



Oracle

SQL

SQL

- **SQL: Structured Query Language**
 - DBMS(Database Management System)에서 데이터를 읽고 쓰고 삭제하는 등 데이터를 관리하기 위한 일종의 프로그램 언어
 - IBM 연구소에서 개발한 SEQUEL(Structured English Query Language)에서 유래
 - 국제표준기구(ISO)와 미국국립표준협회(ANSI)에서 RDBMS의 표준 언어로 SQL을 채택
- **SQL의 종류**
 - **DDL(Data Definition Language): CREATE, DROP, ALTER, TRUNCATE**
 - **DQL(Data Query Language): SELECT**
 - **DML(Data Manipulation Language): INSERT, UPDATE, DELETE**
 - **TCL(Transaction Control Language): COMMIT, ROLLBACK**
- **PL/SQL: Procedural Language extension to SQL**
 - SQL을 이용한 절차적 프로그래밍 언어
 - 변수 선언과 사용, 제어문(IF, CASE, LOOP), 함수, 프로시저, 예외처리 등

Table

Column (열), Variables, Fields, Attributes

Row(행)
Observation
Records
Examples

PatientID	AdmDate	Age	Diabetes	Status
1	2015/10/15	25	Type1	Poor
2	2015/11/14	34	Type2	Improved
3	2015/10/21	28	Type1	Excellent
4	2015/10/28	52	Type1	Poor

- DBMS: records / fields
- 통계: observations(관측치) / variables(변수)
- Data-mining, Machine-learning: examples / attributes

Oracle 11g XE 실습 계정 활성화

```
$ sqlplus / as sysdba
```

```
SQL> create user scott identified by tiger;
```

```
SQL> grant dba to scott;
```

```
SQL> connect scott/tiger;
```

```
SQL> select table_name from user_tables;
```

```
SQL> exit
```

```
$ sqlplus scott/tiger
```



CREATE USER

CREATE USER 아이디 **IDENTIFIED BY** 비밀번호;

- 새로운 생성자 생성

```
SQL> CREATE USER scott IDENTIFIED BY tiger;
```

GRANT 권한, ... **TO** 사용자 아이디;

- 생성된 사용자에게 권한 부여

```
SQL> GRANT connect , resource TO scott;
```

COMMIT;

- DB에 변경 내용 저장

```
SQL> commit;
```



CREATE TABLE (1)

Inline/Column-Level 제약 조건 정의

CREATE TABLE 테이블이름 (

컬럼1 데이터타입 [**DEFAULT** 기본값] [[**CONSTRAINT** 제약조건이름] 제약조건내용],

컬럼2 데이터타입 [**DEFAULT** 기본값] [[**CONSTRAINT** 제약조건이름] 제약조건내용],

...,

);

- 테이블 객체를 생성
- 데이터 타입(data type)
 - NUMBER, VARCHAR2, DATE, TIMESTAMP, CLOB, BLOB 등
- 제약 조건(constraints)
 - NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY
 - CHECK

CREATE TABLE (2)

Out-of-Line/Table-Level 제약 조건 정의 - NOT NULL 제약 조건을 제외한 제약조건 지정

```
CREATE TABLE 테이블이름 (  
    컬럼1 데이터타입 [DEFAULT 기본값],  
    컬럼2 데이터타입 [DEFAULT 기본값],  
    ...,  
    CONSTRAINT 제약조건이름 제약조건내용 (컬럼)],  
    CONSTRAINT 제약조건이름 제약조건내용 (컬럼)],  
    ...  
);
```



Constraints(제약 조건)

종류	설명
NOT NULL	지정한 열에 NULL을 허용하지 않습니다. NULL을 제외한 데이터의 중복은 허용됩니다.
UNIQUE	지정한 열이 유일한 값을 가져야 합니다. 즉 중복될 수 없습니다. 단 NULL은 값의 중복에서 제외됩니다.
PRIMARY KEY	지정한 열이 유일한 값이면서 NULL을 허용하지 않습니다. PRIMARY KEY는 테이블에 하나만 지정 가능합니다.
FOREIGN KEY	다른 테이블의 열을 참조하여 존재하는 값만 입력할 수 있습니다.
CHECK	설정된 조건식을 만족하는 데이터만 입력 가능합니다.

ALTER

ALTER TABLE 테이블이름 **RENAME COLUMN** 변경전컬럼이름 **TO** 변경후컬럼이름;

- 테이블의 컬럼 이름 변경

ALTER TABLE 테이블이름 **MODIFY** 컬럼이름 데이터타입;

- 테이블의 데이터 타입 변경

ALTER TABLE 테이블이름 **ADD** 컬럼이름 데이터타입;

- 테이블에 새로운 컬럼을 추가

ALTER TABLE 테이블이름 **DROP COLUMN** 컬럼이름;

- 테이블의 컬럼을 삭제

ALTER TABLE 테이블이름 **ADD CONSTRAINTS** 제약조건이름;

- 테이블에 제약 조건을 추가

ALTER TABLE 테이블이름 **DROP CONSTRAINTS** 제약조건이름;

- 테이블에서 제약 조건을 삭제

Delete, Truncate, Drop

DELETE FROM 테이블이름;

- 테이블의 모든 행(row)의 데이터를 삭제, 테이블 크기가 줄어들 지는 않음

TRUNCATE TABLE 테이블이름;

