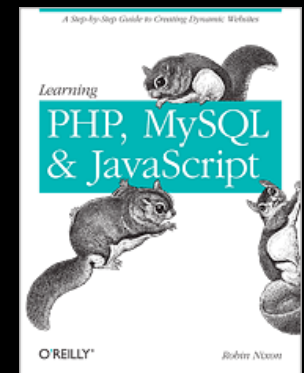


Mastering MySQL

Chapter 9

Dr. Charles Severance

To be used in association with the book:
PHP, MySQL, and JavaScript by Robin Nixon



open.michigan

Unless otherwise noted, the content of this course material is licensed under a Creative Commons Attribution 3.0 License.

<http://creativecommons.org/licenses/by/3.0/>.

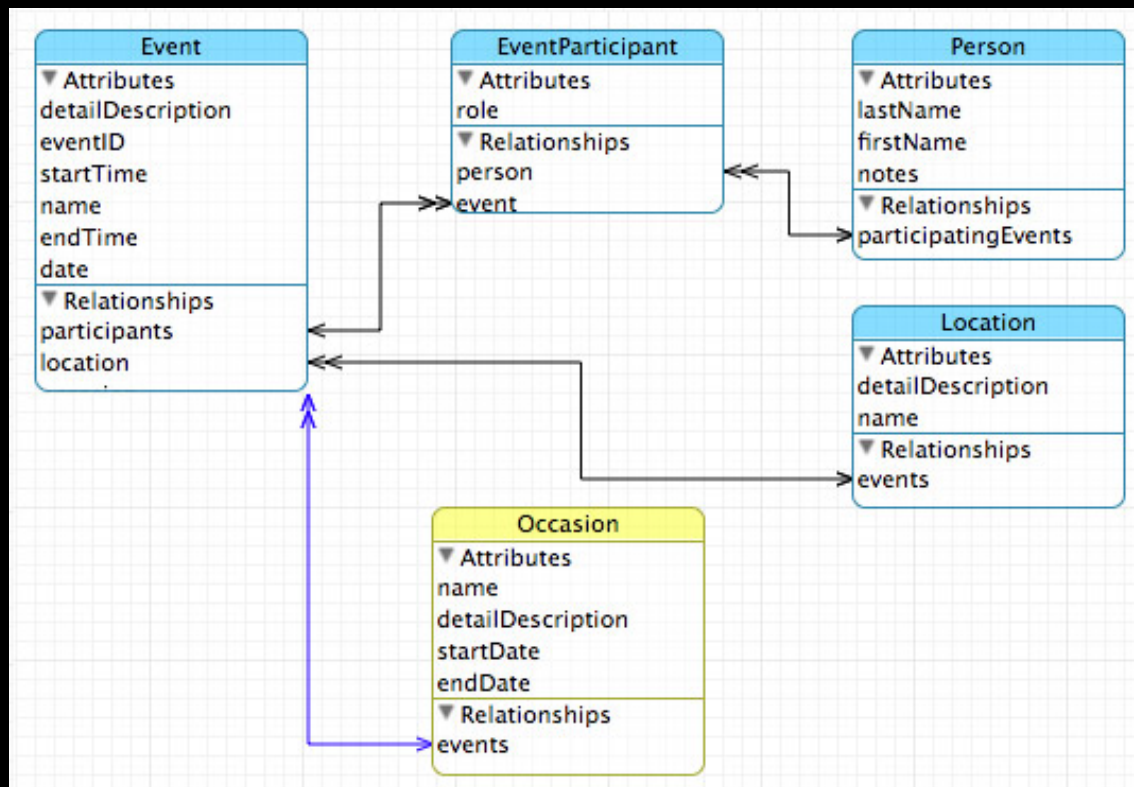
Copyright 2011, Charles Severance



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 or ends with a picture...

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



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 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

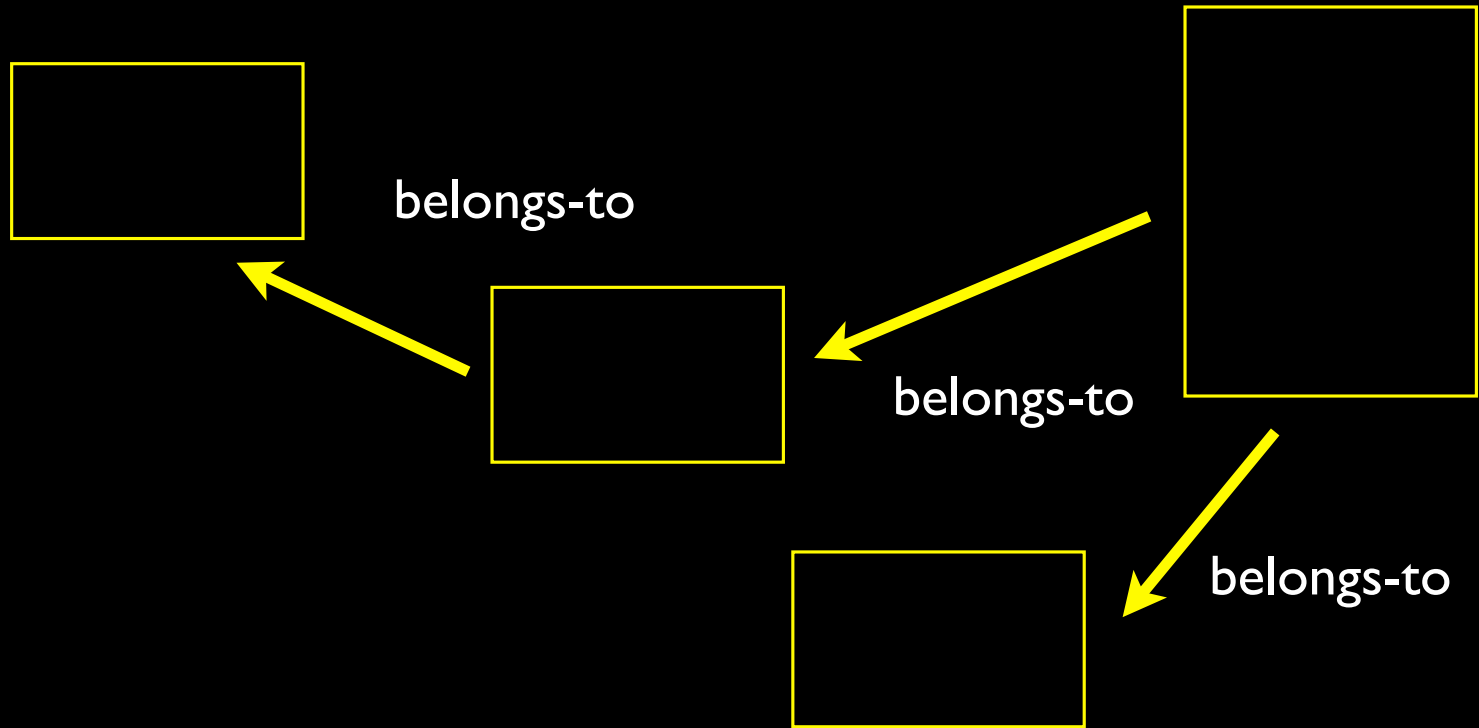
For each “piece of info”...

