데이터베이스

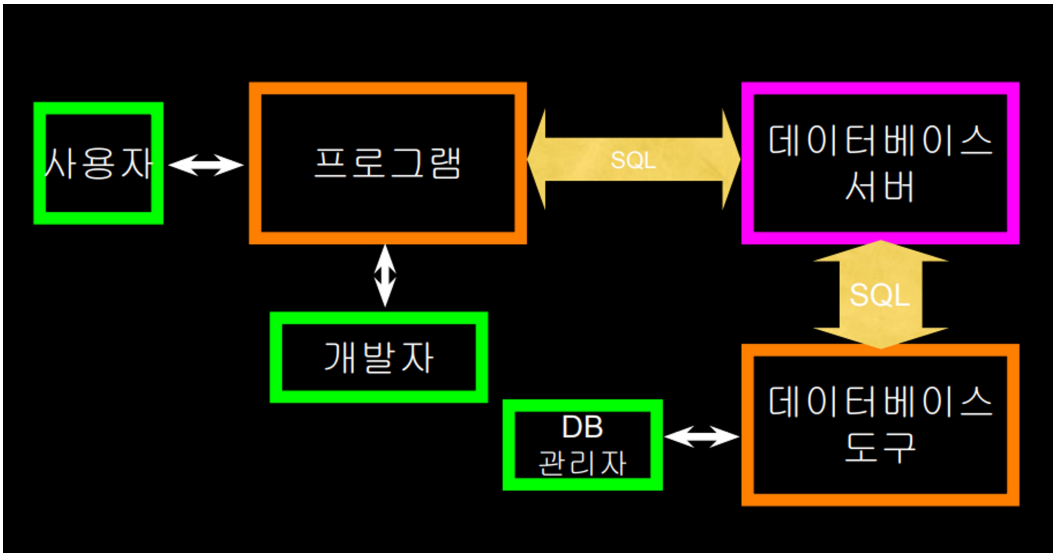
2019년 6월 10일 월요일 오후 3:44

여기서는 SQLite 를 사용함

용어

- 데이터베이스 - 여러 개의 테이블을 포함
- 관계(또는 테이블) - 튜플과 속성을 포함
- 튜플(또는 로우) - 사람/노래처럼 객체를 표현할 수 있는 필드의 집합
- 속성(또는 칼럼/필드) - 객체를 나타내는 로우에 있는 데이터 중 하나

❖ 데이터베이스를 이용한 웹 프로그램



- 프로그램 개발자 - 프로그램의 논리를 세우고 외형, 느낌을 관리 - 프로그램의 문제점 모니터링
- 데이터베이스 관리자 - 프로그램이 서비스 환경에서 실행되는 동안 데이터베이스를 모니터링 및 수정
- 두 사람 모두 데이터 모델링에 관여

❖ SQL

구조화된 쿼리 언어는 데이터베이스에 명령을 내리기 위해 우리가 사용할 언어

✓ **테이블 생성하기**

테이블을 생성할 때는 CREATE TABLE '테이블이름'과 같이 SQL 명령어를 작성합니다.
소괄호 안에는 컬럼(열)의 이름과 데이터의 특성을 작성해줍니다.

```
CREATE TABLE Users(
  name VARCHAR(128),
  email VARCHAR(128)
)
```

✓ **데이터 입력하기**

데이터를 입력할 때는 INSERT INTO 테이블이름(컬럼1, 컬럼2, ...) VALUES(값1,값2,...) 과 같은 형태로 SQL문을 작성합니다.

```
INSERT INTO Users (name, email) VALUES ('Kristin', 'kf@umich.edu')
```

✓ **데이터 삭제하기**

데이터를 삭제할 때는 DELETE FROM 테이블이름 WHERE 조건 과 같은 형태로 SQL문을 작성합니다.
다음 코드는 email 컬럼의 값이 'ted@umich.edu'인 행을 찾아서 삭제하라는 의미입니다.

```
DELETE FROM Users WHERE email='ted@umich.edu'
```

✓ **데이터 갱신(update)하기**

데이터를 갱신할 때는 UPDATE 테이블이름 SET 갱신데이터 WHERE 조건 과 같은 형태로 SQL문을 작성합니다.
다음 코드는 email 컬럼의 값이 'csev@umich.edu'인 행을 찾아서 name 컬럼 값을 'Charles'로 갱신하라는 의미입니다.

```
UPDATE Users SET name='Charles' WHERE email='csev@umich.edu'
```

