

# 데이터베이스

- 순서 : Ch8(뷰와 시스템 카탈로그 )
- 학기 : 2018학년도 2학기
- 학과 : 가천대학교 컴퓨터공학과 2학년
- 교수 : 박양재

# 데이터베이스

- 목차

8.1 뷰

8.2 관계 DBMS의 시스템 카탈로그

8.3 오라클의 시스템 카탈로그

## 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

## 8.1 뷰

### □ 뷰의 개요

- ✓ ANSI/SPARC 3단계 아키텍처에서 외부 뷰는 특정 사용자가 보는 데이터베이스의 구조
- ✓ 관계 데이터베이스에서의 뷰는 한 사용자의 전체 외부 뷰 대신에 하나의 가상 릴레이션(virtual relation)을 의미(실제로 튜플을 갖지 않는다)
- ✓ 뷰는 기존의 기본 릴레이션(base relation, 실제 릴레이션)에 대한 SELECT문의 형태로 정의됨
- ✓ 사용자는 여러 개의 릴레이션과 뷰를 사용할 수 있음
- ✓ 사용자에게는 기본 릴레이션과 같은 방법으로 조작할 수 있는 정상적인 릴레이션으로 보인다.
- ✓ 뷰는 릴레이션으로부터 데이터를 검색하거나 갱신할 수 있는 동적인 창(dynamic window)의 역할-뷰 내용은 질의 할 때 마다 달라진다.

## 8.1 뷰(계속)

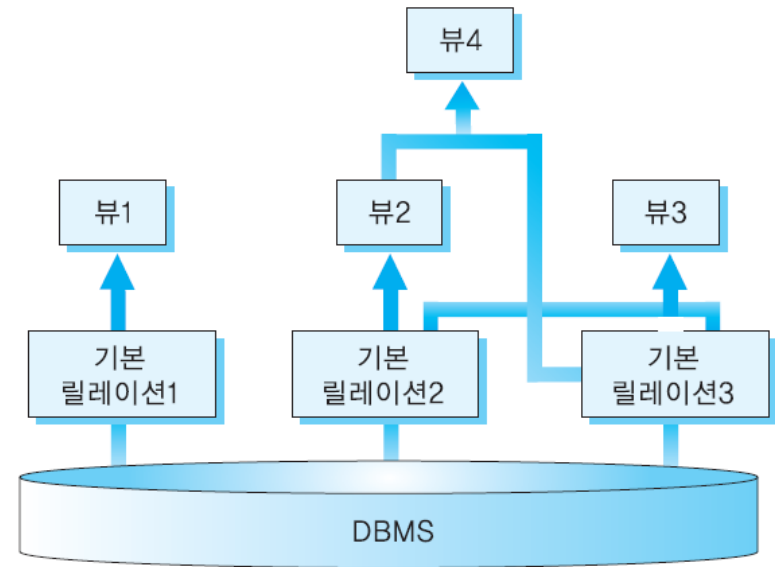
### □ 스냅샷(snapshot)

- ✓ 어느 시점에 SELECT문의 결과를 기본 릴레이션의 형태로 저장해 놓은 것  
일부 관계 DBMS들에서는 구체화된 뷰(materialized view)라 부른다.
- ✓ 스냅샷은 사진을 찍은 것과 같아서 스냅샷을 정의하는 시점의 기본 릴레이션의 내용이 스냅샷에 반영되므로 기본 릴레이션이 갱신되면 스냅샷은 반영하지 않으므로 주기적으로 refresh 해야 됨.
- ✓ 어떤 시점의 조직체의 현황, 예를 들어 몇년 몇월 시점에 근무하던 직원들의 정보, 재고 정보 등이 스냅샷으로 정의될 수 있음
- ✓ 뷰를 사용하면 여러 사용자가 여러 방식으로 데이터를 볼 수 있다.  
데이터베이스는 공유 자원이므로 저장된 데이터가 각 사용자에게 서로 다른 뷰로 제공하는 것은 유용하다.

## 8.1 뷰(계속)

항목	내용	비고
외부스키마 (External Schema)	<ul style="list-style-type: none"> <li>- 뷰 단계가 여러 개의 사용자 관점으로 구성. 즉 각 사용자가 보는 개인적인 DB 스키마</li> <li>- DB의 개별 사용자나 응용프로그램에 접근하는 DB를 정의</li> </ul>	사용자 관점으로 접근하는 특성에 따른 스키마 구성
개념스키마 (Conceptual Schema)	<ul style="list-style-type: none"> <li>- 개념 단계 하나마다 개념적 스키마로 구성. 모든 사용자 관점을 통합한 조직 전체의 DB를 기술하는 것</li> <li>- 모든 응용 시스템이나 사용자가 필요로 하는 데이터를 통합한 조직전체의 DB를 기술한 것 DB에 저장되는 데이터와 그들의 관계를 표현한 스키마</li> </ul>	통합관점
내부스키마 (Internal Schema)	<ul style="list-style-type: none"> <li>- 내부 단계와 내부 스키마로 구성. DB가 물리적으로 저장된 형식</li> <li>- 물리적 장치에서 데이터가 실제로 저장되는 방법을 표현한 스키마</li> </ul>	물리적 저장구조

