Introduction to Structured Query Language

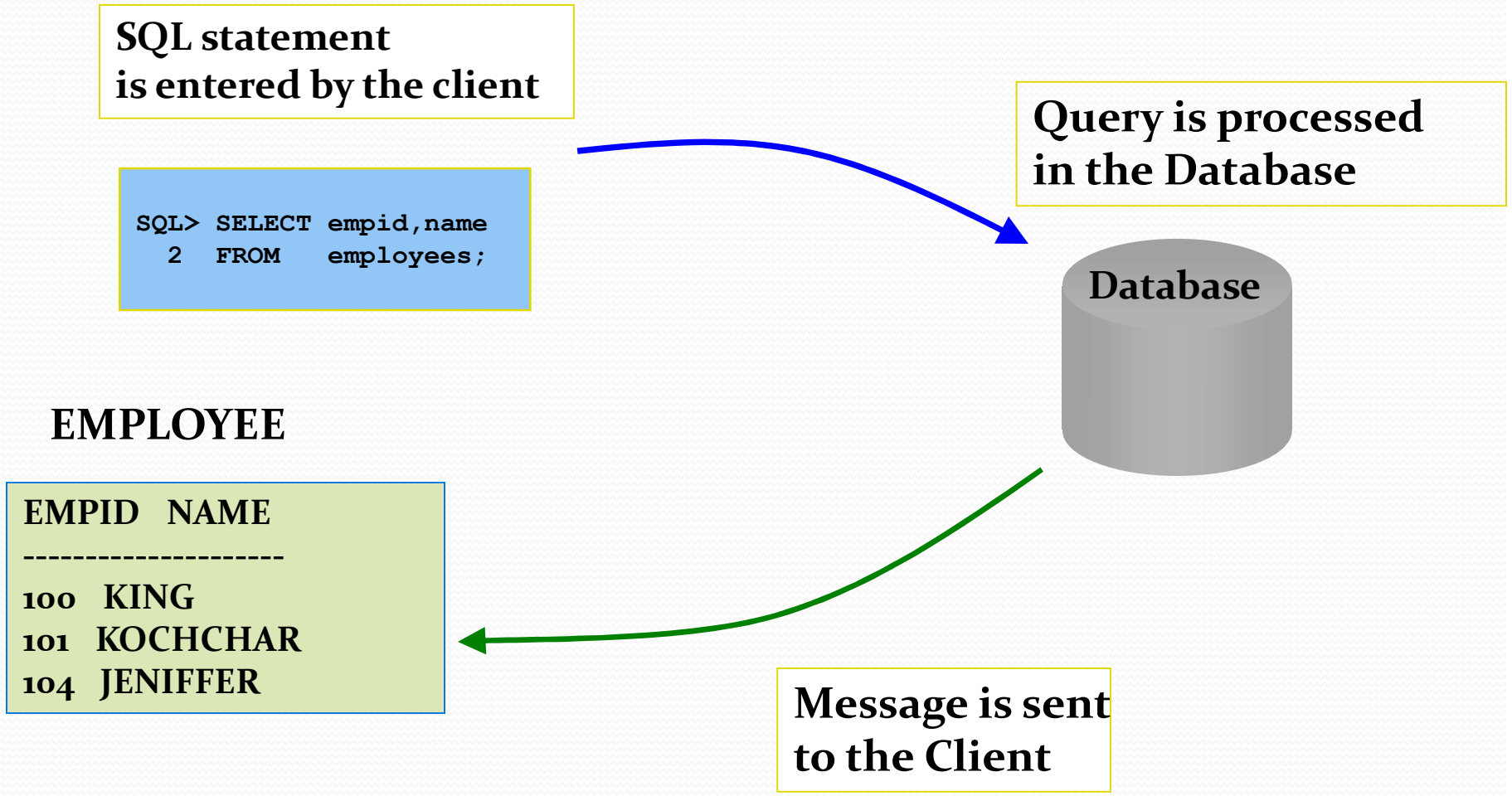
Objectives

- Structured query Language.
- Different SQL Statements.
- Data Types
- Writing SQL Statements
- Retrieving information from table.
- Using Operators
- Where & Order by clauses.

