10장 기타 스키마 객체 생성

목적

- 뷰 설명
- 뷰 생성
- 뷰를 통한 데이터 검색
- 뷰의 정의 변경
- 뷰를 통한 데이터 삽입, 갱신, 삭제
- 뷰 삭제

데이터베이스 객체

객체	설명
Table	행과 열로 구성된 기본적인 저장 매체의 단위
View	하나 이상의 테이블로부터 데이터의 부분집합을 논리적으로 표현
Sequence	기본 키 값을 발생
Index	어떤 질의의 성능을 향상
Synonym	객체에 대체 이름을 부여

뷰(View) 란?

- 테이블 또는 다른 뷰를 기초로 하는 논리적 테이블
- 뷰는 그자체로서 소유하는 데이터는 없지만 창문처럼 창문을 통해 어떤 데이터를 보거나 변경할 수 있다.
- 물리적으로 존재하는 릴레이션은 아니며 질의와 같은 수식에 의해 정의된 것
- 그러나 물리적으로 존재하는 일반 릴레이션과 유사하게 뷰에 대해 질의 할 수 있으며 어떤 경우에는 뷰 자체를 변경할 수 있다.

뷰의 사용 목적은?

- **DB** 선택적인 부분을 디스플레이 할 수 있기 때문에 데이터베이스에 대한 엑세스를 제한 할 수 있다.
- 복잡한 질의로부터 결과를 검색하기 위한 단순한 질의를 만든다.
- 사용자와 어플리케이션 프로그램에 대한 데이터 독립성을 제공한다.
- 하나의 뷰는 여러 개의 테이블로부터 데이터를 검색하는 데 사용될 수 있다.

단순 뷰와 복합 뷰

특징	단순 뷰	복합 뷰
테이블 수	하나	하나 이상
함수 포함	없음	있음
데이터그룹 포함	없음	있음
뷰를 통한 DML	있음	항상 가능한 것은 아님

뷰 생성

- **CREATE VIEW** 문장 내에서 서브쿼리를 내장한다.

```
CREATE [OR REPLACE] [FORCE | NOFORCE] VIEW view [(alias[, alias]….)]
AS subquery
[WITH CHECK OPTION [CONSTRAINT constraint] ]
[WITH READ ONLY]
```

- 서브쿼리는 복합 **SELECT** 구문을 포함할 수 있다.
- 서브쿼리는 **ORDER BY**절을 포함할 수 없다.

뷰 생성

- 부서 **10**의 종업원의 세부사항을 포함하는 **EMPVU10**뷰를 생성

```
SQL> CREATE  VIEW     empvu10
2       AS  SELECT          empno, ename, job
3       FROM                emp
4       WHERE               deptno = 10;
```

- **SQL * Plus DESCRIBE** 명령어를 사용하여 뷰의 구조를 기술

```
SQL>  DESCRIBE   empvu10
```

뷰 생성

- 서브 쿼리에서 열 별칭을 사용하여 뷰를 생성

```
SQL> CREATE  VIEW        salvu30
2       AS  SELECT       empno  EMPLOYEE_NUMBER,  ename  NAME
3                        sal  SALARY
4       FROM             emp
5       WHERE            deptno – 30;
```

