

## 11. 가상 테이블인 뷰

# 01. 뷰의 개념

- ❖ 뷰(View)는 한마디로 물리적인 테이블을 근거한 논리적인 가상 테이블이라고 정의할 수 있습니다.
- ❖ 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 쿼리문입니다.
- ❖ 뷰(View)란 '보다'란 의미를 갖고 있는 점을 감안해 보면 알 수 있듯이 실제 테이블에 저장된 데이터를 뷰를 통해서 볼 수 있도록 합니다.
- ❖ 사용자에게 주어진 뷰를 통해서 기본 테이블을 제한적으로 사용하게 됩니다.

## 1.2 뷰 정의하기

- ❖ 뷰를 생성하여 자주 사용되는 SELECT 문을 간단하게 접근하는 방법을 학습해 봅시다. 다음은 뷰를 생성하기 위한 기본 형식입니다.

```
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 시작합니다. AS 다음은 마치 서브 쿼리문과 유사합니다.
- ❖ *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 됩니다.
- 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 있습니다.
- 그래서 대부분 뷰를 만들 때는 CREATE VIEW 대신 CREATE OR REPLACE VIEW를 사용하는 편입니다.

### ❖ FORCE

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

## 1.2 뷰 정의하기

### ❖ WITH CHECK OPTION

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

### ❖ WITH READ ONLY

- WITH READ ONLY를 사용하면 해당 뷰를 통해서는 SELECT만 가능하며 INSERT/UPDATE/DELETE를 할 수 없게 됩니다.
- 만약 이것을 생각한다면, 뷰를 사용하여 추가, 수정, 삭제(INSERT/UPDATE/DELETE)가 모두 가능합니다.

## 1.2 뷰 정의하기

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

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

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

## 1.2 뷰 정의하기

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

```
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 있습니다.

```
SELECT * FROM EMP_VIEW30;
```

## 02. 뷰의 내부구조와 USER\_VIEWS 데이터 딕셔너리

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

```
SELECT * FROM EMP_VIEW30;
```

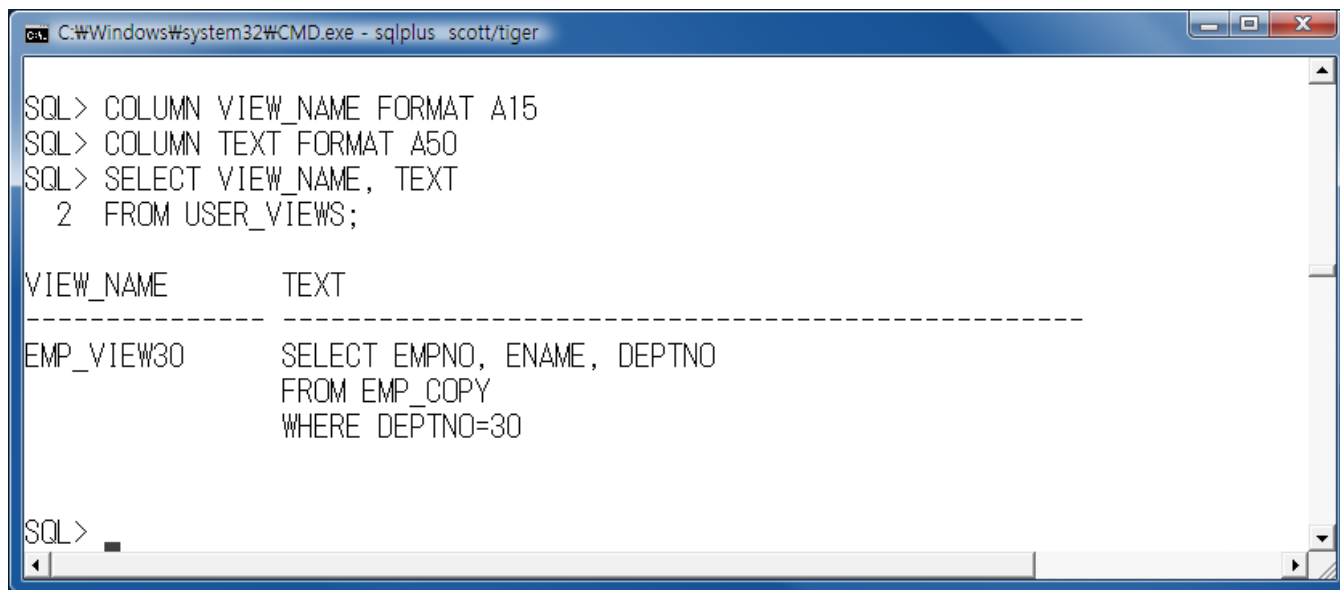
- ❖ EMP\_VIEW30라는 뷰는 데이터를 물리적으로 저장하고 있지 않습니다.
- ❖ CREATE VIEW 명령어로 뷰를 정의할 때 AS 절 다음에 기술한 쿼리 문장 자체를 저장하고 있습니다.
- ❖ 뷰 정의할 때 기술한 쿼리문이 궁금하다면 데이터 딕셔너리 USER\_VIEWS 테이블의 TEXT 컬럼 값을 살펴보면 됩니다.



