INTRODUCTION 41061810

A DataBase Management System (DBMS) is a collection of interrelated data and a set of programs to access those data.

Collection of Data is called as DATABASE which contains information relevant to an enterprise. The primary goal of a DBMS is to provide a way to store and retrieve database information.

PURPOSE OF DATABASE DATABASE SYSTEM.

Earlier database systems are created in response to manage the commercial data. These data is typicaly stored in files To allow users to manipulate these files various programs are written for

→ addition of new data

→ Updating the data

→ Deleting the data.

CONVENTIONAL FILE PROCESSING:

The information can be either a conventional file processing system or a database system. In the conventional file processing system, each and every sub system of information s/m will have its own set of files. As a result, there will be duplicated data between various subsystems of the information system.

The concept of the conventional file processing system is shown below.

Typical File Processing s/m

Name of application	Input	Output
Application-X	File 1, 2	Report 1, 2, 3
Application-Y	File 1, 3, 4	Report 4, 5.
Application-Z	File 5	Report 6

Drawbacks of Conventional File Processing System.

1. Data Redundancy and inconsistency.

2. Difficulty in accessing data

3. Data isolation

4. Concurrent access anomalies

5. Security problems

6. Integrity problems

Functions of DBMS

1. Data dictionary management

2. Modify, delete, insert the data.

3. Security

4. Control multi-user access management

5. Acheived data integrity

6. Transaction management

7. Data transformation & presentation

Diff [illegible] DBMS

File S/m	DBMS
1. Easy to use to store general files which require less security	DBMS is used when security constraints are high
2. Data Redundancy is more	Data Redundancy is less
3. Data Inconsistency is more	Data Inconsistency is less
4. Centralization is hard to get.	Centralization is achieved.
5. Stored unstructured data	Stores structured data.

Characteristics or Features of DBMS.

- Persistence - Permanent data stored
- Validity - Validation of field.
- Consistency - The value of data is the same at all place
- Security
- Non- Redundancy - No multiple cops.
- Independence
- Concurrency - Multiple users can share a file at same time

File Processing s/m

Name of application	Input	Output
Application-X	File 1, 2	Report 1, 2, 3
Application-Y	File 1, 3, 4	Report 4, 5.
Application-Z	File 5	Report 6

Drawbacks of Conventional File Processing System.

1. Data Redundancy and inconsistency
2. Difficulty in accessing data
3. Data isolation
4. Concurrent access anomalies
5. Security problems
6. Integrity problems

Functions of DBMS

1. Data dictionary management
2. Modify, delete, insert the data
3. Security
4. Control multi-user access management
5. Acheived data integrity
6. Transaction management
7. Data transformation & presentation

Diffe S/m vs DBMS

File S/m	DBMS
1. Easy to use to store general files which require less security	DBMS is used when security constraints are high.
2. Data Redundancy is more	Data Redundancy is less
3. Data Inconsistency is more	Data Inconsistency is less
4. Centralization is hard to get.	Centralization is achieved.
5. Stored unstructured data	Stores structured data.

Characteristics or Features of DBMS.

- Persistence - Permanent data stored
- Validity - Validation of field.
- Consistency - The value of data is the same at all place
- Security
- Non- Redundancy - No multiple cops.
- Independence
- Concurrency - Multiple users can share a file at same time

Advantages of DBMS

- Consumes less space
- Reduction of Redundancy
- Data integrity, security & continuity
- Backup and recovery process
- Data model can be developed.
- Concurrency control
- Good Performance.

Disadvantages of DBMS

- Requires large size of memory.
- Time consuming.
- Requires a processor with the high speed of data processing
- Cost of data conversion
- DB corrupted.
- Expensive & complex

Application

Applications of DBMS

- Airlines
- Universities
- Banking
- Manufacturing

VIEWS OF DATA.

Database is a collection of inter-related data and set of programs that allow users to access or modify the data..

Abstract view of the system is a view in which the system hides certain details of how the data are stored and maintained.

The main purpose of db s/ms is to provide users with abstract view of data.

The view of the system helps the user to retreive data efficiently.

Data Abstraction:

Data abstraction means retreiving only required amount of information of the system and hiding background details.

There are several levels of abstraction that simplify the user interactions with the s/m.

Physical level.

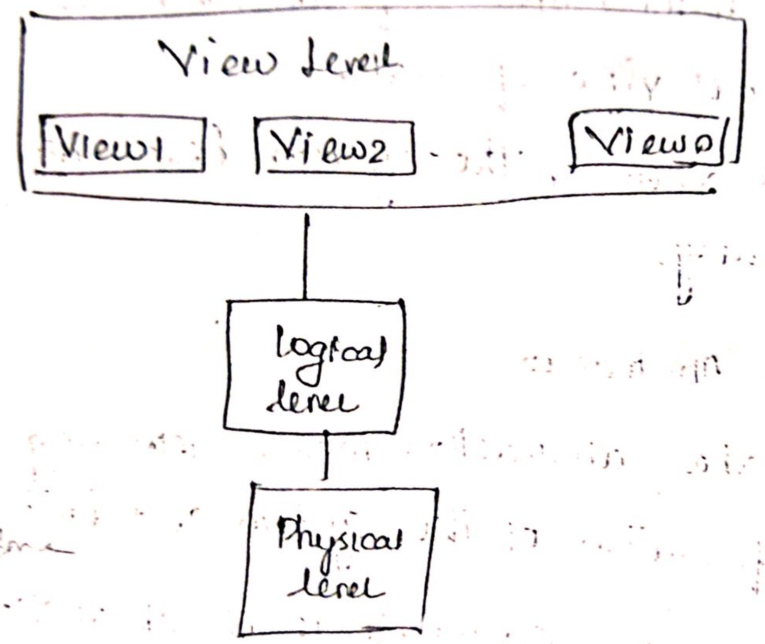
- Lowest level.
- Describes how actually the data are stored.
- Describes complex low level data structures.

Logical level.

- Next higher level.
- Describes relationship among data
- Describes the entire db in terms of small number of relatively simple structures.

View level:

- Highest level of abstraction.
- View level can provide the access to only part of the database.
- Helps in simplifying the interaction with the s/m.
- System can provide multiple views.



INSTANCES & SCHEMA.

Schema: The overall design of the database is called schema.

Eg.

