THE STRUCTURED QUERY LANGUAGE

Unit 1

INTRODUCTION

- Database is an organized collection of information about an entity having controlled redundancy and serves multiple applications.
- DBMS(Database Management System) is an application software that is developed to create and manipulate data in a database.
- Query language can access data in a database. SQL is used by most relational database systems.

WHAT IS SQL? /FEATURES OF SQL

- Stands for Structured Query Language
- It's a standard query language.
- Its an ANSI (American National Standards Institute) standard.
- Its used with all relational databases for data definition and manipulation.
- It's a non-procedural language ; because it only specifies what is to be done and not how its to be done.
- It's a higher-level language; Its an English-like language.
- Its very user friendly.
- It can process a single record as well as a set of records at a time.

- It’s a data sub-language consisting of three built in languages: DDL, DML and DCL.
- In insulates the user from underlying structure and algorithm.
- Has facilities for defining views, security, integrity constraints , transaction controls etc..

DIFFERENCE BETWEEN SQL AND SQL* PLUS

SQL	SQL* Plus
It's a language	Its an environment
Its based on ANSI(American National Standard Institute) standard SQL.	Its oracle proprietary interface for executing SQL statements.
It is entered into the SQL buffer on one or more lines.	It is entered one line at a time, not stored in the buffer.
It cannot be abbreviated.	It can be abbreviated
It uses a termination character to execute command immediately.	It does not require termination character. Commands are executed immediately.
SQL statements manipulate data and table definition in the database.	It does not allow manipulation of values I in the database.

DATA DEFINITION LANGUAGE

- It defines a set of commands used in the creation and modification of schema objects like table, views, indexes etc..
- It provides the able to create, alter and delete these objects.
- The current transactions are implicitly committed ie, the changes made by these are permanently stored in the database.

➢ **CREATE TABLE Command**

Syntax:

CREATE TABLE <table_name>(

Column_name1 data_type(size)[constraints],

Column_name2 data_type(size)[constraints],

Column_name3 data_type(size)[constraints],

……..

);

- Where,

  table_name : name of the table

  Column_name : name of the field

  Data_type : data type of the field

  Size: allocates specified size to the field

- Rules for Creation of TABLE
  - Table name should start with an alphabet.
  - Blank spaces and single quotes are not allowed in table name.
  - Table name cannot take reserved words as values.

- Proper data type and size should be specified.
- Column name should be unique.

Column Constraints: NOT NULL, UNIQUE, PRIMARY KEY, CHECK, DEFAULT, REFERENCES.

- Whenever a parent row is deleted the corresponding child rows are deleted from the referencing table.

➢ **ALTER TABLE Command:**

This command is used for modification of existing structure of the table in the following situation:

- When a new column is to be added to the table structure.
- When the existing column definition has to be changed(changing the width of the data type or the data type itself)
- When integrity constraints have to be added or deleted.

- When a constraint has to be enabled or disabled.

Syntax:

ALTER TABLE <table name> ADD(<column name><data type>…);

ALTER TABLE<table name> MODIFY(<column name><data type>….);

ALTER TABLE<table name>ADD CONSTRAINT<constraint name><constraint type>(field name);

ALTER TABLE <table name>DROP<constraint name>;

ALTER TABLE <table name> ENABLE/DISABLE<constraint name>;

Example:

ALTER TABLE Customers MODIFY(cno NUMBER(7));

➢ DROP TABLE Command

- This command is used to delete the table from the database.

Syntax:

DROP TABLE <table_name>

Example:

DROP TABLE Customers;

DATA MANIPULATION LANGUAGE

- It defines a set of commands that are used to query and modify data within existing schema objects.
- Commit is not implicit ;i.e. changes are not permanent till explicitly committed.
- DML commands available are:
  - SELECT
  - INSERT
  - UPDATE
  - DELETE

SELECT STATEMENT

- Its used to retrieve data from the databases.
- Result is stored in a result table, called the result-set.
- It can be used with many clauses.
- Some of the clauses used are:

1. Arithmetic operator:

Example: SELECT ENAME, SAL, SAL + 300 FROM EMP;

2. Operator Precedence

The operators used in SQL are*,/,+,-.

