

Relational Database Design

Charles Severance
www.wa4e.com



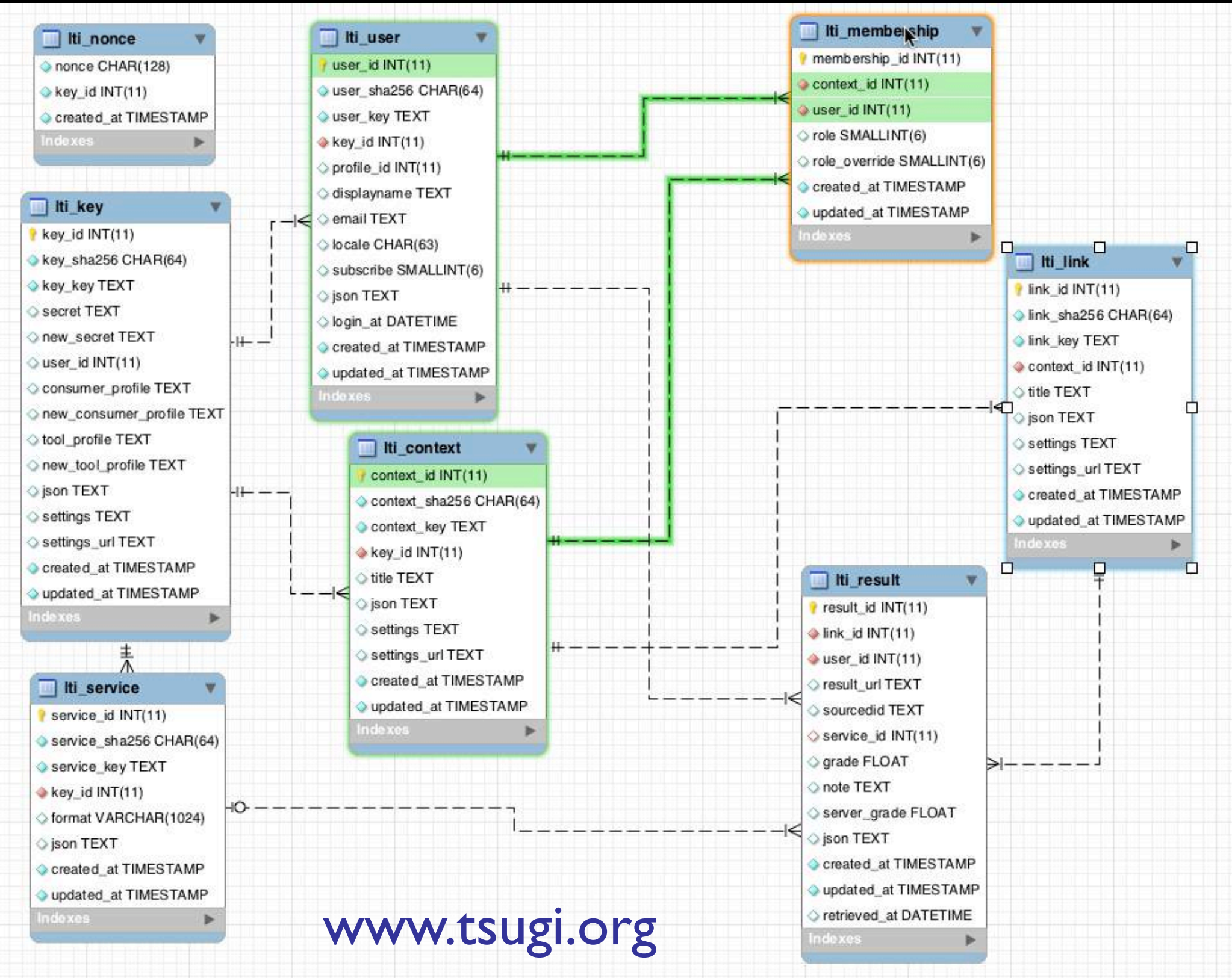
<http://www.wa4e.com/lectures/SQL-02-MySQL-Design-Handout.txt>

Relational Database Design

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

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.
- Database design starts with a picture...





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 twice - use a relationship instead
- When there is one thing in the “real world” there should only be one copy of that thing in the database

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



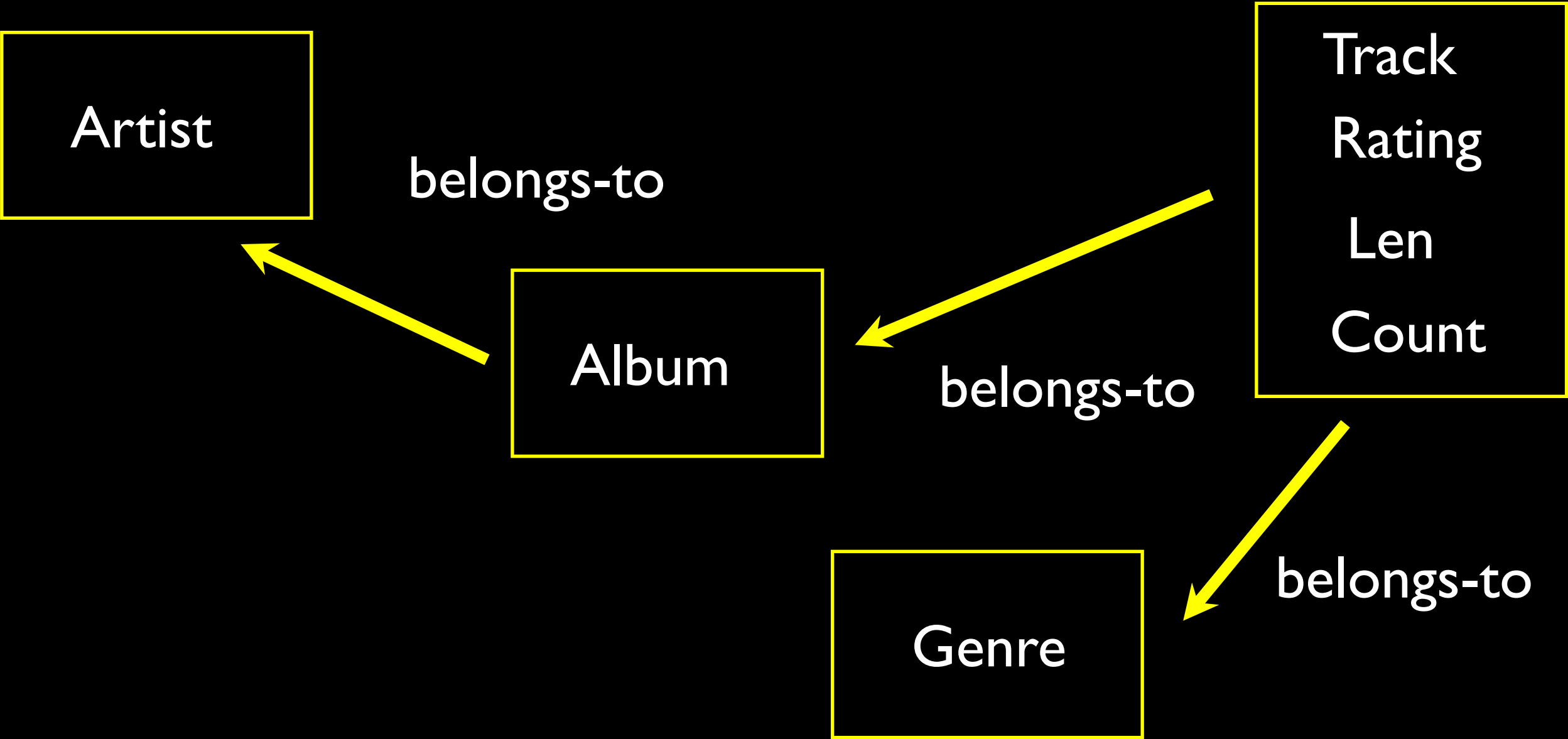
For each “piece of info”...

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

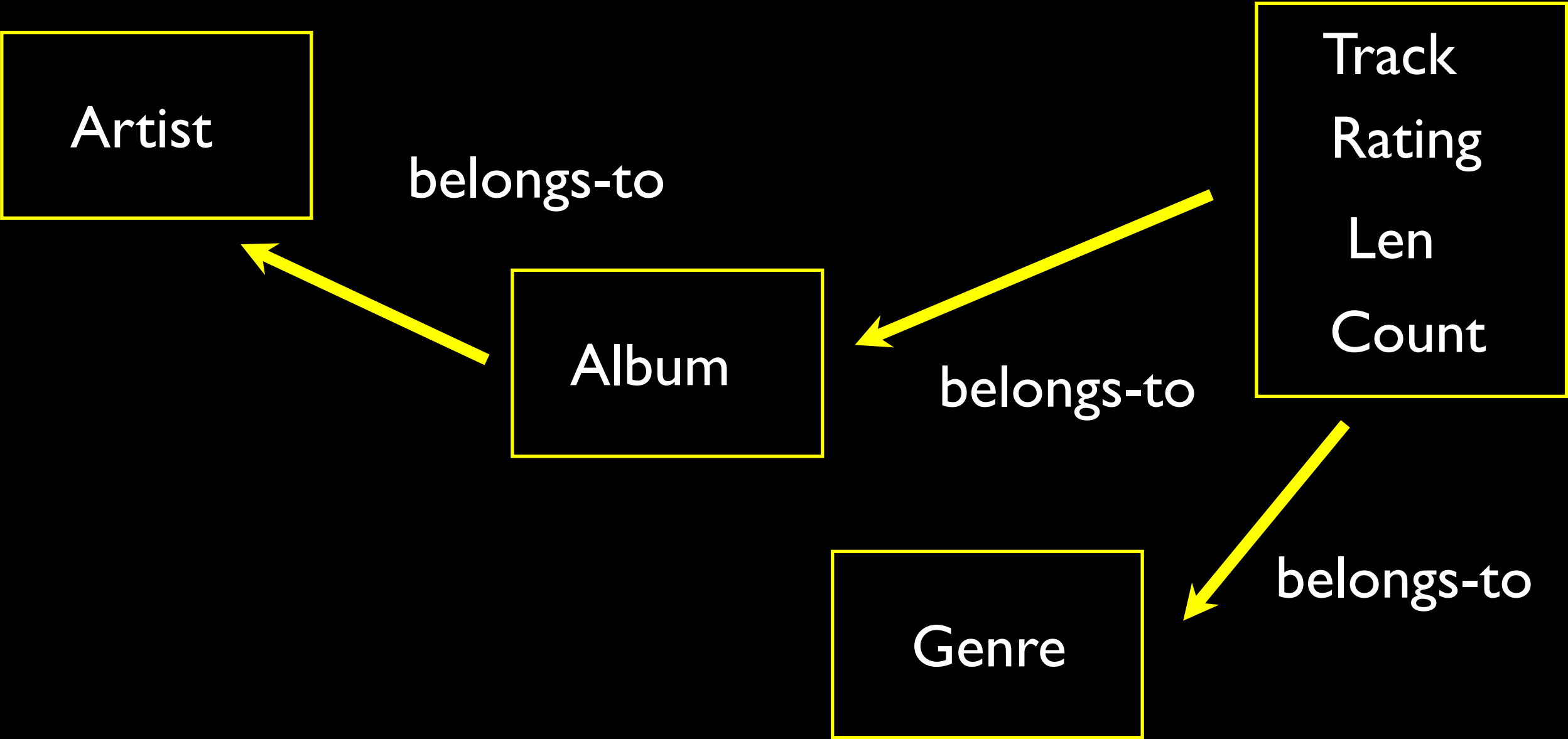
Len Album
Genre
Artist Rating
Track Count

