Advanced SQL in Oracle and SQL Server

The WITH Clause – PART II (Recursive)

Scott L. Hecht
http://www.sheepsqueezers.com
@sheepsqueezers

pluralsight
hardcore developer training

Module Contents

- **The WITH Clause**
  - Introduction
  - Data Used in Module
  - Recursive WITH Clause
    - Preliminaries
      - What is Recursion?
      - What is a Hierarchy?
      - Depth vs. Breadth Explained
      - Iteration is Not a Dirty Word!
    - Anchor Query and Recursive Query
    - Example #1: Factorials using a Recursive WITH Clause
    - Example #2: Family Hierarchical Queries
    - Example #3: Manager/Employee Hierarchical Queries
    - Using the MAXRECURSION Option (SQL SERVER-SPECIFIC)
    - SEARCH, SET and CYCLE (ORACLE-SPECIFIC)

Introduction

- **Why Learn the WITH Clause?**
  - Complex (Recursive)
    - Allows you to spin through table arranged hierarchically
    - Allows for parent/child relationships to be accessed via SQL
  - Availability:
    - Oracle: 11gR2
    - SQL Server: 2005

Data Used in Module

- **Table**
  - NUMBERS
- **Columns**
  - NUM – column containing the numbers 1 to 4
- **Data**

NUM
1
2
3
4

Data Used in Module

- **Table**
  - FAMILY
- **Columns**
  - CHILD_KEY – column containing child description (non-nullable)
  - PARENT_KEY – column containing the parent description (nullable)
- **Data**

CHILD_KEY	PARENT_KEY
GRANDPARENTS	
PARENTS	GRANDPARENTS
CHILD-1	PARENTS
CHILD-2	PARENTS

Data Used in Module

- **Table**
  - COMPANY
- **Columns**
  - EMPLOYEE_ID – employee identification number (non-nullable)
  - EMPLOYEE_NAME – name of the employee
  - MANAGER_ID – EMPLOYEE_ID of the manager (nullable)
  - SALARY – the salary of the employee
- **Data**

EMPLOYEE_ID	EMPLOYEE_NAME	MANAGER_ID	SALARY
1	FRED		100000
2	BARNEY	1	50000
3	WILMA	1	50000
4	BETTY	3	**40000**
5	PEBBLES	3	40000
6	BAM-BAM	4	20000
7	DINO	4	20000
8	HOPPY	4	**40000**

Recursive WITH Clause

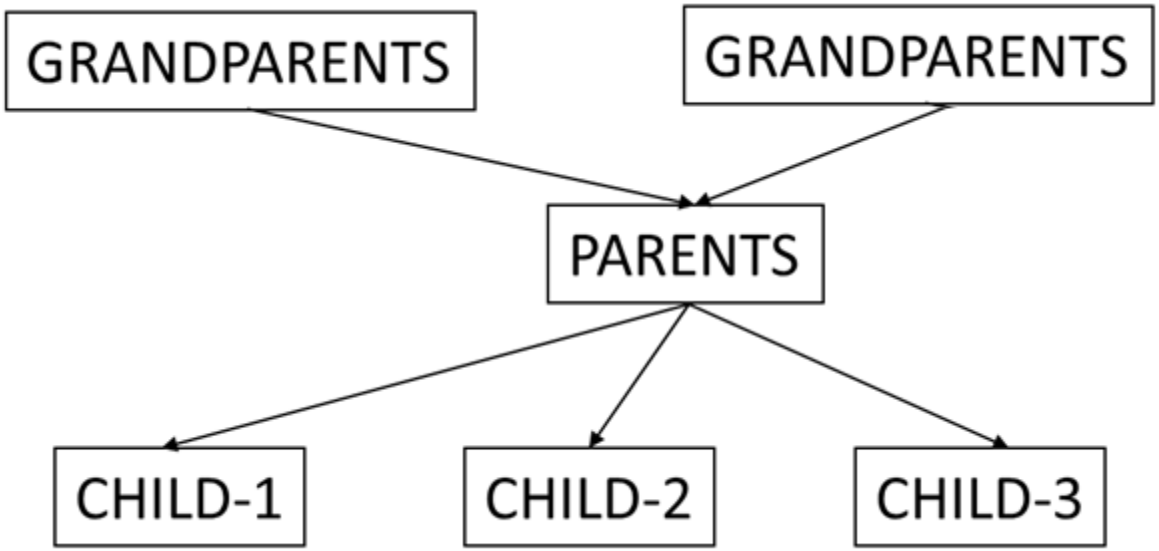
- **Preliminaries – What is Recursion?**
  - *Recursion* is the process a procedure goes through when one of the steps of the procedure involves invoking the procedure itself. A procedure that goes through recursion is said to be *recursive*.
  - The Recursive WITH Clause uses recursion (of course!)
  - One very popular example of recursion is the factorial function:

$$n! = n * (n-1) * (n-2) * \ldots * 1 = n * (n-1)!$$

$$\text{FACT}(n) \leftarrow \{ n * \text{FACT}(n-1) \}$$

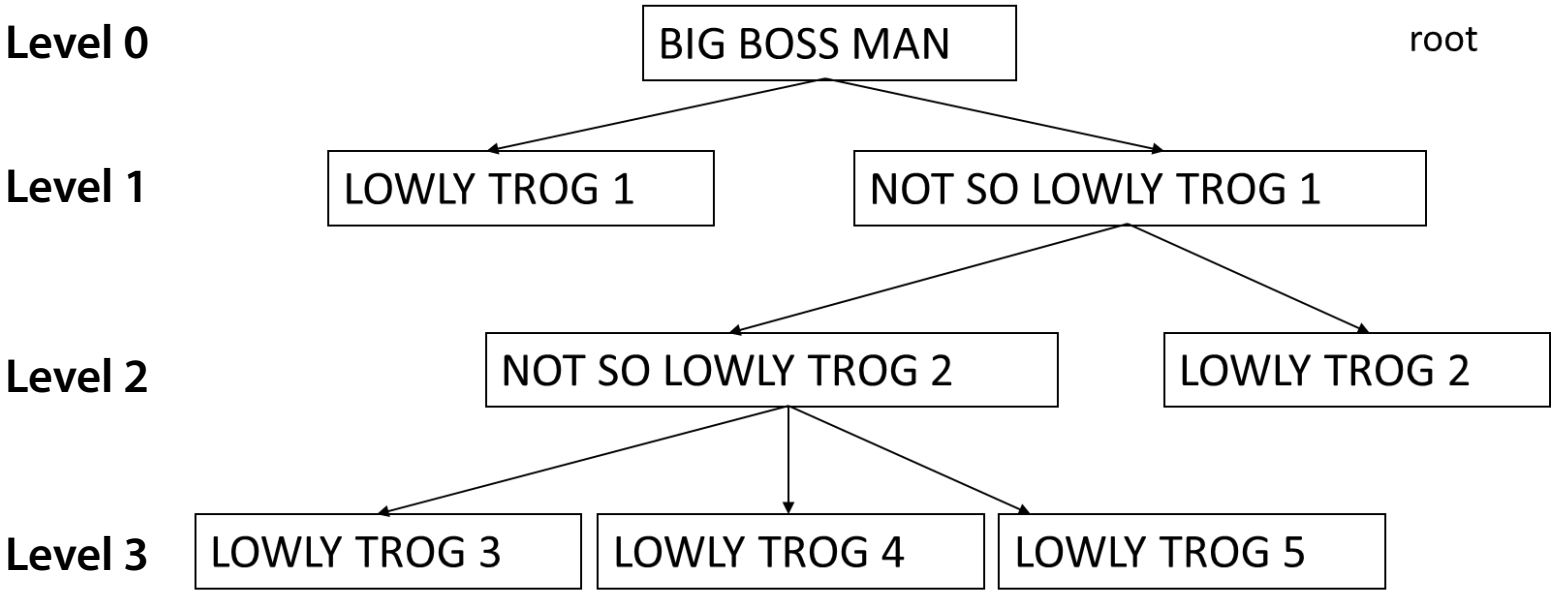
Recursive WITH Clause

- **Preliminaries – What is a Hierarchy?**
  - A *hierarchy* is an arrangement of items (e.g., people, plants, animals, categories, languages, etc.) in which the items are represented as being *above*, *below*, or *at the same level as* one another.
  - The Recursive WITH Clause allows you to work with hierarchical data



