



View

1. View

- ❖ 뷰(View)는 물리적인 테이블을 근거한 논리적인 가상 테이블
- ❖ 가상이란 단어는 실질적으로 데이터를 저장하고 있지 않기 때문에 붙인 것이고 테이블이란 단어는 실질적으로 데이터를 저장하고 있지 않더라도 사용자는 마치 테이블을 사용하는 것과 동일하게 뷰를 사용할 수 있기 때문에 붙인 것
- ❖ 뷰는 기본 테이블에서 파생된 객체로서 기본 테이블에 대한 하나의 쿼리문
- ❖ 사용자에게 주어진 뷰를 통해서 기본 테이블을 제한적으로 사용하도록 함
- ❖ 뷰를 생성하기 위해서는 실질적으로 데이터를 저장하고 있는 물리적인 테이블이나 다른 뷰가 존재해야 함



실습

❖ 기본 테이블을 생성

1. DEPT_COPY를 DEPT 테이블의 복사본으로 생성

```
CREATE TABLE DEPT_COPY  
AS  
SELECT * FROM DEPT;
```

2. EMP 테이블의 복사본으로 EMP_COPY를 생성

```
CREATE TABLE EMP_COPY  
AS  
SELECT * FROM EMP;
```



1. View

- ❖ 뷰를 생성하기 위한 기본 형식

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

- ❖ 테이블을 생성하기 위해서 CREATE TABLE 로 시작하지만 뷰를 생성하기 위해서는 CREATE VIEW로 시작
- ❖ AS 다음은 서브 쿼리문과 유사
- ❖ subquery에는 SELECT 문을 기술



1. View

❖ CREATE OR RELPACE VIEW

- ✓ 뷰를 만들 때 CREATE OR RELPACE VIEW 대신 CREATE VIEW 만 사용 가능
- ✓ CREATE VIEW를 통해 만들어진 뷰의 구조를 바꾸려면 뷰를 삭제하고 다시 만들어야 되는 반면 CREATE OR REPLACE VIEW는 새로운 뷰를 만들 수 있을 뿐만 아니라 기존에 뷰가 존재하더라도 삭제하지 않고 새로운 구조의 뷰로 변경(REPLACE)

❖ FORCE

- ✓ FORCE를 사용하면, 기본 테이블의 존재 여부에 상관없이 뷰를 생성

❖ WITH CHECK OPTION

- ✓ WITH CHECK OPTION을 사용하면, 해당 뷰를 통해서 볼 수 있는 범위 내에서만 UPDATE 또는 INSERT가 가능

❖ WITH READ ONLY

- ✓ WITH READ ONLY를 사용하면 해당 뷰를 통해서만 SELECT만 가능하며 INSERT/UPDATE/DELETE를 할 수 없음
- ✓ 생략하면 뷰를 사용하여 추가, 수정, 삭제(INSERT/UPDATE/DELETE)가 모두 가능



1. View

- ❖ 만일, 30번 부서에 소속된 직원들의 사번과 이름과 부서번호를 자주 검색한다고 한다면 같은 SELECT 문을 여러 번 입력해야 함

```
SELECT EMPNO, ENAME, DEPTNO  
FROM EMP_COPY  
WHERE DEPTNO=30;
```

- ❖ 동일한 결과를 출력하기 위해서 매번 SELECT 문을 입력하기란 번거로운 일
- ❖ 뷰는 이와 같이 번거로운 SELECT 문을 매번 입력하는 대신 보다 쉽게 원하는 결과를 얻고자 하는 바람에서 출발한 개념



1. View

- ❖ 자주 사용되는 30번 부서에 소속된 직원들의 사번과 이름과 부서번호를 출력하기 위한 SELECT 문을 하나의 뷰로 정의

```
CREATE VIEW EMP_VIEW30  
AS  
SELECT EMPNO, ENAME, DEPTNO  
FROM EMP_COPY  
WHERE DEPTNO=30;
```

- ❖ 뷰는 테이블에 접근(SELECT)한 것과 동일한 방법으로 결과를 얻을 수 있음

```
SELECT * FROM EMP_VIEW30;
```

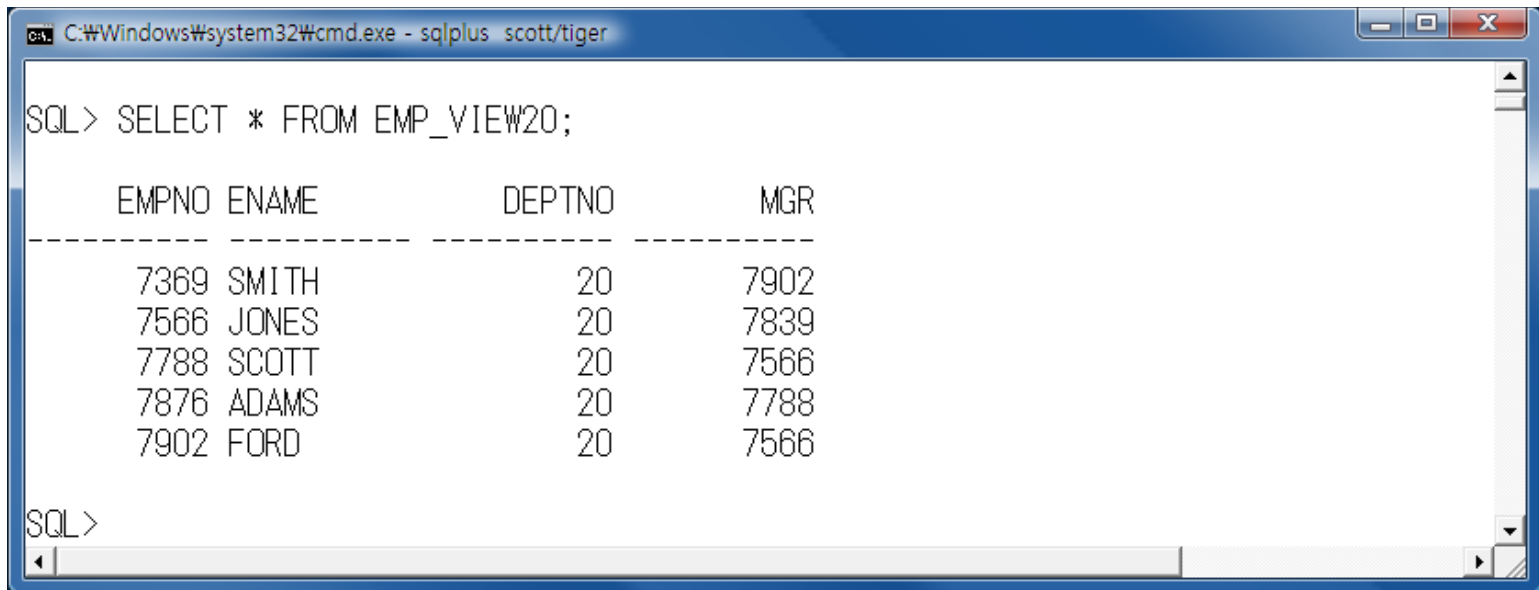
- ❖ 뷰 구조 확인

```
DESC EMP_VIEW30;
```



연습문제

❖ 기본 테이블은 EMP_COPY를 이용하고 deptno가 20번인 부서에 소속된 직원들의 사번(empno)과 이름(ename)과 부서번호(deptno)와 상관의 사번(mgr)을 출력하기 위한 SELECT문을 EMP_VIEW20 이란 이름의 뷰로 정의



```
SQL> SELECT * FROM EMP_VIEW20;
```

EMPNO	ENAME	DEPTNO	MGR
7369	SMITH	20	7902
7566	JONES	20	7839
7788	SCOTT	20	7566
7876	ADAMS	20	7788
7902	FORD	20	7566

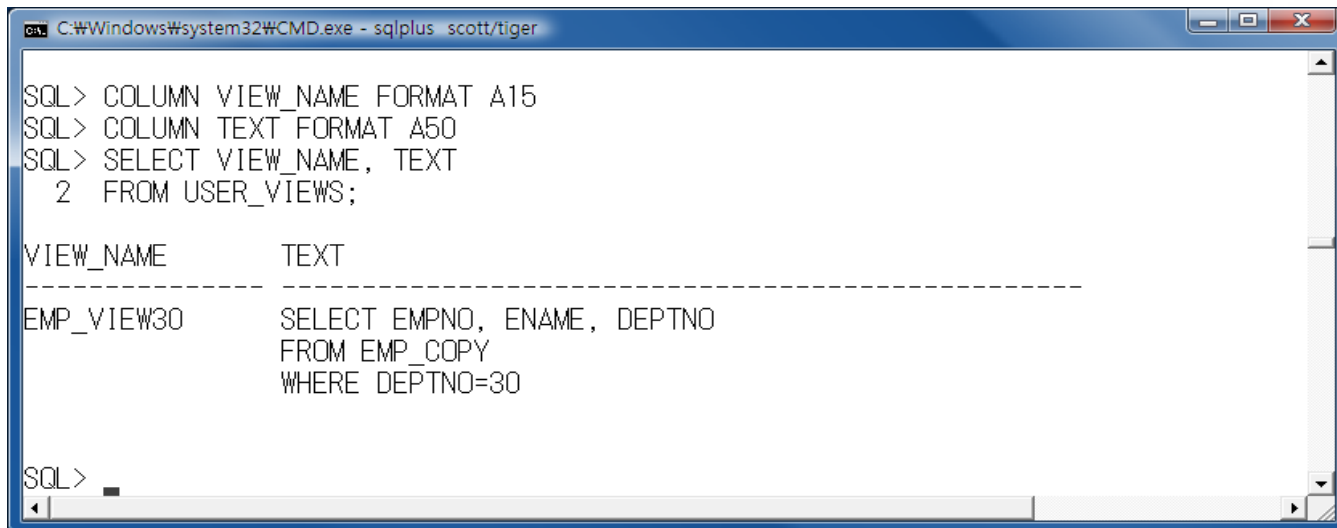
```
SQL>
```


2. 뷰의 내부구조

- ❖ EMP_VIEW30라는 뷰는 데이터를 물리적으로 저장하고 있지 않으며 CREATE VIEW 명령어로 뷰를 정의할 때 AS 절 다음에 기술한 쿼리 문장 자체를 저장하고 있음
- ❖ 뷰 정의할 때 기술한 쿼리문을 확인하고자 하는 경우에는 USER_VIEWS 테이블의 TEXT 컬럼 값을 확인

```
SELECT VIEW_NAME, TEXT  
FROM USER_VIEWS;
```