Operators are evaluated from left to right. Parentheses are used to force prioritized evaluation.

Example:

SELECT ENAME,SAL,12*SAL+100 FROM EMP;

SELECT ENAME,SAL,12*(SAL+100) FROM EMP;

3. Using Column aliases

To print column name as NAME and ANNUAL SALARY

SELECT ENAME "NAME", SAL*12 "ANNUAL SALARY" FROM EMP;

4. Concatenation operator

To print name and job as a single string with column name employees.

SELECT ENAME||JOB "EMPLOYEES" FROM EMP;

5. Distinct operator

- Used to eliminate duplicate rows.
- Used to return only distinct values.

Syntax:

SELECT DISTICT Column_name FROM table_name;

PERSON TABLE

P_Id	LastName	Firstname	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Example:

SELECT DISTICT City FROM PERSON;

The result set :

City
Sandnes
Stavanger

6. ORDER BY Keyword

- Its used to sort the result-set by a specified column.
- Its used in the last portion of select statement.
- It sorts the records in ascending order by default.
- To sort the records in descending order, DESC keyword is used.
- Sorting by column and column Alias is possible.

Syntax:

SELECT column_name(s) FROM table_name ORDER BY column_name(s) ASC|DESC;

Example:

To sort the persons in PERSON table by their last name.

SELECT * FROM PERSON ORDR BY LastName;

The result set:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger
2	Svendson	Tove	Borgvn 23	Sandnes

- To sort based on multiple columns; ascending order on department number and descending order of salary.

Example: SELECT ENAME,DEPTNO,SAL FROM EMP ORDER BY DEPTNO,SAL DESC;

7. Special Comparison operator used in where clause

i. LIKE

- Its used to search for a specified pattern in a column.
- Used only with char and varchar2.
- % denotes zero or many characters.
- _ denotes one character.
- Combination of % and _ can be used.

Example 1: Person table

To select the persons living in a city that starts with "s".

SELECT * FROM PERSONS WHERE City LIKE's%';

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Example 2:

To select the person who are living in a city that ends with an “s” from the PERSON table.

SELECT * FROM PERSON WHERE City LIKE ‘%s’;

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteiven 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes

Example 3:

To print the persons living in a city that contains the pattern “tav” from the PERSON table.

SELECT * FROM PERSON WHERE City LIKE ‘%tav%’;

Example 4:

To print the persons living in a city that does not contain the pattern “tav” from the PERSON table.

SELECT * FROM PERSON WHERE City NOT LIKE ‘%tav%’;

Example 5:

To print the persons having A as second character in LastName.

SELECT * FROM PERSON WHERE LastName LIKE ‘_A%’;

ii. between…. and….

- It gives range between two values(inclusive of both the values).

Example:

SELECT ENAME,SAL FROM EMP WHERE SAL BETWEEN 1000 and 1500;

iii. In(list)

- Used to match any of list values.

Example:

SELECT EMPNO,ENAME,SAL,MGR FROM EMP WHERE MGR IN(7902,7566,7788);

7902,7566,7788 are Employee numbers .

iv. Is null operator

Example: To find employee whose manager-id is not specified.

SELECT ENAME,MGR FROM EMP WHERE MGR IS NULL;

8. Logical operators:

Rules of precedence:

Order evaluated	Operator
1	All comparison operators
2	NOT
3	AND
4	OR

Example: To print those records of salesman or president who are having salary above 15000/-

SELECT ENAME,JOB,SAL FROM EMP WHERE(JOB='SALESMAN' or JOB='PRESIDENT') AND SAL>15000;

9. Aggregate functions:

- These functions help in getting consolidated information from a group of tuples.
- The aggregate functions are:
  - AVG: calculates the average of a set of values.

