UNIT IV

FUNCTIONS

DEFINITION:

Functions are a collection of executable statement. Functions are used to perform a particular task. There are two types

1. Pre-defined functions
2. User defined functions

PRE-DEFINED FUNCTIONS:

Those function has pre-defined meaning/task.User cannot change task of functions.

USER DEFINED FUNCTIONS

Functions developer by users is called user defined function. Function may perform more than one task. But, assigning one task to function is advisible.

DATE FUNCTIONS

Those functions are used to perform operations on data values. We are going to discuss the following important Data Functions.

1. Now ()
2. Sysdate ()

The above two functions are used to return current date and time.

Eg: selectNow () from dual

Eg: selectSysdate () from dual

Curdate ()

This function is used to return only data from current data.

Eg: selectcurdate () from dual

Curtime ()

This function is used to return time of the current data.

Eg: selectcurtime () from dual

Datediff ()

It is used to find out the difference between two dates in term of days only.

Eg: selectdatediff (‘2018-08-31’,’2018-08-28’) from dual

Ans:3

Extract ()

This function is used to fetch any part from the system date.

Eg:

Order

Oid	Orddate		Amt
1	2018-08-31	10:30:05	1000

Eg: selectextract (year from orddate) As ‘orderyear’ from order

Eg: selectextract (month from orddate) As ‘ordermonth’ from order

Eg: selectextract (day from orddate) As ‘orderday’ from order

Eg: selectextract (hour from orddate) As ‘orderhour’ from order

Eg: selectextract (minutes from orddate) As ‘MINUTES’ from order

Eg: selectextract (seconds from orddate) As ‘SECONDS’ from order

Order year	ordermonth	Oreder day	HOUR	MINUTES	SECONDS
2018	**08**	**31**	**10**	**30**	**05**

CHARACTER FUNCTIONS

These functions are used to perform operations on strings or characters. There are many character functions in sql.we are discussing the following important functions.

1. ASCII()

This function return ASCII value of the first character of given string. We are permitted to pass a single character or a string to this function as a argument.

Eg:

Select ASCII(‘anbu’) from dual

Ans:97

Select ASCII (‘2’) from dual

Ans:50

2.BIN()

This function return Binary equivalent of the given numeric value as a argument.

Eg:

Select BIN(12) from dual

Ans:1100

3.Bit_length()

This function returns the length of the given string in bits.

Eg:

Select Bit_length(‘ram’) from dual

Ans:24bits

4.char()

This function returns equivalent character to every argument. This function may have one argument, more than one argument apart from that this function accepts numeric value as a argument.

Eg:Select char(65,97,68,99) from dual

Ans:AaDc

5.concat()

This function is used to join or concat more than one strings.we are permitted to pass more than two strings to this function as a argument.

Eg:

Select concat(‘Ram’,’kumar’) from dual

Ans:Ramkumar

Note:

If you pass null as a parameter to this function, null will be skipped.

Eg:

Select concat(‘siva’,’null’,’sindhu’) from dual

Ans:sivasindhu

6.concat_ws()

This function concat more than one string with separator .we should pass separator as a first argument in this function.

Eg:

Select concat_ws(‘,’,’firstname’,’lastname,) from dual

Ans:firstname,lastname

7.Lower()

This function used to convert capital letters into small letter.The function accept only one string as a argument.we can pass strings in capital letter,string in small letter to this string function as a value.

Eg:

Select Lower(“RAMAN”) from dualAns:raman

8.UPPER()

This Function is used to convert small characters into capital letters. This function accept only one argument.

Eg:select UPPER(‘kannan’) from dual

Ans:KANNAN

9.LEFT()

This function is used to fetch the single or set of characters from left side of the string. This function accept two parameters.

Eg:select LEFT(‘ramkumar’,3) from dual

Ans:ram

10.RIGHT()

This function acts against to the left function.This function returns a single character or set of character from the rightside of the given string.

Eg:select RIGHT(‘ramkumar’,3)

Ans:mar

11.LTRIM()

This function is used to remove leading spaces from the given string.

