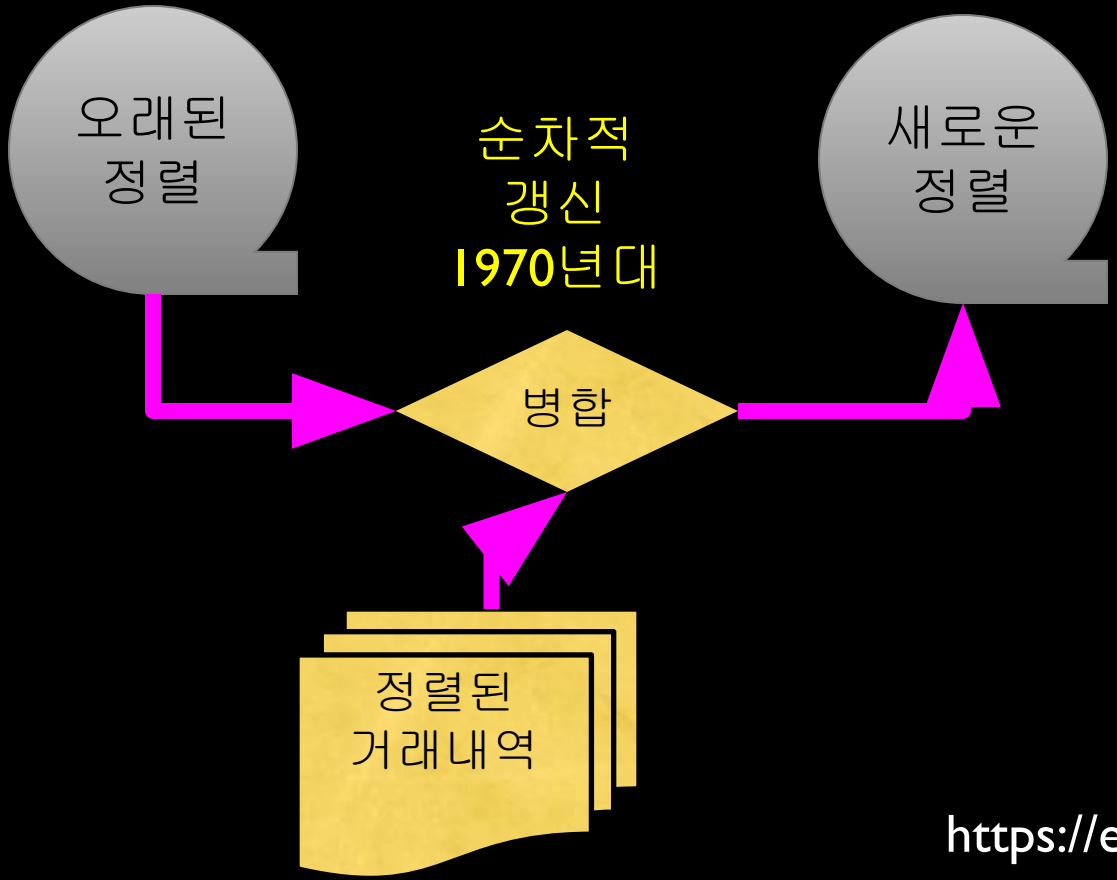
관계형 데이터베이스와 SQLite

Charles Severance

모두를 위한 파이썬

www.py4e.com/lectures3/

http://sqlitebrowser.org/



https://en.wikipedia.org/wiki/IBM_729

랜덤한 접근

- 만약 데이터에 랜덤하게 접근한다면...
- 어떻게 데이터를 효율적으로 뽑아낼까?
- 정렬이 최선은 아님

https://en.wikipedia.org/wiki/Hard_disk_drive_platter

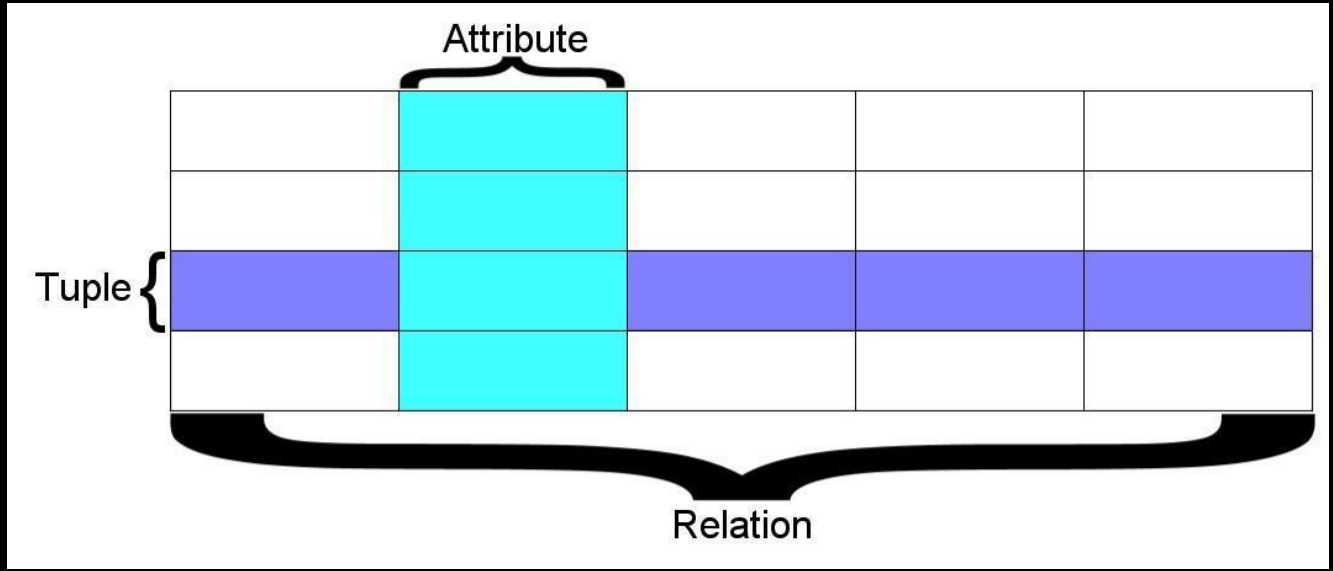
관계형 데이터베이스

- 관계형 데이터베이스는 테이블에서의 행과 열을 이용하여 데이터를 모델링
- 관계형 데이터베이스를 이용하면 단일 테이블이나 복잡하게 얽혀있는 다중 테이블에서 데이터를 효율적으로 추출해낼 수 있음

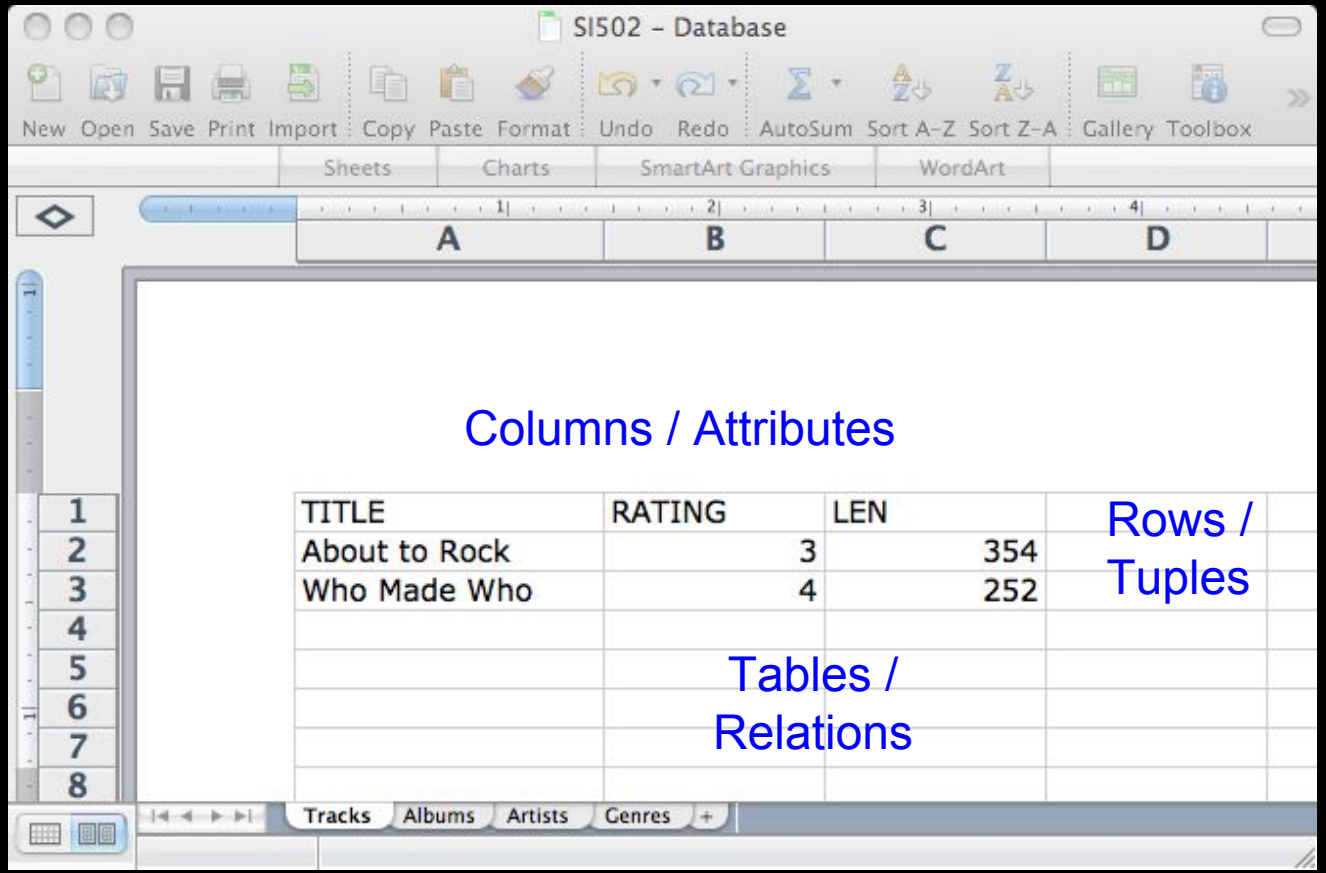
http://en.wikipedia.org/wiki/Relational_database

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



관계는 같은 속성을 갖고 있는 튜플의 집합으로 정의. 튜플은 주로 객체와 그 객체에 관한 정보를 담고 있다. 객체는 일반적으로 물리적인 물체나 개념. 관계는 주로 테이블을 이용해 나타내며 로우와 칼럼으로 이뤄져 있음. 하나의 속성이 참조하는 모든 데이터는 같은 도메인에 속하며 동일한 제약 조건을 따름. (Wikipedia)

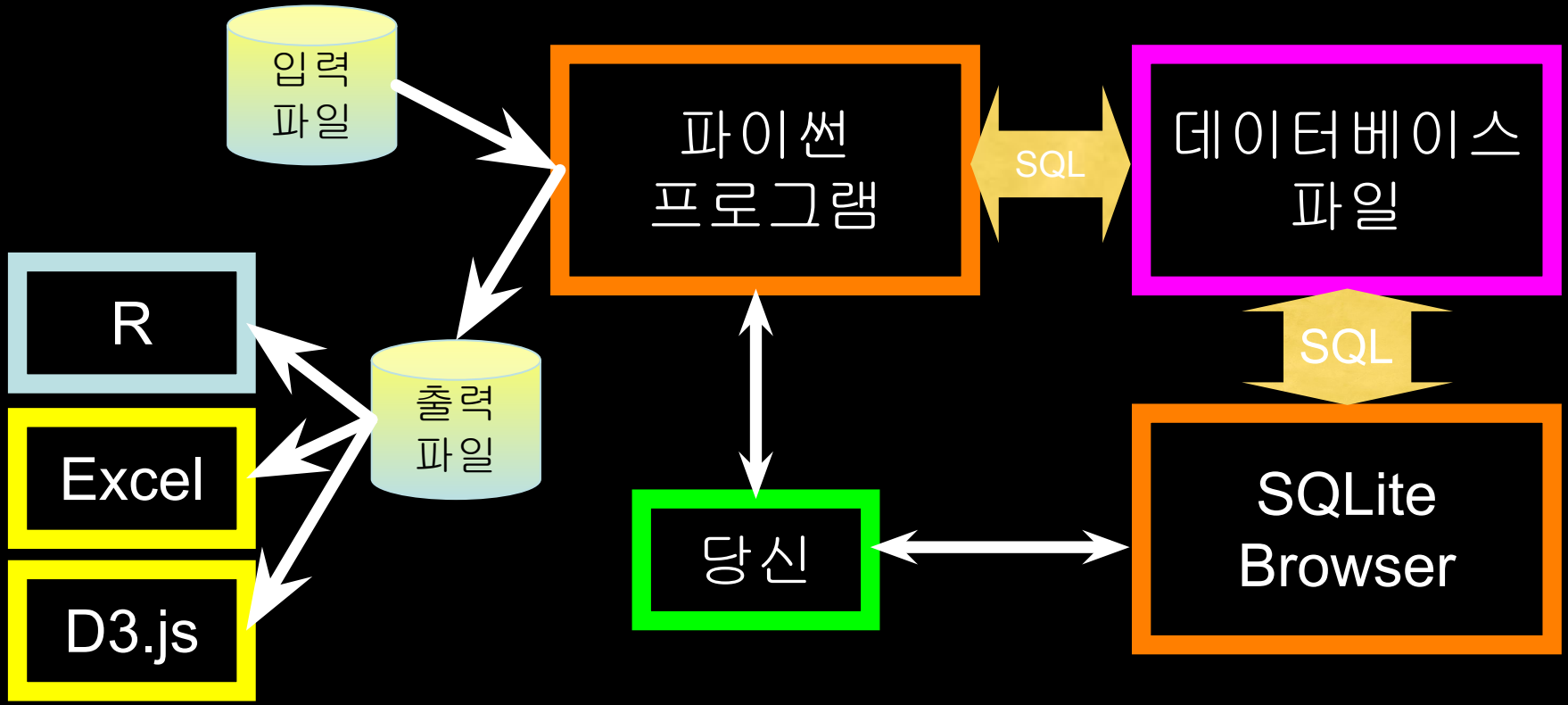

SI502 – Database
New Open Save Print Import Copy Paste Format Undo Redo AutoSum Sort A-Z Sort Z-A Gallery Toolbox
Sheets Charts SmartArt Graphics WordArt
A B C D
Columns / Attributes
TITLE RATING LEN
About to Rock 3 354
Who Made Who 4 252
Rows / Tuples
Tables / Relations
Tracks Albums Artists Genres

SQL

구조화된 쿼리 언어는 데이터베이스에 명령을 내리기 위해 우리가 사용할 언어

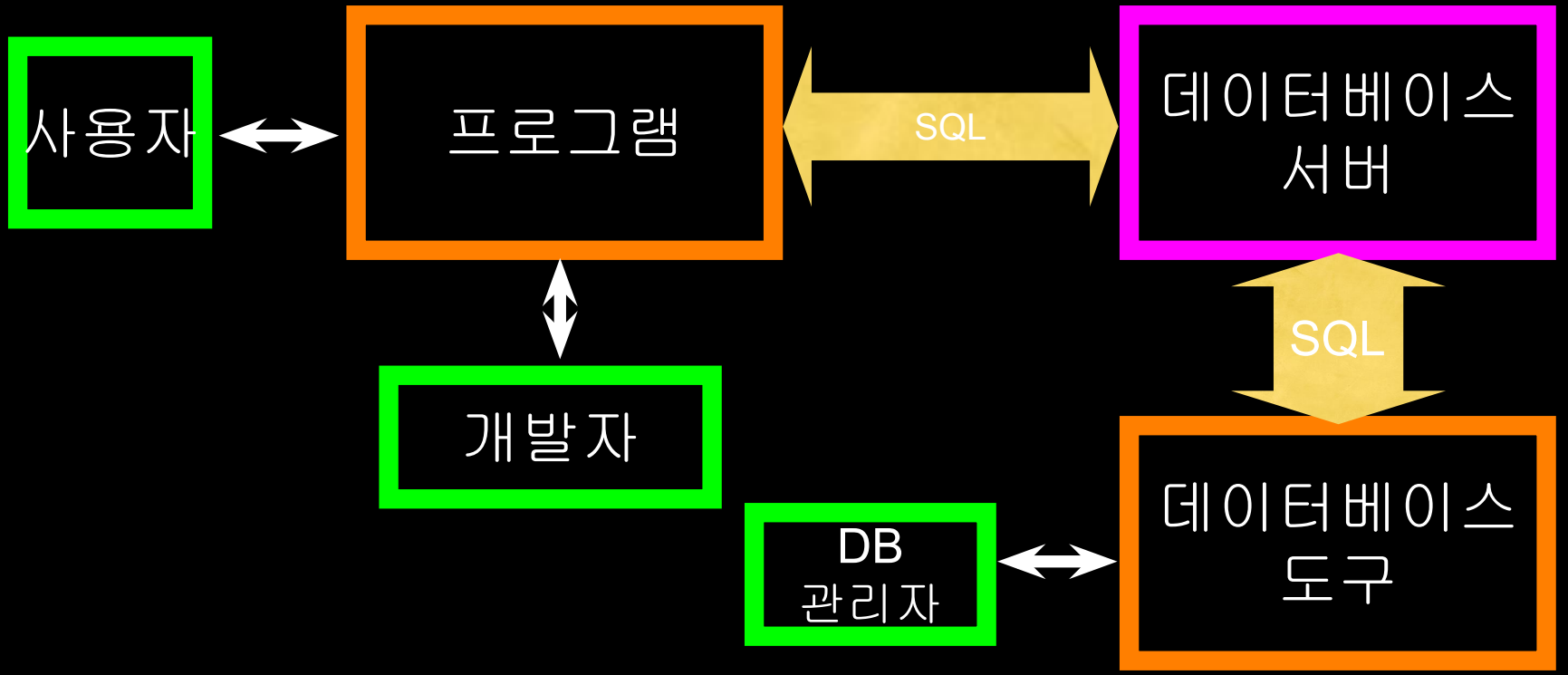
- 데이터 형성 (또는 삽입)
- 데이터 추출
- 데이터 업데이트(갱신)
- 데이터 삭제

http://en.wikipedia.org/wiki/SQL


입력
파일
파이썬
프로그램
SQL
데이터베이스
파일
R
Excel
D3.js
출력
파일
SQL
당신
SQLite
Browser

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


사용자
프로그램
SQL
데이터베이스
서버
SQL
개발자
DB
관리자
데이터베이스
도구

데이터베이스 관리자

- 데이터베이스 관리자는(DBA) 기관의 데이터베이스 디자인, 실행, 유지 및 보수를 담당하는 사람
- 데이터베이스 개발/디자인 전략, 데이터베이스의 성능과 용량 모니터링 및 개선, 추후 확장을 위한 계획 등의 역할
- 데이터베이스의 보호를 위해 보안에 관련하여 계획 및 실행을 담당하기도 함

http://en.wikipedia.org/wiki/Database_administrator

데이터베이스 모델

데이터베이스 모델 또는 데이터베이스 스키마는
데이터베이스 시스템이 지원하는 형식 언어로 표현된
데이터베이스의 구조를 지칭
다시 말해서, “데이터베이스 모델”은 데이터베이스 관리
시스템과 결합을 위해 데이터 모델을 응용한 것

http://en.wikipedia.org/wiki/Database_model

일반적인 데이터베이스 시스템

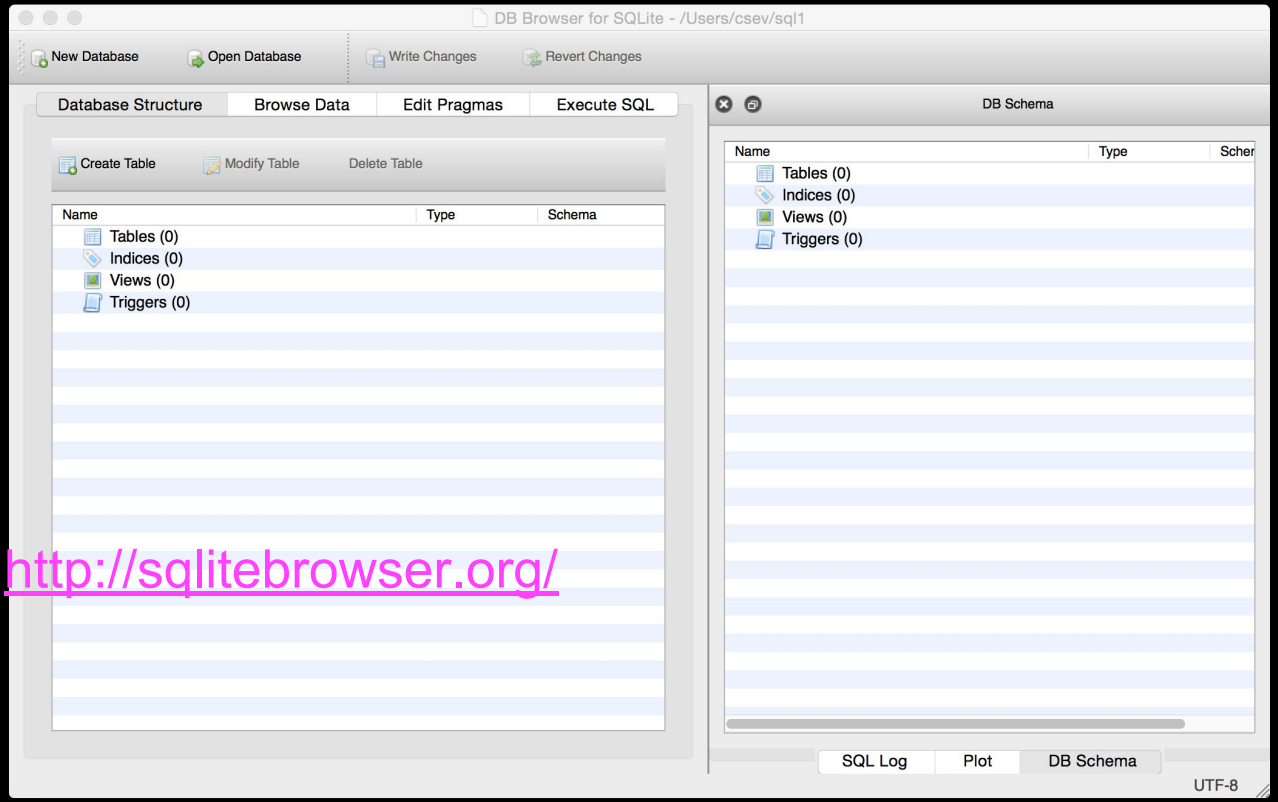
- 널리 쓰이는 세 가지 주요 데이터베이스 관리 시스템
  - Oracle - 크고 상업적이며 기업 단위로 사용. 유연함.
  - MySql - 간단하나 매우 빠르고 확장이 용이. 상업적인 오픈 소스
  - SqlServer - Microsoft가 만듦. 매우 괜찮음.
- 무료이고 오픈 소스인 소규모 프로젝트도 많음
  - HSQL, SQLite, Postgres, ...

