● 계층적 검색

EMPNO	ENAME	MGR
7369	SMITH	7902
7499	ALLEN	7698
7521	WARD	7698
7566	JONES	7839
7654	MARTIN	7698
7698	BLAKE	7839
7782	CLARK	7839
7788	SCOTT	7566
7839	KING	
7844	TURNER	7698
7876	ADAMS	7788
7900	JAMES	7698
7902	FORD	7566
7934	MILLER	7782

EMPNO	ENAME	LV	MGR	MGR_ENAME	ENAMES
7839	KING	1			KING
7566	JONES	2	7839	KING	KING-JONES
7788	SCOTT	3	7566	JONES	KING-JONES-SCOTT
7876	ADAMS	4	7788	SCOTT	KING-JONES-SCOTT-ADAMS
7902	FORD	3	7566	JONES	KING-JONES-FORD
7369	SMITH	4	7902	FORD	KING-JONES-FORD-SMITH
7698	BLAKE	2	7839	KING	KING-BLAKE
7499	ALLEN	3	7698	BLAKE	KING-BLAKE-ALLEN
7521	WARD	3	7698	BLAKE	KING-BLAKE-WARD
7654	MARTIN	3	7698	BLAKE	KING-BLAKE-MARTIN
7844	TURNER	3	7698	BLAKE	KING-BLAKE-TURNER
7900	JAMES	3	7698	BLAKE	KING-BLAKE-JAMES
7782	CLARK	2	7839	KING	KING-CLARK
7934	MILLER	3	7782	CLARK	KING-CLARK-MILLER

Hierarchical Query

```
SELECT [LEVEL], column, expr...
FROM   table
[WHERE condition(s)]
[START WITH condition(s)]
[CONNECT BY PRIOR condition(s)] ;
```

조건:

```
expr comparison_operator expr
```

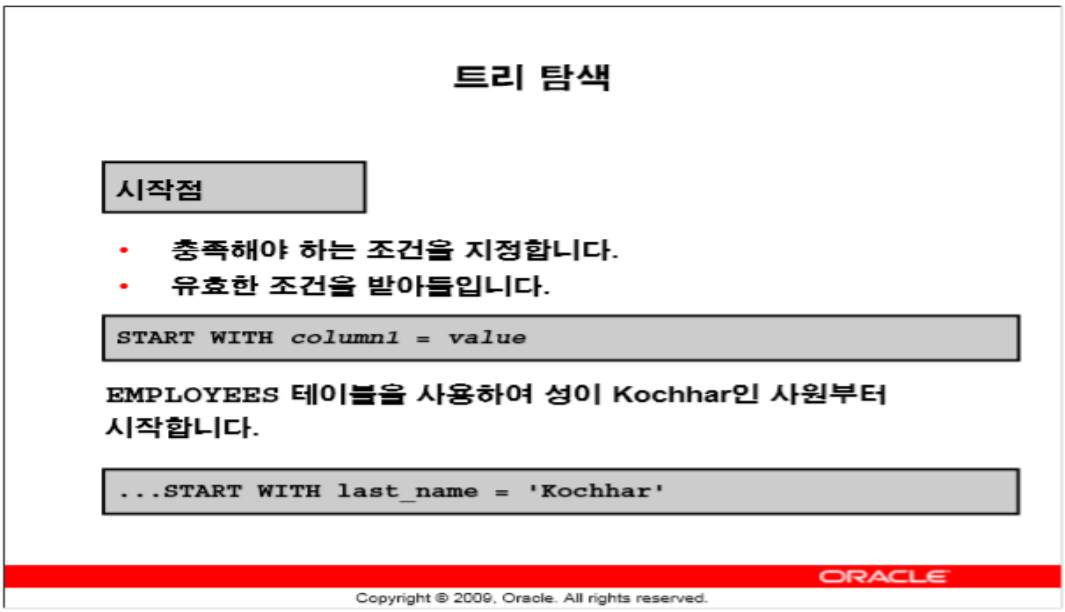
ORACLE

키워드와 절

Hierarchical query는 CONNECT BY 절과 START WITH 절의 존재 여부로 식별할 수 있습니다.

이 구문에서 다음이 적용됩니다.

