

1 REM Author : Henry
2 REM Date : 2024.06.12
3 REM Objective :
4 REM Environment : Ubuntu Server 22.04 LTS, MySQL Workbench 8.0 CE, MySQL Community
Server 8.0.37-0ubuntu0.22.04.3 (ubuntu)

6 REM 데이터 조작(DML)
7 -데이터베이스에 데이터를 추가, 갱신, 삭제할 때 사용

10 REM INSERT 문

- 11 테이블에 새 행 삽입할 때 사용
- 12 Syntax

```
INSERT INTO table_name [ (column1[, column2[...]])]  
VALUES (value1, [, value2[...]]);
```

```
VALUES (99, '충무과', '서울', 98, '인사과', '대전');
```

17 3. Guideline

- 18 1)VALUES 절에서는 동시에 한개의 레코드만 삽입된다.
- 19 2)테이블 이름 뒤에 특정 칼럼을 지정하지 않으면 반드시, 스키마 순서대로 VALUES에서
사용해야 한다.
- 20 3)명확한 Query문을 위해 가급적이면 컬럼이름을 지정하는 것이 좋다.(권장)
- 21 4)문자형과 날짜형은 반드시 단일 따옴표를 사용하고, 숫자형은 단일 따옴표를 사용하지
않는다.
- 22 5)각 열의 값을 포함하는 새 행을 삽입할 수 있으므로 INSERT 절에 열 목록이 필요 없지만 열
목록을 사용하지 않는 경우에는 값을 테이블의 기본 열 순서에 따라 나열해야 한다.
- 23 6)테이블 리스트에 있는 칼럼 갯수와 VALUES 절의 값 갯수는 일치해야 한다.

```
INSERT INTO dept(deptno, dname)  
VALUES (99, '충무과', '서울');
```

- 27 7)입력될 값의 데이터 타입은 칼럼의 데이터 타입과 일치해야 한다.

```
INSERT INTO dept(deptno, dname)  
VALUES ('99', '충무과');
```

- 31 8)입력될 값의 크기는 칼럼의 크기보다 크지 않아야 한다.

```
INSERT INTO dept(deptno, dname)  
VALUES (999, '충무과');
```

- 35 9)NULL 값에 주의하자.

- 36 10)Foreign Key에 주의하자.

- 37 11)오직 한번에 하나의 행만 입력할 수 있다.

```
INSERT INTO dept  
VALUES (99, '충무과', '서울', 98, '인사과', '대전');
```

```
INSERT INTO dept(deptno, dname, loc)  
VALUES(50, 'DEVELOPMENT', 'SEOUL');
```

46 REM NULL 값을 갖는 행 삽입

- 47 1. 암시적 방법

```
48 INSERT INTO dept(deptno, dname)
49 VALUES (60, 'MIS');
```

51 2. 명시적 방법

```
52 INSERT INTO dept
53 VALUES (70, 'FINANCE', NULL);
```

57 REM NOW 함수 또는 CURDATE(), CURTIME()를 사용하여 현재 날짜 및 시간 삽입

```
58 INSERT INTO emp (empno, ename, job, mgr, hiredate, sal, comm, deptno)
59 VALUES (7196, 'GREEN', 'SALESMAN', 7782, CURDATE(), 2000, NULL, 10);
```

```
60
61 CREATE TABLE emp_copy
```

```
62 AS
```

```
63 SELECT *
```

```
64 FROM emp;
```

```
65
66 --Database의 Character set 변경, Table Collation 변경 --> utf8_general_ci
```

```
68 INSERT INTO emp_copy(empno, ename, job, mgr, hiredate, sal, comm, deptno)
69 VALUES(9999, 'Sujan', '', NULL, SYSDATE, 3000, NULL, 10);
```

```
70
71 INSERT INTO emp_copy
```

```
72 VALUES(8888, '홍길동', '', NULL, CURDATE(), 4000, NULL, 20)
```

```
73
74 INSERT INTO emp_copy(empno, ename, hiredate, deptno)
```

```
75 VALUES(7777, '백두산', CURDATE(), 40);
```

```
76
77 INSERT INTO emp_copy(empno, ename, hiredate, deptno)
```

```
78 VALUES (6666, '한라산', STR_TO_DATE('20080501', '%Y%m%d'), 30);
```

```
80
81
82 REM 다른 테이블로부터 행 복사
```

```
83 CREATE TABLE emp_clone
```

```
84 AS
```

```
85 SELECT empno, ename, sal, hiredate
```

```
86 FROM emp
```

```
87 WHERE 1 < 0;
```

```
88
89
90 --사원테이블에서 부서번호 10번의 레코드만 emp_clone 으로 복사하시오.
```

```
91 INSERT INTO emp_clone(empno, ename, sal, hiredate)
```

```
92 SELECT empno, ename, sal, hiredate
```

```
93 FROM emp
```

```
94 WHERE deptno = 10;
```

```
95
96
97
98 REM UPDATE 문
```

```
UPDATE
```

```
NOW()      X
NOW()      - - - -
           가   가   .(      )
=> CURDAET()
```

```
UTF8mb4
```

- 99 1. 기본 행 수정
100 2. 필요한 경우 한번에 여러 행 갱신 가능
101 3. Syntax

102
103 **UPDATE** table_name
104 **SET** column = value [, column = value, ...]
105 **[WHERE** condition];

106
107
108 **UPDATE** emp
109 **SET** deptno = 20
110 **WHERE** empno = 7782;

111
112 **UPDATE** emp
113 **SET** deptno = 20;

114
115
116
117 REM 무결성 제약 조건 오류
118 -무결성 제약 조건의 영향을 받는 값을 포함하는 레코드를 갱신하면 오류가 발생

119 **UPDATE** emp
120 **SET** deptno = 55
121 **WHERE** deptno = 10;

122
123
124
125 REM **DELETE** 문

- 126 1. 기존 행 제거

- 127 2. Syntax

128 **DELETE** **[FROM]** table_name
129 **[WHERE** condition];

130
131 **DELETE** **FROM** dept
132 **WHERE** dname = 'DEVELOPMENT';

133
134 **DELETE** **FROM** emp;

135
136
137
138 REM 무결성 제약 조건 오류
139 -무결성 제약 조건의 영향을 받는 값을 포함하는 레코드를 삭제하면 오류가 발생

140
141 **DELETE** **FROM** dept
142 **WHERE** deptno = 10;

143
144
145 --emp_copy테이블에서 1987년에 입사한 사원을 제거하시오.

146 **DELETE** **FROM** emp_copy
147 **WHERE** YEAR(hiredate) = '1987';

148
149

150
151 REM DML과 TRANSACTION
152 1. DDL 명령어인 경우에는 직접 Database의 table에 영향을 미치기 때문에 DDL명령어를 입력
하는 순간 명령어에 해당하는 작업이 즉시(AUTO COMMIT) 완료된다.
153 2. 하지만, DML 명령어의 경우, 조작하려는 table을 memory buffer에 올려놓고 작업을 하기
때문에 실시간으로 table에 영향을 미치는 것이 아니다.
154 3. 따라서 buffer에서 처리한 DML 명령어가 실제 table에 반영되기 위해서는 COMMIT 명령어를
입력하여 Transaction을 종료해야 한다.
155 4. table의 전체 data를 삭제하는 경우, System 활용 측면에서는 삭제된 data를 log로 저장하는
DELETE TABLE 보다는 System 부하가 적은 TRUNCATE TABLE을 권고한다.
156 5. 단, TRUNCATE TABLE의 경우 삭제된 data의 log가 없으므로 ROLLBACK이 불가능하므로
주의해야 한다.