Java Web Programming 입문 17

(Oracle PL/SQL #09)

오늘의 키워드

- Object
  - VIEW
  - INDEX
  - SEQUENCE
  - SYNONYM
- CONSTRAINT(제약조건)
  - Integrity
  - Primary Key
  - Not Null
  - Unique
  - Check
  - Foreign Key
  - Tip
- Etc…

Object

- VIEW
  - 생성

```
SQL> CREATE VIEW EMPVW30
          AS SELECT EMPNO, ENAME, SAL
               FROM EMP
              WHERE DEPTNO = 30;
```

```
-- SYSTEM 계정으로 접속
SQL> CONN SYSTEM/ORACLE
```

```
-- SCOTT에게 View를 생성할수 있는 권한을 허가
SQL> GRANT CREATE VIEW TO SCOTT;
```

```
-- 다시 SCOTT 접속
SQL> CONN SCOTT/TIGER
```

```
SQL> DESC EMPVW30
```

```
SQL> SELECT *
       FROM EMPVW30;
```

Object

- VIEW
  - Data Dictionary

```
SQL> DESC USER_VIEWS
```

```
SQL> SELECT VIEW_NAME, TEXT
       FROM USER_VIEWS;
```

  - 사실상…

```
SQL> SELECT *
       FROM EJPVW30;
```

```
SQL> SELECT *
       FROM (SELECT EMPNO, ENAME, SAL
               FROM EMP
              WHERE DEPTNO = 30);
```

  - 그럼 쓸 이유가?

```
1) 보안 효과 (Table을 직접 조회하지 못하게)
2) Query Access 가 단순화
```

Object

- VIEW
  - View 에 INSERT ?

```
SQL> INSERT INTO EMPVW30
     VALUES(1111,'홍길동', 3000, 30);
```

```
SQL> SELECT *
       FROM EMPVW30;
```

```
SQL> SELECT *
       FROM EMP;
```

  - View 의 수정

```
SQL> ALTER VIEW ...(후략)
```

```
SQL> CREATE OR REPLACE VIEW EMPVW30
         AS SELECT EMPNO, ENAME,SAL,DEPTNO
              FROM EMP
             WHERE DEPTNO=30;
```

Object

- INDEX
  - 생성

```
SQL> CREATE INDEX IDX_EMP_ENAME
          ON EMP(ENAME);
```

```
SQL> SELECT EMPNO, ENAME, SAL
       FROM EMP
      WHERE ENAME = 'SCOTT';
```

  - Data Dictionary

```
SQL> DESC USER_INDEXES
```

```
SQL> SELECT INDEX_NAME, INDEX_TYPE
       FROM USER_INDEXES;
```

Object

- INDEX
  - 삭제

```
SQL> DROP INDEX INDEX_NAME
```

```
-- 일반 Index 지우기
SQL> DROP INDEX IDX_EMP_ENAME;
```

  - 제약조건(Constraint)으로 인해 자동 생성된 INDEX 의 삭제는 까다롭다.

```
-- 바로는 못지움
SQL> DROP INDEX IDX_EMP1_EMPNO;
```

```
-- 제약조건을 지워버리면 해당 Index도 함께 삭제
SQL> DROP EMP1_EMPNO_PK;
```

```
SQL> ALTER TABLE EMP1
       DROP CONSTRAINT EMP1_EMPNO_PK;
```

Object

- SEQUENCE
  - 생성

```
SQL> CREATE SEQUENCE DEPT_DEPTNO
         START WITH    1
         INCREMENT BY  1
         MINVALUE      1
         MAXVALUE      100
         NOCACHE
         NOCYCLE;
```

```
- START WITH              : Sequence의 시작번호
- INCREMENT BY            : Sequence의 증가량
- MINVALUE/MAXVALUE       : Sequence의 최소/최대값
- NOCYCLE                 : MAXVALUE 도달후 순환하지 않음
  CYCLE                   : MAXVALUE 도달후 MINVALUE 값으로 순환
- NOCACHE                 : Chache 사용안함
  CHACHE                  : Chache 사용, Chahe는 임시저장소
                            CHACHE 20 으로 숫자를 지정해놓으면
                            한꺼번에 다음 차수 20개를 뽑아
                            미리 Memory에 올려놓겠다는 의미
```