〈표 1〉 데이터 독립성의 구성



〔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];
```

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

## 8.1 뷰(계속)

예: 한 릴레이션 위에서 뷰를 정의

그림 4.8의 EMPLOYEE 릴레이션에 대해서 “3번 부서에 근무하는 직원들의 직원번호, 직원이름, 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 ;
```

\*.EMP\_DNO3 뷰에는 실제로 튜플들이 저장되어 있지 않지만 EMP\_DNO3을 통해서 기본 릴레이션인 EMPLOYEE에 접근하면 파란색 음영 표시부분만 접근이 가능

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)

```
CREATE VIEW EMP_DN03 (ENO, ENAME, TITLE)
AS SELECT EMPNO, EMPNAME, TITLE
FROM EMPLOYEE
WHERE DNO=3;
```

```
SELECT *
FROM EMP_DN03;
```

CREATE 을 (를) 성공했습니다.

ENO	ENAME	TITLE
3011	이수민	부장
3427	최종철	사원

2 rows selected

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\_DN03을 통해서 접근할 수 있는 부분(파란색 음영)

## 8.1 뷰(계속)

예: 두 릴레이션 위에서 뷰를 정의

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';
```

## 8.1 뷰(계속)

```
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' ;
```

```
SELECT *
FROM EMP_PLANNING ;
```

EMPNAME	TITLE	SALARY
이성래	사장	5000000
조민희	과장	3000000
김창섭	대리	2500000

3 rows selected

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

DEPARTMENT


DEPTNO	DEPTNAME	FLOOR
1	영업	8
2	기획	10
3	개발	9
4	총무	7

## 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

```
SELECT *
FROM EMP_DNO3
WHERE TITLE='사원';
```



```
SELECT EMPNO, EMPNAME, TITLE
FROM EMPLOYEE
WHERE TITLE='사원'
AND DNO=3;
```

- ✓ 예: EMP\_DNO3뷰에 대해 SELECT문을 수행하면 기본 릴레이션 EMPLOYEE에 대한 질의로 변환되어 수행한다. **뷰의 SELECT절에 \* 명시했어도 기본 릴레이션에 대한 SELECT절에서는 뷰의 정의에 사용된 세 개 애트리뷰트만 열거된다.** EMP\_DNO3는 EMPLOYEE로 뷰의 WHERE절 TITLE='사원'만 명시했지만 기본 릴레이션 WHERE절은 TITLE='사원'과 뷰 정의시 사용된 조건인 DNO=3이 AND로 연결된다.

## 8.1 뷰(계속)

### □ 뷰 삭제

- ✓ 뷰를 생성한 후 뷰가 더 이상 필요없거나 뷰 생성자가 뷰 정의에서 참조된 기본 릴레이션에 대한 SELECT 권한을 잃으면 뷰를 삭제 한다.
- ✓ 뷰가 삭제 될 때 이 뷰가 기반으로 하는 기본 릴레이션은 아무런 영향을 받지 않는다.

DROP VIEW 뷰이름 ;

- ✓ 뷰의 생성자 또는 적절한 권한을 가진 사용자만 뷰를 제거 할 수 있다.
- ✓ 삭제된 뷰를 기반으로 하는 뷰나 기타 응용들은 무효화된다.

## 8.1 뷰(계속)

### □ 뷰의 장점

✓ 뷰는 복잡한 질의를 간단하게 표현할 수 있게 함

- 기획부에 근무하는 사원들 중에서 직책이 **부장**인 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
        AND E.TITLE = '부장';
```

- 뷰에 대해서 같은 결과를 검색하는 질의를 표현하면

```
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' ;
```

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

## 8.1 뷰(계속)

□ 뷰의 장점(계속) -WITH CHECK OPTION 명시 후 UPDATE 해야 한다.

✓ 뷰는 데이터 무결성을 보장하는데 활용됨

- 기본적으로 뷰를 통해 튜플을 추가하거나 수정할 때 튜플이 뷰를 정의하는 SELECT문의 WHERE절의 기준에 맞지 않으면 뷰의 내용에서 사라짐
- 뷰를 만들 때, 3번 부서에 근무하는 직원들의 사원번호, 사원이름, 직책으로 이루어진 뷰 EMP\_DNO3 정의하였으므로 DNO=2로 수정 되면 뷰에서 사라진다.

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

- 이 뷰의 정의할 때 WITH CHECK OPTION을 명시했다고 가정

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

```
WITH CHECK OPTION;
```

- 뷰가 선택할 수 없는 튜플들을 생성할 수 없도록 보장한다.(삽입 또는 수정 되는 데이터의 무결성 제약조건과 데이터 유효성 검사가 시행되도록 하는 옵션

## 8.1 뷰(계속)

□ 뷰의 장점(계속) -WITH CHECK OPTION 명시 후 UPDATE 해야 한다.

✓ 실습 1 : 이 뷰의 정의할 때 WITH CHECK OPTION을 명시

```
CREATE VIEW EMP_DN03 (ENO, ENAME, TITLE)
AS SELECT EMPNO, EMPNAME, TITLE
FROM EMPLOYEE
WHERE DNO =3
WITH CHECK OPTION ;
```

CREATE VIEW을(를) 성공했습니다.

