ojoen.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 © 2009, Charles Severance.

You assume all responsibility for use and potential liability associated with any use of the material. Material contains copyrighted content, used in accordance with U.S. law. Copyright holders of content included in this material should contact open.michigan@umich.edu with any questions, corrections, or clarifications regarding the use of content. The Regents of the University of Michigan do not license the use of third party content posted to this site unless such a license is specifically granted in connection with particular content. Users of content are responsible for their compliance with applicable law. Mention of specific products in this material solely represents the opinion of the speaker and does not represent an endorsement by the University of Michigan. For more information about how to cite these materials visit http://michigan.educommons.net/about/terms-of-use.

Any medical information in this material is intended to inform and educate and is not a tool for self-diagnosis or a replacement for medical evaluation, advice, diagnosis or treatment by a healthcare professional. You should speak to your physician or make an appointment to be seen if you have questions or concerns about this information or your medical condition. Viewer discretion is advised: Material may contain medical images that may be disturbing to some viewers.





Relational Databases

Charles Severance



Relational Databases

Relational databases model data by storing rows and columns in tables. The power of the relational database lies in its ability to efficiently retrieve data from those tables and in particular where there are multiple tables and the relatinships between those tables involved in the query.

http://en.wikipedia.org/wiki/Relational database

SQLite Database Browser

- SQLite is a very popular browser it is free and fast and small
- We have a program to manipulate SQLite databases
- http://sqlitebrowser.sourceforge.net/
- SQLite is embedded in Python and a number of other languages



SQLite is in lots of software...

SymbianPythonPhilipsGEMicrosoftMcAfeeAdobeFirefoxPHPToshiba

Google

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

Skype

Apple

Sun Microsystems

000	SQLite Database Brow	wser – /Users/csev/sql1	
🗋 🚰 🖬		₽ } ?	
	Database Structure Bro	wse Data Execute SQL	
Table:	:	New Record Delete	R
< 0 -	0 of 0 >	Go to: 0	

http://sqlitebrowser.sourceforge.net/

Source: SQLite Terminal



Start Simple - A Single Table

Lets make a table of People - with a Name and an E-Mail

000	SQLite Database Browser - /Users/csev/sql1
c	Database Structure Browse Data Execute SQL
Name Object ▼Users table email field name field	Type Schema CREATE TABLE Users (email TEXT, name TEXT TEXT
	Our first table with two colu



Source: SQLite Terminal

0	0	SQLite Database Browse	er – /Users/csev/sql1
	🎽 🖬 🕫 🖬 🛙		₽
	Da	tabase Structure Brows	e Data Execute SQL
Tab	le: Users 🛟		New Record
	email	name	
1	csev@umich.edu	Chuck	
2	botimer@umich.edu	Noah	
3	mrzhou@umich.edu	Daniel	
4	ksnam@umich.edu	Kevin	
		Our tabl	e with four rows
C	<) 1 - 4 of 4 >>		Go to: 0
_			



Source: SQLite Terminal

SQL

- Structured Query Language is the language we use to issue commands to the database
 - Create a table
 - Retieve some data
 - Insert data
 - Delete data

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

SQL Insert

The Insert statement inserts a row into a table

insert into Users (name, email) values ('Ted', 'ted@umich.edu')



SQLite Database Browser - /Users/c	csev/sql1 🖂
Database Structure Browse Data Ex	kecute SQL
SQL string:	
insert into Users (name, email) values ('Ted', 'ted@umich.edu')	SQLite Database Browser - /Users/csev/sql1
	Database Structure Browse Data Execute SQL
	Table: Users Image: Comparison New Record Delete Record
Execute query	email name
Error message from database engine:	1 csev@umich.edu Chuck
No error	2 botimer@umich.edu Noah
Data returned:	3 mrznou@umich.edu Daniel
	5 ted@umich.edu Ted
	<pre></pre>

Sources: SQLite Terminal

SQL Delete

Deletes a row in a table based on a selection criteria

delete from Users where email='ted@umich.edu'

