Day 7

This is CS50 for MBAs. Harvard Busines School. Spring 2015.

Cheng Gong

Table of Contents

News

- Check out some of Colton's music here[1].
- In other news, the DNS for `cs50.harvard.edu` should be fully updated by now across HBS' servers.
- Tor is an anonymization network, where people's data is bounced between other people on the network, so tracing where the data really came from becomes much harder.
- There's this device available, called an Anonabox, that you can just plug into to access the Tor network, instead of downloading and configuring software. But there were major security flaws, like the lack of password protection for its WiFi and the ability for anyone nearby to snoop on its traffic, according to Wired[2]. So the takeaway here is to not trust mysterious projects so easily.

Questions

- Are gem installs in ruby similar to installing frameworks like Flask in Python?

[1] https://cs50.harvard.edu/music
[2] http://www.wired.com/2015/04/anonabox-recall/

  - We installed the package with a program called `pip`, and in Ruby packages are called "gems," so the short answer is yes.
- What's the difference between an object oriented language and the languages we've talked about so far?
  - Object-oriented languages have the convention of storing functionality inside objects that contain related data and functions, which means that, for example, a data structure like a stack will have operations like pop or push (getting or putting items, respectively) and properties like size. In a language like C, the same can be achieved, but with more code and less readability than in object-oriented languages. Most of what we do now actually is object-oriented in some sense.
- Christina introduces herself as a course assistant who's been at office hours to answer questions, and is also a full-time PM at TripAdvisor.
- Are there inherent advantages in using procedural, functional, and object-oriented languages in the same way that there are advantages in compiled vs. interpreted languages?
  - Different languages lend themselves to different kinds of jobs more easily, so yes, as they are different tools.
- Are all programming languages in English?
  - In general they are, though programmers might put comments and variable names in their own language. There is a list here: wikipedia.org/wiki/Non-English-based_programming_languages[3] and a discussion here: blogs.msdn.com/b/alfredth/archive/2011/07/21/why-are-all-programming-languages-in-english.aspx[4].
- What were the key takeaways of today's [yesterday's] class? I followed the demos, but was not sure what the generalizable principles are. I took away:
  - You can create functions so you dont have to re-write. These can be powerful and speed up things in all manner of ways
    - David: This speeds up development time, but performance might be a tiny bit slower. Though the benefit does exceed the costs in this case, since we have such fast computers these days.
  - You can use other people's pre-coded functions via libraries

[3] http://en.wikipedia.org/wiki/Non-English-based_programming_languages
[4] http://blogs.msdn.com/b/alfredth/archive/2011/07/21/why-are-all-programming-languages-in-english.aspx

  - You can use python to change the html code itself
  - Your code can pull from the url line What else was a generalizable lesson? (there were certainly several wow moments too—the spiderman ones, but were they generalizable?)
  - David has these:
    - Using one language to generate another, particularly for the web, like Python generating HTML.
    - Code can be structured according to "design patterns" and MVC is one such example, where through years of collective experience models of development that work well have become conventions known as "design patterns". Good developers would be familiar with these patterns.
    - There are multiple ways to solve a problem. Python is certainly not the only back-end language that can generate HTML, and today we'll look at multiple database technologies.
- I'm afraid that I cannot access "Discuss" so would like to mention my problem here. I could run all python programs in the folder "src6" well except froshims-3 and hello-2.
  - Ask us at the end of class so we can help in person.
- For this assignment, I changed the sentence within the hello.html file to something other than hello, world and it did not update on the website. Is this because I had to change the python code instead of the html file?
  - You might need to save the file and reload the browser, but if still having trouble we can definitely help at the end of class.

