Database Part3

테이블 생성과 삭제
데이터 삽입
COMMIT과 ROLLBACK
데이터 수정
데이터 삭제

Database | 테이블 생성하기- /

데이터베이스의 모든 데이터는 테이블(릴레이션)에 저장된다. 그렇기 때문에 데이터 저장을 위해서는 먼저 테이블을 생성하는 문법을 알아야한다.

테이블을 생성하는 문법은 다음과 같다.

```
CREATE TABLE 테이블명(
    컬럼명 테이터타입 [제약조건],
    컬럼명 테이터타입 [제약조건],
    컬럼명 테이터타입 [제약조건],
    …
);
```

제약조건은 필요 시에만 작성하기 때문에 생략 가능하다.

생략이 가능하다는 의미에서 [] 안에 작성하였다.

데이터타입	설명	비고
INT	정수	-21억~21억
DECIMAL(m, [d])	전체 자릿수(m)와 소수점 이하 자릿수(d)를 가진 숫자형	DECIMAL(5,2) 전체 자릿수5자리, 그 중 소수점 이하를 두자리로 표현
CHAR(n)	고정길이 문자	1 <= n <= 255
VARCHAR(n)	가변길이 문자 --넣는만큼만 쓰임	1 <= n <= 65535
DATE	날짜 형식 데이터	YYYY-MM-DD 형식
DATETIME	날짜 및 시간 데이터	YYYY-MM-DD HH:MM:SS 형식

많이 사용하는 데이터타입

제약조건의 정식 명칭은 '무결성 제약조건'이다. 이는 데이터에 결함이 없도록 특별한 제약 사항을 컬럼에 추가하여 데이터 삽입, 수정, 삭제 등 데이터에 변화가 일어날 때 데이터의 신뢰성을 지키기 위한 용도로 사용한다. '제약조건'이라는 단어가 부정적 느낌을 주지만, 실은 잘못된 데이터로 인해 데이터의 신뢰도가 떨어지는 것을 막기 위한 안전 장치이다. 무결성 제약조건은 필요할 때마다 조금씩 소개하겠다.

다음은 학생정보 테이블의 예시이다.

```
CREATE TABLE STUDENT(
    STU_NUM INT PRIMARY KEY,              -- 학번 컬럼 , 정수타입
    STU_NAME VARCHAR(10) NOT NULL,   -- 학생명 컬럼, 문자타입, 가변길이로 최대 10글자
    STU_TEL VARCHAR(20) UNIQUE,         -- 연락처 컬럼, 문자타입, 가변길이로 최대 20글자
    INTRO VARCHAR(100)                        -- 학생소개, 문자타입, 가변길이로 최대 100글자
);
```

위 테이블에 작성한 무결성 제약조건은 아주 기본이면서, 반드시 알아야 하는 제약조건이다.

PRIMARY KEY 제약조건 : 기본키 제약조건이라고도 하며, 모든 테이블에 반드시 있어야 하는 제약조건이다. 중복 X, NULL X

기본키는 각 행을 구분하는 식별자로 유일성과 최소성을 만족해야 하며, NULL 데이터가 들어갈 수 없다.

NOT NULL 제약조건 : NOT NULL 제약조건이 붙은 컬럼은 NULL 데이터가 들어갈 수 없다.

UNIQUE 제약조건 : UNIQUE 제약조건이 붙은 컬럼은 중복 데이터가 들어 갈 수 없다. NULL데이터는 허용한다.

다음은 게시글 정보 테이블의 예시이다.

```
CREATE TABLE BOARD(
    BOARD_NUM INT PRIMARY KEY,          -- 글 번호 컬럼
    TITLE VARCHAR(10) NOT NULL,         -- 제목 컬럼
    WRITER VARCHAR(20) NOT NULL,          -- 작성자 컬럼
    CONTENT VARCHAR(100),               -- 내용컬럼
    READ_CNT INT DEFAULT 0,             -- 조회수 컬럼
    CREATE_DATE DATETIME DEFAULT CURRENT_TIMESTAMP()  -- 작성일 컬럼
);
```

DEFUALT 제약조건

데이터 삽입 시 해당 컬럼의 값을 지정하지 않을 경우 기본값을 설정하는 제약조건이다.