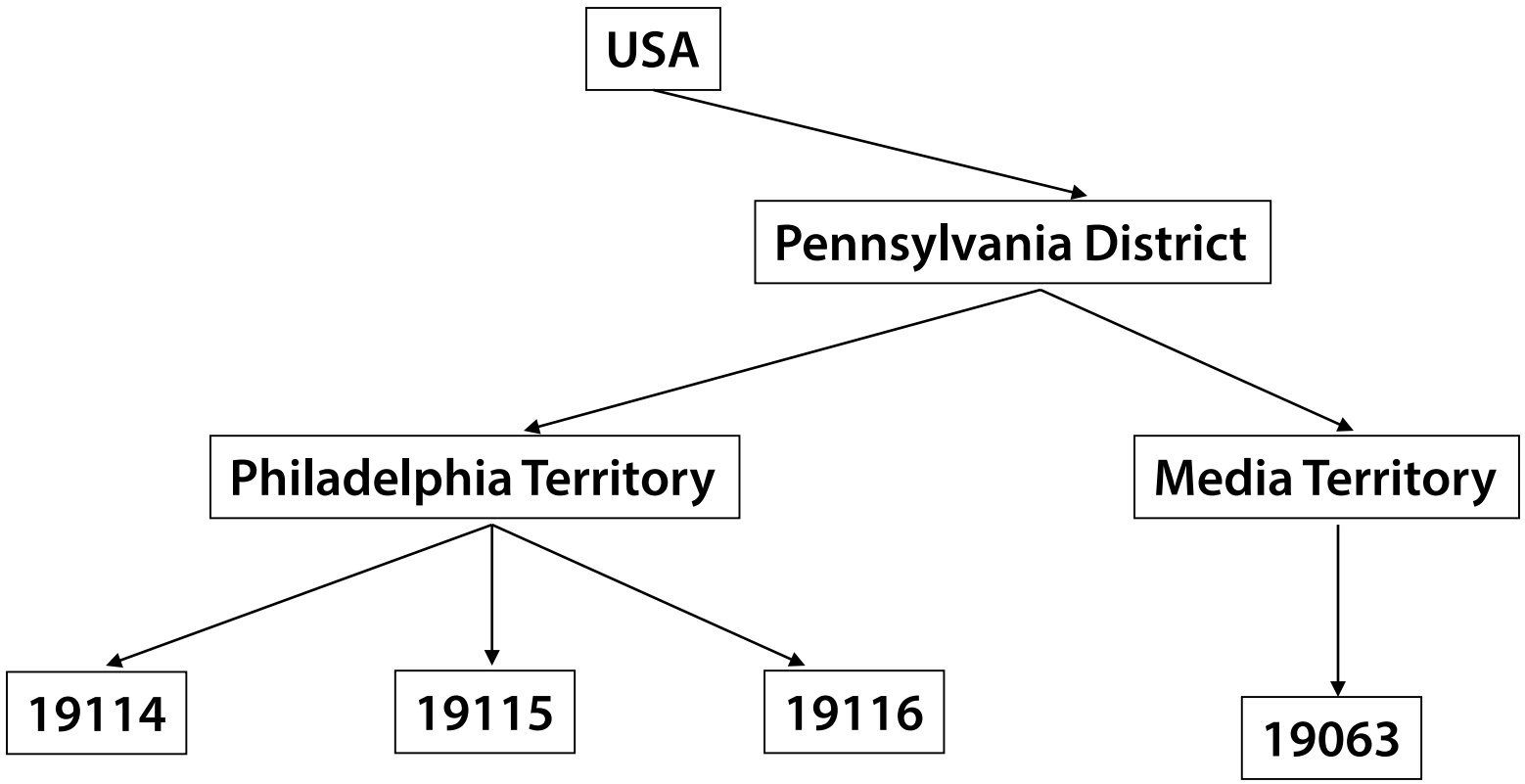
Recursive WITH Clause

- **Preliminaries – What is a Hierarchy? (*continued*)**
  - A hierarchy is also known as a *Parent-Child Relationship*, although that term is used for things not involving grandparents, parents, whining kids, etc.