<input checked="" type="checkbox"/> Hells Bells	5:13	AC/DC	Who Made Who	Rock	★★★★★	61
<input checked="" type="checkbox"/> Shake Your Foundations	3:54	AC/DC	Who Made Who	Rock	★★★★★	70
<input checked="" type="checkbox"/> Chase the Ace	3:01	AC/DC	Who Made Who	Rock		56
<input checked="" type="checkbox"/> For Those About To Rock (We ...	5:54	AC/DC	Who Made Who	Rock	★★★★★	61
<input checked="" type="checkbox"/> Dúlamán	3:43	Altan	Natural Wonders M...	New Age		31
<input checked="" type="checkbox"/> Rode Across the Desert	4:10	America	Greatest Hits	Easy Listen...	★★★★★	23
<input checked="" type="checkbox"/> Now You Are Gone	3:08	America	Greatest Hits	Easy Listen...	★★★★★	18
<input checked="" type="checkbox"/> Tin Man	3:20	America	Greatest Hits	Easy Listen...	★★★★★	22

Track
Album
Artist
Genre
Rating
Len
Count



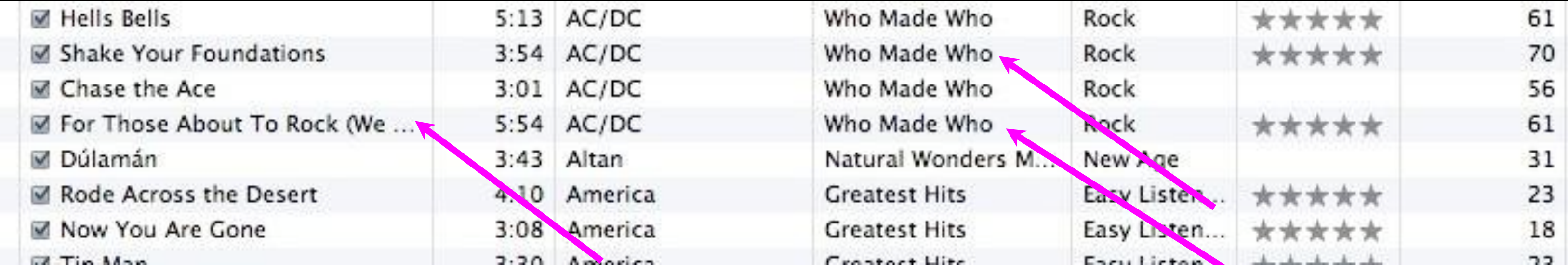
<input checked="" type="checkbox"/> Hells Bells	5:13	AC/DC	Who Made Who	Rock	★★★★★	61
<input checked="" type="checkbox"/> Shake Your Foundations	3:54	AC/DC	Who Made Who	Rock	★★★★★	70
<input checked="" type="checkbox"/> Chase the Ace	3:01	AC/DC	Who Made Who	Rock	★★★★★	56
<input checked="" type="checkbox"/> For Those About To Rock (We ...	5:54	AC/DC	Who Made Who	Rock	★★★★★	61
<input checked="" type="checkbox"/> Dúlamán	3:43	Altan	Natural Wonders M...	New Age	★★★★★	31
<input checked="" type="checkbox"/> Rode Across the Desert	4:10	America	Greatest Hits	Easy Listen...	★★★★★	23
<input checked="" type="checkbox"/> Now You Are Gone	3:08	America	Greatest Hits	Easy Listen...	★★★★★	18
<input checked="" type="checkbox"/> Tin Man	3:30	America	Greatest Hits	Easy Listen...	★★★★★	22



<input checked="" type="checkbox"/> Hells Bells	5:13	AC/DC	Who Made Who	Rock	★★★★★	61
<input checked="" type="checkbox"/> Shake Your Foundations	3:54	AC/DC	Who Made Who	Rock	★★★★★	70
<input checked="" type="checkbox"/> Chase the Ace	3:01	AC/DC	Who Made Who	Rock		56
<input checked="" type="checkbox"/> For Those About To Rock (We ...	5:54	AC/DC	Who Made Who	Rock	★★★★★	61
<input checked="" type="checkbox"/> Dúlamán	3:43	Altan	Natural Wonders M...	New Age		31
<input checked="" type="checkbox"/> Rode Across the Desert	4:10	America	Greatest Hits	Easy Listen...	★★★★★	23
<input checked="" type="checkbox"/> Now You Are Gone	3:08	America	Greatest Hits	Easy Listen...	★★★★★	18
<input checked="" type="checkbox"/> Tip Man	3:30	America	Greatest Hits	Easy Listen...	★★★★★	22



Normalization and Foreign Keys



<input checked="" type="checkbox"/> Hells Bells	5:13	AC/DC	Who Made Who	Rock	★★★★★	61
<input checked="" type="checkbox"/> Shake Your Foundations	3:54	AC/DC	Who Made Who	Rock	★★★★★	70
<input checked="" type="checkbox"/> Chase the Ace	3:01	AC/DC	Who Made Who	Rock		56
<input checked="" type="checkbox"/> For Those About To Rock (We ...)	5:54	AC/DC	Who Made Who	Rock	★★★★★	61
<input checked="" type="checkbox"/> Dúlamán	3:43	Altan	Natural Wonders M...	New Age		31
<input checked="" type="checkbox"/> Rode Across the Desert	4:10	America	Greatest Hits	Easy Listen...	★★★★★	23
<input checked="" type="checkbox"/> Now You Are Gone	3:08	America	Greatest Hits	Easy Listen...	★★★★★	18
<input checked="" type="checkbox"/> Tip Man	3:20	America	Greatest Hits	Easy Listen...	★★★★★	22

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?

Database Normalization (3NF)

There is *tons* of database theory - way too much to understand without excessive predicate calculus

- **Do not replicate data.** Instead, reference data. Point at data.
- Use **integers for keys** and for references.
- Add a special “**key**” column to each table, which you will make references to.

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



Integer Reference Pattern

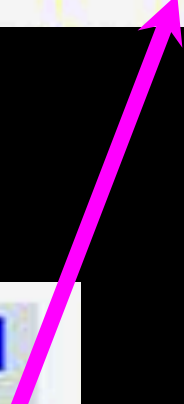
We use integer columns in one table to reference (or look up) rows in another table.

Artist

				artist_id	name
<input type="checkbox"/>				1	Led Zeppelin
<input type="checkbox"/>				2	AC/DC

Album

				album_id	title	artist_id
<input type="checkbox"/>				1	Who Made Who	2
<input type="checkbox"/>				2	IV	1



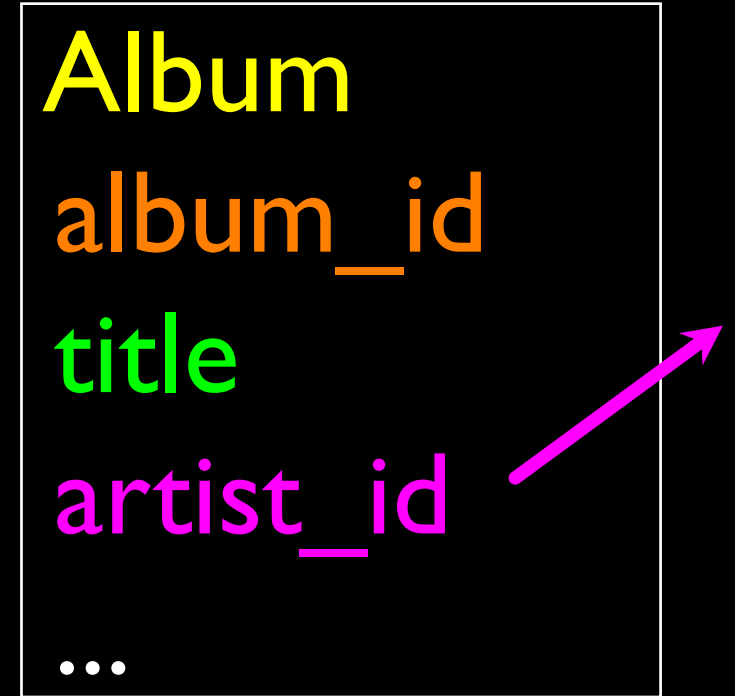


Key Terminology

Finding our way around....

Three Kinds of Keys

- **Primary key** - generally an integer auto-increment field
- **Logical key** - what the outside world uses for lookup
- **Foreign key** - generally an integer key pointing to a row in another table



Album
album_id
title
artist_id
...

Primary Key Rules

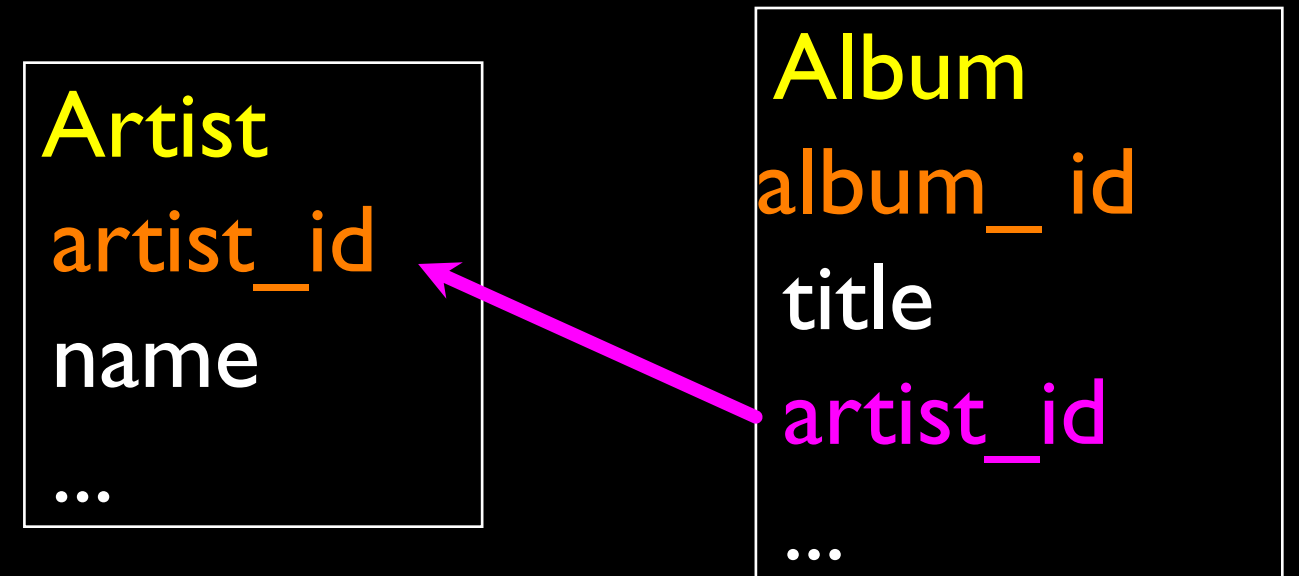
Best practices:

- 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
user_id
email
password
name
created_at
modified_at
login_at

