

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)

5

6

7 REM MySQL Objects

8 1. 종류

9 TABLE, VIEW, INDEX, Stored Procedure, Stored Function, Trigger, Cursor

10

11 2. Data Dictionary

12 -INFORMATION_SCHEMA Database

13

14 3. INFORMATION_SCHEMA

15 -Database Metadata에 대한 액세스

16 -Database 또는 Table 이름, 열의 데이터 유형 또는 액세스 권한과 같은 MySQL 서버에
대한 정보 제공

17 -데이터 사전 및 시스템 카탈로그라고도 한다.

18

19 4. SHOW Statements 가능

20 -SHOW CHARACTER SET

21 -SHOW COLLATION

22 -SHOW COLUMNS

23 -SHOW DATABASES

24 -SHOW FUNCTION STATUS

25 -SHOW INDEX

26 -SHOW OPEN TABLES

27 -SHOW PROCEDURE STATUS

28 -SHOW STATUS

29 -SHOW TABLE STATUS

30 -SHOW TABLES

31 -SHOW TRIGGERS

32 -SHOW VARIABLES

33

34 --SHOW CHARACTER SET;

35 --SHOW CHARACTER SET WHERE `Default collation` LIKE '%korean%';

36

37 5. INFORMATION_SCHEMA Tables

38 1) CHARACTER SET

39 -Available character sets

40

41 2) COLUMN_PRIVILEGES

42 -Privileges defined on columns

43

44 3) COLUMNS

45 -Columns in each table

46

47 4) GLOBAL_VARIABLES

48 -Global system variables

49

5) PARAMETERS

-Stored **routine parameters and** stored **function return values**

6) ROUTINES

-Stored **routine** information

7) SCHEMA_PRIVILEGES

-**Privileges** defined **on** schemas

8) SESSION_VARIABLES

-System variables **for current session**

9) TABLE_CONSTRAINTS

-Which tables have **constraints**

10) TABLE_PRIVILEGES

-**Privileges** defined **on** tables

11) TABLES

-**Table** information

12) TRIGGERS

-**Trigger** information

13) USER_PRIVILEGES

-**Privileges** defined globally per **user**

14) VIEWS

-**View** information

REM **View**

1. 테이블 뷰를 통한 데이터의 **논리적** 부분 집합 또는 조합

2. **논리** 테이블

3. **자체적**으로 데이터를 갖고 있지 않다.

4. 데이터를 보거나 변경할 수 있는 **창**이다.

5. 뷰의 기반이 되는 테이블을 기본 테이블이라 한다.

6. **Data Dictionary**에 **SELECT**문으로 저장

7. 공간을 **차지**하지도 않는다.



REM **View** 의 목적

1. 데이터베이스의 선택적인 내용을 보여줄 수 있기 때문에 데이터베이스에 대한 액세스를 제한한다. --> 보안에 도움이 된다.

2. 복잡한 질의어를 통해 얻을 수 있는 결과를 간단한 질의어를 써서 구할 수 있게 한다. --> 성능향상

3. 데이터 독립성을 허용한다.

4. 동일한 데이터의 다른 **VIEW**를 나타낸다.

5. 조인을 한 것처럼 여러 테이블에 대한 데이터를 **VIEW**를 통해 볼 수 있다.

6. 한개의 **VIEW**로 여러 테이블에 대한 데이터를 검색할 수 있다.

99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148

REM View 종류

1. 단순뷰(Simple View)

- 1) 오직 하나의 테이블에서만 데이터를 가져온다.
- 2) 그룹 이나 다중행 함수를 포함하지 않는다.
- 3) 뷰를 이용해서 DML 을 수행할 수 있다.
- 4) DISTINCT 사용 불가능.



2. 복합뷰(Complex View)

- 1) 다중 테이블에서 데이터를 가져온다.
- 2) 그룹, 다중행 함수를 포함한다.
- 3) DML 문장을 수행할 수 없다.
- 4) DISTINCT 사용 가능.



REM View Syntax

CREATE [OR REPLACE] VIEW view_name(alias,...)