✓ **데이터 추출(select)하기**

데이터를 추출할 때는 SELECT 추출데이터 FROM 테이블이름 WHERE 조건 과 같은 형태로 SQL문을 작성합니다.
다음 코드는 Users 테이블에서 email이 csev@umich.edu인 데이터의 모든 컬럼을 추출하라는 의미입니다.

```
SELECT * FROM Users WHERE email='csev@umich.edu'
```

여기에서 데이터를 특정 컬럼 중심으로 정렬하고 싶을 경우 ORDER BY를 사용할 수 있습니다.
기본적으로 오름차순으로 정렬하지만 DESC 옵션을 사용하면 내림차순으로 정렬이 됩니다.

```
SELECT * FROM Users ORDER BY email
SELECT * FROM Users ORDER BY name DESC
```

❖ **데이터베이스 모델 만들기**

- 프로그램을 위한 데이터 객체의 그림을 그린 뒤, 개체와 관계를 어떻게 나타낼 것인지 고민
- 기본 규칙: 동일한 문자열을 중복으로 넣지 말 것 - 관계를 대신 이용
- "실제"로 무언가가 존재한다면 데이터베이스에는 반드시 복사본 하나만 있어야 함

❖ **데이터베이스에서의 관계 표현**

데이터베이스 정규화 (3NF)

- "수많은" 데이터베이스 이론이 있음. - 서술 논리(predicate calculus)를 모르면 이해하기 어려움
- 데이터 중복 불가 - 데이터 참조 - 데이터 가리키기
- 정수를 사용하여 키 값 및 참조 표현
- 참조하기 쉽도록 각 테이블에 특별한 "키" 칼럼을 추가. 관례에 따라, 많은 개발자들이 이 칼럼을 "id"라고 함

세 종류의 키

- 기본 키 - 일반적으로 정수이며 자동으로 증가
- 논리 키 - 외부에서 검색 시 사용
- 외래 키 - 일반적으로 정수이며 다른 테이블의 로우를 가리킴

키 규칙

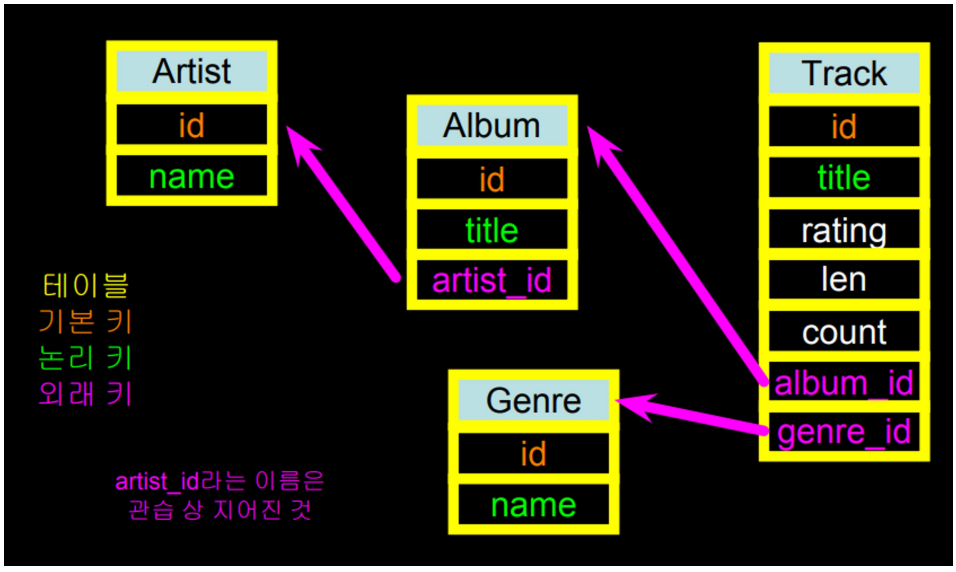
대표적인 사례

- 절대 논리 키를 기본 키로 사용하지 말 것.
- 논리 키는 느리지만 바뀔 수 있음.
- 문자열의 일치를 기반으로 한 관계는 정수의 일치를 기반으로 한 것보다 덜 효율적

