

Oracle Database

DDL - Data Definition Language

DDL 요약

- ▶ CREATE TABLE : 테이블 생성
- ▶ ALTER TABLE : 테이블 관련 변경
- ▶ DROP TABLE : 테이블 삭제
- ▶ RENAME : 이름 변경
- ▶ TRUNCATE : 테이블의 모든 데이터 삭제
- ▶ COMMENT : 테이블에 설명 추가

테이블 생성

- ▶ CREATE TABLE 문 이용

- ▶ Syntax

```
CREATE TABLE [schema.]table_name  
    (column datatype [DEFAULT expr]  
      [column_constraints],  
      ..... ,  
      [table_constraints]);
```

- ▶ Oracle에서의 Table 관리

- ▶ Oracle은 Database의 공간을 Tablespace라는 논리적 공간으로 분할하여 관리
- ▶ Table을 만들기 위해서는 CREATE TABLE 권한과 객체를 생성할 수 있는 저장장소가 있어야 한다
- ▶ DBA는 user에게 권한을 주는 데이터 조작어(DCL) 명령을 사용한다
- ▶ 다른 user의 table을 참조하려면 참조되는 테이블은 동일한 데이터베이스 내에 있어야 한다
- ▶ 참조하는 테이블이 제약조건을 만드는 user의 소유가 아니라면 소유자 이름(Schema)이 제약조건에서 참조되는 table 이름 앞에 붙어야 한다
- ▶ DEFAULT 옵션을 이용하면 column의 기본값을 지정할 수 있다. 새로운 row 입력시 해당 컬럼에 값이 입력되지 않으면 default 옵션에 설정된 값이 자동으로 부여된다

테이블 생성

- ▶ 테이블명, 컬럼명, 데이터 타입 등 정의
- ▶ [연습] 오른쪽 DDL 쿼리를 이용, book 테이블을 만들어 봅시다
- ▶ DESC 명령을 이용, 생성된 테이블이 원하는 구조대로 만들어졌는지 확인해 봅니다

```
CREATE TABLE book (  
    book_id    NUMBER(5),  
    title      VARCHAR2(50),  
    author     VARCHAR2(10),  
    pub_date   DATE DEFAULT SYSDATE  
);
```



book_id	title	author	pub_date
1	토지	박경리	2005-03-12
2	슬램덩크	다케이코	2006-04-05
...

Subquery를 이용한 테이블 생성

- ▶ Subquery의 결과와 동일한 테이블 생성됨
- ▶ 질의 결과 레코드들이 포함됨
- ▶ NOT NULL 제약 조건만 상속됨

```
CREATE TABLE  account_employees
AS (
    SELECT  *
        FROM  employees
        WHERE job_id = 'FI_ACCOUNT'
) ;
```

Naming Rules

- ▶ 테이블, 컬럼 등의 이름 명명 규칙
 - ▶ 문자로 시작
 - ▶ 30자 이내
 - ▶ A-Z, a-z, 0-9, _, \$, #
 - ▶ 같은 사용자가 소유한 다른 객체의 이름과 겹치지 않아야 함
(다른 사용자 소유의 객체와는 같을 수도 있음)
 - ▶ 오라클 예약어는 사용할 수 없음

TABLE의 종류

- ▶ 사용자 테이블 (User Tables)

- ▶ 사용자에 의해 만들어지고 관리되는 테이블의 집합
- ▶ 사용자 정보를 포함함

- ▶ Data Dictionary

- ▶ 오라클 서버에 의해 만들어지고 관리되는 테이블의 집합
- ▶ 데이터베이스 정보를 포함

기본 데이터 타입

▶ 사용 빈도가 가장 높은 타입들

Data type	Description
VARCHAR2(size)	가변길이 문자열 (최대 4000byte)
CHAR(size)	고정길이 문자열 (최대 2000byte)
NUMBER(p,s)	가변길이 숫자. 전체 p자리 중 소수점 이하 s자리 (p:38, s:-84~127, 21Byte) 자리수 지정 없으면 NUMBER(38)
DATE	고정길이 날짜+시간, 7Byte