## 02. 뷰의 내부구조와 USER\_VIEWS 데이터 덱서너리

- ❖ USER\_VIEWS에서 테이블 이름과 텍스트만 출력해 보겠습니다.

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



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> COLUMN VIEW_NAME FORMAT A15  
SQL> COLUMN TEXT FORMAT A50  
SQL> SELECT VIEW_NAME, TEXT  
2 FROM USER_VIEWS;
```

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

VIEW_NAME	TEXT
EMP_VIEW30	SELECT EMPNO, ENAME, DEPTNO FROM EMP_COPY WHERE DEPTNO=30

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

- 기본 테이블은 디스크 공간을 할당 받아서 실질적으로 데이터를 저장하고 있지만, 뷰는 데이터 덱서너리 USER\_VIEWS 에 사용자가 뷰를 정의할 때 기술한 서브 쿼리문(SELECT 문)만을 문자열 형태로 저장하고 있습니다.

## 02. 뷰의 내부구조와 USER\_VIEWS 데이터 덱서너리

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




1. 사용자가 뷰에 대해서 질의를 하면 USER\_VIEWS에서 뷰에 대한 정의를 조회합니다.
2. 기본 테이블에 대한 뷰의 접근 권한을 살핀다.
3. 뷰에 대한 질의를 기본 테이블에 대한 질의로 변환합니다.
4. 기본 테이블에 대한 질의를 통해 데이터를 검색합니다.
5. 검색된 결과를 출력합니다.

## 02. 뷰의 내부구조와 USER\_VIEWS 데이터 덱서너리

- ❖ 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 가져옵니다.

```
SELECT *  
FROM EMP_VIEW30;
```



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

- ❖ 질의는 기본 테이블인 EMP\_COPY를 통해 일어납니다. 즉, 기본 테이블인 EMP\_COPY에 대해서 서브 쿼리문을 수행하게 됩니다. 이러한 동작 원리 덕분에 뷰가 실질적으로 데이터를 저장하고 있지 않더라도 데이터를 검색할 수 있는 것입니다.

## 02. 뷰의 내부구조와 USER\_VIEWS 데이터 덱서너리`

- ❖ 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.

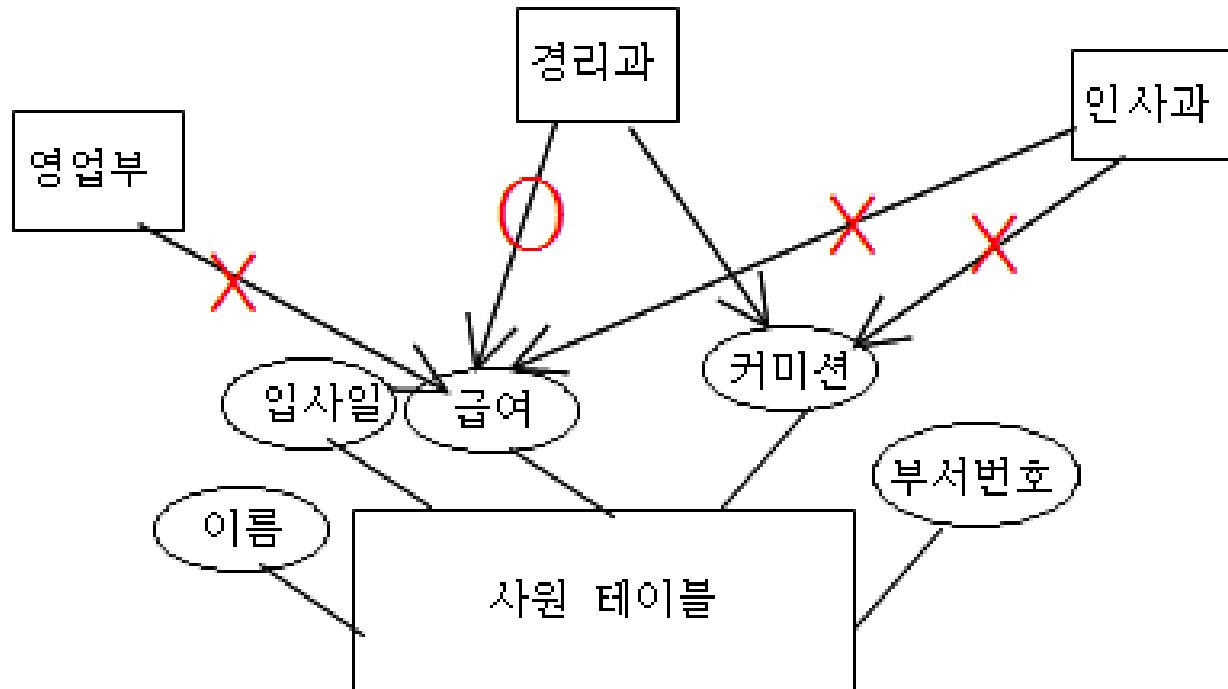
## 03. 뷰를 사용하는 이유

❖ 뷰를 사용하는 이유는 두 가지로 설명할 수 있습니다.

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

❖ 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

# 뷰 삭제하기

- ❖ 지금까지 생성한 뷰 중에서 VIEW\_SAL을 삭제합니다.

