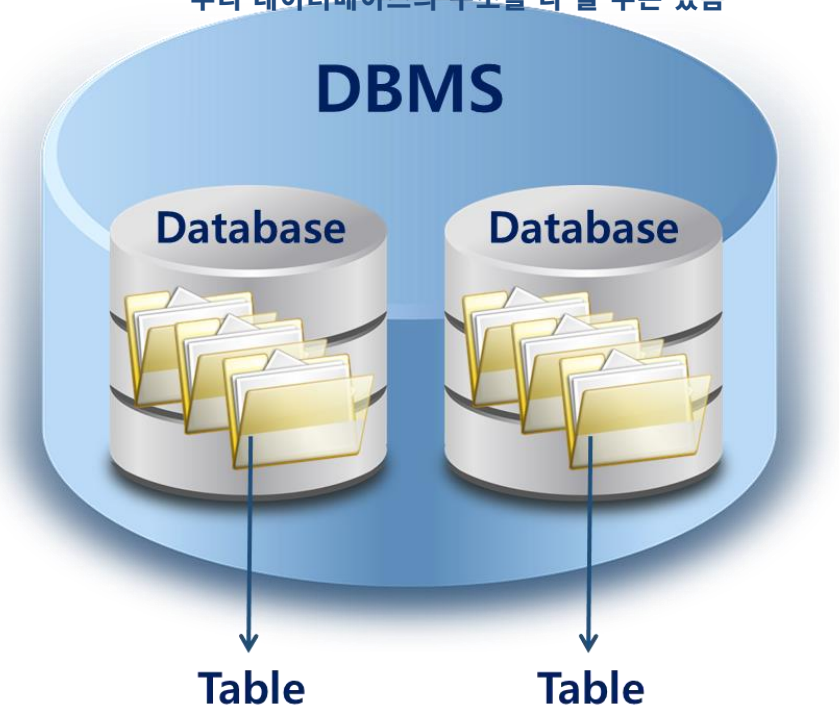
데이터베이스 구조

▶ 데이터베이스 구조

Oracle DBMS 의 구조는 MySQL이나 MSSQL 과 조금 다름

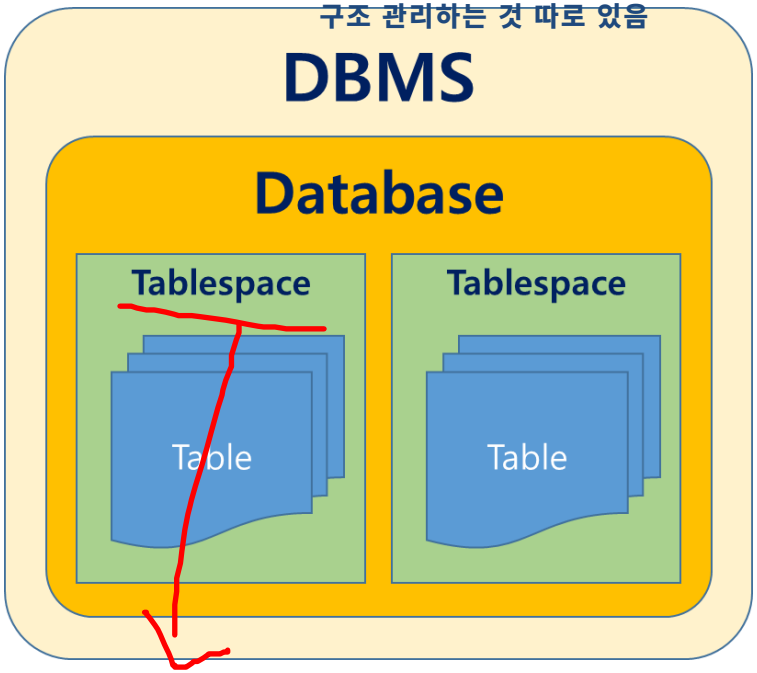
✓ MySQL, MSSQL

일반 사용자 계정으로 들어가도
우리 데이터베이스의 구조를 다 볼 수는 있음



✓ Oracle

일단 데이터를 쓰려면 사용자 계정이 있어야 함
사용자 기반
계정별로 쓸 수 있는 것이 정해져있다
내가 만든 내 테이블에 대해서는
내가 관리할 수 있도록
사용자 계정으로 들어가면 딱 내 것만 볼 수있음
구조 관리하는 것 따로 있음



파일철에 한번 더 꽂아놓는다고 생각하면 된다

우리가 제어 할 수 있는 것:
Tablespace(Table들을 넣어놓는 폴더), Table

데이터베이스 구조

✓ 데이터베이스

체계화된 데이터의 모임

✓ 테이블 스페이스

테이블이 저장되는 공간

기본적으로 생성되어 있음 (SYSTEM , UNDOTBS, TEMP 등..)

