

제 8 장



뷰와 시스템 카탈로그

- 8.1 뷰
- 8.2 관계 DBMS의 시스템 카탈로그
- 8.3 MS SQL Server의 시스템 카탈로그
 - 연습문제

제 8 장



뷰와 시스템 카탈로그

- 8.1 뷰
- 8.2 관계 DBMS의 시스템 카탈로그
- 8.3 MS SQL Server의 시스템 카탈로그
 - 연습문제

제 8 장



뷰와 시스템 카탈로그

- 8.1 뷰
- 8.2 관계 DBMS의 시스템 카탈로그
- 8.3 MS SQL Server의 시스템 카탈로그
 - 연습문제

제 8 장



뷰와 시스템 카탈로그

- 8.1 뷰
- 8.2 관계 DBMS의 시스템 카탈로그
- 8.3 MS SQL Server의 시스템 카탈로그
 - 연습문제

8장. 뷰와 시스템 카탈로그

□ 뷰와 시스템 카탈로그

- ✓ 관계 데이터베이스 시스템의 뷰(view)는 다른 릴레이션으로부터 유도된 릴레이션(derived relation)으로서 ANSI/SPARC 3단계 아키텍처의 외부 뷰와 다름
- ✓ 뷰는 관계 데이터베이스 시스템에서 데이터베이스의 보안 메카니즘으로서, 복잡한 질의를 간단하게 표현하는 수단으로서, 데이터독립성을 높이기 위해서 사용됨
- ✓ 시스템 카탈로그는 시스템내의 객체(기본 릴레이션, 뷰, 인덱스, 사용자, 접근 권한 등)에 관한 정보를 포함
- ✓ 시스템 카탈로그를 적절히 활용하면 원하는 릴레이션을 데이터베이스에서 찾고, 그 릴레이션에 어떤 애트리뷰트들이 들어 있으며, 각 애트리뷰트의 데이터 타입은 무엇인가 등을 쉽게 파악할 수 있음
- ✓ 시스템 카탈로그는 데이터베이스를 효율적으로 활용하는데 크게 도움이 됨

8.1 뷰

□ 뷰의 개요 – 뷰 – materialized view, modification view

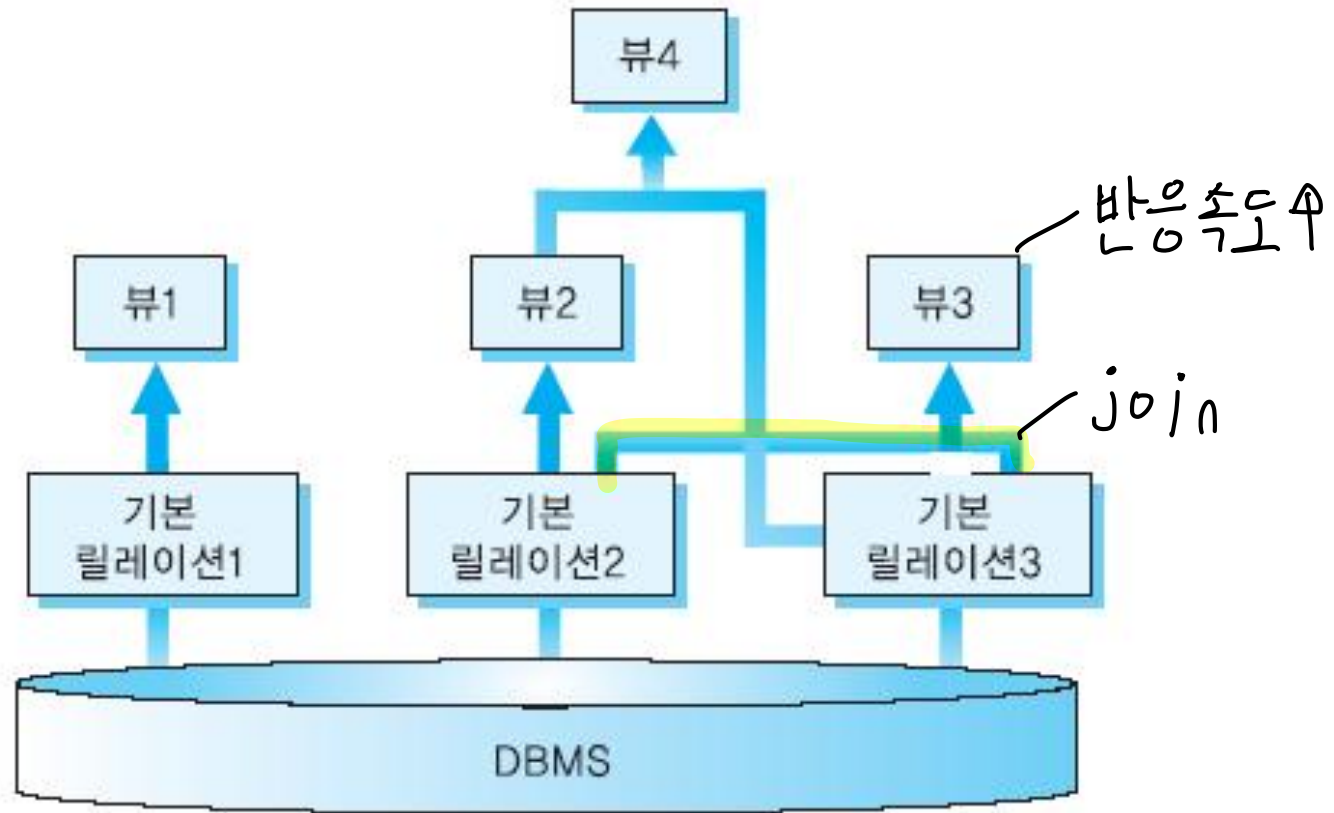
- ✓ ANSI/SPARC 3단계 아키텍처에서 외부 뷰는 특정 사용자가 보는 데이터베이스의 구조 \neq 뷰
- ✓ 관계 데이터베이스에서의 뷰는 한 사용자의 전체 외부 뷰 대신에 하나의 가상 릴레이션(virtual relation)을 의미하는데 사용됨
- ✓ 뷰는 기존의 기본 릴레이션(base relation. 실제 릴레이션)에 대한 SELECT문의 형태로 정의됨
- ✓ 사용자는 여러 개의 릴레이션과 뷰를 사용할 수 있음
- ✓ 뷰는 기본 릴레이션이 나타날 수 있는 곳은 어디든지 사용될 수 있음
- ✓ 뷰는 릴레이션으로부터 데이터를 검색하거나 갱신할 수 있는 동적인 창(dynamic window)의 역할

8.1 뷰(계속)

□ 뷰의 개요(계속)

- ✓ 이에 반해서 어느 시점에 SELECT문의 결과를 기본 릴레이션의 형태로 저장해 놓은 것을 **스냅샷(snapshot)**이라고 부름
- ✓ 스냅샷은 사진을 찍은 것과 같아서 스냅샷을 정의하는 시점의 기본 릴레이션의 내용이 스냅샷에 반영됨
- ✓ 어떤 시점의 조직체의 현황, 예를 들어 몇년 몇월 시점에 근무하던 직원들의 정보, 재고 정보 등이 스냅샷으로 정의될 수 있음

8.1 뷰(계속)



[그림 8.1] 뷰와 기본 릴레이션의 관계

8.1 뷰(계속)

□ 뷰의 정의

- ✓ 뷰를 정의하는 SQL문의 구문

CREATE VIEW 뷰이름 [(애트리뷰트(들))]
AS SELECT문
[WITH CHECK OPTION];

CREATE VIEW __ (,)
AS SELECT
FROM
WHERE
WITH CHECK OPTION;

- ✓ 뷰의 이름 다음에 애트리뷰트들을 생략하면 뷰를 정의하는데 사용된 SELECT문의 SELECT절에 열거된 애트리뷰트들의 이름과 동일한 애트리뷰트들이 뷰에 포함됨
- ✓ 뷰를 정의하는 SELECT절에 산술식 또는 집단 함수에 사용된 애트리뷰트가 있는 경우, 뷰의 정의에 조인이 포함되어 있고 두 개 이상의 다른 릴레이션으로부터 가져온 애트리뷰트들의 이름이 같아서 뷰에서 두 개 이상의 애트리뷰트의 이름이 같게 되는 경우에는 뷰를 정의할 때 모든 애트리뷰트들의 이름을 지정해야 함

AVG, SUM...
SALARY * 1.02..

한개까진
OK.

8.1 뷰(계속)

예: 한 릴레이션 위에서 뷰를 정의

그림 4.8의 EMPLOYEE 릴레이션에 대해서 “3번 부서에 근무하는 직원들의
사원번호, 사원이름, 직책으로 이루어진 뷰”를 정의해보자. 아래의 뷰의
정의에는 뷰의 애트리뷰트들을 별도로 명시했기 때문에 뷰에는 EMPNO,
EMPNAME, TITLE의 세 애트리뷰트가 포함됨

```
CREATE VIEW    EMP_DNO3 (ENO, ENAME, TITLE)
AS SELECT    EMPNO, EMPNAME, TITLE
FROM         EMPLOYEE
WHERE        DNO=3;
```

8.1 뷰(계속)

EMPLOYEE	EMPNO	EMPNAME	TITLE	MANAGER	SALARY	DNO
	2106	김창섭	대리	1003	2500000	2
	3426	박영권	과장	4377	3000000	1
	3011	이수민	부장	4377	4000000	3
	1003	조민희	과장	4377	3000000	2
	3427	최종철	사원	3011	1500000	3
	1365	김상원	사원	3426	1500000	1
	4377	이성래	사장	^	5000000	2

[그림 8.2] EMP_DNO3을 통해서 접근할 수 있는 부분(파란색 음영)

8.1 뷰(계속)

예: 두 릴레이션 위에서 뷰를 정의 조건.