▶ 참고

- ▶ VARCHAR2와 CHAR의 차이점을 구분한다
- ▶ INT, FLOAT 등의 ANSI Type도 내부적으로 NUMBER(38)로 변환됨

기본 데이터 타입

Data type	Description
NCHAR(size)	national character set에 따라 결정되는 size 만큼의 고정길이 character data로 최대 2000byte까지 가능. 디폴트는 1 character.
NVARCHAR2 (size)	national character set에 따라 결정되는 size 만큼의 가변길이 character data로 최대 4000 byte까지 가능하며 반드시 길이를 정해 주어야 함.
LONG	가변 길이 character data로 최대 2 gigabyte까지 가능.
RAW (size)	가변 길이 raw binary data로 최대 2000 까지 가능하며 반드시 길이를 주어야 함.
LONG RAW	가변길이 raw binary data로 최대 2 gigabyte까지 가능.
BLOB	Binary data로 4 gigabyte까지 가능.
CLOB	Single-byte character data로 4 gigabyte까지 가능.
NCLOB	national character set까지 포함한 모든 character data로 4 gigabyte까지 가능.
BFILE	외부 화일로 저장된 binary data로 4 gigabyte까지 가능.
ROWID	Row의 물리적 주소를 나타내는 binary data로 extended rowid 는 10 byte, restricted rowid는 6 byte 길이.
TIMESTAMP	Date값을 미세한 초 단위까지 저장. NLS_TIMESTAMP_FORMAT 형식으로 처리.
INTERVAL YEAR TO MONTH	두 datetime 값의 차이에서 YEAR와 MONTH값 만을 저장.
INTERVAL DAY TO SECOND	두 datetime 값의 차이를 DAY, HOUR, MINUTE, SECOND 까지 저장.

ALTER TABLE

▶ 컬럼 추가 (ADD)

- ▶ `ALTER TABLE book ADD (pubs VARCHAR2(50));`

▶ 컬럼 수정 (MODIFY)

- ▶ `ALTER TABLE book MODIFY (title VARCHAR2(100));`

▶ 컬럼 삭제 (DROP)

- ▶ `ALTER TABLE book DROP (author);`

▶ UNUSED 컬럼

- ▶ `ALTER TABLE book SET UNUSED (author);`

- ▶ `ALTER TABLE book DROP UNUSED COLUMNS;`

기타 테이블 관련 명령

- ▶ 테이블 삭제 (DROP TABLE)

- ▶ `DROP TABLE book;`

- ▶ 데이터 삭제 (TRUNCATE TABLE)

- ▶ `TRUNCATE TABLE book;`

- ▶ Comment

- ▶ `COMMENT ON TABLE book IS 'this is comment';`

- ▶ `SELECT * FROM user_tab_comments;`

- ▶ RENAME

- ▶ `RENAME book TO article;`

- ▶ ROLLBACK 대상이 아님

Constraint(제약조건)

- ▶ Database 테이블 레벨에서 특정한 규칙을 설정함
- ▶ 예상치 못한 데이터의 손실이나 일관성을 어기는 데이터의 추가, 변경 등을 예방
- ▶ 종류
 - ▶ NOT NULL
 - ▶ UNIQUE
 - ▶ PRIMARY KEY
 - ▶ FOREIGN KEY
 - ▶ CHECK

제약조건 정의

▶ Syntax

- ▶ CREATE TABLE 테이블명 (
 컬럼명 DataType [DEFAULT 기본값][컬럼 제약 조건],
 컬럼명 DataType [DEFAULT 기본값][컬럼 제약 조건],
 ...
 [테이블 제약조건] ...);
- ▶ 컬럼 제약 조건 : [CONSTRAINT 이름] constraint_type
- ▶ 테이블 제약 조건 : [CONSTRAINT 이름] constraint_type(column, ...)

