

Unit-1

Introduction to DBMS

Data:

Data is a raw fact or unorganized form (such as alphabets, numbers, or symbols) that refers to or represent, condition, ideas or objects. Data is limitless and present everywhere in the universe.

Information:

- Data that has been verified to be accurate and timely.
- Is specific and organized for a purpose.
- Is represented within a context that gives it meaning and relevance.
- That can lead to an increase in understanding and decrease in uncertainty is called information.

Database:

A database is a collection of information that is organized so that it can easily be accessed managed and update. There are two type of database:

- **Conventional Database:**
Collection of data without use of DBMS. E.g. Telephone diary, register.
- **Computerized database:** The collection of data with the help of computer and DBMS in the form of data file is known as computerized database. The types of computer systems that can run database management system are Centralized PC and Client/Server and distributed.

Elements of Database:

1. **Field:** Smallest unit of Database.
2. **Record:** A Collection of multiple related fields.
3. **Table:** A collection of records or group of records with the row and column order.
4. **Tuple:** A record row in the database.
5. **Index:** It is a process of organizing data in specific order.
6. **Cell:** A cell is a inter section of rows and columns.

Database System:

The database, data collection with DBMS is called data base system. A database system is a way of organizing information on a computer, implemented by a set of computer programs. This kind of organization should offer:

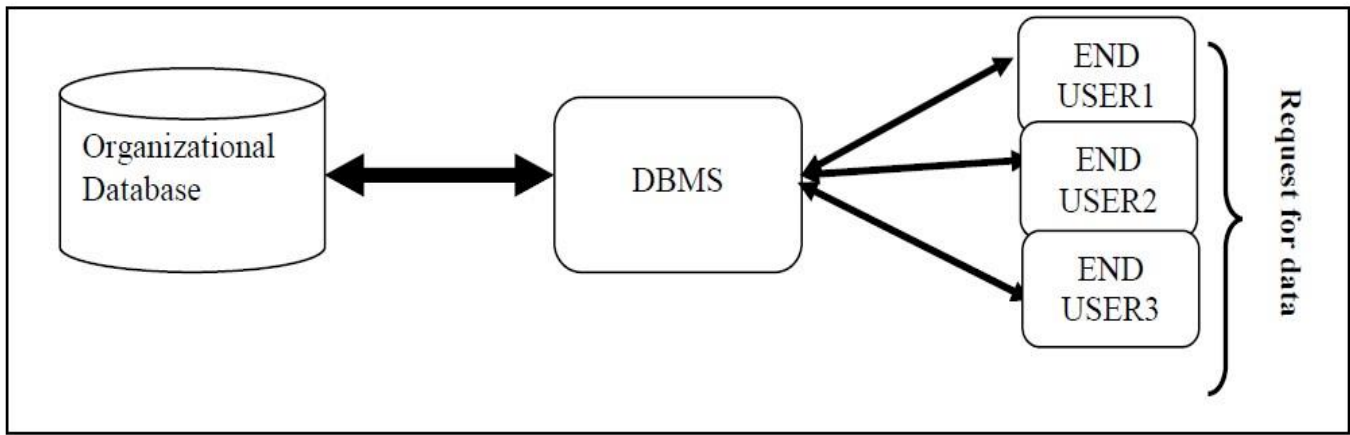
- **Simplicity** - an easy way to collect, access-connect, and display information; o stability - to prevent unnecessary loss of data;
- **Security** - to protect against unauthorized access to private data;
- **Speed** - for fast results in a short time, and for easy maintenance in case of changes in the organization that stores the information.

DBMS (Database management system):

Database management systems (DBMSs) are specially designed software applications that interact with the user, other applications, and the database itself to capture and analyze data. A general-purpose DBMS is a software system designed to allow the definition, creation, querying, update, and administration of databases.

Well-known DBMSs include [MySQL](#), [MariaDB](#), [PostgreSQL](#), SQLite, Microsoft SQL Server, Oracle, SAP HANA, [dBASE](#), FoxPro, IBM DB2, LibreOffice Base, FileMaker Pro, Microsoft Access .

In this figure illustrate that the DBMS stands between the database and the users.



Advantages of DBMS:

- ✓ Controlling data redundancy
- ✓ Elimination of inconsistency
- ✓ Better service to the user
- ✓ Flexibility of the system is improved □ Integrity can be improved □ Standards can be enforced.
- ✓ Security can be improved
- ✓ Provides backup and recovery

Disadvantages of DBMS:

- ✓ Database system are complex, difficult, and time consuming to design.
- ✓ Qualified personnel.
- ✓ Extensive conversion costs in moving from a file system to a database system.
- ✓ Initial training required for all programmers and users.

DBMS Evolution/history of DBMS:

i) 1960's(Traditional File System) :

In 1960's the traditional file system was invented. The traditional file based system is basically file based system in which we manually or through computer handle the database.

ii) Early 1970's(Tree Structure Model) :

The emergence of the first types of DBMS, the hierarchical DBMS. IBM had the first model developed on IBM 360, and their DBMS was called IMS, originally it was written for the Apollo program. This types of DBMS was based in binary trees, where the shape was like a tree and relations were only limited between parent and child record.

Advantage:

- Less redundant data
- Data independence
- Security & integrity
- Efficient searching
- Disadvantages :
 - Complex implementation

- Harder to handle many relationship

iii) Late 1970's (Network Data Model):

The emergence of the network DBMS, Charles Bachman developed first DBMS at Honeywell database named Integrated data store (IDS). A group called COD ASYL who is responsible for the creation of COBOL and that system standardized network DBMS where we developed for business use. In this model, each record can have multiple parents.

Disadvantage: -

- Complex to understand
- Difficult to design & maintenance

iv) 1980's(Relational Data Model):

The emergence of relational DBMS on the hands of Edgar cod. He worked at IBM and he was invented relational data model. This was a new system for entering data and working with big database where the idea was use a table of records.

v) 1990's & onwards(Object Oriented Model) :

In 1990 the DBMS took on a new objects oriented approach, joint with relational DBMS. In This approach, text multimedia internet and web use in conjunction with DBMS were available and possible.

Data warred house (DW, DWH and EDW):

Data warred house is a data structure that is optimized for distribution. It collects and stores integrated sets of historical data from multiple operational systems and feeds them to one or more data marts. It may also provide end-user access to support enterprise views of data.

Data Mart:

A data structure that is optimized for access. It is designed to facilitate end-user analysis of data. It typically supports a single, analytic application used by a distinct set of workers.

Data mining:

Data mining or knowledge discovery is the computer assisted process of digging through and analyzing enormous sets of data and then extracting the meaning of the data. Data mining tools predict behaviors and future, allowing business to make proactive knowledge driven decisions.

APPLICATION DEVELOPMENT WITHOUT DATABASE:

Application development would be complicated and would cause the serious problems during the data access, without a good database system. Programs need to store data and retrieve those stored data later in future when required. Using the traditional file processing system to store the data can cause various problems, some of problems described below:

- Data redundancy
- Program/data dependency
- Lack of flexibility
- Poor programmer productivity
- Excessive program maintenance

Data independence:

Data independence is usually consisting from the two points of view.

- (i) Physical Data independence
- (ii) Logical data independence

i) Physical data independence

It allows changes in the physical storage devices or organization of the file to be made without requiring changes in the concept view or any of the external view and hence in the application program using database.

ii) Logical data independence

The ability to change the logical (conceptual) schema without changing the External schema (User View) is called logical data independence. For example, the addition or removal of new entities, attributes, or relationships to the conceptual schema should be possible without having to change existing external schemas or having to rewrite existing application programs.

Data Abstraction in DBMS:

The purpose of database users with an abstracts view of the data that is system hides certain details of how the data are stored and maintained. It gives an architecture to separate the user application and physical database.

There are several levels of abstraction:

Physical Level:

- How the data are stored.
- E.g. index, B-tree, hashing.

Lowest level of abstraction

- Complex low-level structures described in detail. Conceptual Level:
- Next highest level of abstraction.
- Describes what data are stored.
- Describes the relationships among data.
- Database administrator level. View Level:

Highest level

- Describes part of the database for a particular group of users.
- Can be many different views of a database.
- E.g. tellers in a bank get a view of customer accounts, but not of payroll data.

Database Manager (DB Manager)

A database manager (DB manager) is a computer program, or a set of computer programs, that provide basic database management functionalities including creation and maintenance of databases. Database managers have several capabilities including the ability to back up and restore, attach and detach, create, clone, delete and rename the databases.

Database Users

Database administrators – DBA is responsible for authorizing access to the database, for coordinating and monitoring its use, and acquiring software and hardware resources as needed.

- **Database designers** – identify data to be stored in the database and choosing appropriate structures to represent and store the data. Most of these functions are done before the database is implemented and populated with the data. It is the responsibility of the database designers to communicate with all prospective users to understand their requirements and come up with a design that meets these requirements.
- **End Users :**
 - **Casual End Users** – occasionally access, may need different information each time. Use query language to specify requests.
 - **Naïve or parametric end users** – main job is to query and update the database using standard queries and updates.
 - **Sophisticated end users** – engineers, scientists, analysts who implement applications to meet their requirements.
 - **Stand alone users** – maintain personal databases using ready made packages.

Unit 2

Data model:

Data model is a collection of conceptual tools used for describing data, data relationships and data constraints. The model should enable the designer to incorporate a major portion of semantics of the database in the system.

Type of Data Model:

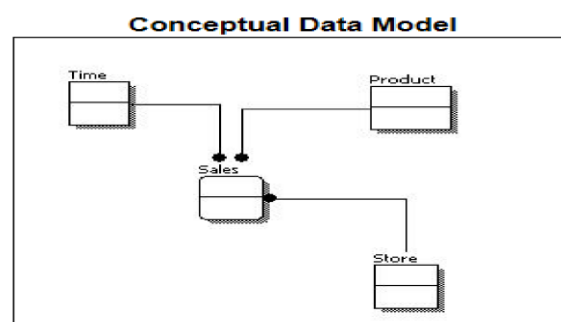
- Conceptual data model
- Logical data model
- Physical data model

i. Conceptual data model:

It identifies the highest-level relationships between the different entities. Features of conceptual data model include:

- Includes the important entities and the relationships among them.
- No attribute is specified.
- No primary key is specified.

The figure below is an example of a conceptual data model.



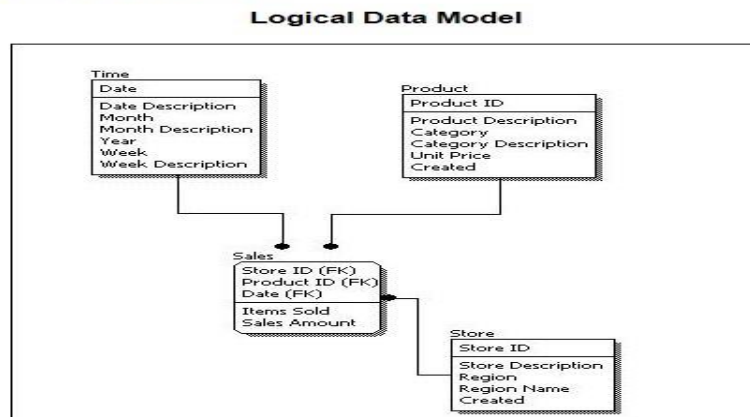
From the figure above, we can see that the only information shown via the conceptual data model is the entities that describe the data and the relationships between those entities. No other information is shown through the conceptual data model.

ii. Logical data model

It describes the data in as much detail as possible, without regard to how they will be physical implemented in the database. Features of a logical data model include:

- Includes all entities and relationships among them.
- All attributes for each entity are specified.
- The primary key for each entity is specified.
- Foreign keys (keys identifying the relationship between different entities) are specified.
- Normalization occurs at this level.