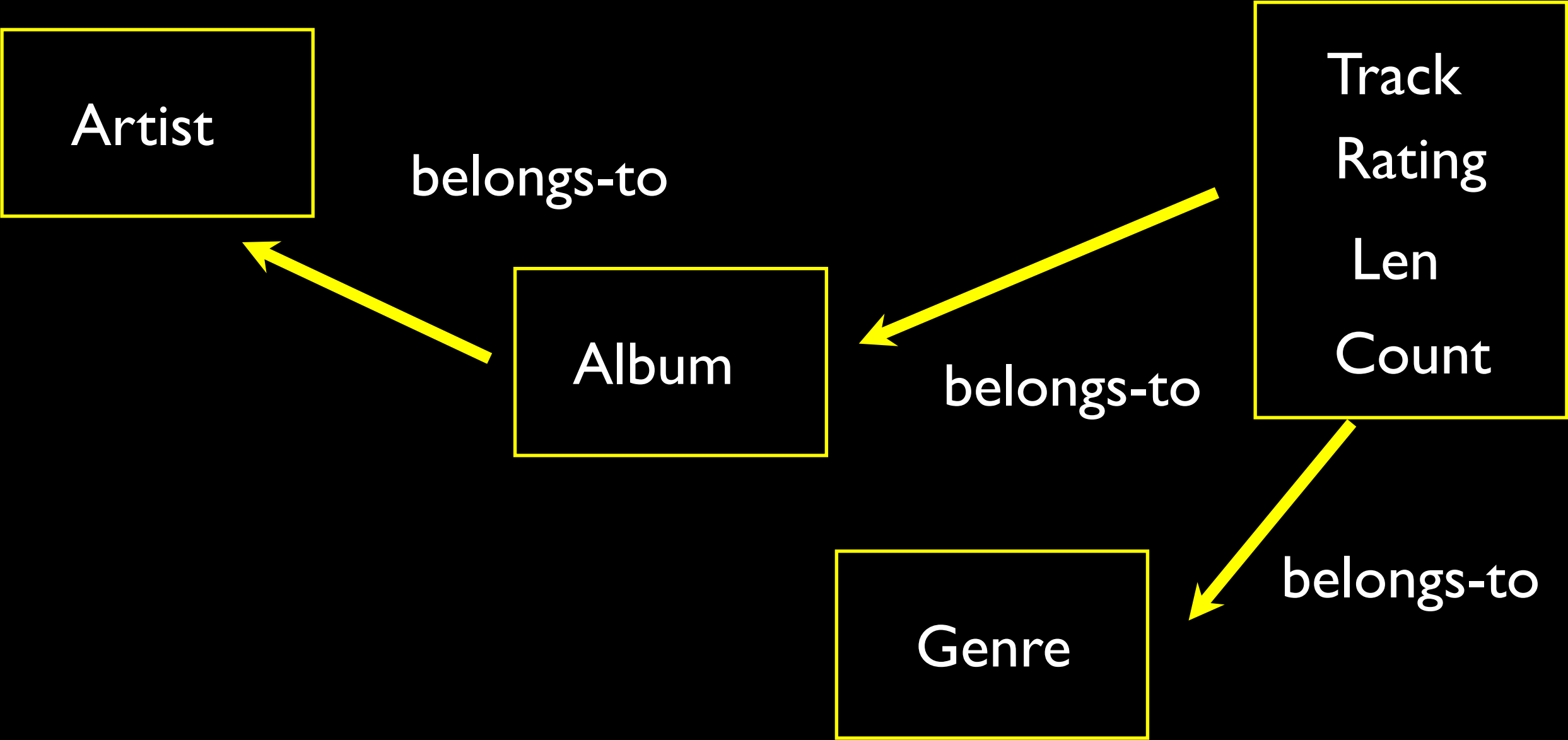
Foreign Keys

- A **foreign key** is when a table has a column containing a key that 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.

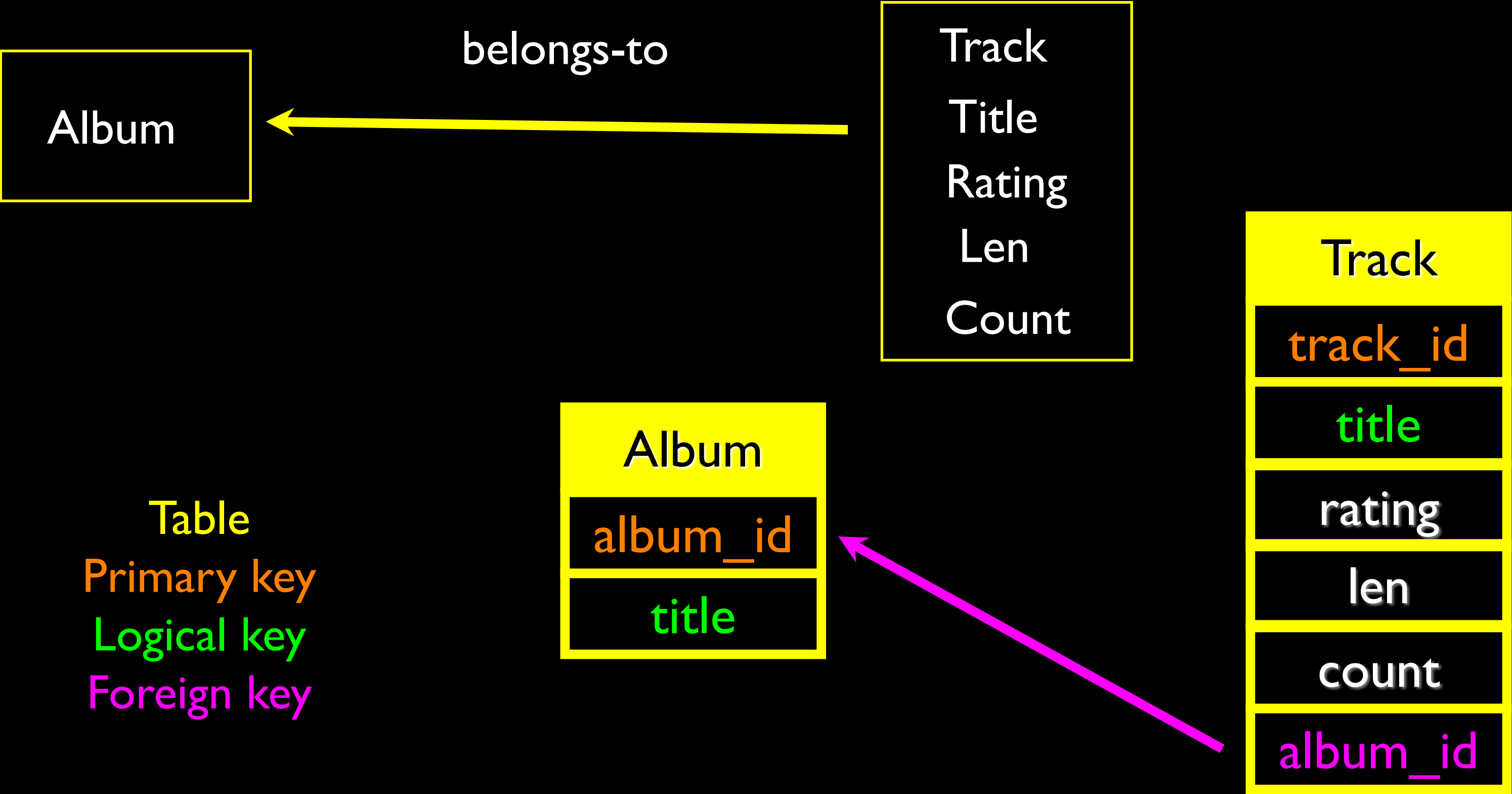


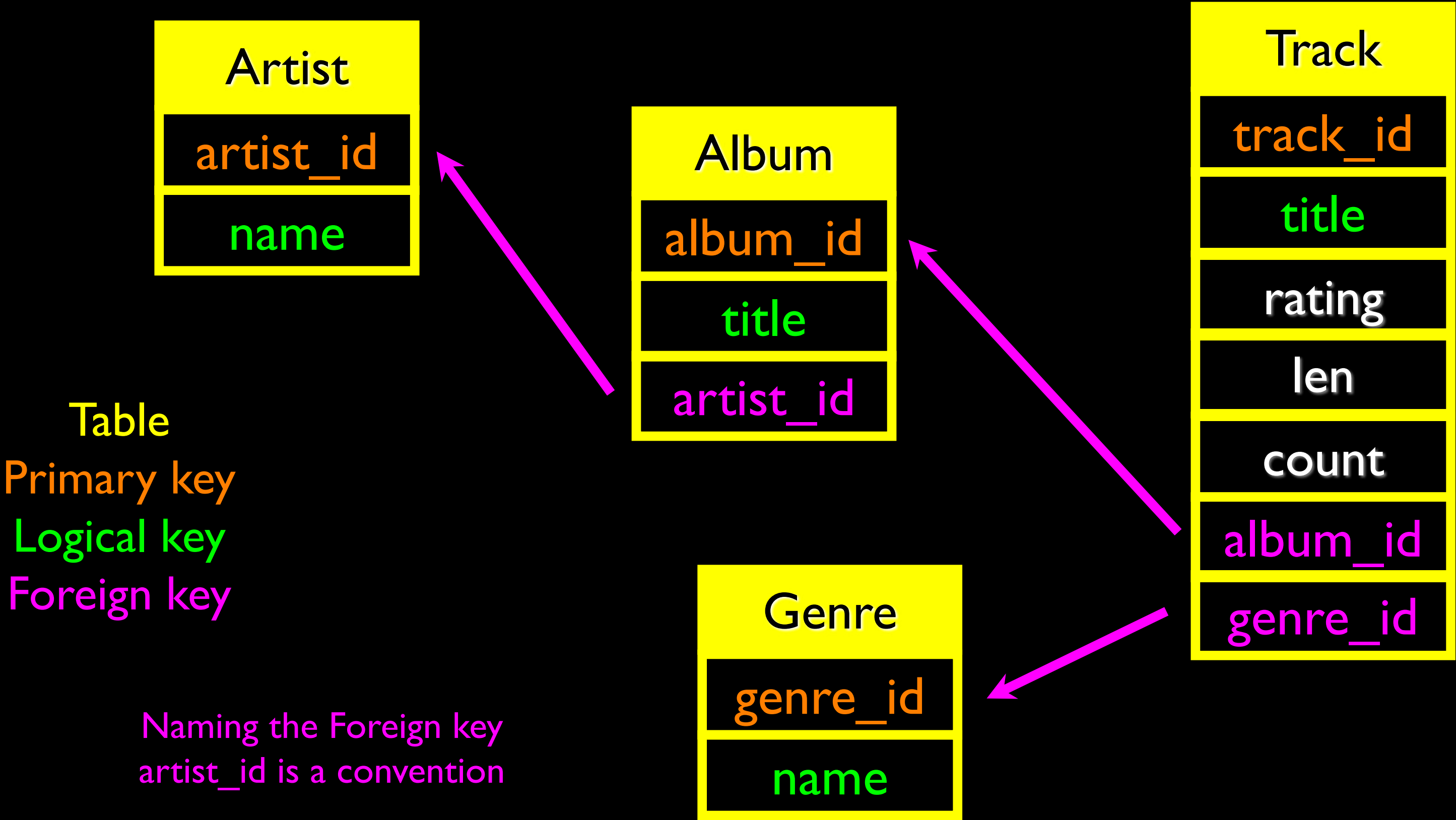


Building a Physical Data Schema



<input checked="" type="checkbox"/> Hells Bells	5:13	AC/DC	Who Made Who	Rock	★★★★★	61
<input checked="" type="checkbox"/> Shake Your Foundations	3:54	AC/DC	Who Made Who	Rock	★★★★★	70
<input checked="" type="checkbox"/> Chase the Ace	3:01	AC/DC	Who Made Who	Rock		56
<input checked="" type="checkbox"/> For Those About To Rock (We ...	5:54	AC/DC	Who Made Who	Rock	★★★★★	61
<input checked="" type="checkbox"/> Dúlamán	3:43	Altan	Natural Wonders M...	New Age		31
<input checked="" type="checkbox"/> Rode Across the Desert	4:10	America	Greatest Hits	Easy Listen...	★★★★★	23
<input checked="" type="checkbox"/> Now You Are Gone	3:08	America	Greatest Hits	Easy Listen...	★★★★★	18
<input checked="" type="checkbox"/> Tip Man	2:20	America	Greatest Hits	Easy Listen...	★★★★★	22