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

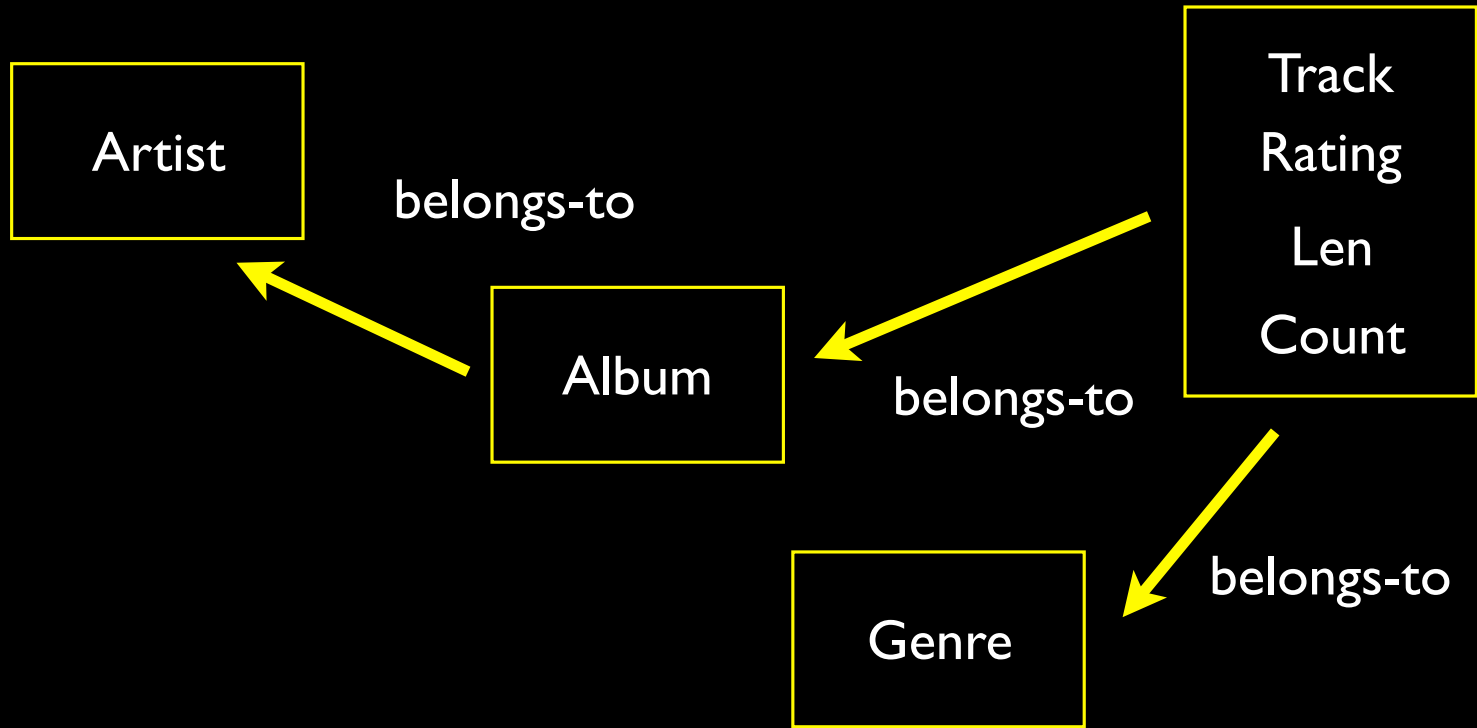
Len Album
Genre
Artist Rating
Track Count

<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"/> Tie Man	3:20	America	Greatest Hits	Easy Listen...	★★★★★	22

Track
Artist
Album
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"/>	Tie Man	3:20	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	3:20	America	Greatest Hits	Easy Listen...	★★★★★	22

Representing Relationships in a Database

<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?

Practical Normalization

- There is *tons* of database theory - we will cover some later in this lecture
 - Do not replicate data - reference data - point at data
 - 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

Integer Reference Pattern

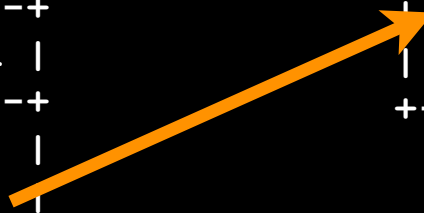
We use integer columns to reference rows in another table.

Album

id	title	artist_id
1	Who Made Who	2
2	IV	1

Artist

id	name
1	Led Zeppelin
2	AC/DC

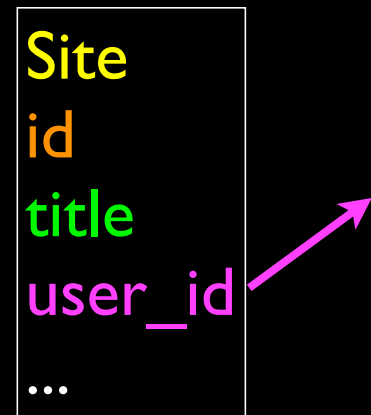


Keys

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 point to a row in another table



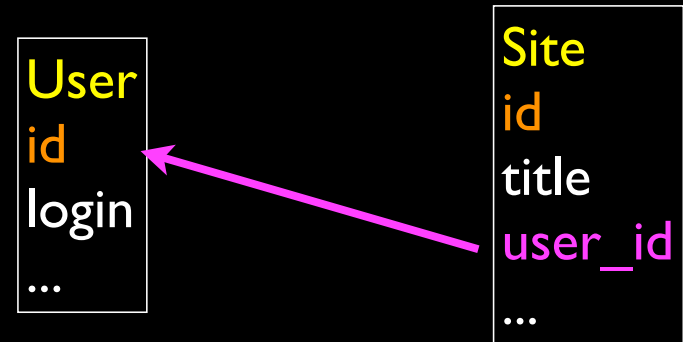
Primary Key Rules

- 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 far less efficient than integers performance-wise