Rollno	Name	marks

Instances: Collection of information at particular moment is called instances.

RollNo	Name	Marks
1	John	85
2	Peter	90

Database languages.

DDL : Data Definition Language.

It is a specialized language used to specify schema. It is a language used for creating and modifying structures of tables, views, indexes and so on.

DDL is also used to specify additional properties of data. DDL commands are create, alter, drop.

DML : Data Manipulation Language.

Enables users to access or manipulate data as organized by appropriate data model.

Types of access

- Retreival of information
- Insertion
- Deletion
- Modification.

Data Model.

Different types of data models are

- ER model
- Relational model
- Hierarchical model
- Network model.
- Object oriented model.

ER Model:

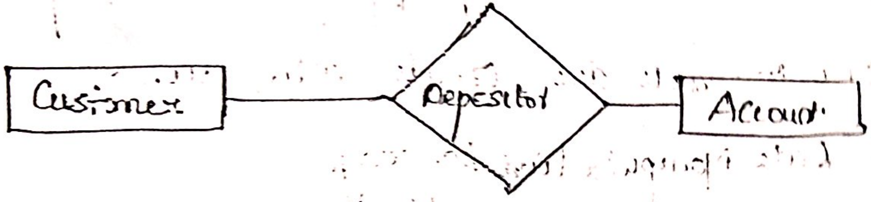
Consists of a collection of basic objects called entities and relationship among entities.

Entity: An entity is a thing or object in the real world that is distinguishable from other objects.

Relationship:

Association among several entities.

Eg



Adv:

1. Easy to develop relational model using ER model.
2. Specifies mapping cardinalities
3. Specifies keys like Primary Key.

Disadvantage

Only for db design not for implementation.

Relational Model:

Represents data and relationships among data by a collection of tables.

Eg.

Customer

Name	Street	City
John	North	Queens
Smith	Sidehill	Brooklyn
Smith	Sidehill	Brooklyn
Jim	Lakeview	Perryridge

column attributes

rows records

Advantages

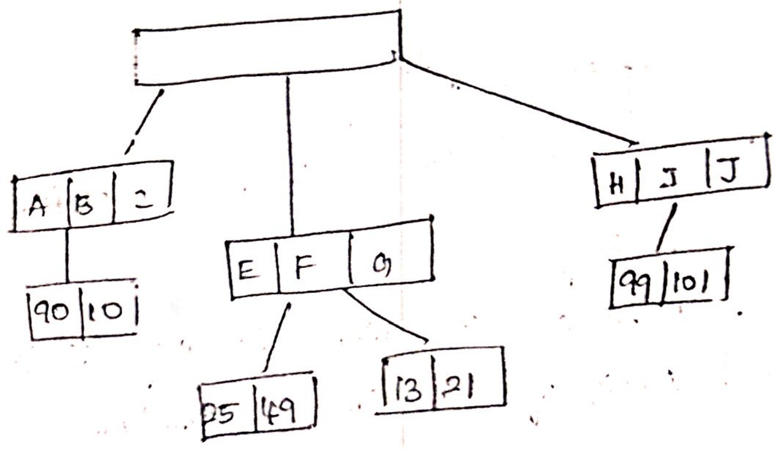
1. Structural independence.
2. Conceptual simplicity
3. Design Implementation, maintenance.
4. Operation flexibility

Disadvantages.

1. H/w & S/w Overheads.

Hierarchical Model

This model links data/records together in a tree data structure.



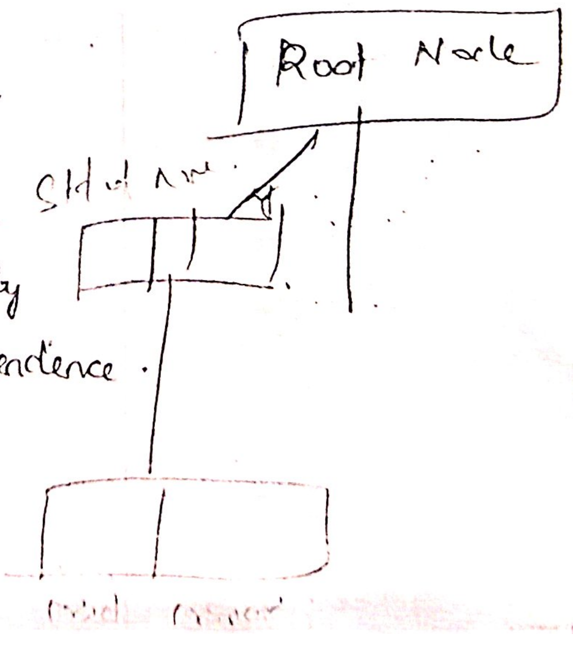
Advantages.

- High speed of access
- Ease of update.
- Simplicity.
- Data Security,
  Data Independent

Disadvantages.

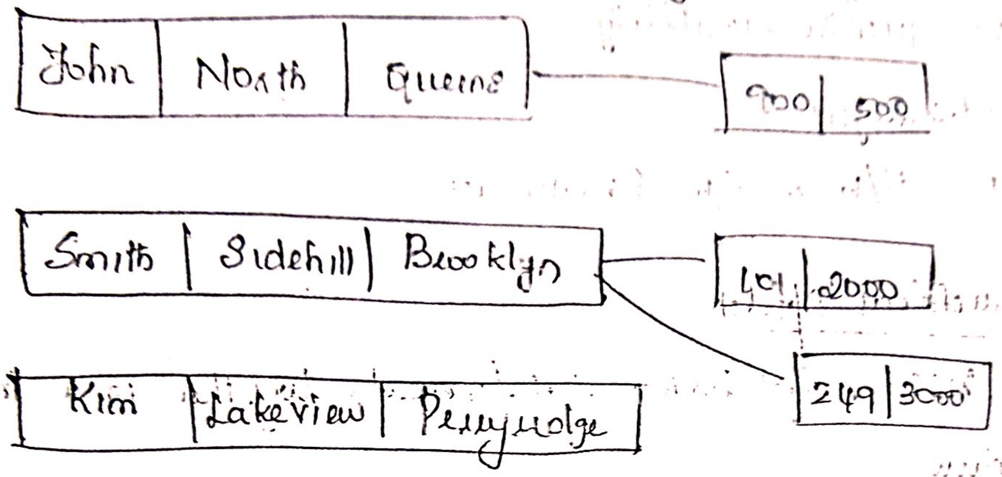
- Implementation Complexity
- Lack of structural independence.
  Difficult to manage

Organizes data in Tree Structure



Network Model.