그림 4.8의 (EMPLOYEE와 DEPARTMENT) 릴레이션에 대해서 “기획부에 근무하는 (사원들의 이름, 직책, 급여로 이루어진 뷰”를 정의해보자. 아래의 뷰의 정의에는 뷰의 애트리뷰트들을 별도로 명시하지 않았기 때문에 뷰에 속하는 애트리뷰트들의 이름은 기본 릴레이션의 애트리뷰트들의 이름과 같다. 즉 뷰에는 EMPNAME, TITLE, SALARY의 세 애트리뷰트가 포함된다.

```
CREATE VIEW EMP_PLANNING
AS SELECT E.EMPNAME, E.TITLE, E.SALARY
FROM EMPLOYEE E, DEPARTMENT D
WHERE E.DNO=D.DEPTNO
AND D.DEPTNAME = '기획';
```

← join했으니
필하는 애트리뷰트 이름
X.

객체에 관한 정보
모여있음.

8.1 뷰(계속)

- 뷰를 사용하여 데이터를 접근할 때 관계 DBMS에서 거치는 과정
- ✓ 시스템 카탈로그로부터 뷰의 정의, 즉 SELECT문을 검색
 - ✓ 기본 릴레이션에 대한 뷰의 접근 권한을 검사
 - ✓ 뷰에 대한 질의를 기본 릴레이션에 대한 동등한 질의로 변환

<pre>SELECT * FROM EMP_DNO3 WHERE TITLE='사원' ;</pre>	→	<pre>SELECT EMPNO, EMPNAME, TITLE FROM EMPLOYEE WHERE TITLE='사원' AND DNO=3;</pre>
--	---	---

정의 접근권한 동등한 질의

8.1 뷰(계속)

□ 뷰의 장점

✓ 뷰는 복잡한 질의를 간단하게 표현할 수 있게 함

- 기획부에 근무하는 사원들 중에서 직책이 부장인 사원의 사원이름과 급여를 검색하는 질의를 기본 릴레이션을 사용하여 표현하면 아래와 같이 다소 복잡한 형태의 질의가 됨

```
SELECT  E.EMPNAME, E.SALARY
FROM    EMPLOYEE E, DEPARTMENT D
WHERE   D.DEPTNAME = '기획'
        AND D.DEPTNO = E.DNO — join
        AND E.TITLE = '부장';
```

- 뷰에 대해서 같은 결과를 검색하는 질의를 표현하면

```
SELECT  EMPNAME, SALARY
FROM    EMP_PLANNING
WHERE   TITLE = '부장';
```

8.1 뷰(계속)

□ 뷰의 장점(계속)

✓ 뷰는 데이터 무결성을 보장하는데 활용됨

- 기본적으로 뷰를 통해 튜플을 추가하거나 수정할 때 튜플이 뷰를 정의하는 SELECT문의 WHERE절의 기준에 맞지 않으면 뷰의 내용에서 사라짐

```
UPDATE EMP_DNO3  
SET DNO = 2  
WHERE ENO = 3427;
```

거부

- ✱ 이 뷰의 정의할 때 WITH CHECK OPTION을 명시했다고 가정하자

```
CREATE VIEW EMP_DNO3 (ENO, ENAME, TITLE)  
AS SELECT EMPNO, EMPNAME, TITLE  
FROM EMPLOYEE  
WHERE DNO = 3  
WITH CHECK OPTION;
```

8.1 뷰(계속)

□ 뷰의 장점(계속)

✓ 뷰는 데이터 독립성을 제공함

- 뷰는 데이터베이스의 구조가 바뀌어도 기존의 질의(응용 프로그램)를 다시 작성할 필요성을 줄이는데 사용될 수 있음
- 예: 응용의 요구사항이 변경되어 기존의 EMPLOYEE 릴레이션이 두 개의 릴레이션 EMP1(EMPNO, EMPNAME, SALARY)과 EMP2(EMPNO, TITLE, MANAGER, DNO)로 분해되었다고 가정하자. 응용 프로그램에서 기존의 EMPLOYEE 릴레이션을 접근하던 SELECT문은 더 이상 수행되지 않으므로, EMP1과 EMP2에 대한 SELECT문으로 변경해야 한다.
- 아래와 같이 EMPLOYEE라는 뷰를 정의했다면 응용 프로그램에서 EMPLOYEE 릴레이션을 접근하던 SELECT문은 계속해서 수행될 수 있음

8.1 뷰(계속)

□ 뷰의 장점(계속)

- ✓ 뷰는 데이터 독립성을 제공함(계속)

정의만 다시 해주면 된다.

```
CREATE VIEW EMPLOYEE
AS SELECT E1.EMPNO, E1.EMPNAME, E2.TITLE, E2.MANAGER,
          E1.SALARY, E2.DNO
FROM EMP1 E1, EMP2 E2
WHERE E1.EMPNO = E2.EMPNO;
```

8.1 뷰(계속)

□ 뷰의 장점(계속)

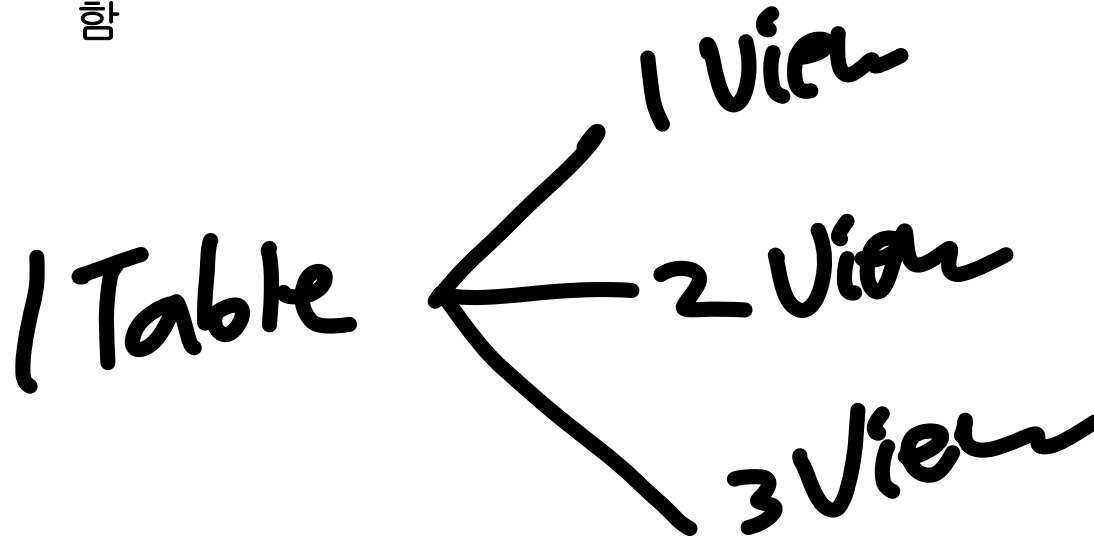
✓ 뷰는 데이터 보안 기능을 제공함

- 뷰는 뷰의 원본이 되는 기본 릴레이션에 직접 접근할 수 있는 권한을 부여하지 않고 뷰를 통해 데이터를 접근하도록 하기 때문에 보안 메커니즘으로 사용할 수 있음
- 뷰는 일반적으로 기본 릴레이션의 일부 애트리뷰트들 또는 일부 튜플들을 검색하는 SELECT문으로 정의되므로 뷰를 통해서 기본 릴레이션을 접근하면 기본 릴레이션의 일부만 검색할 수 있음
- 예: EMPLOYEE 릴레이션의 SALARY 애트리뷰트는 숨기고 나머지 애트리뷰트들은 모든 사용자가 접근할 수 있도록 하려면 SALARY 애트리뷰트를 제외하고 EMPLOYEE 릴레이션의 모든 애트리뷰트를 포함하는 뷰를 정의하고, 사용자에게 뷰에 대한 SELECT 권한을 허가

8.1 뷰(계속)

□ 뷰의 장점(계속)

- ✓ 동일한 데이터에 대한 여러 가지 뷰를 제공함
 - 뷰는 사용자들의 그룹이 각자 특정한 기준에 따라 데이터를 접근하도록 함



8.1 뷰(계속)

□ 뷰의 갱신

- ✓ 뷰에 대한 검색이 기본 릴레이션에 대한 검색으로 변환되듯이 뷰에 대한 갱신도 기본 릴레이션에 대한 갱신으로 변환됨
- ✓ 아래의 갱신들이 성공적으로 수행될 수 있는가?
- ✓ 갱신 1: 한 릴레이션 위에서 정의된 뷰에 대한 갱신

```
INSERT INTO EMP_DNO3  
VALUES (4293, '김정수', '사원');
```



```
INSERT INTO EMPLOYEE  
VALUES (4293, '김정수', '사원', , , );
```

기본키 다중성

MANAGER, SALARY: null

DNO: 1 (다중성)

8.1 뷰(계속)

❌ 뷰의 갱신(계속)

✓ 갱신 2: 두 개의 릴레이션 위에서 정의된 뷰에 대한 갱신 ^{조건}

✗

```
INSERT INTO EMP_DNO3  
VALUES (4293, '김정수', '사원');
```

```
INSERT INTO EMPLOYEE  
VALUES (4293, '김정수', '사원', , , );
```

조건 뷰 갱신 가능..?

✗
INSERT INTO EMP_PLANNING
VALUES ('박지선', '대리', 2500000);

➡
INSERT INTO EMPLOYEE
VALUES (, '박지선', '대리', 2500000,);

EMP_PLANNING! 두개의 릴레이션
조인하여 정의된 뷰. 갱신 불가능.
(대부분).

8.1 뷰(계속)

□ 뷰의 갱신(계속) SUM, AVG, MAX, MIN, COUNT, DISTINCT, GROUP BY, HAVING

✓ 갱신 3: 집단 함수 등을 포함한 뷰에 대한 갱신 → 불가능.