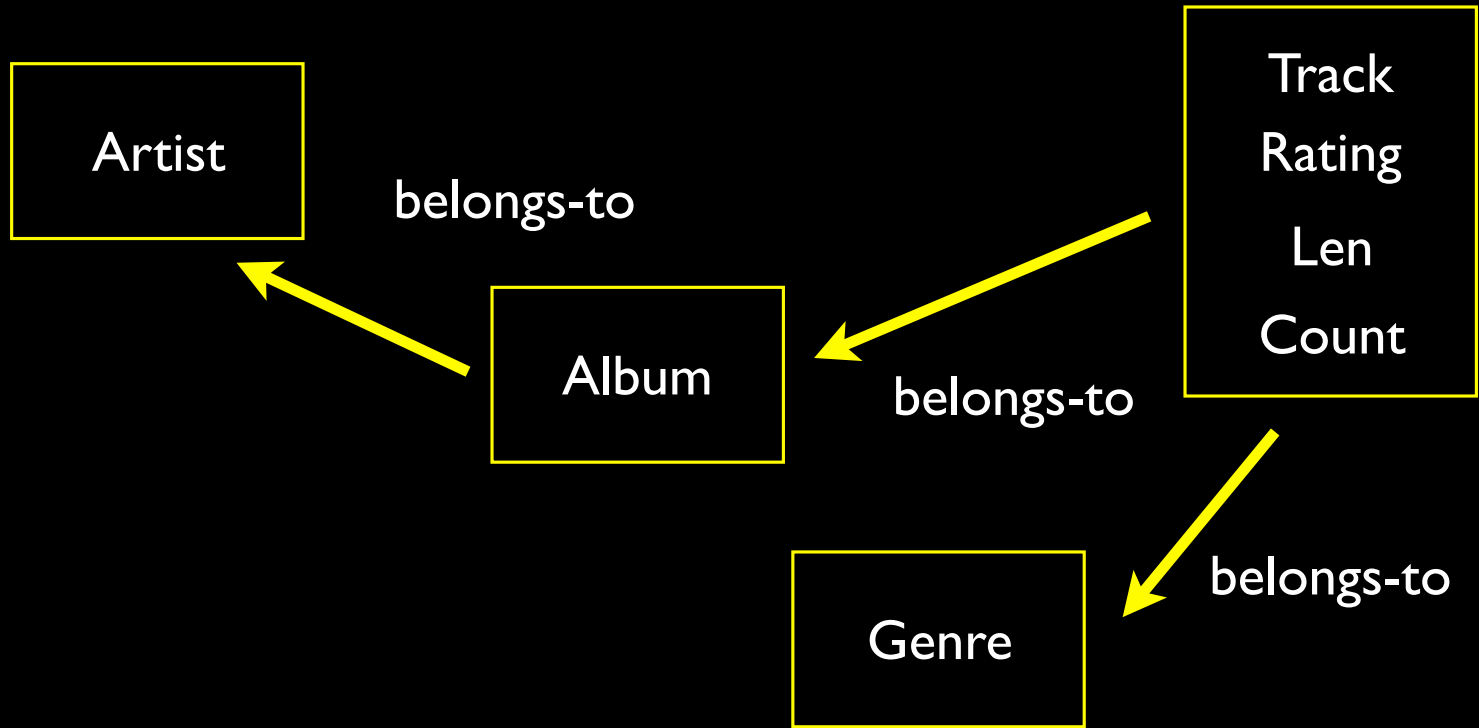
```
User
id
login
password
name
email
created_at
modified_at
login_at
```

Foreign Keys

- A **foreign key** is when a table has a column that contains a key which points the **primary key** of another table.
- When all primary keys are integers, then all foreign keys are integers - this is good - very good
- If you use strings as foreign keys - you show yourself to be an uncultured swine



Relationship Building (in tables)



<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"/>	Tie Man	3:20	America	Greatest Hits	Easy Listen...	★★★★★	22

Album

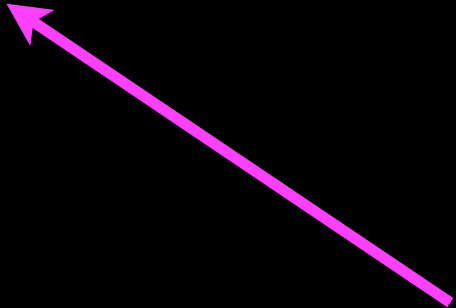
belongs-to

Track
Title
Rating
Len
Count

Track
id
title
rating
len
count
album_id

Album
id
title

Table
Primary key
Logical key
Foreign key



Artist
id
name

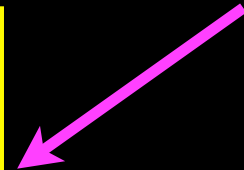
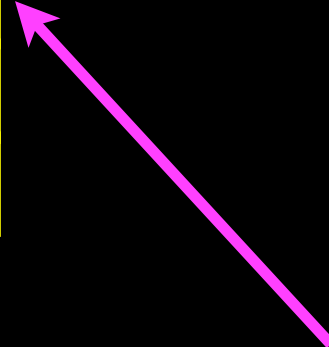
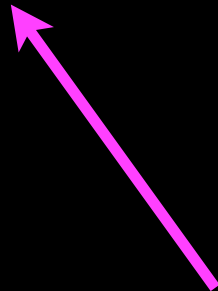
Album
id
title
artist_id

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.



```
CREATE TABLE Track (  
    id INT UNSIGNED NOT NULL AUTO_INCREMENT KEY,  
    title VARCHAR(128),  
    rating INTEGER,  
    len INTEGER,  
    count INTEGER,  
    album_id INTEGER,  
    genre_id INTEGER  
);  
  
CREATE TABLE Genre (  
    id INT UNSIGNED NOT NULL AUTO_INCREMENT KEY,  
    name VARCHAR(128)  
);  
  
CREATE TABLE Album (  
    id INT UNSIGNED NOT NULL AUTO_INCREMENT KEY,  
    title VARCHAR(128),  
    artist_id INTEGER  
);  
  
CREATE TABLE Artist (  
    id INT UNSIGNED NOT NULL AUTO_INCREMENT KEY,  
    name VARCHAR(128)  
);
```

```
mysql> describe Track;
```

Field	Type	Null	Key	Default	Extra
id	int(10) unsigned	NO	PRI	NULL	auto_increment
title	varchar(128)	YES		NULL	
rating	int(11)	YES		NULL	
len	int(11)	YES		NULL	
count	int(11)	YES		NULL	
album_id	int(11)	YES		NULL	
genre_id	int(11)	YES		NULL	

```
7 rows in set (0.01 sec)
```

```
mysql> describe Artist;
```

