```
REM Author :
REM Date :
REM Objective : Chapter 8. 계층형 쿼리
REM Environment : CentOS 6.5, SQLGate 2010 for Oracle, Oracle 11g Enterprise Edition 11.2.0

REM 수업을 위한 준비하기
/*
    DROP USER HR CASCADE;

    CREATE USER HR IDENTIFIED BY hr
    DEFAULT TABLESPACE USERS
    TEMPORARY TABLESPACE TEMP;

    GRANT RESOURCE, CONNECT TO HR;
    GRANT CREATE VIEW TO HR;
    GRANT CREATE SYNONYM TO HR;

    CONN HR/hr

    SQL>SPOOL D:\hr_test.txt
    START Human_Resources\hr_cre.sql
    START Human_Resources\hr_popul.sql
    START Human_Resources\hr_idx.sql
    START Human_Resources\hr_comnt.sql
    START Human_Resources\hr_schema_data.sql
    --쿼리 수행 도중 다음과 같은 오류발생할 것임--
    1 row updated.
       SET department_name = '공공기관판매(Government Sales)'
                             *
    ERROR at line 2:
    ORA-12899: value too large for column "HR"."DEPARTMENTS"."DEPARTMENT_NAME"
    (actual: 36, maximum: 30)

    1 row updated.
    ...
    ...
    SQL>SPOOL OFF

    --저장 파일을 열어 오류확인할 것
    --아래와 같이 오류처리할 것

        --Human_Resources\hr_cre.sql 파일을 열어서
        --DEPARTMENTS.DEPARTMENT_NAME 를 30에서 40으로 수정할 것

    CONN SYSTEM/javaoracle

    DROP USER HR CASCADE;

    CREATE USER HR IDENTIFIED BY hr
    DEFAULT TABLESPACE USERS
    TEMPORARY TABLESPACE TEMP;

    GRANT RESOURCE, CONNECT TO HR;
    GRANT CREATE VIEW TO HR;
    GRANT CREATE SYNONYM TO HR;

    CONN HR/hr

    SQL>SPOOL D:\hr_test.txt
    START Human_Resources\hr_cre.sql
    START Human_Resources\hr_popul.sql
    START Human_Resources\hr_idx.sql
    START Human_Resources\hr_comnt.sql
    START Human_Resources\hr_schema_data.sql
    SQL>SPOOL OFF

    --파일을 열어서 오류가 없는 것을 확인해야 함.
*/

REM 계층형 쿼리(Hierarchical Query) 소개
--1. ORACLE 에서만 지원하는 막강한 기능 중의 하나(다른 데이터베이스에서도 불가능한 것은 아니지만, ORACLE 에서는 단 하나의 SQL 문장만 사용하여 계층형 정보를 표현하는
```

```
것이 가능)
--2. 관계형 데이터베이스에서 말하는 관계(relation)라는 의미와 계층형과는 상반된 개념
--3. 프로그램을 개발하다보면 관계형 데이터를 계층형 정보로 표현해야 하는 경우 발생

REM 계층형 정보란?
--1. 일반적으로 RDBMS 는 데이터베이스에 저장된 데이터들이 서로 연관성있는 데이터를
갖도록 2차원적 구조로 저장 --> Microsoft Office Excel Program
--2. 관계형이라는 의미가 서로 평등하고 수평적인 관계를 의미하는 반면 계층형 구조는
이와 다르게 계급적이고 수직적인 관계를 갖는다.
--3. 예:EMP table 에서 각 사원은 MGR 하위에 소속되어 있는 계층형이다. 답변형
게시판,                                  <BOM(Bill of Material)>

컴퓨터                                          ---1레벨
    |
    |----------------|---------|
  모니터             본체      프린터          ---2레벨
                     |
          |----------|--------|
Mother Board   LAN Card  Power Supply  ---3레벨
          |
          |----|-------|---------|
       RAM   CPU Graphic Card  etc...  ---4레벨

  --1)Node : 각각의 품목. 실제 테이블에서는 하나의 row 에 대응
  --2)Parent : 트리구조의 상위에 있는 노드
  --3)Child
  --4)Leaf
  --5)Root
  --6)Level

REM 계층형 쿼리의 구조
--1. 부모테이블과 자식테이블 관계에서는 계층형 데이터를 표현하기 어렵다.
