**HIERARCHICAL Queries**

* These are queries that are executed upon tables that contain hierarchical data.
* To Execute the hierarchical queries, we need the following queries
* START WITH: It specifies the root rows of the hierarchy.
* CONNECT BY: It is used to specify the relationship between parent rows and child rows of the hierarchy.
* WHERE: It is used to restrict the rows returned by the Query without affecting other row of the hierarchy.

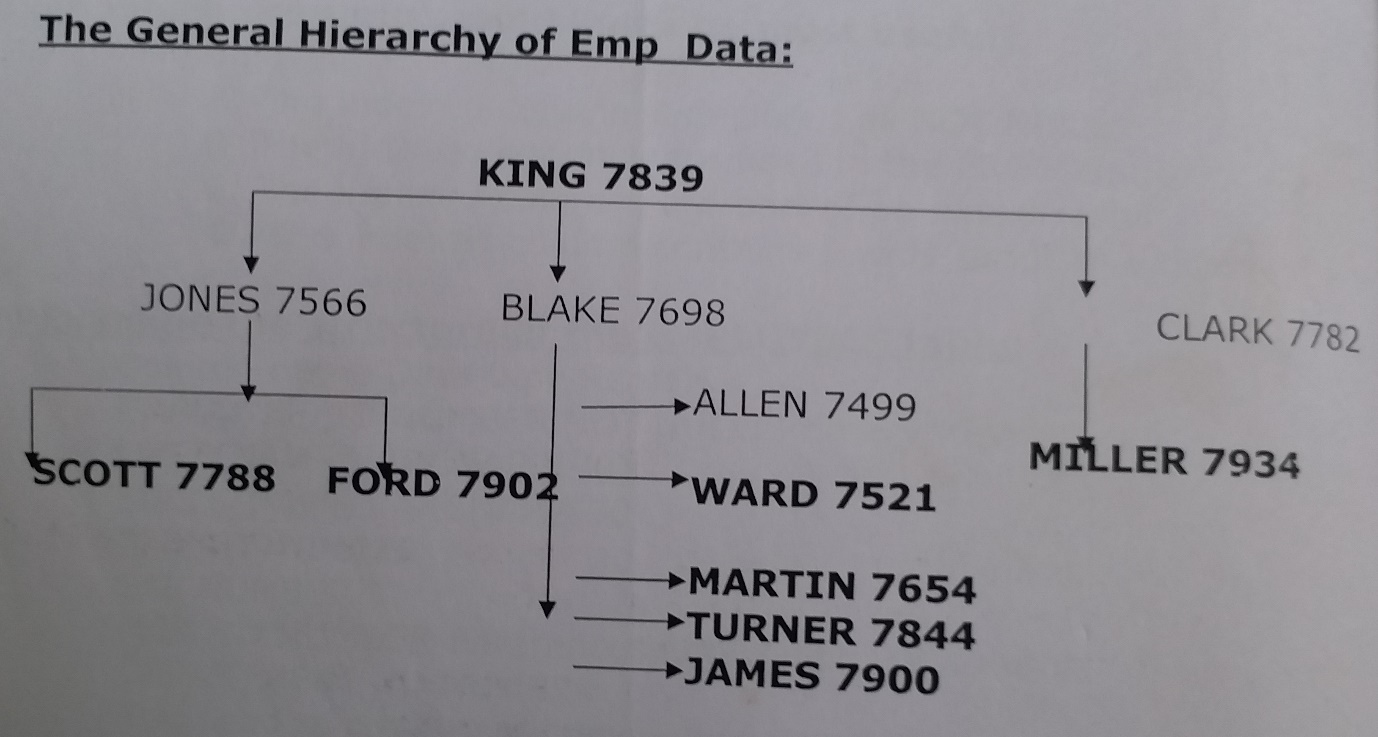
**Restriction:**

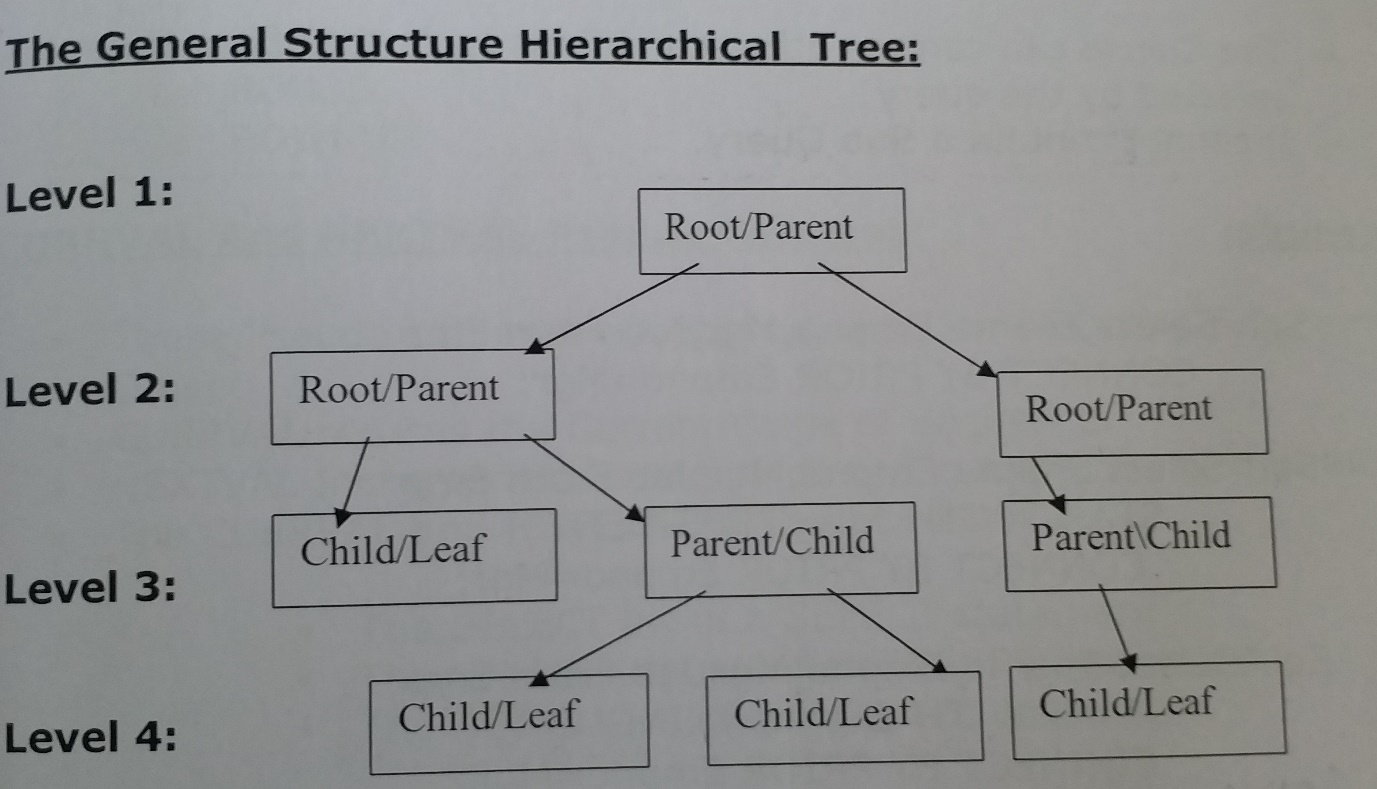
• They cannot be used to perform joins.

• They cannot select data from a View, Whose Query performs a Join

• If ORDER BY Clause is used, then the rows are returned as per the specification in the ORDER BY Clause.

**The General Hierarchy of Emp Data:**

****

****

**Note:** To define hierarchical queries properly we must use the following clauses.

* Start with
* Connect by

**START WITH Clause:**

• It identifies the row(s) to be used as the Root(s) of a hierarchical Query.

• It specifies a condition that the roots must specify.

• If START WITH is omitted, Oracle uses all rows in the table as ROOT

rows.

• A START WITH Condition can contain a Sub Query.

