SQL

✓ 원리를 알면 IT가 맛있다

SQL for Beginners

chapter 11.

기타 스키마 객체

❑ 학습목표

- 뷰 (view)
- 시퀀스 (sequence)
- 인덱스 (index)

❑ 1) 뷰 [view]

- 뷰 (VIEW) 정의

  - 테이블 또는 다른 뷰를 기초로 하는 논리적 테이블.
  - 뷰는 그 자체로서 소유하는 데이터는 없지만, 창문처럼 어떤 데이터를 보거나 변경할 수 있다.
  - 뷰에서 참조하는 테이블을 기본 테이블(Base Table) 이라고 한다.

- 뷰 (VIEW) 사용 목적 및 특징

  - 데이터베이스에서 선택적으로 데이터를 보여줄 수 있기 때문에, 데이터베이스에 대한 접근을 제한 할 수 있다.
  - 복잡한 질의로부터 결과를 검색하기 위한 단순한 질의를 만들 수 있다.
  - 하나의 뷰는 여러 개의 테이블로부터 데이터를 검색하는데 사용 가능하다.
  - 조인을 한 것처럼 여러 테이블에 대한 데이터를 VIEW을 통해볼 수 있다

- 뷰 (VIEW) 종류

  - 단순 뷰(simple view) : 1개의 테이블로 구성
  - 복합 뷰(complex view) : 여러 개의 테이블로 구성

❑ 1) 뷰 [view]

- 뷰 (VIEW) 작성법

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW view
      [(alias[, alias] ...)]
AS subquery
[WITH CHECK OPTION [CONSTRAINT constraint]]
[WITH READ ONLY [CONSTRAINT constraint]];
```

* Force는 기본테이블이 없어도 강제로 뷰 생성.

- 뷰 (VIEW) 작성시 주의할 점.

: subquery는 조인, set 연산, 서브쿼리가 포함된 복잡한 SELECT 문이 정의 가능하다.
: subquery에는 **ORDER BY 절을 사용할 수 없다.**
ORDER BY를 사용하려면 검색시 뷰에 기술한다.
: 뷰를 수정하기 위해서는 CREATE OR REPLACE 을 이용한다.
: CREATE VIEW 권한을 가져야 뷰를 생성할 수 있다.

- CREATE VIEW 권한 할당

```
SQL> conn / as sysdba
연결되었습니다.
SQL> GRANT CREATE VIEW TO SCOTT;

권한이 부여되었습니다.

SQL> conn scott/tiger
연결되었습니다.
```

❑ 1) 뷰 [view]

- 뷰 (VIEW) 생성

```
SQL> CREATE VIEW EMP_VIEW
  2  AS
  3  SELECT EMPNO, ENAME, SAL, HIREDATE
  4  FROM EMP
  5  WHERE DEPTNO = 10;

뷰가 생성되었습니다.
```

```
SQL> SELECT *
  2  FROM EMP_VIEW;

     EMPNO ENAME                       SAL HIREDATE
---------- ------------------ ---------- --------
      7782 CLARK                     2450 81/06/09
      7839 KING                      5000 81/11/17
      7934 MILLER                    1300 82/01/23
```

```
SQL> CREATE VIEW DEPT_VIEW
  2  AS
  3  SELECT DEPTNO NO , DNAME name
  4  FROM DEPT;

뷰가 생성되었습니다.

SQL> DESC DEPT_VIEW;
 이름                                      널?      유형
 ----------------------------------------- -------- ------------

 NO                                        NOT NULL NUMBER(2)
 NAME                                               VARCHAR2(14)
```

```
SQL> SELECT VIEW_NAME
  2  FROM USER_VIEWS;

VIEW_NAME
------------------------
EMP_VIEW
DEPT_VIEW
```

- 뷰 (VIEW) 수정
  : CREATE OR REPLACE 명령 이용한다.

```
SQL> CREATE OR REPLACE VIEW DEPT_VIEW
  2  AS
  3  SELECT DEPTNO NO, DNAME NAME, LOC
  4  FROM DEPT;

뷰가 생성되었습니다.

