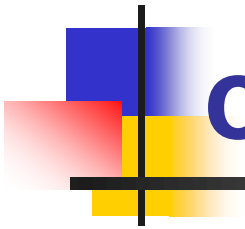
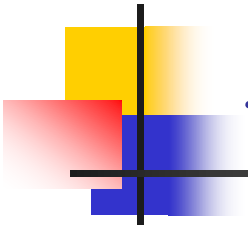


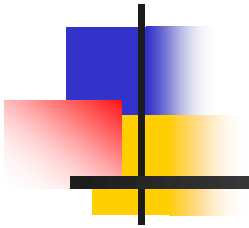
Oracle 7강 –DDL문장, Table 제약조건

양 명 숙
[now4ever7@gmail.com]



목차

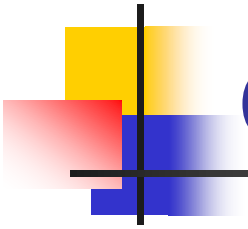
- DDL
  - CREATE
  - ALTER
  - DROP
  - TRUNCATE
  - DELETE, TRUNCATE, DROP 명령어의 차이점
- Table 만들기
- 무결성 제약조건



DDL

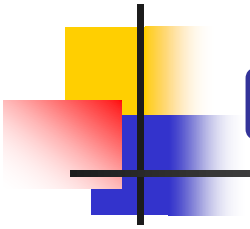
DDL

- 오라클 – 데이터를 저장하고 관리해주는 프로그램
  - 오라클 데이터베이스 내부에 데이터를 관리하기 위해 다양한 저장 객체를 생성 => 오브젝트(object)
    - 데이터베이스 객체라고도 표현, 테이블, 뷰, 인덱스, 시퀀스, 저장 프로시저, 트리거 등
- DDL – 데이터를 저장하고 관리하기 위해 오브젝트를 사용하는데 이런 오브젝트를 생성하고, 변경하고, 관리하는 명령어를 DDL(Data Definition Language)이라고 함
- DDL
  - CREATE
  - ALTER
  - DROP
  - TRUNCATE



CREATE 문

- CREATE – 새로운 오브젝트나 스키마를 생성할 때 사용하는 명령어
  - 데이터베이스 객체들을 생성할 때 사용
- 기본 구조
  - CREATE 객체종류 객체명 …
- 예)
  - 테이블 생성
    - CREATE TABLE 테이블명 …
  - 인덱스 생성
    - CREATE INDEX 인덱스명..



DROP문

- CREATE 와는 반대로 이미 생성된 객체들을 삭제할 때 사용
- DELETE – 테이블 상의 데이터를 삭제할 때 사용
- DROP – 객체 자체를 데이터베이스에서 영구히 없앨 때 사용
  - 객체를 소멸시킴, 데이터도 사라짐
- 기본 구조
  - DROP 객체종류 객체명..
  - 예) drop table 테이블명

ALTER문

- 기존에 생성된 객체를 변경할 때 사용하는 문장
- 예) 테이블 사용 중 컬럼 추가하는 것처럼 테이블 구조를 변경하는 경우
  - alter 객체종류 객체명 …

예) 장소명을 가지는 LOC 칼럼 추가

```
alter table dept6
add(loc  varchar2(10));
```

TRUNCATE문

- DELETE 문과 동일한 역할
  - 테이블에 있는 데이터들을 삭제할 때 사용
- DELETE 문과 차이점
  - TRUNCATE문을 사용하여 데이터를 삭제했을 경우에는 ROLLBACK을 사용하여 데이터를 복구할 수 없음
  - TRUNCATE가 실행되면서 자동 COMMIT이 되기 때문
- TRUNCATE문의 특징
  - DELETE 보다 수행속도가 빠르며 수행 비용이 적다
  - 테이블에 있는 인덱스나 트리거에 영향을 주지 않고 데이터만 삭제함
  - DROP은 테이블에 있는 데이터를 삭제함은 물론 **테이블 구조까지 소멸**시킴
  - DELETE는 메모리상에 존재하는 테이블의 데이터를 삭제
  - TRUNCATE는 **메모리뿐만 아니라 데이터 파일에 있는 데이터까지 삭제**하지만 **테이블 구조는 그대로** 보존함
  - TRUNCATE는 DROP과 DELETE의 중간에 위치한 역할 수행
  - TRUNCATE는 테이블과 클러스터에만 적용 가능

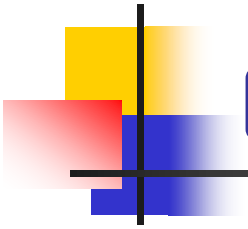
TRUNCATE문

- delete, truncate, drop 명령어의 차이점
  - delete – delete 후 데이터만 지워지고, 쓰고 있던 디스크상의 공간은 그대로 가지고 있다.
    - delete 되어도 테이블 용량은 줄어 들지 않는다
  - truncate – 최초에 테이블이 만들어졌던 상태, 즉 데이터가 1건도 없는 상태로 모든 데이터를 삭제하고 컬럼값만 남겨 놓는다
    - 용량도 줄어들고, 인덱스 등도 모두 삭제됨
    - 테이블을 삭제하지는 않고 데이터만 삭제
  - drop – 데이터와 테이블 전체를 삭제함
    - 사용하고 있던 공간도 모두 반납하고 인덱스나 제약조건 등 오브젝트도 삭제됨

TRUNCATE문

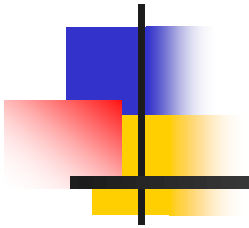
- 구문 형식

```
TRUNCATE  TABLE  테이블명...
```

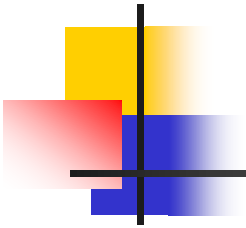


DCL(Data Control Language)

- DCL(데이터 제어어)
  - 데이터베이스에 있는 데이터에 접근을 제어하는 언어
  - GRANT – 접근 제어나 어떤 작업을 허용하는 권한을 주는 역할
  - REVOKE – 허용된 권한을 없애는 역할



Table



테이블(TABLE)

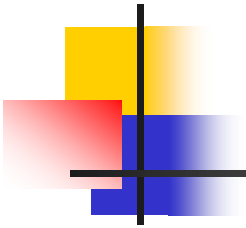
- 테이블(TABLE)
  - 데이터를 저장하고 있는 테이블
  - 데이터베이스상에서 가장 기본이 되는 객체
  - 데이터를 담고 있는 상자
  - 상자에 물건들을 보관하고 꺼내고 하듯이 데이터베이스에서는 데이터들을 테이블을 통해서 관리함
  - 2차원의 평면 구조, 표를 데이터베이스에 그대로 옮겨 놓은 것
  - row(행, 가로축)와 column(열, 세로축)으로 구성
  - 저장되는 데이터는 몇 번째 row의 특정 column값 단위로 구분할 수 있음

데이터베이스와 테이블의 관계

데이터베이스라는 큰 통 안에 정보가 들어있는 다수의 테이블이 존재함

테이블(TABLE)

- 데이터베이스
  - 데이터들을 저장하는 곳
  - 데이터베이스 전체를 하나의 거대한 창고라고 생각하자
- 창고에서 물건을 관리하는 방법은 데이터베이스에서 데이터를 관리하는 것과 유사함
  - 창고 – 여러 가지 종류의 물품들을 보관
  - 매일 많은 물품들이 창고에 입고되거나 출고
  - 창고의 규모가 크기 때문에 창고 전체를 관리하는 총괄 책임자가 존재 (DBA)
  - 창고 내에서는 구간을 나누어 물품들을 보관 (테이블스페이스)
  - 각 구간에는 물품의 성격에 따라 이를 관리하는 담당자들이 존재 – 이들이 물품의 입고, 출고, 관리를 맡고 있음 (스키마소유자)
  - 개별 물품들은 하나의 큰 컨테이너 박스에 보관됨, 이 박스들에는 동일한 성격을 가진 물품들을 보관하고 있음 (테이블)



