SQL

✓ 원리를 알면 IT가 맛있다

SQL for Beginners

chapter 09.

DDL

❑ 학습목표

- DDL 개요
- 스키마(schema) 개요
- 테이블 생성 및 삭제
- 제약조건
- 컬럼 수정 (추가,수정,삭제)
- flashback 기술

❑ 1) DDL (Data Definition Language)

- DDL 용도 및 종류

문장	설명
SELECT	데이터베이스로부터 데이터를 검색
INSERT UPDATE DELETE MERGE	데이터베이스 내의 테이블에 새로운 행을 입력하거나, 기존의 행을 수정 또는 삭제하는 명령어로 데이터 조작어(DML : Data Manipulation Language)라고 함
CREATE ALTER DROP RENAME TRUNCATE	테이블을 생성, 변경, 삭제하는 명령어로 데이터 정의어(DDL : Data Definition Language)라고 함
COMMIT ROLLBACK SAVEPOINT	DML 문장에 의한 변경 사항을 관리하거나, 변경사항을 하나의 논리적 트랜잭션으로 포함시키는 명령어
GRANT REVOKE	데이터베이스와 데이터베이스를 구성하는 구조(테이블, 뷰 등)에 접근 권한을 부여하거나 회수하는 명령어로 데이터 제어어(DCL : Data Control Language)라고 함

- DDL 용도

– 데이터베이스 구조를 생성, 수정, 삭제하는데 사용되는 SQL문장이다. 이러한 문장은 데이터베이스에 즉각 영향을 미치며 데이터베이스 사전(DATA DICTIONARY)에 정보를 기록 한다.

- 오라클에서 사용하는 객체

– 오라클 데이터베이스는 여러 개의 데이터 구조를 가지고 있다.

객체명	설명
테이블(Table)	기본적인 저장 단위로 행과 컬럼으로 구성
뷰(View)	한개 이상의 테이블의 논리적인 부분 집합을 표시
시퀀스(Sequence)	숫자 값 생성기
인덱스(Index)	데이터 검색 성능 향상
동의어(Synonym)	객체에 대한 별칭

- 데이터베이스 객체 이름 지정방법

  - 테이블 및 컬럼명은 문자로 시작하며 1 ~ 30 문자 이내로 작성한다.(30btye)
  - 테이블 및 컬럼명은 A~Z , a~z , 0~9 , _ , $, # 로 작성한다.
    한글 작성도 가능하지만 권장하지 않는다.
  - 동일한 사용자의 다른 객체와 이름이 중복되지 않도록 한다.
  - Oracle 의 예약어는 사용 불가.
  - 대소문자 구별 안함.

- 테이블 생성

```
CREATE TABLE [schema.]table
             (column datatype [DEFAULT expr][, ...]);
```

```
SQL> CREATE TABLE DEPT_2
  2  ( DEPTNO NUMBER(2),
  3    DNAME VARCHAR2(10),
  4    LOC VARCHAR2(10) );

테이블이 생성되었습니다.
```

```
SQL> CREATE TABLE SCOTT.DEPT_3
  2  ( DEPTNO NUMBER(2),
  3    DNAME VARCHAR2(10),
  4    LOC  VARCHAR2(10) );

테이블이 생성되었습니다.
```

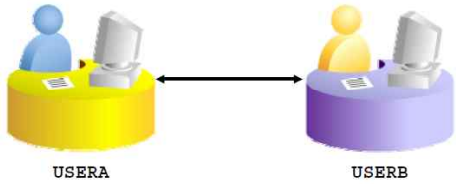
- 스키마 (schema)

– 특정 사용자가 데이터베이스에 접근하여 생성한 객체들의 대표이름을 의미한다. 일반적으로 **사용자의 계정명과 동일**하다.

– 생성한 객체들의 소유는 생성한 계정이 갖는다. 따라서 다른 스키마는 기본적으로 접근이 불가능하다. (권한 할당 필요)

만약 권한을 가지고 다른 스키마에 접근하기 위해서는 '스키마.객체명' 형식으로 접근해야 된다.

예> scott 계정의 스키마 명은 scott이다.



- 다른 사용자의 테이블 검색

```
SQL> show user;
USER은 "SYS"입니다
SQL>
SQL> CONN / as sysdba
연결되었습니다.
SQL>
SQL> SELECT * FROM DEPT;
SELECT * FROM DEPT
              *
1행에 오류:
ORA-00942: 테이블 또는 뷰가 존재하지 않습니다
```

```
SQL> SELECT * FROM SCOTT.DEPT;

    DEPTNO DNAME      LOC
---------- ---------- ----------
        90 경리과     부산
        10 ACCOUNTING NEW YORK
        20 RESEARCH   DALLAS
        30 SALES      CHICAGO
        40 OPERATIONS BOSTON
```

❑ 3) 오라클 데이터 타입

- 데이터 타입 종류

  : 테이블 생성시 컬럼에 지정할 수 있는 데이터 타입은 다음과 같다.

데이터 타입	설명
VARCHAR2(*size*)	가변 길이 문자열 데이터. (최소 길이 1, 최대 길이 4000 바이트)
CHAR[(*size*)]	고정 길이 문자열 데이터. (디폴트 및 최소 길이 1, 최대 길이 2000 바이트)
NUMBER(*p*, *s*)	가변 길이 숫자 데이터. (전체 자리수 *p*, 소수점 자리수 *s*. 전체 자리수는 1~38, 소수점 자리수는 -84~127)
DATE	날짜 및 시간 데이터. (B.C 4712년 1월 1일~ A.D 9999년 12월 31일)
LONG	가변 길이 문자열 데이터(2GB)
CLOB	문자 데이터(4GB)
RAW(*size*)	이진 데이터. (최대 2000 바이트)
LONG RAW	가변 길이 이진 데이터(2GB)
BLOB	이진 데이터(4GB)
BFILE	외부 파일에 저장된 이진 데이터(4GB)
ROWID	시스템에서 테이블내의 행들을 유일하게 식별할 수 있는 64 비트 숫자

❑ 3) 오라클 데이터 타입

- LONG 타입 특징

  : 2기가 바이트의 가변 길이 문자 저장.
  : VARCHAR2와 유사한 특징을 가지나 아래와 같은 특징이 있다.
  - 하나의 테이블에 하나의 LONG 타입만 사용 가능
  - NOT NULL 제약 조건 이외의 다른 제약조건 사용 불가
  - 인덱스 사용 불가
  - SELECT문에서 WHERE,GROUP BY, ORDER BY, CONNECT BY, DISTINCT 사용 불가
  - CTAS 사용 불가

❑ 2) DDL – DEFAULT 옵션

- DEFAULT 옵션

– 해당 테이블에 행을 입력할 때, 해당 컬럼에 값을 지정하지 않은 경우 자동으로 디폴트 값이 입력되어 NULL 값이 저장되는 것을 방지할 수 있다.