여러 소프트웨어에서 사용되는 SQLite

http://www.sqlite.org/famous.html

SQLite Browser

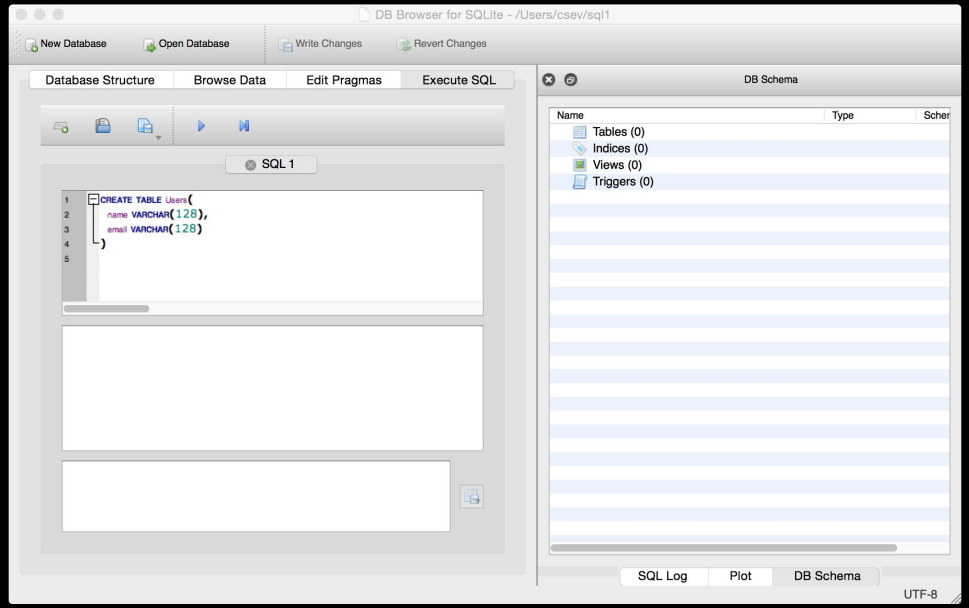
- SQLite는 매우 유명한 데이터베이스로 무료이고 빠르며 가벼움
- SQLite Browser를 이용하면 SQLite 파일을 직접 다룰 수 있음
  - http://sqlitebrowser.org/
- SQLite는 파이썬과 그 밖의 언어들에 내장되어 있음


DB Browser for SQLite - /Users/csev/sql1
New Database
Open Database
Write Changes
Revert Changes
Database Structure
Browse Data
Edit Pragmas
Execute SQL
Create Table
Modify Table
Delete Table
Name
Type
Schema
Tables (0)
Indices (0)
Views (0)
Triggers (0)
http://sqlitebrowser.org/
DB Schema
Name
Type
Scher
Tables (0)
Indices (0)
Views (0)
Triggers (0)
SQL Log
Plot
DB Schema
UTF-8

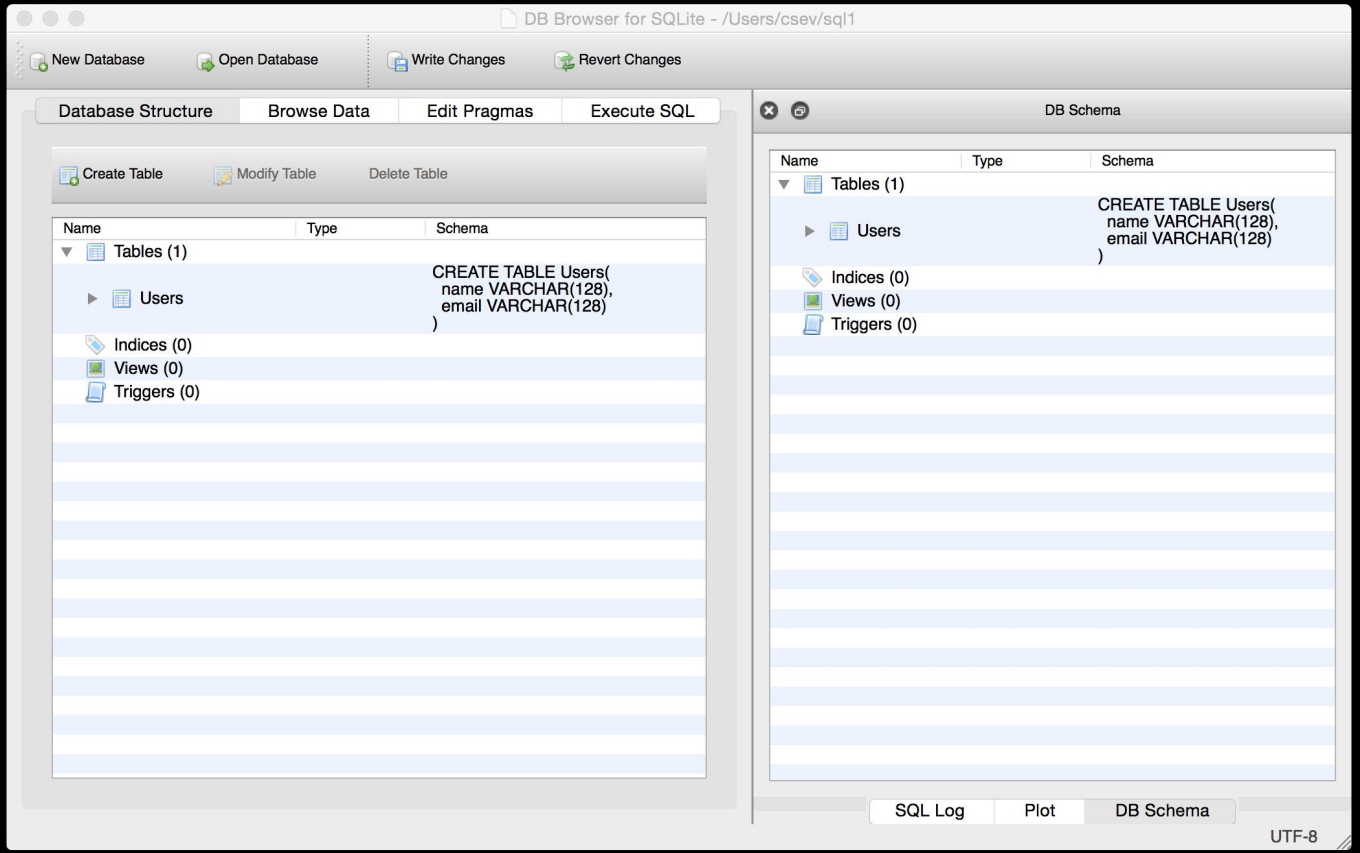
데이터베이스를 만들어 봅시다

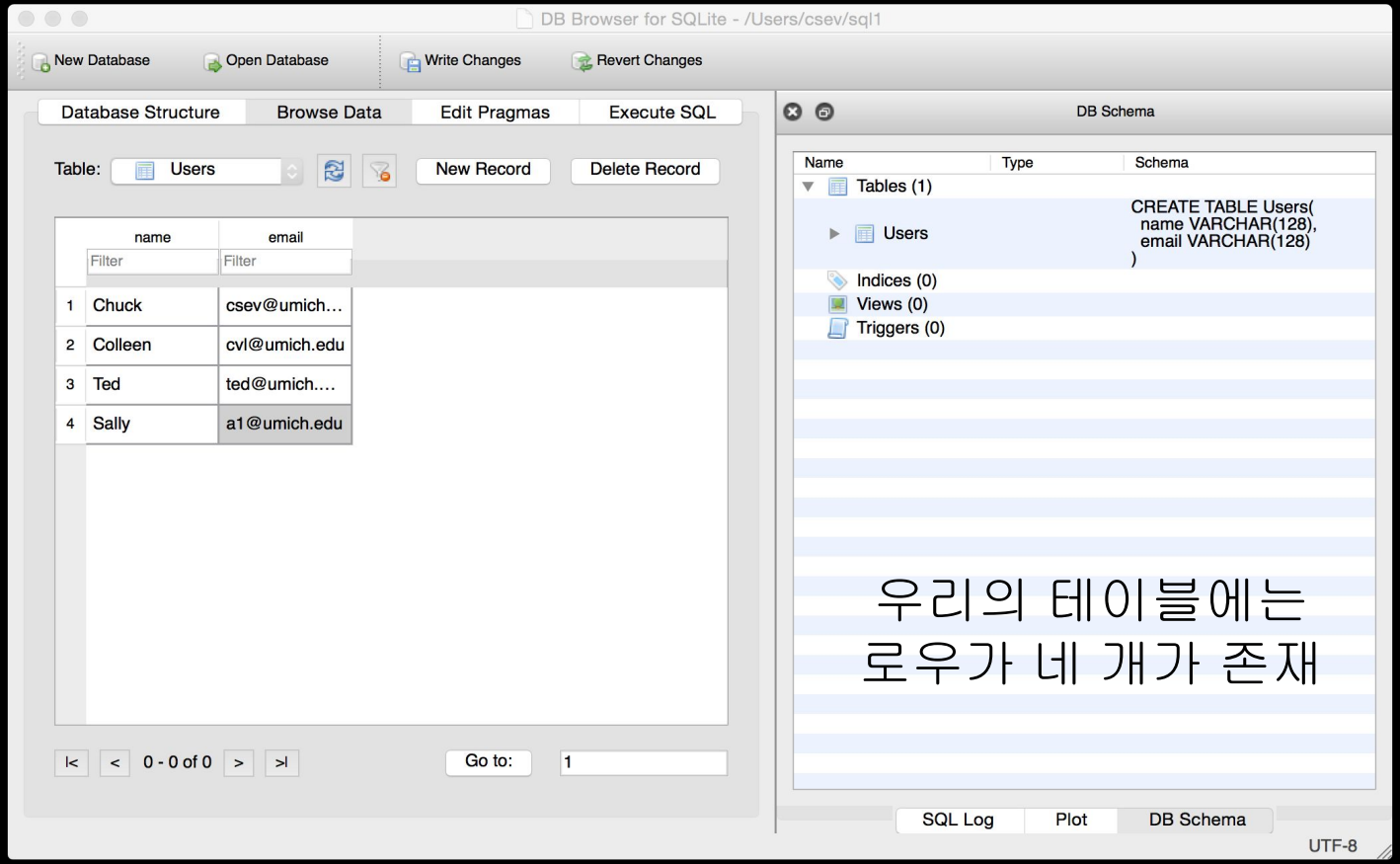
https://www.py4e.com/lectures3/Pythonlearn-15-Database-Handout.txt

시작은 간단히 - 단일 테이블




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


DB Browser for SQLite - /Users/csev/sql1
New Database
Open Database
Write Changes
Revert Changes
Database Structure
Browse Data
Edit Pragmas
Execute SQL
Create Table
Modify Table
Delete Table
Name
Type
Schema
Tables (1)
Users
CREATE TABLE Users(
name VARCHAR(128),
email VARCHAR(128)
)
Indices (0)
Views (0)
Triggers (0)
DB Schema
Name
Type
Schema
Tables (1)
Users
CREATE TABLE Users(
name VARCHAR(128),
email VARCHAR(128)
)
Indices (0)
Views (0)
Triggers (0)
SQL Log
Plot
DB Schema
UTF-8


DB Browser for SQLite - /Users/csev/sql1
New Database
Open Database
Write Changes
Revert Changes
Database Structure
Browse Data
Edit Pragmas
Execute SQL
Table:
Users
New Record
Delete Record
name
email
Filter
Filter
1 Chuck csev@umich…
2 Colleen cvl@umich.edu
3 Ted ted@umich.…
4 Sally a1@umich.edu
|<
<
0 - 0 of 0
>
>|
Go to:
1
DB Schema
Name
Type
Schema
Tables (1)
Users
CREATE TABLE Users(
name VARCHAR(128),
email VARCHAR(128)
)
Indices (0)
Views (0)
Triggers (0)
우리의 테이블에는
로우가 네 개가 존재
SQL Log
Plot
DB Schema
UTF-8

SQL

구조화된 쿼리 언어는 데이터베이스에 명령을 내리기 위해 우리가 사용할 언어

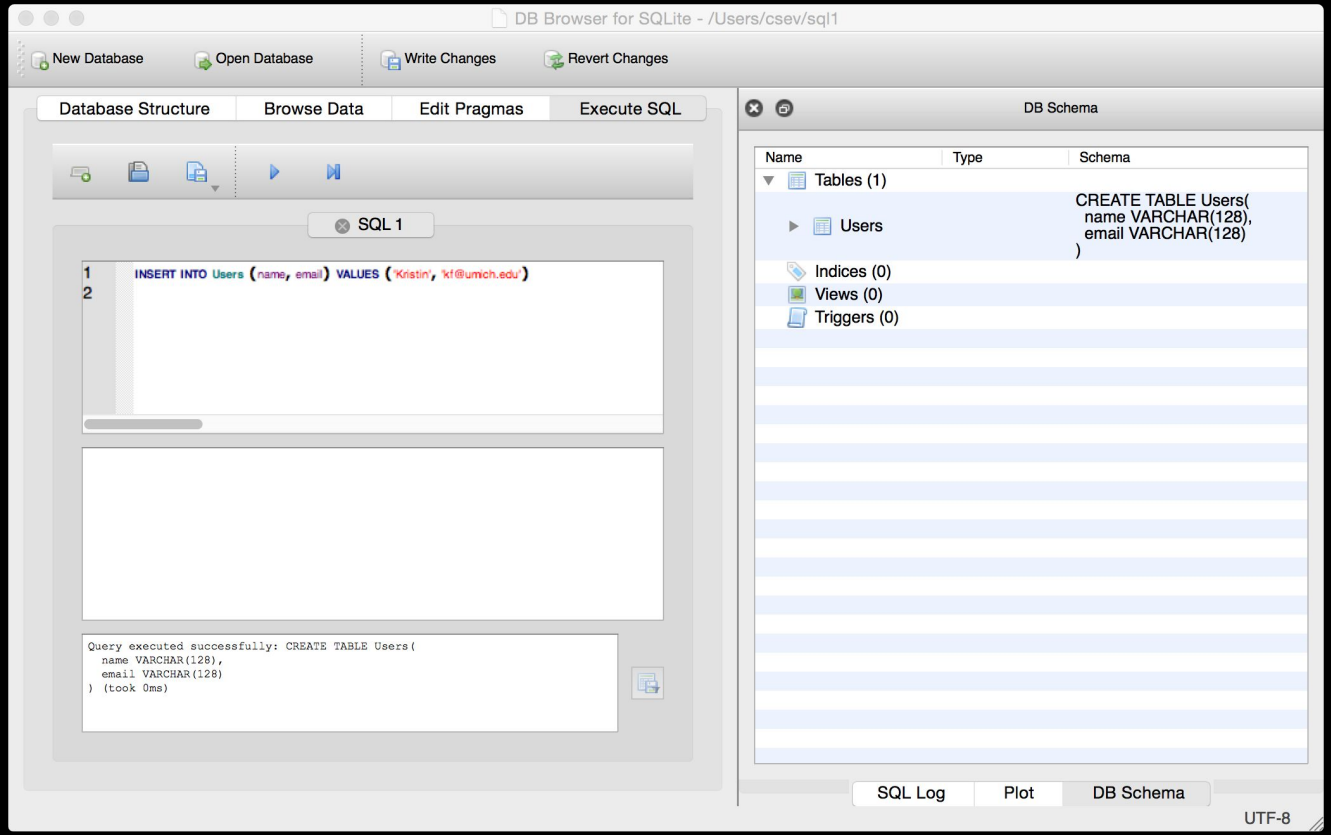
- 데이터 형성 (또는 삽입)
- 데이터 추출
- 데이터 업데이트(갱신)
- 데이터 삭제

