SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

삼성 청년 SW아카데미

MySQL

Confidential

<알림>

본 강의는 삼성 청년 SW아카데미의 컨텐츠로
보안서약서에 의거하여
강의 내용을 어떠한 사유로도 임의로 복사,
촬영, 녹음, 복제, 보관, 전송하거나
허가 받지 않은 저장매체를
이용한 보관, 제3자에게 누설, 공개,
또는 사용하는 등의 행위를 금합니다.

SAMSUNG SW ACADEMY FOR YOUTH

목차

Confidential

- View
- Index

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

View

View

Confidential

❖ View는 우리말로 해석하면 관점

- 하나의 테이블 혹은 여러 테이블에 대하여 특정 사용자나 조직의 관점에서 데이터를 바라볼 수 있도록 해주는 수단
- View는 가상 테이블이라고도 부름
  - 일반 테이블: 실제로 물리적인 데이터를 갖고 있음
  - View: 물리적인 데이터를 갖고 있지 않고, **뷰**가 **이미 정의된 일반 테이블**로부터 데이터를 가져다 보여줌
- 사용 예제 시나리오
  - 한 회사의 사원 정보는 여러 부서에서 사용됨
  - 인사팀에서는 급여 지급을 위해 [사번, 이름, 입사일자, 급여액] 데이터 필요
  - 기획실에서는 인력 배치를 위해 [사번, 이름, 근무부서, 담당업무] 데이터 필요
  - 사내복지팀에서는 근속년수에 따른 선물을 보내기 위해 [사번, 이름, 입사일자, 근무부서 주소] 데이터 필요
- 과거의 파일 시스템 환경에서는 사원 정보를 각 부서마다 따로 관리함에 따라 데이터의 중복, 불일치 발생
- 각 부서의 관점에 따라 보이지 않도록 숨겨야 하는 컬럼이 발생 (급여액)

❖ 각 부서별 View는 아래와 같이 생성

```
# 인사팀
CREATE VIEW view_emp1 AS
SELECT employee_id "사번", first_name "이름", hire_date "입사일자", salary "급여액"
FROM employees;
```

```
# 기획실
CREATE VIEW view_emp2 AS
SELECT e.employee_id "사번", e.first_name "이름", d.department_name "근무부서", j.job_title "담당업무"
FROM employees e
LEFT OUTER JOIN departments d ON e.department_id = d.department_id
LEFT OUTER JOIN jobs j ON e.job_id = j.job_id;
```

```
# 사내복지팀
CREATE VIEW view_emp3 AS
SELECT e.employee_id "사번", e.first_name "이름", d.department_name "근무부서",
    CONCAT(l.street_address, ", ", l.postal_code, ", ", l.city) "근무부서 주소"
FROM employees e
LEFT OUTER JOIN departments d ON e.department_id = d.department_id
LEFT OUTER JOIN locations l ON d.location_id = l.location_id;
```

View

❖ View는 다음과 같은 목적으로 사용

- 하나의 테이블에 대하여 여러 부서에서 서로 다른 관점으로 보기를 원할 때 사용
- 테이블에 급여와 같이 일반 사용자에게는 감추어야 할 컬럼이 있을 때,
  급여 컬럼을 제외하고 View를 만들어 제공함으로써 보안 유지할 필요가 있을 때 사용
- 자주 사용하는 복잡한 질의문을 미리 View로 만들어 두고 간편하게 사용

❖ View에 대해서도 행 추가, 수정, 삭제가 가능

- 단, 기존 테이블의 무결성 규칙을 만족한다면 실행될 것이고, 문제가 된다면 실행되지 않음

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

Index

❖ 테이블에 대한 **검색 속도를 향상**시킬 수 있는 수단

- 책 뒤에 있는 색인을 생각하자
- 10만 명의 사원정보가 테이블에 저장되어 있다고 가정해보자
  - 사원의 이름을 가지고 사원정보를 검색한다면, 첫 번째 행부터 마지막 행까지 순차적으로 검색하여 이름이 일치하는 사원을 검색
    - 순차적으로 검색을 한다면 최악의 경우 10만 번째 검색에서 원하는 사원정보를 얻게 됨
  - 사원의 이름으로 오름차순 정렬하여 이진 검색을 한다면, 순차적으로 검색하는 것보단 적은 횟수로 원하는 사원정보를 얻게 됨
    - 수시로 입력, 삭제되는 사원정보를 매번 정렬한 상태로 유지하는 비용이 많이 듦
    - 이런 경우 인덱스를 활용하자!

❖ 인덱스의 동작 원리

- 검색의 기준이 되는 컬럼만을 뽑아 정렬한 상태를 유지
- 인덱스의 각 행은 원래 데이터가 저장되어 있는 테이블에 대응하는 행의 주소 값을 가지고 있음
- 예: 사용자가 사원 'Steven'에 대한 검색을 DBMS에게 요청한 경우
  - DBMS는 먼저 인덱스에서 'Steven'을 검색하여 원래 데이터의 행 주소를 알아낸다.
  - 알아낸 원래 데이터의 행 주소를 가지고 해당 행을 찾게 된다.
  - 인덱스는 항상 정렬된 상태를 유지하고 있으므로 인덱스에서 사원의 이름을 찾는 것은 빠른 시간에 할 수 있음
- 모든 컬럼에 대해 인덱스를 만들어도 될까?
  - 바람직하지 않다.
  - 인덱스는 저장 공간을 차지하고, 항상 정렬 상태를 유지해야 하는 비용이 발생
  - 인덱스가 많아지면 인덱스를 재정렬하는데 많은 시간이 필요하므로 DBMS의 성능을 저하시킴
  - 꼭 필요한 컬럼에 대해서만 인덱스를 적용하자

❖ 인덱스 지정에 대한 일반적인 지침

- 인덱스로 지정하는 컬럼은 SQL의 WHERE 절에서 비교 대상이 되는 컬럼 또는 JOIN에 사용되는 컬럼이다.
- 행의 수가 적으면 (예: 200~300개) 인덱스를 지정하여도 별 효과가 없다.
- 인덱스로 지정한 컬럼에 의해 검색했을 때, 검색 결과가 전체 행의 10~15% 미만일 때 효과가 있다.
  - 예를 들어, '성별' 컬럼으로 검색하면 전체 행의 50% 정도가 검색 결과로 나올 것이다.
    이런 경우는 순차 검색이 더 빠르기 때문에 '성별' 컬럼에 인덱스를 지정하지 않는다.

❖ 대부분의 DBMS에서는 테이블의 기본키(Primary Key) 또는 UNIQUE에 자동으로 인덱스를 만들어 준다.

- 클러스터형 인덱스
  - 보통 기본키(Primary Key) 컬럼에 자동생성. 검색의 기준이 되는 경우가 많고, 행의 중복 여부 확인 작업에 이용하기 위해 생성
  - 내용 자체가 순서대로 정렬되어 있어서 인덱스 자체가 데이터의 내용과 같음
- 보조 인덱스
  - 보통 UNIQUE 지정된 컬럼에 자동생성. 새로 만드는 인덱스들은 대부분 보조 인덱스가 됨
  - 인덱스가 별도로 있고, 인덱스의 각 행에 있는 원래의 데이터의 행 주소를 가지고 실제 데이터를 찾음

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

내일 방송에서 만나요!

삼성 청년 SW 아카데미