

제3장

SQL 기초-2



1. Insert문

- INSERT 문

- 테이블에 데이터를 삽입하는 명령어

```
INSERT [INTO] 테이블이름[(열1, 열2, ...)] VALUES (값1, 값2, ...)
```

- INSERT 문에서 테이블 이름 다음에 나오는 열 생략 가능(단 열의 순서 및 개수는 동일해야 함)

```
USE sqlldb;  
CREATE TABLE testTBL1 (id int, userName char(3), age int);  
INSERT INTO testTBL1 VALUES (1, '뽀로로', 16);
```

- id와 이름만 입력하고 나이는 입력하고 싶지 않다면

```
INSERT INTO testTBL1 (id, userName) VALUES (2, '크롱');
```

- 열의 순서를 바꾸어 입력하고 싶을 때

```
INSERT INTO testTBL1 (userName, age, id) VALUES ('루피', 14, 3);
```

1. Insert문

■ AUTO_INCREMENT 키워드

- 자동으로 1부터 증가하는 값을 입력하는 키워드
- 특정 열을 AUTO_INCREMENT로 지정할 때는 반드시 **PRIMARY KEY(기본키)** 또는 **UNIQUE(유일한 값)**로 설정해야 함
- 데이터 형식이 숫자인 열에만 사용 가능
- AUTO_INCREMENT로 지정된 열은 INSERT 문에서 NULL 값으로 지정하면 자동으로 값이 입력됨

```
USE sqldb;
CREATE TABLE testTBL2
( id int AUTO_INCREMENT PRIMARY KEY,
  userName char(3),
  age int
);
INSERT INTO testTBL2 VALUES (NULL, '에디', 15);
INSERT INTO testTBL2 VALUES (NULL, '포비', 12);
INSERT INTO testTBL2 VALUES (NULL, '통통이', 11);
SELECT * FROM testTBL2;
```

	id	userName	age
▶	1	에디	15
	2	포비	12
	3	통통이	11
•	NULL	NULL	NULL

■ AUTO_INCREMENT 입력 값을 100부터 시작하도록 변경하고 싶다면

```
ALTER TABLE testTBL2 AUTO_INCREMENT=100;
INSERT INTO testTBL2 VALUES (NULL, '패티', 13);
SELECT * FROM testTBL2;
```

	id	userName	age
▶	1	에디	15
	2	포비	12
	3	통통이	11
	100	패티	13
•	NULL	NULL	NULL

1. Insert문

- AUTO_INCREMENT로 증가되는 값을 지정하기 위해서는 서버 변수인 @@auto_increment_increment 변수 변경(초깃값을 1000으로 하고 증가 값을 3으로 변경하는 구문)

```
USE sqldb;
CREATE TABLE testTBL3
( id int AUTO_INCREMENT PRIMARY KEY,
  userName char(3),
  age int
);
ALTER TABLE testTBL3 AUTO_INCREMENT=1000;
SET @@auto_increment_increment=3;
INSERT INTO testTBL3 VALUES (NULL, '우디', 20);
INSERT INTO testTBL3 VALUES (NULL, '버즈', 18);
INSERT INTO testTBL3 VALUES (NULL, '제시', 19);
SELECT * FROM testTBL3;
```

	id	userName	age
▶	1000	우디	20
	1003	버즈	18
	1006	제시	19
•	NULL	NULL	NULL

- 데이터를 삽입할 때 코드를 줄이려면 여러 행을 한꺼번에 입력

```
INSERT INTO testTBL3 VALUES
(NULL, '토이', 17),
(NULL, '스토리', 18),
(NULL, '무비', 19);
SELECT * FROM testTBL3;
```

	id	userName	age
▶	1000	우디	20
	1003	버즈	18
	1006	제시	19
	1009	토이	17
	1012	스토리	18
	1015	무비	19
•	NULL	NULL	NULL

1. Insert문

- 대량 데이터 삽입 형식

```
INSERT INTO 테이블이름 (열1, 열2, ...)  
SELECT 문;
```

- employees 테이블의 데이터를 가져와 testTBL4 테이블에 입력

```
USE sqlpdb;  
CREATE TABLE testTBL4 (id int, Fname varchar(50), Lname varchar(50));  
INSERT INTO testTBL4  
SELECT emp_no, first_name, last_name FROM employees.employees;
```