- Based on directed graph theory.
- Replaces the hierarchical tree with a graph thus allowing more general connection among the nodes.



Adv

- Conceptual simplicity
- Capability to handle more relationship

Disadvantages.

- Detailed structural knowledge is required.
- Lack of structural independence.

Object Oriented Model.

- Based on collection of objects.
- Contains values in instance variables within the object.
- Object also contains bodies of node that operate on the object.

Adv

Code is easier to maintain

Access is easy.

DBMS Architecture

Storage manger :

A program module that provides the interface between the low level data stored in the database.

Storage manager is responsible for the interaction with file manager.

It translates various DML statements into low level file s/m commands.

Storage manager is responsible for storing, retrieving and updating data.

Components of Storage Manager.

a. Authorization and integrity manager :

Tests for satisfaction of integrity constraints and checks the authority of users.

b. Transaction Manager. Ensures the db remains in a consistent state despite of system failures and ensures concurrent transactions proceed without conflicting.

c. File Manager :

Manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

d. Buffer Manager.

Responsible for fetching data from disk storage into main memory and deciding what data to cache in main memory. It is the critical part of the db s/m, since it enables the db to handle data sizes that are much larger than the size of main memory.

Storage Manager implements several data d

a. Data files

b. Data dictionary - Stores metadata

c. Indices - Provides fast access to data items

Query Processor:

DDL Interpretor:

· Interprets DDL statements and records the definitions in data dictionary.

DML compiler:

Translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation understands.

Query is translated into any number of alternative Evaluation plans that all gives the same result.

Query Optimization:

→ Picks the lowest cost evaluation plan among the alternatives.

Query evaluation engine.

Low level instructions are executed by DML Compiler.

Naive Users:

· Unsophisticated users who interact with the system by invoking one of the application programs.

Application Programmers:

· Computer Professionals who write application program.

· They will choose many tools to develop user interfaces

· RAD tools are tools that enable an application programmer to construct forms and reports without writing a program.

Sophisticated Users:

- Interacts with the s/m without utility programs.
- Submits each query to query processor whose function is to break down DML statements into instructions that the storage manager understands

Specialized users:

Writes specialized applications that do not fit into traditional data processing framework.

DB Administrator:

Central control of both the data and programs that access the data.

Functions:

1. Schema Definition: Creates the original db. schema by executing a set of data definition statements in DDL.
2. Storage structure and access method definition
3. Schema & Physical organization modification

INTRODUCTION FOR RELATIONAL MODEL:

The relational model is the primary data model for commercial data processing applications.

A Relational database consists of a collection of tables, each of which is assigned a unique name. A row in a table represents a relationship among a set of values.

BASIC STRUCTURE:

Table or Relation:

- Table is a collection of data items arranged in rows and columns.
- Table will not have duplicate data or rows.

S-NO	Reg No	Name	Dept	Age.
1	4001	James	CSE	17
2	4002	Peter	ESE	17
3	4003	Jackson	CSE	18

Tuple or Record or Row:

- Single entry in the table is defined as row/tuple/record.
- Tuple represents a set of related data.

3	4003	Jackson	CSE	18

Attribute or Columns:

Attribute is a part of table that contains several records.

- Attribute is a part of table that contains several records.
- Each record can be broken down into several small parts of data is known as attribute.

Relation Schema:

- A Relation Schema describes the structure of the relation, with the name of the relation, attributes and their types.

Relation Instance :

- Relation instance refers to specific instance of relation.
- Instance - contains specific set of rows.

Domain:

For each attribute of relation, there is a set of permitted values called domain.

Atomic:

The domain is atomic if elements of the domain are considered to be indivisible units.

Null Attribute:

A null is a special symbol, independent of data type. This value is represented when the value of any record is missed.

Degree:

Degree refers to the total numbers of columns in a relation

Cardinality:

Cardinality refers to the total number of tuples present in a relational database.

KEYS IN DBMS.

Keys are used to establish and identify relation between tables. Each record within a table can be uniquely identified by the combination of one or more fields in a table.

Keys help to enforce integrity and identify relationship.

Why do we need keys ?

Problems :

- In real world applications, huge datas must be stored in a table.

• Tables can extend to 1000 of records stored in them:
be unsorted or unorganized.

• Any Record must be fetched.

To avoid all the above problems, keys are defined to easily identify any rows of data in a table.

Types of Keys:

There are different types of keys in DBMS. They are as follows:

a. Super Key.
b. Primary Key.
c. Candidate Key.
d. Foreign Key.
e. Alternate Key.

SUPER KEY:

Super Keys are defined a set of attributes within a table that helps to identify a record in a unique manner within a table.

Eg:

studid	Name	Contact	age
1	Akhil	9841263540	17
2	Akhil	8838562190	17
3	John	7452612380	18
4	James	8896523122	17
5	Peter	9840096070	18

studid, {studid, name}, {studid, name, contact} contact.

PRIMARY KEY :

A Primary key is a special relational database table column designated to uniquely identify all table records. The main features are a primary key must contain a unique value for each row of data. It cannot have null values.

Primary key

Reg.No	Name	Dept	Age
4001	James	CSE	18
4002	Peter	ECE	18
4003	Jackson	ECE	17
4004	Williams	Mech	19.

CANDIDATE KEY

Candidate key is a single attribute or least or minimal combination of attributes that uniquely identify each record in a table. Every candidate key is a super key. But every super key is not candidate key.

Candidate Key

studid	name	contact	age
001	Akhil	9841563820	17
002	Akhil	9962589161	17
003	Pete	8837116320	19
004	Jones	7409623450	17

FOREIGN KEY.

Foreign key is a single attribute or collection of attributes in one table that refers to the primary key of other table. Foreign key refers to the primary key.

The table containing the primary key is called parent table and the table containing the foreign key is called child table.

Eg: Refer Database Integrity Example.

INTEGRITY CONSTRAINTS:

Database integrity means correctness or accuracy of data in the database. A database may have integrity constraints.

