Oracle DBMS

CRUD
Create, Read, Update, Delete
Read에서 가장 많이 쓰는 것은 Select문
실질적으로 Select문을 많이 쓴다.

<Select문 구조>
select 열
from 테이블
where 조건
group by 열
order by 열
형태로 구성이 된다.

select → group by → order by 순으로 데이터 읽기를 한다.

<데이터 조작하기>
DML(Data Manipulation Language)
CRUD를 조작 제어하는 것을 DML이라고 한다.

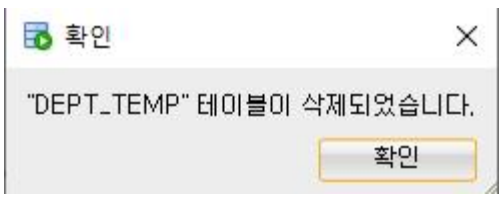
<CRUD에서 사용하는 명령어>
<Data>
Create → Insert 명령어
Read → Select 명령어
Update → Update 명령어
Delete → Delete 명령어

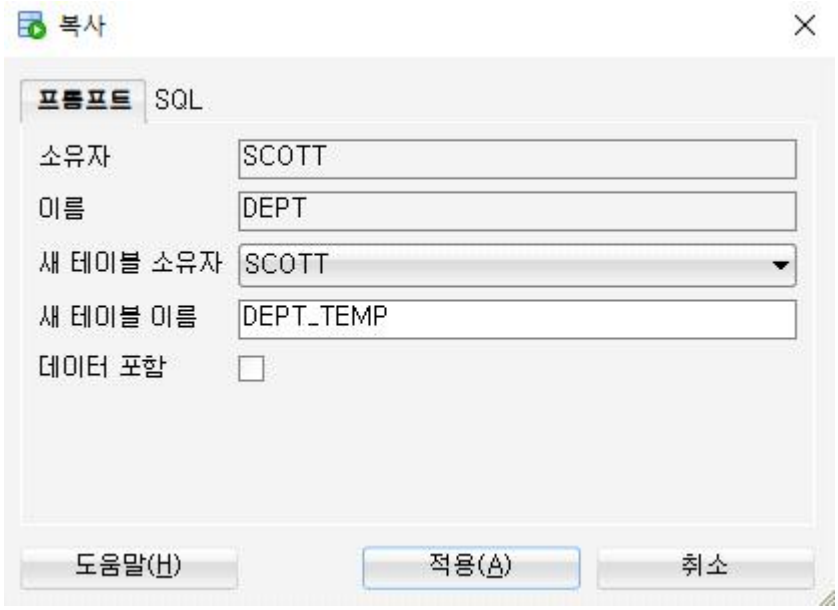
<Table>
Create → Create 명령어
Read → Select 명령어
Update → Update 명령어
Delete → Delete 명령어

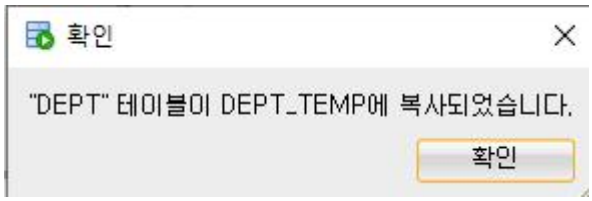
<Create 명령어 - 데이터 생성>
데이터를 생성시 데이터를 집어 넣는다. 데이터를 Insert 한다고 부름.
<테이블 생성>
```
CREATE TABLE DEPT_TEMP
    AS SELECT * FROM DEPT;
```
Select에서 AS는 Alias(별칭) 여기서 AS는 ~로 부터의 의미

실행하면 Table DEPT_TEMP이(가) 생성되었습니다. 표시가 됨.
만들고 즉시 반영이 안 되서 새로고침을 해서 동기화를 시켜주어야 함.
GUI환경에서는 아래 그림과 같이 속성 변경을 할 수 있음

테이블 - 삭제- 적용 눌러서 하게 되면 아래 같이 즉시 된다.







즉시 반영이 된다.

GUI로 하게 되면 내용까지는 복사는 안 된다.

```
CREATE TABLE DEPT_TEMP
    AS SELECT * FROM DEPT;
```

내용까지 복사가 된다.

