ch14. 제약조건

14-1. 제약 조건 종류

> 제약 조건이란?

- 오라클에서 사용하는 제약 조건은 테이블의 특정 열에 지정한다.
- 제약 조건을 지정한 열에 제약 조건에 부합하지 않는 데이터를 저장할 수 없다.
- 제약 조건 지정 방식에 따라 기존 데이터의 수정이나 삭제 가능 여부도 영향을 받는다.
- 제약 조건 종류

NOT NULL	지정한 열에 NULL을 허용하지 않는다. NULL을 제외한 데이터의 중복은 허용.
UNIQUE	열이 유일한 값을 가져야 하고 중복이 없다. 단 NULL은 값의 중복에서 제외.
PRIMARY KEY	지정한 열이 유일한 값이면서 NULL을 허용하지 않는다. PRIMARY KEY는 테이블에 하나만 지정 가능.
FOREIGN KEY	다른 테이블의 열을 참조하여 존재하는 값만 입력할 수 있다.
CHECK	조건식을 만족하는 데이터만 입력 가능.

- 데이터 무결성 *Data integrity*
  - 데이터베이스에 저장되는 데이터의 정확성과 일관성을 보장한다는 의미
  - 이를 위해 항상 유지해야 하는 기본 규칙을 가지고 있다.

영역 무결성 domain integrity	열에 저장되는 값의 적정 여부를 확인. 자료형, 적절한 형식의 데이터, NULL 여부같은 정해 놓은 범위를 만족하는 데이터임을 규정.
개체 무결성 entity integrity	테이블 데이터를 유일하게 식별할 수 있는 기본키는 반드시 값을 가지고 있어야 하며 NULL이 될 수 없고 중복될 수도 없음.
참조 무결성 referential integrity	참조 테이블의 외래키 값은 참조 테이블의 기본키로서 존재해야 하며 NULL이 가능.

- 제약 조건은 데이터 정의어 DDL에서 활용함.

14-2. 빈 값을 허락하지 않는 NOT NULL

테이블을 생성하며 제약 조건 지정

- NOT NULL은 특정 열에 데이터의 중복 여부와는 상관없이 NULL의 저장을 허용하지 않는
  제약 조건.
- 반드시 열에 값이 존재해야만 하는 경우에 지정.

- 테이블을 생성할 때 NOT NULL 지정

```
CREATE TABLE TABLE_NOTNULL(
   LOGIN_ID VARCHAR2(20) NOT NULL,
   LOGIN_PWD VARCHAR2(20) NOT NULL,
   TEL VARCHAR2(20)
);

DESC TABLE_NOTNULL;
```

제약 조건 확인

- 지정한 제약 조건 정보를 확인하려면 다음과 같은 USER_CONSTRAINTS 데이터 사전을 활용

OWNER	제약 조건 소유 계정
CONSTRAINT_NAME	제약 조건 이름 (직접 지정하지 않을 경우 오라클이 자동으로 지정함)
CONSTRAINT_TYPE	제약 조건 종류 C : CHECK, NOT NULL U : UNIQUE P : PRIMARY KEY R : FOREIGN KEY
TABLE_NAME	제약 조건을 지정한 테이블 이름

제약 조건 이름 직접 지정

- 제약 조건에 이름을 직접 지정하려면 CONSTRAINT 키워드를 사용한다.

```
CREATE TABLE TABLE_NOTNULL2(
   LOGIN_ID VARCHAR2(20) CONSTRAINT TBLNN2_LGNID_NN NOT NULL,
   LOGIN_PWD VARCHAR2(20) CONSTRAINT TBLNN2_LGNPW_NN NOT NULL,
   TEL VARCHAR2(20)
```

```
);

SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
  FROM USER_CONSTRAINTS;
```

이미 생성한 테이블에 제약 조건 지정

- 생성한 테이블에 제약 조건 추가하기
  - NOT NULL 제약 조건의 추가는 ALTER 명령어와 MODIFY 키워드를 사용한다.

```
ALTER TABLE TABLE_NOTNULL
MODIFY(TEL NOT NULL);
```

- TEL 열을 NULL 이 아닌 데이터로 수정

```
UPDATE TABLE_NOTNULL
   SET TEL = '010-1234-5678'
 WHERE LOGIN_ID = 'TEST_ID_01';

SELECT * FROM TABLE_NOTNULL;
```

- 생성한 테이블에 제약 조건 이름 직접 지정해서 추가

```
ALTER TABLE TABLE_NOTNULL2
MODIFY(TEL CONSTRAINT TBLNN_TEL_NN NOT NULL);

SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
  FROM USER_CONSTRAINTS;
```

