ORACLE

13. 뷰

목차

1. 뷰의 개념

- 뷰(View)
  - 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 가상 테이블이라 불린다.
  - 기본 테이블에서 파생된 객체로서 기본 테이블에 대한 하나의 쿼리문이다.
  - 사용자에게 주어진 뷰를 통해서 기본 테이블을 제한적으로 사용하게 된다.

1.1 뷰의 기본 테이블(1)

- 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 사용하자.
- 테이블의 내용을 복사할 때 제약조건은 복사되지 않는다.

1.1 뷰의 기본 테이블(2)

- ex) 뷰의 기본 테이블을 생성한다.
- 1. DEPT_COPY를 DEPT 테이블의 복사본으로 생성한다.

예

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

- 2. EMP 테이블의 복사본으로 EMP_COPY를 생성한다.

예

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

1.2 뷰 정의하기(1)

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

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

- 테이블을 생성하기 위해서 CREATE TABLE 로 시작하지만, 뷰를 생성하기 위해서는 CREATE VIEW로 시작합니다. AS 다음은 마치 서브 쿼리문과 유사하다.
- *subquery*에는 우리가 지금까지 사용하였던 SELECT 문을 기술하면 된다.

1.2 뷰 정의하기(2)

- CREATE OR REPLACE VIEW
  - 뷰를 만들 때 CREATE OR REPLACE 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 사용하는 편이다.

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

1.2 뷰 정의하기(3)

- 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 통해서는 SELECT만 가능하며 INSERT/UPDATE/DELETE를 할 수 없게 된다.
  - 만약 이것을 생략한다면, 뷰를 사용하여 추가, 수정, 삭제 (INSERT/UPDATE/DELETE)가 모두 가능하다.

1.2 뷰 정의하기(4)

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

예

```
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 바람에서 출발한 개념이다.

1.2 뷰 정의하기(5)

- 자주 사용되는 30번 부서에 소속된 사원들의 사번과 이름과 부서번호를 출력하기 위한 SELECT문을 하나의 뷰로 정의해 보자.

예

```
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.

예

```
SELECT * FROM EMP_VIEW30;
```

1.2 뷰 정의하기(6)

- 뷰를 생성하려는데 권한이 불충분하다고 오류가 발생할 경우가 있다.
- 이럴 경우에는 DBA인 SYSTEM 계정으로 로그인하여 뷰를 생성할 권한을 부여해야 한다.
- 특정 사용자에 대해서 아무 문제없이 뷰가 생성된다면 괜찮지만, 그렇지 않을 경우 GRANT 명령어로 특정 사용자에게 권한을 부여해야 한다.
- 아래 문장은 사용자에게 테이블을 생성할 CREATE VIEW 권한을 부여하는 명령어이다.
- 이 명령어는 DBA 권한이 있는 사용자만이 부여할 수 있으므로 SYSTEM계정으로 접속한다.

예

```
CONN SYSTEM/PASSWORD

GRANT CREATE VIEW TO USER;
```

문제

1. 기본 테이블은 EMP_COPY로 하여 20번 부서에 소속된 사원들의 사번과 이름과 부서번호와 상관의 사번을 출력하기 위한 SELECT문을 EMP_VIEW20 이란 이름의 뷰로 정의하시오.

EMPNO	ENAME	DEPTNO	MGR
1001	김사랑	20	1013
1004	이병헌	20	1008
1009	안성기	20	(null)
1012	강혜정	20	1006
1013	박숭훈	20	1003

2. 뷰의 구조(1)

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

  예 **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.
- 뷰를 정의할 때 기술한 쿼리문이 궁금하다면 데이터 딕셔너리 USER_VIEWS 테이블의 TEXT 컬럼 값을 살펴보면 된다.