Object

- SEQUENCE
  - 생성했으니 사용 해보자

```
SQL> INSERT INTO DEPT
     VALUES(DEPT_DEPTNO.NEXTVAL, 'A', 'B');
```

```
SQL> INSERT INTO DEPT
     VALUES(DEPT_DEPTNO.NEXTVAL, 'C', 'D');
```

```
SQL> INSERT INTO DEPT
     VALUES(DEPT_DEPTNO.NEXTVAL, 'E', 'F');
```

  - Data Dictionary

```
SQL> SELECT *
       FROM USER_SEQUENCES;
```

  - 변경은?

```
ALTER SEQUENCE [SEQUENCE명]
               [바꾸고싶은 OPTION]
```

START WITH 값은 변경 불가

```
SQL> ALTER SEQUENCE DEPT_DEPTNO
       CYCLE
       CACHE 20;
```

Object

- Synonym
  - 생성

```
SQL> CREATE TABLE COPY_EMP1
          AS SELECT *
               FROM EMP;
```

```
SQL> CREATE SYNONYM CE
          FOR COPY_EMP1;
```

  - 권한부여

```
SQL> CONN SYSTEM/ORACLE
```

```
SQL> GRANT CREATE SYNONYM
          TO SCOTT;
```

  - 비교

```
SQL> SELECT *
       FROM CE;
```

```
SQL> SELECT *
       FROM COPY_EMP1;
```

  - 삭제

```
SQL> DROP SYNONYM CE;
```

  - Data Dictionary

```
SQL> SELECT *
       FROM USER_SYNONYMS
```

CONSTRAINT(제약조건)

- EMP, DEPT Table

EMP Table

EMPNO	ENAME	SAL	DEPTNO
문자(X) NULL(X) 중복(X)	NULL(X) 중복(O)	0	DEPT Table의 DEPTNO를 참조

DEPT Table

DEPTNO	DNAME	LOC
NULL(X) 중복(X)	고유값	

- 종류

- Primary Key : NULL(X), 중복값(X)
- NOT NULL : NULL(X)
- Unique : 고유한 값 (중복값(X))
- CHECK : Business Logic Check
  (ex: 급여의 범위는 500~5000 사이 등)
- Foreign Key : 타 Table에 Column 값을 현재 나(Table)와 관련지어 사용
  (외국에서 온 대사관?)

CONSTRAINT(제약조건)

- 생성 시기

```
- Table 최초 생성시 Option
- 이미 생성되어있는 Table에 추가/수정
```

```
SQL> CREATE TABLE DEPT1 (
        DEPTNO   NUMBER(2)        PRIMARY KEY      ,
        DNAME    VARCHAR2(20)     UNIQUE           ,
        LOC      VARCHAR2(20)
      );
```

```
- DEPTNO : 두자리 숫자 Data Type, PRIMARY KEY 로 사용
- DNAME  : UNIQUE (중복값(X))
```

CONSTRAINT(제약조건)

- 생성방법

```
SQL> CREATE TABLE EMP1 (
        EMPNO    NUMBER(4)        CONSTRAINT EMP1_EMPNO_PK         PRIMARY KEY      ,
        ENAME    VARCHAR2(20)                                      NOT NULL         ,
        SAL      NUMBER(7,2)                                                        ,
        DEPTNO   NUMBER(2)                                                          ,
        CONSTRAINT EMP1_SAL_CK
                 CHECK (SAL BETWEEN 500 AND 5000)                                   ,
        CONSTRAINT EMP1_DEPTNO_FK
                 FOREIGN KEY (DEPTNO) REFERENCES DEPT1(DEPTNO)
     );
```

```
------------------------------
  1. COLUMN LEVEL 정의 방식
------------------------------

  - [Column명] [DataType] [제약조건종류]
  - DEPTNO     NUMBER(2)  PRIMARY KEY

FM

  - [Column명]  [DataType]       [CONSTRAINT 제약조건명]          [제약조건종류]
  - EMPNO       NUMBER(4)        CONSTRAINT EMP1_EMPNO_PK        PRIMARY KEY
```