The figure below is an example of a logical data model.



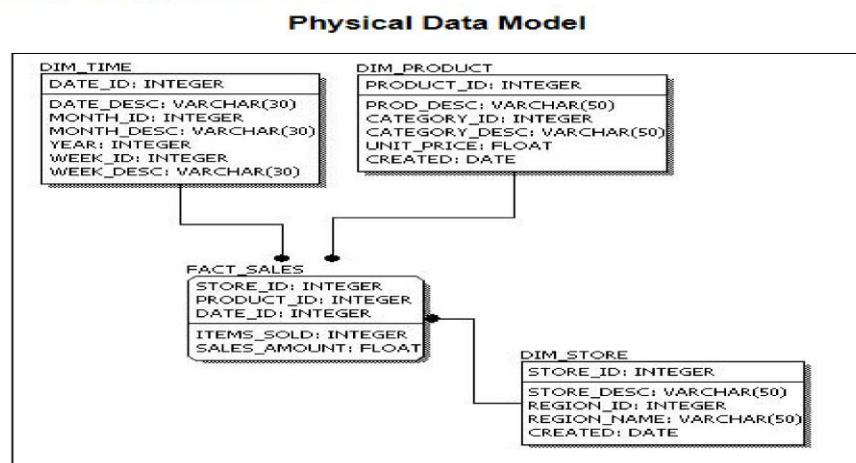
iii. Physical data model

It represents how the model will be built in the database. A physical database model shows all table structures, including column name, column data type, column constraints, primary key, foreign key, and relationships between tables.

Features of a physical data model include:

- Specification all tables and columns.
- Foreign keys are used to identify relationships between tables.
- Demoralization may occur based on user requirements. Physical data model will be different for different RDBMS. For example, data type for a column may be different between MySQL and SQL Server.

The figure below is an example of a physical data model.



Other types of data model:

i) Object based data model :

Object based data model are used in describing data and data relationship in accordance with concept. In general, the object based data models are gaining wide acceptance for their flexible structuring capabilities. The Entity relationship model which is an object based model is widely used in practices as an appropriate database design tool.

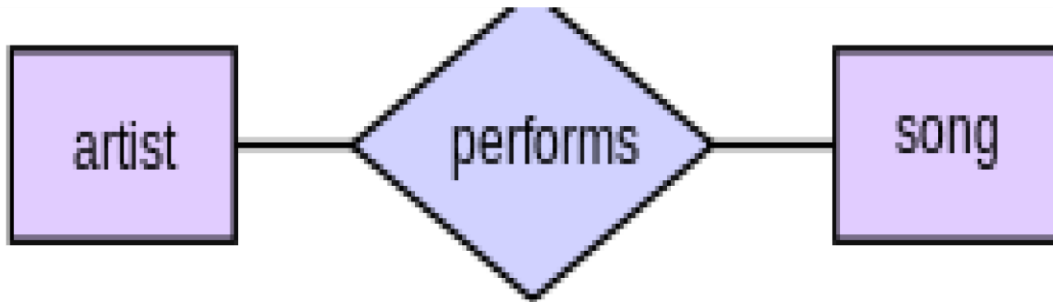
ii) Record based data model :

These data models are used to specify the overall logical structure of database. The three widely accepted record based data models are relational model ,network model and hierarchical model.

Entity relationship model:

An **entity–relationship model (ER model)** is a data model for describing the data or information aspects of a business domain or its process requirements, in an abstract way that lends itself to ultimately being implemented in a database such as a relational database. The main components of ER models are entities (things) and the relationships that can exist among them, and databases.

Fig: e-r model



• Association (Relationship):

A relationship is an associating among several entities. For example working in relationship associate a project with each employee that she/he is working in.

E-R Diagram:

The overall logical structure of a database can be expressed graphically by E-R diagram which consist of following components:

- Rectangles, which represent entity sets.
- Ellipses, which represent attributes.
- Diamonds which represent relationships among entity sets.
- Lines, which links attributes to entity sets and entity set to relationship

Example

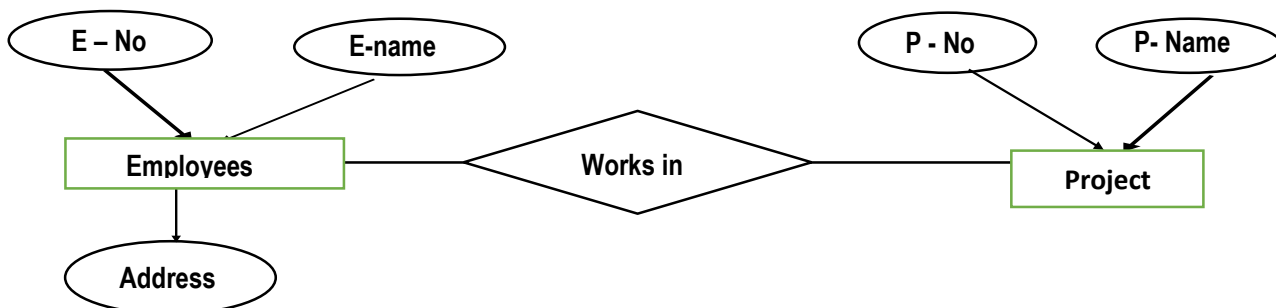


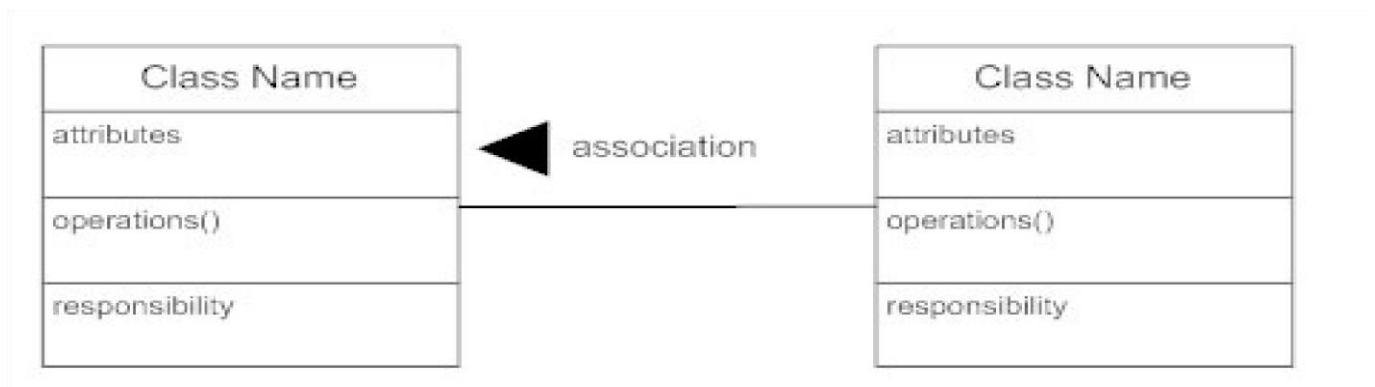
Figure: E-R Diagram for a part of the database a project management system

Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

Sample of class diagram



Entity set :

An **entity set** is represented by a set of attributes. Possible attributes of the employee entity set are E_No, E_Name and Address.

Relationship sets:

A relationship set is a set of relationship of the same type. Finally, it is a mathematical relation on $n \geq 2$ entity sets.

Mapping constraints/cardinalities

One important constraint is mapping cardinalities which express the no. of entities of an entity set to which entities of another entity set can be associated with a relationship. There are four types of mapping cardinality (association).

i) **One to One :**

An entity in A is associated with at most one entity in B and an entity B is also associated with

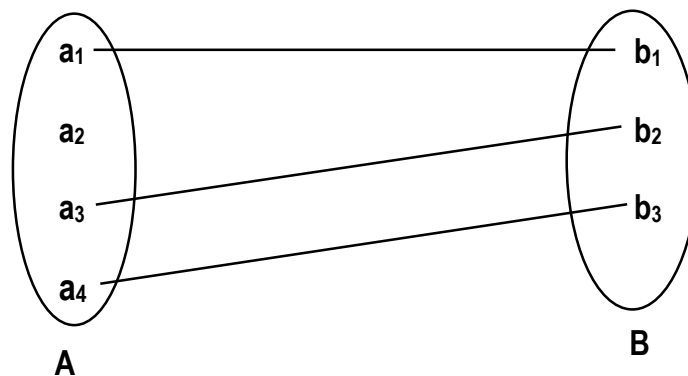


Fig: One to one relationship

ii) **One to many :**

An entity in A is associated with any number entities in B. An entity in B, however can be associated with at most one entity.

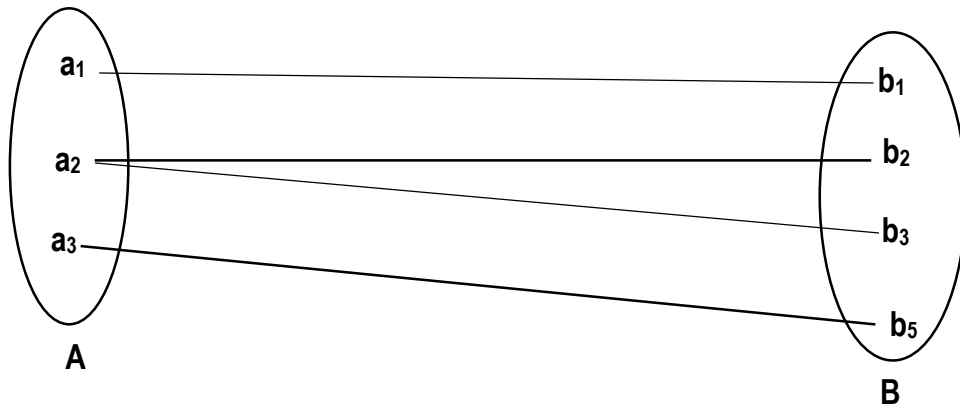


Fig: One to many relationships

iii) Many to one:-

An entity in A is associated with at most one entity in B. An entity in B, however can be associated with any no. of entities in A.