--2. 결국 self 조인처럼 자기의 상사 정보는 역시 자기 테이블안에 있는 구조이기
때문이다.
--3. 위의 BOM 에서 각 아이템의 상위레벨 역시 같은 테이블에 존재하기 때문이다.
--4. 위의 BOM 을 EMP 테이블처럼 작성하면 아래와 같다.
   --1) 품목 식별자(PK)
   --2) 상위품목 식별자
   --3) 품목 명
   --4) 부품 식별자(FK)
   --5) 품목 수량
/*
BOM TABLE
-----------------------------------------------------------
ITEM_ID   |   PARENT_ITEM_ID  |  ITEM_NAME   |   ITEM_QTY
-----------------------------------------------------------
   1001   |        NULL       |      컴퓨터    |       1
   1002   |        1001       |      본체     |       1
   ...
   1004   |        1001       |      프린터    |       1
   1006   |        1002       |      랜카드    |       1
   ...----------------------------------------------------
*/

REM 계층형 쿼리 작성하기
/*
    CREATE TABLE BOM
    (
        ITEM_ID           NUMBER        NOT NULL,
        PARENT_ID       NUMBER,
        ITEM_NAME      VARCHAR2(20)   NOT NULL,
        ITEM_QTY        NUMBER,
        PRIMARY KEY(ITEM_ID)
    );
    INSERT INTO BOM VALUES (1001, NULL, '컴퓨터', 1);
    INSERT INTO BOM VALUES (1002, 1001, '본체', 1);
    INSERT INTO BOM VALUES (1003, 1001, '모니터', 1);
    INSERT INTO BOM VALUES (1004, 1001, '프린터', 1);

    INSERT INTO BOM VALUES (1005, 1002, 'Mother Board', 1);
    INSERT INTO BOM VALUES (1006, 1002, 'LAN Card', 1);
    INSERT INTO BOM VALUES (1007, 1002, 'Power Supply', 1);
    INSERT INTO BOM VALUES (1008, 1005, 'RAM', 1);
```

```
    INSERT INTO BOM VALUES (1009, 1005, 'CPU', 1);
    INSERT INTO BOM VALUES (1010, 1005, 'Graphics Card', 1);
    INSERT INTO BOM VALUES (1011, 1005, '기타장치', 1);
*/
--1. 각각의 품목들이 서로 부모와 자식 관계로 연결되어 있기 때문에 조인을 사용해야 하는데, 셀프조인을 사용해야 한다.
--2. 하지만, 컴퓨터의 경우 가장 상위의 레벨이기 때문에 PARENT_ITEM_ID 값이 NULL이다. (EMP 테이블의 MGR이 NULL 인 KING 이 있다.)
--3. 따라서 OUTER JOIN 도 필요하다.
/*
    SELECT b1.ITEM_NAME, b1.ITEM_ID, b2.ITEM_NAME AS PARENT_ITEM
    FROM BOM b1, BOM b2
    WHERE  b1.PARENT_ID = b2.ITEM_ID(+)
    ORDER BY b1.ITEM_ID;
*/
--4. 하지만, 완전히 계층적으로 조회되는 것이 아니다. 본체의 하위 품목인 랜카드, 마더보드, 파워 서플라이 품목이 프린터 바로 아래에 있음으로, 마치 프린터 하위 품목인 것 처럼 오해가 될 수 있다.
--5. 조회 결과가 트리 구조와 다르다.
--6. 그래서 계층형 쿼리가 필요하다.

REM START WITH ... CONNECT BY 절을 사용한 계층형 쿼리
--1. ORACLE 에서 계층적인 정보를 표현하기 위해 만든 명령어이다.
--2. ORACLE 8i에서부터 지원
--3. Syntax
/*
    SELECT [LEVEL], COLUMN1, COLUMN2,...
