

DDL (Data Definition Language)

▶ 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(PROCEDURAL), 함수(FUNCTION), 트리거(TRIGGER), 동의어(SYNONYM), 사용자(USER)

▶ CREATE

테이블이나 인덱스, 뷰 등 데이터베이스 객체를 생성하는 구문

✓ 표현식

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

```
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) | |

▶ 오라클의 데이터형

| 데이터형 | 설명 |
|------------------------|--|
| CHAR(크기) | 고정길이 문자 데이터 |
| VARCHAR2(크기) | 가변길이 문자 데이터(최대 2,000 Byte) |
| NUMBER | 숫자 데이터(최대 40자리) |
| NUMBER(길이) | 숫자 데이터로, 길이 지정 가능하다 (최대 38자리) |
| DATE | 날짜 데이터(BC 4712년 1월 1일 ~ AD 4712년 12월 31일) |
| LONG | 가변 길이 문자형 데이터(최대 2GB) |
| LOB | 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 |
| INTERVAL YEAR TO MONTH | 년과 월을 이용하여 기간 저장 |
| INTERVAL DAY TO SECONT | 일, 시, 분, 초를 이용하여 기간 저장 |

▶ CHARACTER

✓ CHAR

CHAR(SIZE [(byte | char)]

- * SIZE : 포함될 문자(열)의 크기
- * 지정한 크기보다 작은 문자(열)가 입력되면 남은 공간은 공백으로 채움
- * 데이터는 "를 사용하여 표기하고 대·소문자를 구분함

| 실제 값 | 데이터 타입 | 저장 되는 값 | 설명 |
|--------|---------|-----------|------------------------------------|
| KIMCHI | CHAR(6) | KIMCHI | |
| | CHAR(9) | KIMCHI*** | 공백 3칸(3byte) |
| | CHAR(3) | (오류) | 저장되는 글자는 6글자인데 공간은 3자리이기 때문에 오류 |
| 김치 | CHAR(6) | 김치 | 한글은 한 글자 당 3byte이므로 공간에 딱 맞음 |
| | CHAR(9) | 김치*** | 공백 3byte |
| | CHAR(3) | (오류) | 저장되는 글자는 총 6byte인데 공간은 3byte이므로 오류 |

▶ CHARACTER

✓ VARCHAR2

VARCHAR2(SIZE [(byte | char)]

- * SIZE : 포함될 문자(열)의 크기
- * 크기가 0인 값은 NULL로 인식
- * 데이터는 "를 사용하여 표기하고 대·소문자를 구분함

| 실제 값 | 데이터 타입 | 저장 되는 값 | 설명 |
|--------|-------------|---------|------------------------------------|
| KIMCHI | VARCHAR(6) | KIMCHI | |
| | VARCHAR(10) | KIMCHI | |
| | VARCHAR(3) | (오류) | 저장되는 글자는 6글자인데 공간은 3자리이기 때문에 오류 |
| 김치 | VARCHAR(6) | 김치 | 한글은 한 글자 당 3byte이므로 공간에 딱 맞음 |
| | VARCHAR(10) | 김치 | |
| | VARCHAR(3) | (오류) | 저장되는 글자는 총 6byte인데 공간은 3byte이므로 오류 |

▶ NUMBER

NUMBER[(P [, S])]

* P : 표현할 수 있는 전체 숫자 자리 수 (1 ~ 38)

* S : 소수점 이하 자리 수 (-84 ~ 127)

| 실제 값 | 데이터 타입 | 저장 되는 값 | 설명 |
|------------|---------------|-----------|--|
| 12345.678 | NUMBER | 12345.678 | |
| | NUMBER(7) | 12346 | 7자리이지만 정수는 5자리이므로 5개만 표현, 첫 번째 소수로 인해 반올림 되어 저장 |
| | NUMBER(7, 1) | 12345.7 | 7자리이지만 정수 5자리와 소수 1자리만 표현, 두 번째 소수로 인해 반올림 되어 저장 |
| | NUMBER(7, 3) | (오류) | 7자리, 소수점 이하 3자리로 정수는 총 4자리인데 실제 값의 정수는 5자리이므로 오류 |
| | NUMBER(5, -2) | 12300 | s가 -2여서 소수점 왼쪽 두 번째 자리 4가 반올림되어 저장 |
| 0.1234 | NUMBER(4, 5) | (오류) | 유효숫자는 4개가 맞지만 소수점 아래가 5자리인데 4자리이므로 오류 |
| 0.01234 | NUMBER(4, 5) | 0.01234 | |
| 0.0001234 | NUMBER(3, 7) | (오류) | 소수점 이하 일곱 째 자리까지 유효숫자는 4개인데 p가 3이므로 오류 |
| 0.00001234 | NUMBER(3, 7) | 0.0000123 | 소수점 이하 일곱 째 자리까지 유효숫자는 3개이기 때문에 마지막 4 제외 |

▶ DATE

DATE

- * 일자(세기/년/월/일) 및 시간(시/분/초) 정보 관리
- * 기본적으로 화면에 년/월/일 정보만 표기
- * 날짜 연산 및 비교 가능

| 연산 | 결과 타입 | 설명 |
|------------|--------|----------------|
| 날짜 + 숫자 | DATE | 날짜에서 숫자만큼 며칠 후 |
| 날짜 - 숫자 | DATE | 날짜에서 숫자만큼 며칠 전 |
| 날짜 - 날짜 | NUMBER | 두 날짜의 일수 차 |
| 날짜 + 숫자/24 | DATE | 날짜 + 시간 |

▶ 컬럼 주석

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 |

▶ 제약 조건(CONSTRAINTS)

테이블 작성 시 각 컬럼에 기록될 데이터에 대해 제약조건을 설정할 수 있는데
이는 데이터 무결성 보장을 주 목적으로 함
입력 데이터에 문제가 없는지에 대한 검사와 데이터의 수정/삭제 가능 여부 검사 등을 위해 사용

| 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 | 참조되는 테이블의 컬럼의 값이 존재하면 허용 |
| CHECK | 저장 가능한 데이터 값의 범위나 조건을 지정하여 설정한 값만 허용 |

▶ 제약 조건(CONSTRAINTS)

✓ 제약조건 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 저장하거나 수정할 때 NULL 값을 허용하지 않도록 컬럼 레벨에서 제한

✓ 예시