X

```
CREATE VIEW EMP_AVGSAL (DNO, AVGSAL)
AS SELECT DNO, AVG(SALARY)
FROM EMPLOYEE
GROUP BY DNO;
```

집단 함수의 결과가 세트인 뷰트에 포함. 뷰의 이름 때문에 반드시 뷰의 세트 뷰드들의 이름 명시. ○

```
X UPDATE EMP_AVGSAL
SET AVGSAL = 3000000
WHERE DNO = 2;
```

계산된 평균급여값 포함.

2번부서의 평균급여 값을 3000000으로 수정하라? (X)

기존 릴레이션에서도 불가능.

```
X INSERT INTO EMP_AVGSAL
VALUES (3, 3200000);
```

3번 부서이면서 평균급여가 3200000 인투플 삽입. → 릴레이션에 반영 결정함. X.

부서 평균급여

UPDATE, INSERT → 집단 함수.

8.1 뷰(계속)

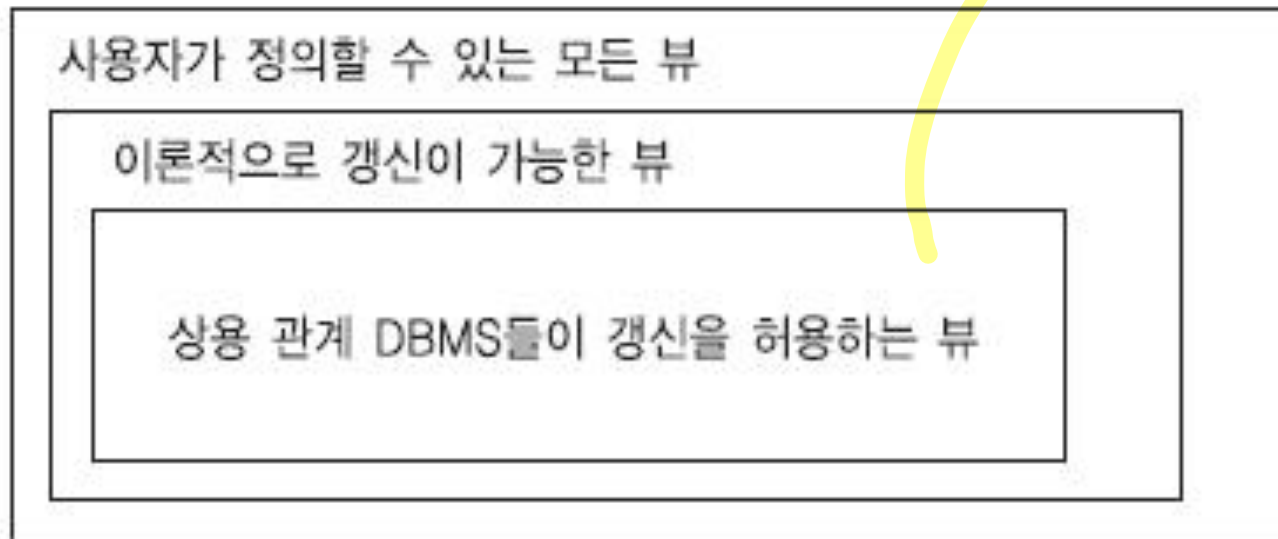
❑ 갱신이 불가능한 뷰

- ✓ 한 릴레이션 위에서 정의되었으나 그 릴레이션의 기본 키가 포함되지 않은 뷰
- ✓ 기본 릴레이션의 атри뷰트들 중에서 뷰에 포함되지 않은 атри뷰트에 대해 NOT NULL이 지정되어 있을 때 ex) 기본 키
- ✓ 집단 함수가 포함된 뷰
- ✓ 조인으로 정의된 뷰

디폴트값
적용시 허용.

8.1 뷰(계속)

실제로 갱신이
허용되는 경우 적음.



[그림 8.3] 갱신 가능성 기준에 따른 뷰들의 유형

8.2 관계 DBMS의 시스템 카탈로그

□ 시스템 카탈로그

- ✓ 시스템 카탈로그는 데이터베이스의 객체(사용자, 릴레이션, 뷰, 인덱스, 권한 등)와 구조들에 관한 모든 데이터를 포함
- ✓ 시스템 카탈로그를 메타데이터라고 함. 메타데이터는 데이터에 관한 데이터라는 의미
- ✓ 시스템 카탈로그는 사용자 및 질의 최적화 모듈 등 DBMS 자신의 구성요소에 의해서 사용됨
- ✓ 시스템 카탈로그는 관계 DBMS마다 표준화되어 있지 않아서 관계 DBMS마다 서로 다른 형태로 시스템 카탈로그 기능을 제공함
- ✓ 시스템 카탈로그는 데이터 사전(data dictionary) 또는 시스템 테이블이라고도 부름

뷰: 유도된 릴레이션
가상 릴레이션
동적인 카.

8.2 관계 DBMS의 시스템 카탈로그(계속)

- 시스템 카탈로그가 질의 처리에 어떻게 활용되는가

```
SELECT  EMPNAME, SALARY, SALARY * 1.1
FROM    EMPLOYEE
WHERE   TITLE = '과장' AND DNO = 2;
```

카탈로그를 처리면서
수행되는 것.
카탈로그에 정보 있음.

- DBMS 수행 →
- ✓ SELECT문이 문법적으로 정확한가를 검사함 세미콜론 찍었나? ...
 - ✓ SELECT문에서 참조하는 EMPLOYEE 릴레이션이 데이터베이스에 존재하는가를 검사함
 - ✓ EMPLOYEE 릴레이션에 SELECT절에 열거된 애트리뷰트와 WHERE절에서 조건에 사용된 애트리뷰트가 존재하는가를 확인함
 - ✓ SALARY 애트리뷰트가 수식에 사용되었으므로 이 애트리뷰트의 데이터 타입이 숫자형(정수형이나 실수형)인가를 검사하고, TITLE이 문자열과 비교되었으므로 이 애트리뷰트의 데이터 타입이 문자형(CHAR(n) 또는 VARCHAR(n) 등)인가를 검사함

8.2 관계 DBMS의 시스템 카탈로그(계속)

❑ 시스템 카탈로그가 질의 처리에 어떻게 활용되는가(계속)

- ✓ 이 질의를 입력한 사용자가 EMPLOYEE 릴레이션의 EMPNAME, SALARY 애트리뷰트를 검색할 수 있는 권한이 있는가를 확인함 *사용 권한*
- ✓ TITLE 애트리뷰트와 DNO 애트리뷰트에 인덱스가 정의되어 있는지 확인함
- ✓ 두 애트리뷰트에 각각 인덱스가 존재한다고 가정하자. DBMS가 두 인덱스 중에서 조건을 만족하는 튜플 수가 적은 것을 선택하기 위해서는 관계 데이터베이스 시스템에 데이터베이스 외에 추가로 정보를 유지해야 함
- ✓ 한 릴레이션의 전체 튜플 수와 그 릴레이션에 정의된 각 인덱스에 존재하는 상이한 값들의 개수를 유지한다면 어느 인덱스를 사용하는 것이 유리한가를 예상할 수 있음

적은수의 튜플이
검색되는 인덱스가
조용함.

$$\begin{array}{r} 106/5 \\ 100/3 \end{array}$$

8.2 관계 DBMS의 시스템 카탈로그(계속)

□ 시스템 카탈로그가 질의 처리에 어떻게 활용되는가(계속)

- ✓ 그림 4.8에서 EMPLOYEE 릴레이션의 전체 튜플 수는 7이고, TITLE 애트리뷰트에는 사원, 대리, 과장, 부장, 이사의 다섯 가지 값들이 존재함
- ✓ DNO 애트리뷰트에는 1, 2, 3의 세 가지 값들이 존재함
- ✓ 따라서 TITLE 애트리뷰트에 정의된 인덱스가 DNO에 정의된 인덱스보다
(대상 튜플들을 더 좁혀 주므로) 유리함

8.2 관계 DBMS의 시스템 카탈로그(계속)

□ 질의 최적화

- ✓ DBMS가 질의를 수행하는 여러 가지 방법들 중에서 가장 비용이 적게 드는 방법을 찾는 과정
- ✓ 질의 최적화 모듈이 정확한 결정을 내릴 수 있도록 DBMS는 자체 목적을 위해서 시스템 카탈로그에 다양한 정보를 유지함
- ✓ 사용자가 질의 최적화 모듈을 깊이 있게 이해할 필요는 없지만 질의 최적화 모듈이 정확한 수행 방법을 결정하기 위해서는 릴레이션에 관한 다양한 통계 정보가 정확하게 유지돼야 한다는 것을 알고 있는 것이 바람직

8.2 관계 DBMS의 시스템 카탈로그(계속)

— 관리의 목적

□ 관계 DBMS의 시스템 카탈로그

- ✓ 사용자 릴레이션과 마찬가지로 형태로 저장되기 때문에 사용자 릴레이션에 적용되는 회복 기법과 동시성 제어 기법을 동일하게 사용할 수 있음
- ✓ 시스템 카탈로그는 사용자 릴레이션처럼 SELECT문을 사용하여 내용을 검색할 수 있음 UPDATE, INSERT(x) 관리자가 사용자 임의로 넣지 x.
- ✓ 시스템 카탈로그에는 릴레이션, 애트리뷰트, 인덱스, 사용자, 권한 등 각 유형마다 별도의 릴레이션이 유지됨
- ✓ EMPLOYEE 릴레이션과 DEPARTMENT 릴레이션에 대해서 시스템 카탈로그에 어떤 정보들이 유지되는가를 이해하기 쉽도록 시스템 카탈로그를 매우 단순화하여 설명함
- ✓ 릴레이션에 관한 정보를 유지하는 릴레이션의 이름이 SYS_RELATION, 애트리뷰트에 관한 정보를 유지하는 릴레이션의 이름이 SYS_ATTRIBUTE라고 가정