테이블(TABLE)

- 총괄 책임자 => DBA
- 구간 => 테이블스페이스
- 담당자 => 사용자로서 여기서는 스키마 소유자
- 컨테이너 박스 => 테이블
- 개별 물품 => 테이블에 저장되는 데이터
- 입고, 출고 => insert, delete 등과 같은 DML

TABLE 만들기

- TABLE 만들기
  - 테이블 생성 패턴 – CREATE TABLE
  - 구문형식

```
CREATE TABLE   테이블명
(
        컬럼1   데이터타입,
        컬럼2   데이터타입,
        …
)
TABLESPACE  테이블스페이스명 ;
```

Tablespace – 이 테이블이 저장되는 테이블스페이스 지정
생략 가능, 생략할 경우 현재 로그인 한 사용자의 기본 테이블스페이스로 자동 저장됨

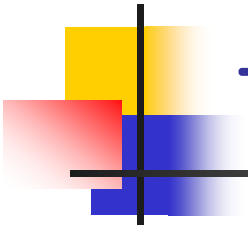


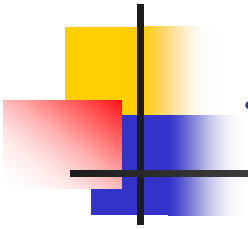
TABLE 만들기

- TABLE 만들기
  - 테이블에 있는 데이터들의 조회, 삽입, 삭제 등은 모두 ROW 단위로 이루어지고, 테이블에 있는 컬럼들이 모여 하나의 ROW를 구성하게 됨
    - (UPDATE 와 같이 데이터의 변경은 ROW 단위이기는 하지만 컬럼별로 이루어짐)
  - 테이블이름 – 최대 30 byte, 한글의 경우 15자
  - 컬럼명, 스키마, 사용자 이름도 30 byte까지 사용
  - 한 테이블에는 최대 255개까지 컬럼을 만들 수 있음
  - 컬럼이 가질 수 있는 데이터 타입
    - 문자, 숫자, 날짜 등 오라클에서 제공하는 기본 데이터 타입
    - 사용자 정의 데이터 타입

기본 데이터 타입

- 기본 데이터 타입(built-in datatype)
  - 문자형, 숫자형, 날짜형 데이터
- 문자형 데이터 (CHAR형 데이터) – CHAR, VARCHAR2, CLOB, NCLOB

데이터 유형	설명	길이
CHAR [(size [BYTE| CHAR])]	고정 길이의 문자형 데이터 타입으로 그 크기는 size 만큼의 byte 수 혹은 문자 개수가 됨	최대크기는 2000byte가 될 수 있으며, BYTE나 CHAR를 명시하지 않을 경우 디폴트 값은 BYTE. Size 값을 명시하지 않을 경우 디폴트값은 1
VARCHAR2 (size [BYTE| CHAR])	가변 길이의문자형데이터 타입으로 그 크기는 최대 size만큼의 byte 수 혹은 문자수가 됨. 반드시 size 값을 명시해야 함	최대크기는 4000byte가 될 수 있으며, BYTE나 CHAR를 명시하지 않을 경우 디폴트 값은 BYTE. Size 값을 생략할 수 없음
NCHAR[(size)]	고정 길이의 유니코드 문자형 데이터 타입으로 그 크기는 utf8 인코딩의 경우 size*3byte, AL16UTF16 인코딩의경우 size*2byte가 됨	최대크기는 2000byte. Size 값을 명시하지 않을 경우 디폴트값은 1
NVARCHAR2(size)	가변 길이의 유니코드 문자형 데이터 타입으로 그 크기는 최대 size 값이 되며, utf8의 경우 3배, AL16UTF16 인코딩의 경우 2배의 byte가 됨	최대크기는 4000byte. Size 값을 생략할 수 없음
LONG	가변 길이의 문자형 데이터 타입	최대 크기는 2GB



문자형 데이터

- 문자형 데이터
  - CHAR – 고정된 길이의 데이터 타입
  - VARCHAR2 – 가변 길이의 데이터 타입
- 고정 길이
  - 저장될 공간의 크기가 미리 정해져 있어 데이터가 저장될 때 그 크기만큼의 저장 공간이 그대로 보존되는 것
- 가변 길이
  - 공간의 크기가 미리 정해지기는 하지만 저장되는 데이터의 크기에 따라 실제 저장 공간의 크기가 바뀔 수 있는 타입

문자형 데이터

예)

```
CREATE TABLE chr_exam1(
   names1   char(3 byte),
   names2   varchar2(3 byte)
);

insert into chr_exam1 values('AA','AA');

select replace(names1, ' ', 'B') nm1,
    replace(names2, ' ', 'B') nm2
from chr_exam1;
```

NM1	NM2
AAB	AA

공백이 있으면 그 자리를 'B' 문자로 채워서 보여주는 문장

- 고정 길이인 names1 칼럼 – 2개의 문자를 입력했지만 총 3문자의 공간 차지
  - 실제 저장되는 공간은 3byte
  - 데이터의 크기가 명확하게 정해져 있는 컬럼의 경우
- 가변 길이인 names2 칼럼 – 크기는 3자리이지만 2자리 데이터가 들어가서 실제 크기도 2자리를 차지
  - 실제 저장되는 공간은 2byte
  - 저장 공간의 효율성 측면에서 훨씬 경제적
  - 저장되는 데이터의 크기가 변동적일 경우

문자형 데이터

```
CREATE TABLE chr_exam2(
   names3  char(3 byte),
   names4  char(3 char), --3글자의 크기를 갖는다(영문 3byte, 한글 6byte)
   names5  char(3) - 디폴트값으로 byte 적용
);

select * from chr_exam2;

insert into chr_exam2 values ('AAA','AAA','AAA');
insert into chr_exam2 (names3, names4) values ('AAA','AAA');
insert into chr_exam2 (names3) values ('홍길동'); --에러, 6byte 이므로 저장 불가
insert into chr_exam2 (names4) values ('홍길동');

commit;
```

- 오라클 환경변수인 NLS_LENGTH_SEMANTICS 변수에 할당된 값이 BYTE냐 CHAR 냐에 따라
- NLS_LENGTH_SEMANTICS 변수의 디폴트값 : BYTE

오라클 파라미터

- 오라클에는 많은 파라미터가 존재함
- 이 파라미터에 설정된 값에 따라 데이터베이스 동작이나 환경이 바뀌게 됨
- 파라미터에 설정된 값을 보는 방법
  - [1] SQL*PLUS 를 사용
    - SHOW PARAMETER 파라미터명;
    - SHOW PARAMETER NLS_LENGTH_SEMANTICS;

```
SQL> CONN as sysdba
사용자명 입력: sys
암호 입력:
연결되었습니다.
SQL> SHOW PARAMETER NLS_LENGTH_SEMANTICS;

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------
nls_length_semantics                 string      BYTE
```

```
select * from nls_session_parameters
where parameter = 'NLS_LENGTH_SEMANTICS';
```

오라클 파라미터

- [2] 데이터 딕셔너리인 시스템 뷰를 조회해보는 것
- 파라미터 정보를 갖고 있는 시스템 뷰는 V_$PARAMETER 이며, 소유자는 SYS 임

```
SQL> SELECT name, type, value
  2  FROM sys.V_$PARAMETER
  3  WHERE name='nls_length_semantics';

NAME
----------------------------------------

     TYPE
----------
VALUE
----------------------------------------

nls_length_semantics
         2
BYTE
```

