All Your Database Are Belong To Us

A bottom up, interactive learning spree

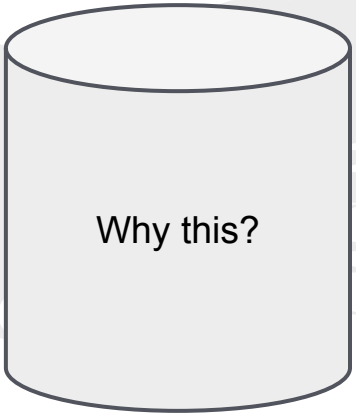
by
Jess Stanton
@tiny_mouse
February 27, 2014
wifi: uberguest

Outline

- Who? Roles and Permissions
- Wat? Schema
- How do? SQL
- Python Example App
- Failure Minimization
- Advanced Topics
- RDBMS Alternatives
- Questions!

Disclaimer

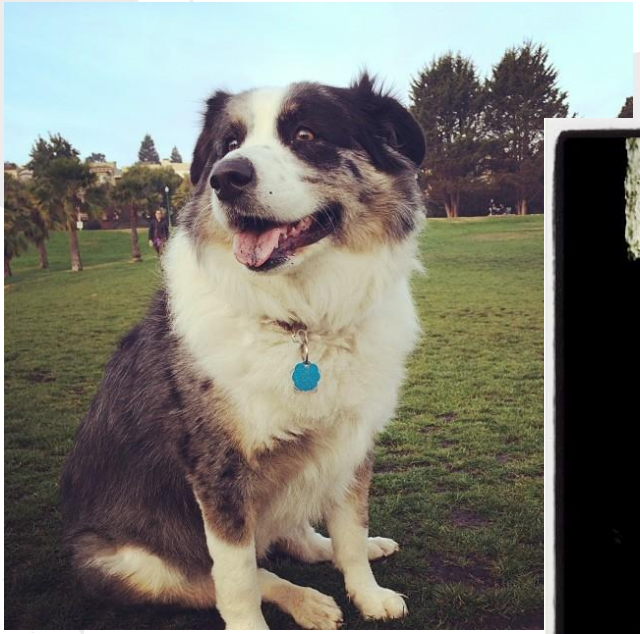
In the scheme of things, the topic of databases and all things pertaining to them is a huge one. It is a lot to cover, so I'll be glazing over a number of things and leaving out some others so I can cover the most commonly used or important parts.


Why this?


Bedroom
Bedroom
Closet
Bathroom
Bedroom
Office
Kitchen
Living Room

Who?

My Roommates



Wat?

Topography

Most Important Parts

- Tables
- Indexes

Additional Features

- Views
- Stored Procedures
- Lots more...

Tables

- Columns
- Rows
- Naming
  - Tables
  - Columns

Column Data Types

Strong vs. Weak Typing

Nullable vs. Not

Common Ones:

- text
- variable length characters (varchars)
- ints
- floats/reals
- blob
- a bunch more

Data Types pt. 2

- Non-SQLite DBs:
  - date (with or without timezone)
  - booleans
  - enums
- Uncommon ones:
  - Postgres:
    - geospatial points/polygons
    - json

Primary Key

Has to be unique.

Has to be not null.

Your Stuff aka “items”

name
color
color family (red, orange, yellow, …, violet)
type (shirt, book, dish, tech)
storage location (kichen, living room, my room)

What types are these things?

Is name an int?

How would you store color?

This is your “items” table

1	Name	Type	Location	Color Family	Hex Color
2	My Favorite Boots	Shoes	Bedroom	Black	#000000
3	Suede Oxfords	Shoes	Bedroom	Blue	#007FFF
4	Flip Flops	Shoes	Bedroom	Blue	#62B1F6
5	Motorcycle Jeans	Pants	Living Room	Black	#000000
6	Teal Pants	Pants	Living Room	Blue	#00868B
7	Big Teacup	Dishes	Kitchen	Blue	#8FD8D8
8	Chambray Button Up	Shirt	Bedroom	Blue	#9BC4E2
9	White T-Shirt	Shirt	Bedroom	White	#FFFFFF
10	Red Check Silk Button Up	Shirt	Bedroom	Red	#FF2400

Can we do better?

What would happen if you misspelled ‘blue’ as ‘bule’?

Normalization

Minimize redundancy and dependency unless you have a good reason.

These are your tables

name	item_type_id	location_id	hex_color_id
My Favorite Boots	1	1	1
Suede Oxfords	1	1	2
Flip Flops	1	1	3
Motorcycle Jeans	2	2	4
Teal Pants	2	2	5
Big Teacup	3	3	6
Chambray Button Up	4	1	7
White T-Shirt	4	1	8
Red Check Silk Button Up	4	1	9

id	name
1	Shoes
2	Pants
3	Dishes
4	Shirt

id	name
1	Bedroom
2	Living Room
3	Kitchen

id	name
1	Blue
2	White
3	Red
4	Black

id	hex_value	color_family_id
1	#000000	4
2	#007FFF	1
3	#62B1F6	1
4	#000000	4
5	#00868B	1
6	#8FD8D8	1
7	#9BC4E2	1
8	#FFFFFF	2
9	#FF2400	3

How?