000	SQLite Database Browser - /Users/cse	ev/sql1				
D 🚰 🖬 🔊						
	Database Structure Browse Data Exec	ute SQL	1			
SQL string:						
delete from Users v	vhere email='ted@umich.edu'			SQLite Database Browse	er – /Users/csev/sql1	
Execute query		Tak	le: Users ‡	tabase Structure Brows	e Data Execute SQL	Delete Record
Error message from	database engine:	1	email csev@umich.edu	name Chuck		
No error		2	botimer@umich.edu	Noah		
Data returned:		4	mrzhou@umich.edu ksnam@umich.edu	Daniel Kevin		
		C	<) 1 - 4 of 4 >		Go to: 0	

Sources: SQLite Terminal

SQL: Update

Allows the updating of a field with a where clause

update Users set name="Charles" where email='csev@umich.edu'

SQLite Database Browser - /Users/csev/	/sql1
Database Structure Browse Data Execut	te SQL
SQL string:	
update Users set name="Charles" where email='csev@umich.edu'	SQLite Database Browser - /Users/csev/sql1
	Database Structure Browse Data Execute SQL
	Table: Users Image: Table New Record Delete Record
Execute query	email name
Error message from database engine:	1 csev@umich.edu Charles
No error	3 mrzhou@umich.edu Daniel
Data returned:	4 ksnam@umich.edu Kevin
	(<) 1 - 4 of 4 (>) Go to: 0

Sources: SQLite Terminal

Retrieving Records: Select

• The select statement retrieves a group of records - you can either retrieve all the records or a subset of the records with a WHERE clause

select * from Users

select * from Users where email='csev@umich.edu'

000	SQLite Database Browser - /U	sers/csev/sql1 🛛 🕀	
🗅 🚄 🖬 K	▶ 💣 💣 🖬 💣 💣 🖳 🐶?		
	Database Structure Browse Data	Execute SQL	
SQL string:			
select * from Us	sers		te Database Browser -
			1 📑 📑 📟 🗟
		Database	Structure Browse Da
		SQL string:	
Execute quer	y)	select * from Users where email=	csev@umich.edu'
Error message fr	om database engine:		
No error			
Data returned:			
csev@umich.edu	Charles	(Execute query)	
botimer@umich.e	du Noan du Daniel	Error message from database eng	ine:
ksnam@umich.ed	u Kevin	Ne sees	
		No error	
		Data returned:	
		csev@umich.edu	Charles



Sources: SQLite Terminal

Sorting with ORDER BY

• You can add an ORDER BY clause to SELECT statements to get the results sorted in ascending or descending order

> select * from Users order by email select * from Users order by name





000	SQLite Database Browser - /Users/csev/sql1	
🗅 🚅 🖬 🍤 🛒		
	Database Structure Browse Data Execute SQL	
SQL string: select * from Users order by name		SQLite Database Browser - /Users/csev/sql1 SQLite Database Browser - /Users/csev/sql1 Database Structure Browse Data Execute SQL SQL string: select * from Users order by email
Error message from data No error	abase engine:	
Data returned:		Execute query
csev@umich.edu mrzhou@umich.edu	Charles Daniel	Error message from database engine:
ksnam@umich.edu botimer@umich.edu	Kevin Noah	Data returned:
		botimer@umich.edu Noah csev@umich.edu Charles ksnam@umich.edu Kevin mrzhou@umich.edu Daniel

Sources: SQLite Terminal

SQL Summary

insert into Users (name, email) values ('Ted', 'ted@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

This is not too exciting (so far)

Tables pretty much look like big fast programmable spreadsheet with rows, columns, and commands

• The power comes when we have more than one table and we can exploit the relationships between the tables

Complex Data Models and Relationships

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...

Event	EventParticipant	Person
Attributes	Attributes	Attributes
detailDescription	role	lastName
eventID	Relationships	firstName
startTime	person 🛹	- notes
name	>> event	Relationships
endTime		participatingEvents
date		
Relationships		Location
participants <		Location
location	<u> </u>	detailDescription
*		name
		Relationships
		events
	Occasion	
	Attributes	
	name	
	detailDescription	
	startDate	
	endDate	
	Relationships	
	events	



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

WAREHOUSE



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
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

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.

