suanlab

데이터베이스
Database

ORACLE®

DATABASE

14
뷰(View)

VIEW
MATERIALIZED VIEW

뷰(View)

- 뷰(View)
  - 사용자에게 접근이 허용된 데이터만 제한적으로 제공
  - 하나 이상의 테이블로부터 유도된 가상 테이블
  - 뷰에 대한 질의 실행시 정의된 테이블로 대체되어 실행
  - 임시 작업을 위한 용도도 활용되고 사용상의 편의성을 최대화함
- 뷰의 특징
  - 테이블에서 유도되었기 때문에 구조가 같음
  - 가상 테이블이라 물리적으로 구현되지 않음
  - 데이터의 논리적 독립성 제공
  - 뷰로 필요한 데이터만 처리하므로 관리 용이
  - 여러 테이블을 조인하여 뷰 생성 가능
  - 뷰에 나타나지 않은 데이터를 안전하게 보호 (사용자별로 접근 권한 설정)
  - 테이블의 기본키를 포함하여 뷰를 구성하면 삽입, 삭제, 갱신 가능
- 뷰의 장단점

장점	단점
• 논리적 데이터 독립성 제공 • 동일 데이터에 대해 동시에 여러 사용자 요구 지원 • 사용자의 데이터 관리 편의성 제공 • 접근 제어를 통한 보안 제공	• 독립적 인덱스 생성 불가 • 뷰의 정의 변경 불가 • 삽입, 수정, 삭제 연산에 제약

- 뷰의 종류
  - 단순 뷰(Simple View): 하나의 테이블에서 뷰 생성
  - 복합 뷰(Complex View): 두개 이상의 테이블을 조인하여 뷰 생성
  - 인라인 뷰(Inline View): SELECT문의 FROM 절에 기술된 SELECT 문
- 뷰 예제

```
SELECT *
FROM emp_details_view;
```

CREATE OR REPLACE VIEW

- 뷰 생성 및 변경

```
CREATE OR REPLACE VIEW 뷰 이름
AS
    SELECT 질의
```

```
SELECT *
FROM employees;
```

```
CREATE OR REPLACE VIEW emp_view
AS
    SELECT employee_id, first_name, last_name, email
    FROM employees;
```

```
SELECT *
FROM emp_view;
```

```
CREATE OR REPLACE VIEW new_employee_view
AS
    SELECT employee_id, first_name, last_name,
           email, hire_date, job_id
    FROM employees
    WHERE employee_id > 206;
```

```
INSERT INTO new_employee_view
VALUES (207, 'Suan', 'Lee', 'suan', '21/01/01', 'IT_PROG');
```

```
SELECT *
FROM new_employee_view;
```

- 뷰 제거

```
DROP VIEW emp_view;
```

```
DROP VIEW new_employee_view;
```

CREATE OR REPLACE VIEW

- 읽기 전용 뷰 생성

```
CREATE OR REPLACE VIEW 뷰 이름
AS
    SELECT 질의
    WITH READ ONLY;
```

```
CREATE OR REPLACE VIEW salary_order_view
AS
    SELECT first_name, last_name, job_id, salary
    FROM employees
    ORDER BY salary DESC
    WITH READ ONLY;
```

```
SELECT *
FROM salary_order_view;
```

```
INSERT INTO salary_order_view
VALUES ('Suan', 'Lee', 'IT PROG', 10000);
```

```
CREATE OR REPLACE VIEW job_salary_view
AS
    SELECT job_id, SUM(salary) sum_salary,
           MIN(salary) min_salary, MAX(salary) max_salary
    FROM employees
    GROUP BY job_id
    ORDER BY SUM(salary)
    WITH READ ONLY;
```

```
SELECT *
FROM job_salary_view;
```

