2장. SQL – DDL, DML

데이터 베이스 구성요소

데이터베이스 구성 객체

❖ 데이터베이스 객체의 종류

DB 객체	설명
테이블	데이터를 담고 있는 객체
뷰	하나 이상의 데이블을 연결해 마치 테이블인 것처럼 사용하는 객체
인덱스	테이블에 있는 데이터를 빠르게 찾기 위한 객체
시노님	데이터베이스 객체에 대한 별칭을 부여한 객체
시퀀스	일련번호를 사용하는 객체
함수	특정 연산을 하고 값을 반환하는 객체
프로시저	함수와 비슷하지만 값을 반환하지 않는 객체

데이터베이스 구성 객체

테이블(TABLE)

데이터를 넣고 수정하고 삭제하는, 즉 데이터를 담고 있는 객체가 테이블이다. 테이블은 DBMS상에서 가장 기본적인 객체로 로우(행)와 컬럼(열)으로 구성된 2차원 형태(표)의 객체로 엑셀과 구조가 비슷하다.

```
CREATE TABLE 테이블명(
    칼럼1    데이터타입 [NULL, NOT NULL],
    칼럼1    데이터타입 [NULL, NOT NULL],
...
);
```

데이터베이스 구성 객체

데이터 타입(Type)

데이터 타입	설명
CHAR	고정길이 문자, 최대 2000byte, 디폴트값은 1byte
VARCHAR2	가변길이 문자, 최대 4000byte, 디폴트값은 1byte
NUMBER	가변 숫자, 십진수 기준 최대 220byte
FLOAT	NUMBER의 하위 타입, 이진수 기준 22byte
DATE	날짜 – 연, 월, 일
TIMESTAMP	날짜 – 연, 월, 일, 시, 분, 초, 밀리초
NULL	'값이 없음'을 의미하며 디폴트값이 NULL이다.

데이터베이스 구성 객체

테이블(TABLE) 생성과 데이터 타입

```
-- ex2_1 테이블 --
CREATE TABLE ex2_1(
    column1 CHAR(10),
    column2 VARCHAR2(10),
    column3 NUMBER
);

INSERT INTO ex2_1(column1, column2, column3) VALUES ('abc', 'abc', 100);

SELECT * FROM ex2_1;

-- CHAR와 VARCHAR2의 크기 비교--
SELECT column1, LENGTH(column1) as len1,
       column2, LENGTH(column2) as len2
FROM ex2_1;
```

COLUMN1	LEN1	COLUMN2	LEN2
abc	10	abc	3

데이터베이스 구성 객체

테이블(TABLE) 생성과 데이터 타입

```
-- ex2_2 테이블 --
CREATE TABLE ex2_2(
    col_date    DATE,
    col_timestamp   TIMESTAMP
);

-- 날짜 데이터 비교
INSERT INTO ex2_2 VALUES (SYSDATE, SYSTIMESTAMP);

SELECT * FROM ex2_2;
```

COL_DATE	COL_TIMESTAMP
2021/01/01	21/01/01 06:08:21.980000000

데이터베이스 구성 객체

제약조건

테이블들은 각 속성(칼럼)에 대한 무결성을 유지하기 위한 다양한 제약 조건(Constraints)이 적용되어 있다.

제약 조건에는 NOT NULL, 기본키, 외래키, CHECK 등이 있다.

NOT NULL

칼럼을 정의할 때 NOT NULL 제약 조건을 명시하면 반드시 데이터를 입력해야한다.

```
칼럼명  데이터 타입 NOT NULL
```

데이터베이스 구성 객체

```
-- ex2_3 테이블
CREATE TABLE ex2_3(
    col_null    VARCHAR2(10),
    col_not_null    VARCHAR2(10) NOT NULL
);

INSERT INTO ex2_3 VALUES ('hello', '');

INSERT INTO ex2_3 VALUES ('hello', 'Thank you!');
```

```
오류 보고 -
ORA-01400: NULL을 ("HR"."EX2_3"."COL_NOT_NULL") 안에 삽입할 수 없습니다
```