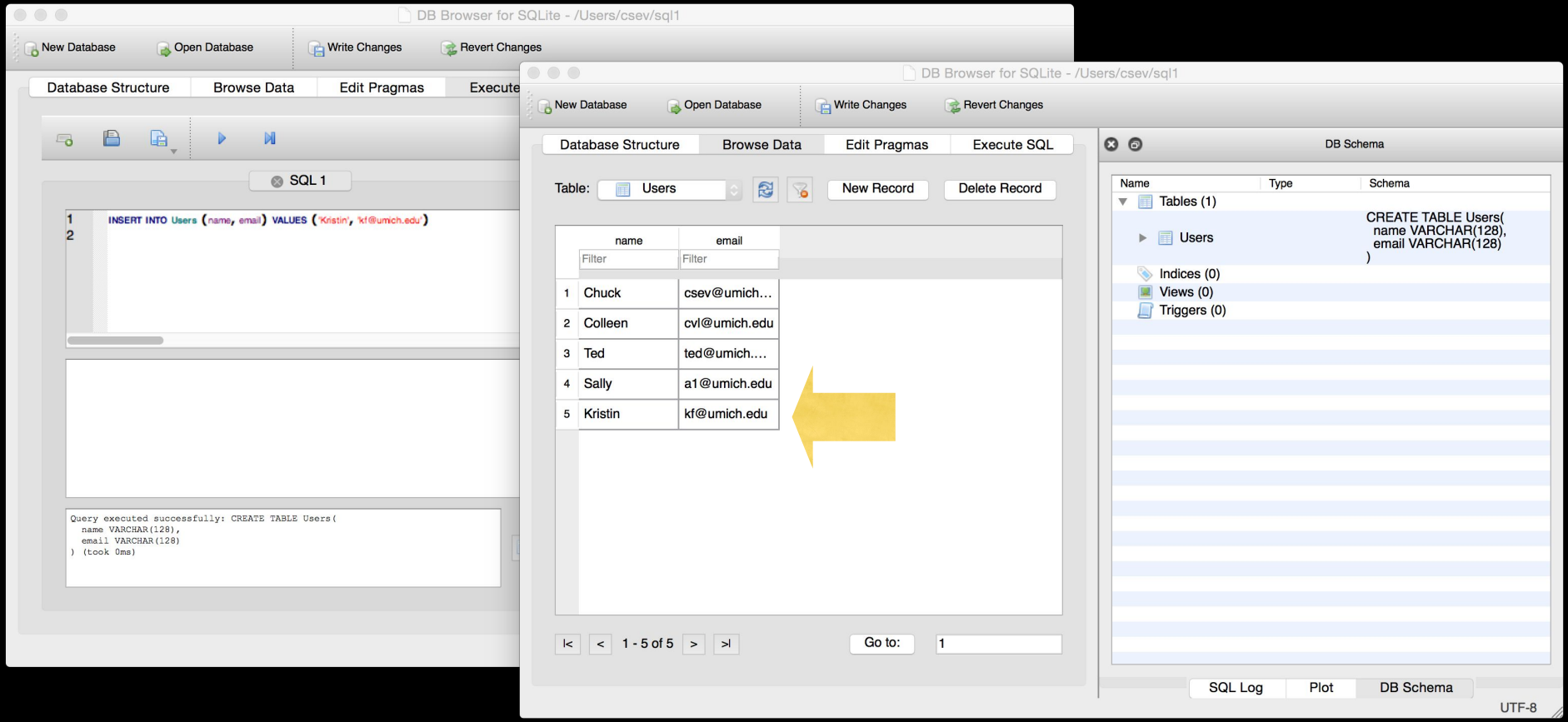
http://en.wikipedia.org/wiki/SQL

SQL: INSERT

INSERT문은 테이블에 로우를 삽입

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

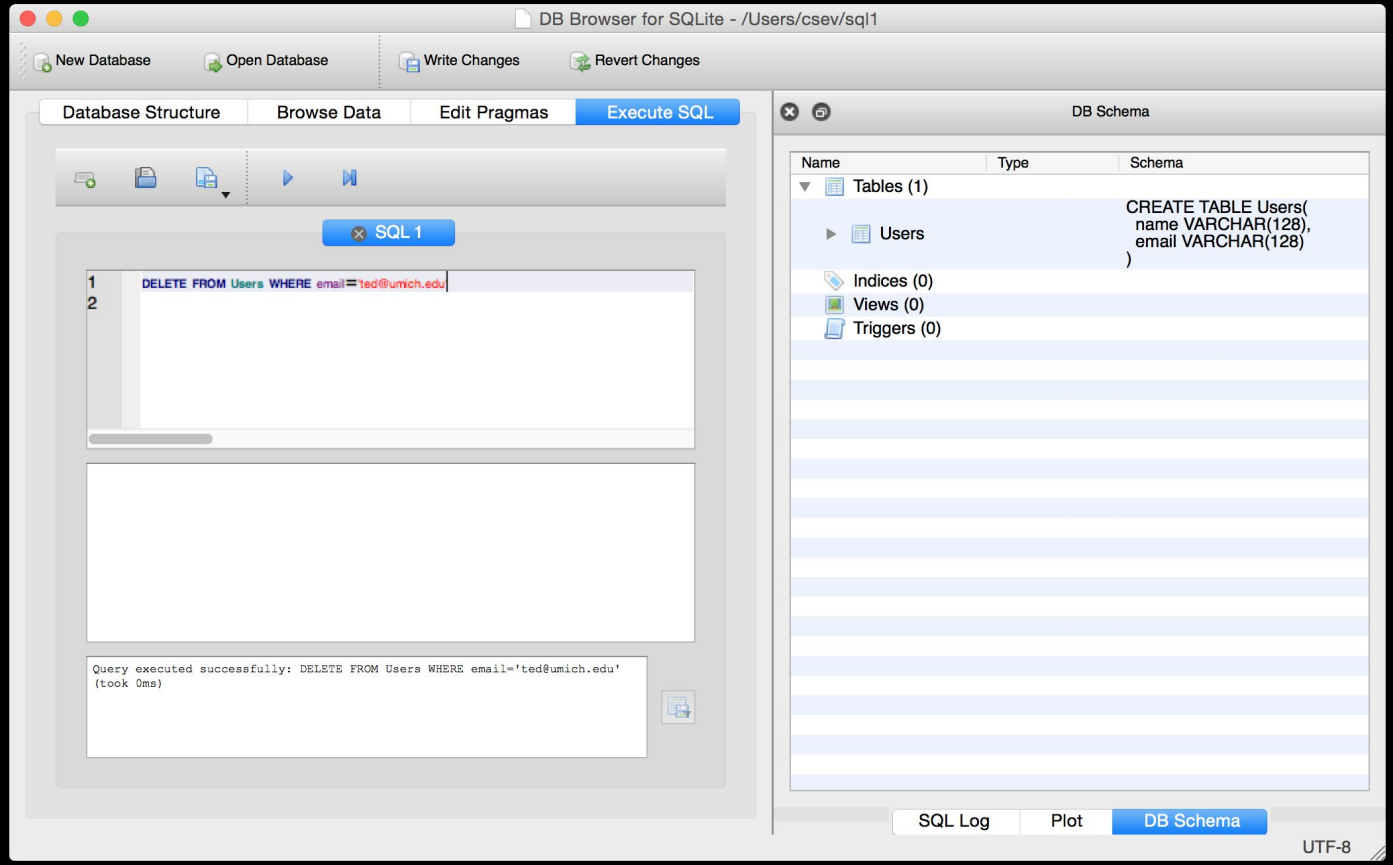

DB Browser for SQLite - /Users/csev/sql1
New Database
Open Database
Write Changes
Revert Changes
Database Structure
Browse Data
Edit Pragmas
Execute SQL
SQL 1
1 INSERT INTO Users (name, email) VALUES ('Kristin', 'kf@umich.edu')
2
Query executed successfully: CREATE TABLE Users(
name VARCHAR(128),
email VARCHAR(128)
) (took 0ms)
DB Schema
Name
Type
Schema
Tables (1)
Users
CREATE TABLE Users(
name VARCHAR(128),
email VARCHAR(128)
)
Indices (0)
Views (0)
Triggers (0)
SQL Log
Plot
DB Schema
UTF-8

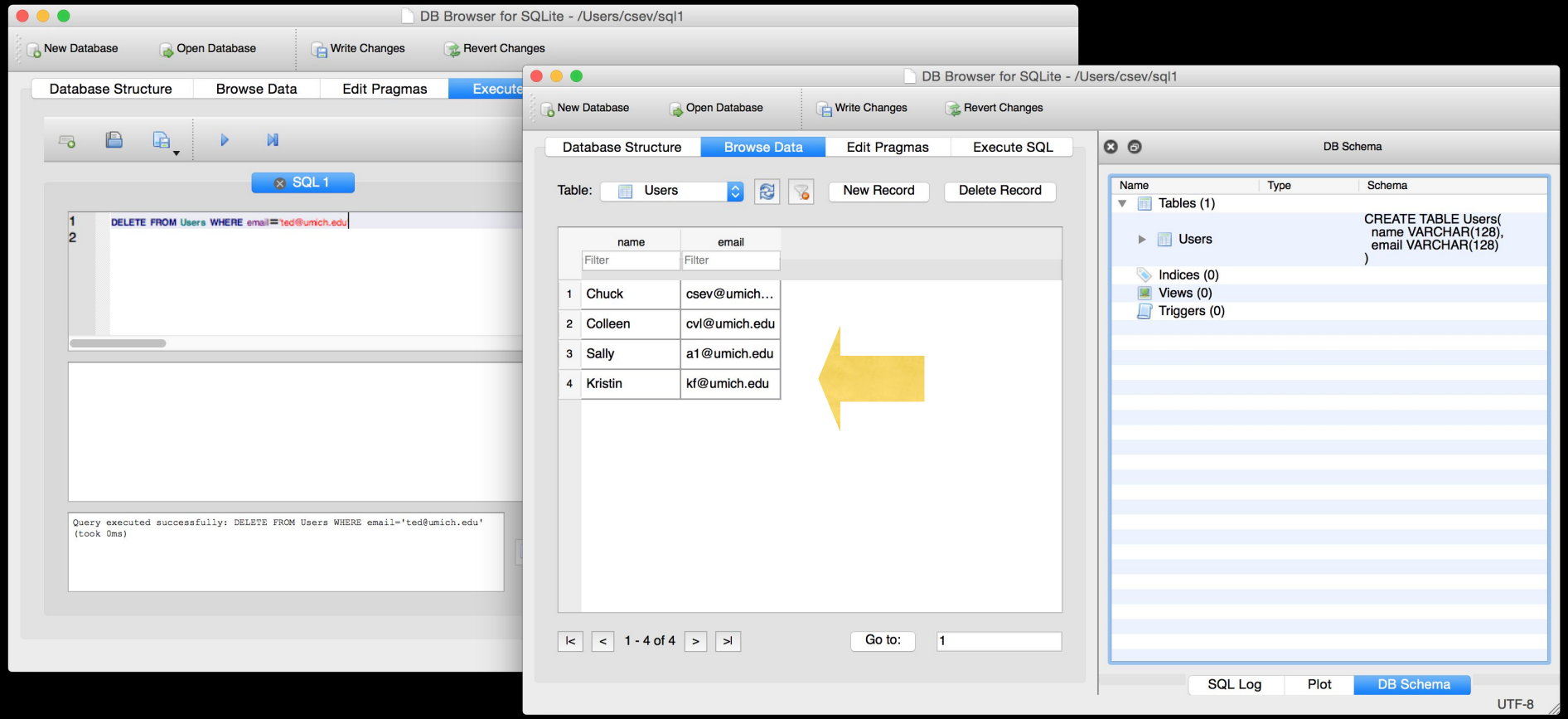

DB Browser for SQLite - /Users/csev/sql1
New Database
Open Database
Write Changes
Revert Changes
Database Structure
Browse Data
Edit Pragmas
Execute SQL
SQL 1
1 INSERT INTO Users (name, email) VALUES ('Kristin', 'kf@umich.edu')
2
Query executed successfully: CREATE TABLE Users(
name VARCHAR(128),
email VARCHAR(128)
) (took 0ms)
DB Browser for SQLite - /Users/csev/sql1
New Database
Open Database
Write Changes
Revert Changes
Database Structure
Browse Data
Edit Pragmas
Execute SQL
Table: Users
New Record
Delete Record
name
email
Filter
Filter
1 Chuck csev@umich...
2 Colleen cvl@umich.edu
3 Ted ted@umich....
4 Sally a1@umich.edu
5 Kristin kf@umich.edu
|< < 1 - 5 of 5 > >|
Go to: 1
DB Schema
Name
Type
Schema
Tables (1)
Users
CREATE TABLE Users(
name VARCHAR(128),
email VARCHAR(128)
)
Indices (0)
Views (0)
Triggers (0)
SQL Log
Plot
DB Schema
UTF-8

SQL: DELETE

DELETE문은 테이블에서 조건을 만족하는 로우를 삭제

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

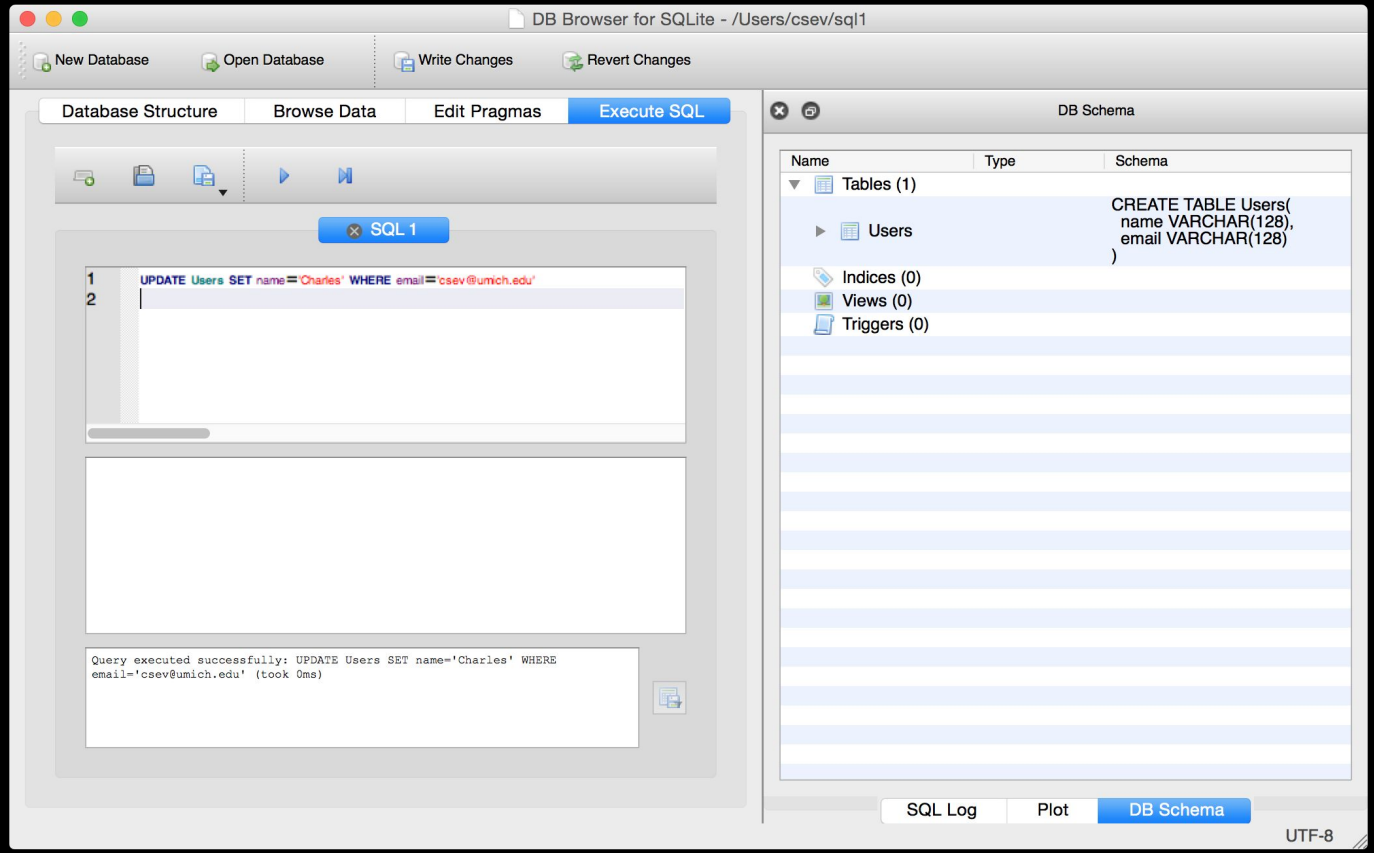

DB Browser for SQLite - /Users/csev/sql1
New Database
Open Database
Write Changes
Revert Changes
Database Structure
Browse Data
Edit Pragmas
Execute SQL
SQL 1
1 DELETE FROM Users WHERE email='ted@umich.edu'
2
Query executed successfully: DELETE FROM Users WHERE email='ted@umich.edu' (took 0ms)
DB Schema
Name
Type
Schema
Tables (1)
Users
CREATE TABLE Users(name VARCHAR(128), email VARCHAR(128))
Indices (0)
Views (0)
Triggers (0)
SQL Log
Plot
DB Schema
UTF-8

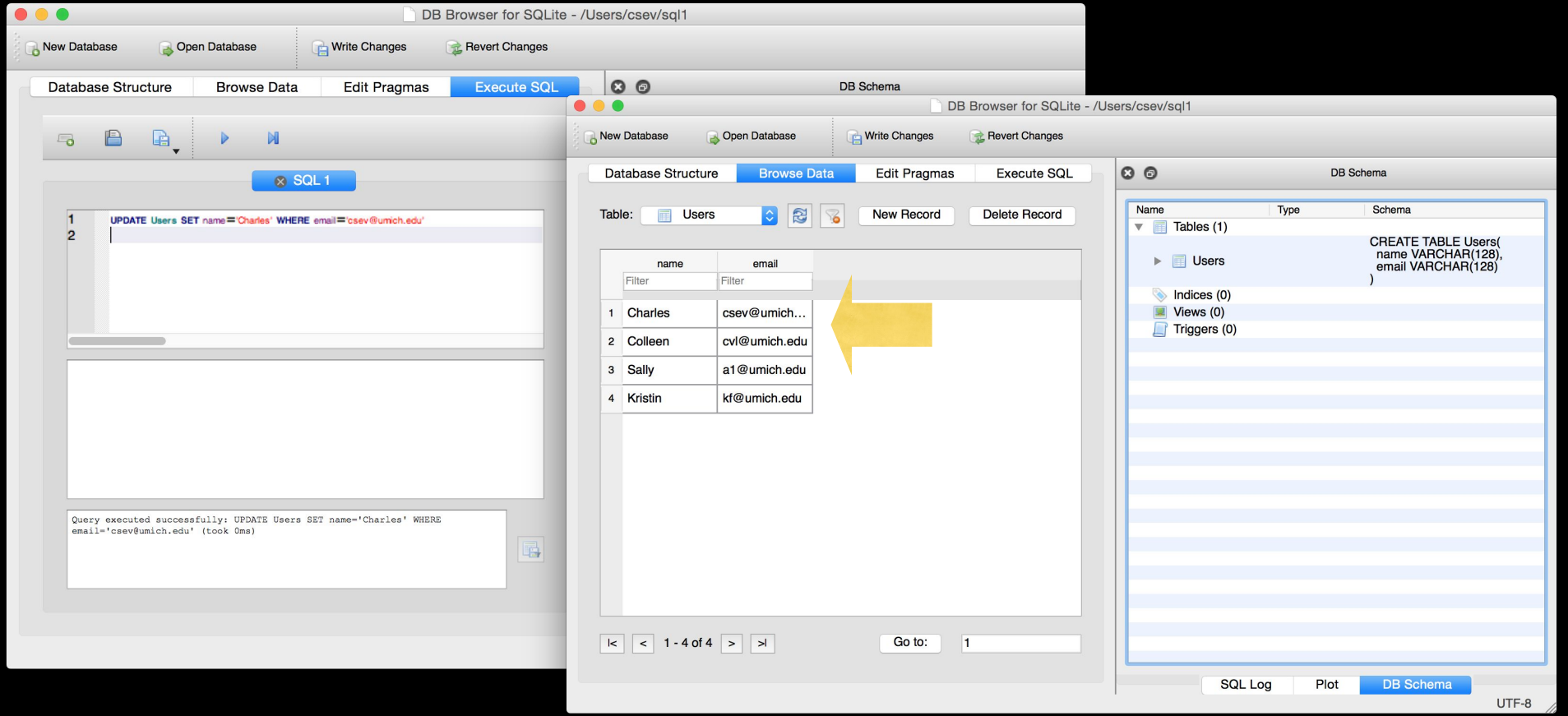

DB Browser for SQLite - /Users/csev/sql1
New Database
Open Database
Write Changes
Revert Changes
Database Structure
Browse Data
Edit Pragmas
Execute
SQL 1
1 DELETE FROM Users WHERE email='ted@umich.edu'
2
Query executed successfully: DELETE FROM Users WHERE email='ted@umich.edu' (took 0ms)
DB Browser for SQLite - /Users/csev/sql1
New Database
Open Database
Write Changes
Revert Changes
Database Structure
Browse Data
Edit Pragmas
Execute SQL
Table: Users
New Record
Delete Record
name
email
Filter
Filter
1 Chuck csev@umich...
2 Colleen cvl@umich.edu
3 Sally a1@umich.edu
4 Kristin kf@umich.edu
|< < 1 - 4 of 4 > >|
Go to: 1
DB Schema
Name
Type
Schema
Tables (1)
Users
CREATE TABLE Users(name VARCHAR(128), email VARCHAR(128))
Indices (0)
Views (0)
Triggers (0)
SQL Log
Plot
DB Schema
UTF-8

SQL: UPDATE

WHERE절과 함께 쓰이며 필드를 갱신

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


DB Browser for SQLite - /Users/csev/sql1
New Database
Open Database
Write Changes
Revert Changes
Database Structure
Browse Data
Edit Pragmas
Execute SQL
SQL 1
1 UPDATE Users SET name='Charles' WHERE email='csev@umich.edu'
2
Query executed successfully: UPDATE Users SET name='Charles' WHERE email='csev@umich.edu' (took 0ms)
DB Schema
Name
Type
Schema
Tables (1)
Users
CREATE TABLE Users(name VARCHAR(128), email VARCHAR(128))
Indices (0)
Views (0)
Triggers (0)
SQL Log
Plot
DB Schema
UTF-8