- ❖ 기본 테이블은 디스크 공간을 할당 받아서 실질적으로 데이터를 저장하고 있지만, 뷰는 데이터 덱서너리USER_VIEWS 에 사용자가 뷰를 정의할 때 기술한 서브 쿼리문(SELECT 문)만을 문자열 형태로 저장하고 있음



```
C:\Windows\system32\CMD.exe - sqlplus scott/tiger  
  
SQL> COLUMN VIEW_NAME FORMAT A15  
SQL> COLUMN TEXT FORMAT A50  
SQL> SELECT VIEW_NAME, TEXT  
2 FROM USER_VIEWS;  
  
VIEW_NAME          TEXT  
-----  
EMP_VIEW30         SELECT EMPNO, ENAME, DEPTNO  
                     FROM EMP_COPY  
                     WHERE DEPTNO=30  
  
SQL>
```

3. 뷰를 사용하는 이유

- ❖ 뷰를 사용하는 이유
 - ✓ 복잡하고 긴 쿼리문을 뷰로 정의하면 접근을 단순화시킬 수 있음
 - ✓ 보안에 유리
- ❖ 사용자마다 특정 객체만 조회할 수 있도록 권한을 부여할 수 있기에 동일한 테이블을 접근하는 사용자마다 서로 다르게 보도록 여러 개의 뷰를 정의해 놓고 특정 사용자만이 해당 뷰에 접근할 수 있도록 하면 보안에 유리



4. 뷰의 종류

- ❖ 뷰는 뷰를 정의하기 위해서 사용되는 기본 테이블의 수에 따라 단순 뷰(Simple View)와 복합 뷰(Complex View)로 나뉨

단순 뷰	복합 뷰
하나의 테이블로 생성	여러 개의 테이블로 생성
그룹 함수의 사용이 불가능	그룹 함수의 사용이 가능
DISTINCT 사용이 불가능	DISTINCT 사용이 가능
DML 사용 가능	DML 사용이 제한적(일반적으로 불가능)



실습

❖ 단순 뷰에 대해서 DML(INSERT/UPDATE/DELETE) 문을 사용할 수 있음을 확인

1. EMP_VIEW30 뷰에 데이터를 추가

```
INSERT INTO EMP_VIEW30  
VALUES(8000, 'ANGEL', 30);
```

```
SELECT * FROM EMP_VIEW30;
```

2. 단순 뷰를 대상으로 실행한 DML 명령문의 처리 결과는 뷰를 정의할 때 사용한 기본 테이블에 적용

```
SELECT * FROM EMP_COPY;
```



실습

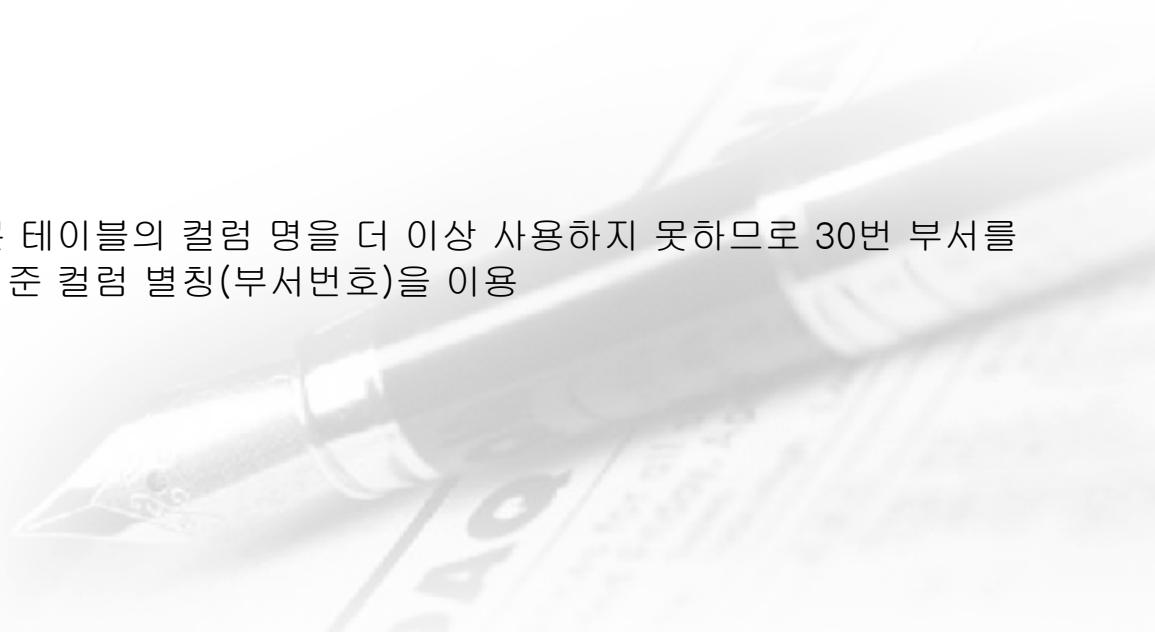
- ❖ 기본 테이블(EMP_COPY)의 컬럼 명을 한글화 하여 컬럼 명이 사원번호, 사원명, 급여, 부서번호로 구성

```
CREATE OR REPLACE  
VIEW EMP_VIEW(사원번호, 사원명, 급여, 부서번호)  
AS  
SELECT EMPNO, ENAME, SAL, DEPTNO  
FROM EMP_COPY;
```

- ❖ EMP_VIEW30 뷰에 데이터를 추가

```
SELECT * FROM EMP_VIEW  
WHERE 부서번호=30;
```

- ❖ 뷰는 컬럼에 별칭을 주게 되면 기본 테이블의 컬럼 명을 더 이상 사용하지 못하므로 30번 부서를 검색하기 위해서는 뷰를 생성할 때 준 컬럼 별칭(부서번호)을 이용



실습

❖ 그룹 함수 SUM과 AVG를 사용해서 각 부서별 급여 총액과 평균을 구하는 뷰를 작성 하기 위해서 SELECT 절 다음에 SUM이란 그룹 함수를 사용하면 결과를 뷰의 특정 컬럼처럼 사용하는 것이므로 물리적인 컬럼이 존재하지 않는 가상 컬럼이기에 뷰를 생성할 때 가상 컬럼을 사용하려면 사용자가 반드시 이름을 따로 설정해야 함

- ✓ 부서별 급여 총액과 평균을 구하기 위한 뷰를 생성

```
CREATE VIEW VIEW_SAL
```

```
AS
```

```
SELECT DEPTNO, SUM(SAL) AS "SalSum", AVG(SAL) AS "SalAvg"
```

```
FROM EMP_COPY
```

```
GROUP BY DEPTNO;
```



4.1 단순 뷰

- ❖ 단순 뷰는 DML 명령어를 사용하여 조작이 가능
- ❖ 몇 가지의 경우에는 조작이 불가능
 - ✓ 뷰 정의에 포함되지 않은 컬럼 중에 기본 테이블의 컬럼이 NOT NULL 제약 조건이 지정되어 있는 경우 INSERT 문이 사용 불가능한데 뷰에 대한 INSERT 문은 기본 테이블에 NULL 값을 입력하는 형태가 되기 때문
 - ✓ SAL*12와 같이 산술 표현식으로 정의된 가상 컬럼이 뷰에 정의되면 INSERT나 UPDATE가 불가능
 - ✓ DISTINCT을 포함한 경우에도 DML 명령을 사용할 수 없음
 - ✓ 그룹 함수나 GROUP BY 절을 포함한 경우에도 DML 명령을 사용할 수 없음

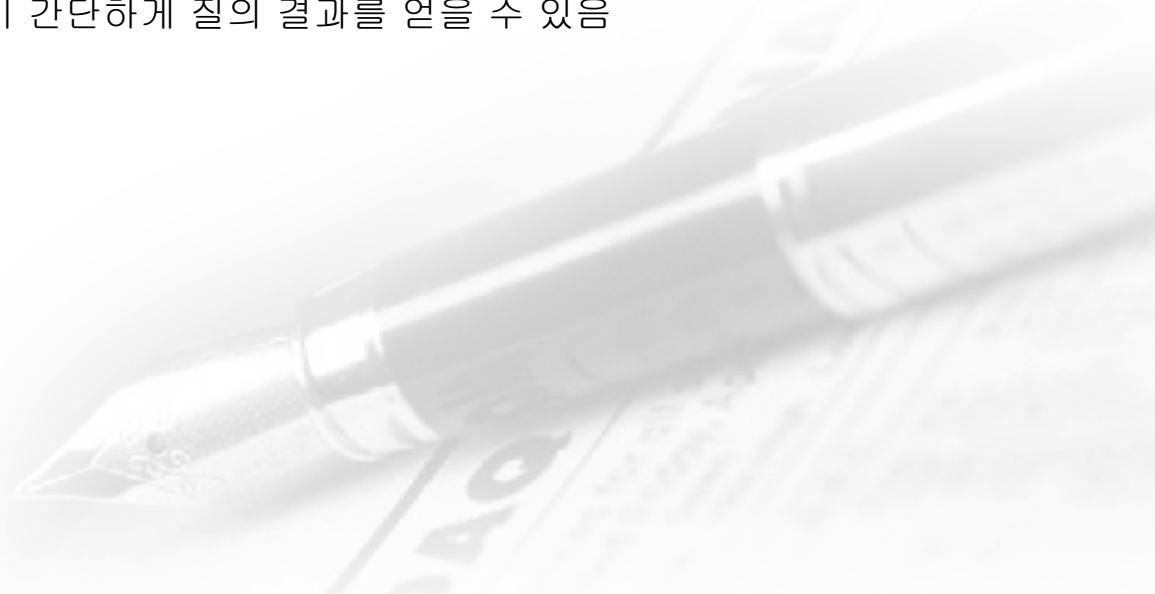


4.2 복합 뷰

- ❖ 뷰를 사용하는 이유 중의 하나가 복잡하고 자주 사용하는 질의를 보다 쉽고 간단하게 사용하기 위해서 인데 사원 테이블과 부서 테이블을 자주 조인하는 경우에 사원 테이블과 부서 테이블을 조인하기 위해서는 다음과 같은 SELECT 문을 매번 작성해야 함

```
SELECT E.EMPNO, E.ENAME, E.SAL, E.DEPTNO, D.DNAME, D.LOC  
FROM EMP E, DEPT D  
WHERE E.DEPTNO = D.DEPTNO  
ORDER BY EMPNO DESC;
```

- ❖ 조인문에 "CREATE VIEW EMP_VIEW_DEPT AS" 만 추가해서 뷰로 작성해 놓으면 "SELECT * FROM EMP_VIEW_DEPT;" 와 같이 간단하게 질의 결과를 얻을 수 있음



