STRUCTURED QUERY LANGUAGE(SQL)

Basic Database concepts

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.

Basic Database concepts

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.

Basic Database concepts

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.

Basic Database concepts

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.

STRUCTURED QUERY LANGUAGE
(S Q L)

Structured Query Language

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

Characteristics of SQL

✓It is very easy to learn and use.

✓Large volume of databases can be handled quite easily.

✓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.

✓SQL can be linked to most of other high level languages that makes it first choice for the database programmers.

Structured Query Language

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**

Structured Query Language

Processing Capabilities of SQL

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.**

Structured Query Language

Processing Capabilities of SQL

3. View Definition:

DDL contains set of command to create a view of a relation.

e.g : **create view**

4. Data Control Language:

This language is used for controlling the access to the data. The commonly used commands DCL are,

GRANT, REVOKE

Structured Query Language

Processing Capabilities of SQL

5. 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

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**

Data types of SQL

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.

Data types of SQL

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.

Data types of SQL

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.

Data types of SQL

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;**

Data types of SQL

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. The general syntax of creating a table is

create table <table> (
<column 1> <data type> [not null] [unique] [<column constraint>],
. . . . . .
<column n> <data type> [not null] [unique] [<column constraint>],
[<table constraint(s)>]
);

SQL Commands

CREATE TABLE Command:

For each column, a name and a data type must be specified and the column name must be unique within the table definition. Column definitions are separated by comma. Uppercase and lowercase letters makes no difference in column names, the only place where upper and lower case letters matter are strings comparisons. A not null Constraint means that the column cannot have null value, that is a value needs to be supplied for that column. The keyword unique specifies that no two tuples can have the same attribute value for this column.

SQL Commands

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.

1. **NOT NULL**
2. **UNIQUE**
3. **PRIMARY KEY**
4. **FOREIGN KEY**
5. **CHECK**
6. **DEFAULT**

SQL Commands

Constraints:

1. NOT NULL: Ensures that we cannot leave a column as null. That is a value has to be supplied for that column.

e.g name varchar(25) not null;

2. UNIQUE: Constraint means that the values under that column are always unique.

e.g Roll_no number(3) unique;

SQL Commands

Constraints:

3. PRIMARY KEY: Constraint means that a column can not have duplicate values and not even a null value.

e.g. Roll_no number(3) primary key;

The main difference between unique and primary key constraint is that a column specified as unique may have null value but primary key constraint does not allow null values in the column.

SQL Commands

Constraints:

4. FOREIGN KEY: Is used to enforce referential integrity and is declared as a primary key in some other table.

e.g cust_id varchar(5) references master(cust_id);

it declares cust_id column as a foreign key that refers to cust_id field of table master. That means we cannot insert that value in cust_id filed whose corresponding value is not present in cust_id field of master table.

SQL Commands

Constraints:

5. CHECK: Constraint limits the values that can be inserted into a column of a table.

e.g marks number(3) check(marks>=0);

The above statement declares marks to be of type number and while inserting or updating the value in marks it is ensured that its value is always greater than or equal to zero.

SQL Commands

Constraints:

6. DEFAULT: Constraint is used to specify a default value to a column of a table automatically. This default value will be used when user does not enter any value for that column.
e.g balance number(5) default = 0;

```
CREATE TABLE student (
Roll_no      number(3) primary key,
Name         varchar(25) not null,
Class        varchar(10),
Marks        number(3) check(marks>0),
City         varchar(25)  );
```

SQL Commands

Operators in SQL:

The following are the commonly used operators in SQL

- ✓**Arithmetic Operators** **+, -, *, /**
- ✓**Relational Operators** **=, <, >, <=, >=, <>**
- ✓**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.

SQL Commands

Data Modifications in SQL

After a table has been created using the create table command, tuples can be inserted into the table, or tuples can be deleted or modified.

SQL Commands

Data Modifications in SQL

INSERT Statement

The simplest way to insert a 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,‘Chennai');

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.

SQL Commands

Data Modifications in SQL

INSERT Statement

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.

SQL Commands

TABLE : STUDENT

Roll_no	Name	Class	Marks	City
101	Rohan	XI	400	Chennai
102	Aneeta	XII	390	Bengaluru
103	Pawan Kumar	IX	298	Mysore
104	Rohan	IX	376	Mangalore
105	Sanjay	VII	240	Mumbai
113	Anju	VIII	432	Delhi

SQL Commands

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.

Projection: This is the capability of SQL to return only specific attributes in the relation.

SQL Commands

SELECT Command

✓ **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 name, class FROM student;**

The above command displays only name and class attributes from student table.

SQL Commands

SELECT Command

✓ **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;**

SQL Commands

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
Pawan Kumar

SQL Commands

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).

