UNIT-3 Database and SQL

Data :- Raw facts and figures which are useful to an organization. We cannot take decisions on the basis of data.

Information:- Well processed data is called information. We can take decisions on the basis of information.

Field: Set of characters that represents specific data element.

Record: Collection of fields is called a record. A record can have fields of different data types.

File: Collection of similar types of records is called a file.

Table: Collection of rows and columns that contains useful data/information is called a table. A table generally refers to the passive entity which is kept in secondary storage device.

Relation: Relation (collection of rows and columns) generally refers to an active entity on which we can perform various operations.

Database: Collection of logically related data along with its description is termed as database.

Tuple: A row in a relation is called a tuple.

Attribute: A column in a relation is called an attribute. It is also termed as field or data item.

Degree: Number of attributes in a relation is called degree of a relation.

Cardinality: Number of tuples in a relation is called cardinality of a relation.

Primary Key: Primary key is a key that can uniquely identifies the records/tuples in a relation. This key can never be duplicated and NULL.

Foreign Key: Foreign Key is a key that is defined as a primary key in some other relation. This key is used to enforce referential integrity in RDBMS.

Candidate Key: Set of all attributes which can serve as a primary key in a relation.

Alternate Key: All the candidate keys other than the primary keys of a relation are alternate keys for a relation.

DBA: Data Base Administrator is a person (manager) that is responsible for defining the data base schema, setting security features in database, ensuring proper functioning of the data bases etc.

Relational Algebra: Relation algebra is a set operation such as select, project, union, & Cartesian product etc.
Select operation: Yields set of set of rows depending upon certain condition. Mathematically it is denoted as
e.g $\sigma_{\text{rollno} > 10}$ (student) -means show those rows of student table whose roll no.'s are >10.

Project Operation : yields set of columns as result which are specified. Mathematically it is denoted as π
e.g. $\sigma_{\text{rollno,name}}$ (student) - means show only rollno and name column only.

Union Operation : Two relation are said to be union compatible if their degree and column are same.

e.g Relation A

X	Y	Z
$X1$	$Y1$	$Z1$
$x2$	$y2$	$z2$

Relation B

X	Y	Z
$X1$	$Y1$	$Z1$
$x3$	$y3$	$z3$

Resultant Relation $A \cup B =$

X	Y	Z
$X1$	$Y1$	$Z1$
$x3$	$y3$	$z3$

Cartesian product: Cartesian Product of two relation A and B gives resultant relation whose no. of column are sum of degrees of two relation and no. of rows are product of cardinality of two relations.

e.g Relation A

X	Y	Z
$x1$	$y1$	$z1$
$x2$	$y2$	$z2$

Relation B

U	V
$u1$	$v1$
$u2$	$v2$

Resultant Relation A\[× \]B =

X	Y	Z	U	V
x1	y1	z1	u1	v1
x1	y1	Z1	u2	v2
x2	y2	z2	u1	v1
x2	y2	z2	u1	v2

Structured Query Language

SQL is a non procedural language that is used to create, manipulate and process the databases(relations).

Characteristics of SQL

1. It is very easy to learn and use.
2. Large volume of databases can be handled quite easily.
3. It is non procedural language. It means that we do not need to specify the procedures to accomplish a task but just to give a command to perform the activity.
4. SQL can be linked to most of other high level languages that makes it first choice for the database programmers.

Processing Capabilities of SQL

The following are the processing capabilities of SQL

1. **Data Definition Language (DDL)**
   DDL contains commands that are used to create the tables, databases, indexes, views, sequences and synonyms etc.
   e.g:Create table, create view, create index, alter table etc.
2. **Data Manipulation Language (DML)**
   DML contains command that can be used to manipulate the data base objects and to query the databases for information retrieval.
   e.g Select, Insert, Delete, Update etc.
3. **Data Control Language:**

This language is used for controlling the access to the data. Various commands like GRANT, REVOKE etc are available in DCL.

4. **Transaction Control Language (TCL)**
   TCL include commands to control the transactions in a data base system. The commonly used commands in TCL are COMMIT, ROLLBACK etc.

Data types of SQL

Just like any other programming language, the facility of defining data of various types is available in SQL also. Following are the most common data types of SQL.

