Database Normalization and Joins

Charles Severance
Barb Ericson

Python for Everybody
www.py4e.com/lectures3/

Database Design

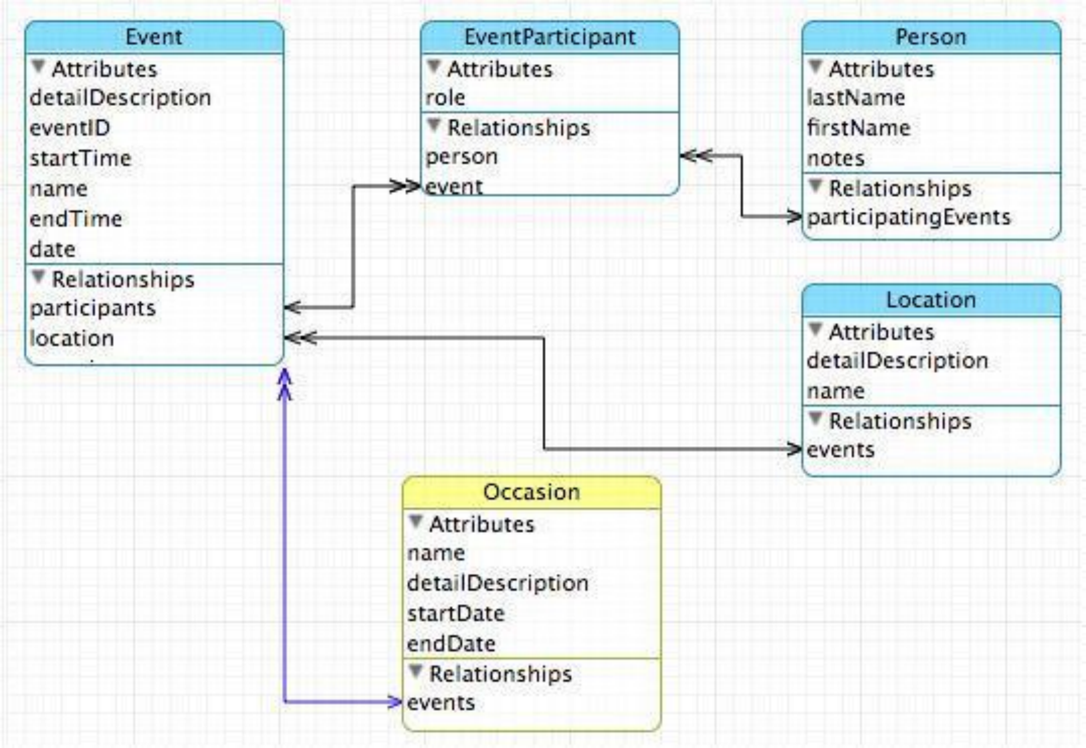
- Database design is an art form of its own with particular skills and experience
- Our goal is to avoid the really bad mistakes and design clean and easily understood databases
- Others may performance tune things later

Building a Data Model

- Drawing a picture of the data objects for our application and then figuring out how to represent the objects and their relationships

- Basic Rule: Don’t put the same string data in a database twice - use a relationship instead

- When there is one thing in the “real world” there should be one copy of that thing in the database



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

Storing Movie Data

- What if we want to store data about movies in a database?
- We can use OMDB to get the data.

What data do you need?

- Title, year, actors

Title	Year	Actors
Shaft	2000	"Samuel L. Jackson, Vanessa Williams, Christian Bale"
Glass	2019	"James McAvoy, Bruce Willis, Samuel L. Jackson"

For each “piece of info”...

- Is the column an object or an attribute of another object?
- Once we define objects, we need to define the relationships between objects

Title

Year

Actors

Movie and Actor

- Break tables into the entities they represent

Movie

ID	Title	Year
1	Shaft	2000
2	Glass	2019

Actor

ID	Name
1	Samuel L. Jackson
2	Vanessa Williams
3	Christian Bale
4	James McAvoy
5	Bruce Willis

Connecting Movie and Actor

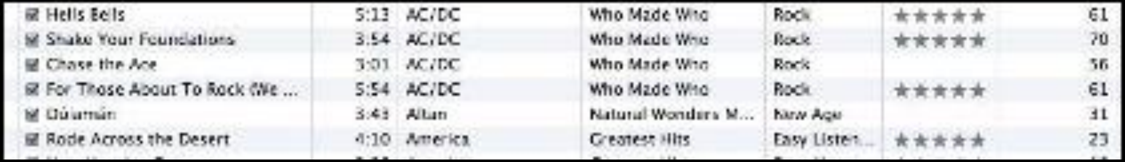
- This is a many to many relationship
  - A movie can have many actors
  - An actor can appear in many movies

Movie_Actor

movie_id	actor_id
1	1
1	2
1	3
2	4
2	5
2	1

Music Tracks

- What if you want to store data for music tracks: title, length, artist, album, genre, rating, count?



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

For each “piece of info”...

- Is the column an object or an attribute of another object?
- Once we define objects, we need to define the relationships between objects

