**BASIC OF**

**PL-SQL and SQL**

**The basic structure of an SQL expression consists of three clauses:**

* The **select** clause which corresponds to the projection operation. It is the list of attributes that will appear in the resulting table.
* The **from** clause which corresponds to the Cartesian-product operation. It is the list of tables that will be joined in the resulting table.
* The **where** clause which corresponds to the selection operation. It is the expression that controls the which rows appear in the resulting table.

**Cursor Type:**

4 types:

1. **sql%found**
2. **sql%notfound**
3. **sql %isopen**
4. **sql %rowcount**

**View and Materialized View:**

|  |  |  |
| --- | --- | --- |
| **Basis for**  **Comparison** | **View** | **Materialized View** |
| Basic | A View is never stored it is only displayed. | A Materialized View is stored on the disk. |
| Define | View is the virtual table formed from one or more base tables or views. | Materialized view is a physical copy of the base table. |
| Update | View is updated each time the virtual table (View) is used. | Materialized View has to be updated manually or using triggers. |
| Speed | Slow processing. | Fast processing. |
| Memory usage | View do not require memory space. | Materialized View utilizes memory space. |
| Syntax | Create View V As | Create Materialized View V Build [clause] Refresh [clause] On [Trigger] As |

**Constrain(Oracle):**

**6 types of integrity constraint** are described briefly here and more fully in ["Semantics"](https://docs.oracle.com/cd/B19306_01/server.102/b14200/clauses002.htm#i1002038):

* A **NOT NULL** **constraint** prohibits a database value from being null.
* A **unique** **constraint** prohibits multiple rows from having the same value in the same column or combination of columns but allows some values to be null.
* A **primary key constraint** combines a NOT NULL constraint and a unique constraint in a single declaration. That is, it prohibits multiple rows from having the same value in the same column or combination of columns and prohibits values from being null.
* A **foreign key constraint** requires values in one table to match values in another table.
* A **check constraint** requires a value in the database to comply with a specified condition.
* A REF column by definition references an object in another object type or in a relational table. A **REF constraint** lets you further describe the relationship between the REF column and the object it references.

**Purpose of oracle index:**

An **index** is a performance-tuning method of allowing faster retrieval of records. An **index** creates an entry for each value that appears in the indexed columns. By default, **Oracle** creates B-tree **indexes**.

**Types of Indexes:**

Oracle Database provides several indexing schemes, which provide complementary performance functionality. The indexes can be categorized as follows:

* B-tree indexes

These indexes are the standard index type. They are excellent for primary key and highly-selective indexes. Used as concatenated indexes, B-tree indexes can retrieve data sorted by the indexed columns. B-tree indexes have the following subtypes:

* + Index-organized tables

An index-organized table differs from a heap-organized because the data is itself the index. See ["Overview of Index-Organized Tables"](https://docs.oracle.com/cd/E11882_01/server.112/e40540/indexiot.htm#CBBJEBIH).

* + Reverse key indexes

In this type of index, the bytes of the index key are reversed, for example, 103 is stored as 301. The reversal of bytes spreads out inserts into the index over many blocks. See ["Reverse Key Indexes"](https://docs.oracle.com/cd/E11882_01/server.112/e40540/indexiot.htm#CBBFDEAJ).

* + Descending indexes

This type of index stores data on a particular column or columns in descending order. See ["Ascending and Descending Indexes"](https://docs.oracle.com/cd/E11882_01/server.112/e40540/indexiot.htm#CBBFFFFG).

* + B-tree cluster indexes

This type of index is used to index a table cluster key. Instead of pointing to a row, the key points to the block that contains rows related to the cluster key. See ["Overview of Indexed Clusters"](https://docs.oracle.com/cd/E11882_01/server.112/e40540/tablecls.htm#CFABHBAG).

* Bitmap and bitmap join indexes

In a bitmap index, an index entry uses a bitmap to point to multiple rows. In contrast, a B-tree index entry points to a single row. A bitmap join index is a bitmap index for the join of two or more tables. See ["Bitmap Indexes"](https://docs.oracle.com/cd/E11882_01/server.112/e40540/indexiot.htm#CBBFJFDD).

* Function-based indexes

This type of index includes columns that are either transformed by a function, such as the UPPER function, or included in an expression. B-tree or bitmap indexes can be function-based. See ["Function-Based Indexes"](https://docs.oracle.com/cd/E11882_01/server.112/e40540/indexiot.htm#CBBGIIFB).