```
DROP VIEW VIEW_SAL;
```



## 05. 인라인 뷰

- ❖ 사원 중에서 입사일이 빠른 사람 5명(TOP-5)만을 얻어 오는 질의문을 작성해 봅시다.
- ❖ TOP-N을 구하기 위해서는 ROWNUM과 인라인 뷰가 사용됩니다.

# ROWNUM 컬럼 성격 파악하기

1. 다음은 ROWNUM 컬럼 값을 출력하기 위한 쿼리문입니다.

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

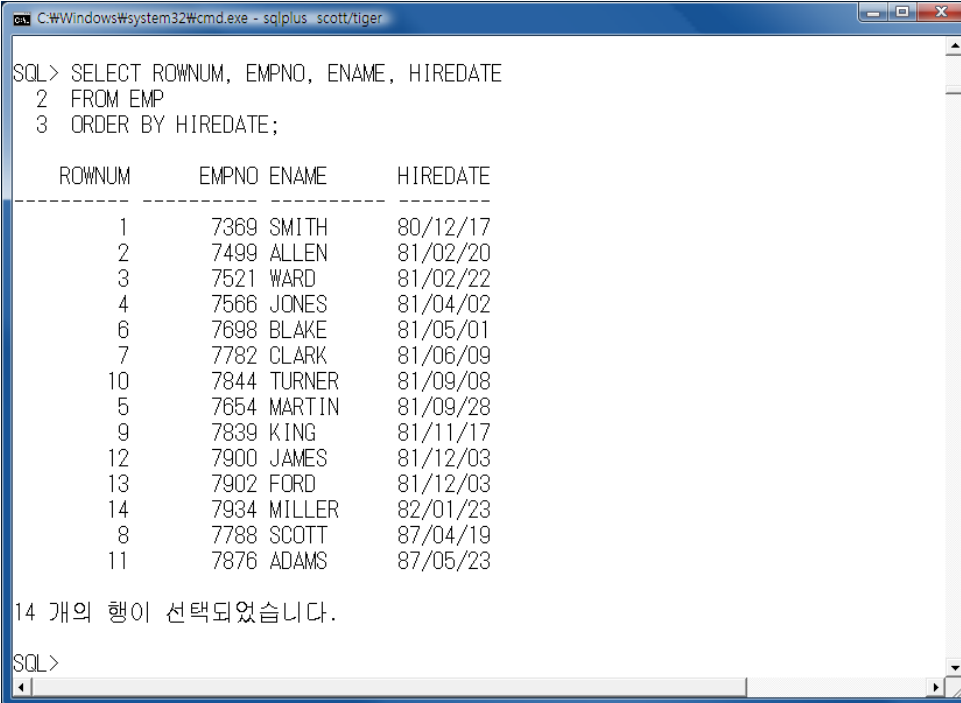
❖ 2. 입사일이 빠른 사람 5명만(TOP-N)을 얻어오기 위해서는 일련의 출력 데이터를 일단 임의의 순서로 정렬한 후에 그 중 일부의 데이터만 출력할 수 있도록 해야 하므로 ORDER BY 절을 사용하여 입사일을 기준으로 오름차순 정렬해 봅시다.

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

# 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 with the following content:

```
C:\Windows\system32\cmd.exe - sqlplus scott/tiger  
SQL> SELECT ROWNUM, EMPNO, ENAME, HIREDATE  
2  FROM EMP  
3  ORDER BY HIREDATE;  
  
   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  
  
14 개의 행이 선택되었습니다.  
SQL>
```

## <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

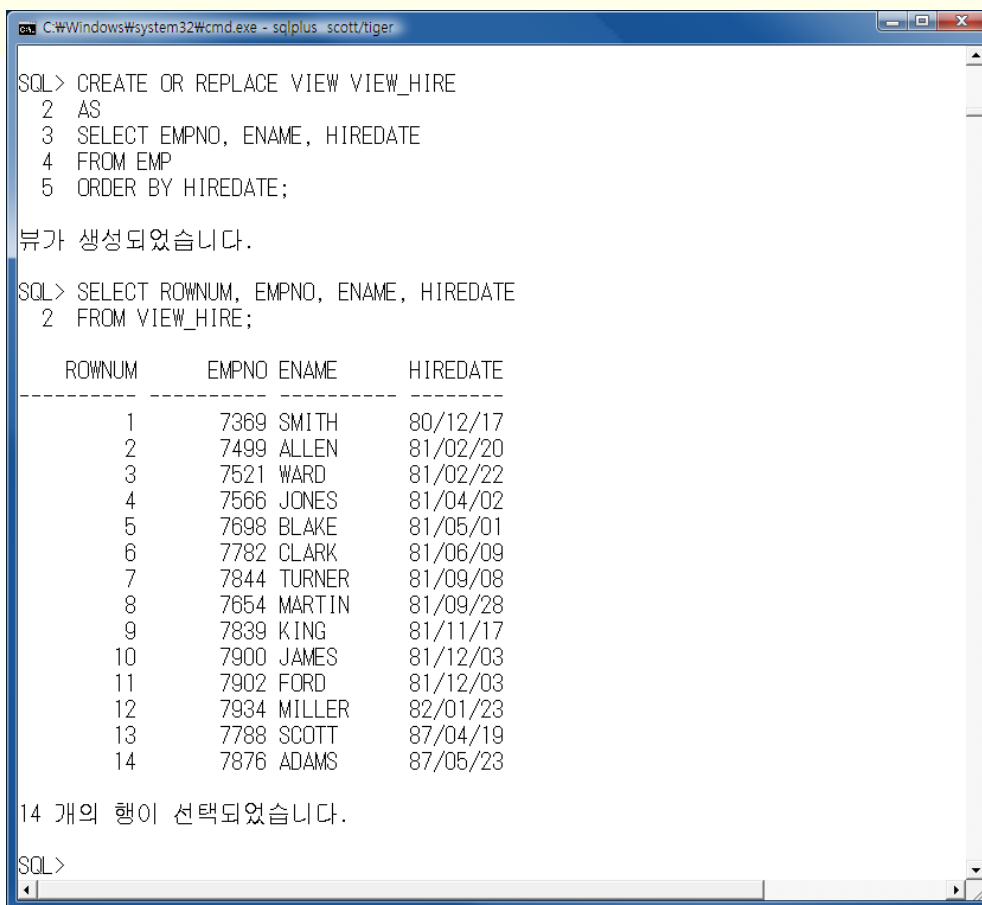
- ❖ 뷰와 함께 사용하여 TOP-N을 구해봅시다. TOP-N은 일련의 출력 데이터를 일단 임의의 순서로 정렬한 후에 그 중 일부의 데이터만 출력할 수 있도록 하여 구합니다.
- ❖ 1. 입사일을 기준으로 오름차순 정렬한 쿼리문으로 새로운 뷰를 생성해 봅시다.