2. 뷰의 구조(2)

- ex) USER_VIEWS에서 테이블 이름과 텍스트만 출력해보자

예

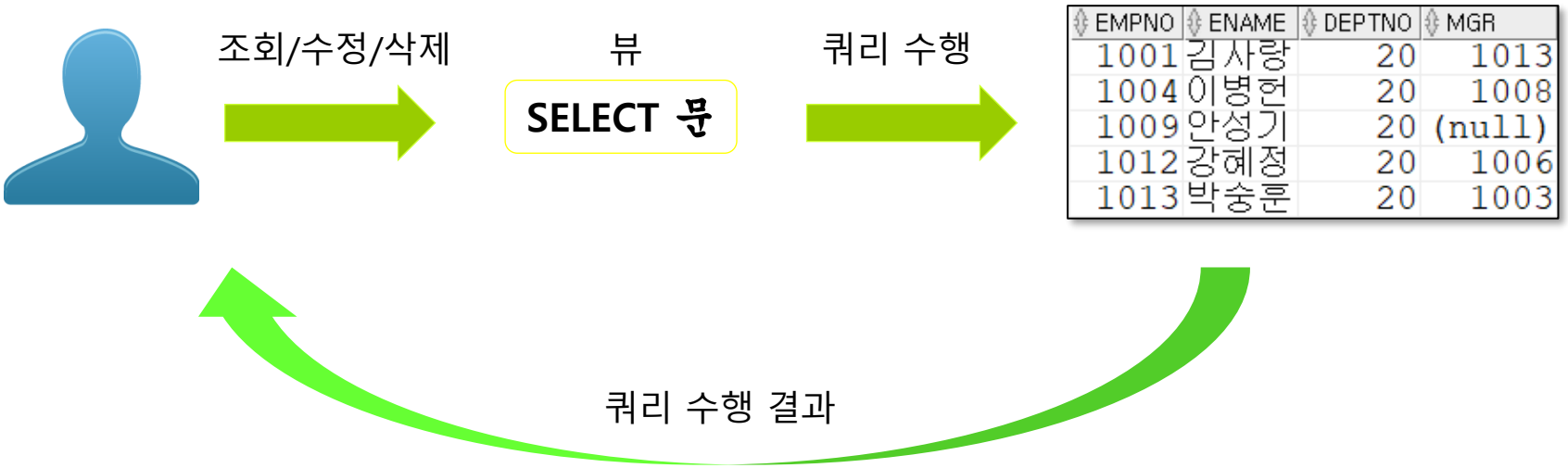
```
SELECT VIEW_NAME, TEXT
FROM USER_VIEWS;
```

VIEW_NAME	TEXT
EMP_VIEW30	SELECT EMPNO, ENAME, DEPTNOFROM EMP COPYWHERE DEPTNO=30

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

2. 뷰의 구조(3)

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

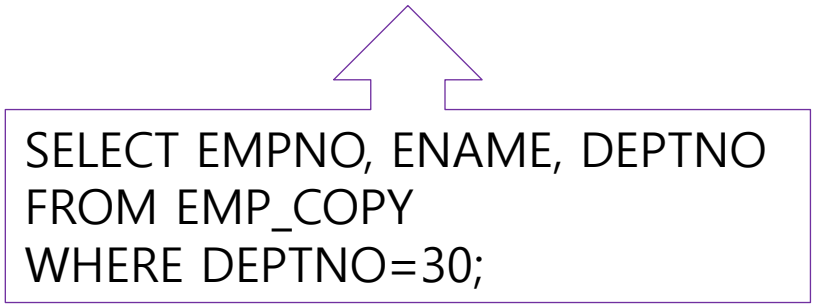


1. 사용자가 뷰에 대해서 질의를 하면 USER_VIEWS에서 뷰에 대한 정의를 조회한다.
2 기본 테이블에 대한 뷰의 접근 권한을 살핀다.
3. 뷰에 대한 질의를 기본 테이블에 대한 질의로 변환한다.
4. 기본 테이블에 대한 질의를 통해 데이터를 검색한다.
5. 검색된 결과를 출력한다.