Field	Type	Null	Key	Default	Extra
id	int(10) unsigned	NO	PRI	NULL	auto_increment
name	varchar(128)	YES		NULL	

```
2 rows in set (0.00 sec)
```

```
mysql> insert into Artist (name) values ('Led Zepplin');  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> insert into Artist (name) values ('AC/DC');  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> select * from Artist;
```

```
+-----+-----+  
| id | name          |  
+-----+-----+  
|  1 | Led Zepplin  |  
|  2 | AC/DC        |  
+-----+-----+
```

```
2 rows in set (0.00 sec)
```

```
mysql> insert into Genre (name) values ('Rock');  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> insert into Genre (name) values ('Metal');  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> select * from Genre;
```

```
+-----+-----+  
| id | name |  
+-----+-----+  
| 1 | Rock |  
| 2 | Metal |  
+-----+-----+
```

```
2 rows in set (0.00 sec)
```

```
mysql> insert into Album (title, artist_id) values ('Who Made Who', 2);  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> insert into Album (title, artist_id) values ('IV', 1);  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> select * from Album;
```

```
+----+-----+-----+  
| id | title          | artist_id |  
+----+-----+-----+  
|  1 | Who Made Who  |         2 |  
|  2 | IV            |         1 |  
+----+-----+-----+
```

```
2 rows in set (0.00 sec)
```

```
mysql> insert into Album (title, artist_id) values ('Who Made Who', 2);  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> insert into Album (title, artist_id) values ('IV', 1);  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> select * from Album;
```

```
+----+-----+-----+  
| id | title          | artist_id |  
+----+-----+-----+  
|  1 | Who Made Who  |         2 |  
|  2 | IV            |         1 |  
+----+-----+-----+
```

```
2 rows in set (0.00 sec)
```

```
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);
```

```
mysql> select * from Track;
```

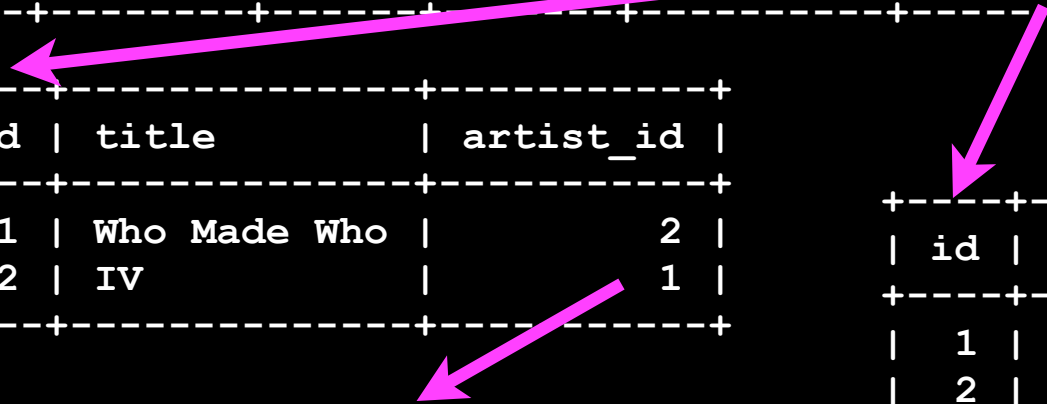
id	title	rating	len	count	album_id	genre_id
1	Black Dog	5	297	0	2	1
2	Stairway	5	482	0	2	1
3	About to Rock	5	313	0	1	2
4	Who Made Who	5	207	0	1	2

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

id	title	artist_id
1	Who Made Who	2
2	IV	1

id	name
1	Rock
2	Metal

id	name
1	Led Zeppelin
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**

id	title	artist_id
1	Who Made Who	2
2	IV	1

title	name
Who Made Who	AC/DC
IV	Led Zeppelin

id	name
1	Led Zeppelin
2	AC/DC

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

What we want
to see

The tables which
hold the data

How the tables
are linked

```
mysql> select * from Album join Artist on Album.artist_id = Artist.id;
```

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

```
2 rows in set (0.00 sec)
```

```
mysql>
```

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.id and Track.album_id = Album.id and
Album.artist_id = Artist.id;
```

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

<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/Fair						
<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 |
+-----+-----+-----+-----+
| Black Dog | Led Zeppelin | IV | Rock |
| Stairway | Led Zeppelin | IV | Rock |
| About to Rock | AC/DC | Who Made Who | Metal |
| Who Made Who | AC/DC | Who Made Who | Metal |
+-----+-----+-----+-----+

```

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

Theory of Normalization

Normalization

- Spreading data across multiple tables and linking the rows with foreign and primary keys
- E. F. Codd invented the relational model and described rules that data models must follow to satisfy first, second, and third normal "forms"
- Reducing duplicate data

Author 1	Author 2	Title	ISBN	Price U.S.	Cust.name	Cust.address	Purch.date
David Sklar	Adam Trachtenberg	<i>PHP Cookbook</i>	0596101015	44.99	Emma Brown	1565 Rainbow Road, Los Angeles, CA 90014	Mar 03 2009
Danny Goodman		<i>Dynamic HTML</i>	0596 527403	59.99	Darren Ryder	4758 Emily Drive, Rich- mond, VA 23219	Dec 19 2008
Hugh E. Williams	David Lane	<i>PHP and MySQL</i>	0596005436	44.95	Earl B. Thurston	862 Gregory Lane, Frank- fort, KY 40601	Jun 22 2009
David Sklar	Adam Trachtenberg	<i>PHP Cookbook</i>	0596101015	44.99	Darren Ryder	4758 Emily Drive, Rich- mond, VA 23219	Dec 19 2008
Rasmus Lerdorf	Kevin Tatroe & Peter MacIntyre	<i>Programming PHP</i>	0596006815	39.99	David Miller	3647 Cedar Lane, Wal- tham, MA 02154	Jan 16 2009

First Normal Form

- There should be no repeating columns containing the same kind of data (1A)
- All columns should contain a single value (1B)
- There should be a primary key to uniquely identify each row (1C)

Author 1	Author 2	Title	ISBN	Price U.S.	Cust.name	Cust.address	Purch.date
David Sklar	Adam Trachtenberg	<i>PHP Cookbook</i>	0596101015	44.99	Emma Brown	1565 Rainbow Road, Los Angeles, CA 90014	Mar 03 2009
1A		1C (ok)					
Danny Goodman		<i>Dynamic HTML</i>	0596 527403	59.99	Darren Ryder	4758 Emily Drive, Rich- mond, VA 23219	Dec 19 2008
Hugh E. Williams	David Lane	<i>PHP and MySQL</i>	0596005436	44.95	Earl B. Thurston	862 Gregory Lane, Frank- fort, KY 40601	Jun 22 2009
David Sklar	Adam Trachtenberg	<i>PHP Cookbook</i>	0596101015	44.99	Darren Ryder	4758 Emily Drive, Rich- mond, VA 23219	Dec 19 2008
1B							
Rasmus Lerdorf	Kevin Tatroe & Peter MacIntyre	<i>Programming PHP</i>	0596006815	39.99	David Miller	3647 Cedar Lane, Wal- tham, MA 02154	Jan 16 2009