1. NUMBER
2. CHAR
3. VARCHAR / VARCHAR2
4. DATE
5. LONG
6. RAW/LONG RAW

1. **NUMBER**
   Used to store a numeric value in a field/column. It may be decimal, integer or a real value. General syntax is Number(n,d)
   Where n specifies the number of digits and d specifies the number of digits to the right of the decimal point.
   e.g marks number(3) declares marks to be of type number with maximum value 999.
   pct number(5,2) declares pct to be of type number of 5 digits with two digits to the right of decimal point.
2. **CHAR**
   Used to store character type data in a column. General syntax is Char (size)
   where size represents the maximum number of characters in a column. The CHAR type data can hold at most 255 characters.
   e.g name char(25) declares a data item name of type character of upto 25 size long.
3. **VARCHAR/VARCHAR2**
   This data type is used to store variable length alphanumeric data. General syntax is varchar(size) / varchar2(size)
   where size represents the maximum number of characters in a column. The maximum

allowed size in this data type is 2000 characters.
e.g address varchar(50); address is of type varchar of upto 50 characters long.

4. **DATE**
   Date data type is used to store dates in columns. SQL supports the various date formats other that the standard DD-MON-YY.
   e.g dob date; declares dob to be of type date.
5. **LONG**
   This data type is used to store variable length strings of upto 2 GB size.
   e.g description long;
6. **RAW/LONG RAW**
   To store binary data (images/pictures/animation/clips etc.) RAW or LONG RAW data type is used. A column LONG RAW type can hold upto 2 GB of binary data.
   e.g image raw(2000);

SQL Commands

CREATE TABLE Command:
Create table command is used to create a table in SQL. It is a DDL type of command.

Syntax :
CREATE TABLE <table name>
(<column name> <data types>[(size)] [,<column name> <data types>[(size)]....);
e.g.
Create table student
(rollno number(2), name
varchar2(20), dob date);

Constraints:

Constraints are the conditions that can be enforced on the attributes of a relation. The constraints come in play when ever we try to insert, delete or update a record in a relation. They are used to ensure integrity of a relation, hence named as integrity constraints.

1. NOT NULL
2. UNIQUE
3. PRIMARY KEY

4. FOREIGN KEY
5. CHECK
6. DEFAULT

i. **Not Null constraint :** It ensures that the column cannot contain a NULL value.
ii. **Unique constraint :** A candidate key is a combination of one or more columns, the value of which uniquely identifies each row of a table.
iii. **Primary Key :** It ensures two things : (i) Unique identification of each row in the table. (ii) No column that is part of the Primary Key constraint can contain a NULL value.
iv. **Foreign Key :** The foreign key designates a column or combination of columns as a foreign key and establishes its relationship with a primary key in different table.
Create table Fee
(RollNo number(2) Foreign key (Rollno) references Student (Rollno),
Name varchar2(20) Not null, Amount
number(4), Fee_Date date);
v. **Check Constraint :** Sometimes we may require that values in some of the columns of our table are to be within a certain range or they must satisfy certain conditions.
Example:
Create table Employee
(EmpNo number(4) Primary Key,
Name varchar2(20) Not Null,
Salary number(6,2) check (salary > 0),
DeptNo number(3)
);

Data Manipulation in SQL

DML Commands are as under:
SELECT - Used for making queries
INSERT - Used for adding new row or record into table
UPDATE- used for modification in existing data in a table
DELETE – used for deletion of records.

INSERT Statement
To insert a new tuple into a table is to use the insert statement

insert into <table> [(<column i, . . . , column j>)] values (<value i, . . . , value j>);

INSERT INTO student VALUES(101,'Rohan','XI',400,'Jammu');
While inserting the record it should be checked that the values passed are of same data types as the one which is specified for that particular column.
For inserting a row interactively (from keyboard) & operator can be used.
e.g INSERT INTO student VALUES(&Roll_no','&Name','&Class','&Marks','&City');

In the above command the values for all the columns are read from keyboard and inserted into the table student.
NOTE:- In SQL we can repeat or re-execute the last command typed at SQL prompt by typing "/" key and pressing enter.