SELECT	표준 SELECT 절입니다.
LEVEL	hierarchical query에 의해 반환되는 각 행의 경우 LEVEL pseudocolumn은 루트 행에 대해 1을 반환하고 루트의 하위 행에 대해 2를 반환합니다(이하 같은 방식).
FROM *table*	열을 포함하는 테이블, 뷰 또는 스냅샷을 지정합니다. 한 테이블에서만 선택할 수 있습니다.
WHERE	계층 구조의 다른 행에 영향을 주지 않으면서 query에 의해 반환되는 행을 제한합니다.
condition	표현식을 사용한 비교입니다.
START WITH	계층의 루트 행(시작 위치)을 지정합니다. 이 절은 실제 hierarchical query에 필요합니다.
CONNECT BY	상위 및 하위 PRIOR 행 사이에 관계가 있는 열을 지정합니다. 이 절은 hierarchical query에 필요합니다.



트리 탐색

트리의 루트로 사용할 행은 START WITH 절에 의해 결정됩니다. START WITH 절은 유효한 조건을 포함할 수 있습니다.

예제

EMPLOYEES 테이블을 사용하여 회사 대표인 King부터 시작합니다.

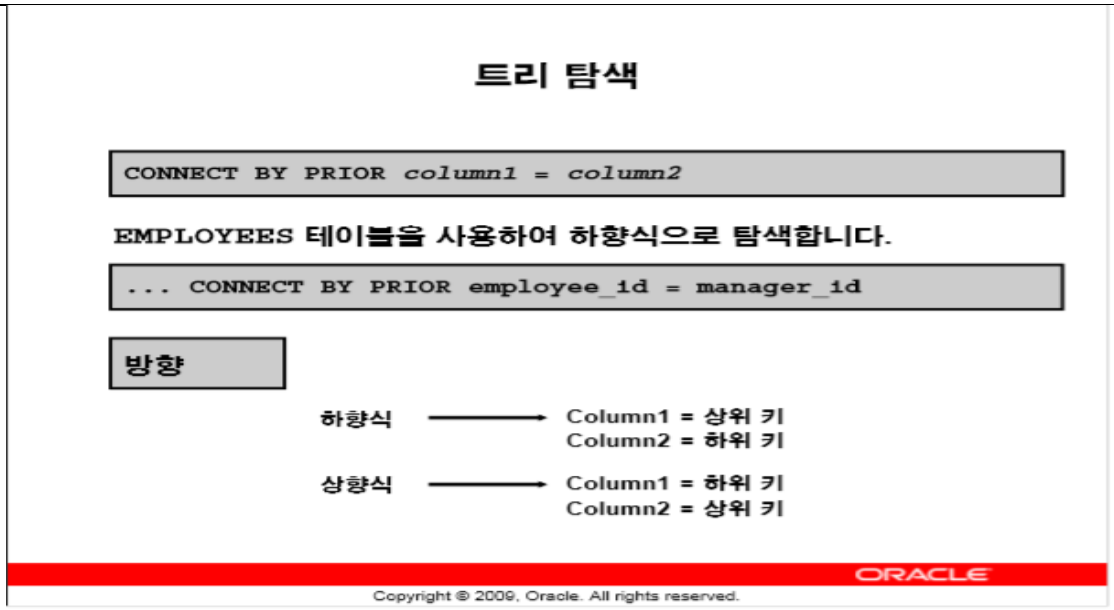
```
... START WITH manager_id IS NULL
```

EMPLOYEES 테이블을 사용하여 사원 Kochhar부터 시작합니다. START WITH 조건은 subquery를 포함할 수 있습니다.

```
... START WITH employee_id = (SELECT employee_id
                              FROM employees
                              WHERE last_name = 'Kochhar')
```

START WITH 절을 생략하면 테이블의 모든 행을 루트 행으로 간주하고 트리 탐색이 시작됩니다.

참고: CONNECT BY 및 START WITH 절은 ANSI(American National Standards Institute) SQL 표준이 아닙니다.



트리 탐색

Query 방향은 CONNECT BY PRIOR 열의 위치에 의해 결정됩니다. 하향식의 경우 PRIOR 연산자를 상위 행을 참조하고, 하향식의 경우 PRIOR 연산자를 하위 행을 참조합니다. 상위 행의 하위 행을 찾기 위해 Oracle 서버는 상위 행에 대해 PRIOR 표현식을 평가하고 테이블의 각 행에 대해 다른 표현식을 평가합니다. 조건이 참인 행은 상위 행의 하위 행입니다. Oracle 서버는 항상 현재 상위 행에 대해 CONNECT BY 조건을 평가하여 하위 행을 선택합니다.