사용자에 의해 추가적으로 여러 개의 테이블스페이스가 생성될 수 있음

✓ 오브젝트(Object)

데이터를 관리하기 위해 생성하는 모든 것

table, index, view, sequence 등

table : **가장 기본이 되는 오브젝트**

▶ 데이터베이스 구조

KH정보교육원

✓ 세그먼트(Segment)

실제 저장공간을 가지고 있는 오브젝트,
구조만 있는 오브젝트
내가 표 그려서 입력하는 대로 저장해주는 오브젝트들이 있고
저장한걸 참조만 하는 오브젝트가 있음
저장공간 가지고 있는 놈 => 세그먼트

Object 중에서 저장공간을 가지고 있는 것

✓ 익스텐드(Extent)

데이터를 저장하는 논리적 단위

조금 더 많이 묶어서 한번에 처리하기 위해
ex) 100글자 가져온다 하면 2글자를 50번 가져와야하기 때문에 번거로움

일정한 수의 연속된 Block들로 구성

I/O를 할 때 속도를 높이기 위해 도입된 개념

✓ 데이터 블록(Block)

오라클에서 사용하는 데이터처리의 최소의 논리단위

I/O의 최소 단위 (I/O -> Input, Output) 로 하나 또는 그 이상의 OS

데이터를 저장하고 꺼내 쓰기 위한 최소 단위
Block으로 이루어짐

테이블 구조

Table

Columns Name

이름	나이	전화번호
홍길동	30	01012341234
김길동	25	01011112222
이길동	34	01043219876

연관된 데이터들
Rows
= Record
= Tuple
= 행

Columns
= Fields
= Attribute
= 열

연관된 하나의 속성을
가진 데이터들

Field

▶ 데이터베이스 구조

KH정보교육원

✓ Table (= Relation)

데이터의 세부 목적에 맞게 구조적 목록으로 묶어 놓은 영역

행(row)과 열(column)으로 구성되는 가장 기본적인 데이터베이스 객체로 데이터 배이스 내에서 모든 데이터는 테이블을 통해서 저장됨

✓ Columns (= Fields = Attribute = 열)

연관된 하나의 속성을 가진 데이터들

특정한 단순 자료 형의 일련의 데이터 값과 테이블에서의 각 열

✓ Rows (= Record = Tuple = 행)

연관된 데이터

테이블에서 한 객체의 대한 전체 정보

✓ Field

딱 데이터 하나

Column 의 대용으로 동일한 의미로 사용되지만 한 행과 한 열의 교차 지점에 있는 단일 값을 특별히 지칭하는 용어

SQL

데이터베이스에서 사용하는 명령어,언어

SQL(Structured Query Language)

데이터베이스에게 명령을 내리기 위한 명령어

구조화된 질의 언어

우리가 명령어를 통해서 데이터베이스한테 물어보면
데이터베이스는 처리해서 그것을 답해줌

DMBS를 관리하고 제어하기 위해 사용하는 인터페이스 또는 언어

데이터베이스로부터 정보를 얻거나 갱신하기 위한 표준 대화식 프로그래밍 언어

DBMS에 따라 사용되는 SQL 문법이 다름

▶ SQL 기본 문법

KH정보교육원

✓ 메타문자

	MS-SQL	MY-SQL	ORACLE
한줄 주석	--	# , --	--
범위 주석	/* */	/* */	/* */
문자열 지정	‘	‘, “	‘

싱글 쿼터

✓ 연산자

	MS-SQL	MY-SQL	ORACLE
산술 연산자	+, –, *, /, %, …	+, –, *, /, div	+, –, *, /
비교 연산자	=, >, <, >=, <=, <>, !=, !>, !<	=, >, <, >=, <=, <> , <=>, !=	=, >, <, >=, <=, <>
문자열 비교	like, not like	like, not like	like, not like
논리 연산자	and, or, not	and(&&), or(||), not(!)	and, or, not
범위 연산자	between A and B not between A and B	between A and B Not between A and B	between A and B Not between A and B
값 출력 연산자	in, not in	in, not in	in, not in
Null 검색	is null, is not null	is null, is not null	is null, is not null

같다 / 같지않다

A부터 B까지

A부터B까지 말고

▶ SQL 기본 문법

기본 명령어들이 크게 세가지로 , 좀더 세부적으로는 다섯가지로 나눠진다.
SELECT는 DML에 속하지만 DQL(데이터 쿼리어)로 따로 구분하기도 한다.
COMMIT, ROLLBACK은 트렌젝션을 수행하기 위한 언어 DTL로 구분되기도 한다.

① ✓ 데이터 정의어(DDL, Data Definition Language)

틀 만들기

DBMS안에서 테이블, 테이블스페이스 그 외 오브젝트 들을 생성하고 삭제하고 수정할 수 있는 것

데이터베이스의 구조를 정의하거나 변경, 삭제하기 위해 사용하는 언어

주로 DB관리자 또는 설계자가 사용

실제 데이터를 쓰고 입력하는게 아니라 그 데이터를 저장할 공간을 만드는 것
데이터를 저장할 공간 = 객체

CREATE(개체 생성), ALTER(개체 수정), DROP(개체 삭제), TRUNCATE(개체 초기화)

트렁케이트

② ✓ 데이터 조작어(DML, Data Manipulation Language)

(실제 데이터를)

data를 조작하기 위해 사용하는 언어

data의 삽입, 수정, 삭제, 조회 등의 동작을 제어

data를 이용하려는 사용자와 시스템 간의 인터페이스를 직접적으로 제공하는 언어

가장 많이 사용됨(공격 시에도 가장 많이 사용)

data 사용 할 때 쓰는 언어

INSERT(데이터 생성), UPDATE(데이터 수정), DELETE(데이터 삭제), SELECT(데이터 조회)

DQL

③ ✓ 데이터 제어어(DCL, Data Control Language)

DBMS 자체를 제어하기 위한 언어

DB에 대한 보안, 무결성, 복구등 DBMS를 제어하기 위한 언어

GRANT(권한할당), REVOKE(권한해제), COMMIT(실행), ROLLBACK(복구)

DTL

무결성을 보장하기 위한 트렌젝션을 수행할 수 있는 명령어들

DDL
(CREATE)