Creating our Music Database

```
CREATE DATABASE Music  
    DEFAULT CHARACTER SET utf8;  
  
USE Music;
```

```
CREATE TABLE Artist (  
    artist_id INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    name VARCHAR(255)  
) ENGINE = InnoDB;
```

```
CREATE TABLE Album (  
    album_id INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    title VARCHAR(255),  
    artist_id INTEGER,  
  
    INDEX USING BTREE (title),  
  
    CONSTRAINT FOREIGN KEY (artist_id)  
        REFERENCES Artist (artist_id)  
        ON DELETE CASCADE ON UPDATE CASCADE  
) ENGINE = InnoDB;
```



```
CREATE TABLE Genre (  
    genre_id INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    name VARCHAR(255),  
    INDEX USING BTREE (name)  
) ENGINE = InnoDB;  
  
CREATE TABLE Track (  
    track_id INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    title VARCHAR(255),  
    len INTEGER,  
    rating INTEGER,  
    count INTEGER,  
    album_id INTEGER,  
    genre_id INTEGER,  
  
    INDEX USING BTREE (title),  
  
    CONSTRAINT FOREIGN KEY (album_id) REFERENCES Album (album_id)  
        ON DELETE CASCADE ON UPDATE CASCADE,  
    CONSTRAINT FOREIGN KEY (genre_id) REFERENCES Genre (genre_id)  
        ON DELETE CASCADE ON UPDATE CASCADE  
) ENGINE = InnoDB;
```



MAMP

localhost:8888/MAMP/?language=English

StartphpInfoXCachephpMyAdminSQLiteManagerFAQTry MAMP PRO

MAMP & MAMP PRO powered by appsolute GmbH

localhost » Music

StructureSQLSearchQueryExportImportOperationsPrivilegesMore

Run SQL query/queries on database Music: ?

1 CREATE TABLE Artist (
2 artist_id INTEGER NOT NULL AUTO_INCREMENT KEY,
3 name VARCHAR(255)
4) ENGINE = InnoDB;

Clear

[Delimiter ;] ☒ Show this query here again ☐ Retain query box

Go

phpMyAdmin

HomeSQLHelpSQLRefresh

(Recent tables)

Music

No tables found in database.
[Create table](#)

MAMP

localhost:8888/MAMP/?language=English









StartphpInfoXCachephpMyAdminSQLiteManagerFAQTry MAMP PRO







MAMP & MAMP PRO powered by appsolute GmbH

phpMyAdmin

localhost » Music » Artist

BrowseStructureSQLSearchInsertExportImportOperationsTriggers

#	Name	Type	Collation	Attributes	Null	Default	Extra	Ac
<input type="checkbox"/>	1	artist_id	int(11)		No	None	AUTO_INCREMENT	   
<input type="checkbox"/>	2	name	varchar(255)	utf8_general_ci	Yes	NULL		   

Check All / Uncheck All With selected: 

Print view

Relation view

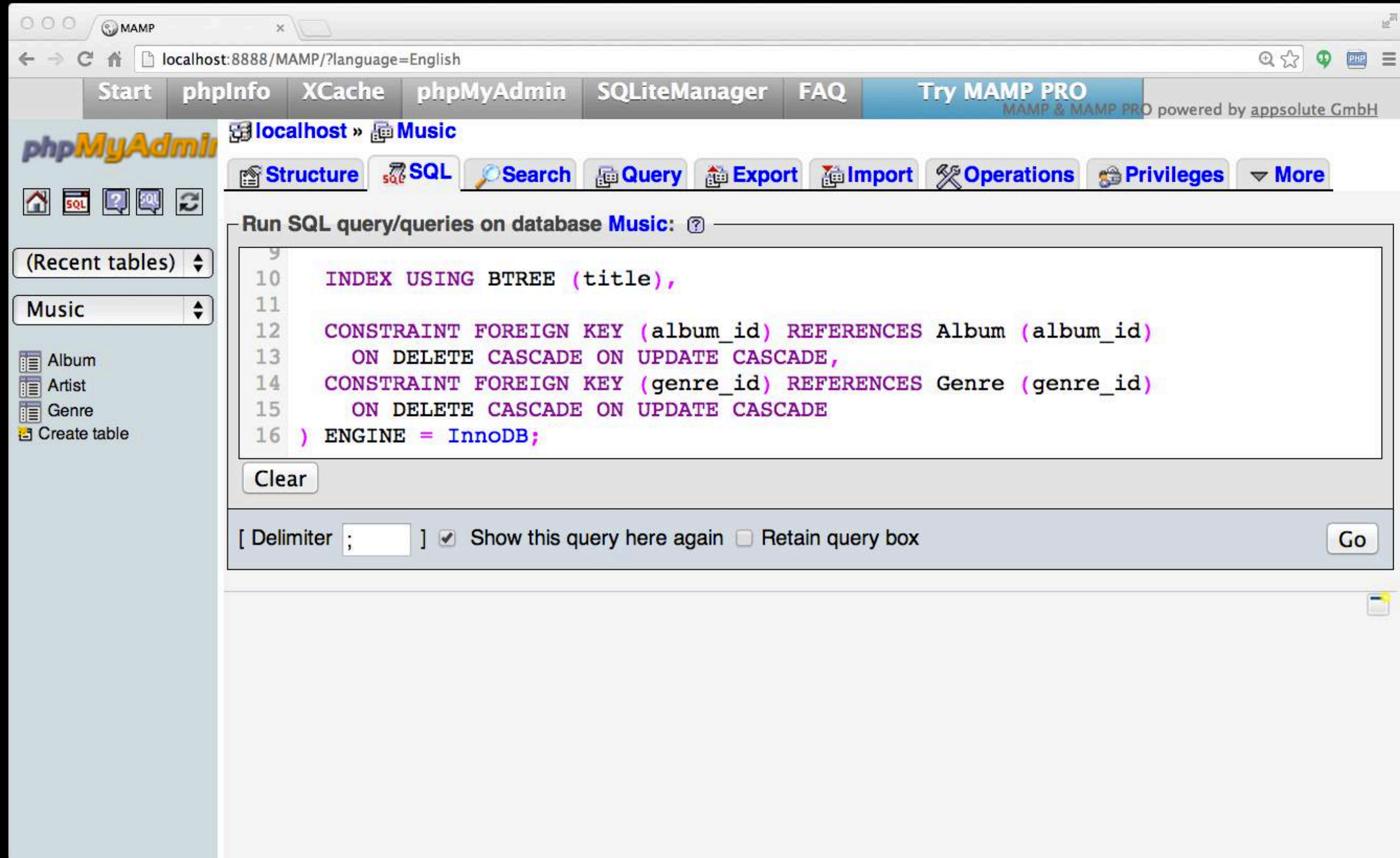
Propose table structure

Add 1 column(s) ☒ At End of Table ☐ At Beginning of Table ☐ After artist_id Go

+ Indexes

Information

Space usage			Row Statistics	
Type	Usage		Statements	Value
Data	16 KiB		Format	Compact
Index	0 B		Collation	utf8_general_ci
Total	16 KiB		Next autoindex	1
			Creation	Jan 26, 2014 at 09:31 PM



The screenshot shows the phpMyAdmin web interface in a browser window. The browser address bar shows 'localhost:8888/MAMP/?language=English'. The interface has a top navigation bar with links: Start, phpInfo, XCache, phpMyAdmin, SQLiteManager, FAQ, and Try MAMP PRO. Below this is a sub-navigation bar for the 'localhost' database, showing 'Music' as the selected database. The main area has a toolbar with icons for Structure, SQL, Search, Query, Export, Import, Operations, and Privileges. The 'SQL' tab is active, and the 'Run SQL query/queries on database Music:' section is visible. The SQL query editor contains the following code:

```
9  
10 INDEX USING BTREE (title),  
11  
12 CONSTRAINT FOREIGN KEY (album_id) REFERENCES Album (album_id)  
13 ON DELETE CASCADE ON UPDATE CASCADE,  
14 CONSTRAINT FOREIGN KEY (genre_id) REFERENCES Genre (genre_id)  
15 ON DELETE CASCADE ON UPDATE CASCADE  
16 ) ENGINE = InnoDB;
```

Below the query editor is a 'Clear' button. At the bottom, there is a 'Go' button and a checkbox for 'Show this query here again' (checked) and 'Retain query box' (unchecked). The left sidebar shows a list of tables: Album, Artist, Genre, and a 'Create table' link.

MAMP

localhost:8888/MAMP/?language=English

Start phpInfo XCache phpMyAdmin SQLiteManager FAQ Try MAMP PRO

MAMP & MAMP PRO powered by appsolute GmbH

phpMyAdmin

localhost » Music » Track

Browse Structure SQL Search Insert Export Import Operations Triggers

	#	Name	Type	Collation	Attributes	Null	Default	Extra				
<input type="checkbox"/>	1	track_id	int(11)			No	None	AUTO_INCREMENT				
<input type="checkbox"/>	2	title	varchar(255)	utf8_general_ci		Yes	NULL					
<input type="checkbox"/>	3	len	int(11)			Yes	NULL					
<input checked="" type="checkbox"/>	4	rating	int(11)			Yes	NULL					
<input type="checkbox"/>	5	count	int(11)			Yes	NULL					
<input type="checkbox"/>	6	album_id	int(11)			Yes	NULL					
<input type="checkbox"/>	7	genre_id	int(11)			Yes	NULL					

Check All / Uncheck All With selected:

Print view

Relation view

Propose table structure

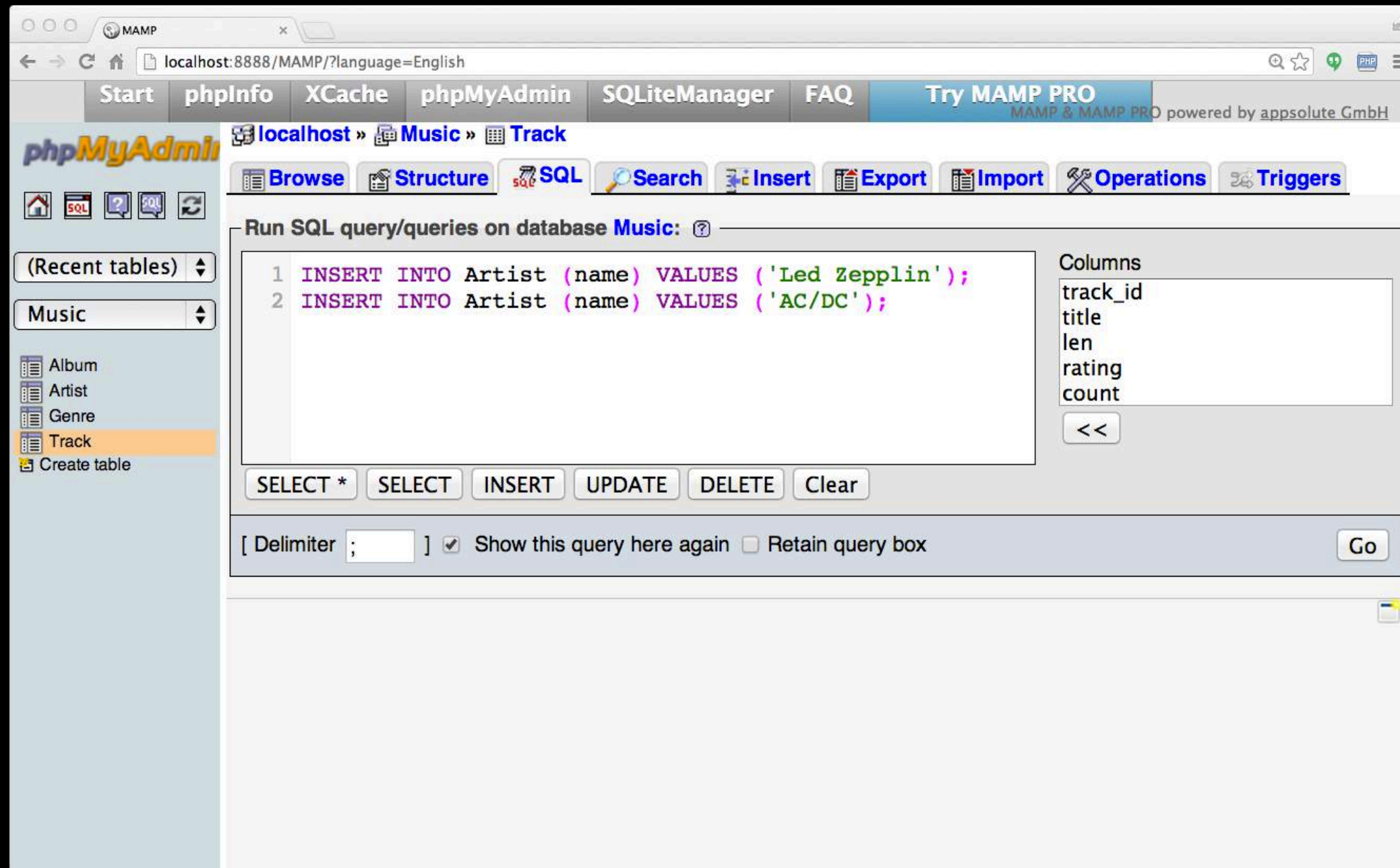
Add 1 column(s) ☒ At End of Table ☐ At Beginning of Table ☐ After track_id

Go

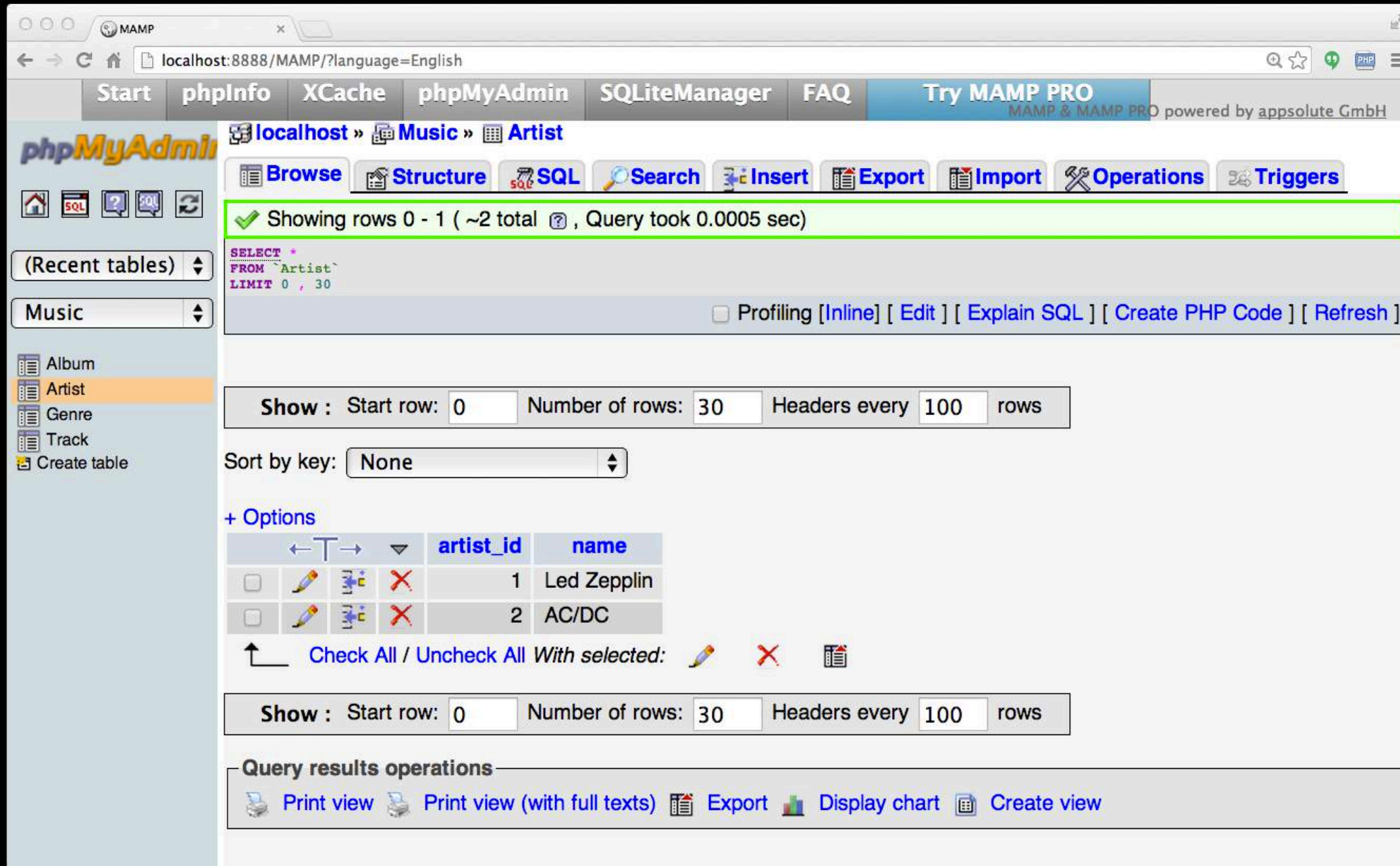
+ Indexes

Information

Space usage			Row Statistics	
Type	Usage		Statements	Value
Data	16 KiB		Format	Compact
Index	48 KiB		Collation	utf8_general_ci
Total	64 KiB		Next autoindex	1



```
INSERT INTO Artist (name) VALUES ('Led Zepplin');  
INSERT INTO Artist (name) VALUES ('AC/DC');
```



The screenshot shows the phpMyAdmin interface for a database named 'Music'. The 'Artist' table is selected, and the 'Structure' tab is active. The table has two columns: 'artist_id' and 'name'. The data is displayed in a table with two rows: '1' for 'Led Zeppelin' and '2' for 'AC/DC'. The SQL query editor shows the query: `SELECT * FROM `Artist` LIMIT 0, 30`. The status bar indicates 'Showing rows 0 - 1 (~2 total , Query took 0.0005 sec)'. The 'Query results operations' section includes links for 'Print view', 'Print view (with full texts)', 'Export', 'Display chart', and 'Create view'.

Showing rows 0 - 1 (~2 total , Query took 0.0005 sec)

```
SELECT *
FROM `Artist`
LIMIT 0, 30
```

Profiling [Inline] [Edit] [Explain SQL] [Create PHP Code] [Refresh]

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

Sort by key: None

+ Options

				artist_id	name
<input type="checkbox"/>				1	Led Zeppelin
<input type="checkbox"/>				2	AC/DC

Check All / Uncheck All With selected:

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

Query results operations

Print view Print view (with full texts) Export Display chart Create view

INSERT INTO Artist (name) VALUES ('Led Zeppelin');
INSERT INTO Artist (name) VALUES ('AC/DC');



MAMP

localhost:8888/MAMP/?language=English

Start phpInfo XCache phpMyAdmin SQLiteManager FAQ Try MAMP PRO

localhost » Music » Genre

Browse Structure SQL Search Insert Export Import Operations Triggers

Showing rows 0 - 1 (~2 total , Query took 0.0003 sec)

```
SELECT *  
FROM `Genre`  
LIMIT 0 , 30
```

Profiling [Inline] [Edit] [Explain SQL] [Create PHP Code] [Refresh]

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

Sort by key: None

+ Options

				genre_id	name
<input type="checkbox"/>				1	Rock
<input type="checkbox"/>				2	Metal

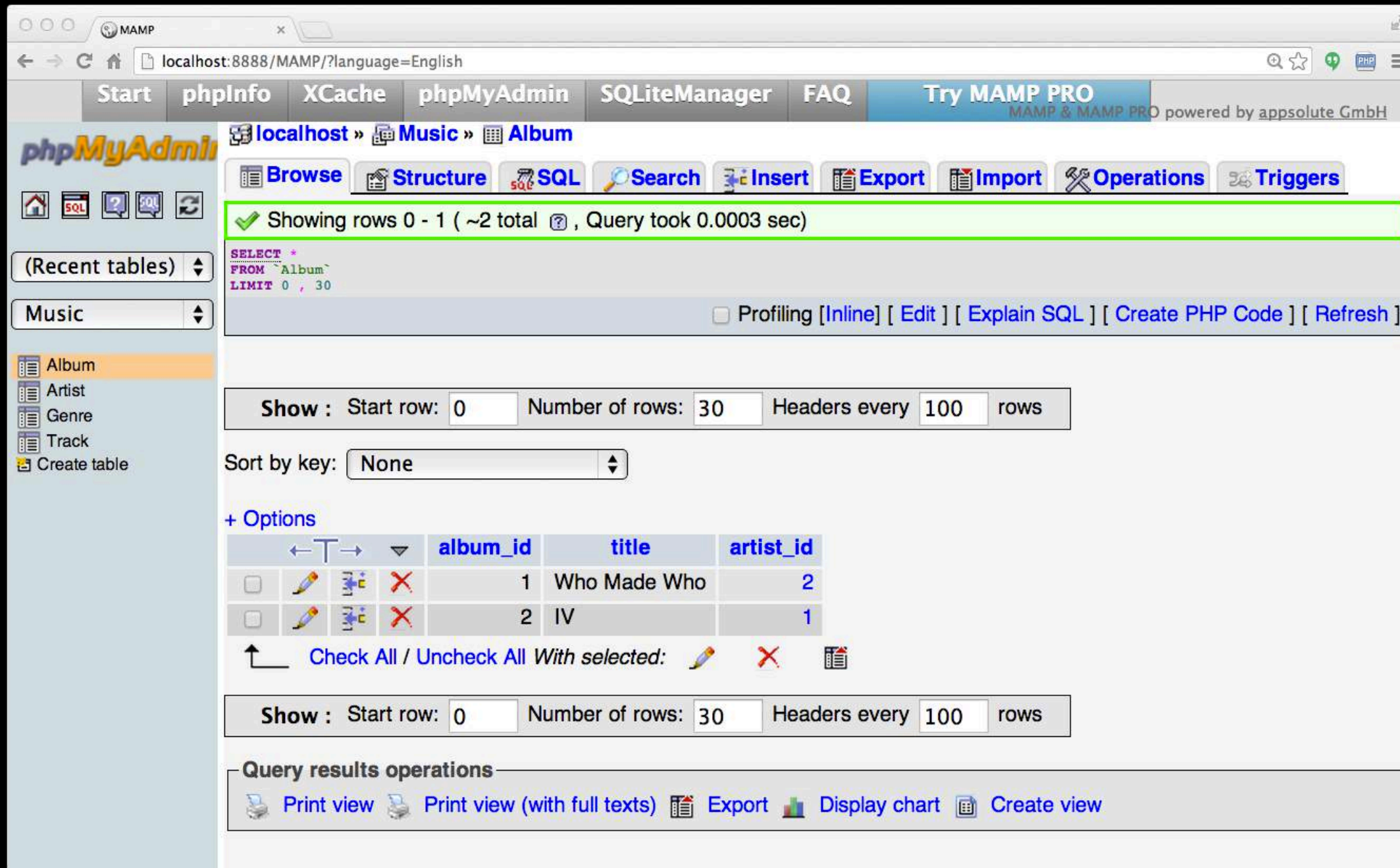
Check All / Uncheck All With selected:

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

Query results operations

Print view Print view (with full texts) Export Display chart Create view

INSERT INTO Genre (name) VALUES ('Rock');
INSERT INTO Genre (name) VALUES ('Metal');



The screenshot shows the phpMyAdmin interface for a database named 'Music'. The 'Album' table is selected, and the 'SQL' tab is active. The query results show two rows:

album_id	title	artist_id
1	Who Made Who	2
2	IV	1














The interface includes navigation tabs (Start, phpInfo, XCache, phpMyAdmin, SQLiteManager, FAQ, Try MAMP PRO), a left sidebar with a table list (Album, Artist, Genre, Track, Create table), and a top navigation bar with icons for Browse, Structure, SQL, Search, Insert, Export, Import, Operations, and Triggers. The SQL query executed is:

```
SELECT *  
FROM `Album`  
LIMIT 0 , 30
```

The results are displayed with options for sorting and pagination. The 'Query results operations' section at the bottom includes links for Print view, Print view (with full texts), Export, Display chart, and Create view.

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);
```

				track_id	title	len	rating	count	album_id	genre_id
<input type="checkbox"/>				1	Black Dog	297	5	0	2	1
<input type="checkbox"/>				2	Stairway	482	5	0	2	1
<input type="checkbox"/>				3	About to Rock	313	5	0	1	2
<input type="checkbox"/>				4	Who Made Who	207	5	0	1	2



We Have Relationships!

Track

				track_id	title	len	rating	count	album_id	genre_id
<input type="checkbox"/>				1	Black Dog	297	5	0	2	1
<input type="checkbox"/>				2	Stairway	482	5	0	2	1
<input type="checkbox"/>				3	About to Rock	313	5	0	1	2
<input type="checkbox"/>				4	Who Made Who	207	5	0	1	2

Album

				album_id	title	artist_id
<input type="checkbox"/>				1	Who Made Who	2
<input type="checkbox"/>				2	IV	1

Genre

				genre_id	name
<input type="checkbox"/>				1	Rock
<input type="checkbox"/>				2	Metal

Artist

				artist_id	name
<input type="checkbox"/>				1	Led Zeppelin
<input type="checkbox"/>				2	AC/DC

Using Join Across Tables

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

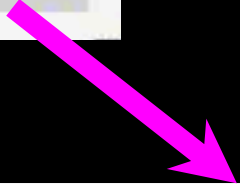
Relational Power

- 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**.

				album_id	title	artist_id
<input type="checkbox"/>				1	Who Made Who	2
<input type="checkbox"/>				2	IV	1



				artist_id	name
<input type="checkbox"/>				1	Led Zeppelin
<input type="checkbox"/>				2	AC/DC


The tables that
hold the data



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

What we want
to see

How the tables
are linked

				album_id	title	artist_id
<input type="checkbox"/>				1	Who Made Who	2
<input type="checkbox"/>				2	IV	1

				artist_id	name
<input type="checkbox"/>				1	Led Zeppelin
<input type="checkbox"/>				2	AC/DC

Album.title Album.artist_id Artist.artist_id Artist.name













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

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

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

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

Joining two tables without an **ON** clause gives all possible combinations of rows.

				track_id	title	len	rating	count	album_id	genre_id
<input type="checkbox"/>				1	Black Dog	297	5	0	2	1
<input type="checkbox"/>				2	Stairway	482	5	0	2	1
<input type="checkbox"/>				3	About to Rock	313	5	0	1	2
<input type="checkbox"/>				4	Who Made Who	207	5	0	1	2

title	name
Black Dog	Rock
Stairway	Rock
About to Rock	Metal
Who Made Who	Metal

				genre_id	name
<input type="checkbox"/>				1	Rock
<input type="checkbox"/>				2	Metal

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

It Can Get Complex...

```
SELECT Track.title, Artist.name, Album.title, Genre.name
FROM Track JOIN Genre JOIN Album JOIN Artist ON
Track.genre_id = Genre.genre_id AND Track.album_id =
Album.album_id AND Album.artist_id = Artist.artist_id
```

title	name	title	name
About to Rock	AC/DC	Who Made Who	Metal
Who Made Who	AC/DC	Who Made Who	Metal
Black Dog	Led Zeppelin	IV	Rock
Stairway	Led Zeppelin	IV	Rock

What we want
to see
The tables that
hold the data
How the tables
are linked



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

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



ON DELETE CASCADE

				track_id	title	len	rating	count	album_id	genre_id
<input type="checkbox"/>				1	Black Dog	297	5	0	2	1
<input type="checkbox"/>				2	Stairway	482	5	0	2	1
<input type="checkbox"/>				3	About to Rock	313	5	0	1	2
<input type="checkbox"/>				4	Who Made Who	207	5	0	1	2

Child

We are telling MySQL to
"clean up" broken references

				genre_id	name
<input type="checkbox"/>				1	Rock
<input type="checkbox"/>				2	Metal

Parent