문자형 데이터

- LONG 타입
  - 2GB 까지 저장할 수 있는 가변 길이를 가진 문자형 데이터 타입
  - 컬럼 하나에 대용량 데이터를 저장하고자 할 경우 사용
  - 텍스트 형태가 아닌 데이터들의 경우 long 타입 사용
    - 예전에는 사진을 저장할 때 LONG 타입 사용
  - CLOB, NCLOB와 같은 LOB 타입이 등장한 이후로 오라클 사에는 더 이상 LONG 타입을 사용하지 않도록 권고
    - LOB(Large Object) 타입들이 훨씬 더 강력한 기능들을 제공하므로
    - LONG 타입이 10g 버전까지 존재하는 이유는 오직 하위 버전과의 호환성을 위해서.
- NCHAR, NVARCHAR2 타입
  - CHAR, VARCHAR2 와 같으며, 유니코드를 사용할 경우에 사용됨
  - 다양한 언어의 문자 값 저장

문자형 데이터

- LOB(Large Object)
  - 대용량의 데이터를 저장하고 관리하기 위해 오라클에서 제공하고 있는 기본 데이터 타입
  - 기존의 텍스트 기반의 구조적인 데이터 뿐만 아니라 사진, 음악, 동영상 등의 비 구조적인 데이터를 저장하고 관리하기 위한 타입
  - long, long row => LOB

문자형 데이터

- LOB 타입의 종류
  - 내부 LOB 타입
    - CLOB 타입(Character Large Object)
      - 데이터베이스 단일 바이트 문자셋으로 저장,
      - 크기가 큰 문자열이나 문서의 저장이 가능,
      - long 타입이 확장된 형태, 4GB 까지 저장
    - BLOB 타입(Binary Large Object)
      - 바이너리 형태로 저장,
      - 사진, 비디오, 오디오 데이터 저장시 주로 사용,
      - long raw 타입이 확장된 형태, 4GB 저장
    - NCLOB 타입(National Character Set Large Object)
      - CLOB와 같으나 National Character Set 으로 저장
  - 외부 LOB 타입
    - Bfile 타입(External Binary File)
    - 오라클 외부 운영체제 파일 시스템에 저장되지만, 오라클 테이블에서 접근이 가능
    - 실제 데이터가 오라클 외부에 저장, 읽기만 가능, DVD나 CD 등 큰 비구조화된 데이터 저장

숫자형 데이터

- 다양한 형태의 숫자 정보들을 저장하고 관리하기 위해 지원되는 데이터 타입

데이터 유형	설명	길이
BYNARY_FLOAT	32 bit 부동 소수	4byte
BYNARY_DOUBLE	64 bit 부동 소수	8byte
NUMBER [(prec | prec, scale)]	가변 숫자 타입으로, prec로 설정한 값(1~38)은 전체 자릿수, Scale 로설정한 값(-84~127)은 소수점 이하 자릿수를 의미함	최대값은 21byte

- 숫자 – 정수(양의정수, 음의 정수, 0), 실수(고정 소수점 수,부동 소수점 수)
- NUMBER 타입 – 정수를 포함한 모든 실수를 표현하고 저장하는 데 사용
  - NUMBER 타입으로 저장할 수 잇는 숫자의 범위
    - $1*10^{-130}$ ~ $9.99\cdots9 * 10^{125}$ 까지의 38자리 양수 및 음수
    - 0

숫자형 데이터

- NUMBER 타입 구문 형식

컬럼명	NUMBER(precision, scale)

- precision – 전체 자릿수
  - 적용할 수 있는 최대값의 precision은 38
- scale – 소수점 이하 자릿수
  - 생략하면 0 이 적용됨
- precision, scale 모두 생략 가능

숫자형 데이터

예)

```
CREATE TABLE num_temp1(
   n1  number,
   n2  number(9),
   n3  number(9,2),
   n4  number(9,1),
   n5  number(7),
   n6  number(7,-2),
   n7  number(6),
   n8  number(3,5)
);
```

	N1	N2	N3	N4	N5	N6
▶	1234567.89	1234568	1234567.89	1234567.9	1234568	1234600
	1234567.89	1234568	1234567.89	1234567.9	1234568	1234600

```
insert into num_temp1(n1, n2, n3, n4, n5, n6)
values(1234567.89,1234567.89,1234567.89,1234567.89,1234567.89,1234567.89);

select n1, n2, n3, n4, n5, n6 from num_temp1;

insert into num_temp1(n1, n2, n3, n4, n5, n6, n7)
values(1234567.89,1234567.89,1234567.89,1234567.89,1234567.89,1234567.89, 1234567.89);
```

--에러

N1	N2	N3	N4	N5	N6
1234567.89	1234568	1234567.89	1234567.9	1234568	1234600
1234567.89	1234568	1234567.89	1234567.9	1234568	1234600

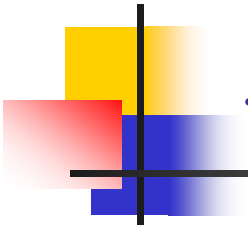
숫자형 데이터

- n1 number
  - 크기를 지정하지 않아 자릿수의 제한이 없다
  - 최대 39~40자리 숫자까지만 정확도가 보장됨
- n2 number(9)
  - 전체 자릿수가 9자리, scale 생략=>0,
  - 소수점 이하 첫 번째 자리에서 반올림
- n3 number(9,2)
  - 전체 9자리, 소수점 이하 2자리
- n4 number(9,1)
  - 소수점 이하 자릿수 1, 소수점 두 번째 자릿수에서 반올림
- n5 number(7)
  - 전체 자리수가 7자리 => 소수점 이하 첫째 자리에서 반올림
- n6 number(7,-2)
  - Scale 이 -2 => 소수점 기준으로 오른쪽이 아닌 왼쪽 두 번째 자리(십의 자리)에서 반올림

숫자형 데이터

- n8 number(3,5)
  - 전체 자릿수 3, 소수점 이하 자릿수 5 => scale 값이 precision 보다 클 경우는 1보다 작은 실수 입력시 사용
  - 소수점 이하 자릿수가 5자리이고, 소수점 이하 맨 앞에 0 이 2개(5-3)가 붙게 됨
  - 예) 0.00123 저장
  - 0.01234는 저장되지 않음
  - 소수점 이하 0 이 2개, 소수점 이하 자릿수가 5개 미만인 숫자 0.0012는 저장 가능

```
select * from num_temp1;

update num_temp1
   set n8=0.00123;

update num_temp1
   set n8=0.01234;  --에러
```



숫자형 데이터

- BYNARY_FLOAT, BYNARY_DOUBLE
  - 2진법의 정밀도를 사용
  - 0과 1의 수만 사용하므로 number 타입보다 정밀도가 낮다
  - Number 타입 – 숫자를 저장할 때 10진법으로 정밀도를 제어

날짜형 데이터

데이터 유형	설명	길이
DATE	고정 길이의 날짜와 시간 데이터로서 BC 4712년 1월 1일부터 9999년 12월 31일까지 표현	7 byte, 표현형태는 NLS_DATE_FORMAT 파라미터에 명시된 값을 따름 (예: DD-MON-RR, 01-JAN-99)
INTERVAL YEAR TO MONTH	연도와 월 형태로 기간을 표현	5 byte
INTERVAL_DAY TO SECOND	요일, 시, 분, 초 형태로 기간 표현	11byte
TIMESTAMP [(frac_sec_prec)]	밀리초 까지 표현 가능한 날짜형 데이터 타입	
TIMESTAMP [(frac_sec_prec)] WITH TIME ZONE	시간대와 더불어 날짜와 시간을 표현 시간대는 '-5:0' 나 'US/Pacific'	
TIMESTAMP [(frac_sec_prec)] WITH LOCAL TIME ZONE	TIMESTAMP WITH TIME ZONE과 비슷. 저장시점에는 데이터베이스 시간대를 준수하나 조회시에는 조회하는 클라이언트의 시간대로 표현	

날짜형 데이터

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY-MM-DD';

세션이 변경되었습니다.

