STRUCTURE QUERY LANGUAGE(SQL): SQL IS A DATABASE LANGUAGE WHICH WAS USED TO COMMUNICATE WITH DATABASE.INTRODUCED BY "IBM" AND INITIAL NAME OF THIS LANGUAGE WAS "SEQUEL" AND LATER RENAMED WITH "SQL".

"SQL" IS ALSO CALLED AS "CLI"(COMMON LANGUAGE INTERFACE) BECAUSE THIS IS THE LANGUAGE WHICH IS USED TO COMMUNICATE WITH ANY RDBMS PRODUCTS SUCH AS ORACLE, SQLSERVER, MYSQL, DB2.....etc.

SQL PRE-DEFINE QUERIES ARE NOT A CASE - SENSITIVE (WRITE QUERIES IN EITHER UPPER & LOWER-CASE CHARACTERS) BUT EVERY SQL QUERY SHOULD ENDS WITH "; ".

SUB - LANGUAGES OF SQL:

1) DDL (DATA DEFINITION LANGUAGE):

> CREATE, ALTER, RENAME, TRUNCATE, DROP

> RECYCLEBIN, FLASHBACK, PURGE (LATEST FEATURES)

2) DML (DATA MANIPULATION LANGUAGE):

> INSERT, UPDATE, DELETE

> INSERT ALL, MERGE (NEW COMMANDS)

3) DQL / DRL (DATA QUERY / DATA RETRIVE LANGUAGE):

> SELECT

4) TCL (TRANSACTION CONTROL LANGUAGE):

> COMMIT, ROLLBACK, SAVEPOINT

5) DCL (DATA CONTROLL LANGUAGE):

> GRANT, REVOKE

1) DDL (DATA DEFINITION LANGUAGE):

> CREATE

> ALTER

> RENAME

> TRUNCATE

> DROP

1. CREATE: CREATE A NEW TABLE IN ORACLE DB.

SYNTAX:

CREATE TABLE <TABLE NAME> (<COLUMN NAME1> <DATATYPE>[SIZE], <COLUMN NAME2> <DATATYPE>[SIZE], ...);

EX:

CREATE TABLE STUDENT (STID INT, SNAME CHAR (10), SFEE NUMBER (6,2));

TO VIEW THE STRUCTURE OF A TABLE IN ORACLE DB:

SYNTAX:

SQL> DESC <TABLE NAME>;

EX:

SQL> DESC STUDENT;

2. ALTER:

IT IS USED TO MODIFY THE STRUCTURE OF A TABLE IN DATABASE.THIS COMMAND IS HAVING THE FOLLOWING FOUR SUB COMMANDS ARE

i) ALTER - MODIFY: TO CHANGE DATATYPE AND ALSO SIZE OF DATATYPE OF A PARTICULAR COLUMN.

SYNTAX:

ALTER TABLE <TN> MODIFY <COLUMN NAME> <NEW DATATYPE>[NEW SIZE];

EX:

SQL> ALTER TABLE STUDENT MODIFY SNAME VARCHAR2(20);

ii) ALTER - ADD: ADDING A NEW COLUMN TO AN EXISTING TABLE.

SYNTAX:

ALTER TABLE <TN> ADD <NEW COLUMN NAME> <DATATYPE>[SIZE];

EX:

SQL> ALTER TABLE STUDENT ADD SADDRESS VARCHAR2(30);

iii) ALTER - RENAME: TO CHANGE A COLUMN NAME IN A TABLE.

SYNTAX:

ALTER TABLE <TN> RENAME <COLUMN> <OLD COLUMN NAME> TO <NEW COLUMN NAME>;

EX:

SQL> ALTER TABLE STUDENT RENAME COLUMN SNAME TO STUDENTNAMES;

iv) ALTER - DROP: TO DROP AN EXISTING COLUMN FROM A TABLE.

SYNTAX:

ALTER TABLE <TN> DROP <COLUMN> <COLUMN NAME>;

EX:

SQL> ALTER TABLE STUDENT DROP COLUMN SFEE;

3. RENAME: IT IS USED TO CHANGE A TABLE NAME IN DATABASE.

SYNTAX:

RENAME <OLD TABLE NAME> TO <NEW TABLE NAME>;

EX:

SQL> RENAME STUDENT TO STUDENTDETAILS;

4. TRUNCATE: TO DELETE ALL ROWS FROM A TABLE AT A TIME. BY USING TRUNCATE COMMAND, WE CANNOT DELETE A SPECIFIC ROW FROM A TABLE BECAUSE TRUNCATE DOES NOT SUPPORTS "WHERE CLAUSE" CONDITION. IS DELETING ROWS BUT NOT COLUMNS OF A TABLE.

