Oracle DBMS

<객체 = Object>
객체(Object)는 속성(attribute)과 동작(behavior)을 가지고 있는 개체(entity)를 의미

Table = Object
Object 속 내용이 Data
<CRUD Table & Data 참고>

데이터 사전(Data Dictionary) - 아래에 테이블에 들어있음
인덱스(Index)
뷰(View)
시퀀스(Sequence)
동의어(SYNONYM)

Oracle DB의 Table은 사용자 테이블과 데이터 사전으로 나뉜다.
테이블을 누르면 열의 이름을 알 수 있다.
테이블을 만들게 되면 별도로 데이터 사전이 만들어져있다. 배경에 숨겨져 있음.
USER, ALL, DBA, V$로 표시하여 데이터 사전 뷰를 본다.

<데이터 사전 보기>
SELECT * FROM DICT;
또는
SELECT * FROM DICTIONARY;

TABLE NAME(테이블 이름)과 COMMENTS(주석)이 있음.
Oracle에서 자체적으로 생성을 한다.

<데이터 사전 객체정보 보기>
SELECT TABLE_NAME FROM USER_TABLES;
<계정이 사용할 수 있는 객체(테이블) 보기>
SELECT OWNER, TABLE_NAME FROM ALL_TABLES;

SYS = 관리자의 의미, 시스템에서 관리를 한다.
DUAL = 임시 테이블

AI와 프로그램은 만들어져있는 것들 잘 활용하여 프로그래밍을 하면 좋음.

<DBA - 시스템(DB 관리자)에서 확인하여야 함>
SELECT * FROM DBA_TABLES;

DBA = DataBase Administrator(데이터베이스 관리자)의 줄임말
보안환경 - 관리를하면 한다.
Windows, Linux, Mac, UNIX의 운영체제가 있음.
Windows - Administrator 나머지 운영체제에서 Administrator는 root라는 이름으로 사용
시스템이라는 이름으로 관리자의 테이블이 System 계정이다.
이 명령어는 SCOTT에서는 에러가 걸린다.
정보처리(기능사, 산업기사, 기사) 실기 시험용으로 oracle 11g 버전을 사용하는 편이다.

튜토리얼(Tutorial) : 따라해보기 - 입문서

<INDEX : 색인>
참고할 특정한 내용을 찾기 위해서 사용한다.

<인덱스 정보 보기>
SELECT * FROM USER_INDEXES;
GUI에서는 더블클릭

<인덱스 생성>
CREATE INDEX IDX_EMP_SAL ON EMP(SAL);
IDX = INDEX

<생성한 인덱스 확인 및 삭제 하기>
SELECT * FROM USER_IND_COLUMNS;
GUI 인덱스 - 새 인덱스 - 인덱스 이름 붙여서 생성(고유값은 없음(Primary Key가 아님), 조금 복잡하니 CLI 사용 권장)
삭제할 때 DROP INDEX로 사용한다.
DROP INDEX IDX_EMP_SAL;
※ INDEX도 Object이기 때문에 테이블 CRUD 명령어와 똑같이 적용하면 됨.(데이터는 없기 때문에 Trancated는 안 씀)

<뷰(View)>
가상 테이블 = Virtual Table = View
Select로 일일이 테이블을 만들기보다 임시로 보여야 하는 경우도 있음
View로 생성하면 편하다.

<View 생성하기 - 시스템에서 권한 부여>
View에는 권한 부여를 해야 View를 쓸수 있음.
시스템에 Grant 자격(권한)부여 명령어
SYSTEM에서 GRANT CREATE VIEW TO SCOTT;
권한 부여를 해줘야 View를 생성 할 수 있다.
권한 부여 성공 - Grant을(를) 성공했습니다.

Grant ↔ Revoke(권한 제거)

<권한 제거 - 시스템에서>
REVOKE CREATE VIEW FROM SCOTT;

<시스템에서 View 권한 주지 않고 실행시>
CREATE VIEW VW_EMP20
    AS (SELECT EMPNO, ENAME, JOB, DEPTNO
          FROM EMP
         WHERE DEPTNO = 20);
사용시 아래같이 오류가 난다.
오류 보고 -
ORA-01031: insufficient privileges
01031. 00000 - "insufficient privileges"
*Cause: An attempt was made to perform a database operation without
the necessary privileges.
*Action: Ask your database administrator or designated security
administrator to grant you the necessary privileges
ORA-01031 에러코드 Insufficient privileges(충분하지 않은 권한 : 즉, 권한 없음)이라고 뜬다.
Cause, Action(원인과 해결 요령법)을 알려준다.