Len

Album

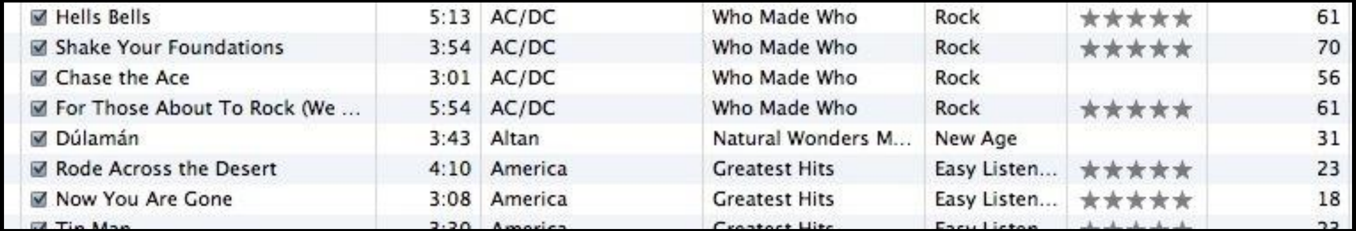
Genre

Artist

Rating

Track

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

Track

Album

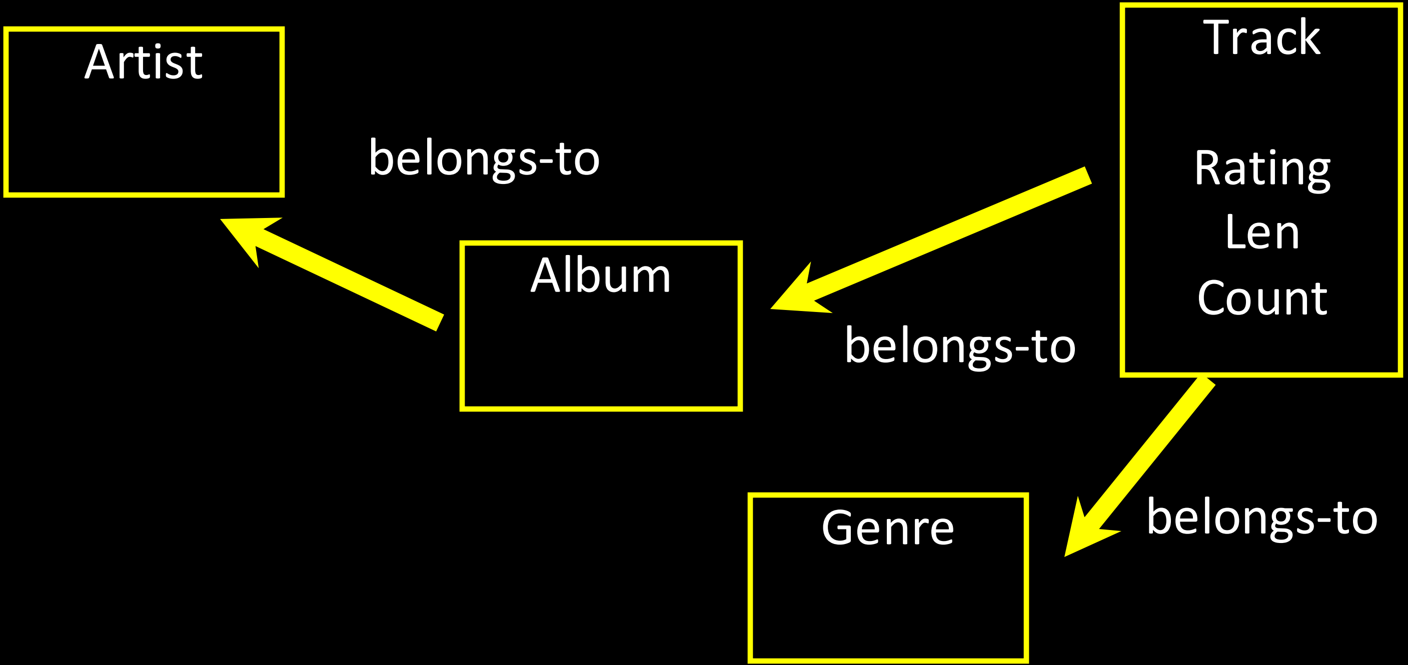
Artist

Genre

Rating

Len

Count



Artist

belongs-to

Album

Track

Rating

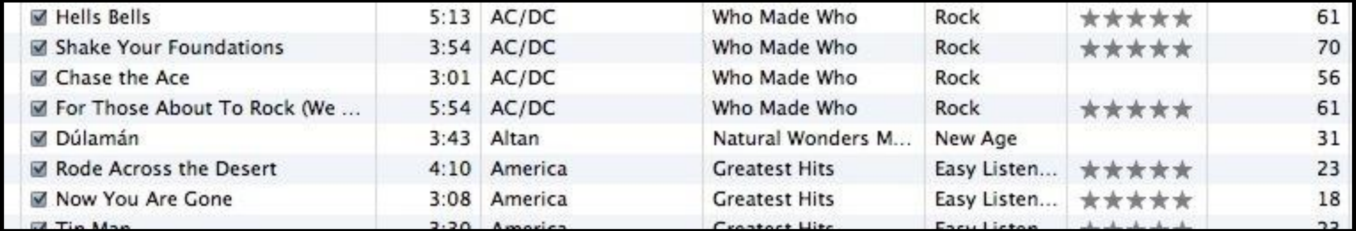
Len

Count

belongs-to

Genre

belongs-to



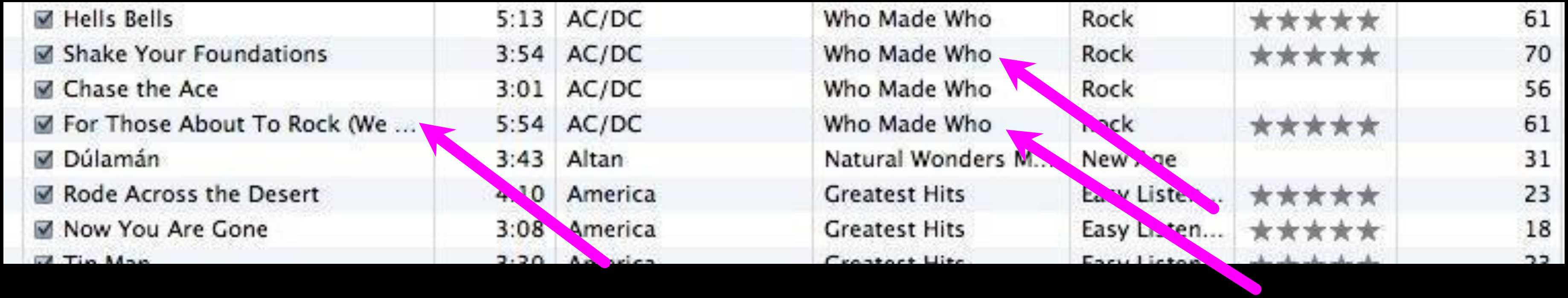
☑	Hells Bells	5:13	AC/DC	Who Made Who	Rock	★★★★★	61
☑	Shake Your Foundations	3:54	AC/DC	Who Made Who	Rock	★★★★★	70
☑	Chase the Ace	3:01	AC/DC	Who Made Who	Rock		56
☑	For Those About To Rock (We ...	5:54	AC/DC	Who Made Who	Rock	★★★★★	61
☑	Dúlamán	3:43	Altan	Natural Wonders M...	New Age		31
☑	Rode Across 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

Representing Relationships in a Database

Database Normalization (3NF)

- There is *tons* of database theory - way too much to understand without excessive predicate calculus
- Do not replicate string data - reference it instead
- Use integers for keys and for references
- Add a special “key” column to each table which we will make references to. By convention, many programmers call this column “id”

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	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

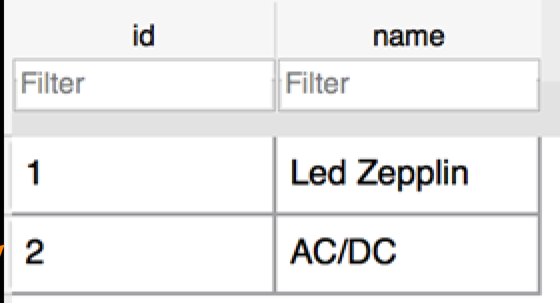
We want to keep track of which band is the “creator” of each music track...

What album does this song “belong to”??

Which album is this song related to?

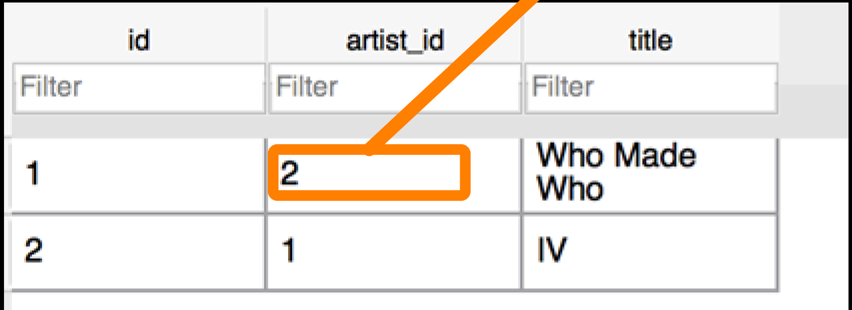
Integer Reference Pattern

We use integers to reference rows in another table



Artist

id	name
Filter	Filter
1	Led Zepplin
2	AC/DC



Album

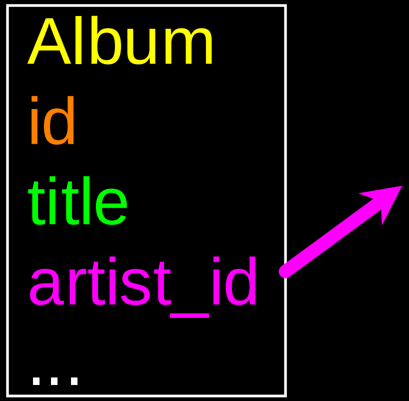
id	artist_id	title
Filter	Filter	Filter
1	2	Who Made Who
2	1	IV

Three Kinds of Keys

- Primary key - generally an integer auto-incremented field

- Logical key - What the outside world uses for lookup

- Foreign key - generally an integer key pointing to a row in another table



Album
id
title
artist_id
...