Eg:Select LTRIM(‘ ram’) from dual

Ans:ram

12.RTRIM()

This Function is used to remove trailing space from the given string.

Eg:select RTRIM(‘ram ‘) from dual

13.TRIM()

This function is used to remove spaces leading to given string, trailing to given string, both from leading and trailing.

Eg: selectTRIM (leading ‘x’ from ‘xxxramxxx’) from dual

Ans: ramxxx

14.LENGTH ()

This Function is used to return/find out the length of the given string.

Eg: selectLENGTH(‘text’) from dual

Ans:4

Eg: selectLENGTH (‘ram Kumar’) from dual

Ans:9

15.REVERSE ()

This function is used to return the reverse value of the given string.

Eg: selectREVERSE(‘text’) from dual

Ans:txet

16.REPEAT ()

This function print the given string more than one time.

Eg: selectREPEAT(‘sql’,3) from dual

Ans:sqlsqlsql

17.REPLACE ()

This function is used to replace any character in the given string by some other character.

Eg: select REPLACE(‘raman’,’a’,’x’)from dual

18.strcmp()

This function is used to compare two string for their equality. If both strings are equal, their function return 0 as a result. If first string is smaller than second string, this function returns -1 as a result. Suppose first string is greater than second string, this function returns +1 as result.

By, default this function accept only two string as an argument at a time. This function is capable for comparing string.

Using this function, we cannot compare two different numbers.

Eg: select strcmp(‘anbu’,’anand’) from dual

Ans:1

Select strcmp (‘ram’, ‘ram’) from dual

Ans:0

Select strcmp(‘anbu’,’man’) from dual

Ans: -1

19.substring ()

This function is used to fetch some part of the given string.

Eg: select substring('ramesh',3,2) From dual

Ans: me

NUMERIC FUNCTION

This Function performs operation on numbers and return numeric values as a result. There are many numeric functions in sql. We are discussing the following important Numeric Function.

1. abs ()

This function returns absolute value of argument persived in the function.

Eg:

Select abs(4) from dual

Select abs(-4)from dual

Note:

Dual is a dummy table which has one row and one column. This table is created by sql automatically.

2.celing()

This function return next integer which is part of the argument.

Eg:

Select ceiling(5.35)from dual

Ans:6

Select ceiling(-5.65)from dual

Ans:-5

3.floor()

This function returns lowest number of the argument.

Eg:

Select floor(5.73)from dual

Ans:5

4.greatest()

This function is used to find out the biggest number from the list of numbers.

Eg:

Select greatest(1,30,50,40,25) from dual

Ans:50

Note:

1. The list contains one number, more than one number.
2. This list contains character.
3. You can use this function on only one list at a time.

5. least()

This function is used to find out the least value or minimum value from the given list.

Eg:

Select least(1,30,50,40,25) from dual

Ans:1

6.power()

This function returns power of the given number.

Eg:

Select power(3,2) from dual

Ans:9

Select power(3,0) from dual

Ans:1

7.sqrt()

This function is used to return sqrt of the given number.

This function accept only one parameter at a time.

Eg:

Select sqrt(49) from dual

Ans:7

8.mod()

This function used to return modules of any division operation.

Eg:

Select mod(9,3) from dual

Ans:0

Select mod(3,9) from dual

Ans:3

AGGREGATE OR GROUP FUNCTION

Those functions apply on set of records and return only one value as a result. There are five important aggregate functions.

Important Functions:

- Count()
- Min()
- Max()
- Avg()
- Sum()

Count ()

This function return number of records stored in DataBase.

Eg:

Select Count (eno) from emp

Min()

This function is used to return minimum value from the DataBase table.

Eg:

Select Min(sal) from emp

Max()

This function is used to return maximum value from the DataBase table.

Eg:

Select Max(sal) from emp

Avg()

This function is used to return average value of particular column.

Eg:

Select Avg(sal) from emp

Sum()

This function is used to find out total value for particular column from our DataBase.