8.2 관계 DBMS의 시스템 카탈로그(계속)

SYS_RELATION	<u>RelId</u>	RelOwner	RelTups	RelAtts	RelWidth
	EMPLOYEE	KIM	7	6	36
	DEPARTMENT	KIM	4	3	18

SYS_ATTRIBUTE	<u>AttrelId</u>	<u>AttID</u>	AttName	AttOff	AttType	AttLen	PkorFk
	EMPLOYEE	1	EMPNO	0	int	4	Pk
	EMPLOYEE	2	EMPNAME	4	char	10	
	EMPLOYEE	3	TITLE	14	char	10	
	EMPLOYEE	4	MANAGER	24	int	4	Fk
	EMPLOYEE	5	SALARY	28	int	4	
	EMPLOYEE	6	DNO	32	int	4	Fk
	DEPARTMENT	1	DEPTNO	0	int	4	Pk
	DEPARTMENT	2	DEPTNAME	4	char	10	
	DEPARTMENT	3	FLOOR	14	int	4	
	

[그림 8.4] SYS_RELATION과 SYS_ATTRIBUTE의 내용

8.2 관계 DBMS의 시스템 카탈로그(계속)

□ 시스템 카탈로그의 갱신

- ✓ 어떤 사용자도 시스템 카탈로그를 직접 갱신할 수 없음 *ex) INSERT, UPDATE, DELETE*
- ✓ 즉 DELETE, UPDATE 또는 INSERT문을 사용하여 시스템 카탈로그를 변경할 수 없음
- ✓ EMPLOYEE 릴레이션의 소유자인 KIM이 EMPLOYEE 릴레이션에서 MANAGER 애트리뷰트를 삭제하기 위해서
- ✓ 라고 하는 대신에 아래와 같이 시스템 카탈로그에 대해 DELETE문을 사용하면 DBMS가 거절함

DELETE FROM SYS_ATTRIBUTE

WHERE AttRelId = 'EMPLOYEE' **AND** AttName = 'MANAGER';

— 시스템 카탈로그

8.2 관계 DBMS의 시스템 카탈로그(계속)

□ 시스템 카탈로그에 유지되는 통계 정보

✓ 릴레이션마다

튜플의 크기, 튜플 수, 각 블록의 채우기 비율, 블록킹 인수,
릴레이션의 크기(블록 수)

✓ 뷰마다

뷰의 이름과 정의

✓ 애트리뷰트마다

애트리뷰트의 데이터 타입과 크기, 애트리뷰트 내의 상이한 값들의 수,
애트리뷰트 값의 범위, 선택율 (조건을 만족하는 튜플 수/전체 튜플 수)

(4/100)

8.2 관계 DBMS의 시스템 카탈로그(계속)

□ 시스템 카탈로그에 유지되는 통계 정보(계속)

✓ 사용자마다

접근할 수 있는 릴레이션과 권한

✓ 인덱스마다

인덱스된 애트리뷰트(키 애트리뷰트 또는 비 키 애트리뷰트),
클러스터링 인덱스/비 클러스터링 인덱스 여부, 밀집/희소 인덱스 여부,
인덱스의 높이, 1단계 인덱스의 블록 수

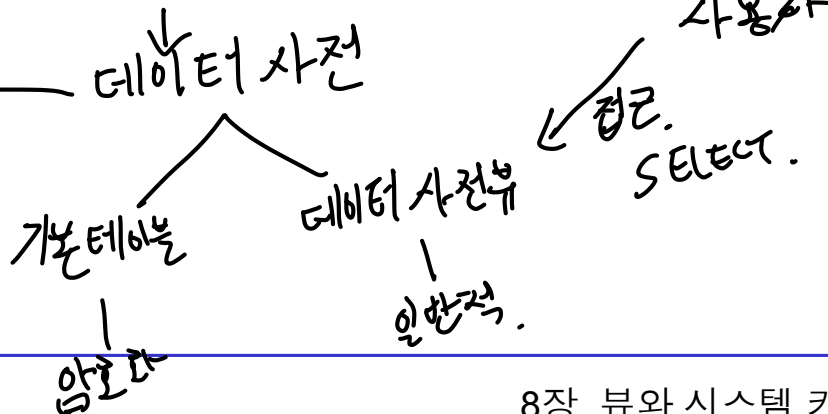
중요

8.3 오라클의 시스템 카탈로그

□ 오라클의 시스템 카탈로그

- ✓ 시스템 카탈로그를 **데이터 사전(data dictionary)**이라고 부름
- ✓ 데이터 사전은 **시스템 테이블스페이스에 저장됨**
- ✓ 데이터 사전은 **기본 테이블과 데이터 사전 뷰로 구성됨** 기본 테이블; 데이터베이스에 대한 설명을 포함
- ✓ 사용자는 **기본 테이블의 정보가 암호화된 형태로 저장되어 있기 때문에 직접 접근할 필요가 거의 없으며, 일반적으로 이해하기 쉬운 형식의 정보를 제공하는 데이터 사전 뷰를 접근**
- ✓ **데이터 사전을 접근하려면 SELECT문만 사용해야 함**

ORACLE



시스템
테이블스페이스

8.3 오라클의 시스템 카탈로그(계속)

❑ 데이터 사전 뷰의 세 부류

✓ DBA_xxx 뷰

데이터베이스 내의 모든 객체들에 관한 정보

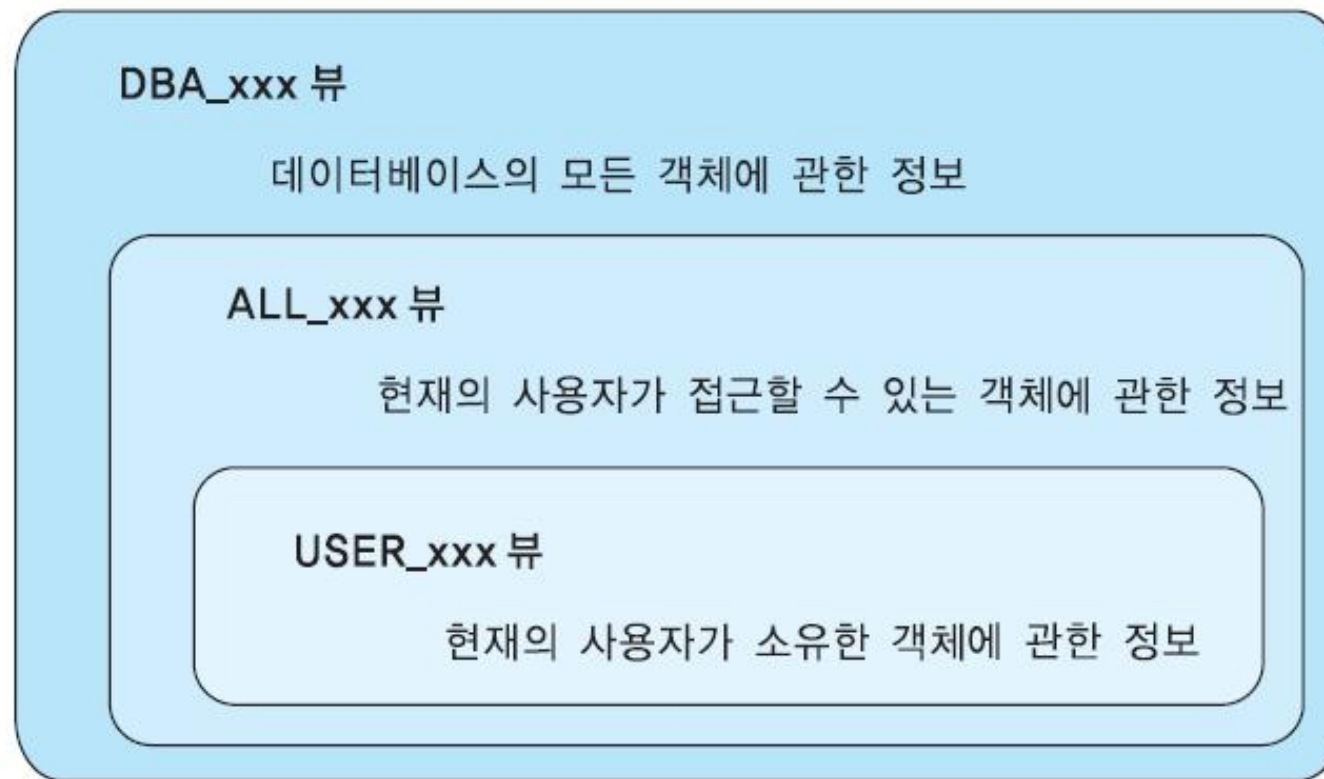
✓ ALL_xxx 뷰

현재의 사용자가 접근할 수 있는 객체들에 관한 정보 권한이 있는

✓ USER_xxx 뷰

현재의 사용자가 소유하고 있는 객체들에 관한 정보

8.3 오라클의 시스템 카탈로그(계속)



[그림 8.4] 데이터 사전 뷰들의 세 부류와 이들의 포함 관계

8.3 오라클의 시스템 카탈로그(계속)