2. 뷰의 구조(4)

- 우리가 앞에서 생성한 뷰인 EMP_VIEW30을 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;
```



- 질의는 기본 테이블인 EMP_COPY를 통해 일어난다. 즉, 기본 테이블인 EMP_COPY 에 대해서 서브 쿼리문을 수행하게 된다. 이러한 동작 원리 덕분에 뷰가 실질적으로 데이터를 저장하고 있지 않더라도 데이터를 검색할 수 있는 것이다.

2. 뷰의 구조(5)

- ex) 기본 테이블을 가져다가 쿼리문을 수행한다는 것을 증명하기 위해서 간단한 예를 살펴보자.
- 1. EMP_VIEW30 뷰에 행을 하나 추가하는 문장이다.

예

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

- 2. INSERT 문으로 뷰에 새로운 행을 추가하였다. 뷰의 내용을 출력해 보면 추가된 행이 뷰에 존재하고 있음을 확인할 수 있다.

EMPNO	ENAME	DEPTNO
1002	한예슬	30
1003	오지호	30
1005	신농협	30
1006	장농건	30
1008	감우성	30
1011	조향기	30
1111	AAAA	30

2. 뷰의 구조(5)

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

예 **SELECT * FROM EMP_COPY;**

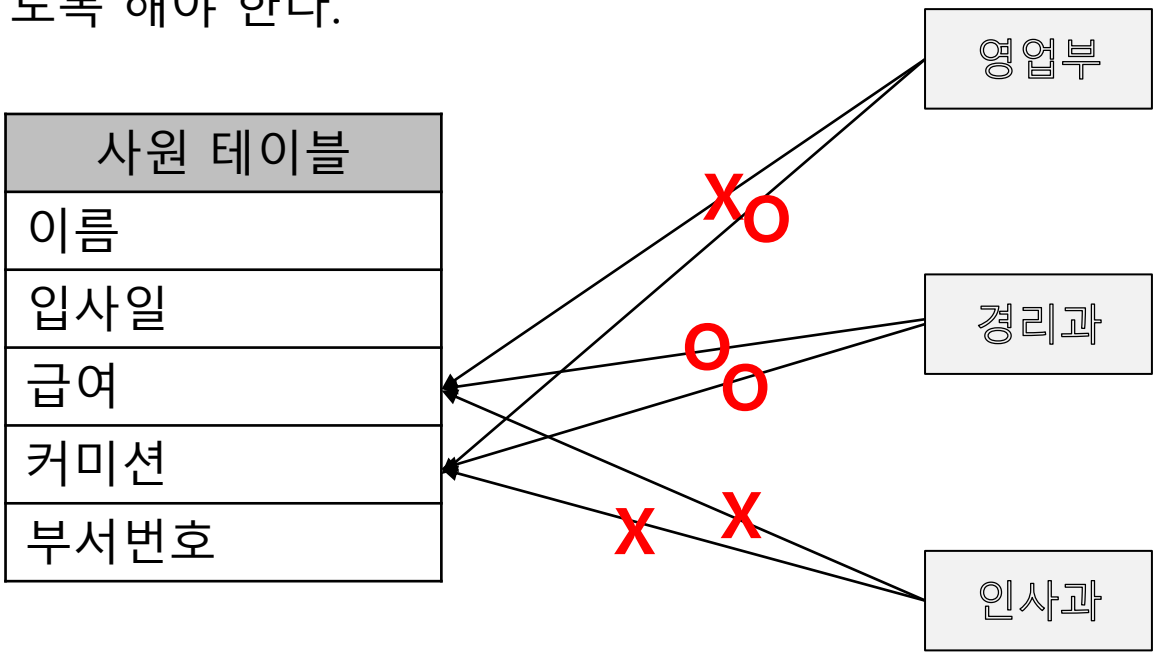
EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1001	김사랑	사원	1013	07/03/01	300	(null)	20
1002	한예슬	대리	1005	07/04/02	250	80	30
1003	오지호	과장	1005	05/02/10	500	100	30
1004	이병헌	부장	1008	03/09/02	600	(null)	20
1005	신농협	과장	1005	05/04/07	450	200	30
1006	장농건	부장	1008	03/10/09	480	(null)	30
1007	이눈세	부장	1008	04/01/08	520	(null)	10
1008	감우성	차장	1003	04/03/08	500	0	30
1009	안성기	사장	(null)	96/10/04	1000	(null)	20
1010	이병헌	과장	1003	05/04/07	500	(null)	10
1011	조향기	사원	1007	07/03/01	280	(null)	30
1012	강혜정	사원	1006	07/08/09	300	(null)	20
1013	박숭훈	부장	1003	02/10/09	560	(null)	20
1014	조인성	사원	1006	07/11/09	250	(null)	10
1111	AAAA	(null)	(null)	(null)	(null)	(null)	30

2. 뷰의 구조(6)

- 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 있다. 뷰(EMP_VIEW30)의 내용을 확인하기 위해 SELECT문을 수행하면 변경된 기본 테이블(EMP_COPY)의 내용에서 일부를 서브 쿼리한 결과를 보여준다.
- 뷰는 실질적인 데이터를 저장한 기본 테이블을 볼 수 있도록 한 투명한 창이다, 기본 테이블의 모양이 바뀐 것이고 그 바뀐 내용을 뷰라는 창을 통해서 볼 뿐이다. 뷰에 INSERT 뿐만 아니라 UPDATE, DELETE 모두 사용할 수 있는데, UPDATE, DELETE 쿼리문 역시 뷰의 텍스트에 저장되어 있는 기본 테이블이 변경하는 것이다.
- 그렇기에 뷰가 물리적인 테이블을 근거로 한 논리적인 가상 테이블이란 뜻이다.

3. 뷰를 사용하는 이유

- 예를 들어 사원 테이블에 개인 적인 정보인 급여와 커미션은 부서에 따라 접근을 제한해야 한다. 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 한다.



4. 뷰의 종류

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

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

4.1 단순 뷰에 대한 데이터 조작

- ex) 단순 뷰에 대해서 DML 즉,INSERT/UPDATE/DELETE 문을 사용할 수 있음을 확인한다.
- 1. EMP_VIEW30 뷰에 데이터를 추가해본다.

예

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

SELECT * FROM EMP_VIEW30;
```