SYNTAX:

TRUNCATE TABLE <TABLE NAME>;

EX:

SQL> TRUNCATE TABLE STUDENT;

5. DROP: TO DROP A TABLE (i.e., ROWS AND COLUMNS) FROM DATABASE.

SYNTAX:

DROP TABLE <TABLE NAME>;

EX:

SQL> DROP TABLE STUDENT;

NOTE: FROM ORACLE 10g ENTERPRISE EDITION ONCE WE DROP A TABLE FROM DATABASE THEN IT WILL DROP TEMPORARILY. AND USER HAS A CHANCE TO RESTORE DROPPED TABLE AGAIN INTO DATABASE BY USING THE FOLLOWING COMMANDS ARE,

1) RECYCLEBIN

2) FLASHBACK

3) PURGE

1) RECYCLEBIN: IT IS A PRE-DEFINE TABLE WHICH IS USED TO STORED INFORMATION ABOUT DROPPED TABLES.IT WILL WORK AS A WINDOWS RECYCLEBIN IN SYSTEM.

HOW TO VIEW THE STRUCTURE OF RECYCLEBIN:

SYNTAX:

SQL> DESC RECYCLEBIN;

Name **Null?** **Type**

---------- **--------** **------------**

OBJECT_NAME **NOT NULL** **VARCHAR2(30)**

ORIGINAL_NAME **VARCHAR2 (32)**

HOW TO VIEW INFORMATION ABOUT DROPPED TABLES IN RECYCLEBIN:

SYNTAX:

SQL> SELECT OBJECT_NAME, ORIGINAL_NAME FROM RECYCLEBIN;

OBJECT_NAME **ORIGINAL_NAME**

---------- **----------**

BIN$EenuRFtsT7ahnHV7rbI71Q==$0 **STUDENT**

2) FLASHBACK: THIS COMMAND IS USED TO RESTORE A DROPPED TABLE FROM RECYCLEBIN.

SYNTAX:

SQL> FLASHBACK TABLE <TABLE NAME> TO BEFORE DROP;

EX:

SQL> FLASHBACK TABLE STUDENT TO BEFORE DROP;

PURGE: THIS COMMAND IS USED TO DROP A TABLE FROM RECYCLEBIN PERMANENTLY (OR) TO DROP A TABLE FROM DATABASE PERMANENTLY.

SYNTAX1: (DROPPING A SPECIFIC TABLE FROM RECYCLEBIN)

SQL> PURGE TABLE <TABLE NAME>;

EX:

SQL> PURGE TABLE TEST1;

SYNTAX2: (DROPPING ALL TABLES FROM RECYCLEBIN)

SQL> PURGE RECYCLEBIN;

EX:

SQL> PURGE RECYCLEBIN;

SYNTAX3: (DROP A TABLE FROM DATABASE PERMANENTLY)

SQL> DROP TABLE <TABLE NAME> PURGE;

EX:

SQL> DROP TABLE STUDENT PURGE;

2) DML (DATA MANIPULATION LANGUAGE)

- **> INSERT**
- **> UPDATE**
- **> DELETE**

1. INSERT: INSERTING A NEW ROW DATA INTO A TABLE.

SYNTAX1:

INSERT INTO <TN> VALUES (VALUE1, VALUE2, ……);

EX:

SQL> CREATE TABLE STUDENT (STID INT, SNAME VARCHAR2(10), SFEE NUMBER (10));

SQL> INSERT INTO STUDENT VALUES (1021,'SAI',2500);

1 row created.

NOTE: IN THIS METHOD WE SHOULD INSERT VALUES TO ALL COLUMNS IN A TABLE.

SYNTAX2:

INSERT INTO <TN> (REQ. COLUMN NAMES) VALUES (VALUE1, VALUE2, …….);

EX:

SQL> INSERT INTO STUDENT (STID, SNAME) VALUES (1022,'SMITH');

1 row created.

NOTE: IN THIS METHOD WE CAN INSERT VALUES FOR REQUIRED COLUMNS ONLY.AND REMAINING COLUMNS WILL TAKE "NULL" BY DEFAULT.

How to insert NULLs into a table:

METHOD1:

INSERT INTO EMP VALUES (NULL, NULL, NULL);

METHOD2:

INSERT INTO EMP (EID, ENAME, EADDRESS) VALUES (NULL, NULL, NULL);

SUBSTITUTIONAL OPERATORS: THESE OPERATORS ARE USED TO INSERT MULTIPLE ROWS DATA INTO A TABLE CONTINUALLY.THESE ARE TWO TYPES,

i) &: WE CAN INSERT VALUES TO COLUMNS DYNAMICALLY.

ii) &&: WE CAN INSERT VALUES TO COLUMNS IN FIXED MANNER.IF WE WANT CHANGE A FIXED VALUE OF COLUMN THEN WE SHOULD "EXIT" FROM ORACLE DATABASE.

