



Database 프로그래밍을 위한
오라클 명령어

11

가상 테이블 view

강 사 : 김 진 성



목 차

1

뷰의 개념

2

뷰의 내부구조

3

뷰를 사용하는 이유

4

뷰의 종류



목 차

5

뷰 삭제와 다양한 옵션

6

뷰 활용하기



<실습하기> 뷰 정의하기

- 뷰를 생성하려는데 다음과 같이 권한을 설정하였다면 30번 부서에 소속된 직원들의 사번과 이름과 부서번호를 출력하기 위한 SELECT 문을 하나의 뷰로 다시 정의한다.

1. 뷰를 정의

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

- 2. 뷰를 생성할 때 컬럼 이름을 명시하지 않으면 뷰(EMP_VIEW30)를 정의하는 기본 테이블(EMP_COPY)의 컬럼 명을 상속받아 사용한다.

```
DESC EMP_VIEW30
```





<실습하기> 뷰 정의하기

5. 데이터를 추가하면서 EMP_SEQ 시퀀스로부터 사원번호를 자동으로 할당받았는지 EMP01 테이블의 내용을 확인

```
SELECT * FROM EMP01;
```

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

SQL> SELECT * FROM EMP01;

      EMPNO ENAME      HIREDATE
-----
         1  JULIA      09/01/22

SQL>
```



01. 뷰의 개념



- ❖ 뷰(View)는 한마디로 물리적인 테이블을 근거한 논리적인 가상 테이블이라고 정의할 수 있다.
- ❖ 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 것입니다.
- ❖ 뷰는 기본 테이블에서 파생된 객체로서 기본 테이블에 대한 하나의 쿼리문이다.
- ❖ 뷰(View)란 실제 테이블에 저장된 데이터를 뷰를 통해서 볼 수 있도록 한다.
- ❖ 사용자에게 주어진 뷰를 통해서 기본 테이블을 제한적으로 사용하게 된다.

1.1 뷰의 기본 테이블



- ❖ 뷰는 이미 존재하고 있는 테이블에 제한적으로 접근하도록 한다.
- ❖ 뷰를 생성하기 위해서는 실질적으로 데이터를 저장하고 있는 물리적인 테이블이 존재해야 하는데 이 테이블을 기본 테이블이라고 한다.
- ❖ 우선 시스템에서 제공하는 dept 테이블과 emp 테이블의 내용이 변경되는 것을 막기 위해서 테이블의 내용을 복사한 새로운 테이블을 생성한 후에 이를 기본 테이블로 사용한다.

〈실습하기〉 뷰의 기본 테이블 생성하기

뷰의 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;
```


1.2 뷰 정의하기



- ❖ 뷰를 생성하여 자주 사용되는 SELECT 문을 간단하게 접근하는 방법을 학습해보자. 다음은 뷰를 생성하기 위한 기본 형식이다.

```
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 시작한다. AS 다음은 마치 서브 쿼리문과 유사하다.
- ❖ *subquery*에는 우리가 지금까지 사용하였던 SELECT 문을 기술하면 됩니다.

1.2 뷰 정의하기



❖ CREATE OR RELPACE VIEW

- 뷰를 만들 때 CREATE OR RELPACE VIEW 대신 그냥 CREATE VIEW만 사용해도 된다.
- 그러나 그냥 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)할 수 있다.
- 그래서 대부분 뷰를 만들 때는 CREATE VIEW 대신 CREATE OR REPLACE VIEW를 사용한다.

❖ FORCE

- FORCE를 사용하면, 기본 테이블의 존재 여부에 상관없이 뷰를 생성한다.

1.2 뷰 정의하기



❖ WITH CHECK OPTION

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

❖ WITH READ ONLY

- WITH READ ONLY를 사용하면 해당 뷰를 통해서만 가능하며 INSERT/UPDATE/DELETE를 할 수 없게 된다.
- 만약 이것을 생략한다면, 뷰를 사용하여 추가, 수정, 삭제(INSERT/UPDATE/DELETE)가 모두 가능

1.2 뷰 정의하기



- ❖ 뷰를 만들기 전에 어떤 경우에 뷰를 사용하게 되는지 다음 예를 통해서 뷰가 필요한 이유를 알아보자.
- ❖ 만일, 30번 부서에 소속된 직원들의 사번과 이름과 부서번호를 자주 검색한다고 한다면 다음과 같은 SELECT 문을 여러 번 입력해야 한다.

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

- ❖ 위와 같은 결과를 출력하기위해서 매번 SELECT 문을 입력하기란 번거로운 일이다.
- ❖ 뷰는 이와 같이 번거로운 SELECT 문을 매번 입력하는 대신 보다 쉽게 원하는 결과를 얻고자 하는 바램에서 출발한 개념이다.

1.2 뷰 정의하기



- ❖ 자주 사용되는 30번 부서에 소속된 직원들의 사번과 이름과 부서번호를 출력하기 위한 SELECT문을 하나의 뷰로 정의해보자

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

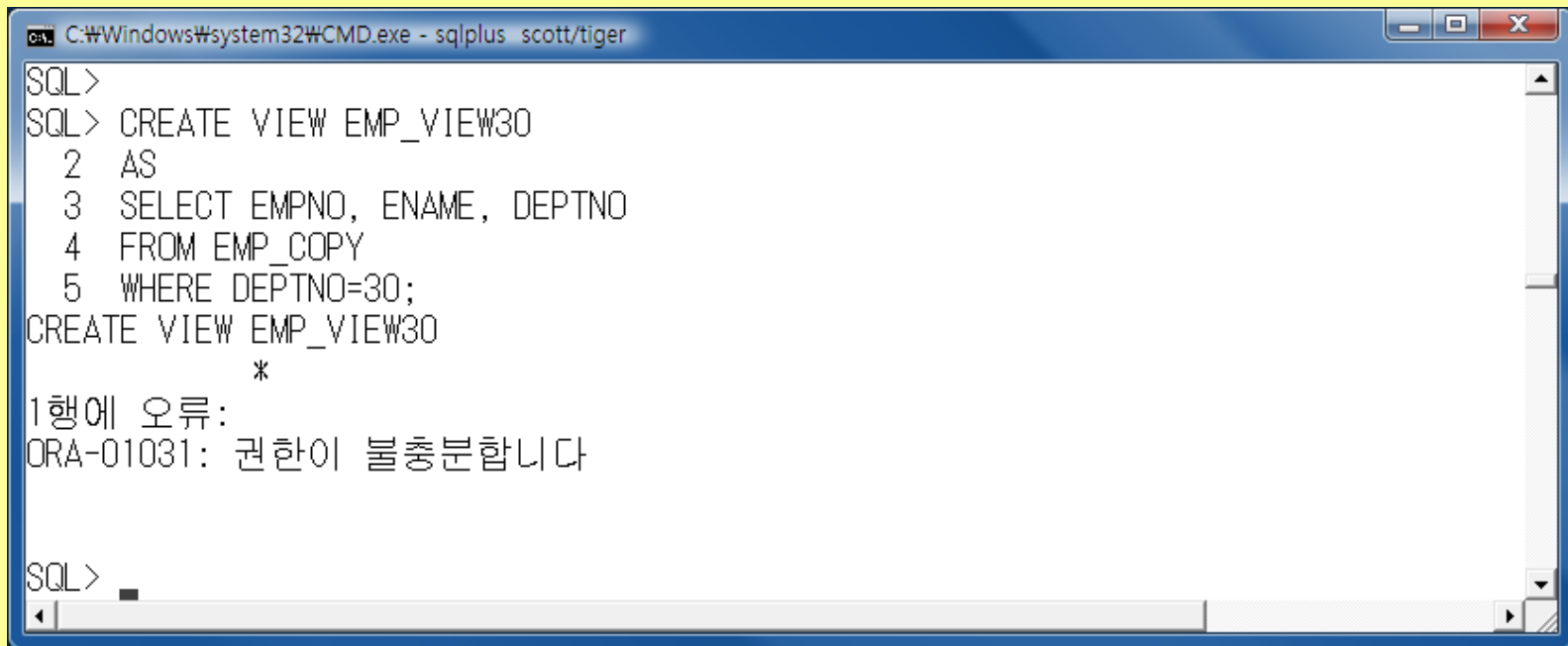
- ❖ 뷰는 테이블에 접근(SELECT)한 것과 동일한 방법으로 결과를 얻을 수 있다.