**CONNECT By Clause:**

* This clause specifies the Relationship between Parent and Child Rows in a hierarchical query.
* This clause contains a condition that defines a relationship.
* This condition can be any condition as defined by the syntax description.
* Within the condition, some part of the condition must use the PRIOR operator, which refers to the Parent row.
* The forms of PRIOR operators are
* PRIOR Expr Comparison -operator Expr.
* Expr Comparison - operator Expr.
* The Clause can contain other conditions to further filter the rows selected by the query.
* It cannot contain a Sub Query.

**Example:**

Sql>Select Ename,Empno,Mgr,Job From EmP CONNECT BY PRIOR Empno = Mgr;

Sql>Select Ename,Empno,Mgr,Job From Emp

START WITH Job = 'PRESIDENT' CONNECT BY PRIOR Empno = Mgr;

Sql>Select Ename,Empno,Mgr,Job From Emp START WITH Ename = 'KING' CONNECT BY PRIOR Empno=Mgr;

Sql>Select Ename,Empno,Mgr,Job From Emp START WITH Sal = 5000

CONNECT BY PRIOR Empno=Mgr;

**SYS\_CONNECT\_BY\_PATH Function:/**

* The function returns the path of a column value from Root to Node, with column values separated by 'Char' for each row returned by CONNECT BY condition.
* Can work on any datatype CHAR,VARCHAR2,NCHAR, or NVARCHAR2.

Sql> select Ename, SYS\_CONNECT\_BY\_PATH(Ename,'/') "Path"

From Emp Start With Ename ='KING' Connect By Prior Empno=MGR;

**PSEUDO Columns:**

• Pseudo column behave like a table column, but is not actual stored in a table.

• Upon Pseudo columns only SELECT’s can implemented, but INSERT, UPDATE or DELETE cannot be implemented.

• The available Pseudo Columns are….

* CURRVAL
* NEXTVAL
* LEVEL
* ROWID
* ROWNUM

**CURRVAL And NEXTVAL Pseudo Columns:**

* These Pseudo columns are applied upon the SEQUENCE Schema Object.
* CURRVAL returns the Current Value of Sequence
* NEXTVAL Increments the sequence and returns the Next Value.
* The CURRVAL and NEXTVAL can be used only in...
* The SELECT list of a SELECT statement
* The VALUES clause of an INSERT Statement
* The SET Clause of an UPDATE Statement.

**Restrictions:**

* The CURRVAL and NEXTVAL cannot be used in……….
* A Sub Query
* A View's Query or SNAPSHOT'S Query.
* A SELECT Statement with a GROUP BY or ORDER BY Clause.
* A SELECT Statement with the DISTINCT operator.
* A SELECT Statement that is combined with another SELECT statement with UNION,INERSECT,MINUS SET Operator.
* The WHERE Clause of a SELECT Statement.
* The DEFAULT Value of Column in a CREATE TABLE or ALTER TABLE Statement.
* The Condition of a CHECK Constraint.
* In a single SELECT Statement, all referenced sequences, LONG Columns, Updated tables, and locked tables, must be located on the same database.
* The First reference to the NEXTVAL return the SEQUENCES Initial Value .
* Before the CURRVAL can be used for a SEQUENCE in a session, first the SEQUENCE should be incremented with NEXTVAL.
* A SEQUENCE can be incremented only in a single SQL.
* A SEQUENCE can be accessed by many users concurrently with no WAITING or LOCKING.
* CURRVAL and NEXTVAL should be qualified with the name of the Sequence.

**Syntax:**

SEQUENCENAME.CURRVAL🡪Returns the Current Value of the Sequence.

SEQUENCENAME.NEXTVAL🡪Increments the Sequence Value by the Declare Specification.

**SEQUENCE:**

* A SEQUENCE is a schema object that can generate unique sequential values.
* Sequence is an independent database object.
* The SEQUENCE Values are often user for PRIMARY KEY'S and UNIQUE KEY'S.
* To refer to the Current or Next Value of a SEQUENCE in the Schema of another used. The following privileges be available.
* SELECT OBJECT PRIVILAGE
* SELECT ANY SEQUENCE
* For SEQUENCES in other Schema the Qualifying Syntax is
* SCHEMANAME.SEQUENCENAME.CURRVAL
* SCHEMANAME.SEQUENCENAME.NEXTVAL
* To refer to the value of a SEQUENCE on a Remote Database, The SEQUENCE should be qualified with a complete or partial name of the Database Link.
* SCHEMANAME.SEQUENCENAME.CURRVAL@DBLINK
* SCHEMANAMF.SEOUENCENAME.NEXTVAL@DBLINK