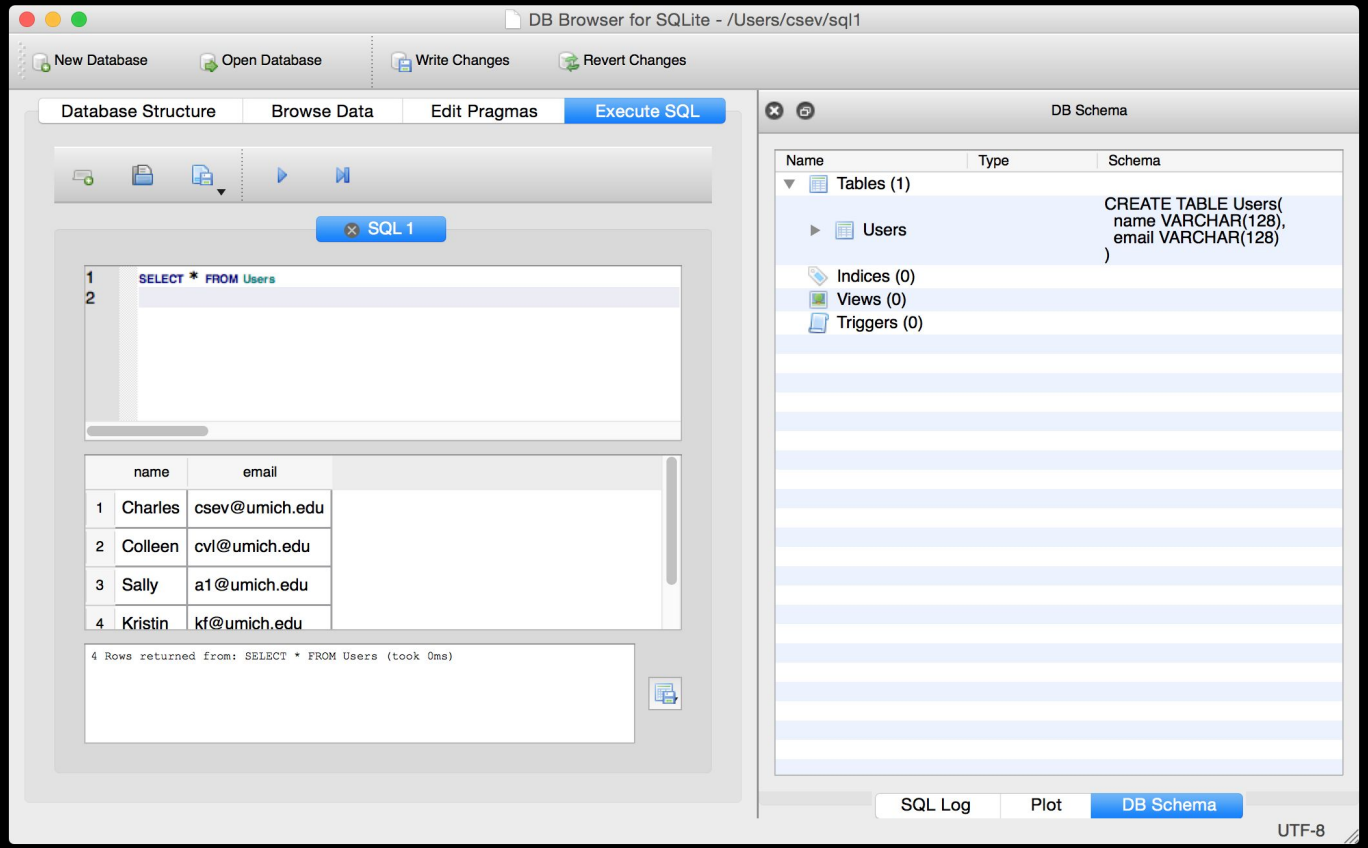

DB Browser for SQLite - /Users/csev/sql1
New Database
Open Database
Write Changes
Revert Changes
Database Structure
Browse Data
Edit Pragmas
Execute SQL
DB Schema
SQL 1
1 UPDATE Users SET name='Charles' WHERE email='csev@umich.edu'
2
Query executed successfully: UPDATE Users SET name='Charles' WHERE email='csev@umich.edu' (took 0ms)
DB Browser for SQLite - /Users/csev/sql1
New Database
Open Database
Write Changes
Revert Changes
Database Structure
Browse Data
Edit Pragmas
Execute SQL
DB Schema
Table: Users
New Record
Delete Record
name
email
Filter
Filter
1 Charles csev@umich...
2 Colleen cvl@umich.edu
3 Sally a1@umich.edu
4 Kristin kf@umich.edu
|< < 1 - 4 of 4 > >|
Go to: 1
Name
Type
Schema
Tables (1)
Users
CREATE TABLE Users(name VARCHAR(128), email VARCHAR(128))
Indices (0)
Views (0)
Triggers (0)
SQL Log
Plot
DB Schema
UTF-8

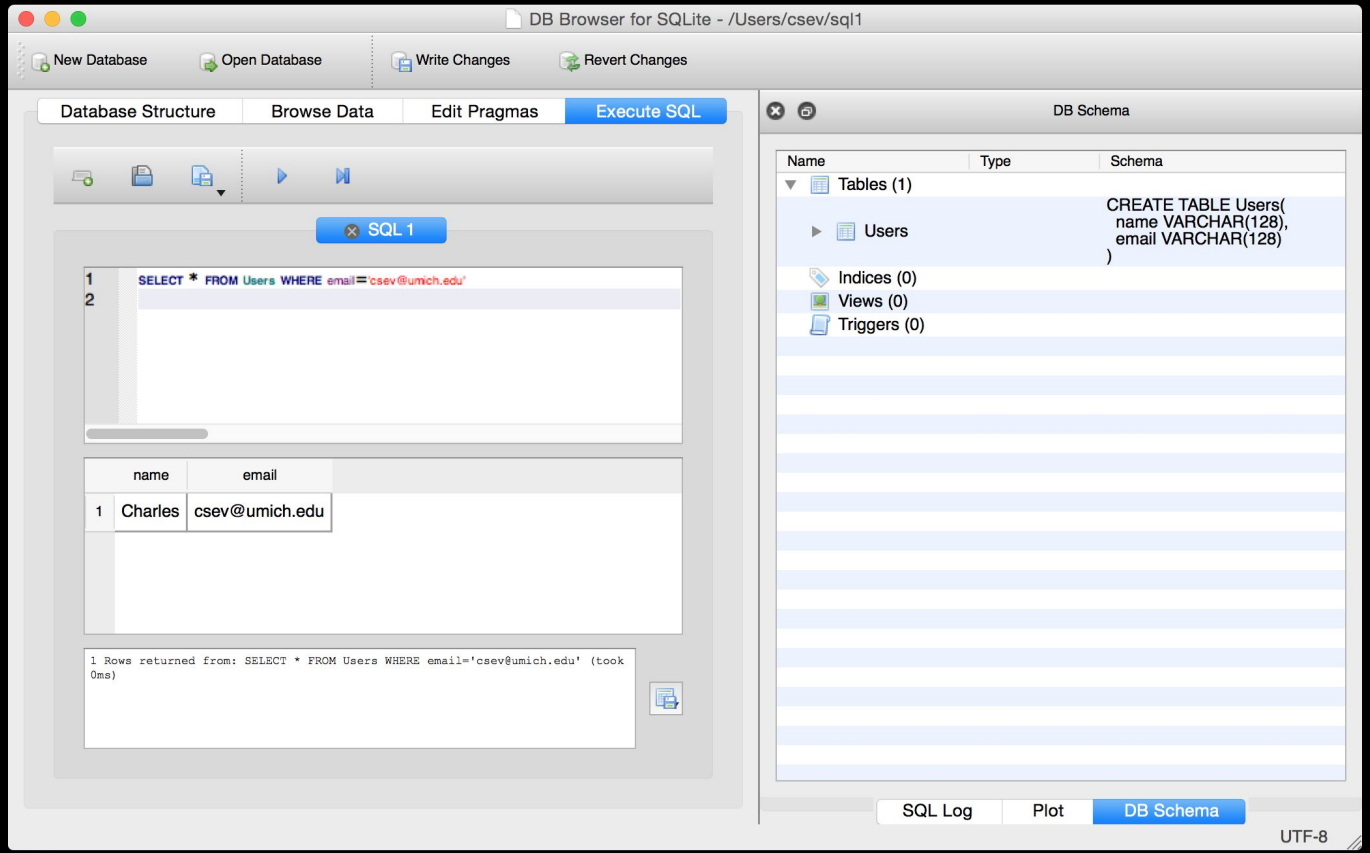
레코드 추출: SELECT

- SELECT문은 레코드 여러 개를 추출
- 모든 레코드를 추출하거나, WHERE절과 함께 쓰여서 일부 레코드만 추출

```
SELECT * FROM Users
```

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


DB Browser for SQLite - /Users/csev/sql1
New Database
Open Database
Write Changes
Revert Changes
Database Structure
Browse Data
Edit Pragmas
Execute SQL
SQL 1
1 SELECT * FROM Users
2
name email
1 Charles csev@umich.edu
2 Colleen cvl@umich.edu
3 Sally a1@umich.edu
4 Kristin kf@umich.edu
4 Rows returned from: SELECT * FROM Users (took 0ms)
DB Schema
Name Type Schema
Tables (1)
Users
CREATE TABLE Users(
name VARCHAR(128),
email VARCHAR(128)
)
Indices (0)
Views (0)
Triggers (0)
SQL Log
Plot
DB Schema
UTF-8

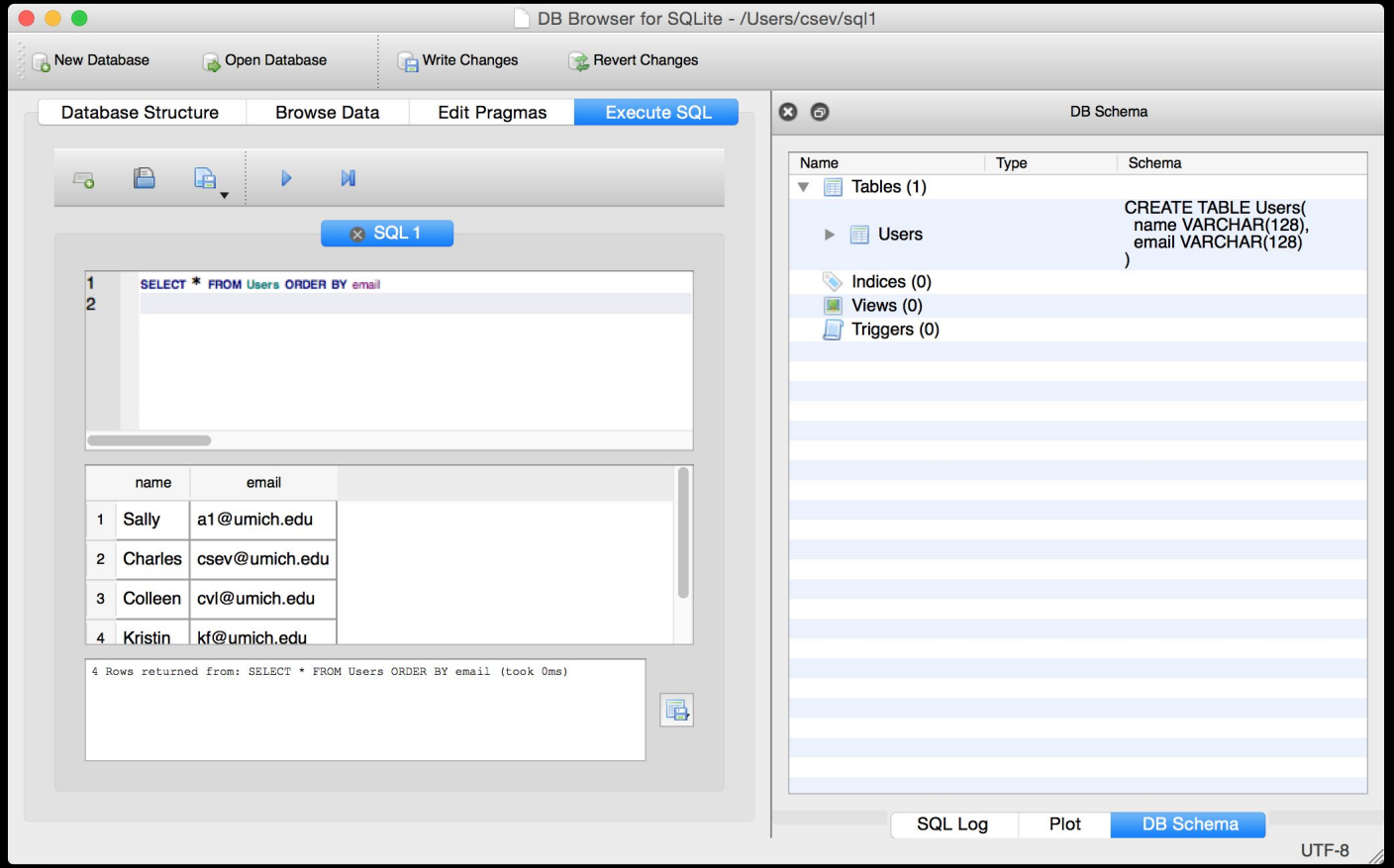

DB Browser for SQLite - /Users/csev/sql1
New Database
Open Database
Write Changes
Revert Changes
Database Structure
Browse Data
Edit Pragmas
Execute SQL
SQL 1
1 SELECT * FROM Users WHERE email='csev@umich.edu'
2
name
email
1
Charles
csev@umich.edu
1 Rows returned from: SELECT * FROM Users WHERE email='csev@umich.edu' (took 0ms)
DB Schema
Name
Type
Schema
Tables (1)
Users
CREATE TABLE Users(name VARCHAR(128), email VARCHAR(128))
Indices (0)
Views (0)
Triggers (0)
SQL Log
Plot
DB Schema
UTF-8

ORDER BY를 이용한 정렬

SELECT문에 ORDER BY절을 추가하여서 오름차순/내림차순으로 정렬된 결과를 얻을 수 있음.

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

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


DB Browser for SQLite - /Users/csev/sql1
New Database
Open Database
Write Changes
Revert Changes
Database Structure
Browse Data
Edit Pragmas
Execute SQL
SQL 1
1 SELECT * FROM Users ORDER BY email
2
name email
1 Sally a1@umich.edu
2 Charles csev@umich.edu
3 Colleen cvl@umich.edu
4 Kristin kf@umich.edu
4 Rows returned from: SELECT * FROM Users ORDER BY email (took 0ms)
DB Schema
Name Type Schema
Tables (1)
Users
CREATE TABLE Users(
name VARCHAR(128),
email VARCHAR(128)
)
Indices (0)
Views (0)
Triggers (0)
SQL Log
Plot
DB Schema
UTF-8