```
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)
```

Table USER_NOTNULL이 (가) 생성되었습니다.

);

INSERT INTO USER_NOTNULL VALUES(1, 'user01', 'pass01', '홍길동', '남', '010-1234-5678', 'hong123@kh.or.kr'); | 1 행 미(가) 삽입되었습니다.

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)

✓ UNIQUE

컬럼 입력 값에 대해 중복을 제한하는 제약조건으로 컬럼 레벨과 테이블 레벨에 설정 가능

✓ 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)
```

Table USER_UNIQUE이(가) 생성되었습니다.

```
);
```

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

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

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
);
```

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

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.

Table USER_UNIQUE2이 (가) 생성되었습니다.

1 행 이 (가) 삽입되었습니다.

1 행 이 (가) 삽입되었습니다.

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

* UNIQUE 제약조건은 NULL 값 중복은 가능 → 테이블 생성 시 컬럼 레벨에 NOT NULL 지정하면 해결

▶ 제약 조건(CONSTRAINTS)

✓ 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 개의 컬럼을 묶어 하나의 UNIQUE 제약조건 설정
);
```

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 |

```
INSERT INTO USER_UNIQUE3 VALUES(1, 'user01', 'pass01', '홍길동', '남', '010-1234-5678', 'hong123@kh.or.kr'); | 1 행 미(가) 삽입되었습니다.
INSERT INTO USER_UNIQUE3 VALUES(2, 'user01', 'pass01', NULL, NULL, '010-1234-5678', 'hong123@kh.or.kr'); | 1 행 미(가) 삽입되었습니다.
INSERT INTO USER_UNIQUE3 VALUES(2, 'user02', 'pass02', NULL, NULL, '010-1234-5678', 'hong123@kh.or.kr'); | 1 행 미(가) 삽입되었습니다.
INSERT INTO USER_UNIQUE3 VALUES(1, 'user01', 'pass01', NULL, NULL, '010-1234-5678', 'hong123@kh.or.kr');
```

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(Identifier)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)
);
```

또는 CREATE TABLE USER_PRIMARYKEY(Table USER_PRIMARYKEY이 (가) 생성되었습니다.