* Application domain indexes

This type of index is created by a user for data in an application-specific domain. The physical index need not use a traditional index structure and can be stored either in the Oracle database as tables or externally as a file. See ["Application Domain Indexes"](https://docs.oracle.com/cd/E11882_01/server.112/e40540/indexiot.htm#CBBFEBGI).

**Table Function:**

**Steps**:

* First have to create “Object Type”
* 2nd have to create “Table Type” of the object
* 3rd have to create a Function for return row of table
* Finally must have to call Function

**Now Example on Employee Table:**

* **create “Object Type”:**

CREATE TYPE t\_emp\_total AS OBJECT (

ENAME VARCHAR2(50),

JOB VARCHAR2(50),

MGR NUMBER,

DATE\_OF\_BIRTH date,

SAL NUMBER,

COMM NUMBER,

DEPTNO NUMBER

);

* **create “Table Type” of the object:**

CREATE TYPE t\_table\_emp\_total IS TABLE OF t\_emp\_total;

* **to create a Function for return row of table:**

CREATE OR REPLACE FUNCTION get\_emp\_details (p\_id IN EMP.EMPNO%type)

RETURN t\_table\_emp\_total PIPELINED IS

out\_rec t\_emp\_total := t\_emp\_total(null, null, null, null, null, null, null);

BEGIN

FOR i in (select ENAME, JOB, MGR, DATE\_OF\_BIRTH, SAL, COMM, DEPTNO from emp where EMPNO = p\_id) LOOP

out\_rec.ENAME := i.ENAME;

out\_rec.JOB := i.JOB;

out\_rec.MGR := i.MGR;

out\_rec.DATE\_OF\_BIRTH := i.DATE\_OF\_BIRTH;

out\_rec.SAL := i.SAL;

out\_rec.COMM := i.COMM;

out\_rec.DEPTNO := i.DEPTNO;

pipe row(out\_rec);

END LOOP;

RETURN;

END;

* **call Function:**

SELECT \*

FROM TABLE(get\_emp\_details(7839)) #here have to pass EMPNO(ID)

**For more info about Table Function:**

* <https://livesql.oracle.com/apex/livesql/file/content_C87XCH8SE085LMS3C5KR03VFS.html>
* <https://docs.oracle.com/cd/B19306_01/appdev.102/b14289/dcitblfns.htm>
* ..
* <http://stevenfeuersteinonplsql.blogspot.com/2015/04/table-functions-introduction-and.html>

**NVL():**

select NVL(:e1, :e2)

from dual

NB: If e1 != Null answer=e1 AND If e1 = Null answer=e2

**NVL2():**

select NVL2(:e1, :e2, :e3)

from dual

NB: If e1 != Null answer=e2 AND If e1 = Null answer=e3

**COALESCE():**

The [COALESCE()](https://www.oracletutorial.com/oracle-comparison-functions/oracle-coalesce/) function evaluates its argument in order and stops evaluation when it can determine the result i.e., when it can find the first non-NULL argument. This feature is known as short-circuit evaluation.

select COALESCE(:e1, :e2, :e3)

from dual

NB: It will find 1st not null value

**Decode():**

select JOB\_ID as ID,

DECODE(JOB\_TITLE ,'President','CEO','Administration Vice President','Vice President','\*','\*')

JOB\_TITLE,

MAX\_SALARY as Salary

from HR.JOBS

**LISTAGG():**

SELECT LISTAGG(FIRST\_NAME||' '||LAST\_NAME, ', ') WITHIN GROUP (ORDER BY EMPLOYEE\_ID) AS "Employee List"

FROM HR.EMPLOYEES;

**OVER (PARTITION BY Column\_name):**

SELECT EMPLOYEE\_ID as ID, FIRST\_NAME||' '||LAST\_NAME as Name, DEPARTMENT\_ID, COUNT(\*)

OVER (PARTITION BY DEPARTMENT\_ID) AS DEPT\_COUNT

FROM HR.EMPLOYEES

ORDER BY EMPLOYEE\_ID

[**MONTHS\_BETWEEN**](https://www.oracletutorial.com/oracle-date-functions/oracle-months_between/)**():**

SELECT MONTHS\_BETWEEN( DATE '2017-07-01', DATE '2017-01-01' )

FROM dual

[**NEXT\_DAY**](https://www.oracletutorial.com/oracle-date-functions/oracle-next_day/)**():**

SELECT NEXT\_DAY( DATE'2020-01-01', 'THURSDAY' )

FROM dual

[**LAST\_DAY**](https://www.oracletutorial.com/oracle-date-functions/oracle-last_day/)**():**