예제

EMPLOYEES 테이블을 사용하여 하향식으로 탐색합니다. 상위 행의 EMPLOYEE_ID 값이 하위 행의 MANAGER_ID 값과 같은 계층 관계를 정의합니다.

```
... CONNECT BY PRIOR employee_id = manager_id
```

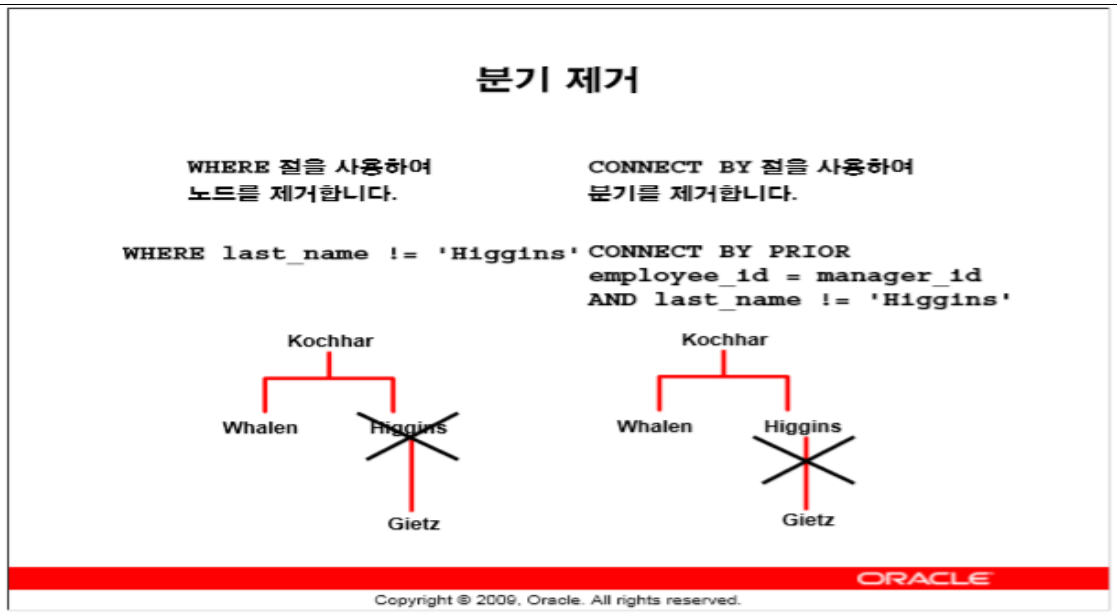
EMPLOYEES 테이블을 사용하여 상향식으로 탐색합니다.

```
... CONNECT BY PRIOR manager_id = employee_id
```

PRIOR 연산자를 반드시 CONNECT BY 바로 뒤에 코딩할 필요는 없습니다. 따라서 다음 CONNECT BY PRIOR 절은 이전 예제의 절과 동일한 결과를 나타냅니다.

```
... CONNECT BY employee_id = PRIOR manager_id
```

참고: CONNECT BY 절은 subquery를 포함할 수 없습니다.



분기 제거

WHERE 절과 CONNECT BY 절을 사용하여 트리를 제거합니다. 즉, 표시할 노드 또는 행을 제어합니다. 사용하는 술어는 부울 조건으로 동작합니다.

예제

루트에서 시작하여 하향식으로 탐색하고 결과에서 Higgins 사원은 제거하고 하위 행은 처리합니다.

```
SELECT  department_id, employee_id,last_name, job_id, salary
FROM    employees
WHERE   last_name != 'Higgins'
START   WITH manager_id IS NULL
CONNECT BY PRIOR employee_id = manager_id;
```

루트에서 시작하여 하향식으로 탐색하고 Higgins 사원과 하위 행을 모두 제거합니다.

```
SELECT  department_id, employee_id,last_name, job_id, salary
FROM    employees
START   WITH manager_id IS NULL
CONNECT BY PRIOR employee_id = manager_id
AND     last_name != 'Higgins';
```

[상위행과 하위행의 관계]