데이터베이스 구성 객체

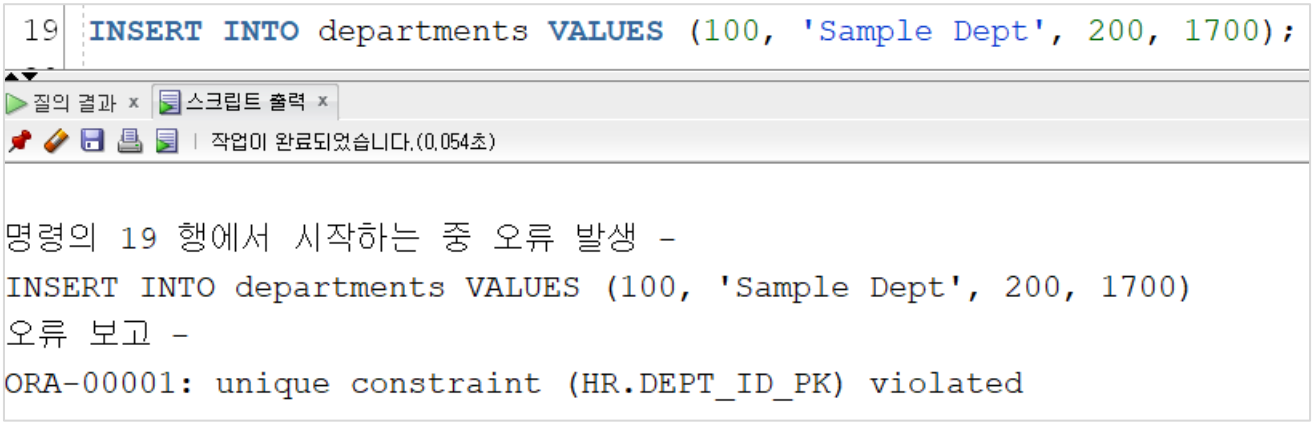
기본키 제약조건

기본키는 Primay Key라고도 하며, UNIQUE와 NOT NULL 속성을 동시에 가진 제약 조건으로 테이블 당 1개의 기본키만 생성할 수 있다.

칼럼명 데이터 타입 **PRIMARY KEY**

또는

CONSTRAINTS 제약조건명 **PRIMARY KEY**(칼럼명, ...)



데이터베이스 구성 객체

```
-- 제약(CONSTRAINT) 조건 --
CREATE TABLE ex2_4(
    col_1 VARCHAR2(10) PRIMARY KEY,
    col_2 NUMBER(2)
);

-- 기본키는 NOT NULL 이므로 NULL 입력 불가
INSERT INTO ex2_4 VALUES ('', 25);

INSERT INTO ex2_4 VALUES ('AA', 25);

-- 기본키는 중복 입력 불가
INSERT INTO ex2_4 VALUES ('AA', 25);
```

```
오류 보고 -
ORA-01400: cannot insert NULL into ("HR"."EX2_4"."COL_1")
```

```
오류 보고 -
ORA-00001: unique constraint (HR.SYS_C007002) violated
```

데이터베이스 구성 객체

외래키 제약조건

기본키는 Primay Key라고도 하며, UNIQUE와 NOT NULL 속성을 동시에 가진 제약 조건으로 테이블 당 1개의 기본키만 생성할 수 있다.

칼럼명 데이터 타입 **PRIMARY KEY**

또는

CONSTRAINTS 제약조건명 **PRIMARY KEY**(칼럼명, ...)

```
19  INSERT INTO departments VALUES (100, 'Sample Dept', 200, 1700);
```