▶ 주의

- ▶ 제약조건에 이름을 부여하지 않으면 Oracle이 **Sys-Cn**의 형태로 자동 부여

제약조건

▶ NOT NULL

- ▶ NULL 값을 허용하지 않음
- ▶ 컬럼 형태로만 제약조건 정의할 수 있음(테이블 제약 조건 불가)

```
CREATE TABLE book (  
    book_id NUMBER(5) NOT NULL  
);
```

▶ UNIQUE

- ▶ 중복된 값을 허용하지 않음 (NULL은 들어올 수 있음)
- ▶ 복합 컬럼에 대해서도 정의 가능
- ▶ 자동적으로 인덱스 생성

```
CREATE TABLE book (  
    book_id NUMBER(5) CONSTRAINT c_nook_u UNIQUE  
);
```

제약조건

▶ PRIMARY KEY

- ▶ NOT NULL + UNIQUE (인덱스 자동 생성)
- ▶ 테이블 당 하나만 나올 수 있음
- ▶ 복합 컬럼에 대하여 정의 가능 (순서 중요)

```
CREATE TABLE book (  
    ...  
    PRIMARY KEY (book_id)  
);
```

▶ CHECK

- ▶ 임의의 조건 검사 조건식이 참이어야 변경 가능
- ▶ 동일 테이블의 컬럼만 이용 가능

```
CREATE TABLE book (  
    rate NUMBER CHECK (rate IN (1,2,3,4,5))  
);
```

실습

▶ [연습] author 테이블 생성

- ▶ book 테이블의 author 컬럼을 author 테이블로 분리, 관리하고자 합니다.
- ▶ 다음 조건으로 author 테이블을 생성해 봅시다

1. 컬럼 타입

```
author_id : NUMBER(10)
author_name : VARCHAR2(100)
author_desc : VARCHAR2(500)
```

2. 제약조건

```
author_id : primary key
author_name : NOT NULL
```


실습

- ▶ [연습] book 테이블 변경
 - ▶ book 테이블의 author 컬럼을 삭제해 봅니다
 - ▶ book 테이블에 author_id 컬럼을 추가합니다
 - ▶ author 테이블의 author_id 컬럼과 같은 형식(NUMBER(10))으로 지정합니다

제약조건

▶ FOREIGN KEY

- ▶ 참조 무결성 제약
- ▶ 일반적으로 REFERENCE 테이블의 PK를 참조
- ▶ REFERENCE 테이블에 없는 값은 삽입 불가
- ▶ REFERENCE 테이블의 레코드 삭제시 동작
 - ▶ ON DELETE CASCADE : 해당하는 FK를 가진 참조행도 삭제
 - ▶ ON DELETE SET NULL : 해당하는 FK를 NULL로 바꿈

```
CREATE TABLE book (  
    ...  
    author_id NUMBER(10),  
    CONSTRAINT c_book_fk FOREIGN KEY (author_id)  
    REFERENCE author(id)  
    ON DELETE SET NULL  
);
```

ADD / DROP CONSTRAINTS

▶ 제약조건 추가

- ▶ ALTER TABLE 테이블명 ADD CONSTRAINT ...
- ▶ NOT NULL은 추가 못함

```
ALTER TABLE book ADD CONSTRAINT c_book_fk  
FOREIGN KEY (author_id) REFERENCES author(author_id);
```

▶ 제약조건 삭제

- ▶ ALTER TABLE 테이블명 DROP CONSTRAINT 제약조건명
- ▶ PRIMARY KEY의 경우 FK 조건이 걸린 경우에는 CASCADE로 삭제해야 함

```
ALTER TABLE book DROP CONSTRAINT c_book_fk;  
ALTER TABLE author DROP PRIMARY KEY CASCADE;
```

ENABLE/DISABLE CONSTRAINTS

▶ 제약조건 비활성화

- ▶ 제약조건 검사를 중지함
- ▶ CASCADE를 사용하여 의존되어 있는 다른 조건을 함께 중지시킬 수 있음
- ▶ 대규모 데이터 변경 등의 속도를 빠르게 함
- ▶ UNIQUE, PRIMARY KEY의 경우 인덱스 제거됨

```
ALTER TABLE book DISABLE CONSTRAINT c_book_fk CASCADE;
```

▶ 제약조건 활성화

- ▶ 중지되어 있던 제약조건 검사를 활성화
- ▶ UNIQUE, PRIMARY KEY의 경우 인덱스 자동 생성

```
ALTER TABLE book ENABLE CONSTRAINT c_book_fk CASCADE;
```

CASCADE CONSTRAINT

- ▶ 제약조건이 걸려 있는 테이블이나 컬럼은 삭제시 에러 발생

```
ALTER TABLE book DROP (author_id) CASCADE CONSTRAINT;
```