- COUNT: counts rows in a specified table or view.
- MIN: gets the minimum value in a set of values.
- MAX: gets the maximum value in a set of values.
- SUM: calculates the sum of values.

Example 1:

To find total number of employees.

```
SELECT COUNT(*) FROM EMP;
```

Example 2:

To find the maximum, minimum and average salaries of employees of department D1.

```
SELECT MIN(SAL),MAX(SAL),AVG(SAL) FROM EMP WHERE DEPTNO='D1';
```

10. GROUP BY

- Used to divide the rows in a table into small groups.

Syntax:

SELECT “Column_name1”, function(“Column_name2”) FROM “table_name” WHERE “CONDITION” GROUP BY “ Column_name1”;

Example:

SELECT Store, MAX(sale_Amount) FROM SALES_STORE GROUP BY Store;

11. Having clause

- Used for creating conditions on grouped information.

Syntax:

SELECT “Column_name1”, Function(“Column_name2”) FROM “table_name” GROUP BY “ Column_name1” HAVING (CONDITION based on Function);

Example:

SELECT Store, SUM(Sales_Amount) FROM SALES_HISTORY GROUP BY Store HAVING SUM(Sales_Amount)>100;

INSERT INTO COMMAND

- Values can be inserted for all columns or for the selected columns.
- Instead of values parameter substitution can be used. In this case, values will be passed at run-time.

Syntax:

INSERT INTO table_name VALUES(value1, value2, value3,…., valueN);

Example 1:

INSERT INTO PERSON(4,'Laxman');

Example 2:

INSERT INTO PERSON(&1,'&2');

UPDATE COMMAND

Syntax:

UPDATE <table name> SET <Column_name>=<value> WHERE <condition>;

- Sub query in the UPDATE command:

UPDATE EMP SET COMM=COMM*2 WHERE 2<=(SELECT COUNT(*) FROM INCR WHERE INCR.EMPNO=EMP.EMPNO GROUP BY EMPNO);

DELETE COMMAND

Delete the records of employees who have not got commission.

```
DELETE FROM EMP WHERE EMPNO NOT IN( SELECT EMPNO FROM INCR);
```

ADVANTAGES OF SQL

- A standard for database query language
- Easy to learn:consists of English statements and it is very easy to learn and understand a SQL query.
- Portability: run in programs in mainframes, PCs, laptops, servers and even mobile phones. It runs in local systems, intranet and internet. Databases using SQL can be moved from device to another without any problems.
- SQL Queries can be used to retrieve large amounts of records from a database quickly and efficiently.
- SQL is used to view the data without storing the data into the object.
- SQL joins two or more tables and show it as one object to user.
- it is easier to manage database systems without having to write substantial amount of code.
- SQL restricts the access of a table so that nobody can insert the rows into the table.

- **Used for relational databases:** SQL is widely used for relational databases.
- **Client/Server language:** SQL is used for linking front end computers and back end databases. Thus, providing client server architecture.
- **Dynamic database language:** By the use of SQL database structure can be changed in a dynamic fashion even when the contents of the database are accessed by users at the same time.
- SQL supports the latest object based programming and is highly flexible.

DISADVANTAGES OF SQL

- **Difficult Interface :** SQL has a complex interface that makes it difficult for some users to access it.
- **Cost**: The operating cost of some SQL versions makes it difficult for some programmers to access it.
- Lack of orthogonality: may different ways to express some queries.
- Its becoming enormous.

DATA CONTROL LANGUAGE

- Commands that allow system and data privileges to be passed to various users.
- Available to database Administrator.
- Some of the DCL commands:

✓ **Create a new user**:

CREATE USER <user_name> IDENTIFIED BY<Password>

Example:

CREATE USER XYZ IDENTIFIED BY abc124

- **Grant**: its used to provide database access permission to users.
- Its of two type:

1. System level permission
2. Object level permission

- System level permission

- On creation of a session.
- Not portable.