SQL Commands

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 ('Bengaluru','Delhi','Chennai');**

The above command displays all those records whose city is either **Bengaluru or Delhi or Chennai**

SQL Commands

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.

SQL Commands

Conditions based on Pattern

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.

SQL Commands

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.

SQL Commands

ORDER BY Clause

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 Commands

GROUP BY Clause

The GROUP BY clause can be used in a SELECT statement to collect data across multiple records and group the results by one or more columns.

The syntax for the GROUP BY clause is:

```
SELECT column1, column2, ... column_n,
aggregate_function (expression)
FROM tables
WHERE conditions
GROUP BY column1, column2, ... column_n;
```

SQL Commands

GROUP BY Clause

aggregate_function can be a function such as SUM, COUNT, MAX, MIN, AVG etc.

e.g SELECT name, COUNT(*) as "Number of

employees“ FROM student WHERE marks>350

GROUP BY city;

SQL Commands

HAVING Clause

The **HAVING** clause is used in combination with the GROUP BY clause. It can be used in a **SELECT** statement to filter the records that a **GROUP BY** returns.

The syntax for the **HAVING** clause is:

```
SELECT column1, column2, ... column_n,
aggregate_function (expression)
FROM tables
WHERE predicates
GROUP BY column1, column2, ... column_n
HAVING condition1 ... condition_n;
```

SQL Commands

HAVING Clause

e.g
```
SELECT SUM(marks) as "Total marks"
FROM student
GROUP BY department
HAVING SUM(sales) > 1000;
```

Note: select statement can contain only those attribute which are already present in the group by clause.

SQL Commands

Functions available in SQL

SQL provide large collection of inbuilt functions also called library functions that can be used directly in SQL statements.

1. **Mathematical functions**
2. **String functions**
3. **Date & Time functions**

SQL Commands

Functions available in SQL

1. Mathematical functions

Some of the commonly used mathematical functions are sum() avg(), count(), min(), max() etc.

e.g. **SELECT sum(marks) FROM student;**

displays the sum of all the marks in the table student.

e.g. **SELECT min(Roll_no), max(marks) FROM student;**

displays smallest Roll_no and highest marks in the table student.

SQL Commands

Functions available in SQL

2. String functions

These functions are used to deal with the string type values like

ASCII, LOWEWR, UPPER, LEN, LEFT, RIGHT, TRIM, LTRIM, RTRIM etc.

ASCII : Returns the ASCII code value of a character (leftmost character of string).

Syntax: **ASCII(character)**

SQL Commands

Functions available in SQL

2. String functions

SELECT ASCII('a')	returns	97
SELECT ASCII('A')	returns	65
SELECT ASCII('1')	returns	49
SELECT ASCII('ABC')	returns	65

For Upper character 'A' to 'Z' ASCII value 65 to 90
For Lower character 'A' to 'Z' ASCII value 97 to 122
For digit '0' to '9' ASCII value 48 to 57

SQL Commands

Functions available in SQL

2. String functions

For Upper character 'A' to 'Z' ASCII value 65 to 90

For Lower character 'A' to 'Z' ASCII value 97 to 122

For digit '0' to '9' ASCII value 48 to 57

NOTE: If no table name is specified then SQL uses Dual table which is a dummy table used for performing operations

SQL Commands

Functions available in SQL

2. String functions

LOWER : Convert character strings data into lowercase.

Syntax: **LOWER(string)**

SELECT LOWER('STRING FUNCTION')

returns **string function**

SQL Commands

Functions available in SQL

2. String functions

UPPER : Convert character strings data into Uppercase.

Syntax**: UPPER(string)**

SELECT UPPER('string function')

returns **STRING FUNCTION**

SQL Commands

Functions available in SQL

2. String functions

LEN : Returns the length of the character string.

Syntax: **LEN(string)**

SELECT LEN('STRING FUNCTION')

returns **15**

SQL Commands

Functions available in SQL

2. String functions

REPLACE : Replaces all occurrences of the second string(string2) in the first string(string1) with a third string(string3).

Syntax: **REPLACE('string1','string2','string3')**

SELECT REPLACE('STRING FUNCTION','STRING','SQL')

returns **SQL Function**

Returns NULL if any one of the arguments is NULL.

SQL Commands

Functions available in SQL

2. String functions

LEFT : Returns left part of a string with the specified number of characters counting from left.LEFT function is used to retrieve portions of the string.

Syntax: **LEFT(string,integer)**

SELECT LEFT('STRING FUNCTION', 6)

returns STRING

SQL Commands

Functions available in SQL

2. String functions

RIGHT : Returns right part of a string with the specified number of characters counting from right.RIGHT function is used to retrieve portions of the string.