```
DELETE FROM Genre WHERE name = 'Metal'
```



ON DELETE CASCADE

				track_id	title	len	rating	count	album_id	genre_id
<input type="checkbox"/>				1	Black Dog	297	5	0	2	1
<input type="checkbox"/>				2	Stairway	482	5	0	2	1
<input type="checkbox"/>				3	About to Rock	313	5	0	1	2
<input type="checkbox"/>				4	Who Made Who	207	5	0	1	2

DELETE FROM Genre WHERE name = 'Metal'

				track_id	title	len	rating	count	album_id	genre_id
<input type="checkbox"/>				1	Black Dog	297	5	0	2	1
<input type="checkbox"/>				2	Stairway	482	5	0	2	1

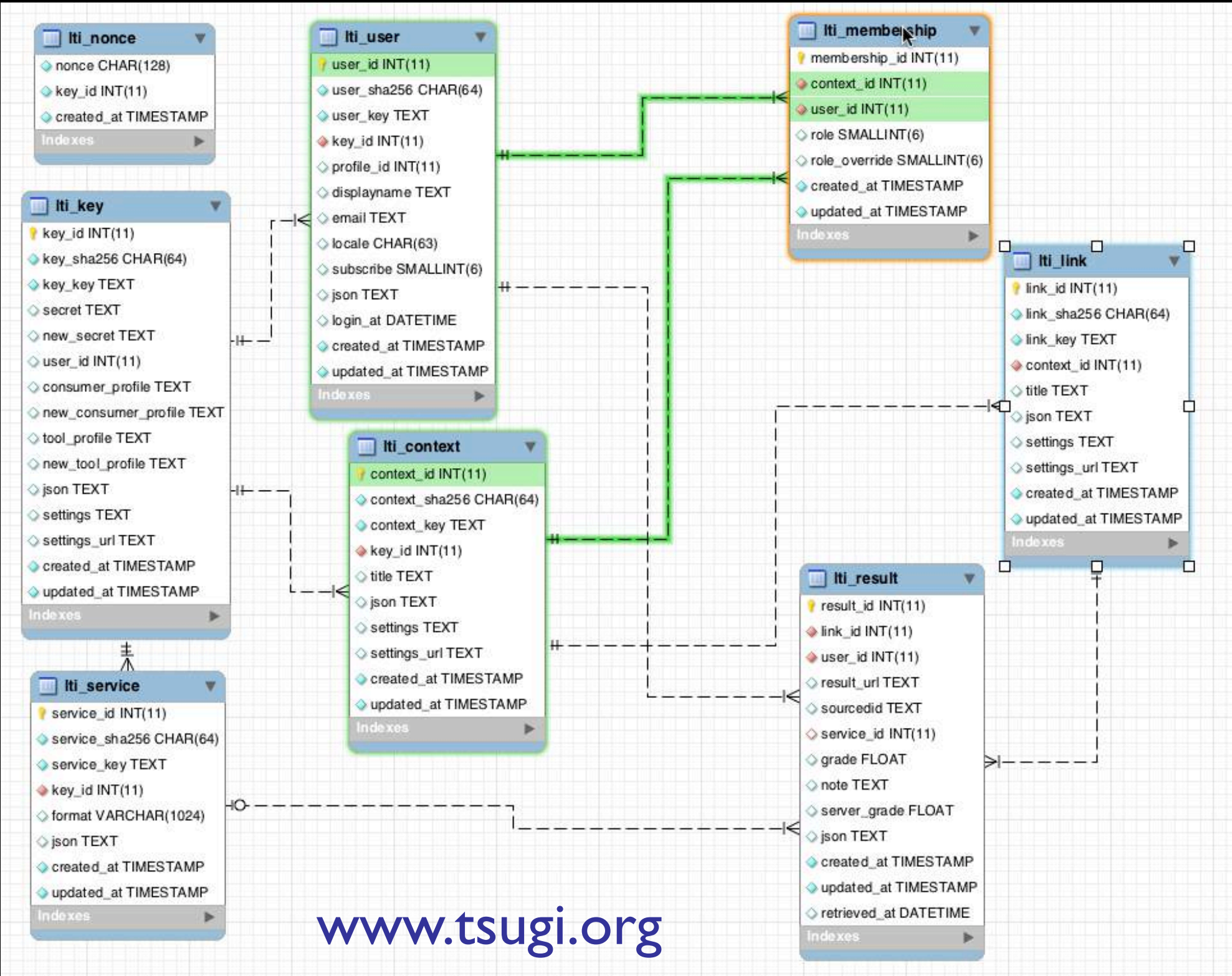
ON DELETE Choices

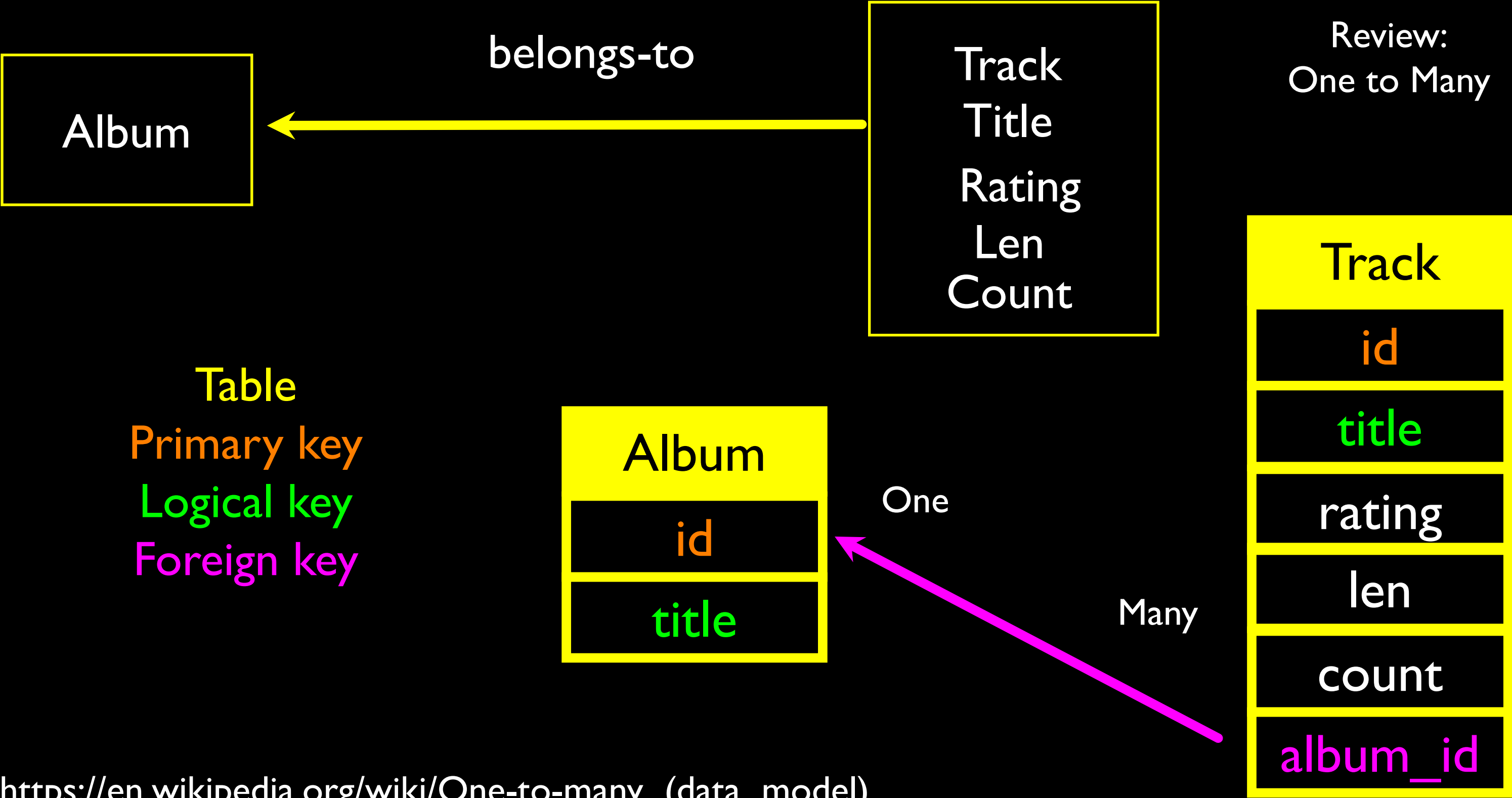
- **Default / RESTRICT** – Don't allow changes that break the constraint
- **CASCADE** – Adjust child rows by removing or updating to maintain consistency
- **SET NULL** – Set the foreign key columns in the child rows to null

<http://stackoverflow.com/questions/1027656/what-is-mysqls-default-on-delete-behavior>



Many-to-Many Relationships

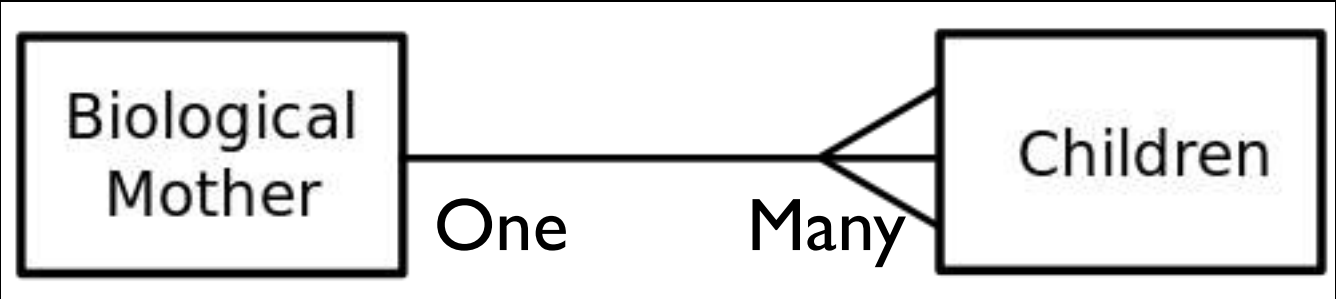






id	name
Filter	Filter
1	Rock
2	Metal

One



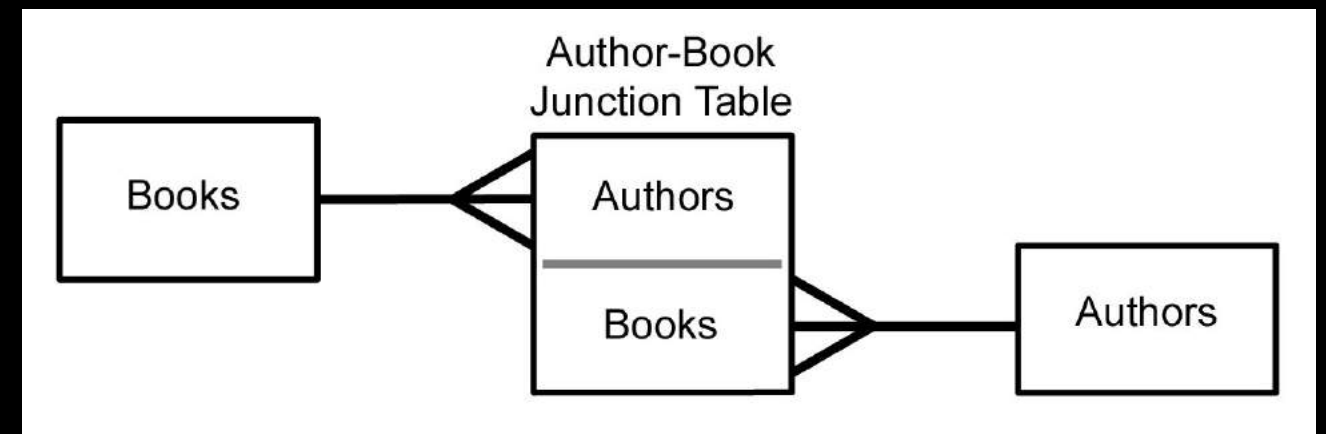
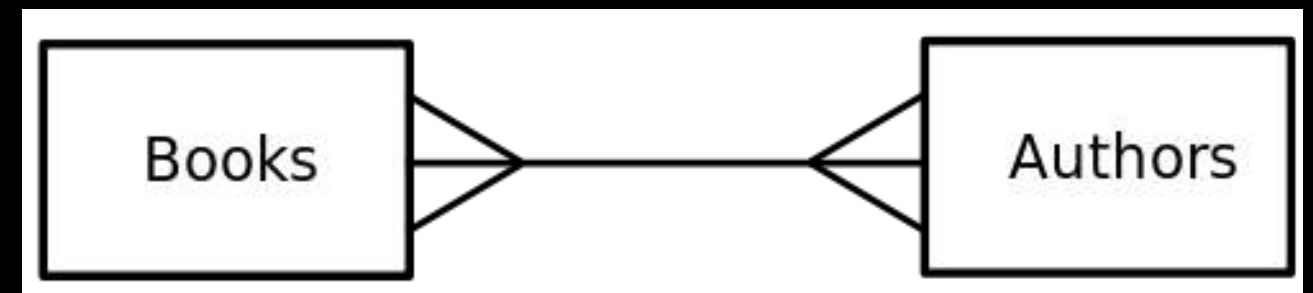
Many

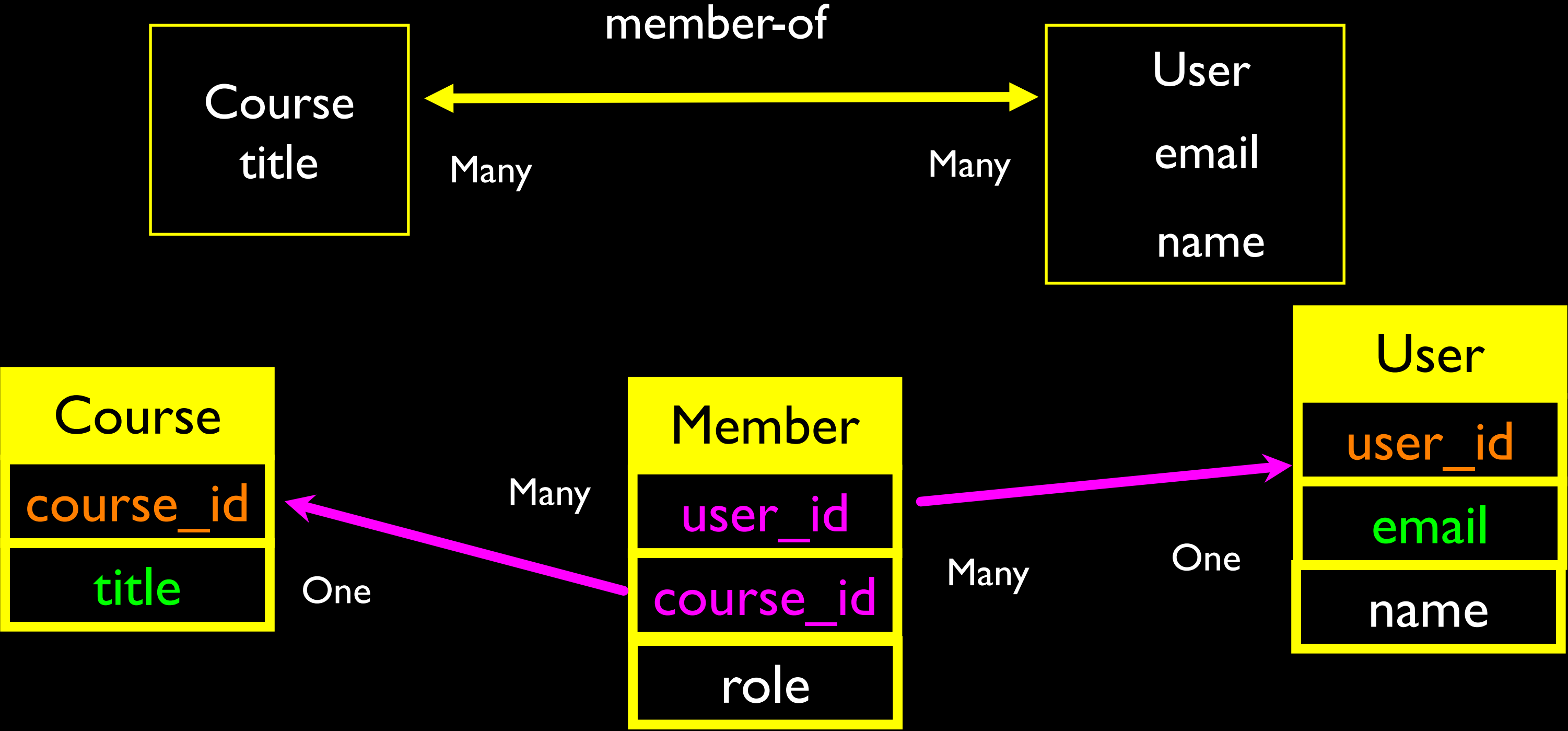
id	title	album_id	genre_id	len	rating	count
Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	Black Dog	2	1	297	5	0
2	Stairway	2	1	482	5	0
3	About to Rock	1	2	313	5	0
4	Who Made Who	1	2	207	5	0

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

Many to Many

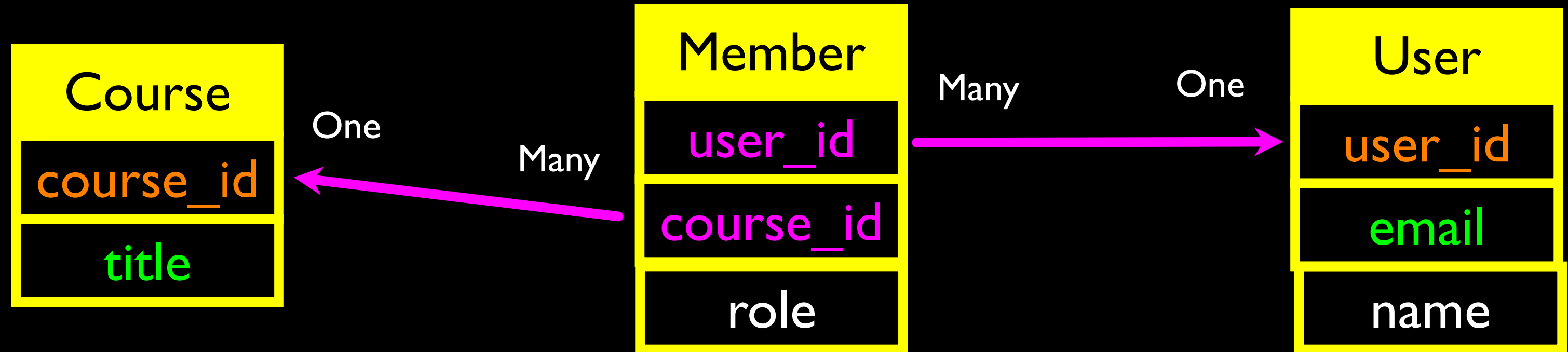
- Sometimes we need to model a relationship that is many to many.
- We need to add a “connection” table with two foreign keys.
- There is usually no separate primary key.





Start with a Fresh Database