SQL 요약

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

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

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

SELECT * FROM Users

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

SELECT * FROM Users ORDER BY email
```

(지금까지는) 재미 없음

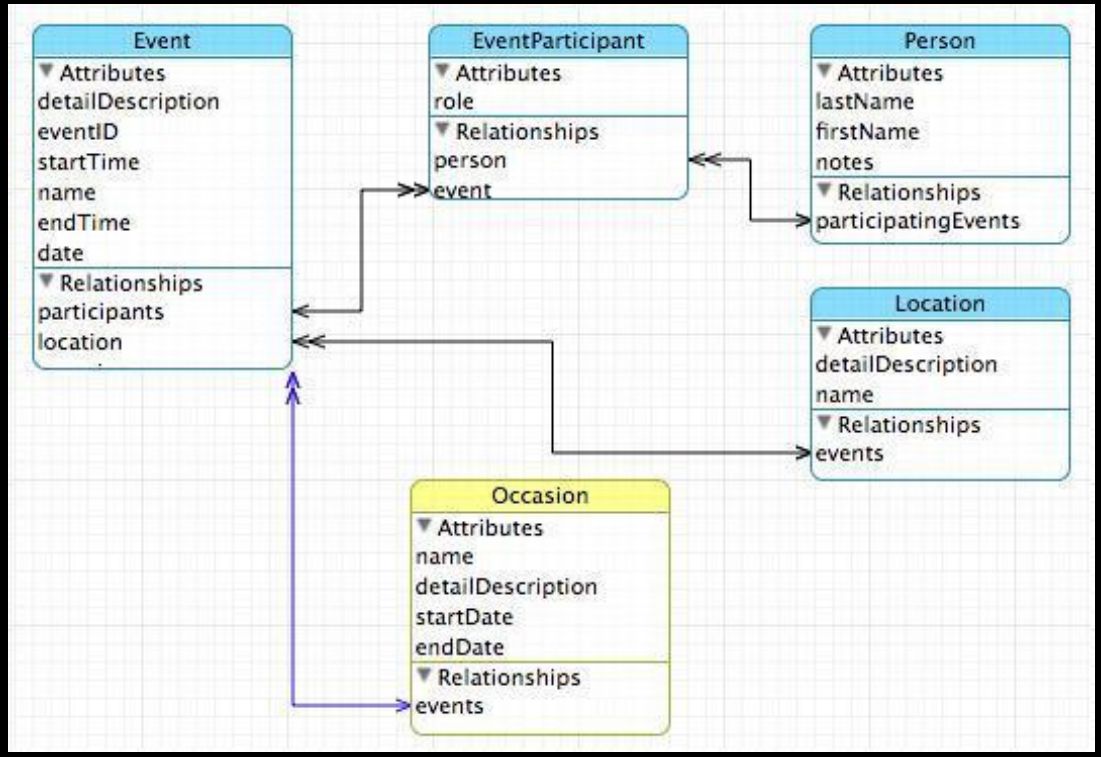
- 테이블은 로우, 칼럼, 그리고 명령문으로 빠르게 프로그래밍할 수 있는 엑셀처럼 보임
- 여러 개의 테이블을 이용해 관계를 이용하면 그 위력을 느낄 수 있을 것임

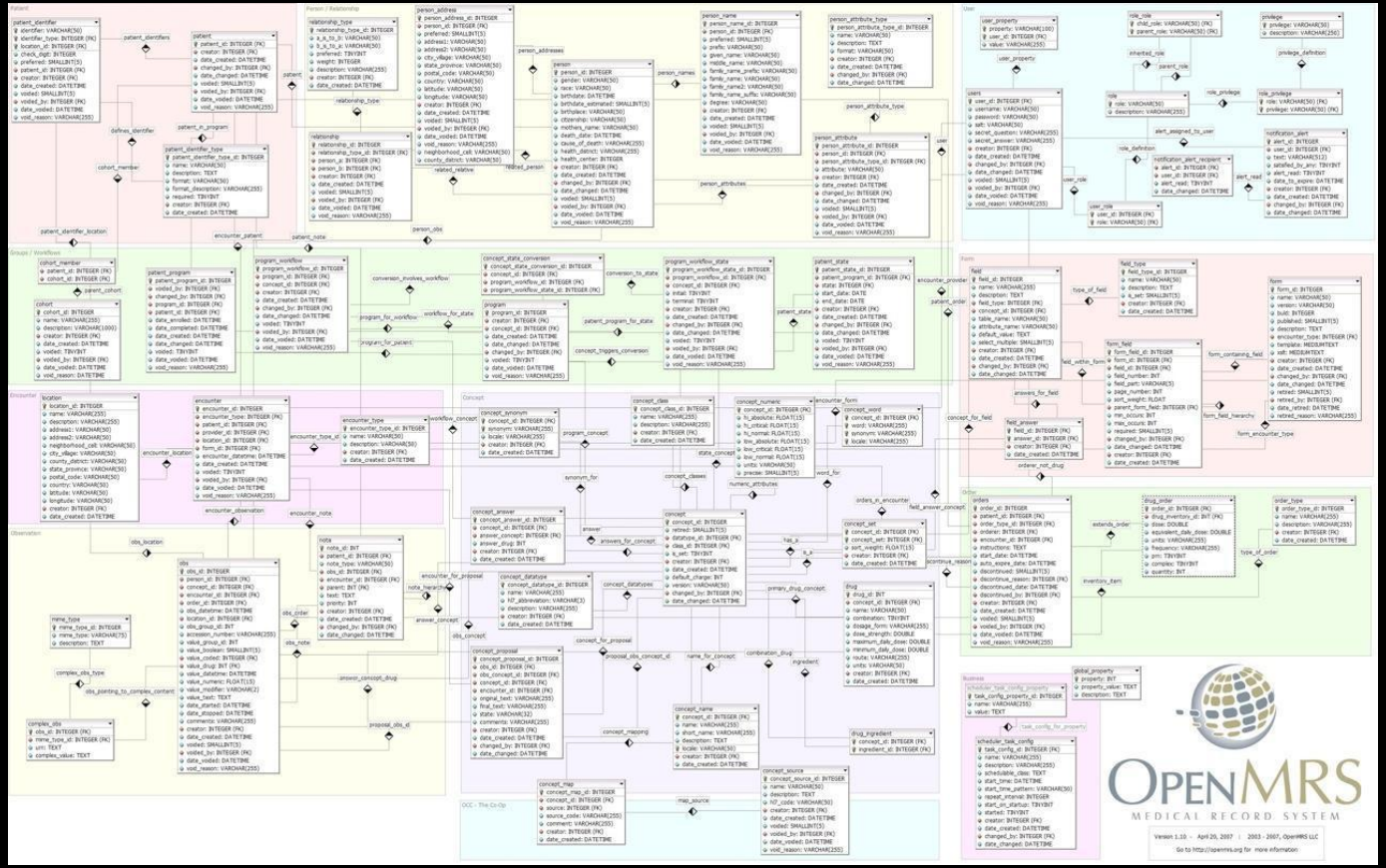
복잡한 데이터 모델과 관계

http://en.wikipedia.org/wiki/Relational_model

데이터베이스 디자인

- 데이터베이스 디자인은 특정 능력과 경험이 빚어내는 예술
- 우리의 목표는 치명적인 실수를 피하고 깔끔하며 알아보기 쉬운 데이터베이스를 디자인하는 것
- 성능 조정 등은 차후의 일
- 데이터베이스 디자인은 그림을 그리며 시작


Event
Attributes
detailDescription
eventID
startTime
name
endTime
date
Relationships
participants
location
EventParticipant
Attributes
role
Relationships
person
event
Person
Attributes
lastName
firstName
notes
Relationships
participatingEvents
Location
Attributes
detailDescription
name
Relationships
events
Occasion
Attributes
name
detailDescription
startDate
endDate
Relationships
events


OpenMRS
MEDICAL RECORD SYSTEM
Version 1.10 - April 20, 2007 | 2003 - 2007, OpenMRS LLC
Go to http://openmrs.org for more information

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

Track	Len	Artist	Album	Genre	Rating	Count
Hells Bells	5:13	AC/DC	Who Made Who	Rock	★★★★★	61
Shake Your Foundations	3:54	AC/DC	Who Made Who	Rock	★★★★★	70
Chase the Ace	3:01	AC/DC	Who Made Who	Rock		56
For Those About To Rock (We ...	5:54	AC/DC	Who Made Who	Rock	★★★★★	61
Dúlamán	3:43	Altan	Natural Wonders M...	New Age		31
Rode Across the Desert	4:10	America	Greatest Hits	Easy Listen...	★★★★★	23
Now You Are Gone	3:08	America	Greatest Hits	Easy Listen...	★★★★★	18
Tin Man	3:30	America	Greatest Hits	Easy Listen...	★★★★★	23
Sister Golden Hair	3:22	America	Greatest Hits	Easy Listen...	★★★★★	24
Track 01	4:22	Billy Price	Danger Zone	Blues/R&B	★★★★★	26
Track 02	2:45	Billy Price	Danger Zone	Blues/R&B	★★★★★	18
Track 03	3:26	Billy Price	Danger Zone	Blues/R&B	★★★★★	22
Track 04	4:17	Billy Price	Danger Zone	Blues/R&B	★★★★★	18
Track 05	3:50	Billy Price	Danger Zone	Blues/R&B	★★★★★	21
War Pigs/Luke's Wall	7:58	Black Sabbath	Paranoid	Metal	★★★★★	25
Paranoid	2:53	Black Sabbath	Paranoid	Metal	★★★★★	22
Planet Caravan	4:35	Black Sabbath	Paranoid	Metal	★★★★★	25
Iron Man	5:59	Black Sabbath	Paranoid	Metal	★★★★★	26
Electric Funeral	4:53	Black Sabbath	Paranoid	Metal	★★★★★	22
Hand of Doom	7:10	Black Sabbath	Paranoid	Metal	★★★★★	23
Rat Salad	2:30	Black Sabbath	Paranoid	Metal	★★★★★	31
Jack the Stripper/Fairies Wear ...	6:14	Black Sabbath	Paranoid	Metal	★★★★★	24
Bomb Squad (TECH)	3:28	Brent	Brent's Album			1
clay techno	4:36	Brent	Brent's Album			2
Heavy	3:08	Brent	Brent's Album			1
Hi metal man	4:20	Brent	Brent's Album			1
Mistro	2:58	Brent	Brent's Album			1

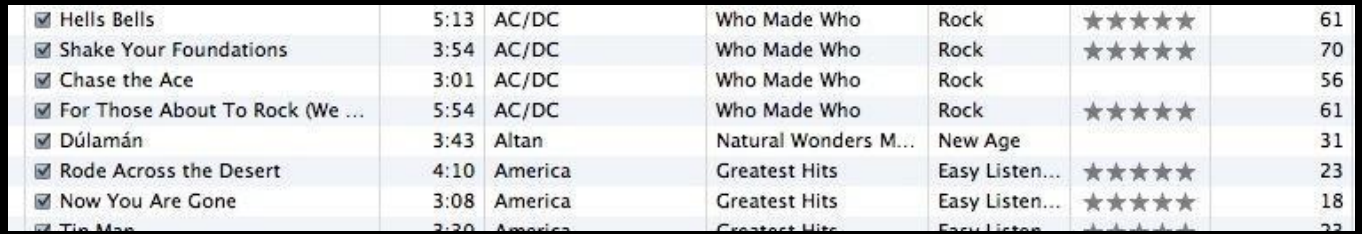
“각각의 정보”에 대하여...

- 칼럼은 객체인가? 아니면 다른 객체의 속성인가?
- 객체를 정의하고 나면 객체 간의 관계를 정의하여야 함

길이 앨범 장르 가수 평점 트랙 횟수

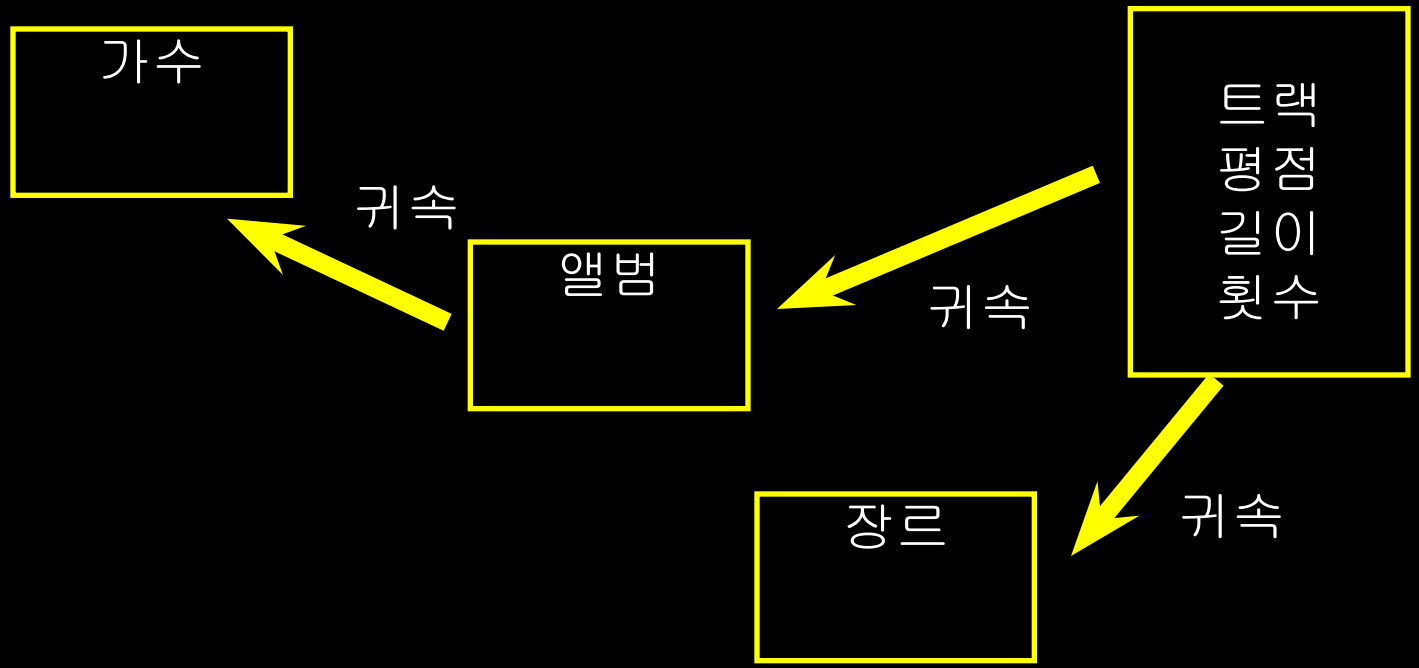
Hells Bells	5:13	AC/DC	Who Made Who	Rock	★★★★★	61
Shake Your Foundations	3:54	AC/DC	Who Made Who	Rock	★★★★★	70
Chase the Ace	3:01	AC/DC	Who Made Who	Rock		56
For Those About To Rock (We ...	5:54	AC/DC	Who Made Who	Rock	★★★★★	61
Dúlamán	3:43	Altan	Natural Wonders M...	New Age		31
Rode Across the Desert	4:10	America	Greatest Hits	Easy Listen...	★★★★★	23
Now You Are Gone	3:08	America	Greatest Hits	Easy Listen...	★★★★★	18
Tin Man	3:30	America	Greatest Hits	Easy Listen...	★★★★★	23

트랙
앨범
가수
장르
평점
길이
횟수

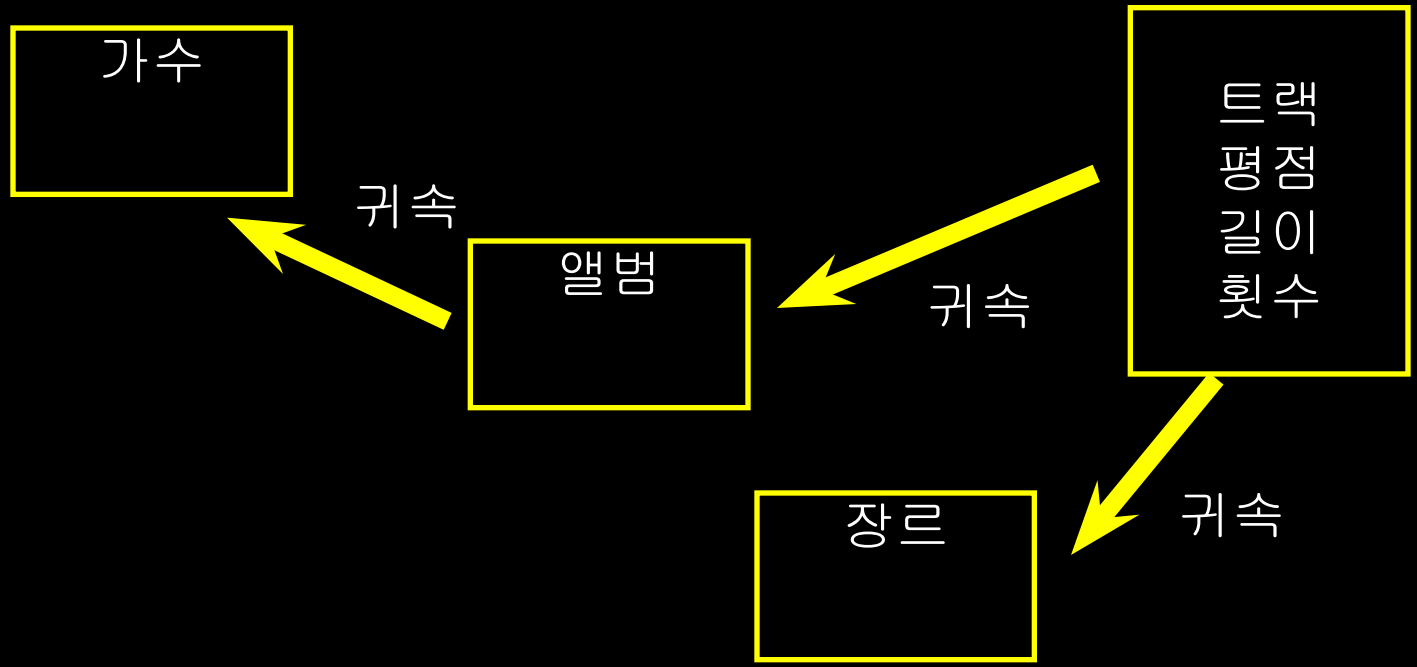


Hells Bells	5:13	AC/DC	Who Made Who	Rock	★★★★★	61
Shake Your Foundations	3:54	AC/DC	Who Made Who	Rock	★★★★★	70
Chase the Ace	3:01	AC/DC	Who Made Who	Rock		56
For Those About To Rock (We ...	5:54	AC/DC	Who Made Who	Rock	★★★★★	61
Dúlamán	3:43	Altan	Natural Wonders M...	New Age		31
Rode Across the Desert	4:10	America	Greatest Hits	Easy Listen...	★★★★★	23
Now You Are Gone	3:08	America	Greatest Hits	Easy Listen...	★★★★★	18

트랙
앨범
가수
장르
평점
길이
횟수



Hells Bells	5:13	AC/DC	Who Made Who	Rock	★★★★★	61
Shake Your Foundations	3:54	AC/DC	Who Made Who	Rock	★★★★★	70
Chase the Ace	3:01	AC/DC	Who Made Who	Rock		56
For Those About To Rock (We ...	5:54	AC/DC	Who Made Who	Rock	★★★★★	61
Dúlamán	3:43	Altan	Natural Wonders M...	New Age		31
Rode Across the Desert	4:10	America	Greatest Hits	Easy Listen...	★★★★★	23
Now You Are Gone	3:08	America	Greatest Hits	Easy Listen...	★★★★★	18



Hells Bells	5:13	AC/DC	Who Made Who	Rock	★★★★★	61
Shake Your Foundations	3:54	AC/DC	Who Made Who	Rock	★★★★★	70
Chase the Ace	3:01	AC/DC	Who Made Who	Rock		56
For Those About To Rock (We ...	5:54	AC/DC	Who Made Who	Rock	★★★★★	61
Dúlamán	3:43	Altan	Natural Wonders M...	New Age		31
Rode Across the Desert	4:10	America	Greatest Hits	Easy Listen...	★★★★★	23
Now You Are Gone	3:08	America	Greatest Hits	Easy Listen...	★★★★★	18

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

http://en.wikipedia.org/wiki/Database_normalization

Hells Bells	5:13	AC/DC	Who Made Who	Rock	★★★★★	61
Shake Your Foundations	3:54	AC/DC	Who Made Who	Rock	★★★★★	70
Chase the Ace	3:01	AC/DC	Who Made Who	Rock		56
For Those About To Rock (We ...	5:54	AC/DC	Who Made Who	[illegible]	★★★★★	61
Dúlamán	3:43	Altan	Natural Wonders M...	New Age		31
Rode Across the Desert	4:10	America	Greatest Hits	Easy Listen...	★★★★★	23
Now You Are Gone	3:08	America	Greatest Hits	Easy Listen...	★★★★★	18
Tin Man	3:30	America	Greatest Hits	Easy Listen...	★★★★★	23

우리는 이런 사항들을 계속 추적하고 싶다
각각의 음악을 어떤 밴드가 "만들었는지"...

이 트랙은 어떤 앨범에 "속하는지"??

이 트랙는 어떤 앨범과 관련되어 있는지?

정수 참조 패턴

정수를 사용하여
다른 테이블에 있는
로우를 참조

id	name
Filter	Filter
1	Led Zepplin
2	AC/DC

가수

id	artist_id	title
Filter	Filter	Filter
1	2	Who Made Who
2	1	IV

앨범

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

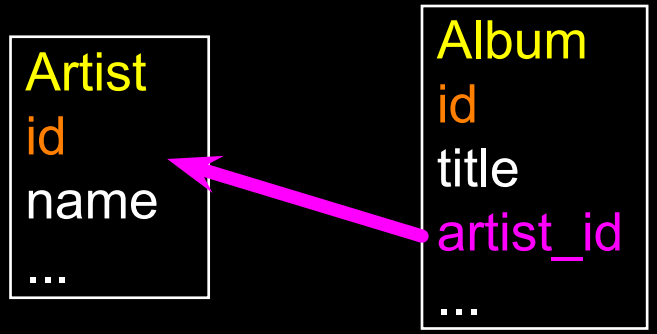
대표적인 사례

- 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

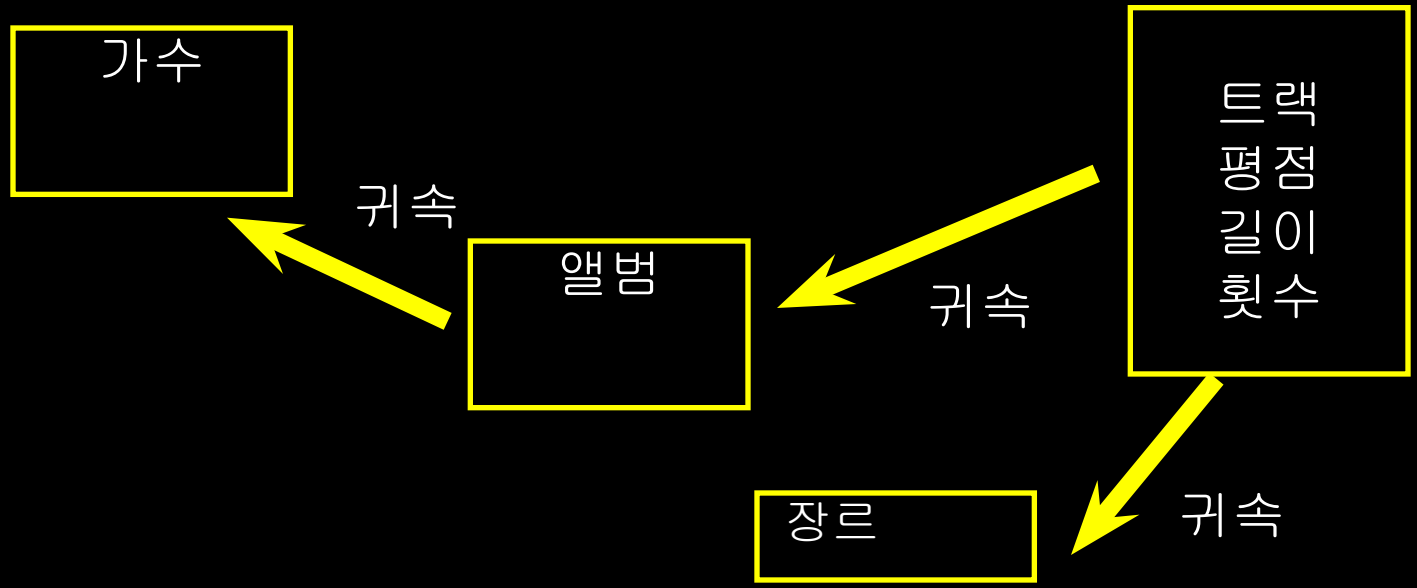
User
id
login
password
name
email
created_at
modified_at
login_at

외래 키

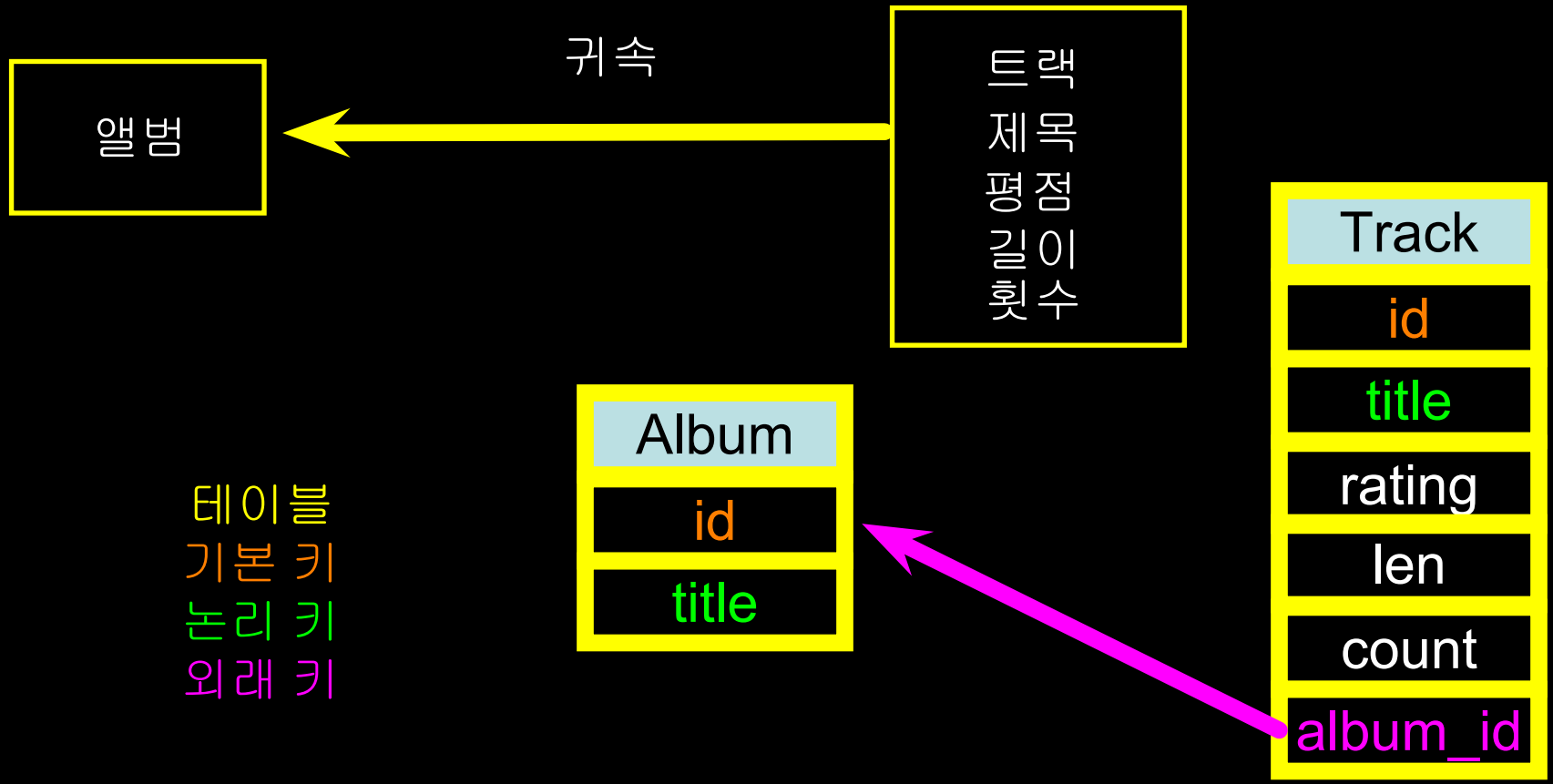
- 외래 키는 다른 테이블의 기본 키 를 가리키는 키를 일컬음
- 모든 기본 키가 정수라면 모든 외래 키도 정수임. - 이게 좋음

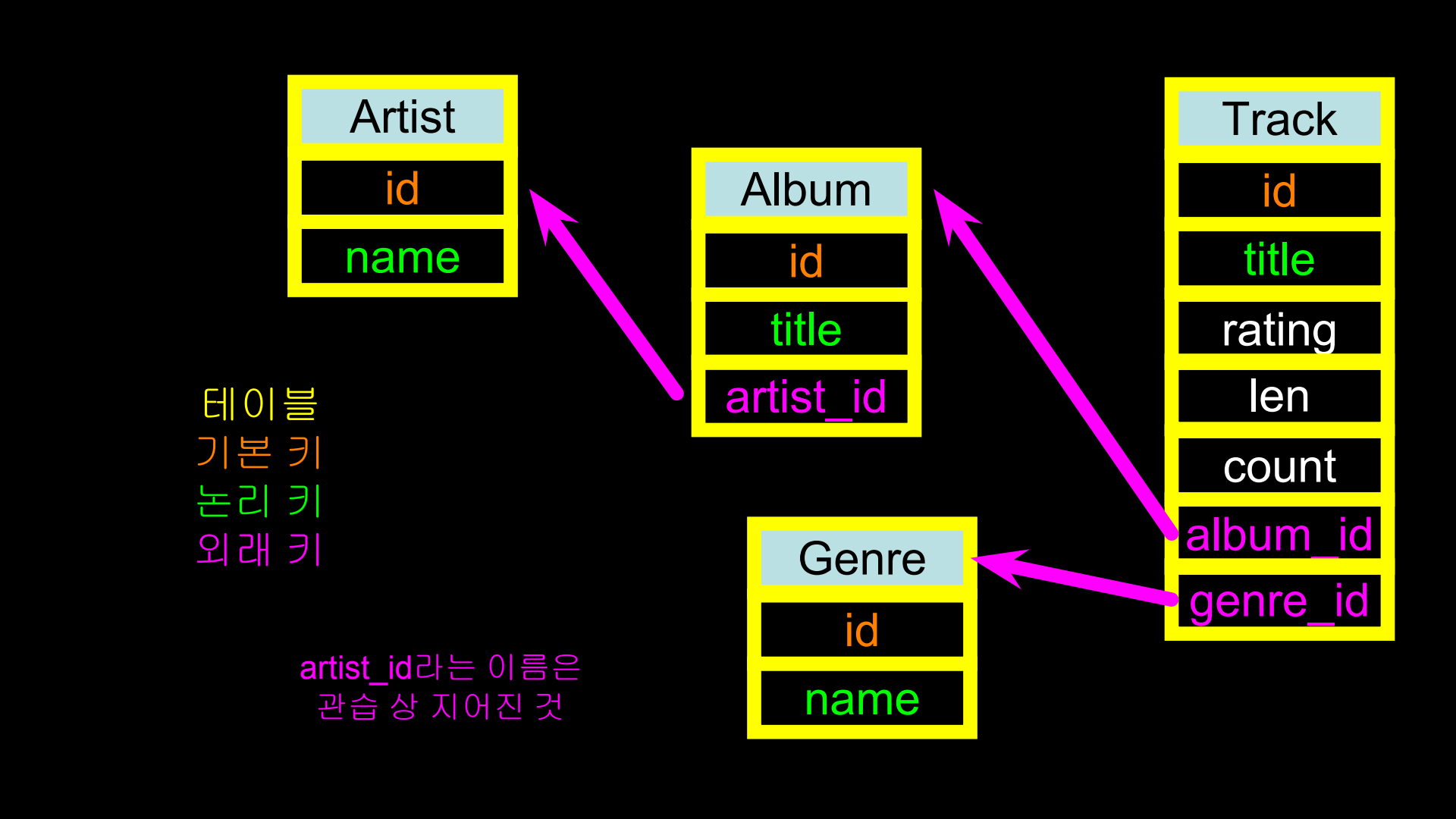


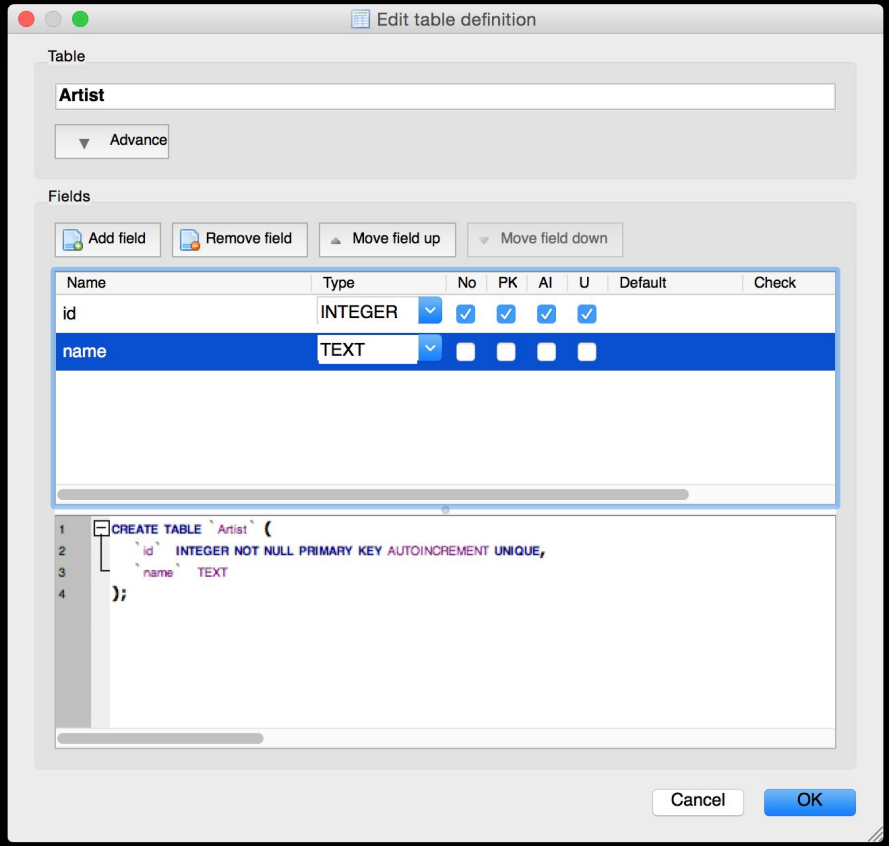
(테이블에서) 관계 형성

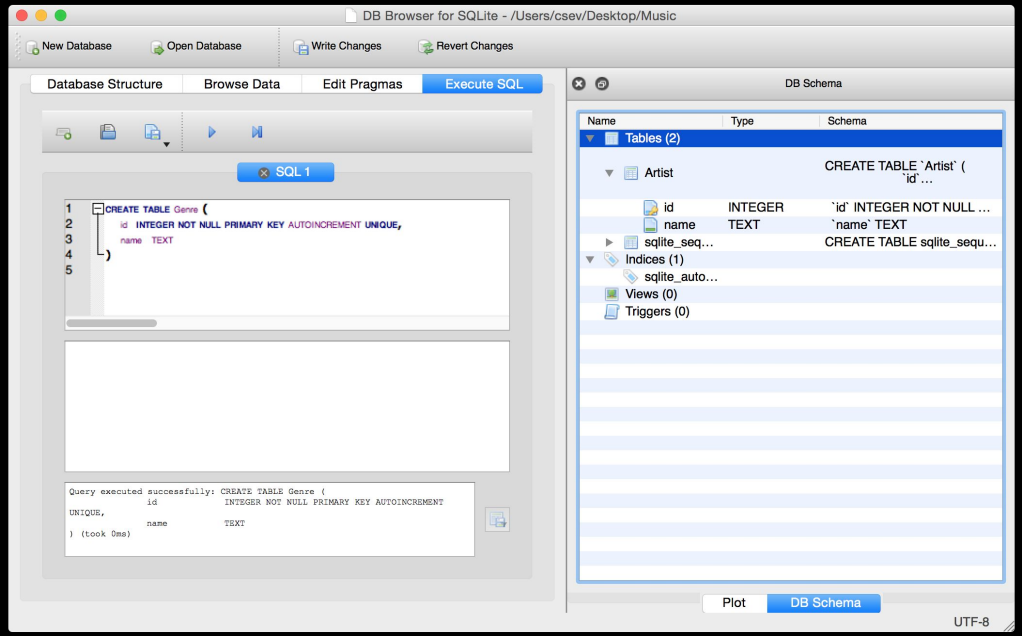


Hells Bells	5:13	AC/DC	Who Made Who	Rock	★★★★★	61
Shake Your Foundations	3:54	AC/DC	Who Made Who	Rock	★★★★★	70
Chase the Ace	3:01	AC/DC	Who Made Who	Rock		56
For Those About To Rock (We ...	5:54	AC/DC	Who Made Who	Rock	★★★★★	61
Dúlamán	3:43	Altan	Natural Wonders M...	New Age		31
Rode Across the Desert	4:10	America	Greatest Hits	Easy Listen...	★★★★★	23
Now You Are Gone	3:08	America	Greatest Hits	Easy Listen...	★★★★★	18


앨범
귀속
트랙
제목
평점
길이
횟수
Track
id
title
rating
len
count
album_id
Album
id
title
테이블
기본 키
논리 키
외래 키


Artist
id
name
Album
id
title
artist_id
Track
id
title
rating
len
count
album_id
genre_id
Genre
id
name
테이블
기본 키
논리 키
외래 키
artist_id라는 이름은
관습 상 지어진 것

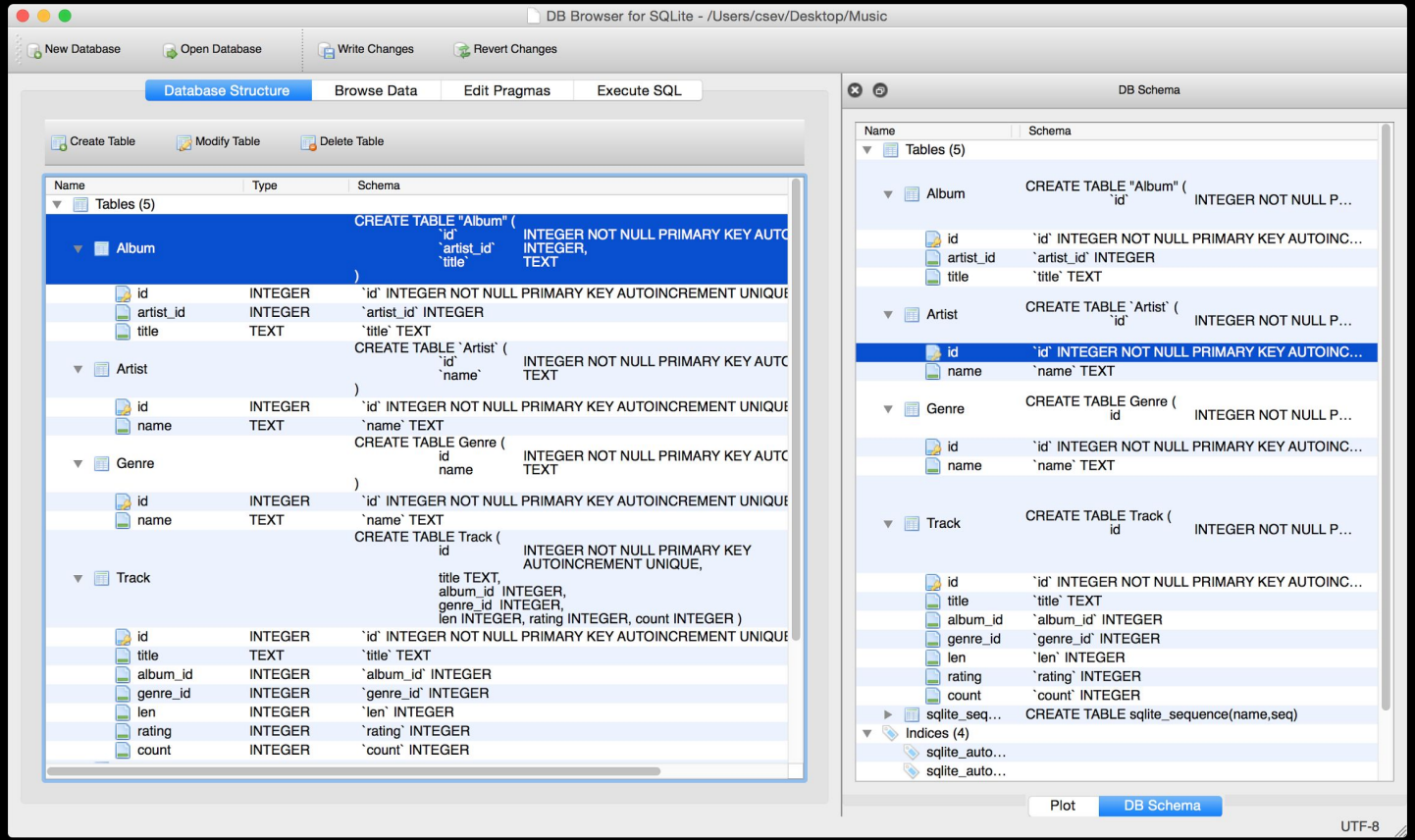

Edit table definition
Table
Artist
Advance
Fields
Add field
Remove field
Move field up
Move field down
Name
Type
No
PK
AI
U
Default
Check
id
INTEGER
name
TEXT
1 CREATE TABLE `Artist` (
2 `id` INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
3 `name` TEXT
4);
Cancel
OK

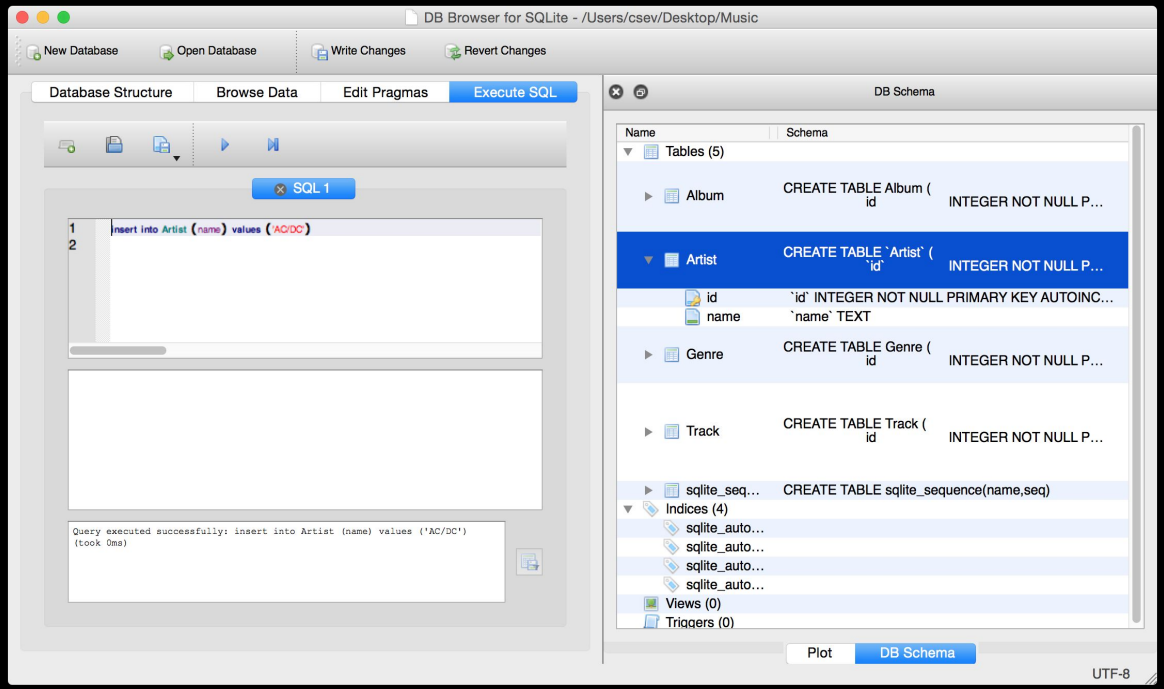