외래 키

- 외래 키는 다른 테이블의 기본 키 를 가리키는 키를 일컬음
- 모든 기본 키가 정수라면 모든 외래 키도 정수임. - 이게 좋음

✓ **(테이블에서) 관계 형성**



❖ **JOIN**

관계의 힘

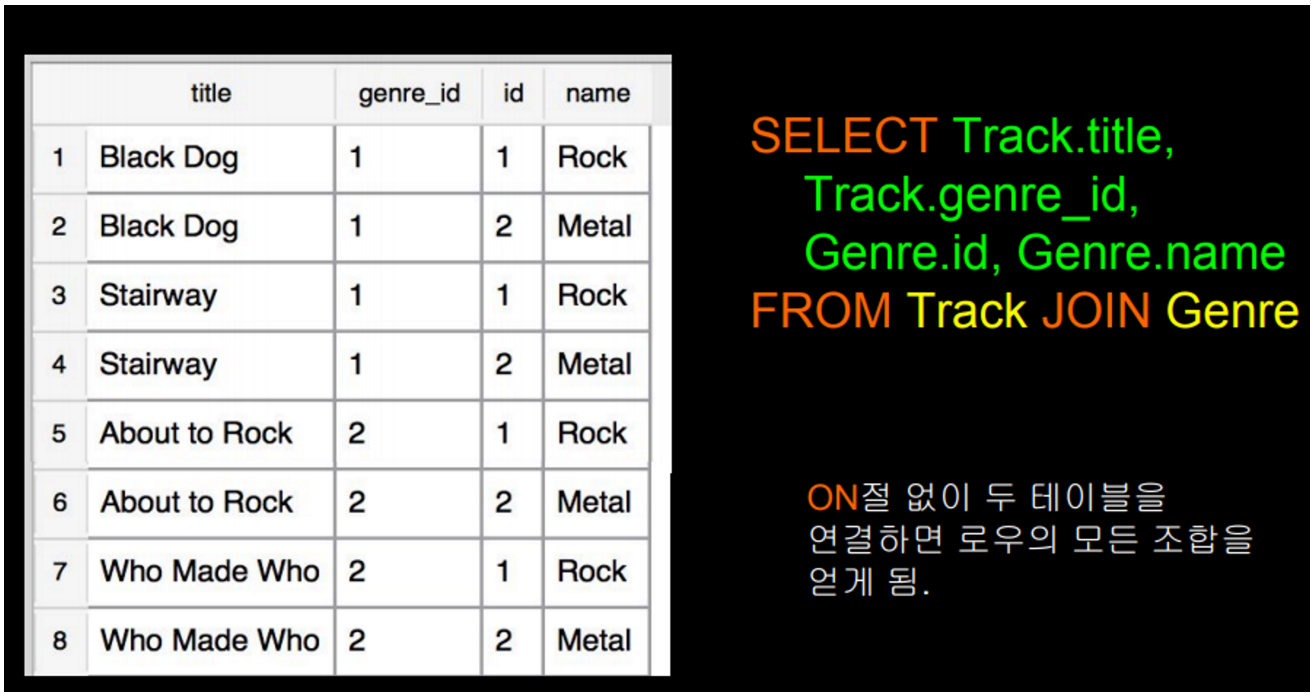
• 중복된 데이터를 제거하는 대신 데이터의 유일한 복사본을 참조함으로써 우리는 관계형 데이터베이스가 아주 큰 규모의 데이터도 매우 빠르게 읽을 수 있는 정보"망"을 구축함