Structured Query Language

Structured Query Language (SQL) is the set of statements with which all programs and users access data in an Oracle database

RDBMS Using SQL



Tasks of SQL

SQL provides statements for a variety of tasks:

- Querying data
- Inserting, updating, and deleting rows in a table
- Creating, replacing, altering, and dropping objects
- Controlling access to the database and its objects
- Guaranteeing database consistency and integrity

SQL unifies all of the preceding tasks in one consistent language.

SQL Statements

- SELECT

Data retrieval

- INSERT
- UPDATE
- DELETE

Data Manipulation Language (DML)

- CREATE
- ALTER
- DROP
- RENAME
- TRUNCATE

Data Definition Language (DDL)

- COMMIT
- ROLLBACK
- SAVEPOINT

Transaction Control Language (TCL)

- GRANT
- REVOKE

Data Control Language (DCL)

Data types

Data type	Description
VARCHAR2(*size*)	Variable-length character data
CHAR(*size*)	Fixed-length character data
NUMBER(*p*,*s*)	Variable-length numeric data
DATE	Date and time values
LONG	Variable-length character data up to 2 gigabytes
CLOB	Single-byte character data up to 4 gigabytes
RAW and LONG RAW	Raw binary data
BLOB	Binary data up to 4 gigabytes
BFILE	Binary data stored in an external file; up to 4 gigabytes

Writing SQL Statements

- SQL is **not** Case Sensitive.
- Keywords **cannot** be split or abbreviated.
- SQL Statements can be **split** across lines.
- Clauses are placed in different lines, to promote readability.

Retrieving Information from Tables

SELECT Statement

```
SELECT    [DISTINCT] {*, column [alias],...}
FROM      table;
```

- Select Clause determines *what* columns from the table has to be retrieved.
- The From Clause determines *which* table.

Selecting All Columns

```
SQL> SELECT *
  2  FROM   departments;
```

```
    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 EXECUTION      FLORIDA
        30 SALES          CHICAGO
        40 STOCK          BOSTON
```

Selecting Specific Columns

```
SQL> SELECT deptno, loc
  2  FROM   departments;
```

```
   DEPTNO LOC
--------- -------------
       10 NEW YORK
       20 FLORIDA
       30 CHICAGO
       40 BOSTON
```

Arithmetic Expressions

- Basic Arithmetic operators

Operator	Description
+	**Add**
-	**Subtract**
*	**Multiply**
/	**Divide**

Using Arithmetic Operators

```
SQL> SELECT name, salary,  salary+300
  2  FROM   employees;
```

```
NAME                 SALARY    SALARY+300
---------- --------- ----------
KING                 5000        5300
BLAKE                2850        3150
CLARK                2450        2750
JONES                2975        3275
MARTIN               1250        1550
ALLEN                1600        1900
...
14 rows selected.
```

Operator Precedence

* / + –

- Parentheses can force precedence

- Multiplication and Division followed by Addition and subtraction.

Operator Precedence

```
SQL> SELECT name, salary, 12*salary+100
  2  FROM   employees;
```

```
NAME                 SALARY  12*SALARY+100
---------- ---------   -----------
KING                 5000      60100
BLAKE                2850      34300
CLARK                2450      29500
JONES                2975      35800
MARTIN               1250      15100
ALLEN                1600      19300
...
14 rows selected.
```

Using Parentheses

```
SQL> SELECT name, salary, 12*(salary+100)
  2  FROM   employees;
```

```
NAME                 SALARY 12*(SALARY+100)
---------- ---------      -----------
KING                 5000         61200
BLAKE                2850         35400
CLARK                2450         30600
JONES                2975         36900
MARTIN               1250         16200
...
14 rows selected.
```

Defining a Null Value

- NULL is UNASSIGNED Value.

```
SQL> SELECT   name, job, comm_pct
  2  FROM     employees;
```

```
NAME       JOB              COMM_PCT
---------- --------- ---------
KING       PRESIDENT
BLAKE      MANAGER
...
TURNER     SALESMAN               0
...
14 rows selected.
```

Null Values in Arithmetic Expr

- NULL as an operand will result NULL

```
SQL> select name NAME, 12*salary+comm
  2  from   employees
  3  WHERE  ename='KING';
```

```
NAME        12*SALARY+COMM
----------- -----------
KING
```

Defining Column Alias

- The Heading name is replaced for the current SELECT Statement.
- AS Keyword [Optional] between the column name and the actual alias name
- Double Quotation Marks.