▶ DDL(Data Definition Language)

데이터 정의 언어로 객체(OBJECT)를 만들고(CREATE), 수정하고(ALTER), 삭제(DROP)하는 구문을 말함

✓ 오라클 객체 종류

테이블(TABLE), 뷰(VIEW), 시퀀스(SEQUENCE), 인덱스(INDEX), 패키지(PACKAGE), 프로시저(PROCEDURE), 함수(FUNCTION), 트리거(TRIGGER), 동의어(SYNONYM), 사용자(USER)

▶ CREATE

데이터베이스의 객체를 생성하는 구문
주로 테이블을 생성할 때 사용하며, 그외에도 다양한 객체들을 생성할 때 사용 함
생성된 구문은 DROP구문을 통해서 제거할 수 있음

테이블이나 인덱스, 뷰 등 데이터베이스 객체를 생성하는 구문

✓ 표현식

뭘 만들건지에 따라서 내용이 조금 달라짐

```
CREATE TABLE 테이블명(컬럼명 자료형(크기),
             컬럼명 자료형(크기),
             ...);
```

뭘 만들건지 / 이름

```
CREATE TABLE MEMBER(
        MEMBER_ID VARCHAR2(20),
        MEMBER_PWD VARCHAR2(20),
        MEMBER_NAME VARCHAR2(20)
);
```

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	MEMBER_ID	VARCHAR2(20 BYTE)	Yes	(null)	1	(null)
2	MEMBER_PWD	VARCHAR2(20 BYTE)	Yes	(null)	2	(null)
3	MEMBER_NAME	VARCHAR2(30 BYTE)	Yes	(null)	3	(null)

▶ 오라클 데이터형

VARCHAR2 같은경우 내 다음 가리킬 공간 조금 만들어놓음
-> 공간할당해야하는 작업을 해야하고 한번 더 주소를 따라가야하기 때문에
성능적인 면에서는 CHAR가 VARCHAR2보다 더 좋다. 한글 3byte, 영어 1byte

데이터형	설명
CHAR(크기) byte	고정길이 문자 데이터 (크기)만큼의 문자열을 저장함 CHAR(20)이라고 적으면 5byte만 적어도 20 공간 늘어남
VARCHAR2(크기) 최대크기	가변길이 문자 데이터(최대 2,000 Byte)
NUMBER	숫자 데이터(최대 40자리)
NUMBER(길이)	숫자 데이터로, 길이 지정 가능 (최대 38자리)
DATE 날짜 저장하는 데이터 타입	날짜 데이터(BC 4712년 1월 1일 ~ AD 4712년 12월 31일)
LONG	가변 길이 문자형 데이터(최대 2GB)
LOB 로비	2GB까지의 가변길이 바이너리 데이터 저장 가능 (이미지, 실행파일 등 저장 가능)
ROWID	DB에 저장되지 않는 행을 식별할 수 있는 고유 값
BFILE	대용량의 바이너리 데이터 저장 가능(최대 4GB)
TIMESTAMP	DATE형의 확장된 형태 0과1의 형태로
INTERVAL YEAR TO MONTH	년과 월을 이용하여 기간 저장
INTERVAL DAY TO SECOND	일, 시, 분, 초를 이용하여 기간 저장

입력할 데이터의 길이가 비슷비슷한 경우 효율적
예) 한국사람 이름, 휴대폰 번호

VARCHAR2(20)이라고 적고 5byte만 적으면 5byte만큼의 크기만 만드는데 나중에 더 입력하면 늘려짐
하지만 20이라고 적었으니까 20까지만

▶ 컬럼 주석

KH정보교육원

테이블의 컬럼에 주석을 다는 구문

✓ 표현식

COMMENT ON COLUMN 테이블명.컬럼명 **IS** '주석 내용 ' ;

COMMENT ON COLUMN MEMBER.MEMBER_ID **IS** '회원아이디';
COMMENT ON COLUMN MEMBER.MEMBER_PWD **IS** '비밀번호';
COMMENT ON COLUMN MEMBER.MEMBER_NAME **IS** '회원이름';

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	MEMBER_ID	VARCHAR2(20 BYTE)	Yes	(null)	1	회원아이디
2	MEMBER_PWD	VARCHAR2(20 BYTE)	Yes	(null)	2	비밀번호
3	MEMBER_NAME	VARCHAR2(30 BYTE)	Yes	(null)	3	회원이름

이 컬럼에 대한 설명
이 컬럼이 뭐하는 컬럼인지
설명을 적어놓는 부분

제약 조건(CONSTRAINTS)

KH정보교육원

데이터를 저장할 때
특정 조건을 강제로 정하는 것

사용자가 원하는 조건의 데이터만 유지하기 위해서 특정 컬럼에 설정하는 제약

각각의 컬럼마다 별도의 조건 제약 가능

데이터 무결성을 지키기 위해 제한된 조건

입력 데이터에 문제가 없는지에 대한 검사와 데이터 수정/삭제 가능 여부 검사 등을 위해 사용

FOREIGN KEY : 테이블을 두개 이상 사용 할 때 사용하는 키
서로 전혀 별개의 테이블 일 수 있지만 연관된 테이블일 수 있다
회원 목록 테이블 ,주문의 목록이 들어가는 테이블
누가 주문을 했는지 회원 정보 들어가야하고, 회원 테이블에도 회원 정보
주문목록에 있는 아이디는 회원목록에 들어있는 아이디만 있을 수 있다 ->
이렇게 제약을 거는 것이 외래키