권한을 주는 이유 : 테이블 정보를 함부로 보지 않게 하기 위해서 권한을 준다.

<뷰 생성>
CREATE VIEW VW_EMP20
    AS (SELECT EMPNO, ENAME, JOB, DEPTNO
          FROM EMP
         WHERE DEPTNO = 20);
VW = View
권한을 주고 뷰를 통해서 임시 테이블을 생성이 되었음.
뷰 - 새 뷰 - 이름 정해주고 서브쿼리 안 내용 복사 하면 GUI에서 뷰 간단하게 생성된다.

<뷰 조회>
SELECT * FROM VW_EMP20;
<생성한 뷰 내용 확인하기 - Data Dictionary 확인하기>
SELECT VIEW_NAME, TEXT_LENGTH, TEXT FROM USER_VIEWS;

<뷰 삭제>
DROP VIEW VW_EMP20;

<ROWNUM - 테이블 번호 주기>

인라인 뷰란?
Inline view는 SQL 쿼리의 일부로서 새로운 가상 테이블을 만들고 이를 쿼리에 삽입하여 사용하는 것을 말합니다. 즉, 서브쿼리와 비슷하지만, 서브쿼리를 실행하여 결과를 반환한 후 이를 다시 쿼리에 삽입하는 대신, 인라인 뷰에서는 새로운 가상 테이블을 생성하여 이를 바로 쿼리에 삽입하여 사용합니다.

인라인 뷰는 특히 여러 테이블로부터 데이터를 추출하고 조합해야 하는 복잡한 쿼리에서 유용합니다. 이를 통해 중첩된 서브쿼리를 대신할 수 있으며, 가독성이 향상되고 성능도 높아질 수 있습니다.

<인라인 뷰의 이해>
```
SELECT  E.* FROM EMP E; - EMP 테이블 전체 보여줌.

SELECT ROWNUM, E.*
  FROM EMP E;
```
ROWNUM(없는 컬럼인데 뷰에 넣어서 EMP 테이블을 보이는 용도로 사용)

<EMP 테이블을 컬럼 기준으로 테이블 정렬>
```
SELECT ROWNUM, E.*
  FROM EMP E
  ORDER BY SAL DESC;
```

<인라인 뷰(서브쿼리 활용>
```
SELECT ROWNUM, E.*
  FROM (SELECT *
          FROM EMP E
        ORDER BY SAL DESC) E;
```
연봉이 높은 순서대로 정렬하여 테이블을 보여주게 된다.

<WITH절 사용하여 Inline View 활용>
```
WITH E AS (SELECT * FROM EMP ORDER BY SAL DESC)
SELECT ROWNUM, E.*
  FROM E;
```

서브쿼리 = With 사용(결과 값이 같다.)

<인라인 뷰로 Top-N(연봉 상위 몇 위 등 형태) 추출하기 - 서브 쿼리, With절 사용>
```
SELECT ROWNUM, E.*
  FROM (SELECT *
          FROM EMP E
```

```
        ORDER BY SAL DESC) E
 WHERE ROWNUM <= 3;

WITH E AS (SELECT * FROM EMP ORDER BY SAL DESC)
SELECT ROWNUM, E.*
  FROM E
 WHERE ROWNUM <= 3;
```

<Sequence - 순번 생성>
Sequence - 순서가 있음.