SQL> select * from date_temp;

DATE1
----------
2013-02-17
```

- DATE 타입이 가장 많이 사용됨
  - 날짜와 시간정보를 저장하는데 세기, 연도, 월, 일, 시, 분, 초 정보 저장

```
CREATE TABLE date_temp(
    date1  date
);

insert into date_temp values(sysdate);

select * from date_temp;
```

```
SQL> select * from date_temp;

DATE1
--------
13/02/17
```

- sysdate – 오라클에서 제공하는 시스템의 현재 날짜 정보를 가지고 있음
- 조회된 날짜의 표현 형식 – NLS_DATE_FORMAT 파라미터에 설정된 값 형태로 표현
- 디폴트값 => YY/MM/DD
- 파라미터 값 변경하기 – ALTER SYSTEM 이나 ALTER SESSION 명령으로.
- ALTER SESSION으로 변경 => 현재세션에만 변경된 값이 적용(다시 로그인하면 원래대로)

날짜형 데이터

- TIMESTAMP
  - DATE 타입보다 확장된 날짜형 데이터 타입
  - 년, 월, 일, 시, 분, 초, 밀리초 정보 값을 저장
  - DATE 타입보다 더 정밀한 시간 정보를 나타냄

```
TIMESTAMP [(fractional_seconds_precision)]
```

  - fractional_seconds_precision – 초 정보의 정밀도를 나타내는데 0~9까지 숫자 사용, 생략시 디폴트값 6
    - 초단위 다음 단위인 Mili Second를 나타낼 자리수
- TIMESTAMP WITH TIME ZONE
  - 각 지역시간(local time)과 그리니치 표준시와의 차이를 포함한 TIMESTAMP 의 변형된 데이터 타입

```
TIMESTAMP[(fractional_seconds_precision)]
WITH  TIME ZONE
```

날짜형 데이터

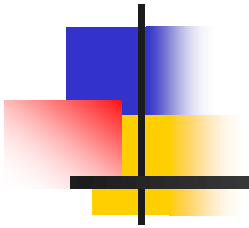
```
CREATE TABLE date_temp2(
 time1  timestamp,
 time2  timestamp with time zone,
 time3  timestamp with local time zone
);

insert into date_temp2 values(sysdate, sysdate, sysdate);

select * from date_temp2;


ALTER SESSION SET NLS_TIMESTAMP_FORMAT = 'YYYY-MM-DD HH:MI:SS.FF';
--ALTER SESSION SET NLS_TIMESTAMP_TZ_FORMAT = 'YYYY-MM-DD HH:MI:SS.FF';
```

```
SQL> select * from date_temp2;

TIME1
---------------------------------
TIME2
---------------------------------
TIME3
---------------------------------
2013-02-17 03:55:03.000000
13/02/17 15:55:03.000000 +09:00
2013-02-17 03:55:03.000000
```

예제 – table 만들기

```
create table test1
(
    name            varchar2(20),  --20byte (최대 4000 byte)
    ssn             char(14), --(최대 2000 byte)
    content         clob,  --(최대 4gb)
    gender          char,  --1byte

    age             number,
    n1              number,
    n2              number(9),
    n3              number(9, 1),
    n4              number(7, -2),
    n5              number(3, 5),

    regdate         date
);

insert into test1
values('홍길동', '900617-1112222', '글쓰기 내용이 많다면', 'M',  20,
1234567.89, 1234567.89, 1234567.89, 1234567.89, 0.00123,  sysdate );
```

```
update test1
set n5 = 0.01234
where name = '홍길동';  --에러

update test1
set n5 = 0.00012
where name = '홍길동';  --0.00123 =>0.00012
```



무결성 제약조건

데이터 무결성(Data Integrity)

- 데이터 무결성
  - 오라클 서버에서 **데이터를 흠 없이 안정되게 지켜 주는 것**
    - 무엇을 지켜 주는가의 관점에서 세 부분으로 나눔
  - 권한이 있는 사람만 해당 데이터를 볼 수 있어야 하며, 데이터의 정확성을 보장한다는 의미를 가짐
  - 제대로 된 데이터들이 올바르게 저장될 수 있도록 하기 위해 데이터베이스 측에서 제공하는 기능들

  - 무엇을 지켜 주는가

설명		방법들
실체, 개체 (entity)	레코드의 존재는 고유해야 한다	primary key, unique
영역 (domain)	특정한 범위의 값만이 와야 한다	check, default
참조 (referential)	다른 속성(컬럼)과의 관계를 위반하지 않아야 한다	foreign key

무엇을 지켜 주는가

- 실체 무결성 — 레코드의 존재는 고유해야 한다
  - primary key 가 대표적
  - 여러 데이터(레코드)들 중에서 유일하게 구분 지어 줄 수 있는 것이 실체 무결성
- 영역 무결성 — 특정한 범위의 값만이 와야 한다
  - 예) “주민등록번호의 값으로 1이나 2, 또는 3, 4만 입력되어야 한다” 와 같은 것들이 영역 무결성임
  - city 라는 컬럼에는 ‘서울’,’대전’,’부산’ 등만이 와야 한다
- 참조 무결성 — 다른 속성(컬럼)과의 관계를 위반하지 않아야 한다
  - foreign key와 primary key 와의 관계
  - 사원 테이블에서 한 사원의 부서코드는 반드시 부서 테이블에 존재해야 한다
    - 만약 그런 부서가 존재하지 않는다면 참조 무결성은 깨진 것임

컬럼 속성(무결성 제약조건)

- 데이터 무결성
  - 무결성을 지키기 위해 제약조건들을 제공함–제약조건들은 테이블의 컬럼에 적용됨
- 무결성 제약조건 (Integrity Constraints)
  - NULL
  - UNIQUE
  - PRIMARY KEY
    - NOT NULL + UNIQUE (INDEX)
    - 하나 이상 컬럼으로 구성됨.
  - FOREIGN KEY
    - 부모 테이블의 P.K가 자식 테이블의 F.K.
  - CHECK
  - DEFAULT

컬럼 속성(무결성 제약조건)

- NULL
  - 데이터가 없음을 의미
  - 컬럼의 속성 중 하나로 해당 컬럼이 null값을 허용하는지 허용하지 않는지 지정
  - 데이터 타입 다음에 명시함
  - Null을 허용하면 null, 허용하지 않으면 not null 명시
  - 명시하지 않으면 디폴트 값 : null

```
CREATE TABLE null_test (
   col1  VARCHAR2(20) NOT NULL,
   col2  VARCHAR2(20) NULL,
   col3  VARCHAR2(20)
);
INSERT INTO null_test ( col1, col2 )  VALUES ('AA', 'BB');
INSERT INTO null_test ( col2, col3 )  VALUES ('CC', 'DD');  --에러
INSERT INTO null_test ( col1, col2, col3 )  VALUES ('TT1', 'SS', NULL);
INSERT INTO null_test ( col1, col2, col3 )  VALUES ('TT2', 'SS', '' );
INSERT INTO null_test ( col1, col2, col3 )  VALUES ('TT3', 'SS', ' ' ); --null아님
```

Null 값을 직접 넣기 위해서는 null 이나 ''을 넣으면 됨
주의 : Null은 값이 없는 것으로 공백(' ')과는 다름

```
ORA-01400: NULL을 ("SCOTT"."NULL_TEST"."NULL_COL1") 안에 삽입할 수 없습니다
```

컬럼 속성(무결성 제약조건)

- UNIQUE 키
  - 테이블에 있는 데이터를 유일하게 식별하기 위한 무결성 제약조건 중 하나
  - 예) 사원 테이블의 사원번호 – unique 키 => 사원번호 컬럼의 값은 중복값을 허용하지 않게 됨
  - 예) 사원 테이블의 이메일 주소 – unique 키
  - 복합 unique 키 – 한 개 이상의 컬럼으로 unique 키를 만드는 것

```
CREATE TABLE unique_test (
        col1  VARCHAR2(10) UNIQUE NOT NULL,
        col2  VARCHAR2(10) UNIQUE,
        col3  VARCHAR2(10) NOT NULL,
        col4  VARCHAR2(10) NOT NULL,
        CONSTRAINTS uni_tmp_uk  UNIQUE ( col3, col4 ) --복합 unique 키
);
INSERT INTO unique_test ( col1, col2, col3, col4 ) VALUES ('A1', 'B1', 'C1', 'D1' );
INSERT INTO unique_test ( col1, col2, col3, col4 ) VALUES ('A2', 'B2', 'C2', 'D2' );