Eg:1

Select Sum (sal) from emp

Eg:2

Select Sum (sal) from emp group by dno

JOINS

It is used to fetch records from two different tables. Whereas, those two tables at least should have one common column. There are many types of join operator. They are,

- Inner Join
- Left join
- Right join
- Full join
- Self join
- Cartesian Product or Cross Join

INNER JOIN

It is used to fetch the common records from two different tables.

For Eg:

We take the following two tables.

Customer

CID	CNAME	ADDRESS	SALARY
1	X	Delhi	5000
2	Y	Chennai	2000
3	Z	Delhi	6000
4	M	Chennai	7000
5	N	Mumbai	8000

Order

OID	CID	AMT
1	2	5000
2	3	3000

Queries:

Select CID, CNAME, AMT from Customer inner join order on Customer.CID=Order.CID

CID	CNAME	AMT
2	Y	5000
3	Z	3000

LEFT JOIN

Left Join or Left Outer Join is also used to fetch records from two different tables. It returns all records from first table where as common records from second table.

Customer

CID	CNAME	ADDRESS	SALARY
1	X	Delhi	5000
2	Y	Chennai	2000
3	Z	Delhi	6000
4	M	Chennai	7000
5	N	Mumbai	8000

Order

OID	CID	AMT
1	2	5000
2	3	3000

QUERIES:

Select CID, CNAME, AMT from Customer LEFT Join order on Customer.CID =Order.CID

CID	CNAME	AMT
1	X	NULL

2	Y	5000
3	Z	3000
4	M	NULL
5	N	NULL

RIGHT JOIN

This join returns all records from second table and common records from first table.

Eg:

Customer

CID	CNAME	ADDRESS	SALARY
1	X	Delhi	5000
2	Y	Chennai	2000
3	Z	Delhi	6000
4	M	Chennai	7000
5	N	Mumbai	8000

Order

OID	CID	AMT
1	2	5000
2	3	3000

Query:

Select CID, CNAME, AMT from Customer RIGHT Join order on Customer.CID = order.CID

CID	CNAME	AMT
2	Y	5000

3	Z	3000

FULL JOIN

Full join or Full Outer Join. It returns all records from table one as well as all records from table two also.

Eg:

Customer

CID	CNAME	ADDRESS	SALARY
1	X	Delhi	5000
2	Y	Chennai	2000
3	Z	Delhi	6000
4	M	Chennai	7000
5	N	Mumbai	8000

Order

OID	CID	AMT
1	2	5000
2	3	3000

Query:

Select CID, CNAME, AMT FROM Customer Full Join order on Customer.CID=Order.CID

CID	CNAME	AMT
1	X	NULL

2	Y	5000
3	Z	3000
4	M	NULL
5	N	NULL
2	Y	5000
3	Z	3000

SELF JOIN

What is Self Join in SQL?

A self join is a join in which a table is joined with itself (which is also called Unary relationships), especially when the table has a FOREIGN KEY which references its own PRIMARY KEY. To join a table itself means that each row of the table is combined with itself and with every other row of the table.

The self join can be viewed as a join of two copies of the same table. The table is not actually copied, but SQL performs the command as though it were.

The syntax of the command for joining a table to itself is almost same as that for joining two different tables. To distinguish the column names from one another, aliases for the actual the table name are used, since both the tables have the same name. Table name aliases are defined in the FROM clause of the SELECT statement. See the syntax :

```
SELECT a.column_name, b.column_name FROM table1 a, table1 b
```

```
WHERE a.common_filed = b.common_field;
```

For this tutorial we have used a table EMPLOYEE, that has one-to-many relationship.

Code to create the table EMPLOYEE

SQL Code:

```
CREATE TABLE employee(emp_id varchar(5) NOT NULL,emp_name
varchar(20) NULL,dt_of_join date NULL,emp_supv varchar(5)
NULL, CONSTRAINT emp_id PRIMARY KEY (emp_id), CONSTRAINT
emp_supv FOREIGN KEY (emp_supv) REFERENCES employee (emp_id));
```

The structure of the table

In the EMPLOYEE table displayed above, emp_id is the primary key. emp_supv is the foreign key (this is the supervisor's employee id).