OH CRUD

- SQL
  - DDL: CREATE, ALTER, RENAME, TRUNCATE, DROP
  - DCL: GRANT, REVOKE
  - DML: INSERT, UPDATE, DELETE, SELECT

Whitespace

```
sqlite> select * from color_families where name like '%r%';
2|purple
4|green
5|orange
7|red
sqlite> select * from color_families
   ...> where
   ...>     name like '%r%';
2|purple
4|green
5|orange
7|red
sqlite>
```

CREATE

CREATE [TABLE, INDEX, etc] (PROPERTIES)

create table hex_colors (id integer primary key autoincrement, hexvalue text);

create index storage_location_idx on items (storage_location_id);

INSERT

INSERT INTO table_name (column1, column2, column3, column4, ...) values (value1, value2, value3, value4);

INSERT INTO ITEMS (name, item_type_id, storage_location_id, hex_color_id) values (“My New Thing!”, 2, 4, 5);

UPDATE

UPDATE table_name SET column1=value1, column2=value2,...columnN=valueN (WHERE columnA=valueA...);

update items set name=”Whoops!”;

update items set name=”New Name” where id = 3;

DELETE

DELETE FROM table_name (WHERE column1=value1...);

DELETE FROM items; # NOOO WHYYYYYYY

DELETE FROM items where id = 1;

DELETE FROM items where name = “Bad Item”;

SELECT

SELECT column1, column2... FROM table_name WHERE column1=value1,... ORDER BY column1 (asc|desc);

SELECT * FROM items; (all rows)

SELECT * FROM items ORDER BY name desc; (all rows)

SELECT * FROM items limit 1; (1 row)

SELECT * FROM items where id = 1; (maybe not the same row)

SELECT Part Deux - JOINS!

select * from items inner join hex_colors

on items.hex_color_id = hex_colors.id; (oooh now we're getting fancy!)

select items.name, hex_colors.hex_value from items inner join hex_colors on items.hex_color_id = hex_colors.id where hex_colors.value like '%FF%'; (what do you think this returns?)

Indexes

Nearly all performance and scalability issues can be linked to interactions with indexes.

There is an automatic index on primary key.

How do you know what to add indexes on?

Transactions

Groups of statements executed all or none.

```
BEGIN;
statement1;
statement2;
COMMIT;
```

DBs + Python

Plain SQL via Python is more painful than it needs to be. Let's use an ORM!

Let's see this in action.

https://github.com/tiny-mouse/daterbasers

These are your tables

name	item_type_id	location_id	hex_color_id
My Favorite Boots	1	1	1
Suede Oxfords	1	1	2
Flip Flops	1	1	3
Motorcycle Jeans	2	2	4
Teal Pants	2	2	5
Big Teacup	3	3	6
Chambray Button Up	4	1	7
White T-Shirt	4	1	8
Red Check Silk Button Up	4	1	9

id	name
1	Shoes
2	Pants
3	Dishes
4	Shirt

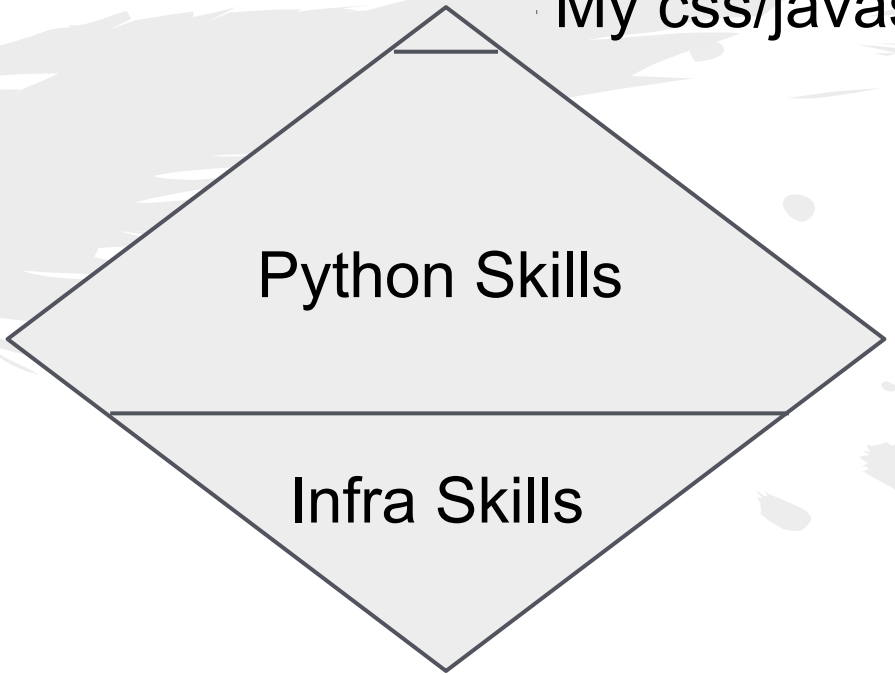
id	name
1	Bedroom
2	Living Room
3	Kitchen

id	name
1	Blue
2	White
3	Red
4	Black

id	hex_value	color_family_id
1	#000000	4
2	#007FFF	1
3	#62B1F6	1
4	#000000	4
5	#00868B	1
6	#8FD8D8	1
7	#9BC4E2	1
8	#FFFFFF	2
9	#FF2400	3

Disclaimer Number Two

My css/javascript/html skills.



All Things Fail Sometime

- Snapshots/Backups: early and often
- Master -> Slave
  - replication
  - promotion to master
- Security
  - password storage (hashes need salt, nom)
  - snapshot/backup storage

DB++

- Migrations

https://pypi.python.org/pypi/alembic

- Sharding
- Readonly/Readwrite

Alternative to RDBMS

NoSQL

MongoDB, Redis

Graph DBs

Questions?

Additional Reading/Resources

Alembic: https://pypi.python.org/pypi/alembic/0.6.3

Sqlalchemy:

http://www.sqlalchemy.org/

MySQL:

- Example DBs: http://dev.mysql.com/doc/index-other.html

- Detailed slides: http://cloud.github.com/downloads/eklitzke/mysql-minutiae/presentation.pdf

Postgres: http://wiki.postgresql.org/wiki/Sample_Databases