AS

Subquery

[WITH CHECK OPTION]

--OR REPLACE : 기존에 존재하는 뷰가 있다면 삭제하고 새로 만든다.

--WITH CHECK OPTION : 서브쿼리 내의 조건을 만족하는 행만 변경 가능

REM VIEW Guide Lines

1. 뷰를 정의하는 하위 절의는 조인, 그룹, 하위 절의 등의 복합 **SELECT** 구문을 포함할 수 있다.
2. 뷰를 정의하는 하위 절의는 **ORDER BY** 절을 포함할 수 없다. **ORDER BY** 절은 뷰에서 데이터를 검색할 때 지정.
3. View 를 수정할 때에는 **ALTER**를 사용하지 않고, **OR REPLACE**를 사용한다.
4. **VIEW** 의 구조를 볼 때는 **DESC** 사용.

CREATE VIEW VIEW_TEST

AS

SELECT * FROM TEST; --ERROR : TEST 테이블이 없음.

CREATE VIEW empview10

AS

SELECT empno, ename, job

FROM emp

WHERE deptno = 10;

DESC empview10 --VIEW 구조보기

SELECT * FROM empview10; --View를 이용한 데이터 조회

CREATE VIEW EMP20

```

149 AS
150 SELECT EMPNO, ENAME, SAL
151 FROM EMP
152 WHERE DEPTNO = 20;
153
154 DESC EMP20;
155
156 SELECT * FROM EMP20;
157
158
159 CREATE OR REPLACE VIEW EMP20(ENO, NAME, PAYROLL)
160 AS
161 SELECT EMPNO, ENAME, SAL
162 FROM EMP
163 WHERE DEPTNO = 20;
164
165
166
167 REM Data Dictionary에서 View정보보기
168 DESC INFORMATION_SCHEMA.VIEWS;
169
170 SELECT * FROM INFORMATION_SCHEMA.VIEWS
171 WHERE TABLE_NAME = 'emp20';
172
173
174 CREATE VIEW EMP_30_VU
175 AS
176 SELECT EMPNO, ENAME, SAL, DEPTNO
177 FROM EMP
178 WHERE DEPTNO = 30;
179
180 DESC EMP_30_VU;
181
182 INSERT INTO EMP_30_VU
183 VALUES(1111, 'Jimin', 500, 30);
184
185 SELECT * FROM EMP_30_VU;
186 SELECT * FROM EMP; --VIEW에 추가한 것이 실제 기본 테이블에도 반영됨.
187
188
189 REM View 수정
190 1. OR REPLACE 옵션을 사용 view가 view ,
191 2. 이미 뷰가 있더라도 뷰를 생성하여 해당 뷰를 대체
192 CREATE OR REPLACE VIEW empview10
193 (employee_number, employee_name, job_title)
194 AS
195 SELECT empno, ename, job
196 FROM emp
197 WHERE deptno = 10;
198
199

```

200 REM **VIEW** 실습
201 1. 부서별로 부서명, 최소 급여, 최대 급여, 부서의 평균 급여를 포함하는 DEPT_SUM **View** 를
생성하라.

```
202 CREATE OR REPLACE VIEW dept_sum(deptno, tmin, tmax, tavg)  
203 AS  
204 SELECT deptno, MIN(sal), MAX(sal), AVG(sal)  
205 FROM emp  
206 GROUP BY deptno;
```

207
208
209 2. emp **table**에서 직원번호, 이름, 업무를 포함하는 emp_view **VIEW**를 생성하시오.

```
210 CREATE OR REPLACE VIEW emp_view(직원번호, 이름, 업무)  
211 AS  
212 SELECT empno, ename, job  
213 FROM emp;
```

214
215
216 3. 위 2번에서 생성한 **VIEW**를 이용하여 10번 부서의 자료만 조회하시오

```
217 CREATE OR REPLACE VIEW emp_view  
218 (직원번호, 이름, 업무)  
219 AS  
220 SELECT empno, ename, job  
221 FROM emp  
222 WHERE deptno = 10;
```