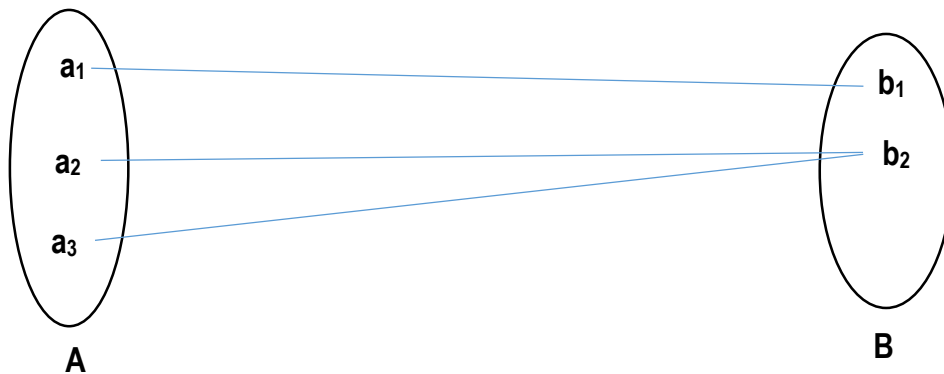
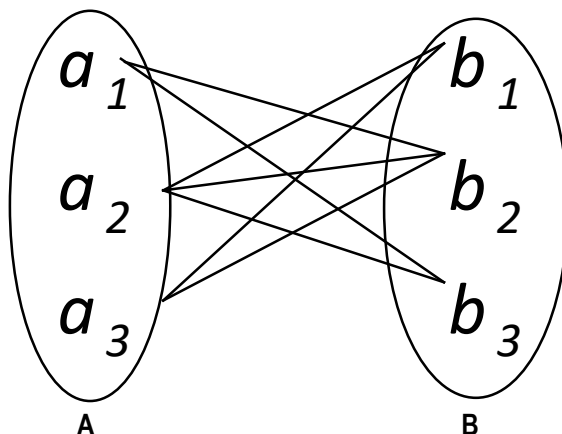


Fig: many to one relationship

iv) Many to many:

An entity in A is associated with any number of entities in B and an entity in B is also associated with any number of entities in A.



Other Associations:

Aggregation

Some special types of associations arise often enough that UML has defined special techniques for handling them. One category is known as an aggregation or a collection. For example, a Sale consists of a collection of Items being purchased. As shown in Figure, aggregation is indicated by a small diamond on the association line next to the class that is the aggregate.

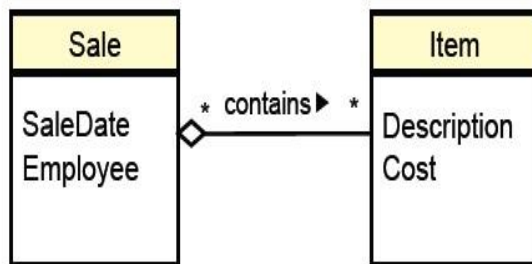


Figure 2.11

Association aggregation. A Sale contains a list of items being purchased. A small diamond is placed on the association to remind us of this special relationship.

Composition

Composition is a stronger aggregate association that does arise more often. In a composition, the individual items become the new object. Consider a bicycle, which is built from a set of components (wheels, crank, stem, and so on). UML provides two methods to display composition. In Figure, the individual classes are separated and marked with a filled diamond.

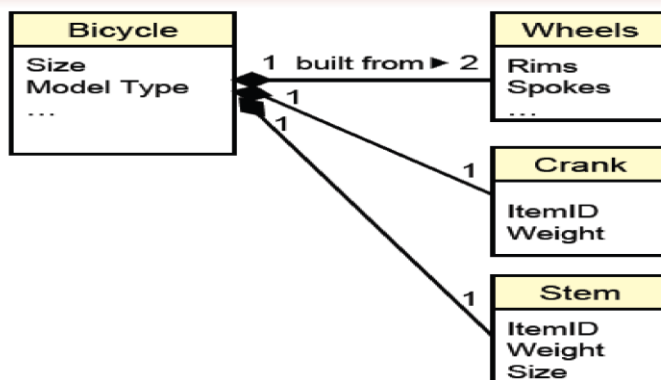


Figure 2.12

Association composition. A bicycle is built from several individual components. These components no longer exist separately; they become the bicycle.

Generalization

Another common association that arises in business settings is generalization. This situation generates a class hierarchy. The most general description is given at the top, and more specific classes are derived from it. Figure presents a sample from Sally's Pet Store.

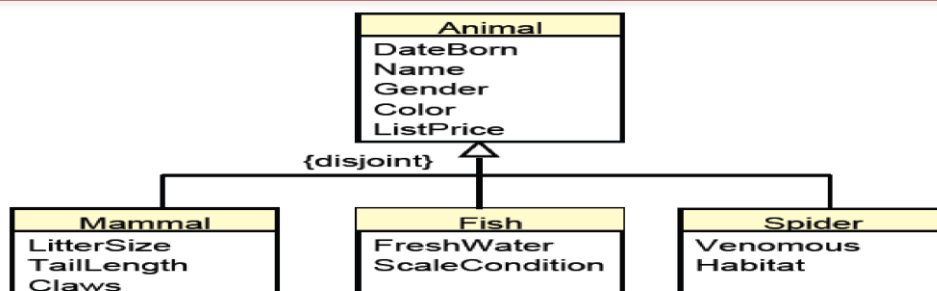


Figure 2.14

Association generalization. The generic Animal class holds data that applies to all animals. The derived subclasses contain data that is specific to each species.

Reflexive Association

A reflexive relationship is another situation that arises in business that requires special handling. A reflexive association is a relationship from one class back to itself. The most common business situation is shown in Figure most employees (worker) have a manager.

Unit 3 Relational Model

Relational Database:-

A relational database consists of a collection of tables i.e., in this model entity sets and relationships all are represented by tables. A row in a table represents a relationship among a set of values. Each row of a relation (table) is called a tuple. Fig shows the Structure of relational Database. Table Name: student

Attribute		
S.N	Name	Address
224	Ram Joshi	Butwal-13
220	Hari Kharel	Sukhanager-09
200	Gita kharel	Milanchowk-10

Values

Important of relational database:-

- ✓ Relational database are based on relational set theory. It supports relational algebra like :- (union, intersection, difference and Cartesian product) and also support select, projection, join, division, operations etc.
- ✓ Relational database supports concept of dynamic views.
- ✓ Relational database use SQL (structured query language)
- ✓ Relational database have an excellent security.
- ✓ Relational database support to new hardware technologies.

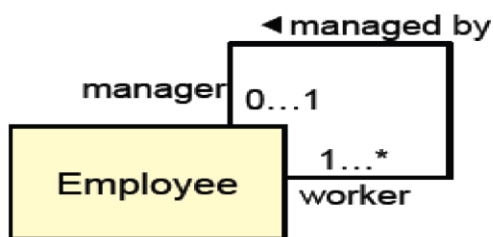


Figure 2.17

Reflexive relationship. A manager is an employee who manages other workers. Notice how the labels explain the purpose of the relationship.

Relational Algebra:-

In Data Management, **relational algebra** describes how data is naturally organized into sets of data, apply so as data is the documentation of a real life person, place or thing and the events or transactions between them at a point in time. The basic set of operation for the relational model is the relational algebra. These operations enable a user to specify basic retrieval request sets of operations include:-

⇒ Select

⇒	Project
⇒	Union
⇒	Set difference
⇒	Rename
⇒	Intersection
⇒	Join

Keys:-

The key is defined as the column or attributes of the database table. For example, if a table has id, name & address as the column names, then each one is known as the key for that table.

1) Super key :-

Super key for an entity set is a set of one or more attributes whose combined value uniquely identifies the entities in the entity set. For example, for an entity set employees, the set of attributes (emp_code, emp_name, address) can be considered to be a super key.

2) Candidate key :

A candidate key is a minimal super key that is a super key which doesn't have any proper subset and which is also a super key. For example, (emp_no) and (emp_name, address) are two candidate keys for the entity set employee.

3) Primary key :-

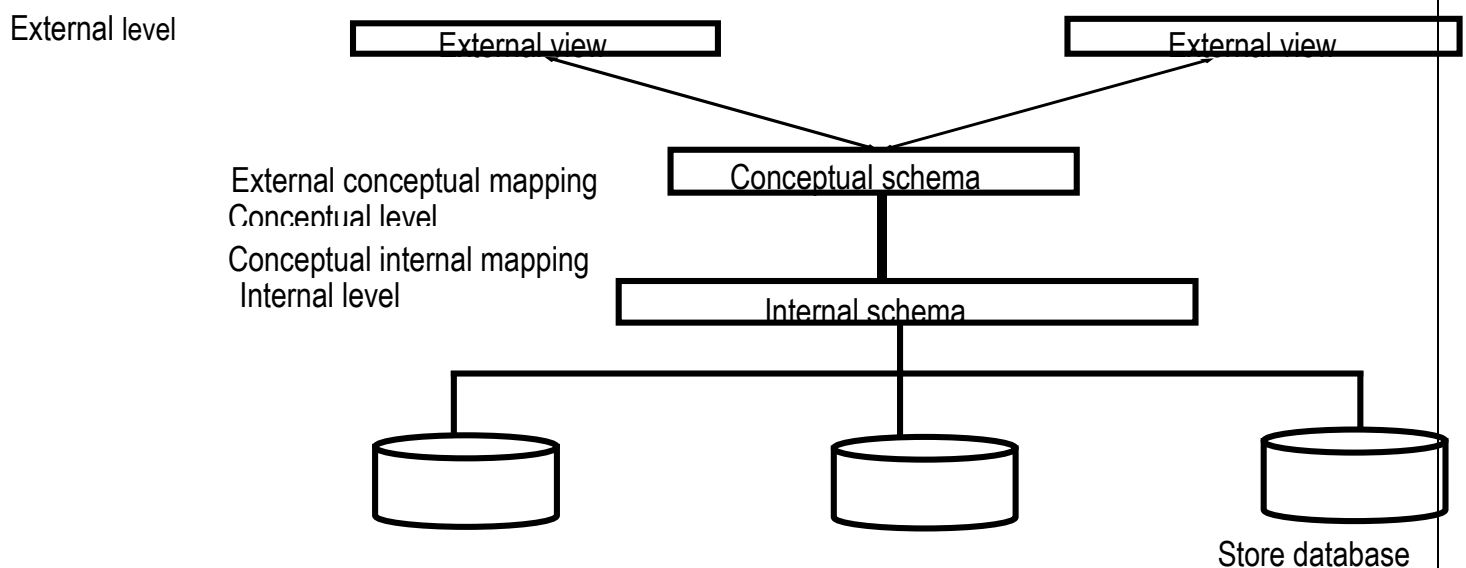
Primary key is usually the key selected to identify a row when the database is physically implemented. For example, (emp_no) can be considered to be the primary key for the entity set employees.

4) Foreign key :

Foreign key is an attribute that appears as a non-key attribute in one relation and as a primary key attribute in another relation.

Schema/schema diagram:

Schema of a database system is its structure described in a formal language supported by DBMS and refers to the organization of data to create a blueprint of how the database will be constructed.



-Internal level:

Internal level has an internal schema which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access path for the database.

- Conceptual level :

Conceptual level has a conceptual schema, which describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structure and concentrates on describing entities, data types, relationship users operation & constraints.

- External level :-

External level includes a number of external schema or user view. Each external schema describes a part of the database that a particular user group is interested in and hides that rest of the database from that user group.

Instance:-

Actual data in the database may change quite frequently. For example, the database change every time we add a student or enter the new grade. The data in the database at a particular moment in time is called a database State or snap shot. It is also called the current set of occurrences or instances in the database.