- 2. 단순 뷰를 대상으로 실행한 DML 명령문의 치리 결과는 뷰를 정의할 때 사용한 기본 테이블에 적용된다.

예

```
SELECT * FROM EMP_COPY;
```

4.2 단순 뷰의 컬럼에 별칭 부여하기

- ex) 기본 테이블(EMP_COPY)의 컬럼 명을 상속받지 않고 한글화 하여 컬럼 명이 사원번호, 사원명, 급여, 부서번호로 구성되도록 한다.
- 1. EMP_VIEW 뷰를 생성한다.

예

```
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 작성하였다. 다음과 같이 EMP_VIEW 를 SELECT 하면서 WHERE 절을 추가하여 30번 부서 소속 사원들의 정보만 볼 수 있다.

예

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

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

4.3 그룹 함수를 사용한 단순 뷰

- 그룹 함수 SUM과 AVG를 사용해서 각 부서별 급여 총액과 평균을 구하는 뷰를 작성해보자. 뷰를 작성하기 위해서 SELECT 절 다음에 SUM이란 그룹 함수를 사용하면 결과를 뷰의 특정 컬럼처럼 사용하는 것이다. 따라서 물리적인 컬럼이 존재하지 않는 가상 컬럼이기에 컬럼 명도 상속 받을 수 없다. 뷰를 생성할 때 가상 컬럼을 사용하려면 사용자가 반드시 이름을 따로 설정해야 한다.
- 1. 부서별 급여 총액과 평균을 구하기 위한 뷰를 생성해보자.

예