```
SELECT * FROM EMP_VIEW30;
```

〈실습하기〉 뷰를 생성할 권한이 불충분하다면

30번 부서에 소속된 직원들의 사번과 이름과 부서번호를 출력하기 위한 SELECT문을 하나의 뷰로 정의해 보자.

1. 뷰를 생성하려는데 다음과 같이 권한이 불충분하다고 오류가 발생할 경우가 있다.



```
C:\Windows\system32\CMD.exe - sqlplus scott/tiger
SQL>
SQL> CREATE VIEW EMP_VIEW30
2  AS
3  SELECT EMPNO, ENAME, DEPTNO
4  FROM EMP_COPY
5  WHERE DEPTNO=30;
CREATE VIEW EMP_VIEW30
          *
1행에 오류:
ORA-01031: 권한이 불충분합니다

SQL>
```

〈실습하기〉 뷰를 생성할 권한이 불충분하다면

2. 이럴 경우에는 DBA인 SYSTEM 계정으로 로그인하여 뷰를 생성할 권한을 부여한다.

특정 사용자에게 대해서 아무 문제없이 뷰가 생성된다면 괜찮지만, 그렇지 않을 경우 GRANT 명령어로 특정 사용자에게 권한을 부여해야 한다.

아래 문장은 SCOTT 사용자에게 테이블을 생성할 CREATE VIEW 권한을 부여하는 명령문이다.

이 명령어는 DBA 권한이 있는 사용자만이 부여할 수 있으므로 SYSTEM 계정으로 접속한다.

```
CONN system/manager
```

```
GRANT CREATE VIEW TO scott;
```

〈실습하기〉 뷰 정의하기

뷰를 생성하려는데 다음과 같이 권한을 설정하였다면 30번 부서에 소속된 사원들의 사번과 이름과 부서번호를 출력하기 위한 SELECT문을 하나의 뷰로 다시 정의해 보자.

1. 뷰를 정의

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

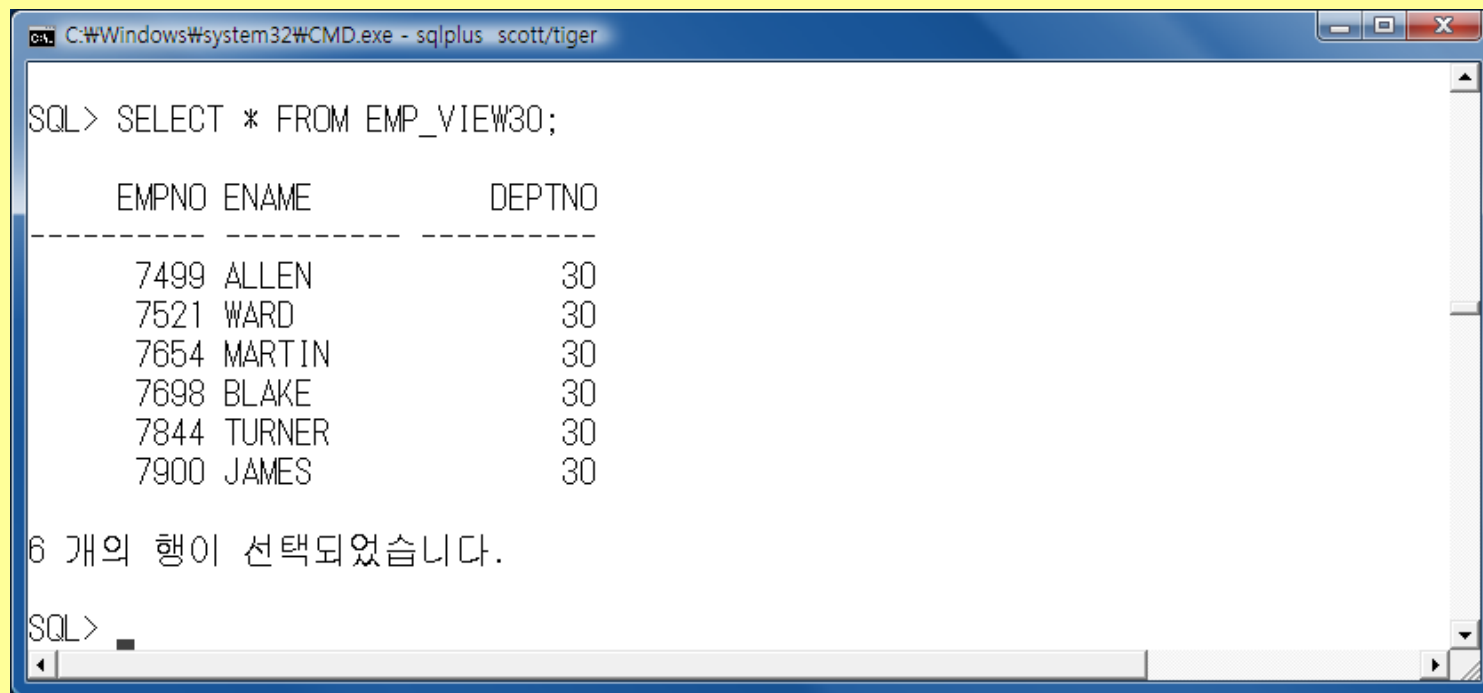
2. 뷰를 생성할 때 컬럼 이름을 명시하지 않으면 뷰(EMP_VIEW30)를 정의하는 기본 테이블(EMP_COPY)의 컬럼 명을 상속받아 사용한다. 다음은 생성된 뷰의 구조를 살펴보자.

```
DESC EMP_VIEW30
```


〈실습하기〉 뷰 정의하기

3. 뷰의 내용을 출력해 봅시다. 테이블의 내용을 출력하는 것과 동일한 방식으로 수행하면 됩니다. SELECT * FROM 다음에 테이블 명 대신에 뷰 이름을 기술하면 됩니다.

```
SELECT * FROM EMP_VIEW30;
```



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

SQL> SELECT * FROM EMP_VIEW30;

  EMPNO ENAME      DEPTNO
-----
  7499 ALLEN        30
  7521 WARD          30
  7654 MARTIN       30
  7698 BLAKE        30
  7844 TURNER       30
  7900 JAMES        30

6 개의 행이 선택되었습니다.

SQL>
```

02. 뷰의 내부구조와 USER_VIEWS 데이터 디렉터리



- ❖ 뷰는 물리적으로 데이터를 저장하고 있지 않다고 하였다. 그런데도 다음과 같은 질의 문을 수행할 수 있는 이유가 무엇일까요?

```
SELECT * FROM EMP_VIEW30;
```

- ❖ EMP_VIEW30라는 뷰는 데이터를 물리적으로 저장하고 있지 않다.
- ❖ CREATE VIEW 명령어로 뷰를 정의할 때 AS 절 다음에 기술한 쿼리 문장 자체를 저장하고 있다.
- ❖ 뷰 정의할 때 기술한 쿼리문이 궁금하다면 데이터 디렉터리 USER_VIEWS 테이블의 TEXT 컬럼 값을 살펴보면 된다.

02. 뷰의 내부구조와 USER_VIEWS 데이터 디렉터리



❖ USER_VIEWS에서 테이블 이름과 텍스트만 출력해 보자.