Eg:

1. The student ID and the department ID must consists of two digits.
2. Every student ID must start with a number.

Entity Integrity Rule:

In a table, the value of attribute of primary key cannot be null. The Null value always deal with incomplete or missing values. The primary key value helps in uniquely identifying every row in the table. Thus if the users of the database want to retrieve any row from the table, the value of that key must be known. Hence it is necessary that the primary key should not have the Null value.

Referential Integrity Rule:

Referential integrity refers to the accuracy and consistency of data within a relationship. In relationship data is linked between two or more tables. This is acheived by having the foreign key reference a primary key value in the parent table.

We must ensure that data on both sides of the relationship remain intact.

The Referential Integrity rule states that whenever a foreign

Key value is used it must reference a valid, existing primary key in the parent table.

Referential Integrity enforces the following three rules.

Consider two tables : Employees table, Managers table.

Rule 1 :

If a primary key for a record in the Manager table changes, all corresponding records in the employees table are modified.

Rule 2 :

If a record in the Managers table is deleted, all corresponding records in the Employees table are deleted.

Advantages of Referential Integrity :

1. Prevents the entry of duplicate data
2. Prevents one table from pointing to a non-existent field in another table.
3. Guaranteed consistency between the partnered tables.
4. Prevents the deletion of a record that contains a value referred to by a foreign key in another table.
5. Prevents the addition of a record to a table that contains a foreign key unless there is a primary key in the linked table.

Database Integrity:

The foreign key [illegible] key in [illegible] table that [illegible] primary key of another table.

Example for foreign key:

Customer

Cust Id	Name	City
C101	AAA	Chennai
C102	BBB	Mumbai
C103	CCC	Pune

Order

Order ID	Description	Cust ID
111	Bolts	C103
222	Nuts	C103
233	Beans	C101
444	Screws	C102
555	Disks	C101

Note that the CustID column in the order table points to the CustID column in the customer table.

CustID → Primary key in customer table.

CustID column is made as a reference to relate the two relations. So it acts as a foreign key.

RELATIONAL ALGEBRA :

A Query language is a language in which a user requests information from the database. Query languages are categorized as

a. Procedural Languages.

b. Non Procedural Languages.

In Procedural language, the user instructs the system to perform a sequence operations on the database to compute the desired result.

• In a Non-Procedural Language, the user describes the desired information without giving a specific procedure for obtaining that information.

The Relational Algebra is a procedural query language. It consists of a set of operations that take one or two operations as input and produce a new relation as their result.

Formal Definition of Relational algebra:

A basic expression in the relational algebra consists of either one of the following :

a. Relation in the database.

b. A constant relation.

The fundamental operations of Relational algebra are.

a. select b. project c. union d. set difference

e. cartesian product f. rename.

The additional operations are

a. set intersection b. natural join c. division

d. assignment.

The select, project and rename operations are called unary operations, because they operate on one relation. The other three operations operate on pair of relations and hence they are called as binary operations

FUNDAMENTAL OPERATIONS:

1) Select Operation:

- The select operation selects tuples that satisfy a given predicate.
- The select operation is represented as follows:

$$\sigma_{\langle \text{select condition} \rangle} (R)$$

The symbol σ is used to describe or denote the select operator. The ⟨select condition⟩ is an expression specified on the attributes of the relation R.

The expression specified in the selection condition is made up of number of clauses of the form:

⟨attribute name⟩ ⟨comparison operator⟩ ⟨constant value⟩

(or)

⟨attribute name⟩ ⟨comparison operator⟩ ⟨attribute name⟩.

where,

→ attribute name is the name of the attribute (column) of the relation R.

→ comparison operator is one of the following comparison operators

$=, \neq, <, \leq, \geq, >$.

→ constant value - Value from the attribute domain. Clauses are connected by boolean operators AND, OR and NOT consider the following Relation

book id	title	Author	Publisher	Year	Price
B0001	DBMS	Korth	McGrawHill	2000	250
B0002	CA	Ulman	Pearson	2004	350
B0003	OS	Rambaugh	Oxford	2003	450
B0004	PQT	Sabista	Pearson	2000	500.

Query 1 : Display Books published in year 2000.

$\sigma_{year = 2000}$ (Book)

book_id	title	Author	Publisher	Year	Price
B0001	DBMS	Korth	McGraw Hill	2000	350
B0004	PQT	Sabista	Pearson	2000	500

Query 2 : Display all the Books having price greater than 300.

$\sigma_{price > 300}$ (Book)

book_id	title	Author	Publisher	Year	Price
B0002	CA	Ulman	Pearson	2004	350
B0003	OS	Rambaugh	Oxford	2003	450
B0004	PQT	Sabista	Pearson	2000	500.

Query 3 : Select the tuples for all books whose publishing year is 2000 or price is greater than 300.

$\sigma_{(year = 2000) \text{ OR } (price > 300)}$ (Book)

Result will be the entire table.

Output for Query 4

book_id	title	Author	Publisher	Year	Price
B0004	PQT	Sabista	Pearson	2000	500

Query 4: Select the tuples for all books whose publ... and price is greater than 300.

$\sigma_{(year = 2000) \text{ and } (price > 300)}$ (Book)

Project operation:

The Project operation selects certain columns from a table while discarding others. It removes any duplicate tuples from the result relation.

The Project operation is represented as follows

$\pi_{<attribute\ list>}(R)$

The π symbol is used to denote the project operation and the attribute list is a list of attributes of the Relation R. The result of the project operation has only the attributes specified in the attribute list and in the same order as they appear in the list.

Query 1: Display all titles with author name.

$\pi_{Title, Author}(Book)$

Title	Author
DBMS	Korth
CA	Ulman
OS	Rambaugh
PQT	Subista

Query 2: Display all book titles with authors and price.

$\pi_{Title, Author, price}(Book)$.

Title	Author	Price
DBMS	Korth	250
CA	Ulman	350
OS	Rambaugh	450
PQT	Subista	500

Composition of Operations :

The relational operations select and project can be combined to form a complicated query.

Query 1 : Display the title of books having price greater than 300.

$$\Pi_{title}(\sigma_{price > 300}(Book))$$

Title.
CA
DS
PGT

Union Operation :

Two relations are said to be union compatible if the following conditions are satisfied.

- The two relations/tables must contain the same number of columns.
- Each column of the first relation/table must be either the same data type as the corresponding column of the second relation/table or convertible to the same data type as corresponding column of the second.