```
CREATE VIEW VIEW_SAL
AS
SELECT DEPTNO, SUM(SAL) AS "SalSum", AVG(SAL) AS
"SalAvg"
FROM EMP_COPY
GROUP BY DEPTNO;
```

4.4 단순 뷰

- 단순 뷰에 대해서 DML 명령어를 사용하여 조작이 가능하다고 하였다.
- 하지만, 다음과 같은 몇 가지의 경우에는 조작이 불가능하다.

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

4.5 복합 뷰

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

예

```
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;
```

- 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 있다.

4.6 복합 뷰 만들기

- ex) 사원 테이블과 부서 테이블을 조인하기 위해서 복합 뷰를 생성해보자.
- 1. 다음은 사번, 이름, 급여, 부서번호, 부서명, 지역명을 출력하기 위한 복합 뷰이다.

예

```
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 있다.

예

```
SELECT * FROM EMP_VIEW_DEPT;
```

문제

2. 각 부서별 최대 급여와 최소 급여를 출력하는 뷰를 SAL_VIEW 란 이름으로 작성하시오.

DNAME	MAX_SAL	MIN_SAL
경리부	520	250
영업부	500	250
인사부	1000	300

5. 뷰 삭제와 옵션(1)

- 뷰는 실체가 없는 가상 테이블이기 때문에 뷰를 삭제한다는 것은 USER_VIEWS 데이터 딕셔너리에 저장되어 있는 뷰의 정의를 삭제하는 것을 의미한다.
- 따라서 뷰를 삭제해도 뷰를 정의한 기본 테이블의 구조나 데이터에는 전혀 영향을 주지 않는다.

- ex) VIEW_SAL을 삭제한다.

예

```
DROP VIEW VIEW_SAL;
```

5. 뷰 삭제와 옵션(2)

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

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

- 이번 절에서는 옵션에 대해서 예를 들어가면서 보다 자세히 살펴보도록 하자.

5.1 OR REPLACE

- CREATE OR REPLACE VIEW 를 사용하면 존재하지 않은 뷰이면 새로운 뷰를 생성하고 기존에 존재하는 뷰이면 그 내용을 변경한다.
- 이전에 작성한 EMP_VIEW30 뷰는 "EMPNO, ENAME, DEPTNO" 3 개의 컬럼을 출력하는 형태였는데 급여와 커미션 컬럼을 추가로 출력할 수 있도록 하기 위해서 뷰의 구조를 변경한다.

예

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

5.2 FORCE(1)

- 뷰를 생성하는 경우에 일반적으로 기본 테이블이 존재한다는 가정 하에서 쿼리문을 작성한다.
- 극히 드물기는 하지만, 기본 테이블이 존재하지 않는 경우에도 뷰를 생성해야 할 경우가 있다. 이럴 경우에 사용하는 것이 FORCE 옵션이다.
- FORCE 옵션과 반대로 동작하는 것으로서 NOFORCE 옵션이 있다.
- NOFORCE 옵션은 반드시 기본 테이블이 존재해야 할 경우에만 뷰가 생성된다.
- 지금까지 뷰를 생성하면서 FORCE/NOFORCE 옵션을 지정하지 않았다. 이렇게 특별한 설정이 없으면 디폴트로 NOFORCE 옵션이 지정된 것이므로 간주한다.

5.2 FORCE(2)

- ex) FORCE/NOFORCE 옵션이 어떤 역할을 하는지 살펴보기 위해서 존재하지 않는 테이블인 EMPLOYEES를 사용하여 뷰를 생성하도록 한다.
- 1. 존재하지 않는 EMPLOYEES를 기본 테이블로 하여 뷰를 생성하게 되면 오류가 발생한다.