UPDATE unique_test
   SET col1 = 'A1'
 WHERE col2 = 'B2';  --에러(unique 키가 걸린 칼럼에 동일한 값 'A1'을 다시 입력하여 제약조건에 위배)
```

Unique 키가 null을 허용하기는 하지만, not null 속성을 주는 것이 좋다

UNIQUE 키

```
INSERT INTO unique_test ( col1, col2, col3, col4 ) VALUES ('A3', null, 'C1', 'D2' );
INSERT INTO unique_test ( col1, col3, col4 ) VALUES ('A4',  'C2', 'D1' );
INSERT INTO unique_test ( col1, col2, col3, col4 ) VALUES ('A5', '', 'C3', 'D3' );
```

null을 여러 번 입력했는데도 별다른 오류 없이 모두 저장됨
⇒null 은 값이 없음을 의미하므로 **unique** 키 비교 대상에서 제외되기 때문
⇒unique 키는 null을 무시하고 있는 것

Unique 키가 null을 허용하기는 하지만, not null 속성을 주는 것이 좋다

컬럼 속성(무결성 제약조건)

- PRIMARY KEY
  - Unique와 동일하게 한 테이블에 있는 데이터들을 유일하게 식별하기 위한 무결성 제약조건
  - 기본키가 걸린 컬럼들은 반드시 not null 속성을 가짐
    - NOT NULL + UNIQUE 인덱스
  - 하나 이상 컬럼으로 구성됨 – 복합키 가능
  - 테이블 하나당 오직 1개만 생성 가능
- 기본 키 정의하는 방법
  - [1] 인라인(inline) 표기방식

```
emp_id number(6,0) PRIMARY KEY
```

  - [2] 아웃라인(out-line) 표기방식

```
emp_id number(6,0),
CONSTRAINTS "emp_pk" PRIMARY KEY(emp_id)
```

후보키(Candidate key)

• 사원테이블의 후보키
사원번호 - 기본키
주민등록 번호 – 대체키
이메일-대체키

- 테이블 정의 후 후보키(Candidate key) 선정
  - 실체 내 특정 건의 유일성을 보장하는 속성을 후보키
  - 레코드를 유일하게 식별할 수 있는 속성이나 속성들의 집합
  - 모든 실체는 최소한 하나 이상의 후보 키 보유
- 대체키(Alternative key) – 후보키 중 기본키를 제외한 나머지 후보키
- CK = PK + AK(Alternate key)
  - 여러 후보키 중에서 **업무적으로 활용도가 높고**, 가능하면 **길이가 짧은 것**을 **기본키**로 선정
  - 기본키의 특성은 Unique 인덱스 & not null의 제약 조건
  - 나머지는 대체키(AK) 로 구분되면서 향후 테이블과 인덱스 생성시 주로 unique 인덱스로 처리

컬럼 속성(무결성 제약조건)

- 생성된 제약조건들에 대한 정보는 USER_CONSTRAINTS 시스템 뷰로 조회하여 확인

```
select * from user_constraints
order by table_name;
```

- 테이블 생성 시 기본키를 지정하지 않고, 생성 후 ALTER TABLE 문을 사용하여 기본키 생성 가능

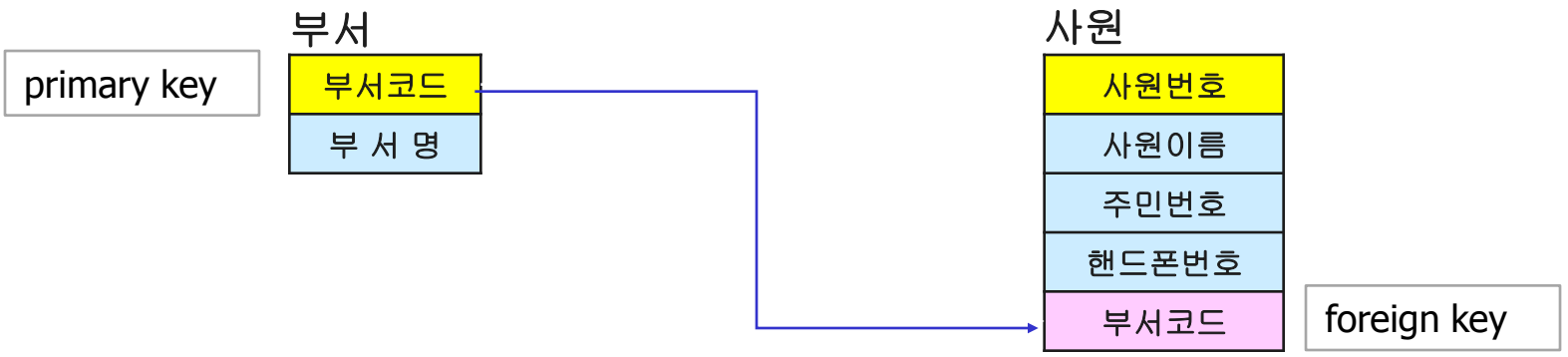
```
ALTER TABLE 테이블명
ADD CONSTRAINTS "기본키 이름" PRIMARY KEY(컬럼);
```