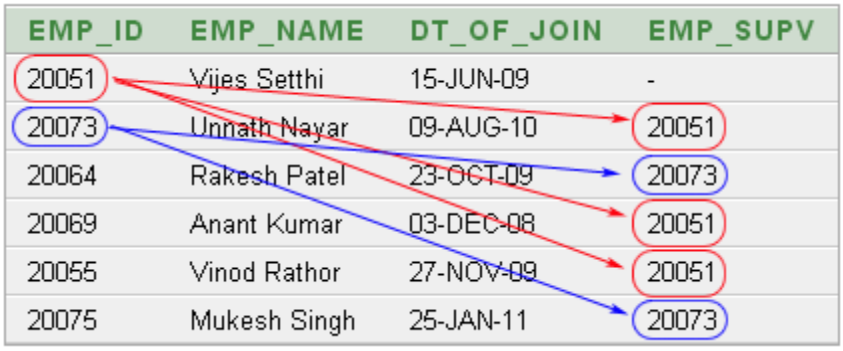
If we want a list of employees and the names of their supervisors, we'll have to JOIN the EMPLOYEE table to itself to get this list.

Unary relationship to employee

How the employees are related to themselves:

- An employee may report to another employee (supervisor).
-
- An employee may supervise himself (i.e. zero) to many employees (subordinates).

We have the following data into the table EMPLOYEE.



EMP_ID	EMP_NAME	DT_OF_JOIN	EMP_SUPV
20051	Vijes Setthi	15-JUN-09	-
20073	Unnath Nayar	09-AUG-10	20051
20064	Rakesh Patel	23-OCT-09	20073
20069	Anant Kumar	03-DEC-08	20051
20055	Vinod Rathor	27-NOV-09	20051
20075	Mukesh Singh	25-JAN-11	20073

The above data shows:

- Unnath Nayar's supervisor is Vijes Setthi
- Anant Kumar and Vinod Rathor can also report to Vijes Setthi.
- Rakesh Patel and Mukesh Singh are under supervison of Unnith Nayar.

Example of SQL SELF JOIN

In the following example, we will use the table EMPLOYEE twice and in order to do this we will use the alias of the table.

To get the list of employees and their supervisor the following SQL statement has used:

SQL Code:

```
SELECT a.emp_id AS "Emp_ID", a.emp_name AS "Employee
Name", b.emp_id AS "Supervisor ID", b.emp_name AS
"Supervisor Name" FROM employee a, employee b WHERE
a.emp_supv = b.emp_id;
```

Output:

Emp_ID	Employee Name	Supervisor ID	Supervisor Name
20055	Vinod Rathor	20051	Vijes Setthi
20069	Anant Kumar	20051	Vijes Setthi
20073	Unnath Nayar	20051	Vijes Setthi
20075	Mukesh Singh	20073	Unnath Nayar
20064	Rakesh Patel	20073	Unnath Nayar

Outputs of the said SQL statement shown here is taken by using Oracle Database 10g Express Edition.

Query:

Select a.CID,b.CNAME,a.SALARY from Customer a,Customer b where a.SALARY <b.SALARY

CARTESIAN JOIN or CROSS JOIN

The CARTESIAN JOIN or CROSS JOIN returns the Cartesian product of the sets of records from two or more joined tables. Thus, it equates to an inner join where the join-condition always evaluates to either True or where the join-condition is absent from the statement.

Syntax

The basic syntax of the **CARTESIAN JOIN** or the **CROSS JOIN** is as follows –

```
SELECT table1.column1, table2.column2...
FROM table1, table2 [, table3]
```

Example

Consider the following two tables.

