DDL : Data Definition Language

데이터 정의어(Data Definition Language)

데이터 정의어(이하 DDL)는 데이터 간에 관계를 정의하여 데이터베이스 구조를 설정하는 SQL 문장

구분	명령어	설명
데이터베이스	CREATE DATABASE	데이터베이스를 생성한다.
	ALTER DATABASE	데이터베이스를 변경한다.
테이블	CREATE TABLE	테이블을 생성한다.
	ALTER TABLE	테이블을 변경한다.
	DROP TABLE	테이블을 제거한다.
테이블 스페이스	CREATE TABLESPACE	테이블 스페이스를 생성한다.
	ALTER TABLESPACE	테이블 스페이스를 변경한다.
	DROP TABLESPACE	테이블 스페이스를 제거한다.

데이터 정의어(Data Definition Language)

구분	명령어	설명
인덱스	CREATE INDEX	인덱스를 생성한다.
	ALTER INDEX	인덱스를 변경한다.
	DROP INDEX	인덱스를 제거한다.
뷰	CREATE VIEW	뷰를 생성한다.
	ALTER VIEW	뷰를 변경한다.
	DROP VIEW	뷰를 제거한다.
동의어	CREATE SYNOMYM	동의어를 생성한다.
	DROP SYNOMYM	동의어를 제거한다.
사용자	CREATE USER	사용자를 생성한다.
	ALTER USER	사용자를 변경한다.
	DROP USER	사용자를 제거한다.

데이터 정의어(Data Definition Language)

구분	명령어	설명
함수	CREATE FUNCTION	함수를 생성한다.
	ALTER FUNCTION	함수를 변경한다.
	DROP FUNCTION	함수를 제거한다.
프러시저	CREATE PROCEDURE	프러시저를 생성한다.
	ALTER PROCEDURE	프러시저를 변경한다.
	DROP PROCEDURE	프러시저를 제거한다.
타입	CREATE TYPE	타입을 생성한다.
	ALTER TYPE	타입을 변경한다.
	DROP TYPE	타입을 제거한다.

데이터 정의어(Data Definition Language)

구분	명령어	설명
권한	GRANT	사용자에게 특권을 부여한다.
	REVOKE	사용자에게 특권을 회수한다.
역할	CREATE ROLE	역할을 생성한다.
	ALTER ROLE	역할을 변경한다.
	DROP ROLE	역할을 제거한다.
객체	RENAME	테이블, 뷰, 동의어, 시퀀스 등의 스키마 객체의 이름을 변경한다.

Create Tables to Store Data

테이블 인스턴스 –테이블의 구조와 칼럼의 특성을 요약

Symbols	Explanation
PK	기본 키 열
FK	외래 키 열
FK1, FK2	동일한 테이블에 있는 두 개의 외부 키
NN	NOT NULL 열
U	고유 열

Create Tables to Store Data

Table Instance Chart

Table Name: E_EMP

Column Name	ID	LAST_NAME	FIRST_NAME	START_DATE	SALARY	MANAGER_ID	DEPT_ID
Key Type	PK					FK1	FK2
Nulls / Unique	NN, U					NN	
FK Ref Table						S_EMP	S_DEPT
FK Ref Column						ID	ID
Data Type	NUMBER	CHAR	CHAR	DATE	NUMBER	NUMBER	NUMBER
Maximum Length	7	25	25		11	7	7
Sample Data	1	Alexander	Lee	2022-09-01	1500		10
	2	Jonathan	Hong	2022-10-05	1000	1	20

Create Tables to Store Data

Syntax

```
CREATE TABLE [user.] table_name
        ({column_name datatype | table_constraint
          ,column_name datatype | table_constraint});
```

where	*table_name*	테이블 이름
	column_name	열 이름
	datatype	데이터 타입
	table_constraint	제약 조건

Create Tables to Store Data

스키마 객체 이름

- 이름 부여 시 따옴표(“ ”) 없는 식별자는 모두 대문자로 간주
- 따옴표 있는 식별자는 대소문자를 구분

모두 다른 식별자	모두 같은 식별자
department **“department”** **“Department”**	**department** **DEPARTMENT** **“DEPARTMENT”**

식별자를 기술 할 때 규칙

- 길이가 **30bytes**를 넘으면 안된다.
- 따옴표 없는 식별자는 알파벳, 한글, 숫자, 언더바(_), **$**, #만 사용. 단, 숫자, ‘**$**’, ‘#’는 첫 글자로 올 수 없다.
- 따옴표 있는 식별자는 공백을 포함한 어떤 문자 사용 가능. 다만, 큰따옴표(“ ”)는 사용 불가능.
- 하나의 네임스페이스 안에 서로 다른 두 객체가 동일한 이름을 사용할 수 없다.