SYNTAX1 (&):

INSERT INTO <TN> VALUES (&<COLUMN NAME1>,&<COLUMN NAME2>,.....);

EX:

SQL> INSERT INTO STUDENT VALUES (&STID,'&SNAME',&SFEE);

Enter value for stid: 1023

Enter value for sname: ALLEN

Enter value for sfee: 1500

SQL> / ---------- →(HERE " / " IS USED TO RE-EXECUTE THE LAST EXECUTED SQL QUERY IN SQLPLUS EDITOR)

Enter value for stid: 1024

Enter value for sname: WARD

Enter value for sfee: 4500

SYNTAX2 (&):

INSERT INTO <TN> (REQ.COLUMN NAMES) VALUES (&<COLUMN NAME1>, ……);

EX:

SQL> INSERT INTO STUDENT (STID)VALUES(&STID);

Enter value for stid: 1026

SQL> /

Enter value for stid: 1027

SQL> /

Enter value for stid: 1028

SYNTAX1 (&&):

INSERT INTO <TN> VALUES (&&<COLUMN NAME1>,&&<COLUMN NAME2>,……);

EX:

SQL> INSERT INTO STUDENT VALUES (&STID,'&SNAME',&&SFEE);

Enter value for stid: 1029

Enter value for sname: SCOTT

Enter value for sfee: 8000

SQL> /

Enter value for stid: 1030

Enter value for sname: WARNER

SQL> /

……

……

SYNTAX2 (&&):

INSERT INTO <TN>(REQ.COLUMN NAMES)VALUES(&&<COLUMN NAME1>,.....);

UPDATE: UPDATING ALL ROWS DATA AT A TIME IN A TABLE (OR) UPDATING A SINGLE ROW DATA IN A TABLE BY USING "WHERE CLAUSE"CONDITION.

SYNTAX:

UPDATE <TN> SET <COLUMN NAME1> = <VALUE1>,<COLUMN NAME2>=<VALUE2>,.....[WHERE <CONDITION>];

EX1:

SQL> UPDATE STUDENT SET SNAME='JONES', SFEE=6500 WHERE STID=1027;

SQL> UPDATE EMP SET COMM=500;

SQL> UPDATE EMP SET SAL=NULL;

SQL> UPDATE EMP SET SAL=5000 WHERE SAL IS NULL;

DELETE: TO DELETE ALL ROWS FROM A TABLE AT A TIME (OR) TO DELETE A SPECIFIC ROW FROM A TABLE BY USING "WHERE CLAUSE" CONDITION.

SYNTAX:

DELETE FROM <TN> [WHERE <CONDITION>];

EX:

SQL> DELETE FROM STUDENT WHERE STID=1023;

EX:

SQL> DELETE FROM STUDENT;

SQL> DELETE FROM EMP WHERE COMM IS NULL;

DIFFERENCE BETWEEN DELETE & TRUNCATE COMMAND:

DELETE	TRUNCATE
1. IT IS A DML COMMAND.	1. IT IS A DDL COMMAND.
2. IT CAN DELETE A SPECIFIC ROW FROM A TABLE.	2. IT CANNOT DELETE A SPECIFIC ROW FROM A TABLE.
3. IT SUPPORTS "WHERE CLAUSE" CONDITION.	3. IT DOES NOT SUPPORTS "WHERE CLAUSE" CONDITION.
4. IT TEMPORARY DATA DELETION.	4. IT IS PERMANENT DATA DELETION.
5. WE CAN RESTORE DELETED DATA BY USING "ROLLBACK" COMMAND.	5. WE CANNOT RESTORE DELETED DATA BY USING "ROLLBACK".
6. EXECUTION SPEED IS SLOW. (DELETING ROWS IN ONE BY ONE MANNER)	6. EXECUTION SPEED IS FAST (DELETING GROUP OF ROWS AT A TIME)

3) DQL / DRL (DATA QUERY LANGUAGE / DATA RETRIVE LANGUAGE):

> SELECT

SELECT: TO RETRIEVE ALL ROWS FROM A TABLE AT A TIME (OR) TO RETRIEVE A SPECIFIC ROW FROM A TABLE BY USING "WHERE CLAUSE" CONDITION.

SYNTAX:

SELECT * FROM <TABLE NAME> [WHERE <CONDITION>];

Here, " * " IS REPRESENT ALL COLUMNS IN A TABLE.

