8253jang@daum.net
장희정

VIEW(뷰)

- 뷰(view)는 데이터베이스에 존재하는 가상의 테이블이다.
- 뷰는 테이블이나 데이터를 직접 소유하지 않고 테이블의 형태만 차용하여 테이블처럼 사용할 수 있게 해주는 역할을 한다. 사용자는 뷰를 테이블과 거의 동일하게 취급한다.
- 뷰는 SELECT 문으로 만들며, 실제 사용자가 뷰를 호출했을 때 SELECT 문이 실행되고 그 결과가 화면에 출력된다.
- 뷰는 한 테이블만 구성할 수도 있고, 다수의 테이블로 조인하여 구성할 수도 있다.
- 생성할 때 옵션을 어떻게 설정하느냐에 따라 DML 여부가 달라진다.
  JOIN을 뷰로 만들었을 때는 DML 안된다.

VIEW를 사용하는 이유

1) **편의성** : 복잡한 쿼리문을(조인, 서브쿼리) 미리 뷰로 만들어 사용하면 간단하게 검색 가능하다.

2) **보안성** : 보안을 위해 사용한다.(관리자 유형에 따라 특정 컬럼 만 선택하여 보여준다)
   테이블의 특정 컬럼과 레코드를 선택해서 뷰를 만들어 권한에 제한이 있는 DBA에게 원본테이블이 아닌 뷰의 이름을 알려줌으로써 DBA는 뷰만 접근 할 수 있도록 제한을 둔다.
   예를 들어 회사에서 부서별로 필요한 고객 정보 외에 다른 정보를 보여주고 싶지 않을 때, 별도의 테이블을 만들어 데이터를 복사하는 방식이 아닌 부서별로 필요한 열로만 구성한 가상의 뷰 테이블을 사용하면 보안 및 편의성이 강화된다.ㄴ

VIEW의 장단점

장점	단점
• 복잡한 쿼리를 단축시켜 놓기 때문에 사용자는 편리하게 사용할 뿐만 아니라 유지·보수에도 유리하다. • 테이블과 열 이름 등을 숨길 수 있으며, 권한에 따라 필요한 열만 구성해 사용할 수 있으므로 보안성이 우수하다.	• 한 번 정의된 뷰는 변경할 수 없다. • 삽입, 삭제, 갱신 작업을 하는 데 제한 사항이 많다. • 뷰 테이블은 인덱스를 가질 수 없다.

☞ VIEW 생성방법

```
CREATE VIEW 뷰이름
AS SELECT문장;
```

8253jang@daum.net
장희정

☞ VIEW 수정 + 생성
CREATE OR REPLACE VIEW 뷰이름
AS SELECT문장;

☞ VIEW 삭제
DROP VIEW 뷰이름;

☞ VIEW를 생성 할 때 옵션 설정
WITH CHECK OPTION : VIEW를 생성할 때 조건을 이용해서 만들게 되면
INSERT, UPDATE, DELETE를 할 때 조건에 충족하지 않는
정보는 DML을 할 수 없도록 하는 것.

☞ VIEW 정보확인하기
DESCRIBE V_EMP;
SHOW CREATE VIEW V_EMP;

INDEX

인덱스는 색인을 만들어서 조회성능을 극대화 하기 위해 만드는 객체이다.
저장된 수많은 데이터 중에 특정 조건을 만족하는 데이터를 조회할 때마다 모든 데이터를 일일이 검사해서 필요한 데이터만 찾아야 한다면 매우 많은 조회 비용이 발생한다. 이는 데이터베이스의 응답이 느려진다는 것으로, 성능이 저하됨을 의미한다.

이러한 문제를 해결하기 위해 사용하는 기술이 인덱스(index)로 필요한 데이터를 바로 찾을 수 있도록 참고할 수 있는 데이터이다.
인덱스는 열 단위로 지정할 수 있다. 예를 들어 책에는 차례나 찾아보기가 있어서 필요한 내용을 빠르게 찾아갈 수 있듯이, 원하는 데이터를 빠르게 찾아갈 수 있도록 하는 기술이 데이터베이스의 인덱스이다.