    FROM TABLE_NAME
    [WHERE 조건절]
    [START WITH 조건절]
    [CONNECT BY PRIOR 조건절];
*/
/*
    SELECT  LPAD('   ', 2 * (LEVEL - 1)) || ITEM_NAME AS ITEM_NAMES
    FROM  BOM
    START WITH PARENT_ID IS NULL
    CONNECT BY PRIOR  ITEM_ID = PARENT_ID;
*/
  --1)START WITH : ROOT 노드를 찾는 역할을 수행하는 절
  --2)CONNECT BY : 부모와 자식노드들 간의 관계를 연결하는 역할
    --CONNECT BY PRIOR ITEM_ID = PARENT_ID  (O)
    --CONNECT BY ITEM_ID = PRIOR PARENT_ID  (X)
    --CONNECT BY PARENT_ID = PRIOR ITEM_ID (O)
--4. START WITH 조건1  ... CONNECT BY 조건2
  --1)START WITH 조건 1 : 루트노드를 식별. 조건1을 만족하는 모든 ROW 들은 루트노드가 된다. START WITH 절을 생략할 수도 있는데, 이러한 경우 모든 ROW 들을 루트노드로 간주한다. 조건1 에서는 서브쿼리도 올 수 있다.
  --2)CONNECT BY 조건 2 : 부모와 자식노드들 간의 관계를 명시하는 부분이다. 조건2에는 반드시 PRIOR 연산자를 포함시켜야 하며, 이는 부모노드의 칼럼을 식별하는데 사용된다. START WITH과는 달리 조건2에서는 서브쿼리가 올 수 없다.
--5. PRIOR 연산자
  --1)PRIOR 키워드는 오직 계층형 쿼리에서만 사용하는 오라클 연산자.
  --2)CONNECT BY 절에서 등호(=)와 동등한 레벨로 사용되는 연산자.
  --3)CONNECT BY 절에서 해당 칼럼의 부모 로우를 식별하는데 사용.
--6. LEVEL 의사칼럼
  --1)계층형 정보를 표현할 때 레벨을 사용
  /*
    SELECT LEVEL, ITEM_NAME
    FROM BOM;
    --ERROR 발생
    --ORA-01788: CONNECT BY clause required in this query block
  */

REM 실습
/*
    CONN HR/hr
    SET PAGESIZE 60
    SET LINESIZE 110
    COL LEVEL FOR 999999
    COL "성명" FOR A40

    SELECT LEVEL, LPAD('  ', 4 * (LEVEL - 1)) || FIRST_NAME || ' ' || LAST_NAME
    "성명"
```

```
204       FROM EMPLOYEES
205       START WITH MANAGER_ID  IS NULL
206       CONNECT BY MANAGER_ID = PRIOR EMPLOYEE_ID;
207
208       --이번에는 직위별로 직위명까지 조회하기
209       COL "직위" FOR A40
210       COL "성명" FOR A35
211       SELECT b.JOB_TITLE "직위",
212                  LPAD('  ', 4 * (LEVEL - 1)) || a.FIRST_NAME || ' ' || a.LAST_NAME
                   "성명"
213       FROM EMPLOYEES a, JOBS b
214       WHERE a.JOB_ID = b.JOB_ID
215       START WITH a.MANAGER_ID  IS NULL
216       CONNECT BY a.MANAGER_ID = PRIOR a.EMPLOYEE_ID;
217
218       --각 사원들의 부서와 부서가 위치한 지역정보까지 포함하자.
219       SELECT b.JOB_TITLE "직위",
220                  LPAD('  ', 4 * (LEVEL - 1)) || a.FIRST_NAME || ' ' || a.LAST_NAME
                   "성명"
221       FROM EMPLOYEES a, JOBS b
222       WHERE a.JOB_ID = b.JOB_ID
223       START WITH a.MANAGER_ID  IS NULL
224       CONNECT BY a.MANAGER_ID = PRIOR a.EMPLOYEE_ID;
225   */
226   --7. 오라클에서 계층형 쿼리를 수행하는 순서
227     --1)조인이 사용되었다면 가장 먼저 조인을 수행한다.
