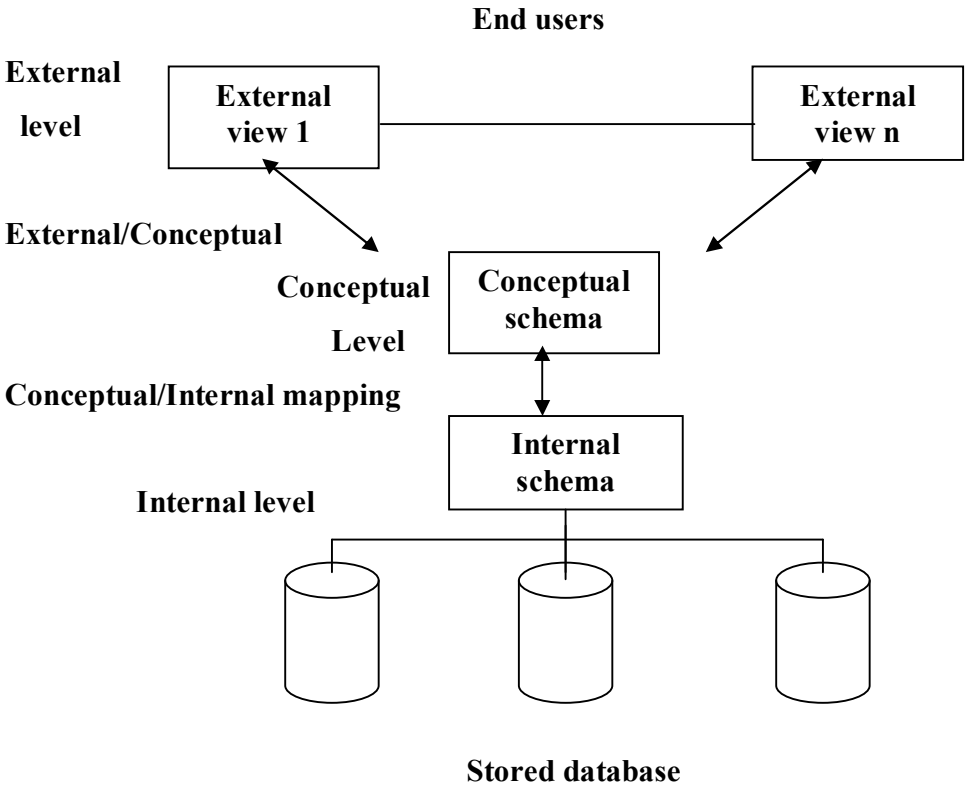
This chapter deals with Architecture of Data Base System, Data Base Administrator (DBA) and database languages like Data Definition Languages and Data Manipulation Languages.

2.2. Architecture of DBMS

The three important characteristics of DBMS are:

1. Insulation of programs and data.
2. Support of multiple use views and
3. Use of a catalog to store database system.

Thus architecture of database system is viewed as three level (schema) architecture.



Above is the architecture of ABMS.

1.Internal Level: The internal view is low-level representation of entire database. i.e., it describes physical storage structure of the database/ the internal level describes how the data are actually are stored in the database. Internal schema specifies the structure of data fields, the indexes and access paths for the database.

2.**Conceptual level:** The conceptual level has a conceptual schema. It describes what data are stored in the database and what relationships exist among those data. The conceptual schema hides details of physical storage structure and it concentrates only in logical structure of database. The conceptual level is use by database administrator who decides what information is to be kept.

3.**External or view level:** It includes number of external schemas or user views. Each external schema describes a part of database in which a particular use group is interested. It hides other part of database, which is not

required to the user group. Thus many users of database system can get required information without accessing entire database.

2.3. Database Administrator (DBA): The person who has the central control in defining the data and to write programs and to access the data in database is known Database Administrator.

The following are the functions of database administrator.

S.A.Q. What is the role of end user?

2. Defining conceptual schema: it is the job of DBA to decide what information is to be stored in the database. The DBA creates conceptual database schema using DDL compiler.

3. Defining internal schema: The DBA must decide how the data is to be represented in the database. This process is referring to as physical database design. DBA defines storage structure using internal DDL.

4. Liaising with users: It is the responsibility of DBA to liaise with users to ensure that the data they require is available and helps the users to write necessary external schemas using applicable external DDL. The mapping between external schema and conceptual schema must be defined by the DBA.

5. Granting of authorization for data access: The granting of different types of authorization allows the DBA to regulate which parts of the database various users con access.

6. Defining integrity rules: The data values stored in the database must satisfy certain consistency constraints. For example, the age of

an employee should not exceed 58 years such type of constraints must be specified explicitly by the dataset administrator.

7. Defining backup and recovery procedures: Once an enterprise committed to the database system it becomes critically dependent on the successfully operation of system. In the event of damages to the any portion of database it is the responsibility of DBA to rectify the errors in database. DBA must define an appropriate recovery procedures to dump (transfer) the data from damaged database to back storage space.

8. Monitoring performance and responding to changing requirements: DBA is responsible for organizing the system and improving the performance. DBA should provide appropriate adjustments to the system whenever requirement changes occur in the system. Whenever changes occur in the database, DBA must ensure that performance level is still acceptable.

S.A.Q.1. What is role of DBA?

2.4.Database Languages: A database system provides two different types of languages. They are Data Definition Language (DDL) and Data Manipulation Language (DML).

2.4.1. Data Definition Language (DDL): A database schema is specified by a set of definitions expressed by a language called as Bata Definition Language (DDL). The result of compilation of DDL Statements is a set of tables. The tables are stored in a special file called data directory or data catalog or data dictionary.

Data catalog is a file that contents meta data. The storage structure and access method used by the database system are specified by a set of definitions in a special type of DDL called a data storage and definition languages compiler. The result of this compiler is a set of instructions, which specify the implementation details database schema. The following are the examples for DDL in SQL

- **CREATE DATABASE** - creates a new database
- **ALTER DATABASE** - modifies a database
- **CREATE TABLE** - creates a new table
- **ALTER TABLE** - modifies a table
- **DROP TABLE** - deletes a table
- **CREATE INDEX** - creates an index (search key)
- **DROP INDEX** - deletes an index

2.4.2. Data Manipulation Language (DML): Manipulation of database includes retrieval of information stored in a database, insertion of new information into the database, deletion of information data, is known as language. There are two types of DML s and they are procedural DML and non-procedural DML. The procedural DML (low level DML) requires a user to specify what data are needed and how to get those data. Procedural DML is embedded in a general purpose programming language. This type of DML retrieves individual reports separately. The non-procedural DML (high level) requires a user to specify what data are needed without specifying how to get those data. Non-procedural DML specifies and retrieves many records in a single statement. This statement is known as query. The portion of a DML that involves information retrieval is known as query language. The following are the some of the example of SQL for DML

- **SELECT** - extracts data from a database
- **UPDATE** - updates data in a database
- **DELETE** - deletes data from a database
- **INSERT INTO** - inserts new data into a database

S.A.Q. What metadata?

Summary

1. Data independence is defined as the capacity to change the schema at one level of database system without affecting the schema in the next higher level.
2. The person who has the central control in defining the data and to write programs and to access the data in database is known DBA.
3. A database system provides two different types of languages. DDL and DML.
4. What data are needed and how to get those data is known as procedural DML and what data are needed without specify how to get those data is known as non-procedural DML.