Consider the two relations.

Depositor

customer_name	city
Hayes	Pune
Johnson	Mumbai
Jones	Solapur
Lindsay	Nasik
Smith	Pune
Turner	Mumbai

Borrower

customer name	city
Adams	Mumbai
Curry	Pune
Hayes	Pune
Jackson	Solapur
Jones	Solapur
Smith	Pune
Williams	Kolhapur

The result of union operation is denoted by

Depositor ∪ Borrower

The above query includes all tuples that are either in Depositor or borrower or in both. Duplicates are eliminated.

The Result of union operation is

Depositor ∪ Borrower

customer_name	city
Hayes	Pune
Johon	Mumbai
Jones	Solapur
Lindsay	Nashik
Smith	Pune
Turner	Mumbai
Adams	Mumbai
Curry	Pune
Jackson	Solapur
Williams	Kohalpur.

Set difference operation

The difference operation is denoted by Depositor - Borrower. The result of the difference operator is the relation that contains all tuples in the Depositor but not in borrower.

customer_name	city
Johson	Mumbai
Lindsay	Nashik
Turnir	Mumbai.

Cartesian Product :

Cartesian Product is also known as CROSS PRODUCT or CROSS JOINS. It is denoted by 'x'.

The Cartesian product of two relations A and B is denoted by A x B. The resulting relation will have A+B columns and n + m tuples.

Consider the following two tables.

publisher_info

publisher_code	Name
P001	Mc GrawHill
P002	PHI
P003	Pearson

book_info

book_id	Title
B001	DBMS
B002	Compiler

The result of cartesian product will be as follows.

publisher_info × book_info

publisher_code	Name	book_id	Title
P001	Mc GrawHill	B001	DBMS
P002	PHI	B001	DBMS
P003	Pearson	B001	DBMS
P001	Mc Graw Hill	B002	Compiler
P002	PHI	B002	Compiler
P003	Pearson	B002	Compiler

Rename Operation.

The relation or the attributes or both can be renamed. The general rename operation can take the following representation.

→ ρ_S (new attribute names) (R)

→ ρ_S (R)

→ ρ (new attribute names) (R)

The symbol ρ (rho) is used to denote the RENAME operator.

'S' ⇒ new relation

'R' ⇒ original relation.

- The first representation renames both the relation and its attributes.
- The second representation renames the relation.
- The third representation renames only the attributes.

Example

Consider a Book Relation with the following attributes

bookname, authorname, publishing-year, bookprice.

Rename the relation name as bookinfo and rename the attributes.

$\rho_{bookinfo\,(bname,\ aname,\ p\text{-}year,\ bprice)}(Book)$

Rename the relation name as bookinfo

$\rho_{bookinfo}(Book)$

Rename the attributes :

$\rho_{(bname,\ aname,\ p\text{-}year,\ bprice)}(Book)$.

ADDITIONAL OPERATIONS IN RELATIONAL ALGEBRA.

Certain common queries are lengthy to express. Therefore additional queries/operations simplify the lengthy queries.

Set intersection:

The result of intersection operation is a relation that includes all tuples that are both in Depositor and Borrower

Depositor ∩ Borrower.

Depositor ∩ Borrower	
customer-name	city
Hayes	Pune
Jones	Solapur
Smith	Pune

Natural Join operation.

The natural-join is a binary operation that allows us to combine certain selection and a cartesian product into one operation. It is denoted by ⋈ symbol.

The natural join operation forms a cartesian product of two arguments, performs a selection forcing equality on those attributes that appear in both relation schema and finally removes the duplicates.

Example :

Consider the two relations.

empcode	empname
E01	Hari
E02	Om
E03	Smith
E04	Jay

empcode	salary
E01	2000
E02	5000
E03	7000
E 04	10000

Display the names of all employees with salary

Query without using join operation.

$\Pi_{empname,\ salary} (\sigma_{employee.empcode\ =\ salary.empcode}$ (Employee × Salary

Query without join:

π empname, salary (Employee ⋈ Salary)

empname	salary
Hari	2000
Om	5000
Smith	7000
Jay	10000.

• division operation:

Consider the following tables.

student

sno	name	address	phone
1	Ram	Delhi	9442356182
2	Peter	Bombay	9840515694
3	Jancy	Calcutta	9943216392
4	Vasanth	Chennai	9710543216

student-sports

sno	sports
1	Tennis
2	Cricket
3	Cricket
4	Tennis

All-sports

sports
Tennis
Cricket

The division operation A ÷ B can be applied if and only if.:

• Attributes of B is proper subset of Attributes of A.

• The relation returned by division operator will have attributes

= (All attributes of A − All attributes of B).

• The relation returned by division operator will return those tuples from relation A which are associated to every B's tuple

SQL FUNDAMENTALS

[illegible]

used to communicate with the relational database management

Characteristics of SQL:

- SQL is extremly flexible.
- Uses a free form syntax that gives the user the ability to structure SQL statements
- Each SQL request is parsed by the RDBMS before execution, to check for proper syntax and to optimize the request.

Advantages of SQL:

- SQL is a high level language that provides a greater degree of abstraction than procedural languages.
- SQL enables the end users and systems personnel to deal with a number of DBMS where it is available.
- Applications written in SQL can be easily ported across systems
- SQL specifies what is required and not how it should be done.

SQL Literals:

The four literals are :

a. Character string:

Written as sequence of characters enclosed in single quotes

Eg: 'Computer'.

b. Bit String :

Bit string is written as a sequence of 0's and 1's enclosed in single quotation preceded by a letter B.

Eg: B'101011010'.

3. Exact numeric:

Written as signed or unsigned decimal number possibly with a decimal point.

Eg. 7, 70.00 +99.99

4. Approximate numeric:

Written as exact numeric literals followed by letter 'E'.

Eg: 5E5, +55E-5.

Types of SQL commands:

SQL provides set of commands for a variety of tasks including the following:

- Querying data
- Updating, inserting and deleting data
- Creating, modifying data
- Providing data integrity and consistency.

SQL statements are divided into following categories.

1. Data Definition Language - Create, alter and delete
2. Data Manipulation Language - Insert, modify and delete
3. Data Query Language - To query one or more tables.
4. Data Control Language - Commands that control the user access. Commands are commit, roll back.