실습

- ❖ 사원 테이블과 부서 테이블을 조인하기 위해서 복합 뷰를 생성

1.다음은 사번, 이름, 급여, 부서번호, 부서명, 지역명을 출력하기 위한 복합 뷰

```
CREATE VIEW EMP_VIEW_DEPT  
AS  
SELECT E.EMPNO, E.ENAME, E.SAL, E.DEPTNO, D.DNAME, D.LOC  
FROM EMP E, DEPT D  
WHERE E.DEPTNO = D.DEPTNO  
ORDER BY EMPNO DESC;
```

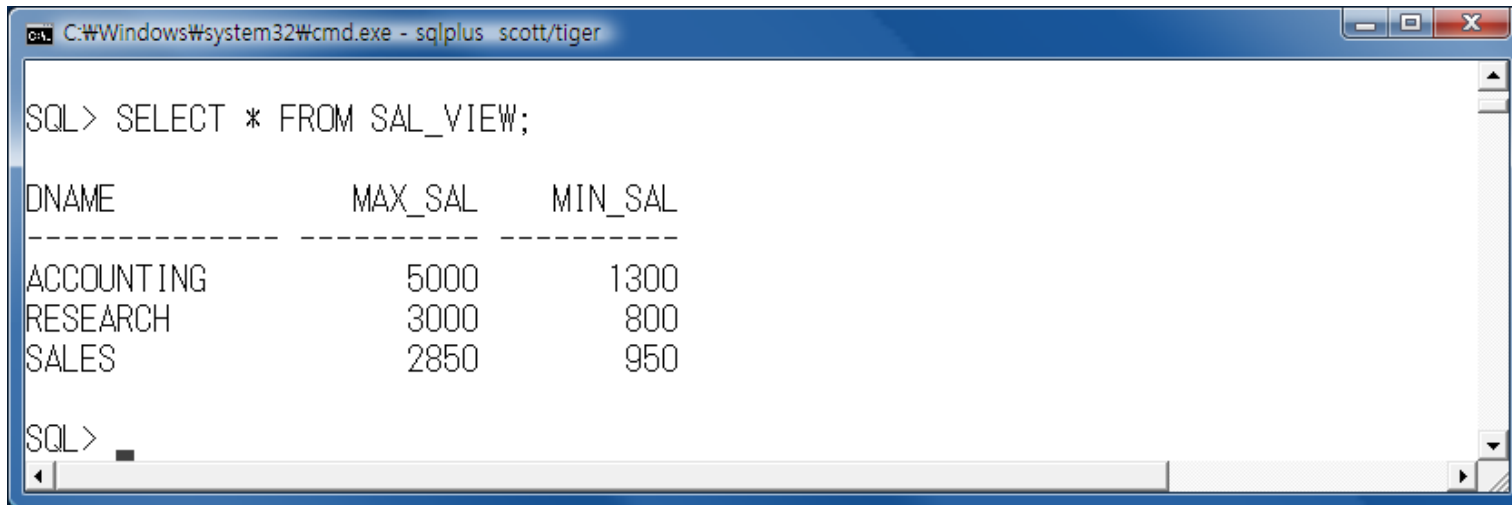
2.뷰를 생성한 후, 이를 활용하면 복잡한 질의를 쉽게 처리

```
SELECT * FROM EMP_VIEW_DEPT;
```



연습문제

❖ EMP 테이블에서 부서별(deptno) 최대 급여(sal)와 최소 급여를 출력하는 뷰를 SAL_VIEW 란 이름으로 작성



```
C:\Windows\system32\cmd.exe - sqlplus scott/tiger

SQL> SELECT * FROM SAL_VIEW;

DNAME                MAX_SAL    MIN_SAL
-----
ACCOUNTING            5000       1300
RESEARCH              3000        800
SALES                 2850        950

SQL>
```

The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe - sqlplus scott/tiger". Inside the window, the SQL command "SELECT * FROM SAL_VIEW;" has been executed. The output is a table with three columns: DNAME, MAX_SAL, and MIN_SAL. The data rows are ACCOUNTING (5000, 1300), RESEARCH (3000, 800), and SALES (2850, 950). The prompt "SQL>" is visible at the bottom.

DNAME	MAX_SAL	MIN_SAL
ACCOUNTING	5000	1300
RESEARCH	3000	800
SALES	2850	950

5. 뷰 삭제와 다양한 옵션

- ❖ 뷰는 실체가 없는 가상 테이블이기 때문에 뷰를 삭제한다는 것은 USER_VIEWS 데이터 디렉터리
에 저장되어 있는 뷰의 정의를 삭제하는 것
- ❖ 뷰를 삭제해도 뷰를 정의한 기본 테이블의 구조나 데이터에는 전혀 영향을 주지 않음
- ❖ VIEW_SAL을 삭제
DROP VIEW VIEW_SAL;



6. 뷰 생성 옵션

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



6.1 OR REPLACE 옵션

- ❖ CREATE OR REPLACE VIEW 를 사용하면 존재하지 않은 뷰이면 새로운 뷰를 생성하고 기존에 존재하는 뷰이면 그 내용을 변경
- ❖ 이전에 작성한 EMP_VIEW30 뷰는 “EMPNO, ENAME, SAL, DEPTNO” 4 개의 컬럼을 출력하는 형태였는데 커미션 컬럼을 추가로 출력할 수 있도록 하기 위해서 뷰의 구조를 변경

```
CREATE OR REPLACE VIEW EMP_VIEW30  
AS  
SELECT EMPNO, ENAME, SAL, COMM, DEPTNO  
FROM EMP_COPY  
WHERE DEPTNO=30;
```



6.2 FORCE 옵션

- ❖ 뷰를 생성하는 경우에 일반적으로 기본 테이블이 존재한다는 가정 하에서 쿼리문을 작성
- ❖ 극히 드물기는 하지만, 기본 테이블이 존재하지 않는 경우에도 뷰를 생성해야 할 경우가 있는데 이럴 경우에 사용하는 것이 FORCE 옵션
- ❖ FORCE 옵션과 반대로 동작하는 것으로서 NOFORCE 옵션이 있음
- ❖ NOFORCE 옵션은 반드시 기본 테이블이 존재해야 할 경우에만 뷰가 생성
- ❖ 특별한 설정이 없으면 디폴트로 NOFORCE 옵션이 지정된 것이므로 간주



실습

- ❖ 존재하지 않는 EMPLOYEES를 기본 테이블로 하여 뷰를 생성하게 되면 다음과 같이 오류가 발생
CREATE OR REPLACE VIEW EMPLOYEES_VIEW
AS
SELECT EMPNO, ENAME, DEPTNO
FROM EMPLOYEES
WHERE DEPTNO=30;
- ❖ 특별한 설정이 없으면 NOFORCE 옵션이 지정된 것이므로 반드시 존재하는 기본 테이블을 이용한 쿼리문으로 뷰를 생성해야 함
CREATE OR REPLACE FORCE VIEW NOTABLE_VIEW
AS
SELECT EMPNO, ENAME, DEPTNO
FROM EMPLOYEES
WHERE DEPTNO=30;
- ❖ FORCE의 반대 옵션이 NOFORCE
CREATE OR REPLACE NOFORCE EXISTTABLE_VIEW
AS
SELECT EMPNO, ENAME, DEPTNO
FROM EMPLOYEES
WHERE DEPTNO=30;

6.3 WITH CHECK OPTION

- ❖ 30 번 부서 소속 직원들의 정보만으로 구성된 뷰

```
CREATE OR REPLACE VIEW EMP_VIEW30  
AS  
SELECT EMPNO, ENAME, DEPTNO, SAL  
FROM EMP_COPY  
WHERE DEPTNO=30;
```

```
SELECT * FROM EMP_VIEW30;
```

- ❖ 뷰는 테이블처럼 SELECT문으로 조회할 수 있음은 물론이고 DML 문으로 내용을 조작할 수 있는데 UPDATE 문으로 30번 부서에 소속된 직원 중에 급여가 1200 이상인 직원은 20번 부서로 변경

```
UPDATE EMP_VIEW30  
SET DEPTNO=20  
WHERE SAL >= 1200;
```

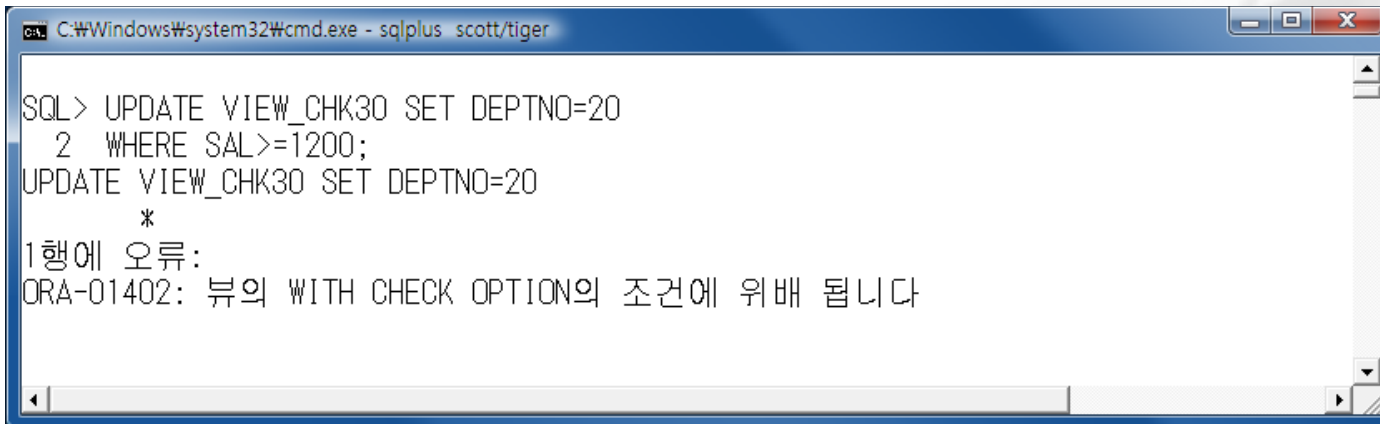


6.3 WITH CHECK OPTION

- ❖ 오라클에서는 WITH CHECK OPTION 으로 뷰의 조건내에서만 삽입, 삭제, 갱신이 가능하도록 제한

```
CREATE OR REPLACE VIEW VIEW_CHK30
AS
SELECT EMPNO, ENAME, SAL, COMM, DEPTNO
FROM EMP_COPY
WHERE DEPTNO=30
WITH CHECK OPTION;
```

- ❖ SAL 이 1200 이상인 사원의 부서를 20번 부서로 변경
UPDATE VIEW_CHK30
SET DEPTNO=20
WHERE SAL >= 1200;