- ▶ 컬럼이나 테이블을 DROP 할 때 관련 제약조건도 함께 삭제할 때

```
DROP TABLE book CASCADE CONSTRAINT;
```

Data Dictionary

- ▶ Oracle이 관리하는 모든 정보를 저장하는 카탈로그
- ▶ 내용
 - ▶ 모든 스키마 객체 정보, 스키마 객체의 공간 정보, 컬럼의 기본값, 제약 조건 정보, 오라클 사용자 정보, 권한 및 롤 정보, 기타 데이터베이스 정보 ...
- ▶ Base-Table과 View로 구성됨
 - ▶ VIEW의 Prefix
 - ▶ USER : 로그인한 사용자 레벨
 - ▶ ALL : 모든 사용자 정보
 - ▶ DBA : 관리자
- ▶ SYS scheme에 속함
- ▶ 주의
 - ▶ DICTIONARY의 테이블이나 컬럼 이름은 모두 대문자 사용!

Data Dictionary

: Example

- ▶ 모든 Dictionary 정보 확인

```
SELECT * FROM DICTIONARY
```

- ▶ 사용자 스키마 객체 확인 (테이블)

```
SELECT object_name  
FROM user_objects  
WHERE object_type = 'TABLE';
```

- ▶ 제약조건 확인 (BOOK 테이블의 예)

```
SELECT constraint_name, constraint_type, search_condition  
FROM user_constraints  
WHERE table_name = 'BOOK';
```

Data Dictionary

: Example 2

- ▶ 제약조건 컬럼 확인

```
SELECT constraint_name, column_name  
FROM user_cons_columns  
WHERE table_name = 'EMP';
```

- ▶ 모든 사용자 확인 (dba에서 확인 가능)

```
SELECT username, default_tablespace, temporary_tablespace  
FROM DBA_USERS;
```


Oracle Database

DCL - Data Control Language

Database User

- ▶ 데이터베이스 사용자
 - ▶ 데이터베이스에 접속하여 데이터를 이용하기 위해 접근하는 모든 사람
 - ▶ 이용 목적에 따라 데이터베이스 관리자(**Database Administrator: DBA**), 최종 사용자, 응용프로그래머로 구분
- ▶ Oracle 데이터베이스 생성시 기본적으로 생성되는 계정
 - ▶ SYS
 - ▶ Data Dictionary Table의 소유자
 - ▶ SYSTEM
 - ▶ 각종 Dictionary View의 소유자
- ▶ 각 **USER**는 동일한 이름의 **SCHEMA** 소유자로, 해당 **SCHEMA**의 모든 객체들에 접근할 수 있다

사용자 관리

▶ Syntax

- ▶ 사용자 생성 : `CREATE USER username IDENTIFIED BY password;`
- ▶ 비밀번호 변경 : `ALTER USER username IDENTIFIED BY password;`
- ▶ 사용자 삭제 : `DROP USER username [CASCADE];`

▶ 주의사항

- ▶ 일반적으로 **DBA**의 일
- ▶ 사용자를 생성하려면 **CREATE USER** 권한 필요
- ▶ 생성된 사용자가 **Login** 하려면 **CREATE SESSION** 권한 필요
- ▶ 일반적으로 **CONNECT, RESOURCE**의 **ROLE**을 부여하면 일반사용자 역할을 수행할 수 있음

사용자 관리

- ▶ [연습] bituser 사용자를 생성하고 비밀번호를 변경해 봅니다

```
SQL> conn system/manager
Connected.
SQL> CREATE USER bituser IDENTIFIED BY bituser;
User created.
SQL> conn bituser/bituser
ERROR:
ORA-01045: user BITUSER lacks CREATE SESSION privilege;
logon denied
```

- ▶ USER를 생성해도 권한(Privilege)을 주지 않으면 아무 일도 할 수 없음
 - ▶ 위 예에서는 사용자가 CREATE SESSION 권한을 갖고있지 않아 오류 발생
- ▶ 사용자는 DBA가 초기화한 암호를 갖게 되며, 로그인 후 ALTER USER 명령을 이용, 각자의 암호를 재설정