INFORMATION_SCHEMA.VIEWS	DB	VIEWS	mysql
view			

223
224
225 4. 위 2번에서 생성한 **VIEW**를 **Data Dictionary** 에서 조회하시오.

```
226 SELECT * FROM INFORMATION_SCHEMA.VIEWS  
227 WHERE TABLE_NAME = 'emp_view';
```

228
229
230 5. 이름, 업무, 급여, 부서명, 위치를 포함하는 emp_dept_name 이라는 **VIEW**를 생성하시오.

```
231 CREATE OR REPLACE VIEW emp_dept_name  
232 AS  
233 SELECT ename, job, sal, dname, loc  
234 FROM emp, dept  
235 WHERE emp.deptno = dept.deptno;
```

236
237
238 6. 87년에 입사한 사람을 볼 수 있는 뷰

```
239 CREATE OR REPLACE VIEW view_emp_87  
240 (sabun, name, hiredate)  
241 AS  
242 SELECT empno, ename, hiredate  
243 FROM emp  
244 WHERE YEAR(hiredate) = '1987';
```

245
246
247 7.부서별로 부서명, 최소급여, 최대급여, 부서별 평균급여를 포함하는 view_dept_sum 뷰를
생성하시오.

```
248 CREATE OR REPLACE VIEW view_dept_sum(v_dname, v_min_sal, v_max_sal, v_avg_sal)
```

```

249 AS
250 SELECT dname, MIN(sal), MAX(sal), AVG(sal)
251 FROM emp, dept
252 WHERE emp.deptno = dept.deptno
253 GROUP BY dept.dname;
254
255
256 REM 복합뷰
257 --두개 이상의 테이블로 부터 값을 출력하고, 그룹함수를 포함하는 복잡한 VIEW
258
259 --사원이름, 업무, 급여, 부서명, 위치를 포함하는 view_emp_dept 뷰를 생성하시오.
260 CREATE OR REPLACE VIEW view_emp_dept
261 AS
262 SELECT ename AS "사원이름", job AS "업무", sal AS "급여", dname AS "부서명", loc AS
"부서의 위치"
263 FROM emp, dept
264 WHERE emp.deptno = dept.deptno AND emp.deptno = 10;
265
266
267 CREATE TABLE dept_clone
268 AS
269 SELECT *
270 FROM dept;
271
272 ALTER TABLE dept_clone
273 ADD CONSTRAINT dept_clone_deptno_pk PRIMARY KEY(deptno);
274
275 ALTER TABLE dept_clone
276 MODIFY dname VARCHAR(14) NOT NULL;
277
278 CREATE OR REPLACE VIEW view_dept_clone
279 AS
280 SELECT deptno, loc
281 FROM dept_clone;
282
283 INSERT INTO DEPT_CLONE
284 VALUES(50, 'SEOUL'); --ERROR
285
286
287 REM WITH CHECK OPTION 절 사용하기
288 --사원테이블과 동일한 emp_20(20번부서만)이라는 뷰를 생성하되, WITH CHECK OPTION 을
사용해서 생성하시오.
289
290 CREATE OR REPLACE VIEW emp_20
291 AS
292 SELECT * FROM emp with check option where
293 WHERE deptno = 20
294 WITH CHECK OPTION CONSTRAINT emp_20_ck;
295
296 UPDATE emp_20
297 SET deptno = 30