Table 1 – CUSTOMERS table is as follows.

```
+----+----------+-----+-----------+----------+
```

```
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

Table 2: ORDERS Table is as follows –

```
+-----+---------------------+-------------+--------+
|OID  | DATE                | CUSTOMER_ID | AMOUNT |
+-----+---------------------+-------------+--------+
| 102 | 2009-10-08 00:00:00 |           3 |   3000 |
| 100 | 2009-10-08 00:00:00 |           3 |   1500 |
| 101 | 2009-11-20 00:00:00 |           2 |   1560 |
```

```
| 103 | 2008-05-20 00:00:00 |        4 |   2060 |

+-----+---------------------+------------+--------+
```

Now, let us join these two tables using CARTESIAN JOIN as follows –

```
SQL> SELECT ID, NAME, AMOUNT, DATE

  FROM CUSTOMERS, ORDERS;
```

This would produce the following result –

```
+----+----------+--------+---------------------+

| ID | NAME     | AMOUNT | DATE                |

+----+----------+--------+---------------------+

|  1 | Ramesh   |   3000 | 2009-10-08 00:00:00 |

|  1 | Ramesh   |   1500 | 2009-10-08 00:00:00 |

|  1 | Ramesh   |   1560 | 2009-11-20 00:00:00 |

|  1 | Ramesh   |   2060 | 2008-05-20 00:00:00 |

|  2 | Khilan   |   3000 | 2009-10-08 00:00:00 |

|  2 | Khilan   |   1500 | 2009-10-08 00:00:00 |

|  2 | Khilan   |   1560 | 2009-11-20 00:00:00 |

|  2 | Khilan   |   2060 | 2008-05-20 00:00:00 |

|  3 | kaushik  |   3000 | 2009-10-08 00:00:00 |

|  3 | kaushik  |   1500 | 2009-10-08 00:00:00 |

|  3 | kaushik  |   1560 | 2009-11-20 00:00:00 |

|  3 | kaushik  |   2060 | 2008-05-20 00:00:00 |
```

```
|  4 | Chaitali |   3000 | 2009-10-08 00:00:00 |
|  4 | Chaitali |   1500 | 2009-10-08 00:00:00 |
|  4 | Chaitali |   1560 | 2009-11-20 00:00:00 |
|  4 | Chaitali |   2060 | 2008-05-20 00:00:00 |
|  5 | Hardik   |   3000 | 2009-10-08 00:00:00 |
|  5 | Hardik   |   1500 | 2009-10-08 00:00:00 |
|  5 | Hardik   |   1560 | 2009-11-20 00:00:00 |
|  5 | Hardik   |   2060 | 2008-05-20 00:00:00 |
|  6 | Komal    |   3000 | 2009-10-08 00:00:00 |
|  6 | Komal    |   1500 | 2009-10-08 00:00:00 |
|  6 | Komal    |   1560 | 2009-11-20 00:00:00 |
|  6 | Komal    |   2060 | 2008-05-20 00:00:00 |
|  7 | Muffy    |   3000 | 2009-10-08 00:00:00 |
|  7 | Muffy    |   1500 | 2009-10-08 00:00:00 |
|  7 | Muffy    |   1560 | 2009-11-20 00:00:00 |
|  7 | Muffy    |   2060 | 2008-05-20 00:00:00 |
+----+----------+--------+---------------------+
```

SUB QUERY

The query appears with another query is called **sub query**. It is also referred as '**INNER'** queries.

Using this sub queries, we can perform the operations such as insertion, deletion, updation, selection on database table. Sub queries always must

be enclosed with in parenthesis '()'. In most of the cases, sub queries appear only in where clause. We are not permitted to use order by clause in sub query whereas, we are permitted to use group by clause in sub query.

Eg: **Emp**

eno	ename	sal	Dno
1	**X**	**1000**	**10**
2	**Y**	**2000**	**10**
3	**Z**	**3000**	**20**
4	**M**	**4000**	**20**

Select * from emp where dno = (select dno from emp where eno=1)

Note

1. **When a query has sub query, sub query will be executed first.**
2. **The result of the sub query is passed to main query as an input to main query.**

RESULT:

EMP

ENO	ENAME	SAL	DNO
1	**X**	**1000**	**10**
2	**Y**	**2000**	**10**
3	**Z**	**3000**	**20**
4	**M**	**4000**	**20**

NOTE:

There is an alternative way for performing operations which are performed using sub queries.

INSERT OPERATION:

Here we insert records to new table using sub queries.

EG:

INSERT INTO EMP1 SELECT * FROM EMP WHERE ENO IN (