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

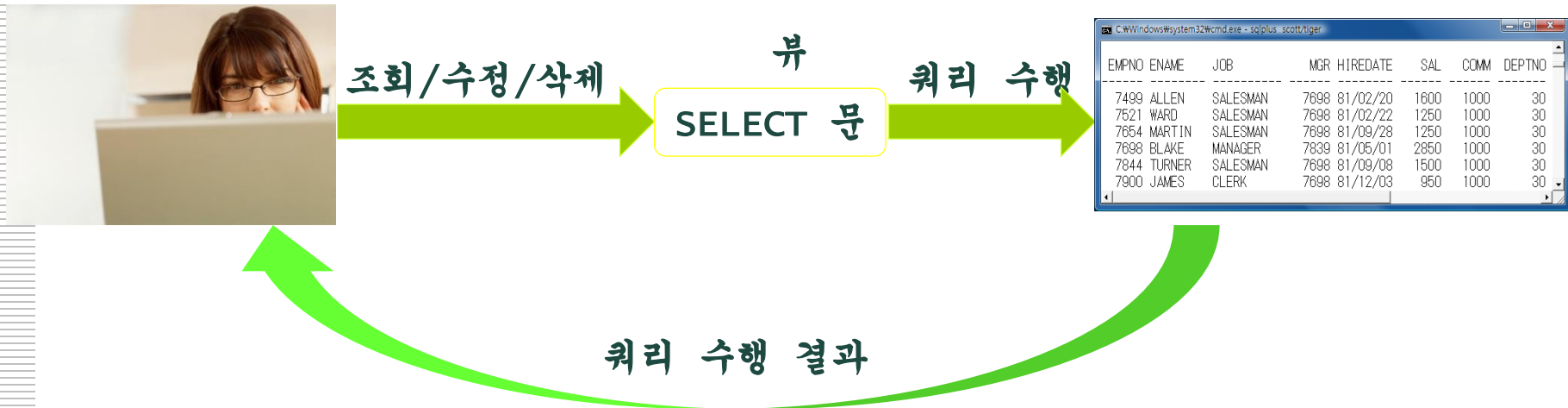
```
C:\Windows\system32\CMD.exe - sqlplus scott/tiger  
  
SQL> COLUMN VIEW_NAME FORMAT A15  
SQL> COLUMN TEXT FORMAT A50  
SQL> SELECT VIEW_NAME, TEXT  
2 FROM USER_VIEWS;  
  
VIEW_NAME          TEXT  
-----  
EMP_VIEW30         SELECT EMPNO, ENAME, DEPTNO  
                   FROM EMP_COPY  
    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

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

02. 뷰의 내부구조와 USER_VIEWS 데이터 디렉터리



- ❖ 뷰의 동작 원리를 이해하기 위해서 뷰에 대한 질의가 어떻게 내부적으로 처리되는지 자세히 살펴보도록 하자.



1. 사용자가 뷰에 대해서 질의를 하면 USER_VIEWS에서 뷰에 대한 정의를 조회
2. 기본 테이블에 대한 뷰의 접근 권한을 살핀다.
3. 뷰에 대한 질의를 기본 테이블에 대한 질의로 변환
4. 기본 테이블에 대한 질의를 통해 데이터를 검색
5. 검색된 결과를 출력

02. 뷰의 내부구조와 USER_VIEWS 데이터 디렉터리



- ❖ 우리가 앞에서 생성한 뷰인 EMP_VIEW30를 SELECT문의 FROM절 다음에 기술하여 질의를 하면 오라클 서버는 USER_VIEWS에서 EMP_VIEW30를 찾아 이를 정의할 때 기술한 서브 쿼리문이 저장된 TEXT 값을 EMP_VIEW30 위치로 가져간다.

```
SELECT *  
FROM EMP_VIEW30;
```

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

- ❖ 질의는 기본 테이블인 EMP_COPY를 통해 일어난다. 즉, 기본 테이블인 EMP_COPY에 대해서 서브 쿼리문을 수행하게 된다. 이러한 동작 원리 덕분에 뷰가 실질적으로 데이터를 저장하고 있지 않더라도 데이터를 검색할 수 있다.

〈실습하기〉 뷰와 기본 테이블 관계 파악하기

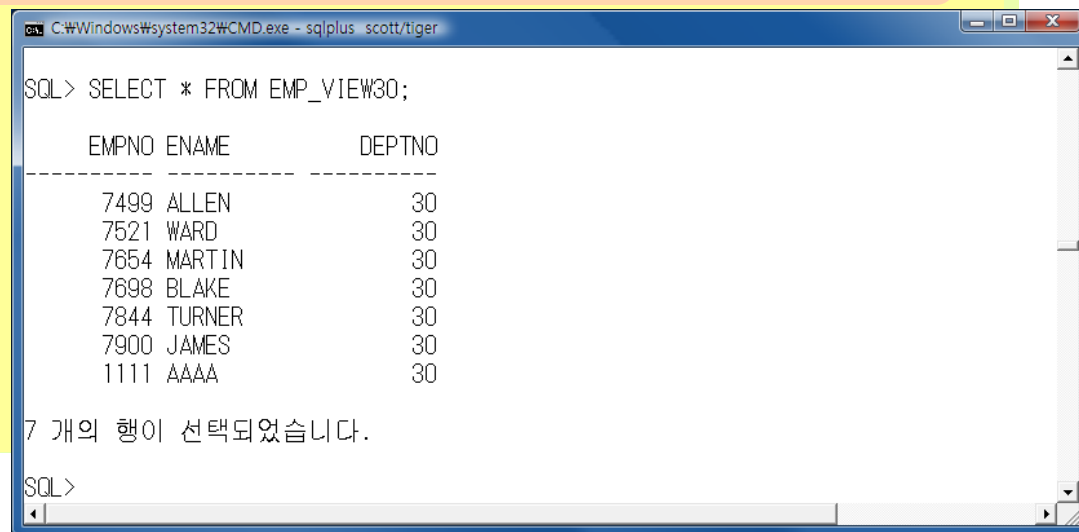
기본 테이블을 가져다가 쿼리문을 수행한다는 것을 증명하기 위해서 간단한 예를 살펴보도록 하겠습니다.

1. 다음은 EMP_VIEW30 뷰에 행을 하나 추가하는 문장입니다.

```
INSERT INTO EMP_VIEW30  
VALUES(1111, 'AAAA', 30);
```

2. INSERT 문으로 뷰에 새로운 행을 추가하였습니다. 뷰의 내용을 출력해 보면 추가된 행이 뷰에 존재하고 있음을 확인할 수 있습니다.

```
SELECT * FROM EMP_VIEW30;
```



```
C:\Windows\system32\CMD.exe - sqlplus scott/tiger  
  
SQL> SELECT * FROM EMP_VIEW30;  
  
  EMPNO ENAME      DEPTNO  
-----  
   7499 ALLEN          30  
   7521 WARD            30  
   7654 MARTIN         30  
   7698 BLAKE          30  
   7844 TURNER         30  
   7900 JAMES          30  
   1111 AAAA           30  
  
7 개의 행이 선택되었습니다.  
  
SQL>
```

〈실습하기〉 뷰와 기본 테이블 관계 파악하기

3. 뷰 뿐만 아니라 기본 테이블의 내용을 출력해 보면 INSERT 문에 의해서 뷰에 추가한 행이 테이블에도 존재함을 확인할 수 있습니다.