**Syntax:**

CREATE SEQUENCE sequenceName

[INCREMENT BY n]

[START WITH n]

[MAXVALUE n | NOMAXVALUE]

[MINVALUE n | NOMINVALUE]

[CYCLE | NOCYCLE]

[CACHE n I NOCACHE]

ORDER/NOORDER;

**INCREMENT BY:**

* Specifies the interval between the Sequence Numbers.
* Value can be Positive or Negative, but cannot be 0.
* If the value is positive it is Incremental Sequence else it Decremental Sequence.

**MINVALUE:**

• Specifies the Sequence's Minimum Value.

**NOMINVALE:**

• Specifies a minimum value of 1 for an ascending sequence and

10^26 for a descending sequence .

**MAXVALUE:**

* Specifies the Maximum value that sequence can generate.

**NOMAXVAULE:**

* Specifies a maximum value of 10^27 for an ascending sequence and -1 for a descending sequence.

**START WITH:**

* + Specifies the first sequence number to be generated.
  + For Ascending sequences the default value is SEQUENCES's MINMUM value.

**CYCLE:**

* Specifies whether the sequence continues to generate values after reaching its maximum or minimum value.

**NOCYCLE:**

* Specifies the SEQUENCE cannot generate more values after the targeted limit.

**CACHE:**

* Specifies how many values the Oracle Server pre-allocates and keep in memory.

**NOCACHE:**

* Specifies the values of a SEQUENCE are not Pre-allocated.
* If the above parameters are not specified by default 20 values are cached.

**ORDER:**

* Guarantee the sequence numbers to be generated in the order of request.

**NOOEDER:**

* Does not guarantee the sequence Number with Order.

**Note:**

* If the Above parameters are not specified by default
* START WITH Will be 1.
* INCREMENT BY will be positive 1.
* SEQUENCE is NOCYCLE.
* The CACHE Value Will be 20
* SEQUENCE is ORDER

**Test Table:**

Sql>CREATE TABLE TDEPT

(

DEPTNO NUMBER(4) Constraint Tdeptno\_PK Primary Key,

DNAME VARCHAR2(14),

LOC VARCHAR2(13)

);

**Creation of Incremental Sequence:**

Sql> CREATE SEQUENCE TDept\_DeptNo\_Seq

INCREMENT BY 10

START WITH 10

MINVALUE 0

MAXVALUE 9999

NOCACHE

NOCYCLE;

• In this sequence named **TDept\_DeptNo\_Seq** to be used for the Deptno column of the Dept table.

• The sequence starts at 10, does not allow caching, and does not cycle.

Sql> Insert into TDept

Values(TDept\_DeptNo\_Seq.NEXTVAL,'SOFTWARE','HYD');

**Creating A Sequence with CYCLE:**

Sql>CREATE SEQUENCE TDept\_\_DeptNo\_Seq

INCREMENT BY 10

START WITH 10

MINVALUE 0

MAXVALUE 9999

NOCACHE

CYCLE;

**Creation Of Decremental Sequence:**

Sql>CREATE SEQUENCE TDept\_DeptNo\_ Seq

INCREMENT BY -1

START WITH 10

MINVALUE 0

MAXVALUE 10

NOCACHE

NOCYCLE;

**Modifying a Sequence:**

* The ALTER command can be used to change the present status of a SEQUENCE.
* The ALTER SEQUENCE command can be used to change….
* Increment Value
* Maximum Value
* Minimum Value
* Cycle Option
* Cache Option

**Syntax:**

>ALTER SEQUENCE sequence

[INCREMENT BY n]

[{MAXVALUE n | N0MAXVALUE}]

[{MINVALUE n | NOMINVALUE}]

[{CYCLE | NOCYCLE}]

[{CACHE n | NOCACHE}];

**Example:**

Sql>Alter SEQUENCE TDept\_DeptNo\_Seq

INCREMENT BY 10

MAXVALUE 500

NOCACHE

NOCYCLE;

**Guidelines for Modifying a sequence:**

• You must be the owner or have the ALTER privilege for the sequence modify it.