Title	ISBN	Price	Cust. name	Cust. address	Purch. date
<i>PHP Cookbook</i>	0596101015	44.99	Emma Brown	1565 Rainbow Road, Los Angeles, CA 90014	Mar 03 2009
<i>Dynamic HTML</i>	0596527403	59.99	Darren Ryder	4758 Emily Drive, Richmond, VA 23219	Dec 19 2008
<i>PHP and MySQL</i>	0596005436	44.95	Earl B. Thurston	862 Gregory Lane, Frankfort, KY 40601	Jun 22 2009
<i>PHP Cookbook</i>	0596101015	44.99	Darren Ryder	4758 Emily Drive, Richmond, VA 23219	Dec 19 2008
<i>Programming PHP</i>	0596006815	39.99	David Miller	3647 Cedar Lane, Waltham, MA 02154	Jan 16 2009

Moving Authors into a separate table keyed by ISBN fixes (1A) and (1B). So data organized into these two tables is in First Normal form. Column repetition has been removed.

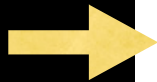
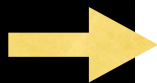
ISBN	Author
0596101015	David Sklar
0596101015	Adam Trachtenberg
0596527403	Danny Goodman
0596005436	Hugh E Williams
0596005436	David Lane
0596006815	Rasmus Lerdorf
0596006815	Kevin Tatroe
0596006815	Peter MacIntyre

Second Normal Form

- Focus on duplicate data in the same column between multiple rows

ISBN	Cust. name	Cust. address	Purch. date
0596101015	Emma Brown	1565 Rainbow Road, Los Angeles, CA 90014	Mar 03 2009
0596527403	Darren Ryder	4758 Emily Drive, Richmond, VA 23219	Dec 19 2008
0596005436	Earl B. Thurston	862 Gregory Lane, Frankfort, KY 40601	Jun 22 2009
0596101015	Darren Ryder	4758 Emily Drive, Richmond, VA 23219	Dec 19 2008
0596006815	David Miller	3647 Cedar Lane, Waltham, MA 02154	Jan 16 2009

ISBN	Cust. name	Cust. address	1B	Purch. date
0596101015	Emma Brown	1565 Rainbow Road, Los Angeles, CA 90014		Mar 03 2009
0596527403	2 Darren Ryder	4758 Emily Drive, Richmond, VA 23219		Dec 19 2008
0596005436	Earl B. Thurston	862 Gregory Lane, Frankfort, KY 40601		Jun 22 2009
0596101015	Darren Ryder	4758 Emily Drive, Richmond, VA 23219		Dec 19 2008
0596006815	David Miller	3647 Cedar Lane, Waltham, MA 02154		Jan 16 2009



CustNo	ISBN	Date
1	0596101015	Mar 03 2009
2	0596527403	Dec 19 2008
2	0596101015	Dec 19 2008
3	0596005436	Jun 22 2009
4	0596006815	Jan 16 2009

CustNo	Name	Address	City	State	Zip
1	Emma Brown	1565 Rainbow Road	Los Angeles	CA	90014
2	Darren Ryder	4758 Emily Drive	Richmond	VA	23219
3	Earl B. Thurston	862 Gregory Lane	Frankfort	KY	40601
4	David Miller	3647 Cedar Lane	Waltham	MA	02154

Third Normal Form (3NF)

- Data that is not dependent on the primary key but instead, dependent on a non-primary key must also be in a separate column

CustNo	Name	Address	City	State	Zip
1	Emma Brown	1565 Rainbow Road	Los Angeles	CA	90014
2	Darren Ryder	4758 Emily Drive	Richmond	VA	23219
3	Earl B. Thurston	862 Gregory Lane	Frankfort	KY	40601
4	David Miller	3647 Cedar Lane	Waltham	MA	02154

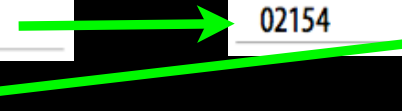
CustNo	Name	Address	City	State	Zip
1	Emma Brown	1565 Rainbow Road	Los Angeles	CA	90014
2	Darren Ryder	4758 Emily Drive	Richmond	VA	23219
3	Earl B. Thurston	862 Gregory Lane	Frankfort	KY	40601
4	David Miller	3647 Cedar Lane	Waltham	MA	02154

CustNo	Name	Address	Zip
1	Emma Brown	1565 Rainbow Road	90014
2	Darren Ryder	4758 Emily Drive	23219
3	Earl B. Thurston	862 Gregory Lane	40601
4	David Miller	3647 Cedar Lane	02154

Zip	CityID
90014	1234
23219	5678
40601	4321
02154	8765

CityID	Name	StateID
1234	Los Angeles	5
5678	Richmond	46
4321	Frankfort	17
8765	Waltham	21

StateID	Name	Abbreviation
5	California	CA
46	Virginia	VA
17	Kentucky	KY
21	Massachusetts	MA



Is 3NF Necessary?

- By ruthlessly removing redundancy, you reduce the size of data and that generally makes your application **faster**
- By having one place where you map a city name to a ZIP code, you have **one place to change** in your entire data model if a city gets a new ZIP code
- Sometimes things are not so perfect though - the city / Zip code may not be so