CONSTRAINT(제약조건)

- 생성방법

```
SQL> CREATE TABLE EMP1 (
       EMPNO    NUMBER(4)        CONSTRAINT EMP1_EMPNO_PK        PRIMARY KEY      ,
       ENAME    VARCHAR2(20)                                     NOT NULL         ,
       SAL      NUMBER(7,2)                                                       ,
       DEPTNO   NUMBER(2)                                                         ,
       CONSTRAINT EMP1_SAL_CK
                CHECK (SAL BETWEEN 500 AND 5000)                                  ,
       CONSTRAINT EMP1_DEPTNO_FK
                FOREIGN KEY (DEPTNO) REFERENCES DEPT1(DEPTNO)
     );
```

```
------------------------------
  2. TABLE LEVEL 정의 방식
------------------------------

  [Column명1]    [DataType]                                                  ,
  [Column명2]    [DataType]                                                  ,
  .
  .
  .
  [Column명N]    [DataType]                                                  ,
  CONSTRAINT     [제약조건명]    [제약조건종류]  [(Column명1)]    ,
  CONSTRAINT     [제약조건명]    [제약조건종류]  [(Column명2)]    ,
```

CONSTRAINT(제약조건)

- Foreign Key
  - 생성

```
... (전략)
CONSTRAINT EMP1_DEPTNO_FK
        FOREIGN KEY (DEPTNO)
        REFERENCES DEPT1(DEPTNO));
```

  - 생성 후 INSERT

```
SQL> INSERT INTO DEPT1
     VALUES(1, 'A', 'B');
```

```
SQL> INSERT INTO DEPT1
     VALUES(1, 'B', 'C');
```

  - 참조하는 쪽에서의 INSERT

```
SQL> INSERT INTO EMP1
     VALUES (1111, '홍길동2', 3000, 1);
```

```
SQL> INSERT INTO EMP1
     VALUES (1111, '홍길동2', 3000, 2);
```

CONSTRAINT(제약조건)

- Foreign Key

```
SQL> DELETE DEPT1
      WHERE DEPTNO = 1;
```

```
SQL> DROP TABLE DEPT1;
```

```
SQL> CREATE TABLE EMP1
       (
        EMPNO  NUMBER(4) CONSTRAINT EMP1_EMPNO_PK PRIMARY KEY          ,
        ENAME  VARCHAR2(20) NOT NULL                                   ,
        SAL    NUMBER(7,2)                                             ,
        DEPTNO NUMBER(2)                                               ,
        CONSTRAINT EMP1_SAL_CK
               CHECK (SAL BETWEEN 500 AND 5000)                        ,
        CONSTRAINT EMP1_DEPTNO_FK
               FOREIGN KEY (DEPTNO) REFERENCES DEPT1(DEPTNO)
               ON DELETE CASCADE
       );
```

CONSTRAINT(제약조건)

- Foreign Key

```
SQL> INSERT INTO DEPT1 VALUES (1, 'A', 'B');
```

```
SQL> INSERT INTO DEPT1
     VALUES (1, 'A', 'B');
```

```
SQL> INSERT INTO EMP1
     VALUES (1111, '홍길동2', 3000, 1);
```

```
SQL> DELETE DEPT1
      WHERE DEPTNO = 1;
```

CONSTRAINT(제약조건)

- Tip
  - 종류

```
UNIQUE      : NULL 을 제외한 고유값(중복불가)
NOT NULL    : NULL은 안되지만 중복은 가능
PRIMARY KEY : NOT NULL + UNIQUE
```