Key Rules

Best practices

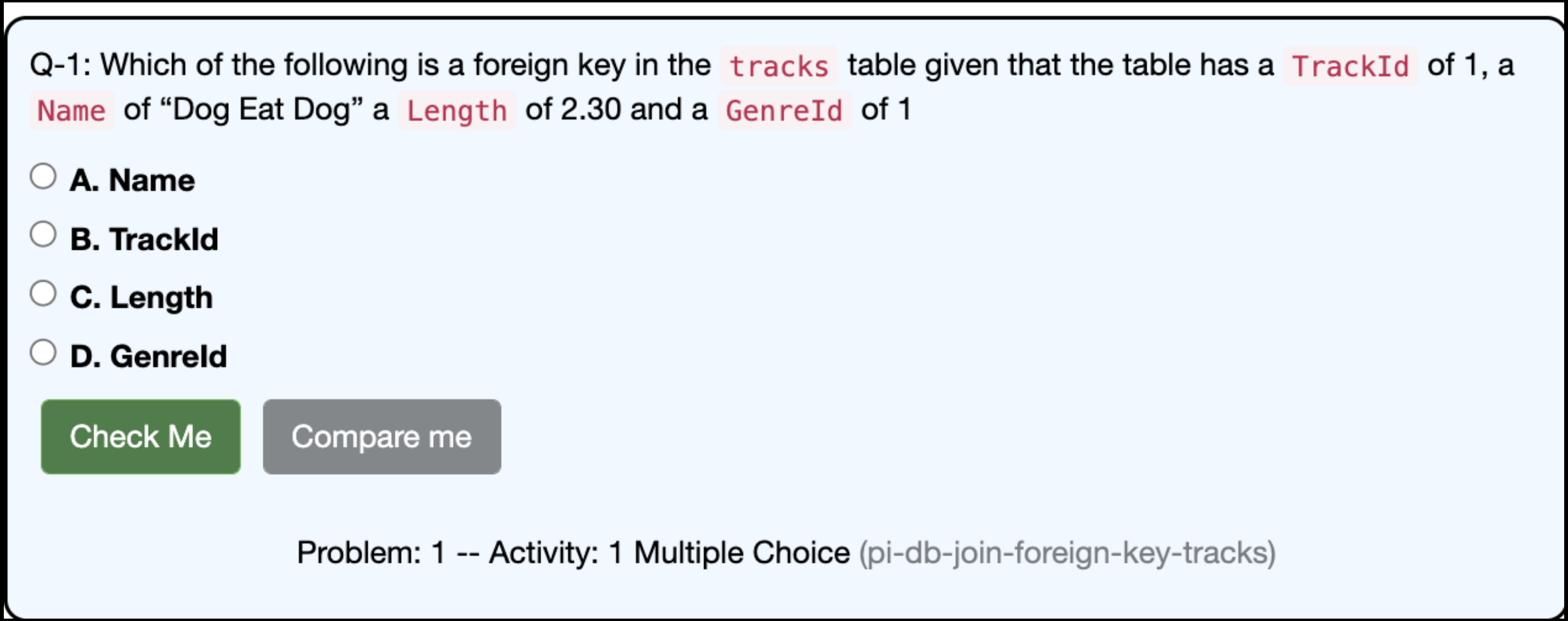
- Never use your logical key as the primary key

- Logical keys can and do change, albeit slowly

- Relationships that are based on matching string fields are less efficient than integers

User
id
login
password
name
email
created_at
modified_at
login_at

PI #1



Q-1: Which of the following is a foreign key in the `tracks` table given that the table has a `TrackId` of 1, a `Name` of “Dog Eat Dog” a `Length` of 2.30 and a `GenreId` of 1

○ **A. Name**

○ **B. TrackId**

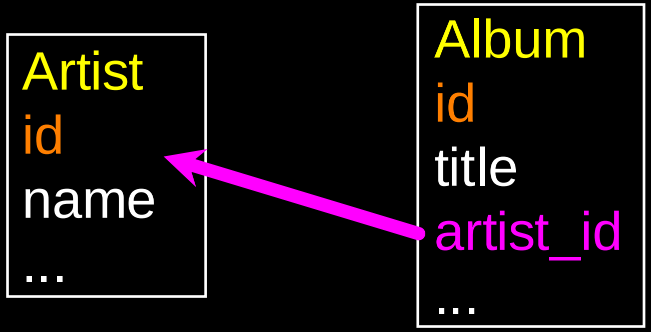
○ **C. Length**

○ **D. GenreId**

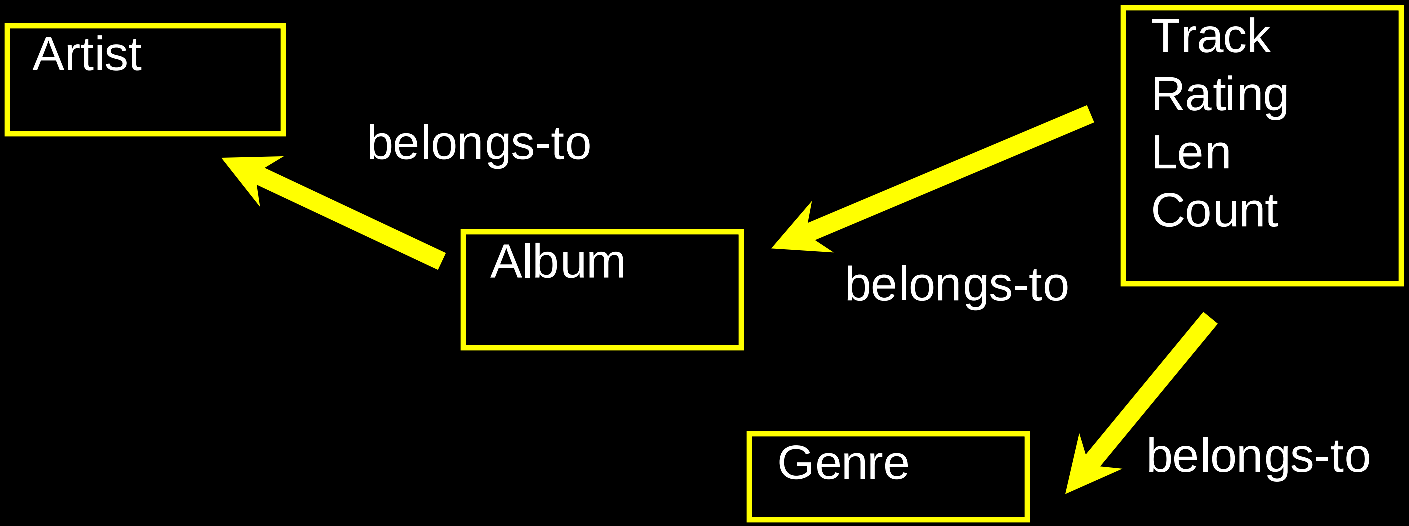
Problem: 1 -- Activity: 1 Multiple Choice (pi-db-join-foreign-key-tracks)