SQL> DESC DEPT_VIEW;
 이름                                      널?      유형
 ----------------------------------------- -------- ------------

 NO                                        NOT NULL NUMBER(2)
 NAME                                               VARCHAR2(14)
 LOC                                                VARCHAR2(13)
```

- 복합 뷰 (VIEW) 생성

```
SQL> CREATE VIEW EMP_DEPT_VIEW
  2  AS
  3  SELECT EMPNO, ENAME , DNAME
  4  FROM EMP , DEPT
  5  WHERE EMP.DEPTNO = DEPT.DEPTNO
  6  AND DEPT.DEPTNO = 30;

뷰가 생성되었습니다.
```

```
SQL> SELECT * FROM EMP_DEPT_VIEW;

     EMPNO ENAME                DNAME
---------- -------------------- ------
      7499 ALLEN                SALES
      7521 WARD                 SALES
      7654 MARTIN               SALES
      7698 BLAKE                SALES
      7844 TURNER               SALES
      7900 JAMES                SALES
```

❑ 1) 뷰 [view]

- 뷰 (VIEW) 에서 DML 작업
  : 단순 뷰에서 DML 연산 수행 가능 하다.
  : 뷰가 다음을 포함한다면 행을 제거할 수 없다.
  - 그룹함수
  - GROUP BY 절
  - DISTINCT 키워드

  : 뷰가 다음을 포함한다면 데이터를 수정할 수 없다.
  - 위의 임의의 조건
  - ROWNUM 의사열
  - 표현식으로 정의된 열 (예: SAL*12)

  : 뷰가 다음을 포함한다면 데이터를 추가할 수 없다.
  - 위의 임의의 조건
  - 뷰에 의해 선택되지 않은 NOT NULL 열이 기본테이블에 있을 때

```
SQL> SELECT * FROM EMP_VIEW;

     EMPNO ENAME                     SAL HIREDATE
---------- ------------------ ---------- --------
      7782 CLARK                    2450 81/06/09
      7839 KING                     5000 81/11/17
      7934 MILLER                   1300 82/01/23
```

```
SQL> DELETE FROM EMP_VIEW
  2  WHERE EMPNO = 7782;

1 행이 삭제되었습니다.
```

```
SQL> SELECT * FROM EMP_VIEW;

     EMPNO ENAME                     SAL HIREDATE
---------- ------------------ ---------- --------
      7839 KING                     5000 81/11/17
      7934 MILLER                   1300 82/01/23
```

- 뷰 (VIEW) 의 제약 조건
  : WITH CHECK OPTION
  - WHERE 조건에 만족하는 데이터만이 INSERT , UPDATE 작업을 수행할 수 있다.

```
SQL> CREATE OR REPLACE VIEW EMP_VIEW
  2  AS
  3  SELECT * FROM EMP
  4  WHERE DEPTNO =10
  5  WITH CHECK OPTION CONSTRAINT EMP_VIEW10_CHCEK;
```

```
SQL> UPDATE EMP_VIEW
  2  SET DEPTNO = 20 WHERE ENAME = 'KING';
UPDATE EMP_VIEW
       *
1행에 오류:
ORA-01402: 뷰의 WITH CHECK OPTION의 조건에 위배 됩니다
```

  : WITH READ ONLY
  - 뷰를 통한 DML 작업은 불가능하다.

```
SQL> CREATE OR REPLACE VIEW  DEPT_VIEW
  2  AS
  3  SELECT * FROM DEPT
  4  WHERE DEPTNO = 10
  5  WITH READ ONLY;
```

```
SQL> DELETE FROM DEPT_VIEW
  2  WHERE DEPTNO  = 10;
DELETE FROM DEPT_VIEW
            *
1행에 오류:
ORA-01752: 뷰으로 부터 정확하게 하나의 키-보전된 테이블 없이 삭제할 수 없습니다
```

- 뷰 (VIEW) 제거
  : 기본 테이블을 기반으로 하기 때문에 데이터 손실 없이 뷰를 삭제한다.
  : 뷰 삭제는 뷰가 만들어진 기본 테이블에는 영향을 미치지 않는다.