-GRANT CREATE SESSION TO MCA12;

- Object level permission:

- GRANT SELECT on EMP TO MCA12;

- **Revoke**: is to cancel the permission granted.

- All permissions or some permissions can be cancelled.

Syntax

REVOKE ALL ON EMP FROM MCA12;(All permissions will be revoked).

- **Drop**: A user-id can be deleted by using drop command.

- DROP USER MCA12;

DATABASE OBJECTS: VIEWS, INDEXES, SEQUENCES AND SYNONYMS

➢ Views

- A view is like a window through which data from tables can be viewed or changed. The table on which a view is based is a base table.
- Its stored as SELECT statement in a data dictionary.
- When a user wants to retrieve data from view, the following steps has to be followed:

1. View definition is retrieved from data dictionary table.
2. Checks access privileges for the view base table.
3. Converts the view query into a n equivalent operation on the underlying base table.

ADVANTAGES OF VIEWS:

- It restricts data access.
- It makes complex queries look easy.
- It allows data independence.
- It presents different views of the same data.

Types of views:

- Simple views
- Complex views

Feature	Simple Views	Complex Views
Number of tables	One	One or more
Contain Functions	No	Yes
Contain groups of data	No	Yes
Data Manipulation	Is allowed	Not always

- To see all details about existing views in Oracle:

SELECT * FROM USER_VIEWS;

Creating a view:

- A query can be embedded within CREATE VIEW STATEMET.
- A query can contain complex select statements including join, groups and sub-queries.
- A query that defies the view cannot contain an order by clause.
- DML operation and Views:

- Delete cannot be performed if view contains:
- Group function
- A group by clause
- A distinct keyword
- Modify cannot be performed if view contains:
- Group functions
- A group by clause
- A distinct keyword
- Columns defined by expressions

- Insert cannot be performed in view if it contains:

- Group functions
- A group by clause
- A distinct keyword
- Columns defined by expressions
- There are NOT NULL Columns in the base tables that are not selected by view.

- Syntax:

```
CREATE VIEW view_name
AS
SELECT column_list
FROM table_name [WHERE condition];
```

Where,

- ***view_name*** is the name of the VIEW, The SELECT statement is used to define the columns and rows that you want to display in the view.

Example:

CREATE VIEW CUSTOMER_VIEW AS SELECT name, age FROM CUSTOMERS;

- Creating views with check option:

The purpose of the WITH CHECK OPTION is to ensure that all UPDATE and INSERTs satisfy the condition(s) in the view definition.

Example:

CREATE VIEW CUSTOMERS_VIEW AS SELECT name, age FROM CUSTOMERS WHERE age is NOT NULL WITH CHECK OPTION;

- Creating views with read only option:

This is done to ensure that no DML operations can be performed on the view.

- Creating views with Replace option:
- Its used to change the definition of the view without dropping and recreating it or regranting object privileges previously granted on it.

SEQUENCES

- Automatically generate unique numbers
- Are sharable
- Are typically used to create a primary key value.
- Replace application code
- Speed up efficiency of accessing sequence values when cached in memory.

Example:

Create a sequence named SEQSS that starts at 105, has a step of 1 and can take maximum value as 2000.

CREATE SEQUENCE SEQSS START WITH 105 INCREMENT BY 1 MAX VALUE 2000;

Example demonstration:

Suppose a table PERSON exist as below:

SELECT * FROM PERSON;

CODE	NAME	ADDRESS
104	RAMESH	MUMBAI

- If we give the command,

  INSERT INTO PERSON VALUES(SEQSS.NEXTVAL, &NAME, &ADDRESS)

  On executing the above query give input as:

  Enter value for name: Nidhi

  Enter value of Address: Delhi

- On giving the command:

  SELECT * FROM PERSON;

CODE	NAME	ADDRESS
104	RAMESH	MUMBAI
105	Nidhi	Delhi

- The sequences minimum value, maximum value and step or increment will be stored in data dictionary.

gaps in sequences occurs when:

- A rollback occurs i.e., when statement changes are not accepted.
- The system crashes
- A sequence is used in another table.