```
CREATE TABLE Genre (
    id      INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    name    TEXT
)
```

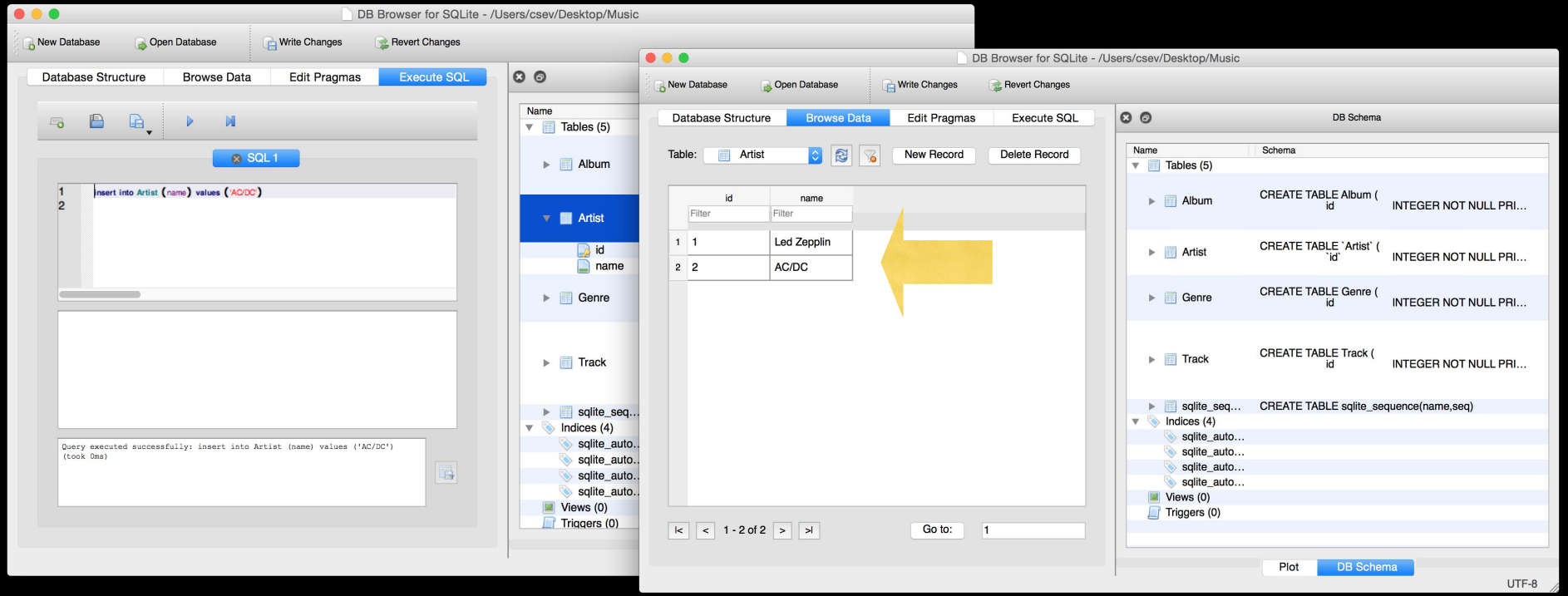
```
CREATE TABLE Album (
    id  INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    artist_id  INTEGER,
    title   TEXT
)