<Table 삭제>

```
DROP TABLE DEPT_TEMP;
```

테스트 테이블(복사본 백업)을 만들고 검증이 필요함.
데이터 복구가 필요할 수 있으므로 백업이 필요함.
원본을 되도록 유지하고 수정 보완 집어넣고 보존을 해야함.
원본 수정시 백업을 반드시 만들어줘야함.
백업을 하는 것 - 버전 관리 때문에 github로 커밋하는 것

<데이터 삽입>

```
INSERT INTO DEPT_TEMP (DEPTNO, DNAME, LOC)
VALUES (50, 'DATABASE', 'SEOUL');
```

1 행 이(가) 삽입되었습니다.

```
SELECT * FROM DEPT_TEMP;
```

테이블에 데이터 삽입된 것을 확인 할 수 있음.

INSERT INTO DEPT_TEMP
VALUES (60 , 'NETWORK', 'BUSAN');
1 행 이(가) 삽입되었습니다.

SELECT * FROM DEPT_TEMP;
로우(행) 이름이 구체적으로 명시가 되어 있지 않더라도, 테이블 열의 순서대로 기본적으로 삽입이 된다.
데이터 추가 된 것이 확인이 됨.

<NULL값 포함 데이터 삽입>
※ NULL값을 넣지 못하는 경우는 삽입 불가
INSERT INTO DEPT_TEMP (DEPTNO, DNAME, LOC)
VALUES (70 , 'WEB', NULL);

INSERT INTO DEPT_TEMP (DEPTNO, DNAME , LOC)
VALUES (80 , 'MOBILE', '');
◎ '' 표시도 NULL값이다. 스페이스바 주의해야 함. 스페이스바를 입력하면 입력값이 들어간다. 스페이스바 없이 해야 NULL값이 정상적으로 입력이 됨.

INSERT INTO DEPT_TEMP (DEPTNO, LOC)
VALUES (90 , 'INCHEON');
◎ 열(컬럼) 데이터를 넣지 않는 방식으로도 NULL을 입력이 가능하긴 하다.
각각 데이터 삽입 후 SELECT * FROM DEPT_TEMP; 하면 NULL값이 들어가 있는 것 확인할 수 있다.

<날짜 데이터 삽입>
CREATE TABLE EMP_TEMP
    AS SELECT *
         FROM EMP
        WHERE 1 <> 1;
<>연산자 : !=(NOT)와 같음
조건과 같은 데이터는 없지만 열(컬럼)은 만들어진다.
테이블의 컬럼 속성만 똑같이 만들어진다.(GUI방식에서 테이블 삽입과 같다.)
SELECT * FROM EMP_TEMP;

GUI에서는 데이터 까지 같이 복사를 할 때는 데이터 포함 눌러서 복사를 하면 된다.

INSERT INTO EMP_TEMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (9999, '홍길동', 'PRESIDENT', NULL, '2001/01/01', 5000, 1000, 10);
SELECT * FROM EMP_TEMP;
컬럼값에 로우값 집어넣기.

오라클에서 날짜 포맷을 설정하는 것에 따라 달라진다.
도구 - 환경설정 - 데이터 - NLS가면 날짜 형식이 표시됨.

```
INSERT INTO EMP_TEMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (1111, '성춘향', 'MANAGER', 9999, '2001-01-05', 4000, NULL, 20);

SELECT * FROM EMP_TEMP;
```

입력을 하였는데 환경설정에서 날짜 형식을 바꾸니 날짜 포맷 형식을 바꾼 모양대로 나타난다.

<날짜 데이터 입력시 유의점>

```
INSERT INTO EMP_TEMP (EMPNO, ENAME, JOB, MGR , HIREDATE, SAL , COMM, DEPTNO)
VALUES (2111, '이순신', 'MANAGER', 9999, '07/01/2001', 4000, NULL, 20);
```

날짜 표기 방식이 달라서 데이터 포맷방식이 맞지 않아서 오류가 발생한다.

<TO_DATE를 활용하여 날짜 입력>

```
INSERT INTO EMP_TEMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (2111, '이순신', 'MANAGER', 9999,
        TO_DATE('07/01/2001', 'DD/MM/YYYY'), 4000, NULL, 20);
SELECT * FROM EMP_TEMP;
```

날짜 데이터 포맷 형식을 같이 입력하니까 날짜 데이터를 제대로 인식한다.