I. Answer the following questions (10 marks)

1. Write short notes on DBA.
2. Write short notes on Architecture of DBMS.
3. Write short notes on Data base languages.

II. Answer the following questions (5 marks)

1. Define DDL.
2. Define DML.
3. What is data independence?
4. What is the architecture of DBMS?

III. Answer the following questions (1 mark)

1. The _____ defines authorized users to access the database (database administrator (DBA)).
2. _____ is a collection of interrelated data and set of programs to access those data. (DBMS).
3. _____ and _____ are two different types of database languages. (DDL , DML).
4. The architecture of database system is viewed as _____ level(schema) architecture. (three)
5. _____ means information appear4s in different files in different formats. (Data redundancy)

Unit – II

Database models

Chapter – I Data models

Structure

2.0. Objective

2.1. Introduction

2.2. Data models

- 2.2.1. Relational data model
- 2.2.2. Network model
- 2.2.3. Hierarchical model

2.0. Objective

The objective of this unit is to handle with different data models in database.

2.1. Introduction

In this chapter we deal with data models like relational, network, hierarchical models.

2.2. Data model

Underlying the structure of a database is the data model. A model is a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints. The various data models that have been proposed fall into three different groups.

1. Object based logical models
2. Record based logical models and
3. physical model

1. Object based logical model: Object based logical models are used in describing data at the logical and view levels. They are characterized by the fact that they provide fairly flexible structuring capabilities and allow data constraints to be specified explicitly. There are many different models. Several of the widely known ones are

1. The Entity relationship model
2. The Object-Oriented model
3. The Semantic data model
4. The Functional Data model

3. Record based logical models: Record based logical models are used in describing data at the logical and view levels. In contrast to object based data models, they are used both to specify the overall logical structure of the database and to provide a higher-level description of the implementation. The database is structured in fixed-format records of several types. Each record type defines a fixed number of fields, or attributes, and each field is usually of a fixed length. The three most widely accepted record-based data models are the relational, network, and hierarchical model.

2.2.1. Relational model: A database based on the relational model developed by E.F. Codd. A relational database allows the definition of data structures, storage and retrieval operations and integrity constraints. In such a database the data and relations between them are organized in tables. A table is a collection of records and each record in a table contains the same fields.

Properties of Relational Tables

- Values Are Atomic
- Each Row is Unique
- Column Values Are of the Same Kind
- The Sequence of Columns is Insignificant
- The Sequence of Rows is Insignificant

 Each Column Has a Unique Name

The following is an example for relational model

Customer table

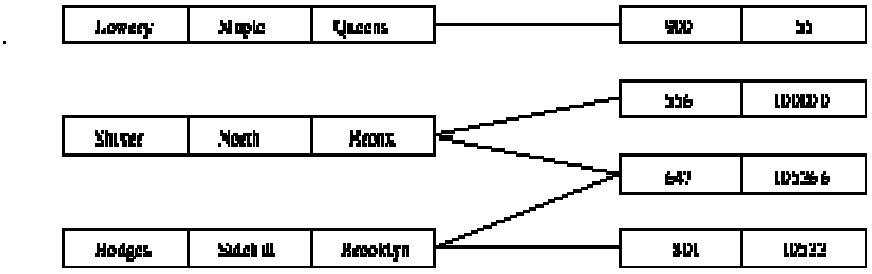
name	street	city	number
Lowery	Maple	Queens	900
Shiver	North	Bronx	556
Shiver	North	Bronx	617
Hodges	Sidehill	Brooklyn	901
Hodges	Sidehill	Brooklyn	617

account table

name	balance
900	55
556	100000
617	105366
901	10533

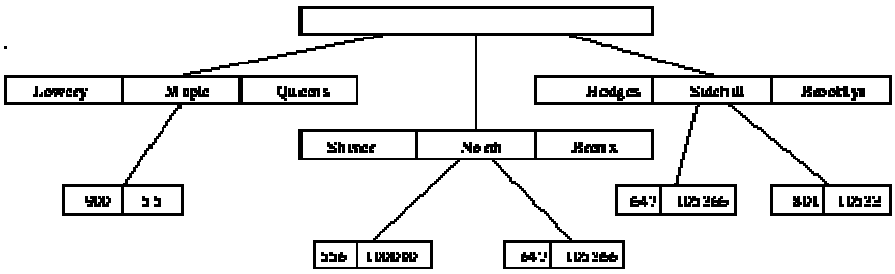
In the above example name, street, city, number are the columns for customer table and name, balance are columns for account table.

2.2.2. The Network Model: Data are represented by collections of records. Relationships among data are represented by links. Organization is that of an arbitrary graph. The following is an example for network database that is the equivalent of the relational database.



In the above example network model is an example for customer and account tables

2.2.3. The Hierarchical Model: Similar to the network model. Organization of the records is as a collection of trees, rather than arbitrary graphs. Hierarchical database that is the equivalent of the relational database.



The relational model does not use pointers or links, but relates records by the values they contain. This allows a formal mathematical foundation to be defined.

4. **Physical data model:** Physical data models are used to describe data at lowest level. In contrast to logical data models, there are few physical data models in use. Two of the widely known ones are the unifying model and the frame-memory model.

Summary

1. Data models are of three types object based logical model, record based logical model, and physical mode.

2. Object based logical models are used in describing data at the logical and view levels.

3. Object based logical mode again divided in the entity relationship model, the object-oriented model, the semantic data model, the functional data model.

4. Record based logical model are used in describing data at the logical and view levels and they are used both to specify the overall logical structure of the database.

5. In this model database structure in fixed-format records of several types.

6. Each record type defines a fixed number of fields, or attributes.

7. This model is further divided into relational model, network model and hierarchical model.

8. Physical data models are used to describe data at the lowest level.

I. Answer the following questions (10 marks)

1. Explain data modes.
2. Write short notes on Record based logical models.
3. Write short notes on Relation model.
4. Write short notes on Network model.
5. Write short notes on Hierarchical model.

II. Answer the following questions (5 marks)

1. What is Relation model?
2. What is Network model?
3. What is Hierarchical model?

III. Answer the following questions (1 mark)

1. A _____ is collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints. (model)

2. The entity-relationship model falls under _____ data model. (Object based logical model)

3. The _____ does not use pointers or links to relate records. (relational model)

4. Record – based data model is divided into _____, _____, and _____ (Relational, Network, and hierarchical model)

5. The semantic data model is example for _____ model. (Object based logical model)

Chapter – II E-R model

Structure

2.0. Objective

2.1. Introduction

2.2. Entity Relationship model

2.3. E-R model

2.3.1. ER diagram

2.0. Objective

The aim of this unit is to understand Entity-Relationship model.

2.1. Introduction

E-R model is falls under Object – based Logical Model which is a category of Data Model. In this chapter we study about what is E-R model and E-R diagrams how to implement.

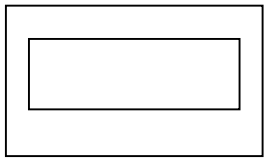
2.1. Entity relationship model: E-R model is a data model that describes data as entities, relationships and attributes.