Create Tables to Store Data

Tibero에서 제공하는 데이터 타입

구분	데이터 타입
문자형	CHAR, VARCHAR, VARCHAR2, NCHAR, NVARCHAR, NVARCHAR2, RAW, LONG, LONG RAW
숫자형	NUMBER, INTEGER, FLOAT
날짜형	DATE, TIME, TIMESTAMP, TIMESTAMP WITH TIME ZONE, TIMESTAMP WITH LOCAL TIME ZONE
간격형	INTERVAL YEAR TO MONTH, INTERVAL DAY TO SECOND
대용량 객체형	CLOB, BLOB, XMLTYPE
내재형	ROWID

Create Tables to Store Data

Datatype	Description
CHAR(s)	고정된 문자열 길이, 최대 2,000자까지 선언, 문자열의 길이가 0인 값은 NULL로 인식
	CHAR(size[BYTE | CHAR]) -> EXAM CHAR(10)
VARCHAR2(s)	가변 문자열 길이, 최대 4,000자까지 선언, 문자열의 길이가 0인 값은 NULL로 인식
	VARCHAR2(size[BYTE | CHAR]) -> EXAM VARCHAR2(10)
VARCHAR(s)	VARCHAR2 타입과 동일
LONG	VARCHAR2와 비슷하지만, 최대 2GB까지 선언
DATE	연도는 BC 9,999 ~ AD 9,999까지 표현,
NUMBER(p,s)	정수 또는 실수를 저장, 음양으로 절댓값이 1.0×10^{-130}보다 크거나 같고, 1.0×10^{126}보다 작은 38자리의 수를 표현할 수 있으며 0과 ±무한대를 포함한다.

Create Tables to Store Data

데이터 무결성 제약 조건

Constraint	Description
NOT NULL	NULL값을 허용하지 않고, 반드시 데이터를 입력. 제약조건이 NULL이면 해당 컬럼은 NULL값을 허용.
UNIQUE	해당 칼럼이 중복되는 데이터가 존재할 수 없는 유일성을 보장하는 제약조건
PRIMARY KEY	NOT NULL, UNIQUE 제약 조건의 결합과 같다. 테이블 또는 뷰는 단 한 개의 PRIMARY KEY 제약조건을 가질 수 있다.
FOREIGN KEY	같은 테이블 또는 서로 다른 두 개 테이블의 키 컬럼 사이의 관계
CHECK	expr로 표현한 조건이 항상 참이 되도록 유지. 특정 조건을 평가 후 만족하지 못하면 에러 발생

Create Tables to Store Data

EXAMPLE

Table Name: S_REGION

Column Name	ID	NAME
Key Type	PK	
Nulls/Unique	NN, U	NN, U
Sample data	1	North America
	2	South America
	3	Africa/Middle East
	4	Asia
	5	Europe

```
SQL> CREATE TABLE s_region
  2     ( id                          NUMBER(7),
  3      name                         VARCHAR2(50)
  4              CONSTRAINT s_region_name_nn NOT NULL,
  5              CONSTRAINT s_region_id_pk PRIMARY KEY (id),
  6              CONSTRAINT s_region_name_uk UNIQUE (name));
Table 'S_REGION' created.
```

Create Tables to Store Data

EXAMPLE

Table Name: S_DEPT

Column Name	deptno	dname	loc
Key Type	PK		
Nulls/Unique			
Datatype	NUMBER	VARCHAR2	VARCHAR2
Maximum Length	2	14	13
Sample data	10	ACCOUNTING	NEW YORK
	20	RESEARCH	DALLAS
	30	SALES	CHICAGO
	40	OPERATIONS	BOSTON

```
SQL> CREATE TABLE s_dept
  2     ( deptno                    NUMBER(2),
  3       dname                     VARCHAR2(14),
  4       loc                       VARCHAR2(13),
  5               CONSTRAINT s_dept_pk PRIMARY KEY(deptno));
Table 'S_DEPT' created.
```

Create Tables to Store Data

EXAMPLE

Table Name: S_DEPT

```
INSERT INTO S_DEPT VALUES (10, 'ACCOUNTING', 'NEW YORK');
INSERT INTO S_DEPT VALUES (20, 'RESEARCH',   'DALLAS');
INSERT INTO S_DEPT VALUES (30, 'SALES',      'CHICAGO');
INSERT INTO S_DEPT VALUES (40, 'OPERATIONS', 'BOSTON');
```

Create Tables to Store Data

EXAMPLE

Table Name: S_EMP

Column Name	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
Key Type	PK			FK1				FK2
Nulls/Unique	NN, U	NN						
FK Ref Table				S_EMP				S_DEPT
FK Ref Column				EMPNO				DEPTNO
Datatype	NUMBER	VARCHAR2	VARCHAR2	NUMBER	DATE	NUMBER	NUMBER	NUMBER
MaxLength	4	10	9	4		7	7	2
Sample data	7839	KING	PRESIDENT	null	1981-11-17	5000	null	10
	7566	JONES	MANAGER	7839	1981-02-04	2975	null	20
	7902	FORD	ANALYST	7566	1981-03-12	3000	null	20

Create Tables to Store Data

```
SQL> create table s_emp
( empno NUMBER (7) CONSTRAINT s_emp_empno_nn NOT NULL,
  ename VARCHAR2 (10) CONSTRAINT s_emp_ename_nn NOT NULL,
  job VARCHAR2 (9),
  mgr NUMBER (4),
  hiredate DATE,
  sal NUMBER (7),
  comm NUMBER (7),
  deptno NUMBER (2),
  constraint s_emp_id_pk PRIMARY KEY (empno),
  constraint s_emp_mgr_fk FOREIGN KEY(mgr) REFERENCES s_emp(empno),
  constraint s_emp_deptno_fk FOREIGN KEY(deptno) REFERENCES s_dept(deptno));
```