Foreign Keys

- A foreign key is when a table has a column that contains a key which points to the primary key of another table.
- When all primary keys are integers, then all foreign keys are integers - this is good - very good



Relationship Building (in tables)



Artist

belongs-to

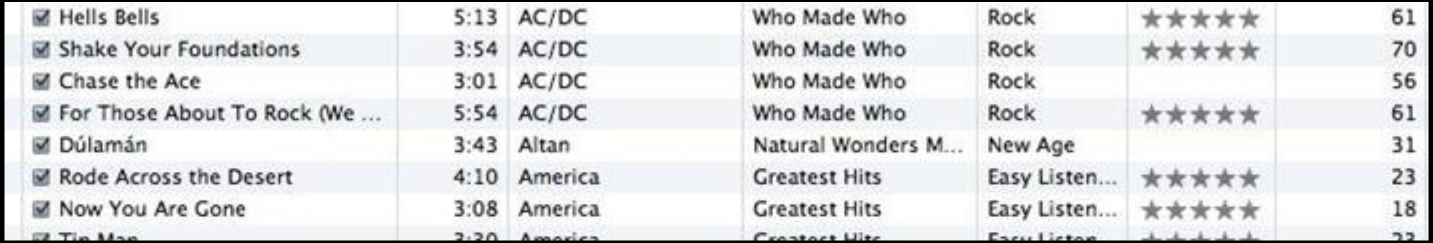
Album

belongs-to

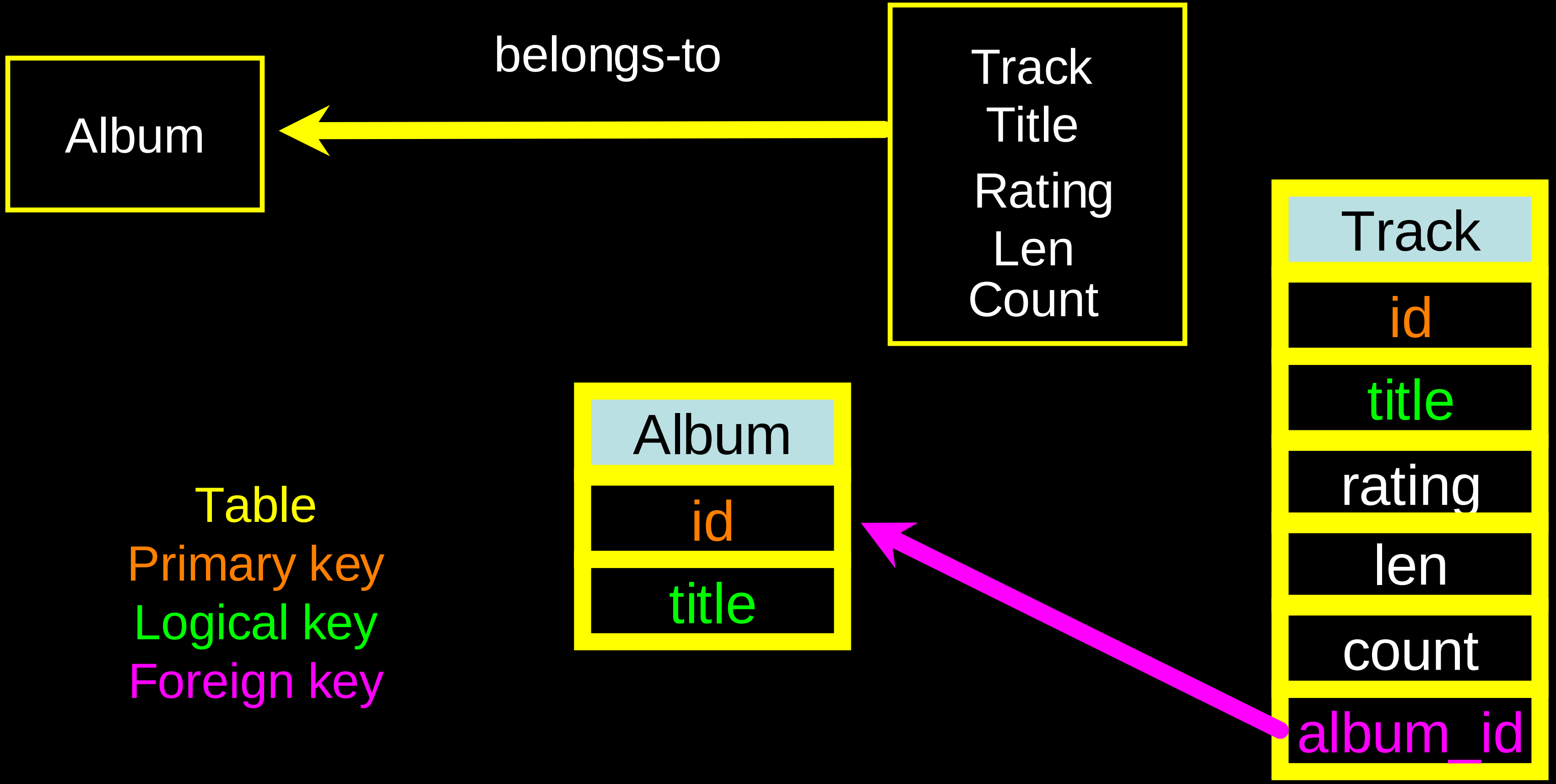
Track
Rating
Len
Count

belongs-to

Genre



☑ 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



Album

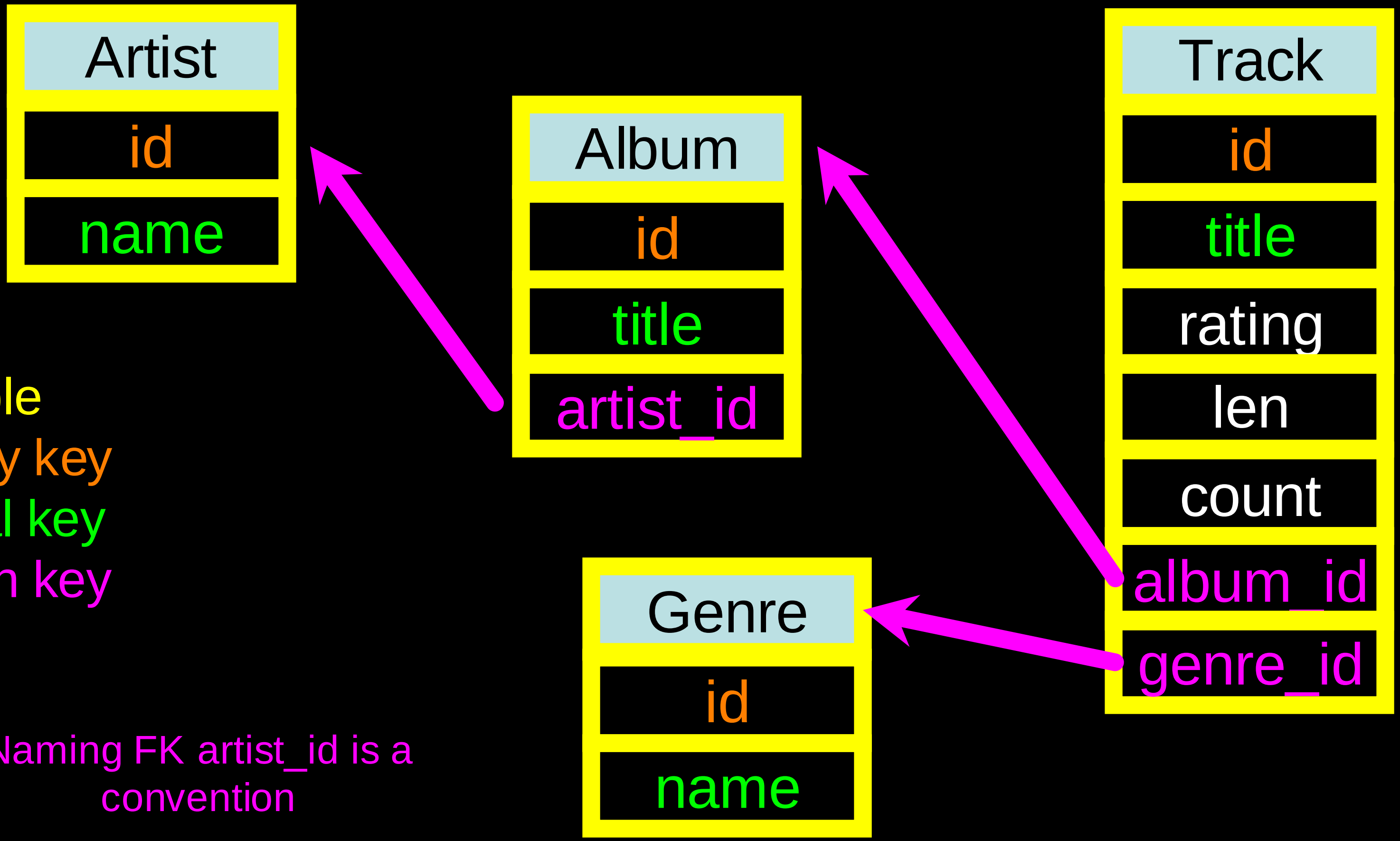
belongs-to

Track
Title
Rating
Len
Count

