MYONGJI UNIVERSITY

OTHER OBJECTS

Dongseop Kwon

Outline

- Sequence
- View
- Trigger

MYONGJI UNIVERSITY

SEQUENCE

Sequence

- 일련번호 자동 생성기
- 용도
  - Unique한 번호를 생성하고자 할 때 (PK 등을 위하여)
  - 별도의 Concurrency나 Performance의 고려를 할 필요 없음

Sequence 생성

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

- INCREMENT BY n:번호 간격 (1)
- START WITH n: 시작번호(1)
- MAXVALUE n: 최대값 (최대 1027)
- MINVALUE n: 최소값 (최소 1)
- CACHE n: 오라클이 미리 생성해놓을 개수 (20)

Sequence 사용하기

- Pseudo Columns
  - CURRVAL: Sequence의 현재 값을 돌려준다.
  - NEXTVAL: Sequence의 다음 값을 돌려주며, 현재값을 다음값으로 바꾼다. (increment)

- Example

```
INSERT INTO emp (empno, ename)
 VALUES (seq_empno.NEXTVAL, 'KIM');
```

```
SELECT seq_empno.CURRVAL FROM dual;
```

Other Syntax

- ALTER SEQUENCE sequence_name
  …;

- DROP SEQUENCE sequence_name;

- Dictionary
  - USER_SEQUENCES
  - USER_OBJECTS

MYONGJI UNIVERSITY

VIEW

View

- 정의: 다른 테이블(들)에 기반한 논리적 가상 테이블
  - 실제로 존재하지 않음
  - 사용자가 View에 대한 질의를 한 순간 실제 테이블들을 통하여 질의 수행
- 용도
  - 보안강화
  - 데이터 복잡성을 단순화
  - 실제 테이블의 정의와 응용프로그램 분리
  - 복잡한 질의 숨김

```
CREATE VIEW staff AS
  SELECT employee_id, last_name, job_id, manager_id, dept_id
  FROM   employees;
```

View 생성

```
CREATE [OR REPLACE] [FORCE|NO FORCE] VIEW  view_name
      [(alias[,alias]...)]
 AS Subquery
      [WITH READ ONLY]
      [WITH CHECK OPTION [CONSTRAINT constraint]];
```

- 설명
  - FORCE: Base 테이블이 없어도 무조건 Create
  - WITH CHECK OPTION: View에 나타날 수 있는 행만 삽입/변경 가능

기타 Syntax

- DROP VIEW *view_name*;

- Dictionary
  - USER_VIEWS
  - USER_OBJECTS

View에 대한 변경

- 제약적으로 가능
  - Simple View는 DML(DELETE, INSERT, UPDATE) 가능
  - 제약조건을 만족시킬 수 없거나 원본 row를 유추할 수 없는 경우 등에는 변경이 불가

```
CREATE OR REPLACE VIEW v1 AS
    SELECT DISTINCT empno FROM emp;

CREATE OR REPLACE VIEW v1 AS
    SELECT COUNT(empno) FROM emp;

CREATE OR REPLACE VIEW v3 AS
 SELECT * FROM emp, dept WHERE emp.deptno =
 dept.deptno;

CREATE OR REPLACE VIEW v4 AS
 SELECT empno, sal * 2 AS dsal FROM emp;
```

Materialized (실체화) View

- VIEW의 내용이 실제 별도의 테이블로 저장됨
- 원 테이블의 내용이 변경되면 자동으로 변경됨
  - 복잡한 질의의 수행 시간이 단축됨
  - Query Rewrite 에 이용될 수 있음

```
CREATE MATERIALIZED VIEW sales_mv AS
  SELECT t.calendar_year, p.prod_id,
         SUM(s.amount_sold) AS sum_sales
  FROM   times t, products p, sales s
  WHERE  t.time_id = s.time_id
  AND    p.prod_id = s.prod_id
  GROUP BY t.calendar_year, p.prod_id;
```

MYONGJI UNIVERSITY

TRIGGER

Trigger

- 특정 Event가 발생될때 자동으로 실행(fire)되는 PL/SQL Block
- 이벤트 종류
  - DML(INSERT, UPDATE, DELETE),
  - DDL (CREATE, ALTER, DROP),
  - DATABASE…
- 타이밍
  - BEFORE, AFTER, …

용도

Feature	Enhancement
Security	The Oracle Server allows table access to users or roles. Triggers allow table access according to data values.
Auditing	The Oracle Server tracks data operations on tables. Triggers track values for data operation on tables.
Data Integrity	The Oracle Server enforces integrity constraints. Triggers implement complex integrity rules.
Referential Integrity	The Oracle Server enforces standard referential integrity rules. Triggers implement nonstandard functionality.
Table Replication	The Oracle Server copies tables asynchronously into snapshots. Triggers copy tables synchronously into replicas.
Derived data	The Oracle Server computes derived data values manually. Triggers compute derived data values automatically.
Event logging	The Oracle Server logs events explicitly. Triggers log events transparently.

Create

```
CREATE [OR REPLACE]  TRIGGER  trigger_name
 timing event1 [ OR  event2  OR  event3 … ]
 ON  {table_name|view_name|SCHEMA|DATABASE}
 [REFERENCING OLD AS old | NEW AS new]
 [FOR EACH ROW  [WHEN  ( condition ) ] ]
trigger_body
```

- timing: BEFORE, AFTER, INSTEAD OF (View일 경우)
- event: DML(INSERT, UPDATE, DELETE), DDL (CREATE, ALTER, DROP), DATABASE…
- FOR EACH ROW: 행트리거
- trigger_body: PL/SQL Block, 프로시저 호출 가능

예제 : 트리거를 이용한 제약조건 처리

```
CREATE  OR  REPLACE  TRIGGER  validate_sal
    BEFORE  INSERT  OR  UPDATE  OF  sal  ON  emp
    FOR EACH ROW
 BEGIN
    IF     :NEW.sal  <= 100    THEN
       RAISE_APPLICATION_ERROR(-20600,
            'Wrong Input/Update : SAL column value must be
      greater than 100.') ;
    ELSIF :NEW.sal >= 10000 THEN
       RAISE_APPLICATION_ERROR(-20601,
            'Wrong Input/Update : SAL column value must be
      less than 10000.') ;
    END  IF ;
 END;
```

예제 : 트리거를 이용한 테이블 통계정보 유지

```
CREATE  OR  REPLACE  TRIGGER  trg_sum_t1
    BEFORE  INSERT  OR  DELETE OR UPDATE  OF  sal  ON  t1
    FOR EACH ROW
 BEGIN
    IF DELETING THEN
       UPDATE t1_sum SET count_sal = count_sal-1,
                    sum_sal = sum_sal - :old.sal;
    ELSIF INSERTING THEN
       UPDATE t1_sum SET count_sal = count_sal+1,
                    sum_sal = sum_sal + :new.sal;
    ELSIF UPDATING THEN
       UPDATE t1_sum SET
                    sum_sal = sum_sal - :old.sal + :new.sal;
    END IF;
 END;
```

기타 Syntax

- 활성화, 비활성화
  - ALTER TRIGGER trigger_name DISABLE | ENABLE ;
  - ALTER TABLE *table_name* DISABLE | ENABLE ALL TRIGGERS ;
- 재컴파일
  - ALTER TRIGGER *trigger_name* COMPILE ;
- 삭제
  - DROP TRIGGER *trigger_name* ;
- Dictionary
  - USER_OBJECTS, USER_TRIGGERS, USER_ERRORS