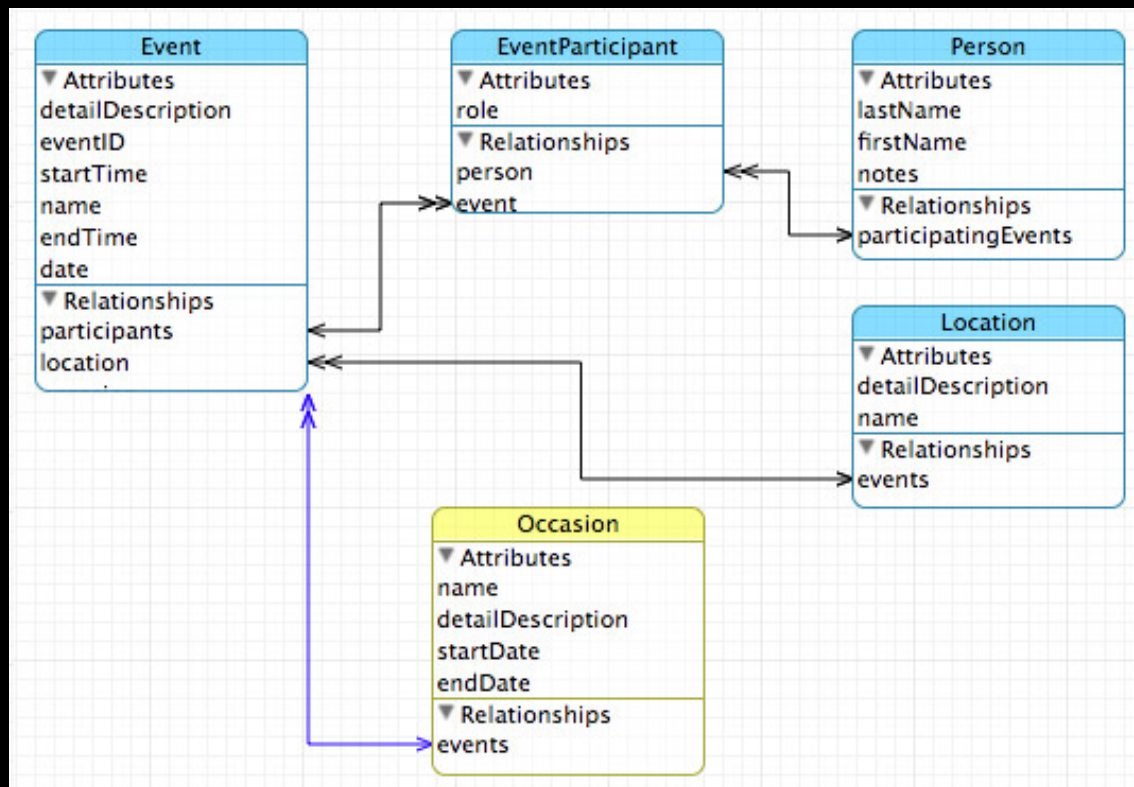
De-normalization on Purpose

- Properly designed relational databases following 3NF have amazingly good performance
- CTools @ UMich supports 20,000 simultaneous logged in users with a total user population of over 200K using 3NF data
- Important data is spread across tables linked with primary and foreign keys and information is assembled using JOINS
- Because of this, if you change your last name, in CTools we change it one place and all of CTools immediately knows of the change on the next page view.

Mega-Sized Applications

- Applications like Facebook, Twitter and Google with 1/2 billion users and millions of messages per second cause a pure 3NF / Relational Model to break down for performance even when every possible trick is used.
- To meet the performance demands of these systems, we allow some replicated data at the cost of the need to update information like "Chuck's last tweet" more than one place
- Data is consistent across the whole system a "little while later"...

Relationship Types



Three Kinds of Relationships

- One to many
- One to one
- Many to many

Album

belongs-to

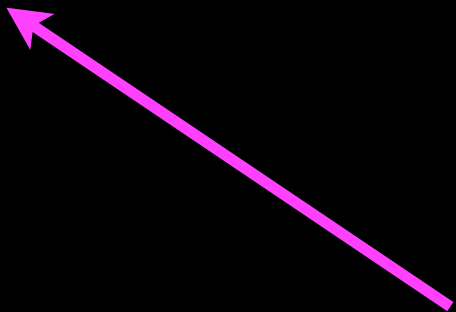


Track
Title
Rating
Len
Count

Track
id
title
rating
len
count
album_id

Album
id
title

One

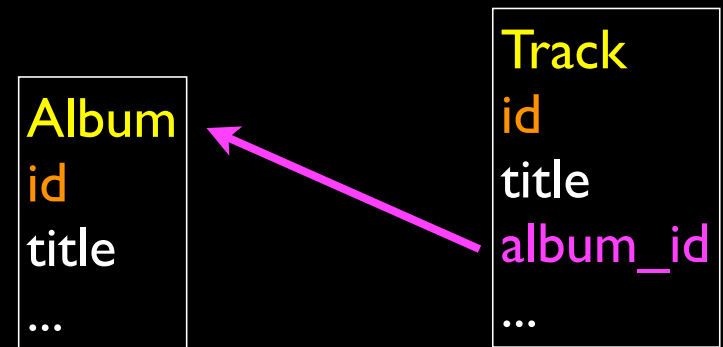


Many

Table
Primary key
Logical key
Foreign key

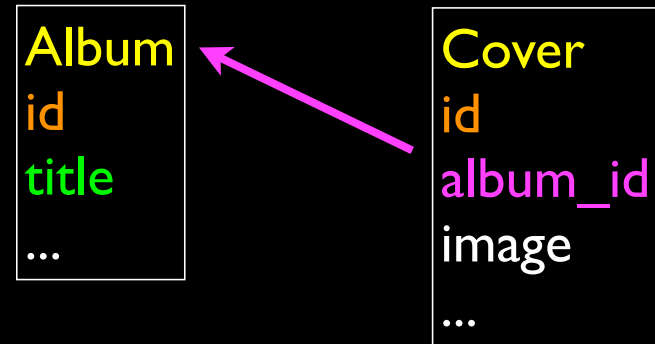
One to Many

- Each Album object may have many Tracks
- A Track always has one single “Album”
- An Album may also have zero Tracks (sometimes)



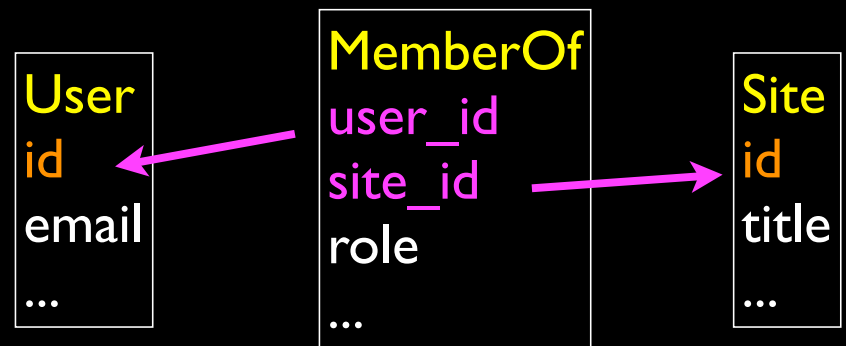
One to One

- Each Album object may have a Cover object
- But a Cover never has more than one Album
- “One to Zero or One”