질의 결과 × 스크립트 출력 ×

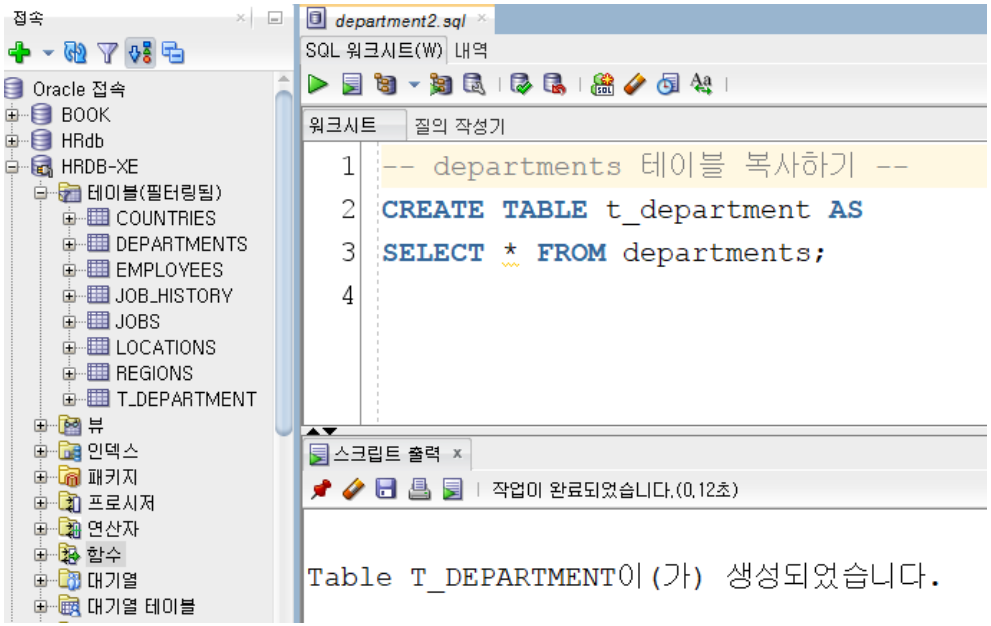
작업이 완료되었습니다.(0.054초)

```
명령의 19 행에서 시작하는 중 오류 발생 -
INSERT INTO departments VALUES (100, 'Sample Dept', 200, 1700)
오류 보고 -
ORA-00001: unique constraint (HR.DEPT_ID_PK) violated
```

데이터베이스 구성 객체

테이블 복사하기

```
CREATE TABLE 새 테이블명 AS
SELECT * FROM 복사할 테이블명
```



데이터베이스 구성 객체

테이블 삭제하기

```
-- employees 테이블 복사(특정 칼럼) --
CREATE TABLE t_employee AS
SELECT employee_id, first_name, last_name, salary, job_id
FROM employees;

SELECT * FROM t_employee;

-- 테이블 삭제 --
DROP TABLE t_employee;
```

데이터베이스 구성 객체

시퀀스(Sequence)

자동 순번을 반환하는 데이터베이스 객체이다.

```
-- 시퀀스 --
CREATE SEQUENCE mySeq
INCREMENT BY 1
START WITH 1
MINVALUE 1
MAXVALUE 1000
NOCYCLE
NOCACHE;

-- 1부터 시작해서 1씩 증가하며 최소값 1 부터
-- 최대값 1000까지 순번을 자동생성한다.
```

데이터베이스 구성 객체

시퀀스(Sequence)

```
-- 데이터 입력
INSERT INTO ex2_8 (col1, col2) VALUES (mySeq.NEXTVAL, 'cat');

INSERT INTO ex2_8 (col1, col2) VALUES (mySeq.NEXTVAL, 'dog');

SELECT * FROM ex2_8;

-- 시퀀스 삭제 --
DROP SEQUENCE mySeq;
```

COL1	COL2
1	cat
2	dog

SELECT – 자료 검색

SELECT 문

SELECT 열이름 (or 별칭)

FROM 테이블 이름

1. 열(칼럼)을 조회하고 정렬하는 방법

전체 검색 및 특정 열 검색

```
1  -- 테이블 전체 조회
2  SELECT * FROM employees;
3
4  --1. 열을 조회하고 정렬하는 방법
5  -- 특정 열만 조회
6  SELECT employee_id, first_name, last_r
7  FROM employees;
8
```

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME
1	100	Steven	King
2	101	Neena	Kochhar
3	102	Lex	De Haan
4	145	John	Russell
5	146	Karen	Partners
6	201	Michael	Hartstein
7	108	Nancy	Greenberg
8	205	Shelley	Higgins

SELECT – 자료 검색

별칭(이름) 만들기

```
-- AS를 사용하여 별칭 만들기(생략 가능)
SELECT employee_id AS 사원번호, first_name 이름, las
FROM employees;
```

사원번호	이름	성
100	Steven	King
101	Neena	Kochhar
102	Lex	De Haan
103	Alexander	Hunold
104	Bruce	Ernst
105	David	Austin

ORDER BY를 이용한 정렬

```
-- 정렬 : ORDER BY - ASC(오름차순), DESC(내림차순)
SELECT employee_id, first_name, last_name, salary
FROM employees ORDER BY salary DESC;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY
100	Steven	King	24000
101	Neena	Kochhar	17000
102	Lex	De Haan	17000
145	John	Russell	14000
146	Karen	Partners	13500

SELECT – 자료 검색

```
-- 데이터 값 계산하기 : 산술 연산자
SELECT employee_id, salary, salary+500, salary-100
FROM employees;