- 생성한 제약 조건의 이름 변경

```
ALTER TABLE TABLE_NOTNULL2
RENAME CONSTRAINT TBLNN_TEL_NN TO TBLNN2_TEL_NN;
```

```
SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
  FROM USER_CONSTRAINTS;
```

제약 조건 삭제

- ALTER 명령어에 DROP CONSTRAINT 키워드를 사용하면 지정한 제약 조건을 삭제할 수 있다.

```
ALTER TABLE TABLE_NOTNULL2
 DROP CONSTRAINT TBLNN2_TEL_NN;

DESC TABLE_NOTNULL2;
```

14-3. 중복되지 않는 값 UNQUE

- UNIQUE 제약 조건은 열에 저장할 데이터의 중복을 허용하지 않고자 할 때 사용
- NULL 은 값이 존재하지 않음을 의미하기 때문에 중복 대상에서 제외된다.
    - 즉 UNIQUE 제약 조건을 지정한 열에 NULL은 여러 개 존재할 수 있다.

테이블을 생성하며 제약 조건 지정

- 제약 조건 지정 (테이블 생성 시)

```
CREATE TABLE TABLE_UNIQUE(
   LOGIN_ID VARCHAR2(20) UNIQUE,
   LOGIN_PWD VARCHAR2(20) NOT NULL,
   TEL VARCHAR2(20)
);

DESC TABLE_UNIQUE;
```

제약 조건 확인

- USER_CONSTRAINTS 데이터 사전에서
  CONSTRAINT_TYPE 열 값이 U 일 경우에 UNIQUE 제약 조건을 의미

- USER_CONSTRAINTS 데이터 사전 뷰로 제약 조건 확인

```
SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
  FROM USER_CONSTRAINTS
 WHERE TABLE_NAME = 'TABLE_UNIQUE';
```

중복을 허락하지 않는 UNIQUE

- UNIQUE 제약 조건을 지정한 LOGIN_ID 열은 중복 값이 저장되지 않는다.

- TABLE_UNIQUE 테이블에 데이터 입력하기

```
INSERT INTO TABLE_UNIQUE(LOGIN_ID, LOGIN_PWD, TEL)
VALUES('TEST_ID_01', 'PWD01', '010-1234-5678');

SELECT * FROM TABLE_UNIQUE;
```

- LOGIN_ID 열에 중복되는 데이터 넣기

```
INSERT INTO TABLE_UNIQUE (LOGIN_ID, LOGIN_PWD, TEL)
VALUES ('TEST_ID_01', 'PWD01', '010-1234-5678');
```

UNIQUE 제약 조건과 NULL 값

- NULL 은 데이터 중복의 의미를 부여할 수 없다.
- 따라서 UNIQUE 제약 조건이 지정된 열에는 NULL이 여러 개 존재할 수 있다.

테이블을 생성하며 제약 조건 이름 직접 지정

- 테이블을 생성할 때 UNQUE 제약 조건 설정하기

```
CREATE TABLE TABLE_UNIQUE2(
   LOGIN_ID VARCHAR2(20) CONSTRAINT TBLUNQ2_LGNID_UNQ UNIQUE,
   LOGIN_PWD VARCHAR2(20) CONSTRAINT TBLUNQ2_LGNPW_NN NOT NULL,
   TEL VARCHAR2(20)
);
```

⇒ USER_CONSTRAINTS 데이터 사전을 조회하면 제약 조건 이름이 앞에서 지정한 대로 저장.

```
SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
  FROM USER_CONSTRAINTS
 WHERE TABLE_NAME LIKE 'TABLE_UNIQUE%';
```

이미 생성한 테이블에 제약 조건 지정

- ALTER 명령어로 이미 생성되어 있는 테이블에 UNIQUE 제약 조건을 추가할 수 있다.

- 이미 생성한 테이블 열에 UNIQUE 제약 조건 추가

```
ALTER TABLE TABLE_UNIQUE
MODIFY(TEL UNIQUE);
```

- 생성한 테이블에 제약 조건 이름 직접 지정하거나 바꾸기

```
ALTER TABLE TABLE_UNIQUE2
MODIFY(TEL CONSTRAINT TBLUNQ_TEL_UNQ UNIQUE);

SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
  FROM USER_CONSTRAINTS
 WHERE TABLE_NAME LIKE 'TABLE_UNIQUE%';
```

- 이미 만들어져 있는 UNIQUE 제약 조건 이름 수정

```
ALTER TABLE TABLE_UNIQUE2
RENAME CONSTRAINT TBLUNQ_TEL_UNQ TO TBLUNQ2_TEL_UNQ;

SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
  FROM USER_CONSTRAINTS
 WHERE TABLE_NAME LIKE 'TABLE_UNIQUE%';
```