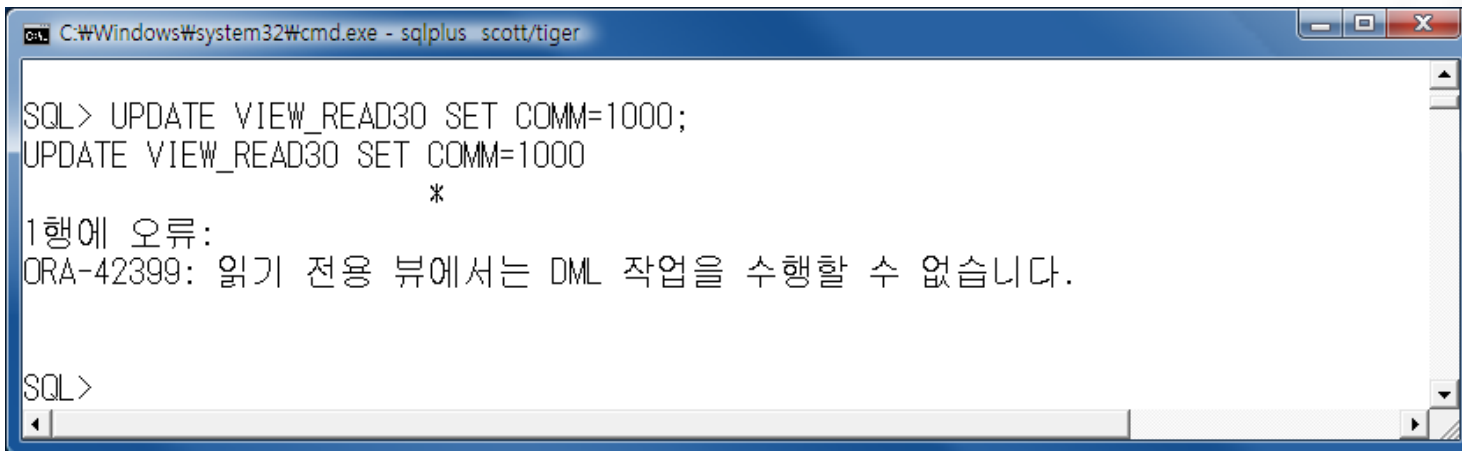
A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe - sqlplus scott/tiger". The window shows the execution of an SQL UPDATE statement. The statement is: "SQL> UPDATE VIEW_CHK30 SET DEPTNO=20 WHERE SAL>=1200;". The output shows "2" rows updated. Then, the statement "UPDATE VIEW_CHK30 SET DEPTNO=20" is entered again, followed by an asterisk "*" on a new line. The output shows "1" row with an error: "1행에 오류: ORA-01402: 뷰의 WITH CHECK OPTION의 조건에 위배 됩니다".

```
C:\Windows\system32\cmd.exe - sqlplus scott/tiger

SQL> UPDATE VIEW_CHK30 SET DEPTNO=20
  2  WHERE SAL>=1200;
UPDATE VIEW_CHK30 SET DEPTNO=20
      *
1행에 오류:
ORA-01402: 뷰의 WITH CHECK OPTION의 조건에 위배 됩니다
```

6.4 WITH READ ONLY 옵션

- ❖ WITH READ ONLY 옵션은 뷰를 통해서는 기본 테이블의 어떤 컬럼에 대해서도 내용을 절대 변경할 수 없도록 하는 것
- ❖ WITH READ ONLY 옵션을 지정한 뷰를 정의
CREATE OR REPLACE VIEW VIEW_READ30
AS
SELECT EMPNO, ENAME, SAL, COMM, DEPTNO
FROM EMP_COPY
WHERE DEPTNO=30 WITH READ ONLY;
- ❖ VIEW_READ30 뷰의 커미션을 모두 2000으로 변경
UPDATE VIEW_READ30 SET COMM=2000;



```
C:\Windows\system32\cmd.exe - sqlplus scott/tiger

SQL> UPDATE VIEW_READ30 SET COMM=1000;
UPDATE VIEW_READ30 SET COMM=1000
          *
1행에 오류:
ORA-42399: 읽기 전용 뷰에서는 DML 작업을 수행할 수 없습니다.

SQL>
```

7. 인라인 뷰

- ❖ 인라인 뷰: FROM 절에 테이블 이름이 아닌 서브쿼리를 사용한 것
- ❖ ROWNUM
 - ✓ 오라클에서 부여하는 행 번호(pseudo column)
 - ✓ SELECT 구문을 수행하게 되면 ROWNUM은 1부터 만들어지는데 WHERE 절이 수행되서 행 단위로 데이터를 가져올 때 부여되고 조건을 만족하면 다음 번호를 부여하고 조건을 만족하지 못하면 이전 번호를 다시 부여



7. 인라인 뷰

- ❖ 사원 중에서 입사일이 빠른 사람 5명(TOP-5)만을 얻어 오는 질의문을 작성
- ❖ TOP-N을 구하기 위해서 ROWNUM과 인라인 뷰를 사용

1. ROWNUM 컬럼 값을 출력하기 위한 쿼리문

```
SELECT ROWNUM, EMPNO, ENAME, HIREDATE  
FROM EMP;
```

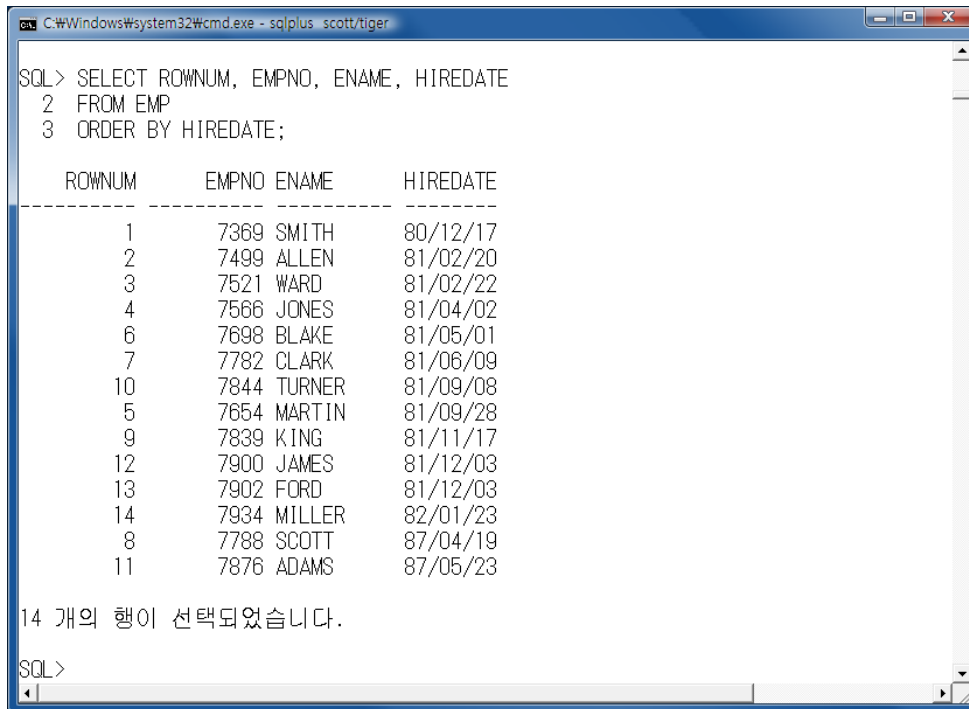
2. 입사일이 빠른 사람 5명만(TOP-N)을 얻어오기 위해서는 일련의 출력 데이터를 일단 임의의 순서로 정렬한 후에 그 중 일부의 데이터만 출력할 수 있도록 해야 하므로 ORDER BY 절을 사용하여 입사일을 기준으로 오름차순 정렬

```
SELECT EMPNO, ENAME, HIREDATE  
FROM EMP  
ORDER BY HIREDATE;
```



실습

3. 입사일을 기준으로 오름차순 정렬을 하는 쿼리문에 ROWNUM 컬럼을 출력
SELECT ROWNUM, EMPNO, ENAME, HIREDATE
FROM EMP
ORDER BY HIREDATE;



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe - sqlplus scott/tiger". The user has entered the following SQL query:

```
SQL> SELECT ROWNUM, EMPNO, ENAME, HIREDATE
2 FROM EMP
3 ORDER BY HIREDATE;
```

The output of the query is displayed in a table format:

ROWNUM	EMPNO	ENAME	HIREDATE
1	7369	SMITH	80/12/17
2	7499	ALLEN	81/02/20
3	7521	WARD	81/02/22
4	7566	JONES	81/04/02
6	7698	BLAKE	81/05/01
7	7782	CLARK	81/06/09
10	7844	TURNER	81/09/08
5	7654	MARTIN	81/09/28
9	7839	KING	81/11/17
12	7900	JAMES	81/12/03
13	7902	FORD	81/12/03
14	7934	MILLER	82/01/23
8	7788	SCOTT	87/04/19
11	7876	ADAMS	87/05/23

Below the table, the message "14 개의 행이 선택되었습니다." (14 rows selected) is displayed. The prompt "SQL>" is visible at the bottom of the window.

실습

- ❖ 결과를 보면 입사일을 기준으로 오름차순 정렬을 하였기에 출력되는 행의 순서는 바뀌더라도 해당 행의 ROWNUM 컬럼 값은 바뀌지 않음
- ❖ ROWNUM 컬럼은 where 구문에서 만들어지므로 이 컬럼의 값을 다른 절에서 사용하려면 새로운 테이블이나 뷰로 새롭게 데이터를 저장해서 사용해야 함
- ❖ TOP-N은 일련의 출력 데이터를 일단 임의의 순서로 정렬한 후에 그 중 일부의 데이터만 출력할 수 있도록 하여 구해야 함

1. 입사일을 기준으로 오름차순 정렬한 쿼리문으로 새로운 뷰를 생성

```
CREATE OR REPLACE VIEW VIEW_HIRE
```

```
AS
```

```
SELECT EMPNO, ENAME, HIREDATE
```

```
FROM EMP
```

```
ORDER BY HIREDATE;
```

2. 입사일을 기준으로 오름차순 정렬을 하는 뷰에 ROWNUM 컬럼을 함께 출력

```
SELECT ROWNUM, EMPNO, ENAME, HIREDATE
```

```
FROM VIEW_HIRE;
```



실습

```
C:\Windows\system32\cmd.exe - sqlplus scott/tiger

SQL> CREATE OR REPLACE VIEW VIEW_HIRE
 2 AS
 3 SELECT EMPNO, ENAME, HIREDATE
 4 FROM EMP
 5 ORDER BY HIREDATE;

뷰가 생성되었습니다.

SQL> SELECT ROWNUM, EMPNO, ENAME, HIREDATE
 2 FROM VIEW_HIRE;
```