DEFAULT 0 은 해당 컬럼에 데이터 삽입 시 값을 명시하지 않으면 0을 삽입하라는 의미이다.

DEFAULT CURRENT_TIMESTAMP()는 데이터타입이 DATETIME인 컬럼에 사용할 수 있으며 데이터 삽입 시 명시적으로 값을 지정하지 않으면 데이터 삽입 명령어가 실행되는 시점의 날짜 및 시간을 자동으로 삽입한다.

생성한 테이블을 삭제하는 쿼리는 간단한다. 테이블 삭제 쿼리의 문법은 다음과 같다.

DROP TABLE 테이블명;

주의!!!

테이블을 삭제하는 것과 테이블에 저장된 데이터를 삭제하는 것은 전혀 다른 작업이다. DROP TABLE 쿼리로 테이블을 삭제하면 테이블 안의 데이터도 모두 사라진다. 테이블에 저장된 데이터는 실수로 삭제해도 되돌릴 기회가 있으나, 테이블 삭제 쿼리는 한 번 실행하면 되돌릴 수 없다. 취업 후 신입 사원 신분으로 회사가 몇 년간 쌓은 데이터가 들어있는 테이블을 한 순간의 실수로 날려버릴 수 있는 것이다.. 그 이후의 일은 상상에 맡기겠다...

데이터를 삽입 할 때는 INSERT 쿼리를 사용한다. INSERT 쿼리의 기본문법은 다음과 같다.

INSERT INTO 테이블명 (컬럼명, 컬럼명...) VALUES (데이터, 데이터...);

다음과 같은 테이블을 기준으로 다음 슬라이드부터 데이터 삽입 쿼리의 예시를 보겠다.

```
CREATE TABLE STUDENT(
    STU_NUM INT PRIMARY KEY,              -- 학번 컬럼
    STU_NAME VARCHAR(10) NOT NULL,    -- 학생명 컬럼
    KOR_SCORE INT,                          -- 국어 점수 컬럼
    ENG_SCORE INT DEFAULT 0,             -- 영어 점수 컬럼
    INTRO VARCHAR(50)                     -- 학생소개 컬럼
);
```

INSERT INTO STUDENT (STU_NUM, STU_NAME, KOR_SCORE, ENG_SCORE, INTRO)

VALUES (1, '김자바', 50, 80, '김자바입니다');

테이블명 다음 () 안에는 데이터를 삽입하고자 하는 컬럼을 쉼표로 구분하여 나열한다. VALUES 다음의 () 안에는 나열한 컬럼에 들어갈 데이터를 적는다. 이때, 문자 데이터는 반드시 홀따옴표에 감싼다. 데이터 삽입을 위해 나열한 컬럼명의 순서대로 삽입할 데이터를 작성한다.

INSERT INTO STUDENT (STU_NUM, STU_NAME, ENG_SCORE) VALUES (2, '박자바', 70);

위 쿼리는 STUDENT 테이블의 모든 컬럼을 나열하지 않았다.(KOR_SOCRE, INTRO 컬럼 제외)

이렇게 데이터 삽입 쿼리문을 작성할 때 반드시 모든 컬럼을 작성할 필요는 없다. 단, 나열한 컬럼대로 데이터가 들어가야 하기 때문에 컬럼을 3개 나열하였다면 VALUES 다음 삽입 할 데이터도 나열한 컬럼의 순서와 갯수에 맞춰 작성한다. 삽입 후 확인해보면 나열하지 않은 컬럼에 NULL 데이터가 들어간 것을 확인 할 수 있다.

이 상태에서 위 쿼리를 다시 실행해보자. 그럼 오류가 나는 것을 볼 수 있을 것이다.