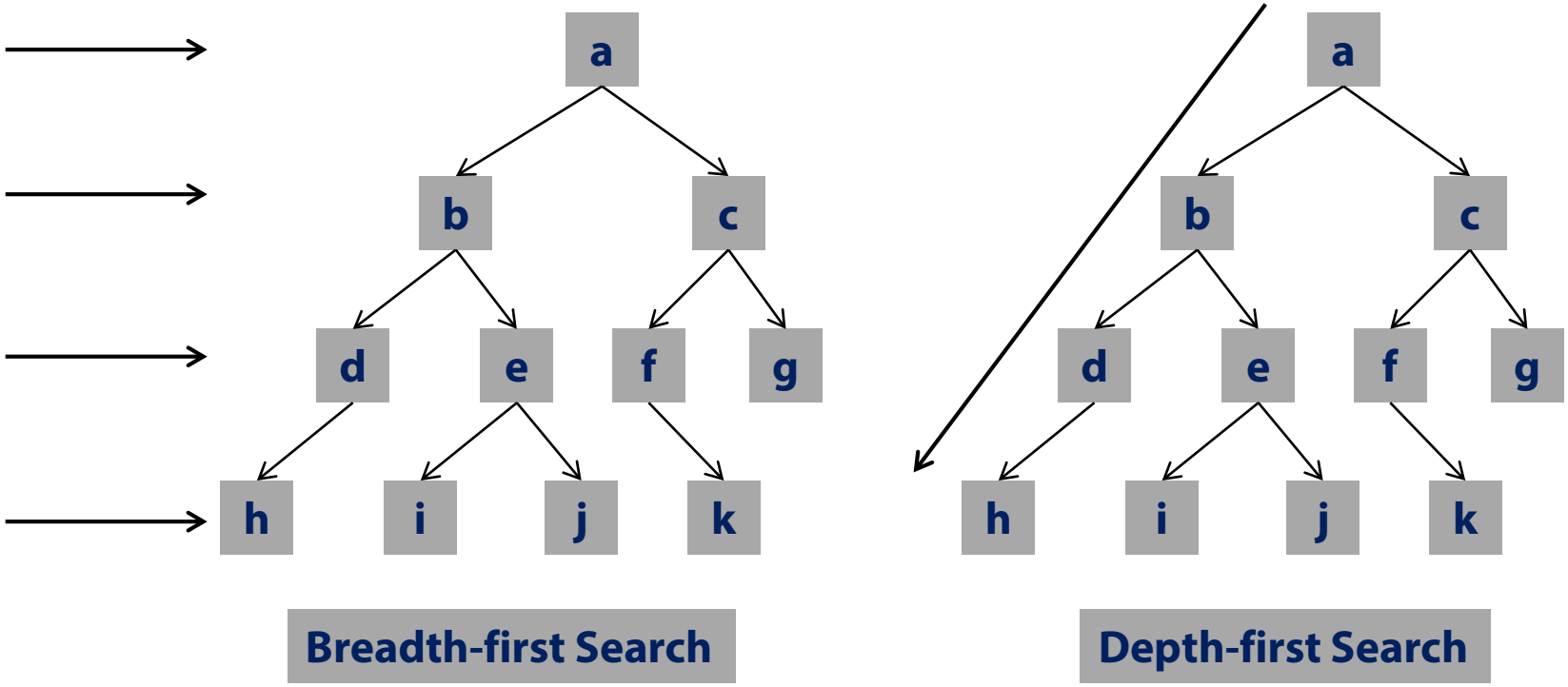
Recursive WITH Clause

- **Preliminaries – What is a Hierarchy? (*continued*)**
  - As another example, you can map a salesperson's zip codes to territories to districts to country.



Recursive WITH Clause

- **Preliminaries – Depth vs. Breadth Explained**
  - Only used in Oracle-specific SEARCH syntax



Recursive WITH Clause

- **Preliminaries – Iteration is Not a Dirty Word!**
  - SQL programmers don't usually iterate through data
  - Think in terms of sets/tables of data
  - Two SQL constructs that require us to think iteratively:
    - Correlated Subqueries
    - Recursive WITH Clause
  - Availability:
    - Oracle: 11g/R2 (Recursive Subquery Factoring Clause)
    - SQL Server: 2005 (Recursive Common Table Expressions)

Recursive WITH Clause

- **Anchor Query and Recursive Query**
  - A Recursive WITH Clause…
    - …is used to iterate through a table *usually* formatted hierarchically
    - …*usually* requires there to be a parent column and child column
  - Parent/Child keys can be character or numeric
  - Syntax similar to Non-Recursive WITH Clause
  - Format of the SQL appearing inside is important:
    - Anchor Query – used for initialization
    - UNION ALL
    - Recursive Query – used for iteration

```
WITH RSFC(cols) AS (
                    ANCHOR QUERY
                    UNION ALL
                    RECURSIVE QUERY
                   )
SELECT cols,…
 FROM RSFC
```

Initialization

Iteration (must refer to RSFC!)