• 종종 어떤 데이터를 원할 때 외래 키로 연결된 수많은 테이블을 거쳐서 얻을 수 있음

JOIN 연산

• JOIN 연산은 SELECT 연산의 일부로써 여러 테이블을 연결

• JOIN을 사용할 때는 반드시 ON절에서 테이블의 연결을 위해 키를 어떻게 사용할지 알려줘야 함

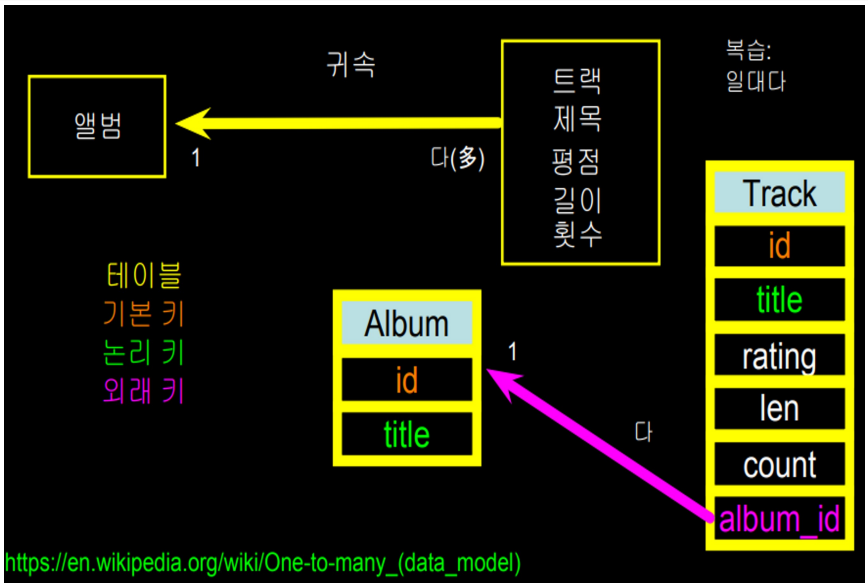


❖ **다대다 관계**

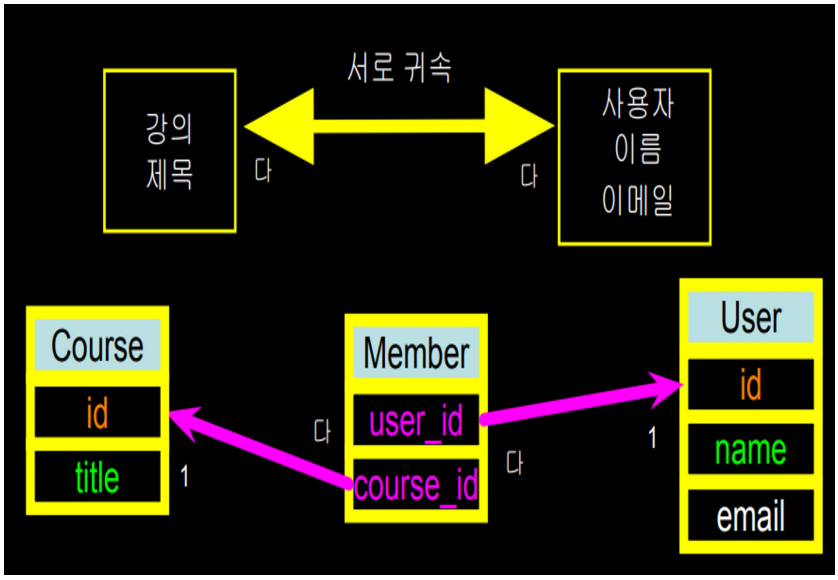
• 가끔 다대다 모델링이 필요할 때가 있음.

• 두 개의 외래 키를 가지고 "연결" 테이블을 추가해야 함
• 따로 기본 키가 존재하지 않음

일대다 관계



다대다 관계



➢ 복잡성으로 속도를 얻다
• 복잡한 대신 데이터가 커지더라도 빠른 속도로 결과를 얻을 수 있음
• 데이터를 정규화한 뒤 정수 키로 연결함으로써, 관계형 데이터베이스가 반드시 훑어야 하는 데이터 총량은 일반적인 (관계가 없는) 데이터보다 훨씬 낮음
• 마치 거래 관계같은 것 - 데이터베이스를 디자인하는 데 시간을 들이는 대신 프로그램을 빠르게 실행할 수 있음

❖ **요약**
• 관계형 데이터베이스를 이용하면 많은 양의 데이터베이스를 다룰 수 있음

- 핵심은 어떤 데이터든 단 하나의 복사본만 가능하며, 관계를 형성하고 JOIN 연산을 이용해 여러 곳의 데이터를 연결할 수 있음
- 이를 통해 많은 양의 데이터에 대해 복잡한 연산을 할 경우 훑어야하는 데이터 크기를 대폭 줄일 수 있음
- 데이터베이스와 SQL을 디자인하는 것은 일종의 예술