Operators in SQL:

Arithmetic Operators:

Used in SQL expressions to add, subtract, multiply, divide and negate data values.

Unary operators +, - Denotes positive or negative expression

Binary operators * / + -

iii) Comparison operators: Used to compare one expression with another. The operators are: =, !=, >, <, >= <=. The other operators are:

Operator	Description
IN	Equal to any member of set
NOT IN	Not Equal to any member of set
IS NULL	Test for NULL
IS NOT NULL	Test for anything other than NULL.
LIKE	Returns true when the first expression matches the pattern of the second expression
ALL	Compares a value to every value in the list
ANY, SOME	Compares a value to any value in the list.
EXISTS	True if subquery returns at least one row.
BETWEEN x and y	>= x and <= y.

Logical Operators: Used to produce a single result from combining the two separate conditions.

Operator	Description
AND	Returns True if both the conditions are true.
OR	Returns True if any one condition is true.
NOT	Returns True if the condition is false.

Set operators: Combine the results of two separate queries into a single result.

Operator	Description.
UNION	Returns all distinct rows from both queries.
INTERSECT	Returns common rows selected by both queries.
MINUS	Returns all distinct rows that are in the first query not in the second.

Data Definition Language Commands.

SQL is a fourth-generation high-level non procedural language, a user requests data from the DBMS. The SQL language uses English-like commands such as CREATE, INSERT, DELETE, UPDATE and DROP.

CREATE TABLE:

This command is used to create a new relation and the corresponding syntax is :

```
CREATE TABLE relation_name ( field1 datatype(size),
                             field2 datatype (size),
                             ... fieldn datatype (size));
```

CREATE TABLE AS SELECT;

This type of create command is used to create the structure of a new table from the structure of existing table.

```
CREATE TABLE relation_name1 (field1, field2, ..., fieldn)
    AS SELECT field1, field2, ..., field n FROM relation_name2;
```

ALTER TABLE ... ADD ...

This command is used to add some extra columns into existing table.

ALTER TABLE relation_name ADD (new field1 datatype
newfield2 datatype (size), ...
newfieldn datatype (size));

ALTER TABLE ... MODIFY :

This command is used to change the width as well as data type of existing relations.

ALTER TABLE relation_name MODIFY (field1 new datatype(size)
field$_2$ new datatype(size), ...
fieldn new data type (size));

DROP TABLE :

This command is used to delete a table.

DROP TABLE relation_name.

RENAMING A TABLE :

We can rename a table by using this command.

RENAME old tablename to new tablename;

TRUNCATING A TABLE:

Truncating a table is removing all records from the table. The Delete statement which can be used to remove one or more rows from a table. Truncation releases storage space occupied by the table, but deletion does not.

TRUNCATE TABLE table_name;

EXAMPLE QUERIES:

1. Create table Customer with the fields cust-no, cust_name, cust-add cust-ph.

```
create table Customer cust-no varchar (4),
cust_name varchar (25), cust_add varchar(30),
cust-ph varchar (15));
```

2. Create the structure for special customer from the structure of Customer relation.

```
create table special-customer (cust-no, cust-name, cust-add)
as select cust-no, cust-name, cust_add from Customer.
```

3. Add customer fax number in the customer relation.

```
ALTER TABLE Customer ADD (fax-no varchar (15));
```

4. Modify the datatype of fax-no to numeric datatype.

```
ALTER TABLE Customer MODIFY (fax-no number (10));
```

5. Write the command to deleting special-customer relation.

```
DROP TABLE special-customer;
```

6. Rename the customer relation to customer-info.

```
RENAME customer TO customer-info;
```

7. Delete all the records in the customer-info relation.

```
TRUNCATE table customer-info;
```

SQL DML Queries:

SQL DML commands allows the user to manipulate the data in database.

To insert values in a table:

INSERT into customer (cust no, cust_name, cust_add, cust. ph)
VALUES ('S01', 'Jerry', 'Chennai', 9841268763);

Consider the following BOOK Relation.

ISBN	TITLE	PUB_YEAR	UNIT_PRICE	AUTHOR_NAME	PUBLISHER_NAME
1001	Oracle	2004	899	Arora	PHI
1002	DBMS	2004	400	Basu	Technical
2001	DOS	2002	250	Sinha	Nirali
2002	ADBMS	2004	450	Basu	Technical
2003	Unix	2000	800	Kapoor	Sci Tech

BASIC STRUCTURE:

The basic structure of an SQL expression consists of three clauses: select, from and where.

The select clause corresponds to projection operation of the relational algebra. It is used to list the attributes desired in the result of a query.

The from clause corresponds to the cartesian product of the relational algebra. It lists the relations to be scanned in the evaluation of the expression.

The where clause corresponds to the selection predicate of the relational algebra. It consists of a predicate involving attributes of the relations that appear in the from clause.

The Select Clause:

This command is used to display all fields [illegible] selected [illegible] for all / selected records from a relation.

Q1: Find the names of all publishers in the book relation

select publisher_name from book;

Q2: Find the names of publishers by eliminating the duplicates

select distinct publisher_name from book;

Q3: Display all the fields from book table.

select * from book;

Q4: Find the titles of books published in year 2004.

select title from book where pub-year = '2004';

Q5: Find the titles of book having price between 300 to 400.

select Title from book where unit_price between 300 and 4

select Title from book
where unit-price >= 300 and unit-price <= 400;

The Rename Operation:

Q6: Rename the column unit-price as new-price.

select Title, unit-price * 10 as new-price.

Title	new-price
Oracle	3990
DBMS	4500
DOS	2500
ADBMS	4500
UNIX	3000

String Operations:

SQL specifies strings by enclosing them in single quotes. The commonly used operation on strings is pattern matching using the operator like. The patterns are described by using two special character:

a) Percentage (%) : The % character matches any substring.

b) Underscore (_) : The _ character matches any character.

Patterns are case sensitive. Upper case characters do not match lower case character or vice versa.

Examples:

- 'computer%' - matches any string beginning with 'computer'.
- '%Engg' - matches any string containing "Engg" as a substring, for eg: "computer Engg department".
- '_s%' - matches any string with second character 's'.
- '___' - matches any string of exactly three characters.
- '___%' - matches any string of atleast three characters.