– 고정된 값만을 가지는 컬럼에 대해서도 사용 가능하다. (예>날짜, 성별)

.

```
SQL> CREATE TABLE DEF_TABLE
  2  ( NUM NUMBER(2),
  3    WRITEDAY DATE
  4  );

테이블이 생성되었습니다.

SQL> CREATE TABLE DEF_TABLE2
  2  ( NUM NUMBER(2),
  3    WRITEDAY DATE DEFAULT SYSDATE
  4  );

테이블이 생성되었습니다.
```

```
SQL> INSERT INTO DEF_TABLE( NUM )
  2  VALUES ( 1 );

1 개의 행이 만들어졌습니다.

SQL> INSERT INTO DEF_TABLE2 ( NUM )
  2  VALUES ( 1 );

1 개의 행이 만들어졌습니다.


SQL> INSERT INTO DEF_TABLE2( NUM , WRITEDAY )
  2  VALUES ( 2, DEFAULT );

1 개의 행이 만들어졌습니다.
```

❑ 2) DDL- 서브쿼리 이용한 테이블 생성

- 서브쿼리를 이용한 테이블 생성

: CTAS (씨탁스)라고 한다.
: 지정된 컬럼의 개수와 서브쿼리에서 리턴된 컬럼의 개수가 일치해야 한다.
: 컬럼을 지정할 때 컬럼명과 디폴트 값만 지정 가능하다.
: 지정된 컬럼을 연산 했을때에는 반드시 alias 를 사용한다.
: NOT NULL를 제외한 제약 조건은 생성된 테이블에 만들어지지 않으며 오직 컬럼의 데이터 타입만 동일하게 생성된다.
: 원본 테이블의 인덱스 및 LONG 타입 복사 불가.

```
CREATE TABLE table
        [(column, column ...)]
AS subquery
```

```
SQL> CREATE TABLE deptA
  2  AS
  3  SELECT * FROM DEPT;

테이블이 생성되었습니다.

SQL> SELECT COUNT(*) FROM deptA;

  COUNT(*)
----------
         4
```

```
SQL> CREATE TABLE deptB( no , name )
  2  AS
  3  SELECT deptno , dname
  4  FROM DEPT;

테이블이 생성되었습니다.

SQL> SELECT count(*) FROM deptB;

  COUNT(*)
----------
         4
```

```
SQL> CREATE TABLE deptC
  2  AS
  3  SELECT * FROM DEPT
  4  WHERE 1= 2;

테이블이 생성되었습니다.

SQL> SELECT COUNT(*) FROM deptC;

  COUNT(*)
----------
         0
```

❑ 3) 제약조건 (constraint)

- 제약조건 정의 (Constraints)

  -부적절한 자료가 입력되는 것을 방지하기 위하여 constraint을 사용한다.
  -제약 조건은 **테이블 LEVEL** 및 **컬럼 LEVEL**에서 규칙을 적용한다.
  -제약 조건은 종속성이 존재할 경우 테이블 삭제를 방지 한다.
  -테이블에서 행이 삽입, 갱신, 삭제될 때마다 테이블에서 규칙을 적용한다.
  -일시적으로 DISABLE할 수 있고 ENABLE할 수도 있다.
  -user_constraints의 DATA DICTIONARY VIEW을 조회하면 지정 테이블에 대해 정의된 제약 조건을 볼 수 있다.

- 제약조건 종류

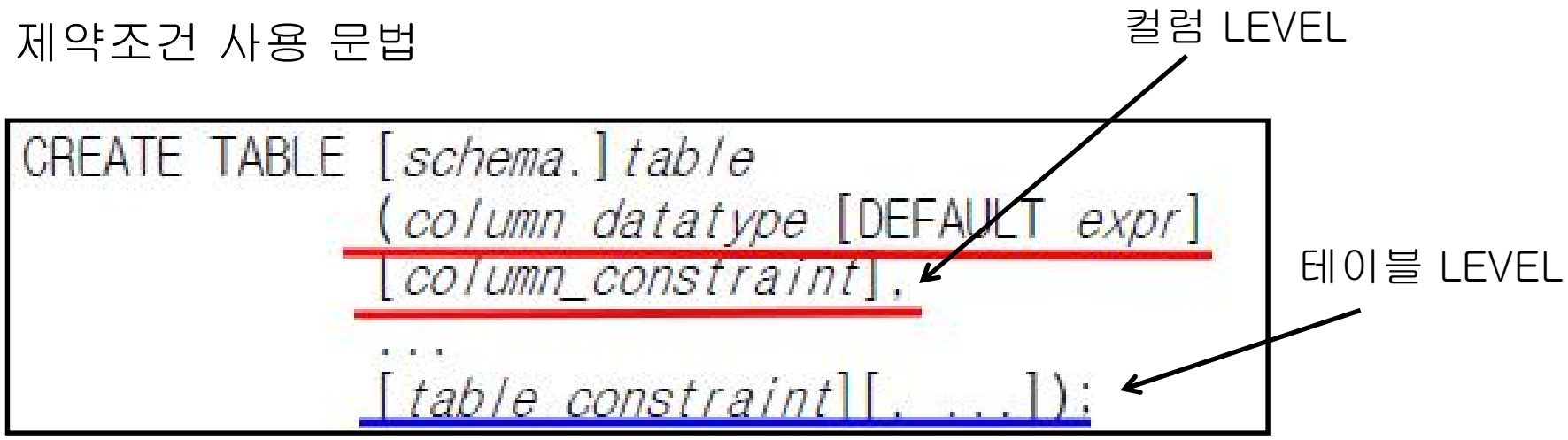
제약조건	기 술
NOT NULL	이 열은 null 값을 포함하지 않음을 지정
UNIQUE	테이블의 모든 행에 대해 유일해야 하는 값을 가진 열 또는 열의 조합을 지정
PRIMARY KEY	유일하게 테이블의 각 행을 식별
FOREIGN KEY	열과 참조된 테이블의 열 사이의 외래키 관계를 적용하고 설정
CHECK	참이어야 하는 조건을 지정

❑ 3) 제약조건 (constraint)

- 제약조건 사용 지침

  -적절한 이름을 지정하여 제약조건을 사용한다. 지정하지 않으면 자동으로 SYS_Cn 형식 으로 임의 저장되기 때문에 필요할 때 관리(검색)이 어려워진다.
  -테이블 생성과 동시에 지정할 수도 있고, 테이블 생성 후에 추가할 수도 있다.
  -제약 조건은 테이블 수준 및 컬럼 수준, 2 가지 방법으로 지정한다.
  -제약 조건은 USER_CONSTRAINTS 데이터 딕셔너리에서 검색할 수 있다.

- 제약조건 사용 문법



❑ 3) 제약조건 (constraint)

- 제약조건 정의 방법

1. 테이블 LEVEL 지정 방법
   - 테이블의 컬럼 정의와는 별개로 정의한다.
   - 한 개 이상의 컬럼에 한 개의 제약 조건을 정의할 수 있다.
   - NOT NULL 제약조건을 제외한 모든 제약조건 정의 가능하다.

```
column, ...
  [CONSTRAINT constraint_name] constraint_type
  (column, ...),
```

2. 컬럼 LEVEL 지정 방법
   - 한 개의 컬럼에 한 개의 제약조건만 정의 가능하다.
   - 모든 제약조건에 대해서 정의 가능하다. (NOT NULL 제약조건 필수)

```
column [CONSTRAINT constraint_name] constraint_type,
```

❑ 3) 제약조건 (constraint)

- PRIMARY KEY (PK)

  - 테이블에 대한 기본 키를 생성합니다.
  - 하나의 기본 키만이 각 테이블에 대해 존재할 수 있다.
  - PRIMARY KEY 제약 조건은 테이블에서 각행을 유일하게 식별하는 열 또는 열의 집합(복합컬럼)이다.(UNIQUE와 NOT NULL조건을 만족)
  - 이 제약 조건은 열 또는 열의 집합의 유일성을 요구하고 NULL값을 포함할 수 없음을 보증 한다.
  - UNIQUE INDEX가 자동 생성된다.

```
column datatype [CONSTRAINT constraint_name] PRIMARY KEY
```

```
column datatype,
. . . . . ,
[CONSTRAINT constraint_name] PRIMARY KEY (column1[, column2,..])
```

```
SQL> CREATE TABLE PK_TAB1(
  2  id          NUMBER(2) CONSTRAINT PK_TAB1_ID_PK PRIMARY KEY,
  3  name        VARCHAR2(10));
```

```
SQL> CREATE TABLE PK_TAB2(
  2  id          NUMBER(2),
  3  name        VARCHAR2(10),
  4  CONSTRAINT PK_TAB2_ID_PK PRIMARY KEY (id));
```

❑ 3) 제약조건 (constraint)

- FOREIGN KEY(FK)