SELECT employee_id AS 사원번호, salary AS 급여, salary+500 AS 추가급여,
FROM employees;
```

	사원번호	급여	추가급여	인하급여
1	100	24000	24500	23900
2	101	17000	17500	16900
3	102	17000	17500	16900
4	103	9000	9500	8900
5	104	6000	6500	5900
6	105	4800	5300	4700

```
-- 데이터 값 연결하기 : 연결 연산자 '||'
SELECT employee_id, first_name ||' '||last_name
FROM employees;

SELECT employee_id, first_name ||'@'||'gmail.com'
FROM employees;
```

SQL | 50개의 행이 인출됨(0.003초)

	EMPLOYEE_ID	NAME
1	100	Steven King
2	101	Neena Kochhar
3	102	Lex De Haan
4	103	Alexander Hunold
5	104	Bruce Ernst
6	105	David Austin

DISTINCT 를 이용한 중복값 제거

```
SELECT DISTINCT job_id, department_id
FROM employees;
```

SQL | 인출된 모든 행: 20(0.002초)

	JOB_ID	DEPARTMENT_ID
1	IT_PROG	60
2	FI_MGR	100
3	PU_MAN	30
4	ST_MAN	50
5	SA_REP	80
6	AD_VP	90
7	AD_ASST	10

SELECT – 자료 검색

2. 행의 데이터 값 조회 방법 – WHERE 조건절 이용

조건과 일치하는 값 검색하기(비교 연산)

```
SELECT *
FROM employees
--WHERE first_name = 'David';
WHERE employee_id <> 105;
```

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL
1	105	David	Austin	DAUSTIN
2	151	David	Bernstein	DBERNSTE
3	165	David	Lee	DLEE

특정 문자가 포함된 데이터 검색 – LIKE 연산자

```
SELECT *
FROM employees
WHERE job_id LIKE 'AD%';
```

EMPLOYEE_ID	JOB_ID	MANAGER_ID
109	FI_ACCOUNT	108
110	FI_ACCOUNT	108
111	FI_ACCOUNT	108
112	FI_ACCOUNT	108
113	FI_ACCOUNT	108
206	AC_ACCOUNT	205

NULL 검색

```
SELECT employee_id, job_id, manager_id
FROM employees
--WHERE manager_id IS NULL;
WHERE manager_id IS NOT NULL;
```

EMPLOYEE_ID	JOB_ID	MANAGER_ID
100	AD_PRES	(null)

SELECT – WHERE 조건절

2. 행의 데이터 값 조회 방법 – WHERE 조건절 이용

논리 연산자 : BETWEEN~AND 와 IN

```
SELECT *
FROM employees
WHERE salary BETWEEN 10000 AND 20000;
```

EMPLOYEE_ID	FIRST_NAME	JOB_ID	SALARY
101	Neena	AD_VP	17000
102	Lex	AD_VP	17000
108	Nancy	FI_MGR	12008
114	Den	PU_MAN	11000
145	John	SA_MAN	14000

```
SELECT first_name, hire_date, job_id, salary
FROM employees
WHERE salary IN (10000, 17000, 24000);
```

FIRST_NAME	HIRE_DATE	JOB_ID	SALARY
Steven	2003/06/17	AD_PRES	24000
Neena	2005/09/21	AD_VP	17000
Lex	2001/01/13	AD_VP	17000
Peter	2005/01/30	SA_REP	10000
Janette	2004/01/30	SA_REP	10000
Harrison	2006/03/23	SA_REP	10000
Hermann	2002/06/07	PR_REP	10000

SELECT – 자료 검색

- employees 테이블 복사

```
-- employees 테이블 복사(특정 칼럼) --
CREATE TABLE t_employee AS
SELECT employee_id, first_name, last_name, salary, job_id
FROM employees;

SELECT * FROM t_employee;
```

```
-- first_name이 'David'이고, salary가 4000을 초과하는 직원
SELECT * FROM t_employee
WHERE first_name = 'David'
AND salary > 4000;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	JOB_ID
105	David	Austin	4800	IT_PROG
151	David	Bernstein	9500	SA_REP
165	David	Lee	6800	SA_REP

SELECT – 자료 검색

- 실습 문제

아래의 실행 화면처럼 salary(급여)가 7000을 초과하고, job_id(직무)가 'IT' 나 'FI'로 시작하는 직원을 검색하세요.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	JOB_ID
103	Alexander	Hunold	9000	IT_PROG
108	Nancy	Greenberg	12008	FI_MGR
109	Daniel	Faviet	9000	FI_ACCOUNT
110	John	Chen	8200	FI_ACCOUNT
111	Ismael	Sciarra	7700	FI_ACCOUNT
112	Jose Manuel	Urman	7800	FI_ACCOUNT
113	Luis	Popp	6900	FI_ACCOUNT

DML이란?