Q1: Find the names of author from book table where the first two characters of name are 'Ba'.

select author_name from book where author_name like 'Ba%'

Q2: Select the author name where the second character of name is 'r' or 'a'.

select author_name from book ~~from~~ where author_name like '_r%' or author_name like '_a%';

rdering the Display of Tuples:

SQL uses order by clause to display the tuples in the result of the query to appear in sorted order.

Q1: Display all titles of books with price in ascending order of titles.

```
select title, unit-price from book order by title;
```

Q2: Display all titles of books with price and year in descending order of year.

```
Select title, unit-price, pub-year from book
order by pub-year desc;
```

Aggregate Functions:

Aggregate functions are functions that take a collection of values as input and return a single value. SQL offers five built-in aggregate functions.

a. avg
b. min
c. max
d. sum
e. count

1. avg: Returns average value of n, ignoring null values.

Q1: Display the average price from book.

```
select avg(unit-price) "Average Price" from Book;
```

2. min: Returns minimum value of expression.

Q2: Display the minimum price from book.

```
Select min(unit-price) "Minimum Price" from Book;
```

3. max : Returns maximum value of expression.

Q3: Display the maximum price from book.

```
Select max(unit-price) "Maximum Price" from Book;
```

4. sum : Return sum of values of n.

Q4: Display the total price from Book.

```
select sum(unit-price) "Total price" from Book;
```

5. count : Returns the number of rows where expression is not null.

Q5: Display the total number of books available in the book table.

```
select count(title) "No. of Books" from Book;
```

SET Operations :

The SQL operations union, intersect and except operate on relations and correspond to the relational-algebra operations ∪, ∩ and −.

Consider the following tables:

Depositor (customer_name, acct_no)

Borrower (customer_name, loan_no)

Depositor

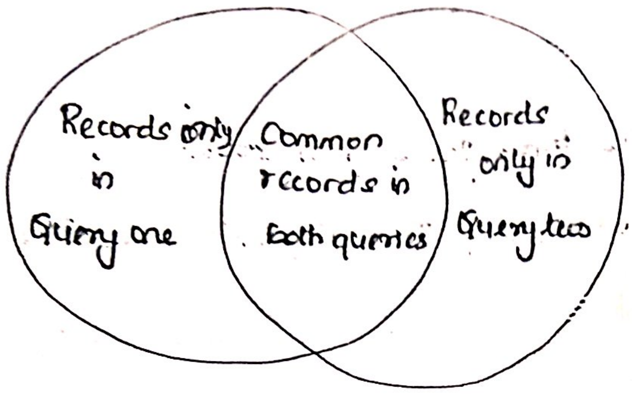
customer_name	acct_no
John	1001
Sita	1002
Vishal	1003
Ram	1004

Borrower.

customer_name	loan_no
John	2001
~~Sita~~ Tonny	2003
~~Vishal~~ Rohit	2004
~~Ram~~ Vishal	2002

a. Union Operation :

Union operation merges the output of two or more queries into a single set of rows and column.


Records only in Query one
Common records in Both queries
Records only in Query two

- Find all the customers having a loan, account or both at the bank.

  Select customer_name from borrower union select customer_name from depositor;

- The union operation automatically eliminates the duplicates.

customer_name
John
Ram
Rohit
Sita
Tonny
Vishal.

To retain all duplicates, then the query will be as follows:

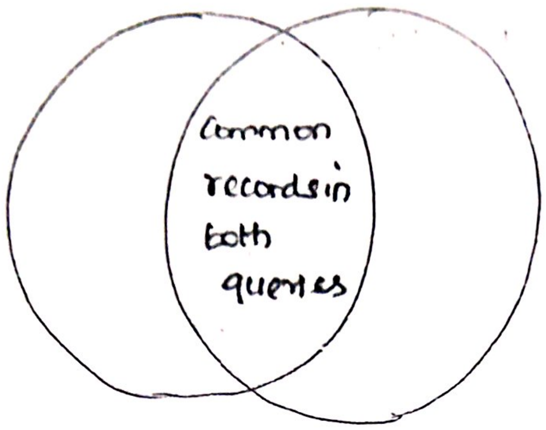
Select customer_name from borrower

union all

Select customer_name from depositor;

b. The Intersect Operation :

The intersect clause outputs only rows produced by both the queries intersected ie, the intersect query [illegible] as common rows from the output of both queries.



Find all the customers who have an account and loan at the bank.

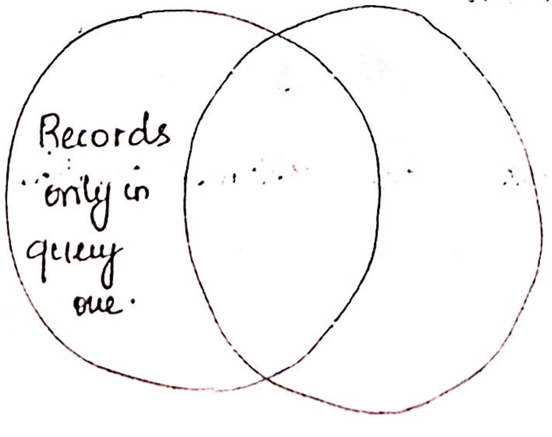
```
Select customer_name from Depositor
intersect
Select customer_name from Borrower;
```

customer_name
John
Vishal

The intersect operation automatically eliminates duplicates.

c. The except operation :

The Except also called as Minus operation outputs rows that are in first table not in second table.



Find all the customers who have an account but no loan at the bank.

```
select customer_name from depositor
minus
select customer_name from borrower;
```

NULL Values:

SQL allows the use of null values to indicate absence of information about the value of an attribute.

The special keyword null can be used in a predicate.

Consider the following table.

cust_no	cust_name	cust_phone
C101	John	22683491
C102	Peter	22384156
C103	Jerry	

Find all the customers from customer relation with null values for cust_phone.

```
select cust_name from customer
where
cust_phone is null;
```

Output:

cust_name
Jerry.

Find all the customers from customer relation where phone is not null.

```
select cust_name from customer
where
cust_phone is not null;
```

cust_name
John
Peter

Group by:

Group by clause is used to group the rows based on certain criteria. Group by is usually used in conjunction with aggregate functions like sum, avg, min, max etc.