  - FOREIGN KEY는 DETAIL쪽에서 정의한다.
  - MASTER TABLE의 PRIMARY KEY, UNIQUE KEY로 정의된 열을 지정할 수 있으며 열의 값과 일치하거나 NULL값이어야 한다.
  - FOREIGN KEY는 열 또는 열의 집합을 지정할 수 있으며 동일 테이블 또는 다른 테이블간의 관계를 지정할 수 있다.
  - ON DELETE CASCADE을 사용하여 DETAIL TABLE에서 관련된 행을 삭제하고 MASTER TABLE에서 삭제를 허용할 수 있다.

```
column datatype [CONSTRAINT constraint_name]
       REFERENCES table_ name (column1[,column2,..] [ON DELETE CASCADE])
```

```
column datatype,
. . . . . .
[CONSTRAINT constraint_name] FOREIGN KEY (column1[,column2,..])
       REFERENCES table_name  (column1[,column2,..] [ON DELETE CASCADE])
```

❑ 3) 제약조건 (constraint)


```
SQL> CREATE TABLE DEPT_2
  2  ( DEPTNO NUMBER(2) CONSTRAINT DEPT_2_DEPTNO_PK PRIMARY KEY,
  3    DNAME VARCHAR2(10),
  4    LOC VARCHAR2(10) );

테이블이 생성되었습니다.

SQL> INSERT INTO DEPT_2 VALUES ( 10, '인사', '서울' );

SQL> INSERT INTO DEPT_2 VALUES ( 20, '경리', '부산' );

SQL> INSERT INTO DEPT_2 VALUES ( 30, '관리', '대구' );
```

```
SQL> CREATE TABLE EMP_2
  2  ( EMPNO NUMBER(4) CONSTRAINT EMP_2_EMPNO_PK PRIMARY KEY,
  3    ENAME VARCHAR2(10),
  4    SAL  NUMBER(10),
  5    DEPTNO NUMBER(2) CONSTRAINT EMP_2_DEPTNO_FK REFERENCES DEPT_2(DEPTNO));

SQL> INSERT INTO EMP_2 VALUES ( 1111, '홍길동', 2000 , 40 );
INSERT INTO EMP_2 VALUES ( 1111, '홍길동', 2000 , 40 )
*
1행에 오류:
ORA-02291: 무결성 제약조건(SCOTT.EMP_2_DEPTNO_FK)이 위배되었습니다- 부모 키가
없습니다
SQL> CREATE TABLE EMP_3
  2  ( EMPNO NUMBER(4) CONSTRAINT EMP_3_EMPNO_PK PRIMARY KEY,
  3    ENAME VARCHAR2(10),
  4    SAL  NUMBER(10),
  5    DEPTNO NUMBER(2),
  6  CONSTRAINT EMP_3_DEPTNO_FK FOREIGN KEY ( DEPTNO ) REFERENCES DEPT_2(DEPTNO
);
```

❑ 3) 제약조건 (constraint)

- FOREIGN KEY 추가 옵션
  - 부모 테이블의 행 삭제 시 문제될 수 있는 자식 테이블 행 설정법.

  : ON DELETE CASCADE
  - FK 제약조건에 의해 참조되는 테이블(부모 테이블)의 행이 삭제되면, 해당 행을 참조하는 테이블(자식 테이블)의 행도 같이 삭제되도록 한다.

  : ON DELETE SET NULL
  - FK 제약조건에 의해 참조되는 테이블(부모 테이블)의 행이 삭제되면, 해당 행을 참조하는 테이블(자식 테이블)의 컬럼을 NULL로 설정한다.

❑ 3) 제약조건 (constraint)

- UNIQUE (UK)

  - 해당 컬럼에 중복된 값이 저장되지 않도록 제한한다.
  - 한 개 이상의 컬럼(복합컬럼)으로 구성 할 수 있다.
  - NULL 값 저장 가능하다.
  - 테이블 LEVEL 및 컬럼 LEVEL 모두 지정 가능하다. 하지만 복합컬럼 지정시에는 테이블 LEVEL만 가능하다.
  - 해당 컬럼에 UNIQUE INDEX 가 자동 생성된다.

```
column datatype [CONSTRAINT constraint_name] UNIQUE
```

```
column datatype,
. . . . . ,
[CONSTRAINT constraint_name] UNIQUE (column1[,column2,..])
```

```
SQL> CREATE TABLE UNI_TAB1 (
  2  DEPTNO      NUMBER(2) CONSTRAINT UNI_TAB1_DEPTNO_UK UNIQUE,
  3  DNAME       CHAR(14),
  4  LOC         CHAR(13));
```

```
SQL> CREATE TABLE UNI_TAB2 (
  2  DEPTNO      NUMBER(2),
  3  DNAME       CHAR(14),
  4  LOC         CHAR(13),
  5  CONSTRAINT UNI_TAB2_DEPTNO_UK UNIQUE (DEPTNO));
```

❑ 3) 제약조건 (constraint)

```
SQL> CREATE TABLE SAWON_2 (
  2  S_NO NUMBER(2),
  3  S_NAME VARCHAR2(10) NOT NULL,
  4  S_EMAIL VARCHAR2(20) CONSTRAINT SAWON_S_EMAIL_UK UNIQUE );

SQL> INSERT INTO SAWON_2 VALUES( 2, '유관순', 'hong@abc.com' );
INSERT INTO SAWON_2 VALUES( 2, '유관순', 'hong@abc.com' )
*
1행에 오류:
ORA-00001: 무결성 제약 조건(SCOTT.SAWON_S_EMAIL_UK)에 위배됩니다

SQL> CREATE TABLE SAWON_3
  2  ( S_NO NUMBER(2),
  3    S_NAME VARCHAR2(10) NOT NULL,
  4    S_EMAIL VARCHAR2(20),
  5    CONSTRAINT SAWON_3_S_EMAIL_UK UNIQUE (S_EMAIL ));

테이블이 생성되었습니다.
```

❑ 3) 제약조건 (constraint)

- 복합 컬럼 UNIQUE 제약 조건

```
SQL> CREATE TABLE SAWON_4
  2  ( S_NO NUMBER(2),
  3    S_NAME VARCHAR2(10),
  4    S_EMAIL VARCHAR2(20),
  5   CONSTRAINT SAWON_4_UK UNIQUE( S_NAME , S_EMAIL ) );

테이블이 생성되었습니다.
```

```
INSERT INTO SAWON_4 VALUES ( 1, '홍길동','abc@abc.com');

INSERT INTO SAWON_4 VALUES ( 2, '홍길동','ddd@abc.com');


INSERT INTO SAWON_4 VALUES ( 3, '이순신', NULL );

INSERT INTO SAWON_4 VALUES ( 4, '유관순', NULL);

INSERT INTO SAWON_4 VALUES ( 5, '유관순', NULL);

INSERT INTO SAWON_4 VALUES ( 6, NULL , NULL );

INSERT INTO SAWON_4 VALUES ( 7, NULL , NULL );
```

❑ 3) 제약조건 (constraint)

- CHECK 제약 조건

: 해당 컬럼에 반드시 만족해야 될 조건을 지정하는 제약 조건이다. (회사의 업무 규칙등)

```
SQL> CREATE TABLE SAWON_7
  2  ( S_NO NUMBER(2),
  3    S_NAME VARCHAR2(10),
  4    S_SAL NUMBER(10) CONSTRAINT SAWON_7_S_SAL_CK CHECK( S_SAL < 500 ));

테이블이 생성되었습니다.

SQL> INSERT INTO SAWON_7 VALUES ( 1, '홍길동' , 600 );
INSERT INTO SAWON_7 VALUES ( 1, '홍길동' , 600 )
*
1행에 오류:
ORA-02290: 체크 제약조건(SCOTT.SAWON_7_S_SAL_CK)이 위배되었습니다


SQL> INSERT INTO SAWON_7 VALUES( 1 , '홍길도', 400 );

1 개의 행이 만들어졌습니다.
```

```
SQL> CREATE TABLE SAWON_8
  2  ( S_NO NUMBER(2),
  3    S_NAME VARCHAR2(10),
  4    S_SAL NUMBER(10),
  5  CONSTRAINT SAWON_8_S_SAL_CK CHECK ( S_SAL < 500 ) );

테이블이 생성되었습니다.
```