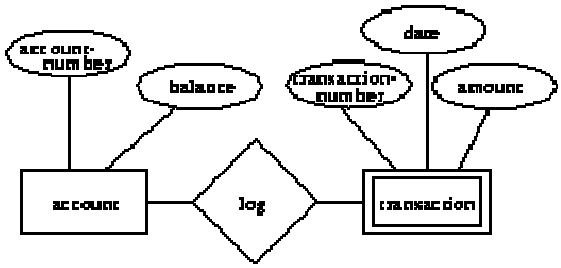
2.2. E-R model: The **E-R** (entity-relationship) data model views the real world as a set of basic objects (entities) and relationships among these objects. It is intended primarily for the DB design process by allowing the specification of an enterprise scheme. This represents the overall logical structure of the DB.

Entity: An entity is an object that exists and it is distinguishable from other objects. A distinction is accomplished by a set of attributes, which describe entity. A relationship is association among several entities. The relationship between two entity sets can be classified according to number of entities in one entity set that are associated with number of entities in another entity set.

Entity set: It is a set of Entities of same type. Rectangle is used to represent the Entity.

Weak Entity set: An entity set which doesn't have sufficient number of attributes to form a primary key is termed as weak entity set. It is represented with double lined rectangle.





Strong entity: An entity set which has which has a primary key is known as strong entity set.

Attributes: The properties of entities are called attributes. There are different types of attributes in the entity. For example, ROLL NO is a property of entity STUDENT. An attribute instance is a particular property of an individual entity instance. A key property uniquely identifies an entity instance .In the example below; a particular student can be uniquely identified from the roll number rather than the name. It is possible for more than one student to have the same number, but a roll number is never duplicated. A relationship can also have attributes. Ellipses are used to represent the attribute. There are seven types of attributes.

1. Simple or atomic attributes: An attribute which can be further divided into simple or atomic attributes.

2. Composite attributes: An attribute which can be further divided into simple attribute is called composite attribute.

3. Single value attribute: Attributes that have single value for a particular entity is called single value attribute.

4. Multi Value Attribute: Attribute that can have set of values for a particular entity are called multi value attributes.

5. Key attribute: An attribute which contains primary key is called key attribute.

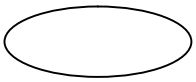
6. Null attribute: An attribute which doesn't have any value is called null attribute. Key attribute doesn't allow null values.

7. Derived attribute: An attribute which derives the value for other related attributes is known as derived attribute. For example, an employee's monthly salary is based on the employee's annual salary.

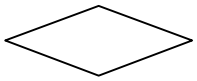
2.2.1. E-R diagram: The overall logical structure of database can be expressed graphically by the E-R diagram, which consists of following components.

- **Rectangles** representing entity sets.

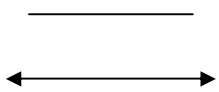
- **Ellipses** representing attributes.



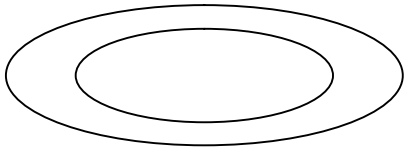
- **Diamonds** representing relationship sets.



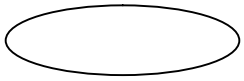
- **Lines** linking attribute to entity sets and entity sets to relationship sets.



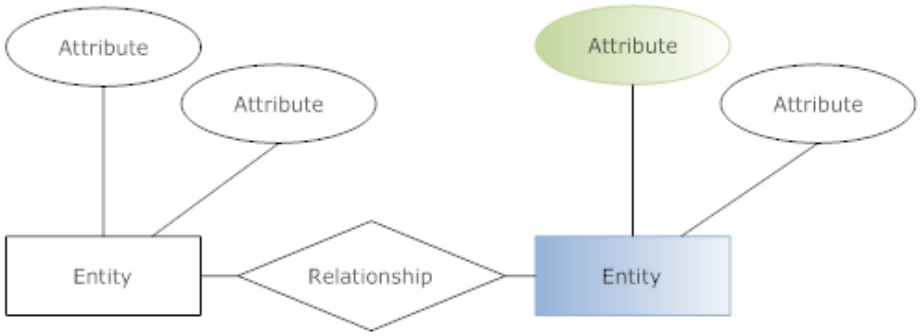
- **Double ellipses** which represent multi valued attributes.



- **Dashed ellipses** which represent derived attributes.



- **Double lines** which represent total participation of an entity in a relationship set.



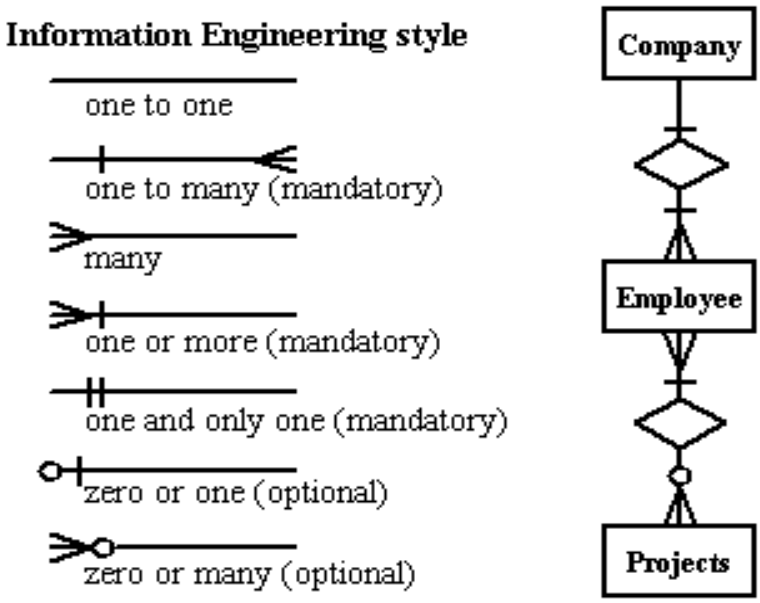
1. S.A.Q. Draw a E-R diagram for teacher and student data with attributes
T_NO, T_NAME, DEPT_NAME AND S_NO, S_NAME, SUBJECT with relation TEACH.

Relationship set: Relationship set is a set of relations of some type if E1, E2, E3, .En are entity sets then relationships set R C {e1, e2, e3, …, en} where (e1, e2, e3, …, en) are relationships.

Types of relationships: There are four types of relationships in DBMS which are as follows

1. One to one: An entity in A is associated with only one entity in another entity set B and Vice Versa. This is called as one to one relationship.
2. One to many: An entity in A is associated with number of entities in B and an entity in B is associated with only one entity in A. This is known as one to many relationships.
3. Many to many: An entity in A is associated with any number of entities in B and an entity in B is associated with any number of entities in A. This is known as many to many relationships.
4. Many to one: An entity in A is associated with at-most one number of entities in B is associated with any number of entities in A.

The following are the lines which is used to represent different relationships diagrammatically.