View:-

View is a single table that is described from other tables. These other tables can be base tables or previously defined views. A view doesn't necessarily exist in physical form, it is considered a virtual table.

Advantages of views:-

- View can represent a subset of the data contained in a table.
- Views can join multiple table into a single virtual table.
- View can hide the complexity of data.
- View take very little space to store.

SQL CREATE VIEW Syntax

```
CREATE VIEW view name AS SELECT column1, column2..... FROM table_name  
WHERE [condition];
```

Example:

Consider the CUSTOMERS table having the following records:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Now, following is the example to create a view from CUSTOMERS table. This view would be used to have customer name and age from CUSTOMERS table:

```
SQL > CREATE VIEW CUSTOMERS_VIEW AS  
SELECT name, age  
FROM CUSTOMERS;
```

Now, you can query CUSTOMERS_VIEW in similar way as you query an actual table. Following is the example:

Data Dictionary: Data dictionary or metadata repository is a centralized repository of information about data such as meaning relationship to other data origin, usage and format. Most database management system keeps the data dictionary hidden from users to prevent them from accidentally destroying its content.



Unit-4

Relational database query languages

SQL (Structure Query Language)

SQL (Structured Query Language) is a special purpose programming language designed for managing data held in a relational database management system (RDBMS).

Originally based upon relational algebra and tuple relational calculus, SQL consists of a data definition language and a data manipulation language. The scope of SQL includes data insert, query, update and delete, schema creation and modification, and data access control.

Query:

- In computers, what a user of a search engine or database enters is sometimes called the query. To query (verb) means to submit a query (noun).
- A database query can be either a select query or an action query. A select query is simply a data retrieval query. An action query can ask for additional operations on the data, such as insertion, updating, or deletion.

Sub queries:

A sub query is a form of an SQL statement that appears inside another SQL statement; it is also termed a nested query. Sub queries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN etc.


It can be used by the following commands.


- To insert records in a target table.
- To update records in a target table.
- To create view


Data types (Domain) in SQL:


Is an attribute that specifies the types of data that the object can hold: integer data, character data, and date and time data and so on.

Some of the data types are given below.

 **Char (size)**
This data type is used to store character string values of fixed length. The size in brackets determines the number of characters the cell can hold.

 **Varchar (size)/ Varchar2 (size):**
This data type is used to store variable length alphanumeric data.

 **Number:**
The number is used to store number (fixed or floating point) in database.

 **Date:**
This data type is used to represent date & time. The standard format is DD-MM-YY as 21-jun-2012.



Long :

This data type is used to store variable length character string containing upto 2 GB.



Boolean :-

This data type has values of True and False.

- ☐ **Date/Time** Use for dates and times
- ☐ **Text:** Use for text or combinations of text and numbers. 255 characters maximum.
- ☐ **Memo:** Memo is used for larger amounts of text. Stores up to 65,536 characters.
- ☐ **Yes/No** A logical field can be displayed as Yes/No, True/False, or On/Off.
- ☐ **AutoNumber:** AutoNumber fields automatically give each record its own number, usually starting at 1.
- ☐ **Currency:** Use for currency. Holds up to 15 digits of whole dollars, plus 4 decimal places.

- **Binary Objects:** A relatively new domain is to separate category for objects or binary large objects (BLOB). It enables us to store any type of object created by the computer. A good e.g. is to use a BLOB to hold files from other software packages. An engineering database might hold drawings and specifications for various components. The advantage is that all of the data is stored together making it easier for users to find the information they need and simplifying backups.

- **Computed values:** Some business attributes can be computed. For instance the total value of a sale can be calculated as the sum of the individual sale prices plus the sales tax, or an employee's age can be computed as the difference between today's date and Date of Birth. At the design stage, you should indicate which data attributes could be computed. The UML notation is to precede the name with a Slash (/) sign and then describe the computation in a note.

- **Events:** Events are another important components of modern database systems that we need to record. Three basic types of events occur in a database environment.
 - Business events that trigger some functions, such as a sale triggering a reduction in inventory
 - Data changes that signal some alert, such as an inventory that drops below a present level, which triggers a new purchase order, and
 - User interface events that trigger some action, such as a user clicking on an icon to send a purchase order to a supplier.
 Events are action that is dependent on time.

DDL: (Data definition language)

DDL is a syntax similar to a computer programming language for defining data structures, especially database schemas. Many data description, language SQL uses a collection of imperative verb whose effect is to modify the schema of database by adding changing and deleting definitions of table or other objects.

Create statements:

To make a new database table, index or view etc.

Syntax.

CREATE TABLE [table name]

(columnname1 data types (size), column name2 data types (size).....);

Example:-

Create employee table having some attributes.

Create table employee (id integer primary key, fname varchar(50) , Lname varchar (50) , date_of_Birth date);

Drop statement:

To destroy an existing database, table, index or view.

Syntax:

Drop object_type object_name;

Example:

Destroy employee table

Drop TABLE employee

Alter statement: To modify an existing data base object.

Syntax:

ALTER object type object of name parameters;

Example:

The command to add (then remove)

Add a column named phone_no for an existing table named employee.

ALTER TABLE employee Add phone_no number;

Drop a column named phone_no for an existing table named employee .

ALTER TABLE employee Drop column phone_no;

DML (Data Manipulation Language): statements are used for managing data within schema objects.

DML is a family of syntax element similar to a computer programming language used for inserting deleting and updating data in database. DML statements are.

❏Select.....from.....where ❏Insert.....intovalues.....

❏Update.....set.....where.....

❏Deletefrom.....where.....

The SQL SELECT Statement

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

SQL SELECT Syntax

SELECT column_name, column_name

FROM table_name; Example:

SELECT * FROM employee;

The SQL INSERT INTO Statement

- The INSERT INTO statement is used to insert new records in a table.

SQL INSERT INTO Syntax

INSERT INTO table_name

VALUES (value1, value2, value3...);

Or

INSERT INTO table_name

(column1, column2, column3...)

VALUES (value1, value2, value3...);

Example:

INSERT INTO employee VALUES (111, 'Sita Sharma', 'butwal');

The SQL UPDATE Statement

- The UPDATE statement is used to update existing records in a table. **SQL UPDATE Syntax** UPDATE table_name SET column1=value1,column2=value2,... WHERE some_column=some_value;

Example

UPDATE employee

SET name='sunita joshi'WHERE address='Butwal';

The SQL 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;

Example

```
DELETE FROM employee
WHERE name='sunita joshi'
```

Data Control Language (DCL)

The Data Control Language (DCL) is a subset of the Structured Query Language (SQL) that allows database administrators to configure security access to relational databases. Some examples:

- GRANT - gives user's access privileges to database
- REVOKE - withdraw access privileges given with the GRANT command

Operator in SQL:

- Relational operator : =, >=, <, >, <=, !=
- Arithmetic operator : +, -, /, *
- Logical operator : AND, OR, NOT

And operator:

Process all rows in a table & display the result only when all of the conditions specified using AND operator is satisfied.

Example:

Retrieve the contents of the columns product_no, description, profit_percent, sell-price from the table product master where the values contained in the field profit_percent is between 10 & 20 both inclusive.

```
select product_no, description, profit_percent,
sell_price from product_master where profit_percent>=10 AND
Profit_percent<=20;
```

OR operator:

Process all rows in a table & display the result only when any of the conditions, specified using OR operator is satisfied.

Example

Retrieve client information like client_no, name, address, and city, pin_code for all the clients where the pin code has the value 400054 or 400057;

```
Select * from client_master where pin_code =400054 OR pin_code=400057 ;
```

Not operator:-

Process all the rows in a table and display the result only when none of the conditions specified using the Not operator are satisfied.

Example:

Retrieve specified client information for the clients who are not in "Butwal" OR "Pokhara"

```
Select client_no, name, address, city pin_code from client_master
```

```
Where Not (city='Butwal' or city=' pokhara');
```

Range Searching:

BETWEEN Operator:

Allows the selection of rows that contains values within a specified lower and upper limit.

Example:

- Retrieve all information from the table productmaster where the values contain within the field profit-percent is betⁿ 10 & 20 both inclusive.

```
Select* from product-master where profit_percent between 10 AND 20;
```

Pattern matching:

LIKE predicate:

Allows for a comparison of one string value with another string value which is not identical.

For character data types:

- The percent sign (%) matches any string.

- The underscore (_) matches any single character. Example:

Retrieve all information about supplier whose name begin with letters 'ja' from supplier_master.

```
Select *from supplier_master where supplier_name like
'ja%';
```

```
Select* from supplier_master Where supplier_name like
'l_r%' or
```

```
Supplier_name Like ' _h%';
```

Selection operation:-

To identify a set of tuples which is a part of a relation and to extract only these tuples out. The select operation selects tuples that satisfy a given predicate or condition.

- ⇒ It is a unary operation defined on a single relation.
- ⇒ It is denoted as sigma (σ).

⇒ It supports arithmetic comparison operator and logical operator.

Note: Arithmetic comparison operator:

⇒ (<, >, <=, >=, ==, !=)

Logical operator: (NOT, OR, AND)

Example:

Table: Book

Acc.No	YR_PUB	TITLE
734216	1982	Algorithm Design
23767	1995	Database Design
89476	1992	Compiler Design
54678	1991	Programming Design
376112	1992	Machine Design

Select from relation "Book" all books whose year of publication is 1992.

Solution: Relational Algebra:

$\sigma_{YR_PUB=1992}(Book)$

SQL: Select* from Book where YR_PUB=1992

Projection operation:

Returns its argument relation with certain attributes left out.

□ It is an unary operation defined on a single relation.

□ It is denoted as π (Pie)

Example;

List all the Title and Acc-No of the "book" relation.

Relation algebra= $\pi_{Acc_No, Title}(Book)$

SQL=Select Acc_No, TITLE from Book

Set Operations:

Union:

Union is used when we need attributes that appear in either or both of two relations. It is denoted as U.

Ex.

R

a	b	c
d	c	a
b	f	e

S

b	c	a
c	d	f
b	f	e

Output:- R U S=

a	b	c
d	c	a
b	f	e
b	c	a
c	d	f

For example:

Borrower (customer_name, loan_no)

Depositor (customer_name, Account_number)

List all the customers who have either an account or loan number or both

relational Algebra= π customer_name (borrower) \cup π customer_name (depositor)

SQL= Select customer_name from Borrower union

Select customer name from depositor

Difference Operation (MINUS)/ Except:

Finds tuples in one relation but not in other. It is denoted as “-”. Example : (output)

R-S= a b c d c a

Example :

Find the names of all customers who have an account but not a loan numbers

Relational Algebra:

π customer_name (Depositor) - π customer_name (Borrower)

SQL= Select customer_name from Depositor MINUS select customer_name from borrower

Intersection operation :

Find tuples in both the relation. It is denoted as \cap .