PRIMARY KEY:
각각의 로우,레코드들이 들어가는데
기준이 되는 컬럼을 하나 정함 -> 프라이머리키를 가지는 컬럼
다른 레코드들과 유일하게 구분될 수 있는
기준이 되는 컬럼을 설정하는 것, 한 테이블에 무조건 하나만 가능

제약 조건	설명
NOT NULL	데이터에 NULL을 허용하지 않음
UNIQUE	중복된 값을 허용하지 않음
PRIMARY KEY	NULL과 중복 값을 허용하지 않음(컬럼의 고유 식별자로 사용하기 위해)
FOREIGN KEY 외래키	두 테이블의 데이터 간 연결을 설정하고 강제 적용하여 외래 키 테이블에 저장될 수 있는 데이터를 제어함
CHECK	저장 가능한 데이터 값의 범위나 조건을 지정하여 설정한 값만 허용

해당 컬럼에 들어갈 수 있는 값을 내가 직접 지정하는 것
해당 컬럼에 들어갈 데이터의 목록을 미리 만들어 놓는 것
ex) 성별은 남,여만 들어갈 수 있게

▶ 제약 조건(CONSTRAINTS)

KH정보교육원

✓ 제약 조건 확인

DESC USER_CONSTRAINTS;

```
이름                 널       유형
---------------- -------- -------------
OWNER                       VARCHAR2(120)
CONSTRAINT_NAME  NOT NULL   VARCHAR2(30)
CONSTRAINT_TYPE             VARCHAR2(1)
TABLE_NAME       NOT NULL   VARCHAR2(30)
SEARCH_CONDITION            LONG
R_OWNER                     VARCHAR2(120)
R_CONSTRAINT_NAME           VARCHAR2(30)
DELETE_RULE                 VARCHAR2(9)
STATUS                      VARCHAR2(8)
DEFERRABLE                  VARCHAR2(14)
DEFERRED                    VARCHAR2(9)
VALIDATED                   VARCHAR2(13)
GENERATED                   VARCHAR2(14)
BAD                         VARCHAR2(3)
RELY                        VARCHAR2(4)
LAST_CHANGE                 DATE
INDEX_OWNER                 VARCHAR2(30)
INDEX_NAME                  VARCHAR2(30)
INVALID                     VARCHAR2(7)
VIEW_RELATED                VARCHAR2(14)
```

DESC USER_CONS_COLUMNS;

```
이름               널       유형
-------------- -------- --------------
OWNER          NOT NULL VARCHAR2(30)
CONSTRAINT_NAME NOT NULL VARCHAR2(30)
TABLE_NAME     NOT NULL VARCHAR2(30)
COLUMN_NAME             VARCHAR2(4000)
POSITION                NUMBER
```

▶ 제약 조건(CONSTRAINTS)

✓ NOT NULL

해당 컬럼에 반드시 값이 기록되어야 하는 경우 사용

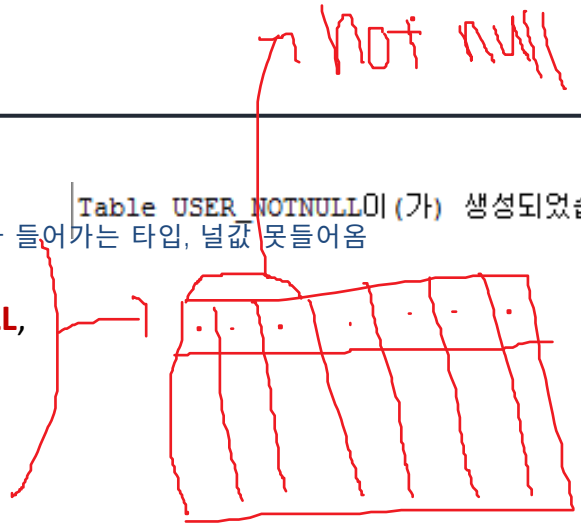
특정 컬럼에 값을 저장/수정할 때는 NULL값을 허용하지 않도록

컬럼 레벨에서 제한

잘못해서 NULL이 들어오지 않도록

▶ 제약 조건(CONSTRAINTS)

✓ NOT NULL 예시



테이블 이름

```
CREATE TABLE USER_NOTNULL(
        USER_NO NUMBER NOT NULL,
        USER_ID VARCHAR2(20) NOT NULL,
        USER_PWD VARCHAR2(30) NOT NULL,
        USER_NAME VARCHAR2(30),
        GENDER VARCHAR2(10),
        PHONE VARCHAR2(30),
        EMAIL VARCHAR2(50)
);
```

숫자가 들어가는 타입, 널값 못들어옴

```
Table USER_NOTNULL이(가) 생성되었습니다.
```

```
1 행 이(가) 삽입되었습니다.
```

```
INSERT INTO USER_NOTNULL VALUES(1, 'user01', 'pass01', '홍길동', '남', '010-1234-5678',
'hong123@kh.or.kr');
INSERT INTO USER_NOTNULL VALUES(2, NULL, NULL, NULL, NULL, '010-1234-5678',
'hong123@kh.or.kr');
```

```
오류 보고 -
SQL 오류: ORA-01400: cannot insert NULL into ("EMPLOYEE"."USER_NOTNULL"."USER_ID")
01400. 00000 -  "cannot insert NULL into (%s)"
*Cause:    An attempt was made to insert NULL into previously listed objects.
*Action:   These objects cannot accept NULL values.
```

	USER_NO	USER_ID	USER_PWD	USER_NAME	GENDER	PHONE	EMAIL
1	1	user01	pass01	홍길동	남	010-1234-5678	hong123@kh.or.kr