Specialization: It is a process of defining a set of sub classes of an entity type. This entity type is known as super class or higher level entity set. The sub classes are known as lower level entity set. In specialization the division of sub classes is defined on the basis of same distinguishing characteristics of entities in the super class. The refinement process follows top down approach

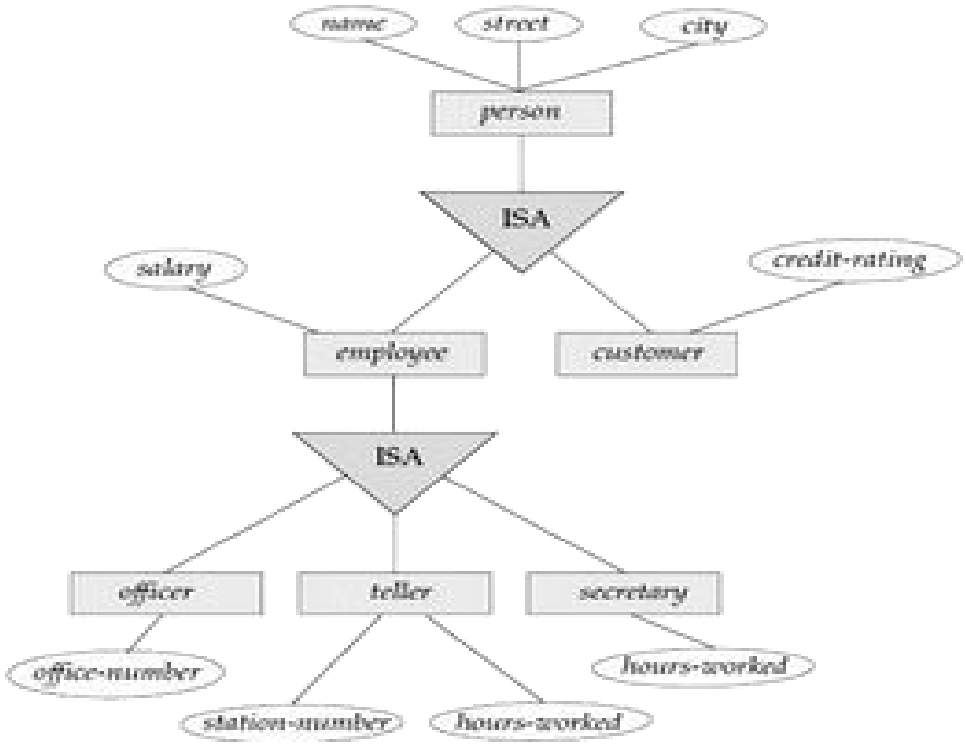
in deriving lower level entity sets. The entities of sub class may have attributes that are not shared by the entities of super class. In E-R diagram specialization is shown by ' ' triangle symbol and is labeled as 'Is A'( ) . The triangle with 'Is' a refers as relationship between super class and sub class.

Consider the following example. Consider the entity set 'account' in banking enterprise

The entity account has attributes accno and bal. Account is divided into two lower level entity sets savings_acc and current_acc respectively. Savings_acc entity has special attribute, interest_rate. Checking_acc hav over_draft attribute. Also both entity sets in common.

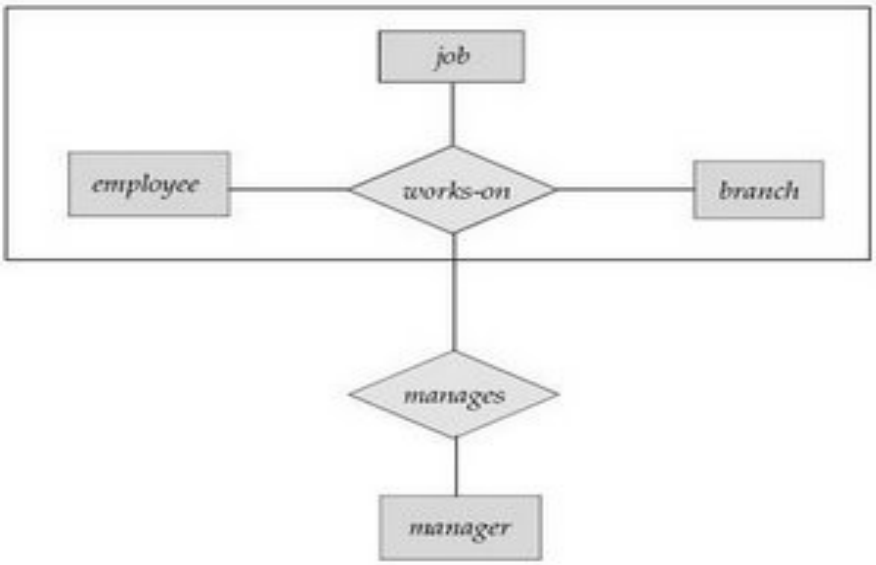
Generalization: The design process may also proceed in a bottom-up manner, in which multiple entity sets are synthesized into a higher-level entity set on the basis of common features. The database designer may have first identified a customer entity set with the attributes name, street, city, and customer-id, and an employee entity set with the attributes name, street, city, employee-id, and salary. There are similarities between the customer entity set and the employee entity set in the sense that they have several attributes in common. This commonality can be expressed by generalization, which is a containment relationship that exists between a higher-level entity set and one or more lower-level entity sets. In our example, person is the higher-level entity set and customer and employee are lower-level entity sets. Higher- and lower-level entity sets also may be designated by the terms super class and subclass, respectively. The person entity set is the super class of the customer and employee subclasses. For all practical purposes, generalization is a simple inversion of specialization. We will apply both processes, in combination, in the course of designing the E-R schema for an enterprise. In terms of the E-R diagram itself, we do not distinguish between specialization and generalization. New levels of entity representation will be distinguished (specialization) or synthesized (generalization) as the design schema comes to express fully the database application and the user requirements of the

database. Differences in the two approaches may be characterized by their starting point and overall goal. Generalization proceeds from the recognition that a number of entity sets share some common features (namely, they are described by the same attributes and participate in the same relationship sets).



Aggregation:

Aggregation is an abstraction in which relationship sets (along with their associated entity sets) are treated as higher-level entity sets, and can participate in relationships.



Summary

1. E-R model used to describes data as entities, relationships and attributes.
2. An entity is an object that exists and it is distinguishable from other objects
3. A relationship is association among several entities.
4. Attribute is the property of an entity.
5. Attributes are of seven types.
6. An entity set which doesn't have sufficient number of attributes to form a primary key is termed as weak entity set.
7. One-to-one, one-to-many, many-to-one, many-to-many are the four types of relationships.
8. Specialization is a process of defining a set of sub classes of an entity type.
9. Generalization is reverse process of specialization. It processed in bottom up manner.
10. Aggregation is the process of providing relationship between relationships.

I. Answer the following questions (10 marks)

1. Write short notes on E-R model.
2. Write short notes on attributes.
3. Write short notes on relationship

II. Answer the following questions (5 marks)

1. Define Aggregation
2. Define Specialization.
3. Define Generalization.
4. What is E-R model?
5. What is E-R diagram?

III. Answer the following questions (1 mark)

1. _____ are functions that take a collection of values as input and return a single value.(Aggregate functions)
2. Entity sets in E – R diagram are represented as _____. (rectangles)
3. Attributes in E – R diagram are represented as _____. (Ellipses)
4. A _____ key is a column in the table whose purpose is to uniquely identify records from the same table. (Primary key)
5. A _____ key is used to store values of another table's primary key to describe the relationship between data from different tables. (Foreign key)

Unit – III

Data base design

Chapter – I Normalization

Structure

3.0. Objective

3.1. Introduction

3.2. Database Design

- 3.2.1. Normalization
- 3.2.2. 1NF
- 3.2.3. 2NF
- 3.2.4. 3NF
- 3.2.5. BCNF

3.0. Objective

The objective of this chapter is to design data base.