❑ 3) 제약조건 (constraint)

- NOT NULL 제약 조건

  -해당 컬럼에 NULL 값이 입력되는 것을 방지하는 제약조건이다.
  -NOT NULL 제약조건은 컬럼 LEVEL에서만 지정 가능하다.

```
SQL> CREATE TABLE SAWON (
  2  S_NO NUMBER(4),
  3  S_NAME VARCHAR2(10) NOT NULL,
  4  S_HIREDATE DATE CONSTRAINT SAWON_S_HIREDATE_NN NOT NULL);

테이블이 생성되었습니다.

SQL> INSERT INTO SAWON
  2  VALUES(1, '길동', NULL);
INSERT INTO SAWON
*
1행에 오류:
ORA-01400: NULL을 ("SCOTT"."SAWON"."S_HIREDATE") 안에 삽입할 수 없습니다

SQL> SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE
  2  FROM USER_CONSTRAINTS
  3  WHERE TABLE_NAME = 'SAWON';

CONSTRAINT_NAME                C
------------------------------ -
SAWON_S_HIREDATE_NN            C
SYS_C003005                    C
```

❑ 4) 테이블 삭제

- 테이블 삭제

: 데이터베이스에서 해당 테이블을 제거하는 것이다.
: 테이블에 저장된 모든 데이터와 관련 INDEX및 제약조건이 삭제된다. (FK 제외)

예 > DROP TABLE dept CASCADE CONSTRAINTS;

```
DROP TABLE table
```

```
SQL> drop table sawon_8;

테이블이 삭제되었습니다.
```

CASCADE CONSTRAINT; 도 가능

- 테이블 이름 변경

```
RENAME old_name TO new_name;
```

```
SQL> rename sawon_7 to sawon_77;

테이블 이름이 변경되었습니다.
```

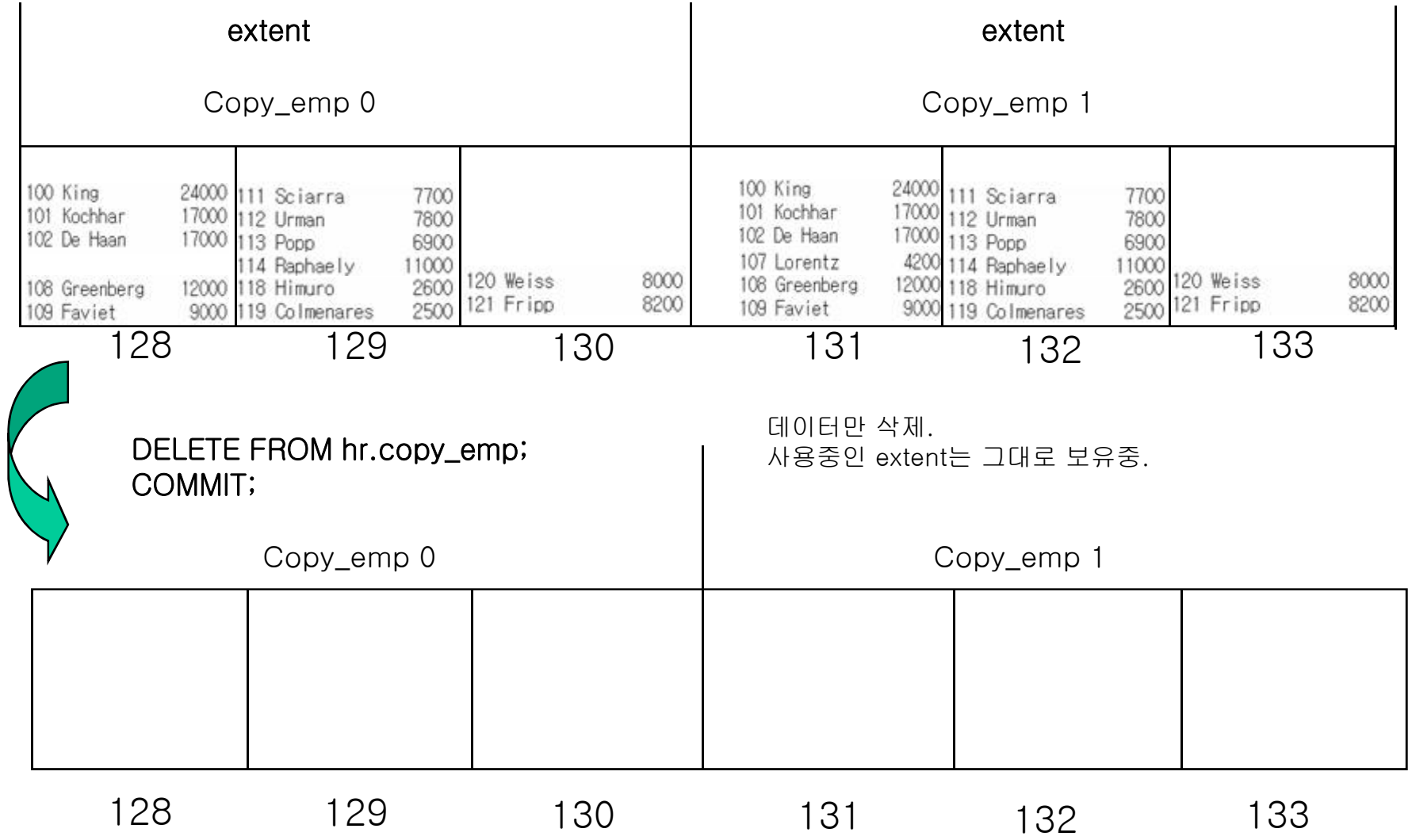
- 테이블 잘라내기

: 테이블의 모든 행들을 삭제 할 수 있다.
: 테이블이 사용하고 있던 저장 공간을 해제하여 다른 테이블들이 사용 할 수 있도록 한다.
: DELETE 명령은 저장공간을 해제하지 않는다.
: ROLLBACK 정보를 발생시키지 않아서 DELETE 보다 수행속도가 빠르다.
: 단, DELETE와 달리 ROLLBACk 은 불가능하다.

```
TRUNCATE TABLE table;
```