```
ALTER TABLE  emp
ADD CONSTRAINTS "emp_pk" PRIMARY KEY(emp_id);
```

컬럼 속성(무결성 제약조건)

- 다른 테이블을 참조하기 위하여 사용되는 속성들
- 테이블 간에 관계를 설정할 때 사용되는 키

- 외래키(Foreign Key)
  - 관계(relationship) – 테이블 간의 관계
  - 관계는 테이블 간에 공통값을 가진 컬럼을 통해서 이루어짐
  - 예) emp(사원테이블), dept(부서 테이블) => deptno(부서코드) 컬럼으로 관계를 맺을 수 있음
    - deptno는 dept 테이블에서는 기본키, emp 테이블에서는 외래키로 정의(참조정보를 갖고 있을 뿐)
    - emp 테이블이 dept 테이블을 참조한다고 표현
    - dept : 부모 테이블, emp : 자식 테이블
  - 부모 테이블의 P.K(기본키)가 자식 테이블의 F.K.(외래키)
  - emp 테이블의 deptno는 반드시 dept 테이블에 존재하는 deptno 값을 가져야 함
    - 외래 키를 지정했다면 존재하지 않는 부서코드 값은 저장 자체가 되지 않음



Foreign Key

사원 테이블(자식)

사번	이름	주민번호	핸드폰번호	부서코드
1	홍길동	801011-1112222	010-111-2222	A01
2	김길동	820331-1444422	011-155-2265	A02
3	김창원	781023-1322222	019-224-3443	A01
4	이재석	830812-1756442	016-656-1654	A03
5	김석기	750419-1343432	017-231-8766	A02
6	이주원	860621-1134453	010-145-3566	A04
7	김남희	640101-1234556	010-456-2454	A04

FK(외래키)

부서 테이블(부모)

부서코드	부서명
A01	총무부
A02	인사부
A03	영업부
A04	홍보부

PK(기본키)

예제 – 게시판 테이블

번호	이름	작성일	제목	한줄답변 작성자	한줄답변 내용	한줄답변 작성일
1	홍길동	2012-10-04	좋은 하루~	이지성	너두	2012-10-04
1	홍길동	2012-10-04	좋은 하루~	김재성	오늘도 힘내자	2012-10-05
2	김창원	2012-10-06	안녕			
3	이재석	2012-10-08	시험 잘봐	홍길동	ok	2012-10-08
3	이재석	2012-10-08	시험 잘봐	김창원	^^	2012-10-08
3	이재석	2012-10-08	시험 잘봐	이지성	화이팅	2012-10-09
4	김남희	2012-10-10	잘 지내			

예제 – 게시판 테이블 분할

게시판 테이블

번호	이름	작성일	제목
1	홍길동	2012-10-04	좋은 하루~
2	김창원	2012-10-06	안녕
3	이재석	2012-10-08	시험 잘봐
4	김남희	2012-10-10	잘 지내

PK(기본키)

한줄답변 테이블

원본번호	번호	한줄답변 작성자	한줄답변 내용	한줄답변 작성일
1	1	이지성	너두	2012-10-04
1	2	김재성	오늘도 힘내자	2012-10-05
3	3	홍길동	ok	2012-10-08
3	4	김창원	^^	2012-10-08
3	5	이지성	화이팅	2012-10-09

FK(외래키)

emp2

외래키(foreign key)

EMPNO	NAME	BIRTHDAY	DEPTNO	EMP_TYPE	TEL	HOBBY	PAY	POSITION	PEMPNO
19900101	나사장	1964/01/25	0001	정규직	054)223-0001	음악감상	100000000	대표이사	
19960101	전부장	1973/03/22	1000	정규직	02)6255-8000	독서	72000000	부장	19900101
19970201	최일도	1975/04/15	1000	정규직	02)6255-8005	운동	50000000	과장	19960101
19930331	백원만	1976/05/25	1001	정규직	02)6255-8010	자전거타기	60000000	차장	19960101
19950303	천만득	1973/06/15	1002	정규직	02)6255-8020	마라톤	56000000	과장	19960101
19966102	일지매	1972/07/05	1003	정규직	052)223-4000	음악감상	75000000	부장	19900101
19930402	유관순	1972/08/15	1004	정규직	042)998-7005	등산	51000000	과장	19966102
19960303	김문호	1971/09/25	1005	정규직	031)564-3340	등산	35000000	대리	19966102
19970112	노정호	1976/11/05	1006	정규직	054)223-4500	수영	68000000	부장	19900101
19960212	이윤나	1972/12/15	1007	정규직	054)223-4600		49000000	과장	19970112
20000101	이태백	1985/01/25	1008	계약직	051)123-4567	등산	30000000		19960212

dept2

DCODE	DNAME	PDEPT	AREA
0001	사장실		포항본사
1000	경영지원부	0001	서울지사
1001	재무관리팀	1000	서울지사
1002	총무팀	1000	서울지사
1003	기술부	0001	포항본사
1004	H/W지원	1003	대전지사
1005	S/W지원	1003	경기지사
1006	영업부	0001	포항본사
1007	영업기획팀	1006	포항본사
1008	영업1팀	1007	부산지사
1009	영업2팀	1007	경기지사
1010	영업3팀	1007	서울지사
1011	영업4팀	1007	울산지사

M : 1 관계

M : 1
- dept2 테이블에 있는 한 row의 dcode값이 emp2 테이블의 deptno 에는 여러 row 가 올 수 있음을 의미

Primary Key(기본키)

emp

외래키(foreign key)

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	1980/12/17	800		20
7499	ALLEN	SALESMAN	7698	1981/02/20	1600	300	30
7521	WARD	SALESMAN	7698	1981/02/22	1250	500	30
7566	JONES	MANAGER	7839	1981/04/02	2975		20
7654	MARTIN	SALESMAN	7698	1981/09/28	1250	1400	30
7698	BLAKE	MANAGER	7839	1981/05/01	2850		30
7782	CLARK	MANAGER	7839	1981/06/09	2450		10
7788	SCOTT	ANALYST	7566	1987/04/19	3000		20
7839	KING	PRESIDENT		1981/11/17	5000		10

dept

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Primary Key(기본키)

student

외래키(foreign key)

STUDNO	NAME	ID	GRADE	JUMIN	BIRTHDAY	TEL	HEIGHT	WEIGHT	DEPTNO1	DEPTNO2	PROFNO
9411	서진수	75true	4	7510231901813	1975/10/23	055)381-2158	180	72	101	201	1001
9412	서재수	pooh94	4	7502241128467	1975/02/24	051)426-1700	172	64	102		2001
9413	이미경	angel000	4	7506152123648	1975/06/15	053)266-8947	168	52	103	203	3002
9414	김재수	gunmandu	4	7512251063421	1975/12/25	02)6255-9875	177	83	201		4001
9415	박동호	pincle1	4	7503031639826	1975/03/03	031)740-6388	182	70	202		4003
9511	김신영	bingo	3	7601232186327	1976/01/23	055)333-6328	164	48	101		1002
9512	신은경	jjang1	3	7604122298371	1976/04/12	051)418-9627	161	42	102	201	2002
9513	오나라	nara5	3	7609112118379	1976/09/11	051)724-9618	177	55	202		4003
9514	구유미	guyume	3	7601202378641	1976/01/20	055)296-3784	160	58	301	101	4007

department

DEPTNO	DNAME	PART	BUILD
101	컴퓨터공학과	100	정보관
102	멀티미디어공학과	100	멀티미디어관
103	소프트웨어공학과	100	소프트웨어관
201	전자공학과	200	전자제어관
202	기계공학과	200	기계실험관
203	화학공학과	200	화학실습관
301	문헌정보학과	300	인문관
100	컴퓨터정보학부	10	
200	메카트로닉스학부	10	
300	인문사회학부	20	
10	공과대학		
20	인문대학		

Primary Key(기본키)

student

외래키(foreign key)

STUDNO	NAME	ID	GRADE	JUMIN	BIRTHDAY	TEL	HEIGHT	WEIGHT	DEPTNO1	DEPTNO2	PROFNO
9411	서진수	75true	4	7510231901813	1975/10/23	055)381-2158	180	72	101	201	1001
9412	서재수	pooh94	4	7502241128467	1975/02/24	051)426-1700	172	64	102		2001
9413	이미경	angel000	4	7506152123648	1975/06/15	053)266-8947	168	52	103	203	3002
9414	김재수	gunmandu	4	7512251063421	1975/12/25	02)6255-9875	177	83	201		4001
9415	박동호	pincle1	4	7503031639826	1975/03/03	031)740-6388	182	70	202		4003
9511	김신영	bingo	3	7601232186327	1976/01/23	055)333-6328	164	48	101		1002
9512	신은경	jjang1	3	7604122298371	1976/04/12	051)418-9627	161	42	102	201	2002
9513	오나라	nara5	3	7609112118379	1976/09/11	051)724-9618	177	55	202		4003

professor

PROFNO	NAME	ID	POSITION	PAY	HIREDATE	BONUS	DEPTNO	EMAIL	HPAGE
1001	조인형	captain	정교수	550	1980/06/23	100	101	captain@abc.net	http://www.abc.net
1002	박승곤	sweety	조교수	380	1987/01/30	60	101	sweety@abc.net	http://www.abc.net
1003	송도권	powerman	전임강사	270	1998/03/22		101	pman@power.com	http://www.power.com
2001	양선희	lamb1	전임강사	250	2001/09/01		102	lamb1@hamail.net	
2002	김영조	number1	조교수	350	1985/11/30	80	102	number1@naver.com	http://num1.naver.com
2003	주승재	bluedragon	정교수	490	1982/04/29	90	102	bdragon@naver.com	
3001	김도형	angel1004	정교수	530	1981/10/23	110	103	angel1004@hanmir.com	
3002	나한열	naone10	조교수	330	1997/07/01	50	103	naone10@empal.com	

Primary Key(기본키)

컬럼 속성(무결성 제약조건)

- FOREIGN KEY

customerid varchar2(30) **references member(id)** not null,

```
컬럼명 [CONSTRAINTS "외래키 이름" ]
        REFERENCES 참조테이블명(참조 컬럼);
```

```
ALTER TABLE  테이블명
ADD CONSTRAINTS "외래키 이름"  FOREIGN KEY(컬럼)
REFERENCES 참조테이블명(참조 컬럼);
```

```
deptno  number CONSTRANTS fk_deptno
            REFERENCES  dept2(dcode);  --인라인

deptno number,
CONSTRANTS  fk_deptno
            FOREIGN KEY(deptno)
            REFERENCES  dept2(dcode);  --아웃라인

--테이블 생성 후
ALTER TABLE emp2
ADD CONSTRANTS  fk_deptno
            FOREIGN KEY(deptno)
            REFERENCES  dept2(dcode);
```

