INDEX, Transactions & In-Built Functions

INDEXES

- Indexes are optional structures associated with tables.
- There are different types of indexes including those used to enforce primary key constraints, unique indexes, non-unique indexes, concatenated indexes, and others.

PRIMARY KEY indexes

- When a PRIMARY KEY constraint is specified, Oracle will automatically create a unique index to support rapid data retrieval for the specified table.
- Without an index, a command that retrieves data will cause Oracle to completely scan a table for rows that satisfy the retrieval condition.

```
select rowid,empno from emp;
```

Creating an Index

- The general form of the CREATE INDEX command is:

CREATE INDEX <index name>
ON <table name> (column1, column2…);

INDEXES

- Use indexes to speed up queries. Indexes speeds up searching of information in tables. So create indexes on those columns which are frequently used in WHERE conditions.

Creating Indexes

To create an Index give the create index command. For example the following statement creates an index on empno column of emp table.

Example:

To create an Index give the create index command. For example the following statement creates an index on empno column of emp table.

create index empno_ind on emp (empno);

INDEXES

- If two columns are frequently used together in WHERE conditions then create a composite index on these columns. For example, suppose we use EMPNO and DEPTNO often together in WHERE condition. Then create a composite index on these column as given below
  - create index empdept_ind on emp (empno,deptno);

Creating a UNIQUE Index

- The general form of the CREATE UNIQUE INDEX command is:

CREATE UNIQUE INDEX <index name>
ON <table name> (column1, column2…);

DROPPING Indexes

- The syntax of the DROP INDEX command is simple:

```
DROP INDEX index_name;
```

Information on Indexes

- select index_name,index_type,table_name from user_indexes where table_name='EMP';

- select index_name,table_name,column_name from user_ind_columns where table_name='EMP';

Transactions

- Unit of work.
- Atomic transaction
  - either fully executed or rolled back as if it never occurred
- Isolation from concurrent transactions
- Transactions begin implicitly
  - Ended by **commit work** or **rollback work**
- TCL Statements available in Oracle are
- **COMMIT** : Make changes done in transaction permanent.

  ROLLBACK : Rollbacks the state of database to the last commit point.

  SAVEPOINT : Use to specify a point in transaction to which later you can rollback.

Transactions -Example

- **SQL>insert into emp (empno,ename,sal) values (109,'Sami',3000);**

  SQL>savepoint a;

  SQL>insert into dept values (10,'Sales','Hyd');

  SQL>savepoint b;

  SQL>insert into salgrade values ('lll',9000,12000);

- Now if you give

  SQL>rollback to a;

- Then row from salgrade table and dept will be roll backed. At this point you can commit the row inserted into emp table or rollback the transaction.

Transactions -Example

- If you give

  SQL>rollback to b;

- Then row inserted into salgrade table will be roll backed. At this point you can commit the row inserted into dept table and emp table or rollback to savepoint a or completely roll backed the transaction.
- If you give

  SQL>rollback;

- Then the whole transactions is roll backed.
- If you give

  SQL>commit;

- Then the whole transaction is committed and all savepoints are removed.

Rownum, rowid

- In sum, the **difference between ROWNUM and ROWID** is that **ROWNUM** is temporary while **ROWID** is permanent. Another **difference** is that **ROWID** can be used to fetch a row, while **ROWNUM** only has meaning within the context of a single SQL statement, a way of referencing rows within a fetched result set.

SQL>select rownum,rowid,empno from scott.emp;

ROWNUM	ROWID	EMPNO
1	AAAR3sAAEAAAACUAAA	50
2	AAAR3sAAEAAAACVAAD	109
3	AAAR3sAAEAAAACWAAA	1000
4	AAAR3sAAEAAAACWAAB	1001
5	AAAR3sAAEAAAACVAAA	1212
6	AAAR3sAAEAAAACVAAB	1213
7	AAAR3sAAEAAAACXAAO	2000

SQL> select rownum,rowid,empno from scott.emp where empno>1000;

ROWNUM	ROWID	EMPNO
1	AAAR3sAAEAAAACWAAB	1001
2	AAAR3sAAEAAAACVAAA	1212
3	AAAR3sAAEAAAACVAAB	1213
4	AAAR3sAAEAAAACXAAO	2000
5	AAAR3sAAEAAAACXAAA	7369
6	AAAR3sAAEAAAACXAAB	7499
7	AAAR3sAAEAAAACXAAC	7521

Character Functions

- **LOWER**

  Converts all characters to lower case

  Syntax : LOWER(argument)

  Example

  SELECT LOWER('VINAYAK') FROM DUAL; returns vinayak

- **UPPER**

  Converts all characters to upper case

  Syntax : UPPER(argument)

  Example

  SELECT UPPER('java') FROM DUAL; returns JAVA

- **INITCAP**

  Forces the first letter of each word to be in upper case

  Syntax :INITCAP(argument)

  Example

  SELECT INITCAP('manipal university') FROM DUAL; returns Manipal University

Character Functions

- **LPAD**

Pads string to the left with a specified character until a specified length is reached

Syntax : LPAD(string,padstrlen,padstring)

string -the string to be padded

padstrlen -the length of the final string after padding

padstring -the string to use for padding

Example:

SELECT LPAD('Two Thousands',18,'*') from dual; returns ********Two Thousands**

Similarly RPAD works.

- **LENGTH**

The length function returns the number of characters in a string.

Syntax : LENGTH(string)

string -The string you want the length of.

Example: SELECT LENGTH('Computer') From Dual; Returns 8

Character Functions

- **SUBSTR**

  The SUBSTR function is used to extract a portion of a string.

  Syntax : SUBSTR(string,stratpos,no_of_char)

  string -the string to be extracted from.