Example #1

- **Task: Compute *4!* using a Recursive WITH Clause**

TABLE: NUMBERS

NUM
1
2
3
4

```
WITH RSFC(ITERATION,RUNNING_FACTORIAL) AS (
  SELECT NUM AS ITERATION,
         1 AS RUNNING_FACTORIAL
   FROM NUMBERS
   WHERE NUM = 1
  UNION ALL
  SELECT R.ITERATION+1,
         R.RUNNING_FACTORIAL * B.NUM
   FROM RSFC R INNER JOIN NUMBERS B
   ON (R.ITERATION+1) = B.NUM
 )
 SELECT ITERATION,RUNNING_FACTORIAL
  FROM RSFC
```

ITERATION	RUNNING_FACTORIAL
1	1
2	2
3	6
4	24

Initialization

ITERATION	RUNNING_FACTORIAL
1	1

Iteration

```
SELECT R.ITERATION+1,
       R.RUNNING_FACTORIAL * B.NUM
FROM RSFC R INNER JOIN NUMBERS B
ON (1+1) = B.NUM
```

ITERATION	RUNNING_FACTORIAL
2	2 = (1*2)

Iteration

```
SELECT R.ITERATION+1,
       R.RUNNING_FACTORIAL * B.NUM
FROM RSFC R INNER JOIN NUMBERS B
ON (2+1) = B.NUM
```

ITERATION	RUNNING_FACTORIAL
3	6 = (1*2*3)

Iteration

```
SELECT R.ITERATION+1,
       R.RUNNING_FACTORIAL * B.NUM
FROM RSFC R INNER JOIN NUMBERS B
ON (3+1) = B.NUM
```

ITERATION	RUNNING_FACTORIAL
4	24 = (1*2*3*4)

Example #2

- **Task: Spin thru FAMILY table starting at PARENT_KEY IS NULL.**

TABLE: FAMILY

CHILD_KEY	PARENT_KEY
GRANDPARENTS	
PARENTS	GRANDPARENTS
CHILD-1	PARENTS
CHILD-2	PARENTS

```
WITH RSFC(CK,PK,LVL) AS (
  SELECT CHILD_KEY,PARENT_KEY,0 as LVL
   FROM FAMILY
   WHERE PARENT_KEY IS NULL
  UNION ALL
  SELECT CHILD_KEY,PARENT_KEY,LVL+1
   FROM RSFC R INNER JOIN FAMILY F
   ON R.CK = F.PARENT_KEY
 )
 SELECT *
  FROM RSFC
```

CK	PK	LVL
GRANDPARENTS		0
PARENTS	GRANDPARENTS	1
CHILD-1	PARENTS	2
CHILD-2	PARENTS	2

Initialization

CK	PK	LVL
GRANDPARENTS	NULL	0

Iteration

```
SELECT CHILD_KEY,PARENT_KEY,lvl+1
 FROM RSFC R INNER JOIN FAMILY F
 ON 'GRANDPARENTS' = F.PARENT_KEY
```

CK	PK	LVL
PARENTS	GRANDPARENTS	1

Iteration

```
SELECT CHILD_KEY,PARENT_KEY,lvl+1
 FROM RSFC R INNER JOIN FAMILY F
 ON 'PARENTS' = F.PARENT_KEY
```

CK	PK	LVL
CHILD-1	PARENTS	2
CHILD-2	PARENTS	2

Iteration

```
SELECT CHILD_KEY,PARENT_KEY,lvl+1
 FROM RSFC R INNER JOIN FAMILY F
 ON F.PARENT_KEY IN
                ('CHILD-1','CHILD-2')
```

---------- YIELDS NO DATA! ---------

Example #3

- **Task: Spin thru COMPANY table starting at MANAGER_ID=3.**

EMPLOYEE_ID	EMPLOYEE_NAME	MANAGER_ID
1	FRED	
2	BARNEY	1
3	WILMA	1
4	**BETTY**	**3**
5	**PEBBLES**	**3**
6	BAM-BAM	4
7	DINO	4
8	HOPPY	4

```
WITH RSFC(EMPID,ENAME,MGRID,LVL) AS (
 SELECT EMPLOYEE_ID,EMPLOYEE_NAME,
        MANAGER_ID,0 as LVL
  FROM COMPANY
  WHERE MANAGER_ID=3
 UNION ALL
 SELECT EMPLOYEE_ID,EMPLOYEE_NAME,
        MANAGER_ID,LVL+1
  FROM RSFC R INNER JOIN COMPANY C
  ON R.EMPID = C.MANAGER_ID
)
SELECT *
 FROM RSFC
```