  - 현재 계정이 생성한 Constraint

```
SQL> DESC USER_CONSTRAINTS
```

```
SQL> SELECT CONSTRAINT_NAME     ,
            CONSTRAINT_TYPE     ,
            TABLE_NAME          ,
            SEARCH_CONDITION
       FROM USER_CONSTRAINTS
      WHERE TABLE_NAME IN ('EMP1', 'DEPT1');
```

```
- P : PRIMARY KEY
- U : UNIQUE
- C : CHECK
- R : REFERENCE
```

CONSTRAINT(제약조건)

- Tip
  - 생성이 되어있는 Table내 Constraint 추가/삭제
    - 추가

```
SQL> ALTER TABLE DEPT1
       ADD CONSTRAINT DEPT1_LOC_UK UNIQUE (LOC);
```

    - 삭제

```
SQL> ALTER TABLE DEPT1
      DROP CONSTRAINT DEPT1_LOC_UK;
```

    - 추가

```
SQL> ALTER TABLE DEPT1
       ADD CONSTRAINT DEPT1_LOC_NN NOT NULL (LOC);
```

```
SQL> ALTER TABLE DEPT1
     MODIFY (LOC VARCHAR2(20) NOT NULL);
```

Etc…

- DQL (Data Query Language)
  - SELECT : Table 조회
- DML (Data Manipulation Language)
  - INSERT : Row 추가
  - UPDATE : Row 변경
  - DELETE : Row 삭제
- TCL (Transaction Control Language)
  - COMMIT : Database 발생한 작업 저장, Transaction 종료
  - ROLLBACK : 이전 Commit 전 단계까지의 작업 취소
  - SAVEPOINT : Transaction Rollback의 시점 지정
- DDL (Data Definition Language)
  - CREATE : Object 생성
  - ALTER : Object 변경
  - DROP : Object 삭제
  - RENAME : Object 이름변경
  - TRUNCATE : Table Data 완전삭제
  - COMMENT : Table/Column 설명작성
- DCL (Data Control Language)
  - GRANT : 권한 부여
  - REVOKE : 권한 회수

Etc…

- PL/SQL
  - SQL 한계 극복
    - 반복 처리 (Loop)
    - 비교처리 (IF)
    - 에러처리
    - SQL문 캡슐화
    - 변수 선언

```
CREATE PROCEDURE

CREATE FUNCTION

CREATE PACKAGE

CREATE TRIGGER
```

```
DECLARE

BEGIN

EXCEPTION

END;
```

Etc…

- PL/SQL
  - Anonymous Procedure
    - 반복적인 SQL 실행을 미리 작성
    - DB에 저장되지 않음
    - [DECLARE]
  - Stored Procedure
    - 생성 후 데이터베이스에 저장
    - [CREATE PROCEDURE procedure_name]
    - 로직 처리 후 종료
  - Stored Function
    - 생성 후 데이터베이스에 저장
    - 로직 처리 후 결과를 리턴
  - Package
    - Procedure / Function Folder
  - Trigger
    - 설정 테이블에 DML 실행 전/후에 필요한 작업을 실행
    - 데이터베이스 감시, 보안, 연속적 작업 수행 자동처리 등
  - Object-Type
    - 객체 옵션이 제공되는 데이터베이스 내 객체 데이터 입력, 수정, 삭제, 조회

프 로 시 저	함 수
PL/SQL 문장으로 실행 (EXECUTE 명령문)	표현식의 일부로 호출 또는 Execute로 실행 (SELECT문의 Seletct-List, WHERE절과 HAVING절, CONNECT BY, START WITH절, INSERT문의 VALUES절, UPDATE문의 SET절)
RETURN 데이터형이 없음	하나의 RETURN 데이터형이 존재
하나 이상의 값을 반환	하나의 값만 반환