SELECT LAST\_DAY(DATE '2020-02-01')

FROM dual

[**ADD\_MONTHS**](https://www.oracletutorial.com/oracle-date-functions/oracle-add_months/)**():**

SELECT ADD\_MONTHS( DATE '2016-02-29', 1 )

FROM dual

**For More Info of Date Function:**

<https://www.oracletutorial.com/oracle-date-functions/>

**Date Functions:**

**SYSDATE:**

SELECT SYSDATE

FROM dual

**SYSTIMESTAMP:**

SELECT SYSTIMESTAMP

FROM dual

[**TO\_DATE**](https://www.oracletutorial.com/oracle-date-functions/oracle-to_date/)**():**

SELECT TO\_DATE('01/Jan/2017', 'DD/MON/YYYY' )

FROM dual

[**TO\_CHAR**](https://www.oracletutorial.com/oracle-date-functions/oracle-to_char/)**():**

SELECT TO\_CHAR(DATE'2017-01-01', 'DL') *--, 'DL' = is a Date Format*

FROM dual

**TO\_NUMBER():**

select TO\_NUMBER('1210.73') *--, '9999.99'*

from dual

**First Date of a Month:**

select trunc(sysdate,'MM') as FIRSTDATE

from dual;

**NB: TRUNC(DATE'2017-07-16', 'MM') 🡺 is used for Specific month**

select TRUNC(DATE '2017-07-16', 'MM') as FIRSTDATE

from dual;

**Last Date of a Month:**

select last\_day(sysdate) as LASTDATE

from dual;

**NB: TRUNC(DATE'2017-07-16', 'MM') 🡺 is used for Specific month**

select last\_day(to\_date('21/06/2019', 'dd/mm/yyyy')) as LASTDATE

from dual;

**First Date of a Previous Month:**

select last\_day(add\_months(sysdate,-2))+1 as FIRSTDATE

from dual;

**Last Date of a Previous Month:**

select last\_day(add\_months(sysdate,-1))as LASTDATE

from dual;

**First Day of a Month:**

select to\_char(trunc(SYSDATE,'MM'),'DAY') as FIRSTDAY

from dual;

**Last Day of a Month:**

select to\_char(last\_day(sysdate),'DAY') as LASTDAY

from dual;

**First Day of a Previous Month:**

select to\_char(last\_day(add\_months(sysdate,-2))+1, 'DAY') as FIRSTDAY

from dual;

**Last Day of a Previous Month:**

select to\_char(last\_day(add\_months(sysdate,-1)), 'DAY') as LASTDAY

from dual;

**Show whole Date and Day of a Month:**

select trunc(sysdate, 'MONTH') + rownum - 1 as DATES,

TO\_CHAR(trunc(sysdate, 'MONTH') + rownum - 1, 'DAY') as day

from dual

connect by rownum <= to\_number(to\_char(last\_day(sysdate), 'DD'));

**Show whole Dates of a Month:**

SELECT trunc(sysdate + level) -1 as dates

from dual

connect by level < (LAST\_DAY(sysdate) - sysdate) + 2

**Number of FRIDAY between the range:**

with t

as

(

select sysdate-31 start\_date, sysdate end\_date

from dual

)

select count(\*)

from (select to\_char(start\_date + (level-1), 'fmday') dt

from t

connect by level <= end\_date-start\_date+1)

where dt in ('friday') *--,'saturday'*

**Collecting IP Address from V\_$Session:**

SELECT username,

UTL\_INADDR.GET\_HOST\_ADDRESS(REGEXP\_REPLACE(machine, '^.+\\')) AS client\_ip,

status,

osuser,

process,

machine,

terminal,

logon\_time,

lockwait,

blocking\_session\_status,

blocking\_instance,

blocking\_session

FROM v$session;

**AGGRIGATE Functions:**

**Max():**

select max(salary)

from hr.employees

**Min():**

select min(salary)

from hr.employees

**AVG():**

select avg(salary)

from hr.employees

**count():**

select count(EMPLOYEE\_ID)

from hr.employees

**SUM():**

select sum(salary)

from hr.employees

**For more info about Aggrigate Functions:**

* <https://www.oracletutorial.com/oracle-aggregate-functions/>

**String Operation:**

**SUBSTR**():

SELECT SUBSTR( 'Oracle Substring', 1, 6 ) as SUBSTRING

FROM dual;

**INSTR() for find the string position:**

SELECT INSTR( 'This is a playlist', 'is' ) as substring\_location

FROM dual;