사용자 정보 확인

▶ 관련 DICTIONARY

- ▶ USER_USERS : 현재 사용자 관련 정보
- ▶ ALL_USERS : DB의 모든 사용자 정보
- ▶ DBA_USERS : DB의 모든 사용자 상세 정보 (DBA만 사용 가능)

```
SELECT * FROM USER_USERS;
```

▶ [연습] USER_USERS, ALL_USERS, DBA_USERS Dictionary로부터 user 목록을 각각 출력해 봅니다

```
SQL> conn scott/tiger
SQL> SELECT username FROM user_users;
SQL> SELECT username FROM all_users;
SQL> SELECT username FROM dba_users;
(ERROR 발생, DBA 권한을 갖고 있지 않음)
SQL> conn system/manager
SQL> /
(SUCCESS, system 계정은 DBA 권한을 갖고 있음)
```

권한(Privilege)과 롤(Role) : Privileges

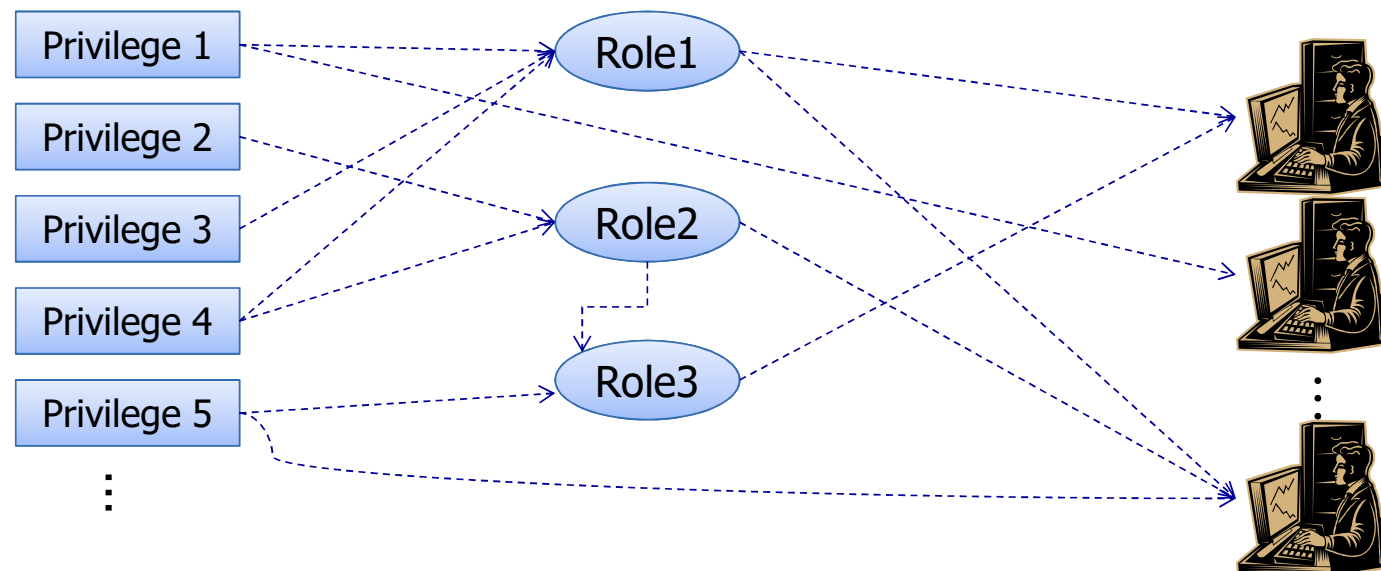
▶ 권한(Privilege)

- ▶ 사용자가 특정 **SQL** 문을 실행하거나 특정 정보에 접근할 수 있는 권리
- ▶ 사용자는 작업에 요구되는 관련 권한에 대한 허가(**GRANT**)를 받아야 한다
- ▶ 종류
 - ▶ 시스템 권한(**80여개**) :
 - ▶ 스키마 객체 권한

권한(Privilege)과 룰(Role)

▶ 룰(Role)

- ▶ 권한을 쉽게 관리하기 위하여 특정한 종류별로 묶어놓은 그룹



GRANT / REVOKE

- ▶ 권한/롤을 부여(GRANT)하거나 회수(REVOKE)