DML(Data Manipulation Language)은 데이터를 조작하는 명령어이다.

INSERT(삽입), UPDATE(변경), DELETE(삭제) 명령어를 사용한다.

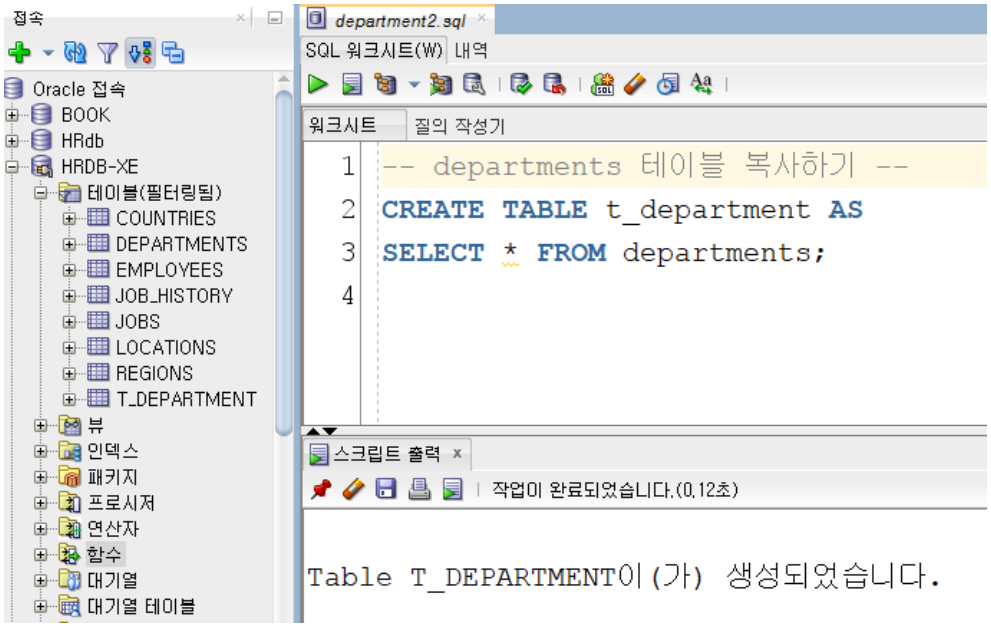
데이터를 조작하여 저장하는 일련의 과정을 **트랜잭션(transition)**이라 하며, DML은 트랜잭션을 다루는 명령어이다.

명령어	설 명
INSERT	테이블에 새로운 행 삽입
UPDATE	테이블에 있는 행의 내용 갱신
DELETE	테이블의 행을 삭제

데이터베이스 구성 객체

- 테이블 복사하기

```
CREATE TABLE 새 테이블명 AS
SELECT * FROM 복사할 테이블명
```



INSERT : 행 삽입하기

INSERT INTO 테이블 이름(열이름1, 열이름2...)

VALUES (데이터 값1, 데이터값 2, ...)

예제1. 1행 추가하기

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	271	Sample Dept	200	1700
2	270	Payroll	(null)	1700
3	260	Recruiting	(null)	1700
4	250	Retail Sales	(null)	1700
5	240	Government Sales	(null)	1700

```
-- 자료 검색
SELECT * FROM t_department
ORDER BY department_id DESC;

-- 자료 추가
INSERT INTO t_department VALUES (271, 'Sample_Dept', 200, 1700);

-- 롤백 - 명령 취소
ROLLBACK;

-- 명령 실행 완료 : 트랜잭션
COMMIT;
```

INSERT : 행 삽입하기

INSERT 후에는 COMMIT을 명령해야함.