Track

Source: Apple iTunes Terminal

✓ 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
Ef Tip Man	2.20	Amorica	Crostoct Hite	Ency Liston		22

Album Len Genre

Artist

Rating

Count

Track Artist Album Genre Rating Len		belongs-to	bel	ongs-to	belongs-t	Ō
Count						
		Source: Apple iTunes Terminal				
Mells Bells	5:13 AC/	/DC Who Mad	de Who	Rock	****	61
Shake Your Foundations	3:54 AC/	/DC Who Mad	de Who	Rock	****	70
Chase the Ace	3:01 AC/	/DC Who Mad	de Who	Rock		56
For Those About To Rock (We	5:54 AC/	/DC Who Mad	de Who	Rock	****	61
🗹 Dúlamán	3:43 Alta	an Natural V	Wonders M	New Age		31
Rode Across the Desert	4:10 Am	erica Greatest	Hits	Easy Listen	****	23
Mow You Are Gone	3:08 Am	erica Greatest	Hits	Easy Listen	****	18
Ef Tin Man	2.20 0.00	orico Creatost	Llitz	Ency Liston	A A A A A	22





Hells Bells	5:13	AC/DC	Who Made Who	Roc
Shake Your Foundations	3:54	AC/DC	Who Made Who	Roc
Chase the Ace	3:01	AC/DC	Who Made Who	Roc
For Those About To Rock (We	5:54	AC/DC	Who Made Who	Roc
🗹 Dúlamán	3:43	Altan	Natural Wonders M	Nev
Rode Across the Desert	4:10	America	Greatest Hits	East
Now You Are Gone	3:08	America	Greatest Hits	East
Tin Man	2.20	Amorica	Crostoct Hite	Enci

	Track Rating Len Count	
igs-to		
	belongs-te	0
ck	****	61
ck	****	70
ck		56
ck	****	61
w Age		31
y Listen	****	23
sy Listen	****	18
nu Lieton		22

Representing Relationships in a Database



We want to keep track of who is the "owner" of each chat message... Who does this chat message "belong to"????

Database Normalization (3NF)

- There is *tons* of database theory way too much to understand without excessive predicate calculus
 - Do not replicate data reference data point at data
 - Use integers for keys and for references
 - Add a special "key" to each table which we will reference by convention many programmers call this "id"

http://en.wikipedia.org/wiki/Database normalization

Better Reference Pattern





csev@umich.edu

gsilver@umich.edu



Finding our way around....

Three Kinds of Keys

- Primary key generally an integer autoinrcement field
- Logical key What the outside world uses for lookup
- Foreign key generally an integer key point to a row in another table



Site id title user • • •

Primary Key Rules

- Rails enourages you to follow 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)



Hells Bells	5:13	AC/DC	Who Made Who	Roc
Shake Your Foundations	3:54	AC/DC	Who Made Who	Roc
Chase the Ace	3:01	AC/DC	Who Made Who	Roc
For Those About To Rock (We	5:54	AC/DC	Who Made Who	Roc
🗹 Dúlamán	3:43	Altan	Natural Wonders M	Nev
Rode Across the Desert	4:10	America	Greatest Hits	East
Now You Are Gone	3:08	America	Greatest Hits	East
Tin Man	2.20	Amorica	Crostoct Hite	Enci

	Track Rating Len Count	
igs-to		
	belongs-te	0
ck	****	61
ck	****	70
ck		56
ck	****	61
w Age		31
y Listen	****	23
sy Listen	****	18
nu Lieton		22









TablePrimary keyLogical keyForeign key

Naming FK artist_id is a convention.