```
DROP VIEW view
```

```
SQL> DROP VIEW DEPT_VIEW;

뷰가 삭제되었습니다.
```

❑ 1) 뷰 [view] 종류

- 인라인 뷰 (inline view)

  - 서브쿼리의 특별한 형태로서 FROM 절에서 사용하는 서브쿼리이다.
  - **인라인 뷰는 SQL명령문이 실행되는 동안만 임시적으로 사용한다.**
    (일반적인 뷰는 생성해서 계속 사용가능)
  - 실무에서 인라인 뷰는 FROM절에서 참조하는 테이블의 크기가 클 경우, 필요한 행과 컬럼 만으로 구성된 집합을 재정의하여 쿼리문을 효율적으로 사용할 수 있다.

```
SELECT column_list
FROM (subquery) alias
WHERE condition;
```

❑ 1) 뷰 (view)

예제> emp 와 dept 테이블에서 부서별 sal 총합과 평균을 출력?

```
select e.deptno , total_sum , total_avg , cnt
from ( select deptno , sum(sal) total_sum, avg(sal) total_avg , count(*) cnt
       from emp
       group by deptno ) e , dept d
where e.deptno = d.deptno;
```

DEPTNO	TOTAL_SUM	TOTAL_AVG	CNT
30	9400	1566.6666666666666666666666666666666667	6
20	10875	2175	5
10	8750	2916.6666666666666666666666666666666667	3

예제> 부서별 평균급여 보다 많은 급여를 받는 사원의 정보 출력?

```
SELECT  A.ENAME , A.SAL , A.DEPTNO ,  B.SALAVG
FROM EMP A, (  SELECT DEPTNO , AVG(SAL ) SALAVG
         FROM EMP
         GROUP BY DEPTNO ) B
WHERE A.DEPTNO  = B.DEPTNO
AND A.SAL > B.SALAVG;
```

❑ 2) 시퀀스

- 시퀀스 (SEQUENCE) 정의

: 여러 사용자들이 공유하는 데이터베이스 객체로서, **호출 될 때마다 중복되지 않은 고유한 숫자를 리턴하는 객체이다.**

: 중복되지 않는 기본키 컬럼에 사용할 값을 발생시키는데 주로 사용한다.

```
CREATE SEQUENCE sequence
       [INCREMENT BY n]
       [START WITH n]
       [{MAXVALUE n | NOMAXVALUE}]
       [{MINVALUE n | NOMINVALUE}]
       [{CYCLE | NOCYCLE)]
       [{CACHE n | NOCACHE}];
```

MAXVALUE: 생성 가능한 시퀀스 최대값
MINVALUE: cycle일 경우 새로 시작값.
감소하는 시퀀스인 경우는 최소값.

```
SQL> CREATE SEQUENCE  EMP_SEQ
  2  INCREMENT BY 1
  3  START WITH 100
  4  MAXVALUE 9999
  5  NOCACHE
  6  NOCYCLE;

시퀀스가 생성되었습니다.
```

```
SQL> SELECT SEQUENCE_NAME , MIN_VALUE, MAX_VALUE, INCREMENT_BY
  2  FROM USER_SEQUENCES;

SEQUENCE_NAME         MIN_VALUE  MAX_VALUE INCREMENT_BY
-------------------- ---------- ---------- ------------
EMP_SEQ                       1       9999            1
```

- 시퀀스 (SEQUENCE) 사용

: NEXTVAL
- 지정된 시퀀스에서 순차적인 시퀀스 번호를 추출할 때 사용.
- **시퀀스명.NEXTVAL**

: CURRVAL
- 사용자가 방금 추출한 시퀀스 번호를 참조할 때 사용.
- **시퀀스명.CURRVAL**
- 반드시 NEXTVAL에 의해서 번호를 추출한 후에 사용해야 한다.

```
SQL> INSERT INTO EMP
  2  VALUES ( EMP_SEQ.NEXTVAL , '홍길동','인사', NULL , SYSDATE,
  3   2500, 300, 40 );

1 개의 행이 만들어졌습니다.