〈표 8.1〉 주요 데이터 사전 뷰들의 이름과 기능

이름	기능
ALL_CATALOG	사용자가 접근할 수 있는 테이블, 뷰, 동의어에 관한 정보
ALL_CONSTRAINTS	사용자가 접근할 수 있는 테이블에 정의된 제약조건에 관한 정보
ALL_CONS_COLUMNS	사용자가 접근할 수 있는 제약조건 정의에 포함된 애트리뷰트에 관한 정보
ALL_SYNONYMS	사용자가 접근할 수 있는 동의어에 관한 정보
ALL_TABLES	사용자가 접근할 수 있는 뷰에 관한 정보. ALL_CATALOG보다 상세한 정보를 볼 수 있음
ALL_VIEWS	사용자가 접근할 수 있는 뷰에 관한 정보
DICTIONARY (또는 DICT)	데이터 사전 테이블과 뷰에 관한 정보
TABLE_PRIVILEGES	사용자가 소유자, 권한 허가자, 권한 피허가자인 객체 또는 권한 피허가자가 PUBLIC인 객체에 관한 정보
USER_CATALOG	사용자가 소유한 테이블, 뷰, 동의어 등에 관한 정보
USER_COL_PRIVS	사용자가 소유자, 권한 허가자, 권한 피허가자인 애트리뷰트에 관한 정보
USER_CONSTRAINTS	사용자가 소유한 제약조건에 관한 정보
USER_CONS_COLUMNS	사용자가 소유한 제약조건에 포함된 애트리뷰트에 관한 정보
USER_SYNONYMS	사용자의 개인적인 동의어에 관한 정보
USER_TABLES	사용자가 소유한 테이블에 관한 정보
USER_TAB_COLUMNS	사용자의 테이블, 뷰에 속한 애트리뷰트에 관한 정보
USER_TAB_PRIVS	사용자가 소유자, 권한 허가자, 권한 피허가자인 객체에 관한 정보
USER_VIEWS	사용자가 소유한 뷰에 관한 정보

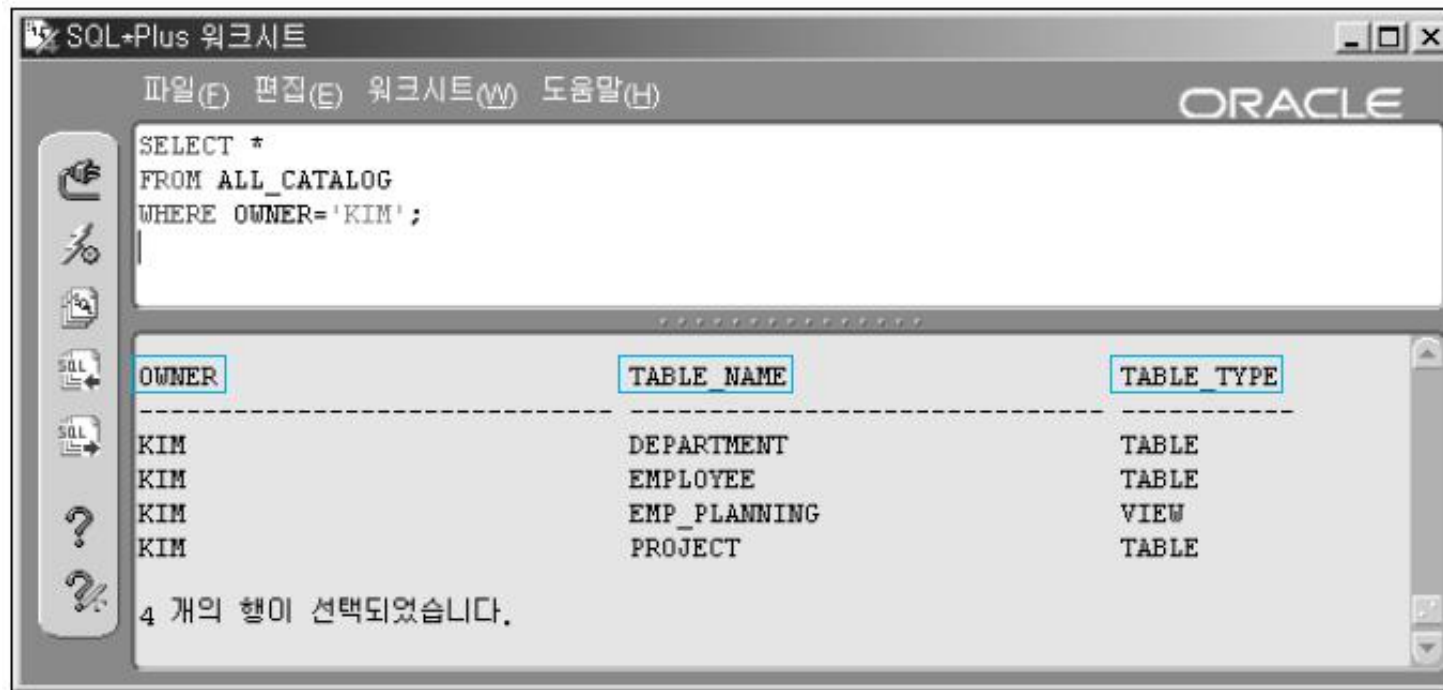
8.3 오라클의 시스템 카탈로그(계속)

- ❑ 사용자 KIM이 소유한 테이블이나 뷰에 관한 정보를 검색하기 위해서 KIM으로 SQL*Plus 워크시트에 로그인한 후에 다음과 같은 질의를 수행

```
SELECT      *  
FROM        ALL_CATALOG  
WHERE       OWNER='KIM';
```

- ✓ OWNER는 사용자의 이름, TABLE_NAME은 테이블이나 뷰의 이름, TABLE_TYPE은 테이블의 유형으로서 테이블, 뷰 등을 나타냄

8.3 오라클의 시스템 카탈로그(계속)



[그림 8.5] ALL_CATALOG를 사용하여 테이블과 뷰에 관한 정보 검색

8.3 오라클의 시스템 카탈로그(계속)

- ❑ 사용자 KIM이 소유한 EMPLOYEE 테이블의 애트리뷰트 정보를 찾기 위해서 다음과 같은 질의를 수행

```
SELECT    TABLE_NAME, COLUMN_NAME, DATA_TYPE
FROM      USER_TAB_COLUMNS
WHERE     TABLE_NAME = 'EMPLOYEE';
```

- ✓ TABLE_NAME은 테이블의 이름, COLUMN_NAME은 애트리뷰트의 이름, DATA_TYPE은 애트리뷰트의 데이터 타입을 각각 나타냄
- ✓ 이밖에도 USER_TAB_COLUMNS 뷰를 통해서 각 애트리뷰트의 길이, 널값 허용 여부, 디폴트값 등을 검색할 수 있음

8.3 오라클의 시스템 카탈로그(계속)



[그림 8.6] EMPLOYEE 테이블의 애트리뷰트들에 관한 정보 검색

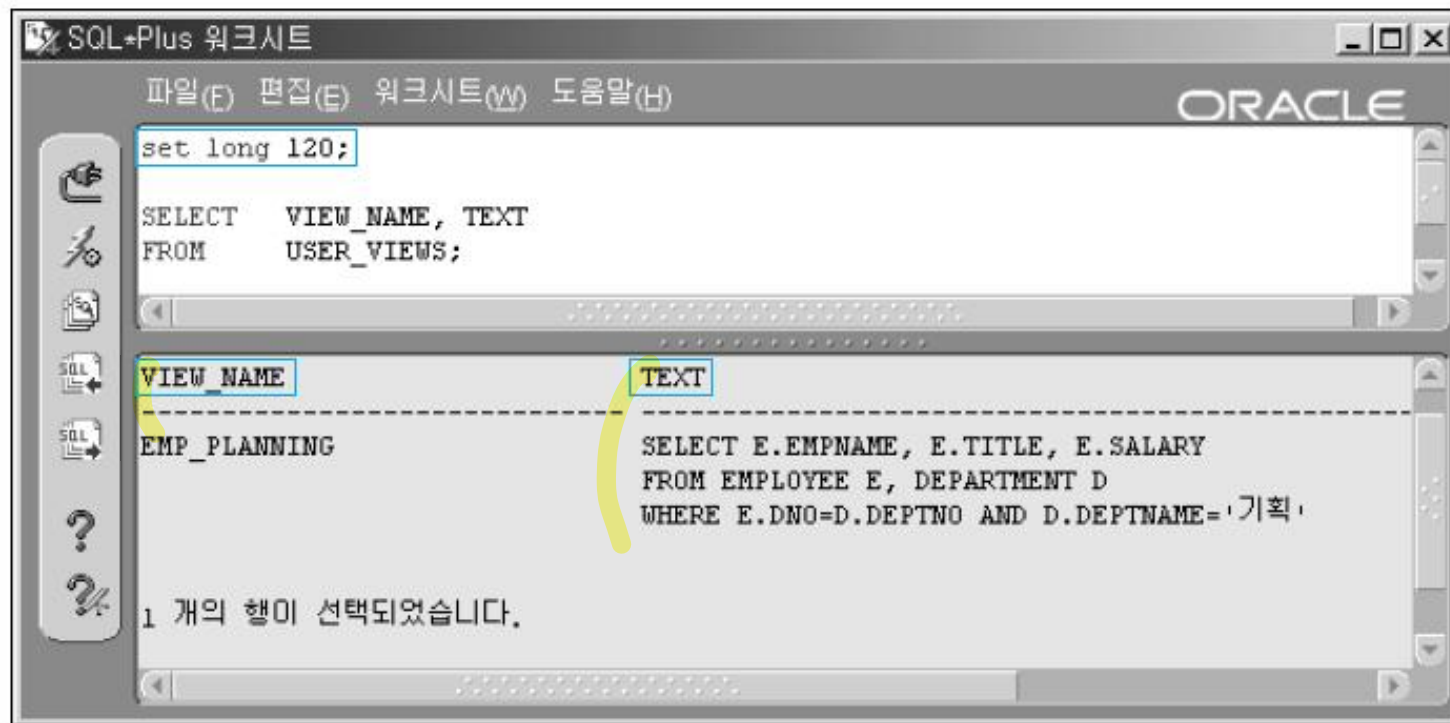
8.3 오라클의 시스템 카탈로그(계속)

- 3장의 예제 3.2에서 생성한 EMP_PLANNING 뷰가 어떤 SELECT문으로 정의되어 있는가를 알기 위해서 다음과 같은 질의를 수행

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

- ✓ 묵시적으로 뷰의 정의문이 80문자까지만 표시되므로 set 명령을 사용하여 120문자까지 나타나도록 설정
- ✓ VIEW_NAME은 뷰의 이름이고, TEXT는 뷰를 정의한 SQL문

8.3 오라클의 시스템 카탈로그(계속)



[그림 8.7] 뷰에 관한 정보 검색

8.3 오라클의 시스템 카탈로그(계속)

→ 갱신불가.