Album
- id
- title

Track
- id
- title
- rating
- len
- count
- album_id

Table
Primary key
Logical key
Foreign key



Artist

id

name

Album

id

title

artist_id

Track

id

title

rating

len

count

album_id

genre_id

Genre

id

name

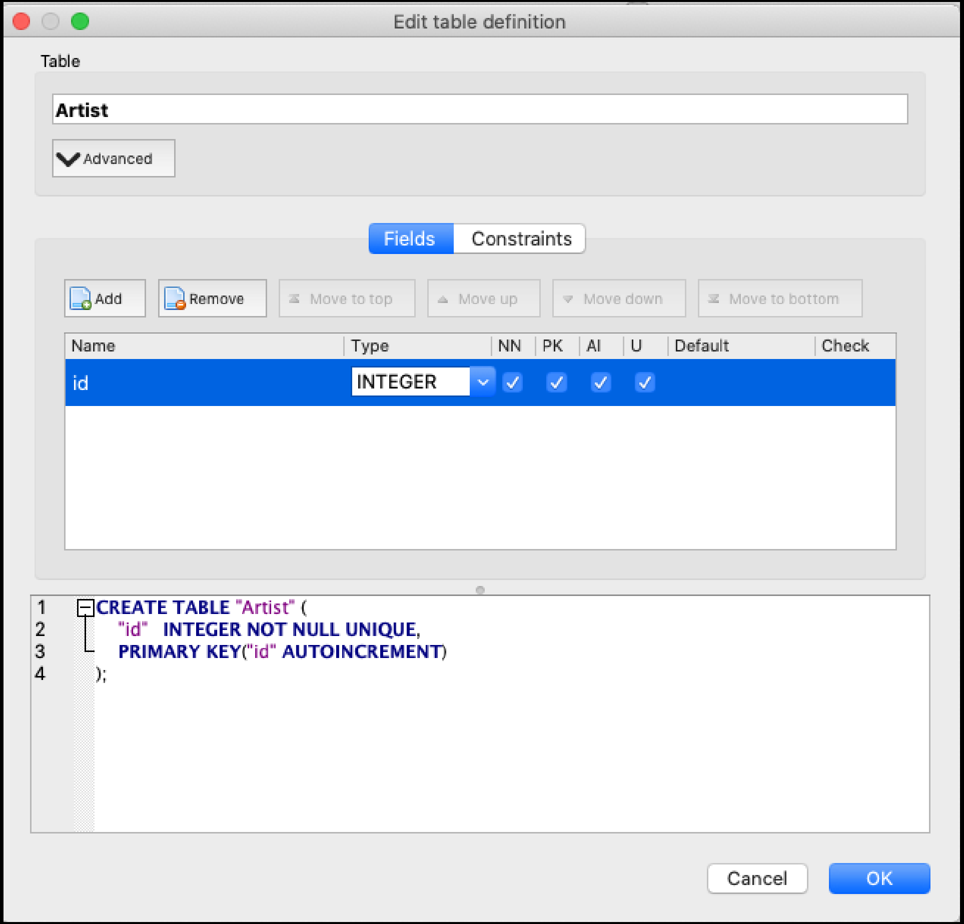
Table

Primary key

Logical key

Foreign key

Naming FK artist_id is a convention



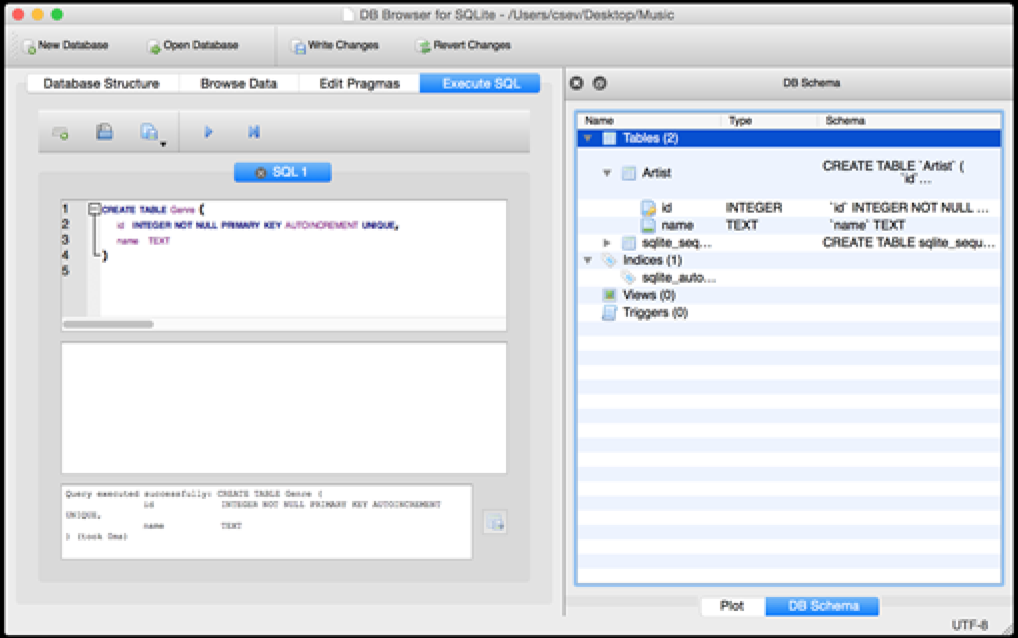
Table

Artist

Name	Type	NN	PK	AI	U	Default	Check
id	INTEGER	✓	✓	✓	✓		

```
CREATE TABLE "Artist" (
    "id"   INTEGER NOT NULL UNIQUE,
    PRIMARY KEY("id" AUTOINCREMENT)
);
```