Another News

- Stack Overflow, a discussion board website that many developers use to ask and answer technical questions, recently released its annual Developer Survey[5].
- This year they had about 26,000 or respondents, and there are facts like the country they are from, the number of developers per capita, age (younger, towards the 20s and 30s), gender (predominantly male), experience (women have less, but it's a sign that

[5] http://stackoverflow.com/research/developer-survey-2015

more women are beginning to enter the field), education (people tend to be self-taught), most popular technologies (JavaScript, SQL, and Java are in the lead), and more.

- Though remember that this sample is of the developers who both use Stack Overflow and respond to their surveys, so it might be fairly biased.

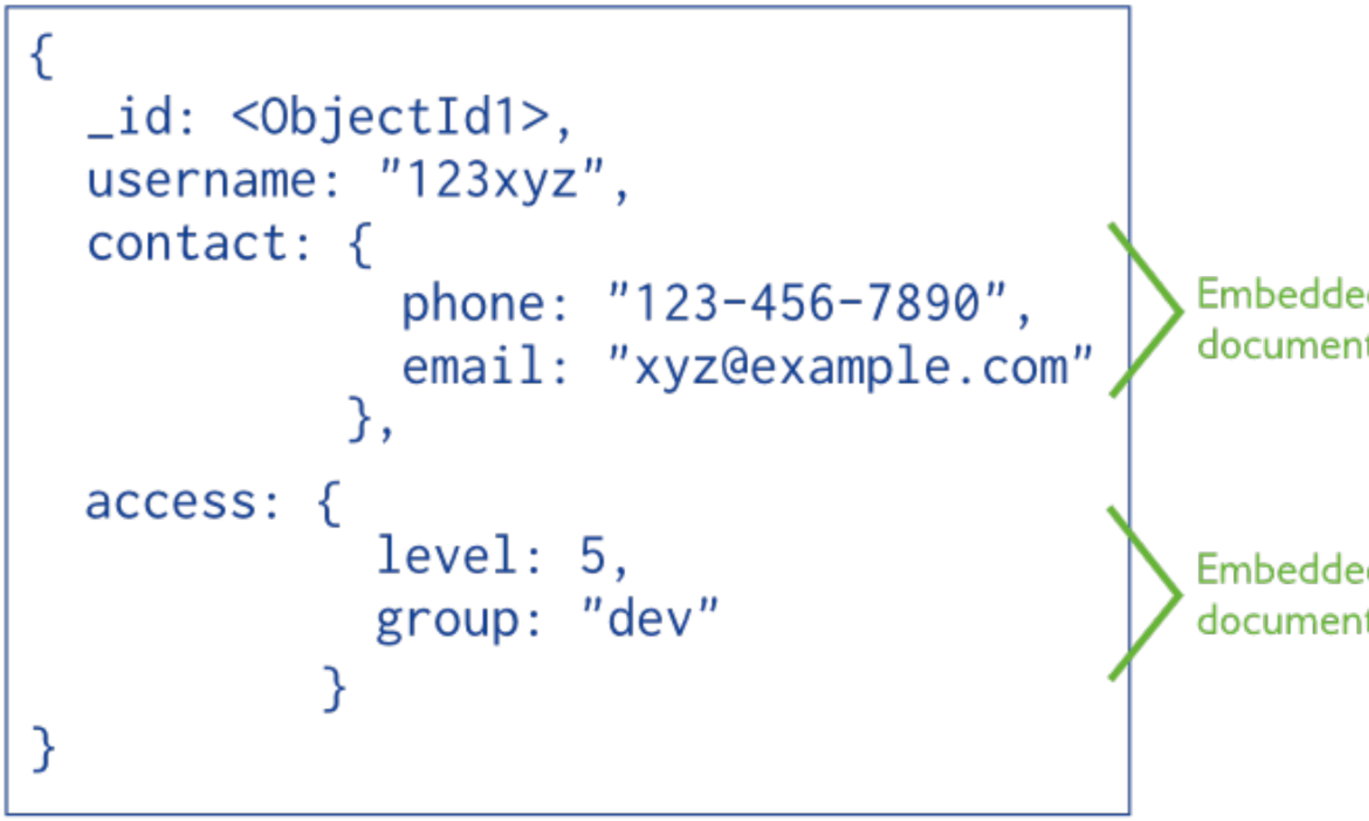
Storing Data

- Yesterday, all the pages we generated were thrown away. We could use caching, which saves the more popular pages so we save some processing time next time we need the page.
- But we need a place to store data, not just for caching but in general, like a customer's purchasing history.
- One of these technologies is called NoSQL, named because it's not SQL, an older query language (language that interacts with a database).
- NoSQL involves storing data in a format that look like this:

```
{
    "_id":"02134",
    "city":"ALLSTON",
    "loc":[
        -71.13286600000001,
        42.353519
    ],
    "pop":23775,
    "state":"MA"
}
```

- This format is called JSON, JavaScript Object Notation, and is just syntax and patterns for saving data.
- Notice that this chunk of data, from an abstract level, seems to represent the city of Allston. And programmers would think of it as an object, with characteristics that other city objects would also have.
- So we have the curly braces that contain everything, and inside a list of key-value pairs, as in a word in quotes, `"_id"` (the key), a colon, `:`, another word in quotes, `"02134"` (the value), and then a comma, `,`, signalling the next key-value pair.