ROWNUM	EMPNO	ENAME	HIREDATE
1	7369	SMITH	80/12/17
2	7499	ALLEN	81/02/20
3	7521	WARD	81/02/22
4	7566	JONES	81/04/02
5	7698	BLAKE	81/05/01
6	7782	CLARK	81/06/09
7	7844	TURNER	81/09/08
8	7654	MARTIN	81/09/28
9	7839	KING	81/11/17
10	7900	JAMES	81/12/03
11	7902	FORD	81/12/03
12	7934	MILLER	82/01/23
13	7788	SCOTT	87/04/19
14	7876	ADAMS	87/05/23

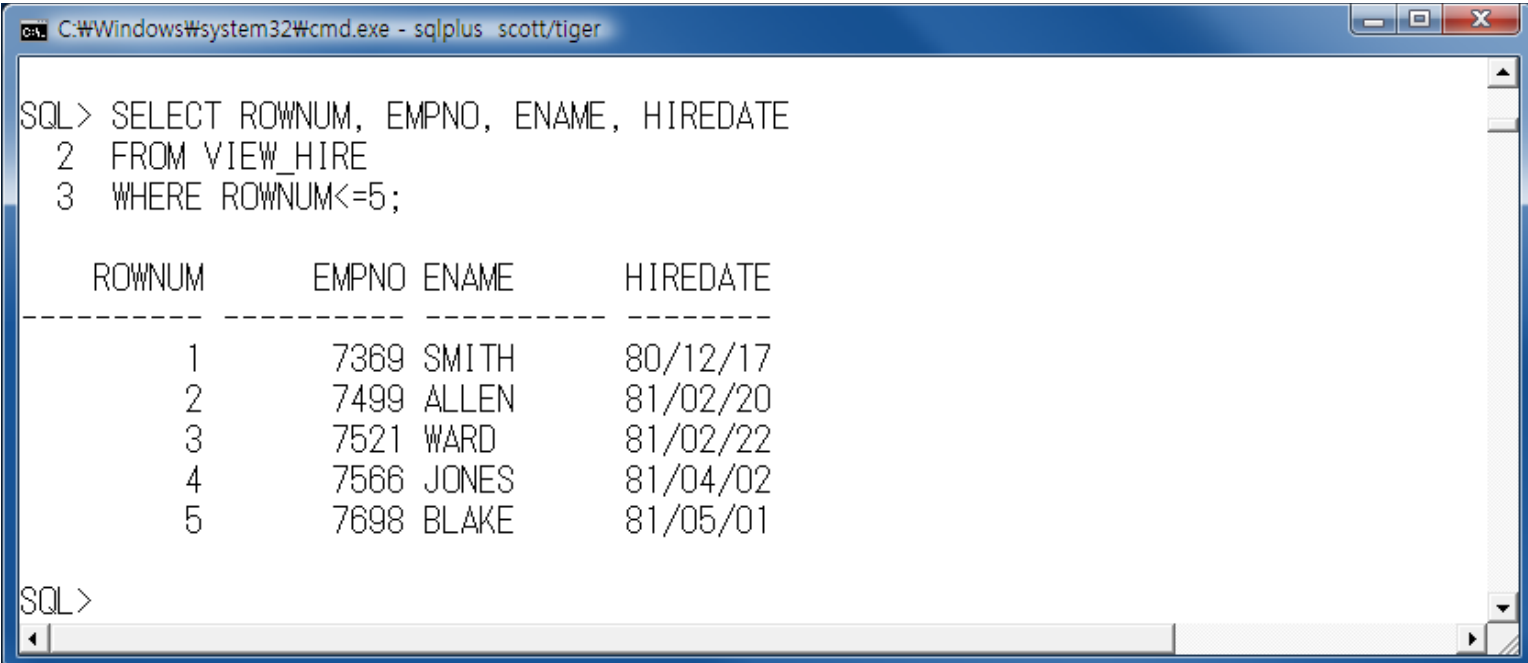
```
14 개의 행이 선택되었습니다.

SQL>
```

실습

3. 입사일이 빠른 사람 5명 가져오기

```
SELECT ROWNUM, EMPNO, ENAME, HIREDATE  
FROM VIEW_HIRE  
WHERE ROWNUM<=5;
```



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe - sqlplus scott/tiger". The user has entered the following SQL query:

```
SQL> SELECT ROWNUM, EMPNO, ENAME, HIREDATE  
2 FROM VIEW_HIRE  
3 WHERE ROWNUM<=5;
```

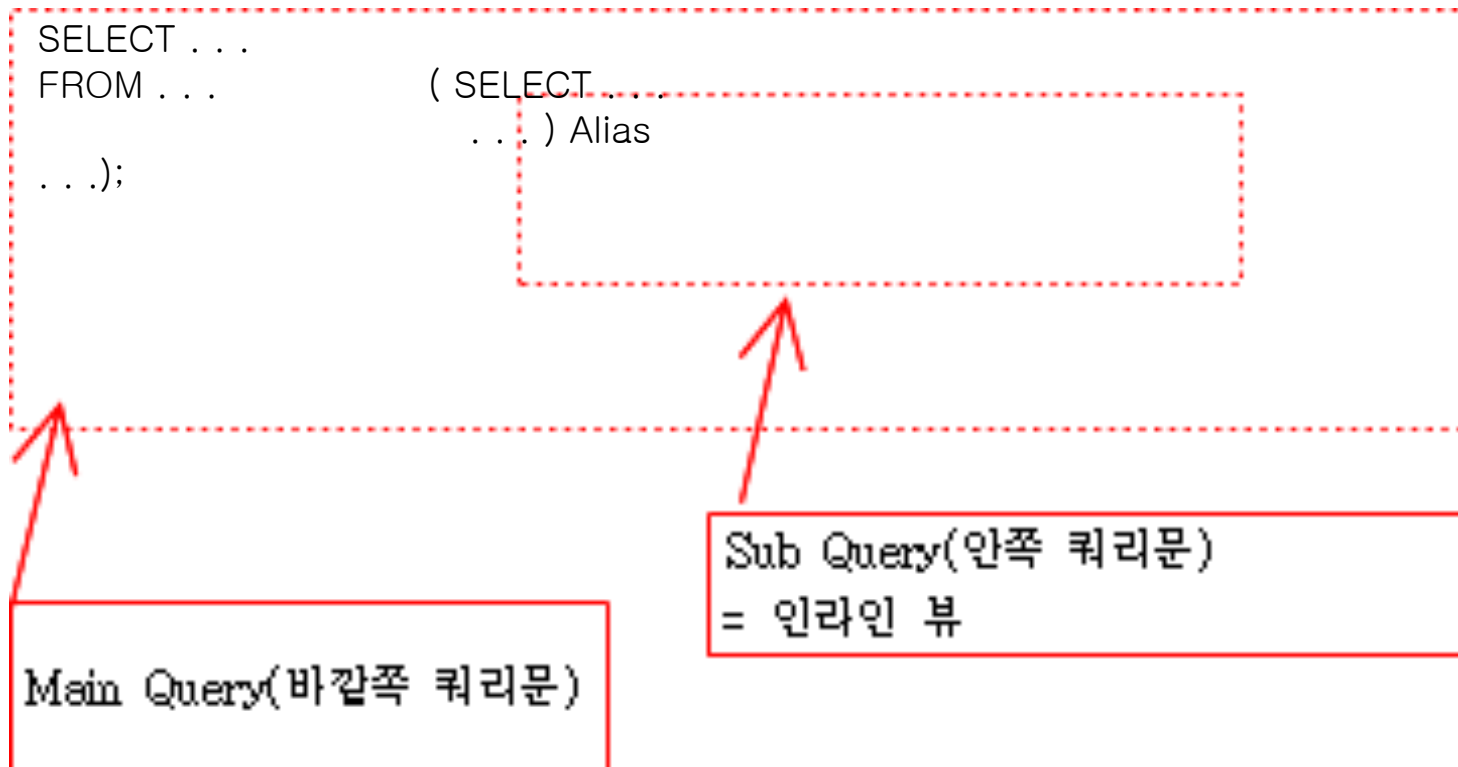
The query results are displayed in a table format:

ROWNUM	EMPNO	ENAME	HIREDATE
1	7369	SMITH	80/12/17
2	7499	ALLEN	81/02/20
3	7521	WARD	81/02/22
4	7566	JONES	81/04/02
5	7698	BLAKE	81/05/01

The command prompt shows the prompt "SQL>" at the bottom left.

6.2 TOP-N 구하기

- ❖ 인라인 뷰는 SQL 문장에서 사용하는 서브 쿼리의 일종으로 보통 FROM 절에 위치해서 테이블처럼 사용하는 것



실습

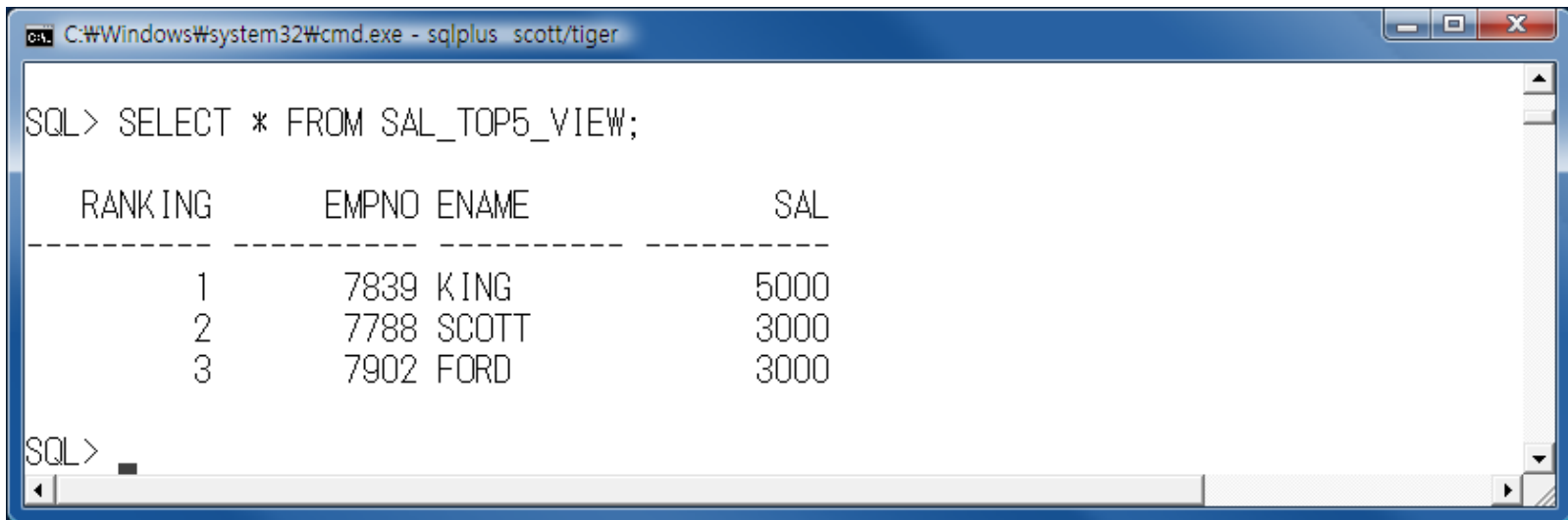
- ❖ 인라인 뷰를 사용해서 입사일이 빠른 사람 5명만을 조회하고자 하는 경우 FROM 절 다음인 VIEW_HIRE 위치에 VIEW_HIRE를 정의할 때 사용한 서브 쿼리문을 기술

```
SELECT EMPNO, ENAME, HIREDATE  
FROM ( SELECT ROWNUM rnum, EMPNO, ENAME, HIREDATE  
FROM EMP  
ORDER BY HIREDATE)  
WHERE rnum<=5;
```