CREATE TABLE Track (
    id  INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    title TEXT,
    album_id  INTEGER,
    genre_id  INTEGER,
    len INTEGER, rating INTEGER, count INTEGER
)
```

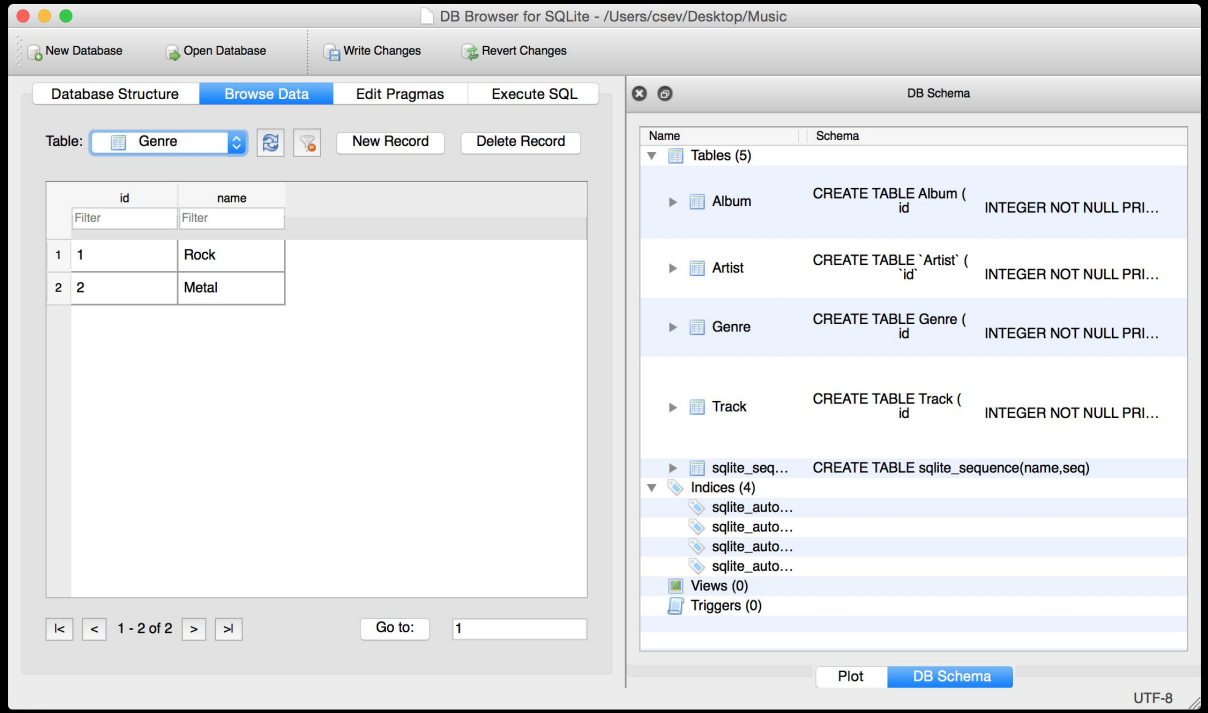

DB Browser for SQLite - /Users/csev/Desktop/Music
New Database
Open Database
Write Changes
Revert Changes
Database Structure
Browse Data
Edit Pragmas
Execute SQL
Create Table
Modify Table
Delete Table
Name
Type
Schema
Tables (5)
Album
CREATE TABLE "Album" (`id` INTEGER NOT NULL PRIMARY KEY AUTO `artist_id` INTEGER, `title` TEXT)
id
INTEGER
`id` INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE
artist_id
INTEGER
`artist_id` INTEGER
title
TEXT
`title` TEXT
Artist
CREATE TABLE `Artist` (`id` INTEGER NOT NULL PRIMARY KEY AUTO `name` TEXT)
id
INTEGER
`id` INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE
name
TEXT
`name` TEXT
Genre
CREATE TABLE Genre (id INTEGER NOT NULL PRIMARY KEY AUTO name TEXT)
id
INTEGER
`id` INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE
name
TEXT
`name` TEXT
Track
CREATE TABLE Track (id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE, title TEXT, album_id INTEGER, genre_id INTEGER, len INTEGER, rating INTEGER, count INTEGER)
id
INTEGER
`id` INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE
title
TEXT
`title` TEXT
album_id
INTEGER
`album_id` INTEGER
genre_id
INTEGER
`genre_id` INTEGER
len
INTEGER
`len` INTEGER
rating
INTEGER
`rating` INTEGER
count
INTEGER
`count` INTEGER
DB Schema
Name
Schema
Tables (5)
Album
CREATE TABLE "Album" (`id` INTEGER NOT NULL P…
id
`id` INTEGER NOT NULL PRIMARY KEY AUTOINC…
artist_id
`artist_id` INTEGER
title
`title` TEXT
Artist
CREATE TABLE `Artist` (`id` INTEGER NOT NULL P…
id
`id` INTEGER NOT NULL PRIMARY KEY AUTOINC…
name
`name` TEXT
Genre
CREATE TABLE Genre (id INTEGER NOT NULL P…
id
`id` INTEGER NOT NULL PRIMARY KEY AUTOINC…
name
`name` TEXT
Track
CREATE TABLE Track (id INTEGER NOT NULL P…
id
`id` INTEGER NOT NULL PRIMARY KEY AUTOINC…
title
`title` TEXT
album_id
`album_id` INTEGER
genre_id
`genre_id` INTEGER
len
`len` INTEGER
rating
`rating` INTEGER
count
`count` INTEGER
sqlite_seq…
CREATE TABLE sqlite_sequence(name,seq)
Indices (4)
sqlite_auto…
sqlite_auto…
Plot
DB Schema
UTF-8



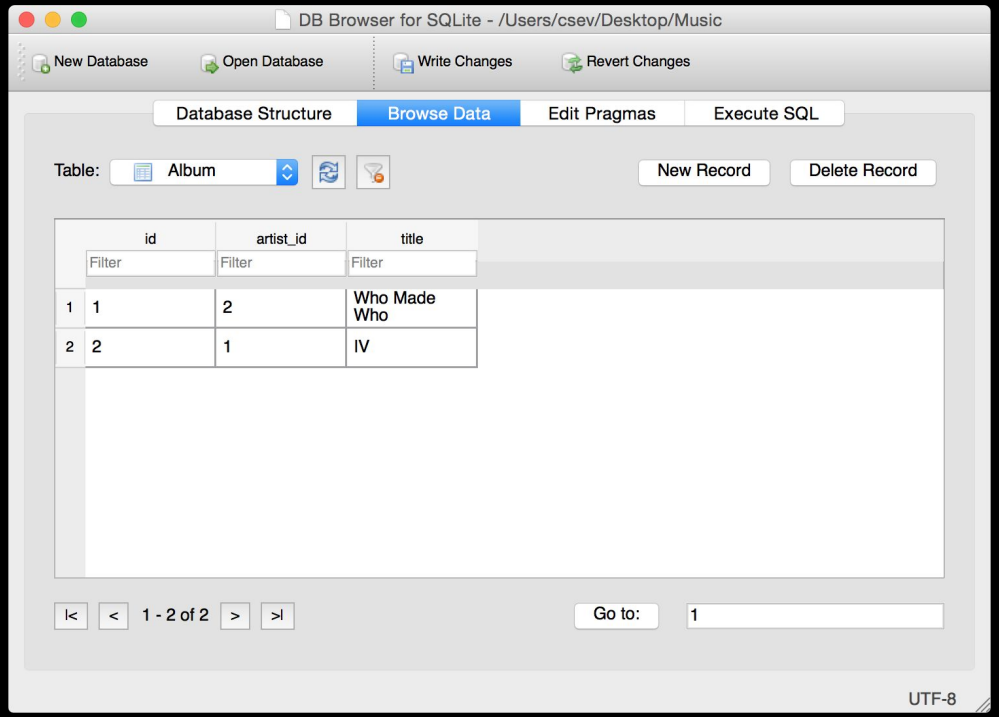
insert into Artist (name) values ('Led Zepplin')
insert into Artist (name) values ('AC/DC')



insert into Artist (name) values ('Led Zepplin')
insert into Artist (name) values ('AC/DC')



insert into Genre (name) values ('Rock')
insert into Genre (name) values ('Metal')




```
insert into Album (title, artist_id) values ('Who Made Who', 2)
insert into Album (title, artist_id) values ('IV', 1)
```

```
insert into Track (title, rating, len, count, album_id, genre_id)
     values ('Black Dog', 5, 297, 0, 2, 1)
insert into Track (title, rating, len, count, album_id, genre_id)
     values ('Stairway', 5, 482, 0, 2, 1)
insert into Track (title, rating, len, count, album_id, genre_id)
     values ('About to Rock', 5, 313, 0, 1, 2)
insert into Track (title, rating, len, count, album_id, genre_id)
     values ('Who Made Who', 5, 207, 0, 1, 2)
```

	id	title	album_id	genre_id	len	rating	count
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Black Dog	2	1	297	5	0
2	2	Stairway	2	1	482	5	0
3	3	About to Rock	1	2	313	5	0
4	4	Who Made Who	1	2	207	5	0

Track

id	title	album_id	genre_id	len	rating	count
Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	Black Dog	2	1	297	5	0
2	Stairway	2	1	482	5	0
3	About to Rock	1	2	313	5	0
4	Who Made Who	1	2	207	5	0

Album

id	artist_id	title
Filter	Filter	Filter
1	2	Who Made Who
2	1	IV

Artist

id	name
Filter	Filter
1	Led Zepplin
2	AC/DC

id	name
Filter	Filter
1	Rock
2	Metal

Genre

여러 테이블에 걸쳐 **JOIN** 사용

http://en.wikipedia.org/wiki/Join_(SQL)

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

id	artist_id	title
Filter	Filter	Filter
1	2	Who Made Who
2	1	IV

Album

Artist

	title	name
1	Who Made Who	AC/DC
2	IV	Led Zepplin

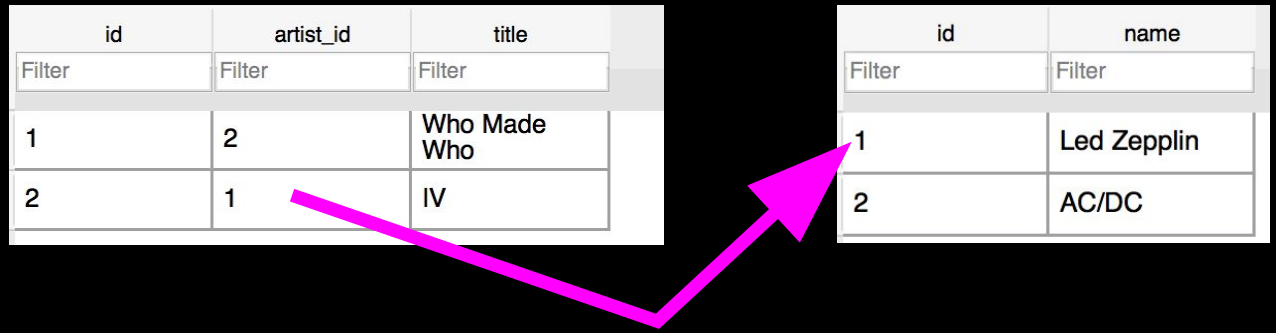
id	name
Filter	Filter
1	Led Zepplin
2	AC/DC

```
select Album.title, Artist.name from Album join Artist on Album.artist_id = Artist.id
```

우리가
보고싶은 것

데이터를
갖고 있는
테이블

테이블의
연결 관계



	title	artist_id	id	name
1	Who Made Who	2	2	AC/DC
2	IV	1	1	Led Zepplin

```
select Album.title, Album.artist_id, Artist.id,Artist.name
from Album join Artist on Album.artist_id = Artist.id
```

	title	genre_id	id	name
1	Black Dog	1	1	Rock
2	Black Dog	1	2	Metal
3	Stairway	1	1	Rock
4	Stairway	1	2	Metal
5	About to Rock	2	1	Rock
6	About to Rock	2	2	Metal
7	Who Made Who	2	1	Rock
8	Who Made Who	2	2	Metal

```
SELECT Track.title,
    Track.genre_id,
    Genre.id, Genre.name
FROM Track JOIN Genre
```

ON절 없이 두 테이블을
연결하면 로우의 모든 조합을
얻게 됨.

	title	name
1	Black Dog	Rock
2	Stairway	Rock
3	About to Rock	Metal
4	Who Made Who	Metal

id	title	album_id	genre_id	len	rating	count
Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	Black Dog	2	1	297	5	0
2	Stairway	2	1	482	5	0
3	About to Rock	1	2	313	5	0
4	Who Made Who	1	2	207	5	0

id	name
Filter	Filter
1	Rock
2	Metal

```
select Track.title, Genre.name from Track join Genre on Track.genre_id = Genre.id
```

우리가
보고싶은 것

데이터를
갖고 있는
테이블

테이블의
연결 관계

```
select Track.title, Artist.name, Album.title,
Genre.name from Track join Genre join Album join
Artist on Track.genre_id = Genre.id and
Track.album_id = Album.id and Album.artist_id =
Artist.id
```

	title	name	title	name
1	Black Dog	Led Zepplin	IV	Rock
2	Stairway	Led Zepplin	IV	Rock
3	About to Rock	AC/DC	Who Made Who	Metal
4	Who Made Who	AC/DC	Who Made Who	Metal

우리가 보고싶은 것

데이터를 갖고 있는 테이블

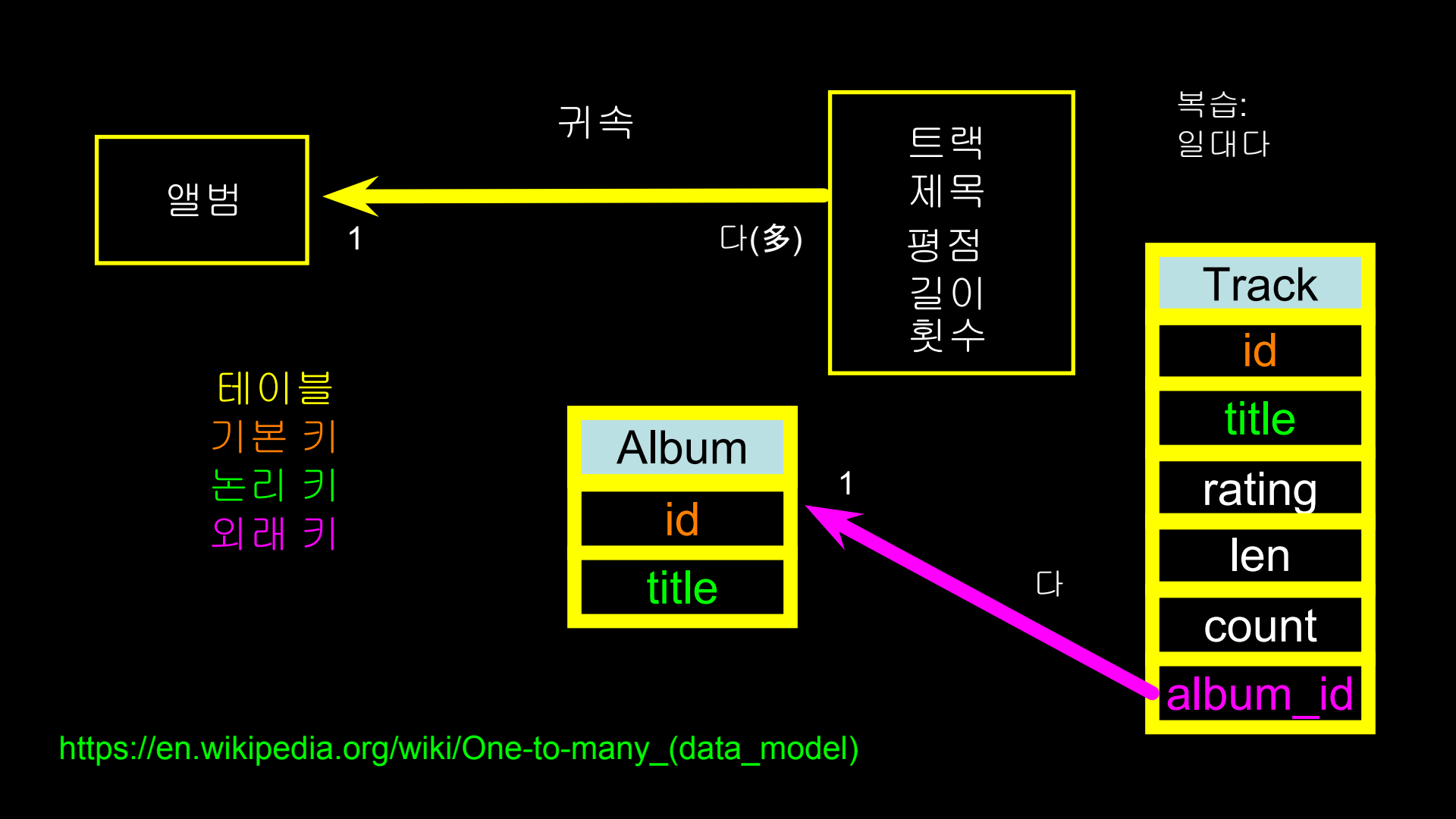
테이블의 연결 관계

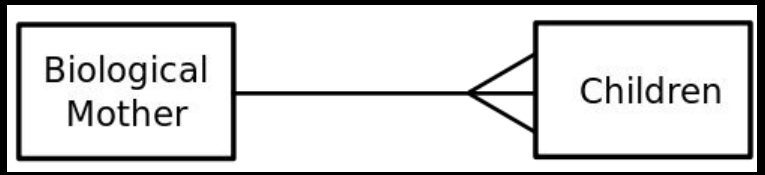
Hells Bells	5:13	AC/DC	Who Made Who	Rock	★★★★★		61
Shake Your Foundations	3:54	AC/DC	Who Made Who	Rock	★★★★★		70
Chase the Ace	3:01	AC/DC	Who Made Who	Rock			56
For Those About To Rock (We ...	5:54	AC/DC	Who Made Who	Rock	★★★★★		61
Dúlamán	3:43	Altan	Natural Wonders M...	New Age			31
Rode Across the Desert	4:10	America	Greatest Hits	Easy Listen...	★★★★★		23
Now You Are Gone	3:08	America	Greatest Hits	Easy Listen...	★★★★★		18
Tin Man	3:30	America	Greatest Hits	Easy Listen...	★★★★★		23
Sister Golden Hair	3:22	America	Greatest Hits	Easy Listen...	★★★★★		24
Track 01	4:22	Billy Price	Danger Zone	Blues/R&B	★★★★★		26
Track 02	2:45	Billy Price	Da				
Track 03	3:26	Billy Price	Da				
Track 04	4:17	Billy Price	Da				
Track 05	3:50	Billy Price	Da				
War Pigs/Luke's Wall	7:58	Black Sabbath	Pa				
Paranoid	2:53	Black Sabbath	Pa				
Planet Caravan	4:35	Black Sabbath	Pa				
Iron Man	5:59	Black Sabbath	Pa				
Electric Funeral	4:53	Black Sabbath	Pa				
Hand of Doom	7:10	Black Sabbath	Pa				
Rat Salad	2:30	Black Sabbath	Pa				
Jack the Stripper/Fairies Wear ...	6:14	Black Sabbath	Pa				
Bomb Squad (TECH)	3:28	Brent	Br				
clay techno	4:36	Brent	Br				
Heavy	3:08	Brent	Br				
Hi metal man	4:20	Brent	Brent's Album				1
Mistro	2:58	Brent	Brent's Album				1

	title	name	title	name
1	Black Dog	Led Zepplin	IV	Rock
2	Stairway	Led Zepplin	IV	Rock
3	About to Rock	AC/DC	Who Made Who	Metal
4	Who Made Who	AC/DC	Who Made Who	Metal

다대다 관계

https://en.wikipedia.org/wiki/Many-to-many_(data_model)


앨범
귀속
1
다(多)
트랙
제목
평점
길이
횟수
복습:
일대다
테이블
기본 키
논리 키
외래 키
Album
id
title
1
다
Track
id
title
rating
len
count
album_id
https://en.wikipedia.org/wiki/One-to-many_(data_model)



id	name
Filter	Filter
1	Rock
2	Metal

1

1 다

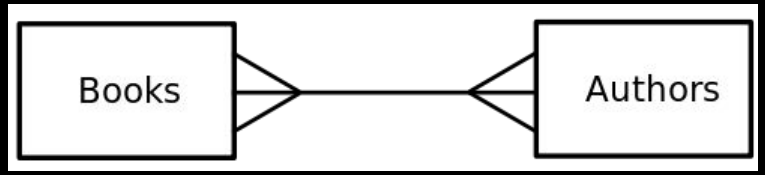
다

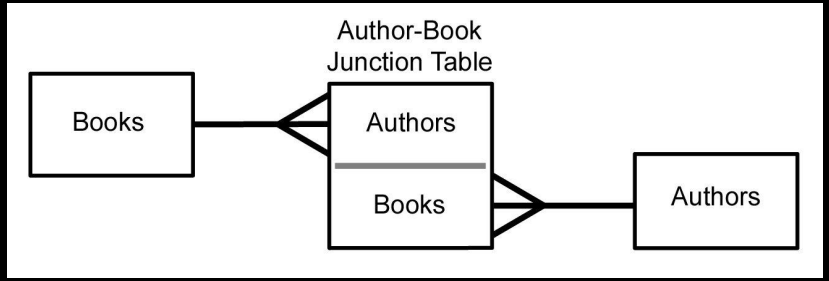
id	title	album_id	genre_id	len	rating	count
Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	Black Dog	2	1	297	5	0
2	Stairway	2	1	482	5	0
3	About to Rock	1	2	313	5	0
4	Who Made Who	1	2	207	5	0

https://en.wikipedia.org/wiki/One-to-many_(data_model)

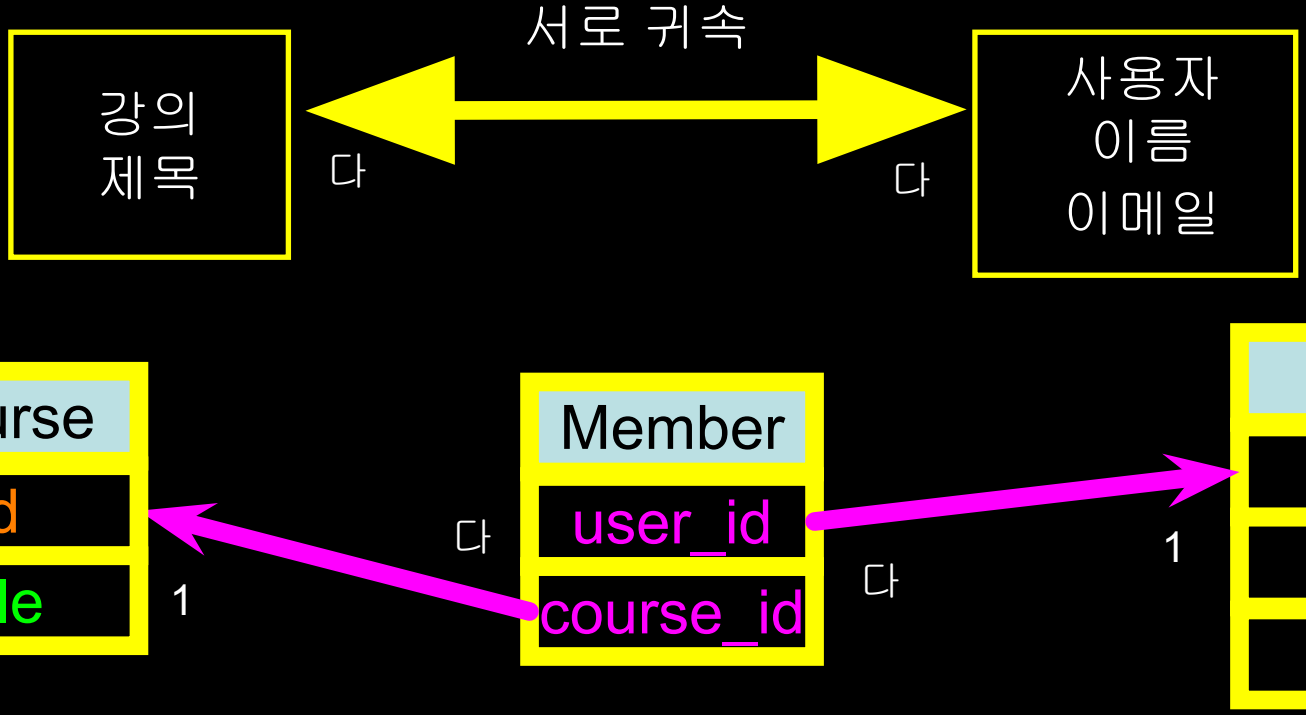
다대다

- 가끔 다대다 모델링이 필요할 때가 있음.
- 두 개의 외래 키를 가지고 “연결” 테이블을 추가해야 함
- 따로 기본 키가 존재하지 않음





https://en.wikipedia.org/wiki/Many-to-many_(data_model)



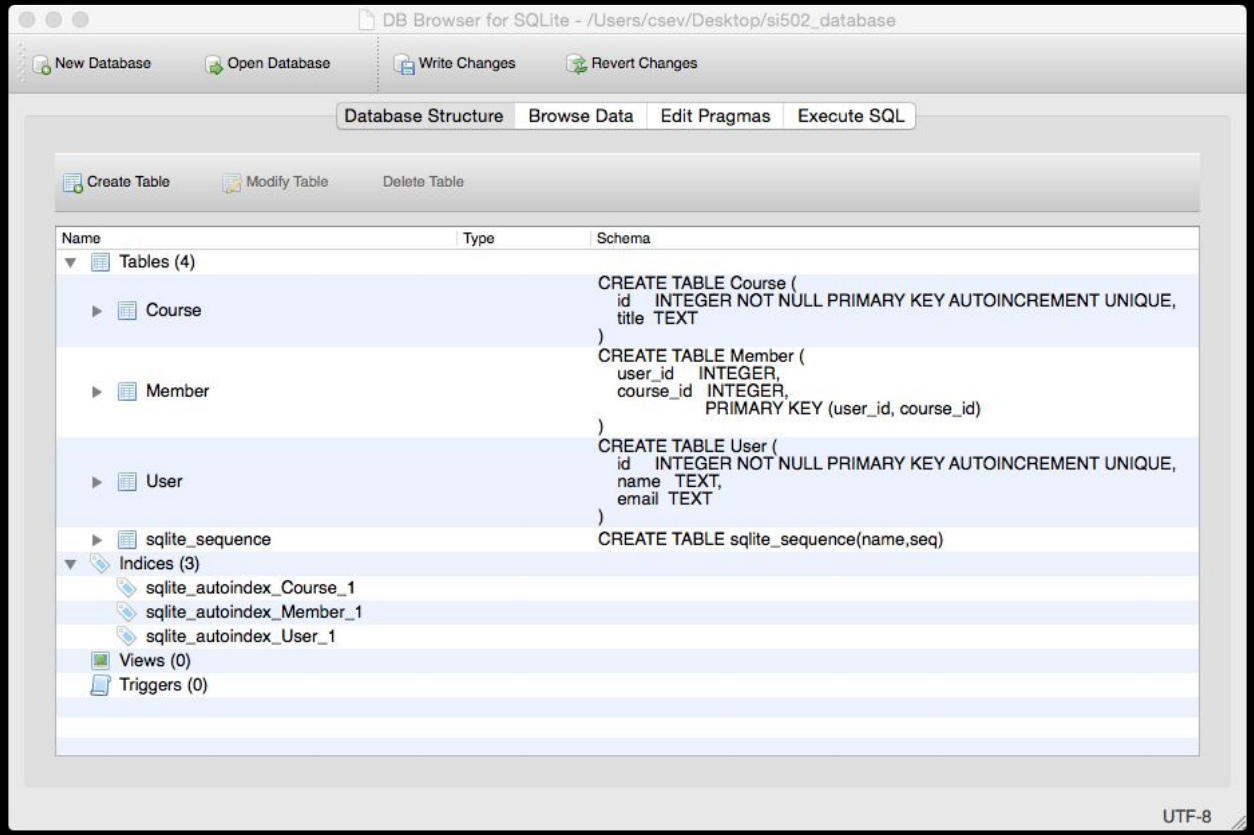
https://en.wikipedia.org/wiki/Many-to-many_(data_model)