SQL> SELECT EMPNO, ENAME, JOB
  2  FROM EMP
  3  WHERE DEPTNO = 40;

     EMPNO ENAME                JOB
---------- -------------------- ---------
       100 홍길동               인사
```

```
SQL> SELECT EMP_SEQ.CURRVAL FROM DUAL;

   CURRVAL
----------
       100
```

- 시퀀스 (SEQUENCE) 변경
  : 증분, 최대값, 최소값, 순환여부 , 캐시여부를 변경할 수 있다.
  : 시퀀스가 변경되면 다음 번 시퀀스 번호 추출부터 변경사항이 적용된다.
  : START WITH 옵션은 변경이 불가능하며, 필요시 시퀀스를 삭제하고 재 생성해야 한다.
  : MAXVALUE 값은 현재 시퀀스 번호보다 큰 번호로 지정해야 한다.

```
SQL> ALTER SEQUENCE EMP_SEQ
  2    INCREMENT BY 2
  3    MAXVALUE 10000
  4    NOCACHE
  5    NOCYCLE;

시퀀스가 변경되었습니다.
```

- 시퀀스 (SEQUENCE) 삭제

```
SQL> DROP SEQUENCE EMP_SEQ;

시퀀스가 삭제되었습니다.
```

❑ 3] 인덱스

▪ 인덱스 (INDEX) 정의

: 테이블에서 행을 검색할 때 검색 속도를 높이기 위해 Oracle 서버가 사용하는 스키마 객체이다.(**데이터의 실제 저장위치인 ROWID를 저장하고 관리함**)
: 인덱스 없이 데이터를 검색하면 테이블의 모든 데이터를 읽어 데이터를 선별한다.(Full Scan)
: 인덱스를 사용하면 디스크의 I/O 를 감소시킬 수 있다.
: 해당 테이블과 논리적으로 독립적이다.
: Oracle 서버에 의해 자동으로 사용 및 관리된다.
반면에 테이블을 삭제하면 관련 인덱스는 자동으로 삭제된다.

▪ 인덱스 (INDEX) 생성

1. 자동 생성
: PRIMARY KEY , UNIQUE 제약 조건 지정 시 UNIQUE INDEX 가 자동 생성 된다.

2. 수동 생성 (non-unique 인덱스 또는 unique 인덱스)
: 한 개 컬럼 또는 여러 컬럼(복합 인덱스)을 이용하여 인덱스 생성 가능하다.

```
CREATE INDEX index
ON table (column[, column] ...);
```

```
SQL> CREATE INDEX EMP_ENAME_IDX
  2  ON EMP( ENAME );

인덱스가 생성되었습니다.
```

create **unique** index ~

❑ 3] 인덱스

▪ 인덱스(INDEX)를 사용할 컬럼 선정

: 값의 범위가 넓은 컬럼 (즉, 컬럼내의 값이 다양할수록 좋다)
: NULL 값이 많은 컬럼 (NULL 값은 인덱스에 포함되지 않기 때문에 인덱스 크기가 감소)
: **WHERE절 또는 JOIN 조건에 사용되는 컬럼**
: 테이블이 크고 대부분의 쿼리 문장이 테이블내 전체 데이터의 약 2 ~ 4% 이내를 검색하는 경우

*** 인덱스가 반드시 성능을 향상시키는 것은 아니다. 테이블에 DML 작업을 수행하면 관련 인덱스도 변경되어야 하므로 오히려 속도가 저하될 수도 있다.**

▪ 인덱스(INDEX) 작성할 필요 없는 경우

: 테이블이 작은 경우
: 쿼리 문장의 조건에 자주 사용되지 않는 컬럼.
: 대부분의 쿼리문장이 테이블내 전체 데이터의 약 2 ~ 4 % 이상을 검색하는 경우
: 테이블이 자주 변경되는 경우
: 인덱스가 작성된 컬럼이 쿼리문장의 조건에서 표현식(함수 및 NOT등)에 포함된 경우

▪ 인덱스(INDEX) 삭제

```
SQL> DROP INDEX EMP_ENAME_IDX;

인덱스가 삭제되었습니다.
```

❑ 3) 인덱스

- INDEX 관리

```
SQL> DROP TABLE hr.copy_emp;

Table dropped.