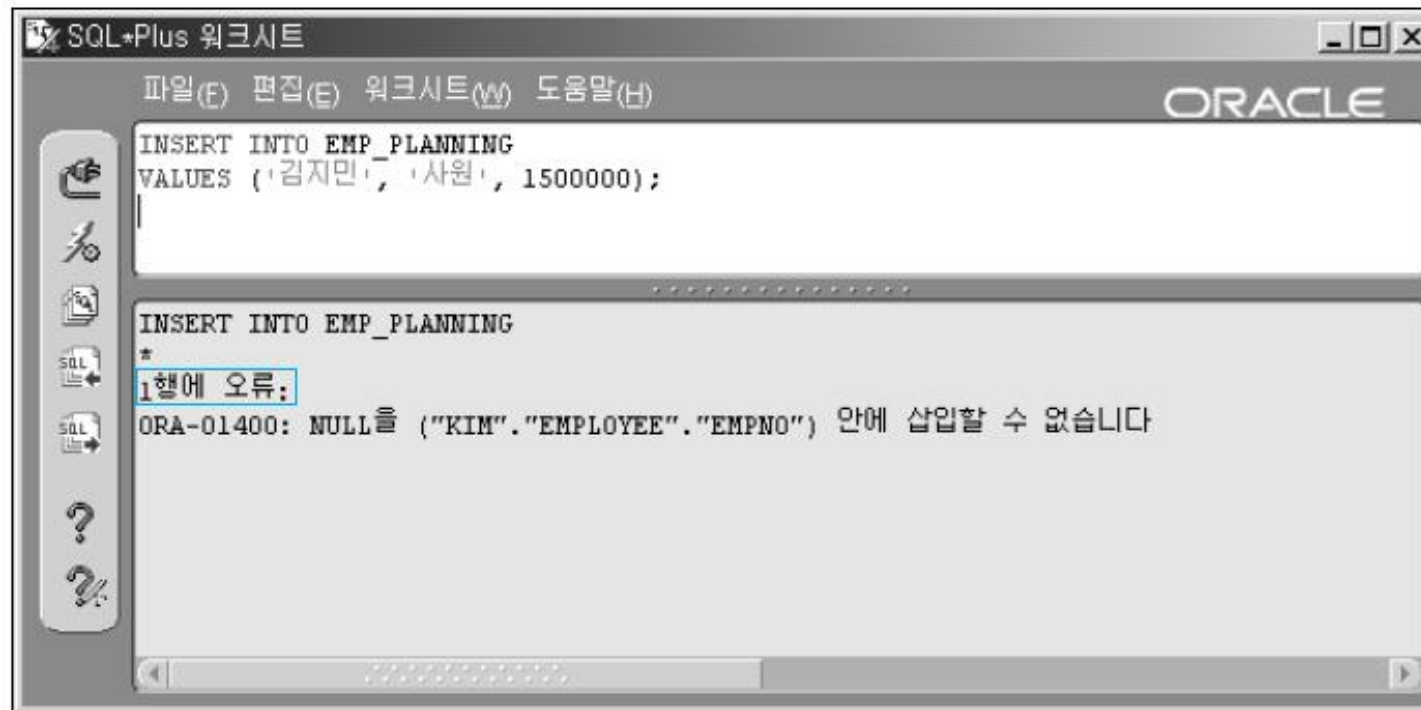
□EMP_PLANNING 뷰는 EMPLOYEE와 DEPARTMENT 테이블을 조인해서 정의한 뷰인데, 이 뷰가 갱신이 가능한지 확인해보기 위해서 아래와 같은 INSERT문을 수행

```
INSERT INTO EMP_PLANNING  
VALUES ('김지민', '사원', 1500000);
```



- ✓ 이 뷰에는 EMPLOYEE 테이블의 기본 키인 EMPNO가 포함되지 않았으므로 뷰를 통해서 아래와 같이 튜플을 삽입하면 EMPNO의 값을 입력하는 것이 불가능

8.3 오라클의 시스템 카탈로그(계속)

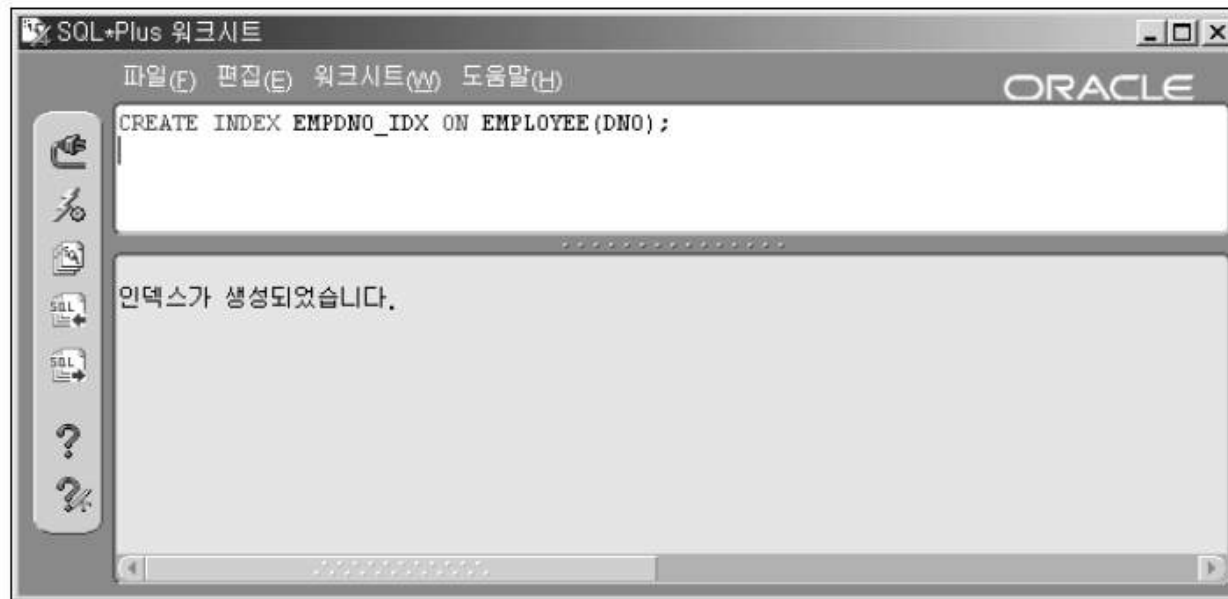


[그림 8.8] 두 테이블의 조인 뷰에 대한 INSERT문 수행

8.3 오라클의 시스템 카탈로그(계속)

- ❑ EMPLOYEE 테이블의 부서번호 애트리뷰트인 DNO에 대해 인덱스를 생성하고, 생성된 인덱스를 통해서 통계 정보를 확인

CREATE INDEX EMPDNO_IDX **ON** EMPLOYEE(DNO);



[그림 8.9] EMPLOYEE 테이블에 인덱스 정의

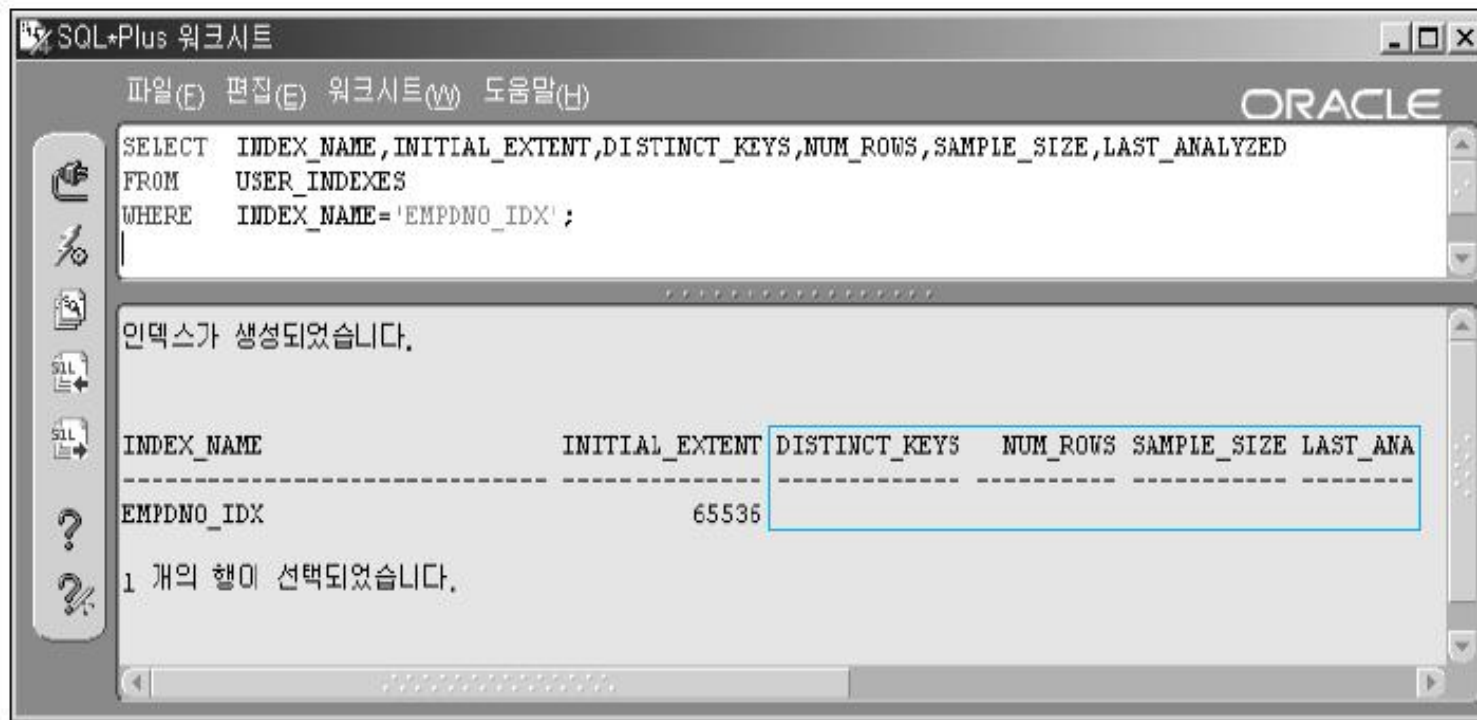
8.3 오라클의 시스템 카탈로그(계속)