예

```
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 한다.

5.2 FORCE(2)

- 2. 기본 테이블이 존재하기 않는 경우에도 뷰를 생성하기 위해서 FORCE 옵션이 적용한다.

예

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

```
경고: 컴파일 오류와 함께 뷰가 생성되었습니다.
```

- 경고 메시지는 출력되지만, USER_VIEWS의 내용을 살펴보면 뷰가 생성된 것을 확인할 수 있다.

5.3 WITH CHECK OPTION(1)

- 뷰를 정의하는 서브 쿼리문에 WHERE 절을 추가하여 기본 테이블 중 특정 조건에 만족하는 로우(행)만으로 구성된 뷰를 생성할 수 있다.
- WITH CHECK 옵션은 특정 조건의 컬럼값을 변경 못하게 하는 옵션이다.

- 다음은 30번 부서 소속 사원들의 정보만으로 구성된 뷰이다.

예

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

5.3 WITH CHECK OPTION(2)

- 뷰를 마치 테이블처럼 SELECT문으로 조회할 수 있음은 물론이고 DML 문으로 내용을 조작할 수 있음을 이미 학습했으므로 UPDATE 문으로 30번 부서에 소속된 사원 중에 급여가 400 이상인 사원은 20번 부서로 이동시켜 보자

예

```
UPDATE EMP_VIEW30 SET DEPT=20
WHERE SAL>=400;
```

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

5.3 WITH CHECK OPTION(3)

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

예

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

5.3 WITH CHECK OPTION(3)

- ex) 위에서 만든 VIEW_CHK30 뷰를 이용하여 급여가 400이상인 사원은 20번 부서로 이동시켜본다.

예

```
UPDATE VIEW_CHK30 SET DEPTNO=20
WHERE SAL>=400;
```

```
명령의 1 행에서 시작하는 중 오류 발생 -
UPDATE VIEW_CHK30 SET DEPTNO=20
WHERE SAL>=400
오류 보고 -
ORA-01402: view WITH CHECK OPTION where-clause violation
```

- VIEW_CHK30 뷰를 생성할 때 부서번호에 WITH CHECK OPTION을 지정하였기에 이 뷰를 통해서는 부서번호를 변경할 수 없다.
- 다른 컬럼은 변경 가능하다.

5.4 READ ONLY(1)

- WITH READ ONLY 옵션은 뷰를 통해서는 기본 테이블의 어떤 컬럼에 대해서도 내용을 절대 변경할 수 없도록 하는 것이다.
- ex) WITH READ ONLY 옵션을 지정한 뷰를 정의한다.

예

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

5.4 READ ONLY(2)

- 1. WITH READ ONLY 옵션을 기술한 VIEW_READ30 뷰의 커미션을 모두 2000으로 변경해보도록 한다.

예 **UPDATE VIEW_READ30 SET COMM=2000;**

```
명령의 1 행에서 시작하는 중 오류 발생 -
UPDATE VIEW_READ30 SET COMM=2000
오류 발생 명령행: 1 열: 24
오류 보고 -
SQL 오류: ORA-42399: cannot perform a DML operation on a read-only view
42399.0000 - "cannot perform a DML operation on a read-only view"
```

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

6. 인라인 뷰(1)

- *사원 중에서 입사일이 빠른 사람 5명(TOP-5)만을 얻어 오는 질의문을 작성 해본다.*

- TOP-N을 구하기 위해서는 ROWNUM과 인라인 뷰가 사용된다. 인라인 뷰는 조금 후에 다루어 보도록 하고, 우선 ROWNUM 컬럼에 대해서 살펴보도록하자.

- ROWNUM컬럼은 데이터가 입력된 순서(몇번째 입력된 데이터인지)를 알려주는 '가상컬럼'이다.

6. 인라인 뷰(2)

- 1. 다음은 ROWNUM 컬럼 값을 출력하기 위한 쿼리문이다.

예

```
SELECT ROWID, 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 정렬해보자.

예

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

6. 인라인 뷰(3)

❖ 3. 이번에는 입사일을 기준으로 오름차순 정렬을 하는 쿼리문에 ROWNUM 컬럼을 출력해보자.