명령을 취소 할때는 ROLLBACK 사용(단, COMMIT을 하기 전에 사용 가능)

롤백(ROLLBACK) 후 검색

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	270	Payroll	(null)	1700
2	260	Recruiting	(null)	1700
3	250	Retail Sales	(null)	1700
4	240	Government Sales	(null)	1700
5	230	IT Helpdesk	(null)	1700
6	220	NOC	(null)	1700
7	210	IT Support	(null)	1700
8	200	Operations	(null)	1700
9	190	Contracting	(null)	1700
10	180	Construction	(null)	1700

UPDATE : 행 갱신하기

UPDATE 테이블 이름

SET 열이름 1 = 데이터 값 1

[WHERE 조건식]

```
-- departments에서 department_name이 Sample_Dept인
-- manager_id를 201, location_id를 1800으로 변경하기
UPDATE departments
SET manager_id = 201,
    location_id = 1800
WHERE department_name = 'Sample_Dept';
```

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	271	Sample Dept	201	1800
2	270	Payroll	(null)	1700
3	260	Recruiting	(null)	1700
4	250	Retail Sales	(null)	1700
5	240	Government Sales	(null)	1700

UPDATE : 행 갱신하기

예제2. departments 테이블을 변경하기

- 실수로 전체를 업데이트 한 경우

```
-- departments에서 department_name이 Sample_Dept인
-- manager_id를 201, location_id를 1800으로 변경하기
UPDATE departments
SET manager_id = 201,
    location_id = 1800;
--WHERE department_name = 'Sample_Dept';

ROLLBACK;
```

이전으로 되돌아감

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	271	Sample Dept	201	1800
2	270	Payroll	201	1800
3	260	Recruiting	201	1800
4	250	Retail Sales	201	1800
5	240	Government Sales	201	1800

DELETE : 행 삭제하기

DELETE [FROM] 테이블 이름

[WHERE 조건식]

[ ] – 대괄호는 생략 가능

예제1.

department_name이 'Sample Dept'행 삭제하기

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	270	Payroll	(null)	1700
2	260	Recruiting	(null)	1700
3	250	Retail Sales	(null)	1700
4	240	Government Sales	(null)	1700
5	230	IT Helpdesk	(null)	1700

DELETE : 행 삭제하기

```
21  DELETE FROM departments
22  WHERE department_name = 'Sample_Dept';
23
24  -- 전체 데이터 삭제하기
25  DELETE FROM departments;
```

스크립트 출력 x | 질의 결과 x

작업이 완료되었습니다.(0.022초)

```
명령의 25 행에서 시작하는 중 오류 발생 -
DELETE FROM departments
오류 보고 -
ORA-02292: 무결성 제약조건(HR.EMP_DEPT_FK)이 위배되었습니다- 자식 레코드가 발견되었습니다
```

INSERT : 행 삽입하기

INSERT 예제2.

```
-- ex3_1 테이블 생성 --
CREATE TABLE ex3_1(
    col1    VARCHAR2(10),
    col2    NUMBER,
    col3    DATE
);
```

```
-- 테이블(해당칼럼), 칼럼에 들어갈 값
INSERT INTO ex3_1(col1, col2, col3)
VALUES ('ABC', 10, SYSDATE);

-- 해당칼럼을 생략한 경우 --
INSERT INTO ex3_1 VALUES ('DEF', 20, SYSDATE);

-- 칼럼을 일부만 사용한 경우 --
INSERT INTO ex3_1(col1, col2)
VALUES ('GHI', 30);
```

	COL1	COL2	COL3
1	ABC	10	2020/12/26
2	DEF	20	2020/12/26
3	GHI	30	(null)

DELETE : 행 삭제하기

예제2. commit 명령어 확인하기

```
select * from tab;  -- 테이블 검색
select * from t_department;
```

```
C:₩Users₩김기용>sqlplus

SQL*Plus: Release 11.2.0.2.0 Production on 목 2월 25 08:06:06 2021

Copyright (c) 1982, 2014, Oracle.  All rights reserved.

Enter user-name: hr
Enter password:

Connected to:
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production

SQL> select * from tab;

TNAME                                                       TABTYPE
----------------------------------------------------------- -------------
 CLUSTERID
----------
COUNTRIES                                                   TABLE

DEPARTMENTS                                                 TABLE

EMPLOYEES                                                   TABLE
```

```
SQL> select * from ex3_1;

no rows selected
```

데이터 삽입 후 검색하면 반영이 안되고, commit을 해야 반영됨

UPDATE : 행 갱신하기

예제1. ex3_1 테이블 데이터 변경하기

```
-- col2의 값을 50으로 변경 --
UPDATE ex3_1
SET col2 = 50;

-- col3의 null을 찾아 현재 날짜로 변경하기
UPDATE ex3_1
SET col3 = SYSDATE
WHERE col3 IS NULL;
```