Using Column Aliases

```
SQL> SELECT ename AS name, sal salary
  2  FROM   employees;
```

```
NAME                 SALARY
-------------------- ---------

...
```

```
SQL> SELECT ename "Name",
  2         sal*12 "Annual Salary"
  3  FROM   employees;
```

```
Name                 Annual Salary
-------------------- -------------

...
```

Using Concatenation Operator (||)

```
SQL> SELECT   name||' '||job AS "Employees"
  2  FROM     employees;
```

```
Employees
------------------
KING PRESIDENT
BLAKE MANAGER
CLARK MANAGER
JONES MANAGER
MARTIN SALESMAN
ALLEN SALESMAN
...
14 rows selected.
```

Literal Character Strings

- Date and character literal values must be enclosed within single quotation marks.

Using 'DISTINCT' Clause

- Eliminate duplicate rows by using the DISTINCT keyword

```
SQL> SELECT DISTINCT deptno
  2  FROM   employees;
```

```
    DEPTNO
----------
        10
        20
        30
```

Using 'Where' and 'Order By'

- The WHERE clause Limit the rows required

- ORDER BY clause Sorts the rows in a particular order.

Using 'WHERE' Clause

- Specify the Selection of rows retrieved by the WHERE Clause.

```
SELECT      [DISTINCT] {*, column [alias], ...}
FROM        table
[WHERE      condition(s)];
```

- The WHERE clause follows the FROM clause.

Using WHERE Clause

```
SQL> SELECT ename, job, deptno
  2  FROM   employees
  3  WHERE  job='CLERK';
```

```
ENAME      JOB          DEPTNO
---------- --------- ----------
JAMES      CLERK            30
SMITH      CLERK            20
ADAMS      CLERK            20
MILLER     CLERK            10
```

Character Strings and Dates

- **Character / Dates are Represented by the Single Quotation Marks.**
- **Default date format is 'DD-MON-YY'**

```
SQL> SELECT   ename, job, deptno
  2  FROM     emp
  3  WHERE    ename = 'JAMES';
```

Comparison Operators

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

Using Comparison Operators

```
SQL> SELECT ename, sal, comm
  2  FROM   employees
  3  WHERE  sal<=comm;
```

```
ENAME          SAL       COMM
---------- ---------- ----------
MARTIN           1250 <--> 1400
```

More Comparison Operators

Operator	Meaning
BETWEEN …AND…	Between two values (inclusive)
IN(list)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value

Using BETWEEN Operator

```
SQL> SELECT   ename, sal
  2  FROM     employees
  3  WHERE    sal BETWEEN 1000 AND 1500;
```

```
ENAME          SAL
---------- ---------
MARTIN        1250
TURNER        1500
WARD          1250
ADAMS         1100
MILLER        1300
```

Lower limit

Higher limit

- Used to compare between range of values.

  Values Specified are inclusive.

Using IN Operator

- IN Operator to check with a List of Values.

```
SQL> SELECT   empno, ename, sal, mgr
  2  FROM     emp
  3  WHERE    mgr IN (7902, 7566, 7788);
```

```
     EMPNO ENAME                SAL        MGR
---------- ---------- ---------- ----------
      7902 FORD                3000       7566
      7369 SMITH                800       7902
      7788 SCOTT               3000       7566
      7876 ADAMS               1100       7788
```

Using LIKE Operator

- Like Keyword Does Wildcard Searches in Valid String Values..