<SYSDATE 활용하여 날짜 데이터 입력>
회원등록, 회원가입 할 때 사용

```
INSERT INTO EMP_TEMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (3111, '심청이', 'MANAGER', 9999, SYSDATE, 4000, NULL, 30);
SELECT * FROM EMP_TEMP;
```

<서브 쿼리 사용하여 데이터 추가>

```
INSERT INTO EMP_TEMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
SELECT E.EMPNO, E.ENAME, E.JOB, E.MGR, E.HIREDATE, E.SAL, E.COMM, E.DEPTNO
  FROM EMP E, SALGRADE S
 WHERE E.SAL BETWEEN S.LOSAL AND S.HISAL
   AND S.GRADE = 1;

SELECT * FROM EMP_TEMP;
```

VALES값에 SELECT를 입력하여서 서브 쿼리를 만들었다.

insert, update, delete - commit을 해줘야 데이터 입력된 것이 보존 된다.
커밋(이전 데이터 저장), 롤백(변경하지 않고 저장)

<데이터 수정하기>
테이블 생성

```
CREATE TABLE DEPT_TEMP2
    AS SELECT * FROM DEPT;

SELECT * FROM DEPT_TEMP2;
```

<UPDATE 사용하기>

```
UPDATE DEPT_TEMP2
   SET LOC = 'SEOUL';

SELECT * FROM DEPT_TEMP2;
```

4개 행 이(가) 업데이트되었습니다. // 문구 표시
// LOC에 서울로 변경이 된다.

<UPDATE 내용을 본래 내용으로 원상복구 할때>
ROLLBACK을 활용한다.
ROLLBACK; // ROLLBACK;을 쓰고 F9 실행하면 롤백 완료.라는 문구가 보인다.

<부분적으로 데이터를 UPDATE할 때>

```
UPDATE DEPT_TEMP2
   SET DNAME = 'DATABASE',
       LOC = 'SEOUL'
 WHERE DEPTNO = 40;
```

1개 행 이(가) 업데이트되었습니다. // 문구 표시
SELECT * FROM DEPT_TEMP2; // 40 부분이 서울 바뀜

<서브쿼리 활용하여 업데이트하기>

```
UPDATE DEPT_TEMP2
   SET (DNAME, LOC) = (SELECT DNAME, LOC
                       FROM DEPT
                       WHERE DEPTNO = 40)
 WHERE DEPTNO = 40;
```

// DNAME, LOC를 DEPT 테이블 40번의 조건에서 가져와서 업데이트 해서 바꿈

또는

```
UPDATE DEPT_TEMP2
   SET DNAME = (SELECT DNAME
                FROM DEPT
```

```
                  WHERE DEPTNO = 40),
       LOC = (SELECT LOC
                FROM DEPT
               WHERE DEPTNO = 40)
 WHERE DEPTNO = 40;
```

// 서브쿼리를 활용해서 열 하나 하나 업데이트 시키는 것

```
SELECT * FROM DEPT_TEMP2;
```

새로운 내용(기존 DEPT 테이블에 있는 내용으로 업데이트) 변경이 된다.

모두 결과 같음

<WHERE절 서브쿼리 사용하여 업데이트>

```
UPDATE DEPT_TEMP2
   SET LOC = 'SEOUL'
 WHERE DEPTNO = (SELECT DEPTNO
                   FROM DEPT_TEMP2
                  WHERE DNAME='OPERATIONS');

SELECT * FROM DEPT_TEMP2;
```

// LOC 부분 서울로 변경이 되었음.

UPDATE - 데이터 변경할 때 사용한다.

<데이터 삭제 하기 - DELETE문>

EMP_TEMP2 생성

```
CREATE TABLE EMP_TEMP2
    AS SELECT * FROM EMP;

SELECT * FROM EMP_TEMP2;
```

<일부 데이터 DELETE 하기>

```
DELETE FROM EMP_TEMP2
 WHERE JOB = 'MANAGER';
```

// 3개 행 이(가) 삭제되었습니다. 문구 보임

```
SELECT * FROM EMP_TEMP2;
```

<WHERE절에 서브쿼리 사용해서 일부 데이터 삭제>

```
DELETE FROM EMP_TEMP2
 WHERE EMPNO IN (SELECT E.EMPNO
                   FROM EMP_TEMP2 E, SALGRADE S
                  WHERE E.SAL BETWEEN S.LOSAL AND S.HISAL
```

```
                    AND S.GRADE = 3
                    AND DEPTNO = 30);
```
// 2개 행 이(가) 삭제되었습니다. 문구 보임, IN 연산자 이하는 테이블 안에 것을 의미

```
SELECT * FROM EMP_TEMP2;
```

<전체 데이터 내용 삭제>

```
DELETE FROM EMP_TEMP2;
```
// 7개 행 이(가) 삭제되었습니다. - 안에 있는 데이터(내용)이 날아감

```
SELECT * FROM EMP_TEMP2;
```

테이블 완전 삭제는 DROP을 사용함.

INSERT, UPDATE, DELETE는 복잡한 내용이 없다.