```
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 보시다.

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



The screenshot shows a SQL\*Plus window with the following content:

```
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;
```

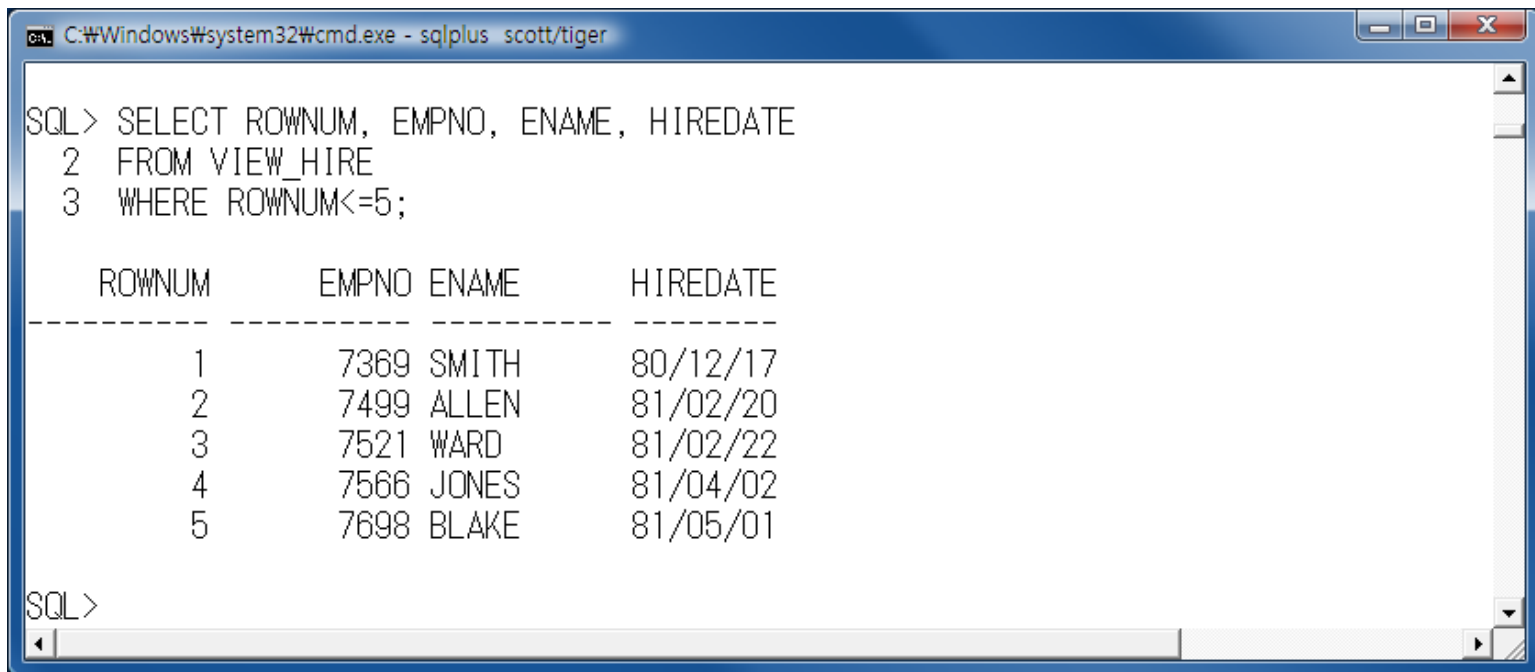
ROWNUM	EMPNO	ENAME	HIREDATE
1	7369	SMITH	80/12/17
2	7499	ALLEN	81/02/20
3	7521	WARD	81/02/22
4	7566	JONES	81/04/02
5	7698	BLAKE	81/05/01
6	7782	CLARK	81/06/09
7	7844	TURNER	81/09/08
8	7654	MARTIN	81/09/28
9	7839	KING	81/11/17
10	7900	JAMES	81/12/03
11	7902	FORD	81/12/03
12	7934	MILLER	82/01/23
13	7788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

14 개의 행이 선택되었습니다.  
SQL>

## <실습하기> 뷰와 ROWNUM 칼럼으로 TON-N 구하기

- ❖ 3. 이제 입사일이 빠른 사람 5명만을 얻어와 봅시다.