<DEPT 활용하여 DEPT_SEQUANCE Table 생성>

```
CREATE TABLE DEPT_SEQUENCE
    AS SELECT *
         FROM DEPT
        WHERE 1 <> 1;
```

<시퀀스 생성>

```
CREATE SEQUENCE SEQ_DEPT_SEQUENCE
   INCREMENT BY 10
   START WITH 10
   MAXVALUE 90
   MINVALUE 0
   NOCYCLE
   CACHE 2;
```

SEQ = SEQUENCE

GUI로 생성 - 시퀀스 마우스 오른쪽 - 새 시퀀스

<생성된 시퀀스 확인>
SELECT * FROM USER_SEQUENCES;
<시퀀스 사용>
<순번을 사용하여 Insert문 실행>
INSERT INTO DEPT_SEQUENCE (DEPTNO, DNAME, LOC)
VALUES (SEQ_DEPT_SEQUENCE.NEXTVAL, 'DATABASE', 'SEOUL');
SELECT * FROM DEPT_SEQUENCE ORDER BY DEPTNO;
★ SEQ_DEPT_SEQUENCE.NEXTVAL : 테이블 객체.속성의 형태로 사용(객체의 다른 속성을 정의하여 불러들임.)
- 순서 자동 매김

<마지막에 넣은 시퀀스를 확인>
CURRVAL(Current Value)를 사용
SELECT SEQ_DEPT_SEQUENCE.CURRVAL
  FROM DUAL;

<시퀀스 반복 사용>
F9를 사용하여 이용(본래는 PL/SQL에서 하면 됨)
9번까지 사용된다.
INSERT INTO DEPT_SEQUENCE (DEPTNO, DNAME, LOC)
VALUES (SEQ_DEPT_SEQUENCE.NEXTVAL, 'DATABASE', 'SEOUL');
SELECT * FROM DEPT_SEQUENCE ORDER BY DEPTNO;

INSERT INTO DEPT_SEQUENCE (DEPTNO, DNAME, LOC)

VALUES (SEQ_DEPT_SEQUENCE.NEXTVAL, 'DATABASE', 'SEOUL')
오류 보고 -
ORA-08004: sequence SEQ_DEPT_SEQUENCE.NEXTVAL exceeds MAXVALUE and cannot be instantiated
최대값을 초과해서 만들 수 없기 때문에 그러하다.

<시퀀스 수정>
ALTER 명령어 사용
```
ALTER SEQUENCE SEQ_DEPT_SEQUENCE
   INCREMENT BY 3
   MAXVALUE 99
   CYCLE;
```
- 시퀀스 값이 수정된다.

<시퀀스 수정 확인>
```
SELECT * FROM USER_SEQUENCES;
```

<수정 후 시퀀스 적용 확인>
```
INSERT INTO DEPT_SEQUENCE (DEPTNO, DNAME, LOC)
VALUES (SEQ_DEPT_SEQUENCE.NEXTVAL, 'DATABASE', 'SEOUL');
SELECT * FROM DEPT_SEQUENCE ORDER BY DEPTNO;
```

<시퀀스 삭제>
```
DROP SEQUENCE SEQ_DEPT_SEQUENCE;
SELECT * FROM USER_SEQUENCES;
```

<동의어(SYNONYM)>
테이블 뷰 시퀀스 등 Object Name대신 사용할 수 있는 다른 Name을 부여할 때 사용.

<동의어를 사용하기 위한 권한 부여 - 권한부여 SYSTEM에서 하기>
```
GRANT CREATE SYNONYM TO SCOTT; - 동의어
GRANT CREATE PUBLIC SYNONYM TO SCOTT; - 공용 동의어
```
<EMP TABLE 동의어 생성>
```
CREATE SYNONYM E FOR EMP;
```

<동의어 활용 전체 조회>
```
SELECT * FROM E; (동의어 사용)
SELECT E.* FROM EMP E; (Alias를 사용하여 나타냄)
```

<동의어 삭제>
```
DROP SYNONYM E;
```

<제약 조건>
제약조건이란? 컬럼에 조건을 달아서 해당되는 구조의 조건을 다는 것
데이터베이스의 무결성(integrity)을 유지하기 위해 사용되는 것으로, 데이터의 일관성과 정확성을 보장하기 위해 사용됩니다. 제약 조건은 테이블의 생성 시 지정하거나, 이미 존재하는 테이블에 추가할 수 있습니다.