	COL1	COL2	COL3
1	ABC	50	2020/12/26
2	DEF	50	2020/12/26
3	GHI	50	2020/12/27

```
SQL> select * from ex3_1;

COL1                 COL2 COL3
-------------------- ---------- --------
ABC                        10 21/02/25
DEF                        20 21/02/25
GHI                        30 21/02/25
```

INSERT ~ SELECT: 행 삽입하기

INSERT ~ SELECT 구문

```
-- ex3_2 테이블 생성 --
CREATE TABLE ex3_2(
    emp_id  NUMBER,
    emp_name    VARCHAR2(20)
);
```

	EMP_ID	EMP_NAME
1	100	King
2	101	Kochhar
3	102	De Haan
4	103	Hunold
5	104	Ernst
6	108	Greenberg
7	109	Faviet
8	110	Chen
9	111	Sciarra
10	112	Urman
11	113	Popp
12	114	Raphaely
13	120	Weiss

```
-- INSERT ~ SELECT --
-- 다른 테이블이나 뷰의 조회 결과로 나온 테이터를
-- 또 다른 테이블에 넣는 형식
INSERT INTO ex3_2(emp_id, emp_name)
SELECT employee_id, last_name
FROM employees
WHERE salary > 5000;
```

스크립트 출력 × | 질의 결과 ×

작업이 완료되었습니다.(0.028초)

58개 행 이(가) 삽입되었습니다.

의사 칼럼 – ROWNUM

의사 칼럼(Pseudo-column)이란

테이블의 칼럼처럼 동작하지만 실제로 테이블에 저장되지 않는 칼럼을 말한다.

- ROWNUM : 쿼리에서 반환되는 각 로우들에 대한 순서값을 나타낸다.

<실습 예제>

```
SELECT ROWNUM, employee_id
    FROM employees
WHERE ROWNUM < 5;

-- ROWNUM은 SELECT만 가능하다.(삭제, 수정 불가)
DELETE FROM employees WHERE ROWNUM = 4;
```

ROWNUM	EMPLOYEE_ID
1	100
2	101
3	102
4	103

0개 행 이(가) 삭제되었습니다.

의사 칼럼 – ROWNUM

- **게시판에서 ROWNUM 이용하기**

  페이지 처리를 할때 게시글이 삭제되므로 순번(SEQUENCE)으로는 계산할 수 없으므로 ROWNUM을 사용 할 수 있다.

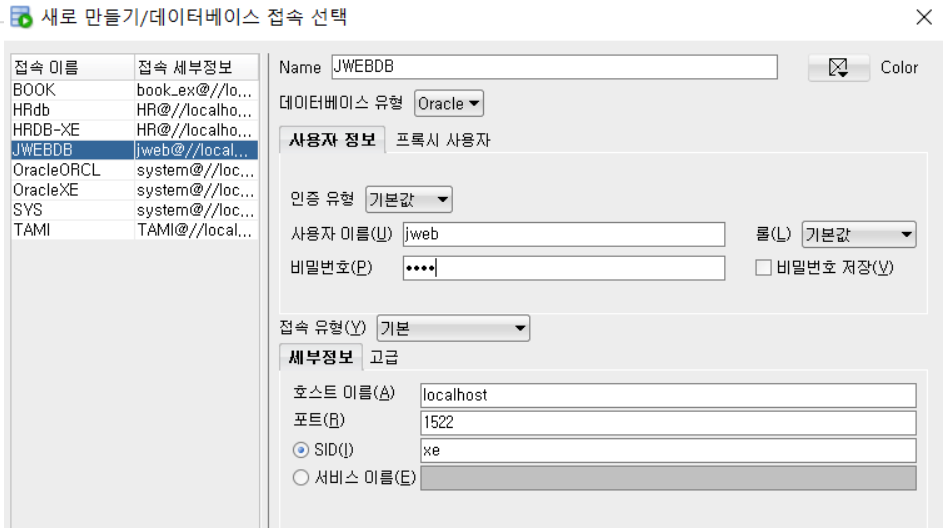
```
SELECT ROWNUM, t_board.* FROM t_board;

SELECT ROWNUM, t_board.* FROM t_board ORDER BY regDate DESC;
```

ROWNUM	BNUM	TITLE	CONTENT
19	29	python	python
18	25	java	java
17	24	c	c
16	19	3월 마지막 주	오늘은 월요일..
15	18	주말이다	fdfdf
14	15	고양이	고양이
13	14	소	소
12	13	강아지	강아지
11	11	대나무	대나무
10	10	소나무	소나무
9	9	매화	매화
8	8	녹련	녹련