• Only future sequence numbers are affected by the ALTER SEQUENCE statement.

• The START WITH option cannot be changed using ALTER SEQUENCE.

• The sequence must be dropped and re-created in order to restart the sequence at different number.

• Some validation is performed. For example, a new MAXVALUE that is less than current sequence number cannot be imposed.

**Viewing the Current Value of a Sequence**

Sql> Select TDept\_DeptNo\_Seq.Currval from Dual;

**Removing a Sequence**

\* Remove a sequence from the data dictionary by using the DROP

SEQUENCE statement.

\* Once removed, the sequence can no longer be referenced.

Sql>DROP SEQUENCE TDept\_DeptNo\_Seq;

**Remember:**

* The owner of the sequence or have the DROP ANY SEQUENCE privilege to remove it.
* Once removed, the sequence can no longer be referenced.
* The Data Dictionary in which the information of SEQUENCES are stored is USER\_OBJECTS.

Sql>Select Object\_Name From User\_Objects where Object\_Name Like'TDEPT\_DEPTNO\_SEQ';

* The Setting of the SEQUENCE can be confirmed by Selecting on

USER\_SEQUENCES catalog.

* Sql>Select Sequence\_Name,Min\_Value,Max\_Value,Increment\_By, Last\_Number From User\_Sequences Where Sequence\_Name=TDEPT\_DEPTNO\_SEQ';

**Level Pseudo Column:**

The LEVEL Pseudo column returns 1 for a root row, 2 for a child of a root, and so on.

• Child🡪Any non-root node.

• Root🡪Highest Node within an Inverted tree.

• Parent🡪Any Node/Row that has Children.

• Leaf🡪any node without children.

• To establish the Hierarchical relationship with LEVEL we need.

* START WITH Clause.
* CONNECT BY Clause.

Sql>Select Ename,Job,MGR,Level From Emp;

Sql> Select level,Ename,Empno,Mgr,Job From Emp

Start with Job='PRESIDENT'

Connect By Prior Empno = Mgr

order by level;

Sql> Select Lpad('',2\*(level-1))||Ename Org\_Level,Empno,Mgr, Job From Emp

Start with Job ='PRESIDENT'

Connect By Prior Empno = Mgr;

Sql> Select Lpad(' ',2\*(level-1)) ||Ename Org\_Level,Empno,Mgr, Job From Emp Where Job<>'ANALYST'

Start with Job='PRESIDENT'

Connect By Prior Empno=Mgr;

Sql> Select Lpad(' ',2\*(level- 1)) ||Ename Org\_Level,Empno,Mgr,Job From Emp Start with Job='PRESIDENT' Connect By Prior Empno=Mgr and Level<=2;

**Select Nth Highest Value from table:**

**Syntax:**

Sql>Select Level,Max(ColName)

From TableName

Where Level=&Levelno

Connect By Prior Colname>Colname

Group by Level;

**Example:**

Sql>Select Level,Max(Sal)

From Emp

Where Level=&Levelno

Connect By Prior Sal>Sal

Group By Level;

**Select Nth Lowest Value From Table:**

**Syntax**

Sql>Select Level,Min(ColName)

From TableName

Where Level=&Levelno

Connect By Prior Colname<Colname

Group by Level;

**Example:**

Sql>Select Level,Min(Sal)

From Emp

Where Level=&Levelno

Connect By Prior Sal<Sal

Group By Level;

**Display top 5 salaries:**

Sql> Select max(Sal) from emp

Where level<=5

Connect by prior sal>sal

Group by level;

**Display the least 5 salaries:**

Sql> Select min(Sal) from emp

Where level<=5

Connect by prior sal<sal

Group by level;

**Display the top 5 salaried Employees:**

Sql>Select \* from emp

Where sal in (select max(sal) from emp

Where level<=5

Connect by prior sal>sal

Group by level)

**Que: Display the least 5 salaried Employees.**

**ROWNUM Pseudo Column:**

* The Oracle Engine assign a ROWNUM value to each row it is retrieved.
* The first row select has a ROWNUM of 1, The second has 2, and so on...
* When ORDER BY clause follows a ROWNUM, the rows will be

Re-Ordered by ORDER BY clause.