- 뷰 제거

```
DROP VIEW salary_order_view;
```

```
DROP VIEW job_salary_view;
```

CREATE MATERIALIZED VIEW

- 구체화된 뷰 생성
  - 뷰는 실체가 없이 SELECT 문으로만 존재하며 데이터는 없음
  - 구체화된 뷰는 실제 데이터가 존재하는 뷰

```
CREATE MATERIALIZED VIEW 뷰 이름
[ BUILD { IMMEDIATE ¦ DEFERRED }
  REFRESH { ON COMMIT ¦ ON DEMAND } { FAST ¦ COMPLETE ¦ FORCE ¦ NAVER }
  ENABLE QUERY REWRITE ]
AS
    SELECT 질의;
```

  - BUILD IMMEDIATE: 구체화된 뷰 생성 후, 동시에 구체화된 내부에 데이터가 채워짐
  - BUILD DEFFERED: 뷰 내부에 데이터가 나중에 채워짐
  - REFRESH ON COMMIT: 원본 테이블에 커밋이 발생될 때마다 구체화된 뷰의 내용이 변경
  - REFRESH ON DEMAND: 직접 DBMS_MVIEW 패키지를 실행해서 구체화된 뷰의 내용을 변경
  - FAST, FORCE: 원본 테이블에 변경된 데이터만 구체화된 뷰에 적용
  - COMPLETE: 원본 테이블이 변경되면 전체를 구체화된 뷰에 적용
  - NEVER: 원본 테이블이 변경되어도 구체화된 뷰에는 적용 안함

```
CREATE MATERIALIZED VIEW country_location_view
    BUILD DEFERRED
AS
    SELECT C.country_name, L.state_province, L.street_address
    FROM HR.countries C, HR.locations L
    WHERE C.country_id = L.country_id;
```

```
SELECT *
FROM country_location_view;
```

```
EXECUTE DBMS_MVIEW.REFRESH(LIST =>'country_location_view');
```

```
DROP MATERIALIZED VIEW country_location_view;
```

CREATE MATERIALIZED VIEW

- 구체화된 뷰 생성

```
CREATE MATERIALIZED VIEW country_location_view
    BUILD IMMEDIATE
    REFRESH ON DEMAND COMPLETE
AS
    SELECT C.country_name, L.state_province, L.street_address
    FROM HR.countries C, HR.locations L
    WHERE C.country_id = L.country_id;
```

```
SELECT *
FROM country_location_view;
```

```
INSERT INTO HR.countries
VALUES ('KR', 'Republic of Korea', 3);
```

```
INSERT INTO HR.locations
VALUES (3300, '1 Cheongwadae-ro', 03048, 'Seoul', 'Jongno-gu',
'KR');
```

```
EXECUTE DBMS_MVIEW.REFRESH(LIST =>'country_location_view');
```

```
SELECT *
FROM country_location_view;
```

```
DELETE HR.locations
WHERE location_id = 3300;
```

```
DELETE HR.countries
WHERE country_id = 'KR';
```

[실습] 뷰 생성

- employees 테이블에서 07년도에 고용된 직원의 employee_id, first_name, last_name, email, hire_date, job_id 컬럼 값을 가지는 employee_07_view 이름의 뷰 생성

- employees 테이블에서 department_id와 job_id로 그룹화하고, 평균 salary가 9000 초과인 직원의 department_id, job_id, salary의 평균값인 salary_avg 컬럼 값을 가지며 평균 salary 값이 높은 순으로 정렬한 high_salary_view 이름의 읽기 전용 뷰 생성

[실습] 뷰 생성

- employees 테이블을 manager_id와 employee_id를 기준으로 자체 조인한 뒤에 department_id와 직원의 first_name과 last_name을 결합하고 관리자의 first_name과 last_name을 결합한 뒤에 department_id를 기준으로 정렬하여 employee_manager_view 이름의 읽기 전용 뷰 생성

- employees, departments, jobs, locations 테이블을 조인하여 first_name, last_name, department_name, job_title, city를 보여주는 구체화 뷰를 내부의 데이터를 채우고 요청 시 변경된 내용을 뷰에 반영하도록 생성


suanlab
이수안 컴퓨터 연구소
suan computer laboratory