Output				
Action Output				
#	Time	Action	Message	Duration / Fetch
1	12:48:52	CREATE TABLE testTBL4 (id int, Fname varchar(50), Lna...	0 row(s) affected	0.094 sec
2	12:48:52	INSERT INTO testTBL4 SELECT emp_no, first_name, l...	300024 row(s) affected Records: 300024 Duplicates: 0 ...	4.187 sec

- 아예 테이블 정의까지 생략하고 싶다면 CREATE TABLE ... SELECT 문 사용

```
CREATE TABLE testTBL5  
(SELECT emp_no, first_name, last_name FROM employees.employees);  
SELECT * FROM testTBL5 LIMIT 3;
```

	emp_no	first_name	last_name
▶	10001	Georgi	Facello
	10002	Bezael	Simmel
	10003	Parto	Bamford

1. Insert문

- **CREATE TABLE ... SELECT** 문에서 열 이름을 바꾸어 테이블을 생성하려면

```
CREATE TABLE testTBL6  
  (SELECT emp_no AS id, first_name AS Fname, last_name AS Lname  
   FROM employees.employees);  
SELECT * FROM testTBL6 LIMIT 3;
```

	id	Fname	Lname
▶	10001	Georgi	Facello
	10002	Bezalel	Simmel
	10003	Parto	Bamford

2. Update문

- UPDATE 문

- 테이블에 입력되어 있는 값을 수정하는 명령어

```
UPDATE 테이블이름  
SET 열1=값1, 열2=값2, ...  
WHERE 조건;
```

- 'Kyoichi'의 Lname을 '없음'으로 수정

```
USE sqldb;  
UPDATE testTBL4  
SET Lname = '없음'  
WHERE Fname = 'Kyoichi';
```

- 전체 테이블의 내용을 수정하고 싶을 때는 WHERE 절 생략

```
UPDATE buyTBL  
SET price = price * 1.5;
```

3. Delete문

- **DELETE 문**

- 테이블에 데이터를 행 단위로 삭제하는 명령어

```
DELETE FROM 테이블이름 WHERE 조건;
```

- **DELETE 문에서 WHERE 절을 생략하면 테이블에 저장된 전체 데이터가 삭제**

```
USE cookDB;  
DELETE FROM testTBL4 WHERE Fname = 'Aamer';
```

- **Aamer 중에서 상위 몇 건만 삭제하고자 할 때는 추가로 LIMIT 절 사용**

```
DELETE FROM testTBL4 WHERE Fname = 'Aamer' LIMIT 5;
```


4. 실습 내용

1 대용량 테이블 생성

1-1 대용량 테이블 3개 생성

```
USE sqldb;  
CREATE TABLE bigTBL1 (SELECT * FROM employees.employees);  
CREATE TABLE bigTBL2 (SELECT * FROM employees.employees);  
CREATE TABLE bigTBL3 (SELECT * FROM employees.employees);
```

2 데이터 삭제하기

2-1 DELETE, DROP, TRUNCATE 문으로 3개의 테이블 삭제

```
DELETE FROM bigTBL1;  
DROP TABLE bigTBL2;  
TRUNCATE TABLE bigTBL3;
```

3 결과 확인하기

3-1 [Output] 창의 결과에서 실행 시간 확인

Output				
Action Output				
#	Time	Action	Message	Duration / Fetch
✓ 1	14:25:12	<u>DELETE</u> FROM bigTBL1	300024 row(s) affected	3.265 sec
✓ 2	14:25:15	<u>DROP</u> TABLE bigTBL2	0 row(s) affected	0.094 sec
✓ 3	14:25:15	<u>TRUNCATE</u> TABLE bigTBL3	0 row(s) affected	0.203 sec

4. 실습 내용

- 실행 시간을 고려한 테이블 삭제 방법
 - 대용량 테이블 전체 내용을 삭제할 때 테이블 자체가 필요 없는 경우에는 **DROP** 문 사용
 - 테이블의 구조를 남겨놓고 싶은 경우에는 **TRUNCATE** 문으로 삭제