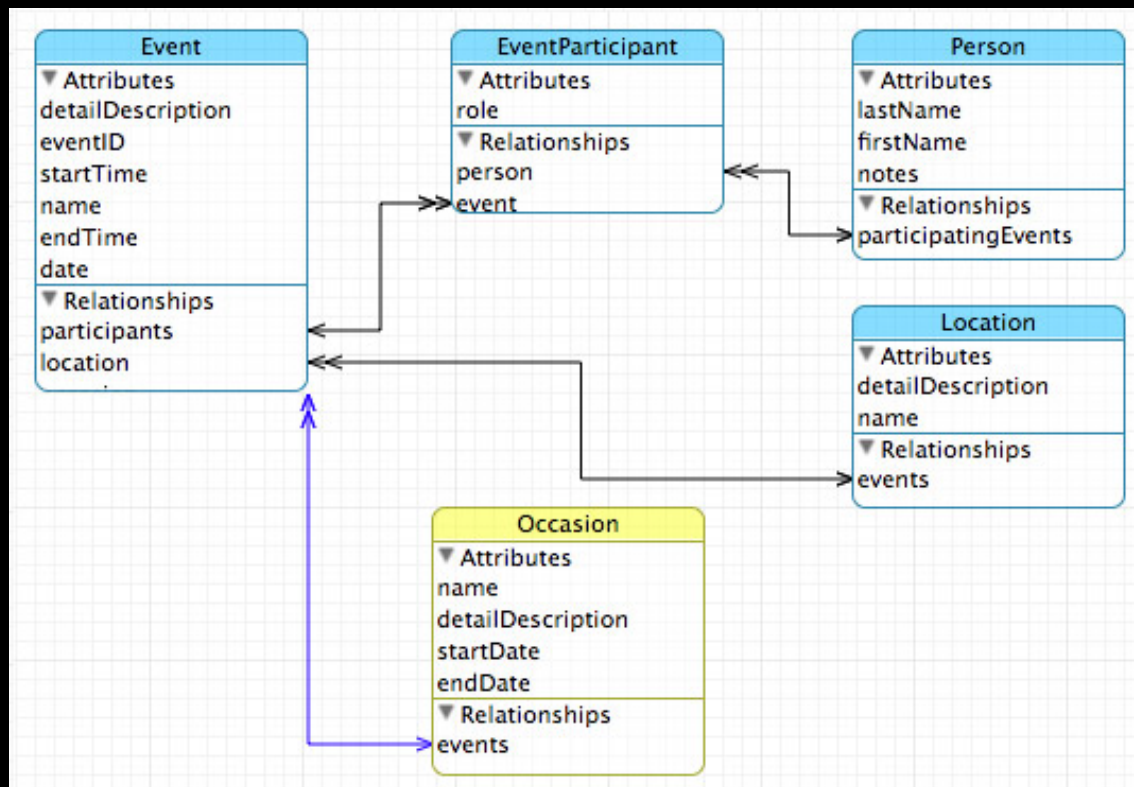


Many to Many

- A User can be a member of many Sites
- Site can have many Users
- We cannot put this in a column of either table
- We need a separate “relates” table - or join table



These relates tables often do not have a separate primary key - the two fields form a composite primary key.



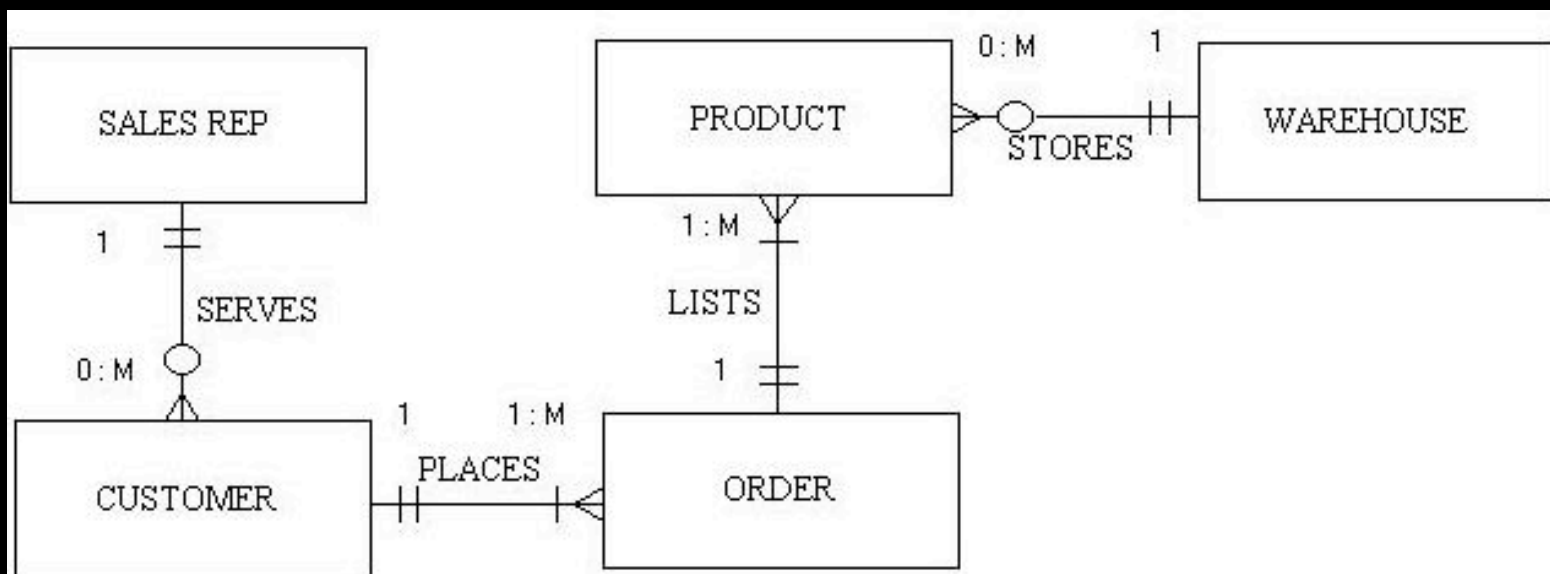


Figure 1. Entity-Relationship Diagram

- * 1 INSTANCE OF A SALES REP SERVES 1 TO MANY CUSTOMERS
- * 1 INSTANCE OF A CUSTOMER PLACES 1 TO MANY ORDERS
- * 1 INSTANCE OF AN ORDER LISTS 1 TO MANY PRODUCTS
- * 1 INSTANCE OF A WAREHOUSE STORES 0 TO MANY PRODUCTS