✓ 실습 2 : 뷰 UPDATE 실시

```
UPDATE EMP_DN03
SET DNO = 2
WHERE ENO = 3427 ;
```

명령의 1 행에서 시작하는 중 오류 발생:

```
UPDATE EMP_DN03
SET DNO = 2
WHERE ENO = 3427
```

오류 발생 명령행: 2, 열: 7

오류 보고:

SQL 오류: ORA-00904: "DNO": 부적합한 식별자  
00904. 00000 - "%s: invalid identifier"

\*Cause:

\*Action:

- 뷰가 선택할 수 없는 튜플들을 생성할 수 없도록 보장한다.(삽입 또는 수정 되는 데이터의 무결성 제약조건과 데이터 유효성 검사가 시행되도록 하는 옵션



## 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)

```
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;
```

- ✓ 응용들이 뷰를 다루도록 하면 응용에 영향을 주지 않으면서 기본 릴레이션들의 스키마를 변경 할 수 있다.
- ✓ 실제의 데이터베이스 응용들에서 뷰가 많이 활용되는 이유는 보안 및 데이터 독립성을 유지하기 위해서이다.

## 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

- ✓ 동일한 데이터에 대한 여러 가지 뷰를 제공함

- 뷰는 사용자들의 그룹이 각자 특정한 기준에 따라 데이터를 접근하도록 함

## 8.1 뷰(계속)

### □ 뷰의 단점

- ✓ 모든 뷰는 갱신이 가능하지 않다.
- ✓ 기본 릴레이션에 접근하는 것보다 성능이 약간 저하 될 수 있다. 이유는 뷰를 참조하는 질의는 시스템 카탈로그에서 뷰의 정의를 가져온 후 기본 릴레이션에 대한 질의로 변환되어 수행되므로 기본 릴레이션에 대한 질의보다 디스크 접근 횟수가 많을 수 있다.

## 8.1 뷰(계속)

### □ 뷰의 갱신

- ✓ 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', , , );
```

## 8.1 뷰(계속)

### □ 뷰의 갱신

- ✓ 실습 : EMP\_DNO3 뷰를 통해서 3개 애트리뷰트에만 값을 명시할 수 있다.(EMPNO, EMPNAME, TITLE), MANAGER, SALARY, DNO에는 값이 명시 되지 않았다.

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

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

MANAGER, SALARY는 널 값, DNO는  
릴레이션 정의시 디폴트 값으로 1로 지정)

```
SELECT *
FROM EMPLOYEE ;
```

EMPNO	EMPNAME	TITLE	MANAGER	SALARY	DNO
4293	김정수	사원			1
4377	이성래	사장		5000000	2
3426	박영권	과장	4377	3000000	1
3011	이수민	부장	4377	4000000	3
3427	최종철	사원	3011	1500000	3
1003	조민희	과장	4377	3000000	2
2106	김향섭	대리	1003	2500000	2
1365	김상원	사원	3426	1500000	1

## 8.1 뷰(계속)

### □ 뷰의 갱신(계속)

- ✓ 갱신 2: 두 개의 릴레이션 위에서 정의된 뷰에 대한 갱신

```
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' ;
```

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

INSERT INTO EMP\_PLANNING  
VALUES ('박지선', '대리', 2500000);

→

INSERT INTO EMPLOYEE  
VALUES ( , '박지선', '대리')

- ✓ 대부분의 관계DBMS에서는 기본 키값이 열인(EMPNO)투플은 엔티티 무결성 제약조건에 의해 삽입을 거절한다.
- ✓ EMP\_PLANNING에 대한 INSERT문은 EMPLOYEE에 대한 INSERT문으로 변환되어 수행되는데 뷰를 통해서 EMPLOYEE 릴레이션의 기본 키에 해당하는 EMPNO의 값을 입력할 수 없으므로 EMPNO가 널 값을 갖게 되고, 따라서 삽입이 거절된다.

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

명령의 1 행에서 시작하는 중 오류 발생:

```
INSERT INTO EMP_PLANNING
VALUES('박지선', '대리', 2500000)
```

오류 보고:

SQL 오류: ORA-01400: NULL을 ("KIM"."EMPLOYEE"."EMPNO") 안에 삽입할 수 없습니다  
01400. 00000 - "cannot insert NULL into (%s)"

## 8.1 뷰(계속)

### □ 뷰의 갱신(계속)

- ✓ 갱신 3 : 집단 함수 등을 포함한 뷰에 대한 갱신
- ✓ 각 부서별로 소속 직원들의 평균급여를 검색하라. 이 뷰에는 집단함수의 결과(AVG(SALARY))가 애트리뷰트에 포함되므로 뷰의 이름 다음에 반드시 뷰의 애트리뷰트들의 이름을 명시해야 한다.

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

CREATE 을(를) 성공했습니다.

- ✓ 이 뷰는 평균 급여 값을 포함하고 있기 때문에 갱신할 수 없다. 즉, 이 뷰를 갱신하는 연산은 이 뷰에서 참조된 EMPLOYEE 릴레이션에 대한 갱신으로 변환될 수 없다.
- ✓ 예: 2번 부서의 평균급여를 3,000,000원으로 수정하라(DBMS가 거절한다)

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

명령의 1 행에서 시작하는 중 오류 발생:

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

오류 발생 명령행: 1, 열: 8

오류 보고:

SQL 오류: ORA-01732: 뷰에 대한 데이터 조작이 부적합합니다

01732. 00000 - "data manipulation operation not legal on this view"



## 8.1 뷰(계속)

### □ 뷰의 갱신(계속)

- ✓ 갱신 3: 집단 함수 등을 포함한 뷰에 대한 갱신
- ✓ 이유 : 2번 부서에 속한 사원이 여러 명 있는데, 2번 부서에 속한 사원들의 평균급여를 3,000,000원으로 하려면 어떻게 수정해야할 지 결정할 수 없으므로 거절한다.
- ✓ 3번 부서이면서 평균급여가 3,200,000원 이라는 튜플을 새로 삽입하라.

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

- ✓ 결론 : EMPLOYEE 릴레이션의 튜플들에 어떻게 반영해야 할지 결정할 수 없으므로 DBMS가 거절한다.

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

명령의 1 행에서 시작하는 중 오류 발생:

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

오류 발생 명령행: 1, 열: 0

오류 보고:

SQL 오류: ORA-01733: 가상 열은 사용할 수 없습니다

01733. 00000 - "virtual column not allowed here"

## 8.1 뷰(계속)

### □ 갱신이 불가능한 뷰

- ✓ 어떤 뷰는 갱신이 가능하고, 어떤 뷰는 갱신이 불가능하다.
- ✓ 한 릴레이션 위에서 정의되었으나 **그 릴레이션의 기본 키가 포함되지 않은 뷰**
  - 이 뷰에 튜플을 삽입하면 기본 릴레이션의 INSERT문으로 변환되는데 뷰에 릴레이션의 기본 키가 포함되어 있지 않았으므로 **뷰에 대한 INSERT문에서 기본 키 값을 정의하는 것은 불가능하다.**
  - 릴레이션의 어떤 튜플에 **기본 키의 값이 없으면 엔티티 무결성 제약조건을 위배하기 때문에 그 튜플의 삽입이 거절된다.**
- ✓ 기본 릴레이션의 애트리뷰트들 중에서 **뷰에 포함되지 않은 애트리뷰트에 대해 NOT NULL이 지정되어 있을 때**
  - 뷰를 통해서 기본 릴레이션에 튜플을 삽입할 때, 뷰에 포함되지 않은 애트리뷰트에 대해서 값을 입력하는 것은 불가능하다. **뷰에 대한 INSERT문에 포함되지 않은 애트리뷰트들에 값을 입력하는 것은 불가능하고 널값이 입력된다.** 따라서 튜플삽입이 거절된다.

## 8.1 뷰(계속)

### □ 갱신이 불가능한 뷰

#### ✓ 집단함수가 포함된 뷰

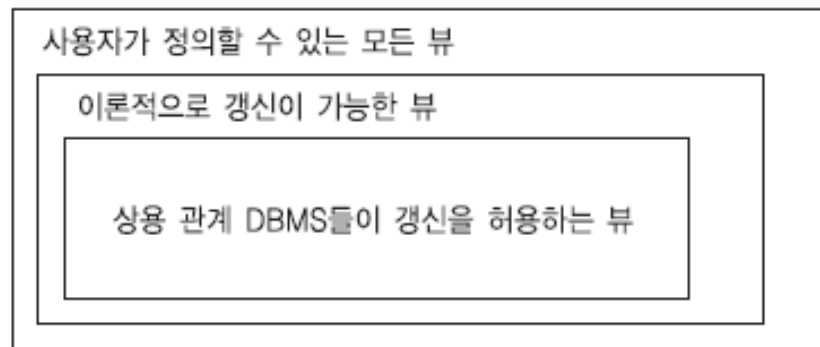
- 뷰의 정의에 SUM, AVG, MAX, MIN, COUNT, DISTINCT, GROUP BY, HAVING등의 구성요소가 하나라도 포함되면 그 뷰는 본질적으로 갱신이 불가능하다.
- 이유 : 뷰에 대한 갱신을 어떻게 기본 릴레이션에 대한 갱신으로 변환해야 할지 결정할 수 없기 때문이다.

#### ✓ 조인으로 정의된 뷰

- 많은 DBMS에서 두 개 이상의 릴레이션을 사용하여 정의된 뷰의 갱신을 허용하지 않는다. 뷰에 대한 갱신이 기본 릴레이션으로 정확하게 대응되지 않을 수 있기 때문이다.

## 8.1 뷰(계속)

### □갱신 가능성 기준에 따른 뷰들의 유형



[그림 8.3] 갱신 가능성 기준에 따른 뷰들의 유형

## 8.2 관계 DBMS의 시스템 카탈로그

### □ 시스템 카탈로그

- ✓ DBMS는 사용자가 저장한 항목의 특성을 알아볼 수 있는 기능을 제공해야 한다.
- ✓ 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 테이블 이라고도 부름

## 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;
```

- ① SELECT문이 **문법적으로 정확한가를 검사함**
- ② SELECT문에서 참조하는 EMPLOYEE 릴레이션이 **데이터베이스에 존재하는가를 검사함**
- ③ EMPLOYEE 릴레이션에 **SELECT절에 열거된 애트리뷰트와 WHERE절에서 조건에 사용된 애트리뷰트가 존재하는가를 확인함**
- ④ 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)인가 등을 검사함**

## 8.2 관계 DBMS의 시스템 카탈로그(계속)

□ 시스템 카탈로그가 질의 처리에 어떻게 활용되는가(계속)