```

dept	50
------	----

```

298 WHERE empno = 7566; : CHECK OPTION Failed
299
300
301 REM Limit ORDER BY , N
302 1. 테이블에서 조건에 대한 최상위 레코드 N개 또는 최하위 레코드 N개를 표시
303 2. Syntax
304 SELECT
305 FROM
306 LIMIT N;
307
308 SELECT empno, ename, hiredate
309 FROM emp
310 ORDER BY hiredate
311 LIMIT 3;
312
313
314 SET @ROWNUM :=0;
315 SELECT @ROWNUM := @ROWNUM + 1 AS rank, empno, ename, hiredate
316 FROM emp
317 ORDER BY hiredate
318 LIMIT 3;
319
320 --emp table에서 가장 최근에 입사한 5명의 사원번호, 사원명, 입사날짜를 출력하시오.
321
322
323 REM View 제거
324 1.뷰가 삭제되도 기본 테이블의 데이터에는 영향이 없음.
325 2. Syntax
326 DROP VIEW view_name;
327
328 DROP VIEW empview10;
329
330
331
332 REM INDEX
333 1. 행에 대한 빠른 참조를 위해서, 테이블에 인덱스를 생성할 수 있다.
334
335 2. 장/단점
336 1)장점
337 -검색 속도가 빨라진다.
338 -시스템에 부하를 줄여서 시스템 전체 성능을 향상시킨다.
339 2)단점
340 -인덱스를 위한 추가 공간이 필요하다.
341 -인덱스를 생성하는 데 시간이 걸린다.
342 -데이터의 변경작업이 자주 일어나는 경우에는 오히려 성능이 더 떨어진다.
343
344 3. 생성해야 할 조건
345 1)테이블의 행의 수가 많다. 아주 작은 크기의 테이블에는 오히려 성능이 떨어진다.
346 2)사용자의 SQL 문에서 WHERE 조건절에 자주 사용되는 칼럼이 대상이 된다.
347 3)WHERE 조건에 의한 결과가 전체 행수의 비율(분포도) 2~4% 인 경우에 효과가 있다.
348 4)분포도가 범위 이상이라도 일부분의 데이터 검색이라면 적용가능

```

5) JOIN 에 자주 사용되는 컬럼이나 NULL 을 포함하는 컬럼이 많은 경우

4. 생성하지 않아야 할 조건

1) 테이블에 행이 적은 경우

2) 컬럼이 WHERE 조건에 자주 사용되지 않을 때

3) WHERE 조건에 의한 결과가 전체 행에 대해 10~15%의 결과보다 높게 리턴될 때

4) 테이블이 자주 입력, 수정, 삭제 될 때는 오히려 검색 속도가 더 떨어진다.

5. INDEX Type

PRIMARY KEY UNIQUE

1) UNIQUE index : 지정된 열의 값이 고유해야 한다.

```
CREATE UNIQUE INDEX index_name  
ON table_name(column1, column2, ...)
```

-Index 확인

```
SHOW INDEX FROM table_name;
```

```
CREATE TABLE dept1
```

```
AS
```

```
SELECT * FROM dept
```

```
WHERE 0 = 1;
```

```
INSERT INTO dept1 VALUES(10, 'ACCOUNTING', 'SEOUL');
```

```
INSERT INTO dept1 VALUES(20, 'SALES', 'PUSAN');
```

```
INSERT INTO dept1 VALUES(30, 'OPERATION', 'PUSAN');
```

```
INSERT INTO dept1 VALUES(40, 'IT', 'DAEJUN');
```

```
CREATE UNIQUE INDEX idx_dept1_deptno ON dept1(deptno); --SUCCESS
```

```
CREATE UNIQUE INDEX idx_dept1_loc ON dept1(loc); --ERROR
```

```
CREATE INDEX idx_dept1_loc ON dept1(loc); --SUCCESS. UNIQUE 를 빼면 됨.
```

2) Non-unique index : 가장 빠름을 보장하는 컬럼, 컬럼의 값이 고유하지 않을 때

```
CREATE INDEX i_emp_ename ON emp(ename);
```

UNIQUE INDEX

3) Single column index : 하나의 컬럼에만 인덱스를 부여

```
CREATE INDEX i_emp_ename ON emp(ename);
```

4) Composite Index : 여러 컬럼에 인덱스부여가능

```
CREATE INDEX l_emp_empno_ename ON emp(empno, ename);
```

6. INDEX 확인

```
SHOW INDEX FROM table_name;
```

```
SHOW INDEX FROM emp;
```


400 7. INDEX 의 제거

401 ALTER TABLE table_name

402 DROP INDEX index_name;

403

404 OR

405 DROP INDEX index_name ON table_name;

406

407

408 8. INDEX 의 수정

409 1)인덱스는 수정할 수 없다.

410 2)수정하기 위해서는 제거하고 새로 생성해야 한다.