연습문제

1. EMP 테이블에서 사원 번호(empno), 이름(ename), 업무(job), 부서번호(DEPTNO)를 포함하는 EMP_VIEW라는 이름의 VIEW를 생성
2. 1번에서 생성한 VIEW를 이용하여 DEPTNO가 10번 부서의 자료만 조회
3. EMP 테이블과 인라인 뷰를 사용하여 급여를 많이 받는 순서대로 3명만 출력하는 뷰 (SAL_TOP3_VIEW)를 작성하는데 급여가 같은 경우 이름의 내림차순으로 조회



```
C:\Windows\system32\cmd.exe - sqlplus scott/tiger

SQL> SELECT * FROM SAL_TOP5_VIEW;

  RANKING      EMPNO ENAME      SAL
-----
         1       7839 KING        5000
         2       7788 SCOTT        3000
         3       7902 FORD        3000

SQL>
```

The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe - sqlplus scott/tiger". Inside the window, the SQL command "SELECT * FROM SAL_TOP5_VIEW;" has been executed. The output is a table with four columns: RANKING, EMPNO, ENAME, and SAL. The data shows three rows: Rank 1 for King (7839) with a salary of 5000, Rank 2 for Scott (7788) with a salary of 3000, and Rank 3 for Ford (7902) with a salary of 3000. The prompt "SQL>" is visible at the bottom left of the window.

RANKING	EMPNO	ENAME	SAL
1	7839	KING	5000
2	7788	SCOTT	3000
3	7902	FORD	3000

Sequence

시퀀스

- ❖ 기본 키가 유일한 값을 갖도록 사용자가 직접 값을 생성해내려면 부담이 큼
- ❖ 시퀀스는 테이블 내의 유일한 숫자를 자동으로 생성하는 자동 번호 발생기로 시퀀스를 기본 키로 사용하게 되면 사용자의 부담을 줄일 수 있음
- ❖ 시퀀스를 생성하기 위한 기본 형식

CREATE SEQUENCE sequence_name

[START WITH n] ①

[INCREMENT BY n] ②

[{MAXVALUE n | NOMAXVALUE}] ③

[{MINVALUE n | NOMINVALUE}] ④

[{CYCLE | NOCYCLE}] ⑤

[{CACHE n | NOCACHE}] ⑥



시퀀스

❖ 옵션

- ① START WITH: 시퀀스 번호의 시작값을 지정할 때 사용하는 것으로 1부터 시작되는 시퀀스를 생성하려면 START WITH 1이라고 기술
- ② INCREMENT BY: 연속적인 시퀀스 번호의 증가치를 지정할 때 사용하는 것으로 1씩 증가하는 시퀀스를 생성하려면 INCREMENT BY 1이라고 기술
- ③ MAXVALUE n | NOMAXVALUE: MAXVALUE 은 시퀀스가 가질 수 있는 최대값을 지정하는 것으로 NOMAXVALUE를 지정하게 되면 ASCENDING 순서일 경우에는 10^{27} 승이고 DESCENDING 순서일 경우에는 -1로 설정
- ④ MINVALUE n | NOMINVALUE: MINVALUE 은 시퀀스가 가질 수 있는 최소값을 지정하는 것으로 NOMINVALUE을 지정하게 되면 ASCENDING 순서일 경우에는 1이고 DESCENDING 순서일 경우에는 10^{26} 승으로 설정
- ⑤ CYCLE | NOCYCLE: CYCLE 은 지정된 시퀀스 값이 최대값까지 증가가 완료되게 되면 다시 START WITH 옵션에 지정한 시작 값에서 다시 시퀀스를 시작하도록 하는 것이고 NOCYCLE 은 증가가 완료되게 되면 에러를 유발
- ⑥ CACHE n | NOCACHE: CACHE 은 메모리상의 시퀀스 값을 관리하도록 하는 것인데 기본 값은 200이며 NOCACHE는 원칙적으로 메모리 상에서 시퀀스를 관리하지 않음

시퀀스

- ❖ 시작 값이 1이고 1씩 증가하는 시퀀스 EMP_SEQ을 생성
CREATE SEQUENCE EMP_SEQ
INCREMENT BY 10
START WITH 10;
- ❖ CURRVAL, NEXTVAL
 - ✓ 시퀀스의 현재 값을 알아내기 위해서 CURRVAL를 사용하고 다음 값을 알아내기 위해서는 NEXTVAL를 사용
 - ✓ CURRVAL에 새로운 값이 할당되기 위해서는 NEXTVAL로 새로운 값을 한 번은 생성해야 함
 - ✓ NEXTVAL로 새로운 값을 생성한 다음에 이 값을 CURRVAL에 대체
 - ✓ NEXTVAL, CURRVAL을 사용할 수 있는 경우
 - 서브 쿼리가 아닌 SELECT 문
 - INSERT 문의 SELECT 절
 - INSERT 문의 VALUE절
 - UPDATE문의 SET 절
 - ✓ NEXTVAL, CURRVAL을 사용할 수 없는 경우
 - VIEW의 SELECT 절
 - DISTINCT 키워드가 있는 SELECT 문
 - GROUP BY, HAVING, ORDER BY 절이 있는 SELECT 문
 - SELECT, DELETE, UPDATE의 서브 쿼리
 - CREATE TABLE, ALTER TABLE 명령의 DEFAULT 값

실습

1. NEXTVAL로 새로운 값을 생성해야 합니다.

```
SELECT EMP_SEQ.NEXTVAL  
FROM DUAL;
```

2. 시퀀스의 현재 값을 알아내기 위해서 CURRVAL를 사용

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



시퀀스

- ❖ 시퀀스는 INSERT 연산과 같이 사용되어 컬럼 값을 자동으로 발생시키는 용도로 사용
- ❖ 사원 테이블을 생성하면서 사원 번호를 기본 키로 설정했는데 기본 키는 반드시 유일한 값을 가져야 하기 때문에 사용자가 새로운 사원을 추가할 때마다 유일한 사원번호를 INSERT 해야 하는 번거로움이 있음
- ❖ 사원 번호를 생성하는 시퀀스 객체를 사용하여 사원 번호가 자동 생성되도록 한다면 이러한 번거로움을 덜어줄 수 있음
- ❖ 시작 값이 1이고 1씩 증가하고 최댓값이 100000이 되는 시퀀스 EMP_PK_SEQ 생성
CREATE SEQUENCE EMP_PK_SEQ
START WITH 1
INCREMENT BY 1
MAXVALUE 100000 ;



시퀀스

- ❖ 사원 번호를 기본 키로 설정하여 EMP01란 테이블을 생성

```
DROP TABLE EMP01;
```

```
CREATE TABLE EMP01(  
EMPNO NUMBER(4) PRIMARY KEY,  
ENAME VARCHAR(10),  
HIREDATE DATE  
);
```

- ❖ 사원 번호를 저장하는 EMPNO 컬럼은 기본 키로 설정하였으므로 중복된 값을 가질 수 없으므로 EMP_PK_SEQ 시퀀스로부터 사원번호를 자동으로 할당받아 데이터를 추가하는 문장을 작성

```
INSERT INTO EMP01  
VALUES(EMP_PK_SEQ.NEXTVAL, 'JULIA' , SYSDATE);
```

- ❖ 시퀀스 삭제

```
DROP SEQUENCE EMP_SEQ;
```



시퀀스

❖ 시퀀스 수정

- ✓ 시퀀스를 변경하려면 ALTER SEQUENCE 문을 사용

ALTER SEQUENCE sequence_name

[INCREMENT BY n]

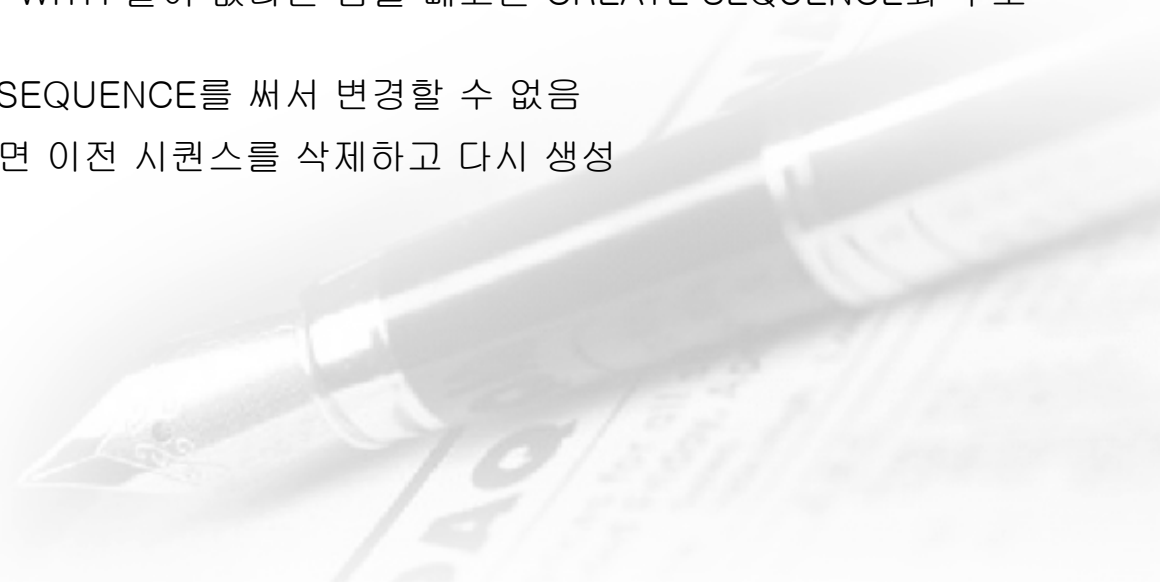
[{MAXVALUE n | NOMAXVALUE}]

[{MINVALUE n | NOMINVALUE}]

[{CYCLE | NOCYCLE}]

[{CACHE n | NOCACHE}]