Output

R \cap S= b f e

Example :

Borrow (customername, loan_number)

Depositor (customername, accountnumber)

List all the customer's who have both a loan & an account. **Relational Algebra=**

π customername (Borrower) \cap customername (depositor) **SQL=**

select customername from Borrower intersect select customername from Depositor

RENAME operation :

RENAME operation is used rename attributes. It is denoted as ρ (row)

Example : Employee (name, eid)

Change attributes name to employee name


Relational Algebra= ρ employee_name/name (employee)

SQL=ALTER TABLE employee RENAME column name to employee_name;


Also, we can rename table name

SQL= RENAME Employee To student;

Join Operation :

The Join operation is defined by the  symbol and is used to compound similar tuples from two relation into single longer tuples. Every row of the first table is joined to every row of second table. The result is tuples taken from both tables.

Syntax:

A  <Join condition>B

Types of join

π inner join (equi-join)

An inner join creates a new results table by combining column value of two tables A and B based upon the join predicate.

The query compares each row of A with each row of B to find all pairs of rows which satisfy the join predicate.

Relational Algebra= R <R. primary-key=s.foreignkey>s

For Example:

Students		Courses
ID	S Name	Course
100	Ram	PH
200	Shyam	CM
300	Hari	CM

Course	c_Name
PH	Pharmacy
CM	Computing

Relation algebra for inner point join or equi-join \bowtie $\langle \text{students. Course} = \text{courses.course} \rangle$

SQL=

Select* from students INNER JOIN courses on students.course= courses.course;

Natural join:-

A natural join offers a further specialization of equijoins. The join predicates arise implicitly by comparing all columns in both tables that have the same columnname in the join tables. The resulting tables one column for each pair of equally named columns.

Relation algebra=

Students \bowtie $\langle \text{course} = \text{course\#} \rangle$ courses

SQL=Select*from students natural join courses

Theta Join:

A Cartesian product with a condition applied $R \langle \text{condition} \rangle S$ is called theta join.

SQL= Select* from student, courses where student=200;

iii) Full other join :

A full other join combines the results of both left and right other joins the join table contains all records from both tables, and fill in nulls for missing matches on either side.

It is denoted by \Join .

Outer Join

An outer join doesn't require each record in the two joined tables to have a matching record. The joined tables retains each-record-even if no other matching records exists.


There are three types of outer join:

- Left outer join
- Right outer join
- Full outer join

i) Left Outer join :-

The result of the left outer join (left join) for table A and B always contains all records of the "left" table A, even if the join condition doesn't find any matching record in the "right" table B. It is denoted by.

Syntax;

R  <r.primary_key =s. foreign_key> s

Example

Student  <student. Course= courses. course> courses

SQL=select*from students left outer join courses on student .course= courses. Course;



ii) Right outer join :

A right outer join (right join) closely resembles a left outer join except with treatment of the tables reversed. Every row from the "right" table B will appear in the joined table at least one. Matching row from the left table A exist . Null will appear in columns from A for those records that have no match in A. It is denoted by .

Syntax;


R  <r.primary_key =s. foreign_key> s

Example

Student  <student. Course= courses. course> courses

SQL=select*from students right outer join courses on student .course= courses. course;

Syntax:

R  <R.primary_key=S.foreign_key> S

Relational Algebra=student  course <student. course = courses.course> Courses

SQL=Select* from student full outer join courses on students.course=courses.course

SELF JOIN

The SQL **SELF JOIN** is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement. The self-join statement is necessary when two sets of data, within the same table, are compared.

Syntax:

The basic syntax of **SELF JOIN** is as follows:

SELECT a.column_name, b.column_name...

FROM table1 a, table1 b WHERE

a.common_field = b.common_field; Here, WHERE clause could be any given expression based on your requirement.

Example:

Consider the following two tables, (a) CUSTOMERS table is as follows:

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Kaushik	25	Delhi	1500.00
3	Kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

Now, let us join this table using SELF JOIN as follows:

```
SQL> SELECT a.ID, b.NAME, a.SALARY
FROM CUSTOMERS a, CUSTOMERS b
WHERE a.SALARY < b.SALARY;
```

This would produce the following result:

ID	NAME	SALARY
2	Ramesh	1500.00
2	Kaushik	1500.00
1	Chaitali	2000.00
2	Chaitali	1500.00
3	Chaitali	2000.00
6	Chaitali	4500.00
1	Hardik	2000.00
2	Hardik	1500.00
3	Hardik	2000.00

Function In SQL :-

SQL has many built-in functions for performing calculations on data. Function are also capable of accepting user-supplied variables or constants and operating on them.

Aggregate functions:

SQL aggregate functions return a single value, calculate from the values in column.

i) Avg : returns the average value. example:

select avg(sell_price) "Average" from product_master;

ii) MIN :

Returns the smallest value.

example:

select MIN(bal_due) "minimum balance" from client_master;

iii) Count (expr):

Returns the number of rows where "expr" is not null. example:

**select count (product_No) "No of product"
from product_master;**

iv) Count (*):

Returns the number of rows in tables including duplicates and those with nulls.

example:

select count(*) "total" from client_master;

Max():

Returns the largest value. Exmaple :

select max (bal_due) "maximum" from client-master;

Sum :

Returns the sum.

Example:

select sum (bal_due) "total balance due" from client_master;

SQRT :-

Returns square root of 'n' .

example :

select sqrt(bal_due) "sqrt of bal_due" from client_master;

Group By clause :

Group by clause is another section of the select statement. To group rows based on distinct values that exists for specified column i.e. it creates a data set, containing several set of records grouped together based on a condition.

Table : sales

Example :

Retrieve the product and the total quantity ordered for each product from the sales tables.

**select product_No, Sum (Qty_ordered) "Total
Qty" from sales group by product_No;**

Having clause:

The HAVING Clause can be used in conjunction with group by clause. HAVING imposes a condition on the group by clause, which further filter the group created by the group by clause. Eg,

Detail ord no.	Product No.	Qty ordered	Qty Dispatched
019001	P0001	10	10
019001	P0004	3	3
019001	P0006	1	7
019002	P0002	4	4
019002	P0005	10	10
019003	P0003	2	2
019004	P0001	6	6
019005	P0006	4	4

019005	P0004	1	1
019006	P0006	8	8

- Retrieve the product No. & total quantity ordered for products 'P0001' or 'P0004' From sale table .

Select product_No, Sum (Qty ordered) "Total Qty" from sales group by product_No having product_No='P0001' OR product_No= 'P0004' ;

ORDER BY clause :

allows data from a table to be viewed in a sorted order. The rows retrieved from the table will be sorted in either ascending or descending order depending on the condition specified in the select sentence.

Eg,

- Retrieve all rows from the table client_masters & display this data sorted on the value contained in the field client no in ascending order.

select* from client_master

ORDER By client_no;

Note : if we want to descending order then,

Select* from client_master

ORDER by client_no Desc;

Stored Procedure

- A **stored procedure** is a subroutine available to applications that access a relational database system. A stored procedure (sometimes called a **proc**, **sproc**, **StoPro**, **StoredProc**, **sp** or **SP**) is actually stored in the database data dictionary.
- Typical use for stored procedures include data validation (integrated into the database) or access control mechanisms. Furthermore, stored procedures can consolidate and centralize logic that was originally implemented in applications. Extensive or complex processing that requires execution of several SQL statements is moved into stored procedures, and all applications call the procedures. One can use nested stored procedures by executing one stored procedure from within another.
- Stored procedures are similar to user-defined functions (UDFs).

Difference between view and procedure

Views

1. Does not accepts parameters
2. Can be used as a building block in large query.
3. Can contain only one single Select query.
4. Can not perform modification to any table except base table from where it's created.
5. Can be used (sometimes) as the target for Insert, update, delete queries.

Stored Procedure

1. Accept parameters
2. Can not be used as a building block in large query.
3. Can contain several statement like if, else, loop etc.
4. Can perform modification to one or several tables.
5. Can not be used as the target for Insert, update, delete queries.

QBE(Query By Example)& QQBE :

is database query language for relational database. It was devised by Moshe M. Zloof at IBM research during the mid 1970's, in parallel to the development of SQL. It is the first graphical query language, using Visual tables where the user would enter commands, example, elements, & condition. Many graphical front- ends for databases use the ideas from QBE today. Originally limited only for the purpose of retrieving data, QBE was later extended to allow other operations, such as inserts, deletes & updates as well as creation temporary tables.

Unit-5

Relational Database Design

Data Normalization:-

The essence of data normalization is to split your data into several tables that will be connected to each other based on the data within them. The goal of data normalization is to identify the business rules so that you can design good database table. By designing database tables carefully, we

- Save space
- Minimize duplication
- Project the data to ensure it's consistency
- Provide faster transactions by sending less data.
- Fundamental rules of Data Normalization.
- Each cell in a table contain atomic (singlevalued) data.
- Each non-key column depends on all of the primary key columns.
- Each non-key column depends on nothing outside of the key columns.

- Types of Normalization :-

i) 1 normal form (1NF) ii) 2 normal form (2NF) iii) 3 normal form (3NF) iv)

Boyce codd Normal form (BCNF)

i) 1 normal form (1NF) :-

An entity is in the first normal form if it contains no repeating groups or each cells contain atomic data

Example :

Entity: Customer (cid, full_name, tele_ph) is not in 1NF & now converting 1NF we have to split the table.

CID	First Name	Surname	Tele Ph
123	Rohan	Pandey	071
456	Hari	Sharma	072
789	Mohan	KC	073

Customer info (CID,first_name, sur_name)

Customer tel (CID, tele_ph)

ii) Second normal form (2NF):-

A relation R is in second normal form (2NF) if & only if it is in 1NF & every non-key attributes is fully functionally dependent on the primary key.

Note: full functional dependency

- A&B are attributes of a relation.

B is fully dependent on A if B is functionally dependent on A (E.g. A→B)



Figure 3.20

Second normal form definition. Each nonkey column must depend on the entire key. It is only an issue with composite keys. The solution is to split off the parts that only depend on part of the key.

Example

Sale line (sale_ID, Item_ID, Description, list price, Quantity, Quantity on Hand) it is not in 2NF, Now converting into 2NF

Item(Item_ID, Description, listprice, Quantity on hand)

Sale_Items (sales_ID, Item_ID, Quantity)

iii) Third normal form (3NF):-

A relation that is in 1NF, 2NF in which no non_primary key attribute is transitively dependent on the primary key.

Note: Transitive Dependency

A,B, & c are attributes of relation such that if

$A \rightarrow B$ and $B \rightarrow C$, then $C \rightarrow A$ is transitively dependent on A through B.

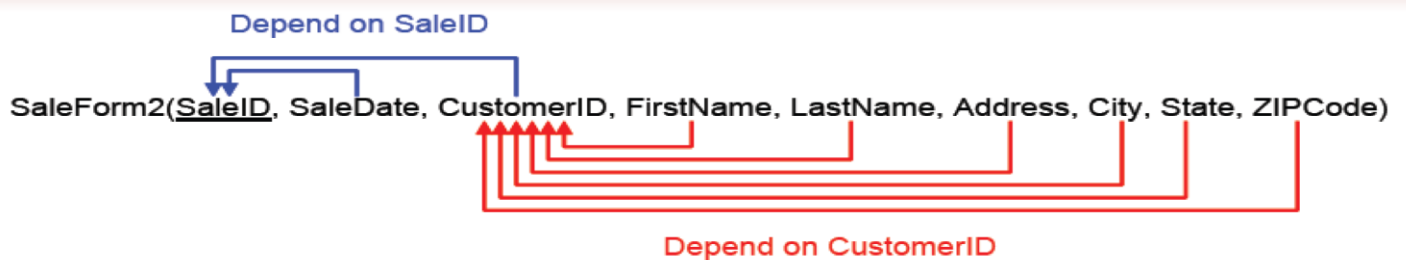


Figure 3.25

Third normal form definition. This table is not in 3NF since some of the columns depend on CustomerID, which is not part of the key.