Modify a sequence:

ALTER SEQUENCE SEQSS INCREMENT 2 MAXVALUE 3000;

Remove a sequence:

DROP SEQUENCE SEQSS;

INDEXES

Properties of Indexes:

- It's a schema object.
- It can be created automatically or explicitly.
- Used to speed up the retrieval of rows.
- They are logically and physically independent of the table. It means they can be created or dropped at any time and will have no affect on the base tables or other indexes.
- When a table is dropped corresponding indexes are also dropped.

Creation of Indexes

- Automatically: When a primary key or UNIQUE constraint is defined in a table definition then a unique index is created automatically.
- Manually: User can create non-unique indexes on columns to speed up access time of rows.

Example:

```
CREATE INDEX EMP_ENAME_IDX ON EMP(ENAME);
```

The above creates an index on employee name.

Finding details of created indexes:

- Data dictionary contains the name of the index, table name and column names.

Removing an index from the data dictionary:

DROP INDEX EMP_ENAME_IDX;

INDEXES cannot be modified.

SYNONYMS

- It gives short names and alternative names for objects.

Example:

CREATE SYNONYM D30 FOR EMPD30;

Removing a synonym:

DROP SYNONYM D30;

TABLE HANDLING

- In RDBMS multiple tables can be handled at a time by using the join operation.
- It's a relational operation that causes two or more with a common domain to be combined into a single table or view.
- Two tables may be joined when each contains a column that shares a common domain with the other. The result of join operation is a single table.
- Important rule of table handling: There should be one condition within the WHERE clause for each pair of tables being joined. For eg. Two combine two tables we will require one condition and to combine three tables we will require two conditions.

TYPES OF JOINS (LEARN FROM UNIT 2 BLOCK 1)

- Equi join:

- It's a join in which the joining condition is based on equality between values in the common columns.

- Common columns appear in the result table.

- Syntax:

SELET column_list FROM table1, table2…. WHERE table1.column_name=table2.column_name;

AGENT TABLE

AGENT_CODE	AGENT_NAME	WORKING_AREA	COMMISSION	PHONE_NO	COUNTRY
A007	Ramasundar	Bangalore	0.15	077-25814763	
A003	Alex	London	0.13	075-12458969	
A008	Alford	New York	0.12	044-25874365	
A011	Ravi Kumar	Bangalore	0.15	077-45625874	
A010	Santakumar	Chennai	0.14	007-22388644	
A012	Lucida	San Jose	0.12	044-52981425	
A005	Anderson	Brisban	0.13	045-21447739	
A001	Subbarao	Bangalore	0.14	077-12346674	
A002	Mukesh	Mumbai	0.11	029-12358964	
A006	McDen	London	0.15	078-22255588	
A004	Ivan	Torento	0.15	008-22544166	
A009	Benjamin	Hampshair	0.11	008-22536178	