SELECT ENO FROM EMP WHERE DNO=10)

NOTE:

In the above query, the table EMP1 is not necessary to create externally. It means the table 'EMP1' will be generated or created automatically by database.

Using the above query, we are passing the traditional method for inserting records into database table.

DELETE OPERATION:

Using the sub query, we can delete database from database table.

EG:

DELETE FROM EMP WHERE ENO IN (SELECT ENO FROM EMP WHERE DNO=10)

UPDATE OPERATION:

We are permitted to update records stored in the database table using sub query.

We can update only one record, set of record, and all records using sub query.

Eg:

UPDATE EMP SET SAL=SAL+3000 WHERE ENO IN (SELECT ENO FROM EMP WHERE DNO=20)

NOTE:

In sql, operators may be defined as mathematical symbols or a collection of character.

TYPES OF SUB QUERIES

Sub queries have the following list of types.

They are,

1. Single row sub queries

2. Multiple row sub queries
3. Multiple column sub queries
4. Nested sub queries
5. Scalar sub queries
    - Correlated
    - Non-correlated

1. SINGLE ROW SUB QUERIES

Those sub queries always return only one value /one row to the outer query or main query.

EG:

SELECT * FROM EMP WHERE ENO = (SELECT ENO FROM EMP WHERE ENAME='X')

2. MULTIPLE ROW SUB QUERIES

Those sub queries will return more than one value to the main query as input.

EG:

SELECT * FROM EMP WHERE ENO IN (SELECT ENO FROM EMP WHERE DNO=10)

3. MULTIPLE COLUMN SUB QUERIES

By default, sub queries return values of one database column, when sub queries return more than one column; we are calling those queries as multiple column sub queries.

EG:

SELECT * FROM EMP WHERE (ENO, SAL) IN (SELECT ENO, MIN (SAL) FROM EMP GROUPBY DNO)

EMP

ENO	ENAME	SAL	DNO
1	X	1000	10
2	Y	2000	20
3	Z	3000	10

4	M	4000	20

RESULT:

EMP

ENO	ENAME	SAL	DNO
1	**X**	**1000**	**10**
2	**Y**	**2000**	**20**

4. NESTED (Multiple) SUB QUERIES

The sub query contains another sub query.

Nested sub query refers, the query written inside another sub query is called nested sub query. Those queries may return values of one column or values of more than one column.

Nested sub query return values to inner sub query as an input.

Nested sub query does not return values to outer sub query/main query.

EG:

SELECT * FROM EMP WHERE ENO= (SELECT ENO FROM EMP WHERE ENAME= (SELECT ENAME FROM EMP WHERE SAL=1000))

SELECT * FROM EMP WHERE ENO IN (SELECT ENO FROM EMP WHERE DNO= (SELECT DNO FROM EMP WHERE ENAME='X'))

Correlated Subqueries

SQL Correlated Subqueries are used to select data from a table referenced in the outer query. The subquery is known as a correlated because the subquery is related to the outer query. In this type of

queries, a table alias (also called a correlation name) must be used to specify which table reference is to be used.

Uncorrelated subquery executes the subquery first and provides the value to the outer query, whereas correlated subquery references a column in the outer query and executes the subquery once for each row in the outer query..

The alias is the pet name of a table which is brought about by putting directly after the table name in the FROM clause. This is suitable when anybody wants to obtain information from two separate tables.

Example:

Order table

OrderNo	Amt	CustNo	AgentNo
1	5000	C001	A001
2	6000	C002	A002
3	7000	C003	A001

Agent table

AgentNo	AgentName
A001	RAM
A002	SEENU

```
SELECT a.ord_num,a.ord_amount,a.cust_code,a.agent_code
FROM orders a WHERE a.agent_code=( SELECT b.agent_code FROM
agents b WHERE b.agent_name='RAM');
```

Result:

OrderNo	Amt	CustNo	AgentNo
1	5000	C001	A001
3	7000	C003	A001