* NOT NULL 제약조건이 설정된 컬럼에 NULL값이 입력되면, 행 자체를 삽입하지 않음

▶ 제약 조건(CONSTRAINTS)

KH정보교육원

✓ UNIQUE

컬럼 입력 값에 대해 중복을 제한하는 제약조건으로

컬럼 레벨과 테이블 레벨에 설정 가능 설정 방법

▶ 제약 조건(CONSTRAINTS)

KH정보교육원

✓ UNIQUE 예시1

```
CREATE TABLE USER_UNIQUE(
        USER_NO NUMBER,
        USER_ID VARCHAR2(20) UNIQUE,
        USER_PWD VARCHAR2(30) NOT NULL,
        USER_NAME VARCHAR2(30),
        GENDER VARCHAR2(10),
        PHONE VARCHAR2(30),
        EMAIL VARCHAR2(50)
);
```

컬럼을 설정할 때 적어준다 -> 컬럼레벨

```
Table USER_UNIQUE이(가) 생성되었습니다.
```

```
INSERT INTO USER_UNIQUE VALUES(1, 'user01', 'pass01', '홍길동', '남', '010-1234-5678',
'hong123@kh.or.kr');
INSERT INTO USER_UNIQUE VALUES(1, 'user01', 'pass01', NULL, NULL, '010-1234-5678',
'hong123@kh.or.kr');
```

```
1 행 이(가) 삽입되었습니다.
```

```
오류 보고 -
SQL 오류: ORA-00001: unique constraint (EMPLOYEE.SYS_C007182) violated
00001. 00000 -  "unique constraint (%s.%s) violated"
*Cause:    An UPDATE or INSERT statement attempted to insert a duplicate key.
           For Trusted Oracle configured in DBMS MAC mode, you may see
           this message if a duplicate entry exists at a different level.
*Action:   Either remove the unique restriction or do not insert the key.
```

▶ 제약 조건(CONSTRAINTS)

✓ UNIQUE 예시2

```
CREATE TABLE USER_UNIQUE2(
        USER_NO NUMBER,
        USER_ID VARCHAR2(20),
        USER_PWD VARCHAR2(30) NOT NULL,
        USER_NAME VARCHAR2(30),
        GENDER VARCHAR2(10),
        PHONE VARCHAR2(30),
        EMAIL VARCHAR2(50),
        UNIQUE (USER_ID) --테이블 레벨 다 적어놓고 마지막에 설정
);
```

```
Table USER_UNIQUE2이(가) 생성되었습니다.
```

```
SQL 오류: ORA-00001: unique constraint (EMPLOYEE.SYS_C007184) violated
00001. 00000 -  "unique constraint (%s.%s) violated"
*Cause:    An UPDATE or INSERT statement attempted to insert a duplicate key.
           For Trusted Oracle configured in DBMS MAC mode, you may see
           this message if a duplicate entry exists at a different level.
*Action:   Either remove the unique restriction or do not insert the key.
```

```
INSERT INTO USER_UNIQUE2 VALUES(1, 'user01', 'pass01', '홍길동', '남', '010-1234-5678',
'hong123@kh.or.kr');
```

```
1 행 이(가) 삽입되었습니다.
```

```
INSERT INTO USER_UNIQUE2 VALUES(1, 'user01', 'pass01', NULL, NULL, '010-1234-5678',
'hong123@kh.or.kr');
```

```
INSERT INTO USER_UNIQUE2 VALUES(1, NULL, 'pass01', '홍길동', '남', '010-1234-5678',
'hong123@kh.or.kr');
```

```
1 행 이(가) 삽입되었습니다.
```

```
INSERT INTO USER_UNIQUE2 VALUES(1, NULL, 'pass01', '홍길동', '남', '010-1234-5678',
'hong123@kh.or.kr');
```

```
1 행 이(가) 삽입되었습니다.
```

	USER_NO	USER_ID	USER_PWD	USER_NAME	GENDER	PHONE	EMAIL
1	1	user01	pass01	홍길동	남	010-1234-5678	hong123@kh.or.kr
2	1	(null)	pass01	홍길동	남	010-1234-5678	hong123@kh.or.kr
3	1	(null)	pass01	홍길동	남	010-1234-5678	hong123@kh.or.kr

* 중복 값이 있는 경우 UNIQUE 제약 조건에 의해 행이 삽입되지 않음(NULL 값 중복은 가능)

▶ 제약 조건(CONSTRAINTS)

KH정보교육원

✓ UNIQUE 예시3

```
CREATE TABLE USER_UNIQUE3(
        USER_NO NUMBER,
        USER_ID VARCHAR2(20),
        USER_PWD VARCHAR2(30) NOT NULL,
        USER_NAME VARCHAR2(30),
        GENDER VARCHAR2(10),
        PHONE VARCHAR2(30),
        EMAIL VARCHAR2(50),
        UNIQUE (USER_NO, USER_ID) --두 컬럼을 묶어 한 UNIQUE제약조건 설정
);
INSERT INTO USER_UNIQUE3 VALUES(1, 'user01', 'pass01', '홍길동', '남', '010-1234-5678', 'hong123@kh.or.kr');
INSERT INTO USER_UNIQUE3 VALUES(2, 'user01', 'pass01', NULL, NULL, '010-1234-5678', 'hong123@kh.or.kr');
INSERT INTO USER_UNIQUE3 VALUES(2, 'user02', 'pass02', NULL, NULL, '010-1234-5678', 'hong123@kh.or.kr');
INSERT INTO USER_UNIQUE3 VALUES(1, 'user01', 'pass01', NULL, NULL, '010-1234-5678', 'hong123@kh.or.kr');
```

Table USER_UNIQUE3이(가) 생성되었습니다.