% ---------- zero or many characters

_ ---------- one character

```
SQL> SELECT   ename
  2  FROM     emp
  3  WHERE    ename LIKE 'S%';
```

Using LIKE Operator

- ESCAPE identifier to search for "%" or "_".

```
SQL> SELECT   ename
  2  FROM     emp
  3  WHERE    ename LIKE '_A%';
```

```
ENAME
----------
JAMES
WARD
```

Using IS NULL Operator

- To Check for Null Values , IS NULL is used.

```
SQL> SELECT   ename, mgr
  2  FROM     emp
  3  WHERE    mgr IS NULL;
```

```
ENAME           MGR
---------- ----------
KING
```

Logical Operators

Operator	Meaning
AND	**Returns TRUE if *both* component conditions are TRUE**
OR	**Returns TRUE if *either* component condition is TRUE**
NOT	**Returns TRUE if the following condition is FALSE**

Using AND Operator

- AND requires both conditions to be TRUE.

```
SQL> SELECT empno, ename, job, sal
  2  FROM   emp
  3  WHERE  sal>=1100
  4  AND    job='CLERK';
```

```
     EMPNO ENAME      JOB              SAL
---------- ---------- --------- ----------
      7876 ADAMS      CLERK            1100
      7934 MILLER     CLERK            1300
```

Using OR Operator

- OR requires either condition to be TRUE.

```
SQL> SELECT empno, ename, job, sal
  2  FROM   emp
  3  WHERE  sal>=1100
  4  OR     job='CLERK';
```

```
     EMPNO ENAME      JOB              SAL
---------- ---------- --------- ----------

      7839 KING       PRESIDENT        5000
      7698 BLAKE      MANAGER          2850
      7782 CLARK      MANAGER          2450
      7566 JONES      MANAGER          2975
      7654 MARTIN     SALESMAN         1250
...
14 rows selected.
```

Using NOT Operator

```
SQL> SELECT ename, job
  2  FROM   emp
  3  WHERE  job NOT IN ('CLERK','MANAGER','ANALYST');
```

```
ENAME      JOB
---------- ---------
KING       PRESIDENT
MARTIN     SALESMAN
ALLEN      SALESMAN
TURNER     SALESMAN
WARD       SALESMAN
```

Rules of Precedence

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

'ORDER BY' Clause

- Sort rows specified by the order: ASC/DESC

```
SQL> SELECT    ename, job, deptno, hiredate
  2  FROM      emp
  3  ORDER BY hiredate;
```

```
ENAME      JOB              DEPTNO HIREDATE
---------- --------- ---------- ---------
SMITH      CLERK              20 17-DEC-80
ALLEN      SALESMAN           30 20-FEB-81
...
14 rows selected.
```

Sorting in Descending Order

```
SQL> SELECT    ename, job, deptno, hiredate
  2  FROM      emp
  3  ORDER BY hiredate DESC;
```

```
ENAME      JOB              DEPTNO HIREDATE
---------- --------- ---------- ---------
ADAMS      CLERK            20 12-JAN-83
SCOTT      ANALYST          20 09-DEC-82
MILLER     CLERK            10 23-JAN-82
JAMES      CLERK            30 03-DEC-81
FORD       ANALYST          20 03-DEC-81
KING       PRESIDENT        10 17-NOV-81
MARTIN     SALESMAN         30 28-SEP-81
...
14 rows selected.
```

Sorting the rows by Alias

```
SQL> SELECT   empno, ename, sal*12 annsal
  2  FROM     emp
  3  ORDER BY annsal;
```

```
     EMPNO ENAME          ANNSAL
---------- ---------- ----------
      7369 SMITH            9600
      7900 JAMES           11400
      7876 ADAMS           13200
      7654 MARTIN          15000
      7521 WARD            15000
      7934 MILLER          15600
      7844 TURNER          18000
...
14 rows selected.
```

Sorting by Multiple Columns

- **The order of ORDER BY list is the order of sort.**

```
SQL> SELECT    ename, deptno, sal
  2  FROM      emp
  3  ORDER BY  deptno, sal DESC;
```

```
ENAME          DEPTNO        SAL
---------- ---------- ----------
KING               10       5000
CLARK              10       2450
MILLER             10       1300
FORD               20       3000
...
14 rows selected.
```

Summary

- SQL queries to retrieve data from tables.
- Displaying alias named for columns.
- Using where clause to restrict data from tables.
- Using operators to display calculated results
- Using order by to display records in sorted order.