사용자 DB 만들기

SYSTEM에서 새 데이터베이스 사용자 만들기

```
1  -- USER(jweb) 생성
2  CREATE USER jweb IDENTIFIED BY 4321;
3
4  -- SESSION 권한 부여
5  GRANT CREATE SESSION TO jweb;
6
7  -- 테이블과 테이블 공간 생성
8  GRANT CREATE TABLE, RESOURCE TO jweb;
9
```



DDL(데이터 정의어)

CREATE : 테이블 생성하기

```
CREATE TABLE 테이블이름(
    열 이름1   데이터타입 ,
    열 이름2   데이터타입 ,
     .....
);
```

```
-- t_student 테이블 생성
CREATE TABLE t_student(
    studentId NUMBER(3),
    studentName VARCHAR2(10) NOT NULL,
    PRIMARY KEY(studentId)
);
```

DDL(데이터 정의어)

ALTER : 테이블 수정하기 – 칼럼(열) 추가하기

```
ALTER TABLE 테이블이름
    ADD (열 이름1   데이터타입 ,
         열 이름2   데이터타입 ,
    );
```

- 새로 생성되는 열은 위치를 지정할 수 없다.(테이블의 마지막에 위치한다.)
- 새로운 열의 데이터 값은 null로 초기화 된다.

```
SELECT * FROM t_student2;

ALTER TABLE t_student2 ADD (age NUMBER(3));

UPDATE t_student2 SET age=25 WHERE studentId = 101;
```

	STUDENTID	STUDENTNAME	AGE
1	103	산들	(null)
2	104	강태양	(null)
3	101	이강	25
4	102	김산	(null)
5	201	박화성	(null)

DDL(데이터 정의어)

ALTER : 테이블 수정하기 – 칼럼(열) 변경하기

ALTER TABLE 테이블이름

MODIFY (열 이름1 데이터타입);

```
-- 칼럼 변경( 자료형의 크기를 10byte-> 20byte)
ALTER TABLE t_student2 MODIFY (studentName VARCHAR2(20));
```

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT
1	STUDENTID	NUMBER(3,0)	Yes	(null)
2	STUDENTNAME	VARCHAR2(20 BYTE)	No	(null)
3	AGE	NUMBER(3,0)	Yes	(null)

DDL(데이터 정의어)

ALTER : 테이블 수정하기 – 칼럼(열) 이름 변경하기

ALTER TABLE 테이블이름

RENAME COLUMN 열 이름1 **TO** 변경할 열 이름

```
-- 칼럼 이름 변경
ALTER TABLE t_student2 RENAME COLUMN age TO studentAge;
```

	COLUMN_NAME	DATA_TYPE	NULLABLE
1	STUDENTID	NUMBER(3,0)	Yes
2	STUDENTNAME	VARCHAR2(20 BYTE)	No
3	STUDENTAGE	NUMBER(3,0)	Yes

DDL(데이터 정의어)

ALTER : 칼럼 삭제하기

ALTER TABLE 테이블이름 **DROP** COLUMN 열 이름;

```
-- 칼럼 삭제
ALTER TABLE t_student2 DROP COLUMN studentId;
```

	STUDENTNAME	AGE
1	산들	(null)
2	강태양	(null)
3	이강	25
4	김산	(null)
5	박화성	(null)

DDL(데이터 정의어)

TRANCATE : 테이블의 내용 모두 삭제하기

```
TRANCATE  TABLE 테이블이름
```

```
-- 테이블의 내용(자료) 모두 삭제하기
TRUNCATE TABLE t_student2;
```

```
Table T_STUDENT2이(가) 잘렸습니다.
```

- 테이블의 모든 데이터를 삭제하고 사용하던 기억 공간도 해제한다.
- 삭제된 데이터는 자동으로 커밋 된다.
- 테이블의 구조 즉, 테이블 자체는 삭제되지 않고 다시 추가가 가능하다.

DDL(데이터 정의어)

DROP : 테이블 삭제하기

DROP TABLE 테이블이름;

```
-- 테이블 삭제
DROP TABLE t_student2;


SELECT * FROM t_student2;
```

```
ORA-00942: table or view does not exist
00942, 00000 - "table or view does not exist"
*Cause:
*Action:
31행, 15열에서 오류 발생
```