Create Tables to Store Data

EXAMPLE

Table Name: S_EMP

```
INSERT INTO S_EMP VALUES (7839,'KING','PRESIDENT',NULL,'81-11-17',5000,NULL,10);
INSERT INTO S_EMP VALUES (7698,'BLAKE','MANAGER',7839,'81-05-01',2850,NULL,30);
INSERT INTO S_EMP VALUES (7782,'CLARK','MANAGER',7839,'81-05-09',2450,NULL,10);
INSERT INTO S_EMP VALUES (7566,'JONES','MANAGER',7839,'81-04-01',2975,NULL,20);
INSERT INTO S_EMP VALUES (7654,'MARTIN','SALESMAN',7698,'81-09-10',1250,1400,30);
INSERT INTO S_EMP VALUES (7499,'ALLEN','SALESMAN',7698,'81-02-11',1600,300,30);
INSERT INTO S_EMP VALUES (7844,'TURNER','SALESMAN',7698,'81-08-21',1500,0,30);
INSERT INTO S_EMP VALUES (7900,'JAMES','CLERK',7698,'81-12-11',950,NULL,30);
INSERT INTO S_EMP VALUES (7521,'WARD','SALESMAN',7698,'81-02-23',1250,500,30);
INSERT INTO S_EMP VALUES (7902,'FORD','ANALYST',7566,'81-12-11',3000,NULL,20);
INSERT INTO S_EMP VALUES (7369,'SMITH','CLERK',7902,'80-12-09',800,NULL,20);
INSERT INTO S_EMP VALUES (7788,'SCOTT','ANALYST',7566,'82-12-22',3000,NULL,20);
INSERT INTO S_EMP VALUES (7876,'ADAMS','CLERK',7788,'83-01-15',1100,NULL,20);
INSERT INTO S_EMP VALUES (7934,'MILLER','CLERK',7782,'82-01-11',1300,NULL,10);
```

CONFIRM THE STRUCTURE OF A TABLE

테이블 구조 확인

Syntax

DESCRIBE *table_name*

Example

```
SQL> DESCRIBE s_emp;

COLUMN_NAME                    TYPE           CONSTRAINT
-------------------------------------- ----------------- --------------------
EMPNO                          NUMBER(7)      PRIMARY KEY
                                               NOT NULL
ENAME                          VARCHAR(10)    NOT NULL
JOB                            VARCHAR(9)
MGR                            NUMBER(4)      REFERENTIAL
HIREDATE                       DATE
SAL                            NUMBER(7)
COMM                           NUMBER(7)
DEPTNO                         NUMBER(2)      REFERENTIAL

INDEX_NAME                     TYPE           COLUMN_NAME
------------------------------ ---------------------- --------------------
S_EMP_ID_PK                    NORMAL         EMPNO
```

ADD A COLUMN TO A TABLE

테이블에 칼럼 추가하기

Syntax

```
ALTER TABLE table_name
ADD ( column_name datatype
[, column_name datatype]…)
```

Example

```
SQL> ALTER TABLE s_region
  2  ADD (comments VARCHAR2 (255));
Table 'S_REGION' altered.
```

MODIFY A COLUMN

테이블에 존재하는 칼럼 속성 변경
데이터 타입, 기본값, 제약조건 변경

Syntax

```
ALTER TABLE table_name
MODIFY ( column_name datatype
[, column_name datatype]…)
```

Example

S_EMP 테이블의 ENAME 칼럼의 길이 20으로 변경 및 SAL 값은 NULL이 될 수 없음

```
SQL> ALTER TABLE s_emp
    2    MODIFY (ename VARCHAR2 (20));
Table 'S_EMP' altered.

SQL> ALTER TABLE s_emp
    2    MODIFY (SAL NOT NULL);
Table 'S_EMP' altered.
```

ADD AND REMOVE DATA CONSTRAINTS

Example

S_EMP 테이블의 외래키 삭제 및 추가

```
SQL> ALTER TABLE s_emp
  2    DROP CONSTRAINT s_emp_mgr_fk;
 Table 'S_EMP' altered.

SQL> ALTER TABLE s_emp
  2    ADD CONSTRAINT s_emp_mgr_fk
  3    FOREIGN KEY (mgr) REFERENCES s_emp(empno);
Table 'S_EMP' altered.
```

DROP A TABLE

DROP TABLE 명령어를 사용해 데이터베이스에서 테이블 제거.
DROP TABLE 명령어는 한번 실행되면 취소할 수 없다.

Syntax

```
DROP TABLE table_name
```

Example

S_REGION 테이블 삭제

```
SQL> DROP TABLE s_region;

Table 'S_REGION' dropped.
```

Simplify Data Access with Views

- 테이블 뷰를 만들어 논리적 하위 집합 또는 데이터 조합을 표시한다.
- 뷰는 실제 데이터가 포함되지 않는다.
- 뷰 이름은 테이블과 같은 네임스페이스를 사용하므로 스키마 내 다른 이름과 중복되면 안된다.

- 장점
  - 접근 제어로 보안 제공
  - 데이터 관리가 편리

- 단점
  - 삽입, 삭제, 갱신 연사에 제약이 있다.