- The value doesn't necessarily need to be a string in quotes. For `loc`, it looks like the value is an array, or list, given that there are square brackets `[]` around two things, separated by a comma `,`. `pop` too has a value of just `23775`.
- To store lots of these objects, we might separate them with commas after the curly braces. But there are no specifications, or schema, for what goes inside. In noSQL, we don't know what types of data we might have, but we have the ability to store nested data.
- For example, let's look at this picture:



```
{
   _id: <ObjectId1>,
   username: "123xyz",
   contact: {
                phone: "123-456-7890",
                email: "xyz@example.com"
              },
   access: {
               level: 5,
               group: "dev"
             }
}
```

# This came from the documentation of MongoDB, a popular NoSQl database. The object here seems to be representing a person's account on some system. It seems to have an `id` for itself, a `username` for this person, and contact information. But `contact` has the value of another object, as denoted by the curly braces, inside which are two more fields, `phone` and `email`. The author used another object instead of a list because now we can label the values within this value, and access

them more accurately. Otherwise, we might have needed to use `contact[0]` or the like to access values. But with nested objects, we can say something like `contact["phone"]`.

  - This also handles the case where the phone number is missing and `contact[0]` actually gets the email, or vice versa.
  - A better design decision, instead of not having anything present if a value is missing, is to use a special word like `null` to indicate that nothing is there.

- We could represent someone in an object like this:

```
{
    name: "Jihad",
    age: 25,
    phone: null,
    address: "123 Main Street Allson MA 02163"
}
```