Roll_no	Name	Class	Marks	City
101	Rohan	XI	400	Jammu
102	Aneeta Chopra	XII	390	Udhampur
103	Pawan Kumar	IX	298	Amritsar
104	Rohan	IX	376	Jammu
105	Sanjay	VII	240	Gurdaspur
113	Anju Mahajan	VIII	432	Pathankot

Operators in SQL:

The following are the commonly used operators in SQL

1. Arithmetic Operators +, -, *, /
2. Relational Operators =, <, >, <=, >=, <>
3. Logical Operators OR, AND, NOT
   Arithmetic operators are used to perform simple arithmetic operations.
   Relational Operators are used when two values are to be compared and Logical operators are used to connect search conditions in the WHERE Clause in SQL.
   Other operators :

4. Range check – between low and high
5. List check – in
6. Pattern check – like , not like (% and _ ‘under score’ is used)

Queries: To retrieve information from a database we can query the databases. SQL SELECT statement is used to select rows and columns from a database/relation.

SELECT Command
This command can perform selection as well as projection.
Selection: This capability of SQL can return you the tuples form a relation with all the attributes.
e.g. SELECT name, class FROM student;
The above command displays only name and class attributes from student table.

Projection: This is the capability of SQL to return only specific attributes in the relation. Use of where clause is required when specific tuples are to be fetched or manipulated.
e.g SELECT * FROM student; command will display all the tuples in the relation student
SELECT * FROM student WHERE Roll_no <=102;
The above command display only those records whose Roll_no less than or equal to 102.
Select command can also display specific attributes from a relation.
* SELECT count(*) AS “Total Number of Records” FROM student;
Display the total number of records with title as “Total Number of Records” i.e an alias
We can also use arithmetic operators in select statement, like
* SELECT Roll_no, name, marks+20 FROM student;
* SELECT name, (marks/500)*100 FROM student WHERE Roll_no > 103;

Eliminating Duplicate/Redundant data
DISTINCT keyword is used to restrict the duplicate rows from the results of a SELECT statement.
e.g. SELECT DISTINCT name FROM student;
The above command returns

Name
Rohan
Aneeta Chopra

Pawan Kumar

Conditions based on a range

SQL provides a BETWEEN operator that defines a range of values that the column value must fall for the condition to become true.

e.g. SELECT Roll_no, name FROM student WHERE Roll_no BETWENN 100 AND 103;

The above command displays Roll_no and name of those students whose Roll_no lies in the range 100 to 103 (both 100 and 103 are included in the range).

Conditions based on a list

To specify a list of values, IN operator is used. This operator select values that match any value in the given list.

e.g. SELECT * FROM student WHERE city IN ('Jammu','Amritsar','Gurdaspur');

The above command displays all those records whose city is either Jammu or Amritsar or Gurdaspur.

Conditions based on Pattern

SQL provides two wild card characters that are used while comparing the strings with LIKE operator.

a. percent(%) Matches any string

b. Underscore(_) Matches any one character

e.g SELECT Roll_no, name, city FROM student WHERE Roll_no LIKE "%3";

displays those records where last digit of Roll_no is 3 and may have any number of characters in front.

e.g SELECT Roll_no, name, city FROM student WHERE Roll_no LIKE "1_3";

displays those records whose Roll_no starts with 1 and second letter may be any letter but ends with digit 3.

ORDER BY Clause

ORDER BY clause is used to display the result of a query in a specific order(sorted order). The sorting can be done in ascending or in descending order. It should be kept in mind that the actual data in the database is not sorted but only the results of the query are displayed in sorted order.

e.g. SELECT name, city FROM student ORDER BY name;
The above query returns name and city columns of table student sorted by name in increasing/ascending order.
e.g. SELECT * FROM student ORDER BY city DESC;
It displays all the records of table student ordered by city in descending order.

Note:- If order is not specifies that by default the sorting will be performed in ascending order.

SQL Functions:

SQL supports functions which can be used to compute and select numeric, character and date columns of a relations. These functions can be applied on a group of rows. The rows are grouped on a common value of a column in the table. These functions return only one value for a group and therefore, they are called aggregate or group functions.