□ 통계 정보를 확인하기 위해서 아래와 같은 명령문을 수행

```
SELECT  INDEX_NAME, INITIAL_EXTENT, DISTINCT_KEYS,  
          NUM_ROWS, SAMPLE_SIZE, LAST_ANALYZED  
FROM    USER_INDEXES  
WHERE    INDEX_NAME = 'EMPDNO_IDX';
```

- ✓ INDEX_NAME은 인덱스의 이름, INITIAL_EXTENT는 초기 익스텐트의 크기, DISTINCT_KEYS는 상이한 인덱스 값들의 개수, NUM_ROWS는 테이블의 튜플 수, SAMPLE_SIZE는 인덱스 분석을 위해 사용된 튜플 수, LAST_ANALYZED는 통계가 마지막으로 갱신된 날짜를 나타냄
- ✓ 인덱스를 정의했다고 해서 자동적으로 통계 정보가 구해지는 것은 아님

8.3 오라클의 시스템 카탈로그(계속)



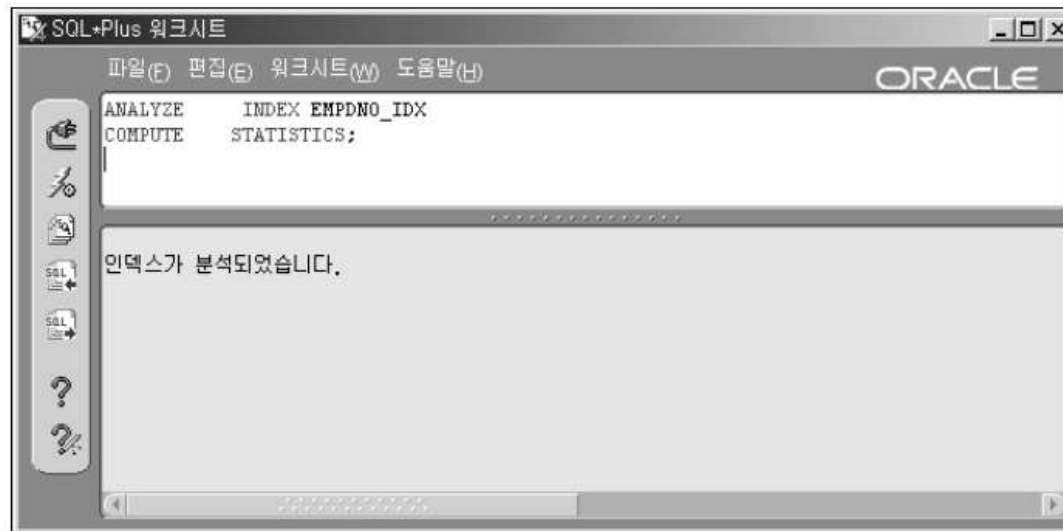
[그림 8.10] 인덱스 EMPDNO_IDX의 통계 정보

8.3 오라클의 시스템 카탈로그(계속)

□ 통계 정보는 ANALYZE문을 사용하여 갱신할 수 있음

```
ANALYZE INDEX EMPDNO_IDX  
COMPUTE STATISTICS;
```

- ✓ 테이블에 대한 통계 정보는 ANALYZE TABLE 명령을 사용해서 갱신하고,
인덱스에 대한 통계 정보는 ANALYZE INDEX 명령을 사용해서 갱신



[그림 8.11] ANALYZE 명령의 수행

8.3 오라클의 시스템 카탈로그(계속)

□ ANALYZE 명령의 문법

ANALYZE 객체_유형 객체_이름 연산 **STATISTICS;**
Table View COMPUTE ESTIMATE

- ✓ 객체_유형은 테이블, 인덱스 등을 나타냄
- ✓ 객체_이름은 객체의 이름을 의미
- ✓ 연산에 **COMPUTE**가 사용되면 전체 테이블을 접근하여 통계 정보를 계산
- ✓ 연산에 **ESTIMATE**가 사용되면 데이터 표본을 추출하여 통계 정보를 구함
- ✓ 주기적으로 ANALYZE 작업을 수행해야 함
- ✓ 테이블을 재생성한 경우, 인덱스를 추가하거나 재생성한 경우, 다량의 데이터를 일괄 작업으로 처리한 경우에는 바로 ANALYZE 작업을 수행하는 것이 필요

연산에 DELETE 사용시 테이블 모든 통계정보 삭제.

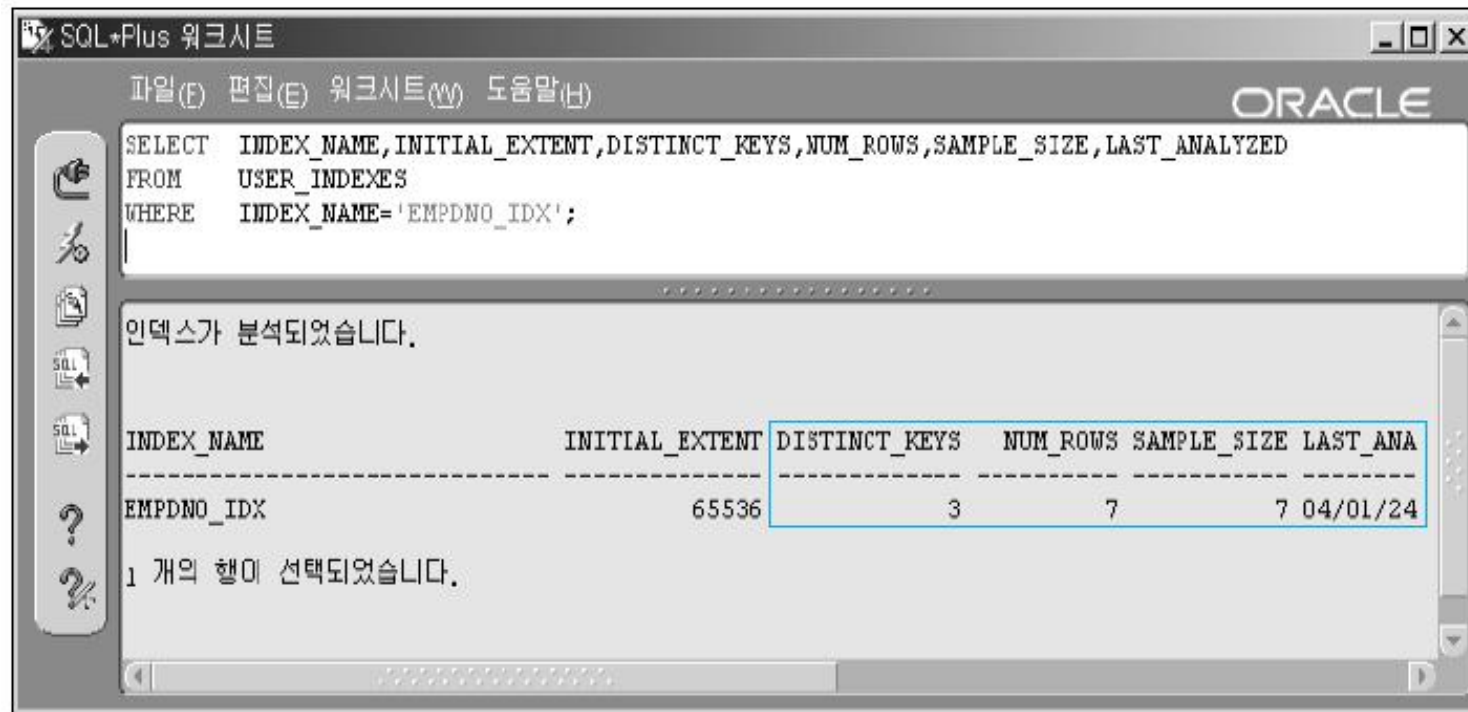
정확한 통계, 처리속도↓
덜정확, 처리속도↑

8.3 오라클의 시스템 카탈로그(계속)

- ❑ ANALYZE 명령의 수행 결과를 확인하기 위해서 아래와 같은 명령문을 다시 수행

```
SELECT  INDEX_NAME, INITIAL_EXTENT, DISTINCT_KEYS,  
          NUM_ROWS, SAMPLE_SIZE, LAST_ANALYZED  
FROM    USER_INDEXES  
WHERE    INDEX_NAME = 'EMPDNO_IDX';
```