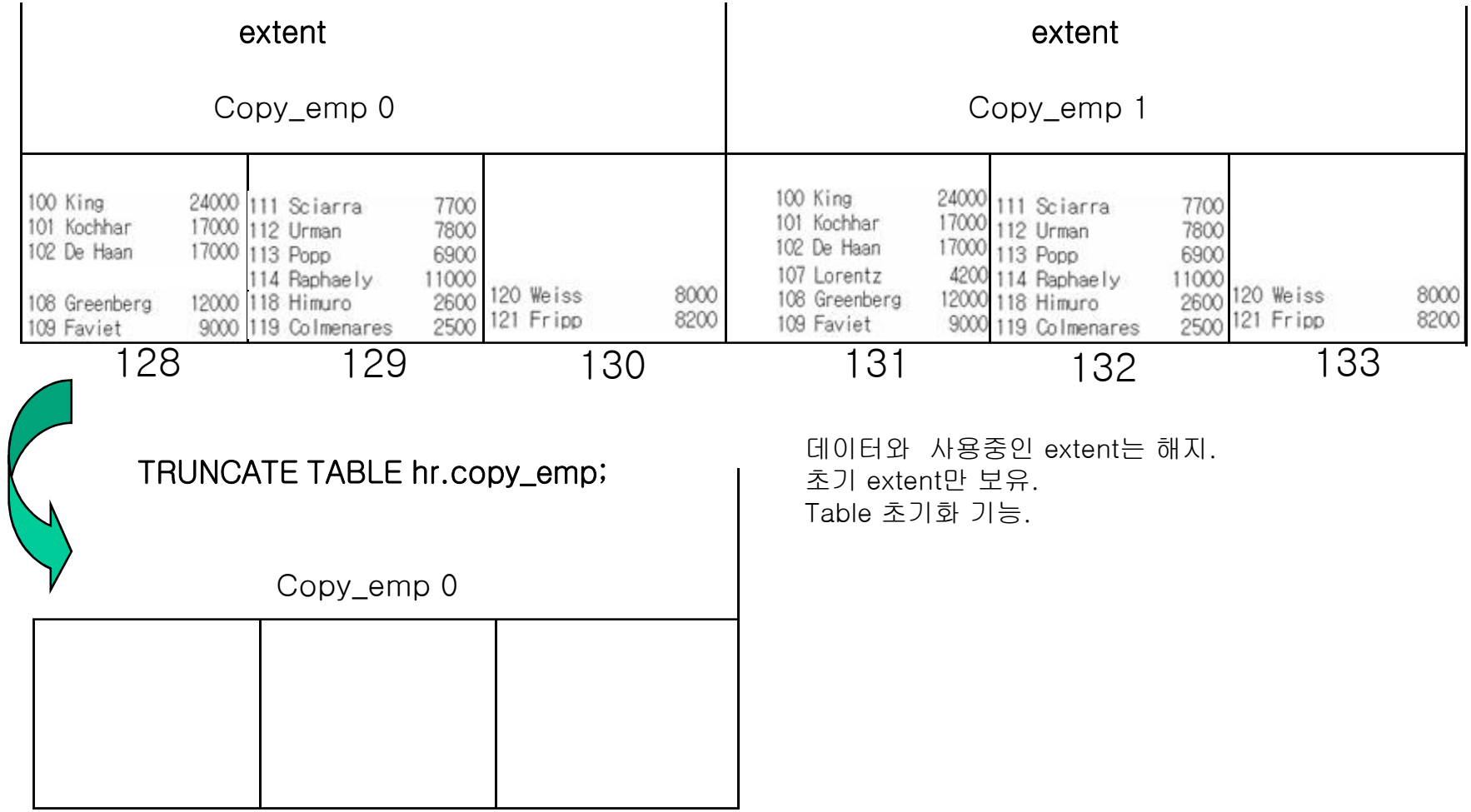
❑ 4) truncate VS delete

- delete 사용



❑ 4) 테이블 삭제

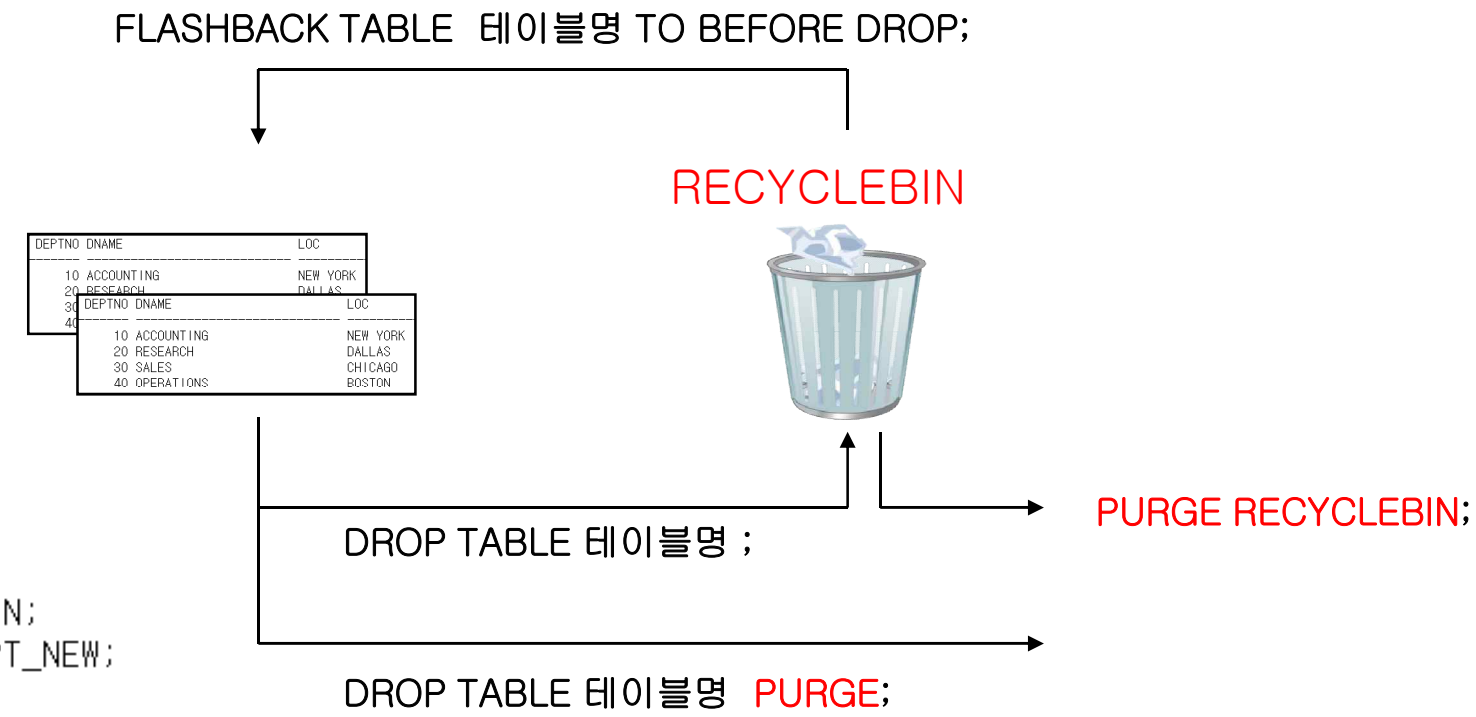
- truncate 사용



데이터와 사용중인 extent는 해지.
초기 extent만 보유.
Table 초기화 기능.

❑ 5) 삭제된 테이블 복구

- Flashback Statement (Flashback Drop)




```
SQL> SHOW RECYCLEBIN;
SQL> DROP TABLE DEPT_NEW;

테이블이 삭제되었습니다.

SQL> SHOW RECYCLEBIN;
ORIGINAL NAME    RECYCLEBIN NAME                OBJECT TYPE  DROP TIME
---------------- ------------------------------ ------------ -------------------

DEPT_NEW         BIN$kJItIBdmTKKHRfZ2gOvNow==$0 TABLE        2009-09-09:15:47:33
```

❑6) 테이블 변경 – 컬럼 추가

- 테이블 변경 (ALTER TABLE 문)
  - 새로운 컬럼 추가
  - 기존 컬럼 수정
  - 컬럼 삭제
  - 컬럼 이름 변경
  - 제약조건 추가 , 삭제
  - 제약조건 활성화 , 비활성화
  - 테이블 읽기 모드(read only)

- 컬럼 추가

```
ALTER TABLE table
ADD (column datatype [DEFAULT expr]
    [, column datatype] ...);
```

```
SQL> CONN SCOTT/TIGER
연결되었습니다.
SQL> CREATE TABLE SCOTT_T
  2  ( NUM NUMBER(4),
  3    NAME VARCHAR(10) );

테이블이 생성되었습니다.

SQL> ALTER TABLE SCOTT_T
  2  ADD ( ADDRESS VARCHAR2(30));

테이블이 변경되었습니다.
```

```
SQL> DESC SCOTT_T;
 이름                                      널?      유형
 ----------------------------------------- -------- ----------------

 NUM                                                NUMBER(4)
 NAME                                               VARCHAR2(10)

SQL> DESC SCOTT_T;
 이름                                      널?      유형
 ----------------------------------------- -------- ----------------

 NUM                                                NUMBER(4)
 NAME                                               VARCHAR2(10)
 ADDRESS                                            VARCHAR2(30)
```

❑ 6) 테이블 변경 – 읽기 전용 (11g, 12C 가능)