- ✓ ALTER SEQUENCE는 START WITH 절이 없다는 점을 빼고는 CREATE SEQUENCE와 구조가 동일
- ✓ START WITH 옵션은 ALTER SEQUENCE를 써서 변경할 수 없음
- ✓ 다른 번호에서 다시 시작하려면 이전 시퀀스를 삭제하고 다시 생성



실습

- ❖ 시퀀스는 최대값을 지정하지 않으면 기본적으로 10^{27} 으로 지정되는데 사용자가 임의로 최대값을 지정할 수 있는데 MAXVALUE에 값을 지정하면 하면 되는데 10 부터 10씩 증가하면서 최대 30까지의 값을 갖는 시퀀스를 생성

```
CREATE SEQUENCE DEPT_DEPTNO_SEQ  
START WITH 10  
INCREMENT BY 10  
MAXVALUE 30;
```



연습문제

- ❖ 초기값 1부터 최대값 999,999까지 1씩 증가하는 TEST_SEQ라는 SEQUENCE를 생성
- ❖ 현재 SESSION을 이루고 있는 사용자가 사용할 수 있는 SRQUENCE를 조회
- ❖ 앞에서 작성한 SRQUENCE의 값을 다음 값으로 변경하고 현재 값을 조회
- ❖ 앞에서 생성한 SRQUENCE를 삭제

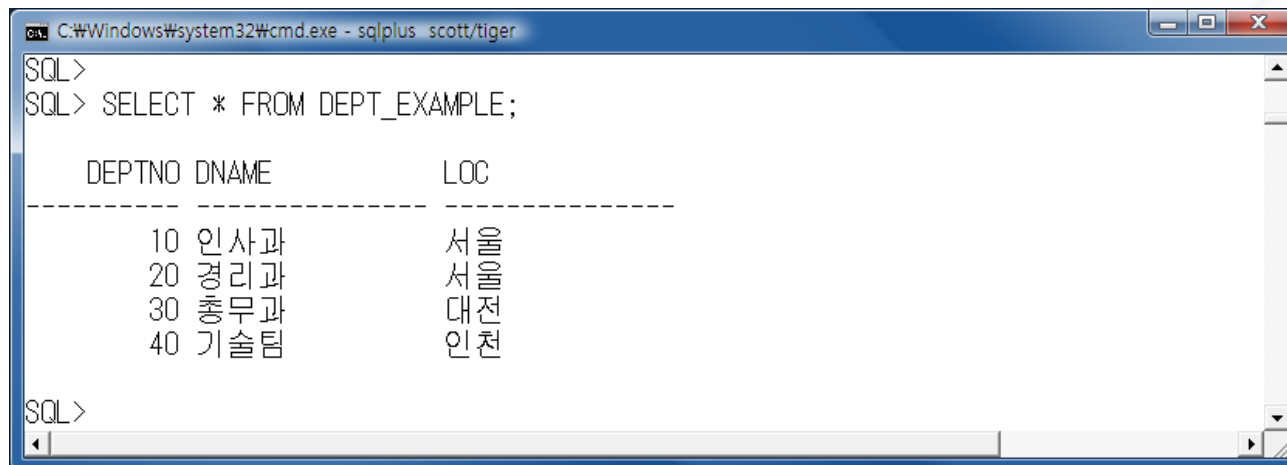


연습문제

- ❖ 부서 번호를 생성하는 시퀀스 객체를 생성하여 시퀀스 객체를 이용하여 부서 번호를 자동 생성하도록 하기 위해서 부서 테이블을 생성

```
CREATE TABLE DEPT_EXAMPLE(  
    DEPTNO NUMBER(4) PRIMARY KEY,  
    DNAME VARCHAR(15),  
    LOC VARCHAR(15));
```

1. DEPTNO 컬럼에 유일한 값을 가질 수 있도록 시퀀스 객체 생성(시퀀스 이름 : DEPT_EXAMPLE_SEQ)
2. 새로운 로우를 추가할 때마다 시퀀스에 의해서 다음과 같이 부서 번호가 자동 부여되도록 생성하고 데이터를 삽입

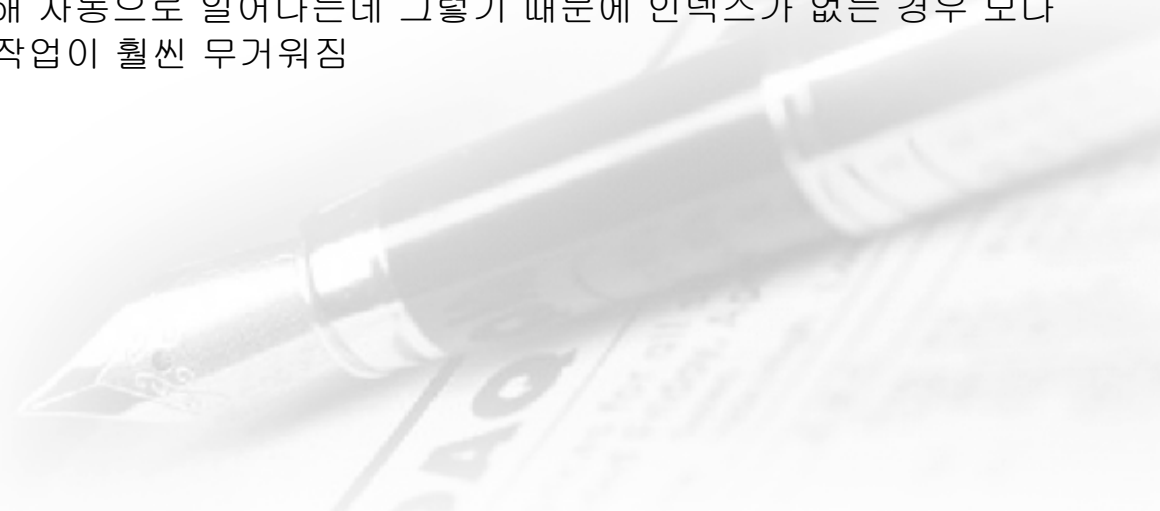


```
C:\Windows\system32\cmd.exe - sqlplus scott/tiger  
SQL>  
SQL> SELECT * FROM DEPT_EXAMPLE;  
  
   DEPTNO DNAME      LOC  
-----  
      10 인사과      서울  
      20 경리과      서울  
      30 총무과      대전  
      40 기술팀      인천  
  
SQL>
```

INDEX

인덱스

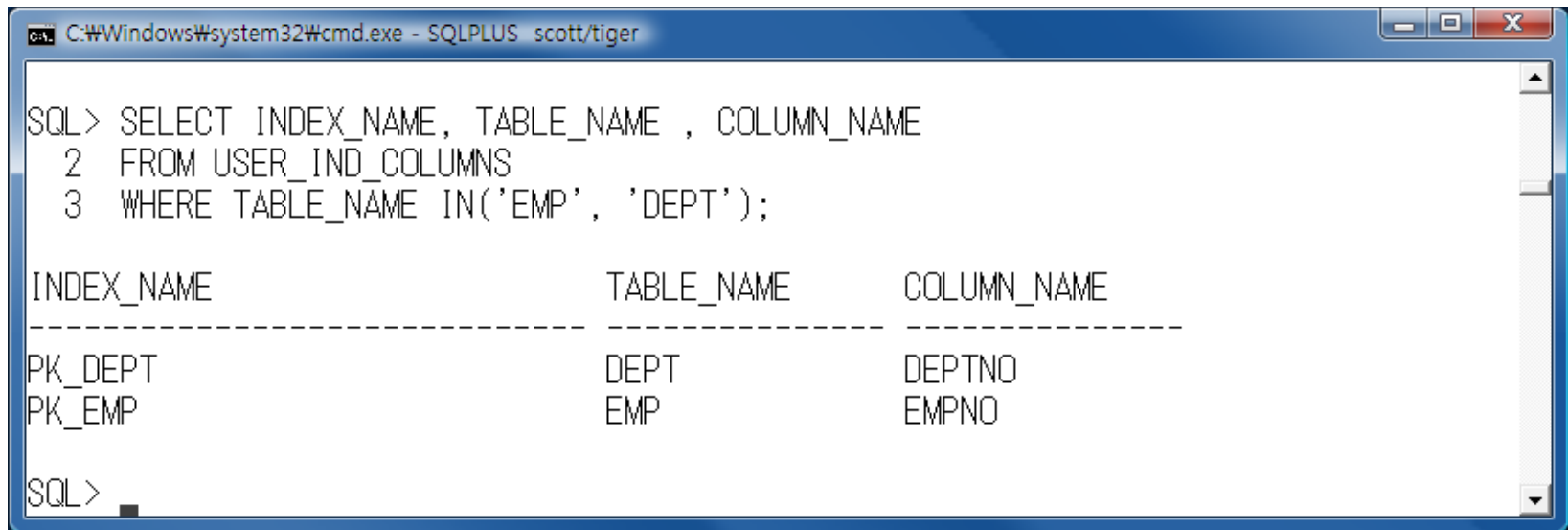
- ❖ SQL 명령문의 처리 속도를 향상시키기 위해서 컬럼에 대해서 생성하는 오라클 객체
- ❖ 인덱스의 장점
 - ✓ 검색 속도가 빨라짐
 - ✓ 시스템에 걸리는 부하를 줄여서 시스템 전체 성능을 향상시킴
- ❖ 인덱스의 단점
 - ✓ 오라클에서의 인덱스의 내부 구조는 B* 트리 형식으로 구성
 - ✓ 컬럼에 인덱스를 설정하면 이를 위한 B* 트리도 생성되어야 하기 때문에 인덱스를 생성하기 위한 시간도 필요하고 인덱스를 위한 추가적인 공간이 필요
 - ✓ 인덱스가 생성된 후에 새로운 행을 추가하거나 삭제할 경우 인덱스로 사용된 컬럼 값도 함께 변경되는 경우가 발생
 - ✓ 인덱스로 사용된 컬럼 값이 변경되는 이를 위한 내부 구조(B* 트리) 역시 함께 수정
 - ✓ 이 작업은 오라클 서버에 의해 자동으로 일어나는데 그렇기 때문에 인덱스가 없는 경우 보다 인덱스가 있는 경우에 DML 작업이 훨씬 무거워짐