- ⑤ 이 질의를 입력한 사용자가 EMPLOYEE 릴레이션의 EMPNAME, SALARY 애트리뷰트를 검색할 수 있는 권한이 있는가를 확인함
- ⑥ TITLE 애트리뷰트와 DNO 애트리뷰트에 인덱스가 정의되어 있는지 확인함
- ⑦ 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(인덱스의 선택율이 높다)
- ⑧ 한 릴레이션의 전체 튜플 수와 그 릴레이션에 정의된 각 인덱스에 존재하는 상이한 값들의 개수를 유지한다면 어느 인덱스를 사용하는 것이 유리한가를 예상할 수 있음

## 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, 사장의 다섯 가지 값들이 존재한다. 그에 반해 DNO 애트리뷰트에는 1, 2, 3의 세 가지 값들이 존재함

EMPLOYEE	<u>EMPNO</u>	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

- ✓ 따라서 TITLE 애트리뷰트에 정의된 인덱스가 DNO에 정의된 인덱스보다 대상 튜플들을 더 좁혀 주므로 유리함
- ✓ 이 처럼 DBMS가 질의를 수행하는 여러가지 방법들 중에서 가장 비용이 적게 드는 방법을 찾는 과정을 질의 최적화(query optimizaton)라 한다.



## 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 모듈 – DBMS의 성능에 매우 중요한 역할 담당
- ✓ 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)

### ❑ 질의 최적화

- ✓ 예:워드프로세서로 문서 작업 시, 여러 개의 문서를 작업하는 사용자는 원하는 문서이름을 정확하게 기억하지 못할 수 있다. 이런 경우에 **사용자는 파일찾기를 이용하거나 원하는 문서가 들어 있을 만한 폴더를 열어본다.**
- ✓ 데이터베이스의 사용자는 릴레이션 이름, 애트리뷰트 이름, 애트리뷰트의 데이터 타입 등 워드프로세서보다 기억할 것이 더 많이 존재하고 이들을 항상 암기 할 수 없다.
- ✓ **사용자는 시스템 카타로그를 통해서 자신이 원하는 릴레이션의 이름, 애트리뷰트들에 관한 정보를 쉽게 찾을 수 있으므로 시스템 카타로그는 사용자를 위한 목적으로 활용된다.**

## 8.2 관계 DBMS의 시스템 카탈로그(계속)

### ❑ 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
- ✓ 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)

### □관계 DBMS의 시스템 카탈로그

✓ SYS\_RELATION 릴레이션 : 데이터베이스 내의 각 릴레이션마다 하나의 튜플로 그 릴레이션에 관한 정보를 나타낸다.

- RelId:릴레이션의 이름 또는 식별자, RelOwner:릴레이션을 생성한 사용자의 식별자, RelTups:릴레이션의 튜플 수, RelAtts:릴레이션의 애트리뷰트 수, RelWidth:튜플의 길이

SYS_RELATION	RelId	RelOwner	RelTups	RelAtts	RelWidth
	EMPLOYEE	KIM	7	6	36
	DEPARTMENT	KIM	4	3	18
	...	...	...	...	...

- EMPLOYEE 릴레이션에 7개의 튜플이 존재를 표시하나 튜플이 삽입된다고 해도 RelTups 개수가 바로 갱신되지 않는다. 시스템 카탈로그는 DBMS가 질의 할 때 마다 접근하기 때문에 동시 접근 수가 사용자 릴레이션보다 많다.
- 삽입 시 바로 갱신되면 동시성 오버헤드가 매우 크고 자원소비가 많아 성능저하 원인

## 8.2 관계 DBMS의 시스템 카탈로그(계속)

### □관계 DBMS의 시스템 카탈로그

- ✓ SYS\_ATTRIBUTE 릴레이션은 각 릴레이션의 각 애트리뷰트마다 하나의 튜플로 그 애트리뷰트에 관한 정보를 나타낸다.
- AttrelId : 애트리뷰트가 속한 릴레이션의 이름 또는 식별자, AttID:애트리뷰트의 번호, AttName:애트리뷰트의 이름, AttOff:튜플내에서 애트리뷰트의 오프셋(off-set), AttType:애트리뷰트의 데이터 타입, AttLen:애트리뷰트의 길이, Pk or Fk:주키 또는 외래키

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	
	...	...	...	...	...	...	

## 8.2 관계 DBMS의 시스템 카탈로그(계속)

### ❑ 시스템 카탈로그의 갱신

- ✓ 어떤 사용자도 시스템 카탈로그를 직접 갱신할 수 없음
- ✓ 즉 DELETE, UPDATE 또는 INSERT문을 사용하여 시스템 카탈로그를 변경할 수 없음
- ✓ EMPLOYEE 릴레이션의 소유자인 KIM이 EMPLOYEE 릴레이션에서 MANAGER 애트리뷰트를 삭제하기 위해서

```
ALTER TABLE EMPLOYEE DROP COLUMN MANAGER;
```

- ✓ 라고 하는 대신에 아래와 같이 시스템 카탈로그에 대해 DELETE문을 사용하면 DBMS가 거절함

```
DELETE FROM SYS_ATTRIBUTE  
WHERE AttRelId = 'EMPLOYEE' AND AttName = 'MANAGER';
```