228     --2)CONNECT BY 조건을 처리한다.
229     --3)나머지 조건(조인을 제외한 조건)을 처리한다.
230
231   REM 실습2
232   /*
233       CONN SCOTT/tiger
234
235       SELECT EMPNO, ENAME, JOB, MGR
236       FROM EMP;
237
238       SELECT EMPLOYEE.EMPNO, EMPLOYEE.ENAME, EMPLOYEE.JOB
239                  MANAGER.ENAME AS MANAGER_NAME
240       FROM EMP EMPLOYEE, EMP MANAGER
241       WHERE EMPLOYEE.MGR = MANAGER.EMPNO;
242
243   KING
                       --1레벨
244      |
245      |---------|-----------------------------------|
246   CLARK            JONES
     BLAKE                                --2레벨
247      |                 |                                          |
248      |               |---------|            |---------|---------|-----------|----------|
249   MILLER  SCOTT  FORD   ALLEN  WARD  MARTIN  TURNER  JAMES  --3레벨
250                   |          |
251                ADAMS
                   SMITH
                       --4레벨
252
253       SELECT ENAME, LEVEL, EMPNO, MGR
254       FROM EMP
255       START WITH ENAME = 'KING'
256       CONNECT BY PRIOR EMPNO = MGR;
257
258           --LPAD 함수를 이용해서 계층적으로 조회하기
259       SELECT LPAD('  ', 3 * LEVEL - 3) || ENAME NAME, LEVEL, EMPNO, MGR
260           FROM EMP
261       START WITH ENAME = 'KING'
262       CONNECT BY PRIOR EMPNO = MGR;
263
264       COL DNAME FOR A15
265       COL LOC FOR A10
266       SELECT LPAD('  ', 3 * LEVEL - 3) || ENAME NAME, LEVEL, E.EMPNO, MGR, DNAME, LOC
267       FROM EMP E, DEPT D
268       WHERE E.DEPTNO = D.DEPTNO
269       START WITH ENAME = 'KING'
270       CONNECT BY PRIOR EMPNO = MGR;
```

```
*/

REM 계층형 쿼리 의사칼럼들
--1. CONNECT_BY_ROOT
  --1)루트 노드 찾기
/*
    COL ITEM_NAMES FOR A20
    SELECT LPAD('  ', 2 * (LEVEL - 1)) || ITEM_NAME ITEM_NAMES,
           CONNECT_BY_ROOT  ITEM_ID  ROOT_ID,
           CONNECT_BY_ROOT  ITEM_NAME  ROOT_NAME
    FROM BOM
    WHERE LEVEL >= 2
    START WITH PARENT_ID IS NULL
    CONNECT BY PRIOR ITEM_ID = PARENT_ID;

    SELECT LPAD('  ', 2 * (LEVEL - 1)) || ITEM_NAME ITEM_NAMES,
           CONNECT_BY_ROOT  ITEM_ID  ROOT_ID,
           CONNECT_BY_ROOT  ITEM_NAME  ROOT_NAME
    FROM BOM
    START WITH ITEM_ID = 1002
    CONNECT BY PRIOR ITEM_ID = PARENT_ID;
*/

--2. CONNECT_BY_ISCYCLE
  --1) 중복 참조값 찾기
  --2) 해당 ROW 의 항목이 자식 노드를 갖고 있는데 동시에 그 자식노드가 다시
  부모노드 인지를 판별하는 의사칼럼
/*
    SELECT ITEM_ID, LEVEL, LPAD('  ', 2 * (LEVEL - 1 )) || ITEM_NAME ITEM_NAMES
    FROM BOM
    START WITH ITEM_ID = 1005
    CONNECT BY PRIOR ITEM_ID = PARENT_ID;

    UPDATE BOM
    SET PARENT_ID = 1010
    WHERE ITEM_ID = 1005;

    --마더보드의 상위품목은 원래 1002 인 본체인데, 잘못 수정해서 그래픽 장치를
    상위품목으로 바꾼것이다. 이렇게 되면 마더보드의 상위품목은 그래픽 장치가 되고
    그렇게 되면 부모노드와 자식노드 간의 관계를 찾을 때 계속 루프를 돌게 된다.