- ▶ Syntax (System Privileges)

```
GRANT [system_priv|role [,system_priv|role ...]  
      TO {user [,user ...]|role|PUBLIC}  
      [WITH ADMIN OPTION];
```

```
REVOKE [system_priv|role [,system_priv|role ...]  
       FROM {user [,user ...]|role|PUBLIC};
```

- ▶ Syntax (Object Privileges)

```
GRANT {[object_priv [,object_priv ...]|ALL}  
      ON object TO {user [,user ...]|role|PUBLIC}  
      [WITH ADMIN OPTION];
```

```
REVOKE {[object_priv [,object_priv ...]|ALL}  
       FROM {user [,user ...]|role|PUBLIC};
```

- ▶ GRANT, REVOKE 문장은 실행 즉시 효력을 발휘한다 (재접속 등 필요 없음)

GRANT / REVOKE

- ▶ 시스템 권한: 관리자로 수행 (ADMIN OPTION/GRANT ANY PRIVILEGES 권한)

```
GRANT create session TO user1;
```

```
REVOKE create session FROM user1;
```

- ▶ 스키마 객체 권한

```
GRANT select ON emp TO user1;
```

```
REVOKE select ON emp FROM user1;
```

- ▶ WITH GRANT OPTION

- ▶ 해당 권한을 받은 사용자가 다시 제3자에게 권한을 부여할 수 있도록 하는 옵션
- ▶ 권한을 REVOKE 하면 해당 사용자가 3자에게 부여한 권한들도 함께 회수됨

```
GRANT select ON emp TO user2  
WITH GRANT OPTION;
```

GRANT / REVOKE

- ▶ [연습] CREATE SESSION 권한을 부여해 봅니다

```
SQL> conn system/manager
SQL> GRANT create session TO bituser;
SQL> conn bituser/bituser
SQL> CREATE TABLE test (a NUMBER);
(* ERROR 발생. WHY? )
```

- ▶ 사용자에게 CONNECT, RESOURCE ROLE을 부여하면 TABLE 생성 등을 할 수 있다

```
SQL> conn system/manager
SQL> GRANT connect, resource TO bituser;
SQL> conn bituser/bituser;
SQL> CREATE TABLE test (a NUMBER);
Table created.
SQL> DESC test
```

ROLE

- ▶ ROLE을 생성한 후 Role에 Privilege를 Grant하여 Role 관리

- ▶ 주로 DBA가 하는 작업

```
CREATE ROLE reviewer;  
GRANT select any table TO reviewer;  
GRANT create session, resource TO reviewer;
```

- ▶ 특정 Role을 사용자에게 Grant / Revoke

```
GRANT reviewer TO user3;
```

권한의 확인

▶ 관련 Dictionary

- ▶ ROLE_SYS_PRIVS: System privileges granted to roles
- ▶ ROLE_TAB_PRIVS: Table privileges granted to roles
- ▶ USER_ROLE_PRIVS: Roles accessible by the user
- ▶ USER_TAB_PRIVS_MADE: Object privileges granted on the user's object
- ▶ USER_TAB_PRIVS_RECD: Object privileges granted to the user
- ▶ USER_COL_PRIVS_MADE: Object privileges granted on the columns of the user's object
- ▶ USER_COL_PRIVS_RECD: Object privileges granted to the user on specific columns
- ▶ USER_SYS_PRIVS: Lists system privileges granted to the user

```
SELECT * FROM USER_ROLE_PRIVS ;  
SELECT * FROM ROLE_SYS_PRIVS ;
```

권한의 확인

▶ [연습]

- ▶ 현재 사용자에게 부여된 Role을 조회해 봅시다

```
SQL> SELECT * FROM user_role_privs;
```

- ▶ 일반적으로 부여되는 CONNECT와 RESOURCE 안에 있는 세부적인 Privilege 들을 확인해 봅시다

```
SQL> DESC role_sys_privs;  
SQL> SELECT privilege FROM role_sys_privs WHERE  
role='RESOURCE';  
SQL> SELECT privilege FROM role_sys_privs WHERE  
role='CONNECT';
```