  Stratpos- starting position from which to extract characters.

  no_of_char- number of characters to be extracted from startpos.

  Example

  SELECT SUBSTR('ComPuter',3,5) from Dual; Returns **mPuter**

Numeric Functions

- **TRUNC**

  The TRUNC function truncates a number to a specified number of decimal places.

  Syntax : TRUNC(number,n)

  Example: SELECT TRUNC(3.142,1)from dual; Returns 3.1 value.

- **POWER** 29

  Syntax: POWER(m,n) Returns m^n value.

  Example: SELECT POWER (5,3) FROM DUAL; Returns 125.

- **ABS**

  Syntax: ABS(m) Returns absolute value of m.

  Example: SELECT ABS (-123) FROM DUAL; Returns 123.

- **SQRT**

  Syntax: SQRT(m) Returns square root of m.

  Example: SELECT SQRT (9) FROM DUAL; Returns 3.

- **MOD**

  Syntax: MOD(m,n) Returns remainder after dividing m by n.

  Example: SELECT MOD(27,7)FROM DUAL; Returns 6.

Conversion Functions

- **TO_CHAR**

  The TO_CHAR function is used convert a value into a char, with or without a specified format.

  Syntax :

  TO_CHAR(number)

  TO_CHAR(number,format)

  TO_CHAR(date)

  TO_CHAR(date,format)

  Example:

  SELECT TO_CHAR(123) FROM DUAL; Returns **123** which will be a character value.

  SELECT TO_CHAR (123,'999.99') FROM DUAL; Returns **123.00** as character value.

- **$** or **999.99** are said to mask formats

  SELECT 'Salary='||TO_CHAR(sal,'$9999.99') FROM EMP;

  Display salary values with **$** symbol prefixed and two leading zeros after decimal point.

  || is used to concatenate strings or strings with data returned by some column.

- **select ename||'Draws Rs.'||sal||'Salary' From emp;**

Conversion Functions

- **TO_NUMBER**

  The TO_NUMBER function is used convert a char into a number.

  Syntax : TO_NUMBER(string)

  Example : SELECT TO_NUMBER ('123') FROM DUAL; Returns **123** which will be a NUMERIC value.

- **TO_DATE**
- The TO_DATE function is used convert a char into a date with default date format or any other format we wish given as format mask.
- Syntax : **TO_DATE(string,format)**

  string The string to be converted

  format The date format mask you wish to apply to the input string.

  This ensures that the string is in a correct date format .
- **insert into emp (empno, ename,hiredate) values(002,'Ramanujan',to_date('10-07-2019','DD-MM-YYYY'));**
- **insert into emp(empno, hiredate)values(1999, to_date('1998/05/31:12:00:00AM', 'yyyy/mm/dd:hh:mi:ssam'));**
- If the **default format set is DD-MON-YYYY** then we can enter date as Character data without using TO_DATE.
- ALTER SESSION SET **NLS_DATE_FORMAT = 'YYYY-MM-DD HH24:MI:SS'**; can be used to set default format for the current session.

Conversion Functions

- **TO_CHAR**
- The TO_CHAR function is used convert a date into a character value with default date format or any other format we wish given as format mask.
- Syntax : **TO_CHAR(Date, format)**

  Date The date to be converted

  format The date format mask you wish to apply to the Date value.
- SELECT **TO_CHAR(HIREDATE)** FROM EMP; DEFAULT FORMAT APPLIED.
- SELECT **TO_CHAR(HIREDATE, 'MONTH-YYYY-DD')** FROM EMP;
- SELECT **TO_CHAR(HIREDATE, 'MON-YY')** FROM EMP;
- SELECT **TO_CHAR(HIREDATE, 'YYYY')** FROM EMP; returns 4 digits of year
- SELECT **TO_CHAR (HIREDATE,'MONTH')** from EMP; returns Month name
- SELECT **TO_CHAR (HIREDATE, 'DAY')** from EMP; returns day name –such as Saturday,..
- SELECT **TO_CHAR(HIREDATE, 'DD-MON-YYYY HH24:MI:SS')** AS "CURRENT DATE TIME" FROM EMP;
- SELECT SYSDATE FROM DUAL; returns current system date.
- Select to_char(sysdate,'HH24:MI:SS AM') from dual; returns only time information

Date function

EXTRACT

- The **EXTRACT** function can be used to retrieve required components (Day, month or year) of the Date .
- **Syntax:**

EXTRACT (YEAR | MONTH | DAY FROM *datetime_value_expression*)

Examples:

SELECT EXTRACT (YEAR FROM TO_DATE ('10-JUL-1999','dd-MON-yyyy')) from dual; Returns 1999

SELECT EXTRACT (YEAR FROM HIREDATE) FROM Emp WHERE Empno=101;

SELECT EXTRACT (MONTH FROM HIREDATE) FROM Emp WHERE Empno=101;

SELECT MONTHS_BETWEEN(SYSDATE,HIREDATE) FROM EMP WHERE EMPNO=7900;

SELECT ADD_MONTHS(SYSDATE,1) FROM DUAL;

Operations on DATE

DATE values can be Compared using the standard comparison operators such as **=, !=, >,** *etc.*

Two **DATE** values can be subtracted, and the result is a **FLOAT** which is the **number of days between** the two DATE values.

For obvious reasons, adding, **multiplying**, and **dividing** two DATE values are **not allowed**.

A numeric constant can be added or subtracted from a DATE value, and these numbers will be interpreted as numbers of days.

For example, SYSDATE+1 will be tomorrow.

With the help of TO_CHAR, string operations can be used on DATE values as well.

For example, to_char(<date>, 'DD-MON-YY') like '%JUN%' evaluates to true if <date> is in June.

END