인덱스

- ❖ 쿼리를 빠르게 수행하기 위한 용도로 사용되는 인덱스는 기본 키나 유일 키와 같은 제약 조건을 지정하면 따로 생성하지 않더라도 자동으로 생성
- ❖ 기본 키나 유일 키는 데이터 무결성을 확인하기 위해서 수시로 데이터를 검색하기 때문에 빠른 조회를 목적으로 오라클에서 내부적으로 해당 컬럼에 인덱스를 자동으로 생성하는 것
- ❖ USER_IND_COLUMNS 데이터 디렉터리 뷰로 인덱스의 생성 유무를 확인

```
SELECT INDEX_NAME, TABLE_NAME , COLUMN_NAME  
FROM USER_IND_COLUMNS  
WHERE TABLE_NAME IN('EMP', 'DEPT');
```



```
C:\Windows\system32\cmd.exe - SQLPLUS scott/tiger  
  
SQL> SELECT INDEX_NAME, TABLE_NAME , COLUMN_NAME  
2 FROM USER_IND_COLUMNS  
3 WHERE TABLE_NAME IN('EMP', 'DEPT');  
  
INDEX_NAME          TABLE_NAME          COLUMN_NAME  
-----  
PK_DEPT             DEPT                 DEPTNO  
PK_EMP              EMP                 EMPNO  
  
SQL>
```

인덱스

- ❖ 제약 조건에 의해 자동으로 생성되는 인덱스 외에 CREATE INDEX 명령어로 직접 인덱스를 생성할 수 있는데 기본 형식은 아래와 같음

```
CREATE INDEX index_name  
ON table_name (column_name);
```

- ❖ CREATE INDEX 다음에 인덱스 객체 이름을 지정
- ❖ 어떤 테이블의 어떤 컬럼에 인덱스를 설정할 것인지를 결정하기 위해서 ON 절 다음에 테이블 이름과 컬럼 이름을 기술
- ❖ 테이블 EMP01의 컬럼 중에서 이름(ENAME)에 대해서 인덱스를 생성

```
CREATE INDEX IDX_EMP01_ENAME  
ON EMP01(ENAME);
```
- ❖ 오라클은 DROP INDEX 명령어를 제공
- ❖ EMP01 테이블의 IDX_EMP01_ENAME 인덱스를 제거

```
DROP INDEX IDX_EMP01_ENAME;
```



인덱스

인덱스를 사용해야 하는 경우	인덱스를 사용하지 말아야 하는 경우
테이블에 행의 수가 많을 때	테이블에 행의 수가 적을 때
WHERE 문에 해당 컬럼이 많이 사용될 때	WHERE 문에 해당 컬럼이 자주 사용되지 않을 때
검색 결과가 전체 데이터의 2%~4% 정도 일 때	검색 결과가 전체 데이터의 10%~15% 이상 일 때
JOIN에 자주 사용되는 컬럼이나 NULL을 포함하는 컬럼이 많은 경우	테이블에 DML 작업이 많은 경우 즉, 입력 수정 삭제 등이 자주 일어날 때



인덱스

❖ 인덱스 종류

- ✓ 고유 인덱스(Unique Index)
- ✓ 비 고유 인덱스(NonUnique Index)
- ✓ 단일 인덱스(Single Index)
- ✓ 결합 인덱스(Composite Index)
- ✓ 함수 기반 인덱스(Function Based Index)

❖ 고유 인덱스 와 비고유 인덱스

- ✓ 고유 인덱스(유일 인덱스라고도 부름)는 기본키나 유일키처럼 유일한 값을 갖는 컬럼에 대해서 생성하는 인덱스
- ✓ 반면 비고유 인덱스는 중복된 데이터를 갖는 컬럼에 대해서 인덱스를 생성하는 경우
- ✓ 고유 인덱스를 설정하려면 UNIQUE 옵션을 추가해서 인덱스를 생성

```
CREATE UNIQUE INDEX index_name
```

```
ON table_name (column_name);
```

❖ 단일 인덱스와 결합 인덱스

- ✓ 한 개의 컬럼으로 구성된 인덱스는 단일 인덱스
- ✓ 두 개 이상의 컬럼으로 인덱스를 구성하는 것을 결합 인덱스

실습

- ❖ 부서 번호와 부서명을 결합하여 인덱스를 설정할 수 있는데 이것이 결합 인덱스

```
CREATE INDEX IDX_DEPT01_COM  
ON DEPT01(DEPTNO, DNAME);
```
- ❖ 데이터 디렉터리인 USER_IND_COLUMNS 테이블에서 IDX_DEPT01_COM 인덱스는 DEPTNO와 DNAME 두 개의 컬럼이 결합된 것임을 확인할 수 있음

```
SELECT INDEX_NAME, COLUMN_NAME  
FROM USER_IND_COLUMNS  
WHERE TABLE_NAME = 'DEPT01';
```



인덱스

- ❖ 검색조건으로 WHERE SAL = 3000이 아니라 WHERE SAL*12 = 3600와 같이 SELECT 문 WHERE 절에 산술 표현 또는 함수를 사용하는 경우가 있음
- ❖ 이 경우 만약 SAL 컬럼에 인덱스가 걸려 있다면 인덱스를 이용해서 빠르게 접근할 것이라고 생각할 수 있지만 SAL 컬럼에 인덱스가 있어도 SAL*12는 인덱스를 생성하지 못함
- ❖ 인덱스 걸린 컬럼이 수식으로 정의 되어 있거나 SUBSTR 등의 함수를 사용해서 변형이 일어난 경우는 인덱스를 이용할 수 없음
- ❖ 이러한 수식으로 검색하는 경우가 많다면 아예 수식이나 함수를 적용하여 인덱스를 만들 수 있는데 SAL*12로 인덱스를 만들어 놓으면 SAL*12가 검색 조건으로 사용될 시 해당 인덱스를 이용할 수 있음



실습

- ❖ 사원 테이블에서 급여 컬럼에 저장된 데이터로 연봉을 인덱스로 지정하기 위한 산술 표현을 인덱스로 설정

1. 함수 기반 인덱스를 생성

```
CREATE INDEX IDX_EMP_ANNSAL  
ON EMP(SAL*12);
```

- ## 2. 다음은 데이터 디렉터리인 USER_IND_COLUMNS에 함수 기반 인덱스가 기록되어 있는 것을 확인하기 위한 쿼리문

```
SELECT INDEX_NAME, COLUMN_NAME  
FROM USER_IND_COLUMNS  
WHERE TABLE_NAME = 'EMP';
```



연습문제

1. INDEX의 장단점을 기술
2. 테이블 생성시 자동적으로 생성되는 INDEX가 있는데 어떤 제약 조건을 기술하면 생성되는가?
3. EMP 테이블의 직급(JOB) 컬럼을 인덱스로 설정하되 인덱스 이름을 IDX_EMP_JOB 설정하고 확인
- 4.

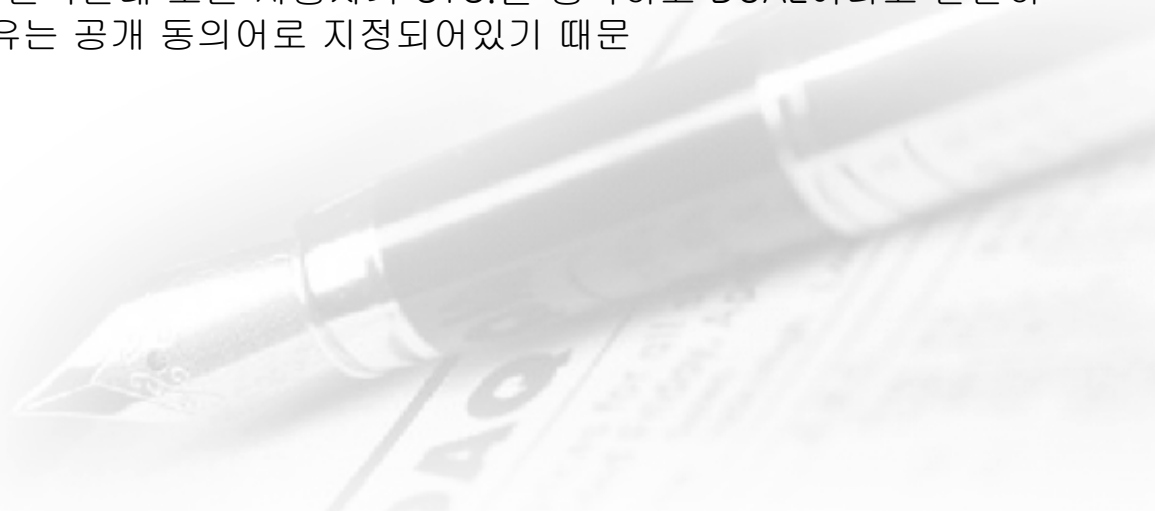




SYNONYM

동의어

- ❖ 데이터베이스의 객체에 대한 소유권은 해당 객체를 생성한 사용자에게 있기 때문에 다른 사용자가 객체에 접근하기 위해서는 소유자로부터 접근 권한을 부여받아야 하며 또한 다른 사용자가 소유한 객체에 접근하기 위해서는 소유자의 이름을 객체 앞에 지정해야 함
- ❖ 객체를 조회할 때마다 일일이 객체의 소유자를 지정하는 것이 번거로울 경우 동의어를 정의하면 긴 이름대신 간단한 이름으로 접근할 수 있음
- ❖ 동의어는 개별 사용자를 대상으로 하는 비공개 동의어와 전체 사용자를 대상으로 한 공개 동의어가 있음
 - ✓ 비공개 동의어 - 객체에 대한 접근 권한을 부여받은 사용자가 정의한 동의어로 해당 사용자만 사용할 수 있음
 - ✓ 공개 동의어 - 권한을 주는 사용자가 정의한 동의어로 누구나 사용할 수 있으며 공개 동의어는 DBA 권한을 가진 사용자만이 생성할 수 있는데 SYNONYM 앞에 PUBLIC를 붙여서 정의
- ❖ DUAL은 원래 SYS가 소유하는 테이블 명이므로 다른 사용자가 DUAL 테이블에 접근하려면 SYS.DUAL로 표현해야 하는 것이 원칙인데 모든 사용자가 SYS.을 생략하고 DUAL이라고 간단하게 사용했는데 이럴 수 있었던 이유는 공개 동의어로 지정되어있기 때문



동의어

- ❖ 동의어를 정의하기 위한 CREATE SYNONYM 명령어의 기본 형식

```
CREATE [PUBLIC] SYNONYM synonym_name  
FOR user_name.object_name;
```

- ❖ synonym_name은 user_name.object_name에 대한 별칭
- ❖ user_name은 객체를 소유한 오라클 사용자이고 object_name은 동의어를 만들려는 데이터베이스 객체 이름
- ❖ SALGRADE 동의어로 GUBUN를 생성하고 동의어로 조회

```
CREATE SYNONYM gubun  
FOR salgrade;
```

```
SELECT *  
FROM gubun;
```



동의어

- ❖ 동의어를 제거하기 위해 DROP SYNONYM문장을 사용
- ❖ Syntax
DROP [PUBLIC] SYNONYM synonym_name;
- ❖ 앞에서 생성한 동의어를 삭제
DROP SYNONYM gubun;
- ❖ 연습문제 - EMP 테이블을 EMPLOYEE라는 동의어를 생성하여라.