	TABLE_NAME	COLUMN_NAME	CONSTRAINT_NAME	CONSTRAINT_TYPE
1	USER_UNIQUE3	USER_NO	SYS_C007186	U
2	USER_UNIQUE3	USER_ID	SYS_C007186	U

테이블레벨에서밖에 설정 못한다
두 개 이상도 가능

이름이 똑같은 다른 사람은 있을 수 있다
전화번호가 같은 다른 사람은 있을 수 있다

그러나 이름이랑 전화번호가 완전히 같은 사람이 있으면 안됨, 둘다 똑같은 것에 대해서만 중복체크 하겠다

1 행 이(가) 삽입되었습니다.

1 행 이(가) 삽입되었습니다.

1 행 이(가) 삽입되었습니다.

```
SQL 오류: ORA-00001: unique constraint (EMPLOYEE.SYS_C007186) violated
00001. 00000 -  "unique constraint (%s.%s) violated"
*Cause:    An UPDATE or INSERT statement attempted to insert a duplicate key.
           For Trusted Oracle configured in DBMS MAC mode, you may see
           this message if a duplicate entry exists at a different level.
*Action:   Either remove the unique restriction or do not insert the key.
```

▶ 제약 조건(CONSTRAINTS)

✓ PRIMARY KEY

테이블에서 한 행의 정보를 구분하기 위한 고유 식별자 역할

NOT NULL의 의미와 UNIQUE의 의미를 둘 다 가지고 있으며

한 테이블 당 하나만 설정 가능

컬럼 레벨과 테이블 레벨 둘 다 지정 가능

▶ 제약 조건(CONSTRAINTS)

✓ PRIMARY KEY 예시1

```
CREATE TABLE USER_PRIMARYKEY(
        USER_NO NUMBER PRIMARY KEY,
        USER_ID VARCHAR2(20) UNIQUE,
        USER_PWD VARCHAR2(30) NOT NULL,
        USER_NAME VARCHAR2(30),
        GENDER VARCHAR2(10),
        PHONE VARCHAR2(30),
        EMAIL VARCHAR2(50)
);
```

```
CREATE TABLE USER_PRIMARYKEY(
        USER_NO NUMBER,
        USER_ID VARCHAR2(20) UNIQUE,
        USER_PWD VARCHAR2(30) NOT NULL,
        USER_NAME VARCHAR2(30),
        GENDER VARCHAR2(10),
        PHONE VARCHAR2(30),
        EMAIL VARCHAR2(50),
        PRIMARY KEY (USER_NO)
);
```

```
Table USER_PRIMARYKEY이(가) 생성되었습니다.
```

```
1 행 이(가) 삽입되었습니다.
```

```
INSERT INTO USER_PRIMARYKEY VALUES(1, 'user01', 'pass01', '홍길동', '남', '010-1234-5678', 'hong123@kh.or.kr');
INSERT INTO USER_PRIMARYKEY VALUES(1, 'user02', 'pass02', '이순신', '남', '010-5678-9012', 'lee123@kh.or.kr');
INSERT INTO USER_PRIMARYKEY VALUES(NULL, 'user03', 'pass03', '유관순', '여', '010-3131-3131',
'yoo123@kh.or.kr');
```

```
SQL 오류: ORA-00001: unique constraint (EMPLOYEE.SYS_C007188) violated
00001. 00000 -  "unique constraint (%s.%s) violated"
*Cause:    An UPDATE or INSERT statement attempted to insert a duplicate key.
           For Trusted Oracle configured in DBMS MAC mode, you may see
           this message if a duplicate entry exists at a different level.
*Action:   Either remove the unique restriction or do not insert the key.
```

```
오류 보고 -
SQL 오류: ORA-01400: cannot insert NULL into ("EMPLOYEE"."USER_PRIMARYKEY"."USER_NO")
01400. 00000 -  "cannot insert NULL into (%s)"
*Cause:    An attempt was made to insert NULL into previously listed objects.
*Action:   These objects cannot accept NULL values.
```

▶ 제약 조건(CONSTRAINTS)

✓ PRIMARY KEY 예시2

```
CREATE TABLE USER_PRIMARYKEY2(
        USER_NO NUMBER,
        USER_ID VARCHAR2(20),
        USER_PWD VARCHAR2(30) NOT NULL,
        USER_NAME VARCHAR2(30),
        GENDER VARCHAR2(10),
        PHONE VARCHAR2(30),
        EMAIL VARCHAR2(50),
        PRIMARY KEY (USER_NO, USER_ID)--두 컬럼을 묶어 한 PRIMARY KEY 제약조건 설정
);
```

Table USER_PRIMARYKEY2이(가) 생성되었습니다.

```
INSERT INTO USER_PRIMARYKEY2 VALUES(1, 'user01', 'pass01', '홍길동', '남', '010-1234-5678', 'hong123@kh.or.kr');
INSERT INTO USER_PRIMARYKEY2 VALUES(1, 'user02', 'pass02', '이순신', '남', '010-5678-9012', 'lee123@kh.or.kr');
INSERT INTO USER_PRIMARYKEY2 VALUES(2, 'user01', 'pass01', '유관순', '여', '010-3131-3131', 'yoo123@kh.or.kr');
INSERT INTO USER_PRIMARYKEY2 VALUES(1, 'user01', 'pass01', '신사임당', '여', '010-1111-1111', 'shin123@kh.or.kr');
```

1 행 이(가) 삽입되었습니다.