Example

Sale_form (sale_ID , sales_date, customers_Id, first_name, last_name, city, state, zip code) above table is not in 3NF because there is transitive dependency, now, convert into 3NF.

Sale (sales_Id, sale_Date, customer_ID)

customer (customer_Id, first_name, last_name, city, state, zip code)

iv) BCNF (Boyce codd Normal form):-

A relation is in Boyce-Codd normal form if and only if it is in third normal form and every determinant is a candidate key. That is, if there is an FD $X \rightarrow Y$, then X must be the primary key (or equivalent to the primary key). In simpler terms: there cannot be a hidden dependency, where hidden means it is not part of the primary key.

example

Employee.specialty (EID, specialty, manager)

Above table is not in BCNF because there is hidden dependency. Now converting into BCNF.

Employee_info (EID, manager) manager_specialty (manager, specialty)

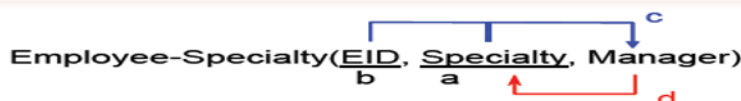


Figure 3.29

Boyce-Codd normal form. There is a hidden dependency (d) between manager and specialty. If we delete rows from the original table, we risk losing data about our managers. The solution is to add a table to make the dependency explicit.

v) Fourth normal form

Fourth normal form (4NF) is a normal form used in database normalization. Introduced by Ronald Fagin in 1977, 4NF is the next level of normalization after Boyce-Codd normal form (BCNF). Whereas the second, third, and Boyce-Codd normal forms are concerned with functional dependencies, 4NF is concerned with a more general type of dependency known as a multivalued dependency. A Table is in 4NF if and only if, for every one of its non-trivial multivalued dependencies $X \twoheadrightarrow Y$, X is a superkey—that is, X is either a candidate key or a superset thereof.

Multivalued dependencies

Multivalued dependencies are also referred to as tuple generating dependencies. After the Boyce -Codd normal form the results may be devoid of any functional dependencies but it may encounter multivalued dependencies as the multivalued dependencies also cause redundancy of data.

For eg:

If there are 3 attributes involved in a relation, A, B, and C..

Then for every value of A we will have respective values for B and C.. But it is a necessary in the 4th normal form that both B and C values are independent of each other.

This is represented by.

A

-

>

>

B

A

-

>

>

C

.

.

MVD or Multivalued Dependency is a dependency where one attribute value is potentially a "multivalued fact" about another and the attributes must be independent of each other.

Example of 4th NF

EmployeeTasks(EID, Specialty, ToolID)

Business rules.

(a) Each employee has many specialties.

(b) Each employee has many tools.

(c) Tools and specialties are unrelated.

EmployeeSpecialty(EID, Specialty)

EmployeeTools(EID, ToolID)

Figure 3.30

Fourth normal form. The original table is 3NF because there are no nonkey columns. The keys are legitimate, but there is a hidden (multivalued) dependency because Specialty and ToolID are unrelated. The solution is to create two tables—one to show each of the two dependencies.

Data constraints/Data rules and integrity:-

Rules which are enforced on data being entered and prevents the user from entering invalid data into tables are called constraints. Thus constraints control data being entered in tables for permanent storage.

Data constraints to be attached to table column via SQL syntax that will check data for integrity.

Types of data constraints:

i) I/O Constraints:

This data constraint determine the speed at which data can be inserted or extracted from a table. I/O constraints divided into two parts.

a. Primary key constraints

- That the data entered in the table column is unique.
- That none of the cells belonging to the table column left empty.

b. Foreign key constraints/referential integrity constraints:

This constraints establishes a relationship between records. This relationship ensures:

- Records cannot be inserted into a detail table if corresponding records in the master table do not exist.

II) **Business Rule constraints :**

DBMS allows the application of business rules to table columns. Business manager determine, business rules. These rules are applied to data prior the data being inserted into table columns eg,
The rule that no employee in the company shall get salary less than Rs.1000/- is a business rule.

Trigger:

A database trigger is procedural code that is automatically executed in response to certain events on a particular table or view in a database. In addition to trigger that fire when data is modified & when user logon or logoff events occur. There are four type of trigger:

Row level Trigger: This gets executed before or after any column value of a row changes

Column level Trigger: This gets executed before or after the specified column changes.

▢ **For each row types:** This trigger gets executed once for each row of the result set cause by insert/update/delete.

▢ **For each statement:** This trigger executed only once for the entire result set, but fires each time the statement is executed.

Unit- 6

Security

Security

- The security of a database can be thought of as a barrier which prevents unauthorized access of data.
- Database security concerns the use of a broad range of information security controls to protect database (potentially including the data, the database application or stored functions, the data base systems, the database servers and the associated networks links) against compromises of their confidentiality, integrity and availability.
- It involves various types or categories of control, such as technical, procedural administrative and physical.

Need of security:

- Unauthorized or unintended activity or misuse by authorized database users, database administrator or network system managers, or by unauthorized user or hackers.
- Mal ware infections causing incidents such as unauthorized access, leakage or disclosure of personal or proprietary data, deletion or damage to the data or program.
- Overloads, performance constraints and capacity issues resulting in the inability of authorized users to use databases as intended.
- Physical damage to database serves caused by computer room fires or floods, overheating, lightning, accidental liquid spills, static discharge, electronic breakdown/ equipment failures.

Access Control :

- Access control mechanisms enforce rules about who can perform what operation or who can access which data.
- Thus any access control mechanism must concern itself which three basic components, namely.

i) Identification and authentication of Accessor :

- The process of identification may involves several parameters such as personal identification of accessor, location of accessory, time and day of access, frequency of access etc.
- Personal identification may be performs with the help of passwords, voice prints, finger prints signatures etc.
- The process of authentication requires supplying information known only to the person the accessor has claimed to be.
- This may be done by quoting a password or by answering some question from system.
- The location of the accessor or time and day of access are also useful in detecting unauthorized access. ii)

Object to be locked :

a) Data object :

The data objects to be locked may be files or some records of a file. Sometimes it may be necessary to lock and privacy keys are used to control access to a particular record type.

b) View :

The another levels of security provided by DBMS is defining views or the external model of the schema. The owner of a database may grant views which may consist of the entire data base or a certain portion of the database.

c) Type of Access :-

Once an object is created, the own may grant other genuine accessors any of the following access right the object. Read, Run, modify, delete insert, create, Destroy.

ii) Crypto System:

A cryptosystem is the combination of three elements an encryption engine, keying information, and operational procedures for their secure use. In order to cryptographically secure high-value data on a hard disk (or on back-up media), it is necessary to employ a high-grade cryptosystem: one which even an attacker processing both a copy of your encryption engine and knowledge of your operating procedures cannot break without your keying information.

Encryption and Decryption:

- Encryption is a method of modifying the original information according to some code so that it can be read only if the user knows the decryption key.
- Encryption should be used when transmitting information from one computer to another particularly when using the internet.
- Sensitive information stored within a database also can be encrypted.
- Decryption is the reverse operation of encryption.
- The process of decoding data that has been encrypted into a secret format decryption requires a secret key or password.