```
CREATE TABLE User (
    id      INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    name    TEXT UNIQUE,
    email   TEXT
)

CREATE TABLE Course (
    id      INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    title   TEXT UNIQUE
)

CREATE TABLE Member (
    user_id     INTEGER,
    course_id   INTEGER,
    role        INTEGER,
    PRIMARY KEY (user_id, course_id)
)
```

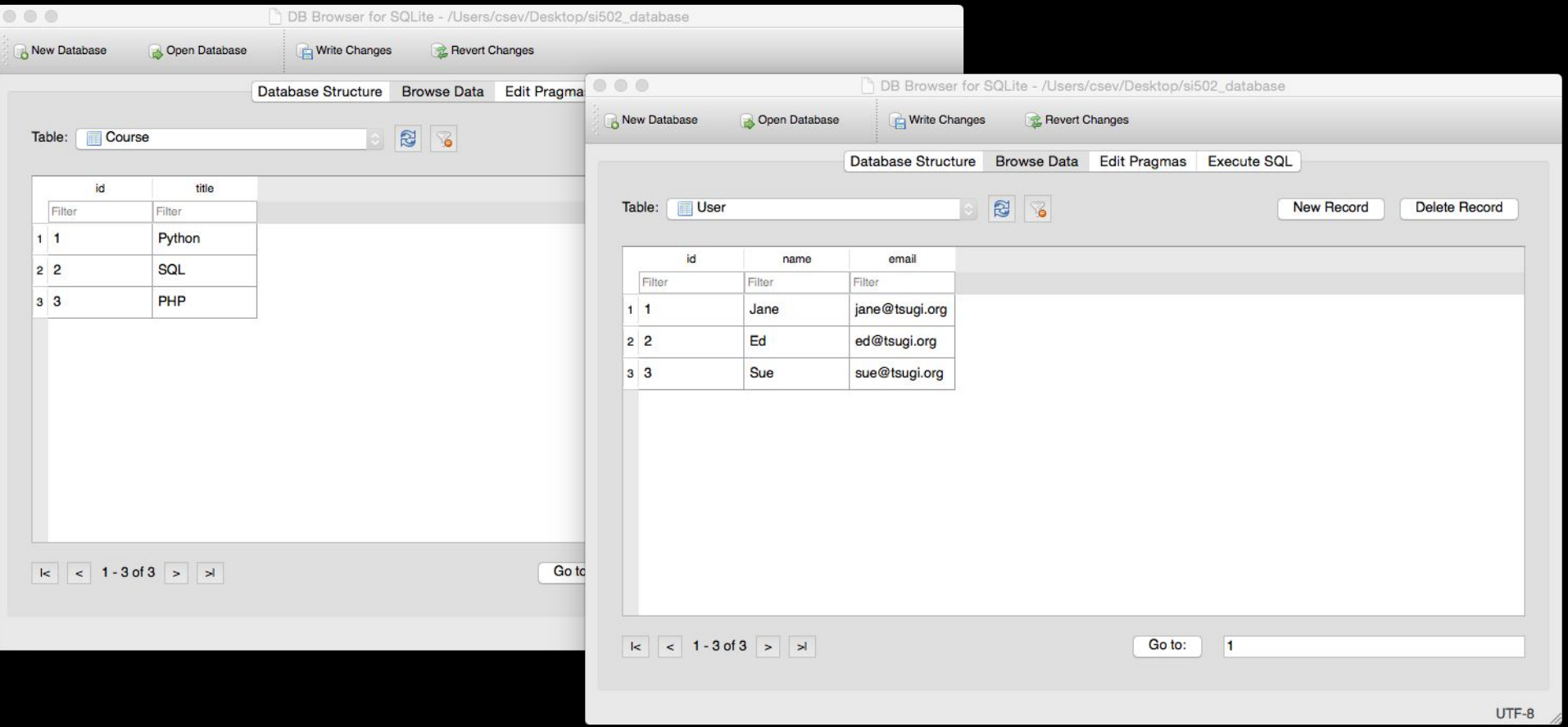
새로운
데이터베이스에서
시작


DB Browser for SQLite - /Users/csev/Desktop/si502_database
New Database
Open Database
Write Changes
Revert Changes
Database Structure
Browse Data
Edit Pragmas
Execute SQL
Create Table
Modify Table
Delete Table
Name
Type
Schema
Tables (4)
Course
CREATE TABLE Course (
id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
title TEXT
)
Member
CREATE TABLE Member (
user_id INTEGER,
course_id INTEGER,
PRIMARY KEY (user_id, course_id)
)
User
CREATE TABLE User (
id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
name TEXT,
email TEXT
)
sqlite_sequence
CREATE TABLE sqlite_sequence(name,seq)
Indices (3)
sqlite_autoindex_Course_1
sqlite_autoindex_Member_1
sqlite_autoindex_User_1
Views (0)
Triggers (0)
UTF-8

사용자와 강의를 삽입

```
INSERT INTO User (name, email) VALUES ('Jane', 'jane@tsugi.org');
INSERT INTO User (name, email) VALUES ('Ed', 'ed@tsugi.org');
INSERT INTO User (name, email) VALUES ('Sue', 'sue@tsugi.org');

INSERT INTO Course (title) VALUES ('Python');
INSERT INTO Course (title) VALUES ('SQL');
INSERT INTO Course (title) VALUES ('PHP');
```


DB Browser for SQLite - /Users/csev/Desktop/si502_database
New Database
Open Database
Write Changes
Revert Changes
Database Structure
Browse Data
Edit Pragma
Table: Course
id
title
Filter
1 1 Python
2 2 SQL
3 3 PHP
I< < 1 - 3 of 3 > >I
Go to
DB Browser for SQLite - /Users/csev/Desktop/si502_database
New Database
Open Database
Write Changes
Revert Changes
Database Structure
Browse Data
Edit Pragmas
Execute SQL
Table: User
New Record
Delete Record
id
name
email
Filter
1 1 Jane jane@tsugi.org
2 2 Ed ed@tsugi.org
3 3 Sue sue@tsugi.org
I< < 1 - 3 of 3 > >I
Go to:
1
UTF-8

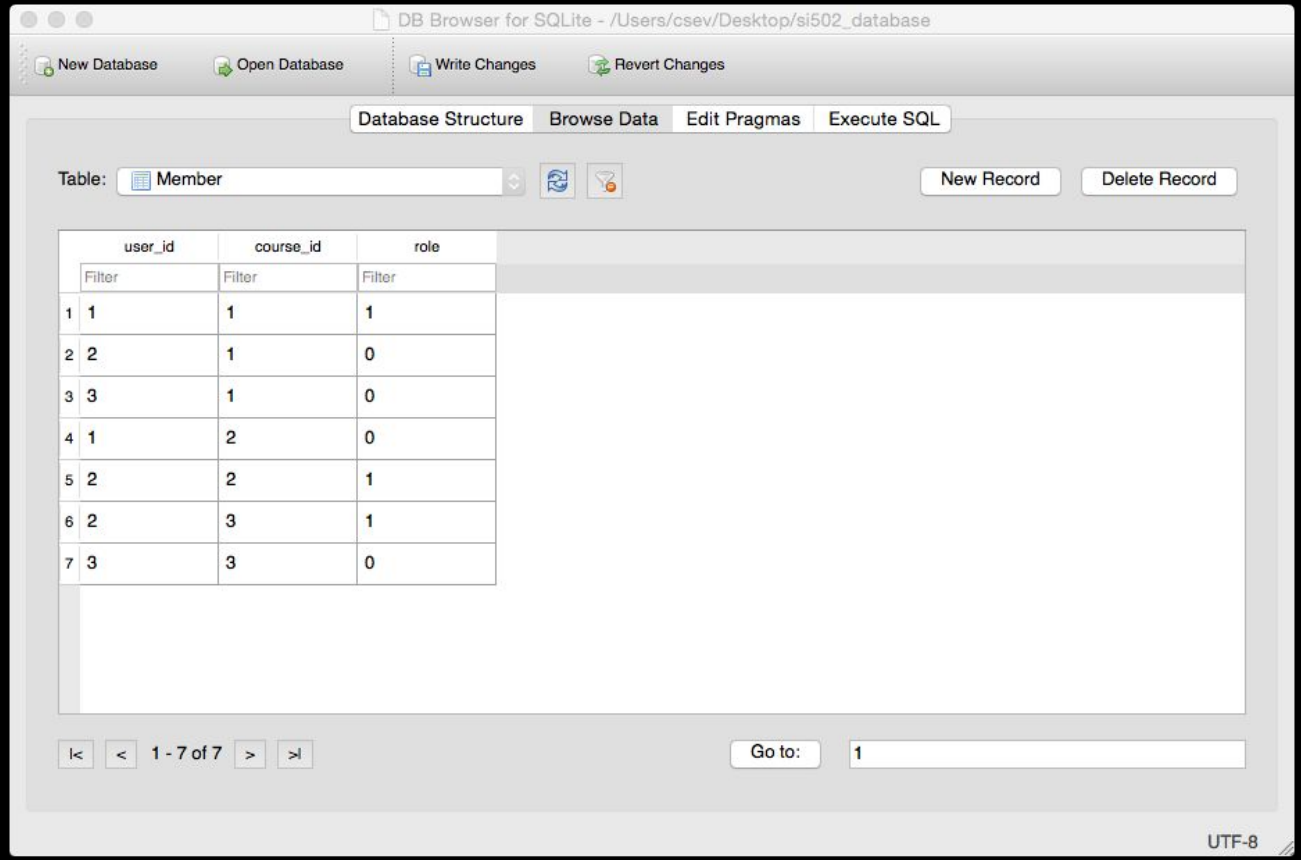
id	name	email
Filter	Filter	Filter
1	Jane	jane@tsugi.org
2	Ed	ed@tsugi.org
3	Sue	sue@tsugi.org

id	title
Filter	Filter
1	Python
2	SQL
3	PHP

```
INSERT INTO Member (user_id, course_id, role) VALUES (1, 1, 1);
INSERT INTO Member (user_id, course_id, role) VALUES (2, 1, 0);
INSERT INTO Member (user_id, course_id, role) VALUES (3, 1, 0);

INSERT INTO Member (user_id, course_id, role) VALUES (1, 2, 0);
INSERT INTO Member (user_id, course_id, role) VALUES (2, 2, 1);

INSERT INTO Member (user_id, course_id, role) VALUES (2, 3, 1);
INSERT INTO Member (user_id, course_id, role) VALUES (3, 3, 0);
```


DB Browser for SQLite - /Users/csev/Desktop/si502_database
New Database
Open Database
Write Changes
Revert Changes
Database Structure
Browse Data
Edit Pragmas
Execute SQL
Table:
Member
New Record
Delete Record
user_id
course_id
role
Filter
Filter
Filter
1 1 1 1
2 2 1 0
3 3 1 0
4 1 2 0
5 2 2 1
6 2 3 1
7 3 3 0
|<
<
1 - 7 of 7
>
>|
Go to:
1
UTF-8

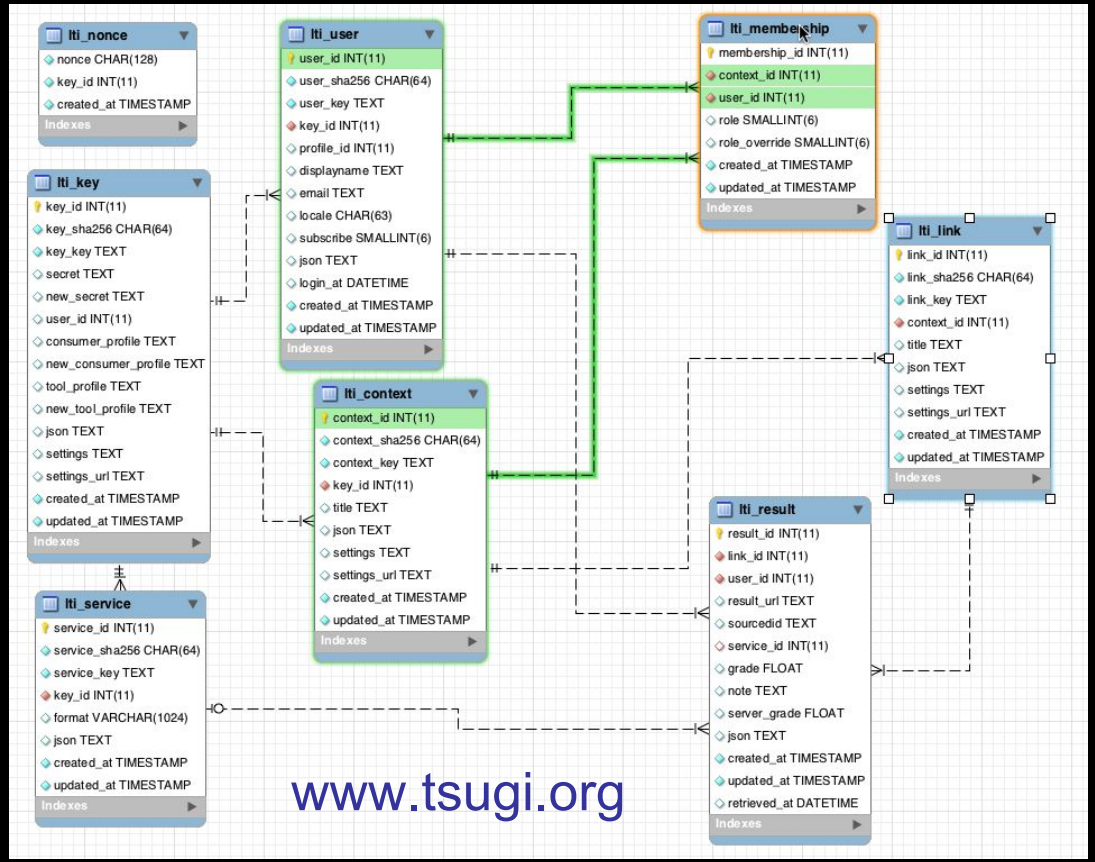
id	name	email
Filter	Filter	Filter
1	Jane	jane@tsugi.org
2	Ed	ed@tsugi.org
3	Sue	sue@tsugi.org

user_id	course_id	role
Filter	Filter	Filter
1	1	1
2	1	0
3	1	0
1	2	0
2	2	1
2	3	1
3	3	0

id	title
Filter	Filter
1	Python
2	SQL
3	PHP

```
SELECT User.name, Member.role, Course.title
FROM User JOIN Member JOIN Course
ON Member.user_id = User.id AND
Member.course_id = Course.id
ORDER BY Course.title, Member.role DESC, User.name
```

	name	role	title
2	Sue	0	PHP
3	Jane	1	Python
4	Ed	0	Python
5	Sue	0	Python
6	Ed	1	SQL


lti_nonce
nonce CHAR(128)
key_id INT(11)
created_at TIMESTAMP
Indexes
lti_user
user_id INT(11)
user_sha256 CHAR(64)
user_key TEXT
key_id INT(11)
profile_id INT(11)
displayname TEXT
email TEXT
locale CHAR(63)
subscribe SMALLINT(6)
json TEXT
login_at DATETIME
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
lti_membership
membership_id INT(11)
context_id INT(11)
user_id INT(11)
role SMALLINT(6)
role_override SMALLINT(6)
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
lti_key
key_id INT(11)
key_sha256 CHAR(64)
key_key TEXT
secret TEXT
new_secret TEXT
user_id INT(11)
consumer_profile TEXT
new_consumer_profile TEXT
tool_profile TEXT
new_tool_profile TEXT
json TEXT
settings TEXT
settings_url TEXT
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
lti_context
context_id INT(11)
context_sha256 CHAR(64)
context_key TEXT
key_id INT(11)
title TEXT
json TEXT
settings TEXT
settings_url TEXT
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
lti_link
link_id INT(11)
link_sha256 CHAR(64)
link_key TEXT
context_id INT(11)
title TEXT
json TEXT
settings TEXT
settings_url TEXT
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
lti_service
service_id INT(11)
service_sha256 CHAR(64)
service_key TEXT
key_id INT(11)
format VARCHAR(1024)
json TEXT
created_at TIMESTAMP
updated_at TIMESTAMP
Indexes
lti_result
result_id INT(11)
link_id INT(11)
user_id INT(11)
result_url TEXT
sourcedid TEXT
service_id INT(11)
grade FLOAT
note TEXT
server_grade FLOAT
json TEXT
created_at TIMESTAMP
updated_at TIMESTAMP
retrieved_at DATETIME
Indexes
www.tsugi.org

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

추가적인 SQL 주제들

- 인덱스는 문자열 등의 필드에 접근할 때 성능을 향상시킴
- 데이터 제약 조건 - (NULL값 불가 등..)
- 트랜잭션 - 여러 SQL 연산을 하나로 묶어 단위로 처리

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

Acknowledgements / Contributions

...

Initial Development: Charles Severance, University of Michigan School of Information

Contributor

- Seung-June Lee (plusjune@gmail.com)
- Connect Foundation

Translator:

- Jeungmin Oh (tangza@gmail.com)
- Donghoon Han (hanuright@gmail.com)