1. **SUM() :** It returns the sum of values of numeric type of a column.
   Eg. Select sum(salary) from employee;
2. **AVG() :** It returns the average of values of numeric type of a column.
   Eg. Select avg(salary) from employee;
3. **MIN() :** It returns the minimum of the values of a column of a given relation.
   Eg. Select min(salary) from employee;
4. **MAX() :** It returns the maximum of the values of a column of a given relation.
   Eg. Select max(salary) from employee;
5. **COUNT() :** It returns the number of rows in a relation.
   Eg. Select count(*) from employee;

Group By Clause :

The rows of a table can be grouped together based on a common value by using the Group By clause of SQL in a select statement.

Syntax : SELECT <attribute name>, <attribute name> ---- [functions]
FROM <relation name>
GROUP BY <group by column>;
Eg. Select age, count (rollno)

From students Group by age;
Output:

Age	Count (rollno)
15	2
14.5	2
14	5

Group-By-Having Clause :
It is used to apply some condition to the Group By clause.
Eg.
Select class
From students Group by class
Having count(*) > 5;

DELETE Command
To delete the record fro a table SQL provides a delete statement. General syntax is:-
DELETE FROM <table_name> [WHERE <condition>];
e.g. DELETE FROM student WHERE city = 'Jammu';
This command deletes all those records whose city is Jammu.

NOTE: It should be kept in mind that while comparing with the string type values lowercase and uppercase letters are treated as different. That is 'Jammu' and 'jammu' is different while comparing.

UPDATE Command
To update the data stored in the data base, UPDATE command is used.
e. g. UPDATE student SET marks = marks + 100;
Increase marks of all the students by 100.
e. g. UPDATE student SET City = 'Udhampur' WHERE city = 'Jammu';
changes the city of those students to Udhampur whose city is Jammu.
We can also update multiple columns with update command, like
e. g. UPDATE student set marks = marks + 20, city = 'Jalandhar'

WHERE city NOT IN ('Jammu','Udhampur');

CREATE VIEW Command

In SQL we can create a view of the already existing table that contains specific attributes of the table.
e. g. the table student that we created contains following fields:
Student (Roll_no, Name, Marks, Class, City)
Suppose we need to create a view v_student that contains Roll_no,name and class of student table, then Create View command can be used:
CREATE VIEW v_student AS SELECT Roll_no, Name, Class FROM student;
The above command create a virtual table (view) named v_student that has three attributes as mentioned and all the rows under those attributes as in student table.
We can also create a view from an existing table based on some specific conditions, like
CREATE VIEW v_student AS SELECT Roll_no, Name, Class FROM student WHERE City <>'Jammu';
The main difference between a Table and view is that
A Table is a repository of data. The table resides physically in the database.
A View is not a part of the database's physical representation. It is created on a table or another view. It is precompiled, so that data retrieval behaves faster, and also provides a secure accessibility mechanism.

ALTER TABLE Command

In SQL if we ever need to change the structure of the database then ALTER TABLE command is used. By using this command we can add a column in the existing table, delete a column from a table or modify columns in a table.
Adding a column : The syntax to add a column is:-

ALTER TABLE table_name

ADD column_name datatype;
e.g ALTER TABLE student ADD(Address varchar(30));
The above command add a column Address to the table atudent.
If we give command
SELECT * FROM student;

The following data gets displayed on screen:

Roll_no	Name	Class	Marks	City	Address
101	Rohan	XI	400	Jammu	
102	Aneeta Chopra	XII	390	Udhampur	
103	Pawan Kumar	IX	298	Amritsar	
104	Rohan	IX	376	Jammu	
105	Sanjay	VII	240	Gurdaspur	
113	Anju MAhajan	VIII	432	Pathankot	

Note that we have just added a column and there will be no data under this attribute. UPDATE command can be used to supply values / data to this column.

Removing a column
ALTER TABLE table_name
DROP COLUMN column_name;
e.g ALTER TABLE Student
DROP COLUMN Address;
The column Address will be removed from the table student.