CREATE VIEWS

Syntax

```
CREATE VIEW view_name [ (alias, [alias]…)
AS query
WHERE
WITH CHECK OPTION
```

where	*view_name*	뷰 이름
	alias	별칭
	query	SELECT 문
	WITH CHECK OPTION	해당 옵션을 사용하면 INSERT/UPDATE 가능
	WITH READ ONLY	읽기 전용 뷰
	constraint	CHECK OPTION에 할당된 제약조건

CREATE VIEWS

Example

부서 번호가 10인 직원 번호, 이름, 직무가 포함된 뷰를 생성

```
SQL> CREATE VIEW empvu10
  2 AS SELECT empno, ename, job
  3 FROM s_emp
  4 WHERE deptno = 10;

View 'EMPVU10' created.

SQL> SELECT *
  2 FROM empvu10;

    EMPNO ENAME              JOB
---------- -------------------- ---------
     7782 CLARK              MANAGER
     7839 KING             PRESIDENT
     7934 MILLER             CLERK

3 rows selected.
```

CREATE VIEWS

Example

부서 번호가 20인 뷰를 생성. 직원 번호는 ID , 이름은 EMPLOYEE, 직무는 TITLE로 표현.

```
SQL> CREATE VIEW empvu20 (id, employee, title)
  2 AS SELECT empno, ename, job
  3 FROM s_emp
  4 WHERE deptno = 20;

View 'EMPVU20' created.

SQL> SELECT *
  2 FROM empvu20;

        ID EMPLOYEE           TITLE
---------- -------------------- ---------
      7369 SMITH              CLERK
      7566 JONES              MANAGER
      7788 SCOTT              ANALYST
      7876 ADAMS              CLERK
      7902 FORD               ANALYST

5 rows selected.
```

CREATE VIEWS

Example

월급이 1500 이상인 뷰를 생성. 직원 번호는 ID , 이름은 NAME, 월급은 MONTHLY_SALARY로 표현.

```
SQL> CREATE VIEW salvu1500
   AS SELECT empno ID, ename NAME, sal MONTHLY_SALARY
   FROM s_emp
   WHERE sal >= 1500;

View 'SALVU1500' created.

SQL> SELECT *
  2 FROM salvu1500;

        ID NAME       MONTHLY_SALARY
---------- ---------- --------------
      7499 ALLEN              1600
      7566 JONES              2975
      7698 BLAKE              2850
      7782 CLARK              2450
      7788 SCOTT              3000
      7839 KING               5000
      7844 TURNER             1500
      7902 FORD               3000

8 rows selected.
```

CREATE VIEWS

Example

S_EMP 테이블에서 부서 번호가 30인 뷰를 'WITH CHECK OPTION'을 이용하여 생성

```
SQL> CREATE VIEW empvu30
  2 AS SELECT *
  3 FROM s_emp
  4 WHERE deptno = 30
  5 WITH CHECK OPTION;
```

부서 번호를 20으로 업데이트

```
UPDATE empvu30
  SET deptno=20
  WHERE deptno=30;
TBR-10010: Statement does not satisfy the WHERE clause of the view.
```

부서 번호가 30인 새로운 사원 정보 입력

```
SQL> INSERT INTO empvu30
   VALUES (9999, 'TABA', 'STUDENT', NULL, '22-10-05', 1500, NULL, 30);

1 row inserted.
```

CONFIRM VIEW NAMES AND STRUCTURES

USER_VIEWS에서 현재 사용자에 속한 뷰의 정보를 조회할 수 있다.

Example

```
SQL> DESCRIBE user_views;

COLUMN_NAME                    TYPE           CONSTRAINT
------------------------------ -------------- ----------
VIEW_NAME                      VARCHAR(128)
TEXT                           LONG
```

Example

```
SQL> SELECT * FROM user_views;

VIEW_NAME
--------------------------------------------------------------------------------
TEXT
--------------------------------------------------------------------------------
SALVU1500
SELECT empno ID, ename NAME, sal MONTHLY_SALARY
   FROM s_emp
   WHERE sal >=
```

DROP A VIEW

Syntax

```
DROP VIEW view_name
```

Example

EMPVU10 뷰 삭제

```
SQL> DROP VIEW empvu10;

View 'EMPVU10' dropped.
```

Generate Primary Key Value

- 유일한 연속적 값을 생성할 수 있는 스키마 객체
- 기본 키 또는 유일 키에 값을 넣을 때 사용

Syntax

```
CREATE SEQUENCE sequence_name
   [ INCREMENT by n ]
   [ START WITH n ]
   [ {MAXVALUE n | NOMAXVALUE } ]
   [ {CYCLE | NOCYCLE } ]
   [ {CACHE n | NOCACHE } ]
   [ {ORDER | NOORDER } ]
```

where	*table_name*	테이블 이름

Generate Primary Key Value

where	*table_name*	테이블 이름
	INCREMENT BY	시퀀스 간격(default : 1)
	START WITH	시퀀스 시작 값
	MAXVALUE	시퀀스 최댓값
	NOMAXVALUE	최댓값 지정 x
	CYCLE	최댓값 도달 시 재시작
	CACHE	캐시를 사용해서 미리 할당 (default : 20)
	ORDER	시퀀스 값 순서 유지