NN = Not null
PK = Primary Key
AI = Autoincrement
U = Unique



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

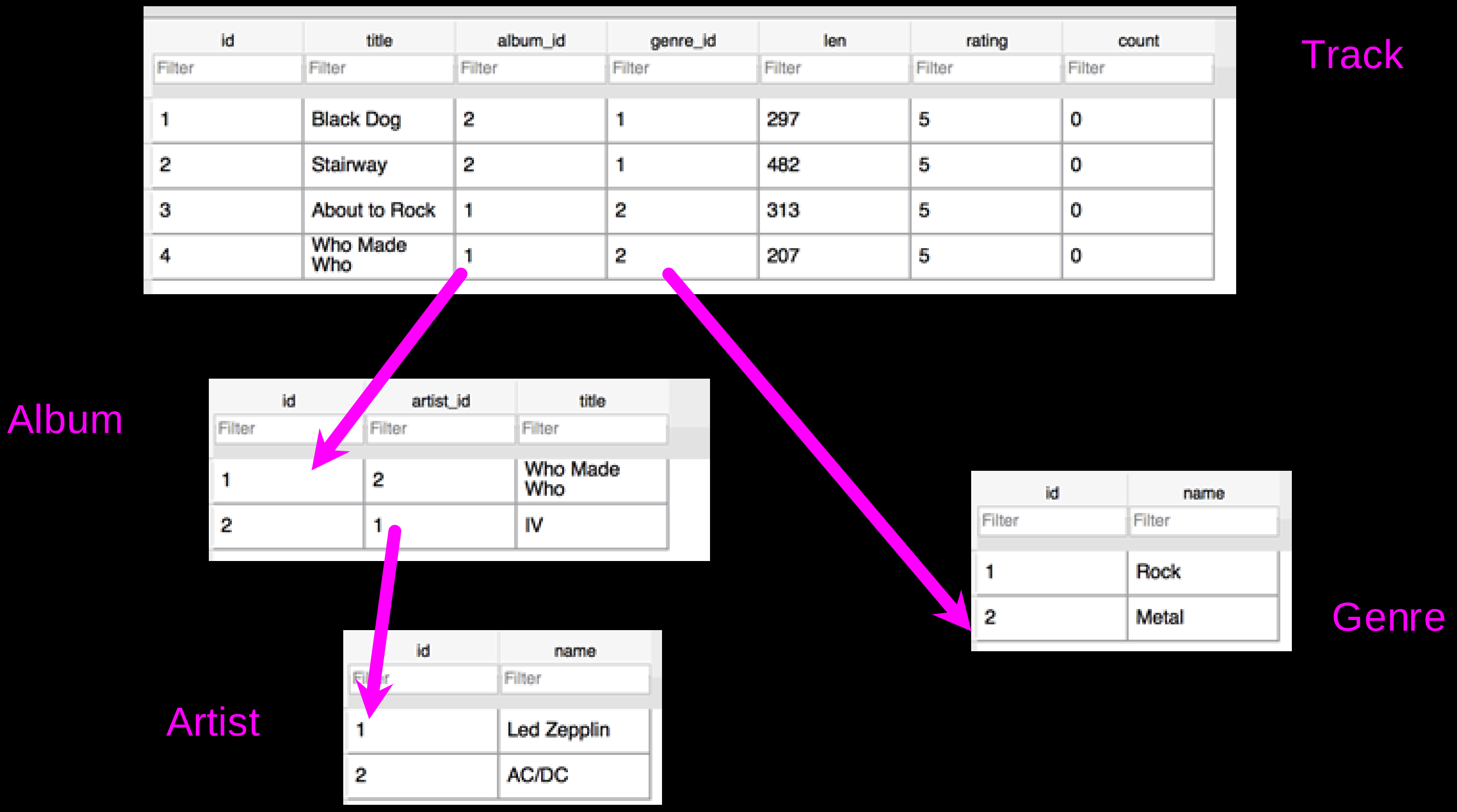
```
Query executed successfully: CREATE TABLE Genre (
                id              INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT
UNIQUE,
                name            TEXT
) (took 0ms)
```

Name	Type	Schema
Tables (2)		
Artist		CREATE TABLE `Artist` (`id` ...
id	INTEGER	`id` INTEGER NOT NULL ...
name	TEXT	`name` TEXT
sqlite_seq...		CREATE TABLE sqlite_sequ...
Indices (1)		
sqlite_auto...		
Views (0)		
Triggers (0)		

```
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
)
```



Track

id	title	album_id	genre_id	len	rating	count
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
1	2	Who Made Who
2	1	IV