Initialization →

EMPID	ENAME	MGRID	LVL
4	BETTY	3	0
5	PEBBLES	3	0

Iteration →

```
SELECT EMPID,ENAME,MGRID,LVL
 FROM RSFC R INNER JOIN COMPANY C
 ON C.MANAGER_ID IN (4,5)
```

EMPID	ENAME	MGRID	LVL
4	BETTY	3	0
5	PEBBLES	3	0
6	BAM-BAM	4	1
7	DINO	4	1
8	HOPPY	4	1

Example #3

- **Task: Spin thru COMPANY table starting at MANAGER_ID=3.**
- **Note: Add the Manager's name to the results.**

```
WITH RSFC(empid,ename,mgrid,lvl) AS (
 SELECT EMPLOYEE_ID,EMPLOYEE_NAME,MANAGER_ID,0 as lvl
  FROM COMPANY
  WHERE MANAGER_ID=3
 UNION ALL
 SELECT EMPLOYEE_ID,EMPLOYEE_NAME,MANAGER_ID,lvl+1
  FROM RSFC R INNER JOIN COMPANY F
  ON F.MANAGER_ID = R.empid
)
SELECT A.EMPID,A.ENAME,A.MGRID,A.LVL,
       B.EMPLOYEE_NAME AS MGR_NAME
 FROM RSFC A LEFT JOIN COMPANY B
 ON A.MGRID=B.EMPLOYEE_ID
 ORDER BY A.LVL,A.EMPID
```

EMPID	ENAME	MGRID	LVL	**MGR_NAME**
4	BETTY	3	0	WILMA
5	PEBBLES	3	0	WILMA
6	BAM-BAM	4	1	BETTY
7	DINO	4	1	BETTY
8	HOPPY	4	1	BETTY

Recursive WITH Clause

- **Using the MAXRECURSION Option (SQL Server-Specific Syntax)**
  - MAXRECURSION can be used to prevent a poorly formed Recursive Common Table Expression (CTE) from entering into an infinite loop.

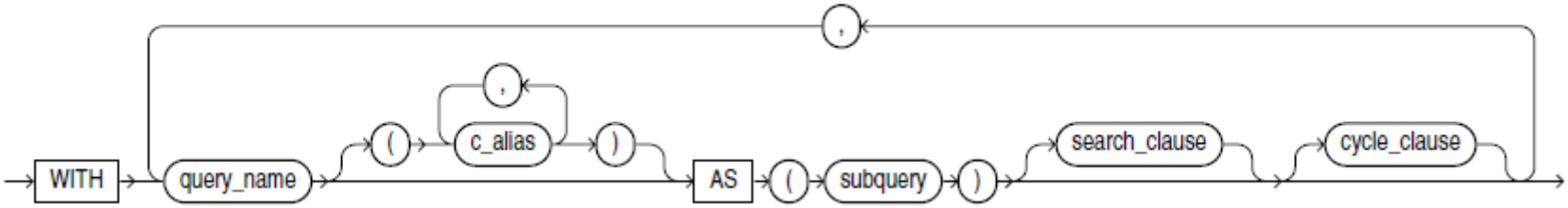
```
WITH RSFC(ITERATION,RUNNING_FACTORIAL) AS (
  SELECT NUM AS ITERATION,
         1 AS RUNNING_FACTORIAL
   FROM NUMBERS
   WHERE NUM = 1
  UNION ALL
  SELECT R.ITERATION+1,
         R.RUNNING_FACTORIAL * B.NUM
   FROM RSFC R INNER JOIN NUMBERS B
   ON (R.ITERATION+1) = B.NUM
 )
 SELECT ITERATION,RUNNING_FACTORIAL
  FROM RSFC
  OPTION (MAXRECURSION 2)
```

  - If number of loops exceeds set limit: *The statement terminated. The maximum recursion 2 has been exhausted before statement completion.*

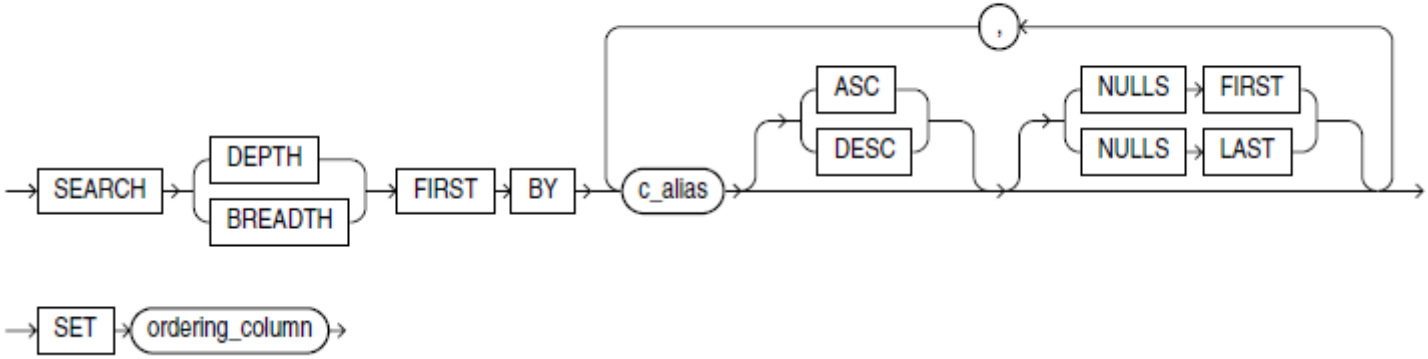
Recursive WITH Clause

- **SEARCH and SET (Oracle-Specific Syntax)**
  - The SEARCH option allow you to return your results in either depth-first or breadth-first ordering
  - The SET option allows you to name the *internally-generated* ordering column which can be used in an ORDER BY Clause