SQL> CREATE TABLE hr.copy_emp
  2  AS
  3  SELECT * FROM hr.employees;

Table created.
```

* Index 가 없는 상태에서의 query

```
SELECT * FROM hr.copy_emp WHERE employee_id = 103;
```

결과 | 스크립트 출력 | 설명 | 자동 추적 | DBMS 출력 | OWA 출력

OPERATION	OBJECT_NAME	OPTIONS	COST
SELECT STATEMENT			3
TABLE ACCESS	COPY_EMP	FULL	3
필터 술어			
EMPLOYEE_ID=103			

- PK와 UK가 없기 때문에 중복 허용 상태이다.

```
EMPLOYEE_ID LAST_NAME
----------- ----------
        198 OConnell
        199 Grant
        200 Whalen
        201 Hartstein
        202 Fay
        203 Mavris
        204 Baer
        205 Higgins
        206 Gietz
        100 King
        101 Kochhar
        102 De Haan
        103 Hunold
        104 Ernst
        105 Austin
        106 Pataballa
        107 Lorentz
        108 Greenberg
```

- Full Table Scan 수행된다.

❑ 3] 인덱스

- 다음 경우에는 index를 이용하지 않는다.

```
SELECT * FROM hr.copy_emp WHERE employee_id != 103;
```

결과 | 스크립트 출력 | 설명 | 자동 추적 | DBMS 출력 | OWA 출력

PERATION	OBJECT_NAME	OPTIONS
SELECT STATEMENT		
TABLE ACCESS	COPY_EMP	FULL
필터 술어		
EMPLOYEE_ID<>103		

← NOT같은 부정

```
SELECT * FROM hr.copy_emp WHERE employee_id IS NOT NULL;
```

결과 | 스크립트 출력 | 설명 | 자동 추적 | DBMS 출력 | OWA 출력

PERATION	OBJECT_NAME	OPTIONS
SELECT STATEMENT		
TABLE ACCESS	COPY_EMP	FULL
필터 술어		
EMPLOYEE_ID IS NOT NULL		

← NULL 관련

```
SELECT * FROM hr.copy_emp WHERE to_number( employee_id ) = 103;
```

결과 | 스크립트 출력 | 설명 | 자동 추적 | DBMS 출력 | OWA 출력

PERATION	OBJECT_NAME	OPTIONS
SELECT STATEMENT		
TABLE ACCESS	COPY_EMP	FULL
필터 술어		
TO_NUMBER(TO_CHAR(EMPL		

← 컬럼 변형

- Function Based INDEX (함수 기반 인덱스)

```
CREATE INDEX HR.COPY_EMP_NAME_IDX
ON HR.COPY_EMP(LAST_NAME);

SELECT * FROM HR.COPY_EMP WHERE UPPER(LAST_NAME) = 'KING';
```

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

OPERATION	OBJECT_NAME	OPTIONS
SELECT STATEMENT		
TABLE ACCESS	COPY_EMP	FULL
Filter Predicates		
UPPER(LAST_NAME)='KING'		

```
LAST_NAME
-----------
OConnell
Grant
Whalen
Hartstein
Fay
Mavris
Baer
Higgins
Gietz
King
Kochhar
```

```
CREATE INDEX HR.COPY_EMP_NAME_IDX2
ON HR.COPY_EMP( UPPER(LAST_NAME));