CUSTOMER TABLE

CUST_CODE	CUST_NAME	CUST_CITY	WORKING_AREA	CUST_COUNTRY	GRADE	OPENING_AMT	RECEIVE_AMT	PAYMENT_AMT	OUTSTANDING_AMT	PHONE_NO	AGENT_CODE
C00013	Holmes	London	London	UK	2	6000.00	5000.00	7000.00	4000.00	BBBBBBB	A003
C00001	Micheal	New York	New York	USA	2	3000.00	5000.00	2000.00	6000.00	CCCCCCC	A008
C00020	Albert	New York	New York	USA	3	5000.00	7000.00	6000.00	6000.00	BBBBSBB	A008
C00025	Ravindran	Bangalore	Bangalore	India	2	5000.00	7000.00	4000.00	8000.00	AVAVAVA	A011
C00024	Cook	London	London	UK	2	4000.00	9000.00	7000.00	6000.00	FSDDSDF	A006
C00015	Stuart	London	London	UK	1	6000.00	8000.00	3000.00	11000.00	GFSGERS	A003
C00002	Bolt	New York	New York	USA	3	5000.00	7000.00	9000.00	3000.00	DDNRDRH	A008
C00018	Fleming	Brisban	Brisban	Australia	2	7000.00	7000.00	9000.00	5000.00	NHBGVFC	A005
C00021	Jacks	Brisban	Brisban	Australia	1	7000.00	7000.00	7000.00	7000.00	WERTGDF	A005
C00019	Yearannaidu	Chennai	Chennai	India	1	8000.00	7000.00	7000.00	8000.00	ZZZZBFV	A010
C00005	Sasikant	Mumbai	Mumbai	India	1	7000.00	11000.00	7000.00	11000.00	147-25896312	A002
C00007	Ramanathan	Chennai	Chennai	India	1	7000.00	11000.00	9000.00	9000.00	GHRDWSD	A010
C00022	Avinash	Mumbai	Mumbai	India	2	7000.00	11000.00	9000.00	9000.00	113-12345678	A002
C00004	Winston	Brisban	Brisban	Australia	1	5000.00	8000.00	7000.00	6000.00	AAAAAAA	A005
C00023	Karl	London	London	UK	0	4000.00	6000.00	7000.00	3000.00	AAAABAA	A006
C00006	Shilton	Torento	Torento	Canada	1	10000.00	7000.00	6000.00	11000.00	DDDDDDD	A004
C00010	Charles	Hampshair	Hampshair	UK	3	6000.00	4000.00	5000.00	5000.00	MMMMMMM	A009
C00017	Srinivas	Bangalore	Bangalore	India	2	8000.00	4000.00	3000.00	9000.00	AAAAAAB	A007
C00012	Steven	San Jose	San Jose	USA	1	5000.00	7000.00	9000.00	3000.00	KRFYGJK	A012
C00008	Karolina	Torento	Torento	Canada	1	7000.00	7000.00	9000.00	5000.00	HJKORED	A004
C00003	Martin	Torento	Torento	Canada	2	8000.00	7000.00	7000.00	8000.00	MYUIRED	A004

```
SELECT agents.agent_name,customer.cust_name,
customer.cust_city
FROM agents,customer
WHERE agents.working_area=customer.cust_city;
```

NATURAL JOIN

- Its same like equi join.
- It's the most commonly used form of join operation.
- columns with the same name of associated tables will appear once only.

SELF JOIN

- Join in which a table is joined with itself.

OUTER JOIN

- It even joins those tuples that do not have matching values in common columns are also included in the result table.
- It places null values in columns that does not have a match between tables.
- Left outer join: it considers only non-matching tuples of tables on the left side of the join expression.
- Right outer join or complete outer join

ACCESS CONTROL MECHANISM

- Each user has an authorization identifier. Its allocated by DBA.
- Each object has an owner. At first only the user has access to an object but the owner passes privileges to carry out certain actions onto other users via GRANT statement and take away given privileges via REVOKE statement.

NESTED QUERIES

- Issues of sub-queries:

- sub-query is a SELECT statement that is embedded in a clause of another SELECT statement. Its also referred to as NESTED SELECT or SUB SELECT or INNER SELECT.
- It executes first before the main query. The result of sub-query is used by the main query(outer query).
- It can be placed in WHERE or HAVING or FROM clauses.
- Order by clause cannot be used in sub-query, if specified it must be the last clause in the main statement.

- Group functions can be used in sub queries.
- Outer and inner queries can get data from different tables.
- Format :

SELECT <select_list> FROM <table> WHERE expr OPERATOR(SELECT <select_list> FROM <TABLE> WHERE);

Operator includes a comparison operator. Operator can be single row operator and multiple row operators.

Single row operator: >,=,>=,<,<=,<>

Multiple row operators: IN,ANY,ALL

- Types of sub-queries:

- Single row sub-query: It returns only one row from the inner select statement.
- Multiple row sub-queries: It returns more than one row from the inner select statement.
- Multiple column sub-queries: It returns more than one column from the inner select statement.