EX:

SQL> SELECT * FROM DEPT;

(OR)

SQL> SELECT DEPTNO, DNAME, LOC FROM DEPT;

EX:

SQL> SELECT * FROM EMP WHERE JOB='CLERK';

SQL> SELECT * FROM EMP WHERE COMM IS NULL;

SQL> SELECT * FROM EMP WHERE COMM IS NOT NULL;

TO VIEW ALL LIST OF TABLES IN ORACLE DATABASE:

SYNTAX:

SQL> SELECT * FROM TAB;

TO VIEW DATA OF A PARTICULAR TABLE:

SYNTAX:

SQL> SELECT * FROM <TABLE NAME>;

EX:

SQL> SELECT * FROM EMP;

NOTE: WHEN WE WANT TO DISPLAY THE INFORMATION / DATA OF A PARTICULAR TABLE IN PROPER SYSTEMATICALLY THEN WE NEED TO SET THE FOLLOWING TWO PROPERTIES ARE,

1) PAGESIZE n:

- NUMBER OF ROWS DISPLAYED PER A PAGE.HERE "n" IS REPRESENTED NO. OF ROWS. BY DEFAULT, A SINGLE PAGE IS DISPLAYED 14 ROWS.RANGE IS 0 TO 50000.

SYNTAX:

SQL> SET PAGESIZE n;

EX:

SQL> SET PAGESIZE 100;

2) LINES n:

- NUMBER OF BYTES IN A SINGLE LINE.HERE "n" IS REPRESENT NO. OF BYTES.RANGE IS 1 TO 32767.

SYNTAX:

SQL> SET LINES n;

EX:

SQL> SET LINES 100;

TO CLEAR SQL PLUS EDITOR SCREEN:

SYNTAX:

SQL> CL SCR;

(OR)

SHIFT+DELETE (FROM KEYBOARD)

TO DISCONNECT / EXIT FROM ORACLE DATABASE:

SQL> EXIT;

ALIAS NAMES: IT IS AN ALTERNATE (OR) TEMPORARY NAME. USER CAN CREATE ALIAS NAMES ON TWO LEVELS IN DB.

i) COLUMN LEVEL:

- IN THIS LEVEL WE ARE CREATING ALIAS NAMES ON COLUMNS.

SYNTAX:

<COLUMN NAME> <COLUMN ALIAS NAME>

EX:

DEPTNO X

ii) TABLE LEVEL:

- IN THIS LEVEL WE CREATING ALIAS NAMES ON TABLE.

SYNTAX:

<TABLE NAME> <TABLE ALIAS NAME>

EX:

DEPT D

SYNTAX TO COMBINED COLUMN + TABLE LEVEL ALIAS NAMES BY USING "SELECT" QUERY:

SELECT <COLUMN NAME1> <COLUMN NAME1 ALIAS NAME>,<COLUMN NAME2> <COLUMN NAME2 ALIAS NAME>,…… FROM <TN> <TABLE ALIAS NAME>;

EX:

SQL> SELECT DEPTNO X, DNAME Y, LOC Z FROM DEPT D;

CONCATENATION OPERATOR (||):

- THIS OPERATOR IS USED TO JOIN TWO STRING VALUES (OR) TWO EXPRESSIONS IN A SELECT QUERY.

EX:

SQL> SELECT 'WELCOME'||' '||'TO ORACLE' FROM DUAL;

OUTPUT:

WELCOME TO ORACLE

EX:

SQL> SELECT 'Mr.'||ENAME||' '||'IS WORKING AS A'||' '||JOB FROM EMP;

OUTPUT:

Mr. SMITH IS WORKING AS A CLERK

DISTINCT KEYWORD:

- THIS KEYWORD IS USED TO ELIMINATE DUPLICATE VALUES AND DISPLAY

UNIQUE VALUES IN QUERY RESULT.

EX:

SQL> SELECT DISTINCT JOB FROM EMP;

SQL> SELECT DISTINCT DEPTNO FROM EMP;

How to create a new table from the old table:

syntax1:

create table <new table name> as select * from <old table name>;

EX:

CREATE TABLE NEWEMP AS SELECT * FROM EMP;

NOTE: created a new table with copy of all rows & columns from the old table.

syntax2:

create table <new table name> as select * from <old table name> where <false condition>;

Ex:

CREATE TABLE DUMMYEMP AS SELECT * FROM EMP WHERE 1=2;

NOTE: created a new table without copy rows from old table. (Columns copy)

EX:

CREATE TABLE SPECEMP AS SELECT EID, EADDRESS FROM EMP;

NOTE: created a new table with specific columns from the old table.