<NULL값 제약조건 정하기>
<테이블 생성하여 제약조건 지정하기 - NOT NULL(NULL값 허용하지 않음) 설정>

```
CREATE TABLE TABLE_NOTNULL(
    LOGIN_ID VARCHAR2(20) NOT NULL,
    LOGIN_PWD VARCHAR2(20) NOT NULL,
    TEL VARCHAR2(20)
);
```

<테이블 구조 확인>

```
DESC TABLE_NOTNULL;
```

<NOT NULL인 값에 NULL 넣어보기>

```
INSERT INTO TABLE_NOTNULL (LOGIN_ID, LOGIN_PWD, TEL)
VALUES ('TEST_ID_01', NULL, '010-1234-5678');
```

아래 같이 에러 발생함

```
오류 보고 -
SQL 오류: ORA-01400: cannot insert NULL into ("SCOTT"."TABLE_NOTNULL"."LOGIN_PWD")
01400. 00000 -  "cannot insert NULL into (%s)"
*Cause:    An attempt was made to insert NULL into previously listed objects.
*Action:   These objects cannot accept NULL value
```

<제약조건이 없는 컬럼에 NULL 입력하기>

```
INSERT INTO TABLE_NOTNULL (LOGIN_ID, LOGIN_PWD)
VALUES ('TEST_ID_01', '1234');

SELECT * FROM TABLE_NOTNULL;
```

TEL값은 NULL이 허용되서 NULL이 입력이 된다.

<NOT NULL 제약 조건이 걸린 컬럼에서 NULL 값으로 업데이트>

```
UPDATE TABLE_NOTNULL
   SET LOGIN_PWD = NULL
 WHERE LOGIN_ID = 'TEST_ID_01';
```

아래와 같이 오류 걸린다.

오류 보고 -
SQL 오류: ORA-01407: cannot update ("SCOTT"."TABLE_NOTNULL"."LOGIN_PWD") to NULL
01407. 00000 - "cannot update (%s) to NULL"
*Cause:
*Action:
%s(%String) = 어떤 문자
NULL 값으로는 업데이트를 하지 못한다.

<제약 조건 확인하기>
Data Dictornary에 제약 조건이 있다.

<제약 조건 보기>
```
SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
  FROM USER_CONSTRAINTS;
```

Table 만들었던 것이 SYS로 표기 되어 있음.

<제약 조건 이름 지정>
```
CREATE TABLE TABLE_NOTNULL2(
    LOGIN_ID VARCHAR2(20) CONSTRAINT TBLNN2_LGNID_NN NOT NULL,
    LOGIN_PWD VARCHAR2(20) CONSTRAINT TBLNN2_LGNPW_NN NOT NULL,
    TEL VARCHAR2(20)
);
```

```
SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
  FROM USER_CONSTRAINTS;
```
→ 제약 조건이 구체적으로 명시되어 보인다.(NOT NULL 제약조건 볼 수 있음)

<CONSTRAINT 안의 타입 - 제약 조건의 의미>
C (Check) : 테이블에 저장되는 데이터의 값이 지정한 조건에 맞는지 검사합니다.
R (Reference) : 외래 키 제약 조건을 설정하는데 사용되며, 다른 테이블의 레코드를 참조합니다.
P (Primary Key) : 테이블에서 하나의 레코드를 식별할 수 있는 유일한 식별자 열입니다.
FK (Foreign Key) : 다른 테이블의 P(Primary Key)에 대한 참조를 가리키는 외래 키 열입니다.
PK (Primary Key) : 테이블에서 하나의 레코드를 식별할 수 있는 유일한 식별자 열입니다. 다른 테이블의 FK(외래 키)를 참조합니다.

<제약 조건 추가>
```
ALTER TABLE TABLE_NOTNULL MODIFY(TEL NOT NULL);
```

MODIFY = 변경, 수정의 의미
오류가 뜬다.

오류 보고 -
ORA-02296: cannot enable (SCOTT.) - null values found
02296. 00000 - "cannot enable (%s.%s) - null values found"
*Cause: an alter table enable constraint failed because the table
         contains values that do not satisfy the constraint.
*Action: Obvious

아래와 같이 기존 정보를 수정해야 한다.