000	SQLite Database Browser – /Users/csev/sql2	
	f ∎ 💣 💣 📟 🐶?	
Data	abase Structure Browse Data Execute SQL	
Name Object Type	Create Table	
	Table name: Artist	
	Define fields:	
	Field name Field type	
	Add database field	
	Field name: id	
	Field type:	
	TEXT	
	NUMERIC	
	BLOB Cancel	Create
	INTEGER PRIMART KET	
	Create Cancel	O O O SQLite Database Browser - /Users/csev/sql2 O
		Database Structure Browse Data Execute SOL
20		
		Name Object Type Schema Artist table CREATE TABLE Artist (id INTECER PRIMARY KEV, name TE
		id field INTEGER PRIMARY KEY
		name field TEXT
Sources: SQLite Terminal		

00		SQLite Database Browser - /Users/csev/sql2
) 🚅 日 !	₽	
		Database Structure Browse Data Execute SQL
Name	Object	Type Schema
▼Album	table	CREATE TABLE Album (artist_id NUMER
artist_id	field	NUMERIC
id	field	INTEGER PRIMARY KEY
title	field	TEXT
▼Artist	table	CREATE TABLE Artist (id INTEGER PRIM.
id	field	INTEGER PRIMARY KEY
name	field	TEXT
▼Genre	table	CREATE TABLE Genre (id INTEGER PRIN
id	field	INTEGER PRIMARY KEY
name	field	TEXT
▼Track	table	CREATE TABLE Track (album_id NUMER
album_id	field	NUMERIC
count	field	NUMERIC
genre_id	field	NUMERIC
id	field	INTEGER PRIMARY KEY
len	field	NUMERIC
rating	field	NUMERIC
title	field	TEXT



SQLite Database Browser - /Users/csev/sql2	
	SQLite Database Browser - /Users/csev/sql2
Database Structure Browse Data Execute SQL	
SQL string:	Database Structure Browse Data Execute SQL
insert into Artist (name) values ('Led Zepplin')	Table: Artist Image: Contract of the second of the se
Execute query Error message from database engine:	
No error	
Data returned:	
	< 1 - 2 of 2 > Go to: 0

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

Sources: SQLite Terminal



000	SQLite Database Brows	er – /Users/csev/sql2
🗋 🖻 🚅 🖬 📭	• 📑 🗃 🖬 🗐	₿?
e	Database Structure Brow	se Data Execute SQL
Table: Genre		New Record Delete Record
id	name	
1	1 Rock	
2	2 Metal	
< 1 - 2 of	2 >	Go to: 0

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





Database Str	title Who Made Who	e Data	Execute SQL New Record) Delete
Database Str	title Who Made Who	e Data	Execute SQL New Record) Delete
1 2	title Who Made Who IV	(New Record	Delete
1	title Who Made Who IV			
2	Who Made Who			
2	IV			
>			Go to:	0
	.)	0	0	Go to:

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

Source: SQLite Terminal



insert into Track (title, rating, len, count, album id, genre id) values ('Black Dog', 5, 297, 0, 1, 1) insert into Track (title, rating, len, count, album id, genre id)

values ('Stairway', 5, 482, 0, 1, 1) insert into Track (title, rating, len, count, album id, genre id) values ('About to Rock', 5, 313, 0, 2, 2) insert into Track (title, rating, len, count, album id, genre id) values ('Who Made Who', 5, 207, 0, 2, 2)

	[Database Stru	cture Brows	e Data Execu	ite SQL	_
Table: Tr	rack 🌻			New	Record	De
count	genr	e_id id	ł	len	rating	
1	0	1	1	297	5	B
2	0	1	2	482	5	S
3	0	2	3	313	5	A
	0	2	4	207	5	V

	l.
elete Record	
title	
Black Dog	
Stairway	
About to Rock	
Who Made Who	

We have relationships!