3.1. Introduction

Data base is designed based on Normalization that consists of 1NF, 2NF, 3NF and BCNF

3.2. **Database Design:** A carefully thought-out database design forms the foundation for future success. These links will help you plan your database designs to maintain performance and integrity through future growth.

3.2. Normalization: Normalization is the process of efficiently organizing data in a database. There are two goals of the normalization process: eliminating redundant data (for example, storing the same data in more than one table) and ensuring data dependencies make sense (only storing related data in a table). Both of these are worthy goals as they reduce the amount of space a database consumes and ensure that data is logically stored.

Normal Forms

The database community has developed a series of guidelines for ensuring that databases are normalized. These are referred to as normal forms and are numbered from one (the lowest form of normalization, referred to as first normal form or 1NF) through five (fifth normal form or 5NF). In practical applications, you'll often see 1NF, 2NF, and 3NF along with the occasional 4NF. But in our syllabus we deal with 1NF, 2NF, and 3NF and BCNF.

3.2.1. 1NF or first normalization: A relation is said to be in First Normal Form (1NF) if and only if each attribute of the relation is atomic. More simply, to be in 1NF, each column must contain only a single value and each row must contain the same columns.

Example:

The following table is in first normal form (1NF/1ST):

Project_Name (primary key field when combined with Employee_Name)
Employee_Name Emp_Hire_Date Project_Manager

Problems With Data Stored in 1NF

Data duplication: within the same project the project manager name will be duplicated. If the manager is replaced then all records for that project will need to be updated (cascade update related records). The result is more space is used for each record and processing time is consumed checking updates and performing updates. To add a project manager you should already have an employee in the project.

3.2.2. 2NF or second normalization: A relation is said to be in Second Normal Form (2NF) if and only in it satisfies following properties

a. the relation must be in 1NF

b. Every non-key attribute is fully functional dependent on the primary key of the relation.

Example:

Project_Name
Employee_Name

The two fields, together, create a unique record and are therefore the primary key of the table.

There are a few problems with this table, however. First, if we delete a project we lose employee names unless they are a member of another project. If the employee gets married and changes their last name then every occurrence of the person's name will need to be updated. Similarly, if someone decides to make a slight change to the project name then many records may have to be updated.

3.2.3. 3NF or Third normalization: A relation is said to be in 3NF if it satisfies following properties:

a. The relation must be in 2NF
b. the Non-key attribute is non transitively dependent on primary key i.e., there is non-functional dependency between non-key attribute.

Example:

Here is the classic employee table example:

Emp_SSN
Emp_Name
Street
City

State
Zip

Here we have a transitive dependency between Zip field and City and State fields - what ever that means. I guess it means they are a little dependent but no big deal. Since a zip will tell you what state and what city an address is in. Therefore, to make this table third normal form we need to split out city state and zip fields into a table of their own and then just have the zip in the employee table:

Emp_SSN
Emp_Name
Street
Zip

The zip tables is as follows:

Zip
City
State

The above two tables are now in third normal form.

3.2.4. Boyce-Codd Normal Form (BCNF)

- When a relation has more than one candidate key, anomalies may result even though the relation is in 3NF.
- 3NF does not deal satisfactorily with the case of a relation with overlapping candidate keys
- I.e. composite candidate keys with at least one attribute in common.
- BCNF is based on the concept of a determinant.

- A determinant is any attribute (simple or composite) on which some other attribute is fully functionally dependent.
- A relation is in BCNF is, and only if, every determinant is a candidate key.

Difference between BCNF and 3NF:

1. 3NF always tries for less join and dependency preservation properties. BCNF also tries for loss less join an dependency preservation properties.
2. In 3NF non-key attributes are non-transitively dependent on primary key. In BCNF all non-key attributes are dependent on super key
3. The design of 3NF may cause null valued attributes or repetition of attributes in some relation, thus it causes redundancy. In BCNF null value attributes or repetitions of attributes are not allowed and thus there is no problem of redundancy.
4. If a relation is in 3NF it need not be in BCNF. All BCNF relations are in 3NF.

Summary

1. Normalization is the process of efficiently organizing data in a database.

2. A relation is said to be in First Normal Form (1NF) if and only if each attribute of the relation is atomic.

3. A relation is said to be in Second Normal Form (2NF) if and only in it satisfies following properties

a. The relation must be in 1NF

b. Every non-key attribute is fully functional dependent on the primary key of the relation.

4. A relation is said to be in 3NF if it satisfies following properties:

a. The relation must be in 2NF

b. The Non-key attribute is non transitively dependent on primary key i.e., there is non-functional dependency between non-key attribute

I. Answer the following questions (10 marks)

1. Explain Normalization.
2. List difference BCNF and 3 ^{NF}.

II. Answer the following questions (5 marks)

1. What is normalization?
2. What 1NF, 2NF, 3NF, BCNF.

III. Answer the following questions (1 mark)

1. BCNF stands for _____ (Boycee-Codd Normal Form

2. RDBMS stands for _____Relational Database Management System)

Chapter –II SQL

Structure

3.0. Objective

3.1. Introduction

3.2. SQL

- 3.2.1. Basic structure
- 3.2.2. Aggregate functions
- 3.2.3. Insert operation
- 3.2.4. Deletes operation
- 3.2.5. Updates operation

3.0. Objective

The objective of this Chapter is to learn SQL queries.

3.1. Introduction

In this chapter we discuss structure of SQL, Aggregate functions, Mathematical function, DDL and DML commands

3.2. SQL

- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- SQL is an ANSI (American National Standards Institute) standard
- SQL is not case sensitive

What Can SQL do?

- SQL can execute queries against a database

- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

Some database systems require a semicolon at the end of each SQL statement.

Semicolon is the standard way to separate each SQL statement in database systems that allow more than one SQL statement to be executed in the same call to the server.

We are using MS Access and SQL Server 2000 and we do not have to put a semicolon after each SQL statement, but some database programs force you to use it.

3.2.1. Basic structure

SQL DML and DDL

SQL can be divided into two parts: The Data Manipulation Language (DML) and the Data Definition Language (DDL).

The query and update commands form the DML part of SQL:

- **SELECT** - extracts data from a database
- **UPDATE** - updates data in a database
- **DELETE** - deletes data from a database
- **INSERT INTO** - inserts new data into a database

The DDL part of SQL permits database tables to be created or deleted. It also define indexes (keys), specify links between tables, and impose constraints between tables. The most important DDL statements in SQL are:

- **CREATE DATABASE** - creates a new database
- **ALTER DATABASE** - modifies a database
- **CREATE TABLE** - creates a new table
- **ALTER TABLE** - modifies a table
- **DROP TABLE** - deletes a table
- **CREATE INDEX** - creates an index (search key)
- **DROP INDEX** - deletes an index

Table creation

Tables are the most fundamental logical storage structures in the oracle relational database. Tables cancan be viewed as two-dimensional arrays that can contain a predetermined number of columns and multiple rows:

Before a table is created the following has to finalize:

1. The name of the table.

2. The number of columns in the table.

3. The name, data type and maximum length of each column of the table.

4. The column and table constraints.

Naming rules

1. The name must begin with an alphabet.

2. Digits and special characters, underscored (_),$ and # are allowed.

3. Maximum length is of 30 characters.

4. It must not be a reserved word.

5. There should not be any other object with the same name in your account.

Syntax:

Create table tablename (columnname datatype(datasize), columnname datatype(datasize), …. columnname datatype(datasize);

Example:

1. Create a table employee table with columns e_no, e_name, sal, deptno, doj.

SQL> create table employee(e_no number(4), e_name varchar2(20), sal number(4,3),deptno number(3), doj date);

S.Q.A. 1. Create a student table with the required details.