Transactions

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

Bank

Balance: 200

ATM

ATM

Bank

Balance: 200

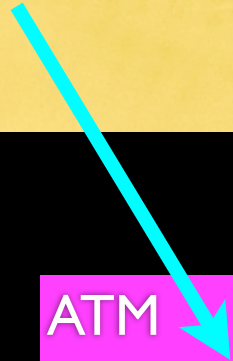
ATM

Balance: 200

ATM

Bank

Balance: 200



ATM

Balance: 200

ATM

Balance: 200

Bank

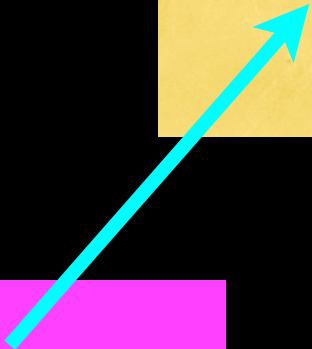
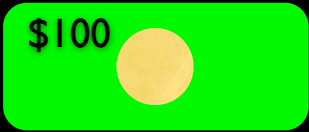
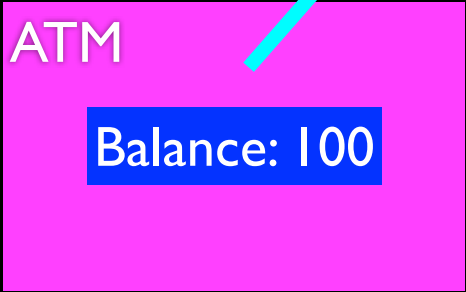
Balance: 200

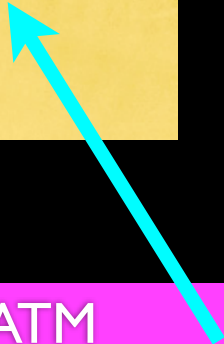
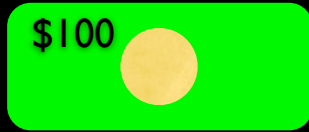
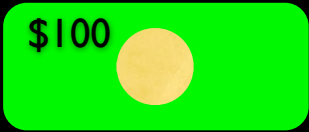
ATM

Balance: 200

ATM

Balance: 200





Bank

Balance: 100

ATM

Balance: 100

ATM

Balance: 100

\$100



\$100



Awesome!

Transactions

- Allows a sequence of steps to work as a unit or not work at all
- Handles the fact that some things take time
- Locks data on read so that it cannot be read by another transaction until the current transaction completes

Transactions in MySQL

- Tables where transactions will be used require the use of the InnoDB engine.
- **BEGIN** - Starts a transaction
- **COMMIT** - Completes a transaction
- **ROLLBACK** - Gives up on a transaction

```
CREATE TABLE accounts (  
    number INT,  
    balance FLOAT,  
    PRIMARY KEY(number)  
) ENGINE InnoDB;
```

```
INSERT INTO accounts(number, balance) VALUES(12345, 1025.50);  
INSERT INTO accounts(number, balance) VALUES(67890, 140.00);
```

```
mysql> select * from accounts;
```

```
+-----+-----+  
| number | balance |  
+-----+-----+  
| 12345  | 1025.5  |  
| 67890  | 140     |  
+-----+-----+
```

```
2 rows in set (0.02 sec)
```

```
mysql>
```

```
BEGIN;  
UPDATE accounts SET balance=balance+25.11 WHERE number=12345;  
COMMIT;
```

```
mysql> select * from accounts;
```

number	balance
12345	1050.61
67890	140

```
2 rows in set (0.00 sec)
```

```
mysql>
```

```
mysql> BEGIN;  
mysql> UPDATE accounts SET balance=balance-250 WHERE number=12345;  
mysql> UPDATE accounts SET balance=balance+250 WHERE number=67890;  
mysql> SELECT * from accounts;
```

number	balance
12345	800.61
67890	390

```
mysql> ROLLBACK;  
Query OK, 0 rows affected (0.34 sec)
```

```
mysql> SELECT * from accounts;
```

number	balance
12345	1050.61
67890	140

```
mysql> select * from accounts;
+-----+-----+
| number | balance |
+-----+-----+
| 12345  |    200  |
| 67890  |    140  |
+-----+-----+
2 rows in set (0.00 sec)

mysql> BEGIN;
Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE accounts SET balance=balance-100 WHERE number=12345;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Transaction started
but not committed.

```
mysql> █
```

```
mysql> select * from accounts;
+-----+-----+
| number | balance |
+-----+-----+
| 12345  |    200  |
| 67890  |    140  |
+-----+-----+
2 rows in set (0.00 sec)

mysql> UPDATE accounts SET balance=balance+50 WHERE number=12345;
█
```

Update waits

```
mysql> select * from accounts;
+-----+-----+
| number | balance |
+-----+-----+
| 12345  |    200  |
| 67890  |    140  |
+-----+-----+
2 rows in set (0.00 sec)

mysql> BEGIN;
Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE accounts SET balance=balance-100 WHERE number=12345;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Transaction started
but not committed.

```
mysql> █
```

```
mysql> select * from accounts;
+-----+-----+
| number | balance |
+-----+-----+
| 12345  |    200  |
| 67890  |    140  |
+-----+-----+
2 rows in set (0.00 sec)

mysql> UPDATE accounts SET balance=balance+50 WHERE number=12345;
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction
mysql> █
```

Update fails

```
mysql> select * from accounts;
+-----+-----+
| number | balance |
+-----+-----+
| 12345  |    200  |
| 67890  |    140  |
+-----+-----+
2 rows in set (0.00 sec)

mysql> BEGIN;
Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE accounts SET balance=balance-100 WHERE number=12345;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> 
```

Transaction started
but not committed.

```
mysql> 
```

```
mysql> select * from accounts;
+-----+-----+
| number | balance |
+-----+-----+
| 12345  |    200  |
| 67890  |    140  |
+-----+-----+
2 rows in set (0.00 sec)

mysql> UPDATE accounts SET balance=balance+50 WHERE number=12345;

```

Try again, waiting

```
mysql> BEGIN;
Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE accounts SET balance=balance-100 WHERE number=12345;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

Transaction is committed.

```
mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

```
mysql> UPDATE accounts SET balance=balance+50 WHERE number=12345;
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction

mysql> UPDATE accounts SET balance=balance+50 WHERE number=12345;
Query OK, 1 row affected (5.80 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

```
mysql> select * from accounts;
+-----+-----+
| number | balance |
+-----+-----+
| 12345  | 150    |
| 67890  | 140    |
+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

Update immediately finishes. And balance correctly reflects both updates.

Transactions

- Transactions are essential in situations where critical data is read, updated, and written in a way that timing is critical
- But transactions incur a cost because they lock areas of the database and pause other operations waiting for the transaction to finish
- There is both a resource and performance cost if transactions are used excessively

Transaction Deadlock

- If process A Locks X, and process B locks Y, and then process A waits for Y and process B waits for X - they wait forever
- This is called a "deadlock"
- Deadlocks are avoided by making sure all transactions take locks in the same order - this way one will "win" at the beginning not the others will wait at the beginning of the sequence.

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 which much be scanned** when doing complex operations across large amounts of data
- There are rules and sometimes we break the rules...
- Database and SQL design is a bit of an **art-form**