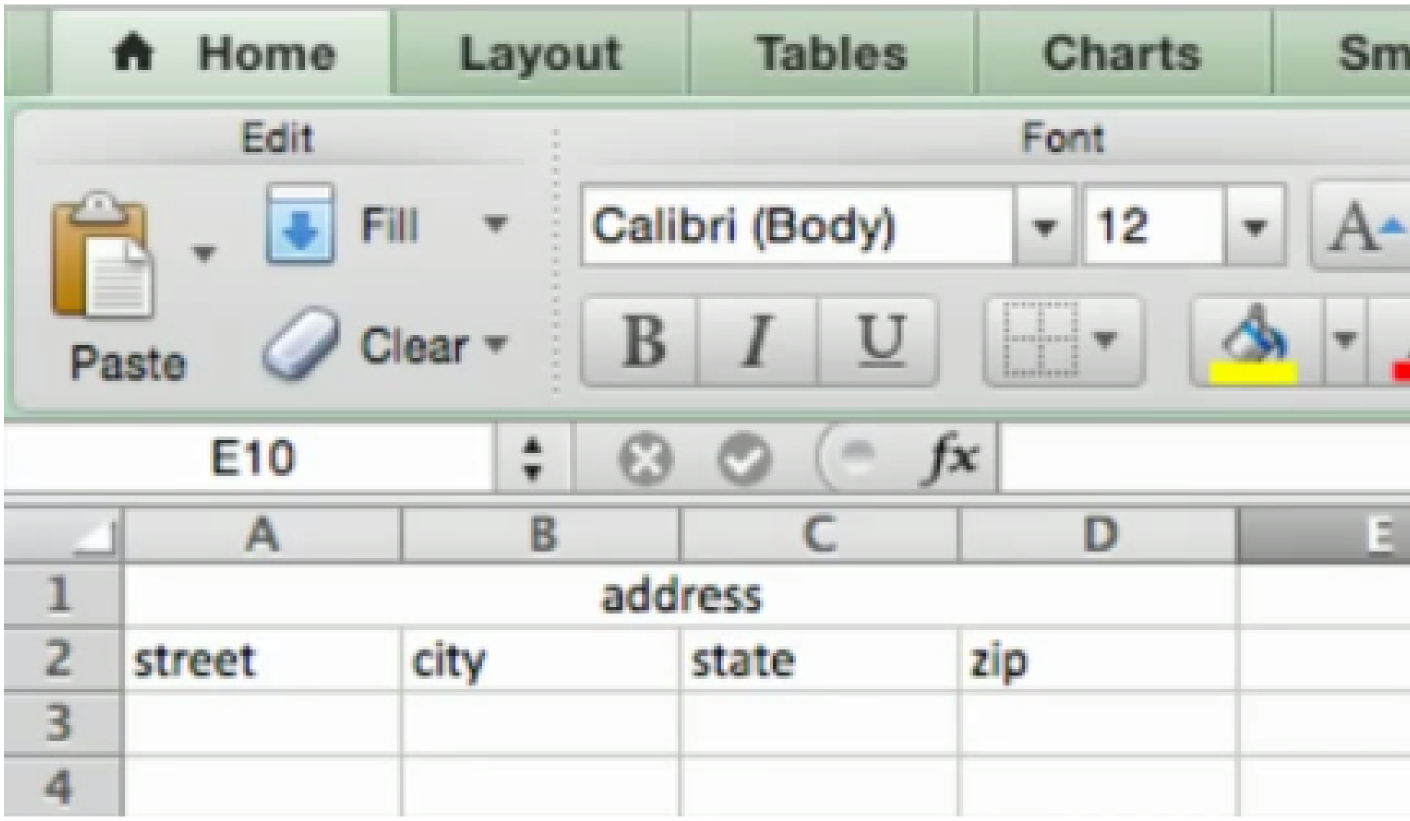
  - But notice that the address actually has a lot of data that is just tossed in, where there's no way to get just the zip code without getting the whole address, and even then, it might not be there in the order we expect it.

- So we can improve it like this:

```
{
    name: "Jihad",
    age: 25,
    phone: null,
    address: {
        street: "123 Main Street",
        city: "Allston",
        state: "MA",
        zip: "02163"
    }
}
```

  - The leading zero in the zip code will disappear when we save it as a number, so we'll save it as a string.

- If we had just an Excel file with columns, we might have something that looks like this:



	A	B	C	D
1	address			
2	street	city	state	zip
3				
4				

# We're trying to store the hierarchy here, and we could actually be even more granular:

```
{
    name: "Jihad",
    age: 25,
    phone: null,
    address: {
        street: {
            number: 123,
            basename: "Main",
            suffix: "Street"
        }
        city: "Allston",
        state: "MA",
        zip: "02163"
    }
}
```

    # So this could still be represented in an Excel file or SQL database, but it would be harder and less straightforward.

- And we have lots of people from the same city, so storing the same information about the city (city name and state and zip) seems unnecessary.
- So all we need is a zip code for the person, but information somewhere else, perhaps in another file:

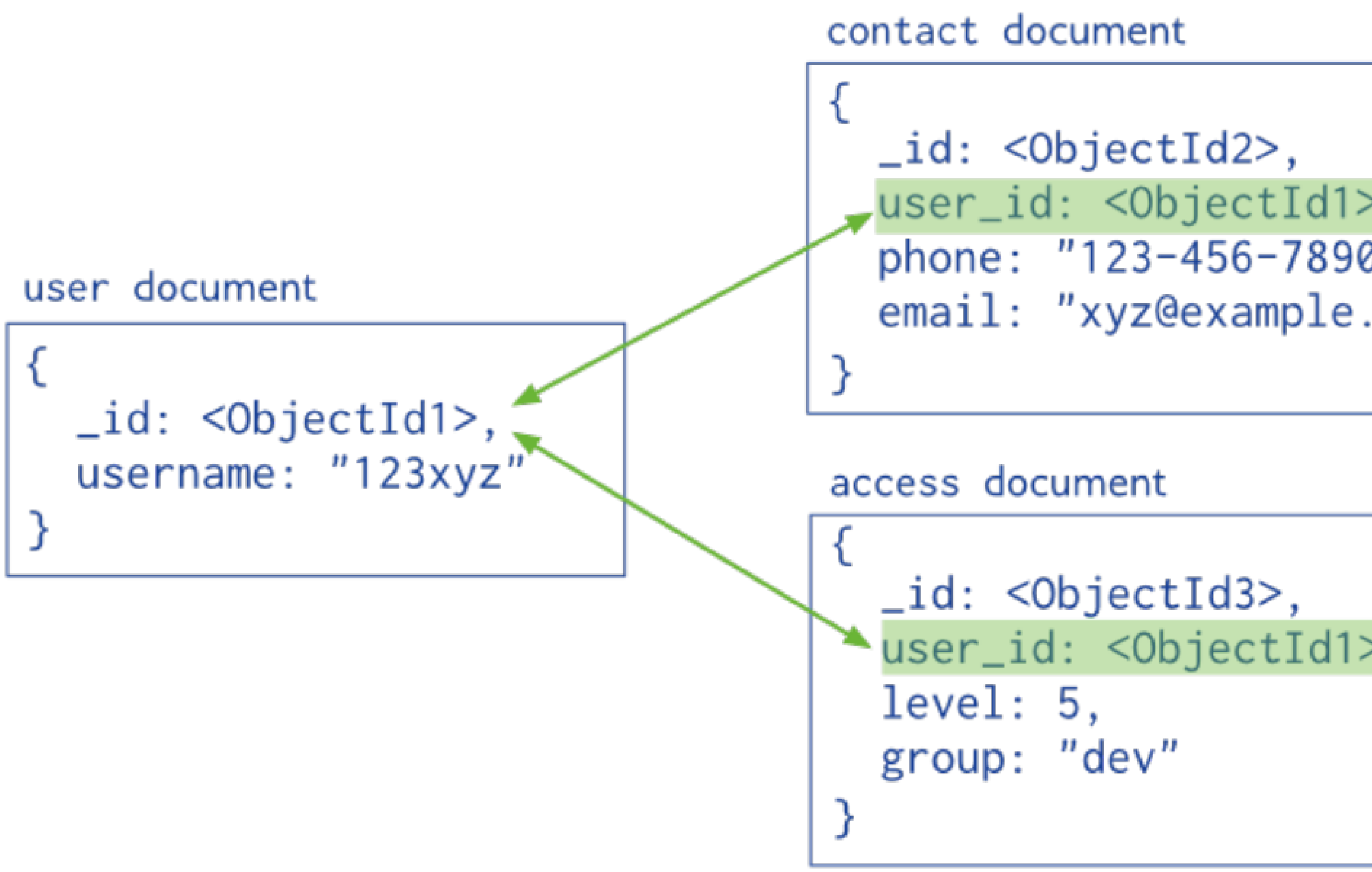
```
{
    name: "Jihad",
    age: 25,
    phone: null,
    address: {
        street: {
            number: 123,
            basename: "Main",
            suffix: "Street"
        }
        city: "Allston",
        state: "MA",
        zip: "02163"
    }
}

____

{
    zip: "02163",
    city: "Allston",
    state: "MA"
}
```