S.A.Q. 1. Create a patent table with the required details.

Describing table with is already created

The table can be described using SQL command describes abbreviated as desc.

Syntax:

desc tablename;

Example

desc employee;

output:

Name	Null	Type
e_no		Number(4)
e_name	Not null	Varchar2(20)
sal		Number(4,2)
deptno		Number(3)
doj		date

S.A.Q. 1. Write a query to describe table employee.

S.A.Q. Write a query to crate a table dept and describe that table

Integrity constraints

A constraint is a rule that restricts the data values for one or more columns in a table.

Constraints are of two types.

1. Table constraints

2. Column constraints

Table constraint: A constraint given at the table level is called as table constraint. It may refer to more that one column of the table. Primary dye constraint used to define composite key is the example of table level constraint.

Column constraint: A constraint given at the column level is known as column level constraint. It defines a rule for a single column. It cannot refer to column other than the column at which it is defined.

1. Not null: The particular column with a not null value can never have a null value.

2. Unique: cannot have two same values in a column. It enforces uniqueness in the given column.

3. Primary key: A primary key is used to uniquely identify rows in the table. There may be only one primary key in the table. It may consist of more than one column. It is used to enforce uniqueness in the primary key. Oracle does the following for the column that has primary key constraint.

- Creates a unique index to enforce uniquness.
- Define not null constraint to prevent null values.

Syntax for column constraint

Create table tablename (columnname datatype(datasize), columnname datatype(datasize)constraintname constraint, …. columnname datatype(datasize);

Example for column constraint

Create table std(s_no number(4), s_name varchar2(20), marks number(3) s_pk primary key);

4. Check constraint: it is used to specify some condition of the column.

The condition may not be

- A reference to pseudo column sysdate.
- Sub query

If it is given as column constraint it can refer only to current column. But if it given as table constraint, it can refer to more than one column of the table.

Syntax

Check columnname condition .

Example:

Check balance>=1000 in the bank

where 1000 is the minimum balance required.

S.A.Q. Write a query to create a table patent that implement constraints.

3.2.2. Aggregate functions

. For example, the purchasing manager may not be interested in a listing of all widget sales, but may simply want to know the number of widgets sold this month. Fortunately, SQL provides **aggregate functions** to assist with the summarization of large volumes of data.

The following are the useful aggregate functions:

- AVG() - Returns the average value
- COUNT() - Returns the number of rows
- MAX() - Returns the largest value
- MIN() - Returns the smallest value
- SUM() - Returns the sum

OrderID	FirstName	LastName	Quantity	UnitPrice	Continent
122	John	Jacob	21	4.52	North America
923	Ralph	Wiggum	192	3.99	North America
238	Ryan	Johnson	87	4.49	Africa
829	Mary	Smith	842	2.99	North America
824	Elizabeth	Marks	48	3.48	Africa
753	James	Linea	9	7.85	North America
942	Alan	Jonas	638	3.29	Europe

Let's begin by taking a look at the SUM function. It is used within a SELECT statement and, predictably, returns the summation of a series of values. If the widget project manager wanted to know the total number of widgets sold to date, we could use the following query:

```
SELECT  SUM (Quantity) AS Total
FROM WidgetOrders
```

Our results would appear as:

Total =1837

The AVG (average) function works in a similar manner to provide the mathematical average of a series of values. Let's try a slightly more complicated task this time. We'd like to find out the average dollar amount of all orders placed on the North American continent. Note that we'll have to multiply the Quantity column by the UnitPrice column to compute the dollar amount of each order. Here's what our query will look like:

```
SELECT AVG(UnitPrice * Quantity) As AveragePrice
FROM WidgetOrders
WHERE Continent = “North America”
```

And the results:

AveragePrice =862.3075

SQL provides the COUNT function to retrieve the number of records in a table that meet given criteria. We can use the COUNT(*) syntax alone to retrieve the number of rows in a table. Alternatively, a WHERE clause can be included to restrict the counting to specific records.

For example, suppose our Widgets product manager would like to know how many orders our company processed that requested over 100 widgets.

Here's the SQL query:

```
SELECT COUNT(*) AS 'Number of Large Orders'
FROM WidgetOrders
WHERE Quantity > 100
```

And the results:

```
Number of Large Orders
----------
3
```

The COUNT function also allows for the use of the DISTINCT keyword and an expression to count the number of times a unique value for the expression appears in the target data. Similarly, the ALL keyword returns the total number of times the expression is satisfied, without worrying about unique values. For example, our product manager would like a simple query that returned the number of unique continents in our orders database.

First, let's take a look at the use of the ALL keyword:

```
SELECT COUNT(ALL Continent) As 'Number of Continents'

FROM WidgetOrders
```

And the result set:

```
Number of Continents
--------------------
7
```

Obviously, this is not the desired results. If you recall the contents of the WidgetOrders table from the previous page, all of our orders came from North America, Africa and Europe. Let's try the DISTINCT keyword instead:

```
SELECT COUNT(DISTINCT Continent) As 'Number of Continents'
FROM WidgetOrders
```

And the output:

Number of Continents

3

In this final segment of our aggregate functions feature article, we'll look at the functionality SQL provides to locate the records containing the smallest and largest values for a given expression.

The MAX() function returns the largest value in a given data series. We can provide the function with a field name to return the largest value for a given field in a table. MAX() can also be used with expressions and GROUP BY clauses for enhanced functionality.

Once again, we'll use the WidgetOrders example table for this query (see the first page of this article for the specification and contents). Suppose our product manager wanted to find the order in our database that produced the most revenue for the company. We could use the following query to find the order with the largest total dollar value:

```
SELECT MAX(Quantity * UnitPrice)As 'Largest Order'
FROM WidgetOrders
```

Our results would look like this:

Largest Order

2517.58

The MIN() function functions in the same manner, but returns the minimum value for the expression. Let's try a slightly more complicated example

utilizing the MIN() function. Our sales department is currently analyzing data on small widget orders. They'd like us to retrieve information on the smallest widget order placed on each continent. This requires the use of the MIN() function on a computed value and a GROUP BY clause to summarize data by continent.

Here's the SQL:

```
SELECT Continent, MIN(Quantity * UnitPrice) AS 'Smallest Order'
FROM WidgetOrders
GROUP BY Continent
```

And our result set:

Continent	Smallest Order
Africa	167.04
Europe	2099.02
North America	70.65

S.Q.A. Write a query to implement all aggregate function on emp table.

3.2.3. Insert

Appends a new record to the end of a table that contains the specified field values. The **INSERT** SQL command has three syntaxes:

- Use the first syntax to insert specified values into specified fields in a table.
- Use the second syntax to insert the contents of elements from an array, memory variable, or property of an object that match the field names in the table.
- Use the third syntax to insert rows from an SQL **SELECT** command into the specified fields in the table.