<UPDATE 문으로 Column Data 수정>
```
UPDATE TABLE_NOTNULL
   SET TEL = '010-1234-5678'
 WHERE LOGIN_ID = 'TEST_ID_01';

SELECT * FROM TABLE_NOTNULL;
```

// ● 먼저 UPDATE로 바꾸고 속성을 변경하면 오류가 발행하지 않는다. //

<NOT NULL 제약 조건 추가>
```
ALTER TABLE TABLE_NOTNULL
MODIFY(TEL NOT NULL);

SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
  FROM USER_CONSTRAINTS;
```

<생성한 테이블에 제약 조건 이름 직접 지정 추가>
```
ALTER TABLE TABLE_NOTNULL2
MODIFY(TEL CONSTRAINT TBLNN_TEL_NN NOT NULL);

SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
  FROM USER_CONSTRAINTS;
```

// 데이터 사전에 이름이 바꾸어져 있다. //

<테이블 구조 확인>
```
DESC TABLE_NOTNULL2;
```

<이미 생성된 제약 조건 이름 변경>
```
ALTER TABLE TABLE_NOTNULL2
RENAME CONSTRAINT TBLNN_TEL_NN TO TBLNN2_TEL_NN;
```

```
SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
  FROM USER_CONSTRAINTS;
```

<제약 조건 삭제>

```
ALTER TABLE TABLE_NOTNULL2
 DROP CONSTRAINT TBLNN2_TEL_NN;

DESC TABLE_NOTNULL2;
```

<UNIQUE - 특수 제약 조건>

// UNIQUE : 유일무이의 뜻 //

```
CREATE TABLE TABLE_UNIQUE(
    LOGIN_ID VARCHAR2(20) UNIQUE,
    LOGIN_PWD VARCHAR2(20) NOT NULL,
    TEL VARCHAR2(20)
);
```

// ID, PW : 모두 NULL값이 안되야 함
ID는 사용자 이름 중복되지 않게 고유한 값이어야 함. 그렇기에 UNIQUE를 활용한다. //

```
DESC TABLE_UNIQUE;
```

<CONSTRAINT(데이터 뷰)로 제약 조건 확인>

```
SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
  FROM USER_CONSTRAINTS
 WHERE TABLE_NAME = 'TABLE_UNIQUE';
```

UNIQUE는 U로 표시 되어있다.
여기서 C는 NULL 또는 NOT NULL값에 대해서 표현을 한다.

<중복을 허락하지 않는 UNIQUE - 데이터 입력 및 가공>

<데이터 입력>

```
INSERT INTO TABLE_UNIQUE(LOGIN_ID, LOGIN_PWD, TEL)
VALUES('TEST_ID_01', 'PWD01', '010-1234-5678');
```

<중복 데이터 입력>

```
INSERT INTO TABLE_UNIQUE(LOGIN_ID, LOGIN_PWD, TEL)
VALUES('TEST_ID_01', 'PWD01', '010-1234-5678');
```

오류 보고 -
ORA-00001: unique constraint (SCOTT.SYS_C007009) violated

// 에러가 걸린다. UNIQUE는 중복값을 허용하지 않음.

```
INSERT INTO TABLE_UNIQUE(LOGIN_ID, LOGIN_PWD, TEL)
VALUES('TEST_ID_02', 'PWD01', '010-1234-5678');
```

→ 아이디가 다르면 값이 정상적으로 입력이 된다.

<UNIQUE와 NULL값>

```
INSERT INTO TABLE_UNIQUE(LOGIN_ID, LOGIN_PWD, TEL)
VALUES(NULL, 'PWD01', '010-2345-6789');

SELECT * FROM TABLE_UNIQUE;
```

NULL값도 삽입이 된다.
NULL값도 UNIQUE에 포함이 된다.
ID는 NOT NULL, UNIQUE 두 가지 모두가 충족이 되어야 함
NOT NULL과 UNIQUE는 같지 않다.

<UNIQUE 제약 조건 수정>

```
UPDATE TABLE_UNIQUE
   SET LOGIN_ID='TEST_ID_01'
 WHERE LOGIN_ID IS NULL;
```

오류 보고 -
ORA-00001: unique constraint (SCOTT.SYS_C007009) violated
앞의 아이디와 중복이 되기 때문에 에러가 뜬다.

```
UPDATE TABLE_UNIQUE
   SET LOGIN_ID='TEST_ID_03'
 WHERE LOGIN_ID IS NULL;

SELECT * FROM TABLE_UNIQUE;
```

// NULL이 TEST_ID_03으로 변경된다.