Generate Primary Key Value

Example

50부터 시작하고 10씩 증가하는 시퀀스 생성

```
SQL> CREATE SEQUENCE s_dept_id
  2 MINVALUE 1
  3 MAXVALUE 99999
  4 INCREMENT BY 10
  5 START WITH 50
  6 NOCACHE
  7 NOORDER
  8 NOCYCLE;
```

CONFIRM SEQUENCES

USER_SEQUENCES Columns

Column	Description
SEQUENCE_NAME	시퀀스 이름
MIN_VALUE	최솟값
MAX_VALUE	최댓값
INCREMENT_BY	증가 값
CYCLE_FLAG	반복 유무
ORDER_FLAG	순서 유무
CACHE_SIZE	캐시할 시퀀스 번호 수
LAST_NUMBER	디스크에 기록된 마지막 시퀀스 번호

CONFIRM SEQUENCES

Example

```
SQL> DESC user_sequences;

COLUMN_NAME                    TYPE           CONSTRAINT
------------------------------ -------------- ----------
SEQUENCE_NAME                  VARCHAR(128)
MIN_VALUE                      NUMBER
MAX_VALUE                      NUMBER
INCREMENT_BY                   NUMBER
CYCLE_FLAG                     VARCHAR(1)
ORDER_FLAG                     VARCHAR(1)
IF_AVAIL                       VARCHAR(1)
CACHE_SIZE                     NUMBER
LAST_NUMBER                    NUMBER
```

CONFIRM SEQUENCES

Example

```
SQL> SELECT sequence_name, min_value, max_value, increment_by, cycle_flag
  2 FROM user_sequences;

SEQUENCE_NAME MIN_VALUE  MAX_VALUE INCREMENT_BY CYCLE_FLAG
------------------------------------------------------------------------------------------------
 S_EMP_ID                       1          99999                    10                    N


1 row selected.
```

REFERENCE PRIMARY KEY VALUES

INSERT 명령에서 시퀀스를 참조하여 값을 자동으로 생성

Expression	Description
sequence_name.NEXTVAL	시퀀스의 다음 값을 반환
sequence_name.CURRVAL	시퀀스의 마지막 값은 반환

Example

새로운 부서 생성

```
SQL> INSERT INTO s_dept
  2  VALUES (s_dept_id.NEXTVAL, 'HR', 'SEOUL');

SQL> INSERT INTO s_dept
   2 VALUES (s_dept_id.NEXTVAL, 'FINANCE', 'MILPITAS');

SQL> SELECT * FROM s_dept;

   DEPTNO DNAME         LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
        50 HR             SEOUL
        60  FINANCE       MILPITAS
```

DROP A SEQUENCE

Syntax

```
DROP SEQUENCE sequence_name
```

Example

S_DEPT_ID 시퀀스 삭제

```
SQL> DROP SEQUENCE s_dept_id;

Sequence 'S_DEPT_ID' dropped.
```

Improve Query Performance

- 데이터베이스 테이블에 하나 이상의 인덱스를 작성하여 일부 쿼리 성능을 향상 시킬 수 있다.
- 자주 사용되는 **WHERE** 조건이나 **JOIN**
- 많은 양의 데이터 값을 가진 열
- 테이블 전체 데이터 중 **10-15%** 데이터를 처리하는 경우 효과적
- 테이블이 작거나 자주 업데이트 되는 경우 인덱스는 비효율적.

데이터베이스에서 테이블 데이터가 액세스 되는 방법

Access Method	Description
by ROWID	데이터의 정확한 위치를 나타내는 행 주소를 사용한 방법
FULL-TABLE SCAN	테이블의 모든 행을 순차적으로 검색하는 방법
by INDEX	열 값의 정렬된 트리 구조를 사용한 이진 검색

Improve Query Performance

Example

ROWID로 테이블 데이터 액세스

```
SQL> SELECT ename
  2 FROM s_emp
  3 WHERE rowid = 'AAAArBAACAAAABFAAK';
```

FULL-TABLE SCAN 으로 테이블 데이터 액세스

```
SQL> SELECT ename
  2 FROM s_emp;
```

CREATE INDEXS

UNIQUE 인덱스를 만들어 칼럼에 중복될 수 없는 유일 값 보장

Syntax

```
CREATE INDEX index_name
ON table_name (column_name [, column_name] ...)
```

where	*index_name*	인덱스 이름
	table_name	테이블 이름
	column_name	열 이름

Example

직원이름 열에 인덱스 생성

```
SQL> CREATE INDEX s_emp_ename_i
  2 ON s_emp(ename);

Index 'S_EMP_ENAME_I' created.
```

CONFIRM THE EXISTENCE OF INDEXES

USER_INDEX에서 인덱스 정보 확인

USER_INDEX 테이블

Column	Description
INDEX_NAME	인덱스 이름
TABLE_OWNER	인덱스 소유자
TABLE_NAME	인덱싱된 개체 이름
TABLE_TYPE	인덱싱된 개체 유형
UNIQUESNESS	인덱스의 고유성: UNIQUE or NONUNIQUE

CONFIRM THE EXISTENCE OF INDEXES

USER_INDEX에서 인덱스 정보 확인

```
SQL> DIESCRIBE user_indexes;

COLUMN_NAME                    TYPE           CONSTRAINT
------------------------------ -------------- ----------
INDEX_NAME                      VARCHAR(128)
INDEX_TYPE                     VARCHAR(26)
TABLE_OWNER                     VARCHAR(128)
TABLE_NAME                      VARCHAR(128)
TABLE_TYPE                     VARCHAR(9)
UNIQUENESS                      VARCHAR(9)
```