SELECT * FROM HR.COPY_EMP WHERE UPPER(LAST_NAME) = 'KING';
```

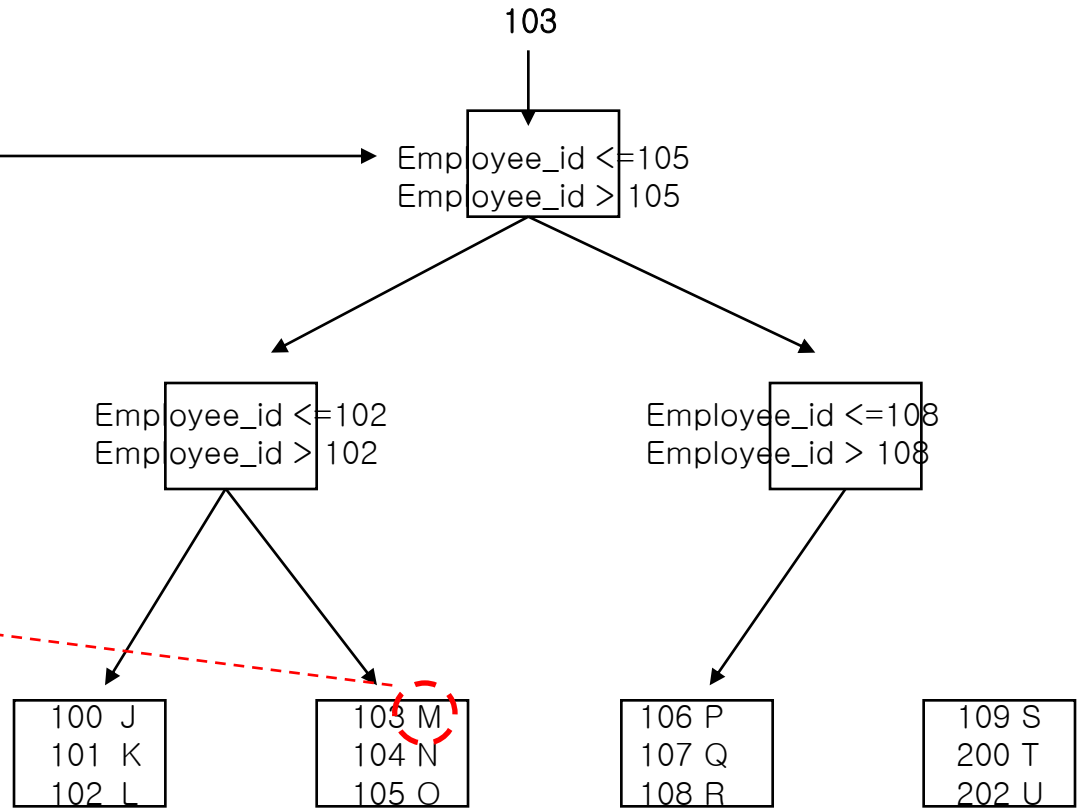
Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output

OPERATION	OBJECT_NAME	OPTIONS
SELECT STATEMENT		
TABLE ACCESS	COPY_EMP	BY INDEX ROWID
INDEX	COPY_EMP_NAME_IDX2	RANGE SCAN
Access Predicates		
UPPER(LAST_NAME)='KING'		

❑ 3] 인덱스

- INDEX 유형 (B-Tree)

EMPLOYEE_ID	LAST_NAME	ROWID
198	OConnell	A
199	Grant	B
200	Whalen	C
201	Hartstein	D
202	Fay	E
203	Mavris	F
204	Baer	G
205	Higgins	H
206	Gietz	I
100	King	J
101	Kochhar	K
102	De Haan	L
103	Hunold	M
104	Ernst	N
105	Austin	O
106	Pataballa	P
107	Lorentz	Q
108	Greenberg	R




```
SQL> SELECT *
  2  FROM hr.copy_emp
  3  WHERE employee_id = 103;
```

```
SELECT * FROM hr.copy_emp WHERE employee_id = 103;
```

결과 | 스크립트 출력 | 설명 | 자동 추적 | DBMS 출력 | OWA 출력

OPERATION	OBJECT_NAME	OPTIONS	COST
SELECT STATEMENT			2
TABLE ACCESS	COPY_EMP	BY INDEX ROWID	2
INDEX	COPY_EMP_IDX	RANGE SCAN	1

액세스 술어
EMPLOYEE_ID=103

Thank you