***subquery_factoring_clause*::=**



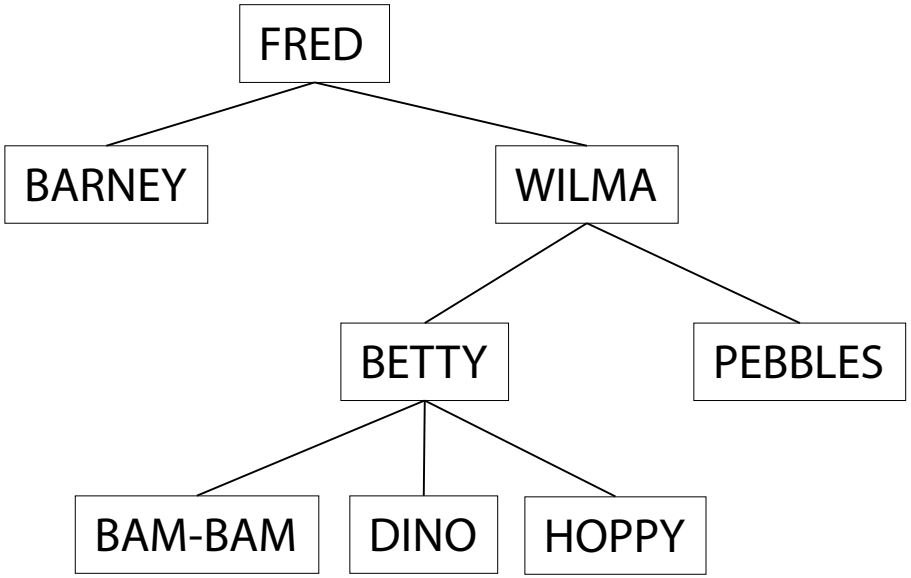
***search_clause*::=**



Example #4

- **Task: Spin thru the COMPANY table sorting BREADTH FIRST.**

EMPLOYEE_ID	EMPLOYEE_NAME	MANAGER_ID
1	FRED	
2	BARNEY	1
3	WILMA	1
4	BETTY	3
5	PEBBLES	3
6	BAM-BAM	4
7	DINO	4
8	HOPPY	4



Example #4

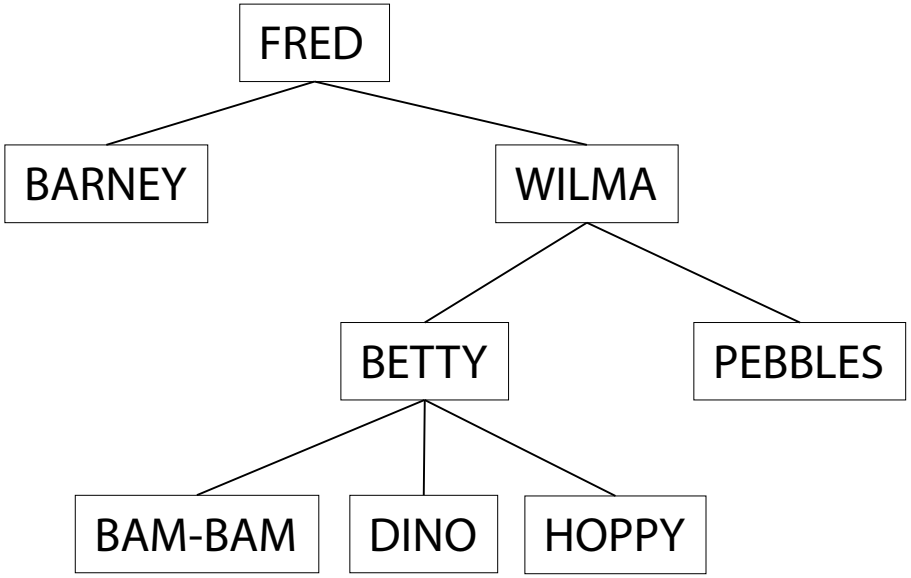
- **Task: Spin thru the COMPANY table sorting BREADTH FIRST.**

```
WITH RSFC(ck,pk,lvl,hier) AS (
   SELECT EMPLOYEE_ID,MANAGER_ID,0 AS lvl,EMPLOYEE_NAME AS hier
     FROM COMPANY
     WHERE MANAGER_ID IS NULL
   UNION ALL
   SELECT EMPLOYEE_ID,MANAGER_ID,lvl+1,hier || '/' || EMPLOYEE_NAME
     FROM RSFC R INNER JOIN COMPANY F
     ON R.ck = F.MANAGER_ID
)
SEARCH BREADTH FIRST BY ck SET ordr
SELECT A.lvl,A.ck,A.pk,A.hier,ordr
 FROM RSFC A
 ORDER BY ordr
```