CONFIRM THE EXISTENCE OF INDEXES

Example

생성한 인덱스 표시

```
SQL> SELECT index_name, uniqueness
  2 FROM user_indexes
  3 WHERE table_name = 'S_EMP';

INDEX_NAME                                                    UNIQUENESS
------------------------------------------------------------------------------------
S_EMP_ID_PK                                                   UNIQUE
S_EMP_ENAME_I                                                 NONUNIQUE
```

	차이점	공통점
PK(Primary Key)	NOT NULL	중복될 수 없는 유일값
	하나의 PK	
	OBJECT – CONSTRAINT	
UNIQUE INDEX	NULL	
	여러 개 생성 가능	
	OBJECT – INDEX	

DROP A INDEX

Syntax

```
DROP INDEX index_name
```

Example

S_DEPT_ID 시퀀스 삭제

```
SQL> DROP INDEX s_emp_ename_i;

Index 'S_EMP_ENAME_I' dropped.
```

Control User Access: Overview

- 데이터베이스 관리자는 사용자에게 **SQL** 보안 명령을 사용해 테이블에 대한 액세스 권한을 제공
- Control User Access
  - 데이터베이스에 대한 권한 제공
  - 테이블 및 시퀀스와 같은 사용자 개체에 대한 액세스를 제공하고 제거
  - 데이터 사전에서 주어진 권한 및 받은 권한 확인
  - 데이터베이스 개체에 대한 동의어 또는 대체 이름 작성

SYSTEM PRIVILEGES: OVERVIEW

- 데이터베이스 관리자는 사용자에게 시스템 권한을 부여하여 사용자는 특정 작업을 수행할 수 있다.
- 시스템 권한은 명령을 실행할 수 있는 권한이다.

Type of System Privileges

System Privilege	Description
In One's Own Schema	자신의 스키마에 테이블 및 시퀀스를 생성할 수 있는 권한
On all Objects of a Specified Type	모든 스키마에서 테이블 생성 및 테이블 또는 뷰를 업데이트 할 수 있는 권한
On the System or a User	사용자를 생성할 수 있는 권한

SYSTEM PRIVILEGES: OVERVIEW

- 60개가 넘는 고유한 시스템 권한이 있다.
- 각 시스템 권한을 통해 사용자는 특정 작업을 수행할 수 있다.

Type of System Privileges

Class	System Privilege	Operations Permitted
SESSION	CREATE SESSION	데이터베이스 연결 허용
TABLE	CREATE TABLE	테이블 및 인덱스 생성
		CONNECT, DML, DROP, ALTER, TRUNCATE 가능
TABLE	SELECT ANY TABLE	모든 스키마에 모든 테이블, 뷰 쿼리 사용 가능

GRANT SYSTEM PRIVILEGES

- 데이터베이스 관리자는 **GRANT SQL** 명령을 사용하여 사용자 및 역할 권한을 부여하고 취소할 수 있다.

Syntax

GRANT *system_priv* TO [*user, role, PUBLIC*] [*WITH ADMIN OPTION*]

where	*system_priv*	부여되는 시스템 권한
	TO	권한을 부여 받는 대상
	user	일반 사용자
	role	권한 받을 역할
	PUBLIC	권한 받을 공유 사용자
	WITH ADMIN OPTION	사용할 수 있는 특권, 남에게 부여할 수 있는 관리 권한

GRANT SYSTEM PRIVILEGES

Example

실습을 위한 새로운 사용자 생성

```
SQL>  CREATE USER scott
   2 IDENTIFIED by tibero;
```

GRANT SQL 명령을 사용해 scott 사용자의 스키마에 테이블을 생성할 수 있는 권한 부여

```
SQL>  GRANT CREATE SESSION, CREATE TABLE TO scott;

Granted.
```

scott 사용자에게 스키마의 테이블을 변경할 수 있는 권한 부여

```
SQL>  GRANT ALTER ANY TABLE TO scott;

Granted.
```

CONFIRM SYSTEM PRIVILEGES GRANTED

DBA_SYS_PRIVS에서 시스템 권한 확인

```
SQL> DIESCRIBE dba_sys_privs;

COLUMN_NAME                    TYPE          CONSTRAINT
-------------------------------- ---------------- -------------------
GRANTEE                        VARCHAR(128)
PRIVILEGE                      VARCHAR(40)
ADMIN_OPTION                    VARCHAR(3)
```

scott 사용자에게 부여된 권한 조회

```
SQL>  SELECT grantee, privilege FROM dba_sys_privs
    2 WHERE grantee='SCOTT';
GRANTEE                                   PRIVILEGE
-----------------------------------------------------------------------------
SCOTT                                    CREATE SESSION
SCOTT                                    CREATE TABLE
SCOTT                                     ALTER ANY TABLE

3 rows selected.
```

OBJECT PRIVILEGES

데이터베이스 관리자가 사용자에게 부여할 수 있는 객체 권한으로는 테이블, 뷰, 시퀀스, 프로시저가 있다.