EX:

CREATE TABLE SPECROWS AS SELECT * FROM EMP WHERE EADDRESS='HYD';

NOTE: created a new table with specific rows from the old table.

How to copy data from one table to another table:

syntax:

insert into <destination table name> select * from <source table name>;

EX:

CREATE TABLE DESTEMP (EMPNO INT, NAME CHAR (10), LOC VARCHAR2(10));

INSERT INTO DESTEMP SELECT * FROM EMP;

EX:

CREATE TABLE DEMP AS SELECT * FROM EMP WHERE 1=0;

INSERT INTO DEMP SELECT * FROM EMP;

INSERT ALL:

- IT IS A DML COMMAND (ORACLE 9i). WHICH IS USED TO INSERT ROWS INTO MULTIPLE TABLES AT A TIME.BUT THE ROWS SHOULD BE AN EXISTING TABLE.

SYNTAX:

INSERT ALL INTO <TN1> VALUES (<COLUMN NAME1>, <COLUMN NAME2>, ……)

INTO <TN2> VALUES (<COLUMN NAME1>, <COLUMN NAME2>, ……)

INTO <TN3> VALUES (<COLUMN NAME1>, <COLUMN NAME2>, ……)

……

……

INTO <TN n> VALUES (<COLUMN NAME1>, <COLUMN NAME2>, ……)

SELECT * FROM <OLD TABLE NAME>;

EX:

STEP1:

SQL> SELECT * FROM DEPT; ----------OLD TABLE

STEP2: CREATING EMPTY TABLES:

SQL> CREATE TABLE TEST1 AS SELECT * FROM DEPT WHERE 1=0;

SQL> CREATE TABLE TEST2 AS SELECT * FROM DEPT WHERE 1=0;

SQL> CREATE TABLE TEST3 AS SELECT * FROM DEPT WHERE 1=0;

STEP3:

```
SQL> INSERT ALL INTO TEST1 VALUES (DEPTNO, DNAME, LOC)
     INTO TEST2 VALUES (DEPTNO, DNAME, LOC)
     INTO TEST3 VALUES (DEPTNO, DNAME, LOC)
     SELECT * FROM DEPT;
```

STEP4:

```
SQL> SELECT * FROM TEST1;
SQL> SELECT * FROM TEST2;
SQL> SELECT * FROM TEST3;
```

MERGE COMMAND:

- IT IS A DML COMMAND (ORACLE 9i).IT IS USED TO TRANSFER DATA FROM SOURCE TABLE TO DESTINATION TABLE.

- IF DATA IS MATCHING IN BOTH TABLES THEN THOSE MATCHING DATA /ROWS ARE OVERRIDE ON DESTINATION TABLE BY USING "UPDATE COMMAND" WHERE AS DATA IS NOT MATCHING THEN THOSE UN MATCHING DATA / ROWS ARE TRANSFERING FROM SOURCE TABLE TO DESTINATION TABLE BY USING "INSERT COMMAND".

SYNTAX:

```
MERGE INTO <DESTINATION TABLE NAME> <ALIAS NAME> USING
<SOURCE TABLE NAME> <ALIAS NAME> ON (<JOIINING CONDITION>)
WHEN MATCHED THEN
UPDATE SET <DEST.TABLE ALIAS NAME>. <COLUMN
NAME1>=<SOUR.TABLE ALIAS NAME>. <COLUMN NAME1>,
.....
WHEN NOT MATCHED THEN
INSERT (<DESTINATION TABLE COLUMNS>) VALUES (<SOURCE TABLE
COLUMNS>);
```

EX:

STEP1:

```
SQL> SELECT * FROM DEPT;
```

STEP2:

```
SQL> CREATE TABLE NEWDEPT AS SELECT * FROM DEPT;
```

STEP3:

```
SQL> INSERT INTO NEWDEPT VALUES (50,'DBA','HYD');
SQL> INSERT INTO NEWDEPT VALUES (60,'SAP','MUMBAI');
```

STEP4:

```
SQL> SELECT * FROM NEWDEPT; ------SOURCE TABLE
SQL> SELECT * FROM DEPT; ----------OLD TABLE
```

STEP5:

```
SQL> MERGE INTO DEPT D USING NEWDEPT S ON (D. DEPTNO=S.DEPTNO)
        WHEN MATCHED THEN
        UPDATE SET D. DNAME=S.DNAME, D.LOC=S.LOC
        WHEN NOT MATCHED THEN
        INSERT (D. DEPTNO, D. DNAME, D.LOC) VALUES (S. DEPTNO,
        S. DNAME, S.LOC);
```