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



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

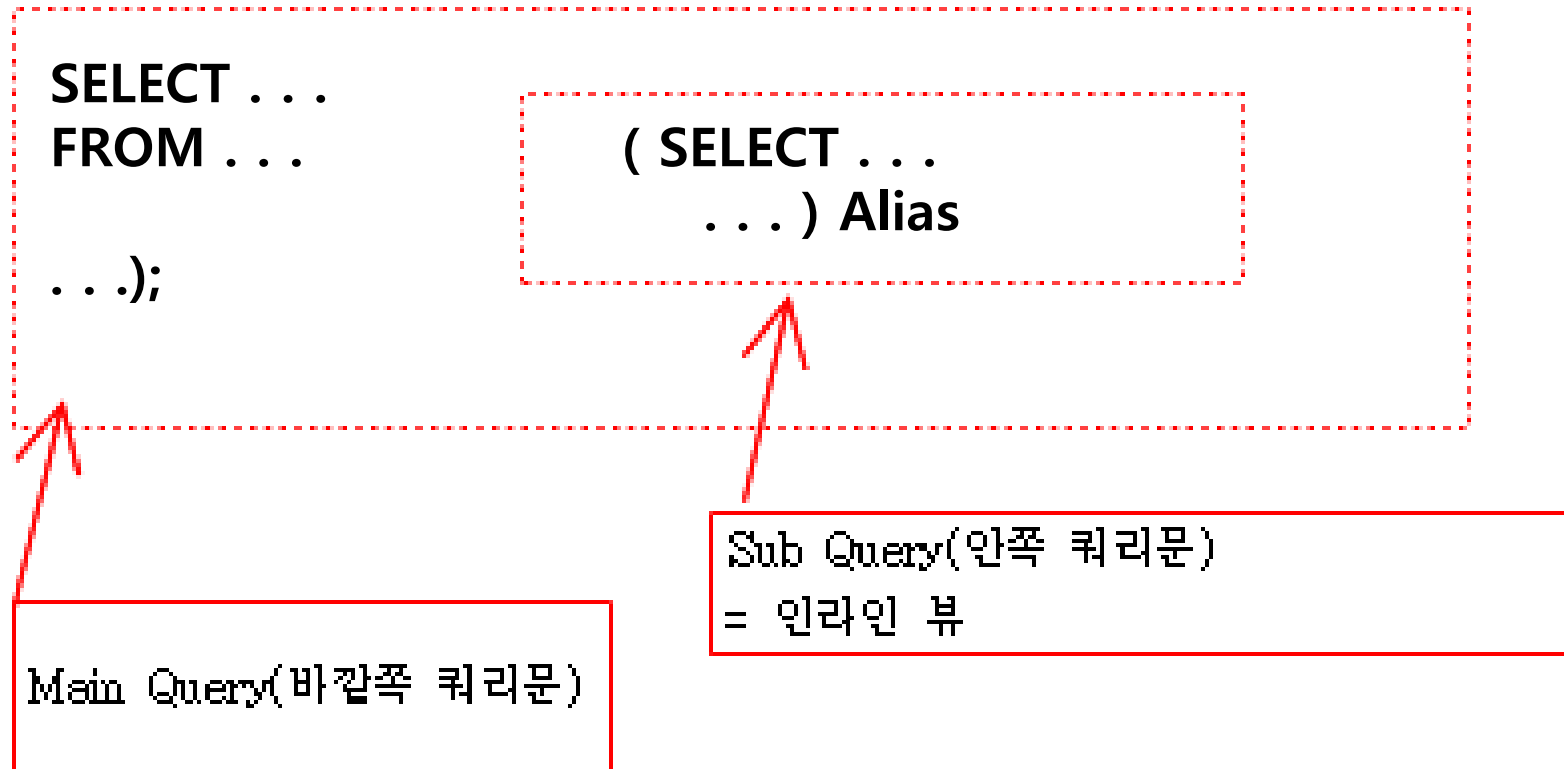
SQL> SELECT ROWNUM, EMPNO, ENAME, HIREDATE
2  FROM VIEW_HIRE
3  WHERE ROWNUM<=5;

   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

SQL>
```

## 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 합니다. 아래 문장을 보면 FROM 절 다음인 VIEW\_HIRE 위치에 VIEW\_HIRE를 정의할 때 사용한 서브 쿼리문을 기술한 것입니다.

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