Etc…

- PL/SQL
  - Stored Procedure

```
CREATE SEQUENCE SQNC_DEPTNO
START WITH 60;
```

```
CREATE OR REPLACE PROCEDURE add_dept
( v_dname IN dept.dname%TYPE DEFAULT 'undecided',
--v_dname IN varchar2
  v_loc IN dept.loc%TYPE  DEFAULT 'undecided')
IS
BEGIN
 INSERT INTO dept
 VALUES (SQNC_DEPTNO.NEXTVAL, v_dname, v_loc);
END add_dept;
```

```
BEGIN
 add_dept('전산과', '서울');
 add_dept(v_loc => '부산', c_dname => '포워딩');
 add_dept('인사과', v_loc => '인천');
 add_dept(v_loc => '포항');
END;
/
```

Etc…

- PL/SQL
  - Stored Procedure

```
CREATE OR REPLACE PROCEDURE name_sal
( v_empno IN emp.empno%TYPE,
  v_ename OUT emp.ename%TYPE,
  v_sal OUT emp.sal%TYPE)
IS
BEGIN
  SELECT ename, sal
    INTO v_ename, v_sal
    FROM emp
   WHERE empno = v_empno;
END name_sal;
/
```

```
VARIABLE emp_name varchar2(15)
VARIABLE emp_sal NUMBER
```

```
EXECUTE name_sal(7900, :emp_name, :emp_sal);
```

```
PRINT emp_name;

PRINT emp_sal;
```

Etc…

- PL/SQL
  - IF

```
DECLARE
v_var NUMBER :=1;
BEGIN
 IF v_var = 1 THEN
  DBMS_OUTPUT.PUT_LINE('데이터 값은 1');
 END IF;
END;
/
```

```
DECLARE
v_var NUMBER :=2;
BEGIN
 IF v_var = 1 THEN
  DBMS_OUTPUT.PUT_LINE('데이터 값은 1');
 ELSE
  DBMS_OUTPUT.PUT_LINE('데이터의 값은 1 아님');
 END IF;
END;
/
```

```
DECLARE
v_var NUMBER :=2;
BEGIN
 IF v_var = 1 THEN
  DBMS_OUTPUT.PUT_LINE('데이터 값은 1');
 ELSIF v_var > 1 THEN
  DBMS_OUTPUT.PUT_LINE('데이터 값은 1보다 큼');
 ELSE
  DBMS_OUTPUT.PUT_LINE('데이터의 값은 1 아님');
 END IF;
END;
/
```

Etc…

- PL/SQL
  - Loop

```
DECLARE
 v_count nuymber :=0;
BEGIN
  LOOP
      v_count := v_count + 1;
      DBMS_OUTPUT.PUT_LINE(v_count);
      EXIT WHEN v_count = 10;
  END LOOP;
END;
/
```

```
DECLARE
    v_num NUMBER;
BEGIN
    FOR v_num IN 1..10 LOOP
        IF(MOD(v_num, 2) = 1 THEN
            DBMS_OUTPUT.PUT_LINE(v_num);
        END if;
    END LOOP;
END;
/
```

Etc…

- PL/SQL
  - Stored Function

```
CREATE OR REPLACE FUNCTION tax
(v_sal IN NUMBER)
RETURN NUMBER
IS
BEGIN
   RETURN (v_sal * 0.08);
END tax;
/
```

```
VARIABLE my_tax number;

EXECUTE :my_tax := tax(100);

PRINT my_tax;
```

Etc…

- Listener
- Package
- Cursor
  - Open, Fetch, Close
- DCL (Data Control Language)
  - Grant, Revoke
- Row Migration
- Trigger
- Redo Log File
  - (No)Archive Log Mode
- Data File
- ERD
- Backup & Recovery
  - exp, imp, Etc…
- Recovery VS Restore
- SQL Tunning
- Oracle Memory Structure
  - SGA(System Global Area)
  - Server Process
  - Background Process
  - Etc…

오늘 숙제

- 복습
  - Java
  - Oracle
- 예습
  - HTML4
  - JSP