```
CREATE TABLE User (  
    user_id      INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    email        VARCHAR(128) UNIQUE,  
    name         VARCHAR(128)  
) ENGINE=InnoDB CHARACTER SET=utf8;  
  
CREATE TABLE Course (  
    course_id    INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    title        VARCHAR(128) UNIQUE  
) ENGINE=InnoDB CHARACTER SET=utf8;
```



```
CREATE TABLE Member (  
  user_id      INTEGER,  
  course_id    INTEGER,  
  role         INTEGER,  
  
  CONSTRAINT FOREIGN KEY (user_id) REFERENCES User (user_id)  
    ON DELETE CASCADE ON UPDATE CASCADE,  
  CONSTRAINT FOREIGN KEY (course_id) REFERENCES Course (course_id)  
    ON DELETE CASCADE ON UPDATE CASCADE,  
  
  PRIMARY KEY (user_id, course_id)  
) ENGINE=InnoDB CHARACTER SET=utf8;
```

Insert Users and Courses

```
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');
```

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

course_id ▲	title
1	Python
2	SQL
3	PHP

Insert Memberships

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

course_id ▲	title
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);
```


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

course_id ▲	title
1	Python
2	SQL
3	PHP

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

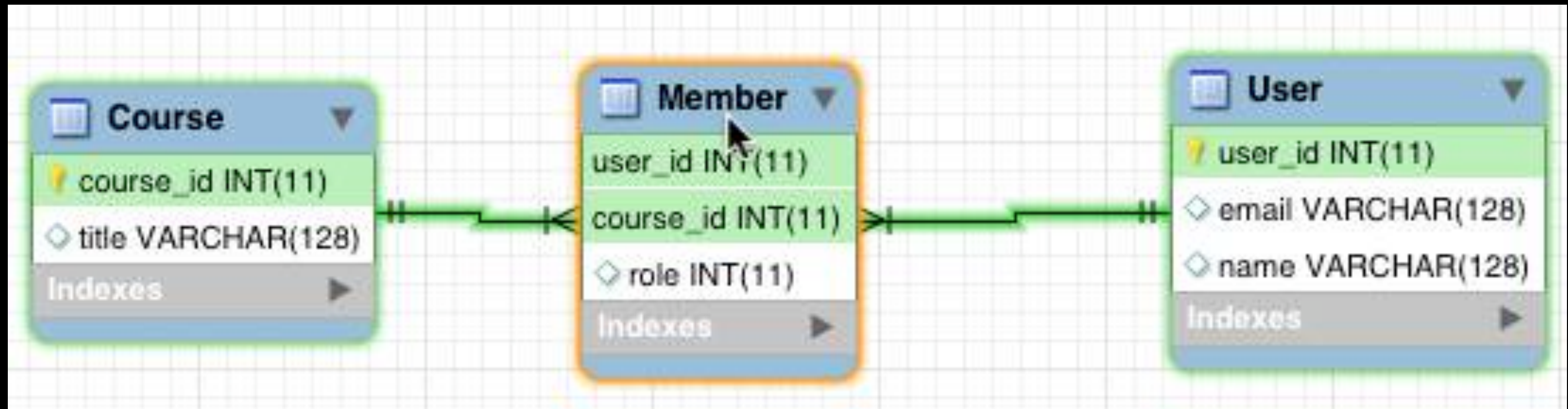
user_id	email	name
1	jane@tsugi.org	Jane
2	ed@tsugi.org	Ed
3	sue@tsugi.org	Sue

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

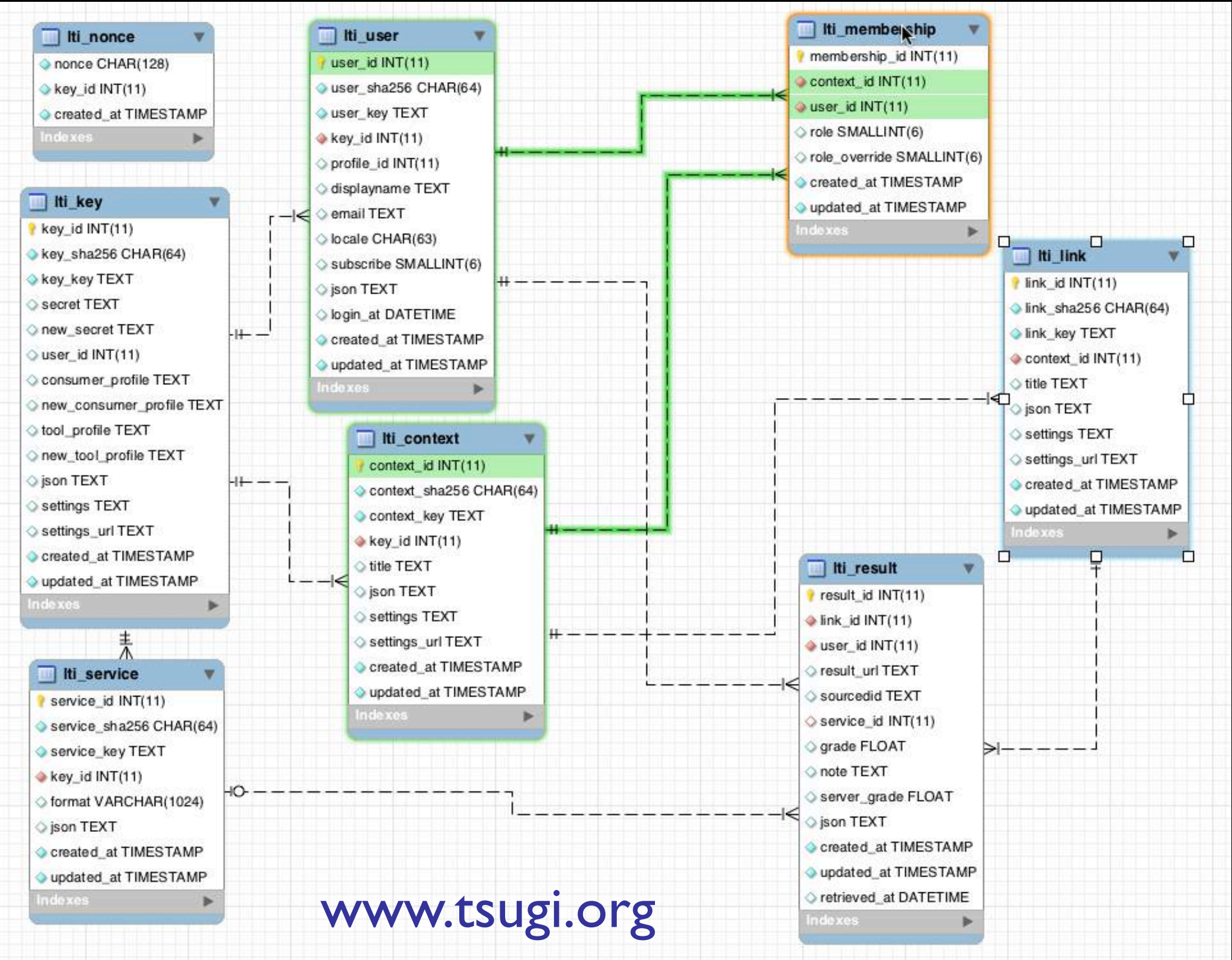
course_id	title
1	Python
2	SQL
3	PHP

name	role	title
Ed	1	PHP
Sue	0	PHP
Jane	1	Python
Ed	0	Python
Sue	0	Python
Ed	1	SQL
Jane	0	SQL

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



<https://www.mysql.com/products/workbench/>



Complexity Enables Speed

- Complexity makes speed possible and allows you to get very fast results as the data size grows.
- By normalizing the data and linking it with integer keys, the overall amount of data which the relational database must scan is far lower than if the data were simply flattened out.
- It might seem like a tradeoff - spend some time designing your database so it continues to be fast when your application is a success.

Summary

- Relational databases allow us to **scale** to very large amounts of data.
- The key is to have **one copy of any data element** and use relations and joins to link the data to multiple places.
- This greatly **reduces the amount of data that must be scanned** when doing complex operations across large amounts of data.
- Database and SQL design is a bit of an **art form**.

Acknowledgements / Contributions



These slides are Copyright 2010- Charles R. Severance (www.dr-chuck.com) as part of www.wa4e.com and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

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

Insert new Contributors and Translators here including names and dates

Continue new Contributors and Translators here