```
deptno  number REFERENCES  dept2(dcode);
```

컬럼 속성(무결성 제약조건)

- 외래 키 제약사항
  - 외래키를 만들기 전에 반드시 **부모 테이블이 먼저 생성되어 있어야** 함
  - **참조하는 부모 테이블의 컬럼은** 반드시 **기본 키** 또는 **unique키** 여야 함
  - 한 개 이상의 컬럼으로 외래 키 생성 가능(복합 외래키)
  - **자식 테이블에 존재하는 값을 부모 테이블에서 삭제할 수 없다**
    - insert 시 : 먼저 부모 테이블 데이터 입력 => 자식 테이블 데이터 입력
    - delete 시 : 먼저 자식 테이블 데이터 삭제(수정) => 부모 테이블 데이터 삭제(수정)
      - on delete cascade 옵션을 주면, 자식 테이블에 데이터가 있어도 부모 테이블에서 삭제 가능하며, 자식 테이블의 데이터도 같이 삭제해버림

댓글 테이블에서
bdNo number references board(no) **on delete cascade** not null --원본 게시판 글번호

컬럼 속성(무결성 제약조건)

- CHECK
  - 입력되는 값을 체크하여 일정한 조건에 해당되는 값만 입력될 수 있게 하는 제약 조건
  - 예) 성별(gender) 컬럼 => 남자, 여자만 입력되고 다른 값은 입력될 수 없도록

```
gender  VARCHAR2(10) NOT NULL
      CHECK ( gender IN ('남자', '여자'))
```

```
CREATE TABLE check_test (
   gender  VARCHAR2(10) NOT NULL
        CONSTRAINT check_gender CHECK ( gender IN ('남자', '여자')) --인라인
);

INSERT INTO check_test VALUES('사람'); --에러
```

```
ORA-02290: 체크 제약조건(SCOTT.CHECK_GENDER)이 위배되었습니다
```

```
gender  VARCHAR2(10) NOT NULL ,
      CONSTRAINT check_gender CHECK ( gender IN ('남자', '여자'))); --아웃라인
```

컬럼 속성(무결성 제약조건)

- DEFAULT
  - 기본값
  - 컬럼에 특정 값을 디폴트 값으로 설정하면 테이블에 데이터를 입력할 때 해당 컬럼에 값을 입력하지 않을 경우 디폴트로 설정한 값이 자동으로 입력됨
  - 컬럼 타입 다음에 'default 디폴트값' 을 명시
  - 반드시 데이터 타입 다음에, null이나 not null 앞에 위치시켜야 함

```
CREATE TABLE default_test (
   gender  VARCHAR2(10) default '남자'  NOT NULL,
   hiredate  date  default sysdate,
   name      varchar2(10)
);

INSERT INTO default_test (name) VALUES ('홍길동');

select * from default_test;
```

예) 성적 테이블의 국어, 영어, 수학 점수 컬럼 : DEFAULT 0
판매 테이블의 판매일자 컬럼 : sysdate

예제 – 무결성 제약조건

```
--부서 테이블(부모)
create table depart
(
    dcode       char(2) primary key,              --부서 코드
    dname       varchar2(50)   not null,          --부서명
    area        varchar2(100)   null              --지역
);

--사원테이블(자식)
create table employee
(
    no          number   primary key,                          --사원번호
    name        varchar2(20)  not null,                       --이름
    pay         number check(pay>=0)    not null,             --급여
    hiredate    date    default  sysdate   null,              --고용일
    email       varchar2(50) unique null,                     --이메일
    deptcd      char(2)
        constraints deptcd_fk references depart(dcode)    --부서코드
);
```

예제 – 무결성 제약조건

```
insert into depart
values('A1','영업부','서울');
insert into depart
values('B1','인사부','인천');
insert into depart
values('C1','총무부','부산');

insert into employee(no, name, pay, email, deptcd)
values(100, '홍길동', 3000000,'hong@nate.com','D1');  --에러

insert into employee(no, name, pay, email, deptcd)
values(100, '홍길동', 3000000,'hong@nate.com','A1');
insert into employee(no, name, pay, email, deptcd)
values(101, '김길동', 4000000,'kim@naver.com','B1');

delete from depart
where dcode='A1';  --에러
delete from depart
where dcode='C1';
commit;
```

예제 – 무결성 제약조건

```
--사원가족 테이블(자식)
create table family
(
    no          number references employee(no) on delete cascade,  --사원번호
    name        varchar2(20) ,  --가족이름
    relation     varchar2(20)  --관계
);

insert into family
values(100, '홍아빠','부');
insert into family
values(100, '홍엄마','모');

insert into family
values(101, '김형','형');
insert into family
values(101, '김동생','제');

delete from employee
where no=101;  --부모 테이블의 데이터를 삭제하면 자식 데이터도 삭제
```

예제2

```
--상품 테이블
create table pdInfo(
    no              number          not null    primary key,
    pdcode          char(3)         not null    unique,
    pdname          varchar2(100)   not null,
    company         varchar2(50)    null,
    price           number          default 0   check(price>=0) null,
    regdate         date            default  sysdate null
);

--장바구니 테이블
create table cart
(
   cartno    number primary key,
   pdcd      char(3) not null  --외래키
       constraints pdcd_fk references pdInfo(pdcode),
   qty       number check(qty >=0 and qty <=1000) null,
   userid    varchar(20) not null
);
```

예제2

```
INSERT into pdInfo(no, pdcode, pdname, price)
values (1, 'h01', '캐논디카',370000);
INSERT  into pdInfo(no, pdcode, pdname)
values (2, 'h02', '캐논디카2');
INSERT  into pdInfo(no, pdcode, pdname, regdate)
values (3, 'h03', '캐논디카3', default);

INSERT  into pdInfo(no, pdcode, pdname, price)  --에러
values (4, 'h04', '캐논디카4',-100);

select * from pdinfo;

INSERT  into cart(cartno, pdcd, qty, userid)
values (1, 'h01', 100, 'hong');
INSERT  into cart(cartno, pdcd, qty, userid)  --에러
values (2, 'h02', 2000, 'hong');

select * from cart;
```

예제-아웃라인 제약조건

```
--아웃라인 제약조건 주기
drop table pd2 cascade constraints;
create table pd2
(
    no          number,
    pdcode    char(3) not null,
    pdname  varchar2(200) not null,
    price       number  default 0,
    company varchar2(100) null,
    regdate     date      default sysdate,

    constraints no_pk           primary key(no),
    constraints pdcode_uk     unique(pdcode),
    constraint price_ck         check(price>=0)
);
```

```
drop table cart2;
create table cart2
(
   cartno    number primary key,
   pdcd      char(3) not null,
   qty       number,
   userid    varchar(20) not null,

   constraints pdcd_fk2 foreign key(pdcd)
         references pd2(pdcode),
    constraint qty_ck check(qty>=0 and     qty<=1000)
);
```

테이블 생성 후 제약조건 추가

- 테이블 생성 후 **alter** 문을 이용하여 제약조건 추가 가능

```
ALTER TABLE check_test
MODIFY gender DEFAULT '남자';

ALTER TABLE check_test
ADD hiredate DATE DEFAULT SYSDATE;

ALTER TABLE check_test
ADD name VARCHAR2(10) NULL;

INSERT INTO check_test (name) VALUES ('홍길동');

SELECT * FROM check_test;
```

스키마 이름.테이블 이름
hr.emp =>hr 스키마에 있는 emp 테이블, hr 사용자로 로그인해 있기 때문에 hr 생략

스키마(Schema) - 임의의 사용자가 생성한 모든 데이터베이스 객체들을 말하며, 스키마 이름은 그 사용자의 이름과 같다.

테이블 복사 및 제거

- 테이블의 복사

```
CREATE TABLE 신규테이블명
AS
   SELECT  컬럼 리스트  from 원본 테이블명;
```

```
CREATE TABLE emp_copy1
AS
SELECT * FROM  emp;