- 읽기 전용 테이블

```
SQL> CREATE TABLE emp_t
  2  AS
  3  SELECT empno, ename, sal
  4  FROM emp
  5  WHERE 1=2;

테이블이 생성되었습니다.

SQL> ALTER TABLE emp_t
  2  ADD ( address VARCHAR2(10) );

테이블이 변경되었습니다.

SQL> ALTER TABLE emp_t READ ONLY;

테이블이 변경되었습니다.

SQL> ALTER TABLE emp_t
  2  ADD ( email VARCHAR2(10) );
ALTER TABLE emp_t
*
1행에 오류:
ORA-12081: "SCOTT"."EMP_T" 테이블에 작업을 갱신하는 것이 허용되지 않습니다
```

```
SQL> ALTER TABLE emp_t READ WRITE;

테이블이 변경되었습니다.

SQL> ALTER TABLE emp_t
  2  ADD ( email VARCHAR2(10) );

테이블이 변경되었습니다.
```

❑6] 테이블 변경 – 컬럼 변경

- 컬럼 변경

  - 숫자 및 문자 컬럼의 전체 길이를 증가 시킬수 있다.
  - 컬럼 길이 축소도 가능하다. (단, 모든 행의 컬럼이 NULL 또는 행이 없는 경우)
  - 모든 행의 해당 컬럼 값이 NULL인 경우에만 데이터 타입을 변경 할 수 있다.
  - 디폴트 값을 변경하면 변경 이후부터 입력되는 행에 대해서만 적용된다.

```
ALTER TABLE table
MODIFY (column datatype [DEFAULT expr]
        [, column datatype] ...);
```

```
SQL> DESC SCOTT_T;
 이름                                      널?      유형
 ----------------------------------------- -------- ----------------

 NUM                                                NUMBER(4)
 NAME                                               VARCHAR2(10)
 ADDRESS                                            VARCHAR2(30)
```

```
SQL> ALTER TABLE SCOTT_T
  2  MODIFY ( ADDRESS NUMBER(4) );

테이블이 변경되었습니다.
```

```
SQL> DESC SCOTT_T
 이름                                      널?      유형
 ----------------------------------------- -------- ----------------

 NUM                                                NUMBER(4)
 NAME                                               VARCHAR2(10)
 ADDRESS                                            NUMBER(4)
```

❑ 6] 테이블 변경 – 컬럼 이름 변경

- 컬럼 이름 변경

```
ALTER TABLE table
RENAME COLUMN old_column TO new_column;
```

```
SQL> DESC SCOTT_T
 이름                                      널?      유형
 ----------------------------------------- -------- ----------------

 NUM                                                NUMBER(4)
 NAME                                               VARCHAR2(10)
 ADDRESS                                            NUMBER(4)

SQL> ALTER TABLE SCOTT_T
  2  RENAME COLUMN ADDRESS TO ADDR;

테이블이 변경되었습니다.

SQL> DESC SCOTT_T
 이름                                      널?      유형
 ----------------------------------------- -------- ----------------

 NUM                                                NUMBER(4)
 NAME                                               VARCHAR2(10)
 ADDR                                               NUMBER(4)
```

❑6] 테이블 변경 – 컬럼 삭제

- 컬럼 삭제
  : 컬럼은 값의 존재 유무에 상관없이 삭제된다.
  : ALTER TABLE 명령을 사용하여 여러 개의 컬럼을 동시 삭제 가능하다.
  : 테이블에는 최소한 하나의 컬럼은 존재해야 한다.

```
ALTER TABLE table
DROP (column );
```

```
SQL> DESC SCOTT_T
 이름
 ---------------------------

 NUM
 NAME
 ADDR

SQL> ALTER TABLE SCOTT_T
  2  DROP ( ADDR );

테이블이 변경되었습니다.

SQL> DESC SCOTT_T
 이름
 ---------------------------

 NUM
 NAME
```

```
SQL> CREATE TABLE SCOTT_T2
  2  ( NUM NUMBER(4),
  3    NAME VARCHAR2(10),
  4    ADDRESS VARCHAR2(30));

테이블이 생성되었습니다.

SQL> INSERT INTO SCOTT_T2 VALUES ( 1, 'AAA', 'SEOUL' );


SQL> ALTER TABLE SCOTT_T2
  2  DROP ( NAME , ADDRESS );

테이블이 변경되었습니다.

SQL> DESC SCOTT_T2
 이름                                      널?      유형
 ----------------------------------------- -------- ----------

 NUM                                                NUMBER(4)
```

❑6] 테이블 변경 – 제약조건 추가

- 제약조건 관리

: 기존 테이블의 제약조건을 추가하거나 삭제, 활성화, 비활성화 할 수 있다.

- 제약조건 추가

```
ALTER TABLE table
ADD [CONSTRAINT constraint] type (column);
```

: NOT NULL 제약조건은 ALTER TABLE ~ MODIFY 명령을 사용한다.

```
SQL> CREATE TABLE scott_t3 (
  2     num NUMBER(4),
  3     name VARCHAR2(10) );

SQL> ALTER TABLE SCOTT_T3
  2  ADD CONSTRAINT SCOTT_T3_NUM_PK PRIMARY KEY(NUM );

테이블이 변경되었습니다.

SQL> ALTER TABLE SCOTT_T3
  2  MODIFY ( NAME VARCHAR2(10) NOT NULL );

테이블이 변경되었습니다.

SQL> DESC SCOTT_T3
 이름                                      널?      유형
 ----------------------------------------- -------- ----------------

 NUM                                       NOT NULL NUMBER(4)
 NAME                                      NOT NULL VARCHAR2(10)
```

❑6) 테이블 변경 – 제약조건 추가

PK FK

EMPNO	ENAME	MGR	DEPTNO
7369	SMITH	7902	20
7499	ALLEN	7698	30
7521	WARD	7698	30
7566	JONES	7839	20
7654	MARTIN	7698	30
7698	BLAKE	7839	30
7782	CLARK	7839	10
7788	SCOTT	7566	20
7839	KING		10
7844	TURNER	7698	30
7876	ADAMS	7788	20

EMPNO	ENAME	MGR	DEPTNO
7900	JAMES	7698	30
7902	FORD	7566	20
7934	MILLER	7782	10

```
SQL> ALTER TABLE EMP
  2  ADD CONSTRAINT EMP_MGR_FK
  3  FOREIGN KEY(MGR) REFERENCES EMP(EMPNO);

테이블이 변경되었습니다.
```

❑6] 테이블 변경 – 제약조건 삭제

- 제약조건 삭제

  : USER_CONSTRAINTS 와 USER_CONS_COLUMNS 에서 제약조건 이름 식별.

  : CASCADE 옵션은 모든 종속적인 제약조건을 같이 삭제한다.

```
ALTER   TABLE table
DROP    PRIMARY KEY | UNIQUE (column) |
        CONSTRAINT constraint [CASCADE];
```

: (SCOTT_T3 테이블 NOT NULL 삭제)

```
SQL> COL CONSTRAINT_NAME FORMAT A20
SQL> COL TABLE_NAME FORMAT A10
SQL> COL COLUMN_NAME FORMAT A10
SQL>
SQL> SELECT CONSTRAINT_NAME,TABLE_NAME,COLUMN_NAME
  2  FROM USER_CONS_COLUMNS
  3  WHERE TABLE_NAME = 'SCOTT_T3';

CONSTRAINT_NAME      TABLE_NAME COLUMN_NAM
-------------------- ---------- ----------
SYS_C006137          SCOTT_T3   NAME
SCOTT_T3_NUM_PK      SCOTT_T3   NUM


SQL> ALTER TABLE SCOTT_T3
  2  DROP CONSTRAINT SYS_C006137;

테이블이 변경되었습니다.
```