```
SELECT * FROM EMP_COPY;
```



EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	80/12/17	800		20
7499	ALLEN	SALESMAN	7698	81/02/20	1600	300	30
7521	WARD	SALESMAN	7698	81/02/22	1250	500	30
7566	JONES	MANAGER	7839	81/04/02	2975		20
7654	MARTIN	SALESMAN	7698	81/09/28	1250	1400	30
7698	BLAKE	MANAGER	7839	81/05/01	2850		30
7782	CLARK	MANAGER	7839	81/06/09	2450		10
7788	SCOTT	ANALYST	7566	87/04/19	3000		20
7839	KING	PRESIDENT		81/11/17	5000		10
7844	TURNER	SALESMAN	7698	81/09/08	1500	0	30
7876	ADAMS	CLERK	7788	87/05/23	1100		20
7900	JAMES	CLERK	7698	81/12/03	950		30
7902	FORD	ANALYST	7566	81/12/03	3000		20
7934	MILLER	CLERK	7782	82/01/23	1300		10
1111	AAAA						30

15 개의 행이 선택되었습니다.

02. 뷰의 내부구조와 USER_VIEWS 데이터 디렉터리



- ❖ INSERT 문에 뷰(EMP_VIEW30)를 사용하였지만, 뷰는 쿼리문에 대한 이름일 뿐이기 때문에 새로운 행은 기본 테이블(EMP_COPY)에 실질적으로 추가되는 것임을 알 수 있습니다. 뷰(EMP_VIEW30)의 내용을 확인하기 위해 SELECT문을 수행하면 변경된 기본 테이블(EMP_COPY)의 내용에서 일부를 서브 쿼리한 결과를 보여줍니다.
- ❖ 뷰는 실질적인 데이터를 저장한 기본 테이블을 볼 수 있도록 한 투명한 창입니다, 기본 테이블의 모양이 바뀐 것이고 그 바뀐 내용을 뷰라는 창을 통해서 볼 뿐입니다. 뷰에 INSERT 뿐만 아니라 UPDATE, DELETE 모두 사용할 수 있는데, UPDATE, DELETE 쿼리문 역시 뷰의 텍스트에 저장되어 있는 기본 테이블이 변경하는 것입니다.
- ❖ 이 정도면 뷰가 물리적인 테이블을 근거로 한 논리적인 가상 테이블이란 말의 의미를 이해할 수 있으리라고 생각됩니다.

03. 뷰를 사용하는 이유

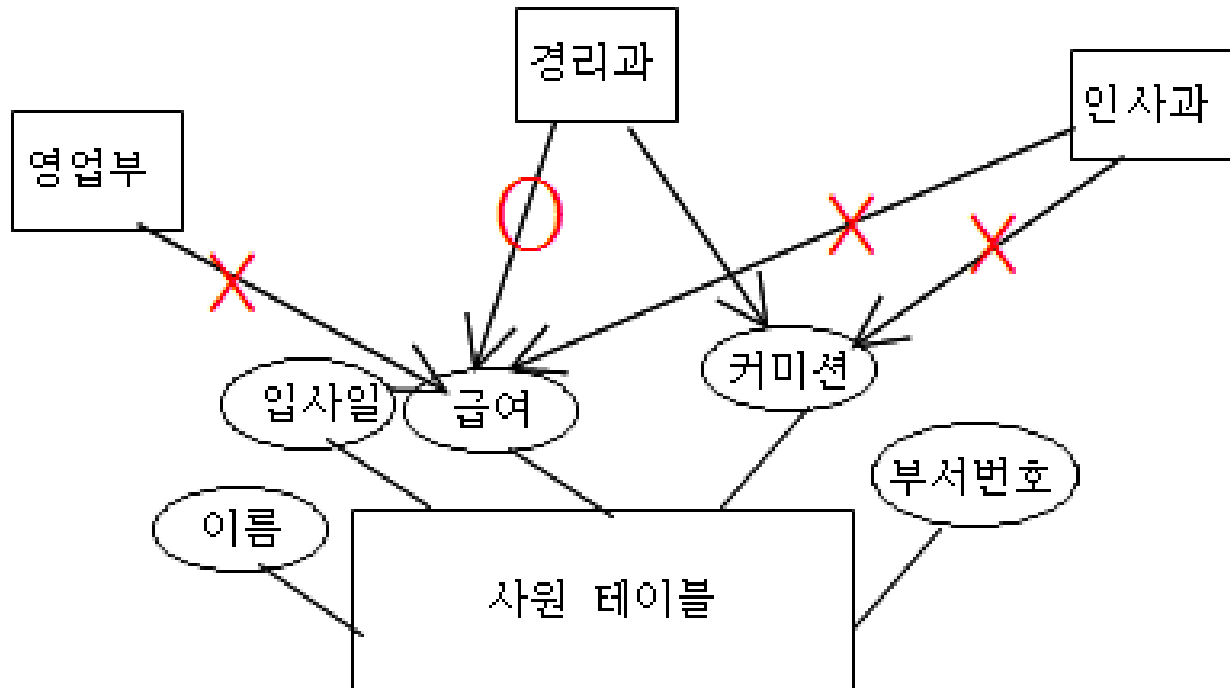


❖ 뷰를 사용하는 이유는 두 가지로 설명할 수 있습니다.

1. 복잡하고 긴 쿼리문을 뷰로 정의하면 접근을 단순화시킬 수 있다.
2. 보안에 유리하다.

- ❖ 1. 번의 경우는 이미 살펴보았으므로 2.번의 경우만 이해해 보도록 합시다.
- ❖ 사용자마다 특정 객체만 조회할 수 있도록 권한을 부여할 수 있기에 동일한 테이블을 접근하는 사용자마다에 따라 서로 다르게 보도록 여러 개의 뷰를 정의해 놓고 특정 사용자만이 해당 뷰에 접근할 수 있도록 합니다.

03. 뷰를 사용하는 이유



- ❖ 예를 들어 사원 테이블에 개인 적인 정보인 급여와 커미션은 부서에 따라 접근을 제한해야 합니다. 급여나 커미션 모두에 대해서 인사과에서는 조회할 수 없도록 하고 경리과에서는 이 모두가 조회될 수 있도록 하지만 영업부서에서는 경쟁심을 유발하기 위해서 다른 사원의 커미션을 조회할 수 있도록 해야 합니다.

04. 뷰의 종류



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

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 사용 불가능

〈실습하기〉 단순 뷰에 대한 데이터 조작

단순 뷰에 대해서 DML 즉, INSERT/UPDATE/DELETE 문을 사용할 수 있음을 확인합시다.

1. EMP_VIEW30 뷰에 데이터를 추가해 봅시다.

```
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 적용됩니다.

```
SELECT * FROM EMP_COPY;
```

〈실습하기〉 단순 뷰의 칼럼에 별칭 부여하기

기본 테이블(EMP_COPY)의 컬럼 명을 상속받지 않고 한글화 하여 컬럼 명이
사원번호, 사원명, 급여, 부서번호로 구성되도록 합시다.

1. EMP_VIEW30 뷰에 데이터를 추가해 봅시다.

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

2. EMP_VIEW 는 전체 사원에 대해서 특정 컬럼만 보여주도록 작성하였습니다.
. 다음과 같이 EMP_VIEW 를 SELECT 하면서 WHERE 절을 추가하여 30번
부서 소속 사원들의 정보만 볼 수 있습니다.

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

EMP_VIEW 뷰는 칼럼에 별칭을 주게 되면 기본 테이블의 칼럼 명을 더 이상
상속받지 못하므로 30번 부서를 검색하기 위해서는 뷰를 생성할 때 준 칼럼
별칭(부서번호)을 사용해야 합니다.

〈실습하기〉 그룹 함수를 사용한 단순 뷰

그룹 함수 SUM과 AVG를 사용해서 각 부서별 급여 총액과 평균을 구하는 뷰를 작성해 봅시다. 뷰를 작성하기 위해서 SELECT 절 다음에 SUM이란 그룹 함수를 사용하면 결과를 뷰의 특정 컬럼처럼 사용하는 것입니다. 따라서 물리적인 칼럼이 존재하지 않는 가상 칼럼이기에 칼럼 명도 상속 받을 수 없습니다. 뷰를 생성할 때 가상 칼럼을 사용하려면 사용자가 반드시 이름을 따로 설정해야 한다는 것을 명심하기 바랍니다.

1. 부서별 급여 총액과 평균을 구하기 위한 뷰를 생성해보도록 합시다.

```
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 뷰에 대해서 DML 명령어를 사용하여 조작이 가능하다고 하였습니다.
- ❖ 하지만, 다음과 같은 몇 가지의 경우에는 조작이 불가능합니다.

- ① 뷰 정의에 포함되지 않은 컬럼 중에 기본 테이블의 컬럼이 NOT NULL 제약 조건이 지정되어 있는 경우 INSERT 문이 사용 불가능합니다. 왜냐하면 뷰에 대한 INSERT 문은 기본 테이블에 NULL 값을 입력하는 형태가 되기 때문입니다.
- ② SAL*12와 같이 산술 표현식으로 정의된 가상 컬럼이 뷰에 정의되면 INSERT나 UPDATE가 불가능합니다.
- ③ DISTINCT을 포함한 경우에도 DML 명령을 사용할 수 없습니다.
- ④ 그룹 함수나 GROUP BY 절을 포함한 경우에도 DML 명령을 사용할 수 없습니다.

4.2 복합 뷰



- ❖ 뷰를 사용하는 이유 중의 하나가 복잡하고 자주 사용하는 질의를 보다 쉽고 간단하게 사용하기 위해서라고 했습니다. 이를 살펴보기 위해서 사원 테이블과 부서 테이블을 자주 조인한다고 합시다.
- ❖ 사원 테이블과 부서 테이블을 조인하기 위해서는 다음과 같이 복잡한 SELECT 문을 매번 작성해야 합니다.