4. 실습 내용

1 새 테이블 생성하기

1-1 멤버 테이블(memberTBL) 새로 만들고 데이터 삽입

```
USE sqlldb;  
CREATE TABLE memberTBL (SELECT userID, userName, addr FROM userTBL LIMIT 3); -- 3건만 가져옴  
ALTER TABLE memberTBL  
ADD CONSTRAINT pk_memberTBL PRIMARY KEY (userID); -- 기본키 지정  
SELECT * FROM memberTBL;
```

	userID	userName	addr
▶	KHD	강호동	경북
	KJD	김제동	경남
	KKJ	김국진	서울
•	NULL	NULL	NULL

2 오류가 발생해도 계속 삽입되도록 설정하기

2-1 첫 번째 데이터에서 기본키를 중복 입력하는 실수 범하기

```
INSERT INTO memberTBL VALUES ('KHD', '강후덜', '미국'); -- 기본키 중복 입력  
INSERT INTO memberTBL VALUES ('LSM', '이상민', '서울');  
INSERT INTO memberTBL VALUES ('KSJ', '김성주', '경기');
```

Output				
Action Output				
#	Time	Action	Message	Duration / Fetch
1	14:49:34	INSERT INTO memberTBL VALUES('KHD', '강후덜', '미	Error Code: 1062. Duplicate entry 'KHD' for key 'PRIMARY'	0.000 sec

4. 실습 내용

2-2 SELECT * FROM memberTBL ; 문으로 조회

2-3 기존의 INSERT INTO 문을 **INSERT IGNORE INTO**문으로 수정한 후 다시 실행

Insert ignore문은 기본키가 중복되더라도 무시하고 넘어가며 오류 메시지만 띄운다.

```
INSERT IGNORE INTO memberTBL VALUES ('KHD', '강후덜', '미국');  
INSERT IGNORE INTO memberTBL VALUES ('LSM', '이상민', '서울');  
INSERT IGNORE INTO memberTBL VALUES ('KSJ', '김성주', '경기');  
SELECT * FROM memberTBL;
```

The screenshot shows a database management tool interface. At the top, there's a toolbar with options like 'Result Grid', 'Filter Rows', 'Edit', 'Export/Import', and 'Wrap Cell Contents'. Below the toolbar is a table with columns 'userID', 'userName', and 'addr'. The table contains five rows: KHD, KJD, KKJ, KSJ, and LSM. The LSM row is highlighted with a red border. Below the table, there's a tab labeled 'memberTBL 13'. To the right of the table, there's a 'Result Grid' button and a 'Form Editor' button. Below the table, there's an 'Output' section with a tab labeled 'Action Output'. The 'Action Output' section shows a list of actions and their results. The first action is 'INSERT IGNORE INTO memberTBL VALUES('KHD', '...', which resulted in '0 row(s) affected, 1 warning(s): 1052 Duplicate entry 'KH...'. The second action is 'INSERT IGNORE INTO memberTBL VALUES('LSM', '이...', which resulted in '1 row(s) affected'. The third action is 'INSERT IGNORE INTO memberTBL VALUES('KSJ', '김...', which resulted in '1 row(s) affected'. The fourth action is 'SELECT * FROM memberTBL', which resulted in '5 row(s) returned'.

#	Time	Action	Message	Duration / Fetch
1	14:52:41	INSERT IGNORE INTO memberTBL VALUES('KHD', '...	0 row(s) affected, <u>1 warning(s): 1052 Duplicate entry 'KH...</u>	0.000 sec
2	14:52:41	INSERT IGNORE INTO memberTBL VALUES('LSM', '이...	1 row(s) affected	0.094 sec
3	14:52:41	INSERT IGNORE INTO memberTBL VALUES('KSJ', '김...	1 row(s) affected	0.015 sec
4	14:53:09	SELECT * FROM memberTBL	5 row(s) returned	0.000 sec / 0.000 sec

4. 실습 내용

3 기본키가 중복되면 새로 삽입한 내용으로 수정하기

3-1 데이터를 삽입할 때 기본키가 중복되면 새로 삽입한 데이터로 내용이 변경되게 하기

ON DUPLICATE KEY UPDATE구문은 기본키가 중복되지 않으면, 일반 **INSERT**문으로 작동하고, 기본키가 중복되면 그 뒤의 **UPDATE**구문을 실행한다.

```
INSERT INTO memberTBL VALUES ('KHD', '강후달', '미국')
ON DUPLICATE KEY UPDATE userName='강후달', addr='미국';
INSERT INTO memberTBL VALUES ('DJM', '동짜몽', '일본')
ON DUPLICATE KEY UPDATE userName='동짜몽', addr='일본';
SELECT * FROM memberTBL;
```

	userID	userName	addr
▶	DJM	동짜몽	일본
	KHD	강후달	미국
	KJD	김재동	경남
	KKJ	김국진	서울
	KSJ	김성주	경기
	LSM	이상민	서울
•	NULL	NULL	NULL

감사합니다.