그 이유는 이미 STU_NUM 값으로 2가 들어가 있는데, 다시 넣으려고 하기 때문이다. STU_NUM에 기본키 제약조건이 걸려있으며 기본키 제약조건이 추가된 컬럼에는 NULL데이터와 중복 데이터가 들어 갈 수 없다. STU_NAME, ENG_SCORE는 특별한 제약조건이 없기 때문에 중복 데이터가 삽입된다.

위 코드에서 STU_NUM 컬럼 값만 3으로 변경 후 다시 쿼리를 실행하면 잘 동작하는 것을 확인할 수 있다.

INSERT 쿼리를 더 학습하기 전에 아주 중요한 내용을 먼저 짚고 넘어가겠다.

엑셀 프로그램에 거래처 정보가 10만개 있다고 해도, 우리가 학습하는 DBMS 프로그램과는 데이터베이스 측면에서 비교불가하다.

데이터는 신뢰도가 아주 중요하다. 10만개의 데이터 중 단 한개의 데이터가 잘못 되더라도, 나머지 9만 9999개의 데이터는 믿을 수 없는 단순 문자열이 되어버린다. 즉, 데이터로서 가치를 잃어버리게 된다. 그렇기 때문에 무결성 제약조건이라는 것은 데이터베이스에서 상당한 역할을 담당하고 있다.

무결성 제약조건과 함께 데이터베이스에 저장된 데이터의 신뢰도를 보존하는 기능이 더 있다. 대표적인 것이 COMMIT과 ROLLBACK이다.

COMMIT은 데이터의 변화를 확정하는 명령어이고, ROLLBACK은 데이터의 변화를 취소하는 명령어이다.

(여기서 말하는 데이터의 변화는 데이터의 삽입, 수정, 삭제를 의미한다. 데이터를 조회한다고 해서 데이터가 달라지진 않으니 SELECT는 제외된다.)

즉, 데이터의 변화가 발생했을 때, 그 변화가 잘못된 변화라면 ROLLBACK을 통해 변화를 취소시키고 데이터의 신뢰도를 유지할 수 있다. COOMIT과 ROLLBACK은 데이터가 변했을 때, 그 변화가 문제가 없는 것인지, 잘못되지는 않았는지 한번 더 확인할 수 있는 기회를 제공하는 것이다.

그래서.. 우리가 지금까지 학습을 위해 INSERT문을 실행한 것은 COMMIT이 되었을까? 아니다. 우리는 COMMIT을 하지 않았다. 다시 말해 우리가 INSERT문을 통해 삽입한 데이터는 저장이 확정되지 않았다는 말이다. ROLLBACK;를 쳐보자. 아마 지금까지 INSERT 쿼리 실행으로 삽입된 데이터가 모두 사라져버리는 것을 확인할 수 있다.

다시 말하지만 COMMIT과 ROLLBACK은 정말 중요한 내용이다. 데이터의 변화(삽입, 삭제, 수정)를 일으키는 쿼리 실행 후 반드시 COMMIT 혹은 ROLLBACK 명령어를 실행하는 것을 잊지 말아야 한다!!!

계속해서 INSERT 쿼리에 대한 설명을 이어가겠다.

INSERT INTO STUDENT (STU_NUM, STU_NAME, ENG_SCORE, INTRO) VALUES (5, '윤자바', 80, '윤자바입니다');

위 쿼리는 컬럼에서 KOR_SCORE 컬럼을 제외하였다. 나열한 컬럼이 4개이니, VALUES 다음의 데이터도 순서에 맞게 4개를 작성하였다.

이 쿼리를 실행하면 오류가 발생할 것이다. 이야기한대로 삽입 시 제외되는 컬럼에는 자동으로 NULL 데이터가 삽입된다. 하지만 테이블 생성 쿼리에서 KOR_SCORE 컬럼을 보면 NOT_NULL 제약조건이 추가된 것을 볼 수 있다. NOT_NULL 제약조건은 NULL 데이터 삽입 불가이기 때문에 쿼리 실행 시 오류가 발생하는 것이다.