```
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;
```

- ❖ 뷰를 사용하는 이유 중의 하나가 복잡하고 자주 사용하는 질의를 보다 쉽고 간단하게 사용하기 위해서라고 했습니다. 위에 작성한 조인문에 "CREATE VIEW EMP_VIEW_DEPT AS" 만 추가해서 뷰로 작성해 놓으면 "SELECT * FROM EMP_VIEW_DEPT;" 와 같이 간단하게 질의 결과를 얻을 수 있습니다.

〈실습하기〉 복합 뷰 만들기

사원 테이블과 부서 테이블을 조인하기 위해서 복합 뷰를 생성해 봅시다.

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

```
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 처리할 수 있습니다.

```
SELECT * FROM EMP_VIEW_DEPT;
```

05. 뷰 삭제와 다양한 옵션



- ❖ 뷰는 실체가 없는 가상 테이블이기 때문에 뷰를 삭제한다는 것은 **USER_VIEWS** 데이터 디렉터리에서 저장되어 있는 뷰의 정의를 삭제하는 것을 의미합니다.
- ❖ 따라서 뷰를 삭제해도 뷰를 정의한 기본 테이블의 구조나 데이터에는 전혀 영향을 주지 않습니다.

〈실습하기〉 뷰 삭제하기

지금까지 생성한 뷰 중에서 VIEW_SAL을 삭제합니다.

```
DROP VIEW VIEW_SAL;
```

06. 뷰 생성에 사용되는 다양한 옵션



- ❖ 뷰 정의하는 방법을 살펴보면서 뷰를 생성하기 위한 사용되는 옵션에 대해서 대략적으로 설명을 했습니다.

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

- ❖ 이번 절에서는 옵션에 대해서 예를 들어가면서 보다 자세히 살펴보도록 합시다.

6.1 뷰 수정을 위한 OR REPLACE 옵션



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

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

6.2 기본 테이블 없이 뷰를 생성하기 위한 FORCE 옵션



- ❖ 뷰를 생성하는 경우에 일반적으로 기본 테이블이 존재한다는 가정 하에서 쿼리문을 작성합니다.
- ❖ 극히 드물기는 하지만, 기본 테이블이 존재하지 않는 경우에도 뷰를 생성해야 할 경우가 있습니다. 이럴 경우에 사용하는 것이 FORCE 옵션입니다.
- ❖ FORCE 옵션과 반대로 동작하는 것으로서 NOFORCE 옵션이 있습니다.
- ❖ NOFORCE 옵션은 반드시 기본 테이블이 존재해야 할 경우에만 뷰가 생성됩니다.
- ❖ 지금까지 뷰를 생성하면서 FORCE/NOFORCE 옵션을 지정하지 않았습니다. 이렇게 특별한 설정이 없으면 디폴트로 NOFORCE 옵션이 지정된 것이므로 간주합니다.

〈실습하기〉 FORCE 옵션으로 기본 테이블 없이 뷰 생성하기

FORCE/NOFORCE 옵션이 어떤 역할을 하는지 살펴보기 위해서 존재하지 않는 테이블인 EMPLOYEES를 사용하여 뷰를 생성하도록 해봅시다.

1. 존재하지 않는 EMPLOYEES를 기본 테이블로 하여 뷰를 생성하게 되면 다음과 같이 오류가 발생합니다.

```
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 합니다.

〈실습하기〉 FORCE 옵션으로 기본 테이블 없이 뷰 생성하기

2. 기본 테이블이 존재하지 않는 경우에도 뷰를 생성하기 위해서 FORCE 옵션이 적용해 보시다.

```
CREATE OR REPLACE FORCE VIEW NOTABLE_VIEW  
AS  
SELECT EMPNO, ENAME, DEPTNO  
FROM EMPLOYEES  
WHERE DEPTNO=30;
```

존재하지 않는 테이블을 기본 테이블로 지정했다는 경고 메시지는 출력되지만, USER_VIEWS의 내용을 살펴보면 뷰가 생성된 것을 확인할 수 있습니다.

〈실습하기〉 FORCE 옵션으로 기본 테이블 없이 뷰 생성하기

3. FORCE의 반대 기능을 가진 옵션은 NOFORCE입니다.

```
CREATE OR REPLACE NOFORCE EXISTTABLE_VIEW  
AS  
SELECT EMPNO, ENAME, DEPTNO  
FROM EMPLOYEES  
WHERE DEPTNO=30;
```

NOFORCE 옵션을 사용하면 뷰를 생성할 때 존재하지 않는 테이블을 기본 테이블로 지정하면 오류 메시지와 함께 뷰가 생성되지 않습니다.

6.3 조건 컬럼 값 변경 못하게 하는 WITH CHECK OPTION



- ❖ 뷰를 정의하는 서브 쿼리문에 WHERE 절을 추가하여 기본 테이블 중 특정 조건에 만족하는 로우(행)만으로 구성된 뷰를 생성할 수 있습니다.
- ❖ 다음은 30 번 부서 소속 직원들의 정보만으로 구성된 뷰입니다.

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

6.3 조건 컬럼 값 변경 못하게 하는 WITH CHECK OPTION



- ❖ 뷰를 마치 테이블처럼 SELECT문으로 조회할 수 있음은 물론이고 DML 문으로 내용을 조작할 수 있음을 이미 학습했으므로 UPDATE 문으로 30번 부서에 소속된 직원 중에 급여가 1200 이상인 직원은 20번 부서로 이동시켜 봅시다.

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

- ❖ EMP_VIEW30 뷰를 여러 사람들이 공유해서 사용하는데 뷰의 부서번호를 변경하지 않은 사용자가 EMP_VIEW30 뷰를 사용하면서 무척이나 혼돈스러울 것입니다.

6.3 조건 컬럼 값 변경 못하게 하는 WITH CHECK OPTION



- ❖ 이러한 결과는 뷰를 공유해서 사용할 경우 혼돈을 초래할 수 있으므로 미연에 방지해야 합니다. 다행히 오라클에서는 WITH CHECK OPTION 으로 이러한 혼돈을 막을 수 있도록 합니다

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

〈실습하기〉 WITH CHECK OPTION 옵션으로 뷰 생성하기

WITH CHECK OPTION을 기술하면 뷰를 정의할 때 조건에 사용되어진 칼럼 값을 뷰를 통해서는 변경하지 못하도록 하여 혼동을 초래할 만한 일이 생기지 않게 해 봅시다.

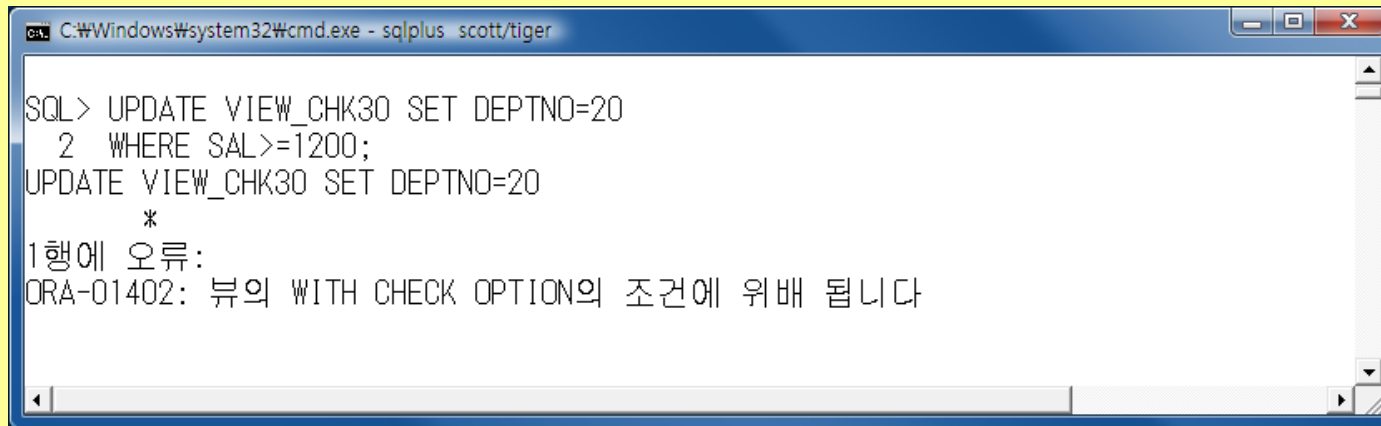
1. 다음과 같이 테이블을 생성합니다.

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

〈실습하기〉 WITH CHECK OPTION 옵션으로 뷰 생성하기

2. 급여가 5000이상인 사원은 20번 부서로 이동시켜봅시다.