**CONCAT():**

SELECT CONCAT( CONCAT( 'Happy', ' coding' ), ' together' )

FROM dual;

**REPLACE():**

SELECT REPLACE( 'This is a test', 'is', 'IS' ) as replace\_word

FROM dual;

**LPAD():**

select LPAD( 'ABC', 5, '\*' )

from dual;

**RPAD():**

select RPAD( 'XYZ', 6, '+' )

from dual;

**TRIM():**

SELECT TRIM( ' ABC ' )

FROM dual;

**LOWER():**

SELECT LOWER('SHIFULLAH')

FROM dual;

**UPPER():**

SELECT UPPER( 'string functions' )

FROM dual;

**INITCAP():**

SELECT INITCAP( 'hi shawon' )

FROM dual;

**For more info about String Function:**

* <https://www.oracletutorial.com/oracle-string-functions/>

**JOIN Query:**

* [A visual explanation of Oracle Joins](https://www.oracletutorial.com/oracle-basics/oracle-joins/) – a brief introduction to joins in Oracle using visual illustrations.
* [INNER JOIN](https://www.oracletutorial.com/oracle-basics/oracle-inner-join/) – show you how to query rows from a table that have matching rows from another table.
* [LEFT JOIN](https://www.oracletutorial.com/oracle-basics/oracle-left-join/) – introduce you to the left-join concept and learn how to use it to select rows from the left table that have or don’t have the matching rows in right table.
* [RIGHT JOIN](https://www.oracletutorial.com/oracle-basics/oracle-right-join/) – explain the right-join concept and show you how to apply it to query rows from the right table that have or don’t have the matching rows in the left table.
* [CROSS JOIN](https://www.oracletutorial.com/oracle-basics/oracle-cross-join/) – cover how to make a Cartesian product from multiple tables.
* [Self-join](https://www.oracletutorial.com/oracle-basics/oracle-self-join/) – show you how to join a table to itself to query hierarchical data or compare rows within the same table.

**INNER JOIN:**

SELECT \*

FROM EMP

INNER JOIN DEPT ON DEPT.DEPTNO = EMP.DEPTNO

ORDER BY EMPNO;

**LEFT JOIN:**

SELECT \*

FROM EMP

LEFT JOIN DEPT ON DEPT.DEPTNO = EMP.DEPTNO

ORDER BY EMPNO;

SELECT \*

FROM EMP

LEFT OUTER JOIN DEPT ON DEPT.DEPTNO = EMP.DEPTNO

ORDER BY EMPNO;

**RIGHT JOIN:**

SELECT \*

FROM EMP

RIGHT JOIN DEPT ON DEPT.DEPTNO = EMP.DEPTNO

ORDER BY EMPNO;

SELECT \*

FROM EMP

RIGHT OUTER JOIN DEPT ON DEPT.DEPTNO = EMP.DEPTNO

ORDER BY EMPNO;

**CROSS JOIN:**

SELECT \*

FROM EMP

CROSS JOIN DEPT

ORDER BY EMPNO;

**SELF JOIN:**

SELECT (a.ENAME||' '||a.JOB) as Employee, (b.ENAME||' '||b.MGR) as Manager

FROM EMP a

LEFT JOIN EMP b ON a.EMPNO = b.MGR

**Number Functions:**

**ROUND**():

SELECT ROUND(-4.53)

FROM dual;

**MOD():**

SELECT MOD(7,4)

FROM dual;

**TRUNC():**

#Truncate from Right side of decimal point

SELECT TRUNC(2.465,1)

FROM dual;

#Truncate from Left side of decimal point

SELECT TRUNC(142.465,-2)

FROM dual;

**Nested\_Query:**

SELECT sal as salary

FROM emp

where sal > (select max(sal) from emp where DEPTNO=30);

**DML(INSERT, UPDATE, DELETE):**

**update\_table:**

update shifullah.MY\_CUSTOMER

set salary = 25000

where id = 2;

commit WORK;

**insert\_table:**

INSERT INTO categories (category\_id, category\_name)

VALUES (150, 'Miscellaneous');

**Delete\_table:**

DELETE

FROM sales

WHERE order\_id = 1 AND item\_id = 1;

**DDL(All except DML. Example CREATE, TRUNCATE etc.):**

**Create\_table:**

CREATE TABLE ot.persons(

person\_id NUMBER GENERATED BY DEFAULT AS IDENTITY,

first\_name VARCHAR2(50) NOT NULL,

last\_name VARCHAR2(50) NOT NULL,

PRIMARY KEY(person\_id)

);