Syntax: **RIGHT(string,integer)**

SELECT RIGHT('STRING FUNCTION', 8)

returns **FUNCTION**

SQL Commands

Functions available in SQL

2. String functions

LTRIM : Returns a string after removing leading blanks on Left side.(Remove left side space or blanks)

Syntax: **LTRIM(string)**

SELECT LTRIM(' STRING FUNCTION')

returns **STRING FUNCTION**

SQL Commands

Functions available in SQL

2. String functions

RTRIM : Returns a string after removing leading blanks on Right side.(Remove right side space or blanks)

Syntax**: RTRIM(string)**

SELECT RTRIM('STRING FUNCTION ')

returns **STRING FUNCTION**

SQL Commands

Functions available in SQL

2. String functions

REVERSE : Returns reverse of a input string.

Syntax: **REVERSE(string)**

SELECT REVERSE('STRING FUNCTION')

returns **NOITCNUF GNIRTS**

SQL Commands

Functions available in SQL

2. String functions

REPLICATE : Repeats a input string for a specified number of times.

Syntax: **REPLICATE (string, integer)**

SELECT REPLICATE('FUNCTION', 3)

returns **FUNCTIONFUNCTIONFUNCTION**

SQL Commands

Functions available in SQL

2. String functions

SPACE : Returns a string of repeated spaces. The SPACE function is an equivalent of using REPLICATE function to repeat spaces.

Syntax: **SPACE (integer)**

(If integer is negative, a null string is returned.)

SELECT ('STRING') + SPACE(1) + ('FUNCTION')

returns **STRING FUNCTION**

SQL Commands

Functions available in SQL

2. String functions

SUBSTRING : Returns part of a given string.

SUBSTRING function retrieves a portion of the given string starting at the specified character(startindex) to the number of characters specified(length).

Syntax: **SUBSTRING (string,startindex,length)**

SQL Commands

Functions available in SQL

2. String functions

SUBSTRING : Returns part of a given string.

SELECT SUBSTRING('STRING FUNCTION', 1, 6)

returns **STRING**

SELECT SUBSTRING('STRING FUNCTION', 8, 8)

returns FUNCTION

SQL Commands

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 = ‘Chennai’;

This command deletes all those records whose city is Chennai.

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.

SQL Commands

UPDATE Command

To update the data stored in the data base, UOPDATE 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 = ‘Hydrabad’**
WHERE city = ‘Bangaluru’;

changes the city of those students to Hydrabad whose city is Bangaluru.

SQL Commands

UPDATE Command

We can also update multiple columns with update command, like

e. g. **UPDATE student set marks = marks + 20, city = 'Mangalore'**
WHERE city NOT IN ('Delhi','Mysore');

SQL Commands

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:

SQL Commands

CREATE VIEW Command

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 <>'Delhi';

SQL Commands

CREATE VIEW Command

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.

SQL Commands

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;
```

SQL Commands

ALTER TABLE Command

e.g **ALTER TABLE student ADD(Address varchar(30));**

The above command add a column Address to the table student.

If we give command

SELECT * FROM student;

The following data gets displayed on screen:

SQL Commands

TABLE : STUDENT

Roll_no	Name	Class	Marks	City	Address
101	Rohan	XI	400	Chennai	
102	Aneeta	XII	390	Bengaluru	
103	Pawan Kumar	IX	298	Mysore	
104	Rohan	IX	376	Mangalore	
105	Sanjay	VII	240	Mumbai	
113	Anju	VIII	432	Delhi	

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.

SQL Commands

ALTER TABLE Command

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

SQL Commands

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

Relational Algebra

It is the collections of rules and operations on relations(tables). The various operations are selection, projection, Cartesian product, union, set difference and intersection, and joining of relations.

Relational Algebra

Define the terms:

i. Database Abstraction

Ans: Database system provides the users only that much information that is required by them, and hides certain details like, how the data is stored and maintained in database at hardware level. This concept/process is Database abstraction.

ii. Data inconsistency

Ans: When two or more entries about the same data do not agree i.e. when one of them stores the updated information and the other does not, it results in data inconsistency in the database.

Relational Algebra

Define the terms:

iii. Conceptual level of database implementation/abstraction

Ans: It describes what data are actually stored in the database. It also describes the relationships existing among data. At this level the database is described logically in terms of simple data-structures.

iv. Primary Key

Ans : It is a key/attribute or a set of attributes that can uniquely identify tuples within the relation.

Relational Algebra

Define the terms:

v. Candidate Key

Ans : All attributes combinations inside a relation that can serve as primary key are candidate key as they are candidates for being as a primary key or a part of it.

vi. Relational Algebra

Ans : It is the collections of rules and operations on relations(tables). The various operations are selection, projection, Cartesian product, union, set difference and intersection, and joining of relations.

Relational Algebra

Define the terms:

vii. Domain

Ans : it is the pool or collection of data from which the actual values appearing in a given column are drawn.