예

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

ROWNUM	EMPNO	ENAME	HIREDATE
9	1009	안성기	96/10/04
13	1013	박중훈	02/10/09
4	1004	이병헌	03/09/02
6	1006	장동건	03/10/09
7	1007	이문세	04/01/08
8	1008	감우성	04/03/08
3	1003	오지호	05/02/10
10	1010	이병헌	05/04/07
5	1005	신농협	05/04/07
1	1001	김사랑	07/03/01
11	1011	조향기	07/03/01
2	1002	한예슬	07/04/02
12	1012	강혜정	07/08/09
14	1014	조인성	07/11/09

6. 인라인 뷰(4)

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

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

- 정렬된 순서대로 ROWNUM 칼럼 값이 매겨지도록 하려면 새로운 테이블이나 뷰로 새롭게 데이터를 저장해야만 한다.

6.1 뷰와 ROWNUM으로 TOP-N 구하기(1)

- ex) ROWNUM 컬럼의 성격은 파악했으므로 이제 뷰와 함께 사용하여 TOP-N을 구해보자. TOP-N은 일련의 출력 데이터를 일단 임의의 순서로 정렬한 후에 그 중 일부의 데이터만 출력할 수 있도록 하여 구한다.

- 1. 입사일을 기준으로 오름차순 정렬한 쿼리문으로 새로운 뷰를 생성한다.

예

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

6.1 뷰와 ROWNUM으로 TOP-N 구하기(2)

❖ 2. 입사일을 기준으로 오름차순 정렬을 하는 뷰에 ROWNUM 칼럼을 함께 출력해보자.

예

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

ROWNUM	EMPNO	ENAME	HIREDATE
1	1009	안성기	96/10/04
2	1013	박중훈	02/10/09
3	1004	이병헌	03/09/02
4	1006	장동건	03/10/09
5	1007	이문세	04/01/08
6	1008	감우성	04/03/08
7	1003	오지호	05/02/10
8	1010	이병헌	05/04/07
9	1005	신농협	05/04/07
10	1001	김사랑	07/03/01
11	1011	조향기	07/03/01
12	1002	한예슬	07/04/02
13	1012	강혜정	07/08/09
14	1014	조인성	07/11/09

6.1 뷰와 ROWNUM으로 TOP-N 구하기(3)

❖ 3. 이제 입사일이 빠른 사람 5명만을 얻어온다.