  # This is called normalization, where we just factor out common pieces that saves space and allows us to change it for everyone more easily, but the price now is more complex code to get the complete data and more time required.

- So in our original example, we can do this:



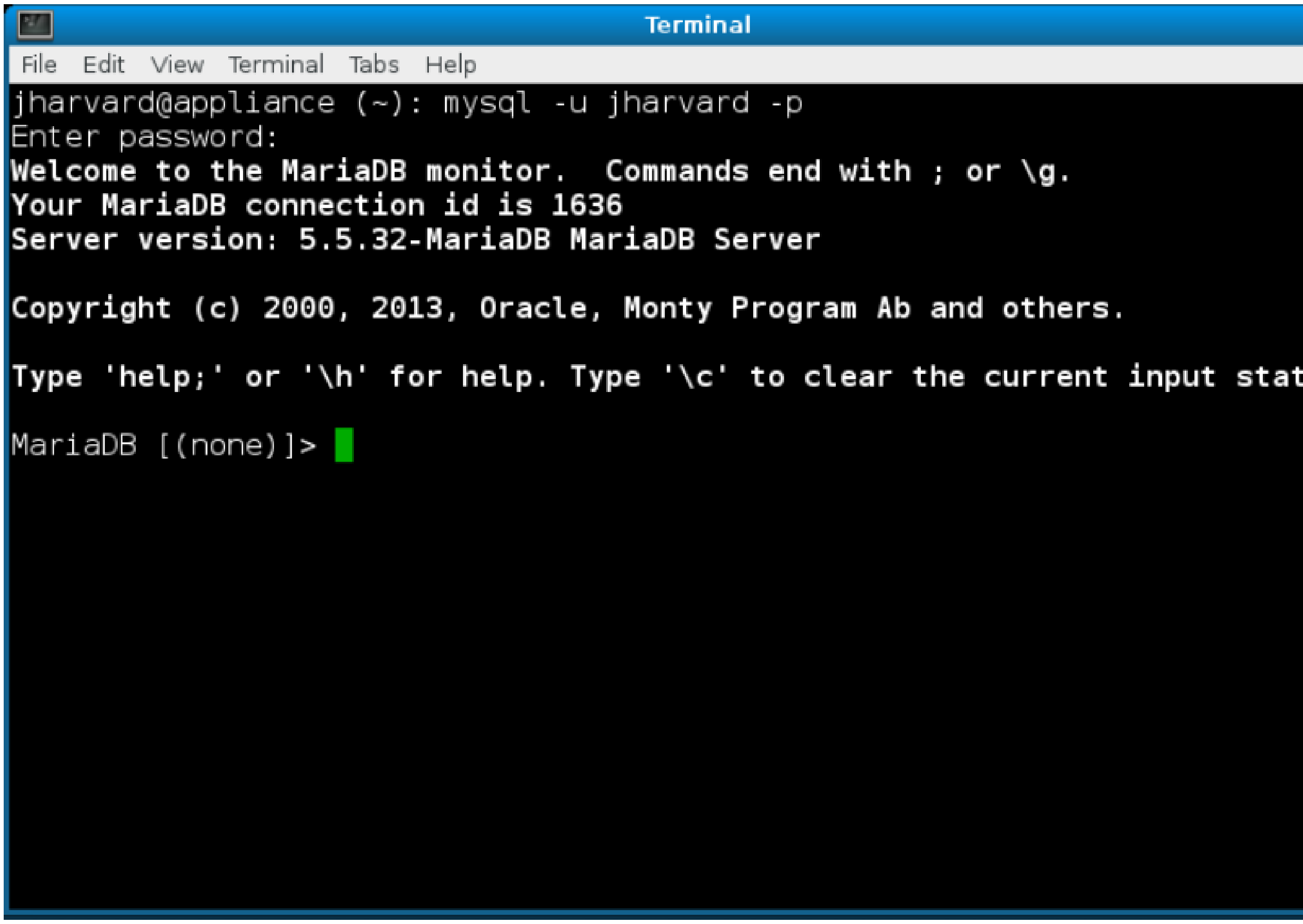
```
{
  _id: <ObjectId1>,
  username: "123xyz"
}
```

```
{
  _id: <ObjectId2>,
  user_id: <ObjectId1>
  phone: "123-456-789
  email: "xyz@example.
}
```

```
{
  _id: <ObjectId3>,
  user_id: <ObjectId1>
  level: 5,
  group: "dev"
}
```