DROP TABLE Command
Sometimes you may need to drop a table which is not in use. DROP TABLE command is used to Delete / drop a table permanently. It should be kept in mind that we can not drop a table if it contains records. That is first all the rows of the table have to be deleted and only then the table can be dropped. The general syntax of this command is:-
DROP TABLE <table_name>;
e.g DROP TABLE student;
This command will remove the table student

Databases Management Systems and SQL(Database Concepts and SQL)

Relations: In relational data model, the data is organised into tables(in the form of rows and columns) called relations.

Domain: A domain is the sets of values from which the actual values appearing in a given column are drawn.

Tuple: A row in a relation.

Attribute: A column in a relation.

Primary Key: A column or set of columns that uniquely identifies a row within a table is called primary key.

Candidate Key: Candidate keys are set of fields (columns with unique values) in the relation that are eligible to act as a primary key.

Alternate Key: A candidate key that is not the primary key is called alternate key.

DDL: Data Definition Language (DDL) or Data Description Language (DDL).

DML: Data Manipulation Language (DML).

String datatypes: CHAR, VARCHAR, VARCHAR2

Numeric datatype: NUMBER, NUMERIC, INT, FLOAT, DECIMAL

Date: DATE

Selection: Selection in relational algebra returns those tuples (records) in a relation that fulfil a condition (Produce table containing subset of rows).

Projection: Projection in relational algebra returns those columns in a relation that given in the attribute list (Produce table containing subset of columns).

Union: The union operator is used to combine two or more tables. In the union operation, duplicate records will be automatically removed from the resultant table.

Cartesian product: SQL joins are used to relate information in different tables. Cartesian product returns a number of rows equal to number of rows in the first table multiply by number of rows in the second table. At the same time, number of columns equal to number of columns in the first table added by number of columns in the second table.

SQL (Structured Query Language)

CREATE TABLE: Used to create the structure of a table.
CREATE TABLE *table_name* (*column1 datatype*, *column2 datatype*, *column3 datatype*,....);

ALTER TABLE: Used to implement structure modification.

To add a new column after creating table:
ALTER TABLE *table_name* ADD *column_name datatype*;

To modify an existing column.
ALTER TABLE *table_name* MODIFY COLUMN *column_name datatype*;

DROP TABLE: To remove a table and remove all of its data
DROP TABLE *table_name* ;

INSERT INTO: is used to insert new records in a table.

INSERT INTO *table_name* (*column1*,*column2*,*column3*, ..) VALUES(*value1*, *value2*, *value3*, ..);

SELECT: The SELECT statement is used to select and display data from a database.
SELECT *column1*, *column2*, ... FROM *table_name*;
SELECT * FROM *table_name*;
SELECT *column1*, *column2*, ... FROM *table_name* WHERE *condition*;

DISTINCT: Distinct keyword eliminates duplicate rows from the result of a select statement.
SELECT DISTINCT *column1*, *column2*, ... FROM *table_name*;

ORDER BY : Used to sort the result-set in ascending or descending order.
SELECT *column1*, *column2*, ... FROM *table_name* ORDER BY *column1* ... ASC|DESC;

GROUP BY : Used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns
SELECT *column_name(s)* FROM *table_name* WHERE *condition* GROUP BY *column_name(s)*

HAVING : Places condition on groups.

SELECT *column_name(s)* FROM *table_name* WHERE *condition* GROUP BY *column_name(s)* HAVI

MAX (): To select the maximum value of a particular column.

SELECT MAX(*column_name*) FROM *table_name* WHERE *condition*;

MIN (): To select the minimum value of a particular column.

SELECT MIN(*column_name*) FROM *table_name* WHERE *condition*;

SUM (): To find the total value of a particular column.

SELECT SUM(*column_name*) FROM *table_name* WHERE *condition*;

AVG (): To find the average value of a particular column.

SELECT AVG(*column_name*) FROM *table_name* WHERE *condition*;

COUNT (): Returns the number of records in the table.

SELECT COUNT(*column_name*) FROM *table_name* WHERE *condition*;

UPDATE : Used to modify the existing records in a table.

UPDATE *table_name* SET *column1* = *value1*, *column2* = *value2*, ...WHERE *condition*;

DELETE : Used to delete existing records in a table.

DELETE FROM *table_name* WHERE *condition*;