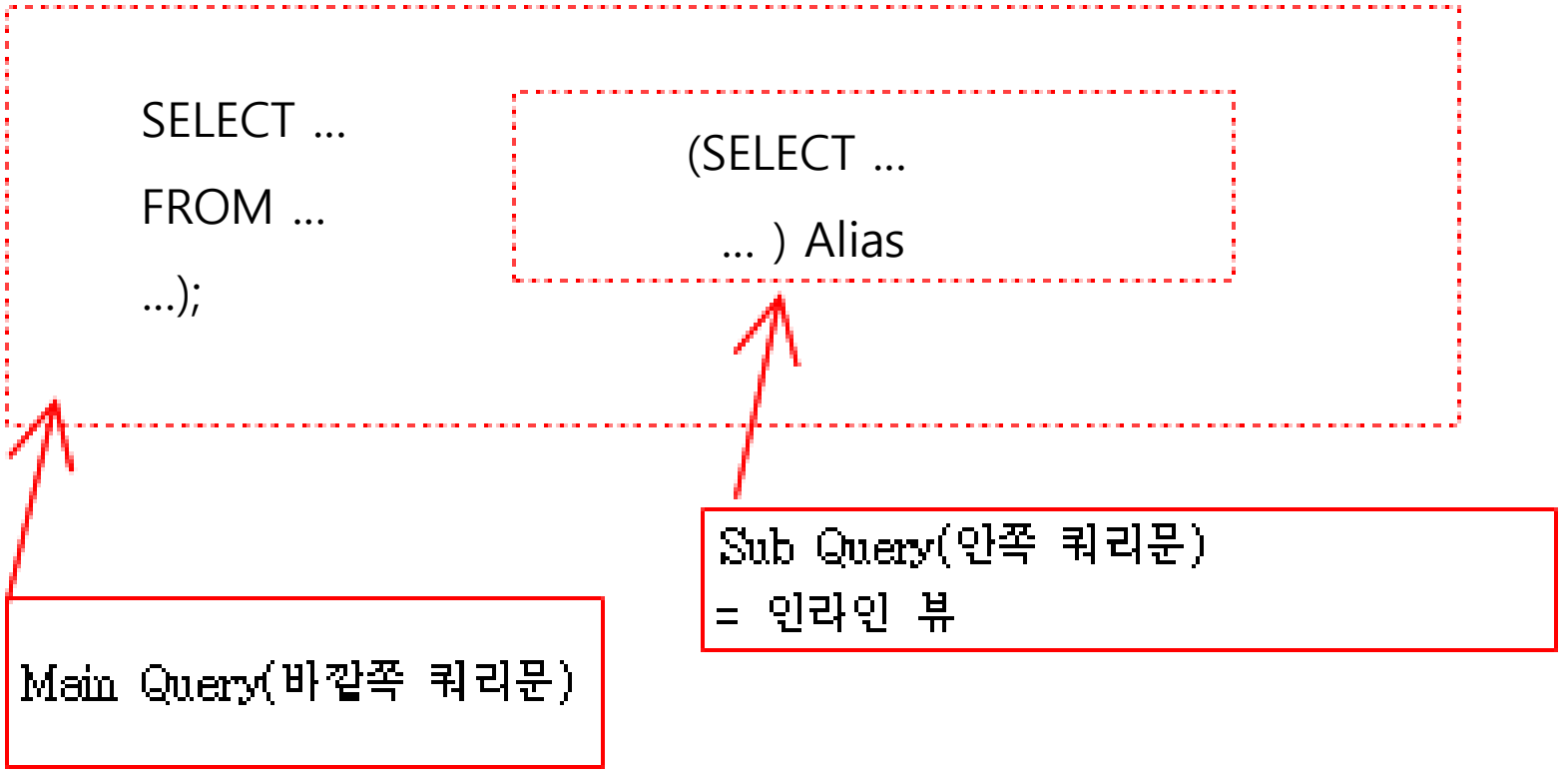
예

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

ROWNUM	EMPNO	ENAME	HIREDATE
1	1009	안성기	96/10/04
2	1013	박중훈	02/10/09
3	1004	이병헌	03/09/02
4	1006	장동건	03/10/09
5	1007	이눈세	04/01/08

6.2 인라인 뷰로 TOP-N 구하기(1)

- 인라인 뷰는 SQL 문장에서 사용하는 서브 쿼리의 일종으로 보통 FROM 절에 위치해서 테이블처럼 사용하는 것이다.



6.2 인라인 뷰로 TOP-N 구하기(2)

- 인라인 뷰란 메인 쿼리의 SELECT 문의 FROM 절 내부에 사용된 서브 쿼리 문을 말한다.
- 우리가 지금까지 생성한 뷰는 CREATE 명령어로 뷰를 생성했지만, 인라인 뷰는 SQL 문 내부에 뷰를 정의하고 이를 테이블처럼 사용한다.
- ex) 인라인 뷰를 사용해서 입사일이 빠른 사람 5명만을 얻어오기로 하자. 아래 문장을 보면 FROM 절 다음인 VIEW_HIRE 위치에 VIEW_HIRE를 정의할 때 사용한 서브 쿼리문을 기술한 것뿐이다.

예

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

문제

3. 인라인 뷰를 사용하여 급여를 많이 받는 순서대로 3명만 출력하시오.

RANKING	EMPNO	ENAME	SAL
1	1009	안성기	1000
2	1004	이병헌	600
3	1013	박중훈	560