## 8.2 관계 DBMS의 시스템 카탈로그(계속)

### □ 시스템 카탈로그에 유지되는 통계 정보

- ✓ 질의 최적화 모듈은 관계 DBMS의 핵심적인 구성요소로 비용 기반으로 하는 질의 최적화 모듈은 다양한 통계정보를 제공하여 질의결과를 추정하는데 사용
- ✓ 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)

## 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



## 8.3 오라클의 시스템 카탈로그

### ❑ 오라클의 시스템 카탈로그

- ✓ 오라클 데이터베이스 내의 테이블들은 사용자 테이블과 데이터 사전으로 구분
- ✓ 사용자 테이블은 사용자가 생성하고 유지하는 테이블
- ✓ 데이터 사전은 오라클 서버가 생성하고 유지하는 테이블들의 모임
- ✓ 시스템 카탈로그를 데이터 사전(data dictionary)이라고 부름
- ✓ 데이터 사전은 시스템 테이블스페이스에 저장됨
- ✓ 데이터 사전은 DDL 명령이 수행될 때마다와 테이블의 튜플 수를 갱신하는 DML명령들에도 오라클 서버가 데이터 사전을 갱신한다.
- ✓ 데이터 사전은 오라클 데이터베이스의 핵심이며, 사용자(최종 사용자, 응용설계자, 데이터베이스 관리자)들에게 중요한 정보의 원천

## 8.3 오라클의 시스템 카탈로그

### ❑ 오라클의 시스템 카탈로그

- ✓ 데이터 사전은 기본 테이블과 데이터 사전 뷰로 구성됨
- ✓ 기본 테이블은 데이터베이스에 대한 설명을 포함하며, CREATE DATABASE 명령 수행 시 자동으로 생성(가장 먼저 생성되는 객체)되며, 오직 오라클 서버만 기본 테이블을 갱신한다.
- ✓ 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 사전에서 사용자가 직접 정보를 갱신하는 것을 지원하지 않으며, 접근하려면 SELECT문을 사용해야 한다.
- ✓ 오라클 설치시 [데이터베이스 생성 및 구성] 선택하였으므로 catalog.sql 스크립트가 수행되어 데이터 사전 뷰가 생성된다.

## 8.3 오라클의 시스템 카탈로그(계속)

### ❑ 데이터 사전 뷰의 세 부류

- 데이터 사전에는 데이터베이스의 모든 객체(테이블, 뷰, 인덱스 등)들에 관한 정의, 객체에 공간이 얼마나 할당되었으며 현재 얼마나 사용 중인가, 애트리뷰트 값, 무결성 제약조건, 사용자들의 이름, 각 사용자가 허가 받은 권한과 역할 등이 포함

#### ✓ DBA\_xxx 뷰

전체 데이터베이스에 관한 전역적인 정보 표시하며, 주로 DBA가 질의 용도로 사용하며, SELECT ANY TABLE 시스템 권한을 허가 받은 사용자가 질의할 수 있다.