- 주어진 별칭 이름으로 이 뷰에서 열을 선택

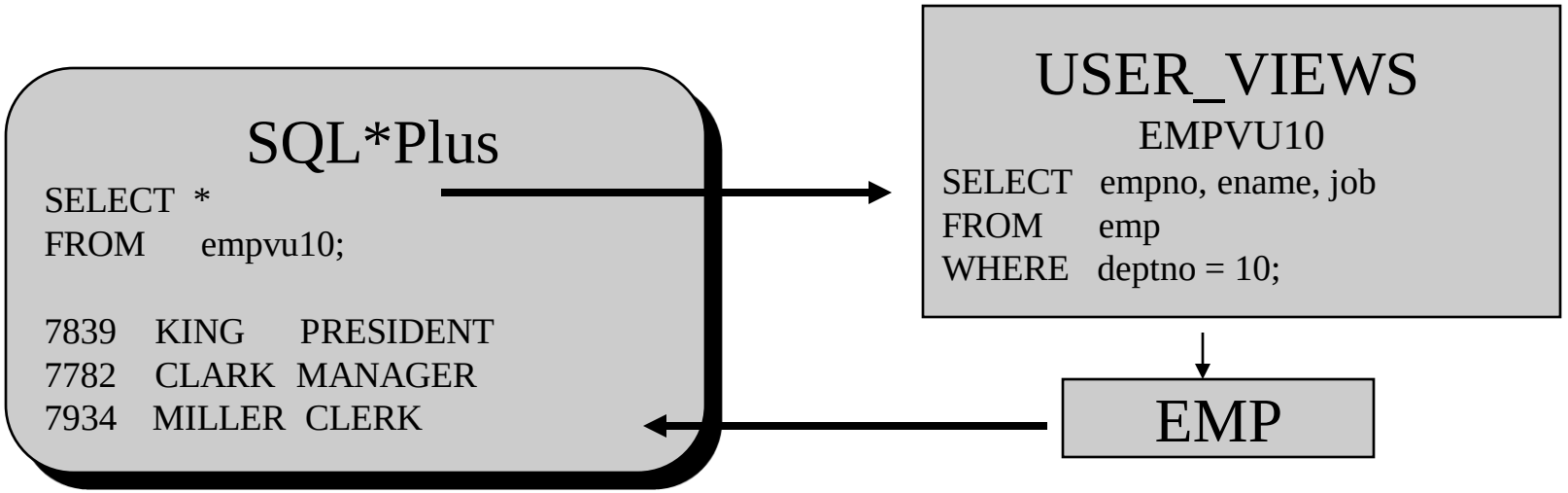
뷰 에서 데이터 검색

```
SQL>   SELECT  *
2          FROM      salvu30;
```

EMPLOYEE_NEMBER	NAME	SALARY
7698	BLAKE	2850
7654	MARTIN	1250
7499	ALLEN	1600
7844	TURNER	1500
7900	JAMES	950
7521	WARD	1250

-> 임의의 테이블에서 할 수 있는 것처럼 뷰로부터 데이터를 검색할 수 있다.
전체 뷰의 내용을 디스플레이 하거나 단지 특정 행과 열만을 볼 수 있다.

뷰 질의



-> 일단 뷰가 생성되면 뷰의 이름과 뷰 정의를 보기 위해 USER_VIEWS라는 데이터 사전 테이블을 질의할 수 있다. 뷰를 만드는 SELECT 문장의 텍스트는 LONG열에 저장된다.

뷰 수정

- **CREATE OR REPLACE VIEW** 절을 사용하여 **EMPVU10** 뷰를 수정한다.

```
SQL> CREATE  OR REPLACE VIEW empvu10
                (employee_number, employee_name, job_title)
2       AS  SELECT          empno, ename, job
3       FROM                emp
4       WHERE               deptno = 10;
```

- **CREATE VIEW** 절에서 열 별칭을 지정할 때 별칭은 서브쿼리의 열과 동일한 명령으로 나열된다.

복합 뷰 생성

- 아래는 부서명, 최소 급여, 최대 급여, 부서의 평균 급여의 복잡한 뷰를 생성한다. 다른 이름이 뷰에 대해 지정됐음을 명심하라. 뷰의 어떤 열이 함수나 표현식에서 유래되었다면 별칭은 필수적이다.

```
SQL> CREATE VIEW    dept_sum_vu
2                   (name, minsal, maxsal, avgsal)
3       AS    SELECT      d.dname, MIN(e.sal), MAX(e.sal),
                          AVG(e.sal)
4       FROM              emp  e,  dept   d
5       WHERE             e.deptno = d.deptno
6       GROUP  BY         d.dname;
```

뷰에서 DML 연산 수행 규칙

- 단순 뷰에서 **DML** 연산을 수행할 수 있다.
- 뷰가 다음을 포함한다면 행을 제거할 수 없다.
  - 그룹 함수
  - **GROUP BY** 절
  - **DISTINCT** 키워드
- 뷰가 다음을 포함한다면 뷰에서 데이터를 수정할 수 없다.
  - 이전 슬라이드에서 언급된 임의의 조건
  - 표현식으로 정의된 열
  - **ROWNUM** 의사열
- 다음을 포함 한다면 뷰에 데이터를 추가할 수 없다.
  - 뷰가 이전 슬라이드 또는 위에 언급된 임의의 조건을 포함할 때
  - 뷰에 의해 선택되지 않은 **NOT NULL** 열이 기본 테이블에 있을 때