# The user object has a `username`, that points to other objects, one called `contact` and one called `access`, that reference each other by means of some unique `id`.

- SQL is yet another language that allow us to query a database to manipulate data efficiently in a large data set.
- A SQL database is like an Excel file with multiple sheets, where each sheet is called a table in SQL.
- Queries include `DELETE`, `INSERT`, `UPDATE`, and `SELECT`.
- We can do this in a command prompt:



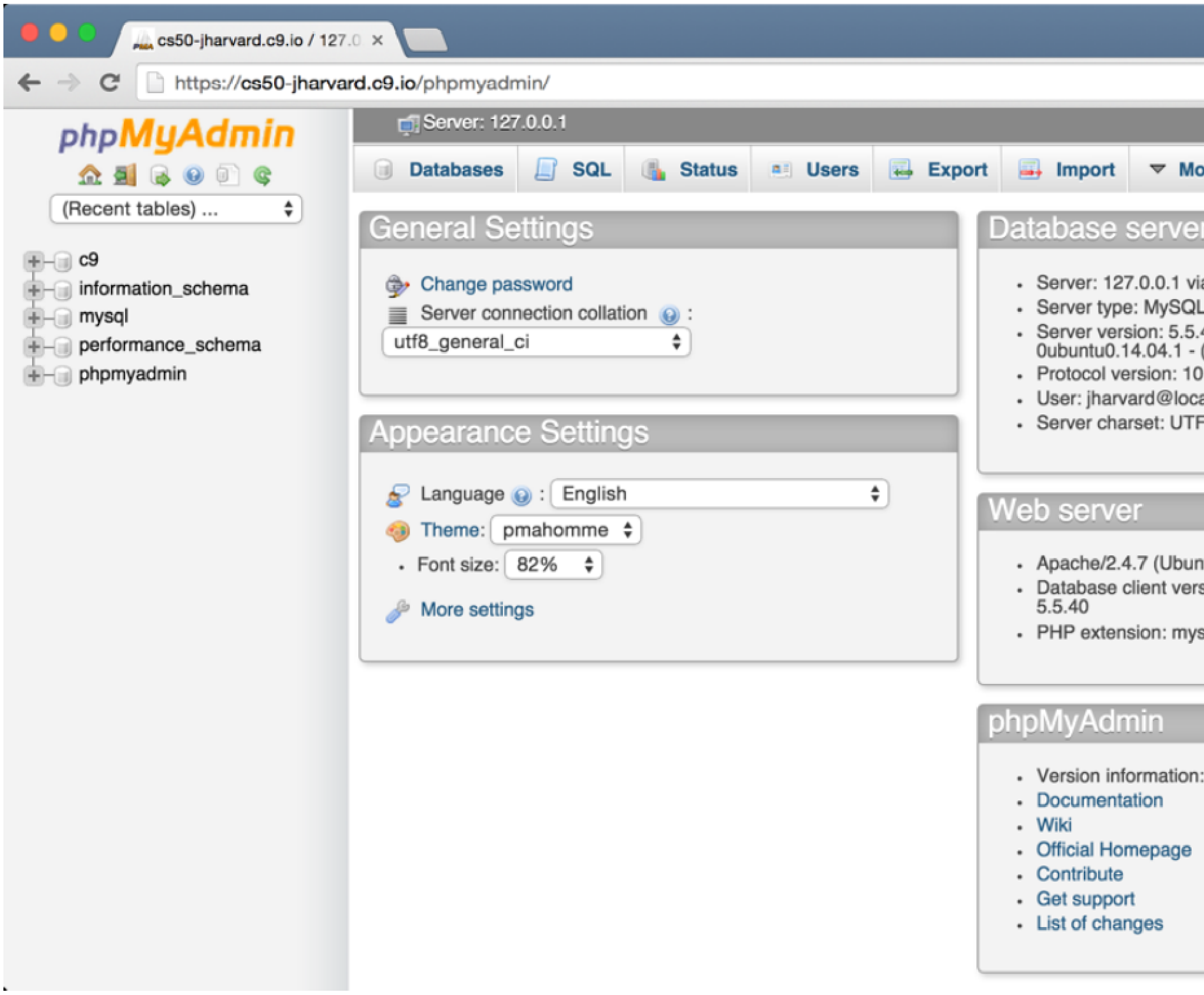
```
jharvard@appliance (~): mysql -u jharvard -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 1636
Server version: 5.5.32-MariaDB MariaDB Server

Copyright (c) 2000, 2013, Oracle, Monty Program Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input stat

MariaDB [(none)]>
```