: (SCOTT_T3 테이블 pk 삭제)

```
SQL> CREATE TABLE SCOTT_T3_REF
  2  ( NUM NUMBER(4)
  3  CONSTRAINT SCOTT_T3_REF_FK REFERENCES SCOTT_T3(NUM ) );

SQL> SELECT CONSTRAINT_NAME, COLUMN_NAME
  2  FROM USER_CONS_COLUMNS
  3  WHERE TABLE_NAME = 'SCOTT_T3_REF';

CONSTRAINT_NAME      COLUMN_NAM
-------------------- ----------
SCOTT_T3_REF_FK      NUM

SQL> ALTER TABLE SCOTT_T3
  2  DROP PRIMARY KEY CASCADE;

테이블이 변경되었습니다.

SQL> SELECT CONSTRAINT_NAME, COLUMN_NAME
  2  FROM USER_CONS_COLUMNS
  3  WHERE TABLE_NAME = 'SCOTT_T3_REF';

선택된 레코드가 없습니다.
```

❑6] 테이블 변경 – 제약조건 활성화/비활성화

- 제약조건 활성화/비활성화
  : 기존 제약조건을 잠시 비활성화하는 방법
  : CASCADE 옵션을 추가하면 해당 제약조건과 관련된 제약조건 모두를 비활성화한다.

```
ALTER  TABLE table
DISABLE  CONSTRAINT constraint [CASCADE];
```

```
ALTER  TABLE table
ENABLE   CONSTRAINT constraint [CASCADE];
```

```
SQL> CREATE TABLE SCOTT_T4
  2  ( NUM NUMBER(4) CONSTRAINT SCOTT_T4_NUM_PK PRIMARY KEY,
  3    NAME VARCHAR2(10) );

테이블이 생성되었습니다.

SQL> DESC SCOTT_T4;
 이름                                      널?      유형
 ----------------------------------------- -------- ------------

 NUM                                       NOT NULL NUMBER(4)
 NAME                                               VARCHAR2(10)

SQL> ALTER TABLE SCOTT_T4
  2  DISABLE CONSTRAINT SCOTT_T4_NUM_PK;

테이블이 변경되었습니다.

SQL> DESC SCOTT_T4
 이름                                      널?      유형
 ----------------------------------------- -------- ------------

 NUM                                                NUMBER(4)
 NAME                                               VARCHAR2(10)
```

```
SQL> ALTER TABLE SCOTT_T4
  2  ENABLE CONSTRAINT SCOTT_T4_NUM_PK;

테이블이 변경되었습니다.

SQL> DESC SCOTT_T4;
 이름                                      널?      유형
 ----------------------------------------- -------- ------------

 NUM                                       NOT NULL NUMBER(4)
 NAME                                               VARCHAR2(10)
```

❑6] 테이블 변경 – 제약조건 활성화/비활성화

- CREATE TABLE을 이용한 CREATE INDEX 생성

  : Non-Unique 인덱스를 생성하는 방법.
  : 데이터베이스 성능과 관련된 내용이다.

```
SQL> CREATE TABLE DEPT_NEW
  2  ( DEPTNO NUMBER PRIMARY KEY,
  3    DNAME VARCHAR2(10),
  4    LOC VARCHAR2(10) );

SQL> SELECT INDEX_NAME, UNIQUENESS
  2  FROM USER_INDEXES
  3  WHERE TABLE_NAME = 'DEPT_NEW';

INDEX_NAME      UNIQUENESS
--------------- ----------------
SYS_C006185     UNIQUE
```

인덱스 삭제됨!!

50억건 데이터 저장!!

```
SQL> ALTER TABLE DEPT_NEW DISABLE PRIMARY KEY;
```

50억건 데이터 저장완료!!!

```
SQL> ALTER TABLE DEPT_NEW ENABLE PRIMARY KEY;
```

인덱스 새로 생성됨!!
(Sort!!)

```
SQL> CREATE TABLE dept_new2
  2  ( deptno NUMBER
  3    PRIMARY KEY USING INDEX
  4    ( CREATE INDEX deptno_idx
  5      ON dept_new2(deptno)   ),
  6    dname VARCHAR2(10),
  7    loc VARCHAR2(10));
SQL> SELECT INDEX_NAME, UNIQUENESS
  2  FROM USER_INDEXES
  3  WHERE TABLE_NAME = 'DEPT_NEW2';

INDEX_NAME      UNIQUENESS
--------------- ----------------
DEPTNO_IDX      NONUNIQUE
```

인덱스 삭제 안됨!!

```
SQL> ALTER TABLE DEPT_NEW2 DISABLE PRIMARY KEY;

SQL> SELECT INDEX_NAME, UNIQUENESS
  2  FROM USER_INDEXES
  3  WHERE TABLE_NAME = 'DEPT_NEW2';

INDEX_NAME      UNIQUENESS
--------------- ----------------
DEPTNO_IDX      NONUNIQUE
```

❑6] 테이블 변경 – 제약조건 컬럼 삭제

- 제약조건 설정된 컬럼 삭제

  : CASCADE CONSTRAINTS 옵션을 사용하여 컬럼 삭제시, 해당 컬럼을 참조하는 모든 제약조건을 삭제 할 수 있다.
  또한, 삭제된 컬럼이 포함되어 있는 모든 제약조건도 삭제된다.

```
SQL> CREATE TABLE SCOTT_CAS
  2  ( PK NUMBER PRIMARY KEY,
  3    FK NUMBER,
  4    C1 NUMBER,
  5    C2 NUMBER,
  6   CONSTRAINT SCOTT_CAS_FK FOREIGN KEY(FK) REFERENCES SCOTT_CAS(PK),
  7   CONSTRAINT SCOTT_CAS_CK1 CHECK( PK > 10 AND C1 > 10 ),
  8   CONSTRAINT SCOTT_CAS_CK2 CHECK( C2 > 10 ) );

SQL> SELECT CONSTRAINT_NAME, COLUMN_NAME
  2  FROM USER_CONS_COLUMNS
  3  WHERE TABLE_NAME = 'SCOTT_CAS';

CONSTRAINT_NAME                                                 COLU
--------------------------------------------------------------- ----
SCOTT_CAS_FK                                                    FK
SYS_C006170                                                     PK
SCOTT_CAS_CK2                                                   C2
SCOTT_CAS_CK1                                                   C1
SCOTT_CAS_CK1                                                   PK
```

```
SQL> ALTER TABLE SCOTT_CAS
  2  DROP (PK);
DROP (PK)
      *
2행에 오류:
ORA-12992: 부모 키 열을 삭제할 수 없습니다


SQL> ALTER TABLE SCOTT_CAS
  2  DROP (C1);
DROP (C1)
      *
2행에 오류:
ORA-12991: 열이 다중-열 제약 조건에 참조되었습니다
```

```
SQL> ALTER TABLE SCOTT_CAS
  2  DROP (PK) CASCADE CONSTRAINTS;

테이블이 변경되었습니다.
```

```
SQL> SELECT CONSTRAINT_NAME, COLUMN_NAME
  2  FROM USER_CONS_COLUMNS
  3  WHERE TABLE_NAME = 'SCOTT_CAS';

CONSTRAINT_NAME                                                 COLUMN_NAM
--------------------------------------------------------------- ----------
SCOTT_CAS_CK2                                                   C2
```

Data Dictionary View

데이터 딕셔너리 개념

: 데이터베이스와 관련된 정보를 제공하기 위한 읽기 전용 테이블과 뷰의 집합.
: 메타정보(META DATA)가 저장된다.
: 사용자는 데이터 딕셔너리 테이블과 뷰에 대해 SELECT 명령문만 실행 가능하다.
: 데이터 딕셔너리 테이블 소유는 SYS 가 소유한다.
일반 사용자가 직접 접근하지 못하고, 뷰를 통해서 접근할 수 있도록 지원
(데이터 딕셔너리 뷰)