스키마 객체 특권	테이블	뷰	시퀀스	PSM 프로그램 (프러시저, 함수 등)	디렉터리
SELECT	O	O	O		
INSERT	O	O			
ALTER	O		O		
UPDATE	O	O			
DELETE	O	O			
TRUNCATE	O				
EXECUTE				O	
INDEX	O				
REFERENCES	O	O			
READ					O
WRITE					O

GRANT OBJECT PRIVILEGES

- 데이터베이스 관리자는 **GRANT** 명령을 사용하여 사용자 및 역할 권한을 부여하고 취소할 수 있다.

Syntax

GRANT *object_priv* ON [*column*] OBJECT TO [*user, role, PUBLIC*] [*WITH GRANT OPTION*]

where	*object_priv*	부여되는 객체 권한
	column	특정 객체 일부 칼럼
	(username.)OBJECT	스키마 객체 권한 대상이 되는 객체
	TO	객체 권한을 부여 받는 사용자
	WITH GRANT OPTION	부여 받은 권한을 다른 사용자에게 부여할 수 있는 권한

GRANT OBJECT PRIVILEGES

Example

scott에게 S_EMP 테이블을 조회 할 수 있는 권한 부여

```
SQL> GRANT SELECT ON s_emp TO scott;
Granted.
```

scott에게 S_EMP 테이블에 직원 번호, 직원 이름, 부서 번호를 삽입할 수 있는 권한과 월급을 수정할 수 있는 권한 부여

```
SQL> GRANT INSERT(empno, ename, deptno),
  2 UPDATE(sal)
  3 ON s_emp TO scott;

Granted.
```

GRANT OBJECT PRIVILEGES

Example

scott에게 S_DEPT 테이블을 조회 할 수 있는 권한과 다른 사람에게 동일한 권한을 부여할 수 있는 권한 부여

```
GRANT SELECT, INSERT
ON s_dept
TO scott
WITH GRANT OPTION;
```

scott이 시스템의 모든 사용자에게 sys의 S_DEPT 테이블을 조회할 수 있는 권한 부여

```
SQL> conn scott
Enter Password:

SQL> GRANT SELECT
  2 ON sys.s_dept
  3 TO PUBLIC;

Granted.
```

CONFIRM OBJECT PRIVILEGES GRANTED

USER_TAB_PRIVS_MADE Columns

Column	Description
GRANTEE	권한이 부여된 사용자 이름
TABLE_NAME	객체 이름
GRANTOR	권한을 부여한 사용자 이름
PRIVILEGE	부여된 권한
GRANTABLE	WITH GRANT OPTION 부여 유무

```
SQL> DESCRIBE user_tab_privs_made

COLUMN_NAME                    TYPE          CONSTRAINT
-------------------------------- ---------------- ----------------
GRANTEE                        VARCHAR(128)
TABLE_NAME                       VARCHAR(128)
GRANTOR                        VARCHAR(128)
PRIVILEGE                      VARCHAR(40)
GRANTABLE                        VARCHAR(3)
```

CONFIRM OBJECT PRIVILEGES GRANTED

Example

현재 사용자가 scott에게 부여된 객체 조회

```
SQL> SELECT *
  2 FROM user_tab_privs_made
  3 WHERE GRANTEE = 'SCOTT';

GRANTEE                        TABLE_NAME  GRANTOR  PRIVILEGE     GRANTABLE
------------------------------ ---------- -------- ---------- ------------
SCOTT                          S_DEPT      SYS      INSERT        YES
SCOTT                          S_DEPT      SYS      SELECT        YES
SCOTT                          S_EMP       SYS      SELECT        NO

3 rows selected.
```

CONFIRM OBJECT PRIVILEGES GRANTED

Example

사용자 scott이 자신에 부여된 객체 조회

```
SQL> SELECT *
  2 FROM user_tab_privs_recd;

OWNER                TABLE_NAME          GRANTOR             PRIVILEGE         GRANTABLE
-------------------------------------------------------------------------------------------------------------------
SYS                  S_EMP               SYS                 SELECT             NO
SYS                  S_DEPT              SYS                 SELECT            YES
SYS                  S_DEPT              SYS                 INSERT            YES

3 rows selected.
```

CREATE SYNONYMS FOR CONVENIENT ACCESS

- 사용자가 소유한 스키마나 다른 사용자가 소유한 스키마에 속하는 동의어를 생성
- **Public Synonyms**
  - 데이터베이스 관리자가 작성한 동의어로, 모든 사용자가 액세스 할 수 있다.
- **Private Synonyms**
  - 데이터베이스 사용자에 의한 동의어로, 동의어 작성자만 액세스 할 수 있다.