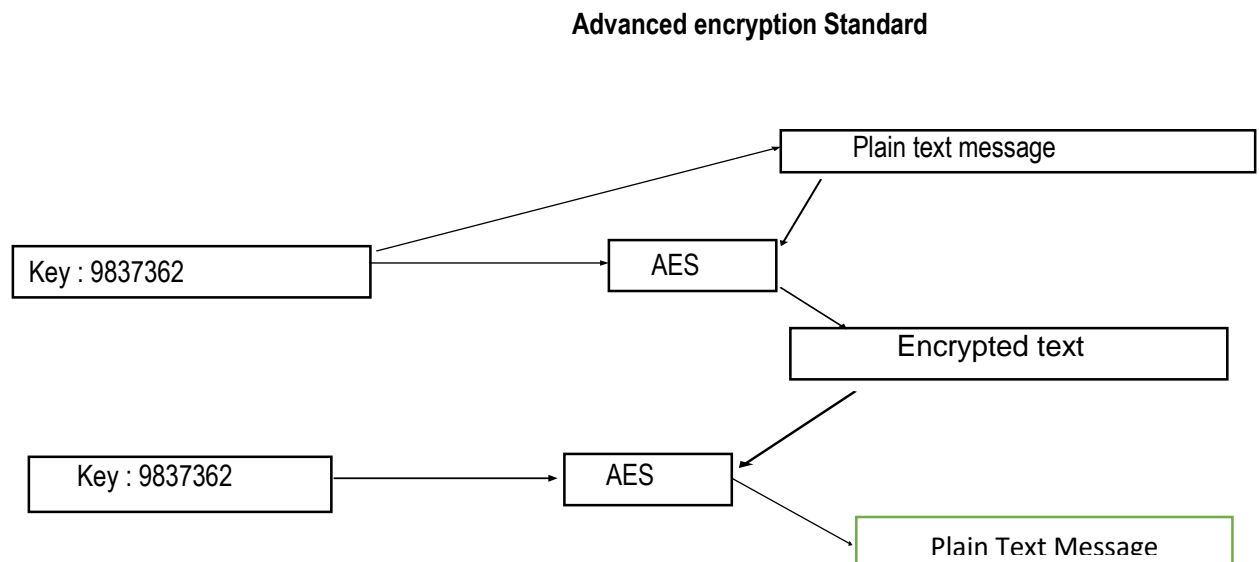


Fig : single key encryption

Unit-7 Query Processing

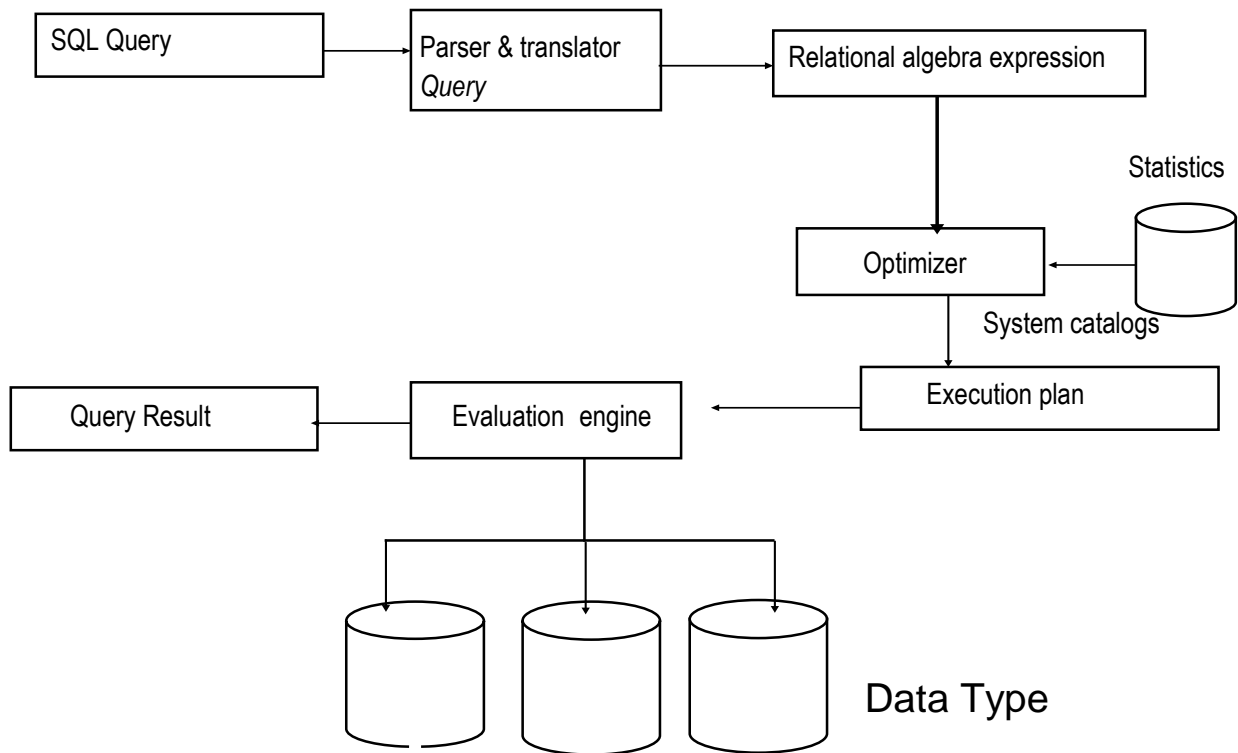
Query Processing:

- All database systems must be able to respond to request for information from the user- I.e process queries.
- Obtaining the desired information from a database system in a predictable and reliable fashion is the scientific art of query processing.

Basic steps in processing on SQL Query

Basic Steps in Query Processing

- Parsing and translation
- Optimization
- Evaluation



1. Parsing and translation

— Translate the query into its internal form. This is then translated into relational algebra.

— Parser checks syntax, verifies relation.

2. Optimization

— SQL is a very high level language:

- The users specify what to search for- not how the search is actually done
- The algorithms are chosen automatically by the DBMS.

— For a given SQL query there may be many possible execution plans.

— Amongst all equivalent plans choose the one with lowest cost.

— Cost is estimated using statistical information from the database catalog.

3. Evaluation

— The query evaluation engine takes a query evaluation plan, executes that plan and returns the answer to that query.

Parsing and translating:

- Translate the query into its internal form (parse free)
- This is then translated into an expression of the relational algebra.
- Parser checks syntax, validates relations, attributes and access permissions.

Query interpretation (Query translation)

Given a query there are a number of ways to process the query; in SQL or relational algebra we can have equivalent expressions. Each method of expressing a query suggests a strategy for processing the query system must transform user query to an equivalent more efficient query- optimization similar to compiler code optimization. In relational models, commercial database system provide a query optimizer there by reducing the burden on the users.

i) Translating SQL Queries into Relational Algebra (Equivalence to expression)

- SQL query is first translated into an equivalent extended relational algebra expression.
- SQL queries are decomposed into query blocks, which form the basic units that translated into the algebraic operators & optimized.
- Query block contains a single SELECT FROMWHERE expression, as well as Group by and Having clauses.
- Nested queries within a query are identified as separate query block.

Query optimization:

- The query optimization or more accurately improving query processing strategy helps in reducing query execution time and overhead.
- Find the “cheapest” execution plan for a query.
- Query optimization is a function of many relational database management systems in which multiple query plans for satisfying a query are examined and a good query plan is identified.
- The set of query plans examined is formed (e.g. primary index access, secondary index access, full file scan) and various relational table join techniques (e.g, merge join,hash join, product join).

Objectives of query optimization

- Main objective is to retrieve data quickly.
- Query optimization is the refining process in database administration and it helps to bring down speed of execution.
- Most of the databases after are built and filled with data, and used come down on speed.
- The time taken to execute a query and return results exponentially grows as the amount of data increases in the database leading to more waiting times on the user, and application sides.

There are two types of optimization:

- These consist of logical optimization which generates a sequences of relational algebra to solve the query.
- In addition, there is physical optimization which is used to determine the means of carrying out each operation to join.

Join strategies:

- The joint operation can be implemented in a variety of ways.
- In terms of disk accesses, the join operations can be very expansive, so implementing.
- Utilizing efficient join algorithms is critical in minimizing a query is execution time.

The following are 4 well-known types of join algorithms.

i) Nested-loop join (Brute force)

For each record t in R (outer loop) retrieve every record D for S (inner loop) and test whether the two record satisfy the join condition $t[A]=s[e]$.

ii) Single-loop join:-

Single-loop join (using an access structure to retrieve the matching records)

- If an index (or hash key) exists for one of the two join attributes (e.g B of S) retrieve each record t in R , one at a time (single loop), and then use the access structure to retrieve directly all matching records S from S that satisfy $S[B]=t[A]$.

iii) Sort-merge join:-

- If the records of R and S are physically sorted (ordered) by value of the join attributes A and B , respectively, we can implemented join in the most efficient way.
- Both files are scanned concurrently in order of the join attributes, matching the records that have the same values for A and B .
- If the files are not sorted they may be sorted first by using external sorting.
- Pairs of file blocks are copied into memory buffers in order and records of each file are scanned only once each for matching with the other file if A and B are key attributes.
- The method is slightly modified in case where A and B are not key attributes.

iv) Query Decomposition:

- The basic idea behind query decomposition is to break a complex query into simpler sub queries which can more easily be evaluated.

Query decomposition consists of 4 steps:

1. **Normalization:** Transform query to a normalized form
2. **Analysis:** Detect and reject "incorrect" queries; possible only for a subset of relational calculus
3. **Elimination of redundancy:** Eliminate redundant predicates
4. **Rewriting:** Transform query to optimize query

Unit-8

Filing and file structure

File:-

The file consist of records and the record may consist of several fields the typical operations that may be performed on the information stored in the file are as follows.

Retrieve:

To find the record having a particular value of the particular field or where the field values satisfy certain conditions.

Insert:

Insert a record at some specific locations.

- **Delete :**
Deletes a particular record.
- **Update :**
Modifies the field values of a record.

Factors affecting physical organization of Data:

- Efficient use of storage.
- Minimization of Data Redundancy.
- Expandability
- Recovery from data loss.

Buffer Management:-

- The buffer is that part of the main memory available for the storage of the contents of some of the blocks.
- The subsystems responsible for the allocation of buffer space is called the buffer manager.
- The buffer manager services all request made by the file management system for blocks on the file currently being operated upon by the DBMS.
- If a requested block is already in the buffer, a address of the block in the main memory is passed on to the file manager & subsequently to the DBMS.
- The most commonly used technique for buffer management is the buffer cache.

File Organization:

A file organization essentially measure organization of records in the file. Some basic file organization techniques are given below.

- Heap (or pile)
- Sequential
- Indexed sequential
- Direct (or hashed)

• Heap (or pile or unordered):

Basically theses files are unordered files. It is the simplest & most basic type. These files consist of randomly ordered records. The records will have no particular order. The operations we can perform on the record are insert, retrieve and Delete. The features of the heap file or the pile File organization are

- New records can be inserted in any empty space that can accommodate them.
- When old records are deleted, the occupied space becomes empty & available for any new insertion.
- If updated records grow; they may need to be relocated (moved) to new empty space. This needs to keep a list of empty space.

Advantages of heap files

- this is a simple file organization method.
- Insertion is some how efficient.
- Good for bulk-loading data into a table best if the file scans are common or insertions are frequent.

Disadvantages of heap files.

- Retrieval requires a linear search and is inefficient.
- Deletion can result in unused space need for reorganization.

Sequential file organization:

- The most basic way to organize the collection of records in a file is to use sequential organization.
- Records of the file are stored in sequence by the primary key field values. They are accessible only in the order stored, i.e., in the primary key order.
- This kind of file organization works well for tasks which need to access nearly every record in a file, e.g. payroll
- Sequential files are inefficient for random access, however, are suitable for sequential access.
- A sequential file can be stored on device like magnetic tape that allow sequential access.

Advantages of sequential file organization.

- It is the fast and efficient which dealing with large volumes of data that need to be processed periodically (batch system).

Disadvantages of sequential file organization:

- Requires that all new transactions be sorted into the proper sequence for sequential access processing.
 - Locating, sorting, modifying, deleting or adding records in the file require rearranging the file.
- The method is too slow to handle applications requiring immediate updating or responses.

Indexed (indexed sequential) file organization.

- It organizes the file like a large dictionary i.e; records are stored in order of the key but an index is kept which also permits a type of direct access.
- The records are stored sequentially by primary key values and there is an index built over the primary key field.
- An index is set of index value, address pairs. Indexing associated a set of objects to a set of orderable quantities that are usually smaller in number or their properties.
- Thus an index is a mechanism for faster search.

Hashed file organization :

- Hashing is the most common form of purely random access to a file or data base.
- It is also used to access columns that do not have an index as an optimization technique.
- Hash functions calculate the address of the page in which the records is to be stored based on one or more filed in the record.
- The record in a hash file appear randomly distributed across the available space.
- It requires some hashing algorithm and the technique.
- Hashing algorithms converts a primary key value into record address .

Advantage of hashed file organization :

- Insertion or search on hash key is fast.
- Best if equality search is needed on hash key.

Disadvantages of hashed file organization :

- It is a complex file organization method.
- Range search is slow.
- It suffers from disk space overhead.
- Unbalanced buckets degrade performance.

B+ Trees

A B+ tree or B plus tree is a type of tree which represents sorted data in a way that allows for efficient insertion, retrieval and removal of records, each of which is identified by a key.

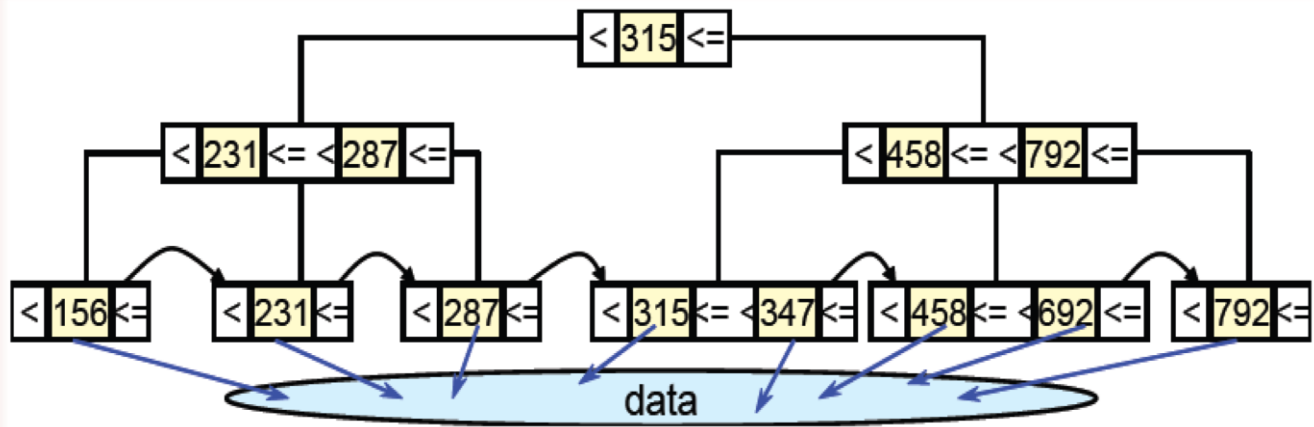
In practice, each node or element on the tree would contain an index element much like those in given figure A. That is, each element would contain the key value, a pointer to the rest of the data, and two link pointers.