					Table: Trac	ck 🛟 (Ne	w Record) Delete Rec
					album_i	d count	genre	_id id	ler	1	rating	title
					1	1	0	1	1	297		5 Black Dog
					2	1	0	1	2	482		5 Stairway
					3	2	0	2	3	313		5 About to F
					4	2	0	2	4	207		5 Who Made
artist_	Id	10		LILIC								
artist_	1d 2 1	Ια	1	Who Mac	de Who			Table:	Genre	A V		
artist_	1d 2 1		1	Who Mac IV	le Who			Table:	Genre	rian	ne	
artist_	1d 2 1		1	Who Mad IV	le Who			Table:	Genre	nan 1 Roc] (((((((((((((((((((
artist_	1d 2 1	Table	1 2 Ar	Who Mac IV	le Who			Table: id 1	Genre	nan 1 Roc 2 Met	ne :k tal	
artist_	1d 2 1	Table	1 2 Ar	Who Mad IV	le Who			Table: id 1	Genre	1 Roc 2 Met	ne :k tal	
artist_	1d 2	Table	1 2 id	Who Mad IV tist	ie Who iname Led Zep	olin		Table: id 1	Genre	1 Roc 2 Met	ne :k tal	

Sources: SQLite Terminal

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 the keys which make the connection between the tables using an ON clause

Table:



Track



	album_id	count	genre_id	id	len	ratin
1	1	0	1	1	297	
2	1	0	1	2	482	
3	2	0	2	3	313	
4	2	0	2	4	207	
-				I		

Data returned:	
Black Date	Death
Black Dog	Rock
Stairway	Rock
About to Rock	Metal
Who Made Who	Metal



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

What we want to see

The tables which hold the data

Sources: SQLite Terminal

3	title
5	Black Dog
5	Stairway
5	About to Rock
5	Who Made Who
Genre	÷ (0,
Genre	ame
Genre	name 1 Rock

How the tables are linked

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

Data returned:

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

What we want to see

The tables which hold the data

How the tables are linked

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
Mow 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	Data raturna						
Paranoid	Data returned:						22
Planet Caravan							25
🗹 Iron Man	Black Dog		AC/DC	Who Made	Who Made Who		26
Electric Funeral	Black Dog		AC/DC	WIIO Made	. WIIO	ROCK	22
✓ Hand of Doom	Stairway	AC/DC		who Made who		Rock	
Rat Salad	About to Roc	k Led Zepplin	IV	N	Metal Metal	81	
☑ Jack the Stripper/Fairies	Who Made W	ho Led Zepplin		IV			24
Bomb Squad (TECH)		no cea ceppini					
🗹 clay techno	<u>805</u>						2
☑ Heavy							1
☑ Hi metal man	4:20	Brent		Brent's Album			1
Mistro	2:58	Brent		Brent's Album			1
erminal							

Sources: SQLite Terminal

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

Python and SQLite3

http://www.python.org/doc/2.5.2/lib/module-sqlite3.html



SQLite3 is built into Python

- Since SQLite is simple and small and designed to be "embedded" -Python decided to embed SQLite into Python
- You simply "import sqlite3" and open a connection to the database and start doing SQL commands

http://www.python.org/doc/2.5.2/lib/module-sqlite3.html

SQLite3 is built into Python

import sqlite3

Open up the database file and get a cursor conn = sqlite3.connect('music.db') c = conn.cursor()

print "Genre Rows" c.execute('select * from Genre') for row in c : print row



SQLite stores all tables and data in a single file.

\$ python sql1.py Genre Rows (1, u'Rock')(2, u'Metal') music.db sql1.py sql2.py

import sqlite3

Open up the database file and get a cursor conn = sqlite3.connect('music.db') c = conn.cursor()

```
print "Inserting Country"
c.execute('insert into Genre (name) values (?)', ('Country', ))
```

```
print "Genre Rows"
c.execute('select * from Genre')
for row in c :
    print row
```

```
print "Deleting Country"
c.execute("delete from Genre where name='Country"")
```

```
print "Genre Rows"
c.execute('select * from Genre')
for row in c :
    print row
```

\$ python sql2.py Inserting Country Genre Rows (1, u'Rock')(2, u'Metal')(3, u'Country') Deleting Country Genre Rows (1, u'Rock')(2, u'Metal')

Additional SQL Topics

- Indexes improve access performance for things like string fields
- Constraints on data (cannot be NULL, etc..) O
- Transactions allow SQL operations to be grouped and done as a unit
- See SI572 Database Design

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