Syntax

CREATE [PUBLIC] SYNONYM *synonym_name*
FOR *object_name*

where	*PUBLIC*	부여되는 객체 권한
	object_name	동의어 개체
	Tpye	테이블
		뷰
		시퀀스
		PSM 함수, 프로시저
		동의어

CREATE SYNONYMS FOR CONVENIENT ACCESS

Example

사용자 scott이 sys의 S_DEPT 테이블에 대한 자신의 개인 동의어 S_DEPT 생성

```
SQL> GRANT CREATE SYNONYM TO scott;

Granted.

SQL> conn scott
Enter Password:
Connected to Tibero.

SQL> CREATE SYNONYM s_dept
  2 FOR sys.s_dept;

Synonym ‘S_DEPT' created.
```

CREATE SYNONYMS FOR CONVENIENT ACCESS

DROP SYNONYM 명령어를 이용해 동의어 제거하기

Syntax

```
CREATE SYNONYM synonym_name
```

Example

```
SQL> DROP SYNONYM S_DEPT;

Synonym ‘S_DEPT' dropped.
```

REMOVE OBJECT PRIVILEGES

SQL 명령어 REVOKE를 사용해 다른 사용자에게 부여된 권한을 제거

Syntax

```
REVOKE privilege, privilege…
ON object_name
FROM [user1_name, user2_name … | PUBLIC | role]
[CASCADE CONSTRAINTS]
```

where	CASCADE CONSTRAINTS	REFERENCE 스키마 객체 권한을 회수하는 경우
		참조 무결성 제약조건을 지운 뒤 권한 회수 (CASCADE CONSTRAINT 을 사용하지 않고, 회수하면 에러 발생)

Example

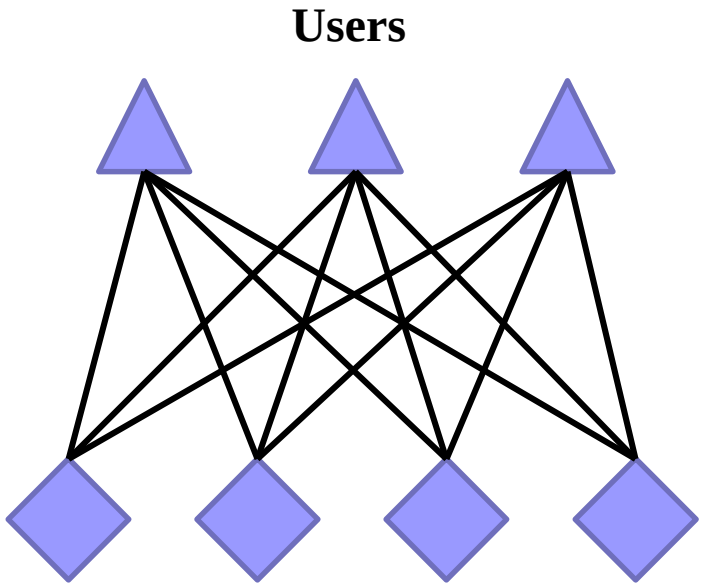
```
SQL> REVOKE SELECT, INSERT
  2 ON s_dept
  3 FROM scott;

Revoked.
```

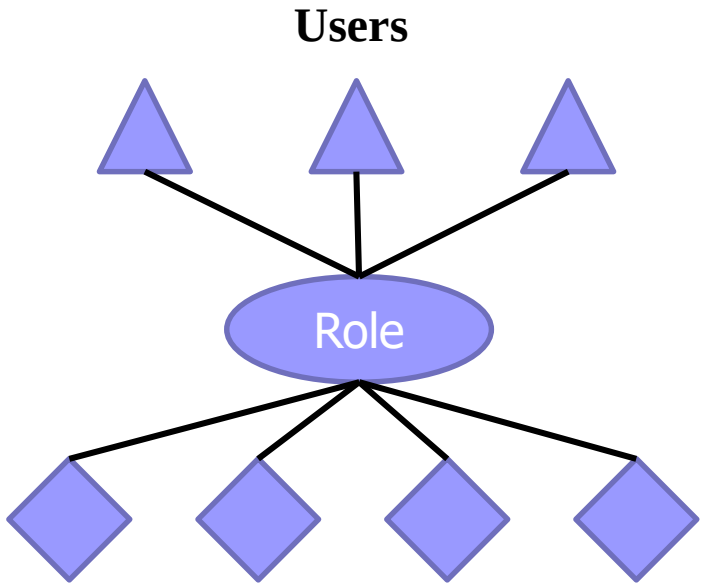
PRIVILEGES GROUPED BY ROLE

- 역할 사용을 통해 권한 관리를 단순화 시킴
- 시스템 권한과 객체 권한으로 구성 가능
- 역할은 사용자가 소유하지 않고, 스키마에도 존재하지 않음.

Grant Privileges Without Roles



Grant Privileges With Roles



CREATE A ROLE

Syntax

CREATE ROLE *role_name* [*NOT IDENTIFIED*] [*IDENTIFIED BY password*]

where	*role_name*	역할 이름
	NOT IDENTIFIED	패스워드 사용하지 않는다.(Default 값)
	IDENTIFIED BY	역할에 패스워드 설정
	password	패스워드

CREATE A ROLE

Example

역할 이름을 acct_rec으로 생성

```
SQL> CREATE ROLE acct_rec;
Role 'ACCT_REC' created.
```

패스워드가 bicentennial이고, 역할 이름을 acct_pay으로 생성

```
SQL> CREATE ROLE acct_pay
  2 IDENTIFIED BY bicentennial;
```

동의어와 테이블 생성 권한이 있는manager 역할 생성

```
SQL> GRANT CREATE TABLE, CREATE SYNONYM
  2 TO manager;
Granted.
```

신규 사용자 kevin에게 manager 권한 부여

```
SQL> GRANT manager TO kevin;

Granted.
```