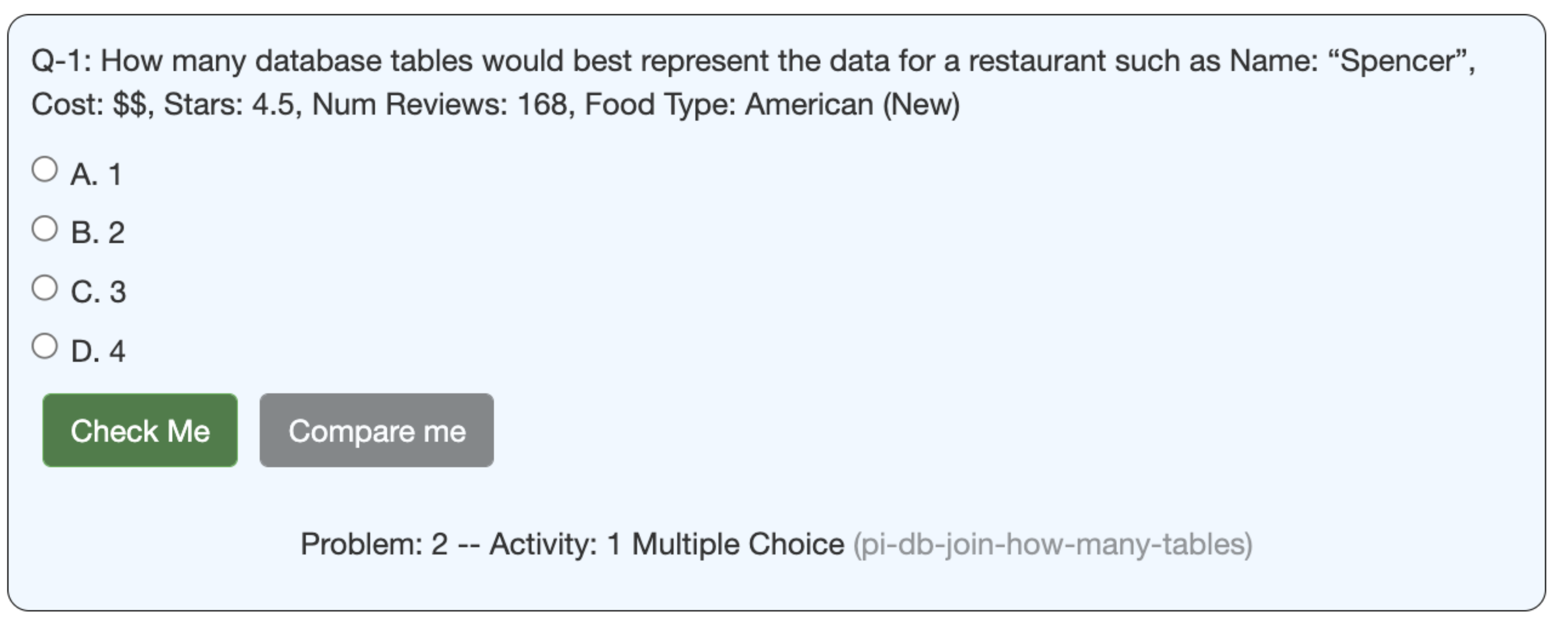
Artist

id	name
1	Led Zepplin
2	AC/DC

Genre

id	name
1	Rock
2	Metal

Peer Instruction #2



Q-1: How many database tables would best represent the data for a restaurant such as Name: “Spencer”, Cost: $$, Stars: 4.5, Num Reviews: 168, Food Type: American (New)

○ A. 1

○ B. 2

○ C. 3

○ D. 4

Problem: 2 -- Activity: 1 Multiple Choice (pi-db-join-how-many-tables)

Using Join Across Tables

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

Relational Power

- By removing the replicated data and replacing it with references to a single copy of each bit of data we build a “web” of information that the relational database can read through very quickly - even for very large amounts of data

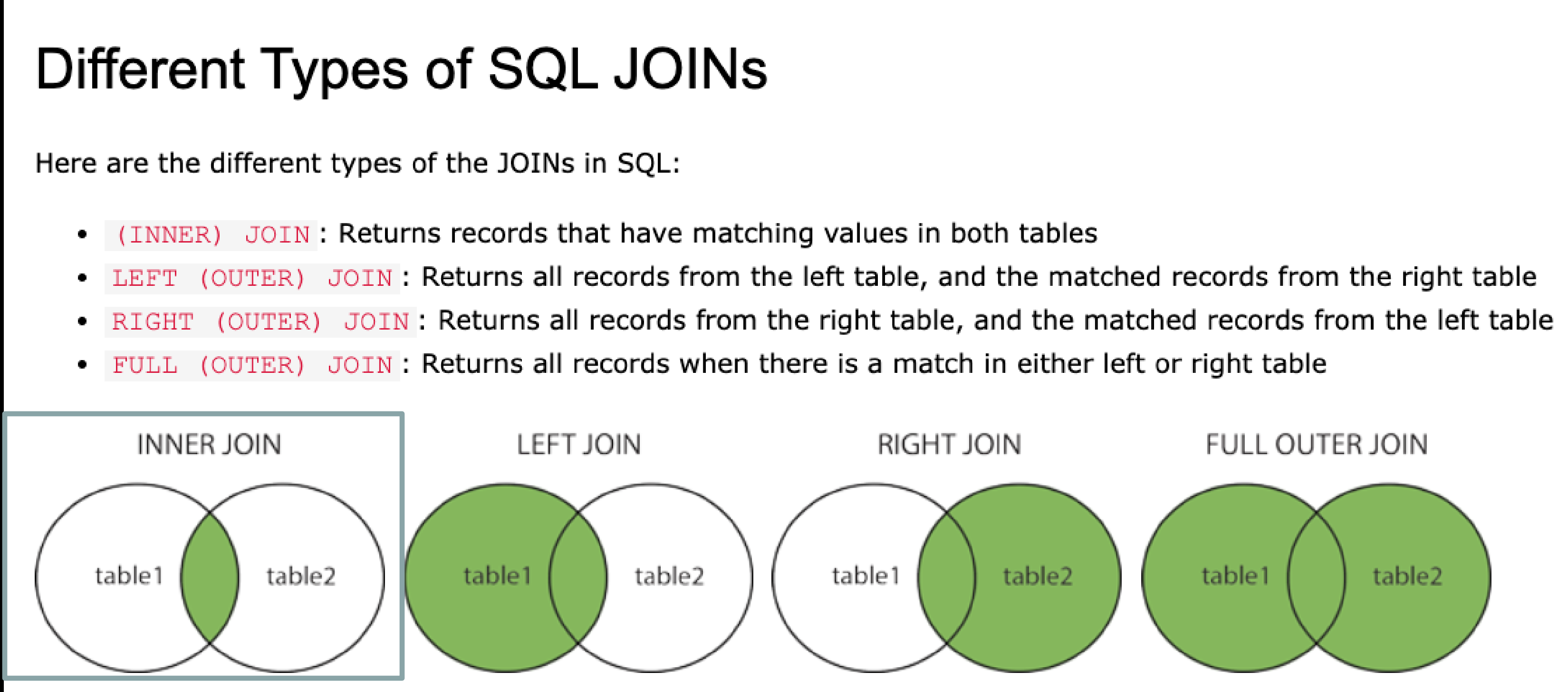
- Often when you want some data it comes from a number of tables linked by these foreign keys

The JOIN Operation

- The JOIN operation links across several tables as part of a select operation

- You must tell the JOIN how to use the keys that make the connection between the tables using an ON clause

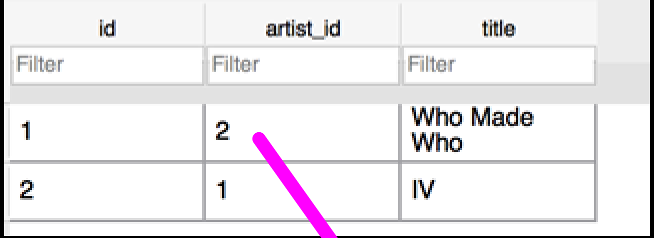
Types of Joins



Different Types of SQL JOINs

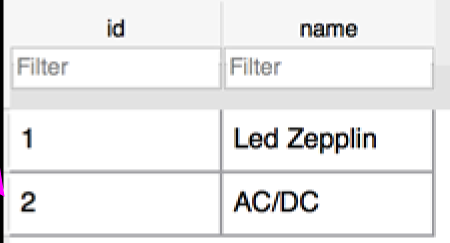
Here are the different types of the JOINs in SQL:

- `(INNER) JOIN`: Returns records that have matching values in both tables
- `LEFT (OUTER) JOIN`: Returns all records from the left table, and the matched records from the right table
- `RIGHT (OUTER) JOIN`: Returns all records from the right table, and the matched records from the left table
- `FULL (OUTER) JOIN`: Returns all records when there is a match in either left or right table