제약 조건 삭제

- UNIQUE 제약 조건은 ALTER 명령어에 DROP CONSTRAINT 키워드를 사용

```
ALTER TABLE TABLE_UNIQUE2
 DROP CONSTRAINT TBLUNQ2_TEL_UNQ;

SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
  FROM USER_CONSTRAINTS
 WHERE TABLE_NAME LIKE 'TABLE_UNIQUE%';
```

14-4. 유일하게 하나만 있는 값 PRIMARY KEY

- UNIQUE와 NOT NULL 제약 조건의 특성을 모두 가지는 제약 조건
- 데이터 중복을 허용하지 않고 NULL 도 허용하지 않는다.
- PRIMARY KEY 제약 조건은 테이블에 하나밖에 지정할 수 없다.
- 특정 열을 PRIMARY KEY 로 지정하면 해당 열에는 자동으로 인덱스가 만들어진다.

```
CREATE TABLE TABLE_PK(
   LOGIN_ID VARCHAR2(20) PRIMARY KEY,
   LOGIN_PWD VARCHAR2(20) NOT NULL,
   TEL VARCHAR2(20)
);

DESC TABLE_PK;
```

- 생성한 PRIMARY KEY 확인하기

```
SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
  FROM USER_CONSTRAINTS
 WHERE TABLE_NAME LIKE 'TABLE_PK%';
```

테이블을 생성하며 제약 조건 이름 직접 지정하기

```
CREATE TABLE TABLE_PK2(
   LOGIN_ID VARCHAR2(20) CONSTRAINT TBLPK2_LGNID_PK PRIMARY KEY,
   LOGIN_PWD VARCHAR2(20) CONSTRAINT TBLPK2_LGNPW_NN NOT NULL,
   TEL VARCHAR2(20)
);

DESC TABLE_PK2;
```

- 제약 조건을 지정하지 않았을 때도 오라클이 자동으로 생성한 이름이 인덱스에 사용된다.

PRIMARY KEY 제약 조건을 지정한 열 확인 (중복 값을 입력했을 때)

- PRIMARY KEY 제약 조건을 지정한 열에는 중복 값과 NULL 이 허용되지 않는다.

- TABLE_PK 테이블에 데이터 입력하기

```
INSERT INTO TABLE_PK(LOGIN_ID, LOGIN_PWD, TEL)
VALUES('TEST_ID_01', 'PWD01', '010-1234-5678');

SELECT * FROM TABLE_PK;
```

- TABLE_PK 테이블에 중복되는 값 입력하기

```
INSERT INTO TABLE_PK(LOGIN_ID, LOGIN_PWD, TEL)
VALUES('TEST_ID_01', 'PWD02', '010-2345-6789');
```

PRIMARY KEY 제약 조건을 지정한 열 확인 (NULL 값을 입력했을 때)

- NULL 값을 명시적으로 입력하기

```
INSERT INTO TABLE_PK(LOGIN_ID, LOGIN_PWD, TEL)
VALUES(NULL, 'PWD02', '010-2345-6789');
```

- NULL 값을 암시적으로 입력하기

```
INSERT INTO TABLE_PK(LOGIN_PWD, TEL)
VALUES('PWD02', '010-2345-6789');
```

⇒ 둘 다 오류 발생 (NULL 값을 허용하지 않음)

- PRIMARY KEY 제약 조건을 지정하려는 열에 중복 값이나 NULL 이 있을 경우에도 동작하지 않음

14-5. 다른 테이블과 관계를 맺는 FOREIGN KEY

서로 다른 테이블 간 관계를 정의하는 데 사용하는 제약 조건이다.

- EMP 테이블과 DEPT 테이블의 제약 조건 살펴보기

```
SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME, R_OWNER, R_CONSTRAINT_NAME
  FROM USER_CONSTRAINTS
 WHERE TABLE_NAME IN ('EMP', 'DEPT');
```

FOREIGN KEY 지정하기

```
CREATE TABLE 테이블 이름(
  ...(다른 열 정의),
  열 자료형 CONSTRAINT [제약 조건 이름] REFERENCES 참조 테이블(참조할 열)
);
```

FOREIGN KEY 로 참조 행 데이터 삭제하기

- 열 데이터를 삭제할 때 이 데이터를 참조하고 있는 데이터도 함께 삭제

```
CONSTRAINT [제약 조건 이름] REFERENCES 참조 테이블(참조할 열) ON DELETE CASCADE
```

- 열 데이터를 삭제할 때 이 데이터를 참조하는 데이터를 NULL로 수정

```
CONSTRAINT [제약 조건 이름] REFERENCES 참조 테이블(참조할 열) ON DELETE SET NULL
```

14-6. 데이터 형태와 범위를 정하는 CHECK

CHECK 제약 조건은 열에 저장할 수 있는 값의 범위 또는 패턴을 정의할 때 사용한다.