For the particular tree in Figure A , each element has at most two links. One link (the line to the left) points to elements that have lower values.

The other link (line to the right) points to elements that have a value greater than or equal to the value in the node. The root is the highest node on the tree.

The bottom nodes are called leaves because they are at the end of the tree branches.

Fig: B+ Tree



Drawbacks

- One criticism has been that the coding is relatively complex.

Unit- 9 Crash Recovery

Crash Recovery:

- Recovery restoring the database to a state that is known be correct after some failure has rendered the current state.
- Correct– does not violate any integrity rule.
A database is correct if it satisfies the logic AND of all known rules.
- Failure- can be local or global.

Importance of DBMS Recovery

- Recovery—that is, the return to a fully operational environment after a hardware or software failure—is an important process.
- Moreover, the effects of a system failure on the organization must be curtailed to minimize any substantial financial loss.
- Actions must be taken to prevent DBMS failures or resolve them quickly if they occur.
- A review of DBMS recovery ensures adherence to appropriate practices and procedures and minimizes business losses.

Categories of failures

There are many causes of DBMS failure. When a DBMS fails, it falls into an incorrect state and will likely contain erroneous data. Typical causes of DBMS failures include errors in the application program, an error by the terminal user, an operator error, loss of data validity and consistency, a hardware error, media failures, an error introduced by the environment, and errors caused by mischief or catastrophe. Type of failures are given below:

- **Transaction Failure.**
Transaction failures occur when the transaction is not processed and the processing steps are rolled back to a specific point in the processing cycle. In a distributed data base environment, a single logical data base may be spread across several physical data bases. Transaction failure can occur when some, but not all, physical data bases are updated at the same time.
- **System Failure.**
System failure can be caused by bugs in the data base, operating system, or hardware. In each case, the Transaction processing is terminated without control of the application. Data in the memory is lost; however, disk storage remains stable. The system must recover in the amount of time it takes to complete all interrupted transactions. At one transaction per second, the system should recover in a few seconds. System failures may occur as often as several times a week.
- **Media Failure.**
Disk crashes or controller failures can occur because of disk-write bugs in the operating system release, hardware errors in the channel or controller, head crashes, or media degradation. These failures are rare but costly.

Recovery facilities:

DBMS should provide following facilities to assist with recovery.

- Backup mechanism, which makes periodic backup copies of database.
- Logging facilities, which keep track of current state of transactions and database changes.
- Check point facility, which enables updates to database in progress to be made permanent.
- Recovery manager, which allows DBMS to store the database to a consistent state following failures.

Log based recovery

- Every action starting from the database start up and also each and every transaction is recorded in a log file step by step
- In case, if within an ongoing transaction, the system crashes or may be for some reason u cannot complete the transaction, then the database will remain in an inconsistent state
- So using the log files we can return back to our previous state as if nothing has happened to the database. and also to recover some data log files.
- There are various types of log files like redo log files, event viewer log files etc.

Shadow Paging

Shadow paging is an alternative to log-based recovery; this scheme is useful if transactions execute serially

Idea: maintain *two* page tables during the lifetime of a transaction – the **current page table**, and the **shadow page table**

Store the shadow page table in nonvolatile storage, such that state of the database prior to transaction execution may be recovered.

- ★ Shadow page table is never modified during execution

To start with, both the page tables are identical. Only current page table is used for data item accesses during execution of the transaction.

Whenever any page is about to be written for the first time

- ★ A copy of this page is made onto an unused page.
- ★ The current page table is then made to point to the copy
- ★ The update is performed on the copy

Sample Page Table



Data Backup

- A **data backup** is the result of copying or archiving files and folders for the purpose of being able to restore them in case of data loss.
- Data loss can be caused by many things ranging from computer viruses to hardware failures to file corruption to fire, flood, or theft (etc.). If you are responsible for business data, a loss may involve critical financial, customer, and company data. If the data is on a personal computer, you could lose financial data and other key files, pictures, music, etc. that would be hard to replace.

As part of a data backup plan, you should consider the following:

- What data (files and folders) to backup
- What compression method to use
- How often to run your backups
- What type of backups to run*
- What kind of media on which to store the backups
- Where to store the backup data for safekeeping

Remote Backup

- In storage terminology, a remote backup refers to an online managed backup service for backing up data to a remote, cloud-based server ("cloud backup").

To update or restore a cloud backup, customers need to use the service provider's specific client application or Web browser interface. Files and data can be automatically saved to the cloud backup service on a regular, scheduled basis, or the information can be automatically backed up anytime changes are made.

Unit 10.

Transaction Processing and Concurrency Control

What is database transaction?

In database, a **database transaction** is a logical unit of database operations which are executed as a whole to process user requests for retrieving data or updating the database. Transactions in a database environment have two main purposes:

- To provide reliable units of work that allow correct recovery from failures and keep a database consistent even in cases of system failure, when execution stops (completely or partially) and many operations upon a database remain uncompleted, with unclear status.
- To provide isolation between programs accessing a database concurrently.

Properties of database transactions are often referred to by the acronym **ACID**.

- **Atomicity:** A transaction is an indivisible unit. It is included in all operations or have done or not done.
- **Consistency:** Transaction must keep the database from one consistent state to another consistent state variable.
- **Isolation:** A transaction is implemented and cannot be interfered with by other transactions. That is, an internal operation of a transaction and the use of the data is isolated with other transaction, the concurrent implementation of all transactions cannot interfere between with each other.
- **Durability:** It's also called permanence. It refers to a transaction which is submitted, and the data in the database it changes should be permanent. The next operation or other faults should not have any impact on them.

Concurrency control

Concurrency control is a database management systems (DBMS) concept that is used to address conflicts with the simultaneous accessing or altering of data that can occur with a multi-user system. Concurrency control, when applied to a DBMS, is meant to coordinate simultaneous transactions while preserving data integrity. The Concurrency is about to control the multi-user access of Database.

Schedules

In the fields of databases and transaction processing (transaction management), a **schedule** (or **history**) of a system is an abstract model to describe execution of transactions running in the system. Often it is a list of operations (actions) ordered by time, performed by a set of transactions that are executed together in the system. If order in time between certain operations is not determined by the system, then a partial order is used. Examples of such operations are requesting a read operation, reading, writing, aborting, committing, requesting lock, locking, etc. Not all transaction operation types should be included in a schedule, and typically only selected operation types (e.g., data access operations) are included, as needed to reason about and describe certain phenomena.

Serializability:

- Serializability is a property of a transaction schedule (history). It relates to the isolation property of a database transaction.
- Serializability of a schedule means equivalence (in the outcomes, the database state, data values) to a serial schedule (i.e; sequential with no transaction overlap in time) with the same transactions. It, is the major criterion for the correctness of concurrent transactions schedule, and thus supported in all general purpose database system.

Locking:-

A lock is a system object associated with the shared resource such as data item of an elementary type, a row in a database or a page or memory.

In a database, a lock on a database object (a data access lock) may need to be acquired by a transactions before accessing the object.

Correct use of locks prevents undesired, incorrect or inconsistent operation on shared resources by other concurrent transactions.

When a database object with an existing lock acquired by one transactions needs to be accessed by another transactions the existing lock for the objects and the type the intended access are checked by the system.

Two phase locking (locking):

In database and transactions processing, two phase locking (2pl) is a concurrency control method that guarantees serializability. It is also the name of the resulting set of the data base transaction schedules (histories).

The protocol utilizes locks, applied by a transaction to data, which may block (interpreted as signals to stop) other transaction from accessing the same data during the transactions life. By the 2PL protocol locks are applied and removed in two phases

Expanding phase:

Locks are acquired and no locks are released.

Shrinking phase:

Locks are released and no lock are acquired.

Two major types of locks are utilized:

Write lock (exclusive lock) is associated with a database object by a transaction (terminology “the transaction locks the object, “or” acquires lock for it”) before writing (inserting/modifying/deleting) this object.

Read lock (shared lock) is associated with a database object by a transaction before reading (retrieving the state of) this object.

Timestamp based concurrency control:

In computer science, a **timestamp-based concurrency control** algorithm is a non-lock concurrency control method. It is used in some databases to safely handle transactions, using timestamps.

- Every timestamp value is unique and accurately represents an instant in time.
- No two timestamps can be the same.
- A higher-valued timestamp occurs later in time than a lowervalued timestamp.

Unit 11

Advanced Database Concepts

OBJECT-ORIENTED DATABASE MODEL:

In the object oriented data model the (OODM). Both data and their relationship are contained in a single structure known as an object. An object includes information about relationship between the facts within the object, as well as information about its relationship with other objects. An object is the abstraction of the real-world entity. An object represents only one occurrence of entity. Attributes describe the property of an object. Objects that are similar in characteristics are grouped in class.

Class: is a collection of similar objects with shared structure (attributes) and behavior (method).

Method: represents a real-world action such as finding a selected person's name, changing person's name or printing a person's address. Inheritance is the ability of an object within the class hierarchy to inherit the attributes and methods of the class above it.

Object-relational database/Model

An **object-relational database (ORD)**, or **object-relational database management system (ORDBMS)**, is a database management system (DBMS) similar to a relational database, but with an object-oriented database model: objects, classes and inheritance are directly supported in database schemas and in the query language. In addition, just as with pure relational systems, it supports extension of the data model with custom data-types and methods.

An object-relational database can be said to provide a middle ground between relational databases and object-oriented databases (OODBMS). In object-relational databases, the approach is essentially that of relational databases: the data resides in the database and is manipulated collectively with queries in a query language; at the other extreme are OODBMSes in which the database is essentially a persistent object store for software written in an object-oriented programming language, with a programming API for storing and retrieving objects, and little or no specific support for querying.

Distributed Databases

A distributed database is a database in which portions of the database are stored on multiple computers within a network. Users have access to the portion of the database at their location so that they can access the data relevant to their tasks without interfering with the work of others. A centralized distributed database management system (DDBMS) manages the database as if it were all stored on the same computer. The DDBMS synchronizes all the data periodically and, in cases where multiple users must access the same data, ensures that updates and deletes performed on the data at one location will be automatically reflected in the data stored elsewhere.

Data warehouse:

A data structure that is optimized for distribution. It collects and stores integrated sets of historical data from multiple operational systems and feeds them to one or more data marts. It may also provide end-user access to support enterprise views of data.

Data Mart:

A data structure that is optimized for access. It is designed to facilitate end-user analysis of data. It typically supports a single, analytic application used by a distinct set of workers.