INSERT INTO STUDENT (STU_NUM, STU_NAME, KOR_SCORE, INTRO) VALUES (5, '윤자바', 80, NULL);

위 쿼리는 컬럼에서 ENG_SCORE 컬럼을 제외하였다. 나열한 컬럼이 4개이니, VALUES 다음의 데이터도 순서에 맞게 4개를 작성하였다.

이 쿼리에서는 INTRO 컬럼에 명시적으로 NULL 데이터를 넣는 문법을 보여주고 있다. INTRO 컬럼은 제약조건이 없기 때문에 NULL 데이터 삽입이 가능하다. 다음으로 ENG_SCORE 컬럼데 삽입된 데이터를 확인해보자. 위의 삽입 쿼리에는 ENG_SCORE 컬럼이 제외되었기 때문에 NULL이 들어가야 하지만 0 데이터가 들어간 것을 확인할 수 있다. ENG_SCORE에는 DEFAULT 0 제약조건이 추가되어, 우리가 삽입되는 값을 명시적으로 작성하지 않으면 NULL이 아닌 0 데이터가 들어가게 된 것이다.

만약, 모든 컬럼에 데이터를 추가한다면 컬럼명을 길게 나열하지 않아도 된다.

1번 쿼리

```
INSERT INTO STUDENT (STU_NUM, STU_NAME, KOR_SCORE, ENG_SCORE, INTRO)
VALUES (8, '홍자바', 60, 70, '홍자바입니다');
```

2번 쿼리

```
INSERT INTO STUDENT VALUES (8, '홍자바', 60, 70, '홍자바입니다');
```

1번과 2번의 쿼리를 정확히 같은 결과를 갖는다. 2번 쿼리를 보면 테이블명 다음 컬럼명이 생략된 것을 알 수 있다.

만약, 삽입 시 제외되는 컬럼이 없으며, 테이블 생성 시 작성한 컬럼명과 동일 순으로 데이터를 작성하면 컬럼명 나열을 INSERT쿼리에서 생략할 수 있다.

데이터 수정 쿼리는 UPDATE문이다. 문법은 아래와 같다. COMMIT, ROLLBACK 잊지 말자!

UPADTE TABLE 테이블명 SET 컬럼명 = 변경할 값, 컬러명 = 변경할 값 , ... WHERE 조건;

```
UPDATE TABLE STUDENT
SET
STU_NAME = '홍길동';
```

위 쿼리문은 STUDENT 테이블에서 STU_NAME의 값을 홍길동으로 변경하는 쿼리이다.

그럼 어떤 학생의 이름이 '홍길동 ' 으로 변경되는가. 위의 쿼리문을 실행하면 모든 학생의 이름이 변경됨을 확인 할 수 있다. 그 이유는 WHERE 조건절을 추가하지 않았기 때문이다.

```
UPDATE TABLE STUDENT
SET
STU_NAME = '이순신'
WHERE STU_NUM = 1;
```

위 쿼리문은 학번이 1번인 학생의 이름을 '이순신 ' 으로 변경한다. . UPDATE문의 조건절은 SELECT 쿼리의 WHERE과 동일하다.

```
UPDATE TABLE STUDENT
SET
STU_NAME = '홍길동',
KOR_SCORE = 100
WHERE STU_NUM = 1;
```

위 쿼리문은 학번이 1번인 학생의 이름과 국어점수를 변경하는 쿼리이다. 변경할 컬럼이 다수이면 위 예제처럼 쉼표로 나열하여 작성한다.

데이터 삭제 쿼리는 DELETE문이다. 문법은 아래와 같다. COMMIT, ROLLBACK 잊지 말자! 테이블 삭제와 혼동하지 말자!!!

```
DELETE FROM 테이블명 WHERE 조건;
```

```
DELETE FROM STUDENT;
```

위 쿼리문은 학생 테이블의 모든 데이터를 삭제한다. WHERE 조건이 없기 때문이다... 언능 ROLLBACK 하자..

```
DELETE FROM STUDENT
WHERE KOR_SCORE <= 80;
```

위 쿼리문은 학생 테이블에서 국어점수가 80점 이하인 학생 데이터를 삭제하는 쿼리문이다.