1 행 이(가) 삽입되었습니다.

1 행 이(가) 삽입되었습니다.

```
오류 보고 -
SQL 오류: ORA-00001: unique constraint (EMPLOYEE.SYS_C007196) violated
00001. 00000 -  "unique constraint (%s.%s) violated"
*Cause:    An UPDATE or INSERT statement attempted to insert a duplicate key.
           For Trusted Oracle configured in DBMS MAC mode, you may see
           this message if a duplicate entry exists at a different level.
*Action:   Either remove the unique restriction or do not insert the key.
```

	USER_NO	USER_ID	USER_PWD	USER_NAME	GENDER	PHONE	EMAIL
1	1	user01	pass01	홍길동	남	010-1234-5678	hong123@kh.or.kr
2	1	user02	pass02	이순신	남	010-5678-9012	lee123@kh.or.kr
3	2	user01	pass01	유관순	여	010-9999-3131	yoo123@kh.or.kr

▶ 제약 조건(CONSTRAINTS)

KH정보교육원

✓ FOREIGN KEY

참조 무결성을 위한 제약조건으로
참조된 다른 테이블이 제공한 값만 사용하도록 제한을 거는 것
참조되는 컬럼과 참조된 컬럼을 통해 테이블 간에 관계가 형성되는데
참조되는 값은 제공되는 값 외에 NULL을 사용 가능하며
참조할 테이블의 참조할 컬럼 명을 생략할 경우
PRIMARY KEY로 설정된 컬럼이 자동으로 참조할 컬럼이 됨

▶ 제약 조건(CONSTRAINTS)

✓ FOREIGN KEY 예시

```
CREATE TABLE USER_GRADE(
            GRADE_CODE NUMBER PRIMARY KEY,
            GRADE_NAME VARCHAR2(30) NOT NULL
);

INSERT INTO USER_GRADE VALUES(10, '일반회원');
INSERT INTO USER_GRADE VALUES(20, '우수회원');
INSERT INTO USER_GRADE VALUES(30, '특별회원');

SELECT * FROM USER_GRADE;
```

```
Table USER_GRADE이(가) 생성되었습니다.

1 행 이(가) 삽입되었습니다.
1 행 이(가) 삽입되었습니다.
1 행 이(가) 삽입되었습니다.
```

	GRADE_...	GRADE_NAME
1	10	일반회원
2	20	우수회원
3	30	특별회원

▶ 제약 조건(CONSTRAINTS)

KH정보교육원

✓ FOREIGN KEY 예시

```
CREATE TABLE USER_FOREIGNKEY(
            USER_NO NUMBER PRIMARY KEY,
            USER_ID VARCHAR2(20) UNIQUE,
            USER_PWD VARCHAR2(30) NOT NULL,
            USER_NAME VARCHAR2(30),
            GENDER VARCHAR2(10),
            PHONE VARCHAR2(30),
            EMAIL VARCHAR2(50),
            GRADE_CODE NUMBER,
            FOREIGN KEY (GRADE_CODE) REFERENCES USER_GRADE( GRADE_CODE)
);
```

또는

```
CREATE TABLE USER_FOREIGNKEY(
            USER_NO NUMBER PRIMARY KEY,
            USER_ID VARCHAR2(20) UNIQUE,
            USER_PWD VARCHAR2(30) NOT NULL,
            USER_NAME VARCHAR2(30),
            GENDER VARCHAR2(10),
            PHONE VARCHAR2(30),
            EMAIL VARCHAR2(50),
            GRADE_CODE NUMBER REFERENCES USER_GRADE (GRADE_CODE)
);
```

▶ 제약 조건(CONSTRAINTS)

KH정보교육원

✓ FOREIGN KEY 예시

```
INSERT INTO USER_FOREIGNKEY
VALUES(1, 'user01', 'pass01', '홍길동', '남', '010-1234-5678', 'hong123@kh.or.kr', 10);
```

1 행 이(가) 삽입되었습니다.

```
INSERT INTO USER_FOREIGNKEY
VALUES(2, 'user02', 'pass02', '이순신', '남', '010-9012-3456', 'lee123@kh.or.kr', 20);
```

1 행 이(가) 삽입되었습니다.

```
INSERT INTO USER_FOREIGNKEY
VALUES(3, 'user03', 'pass03', '유관순', '여', '010-3131-3131', 'yoo123@kh.or.kr', 30);
```

1 행 이(가) 삽입되었습니다.

```
INSERT INTO USER_FOREIGNKEY
VALUES(4, 'user04', 'pass04', '신사임당', '여', '010-1111-1111', 'shin123@kh.or.kr',
NULL);
```

```
INSERT INTO USER_FOREIGNKEY
VALUES(5, 'user05', 'pass05', '안중근', '남', '010-4444-4444', 'ahn123@kh.or.kr', 50);
```

```
오류 보고 -
SQL 오류: ORA-02291: integrity constraint (EMPLOYEE.SYS_C007202) violated - parent key not found
02291. 00000 - "integrity constraint (%s.%s) violated - parent key not found"
*Cause:    A foreign key value has no matching primary key value.
*Action:   Delete the foreign key or add a matching primary key.
```

▶ 제약 조건(CONSTRAINTS)