SELECT * FROM emp_copy1;
```

- 제약 사항
  - 원본 테이블의 인덱스는 복사하지 않음
  - 원본 테이블의 여러 가지 제약사항 중 null, not null 속성만 복사함
  - long 타입 컬럼은 복사 불가

테이블 복사 및 제거

- 테이블의 제거

DROP TABLE [스키마명 .] 테이블명 **[CASCADE CONSTRAINTS];**

  - 테이블을 생성한 소유자만 해당 테이블을 제거할 수 있음
  - sys, system, drop any table 권한을 가진 사용자들은 모든 테이블들을 제거할 수 있음
  - 테이블을 제거하면 참조 제약 조건(외래 키)을 제외하고 해당 테이블에 연관된 제약조건, 인덱스, 트리거들도 자동으로 삭제됨
  - 참조 제약조건들까지 자동으로 삭제시키려면 cascade constraints 옵션을 붙여야 함

- 각종 참조 시스템 뷰
  - USER_TABLES : 해당 사용자가 생성한 테이블 내역
  - USER_CONSTRAINTS
  - USER_INDEXES
  - USER_OBJECTS, …

alter

- alter 명령어 – 만들어져 있는 오브젝트를 변경하는 명령어
  - 테이블 – 칼럼 추가, 칼럼 삭제, 칼럼 이름이나 테이블 이름 변경 등
  - 부하가 많이 걸리는 명령어이므로 주의.
- [1] 새로운 칼럼 추가하기

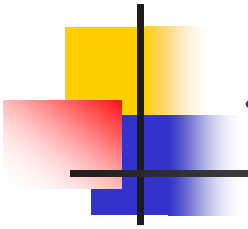
```
create table dept6
as
select dcode, dname from dept2
where dcode in(1000, 1001, 1002);

select * from dept6;

 --장소명을 가지는 LOC 칼럼 추가
 alter table dept6
 add(loc varchar2(10));  --추가되는 칼럼에는 모두 기본값으로 null값이 입력됨

 alter table dept6
add  regdate date default sysdate; --기본값으로 현재일자가 입력됨
```

--[2] 칼럼의 데이터 크기 변경하기
```
desc dept6;

alter table dept6
modify(dcode varchar2(10) primary key);

alter table dept6
modify loc varchar2(10) default '경기' ;

insert into dept6(dcode, dname)
values(1003, '영업팀');
```

--[3] 테이블의 칼럼 이름 변경하기
```
--loc 칼럼을 area 로 변경
alter table dept6
rename column loc to area;

--테이블 이름 변경 -> rename 명령 사용
--dept2 테이블을 dept7로 이름 변경하기
rename dept6 to dept7;

select * from dept7;
```

--[4] 칼럼 삭제하기
```
alter table dept7
drop column regdate;
```

```
--참조키로 설정되어 있는 부모테이블의 칼럼을 삭제
하려 할 경우 에러가 발생하는데
 --다음 방법으로 지우면 됨
alter table depart
drop column dcode cascade constraints;
```



실습1

- 회원 테이블 만들기-member2

no - 번호, 기본키
userid 아이디, unique, 반드시 값 입력되도록
name 이름, 반드시 값 입력되도록
pwd 비밀번호 , 반드시 값 입력되도록
email 이메일
hp 휴대폰번호
zipcode 우편번호
address 주소(시도, 구군, 동)
addressDetail 상세주소
regdate 가입일, 기본값:현재일자
mileage 마일리지, 기본값 :0, 0~1000000 사이의 값만 들어가도록

실습2 – 테이블 분리

- 우편번호 테이블 만들기 – zipcode

zipcode 우편번호
sido 시도
gugun 구군
dong 동
bunji 번지
seq –번호, 기본키

- 회원테이블 만들기 – member1

no – 번호, 기본키
userid 아이디, unique, 반드시 값 입력되도록
name 이름, 반드시 값 입력되도록
pwd 비밀번호 , 반드시 값 입력되도록
email 이메일
hp 휴대폰번호
zipcodeno 우편번호 테이블 번호(seq)와 연결–외래키
addressDetail 상세주소
regdate 가입일, 기본값:현재일자
mileage 마일리지, 기본값 :0, 0~1000000 사이의 값만 들어가도록

과제1

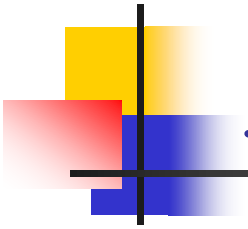
- 게시판(board), 한줄 답변(comments) 테이블 만들기

- 1. 게시판에 글쓰기 – insert
- 2. 게시판의 1번 글에 대해 한줄 답변 2개 쓰기
- 3. 한줄 답변 1개 삭제 – delete
- 4. 게시판의 글 수정 – update
- 5. 게시판의 글 삭제 – 한줄 답변도 같이 삭제되도록
- 6. 게시판 목록 – select
  - 번호 내림차순으로 전체 조회
- 7. 게시판 목록에서 선택한 글 보기
  - 선택한 글만 조회 – select
  - 선택한 글의 조회수 증가 – update
  - 선택한 글의 한줄 답변 모두 조회 – select
- 8. 제목, 내용, 작성자로 검색 – select

과제2

- **상품 정보 테이블 만들기 (product)**

  상품코드번호　기본키
  카테고리
  상품이름
  상품가격　　기본값 0,　0 이상의 값만 입력되도록
  제조회사
  이미지
  요약설명
  상세설명
  등록일　　기본값　현재일자



과제3

- 장바구니 테이블 만들기(cart)
  - 회원아이디　회원테이블(member1)의 userid만 들어가도록
  - 상품코드　상품 정보 테이블(product)의 상품코드 번호만 입력되도록
  - 수량　　기본값 0
  - 생성일　　기본값 현재일자

과제3– 장바구니

장바구니 테이블1

회원아이디	상품코드	상품명	가격	수량
Hong	A01	청바지	90000	1
Hong	B02	티셔츠	98000	1
Park	A02	스커트	110000	2
Kim	A01	청바지	90000	2
Lee	A03	원피스	120000	1

과제3 – 장바구니 테이블 분리

장바구니 테이블(자식)

회원아이디	상품코드	수량
Hong	A01	1
Hong	B02	1
Park	A02	2
Kim	A01	2
Lee	A03	1

FK(외래키)

상품 테이블(부모)

상품코드	상품명
A01	청바지
A02	스커트
A03	원피스
A04	트렌치 코트
B02	스커트

PK(기본키)

- 제약조건 추가

alter table 테이블명
add constraint 제약조건명 ..

예) alter table dept_exam1
add constraint pk_dept_dcode primary key(dcode);

alter table emp_exam1
add constraint chk_emp_pay_comm_pct check(pay > 0 and comm_pct > 0);

- 제약조건의 제거

alter table 테이블명
drop constraint 제약조건명;

예) alter table dept_exam1
drop constraint pk_dept_dcode;

  - NOT NULL 제약만 아래와 같이 제거할 수도 있다.

alter table 테이블명
modify 컬럼명 null;

dname varchar2(30) not null, -- not null 제약조건의 이름을 지정하지 않고 만들면
-- 제약조건의 이름은 자동으로 SYS_로 시작함
loc varchar2(40) constraint NN_dept_loc not null

- index 조회하기

```
select * from user_indexes;
select * from user_ind_columns;
```

- 컬럼에 주어진 default 값 조회

```
select column_name, data_default
from user_tab_columns
where table_name = 'emp_exam1';
```

- 컬럼에 주어진 default 값 변경

```
alter table emp_exam1
modify deptno default 'A01';
```

- 컬럼의 데이터타입 변경

```
--VARCHAR2(30) => VARCHAR2(40)
alter table emp_exam1
modify ename varchar2(40);
```

- 제약조건 이름 변경하기

```
alter table emp_exam1
rename constraint pk_dept_dcode to pk_dept_exam1_dcode;
```

- 테이블 복사를 하면 제약조건 중 NOT NULL 제약만 복사가 되고 나머지 제약조건은 복사가 안됨

```
create table dept_copy  as
select * from dept  where 0=1;
desc dept_copy
```