WITH CHECK OPTION절 사용

- 뷰에 대한 **DML** 연산이 뷰의 조건을 만족할 때만 수행되도록 한다.

```
SQL> CREATE  OR REPLACE VIEW empvu20
2       AS  SELECT        *
3       FROM              emp
4       WHERE             deptno = 20
5       WITH CHECK OPTION CONSTRAINT empvu20_ck;
```

•WITH CHECK OPTION 절 사용

- 뷰를 통해 참조 무결성 체크를 수행하는 것이 가능하다. 또한 DB 레벨에서 제약 조건을 적용할 수 있다. 뷰는 데이터 무결성을 보호하기 위해 사용될 수 있지만 사용은 매우 제한된다. 뷰를 통해 수행되는 INSERT와 UPDATE는 WITH CHECK OPTION 절이 있으면 뷰를 가지고 검색할 수 없는 행 생성을 허용하지 않음을 명시한다.

DML 연산 부정

- 뷰의 정의에 **WITH READ ONLY** 옵션을 추가하여 **DML** 연산이 수행될 수 없게 한다.

```
SQL> CREATE  OR REPLACE VIEW empvu10
                (employee_number, employee_name, job_title)
2       AS  SELECT          empno, ename, job
3       FROM                emp
4       WHERE               deptno = 10
5       WITH READ ONLY;
```

- 뷰의 임의의 행에서 **DML**을 수행하려고 하면 오라클 서버 에러 **ORA-01752**가 발생한다.

뷰 제거

- 뷰는 데이터베이스에서 기본 테이블을 기반으로 하기 때문에 데이터 손실 없이 뷰를 삭제한다.
- 뷰를 제거하기 위해 **DROP VIEW** 문장을 사용한다. 이 문장은 데이터베이스에서 뷰 정의를 제거한다. 뷰 삭제는 뷰가 만들어진 기본 테이블에는 영향을 미치지 않는다. 그 뷰에 기초하여 만들어진 뷰 또는 다른 어플리케이션은 무효화 된다. 생성자 또는 **DROP ANY VIEW**권한을 가진 사람만 뷰를 제거할 수 있다.

```
DROP  VIEW   view
```

요약

- 뷰는 다른 테이블 또는 다른 뷰의 데이터에서 유래된다.
- 뷰는 다음의 장점을 제공한다.
  - 데이터베이스 액세스를 제한한다.
  - 질의를 단순화한다.
  - 데이터 독립성을 제공한다.
  - 동일 데이터의 다중 뷰를 허용한다.
  - 근본적인 데이터를 제거하지 않고도 삭제될 수 있다.

시퀀스

- 자동적으로 유일번호 생성
- 공유 가능한 객체
- 주로 기본키 값을 생성하기 위해 사용
- 메모리에 캐쉬되면 시퀀스 값을 액세스 하는 효율 향상

Create Sequence 문장

- 시퀀스 번호를 자동적으로 생성하기 위해 시퀀스 정의

```
Create Sequence sequence_name
        [INCREMENT BY n]
        [START WITH n]
        [{MAXVALUE n | NOMAXVALUE}]
        [{MINVALUE n | NOMINVALUE}]
        [{CYCLE | NOCYCLE}]
        [{CACHE n | NOCACHE}]
```

```
Create Sequence dept_deptno
        INCREMENT BY 3
        START WITH 91 MAXVALUE 100
        NOCYCLE NOCACHE;
```

시퀀스 확인

- **User_Sequences** 데이터 사전 테이블에서 시퀀스 값을 검사

```
SQL> Select sequence_name, min_value, max_value,
            increment_by, last_number
     From user_sequences;
```

- **Last_Number** 열은 다음 이용 가능한 시퀀스 번호를 표시

Nextval 과 Currval

- **Nextval**은 다음 사용 가능한 시퀀스 값을 반환
- **Currval** 은 현재 시퀀스 값

시퀀스 사용 - 1

- 서브쿼리의 일부가 아닌 **Select** 문장의 **Select** 리스트
- **Insert** 문장에서 서브쿼리 **Select** 리스트
- **Insert** 문장의 **Values** 절
- **Update** 문장의 **Set** 절
- 새로운 레코드 삽입시 **Nextval**을 사용하여 기본 키 생성
- 현재값을 확인 하기 위해서 더미 테이블을 이용하여 **Currval** 값 확인
  Select seq.currval from dual