<테이블 생성하면서 제약 조건 이름 직접 지정>

```
CREATE TABLE TABLE_UNIQUE2(
   LOGIN_ID VARCHAR2(20) CONSTRAINT TBLUNQ2_LGNID_UNQ UNIQUE,
   LOGIN_PWD VARCHAR2(20) CONSTRAINT TBLUNQ2_LGNPW_NN NOT NULL,
   TEL VARCHAR2(20)
);
SELECT * FROM TABLE_UNIQUE2;
```

<UNIQUE 제약 조건 확인 - Data Dictionary 확인하기>

```
SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
  FROM USER_CONSTRAINTS
 WHERE TABLE_NAME LIKE 'TABLE_UNIQUE%';
```

<UNIQUE 제약 조건 추가/변경>

```
ALTER TABLE TABLE_UNIQUE
MODIFY(TEL UNIQUE);
```

오류 보고 -

```
ORA-02299: cannot validate (SCOTT.SYS_C007012) - duplicate keys found
02299. 00000 - "cannot validate (%s.%s) - duplicate keys found"
*Cause:    an alter table validating constraint failed because the table has
           duplicate key values.
*Action:   Obvious
```

기존 정보가 UNIQUE 하지 않음으로 에러 발생한다.

```
UPDATE TABLE_UNIQUE
   SET TEL = NULL;

SELECT * FROM TABLE_UNIQUE;
```

→ NULL값으로 변경이 된다.

```
ALTER TABLE TABLE_UNIQUE
MODIFY(TEL UNIQUE);

SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
  FROM USER_CONSTRAINTS
 WHERE TABLE_NAME LIKE 'TABLE_UNIQUE%';
```

확인하면 유니크값으로 바꿔져있다.

<UNIQUE 제약 조건 이름 직접 지정>

```
ALTER TABLE TABLE_UNIQUE2
MODIFY(TEL CONSTRAINT TBLUNQ_TEL_UNQ UNIQUE);

SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
  FROM USER_CONSTRAINTS
 WHERE TABLE_NAME LIKE 'TABLE_UNIQUE%';
```

UNIQUE도 이름변경 RENAME CONSTRAINT, 삭제 DROP CONSTRAINT를 사용한다.

<PRIMARY KEY>

PRIMARY KEY - C와 U의 속성 모두 가지고 있음.(UNIQUE + NOT NULL)

DB에서 테이블에 하나 이상은 꼭 필요한 것이다. 정보 검색에 중요한 역할은 한다.

테이블 구조를 볼 때, NULLABLE에 NO로 되어 있는 테이블 PK가 있는 테이블

```
<PRIMARY KEY TABLE 생성>
CREATE TABLE TABLE_PK(
    LOGIN_ID VARCHAR2(20) PRIMARY KEY,
    LOGIN_PWD VARCHAR2(20) NOT NULL,
    TEL VARCHAR2(20)
);
DESC TABLE_PK;
```

ID 는 PRIMARY KEY UNIQUE + NOT NULL의 속성임을 알 수 있다.

```
<프라이머리 키 생성 후 데이터 딕셔너리 확인>
SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
  FROM USER_CONSTRAINTS
 WHERE TABLE_NAME LIKE 'TABLE_PK%';
```

PK를 생성하면 INDEX도 같이 자동 생성이 된다.

```
<INDEX 확인>
SELECT INDEX_NAME, TABLE_OWNER, TABLE_NAME
  FROM USER_INDEXES
 WHERE TABLE_NAME LIKE 'TABLE_PK%';
```

INDEX VALUE가 자동 등록된 것이 보인다.

```
<PK 제약조건 이름 직접 지정>
CREATE TABLE TABLE_PK2(
    LOGIN_ID VARCHAR2(20) CONSTRAINT TBLPK2_LGNID_PK PRIMARY KEY,
    LOGIN_PWD VARCHAR2(20) CONSTRAINT TBLPK2_LGNPW_NN NOT NULL,
    TEL VARCHAR2(20)
);
```

ALTER, DROP, RENAME 앞에 쓴 것 그대로 쓸 수 있음.

★ <FK - FOREIGN KEY>
PK(고유키 : PRIMARY KEY)는 하나의 테이블에서 중요한 역할을 차지한다.
FK(외래키)는 참고해야 할 키가 다른 테이블에 있다는 의미
A TABLE, B TABLE 서로 이름이 다른 경우 FK가 있는 경우가 많다.