INSERT INTO dbf_name [(FieldName1 [, FieldName2, ...])]
VALUES (eExpression1 [, eExpression2, ...])

INSERT INTO dbf_name FROM ARRAY ArrayName | FROM MEMVAR | FROM

NAME ObjectName

INSERT INTO dbf_name [(FieldName1 [, FieldName2, ...])]
SELECT SELECTClauses [UNION UnionClause SELECT SELECTClauses ...]

Parameters

INSERT INTO dbf_Name

Specifies the name of the table for appending a new record. dbf_Name can include a path and can be a name expression.

[(FieldName1 [, FieldName2 [, ...]])]

Specifies the names of the fields in the new record into which the values are inserted.

VALUES (eExpression1 [, eExpression2 [, ...]])

Specifies the field values to be inserted into the new record. If you omit the field names, you must specify the field values in the order defined by the table structure. If eExpression is a field name, it must include the table alias.

If **SET NULL** is **ON**, **INSERT** attempts to insert null values into any fields not specified in the **VALUES** clause.

Guidelines for insert command

1. A table must be created before you insert the data.

2. Separate the values by commas.

3. Enclose the strings in single quotes ‘ ‘ .

4. Enter the values in specific sequence, which is sequence of the columns at the time of creation of table.

5. Each value must match the data type of the particular column.

6. Date must be entered in the format dd-mmmm-yy.

7. If you don’t want to enter values for all the columns then specify the column names.

S.A.Q. 1.write a query to insert values in the employee table.(at least five row should be inserted)

Select statement: Select command retrieves the rows from the tables. It implements operators of relational algebra such as projections and selection.

The simplest select command contains the following:

if * is given, all the columns are selected.

The SELECT statement is used to select data from a database.

The result is stored in a result table, called the result-set.

SQL SELECT Syntax

```
SELECT column_name(s)
FROM table_name
```

and

```
SELECT * FROM table_name
```

An SQL SELECT Example

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Now we want to select the content of the columns named "LastName" and "FirstName" from the table above.

We use the following SELECT statement:

```
SELECT LastName,FirstName FROM Persons
```

The result-set will look like this:

LastName	FirstName
Hansen	Ola
Svendson	Tove
Pettersen	Kari

SELECT * Example

Now we want to select all the columns from the "Persons" table.

We use the following SELECT statement:

```
SELECT * FROM Persons
```

Tip: The asterisk (*) is a quick way of selecting all columns!

The result-set will look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

S.A.Q. 1.write a query to select values in the employee table.

2. Write a query to select only address the above(person) table.

3.2.3. Delete

The DELETE statement is used to delete records in a table.

The DELETE Statement

The DELETE statement is used to delete rows in a table.

SQL DELETE Syntax

```
DELETE FROM table_name
WHERE some_column=some_value
```

Note: Notice the WHERE clause in the DELETE syntax. The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

SQL DELETE Example

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger
5	Tjessem	Jakob	Nissestien 67	Sandnes

Now we want to delete the person "Tjessem, Jakob" in the "Persons" table.

We use the following SQL statement:

```
DELETE FROM Persons
WHERE LastName='Tjessem' AND FirstName='Jakob'
```

The "Persons" table will now look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger

Delete All Rows

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:

```
DELETE FROM table_name
or
DELETE * FROM table_name
```

S.A.Q. 1. Write a query to delete all rows in employee table.

3.2.5. Updates:

The UPDATE statement is used to update records in a table.

The UPDATE Statement

The UPDATE statement is used to update existing records in a table.

SQL UPDATE Syntax

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

Note: Notice the WHERE clause in the UPDATE syntax. The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

SQL UPDATE Example

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger
5	Tjessem	Jakob		

Now we want to update the person "Tjessem, Jakob" in the "Persons" table.

We use the following SQL statement:

```
UPDATE Persons
SET Address='Nissestien 67', City='Sandnes'
WHERE LastName='Tjessem' AND FirstName='Jakob'
```

The "Persons" table will now look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
4	Nilsen	Johan	Bakken 2	Stavanger
5	Tjessem	Jakob	Nissestien 67	Sandnes

SQL UPDATE Warning

Be careful when updating records. If we had omitted the WHERE clause in the example above, like this:

```
UPDATE Persons
SET Address='Nissestien 67', City='Sandnes'
```

The "Persons" table would have looked like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Nissestien 67	Sandnes
2	Svendson	Tove	Nissestien 67	Sandnes
3	Pettersen	Kari	Nissestien 67	Sandnes
4	Nilsen	Johan	Nissestien 67	Sandnes
5	Tjessem	Jakob	Nissestien 67	Sandnes

S.A.Q. 1. Write a query to update person table with column pin and also row to implement.

ALTER TABLE Statement

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

ALTER TABLE Syntax

To add a column in a table, use the following syntax:

```
ALTER TABLE table_name
ADD column_name datatype
```

To delete a column in a table, use the following syntax (notice that some database systems don't allow deleting a column):

```
ALTER TABLE table_name
DROP COLUMN column_name
```

To change the data type of a column in a table, use the following syntax:

```
ALTER TABLE table_name
ALTER COLUMN column_name datatype
```

ALTER TABLE Example

Look at the "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Now we want to add a column named "DateOfBirth" in the "Persons" table.

We use the following SQL statement:

```
ALTER TABLE Persons
ADD DateOfBirth date
```

Notice that the new column, "DateOfBirth", is of type date and is going to hold a date. The data type specifies what type of data the column can hold. For a complete reference of all the data types available in MS Access, MySQL, and SQL Server, go to our complete Data Types reference.

The "Persons" table will now like this:

P_Id	LastName	FirstName	Address	City	DateOfBirth
1	Hansen	Ola	Timoteivn 10	Sandnes	
2	Svendson	Tove	Borgvn 23	Sandnes	
3	Pettersen	Kari	Storgt 20	Stavanger	

LIKE Operator

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

The LIKE Operator

The LIKE operator is used to search for a specified pattern in a column.

SQL LIKE Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name LIKE pattern
```

LIKE Operator Example

The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Now we want to select the persons living in a city that starts with "s" from the table above.

We use the following SELECT statement:

```
SELECT * FROM Persons
WHERE City LIKE 's%'
```

The "%" sign can be used to define wildcards (missing letters in the pattern) both before and after the pattern.

The result-set will look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Next, we want to select the persons living in a city that ends with an "s" from the "Persons" table.

We use the following SELECT statement:

```
SELECT * FROM Persons
WHERE City LIKE '%s'
```

The result-set will look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes

Next, we want to select the persons living in a city that contains the pattern "tav" from the "Persons" table.

We use the following SELECT statement:

```
SELECT * FROM Persons
WHERE City LIKE '%tav%'
```

The result-set will look like this:

P_Id	LastName	FirstName	Address	City
3	Pettersen	Kari	Storgt 20	Stavanger

It is also possible to select the persons living in a city that NOT contains the pattern "tav" from the "Persons" table, by using the NOT keyword.

We use the following SELECT statement:

```
SELECT * FROM Persons
WHERE City NOT LIKE '%tav%'
```

The result-set will look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes

Summary

1. In order to access the database, first database is to be started in many installations users are not concerned with either starting or shutting down database. It is taken care of by DBA. SQL*Plus is a program. Written by oracle corporation, using which user can enter SQL and SQL*Plus commands.

2. Create table is a DDL command to create a table definition DML command insert is used to add rows to a table. Select command is used to retrieve rows from the table**.**

I. Answer the following questions (10 marks)

1. Write short notes SQL queries.
2. Write short notes on create query with suitable example.
3. Write short notes on insert query with suitable example.
4. Write short notes on aggregate functions.
5. Write short notes on delete, update queries.
6. Write short notes on Alter query.
7. Write short notes on Update query.

II. Answer the following questions (5 marks)

1. What is SQL?
2. What are different DDL and DML statements?
3. What is the difference between delete table and drop table?
4. What is the difference between select and desc queries.
5. What is like query?

III. Answer the following questions (1 mark)

1. _____ command is used to display definition of a table. (desc)

2. SQL stands for _____ (Structure Query Language)

3. _____ functions are functions that take a collection of values a sinput and return a single value. (Aggregate functions)

Chapter – III SQL Mathematical functions

Structure

3.0. Objective

3.1. Introduction

3.2. Mathematical functions

3.0. Objective

The aim of this chapter is to know about SQL mathematical functions

3.1. Introduction

In this chapter we discuss mathematical functions such as like, upper, lower, round, trunc, initcap, etc.

3.2. Mathematical functions

UCASE() Function

The UCASE() Function
The UCASE() function converts the value of a field to uppercase.
SQL UCASE() Syntax

```
SELECT UCASE(column_name) FROM table_name
```

SQL UCASE() Example
We have the following "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Now we want to select the content of the "LastName" and "FirstName" columns above, and convert the "LastName" column to uppercase.
We use the following SELECT statement:

```
SELECT UCASE(LastName) as LastName,FirstName FROM Persons
```

The result-set will look like this:

LastName	FirstName
HANSEN	Ola
SVENDSON	Tove
PETTERSEN	Kari

LCASE() Function

The LCASE() Function
The LCASE() function converts the value of a field to lowercase.
SQL LCASE() Syntax

```
SELECT LCASE(column_name) FROM table_name
```

SQL LCASE() Example
We have the following "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Now we want to select the content of the "LastName" and "FirstName" columns above, and convert the "LastName" column to lowercase.
We use the following SELECT statement:

```
SELECT LCASE(LastName) as LastName,FirstName FROM Persons
```

The result-set will look like this:

LastName	FirstName
hansen	Ola
svendson	Tove
pettersen	Kari

MID() Function
The MID() Function
The MID() function is used to extract characters from a text field.
SQL MID() Syntax

```
SELECT MID(column_name,start[,length]) FROM table_name
```

Parameter	Description
column_name	Required. The field to extract characters from.
start	Required. Specifies the starting position (starts at 1).
length	Optional. The number of characters to return. If omitted, the MID() function returns the rest of the text.

SQL MID() Example
We have the following "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Now we want to extract the first four characters of the "City" column above.
We use the following SELECT statement:

```
SELECT MID(City,1,4) as SmallCity FROM Persons
```

The result-set will look like this:

SmallCity
Sand
Sand
Stav

LEN() Function

The LEN() Function
The LEN() function returns the length of the value in a text field.
SQL LEN() Syntax

```
SELECT LEN(column_name) FROM table_name
```

SQL LEN() Example

We have the following "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Now we want to select the length of the values in the "Address" column above.
We use the following SELECT statement:

```
SELECT LEN(Address) as LengthOfAddress FROM Persons
```

The result-set will look like this:

LengthOfAddress
12
9
9

ROUND() Function

The ROUND() Function
The ROUND() function is used to round a numeric field to the number of decimals specified.
SQL ROUND() Syntax

```
SELECT ROUND(column_name,decimals) FROM table_name
```

Parameter	Description
column_name	Required. The field to round.
decimals	Required. Specifies the number of decimals to be returned.

SQL ROUND() Example

We have the following "Products" table:

Prod_Id	ProductName	Unit	UnitPrice
1	Jarlsberg	1000 g	10.45
2	Mascarpone	1000 g	32.56
3	Gorgonzola	1000 g	15.67

Now we want to display the product name and the price rounded to the nearest integer.
We use the following SELECT statement:

```
SELECT ProductName, ROUND(UnitPrice,0) as UnitPrice FROM Persons
```

The result-set will look like this:

ProductName	UnitPrice
Jarlsberg	10
Mascarpone	33
Gorgonzola	16

NOW() Function

The NOW() Function
The NOW() function returns the current system date and time.
SQL NOW() Syntax

```
SELECT NOW() FROM table_name
```

SQL NOW() Example
We have the following "Products" table:

Prod_Id	ProductName	Unit	UnitPrice
1	Jarlsberg	1000 g	10.45
2	Mascarpone	1000 g	32.56
3	Gorgonzola	1000 g	15.67

Now we want to display the products and prices per today's date.
We use the following SELECT statement:

```
SELECT ProductName, UnitPrice, Now() as PerDate FROM Persons
```

The result-set will look like this:

ProductName	UnitPrice	PerDate
Jarlsberg	10.45	10/7/2008 11:25:02 AM
Mascarpone	32.56	10/7/2008 11:25:02 AM
Gorgonzola	15.67	10/7/2008 11:25:02 AM

FORMAT() Function

The FORMAT() Function

The FORMAT() function is used to format how a field is to be displayed.
SQL FORMAT() Syntax

```
SELECT FORMAT(column_name,format) FROM table_name
```

Parameter	Description
column_name	Required. The field to be formatted.
format	Required. Specifies the format.

SQL FORMAT() Example
We have the following "Products" table:

Prod_Id	ProductName	Unit	UnitPrice
1	Jarlsberg	1000 g	10.45
2	Mascarpone	1000 g	32.56
3	Gorgonzola	1000 g	15.67

Now we want to display the products and prices per today's date (with today's date displayed in the following format "YYYY-MM-DD").
We use the following SELECT statement:

```
SELECT ProductName, UnitPrice, FORMAT(Now(),'YYYY-MM-DD') as PerDate
FROM Persons
```

The result-set will look like this:

ProductName	UnitPrice	PerDate
Jarlsberg	10.45	2008-10-07
Mascarpone	32.56	2008-10-07
Gorgonzola	15.67	2008-10-07

Summary

1. In order to access the database, first database is to be started in many installations users are not concerned with either starting or shutting down database. It is taken care of by DBA. SQL*Plus is a program. Written by oracle corporation, using which user can enter SQL and SQL*Plus commands.

2. Create table is a DDL command to create a table definition DML command insert is used to add rows to a table. Select command is used to retrieve rows from the table**.**

I. Answer the following questions (10 marks)

1. Write short notes SQL queries.
2. Write short notes on create query with suitable example.
3. Write short notes on insert query with suitable example.
4. Write short notes on aggregate functions.
5. Write short notes on delete, update queries.
6. Write short notes on Alter query.
7. Write short notes on Update query.

II. Answer the following questions (5 marks)

1. What is SQL?
2. What are different DDL and DML statements?
3. What is the difference between delete table and drop table?
4. What is the difference between select and desc queries.
5. What is like query?

III. Answer the following questions (1 mark)

1. _____ command is used to display definition of a table. (desc)

2. SQL stands for _____ (Structure Query Language)

3. _____ functions are functions that take a collection of values a sinput and return a single value. (Aggregate functions)