시퀀스 사용 – 2

- 메모리에서 시퀀스 값을 캐쉬하는 것은 이 값에 대해 더 빠른 액세스를 허용
- 시퀀스 값에서 간격 발생 상황
  - **Rollback** 발생
  - **System** 실패
  - 여러 테이블에서 사용시
- **USER_SEQUENCES** 테이블을 질의하여 **NOCACHE**로 생성된 때에 한해서 다음 사용 가능한 시퀀스 확인

시퀀스 수정

- 증가값, 최대값, 최소값, 사이클 옵션 또는 캐쉬 옵션 변경

```
SQL>Alter SEQUENCE dept_deptno
        INCREMENT BY 1
        MAXVALUE 999999
        NOCACHE
        NOCYCLE;
```

- 시퀀스에 대한 **ALTER** 권한 소유시
- 새로운 번호에서 시작하려면 다시 생성 해야함

시퀀스 제거

- **DROP SEQUENCE** 문장을 사용하여 데이터 사전에서 시퀀스 제거
- 한번 제거되었다면 더 이상 참조될 수 없다.

인덱스란?

- 스키마 객체
- 포인터를 사용하여 행의 검색을 촉진하기 위해 오라클 서버가 사용
- 빠르게 데이터를 찾기 위해 빠른 경로 액세스 방법을 사용하여 디스크 I/O를 경감
- 인덱스하는 테이블에 대해 독립적
- 오라클 서버에 의해 자동적으로 사용되고 유지

인덱스 생성 방법

- 자동
  - 테이블 정의시 **PRIMARY KEY** 또는 **UNIQUE KEY** 제약 조건을 정의 할 때 자동 생성
- 수동
  - 행에 대한 액세스 시간을 향상 시키기 위해 열에서 유일하지 않은 인덱스를 생성할 수 있다.

인덱스 생성

- 하나 이상의 열의 인덱스를 작성

```
CREATE INDEX INDEX_NAME
ON TABLE (COLUMN[, COLUMN] ...);
```

- **Emp** 테이블의 **ename** 열의 질의 액세스 속도 향상

```
SQL> CREATE INDEX emp_ename_idx
  2  ON emp(ename);
```

인덱스 생성 지침

인덱스 생성

- **WHERE** 절이나 조인 조건에 자주 사용하는 열
- 광범위한 값을 포함하는 열
- 많은 수의 널 값을 포함하는 열
- 대형 테이블이 대부분 질의어가 행의 **2-4%**보다 적게 읽어 들일 것으로 예상되는 테이블

인덱스 생성 자제

- 테이블이 작을경우
- 질의 빈도가 낮은 열
- 대부분 질의가 행의 **2-4%** 이상 읽어 들일 것으로 예상되는 테이블
- 자주 갱신되는 테이블

인덱스 확인

- **USER_INDEXES** 데이터 사전 뷰는 인덱스의 이름과 그것의 유일성을 포함
- **USER_IND_COLUMNS** 뷰는 인덱스명, 테이블명, 열명을 포함

```
SQL> SELECT ic.index_name, ic.column_name,
  2         ic.column_position col_pos, ix.uniquencess
  3  FROM user_indexes ix, user_ind_columns ic
  4  where ic.index_name = ix.index_name
  5  AND ic.table_name = 'EMP';
```

인덱스 제거

- 데이터 사전에서 인덱스를 제거

```
SQL> DROP INDEX index;
```

- 데이터 사전에서 **EMP_ENAME_IDX** 인덱스를 제거

- 인덱스를 제거하기위해, 인덱스의 **DROP ANY INDEX** 권한을 갖고 있어야 함

```
SQL> DROP INDEX emp_ename_idx;
```

동의어

동의어(객체의 다른 이름)을 생성하여 객체에 대한 액세스를 단순화

- 다른 사용자가 소유한 테이블 참조
- 객체이름의 길이 단축

```
CREATE [PUBLIC] SYNONYM synonym
FOR object;
```

동의어 생성과 제거

- **DEPT_SUM_VU** 뷰에 대해 단축명을 생성

```
SQL> CREATE SYNONYM d_sum
  2  FOR dept_sum_vu;
```

- 동의어를 제거

```
SQL> DROP SYNONYM d_sum;
```

요약

- 시퀀스를 사용하여 시퀀스 번호 자동 생성
- **USER_SEQUENCES** 사전 테이블에서 시퀀스 정보 확인
- 질의 검색 속도를 향상시키기 위해 인덱스 생성
- **USER_INDEXES** 사전 테이블에서 인덱스 정보 확인
- 객체에 대한 대체 명을 제공하기 위해 동의어 사용