인덱스는 무조건 만든다고 해서 좋은 것은 아니고 전체 테이블의 구조를 잘 보고 선택해야 한다.
왜냐하면, 인덱스는 특정테이블의 레코드검색 속도를 빠르게 하는데 도움이 되지만 전체 시스템 내에 너무 많은 인덱스가 있으면 전체 성능은 저하될 수 있다. 잦은 INSERT, UPDATE, DELETE를 하게 되면 그때마다 INDEX설정이 변경되기 때문에 전체 계정 안에서 시스템 부하가 생긴다.(느려진다) 검색속도를 높이기 위해서 인덱스를 먼저 선택하기 보다는 SQL문장을 좀더 효율적으로 짜

는 노력을 하는 것이 더 좋다.

주로 인덱스 대상이 되는 컬럼은 검색의 조건으로 많이 활용되는 컬럼으로 만든다.

인덱스의 장점 / 단점

장점	단점
• 원하는 데이터를 빠르게 검색할 수 있다. • 불필요한 검색 비용을 절약하고 I/O 성능을 높일 수 있다. • 데이터 검색뿐만 아니라 조인 시에도 빠른 성능을 얻을 수 있다. 특히 조인에서 데이터가 폭발적으로 증가할 수 있으므로 인덱스가 없다면 매우 느린 성능을 보여 준다.	• 인덱스를 별도로 저장하는 인덱스 페이지를 구성하기 위해 추가 공간이 필요하다. • 데이터를 수정할 때 연결된 인덱스 정보도 함께 수정해야 하는 경우 추가 비용이 발생한다. • 인덱스 정보를 수정할 때 잠금이 발생해 데이터베이스 성능이 느려질 수 있다.

MySQL의 클러스터형 인덱스 vs 비클러스터형 인덱스

인덱스의 저장 방식에 따라 클러스터형(Clustered Index)과 비클러스터형(Non-Clustered Index)으로 나뉜다.

클러스터형 인덱스 (Clustered Index)

특징

- **데이터가 인덱스에 직접 저장된다.** (테이블 자체가 인덱스로 정렬됨).
- **기본 키(Primary Key)** 가 자동으로 클러스터형 인덱스로 설정된다..
- **InnoDB 엔진에서만 지원** (MyISAM 에서는 지원하지 않음).
- **테이블의 물리적 순서가 인덱스 순서와 동일**하여 **한 개의 테이블당 하나만 존재 가능**하다.

장점

- **범위 검색(BETWEEN, ORDER BY) 속도가 빠름** (인덱스와 데이터가 함께 저장됨).
- **Primary Key 기반 검색이 빠름**.

단점

- **INSERT/UPDATE 속도가 느려질 수 있다.** (기존 데이터가 정렬된 상태여야 하기 때문).
- **한 테이블당 하나만 만들 수 있다.**

비클러스터형 인덱스 (Non-Clustered Index)

특징

- **데이터가 인덱스에 직접 저장되지 않는다.** 대신 **인덱스는 데이터의 위치(주소)를 저장**한다.
- **테이블 데이터의 물리적 순서와 인덱스 순서는 다를 수 있다.**

8253jang@daum.net
장희정

- **한 테이블에 여러 개의 비클러스터형 인덱스를 생성 가능**하다.

장점

- **하나의 테이블에 여러 개의 인덱스를 만들 수 있다.**
- **자주 조회하는 컬럼에 추가 인덱스를 설정하여 특정 쿼리 성능 향상 가능하다.**

단점

- **데이터를 찾기 위해 한 단계 더 거쳐야한다.** (인덱스를 검색한 후, 데이터 위치를 찾아감 → "LOOKUP" 과정).
- **데이터 수정(INSERT, UPDATE, DELETE) 시 오버헤드 발생 가능하다.** (인덱스를 재조정해야 하므로).

☞ **인덱스 생성**

CREATE INDEX 인덱스명 ON 테이블이름(컬럼명, 컬럼명,....) ;

☞ **인덱스 삭제**

DROP INDEX 인덱스명 ON 테이블이름;