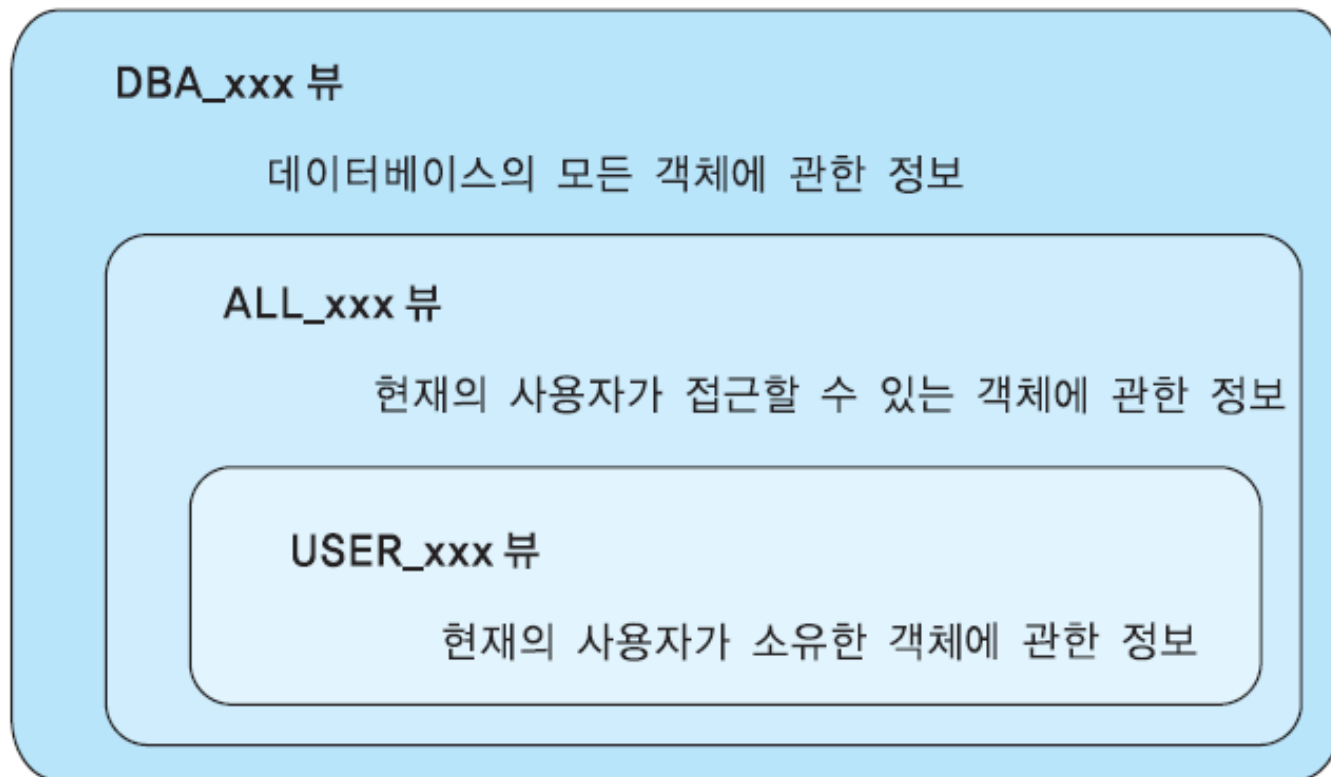
#### ✓ ALL\_xxx 뷰

현재의 사용자가 접근할 수 있는 객체들에 관한 정보 를 표시하며, 어떤 사용자의 데이터베이스에 관한 전체적인 관점을 보여준다. Public 또는 권한이나 역할을 명시적으로 허가 받아서 접근할 수 있는 객체와 이 사용자가 소유한 객체들에 관한 정보 표시

#### ✓ 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_INDEXES	현재 로그인한 사용자가 소유한 인덱스들에 관한 정보
USER_IND_COLUMNS	현재 로그인한 사용자가 소유한 인덱스 및 테이블상의 인덱스에 포함된 애트리뷰트들에 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)

### ❑ Oracle SQL Developer를 사용하여 데이터 사전 뷰를 검색하는 실습

- 사용자 KIM이 소유한 테이블이나 뷰에 관한 정보를 검색하기 위해서 KIM으로 Oracle SQL Developer에 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 (앞 실습에서 생성된 뷰)

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

OWNER	TABLE_NAME	TABLE_TYPE
-----		
KIM	DEPARTMENT	TABLE
KIM	EMPLOYEE	TABLE
KIM	EMP_AVGSAL	VIEW
KIM	EMP_DN03	VIEW
KIM	EMP_PLANNING	VIEW
KIM	PROJECT	TABLE
6 rows selected		

## 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
- ✓ 이 밖에도 USER\_TAB\_COLUMNS 뷰를 통해서 각 애트리뷰트의 길이, 널값 허용 여부, 디폴트값 등을 검색할 수 있음

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

TABLE_NAME	COLUMN_NAME	DATA_TYPE
EMPLOYEE	EMPNO	NUMBER
EMPLOYEE	EMPNAME	CHAR
EMPLOYEE	TITLE	CHAR
EMPLOYEE	MANAGER	NUMBER
EMPLOYEE	SALARY	NUMBER
EMPLOYEE	DNO	NUMBER

6 rows selected

## 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;
```

- ✓ VIEW\_NAME은 뷰의 이름이고, TEXT는 뷰를 정의한 SQL문

SELECT VIEW_NAME, TEXT FROM USER_VIEWS ;	VIEW_NAME	TEXT
	-----	-----
	EMP_AVGSAL	SELECT DNO, AVG(SALARY) FROM employee GROUP BY DNO
	EMP_DN03	SELECT EMPNO, EMPNAME, TITLE FROM employee WHERE DNO =3
	EMP_PLANNING	SELECT E.EMPNAME, E.TITLE, E.SALARY FROM EMPLOYEE E, DEPARTMENT D WHERE E.DNO=D.DEPTNO AND D.DEPTNAME='기획'
	3 rows selected	



## 8.3 오라클의 시스템 카탈로그(계속)

- ❑ 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

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

명령의 1 행에서 시작하는 중 오류 발생:

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

오류 보고:

```
SQL 오류: ORA-01400: NULL을 ("KIM"."EMPLOYEE"."EMPNO") 안에 삽입할 수 없습니다  
01400. 00000 - "cannot insert NULL into (%s)"
```

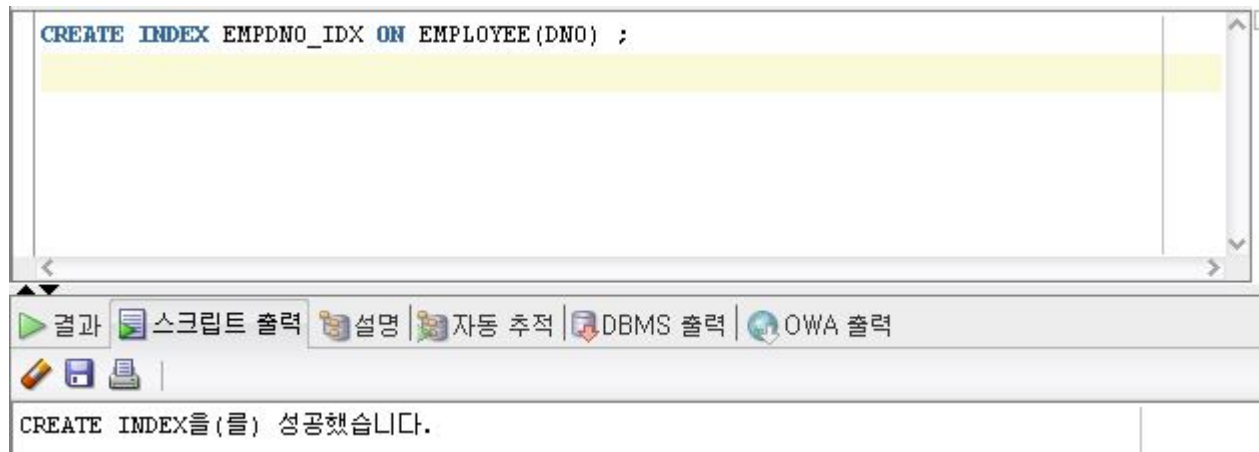
\*Cause:

\*Action:

## 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) ;
```