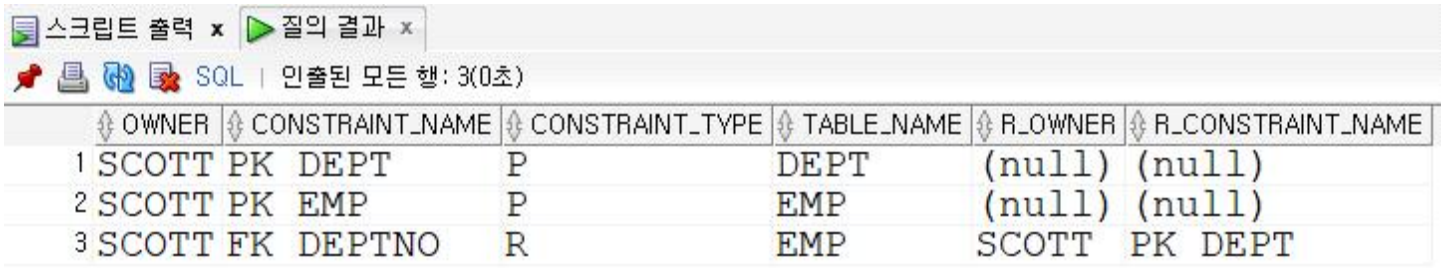
<FK와 PK의 상관관계>
EMP, DEPT DEPTNO라는 공통적인 컬럼이 있다.
테이블끼리 의존한다.
즉, 관계성이 있는 테이블에서 사용한다.(Relationship) 그렇기에 서로 상관관계가 있다.

FK에 의해 테이블이 상호관계(Relationship)를 가진다.
두 테이블을 비교하여 FK를 알아야 DB를 가져오고 제대로 만들기도 할 수 있다. - DB 설계의 핵심

DEPT DEPTNO(PK) - 고유의 값
EMP EMPNO(PK) - 고유의 값
DEPT의 DEPTNO가 연결이 되어서 두 테이블의 상관 관계를 이룬다. 즉, DEPT테이블의 DEPTNO는 FK가 된다.

<R(Relationship)의 개념>
사원정보를 관리할 때 이름으로 관리할 수 있다.
구별을 하기 위해서 사원번호를 만들어서 사용한다.
중복되지 않고 유일무이하고 명확히 구분하기 때문에 사원번호를 사용한다.
사내 조직의 조직도도 중요하다.
부서도 이름인 부서명, 부서 번호로 조직을 관리한다.
일관성있는 데이터 관리를 위해서 사원의 테이블에는 부서 번호로 연결시켜준다.
이것이 상관관계(Relationship)이라 볼 수 있다.
테이블도 각 테이블마다 역할이 다르다.
다른 테이블과 상관관계를 맺을 때 변화무쌍한 것에 상관관계를 맺는다.

```
SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME, R_OWNER, R_CONSTRAINT_NAME
  FROM USER_CONSTRAINTS
 WHERE TABLE_NAME IN ('EMP', 'DEPT');
```


스크립트 출력 | 질의 결과
SQL | 인출된 모든 행: 3(0초)

	OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	R_OWNER	R_CONSTRAINT_NAME
1	SCOTT	PK_DEPT	P	DEPT	(null)	(null)
2	SCOTT	PK_EMP	P	EMP	(null)	(null)
3	SCOTT	FK_DEPTNO	R	EMP	SCOTT	PK_DEPT

<FK 참조하는 열에 존재하지 않는 데이터 입력>
INSERT INTO EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES(9999, '홍길동', 'CLERK', '7788', TO_DATE('2017/04/30', 'YYYY/MM/DD'), 1200, NULL, 50);

입력이 불가하다.

오류 보고 -
ORA-02291: integrity constraint (SCOTT.FK_DEPTNO) violated - parent key not found
부모 키가 없다고 입력이 되지 않음.

EMP 입장에서는 집어넣으면 확인할 방법이 없어서 집어 넣을 수 있지만 DEPTNO(FK)의 테이블을 참조하여 상관관계를 따진다. 상관관계를 따졌을 때 데이터가 없기 때문에 입력이 불가.
parent : DEPTNO의 KEY VALUE를 의미