데이터 딕셔너리 정보

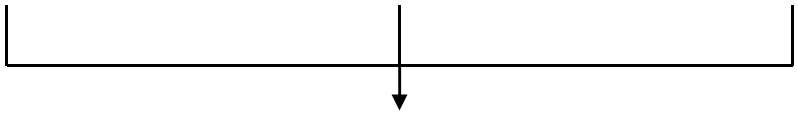
: 데이터베이스의 물리적 구조 또는 객체의 논리적 구조
: 오라클 사용자명과 스키마 객체명
: 각 사용자에게 부여된 권한과 롤(role)
: 무결성 제약조건에 대한 정보
: 컬럼에 대한 기본값
: 스키마 객체에 할당된 영역의 크기와 현재 사용중인 영역의 크기
: 객체 접근 및 갱신에 대한 감사(Audit) 정보
: 데이터베이스 명, 버전, 생성날짜, 시작모드, 인스턴스 명과 같은 일반적인 데이터베이스 정보

- Data Dictionary View

접두사	설명
USER_	사용자가 소유한 객체와 관련된 정보를 포함하고 있는 뷰
ALL_	사용자가 접근 가능한 객체와 관련된 정보를 포함하고 있는 뷰
DBA_	DBA 롤을 부여 받은 사용자들만 접근할 수 있는 제한적인 뷰
V$	데이터베이스 서버 성능, 메모리, 잠금 등의 정보를 포함하고 있는 동적 성능 뷰

```
SELECT * FROM dictionary;
```

DBA_ADVISOR_ACTIONS	ALL_FILE_GROUP_VERSIONS	USER_EXTENTS	V$BACKUP_SPFILE
DBA_ADVISOR_COMMANDS	ALL_HISTOGRAMS	USER_EXTERNAL_LOCATIONS	V$BACKUP_SPFILE_DETAILS
DBA_ADVISOR_DEFINITIONS	ALL_INDEXES	USER_EXTERNAL_TABLES	V$BACKUP_SPFILE_SUMMARY
DBA_ADVISOR_DEF_PARAME...	ALL_INDEXTYPES	USER_FILE_GROUPS	V$BACKUP_SYNC_IO
DBA_ADVISOR_DIRECTIVES	ALL_INDEXTYPE_ARRAYTYP...	USER_FILE_GROUP_EXPORT...	V$BGPROCESS
DBA_ADVISOR_FINDINGS	ALL_INDEXTYPE_COMMENTS	USER_FILE_GROUP_FILES	V$BH
DBA_ADVISOR_JOURNAL	ALL_INDEXTYPE_OPERATORS	USER_FILE_GROUP_TABLES	V$BLOCKING_QUIESCE
DBA_ADVISOR_LOG	ALL_IND_COLUMNS	USER_FILE_GROUP_TABLES...	V$BLOCK_CHANGE_TRACKING
DBA_ADVISOR_OBJECTS	ALL_IND_EXPRESSIONS	USER_FILE_GROUP_VERSIONS	V$BSP
DBA_ADVISOR_OBJECT_TYP...	ALL_IND_PARTITIONS	USER_FREE_SPACE	V$BUFFERED_PUBLISHERS
DBA_ADVISOR_PARAMETERS	ALL_IND_STATISTICS	USER_HISTOGRAMS	V$BUFFERED_QUEUES



Static view

Dynamic Performance View

❑ 7) 데이터 딕셔너리

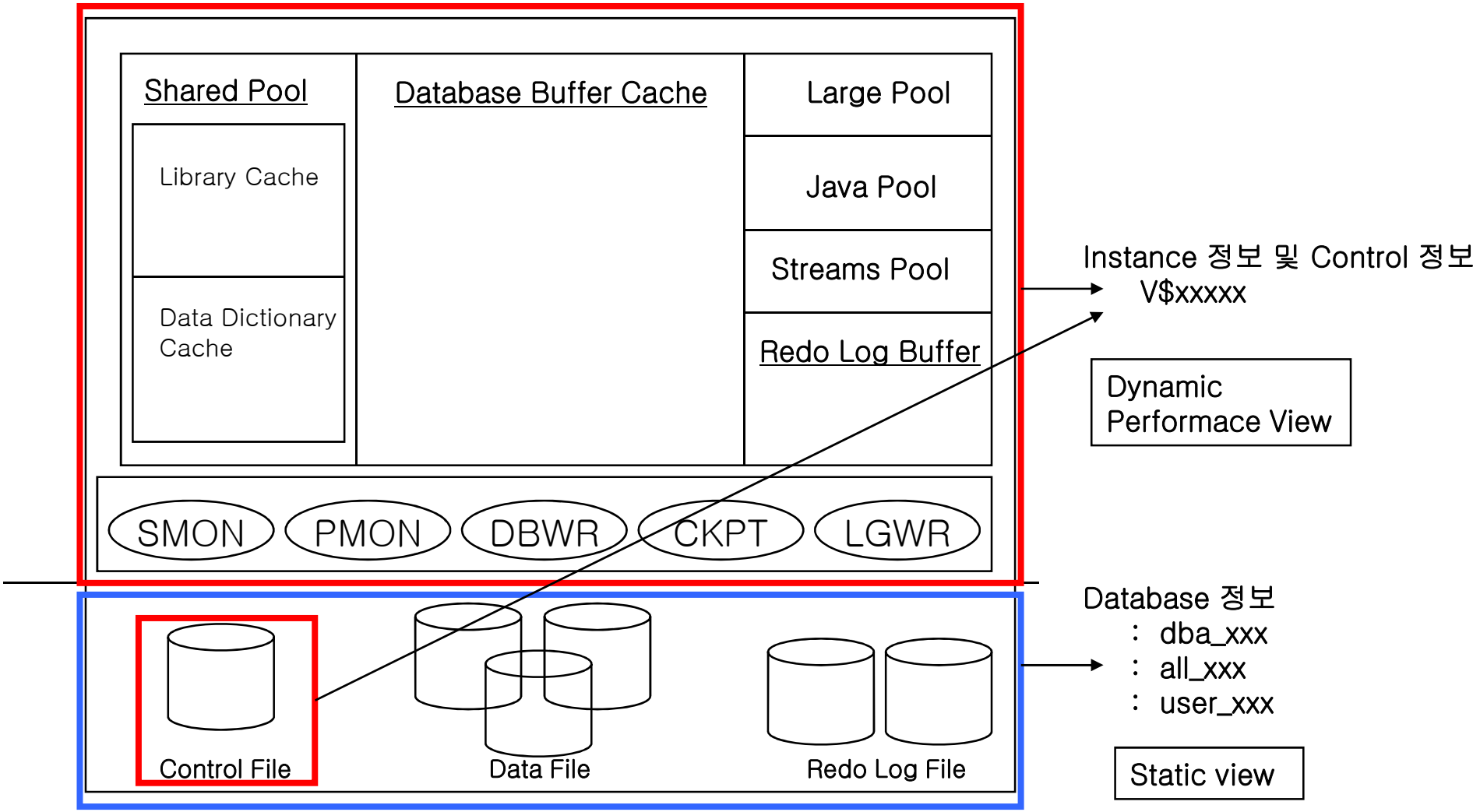
- 동적 성능 뷰 (Dynamic Performance View)

: 현재 데이터베이스 동작 상태를 저장하기 위한 뷰의 집합.
: 시스템 성능 향상을 위해 튜닝 과정에서 주로 활용함.
: DBA는 동적 성능 테이블의 뷰에 질의 하거나 뷰를 생성할 수 있으며, 일반 사용자에게 해당 뷰에 대한 접근 권한을 부여할 수 있다.
: SYS가 소요하며 이름은 모두 V_$ 로 시작한다.
해당 테이블에 대한 뷰가 생성되면 뷰에 대한 동의어가 자동생성됨.
: 동의어 이름은 V$ 로 시작한다.
: 접두어가 V$ 인 데이터딕셔너리는 마운트(mount) 상태에서 조회 가능하다.

- 동적 성능 뷰 (Dynamic Performance View) 종류

: v$datafile
: v$lock
: v$log
: v$logfile
: v$parameter
: v$process
: v$session
: v$sqlarea
: v$sysstat
등이 제공된다.

- Data Dictionary View



Thank you