```
UPDATE VIEW_CHK30 SET DEPTNO=20  
WHERE SAL>=1200;
```



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

```
SQL> UPDATE VIEW_CHK30 SET DEPTNO=20  
2 WHERE SAL>=1200;  
UPDATE VIEW_CHK30 SET DEPTNO=20  
*
```

The output shows an error message:

```
1행에 오류:  
ORA-01402: 뷰의 WITH CHECK OPTION의 조건에 위배 됩니다
```

VIEW_CHK30 뷰를 생성할 때 부서번호에 WITH CHECK OPTION을 지정하였기에 이 뷰를 통해서만 부서번호를 변경할 수 없습니다.

6.4 뷰를 통해 기본 테이블 변경 막는 WITH READ ONLY 옵션

- ❖ **WITH READ ONLY** 옵션은 뷰를 통해서는 기본 테이블의 어떤 컬럼에 대해서도 내용을 절대 변경할 수 없도록 하는 것입니다.



<실습하기> WITH CHECK OPTION과 WITH READ ONLY 비교하기

1. WITH CHECK OPTION을 기술한 VIEW_CHK30 뷰의 커미션을 모두 1000으로 변경해보도록 합시다.

```
UPDATE VIEW_CHK30 SET COMM=1000;
```

WITH CHECK OPTION은 뷰를 설정할 때 조건으로 설정한 컬럼이 아닌 컬럼에 대해서는 변경 가능하므로 커미션이 성공적으로 변경됩니다.

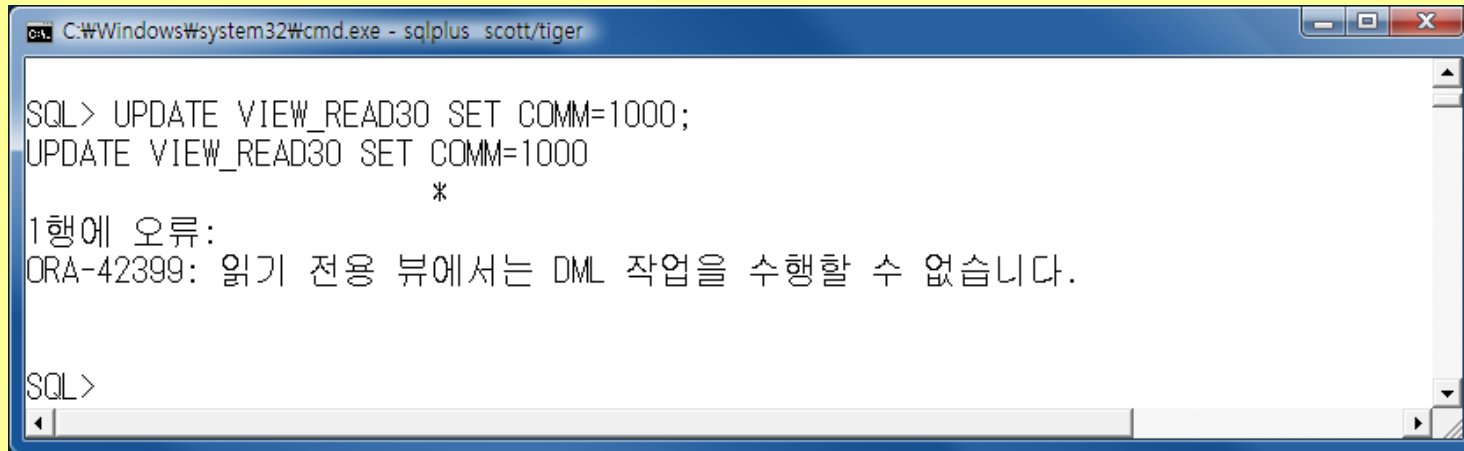
2. WITH READ ONLY 옵션을 지정한 뷰를 정의합니다.

```
CREATE OR REPLACE VIEW VIEW_READ30  
AS  
SELECT EMPNO, ENAME, SAL, COMM, DEPTNO  
FROM EMP_COPY  
WHERE DEPTNO=30 WITH READ ONLY;
```


<실습하기> WITH CHECK OPTION과 WITH READ ONLY 비교하기

3. WITH READ ONLY 옵션을 기술한 VIEW_READ30 뷰의 커미션을 모두 2000으로 변경해보도록 합시다.

```
UPDATE VIEW_READ30 SET COMM=2000;
```



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe - sqlplus scott/tiger". The command prompt displays the following text:

```
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>
```

WITH READ ONLY은 뷰를 설정할 때 조건으로 설정한 컬럼이 아닌 컬럼에 대해서도 변경 불가능하므로 커미션이 컬럼 값 역시 변경에 실패합니다.
뷰를 통해서 기본 테이블을 절대 변경할 수 없게 됩니다.

07. 인라인 뷰



- ❖ **사원 중에서 입사일이 빠른 사람 5명(TOP-5)만을 얻어 오는 질의문을 작성해 봅시다.**
- ❖ **TOP-N을 구하기 위해서는 ROWNUM과 인라인 뷰가 사용됩니다. 인라인 뷰는 조금 후에 다루어 보도록 하고, 우선 ROWNUM 칼럼에 대해서 살펴보도록 합시다.**
- ❖ **ROWNUM 칼럼은 DDL을 학습하면서 살펴보았지만 보다 자세히 살펴보도록 합시다.**

〈실습하기〉 ROWNUM 컬럼 성격 파악하기

1. 다음은 ROWNUM 컬럼 값을 출력하기 위한 쿼리문입니다.

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

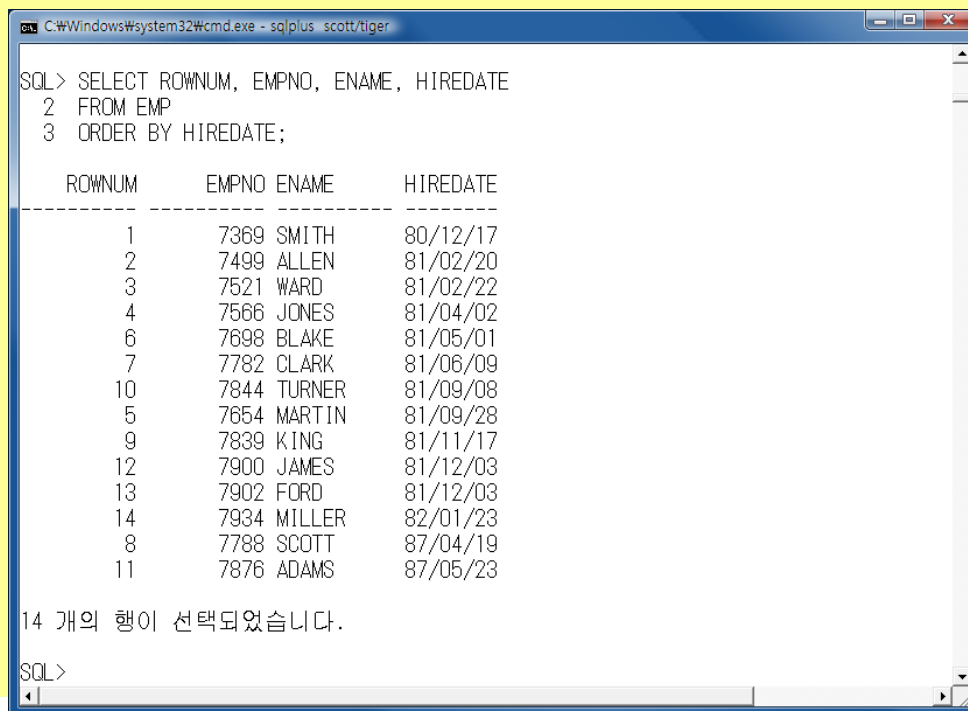
2. 입사일이 빠른 사람 5명만(TOP-N)을 얻어오기 위해서는 일련의 출력 데이터를 일단 임의의 순서로 정렬한 후에 그 중 일부의 데이터만 출력할 수 있도록 해야 하므로 ORDER BY 절을 사용하여 입사일을 기준으로 오름차순 정렬해 봅시다.

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

〈실습하기〉 ROWNUM 컬럼 성격 파악하기

3. 이번에는 입사일을 기준으로 오름차순 정렬을 하는 쿼리문에 ROWNUM 컬럼을 출력해 봅시다.

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



The screenshot shows a SQL*Plus 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 query results are displayed in a table with four columns: ROWNUM, EMPNO, ENAME, and HIREDATE. The results are sorted by HIREDATE in ascending order. There are 14 rows of data.

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, a message states: "14 개의 행이 선택되었습니다." (14 rows selected).

The prompt "SQL>" is visible at the bottom of the window.

〈실습하기〉 ROWNUM 컬럼 성격 파악하기

위 결과를 보면 입사일을 기준으로 오름차순 정렬을 하였기에 출력되는 행의 순서는 바뀌더라도 해당 행의 ROWNUM 컬럼 값은 바뀌지 않는다는 것을 알 수 있습니다.

ROWNUM 컬럼은 오라클의 내부적으로 부여되는데 INSERT 문을 이용하여 입력하면 입력한 순서에 따라 1씩 증가되면서 값이 지정되어 바뀌지 않습니다.

정렬된 순서대로 ROWNUM 컬럼 값이 매겨지도록 하려면 새로운 테이블이나 뷰로 새롭게 데이터를 저장해야만 합니다.

〈실습하기〉 ROWNUM 컬럼 성격 파악하기

위 결과를 보면 입사일을 기준으로 오름차순 정렬을 하였기에 출력되는 행의 순서는 바뀌더라도 해당 행의 ROWNUM 컬럼 값은 바뀌지 않는다는 것을 알 수 있습니다.

ROWNUM 컬럼은 오라클의 내부적으로 부여되는데 INSERT 문을 이용하여 입력하면 입력한 순서에 따라 1씩 증가되면서 값이 지정되어 바뀌지 않습니다.

정렬된 순서대로 ROWNUM 컬럼 값이 매겨지도록 하려면 새로운 테이블이나 뷰로 새롭게 데이터를 저장해야만 합니다.

〈실습하기〉 뷰와 ROWNUM 칼럼으로 TOP-N 구하기

ROWNUM 칼럼의 성격은 파악했으므로 이제 뷰와 함께 사용하여 TOP-N을 구해봅시다. TOP-N은 일련의 출력 데이터를 일단 임의의 순서로 정렬한 후에 그 중 일부의 데이터만 출력할 수 있도록 하여 구합니다.

1. 입사일을 기준으로 오름차순 정렬한 쿼리문으로 새로운 뷰를 생성해 봅시다

```
CREATE OR REPLACE VIEW VIEW_HIRE
AS
SELECT EMPNO, ENAME, HIREDATE
FROM EMP
ORDER BY HIREDATE;
```

<실습하기> 뷰와 ROWNUM 칼럼으로 TON-N 구하기

2. 입사일을 기준으로 오름차순 정렬을 하는 뷰에 ROWNUM 칼럼을 함께 출력해 봅시다.

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