Example #4

- **Task: Spin thru the COMPANY table sorting BREADTH FIRST.**

LVL	CK	PK	HIER	ORDR
0	1		FRED	1
1	2	1	FRED/BARNEY	2
1	3	1	FRED/WILMA	3
2	4	3	FRED/WILMA/BETTY	4
2	5	3	FRED/WILMA/PEBBLES	5
3	6	4	FRED/WILMA/BETTY/BAM-BAM	6
3	7	4	FRED/WILMA/BETTY/DINO	7
3	8	4	FRED/WILMA/BETTY/HOPPY	8

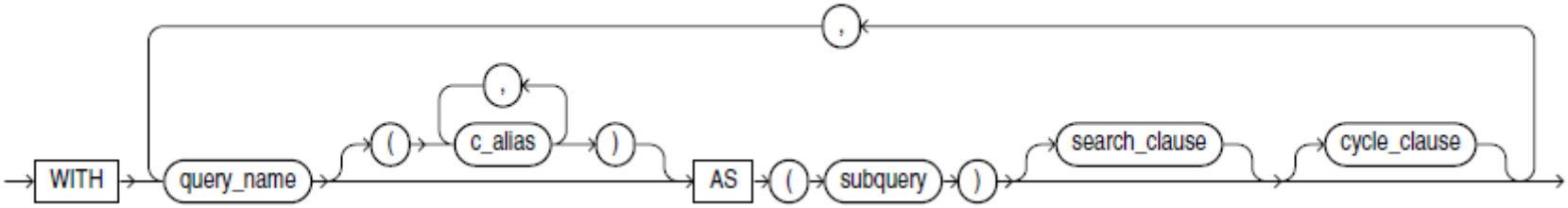


Recursive WITH Clause

- **CYCLE (Oracle-Specific Syntax)**
  - Use CYCLE to determine if a column's value at the lower level appears in an upper level

EMPLOYEE_ID	EMPLOYEE_NAME	MANAGER_ID	SALARY
1	FRED		100000
2	BARNEY	1	50000
3	WILMA	1	50000
4	BETTY	3	**40000**
5	PEBBLES	3	40000
6	BAM-BAM	4	20000
7	DINO	4	20000
8	HOPPY	4	**40000**

subquery_factoring_clause::=



cycle_clause::=

Example #5

- **Task: Determine if an employee is earning the same salary as his/her manager.**

```
WITH RSFC(empid,ename,mgrid,lvl,sal) AS (
 SELECT EMPLOYEE_ID,EMPLOYEE_NAME,MANAGER_ID,0 as lvl,SALARY
  FROM COMPANY
  WHERE MANAGER_ID IS NULL
 UNION ALL
 SELECT EMPLOYEE_ID,EMPLOYEE_NAME,MANAGER_ID,lvl+1,SALARY
  FROM RSFC R INNER JOIN COMPANY F
  ON F.MANAGER_ID = R.empid
)
CYCLE sal SET IS_CYCLE TO 'Y' DEFAULT 'N'
SELECT *
 FROM RSFC
```

EMPID	ENAME	MGRID	LVL	SAL	IS_CYCLE
1	FRED		0	100000	N
2	BARNEY	1	1	50000	N
3	WILMA	1	1	50000	N
4	**BETTY**	**3**	**2**	**40000**	**N**
5	PEBBLES	3	2	40000	N
6	BAM-BAM	4	3	20000	N
7	DINO	4	3	20000	N
8	**HOPPY**	**4**	**3**	**40000**	**Y**

Summary

- Traverse hierarchical (parent-child) tables
- Use SEARCH/SET to order the output in depth/breadth first order
- Use CYCLE to determine if a value in lower level appears in higher level