✓ FOREIGN KEY 예시

<USER_GRADE TABLE>

	GRADE_...	GRADE_NAME
1	10	일반회원
2	20	우수회원
3	30	특별회원

<USER_FOREIGNKEY TABLE>

	USER_NO	USER_ID	USER_PWD	USER_NAME	GENDER	PHONE	EMAIL	GRADE_CODE
1	1	user01	pass01	홍길동	남	010-1234-5678	hong123@kh.or.kr	10
2	2	user02	pass02	이순신	남	010-5678-9012	lee123@kh.or.kr	20
3	3	user03	pass03	유관순	여	010-9999-3131	yoo123@kh.or.kr	30
4	4	user04	pass04	안중근	남	010-2222-1111	ahn123@kh.or.kr	(null)

* FOREIGN KEY 제약조건으로 USER_GRADE TABLE의 GRADE_CODE 컬럼 참조

* USER_GRADE 테이블을 USER_FOREIGNKEY 테이블에서 참조하고 있기 때문에
  USER_GRADE 테이블의 데이터 삭제 시 참조 무결성에 위배되어 삭제 불가능
  → 부모테이블의 데이터 삭제 시 자식 테이블의 데이터를 어떤 방식으로 처리할지에 대한 내용을
  제약조건 설정 시 옵션으로 지정 가능
  기본 삭제 옵션은 **ON DELETE RESTRICTED :** 삭제 불가능으로 지정되어 있음

▶ 제약 조건(CONSTRAINTS)

KH정보교육원

✓ FOREIGN KEY 예시

```
CREATE TABLE USER_FOREIGNKEY2(
            USER_NO NUMBER PRIMARY KEY,
            USER_ID VARCHAR2(20) UNIQUE,
            USER_PWD VARCHAR2(30) NOT NULL,
            USER_NAME VARCHAR2(30),
            GENDER VARCHAR2(10),
            PHONE VARCHAR2(30),
            EMAIL VARCHAR2(50),
            GRADE_CODE NUMBER REFERENCES USER_GRADE (GRADE_CODE) ON DELETE SET NULL
);

DELETE FROM USER_GRADE WHERE GRADE_CODE = 10;
```

	GRADE_CODE	GRADE_NAME
1	20	우수회원
2	30	특별회원

	USER_NO	USER_ID	USER_PWD	USER_NAME	GENDER	PHONE	EMAIL	GRADE_CODE
1	1	user01	pass01	홍길동	남	010-1234-5678	hong123@kh.or.kr	(null)
2	2	user02	pass02	이순신	남	010-5678-9012	lee123@kh.or.kr	20
3	3	user03	pass03	유관순	여	010-9999-3131	yoo123@kh.or.kr	30
4	4	user04	pass04	안중근	남	010-2222-1111	ahn123@kh.or.kr	(null)

* **ON DELETE SET NULL :** 부모 테이블의 데이터 삭제 시 참조하고 있는 테이블의 컬럼 값이 NULL로 변경됨

▶ 제약 조건(CONSTRAINTS)

✓ FOREIGN KEY 예시

```
CREATE TABLE USER_FOREIGNKEY(
                USER_NO NUMBER PRIMARY KEY,
                USER_ID VARCHAR2(20) UNIQUE,
                USER_PWD VARCHAR2(30) NOT NULL,
                USER_NAME VARCHAR2(30),
                GENDER VARCHAR2(10),
                PHONE VARCHAR2(30),
                EMAIL VARCHAR2(50),
                GRADE_CODE NUMBER REFERENCES USER_GRADE (GRADE_CODE) ON DELETE CASCADE
);

DELETE FROM USER_GRADE WHERE GRADE_CODE = 10;
```

	GRADE_CODE	GRADE_NAME
1	20	우수회원
2	30	특별회원

	USE...	USER_ID	USER_PWD	USER_NAME	GENDER	PHONE	EMAIL	GRADE_CODE
1	2	user02	pass02	이순신	남	010-5678-9012	lee123@kh.or.kr	20
2	3	user03	pass03	유관순	여	010-9999-3131	yoo123@kh.or.kr	30
3	4	user04	pass04	안중근	남	010-2222-1111	ahn123@kh.or.kr	(null)

* **ON DELETE CASCADE :** 부모 테이블의 데이터 삭제 시 참조하고 있는 테이블의 컬럼 값이 존재하던 행 전체 삭제

▶ 제약 조건(CONSTRAINTS)

KH정보교육원

✓ CHECK

해당 컬럼에 입력 되거나 수정되는 값을 체크하여 설정된 값 이외의 값이면 에러 발생

비교 연산자를 이용하여 조건을 설정하며 비교 값을 리터럴만 사용 가능하고 변하는 값이나 함수 사용은 불가능

✓ CHECK 예시

```
CREATE TABLE USER_CHECK(
        USER_NO NUMBER PRIMARY KEY,
        USER_ID VARCHAR2(20) UNIQUE,
        USER_PWD VARCHAR2(30) NOT NULL,
        USER_NAME VARCHAR2(30),
        GENDER VARCHAR2(10) CHECK (GENDER IN ('남', '여')),
        PHONE VARCHAR2(30),
        EMAIL VARCHAR2(50)
);
INSERT INTO USER_CHECK VALUES(1, 'user01', 'pass01', '홍길동', '남자', '010-1234-5678',
'hong123@kh.or.kr');
```

```
오류 보고 -
SQL 오류: ORA-02290: check constraint (EMPLOYEE.SYS_C007225) violated
02290. 00000 -  "check constraint (%s.%s) violated"
*Cause:    The values being inserted do not satisfy the named check

*Action:   do not insert values that violate the constraint.
```