```
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)
```

);

| 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 개의 컬럼을 묶어서 하나의 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'); | 1 행 이 (가) 삽입되었습니다.
INSERT INTO USER_PRIMARYKEY2 VALUES(1, 'user02', 'pass02', '이순신', '남', '010-5678-9012', 'lee123@kh.or.kr'); | 1 행 이 (가) 삽입되었습니다.
INSERT INTO USER_PRIMARYKEY2 VALUES(2, 'user01', 'pass01', '유관순', '여', '010-3131-3131', 'yoo123@kh.or.kr'); | 1 행 이 (가) 삽입되었습니다.
INSERT INTO USER_PRIMARYKEY2 VALUES(1, 'user01', 'pass01', '신사임당', '여', '010-1111-1111', 'shin123@kh.or.kr');
```

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)

✓ FOREIGN KEY

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

▶ 제약 조건(CONSTRAINTS)

✓ FOREIGN KEY 예시1

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

Table USER_GRADE이 (가) 생성되었습니다.

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

1 행 이 (가) 삽입되었습니다.

1 행 이 (가) 삽입되었습니다.

1 행 이 (가) 삽입되었습니다.

```
SELECT * FROM USER_GRADE;
```

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

▶ 제약 조건(CONSTRAINTS)

✓ FOREIGN KEY 예시1

```
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)

✓ FOREIGN KEY 예시1

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

INSERT INTO USER_FOREIGNKEY
VALUES(2, 'user02', 'pass02', '이순신', '남', '010-9012-3456', 'lee123@kh.or.kr', 20); | 1 행 미(가) 삽입되었습니다.

INSERT INTO USER_FOREIGNKEY
VALUES(3, 'user03', 'pass03', '유관순', '여', '010-3131-3131', 'yoo123@kh.or.kr', 30); | 1 행 미(가) 삽입되었습니다.

INSERT INTO USER_FOREIGNKEY
VALUES(4, 'user04', 'pass04', '신사임당', '여', '010-1111-1111', 'shin123@kh.or.kr', NULL); | 1 행 미(가) 삽입되었습니다.

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 예시1

<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 RESTRICTED로 지정되어 있음

▶ 제약 조건(CONSTRAINTS)

✓ FOREIGN KEY 옵션(ON DELETE SET NULL)

```
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 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) |

* 부모 테이블의 데이터 삭제 시 참조하고 있는 테이블의 컬럼 값이 NULL로 변경됨

▶ 제약 조건(CONSTRAINTS)

✓ FOREIGN KEY 옵션(ON DELETE CASCADE)

```
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) |

* 부모 테이블의 데이터 삭제 시 참조하고 있는 테이블의 컬럼 값이 존재하던 행 전체 삭제

▶ 제약 조건(CONSTRAINTS)

✓ 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

✓ UNIQUE 예시1

```
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.
```

▶ SUBQUERY를 이용한 CREATE TABLE

서브 쿼리를 이용해서 SELECT의 조회 결과로 테이블을 생성하는 방법으로
컬럼명과 데이터 타입, 값이 복사 되고 제약 조건은 NOT NULL만 복사됨

✓ 예시

```
CREATE TABLE EMPLOYEE_COPY
AS SELECT EMP_ID, EMP_NAME, SALARY, DEPT_TITLE, JOB_NAME
FROM EMPLOYEE
LEFT JOIN DEPARTMENT ON (DEPT_CODE = DEPT_ID)
LEFT JOIN JOB USING(JOB_CODE);
```

SQL | 인출된 모든 행: 23(0,016초)

| EMP_ID | EMP_NAME | SALARY | DEPT_TITLE | JOB_NAME |
|--------|----------|---------|------------|----------|
| 1 214 | 방명수 | 1380000 | 인사관리부 | 사원 |
| 2 216 | 차태연 | 2780000 | 인사관리부 | 대리 |
| 3 217 | 전지연 | 3660000 | 인사관리부 | 대리 |
| 4 219 | 임시환 | 1550000 | 회계관리부 | 차장 |
| 5 220 | 이중석 | 2490000 | 회계관리부 | 차장 |
| 6 221 | 유하진 | 2480000 | 회계관리부 | 차장 |
| 7 206 | 백 나 | 1800000 | 해외영업1부 | 사원 |
| ... | | | | |
| 21 200 | 선동일 | 8000000 | 총무부 | 대표 |
| 22 218 | 미오리 | 2890000 | (null) | 사원 |
| 23 213 | 하동운 | 2320000 | (null) | 대리 |