- 테이블을 생성할 때 CHECK 제약 조건 설정하기

```
CREATE TABLE TABLE_CHECK(
   LOGIN_ID VARCHAR2(20) CONSTRAINT TBLCK_LOGINID_PK PRIMARY KEY,
   LOGIN_PWD VARCHAR2(20) CONSTRAINT TBLCK_LOGINPW_CK CHECK (LENGTH(LOGIN_PWD) > 3),
   TEL VARCHAR2(20)
);

DESC TABLE_CHECK;
```

- CHECK 제약 조건에 맞지 않는 예

```
INSERT INTO TABLE_CHECK
VALUES ('TEST_ID', '123', '010-1234-5678');
```

- CHECK 제약 조건에 맞는 예

```
INSERT INTO TABLE_CHECK
VALUES ('TEST_ID', '1234', '010-1234-5678');
```

- CHECK 제약 조건 확인하기

```
SELECT OWNER, CONSTRAINT_NAME, CONSTRAINT_TYPE, TABLE_NAME
  FROM USER_CONSTRAINTS
 WHERE TABLE_NAME LIKE 'TABLE_CHECK';
```

14-7. 기본값을 정하는 DEFAULT

제약 조건과는 별개로 특정 열에 저장할 값이 지정되지 않았을 경우에 기본값 (default) 을 지정.

- 테이블을 생성할 때 DEFAULT 제약 조건 설정하기

```
CREATE TABLE TABLE_DEFAULT(
   LOGIN_ID VARCHAR2(20) CONSTRAINT TBLCK2_LOGINID_PK PRIMARY KEY,
   LOGIN_PWD VARCHAR2(20) DEFAULT '1234',
   TEL VARCHAR2(20)
);

DESC TABLE_DEFAULT;
```

- DAFAULT로 지정한 기본값이 입력되는 INSERT문 확인하기

```
INSERT INTO TABLE_DEFAULT VALUES ('TEST_ID', NULL, '010-1234-5678');

INSERT INTO TABLE_DEFAULT (LOGIN_ID, TEL) VALUES ('TEST_ID2', '010-1234-5678');

SELECT * FROM TABLE_DEFAULT;
```

제약 조건 비활성화, 활성화

제약 조건은 데이터 무결성을 보장하는 데 중요한 역할을 하는데 신규 기능 개발 또는 테스트 같은 특정 업무를 수행해야 할 때 제약 조건이 걸림돌이 되는 경우가 있다.

```
ALTER TABLE 테이블이름
DISABLE [NOVALIDATE / VALIDATE(선택)] CONSTRAINT 제약조건 이름;
```

```
ALTER TABLE 테이블이름
ENABLE [NOVALIDATE / VALIDATE(선택)] CONSTRAINT 제약조건 이름;
```

Q

1.

```
CREATE TABLE DEPT_CONST (
   DEPTNO NUMBER(2)    CONSTRAINT DEPTCONST_DEPTNO_PK PRIMARY KEY,
   DNAME  VARCHAR2(14) CONSTRAINT DEPTCONST_DNAME_UNQ UNIQUE,
   LOC    VARCHAR2(13) CONSTRAINT DEPTCONST_LOC_NN NOT NULL
);
```

2.

```
CREATE TABLE EMP_CONST (
   EMPNO    NUMBER(4) CONSTRAINT EMPCONST_EMPNO_PK PRIMARY KEY,
   ENAME    VARCHAR2(10) CONSTRAINT EMPCONST_ENAME_NN NOT NULL,
   JOB      VARCHAR2(9),
   TEL      VARCHAR2(20) CONSTRAINT EMPCONST_TEL_UNQ UNIQUE,
   HIREDATE DATE,
   SAL      NUMBER(7, 2) CONSTRAINT EMPCONST_SAL_CHK CHECK (SAL BETWEEN 1000 AND 999
9),
   COMM     NUMBER(7, 2),
   DEPTNO   NUMBER(2) CONSTRAINT EMPCONST_DEPTNO_FK REFERENCES DEPT_CONST (DEPTNO)
);
```

3.

```
SELECT TABLE_NAME, CONSTRAINT_NAME, CONSTRAINT_TYPE
  FROM USER_CONSTRAINTS
 WHERE TABLE_NAME IN ( 'EMP_CONST', 'DEPT_CONST' )
ORDER BY CONSTRAINT_NAME;
```