## 8.3 오라클의 시스템 카탈로그(계속)

- 통계 정보를 확인하기 위해서 USER\_INDEXES 데이터 사전 뷰에 대해 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 .

```
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_ANALYZED
EMPDNO_IDX	65536	3	8	8	15/07/28

1 rows selected

## 8.3 오라클의 시스템 카탈로그(계속)

- ❑ EMPLOYEE 테이블에 새로운 튜플을 한 개 삽입해보자

```
INSERT INTO EMPLOYEE  
VALUES (3428, '김지민', '사원', 2106, 1500000, 2) ;
```

```
INSERT INTO employee  
VALUES (3428, '김지민', '사원', 2106, 1500000, 2) ;|
```

1 행 삽입됨

- ❑ 다시 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' ;
```

## 8.3 오라클의 시스템 카탈로그(계속)

❑ 다시 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 = 'EMPENO_IDX' ;
```

INDEX_NAME	INITIAL_EXTENT	DISTINCT_KEYS	NUM_ROWS	SAMPLE_SIZE	LAST_ANALYZED
EMPENO_IDX	65536	3	8	8	15/07/28

1 rows selected

- ✓ 그림 8.12는 그림 8.10과 동일
- ✓ 한 테이블에 튜플이 삽입되자마자 데이터 사전 뷰의 정보가 갱신되지는 않음

## 8.3 오라클의 시스템 카탈로그(계속)

□ 통계 정보는 ANALYZE문을 사용하여 갱신할 수 있음

ANALYZE 객체\_유형 객체\_이름 연산 STATISTICS ;

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

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

ANALYZE INDEX EMPDNO\_IDX을(를) 성공했습니다.

- 객체유형은 테이블, 인덱스를 나타내며, 연산에 **COMPUTE**가 사용되면 전체 테이블을 접근하여 통계정보를 계산한다. 이 옵션은 정확한 통계를 얻을 수 있지만 처리속도가 느리며, 연산에 **ESTIMATE**(판단치)가 사용되면 데이터 표본을 추출하여 통계정보를 구한다. 따라서 이 옵션은 정확한 정보는 아니지만 처리속도가 **COMPUTE**보다 매우 빠르다. 연산에 **DELETE**가 사용되면 테이블의 모든 통계정보를 삭제한다.
- 질의 최적화 모듈이 정확하게 동작하려면 테이블과 인덱스에 관한 통계정보를 현재 상태로 유지하므로, 주기적으로 **ANALYZE** 작업을 수행해야 한다.
- ✓ 테이블에 대한 통계 정보는 **ANALYZE TABLE** 명령을 사용해서 갱신하고
- ✓ 인덱스에 대한 통계 정보는 **ANALYZE INDEX** 명령을 사용해서 갱신

## 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';
```

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

ANALYZE INDEX EMPDNO_IDX를(를) 성공했습니다.					
INDEX_NAME	INITIAL_EXTENT	DISTINCT_KEYS	NUM_ROWS	SAMPLE_SIZE	LAST_ANALYZED
EMPDNO_IDX	65536	3	9	9	15/07/28
1 rows selected					

- **ANALYZE 명령 수행 후 EMPDNO\_IDX의 통계정보를 확인한 결과 NUM\_ROWS와 SAMPLE\_SIZE가 1씩 증가했음을 알 수 있으며, 방금 INSERT문 수행을 통해 삽입된 튜플에 대한 인덱스 정보가 반영되었음을 알 수 있다.**

## 8.3 오라클의 시스템 카탈로그(계속)

❑ 테이블에 정의된 **인덱스**에 관한 정보를 검색하려면 USER\_IND\_COLUMN 데이터 사전 뷰를 접근해야 한다.

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

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

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



## 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
- ✓ DNO 애트리뷰트에는 사용자가 CREATE INDEX문을 사용하여 명시적으로 인덱스를 정의
- ✓ 결과 화면 우측 끝에 표시

INDEX_NAME	TABLE_NAME	COLUMN_NAME	COLUMN_POSITION	COLUMN_LENGTH	CHAR_LENGTH	DESCEND
SYS_C0011067	EMPLOYEE	EMPNO	1	22	0	ASC
SYS_C0011068	EMPLOYEE	EMPNAME	1	10	10	ASC
EMPDNO_IDX	EMPLOYEE	DNO	1	22	0	ASC

3 rows selected

## 8.3 오라클의 시스템 카탈로그(계속)

- ✓ 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
- ✓ DESCEND는 정렬 방식(오름차순(ASCEND) 또는 내림차순(DESCEND)을 나타냄