<트랜잭션과 세션의 개념>
통장 거래
처음과 끝부분을 관리해주는 격이다.
데이터 수정 변경 만들어내고 삭제할 때는 동시에 일어나면 곤란하다.
트랜잭션과 세션의 개념이다.
INSERT, UPDATE, DELETE에 적용되는 개념

데이터를 접근하는데 2사람이 들어왔는데 2사람이 보는 것은 괜찮으나 A라는 사람이 거래를 하는데 수정 변경 삭제를 하면 B라는 사람도 수정 변경 삭제를 하면 충돌이 일어난다.
데이터를 삽입하거나 삭제하거나 업데이트를 할 때는 동시에 수행되게 할 수는 없다.
그 동작이 일어난 후 데이터에 수정 변경 삭제가 일어나야 한다.
거래의 일관성이 있어야 한다.
동작이 일관성 되도록 만들어 줘야 한다.

DB에서 트랜잭션을 알아서 해준다.
트랜잭션과 세션의 개념을 알기만 하면 됩니다.
트랜잭션과 세션은 오라클 데이터베이스에서 컨트롤을 한다.

스크립트 실행(F5)로 동시에 수행한다.

```
INSERT INTO DEPT_TCL VALUES(50, 'DATABASE', 'SEOUL');
UPDATE DEPT_TCL SET LOC = 'BUSAN' WHERE DEPTNO = 40;
DELETE FROM DEPT_TCL WHERE DNAME = 'RESEARCH';
SELECT * FROM DEPT_TCL;
```
동시에 모든 것들이 되었음.

Transaction 취소할 때는 Rollback사용하면 된다.

```
ROLLBACK; // 원상 복구 ↔ Commit하면 저장이 되니 주의
```

<Transaction 영구적으로 반영할 때는 COMMIT 명령어를 사용>
COMMIT; // 커밋 완료 문구가 뜬다. ROLLBACK 불가이니 신중하게 사용할 필요 있음.

<세션과 읽기 일관성의 의미>
일관성이 없으면 데이터 사고가 일어난다.
두 사람이 동시에 들어온 것을 테스트 하기 위해서 SQL Developer와 함께 SQL PLUS(CLI환경)(책에서는 TOAD로 되어 있음)로 테스트를 진행
cmd : ctrl+c하고 마우스 오른쪽 눌러주면 복붙됨
두 사람이 들어왔는데 원본 데이터 변경이 되지 않음.

<수정 중인 데이터 접근 막는 LOCK>
LOCK은 자동적으로 오라클에서 설정하는 것이다.
LOCK을 걸어서 데이터를 보호한다.

<데이터 정의어>
TABLE에 대한 것이다.
객체라는 개념을 알아야 한다.
테이블 만들기는 고급영역
순수한 CREATE - 테이블 생성
기존에 배운 개념을 이해해야 TABLE에 대해 이해를 한다.
데이터에 대해서 알아야 TABLE을 만들기가 수월하다.

<DATA에서의 CRUD - DML>
Create → Insert 명령어
Read → Select 명령어
Update → Update 명령어
Delete → Delete 명령어

<TABLE에서의 CRUD - DDL>
C - CREATE 명령어
R - DESC(Describe) 명령어
U - ALTER, RENAME 명령어
D - DROP, TRUNCATE 명령어

TABLE = 객체 또는 개체라고도 함. Object이라고 한다.
객체 지향의 언어가 SQL이다.

테이블 이하 메뉴를 객체라고 한다.
DB를 만들 때 객체의 관점에서 만든다.

테이블 생성시에 로우와 컬럼, 필드의 데이터 타입을 고려하여 만들어야 한다.
DDL(Data Definition Language) : 데이터 정의 명령어 라고 한다.
※ DML(Data Manipulation Language) : 데이터 조작 명령어

<테이블 만들기>

```
CREATE TABLE EMP_DDL(
    EMPNO       NUMBER(4),
    ENAME       VARCHAR2(10),
    JOB         VARCHAR2(9),
    MGR         NUMBER(4),
    HIREDATE    DATE,
    SAL         NUMBER(7,2),
    COMM        NUMBER(7,2),
    DEPTNO      NUMBER(2)
);
```

// Table EMP_DDL이(가) 생성되었습니다. (직접 반영이 안 되니 새로고침을 해야함.)

속성 = 컬럼을 잘 구성해야지 데이터베이스를 잘 만든다.
속성과 데이터 타입을 적당히 만들고 추가하면서 갖춰나가는 데이터베이스가 좋은 데이터베이스를 만드는 방법이다.

