Chapter 5

Advanced SQL
**Recursion in SQL**

**Example.** Let Flights(Flight#, Source_City, Dest_City) be a relational schema

**Example.** A recursive SQL query (*not implemented in Oracle 11g*)

WITH RECURSIVE Reaches(Source_City, Dest_City) AS
  (SELECT Source_City, Dest_City FROM Flights)
UNION
  (SELECT R1.Source_City, R2.Dest_City FROM Reaches R1, Reaches R2
   WHERE R1.Dest_City = R2.Source_City)
Recursion in SQL – Hierarchical Queries

- **Start With – Connect By [Prior]**

  ![Diagram](image)

- The *start with-connect by prior* clause can be used to select data with hierarchical relationships
  - usually some sort of parent $\rightarrow$ child (supervisor $\rightarrow$ supervisee, or part $\rightarrow$ subparts) relationships involved

- **Example.** Given the schema `parent(Pname, Cname)`, retrieve all the *descendents* of “Joe Young.”

  ```sql
  SELECT Cname
  FROM Parent
  START WITH Pname = 'Joe Young'
  CONNECT BY PRIOR Cname = Pname;
  ```
Recursion in SQL – Hierarchical Queries

- **Start With – Connect By [Prior]**
  - Using *start with–connect by [prior]* to answer hierarchical queries on a table.

- **Start With** – indicates root node.

- **Connect by** – indicates relationships between parent and child rows within the hierarchy.

- **Prior** - keyword indicates the *parent*.

- **LEVEL** – *pseudocolumn* function that returns 1 for root, 2 for children of root, 3 for next child level, etc.
Hierarchical Queries in SQL – Start With

- The *Start With* clause identifies the row(s) to be used as the root(s) of a hierarchical query.
- The clause specifies a *condition* that the roots must satisfy.
- If this clause is omitted, SQL uses *all* rows in the table as root rows.
- A *Start With* condition can contain a *subquery*, whereas a *Connect By* condition cannot contain a *subquery*.

Example.

```
SELECT PRIOR Cname
FROM Parent
START WITH Pname = 'Joe Young'
...```
Hierarchical Queries in SQL – Connect By

- The *Connect By* clause specifies a *relationship* (condition) between *parent* and *child* rows in a hierarchical query.

- Some part of the condition must use the *Prior* operator to refer to the *parent* row. The part of the condition containing the *Prior* operator must have one of the following forms:

  \[
  \text{PRIOR expr comparison_operator expr, or expr comparison_operator PRIOR expr}
  \]

- **Example.** Given the relation `Emp(Emp#, Mgr#, Sal, Comm)`, the following *Prior* operator applies only to the `Emp#` of the *parent* row and `mgr`, `sal`, and `comm` values for the *child* row:

  \[
  \text{CONNECT BY PRIOR Emp# = mgr# AND sal > comm;}
  \]

- If the *Connect By* clause results in a *loop* in the hierarchy, SQL returns an error. A *loop* occurs if one row is both the *parent* and a *child* of another row.
In Oracle 11g, CONNECT BY could not be used in a query that contains a join.

**Example.** Given the following relation schemas:

\[
\begin{align*}
\text{Emp}(EID, Ename, DNO, MgrID) \\
\text{Dept}(DNO, Dname)
\end{align*}
\]

the following query is invalid in Oracle 11g:

```sql
SELECT Ename, MgrID, Dname
FROM Emp E, Dept D
WHERE E.DNO = D.DNO
START WITH Ename = 'King'
CONNECT BY PRIOR Mgr_ID = EID
```
Hierarchical Queries in SQL – Where Clauses

- SQL evaluates the *Where*-clause condition for each row *individually*
- *Where* clause - restricts the rows returned by the query *without* affecting other rows of the hierarchy.
- If a row does not satisfy the condition specified in the *Where* clause of the query, SQL
  - eliminates it from the hierarchy, but
  - does not remove all the children (and their descendents) of the row, which can be accomplished by specifying the condition in the CONNECT BY clause
- Example.

```sql
SELECT Cname
FROM Parent
WHERE Cname != 'Peter Pan'
START WITH Pname = 'Joe Young'
CONNECT BY PRIOR Cname = Pname;
```
Hierarchical Queries in SQL – Level Number

- Nodes in a tree are assigned level numbers, depending on how far removed they are from the root of the tree.

- **LEVEL** is a pseudo-column, which can be specified in an SQL statement where a column name may appear.

- **Example.** Given the schema `use(Part#, Sub-part#, Quantity)`, retrieve all the subparts used by Part $P_1$, with layers identified.

```sql
SELECT LEVEL AS Layer, Part#, Sub-part# 
FROM use 
START WITH Part# = ‘$P_1$’ 
CONNECT BY PRIOR Sub-part# = Part#;
```
Hierarchical Queries in SQL – To_Number

- Convert a character value containing a number to a number
- Format: To_number(string-expression)
  - The string expression to be converted can be the name of a column, a string literal, or the result of another function, where the underlying data type is of type CHAR.
- Example. Use To_number to convert a string to a number before inserting it in the database:

  UPDATE employees
  SET salary = salary + To_Number('1500.00')
  WHERE Lname = 'Williams';
Hierarchical Queries in SQL – Order By

- Specifies the sorting of rows in a result set.
- Format: ORDER BY ordering-item [ASC | DESC] {, ordering-item [ASC | DESC] ...}
  
  - **Ordering-item**: an expression that determines the sort order. The clause can contain one or more ordering items.
  - **ASC**: sorting in ascending order (default).
  - **DESC**: sorting in descending order.

- **Example**.

```sql
SELECT Dept_no, Emp_name, Job_Title
FROM Employees
ORDER BY Dept_no DESC
```