EMPNO	ENAME	LV	MGR	MGR_ENAME	ENAMES
7839	KING	1			KING
7566	JONES	2	7839	KING	KING-JONES
7788	SCOTT	3	7566	JONES	KING-JONES-SCOTT
7876	ADAMS	4	7788	SCOTT	KING-JONES-SCOTT-ADAMS
7902	FORD	3	7566	JONES	KING-JONES-FORD
7369	SMITH	4	7902	FORD	KING-JONES-FORD-SMITH
7698	BLAKE	2	7839	KING	KING-BLAKE
7499	ALLEN	3	7698	BLAKE	KING-BLAKE-ALLEN
7521	WARD	3	7698	BLAKE	KING-BLAKE-WARD
7654	MARTIN	3	7698	BLAKE	KING-BLAKE-MARTIN
7844	TURNER	3	7698	BLAKE	KING-BLAKE-TURNER
7900	JAMES	3	7698	BLAKE	KING-BLAKE-JAMES
7782	CLARK	2	7839	KING	KING-CLARK
7934	MILLER	3	7782	CLARK	KING-CLARK-MILLER

[계층구조 쿼리]

```
SELECT empno
     , ename
     , mgr
  FROM emp
 START WITH mgr IS NULL
CONNECT BY PRIOR empno = mgr
;
```

계층구조 쿼리 결과

EMPNO	ENAME	MGR
7839	KING	
7566	JONES	7839
7788	SCOTT	7566
7876	ADAMS	7788
7902	FORD	7566
7369	SMITH	7902
7698	BLAKE	7839
7499	ALLEN	7698
7521	WARD	7698
7654	MARTIN	7698
7844	TURNER	7698
7900	JAMES	7698
7782	CLARK	7839
7934	MILLER	7782

[일반정렬 구문]

```
SELECT empno
     , ename
     , LEVEL lv
     , mgr
     , PRIOR ename mgr_ename
     , SUBSTR(SYS_CONNECT_BY_PATH(ename, '-'), 2) enames
  FROM emp
 START WITH mgr IS NULL
CONNECT BY PRIOR empno = mgr
 ORDER BY empno
;
```

[일반정렬 구문 결과]

EMPNO	ENAME	LV	MGR	MGR_ENAME	ENAMES
7369	SMITH	4	7902	FORD	KING-JONES-FORD-SMITH
7499	ALLEN	3	7698	BLAKE	KING-BLAKE-ALLEN
7521	WARD	3	7698	BLAKE	KING-BLAKE-WARD
7566	JONES	2	7839	KING	KING-JONES
7654	MARTIN	3	7698	BLAKE	KING-BLAKE-MARTIN
7698	BLAKE	2	7839	KING	KING-BLAKE
7782	CLARK	2	7839	KING	KING-CLARK
7788	SCOTT	3	7566	JONES	KING-JONES-SCOTT
7839	KING	1			KING
7844	TURNER	3	7698	BLAKE	KING-BLAKE-TURNER
7876	ADAMS	4	7788	SCOTT	KING-JONES-SCOTT-ADAMS
7900	JAMES	3	7698	BLAKE	KING-BLAKE-JAMES
7902	FORD	3	7566	JONES	KING-JONES-FORD
7934	MILLER	3	7782	CLARK	KING-CLARK-MILLER

[최종 쿼리]

```
SELECT empno
     , ename
     , LEVEL lv
     , mgr
     , PRIOR ename mgr_ename
     , SUBSTR(SYS_CONNECT_BY_PATH(ename, '-'), 2) enames
  FROM emp
 START WITH mgr IS NULL
CONNECT BY PRIOR empno = mgr
 ORDER SIBLINGS BY empno
;
```

● 11G에서 새로 등장한 구문 - 타 DBMS에서도 사용 가능한 구문

```
WITH t1(empno, ename, lv, mgr, mgr_ename, enames, empnos) AS
(
SELECT empno
     , ename
     , 1 lv
     , mgr
     , '' mgr_ename
     , ename enames
     , TO_CHAR(empno) empnos
  FROM emp
 WHERE mgr IS NULL
 UNION ALL
SELECT t2.empno
     , t2.ename
     , t1.lv + 1 lv
     , t2.mgr
     , t1.ename mgr_ename
     , t1.enames || '-' || t2.ename enames
     , t1.empnos || '-' || t2.empno empnos
  FROM t1, emp t2
 WHERE t1.empno = t2.mgr
)
SEARCH DEPTH FIRST BY empno SET IDX          --> ORDER SIBLINGS BY 기능에 해당함
CYCLE mgr SET iscycle TO "1" DEFAULT "0"     --> CONNECT_BY_ISCYCLE 기능에 해당함
SELECT *
  FROM t1
;
```