    그러면 무한 루프에 걸린다.

    SELECT ITEM_ID, LEVEL, LPAD('  ', 2 * (LEVEL - 1 )) || ITEM_NAME ITEM_NAMES
    FROM BOM
    START WITH ITEM_ID = 1005
    CONNECT BY PRIOR ITEM_ID = PARENT_ID;

    --이럴때 오라클은 아래와 같은 에러를 발생시킨다.
    ERROR REPORT :
    SQL Error : ORA-01436 : CONNECT BY 의 루프가 발생했습니다.
*/
  --3)계층형 쿼리는 루프를 돌면서 항목 간에 부자 관계를 찾아내게 되는데, 두 항목의
  부모 노드값이 서로를 가리키고 있음으로 오류를 발생한 것이다.
  --4)이럴 때 CONNECT_BY_ISCYCLE 을 사용한다. 이것은 어느 항목이 루프를 발생시키는
  지 찾기 위한 의사칼럼이다.
  --5)제한사항이 있다.
  --6)CONNECT_BY_ISCYCLE 은 반드시 CONNECT BY 절에 NOCYCLE 이 명시되어 있어야 한다.
/*
    SELECT ITEM_ID, LEVEL, LPAD('  ', 2 * (LEVEL - 1 )) || ITEM_NAME ITEM_NAMES
    FROM BOM
    START WITH ITEM_ID = 1005
    CONNECT BY NOCYCLE PRIOR ITEM_ID = PARENT_ID;
    --NOCYCLE 을 사용하면 오류를 발생하지 않는다.

    SELECT  ITEM_ID, LEVEL, CONNECT_BY_ISCYCLE CYCLES,
              LPAD('  ', 2 * (LEVEL - 1 )) || ITEM_NAME ITEM_NAMES
    FROM BOM
    START WITH ITEM_ID = 1005
    CONNECT BY PRIOR ITEM_ID = PARENT_ID;

    --결과에서 CYCLES 가 1이면 바로 거기서 오류의 시작이다.
*/
```

```
--3. CONNECT_BY_ISLEAF
  --1)계층형 쿼리에서 해당 ROW 가 리프노트인지 여부를 체크하는 의사칼럼
  --2)리프노드이면 1, 아니면 0을 리턴
/*
    SELECT  ITEM_ID, LEVEL, CONNECT_BY_ISLEAF  LEAFS,
                LPAD('  ', 2 * (LEVEL - 1)) || ITEM_NAME ITEM_NAMES
    FROM BOM
    START WITH PARENT_ID IS NULL
    CONNECT BY PRIOR ITEM_ID = PARENT_ID;

    SELECT ENAME, CONNECT_BY_ISLEAF ISLEAF,LEVEL
    FROM EMP
    START WITH ENAME = 'KING'
    CONNECT BY PRIOR EMPNO = MGR;
*/

--4. SYS_CONNECT_BY_PATH 함수
  --1)경로 찾아가기
  --2)계층형 쿼리에서만 사용가능
  --3)루트노드로부터 해당 로우의 항목까지의 경로를 반환하는 함수
  --4)Syntax
    SYS_CONNECT_BY_PATH (COLUMN, CHAR)
/*
    SELECT ITEM_ID, LEVEL, SYS_CONNECT_BY_PATH(ITEM_ID, '/') ID_PATH,
                SYS_CONNECT_BY_PATH(ITEM_NAME, '/') NAME_PATH
    FROM BOM
    START WITH PARENT_ID IS NULL
    CONNECT BY PRIOR ITEM_ID = PARENT_ID;

    COL "Path" FOR A40
    SELECT ENAME, SYS_CONNECT_BY_PATH(ENAME, '/') AS "Path"
    FROM EMP
    START WITH ENAME = 'KING'
    CONNECT BY PRIOR EMPNO = MGR;
*/
```