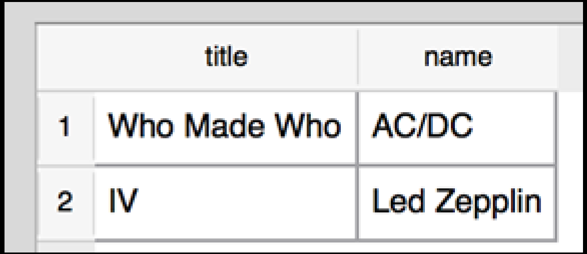
Album

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



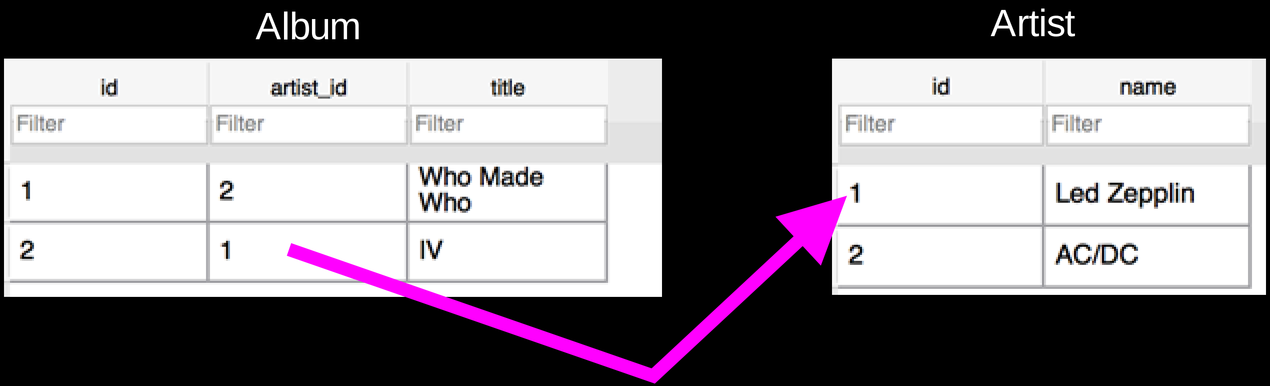
	title	name
1	Who Made Who	AC/DC
2	IV	Led Zepplin

```
SELECT Album.title, Artist.name FROM Album JOIN Artist ON Album.artist_id = Artist.id
```

What we want to see

The tables that hold the data

How the tables are linked



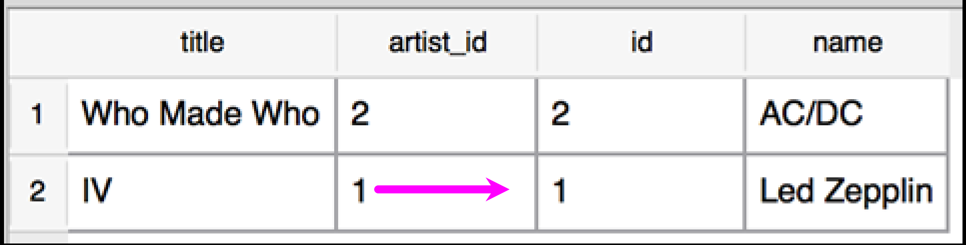
Album

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

Result

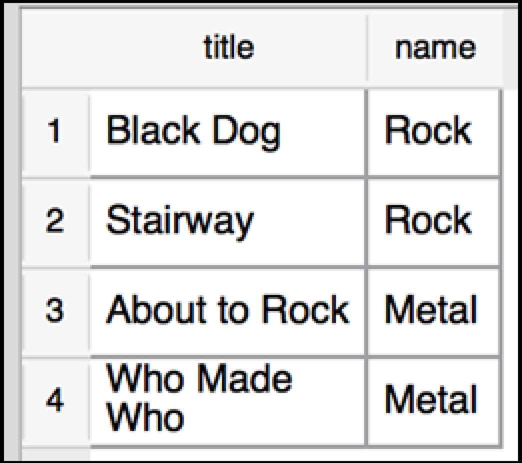


	title	artist_id	id	name
1	Who Made Who	2	2	AC/DC
2	IV	1	1	Led Zepplin

SELECT Album.title, Album.artist_id, Artist.id, Artist.name

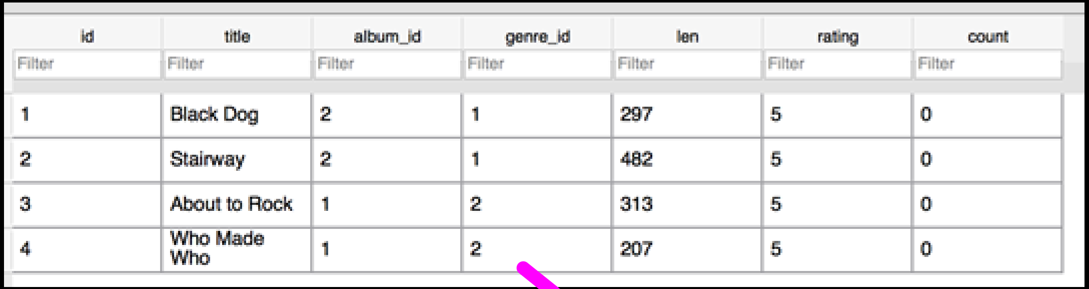
FROM Album JOIN Artist ON Album.artist_id = Artist.id

Result



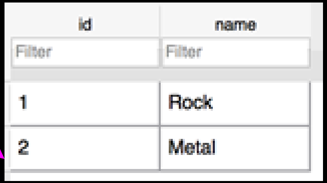
	title	name
1	Black Dog	Rock
2	Stairway	Rock
3	About to Rock	Metal
4	Who Made Who	Metal

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

Genre



id	name
Filter	Filter
1	Rock
2	Metal

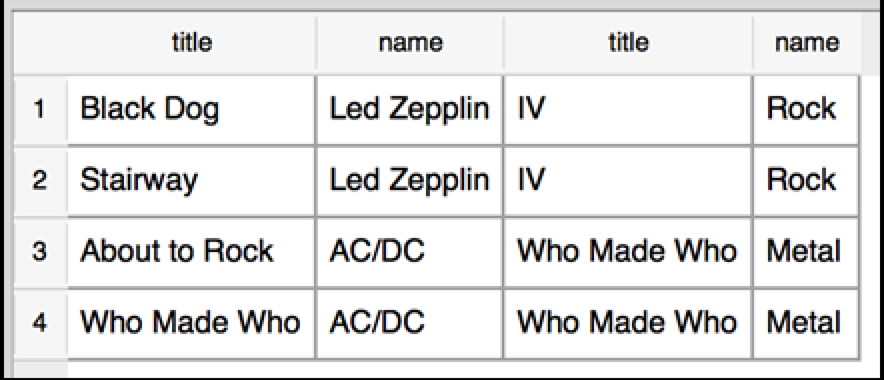
SELECT Track.title, Genre.name FROM Track JOIN Genre ON Track.genre_id = Genre.id

What we want to see

The tables that hold the data

How the tables are linked

```
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

What we want to see

The tables which hold the data

How the tables are linked

Summary

- Relational databases allow us to scale to very large amounts of data

- The key is to have one copy of any string data element and use relations and joins to link the data to multiple places

- This greatly reduces the amount of data which much be scanned when doing complex operations across large amounts of data

- Database and SQL design is a bit of an art form

Lecture Participation

- Go to the ebook and do Practice-Join for up to 10 points

Acknowledgements / Contributions

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

… Insert new Contributors here

...