- 테이블의 모든 행(row)의 데이터를 삭제하고, 행(row) 자체도 삭제

DROP TABLE 테이블이름;

- 테이블 자체를 DBMS에서 삭제

id	pw	email
root	root12	root@test.com
admin	admin12	admin@test.com
sys	sys12	sys@test.com

Delete

id	pw	email

Truncate

id	pw	email

Drop

--	--	--

DESC

DESC 테이블 이름;

➤ 테이블의 구조를 출력

```
SQL> DESC tab;
```

```
SQL> DESC user_tables;
```

```
SQL> DESC departments;
```

```
SQL> DESC employees;
```



CREATE / DROP SEQUENCE

CREATE SEQUENCE 시퀀스이름

[INCREMENT BY 정수]

[START WITH 정수]

[MAXVALUE 정수 | NOMAXVALUE]

[MINVALUE 정수 | NOMINVALUE]

[CYCLE | NOCYCLE]

[CACHE 정수 | NOCACHE]

[ORDER | NOORDER];

DROP SEQUENCE 시퀀스이름;

- 기본값

- INCREMENT BY: 1
- START WITH: 1
- MAXVALUE 정수: NOMAXVALUE (10^{27} , -1)
- MINVALUE 정수: NOMINVALUE (1, -10^{26})
- CYCLE: NOCYCLE
- CACHE: 20
- ORDER: NOORDER

SELECT (1)

SELECT 컬럼이름, 컬럼이름, ... **FROM** 테이블이름;

- 테이블의 특정 컬럼 내용을 확인(출력)

```
SQL> SELECT * FROM department;
```

```
SQL> SELECT deptno, dname FROM department;
```

SELECT DISTINCT 컬럼이름, 컬럼이름, ... **FROM** 테이블이름;

- DISTINCT: 중복된 값은 제거하고 출력

```
SQL> SELECT DISTINCT deptno1 FROM student;
```



SELECT (2)

SELECT 컬럼이름, ... **FROM** 테이블이름 **WHERE** 조건;

➤ 조건에 맞는 데이터들만 출력

```
SQL> SELECT * FROM emp WHERE deptno = 10;
```

```
SQL> SELECT * FROM emp WHERE sal >= 3000;
```

```
SQL> SELECT * FROM emp WHERE sal BETWEEN 1000 AND 3000;
```

```
SQL> SELECT * FROM emp WHERE deptno IN (10, 20);
```

```
SQL> SELECT * FROM emp WHERE ename LIKE 'S%';
```

```
SQL> SELECT * FROM emp WHERE comm IS NOT NULL AND sal > 1500;
```

```
SQL> SELECT * FROM emp WHERE sal < 1000 OR sal > 4000;
```

```
SQL> SELECT * FROM emp WHERE NOT ename LIKE 'S%';
```



SELECT (3)

SELECT 컬럼이름, ... **FROM** 테이블이름 **ORDER BY** 컬럼이름;

- 출력 결과를 정렬해서 보여줌
- **ORDER BY** 컬럼이름 **ASC**: 오름차순(기본값, ASC는 생략 가능)
- **ORDER BY** 컬럼이름 **DESC**: 내림차순

```
SQL> SELECT name FROM student ORDER BY name;
```

```
SQL> SELECT name FROM student ORDER BY 1;
```



INSERT SELECT 구문

INSERT INTO 테이블1 (컬럼1, 컬럼2, ...)

SELECT 컬럼2_1, 컬럼2_2, ... **FROM** 테이블2 **WHERE** 조건;

➤ 테이블2에서 검색한 컬럼의 데이터들을 테이블1의 컬럼들에 삽입

```
INSERT INTO emp1 (emp_id, emp_name)
```

```
SELECT employee_id, employee_name FROM employees;
```



INSERT (1)

INSERT INTO 테이블이름

VALUES (값1, 값2, 값3, ...);

- 테이블의 컬럼 순서대로 해당 컬럼 값을 입력함

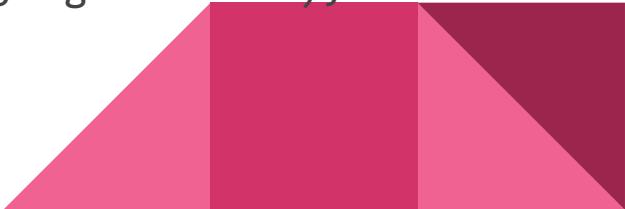
```
SQL> INSERT INTO member VALUES ('guest', 'guest1234', 'guest@test.com');
```

INSERT INTO 테이블이름 (컬럼1, 컬럼2, 컬럼3, ...)

VALUES (값1, 값2, 값3, ...);

- 데이터를 입력할 테이블, 해당 컬럼, 값들을 순서대로 기술하는 형태

```
SQL> INSERT INTO member (id, pw) VALUES ('guest', 'guest1234');
```



UPDATE (1)

UPDATE 테이블이름

SET 컬럼1 = 값1, 컬럼2 = 값2, ...


WHERE 조건;

- 테이블에 저장된 값을 변경

```
SQL> UPDATE member
```

```
SET pw = '123456', email = 'test@test.com'
```

```
WHERE id = 'guest';
```



DELETE (1)

DELETE FROM 테이블 **WHERE** 조건;

- 조건에 맞는 테이블의 내용을 삭제

```
SQL> DELETE FROM member WHERE id = 'test';
```

- WHERE 조건절을 기술하지 않는 경우 테이블의 모든 내용이 삭제