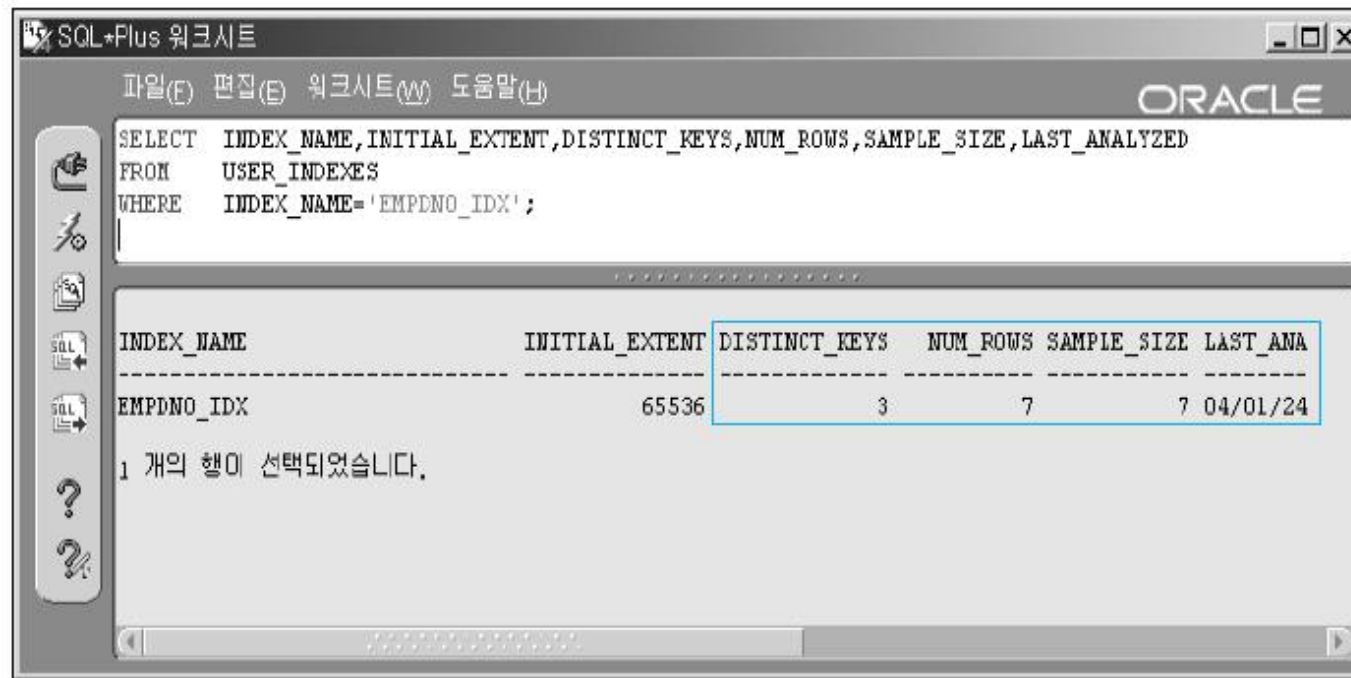
8.3 오라클의 시스템 카탈로그(계속)



[그림 8.12] ANALYZE 수행 후의 인덱스 EMPDNO_IDX의 통계 정보

8.3 오라클의 시스템 카탈로그(계속)

- ❑ 테이블에 튜플이 삽입되자마자 통계 정보가 갱신되지는 않음



The screenshot shows the SQL*Plus interface with a query window titled 'SQL*Plus 워크시트'. The query is: `SELECT INDEX_NAME, INITIAL_EXTENT, DISTINCT_KEYS, NUM_ROWS, SAMPLE_SIZE, LAST_ANALYZED FROM USER_INDEXES WHERE INDEX_NAME='EMPDNO_IDX';`. The result is displayed in a table with the following data:

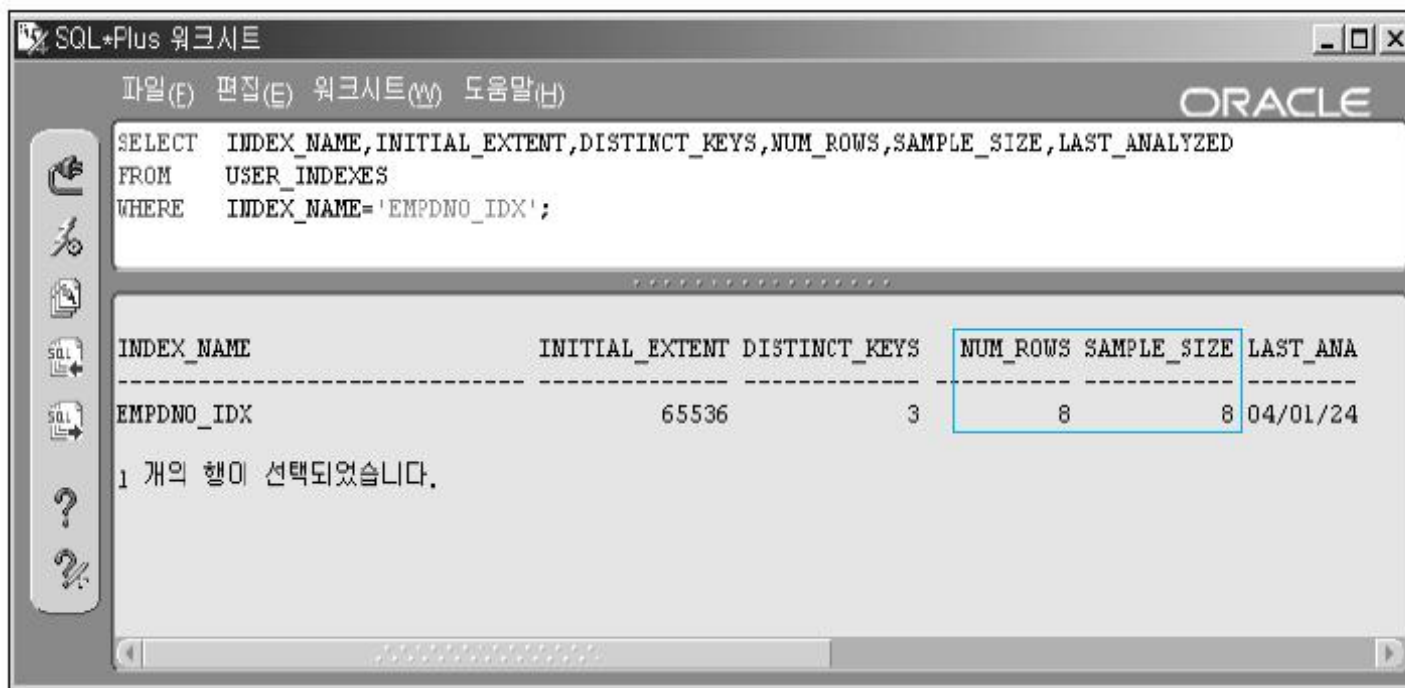
INDEX_NAME	INITIAL_EXTENT	DISTINCT_KEYS	NUM_ROWS	SAMPLE_SIZE	LAST_ANA
EMPDNO_IDX	65536	3	7	7	04/01/24

Below the table, it says '1 개의 행이 선택되었습니다.' (1 row selected).

[그림 8.13] 한 튜플을 삽입한 후의 인덱스 EMPDNO_IDX의 통계 정보

8.3 오라클의 시스템 카탈로그(계속)

- 새로 입력한 튜플이 통계 정보에 반영되도록 **ANALYZE** 명령을 수행한 후에 통계 정보를 확인



SQL*Plus 워크시트

파일(F) 편집(E) 워크시트(W) 도움말(H) ORACLE

```
SELECT INDEX_NAME, INITIAL_EXTENT, DISTINCT_KEYS, NUM_ROWS, SAMPLE_SIZE, LAST_ANALYZED
FROM   USER_INDEXES
WHERE  INDEX_NAME = 'EMPDNO_IDX';
```

INDEX_NAME	INITIAL_EXTENT	DISTINCT_KEYS	NUM_ROWS	SAMPLE_SIZE	LAST_ANA
EMPDNO_IDX	65536	3	8	8	04/01/24

1 개의 행이 선택되었습니다.

[그림 8.14] 한 튜플을 삽입한 후의 인덱스 EMPDNO_IDX의 갱신된 통계 정보

8.3 오라클의 시스템 카탈로그(계속)

- ❑ EMPLOYEE 테이블에 정의된 인덱스 정보를 찾기 위해서 아래와 같은 명령을 수행


```
SELECT *  
FROM USER_IND_COLUMNS  
WHERE TABLE_NAME = 'EMPLOYEE';
```

유저_인덱스_컬럼.

- ✓ INDEX_NAME은 인덱스의 이름, TABLE_NAME은 인덱스가 정의된 테이블의 이름, COLUMN_NAME은 인덱스가 정의된 애트리뷰트의 이름, COLUMN_POSITION은 인덱스가 정의된 애트리뷰트의 위치, COLUMN_LENGTH는 인덱스가 정의된 애트리뷰트의 길이, DESC는 정렬 방식(오름차순 또는 내림차순)을 나타냄

8.3 오라클의 시스템 카탈로그(계속)

- ✓ 3장의 예제 3.2에서 EMPLOYEE 테이블을 정의할 때 EMPNO 애트리뷰트를 기본 키로 선정했으므로 오라클이 자동적으로 인덱스를 생성
- ✓ EMPNAME 애트리뷰트에는 UNIQUE 키워드를 명시했으므로 오라클이 자동적으로 인덱스를 생성



INDEX_NAME	TABLE_NAME	COLUMN_NAME	COLUMN_POSITION	COLUMN_LENGTH	CHAR_LENGTH	DESC
SYS_C003010	EMPLOYEE	EMPNO	1	22	0	ASC
SYS_C003011	EMPLOYEE	EMPNAME	1	10	10	ASC
EMPDNO_IDX	EMPLOYEE	DNO	1	22	0	ASC

3 개의 행이 선택되었습니다.

[그림 8.15] EMPLOYEE 테이블의 인덱스 정보