Q1: Display total price of all books (publisherwise).

```
select publisher_name, sum(unit_price) "Total Amount"
from Book
group by publisher_name;
```

publisher_name	Total Amount
PHI	399
Technical	850
Nirali	250
SciTech	800

Having:

The having clause tells SQL to include only certain groups produced by the group by clause in the query result set. Having clause is equivalent to the where clause and is used to specify the search criteria or search condition when group by clause is specified.

Q1: Display publisherwise total price of books published, except for publisher 'PHI'.

```
select publisher_name, sum(unit_price) 'Total Amount"
from Book
group by publisher_name
having publisher_name <> 'PHI';
```

publisher-name	Total Amount
Nirali	250
Technical	850
Sci Tech	300.

NESTED SUBQUERIES:

SQL provides a mechanism for nesting subqueries. A sub query is a select from where expression that is nested within another query. Mostly sub queries are used to perform tests for set membership to make set comparison and determine set cardinality.

→ Set Membership:

SQL uses in and not in constructs for set membership tests

a) IN :

The in ~~construct~~ connective tests for set membership, where the set is a collection of values produced by a select clause.

Display the title, author and publisher name of all books published in 2000, 2002 and 2004.

```
select title, author_name, publisher_name, pub-year
from book
where pub-year in ('2000', '2002', '2004');
```

b. NOT IN:

The not in connective tests for the absence of set membership.

Display title, author and publisher name of all the books except those which are published in year 2004.

```
select title, author_name, publisher_name, pub-year
from book
where pub-year not in ('2004');
```

→ Tests for Empty Relations:

Exists is a test for non empty set. It is represented expression of the form 'Exists' (Select ... from ...).

Consider the following relations

a. book-info : {Book_ID, Title, author_name, publisher_name, pub_year}.

b. Order_info : {Orderno, Book_ID, order, date, Qty, price}.

bookid	Title	author_name	publisher_name	pub_year
1001	Oracle	Arora	PHI	2004
1002	DBMS	Basu	Technical	2004
2001	DOS	Sinha	Nirali	2003
2002	ADBMS	Basu	Technical	2004
2003	Unix	Kapoor	SciTech	2000

order_no	bookid	date	Qty	price
1	1001	10.10.2004	100	399
2	1002	11.01.2004	60	400.

Get the names of all the books for which order is placed.

```
Select Title from Book_Info
    where exists (select * from order_info
        where Book_info.book_id = Order_Info.book_id);
```

Title
Oracle
DBMS

Selecting data from a view:

Display all the titles of books written by author 'Basu'.

Select Title from v_book where author_name = 'Basu';

Title
DBMS
ADBMS

Updatable Views:

Views can also be used for data manipulation i.e, the user can perform Insert, Update and the Delete operations on the view. The views on which data manipulation can be done are called Updatable views, views that do not allow data manipulation are called Readonly Views.

For the view to be updatable, it should meet following criteria.

- The view must be created on a single table.
- The primary key column of the table should be included in the view.
- Aggregate functions cannot be used in the select statement.
- The select statement used for creating a view should not include Distinct, Group by or Having clause.
- The select statement used for creating a view should not include sub queries.

Destroying a View:

A view can be dropped by using DROP View Command.

DROP VIEW viewname;

DROP VIEW v_book;

JOIN :

Join is a query in which data is retreived from two or more table. A join matches data from two or more tables, based on the values of one or more columns in each table.

Different types of Join operations are

a. Inner Join

b. Outer Join

c. Natural Join.

Inner Join Operation :

Inner Join returns the matching rows from the tables that are being joined. Consider the following two relations :

Employee (emp-name, city)

emp-salary (emp-name, department, salary).

Employee

emp-name	city
Hari	Pune
Om	Mumbai
Smith	Nashik
Jay	Solapur

Emp-salary

emp-name	department	salary
Hari	Computer	10000
Om	IT	7000
Bill	Computer	8000
Jay	IT	5000.

ADVANCED SQL FEATURES:

EMBEDDED SQL

Embedded SQLs are statements included in the programming language. The programming language in which the SQL statements are included is called the host language.

Some of the host languages are C, COBOL, Pascal, FORTRAN. This embedded SQL source code is submitted to an SQL precompiler, which processes the SQL statements. Variables of the host language can be referenced in the embedded SQL statements thus allowing the values calculated by the program to be used by the SQL statements.

Embedded SQL Features:

- The embedded SQL statements appear in the host language SQL statements can be written in upper case or lower case.
- Embedded SQL statements are prefixed by a delimeter - EXEC SQL so that they can be distinguished from host language statements.
- If an embedded SQL statements extend over multiple lines, the host language strategy for statement continuation is used.
- Every embedded SQL statement is terminated with a delimeter In COBOL it is END EXEC.
- Host variables and SQL columns can have same name.

Example:

```
int main ()
EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION
    int orderID
    int custID
    char Salesperson [10].
    char status[6]
EXEC SQL END DECLARE SECTION.
```

```
/* Error processing */
EXEC SQL WHENEVER SQLERROR GOTO query-error;
EXEC SQL WHENEVER [illegible] GOTO bad-number

printf(" Enter order number")
scanf-s ("%d", & orderID)

EXEC SQL SELECT CustID, SalesPerson, Status
 FROM Orders
 WHERE OrderID = :OrderID
 INTO :CustID, :SalesPerson, :Status.

printf ("Customer number : %d\n", CustID);
printf (" Salesperson %s \n", SalesPerson);
printf (" Status : %s\n", status).
exit();

query-error :
    printf ( "SQL error : %ld\n", sqlca -> sqlcode);
    exit ();

bad-number :

    printf (" Invalid order number:\n");
    exit();

}
```

Advantages of Embedded SQL.

- Efficient way of merging the two strengths of programming environments.
- The program's run-time interface to the private database routines is transparent to the application programmer.

DYNAMIC SQL.

The Dynamic SQL component of SQL allows programs to construct and submit SQL queries at run time. Using Dynamic SQL programs can create SQL queries as strings at runtime and either have them executed immediately or have them prepared for subsequent use.

SQL defines standards for embedded dynamic SQL
lls in a host language, such as C in the following example.

```
char * sqlprog = "update account set balance =
                        balance * 1.05
                  where acc_no = ?"

EXEC C  SQL  prepare dynprog from : sqlprog;
 char acc[10] = "101";
 EXEC SQL execute dynprog using :acc;
```