```
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

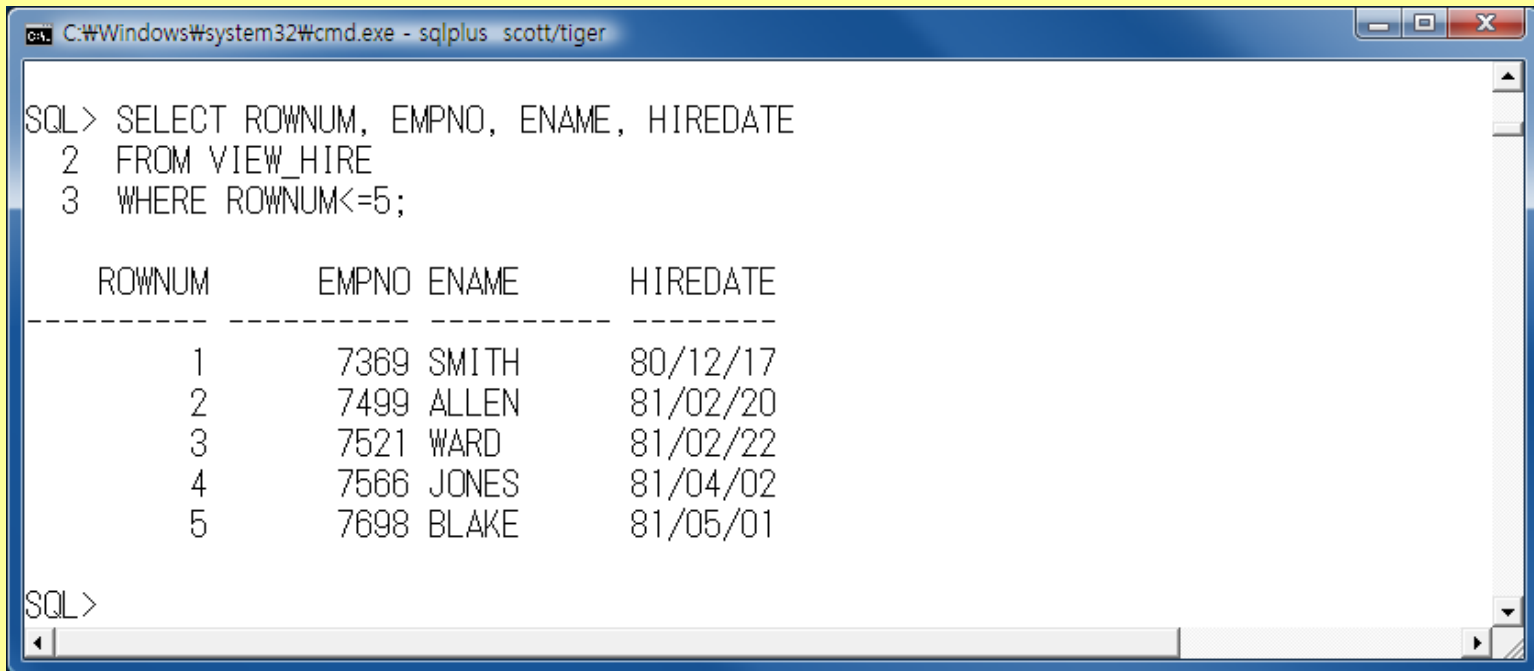
14 개의 행이 선택되었습니다.

```
SQL>
```


〈실습하기〉 뷰와 ROWNUM 칼럼으로 TON-N 구하기

3. 자, 이제 입사일이 빠른 사람 5명만을 얻어와 봅시다.

```
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 with columns: ROWNUM, EMPNO, ENAME, and HIREDATE. The results show the first 5 employees by hire date:

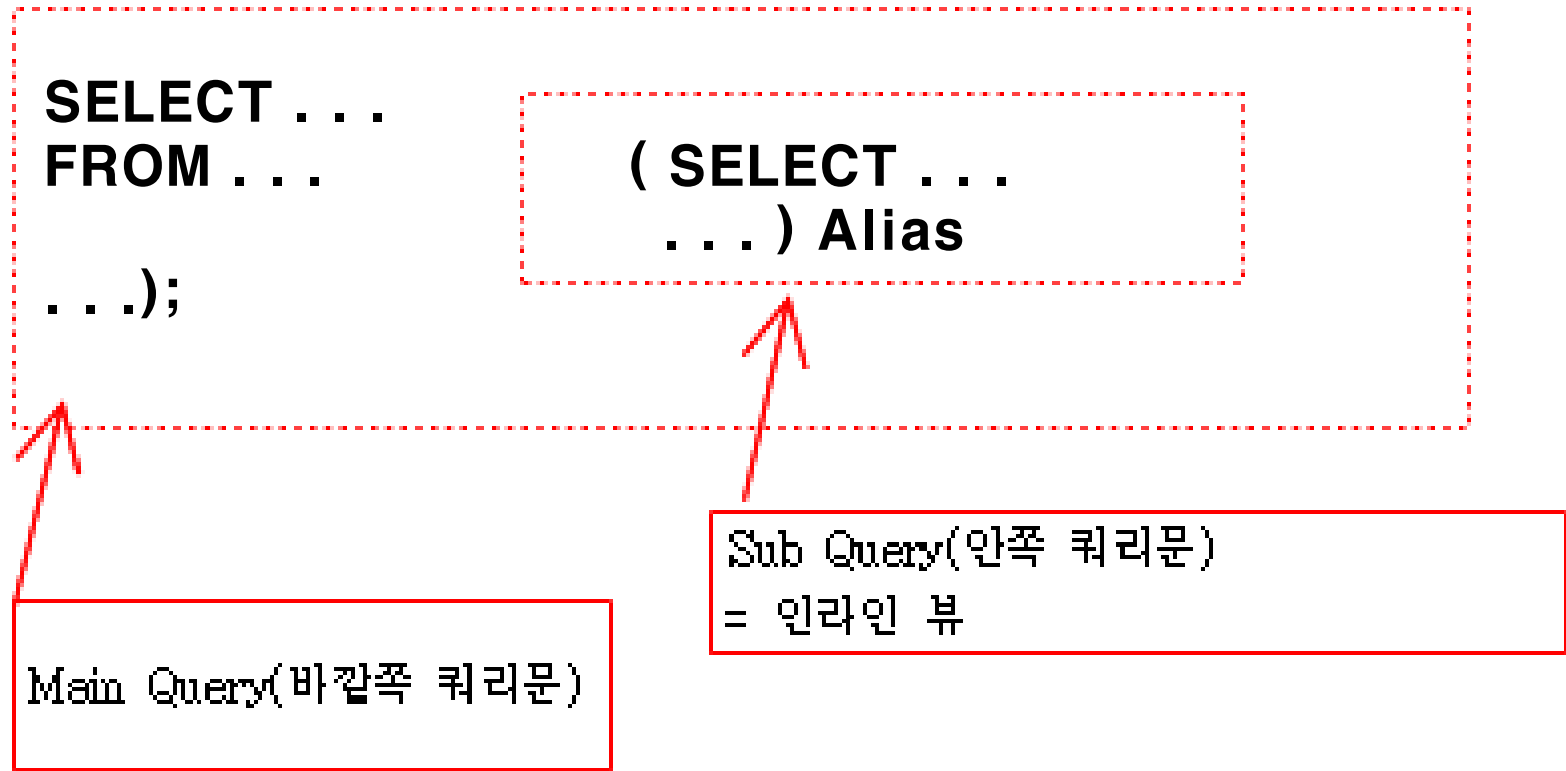
ROWNUM	EMPNO	ENAME	HIREDATE
1	7369	SMITH	80/12/17
2	7499	ALLEN	81/02/20
3	7521	WARD	81/02/22
4	7566	JONES	81/04/02
5	7698	BLAKE	81/05/01

The prompt "SQL>" is visible at the bottom left of the window.

6.2 인라인 뷰로 TOP-N 구하기



- ❖ 인라인 뷰는 SQL 문장에서 사용하는 서브 쿼리의 일종으로 보통 FROM 절에 위치해서 테이블처럼 사용하는 것입니다. 형식은 다음과 같습니다.



6.2 인라인 뷰로 TOP-N 구하기



- ❖ 인라인 뷰란 메인 쿼리의 SELECT 문의 FROM 절 내부에 사용된 서브 쿼리문을 말합니다.
- ❖ 우리가 지금까지 생성한 뷰는 CREATE 명령어로 뷰를 생성했지만, 인라인 뷰는 SQL 문 내부에 뷰를 정의하고 이를 테이블처럼 사용합니다.

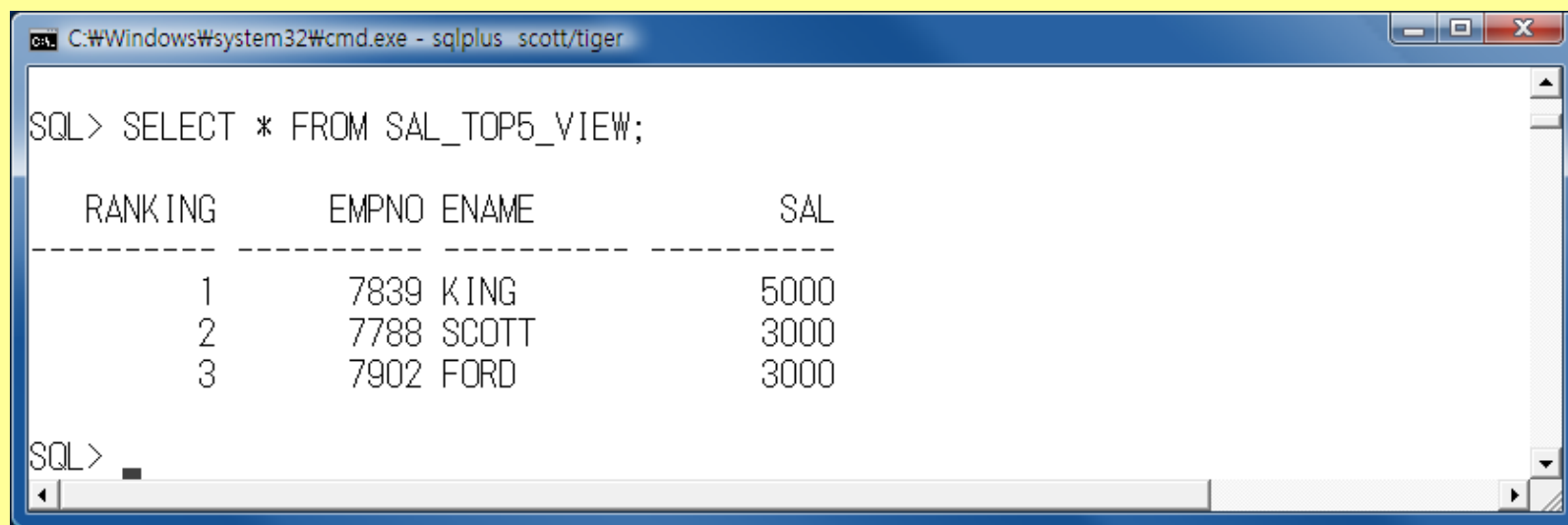
〈실습하기〉 인라인 뷰로 TOP-N 구하기

인라인 뷰를 사용해서 입사일이 빠른 사람 5명만을 얻어오기로 합시다. 아래 문장을 보면 FROM 절 다음인 VIEW_HIRE 위치에 VIEW_HIRE를 정의할 때 사용한 서브 쿼리문을 기술한 것뿐입니다.

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

〈실습하기〉 TOP-N 구하기

4. 인라인 뷰를 사용하여 급여를 많이 받는 순서대로 3명만 출력하는 뷰 (SAL_TOP5_VIEW)를 작성하시오.



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe - sqlplus scott/tiger". The prompt is "SQL>". The user has entered the query "SELECT * FROM SAL_TOP5_VIEW;". The output is a table with four columns: RANKING, EMPNO, ENAME, and SAL. The data is as follows:

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

The prompt "SQL>" is visible at the bottom left of the window.