* If ORDER BY clause is embedded in Sub Query and ROWNUM condition is placed in the TOP\_LEVEL Query, Then the ROWNUM condition can be forced to get applied after the ordering of the rows.
* Condition testing for ROWNUM value equal to any positive integer is always false except digit ‘1’ in where clause.
* It is temporary,
* It can be use at
* Select
* Where
* Group function
* Having

Sql> select rownum,empno,ename,deptno from emp;

Sql> select rownum,empno,ename,deptno from emp where deptno = 10;

Sql> Select Rownum,Empno,Ename,Sal From Emp Where Rownum< = 5;

Sql> Select Lpad(' ',Rownum,'\*') From Emp;

Sql> Select RPAD('&Str', ROWNUM) From Emp Where ROWNUM< = Length('&Str');

Sql> Select ename from emp Where rownum = 1;

Sql> Select RPAD('KRISHNA',ROWNUM)

From Emp

Where ROWNUM<= 7

ORDER BY ROWNUM DESC;

Display the alternative record.

Sql> select Rownum,Ename From Emp

Group By Rownum,Ename

Having Mod(Rownum,2)=0

Order by rownum;

Sql> Select Rownum,Ename From Emp

Group by Rownum,Ename

Having Mod(Rownum,2)!= 0;

**Querying for Top 'N' Records**

* We can ask for Nth largest or smallest values of a column.
* Never use ROWNUM and ORDER BY clause together as oracle first fetches the rows according to ROWNUM and then sorts the found rows.
* From Orcale 8i,ORDER BY clause can be used in INLINE VIEWS.

Sql> Select Ename,Sal,Rownum From emp Where Rownum< = 5 Order By Sal Desc;

Sql>Select Ename,Sal,Rownum

From (select Rownum,Ename,Sal From Emp Order By Sal Desc) Where Rownum< = 5;

Sql> Select \* from (select \* from emp order by sal)

Where rownum< = 5;

Sql>Select \*From Emp Where ROWNUM< = 15 MINUS

Select \*From Emp Where ROWNUM< = 10;

Sql> select rownum,empno,ename,sal From(select \* from emp order by sal desc ) group by rownum,empno,ename,sal having rownum ='&n';

**ROWID Pseudo Column:**

ROWID is an exact physical address of row.

ROWID is used by oracle to locate any row.

The ROWID value is assigned by oracle itself.

ROWID'S are unique identifies for a row in a table.

When a Row is DELETED, ORACLE may reassign its ROWID to a new row that is inserted.

The ROWID can never be INSERTED, UPDATED and DELETED manually.

The ROWID pseudo column can be used in SELECT and WHERE clauses.

Sql>Select Rowid,Empno,Ename From Emp;

Sql>select max(rowid) from emp;

Sql> Select \* from emp

Where rowid=' AAARfRAAEAAAAANAAM';

Sql> Select \* from emp

Where rowid<'AAAW4TAAGAAAACzAAN';

Sql>SELECT \*FROM EMP P

WHERE ROWID<(SELECT MAX(ROWID)

FROM EMP S WHERE P.ENAME = S.ENAME);

Sql> DELETE FROM EMP P

WHERE ROWID<(SELECT MAX(ROWID)

FROM EMP S WHERE P.ENAME = S.ENAME);

Sql>select \* from emp

where rowid not in(select min(rowid)

from emp group by deptno);

Sql>Select rownum,empno,ename From emp Where(rowid,1) in (select rowid,mod(rownum,2) from emp);

Sql> Select B.sal,Sum(A.Sal)

From Emp A,Emp B

Where A.Rowid<=B.Rowid

Group By B.Rowid,B.Sal

order by Sum(A.Sal);

Sql>Select B.Ename,B.Job,B.sal,Sum(A.Sal) "Cum Sal"

From Emp A,Emp B

Where A.Rowid<=B.Rowid

Group By B.Rowid,B.Sal,B.Ename,B.Job

order by Sum(A.Sal);