DESC - 테이블의 속성을 파악할 수 있다.(GUI 환경에서는 테이블 더블클릭)
튜토리얼(Tutorial) - 입문을 하면 따라하라는 의미로 하드웨어 / 소프트웨어마다 제공이 된다.(따라하기 : 무작정 따라해라는 의미)
무작정 따라하면서 하면 스스로 배우는 것이 튜토리얼 개념.

<다른 테이블 복사>

```
CREATE TABLE DEPT_DDL
    AS SELECT * FROM DEPT;

DESC DEPT_DDL;
```

<다른 테이블 일부 부분만 복사>
WHERE절을 사용해서 조건을 걸어서 복사.

```
CREATE TABLE EMP_DDL_30
    AS SELECT *
         FROM EMP
        WHERE DEPTNO = 30;

SELECT * FROM EMP_DDL_30;
```

<테이블 복사하여 테이블 생성>

```
CREATE TABLE EMPDEPT_DDL
    AS SELECT E.EMPNO, E.ENAME, E.JOB, E.MGR, E.HIREDATE,
              E.SAL, E.COMM, D.DEPTNO, D.DNAME, D.LOC
         FROM EMP E, DEPT D
        WHERE 1 <> 1;

SELECT * FROM EMPDEPT_DDL;
```

-> 테이블의 컬럼만 복사가 된다.

<ALTER - 테이블 업데이트>

```
CREATE TABLE EMP_ALTER
    AS SELECT * FROM EMP;

SELECT * FROM EMP_ALTER;
```

<ALTER를 이용하여 컬럼 추가 - 테이블 업데이트>

```
ALTER TABLE EMP_ALTER
ADD HP VARCHAR2(20);
SELECT * FROM EMP_ALTER;
```

열 - 추가에서 생성 가능

<RENAME - 컬럼명 변경하기>

```
ALTER TABLE EMP_ALTER
RENAME COLUMN HP TO TEL;
```

열 - 이름 바꾸기로 가능함.

<MODIFY - 컬럼의 자료형 변환>

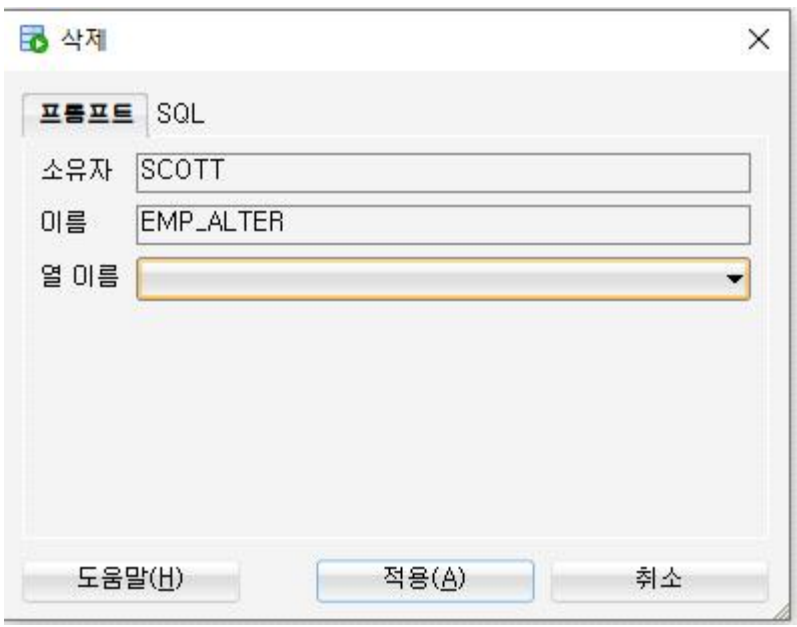
```
ALTER TABLE EMP_ALTER
MODIFY EMPNO NUMBER(5);
```

```
DESC EMP_ALTER;
```

<테이블 삭제 - DROP>

```
ALTER TABLE EMP_ALTER
 DROP COLUMN TEL;
```

```
SELECT * FROM EMP_ALTER;
```



열 - 삭제에서 가능

<테이블 이름 바꾸기 - RENAME>

테이블 - 이름 바꾸기 메뉴

<TRUNCATE - 로우(행) 안의 데이터만 삭제>
컬럼 데이터는 DROP을 사용한다.

```
TRUNCATE TABLE EMP_RENAME;
```

<DROP - 컬럼과 로우로 구성된 테이블 전체를 삭제시킨다>

```
DROP TABLE EMP_RENAME;
```