- But there also exists a graphical user interface called phpMyAdmin:



c9
information_schema
mysql
performance_schema
phpmyadmin

General Settings

Change password
Server connection collation: utf8_general_ci

Appearance Settings

Language: English
Theme: pmahomme
Font size: 82%
More settings

Database serve

- Server: 127.0.0.1 via
- Server type: MySQL
- Server version: 5.5.4 0ubuntu0.14.04.1 - (
- Protocol version: 10
- User: jharvard@loca
- Server charset: UTF

Web server

- Apache/2.4.7 (Ubun
- Database client vers 5.5.40
- PHP extension: mys

phpMyAdmin

- Version information:
- Documentation
- Wiki
- Official Homepage
- Contribute
- Get support
- List of changes

- There are data types in SQL like:

  # `CHAR`

  # `VARCHAR`

  # `INT`

  # `BIGINT`

  # `DECIMAL`

  # `DATETIME`

  # …

- Someone's name might be `VARCHAR`, which means that it will be a variable length string made up of characters. Since people might have shorter or longer names, we should use `VARCHAR` as opposed to `CHAR`, which is a fixed length. But we do have to specify a maximum length, and the higher that length is, the slower the database's performance will be, since we wouldn't be able to randomly access elements. (Think back to arrays that had elements that were all the same size, so we could jump around easily.)
- `INT` is a 32-bit integer, and a `BIGINT` is 64-bits.
- A `DECIMAL` allows us to set the number of significant digits of the number, so we can have decimal numbers without the imprecision of floats.
- A `DATETIME` stores a date and time in the default format of `YYYY-MM-DD hh:mm:ss`, though we can format it in different ways.
- There are also indexes in SQL:
  - `PRIMARY`
  - `INDEX`
  - `UNIQUE`
  - `FULLTEXT`
  - Like Excel, data in SQL is stored in rows. This just means that we can set certain columns to be special in all the rows. For example, making a column `PRIMARY` means that it will be used to uniquely identify a row. This tends to be a number instead of a username, since it will have fixed size and be relatively small and easier to search through. `INDEX` tells the database that we plan to search for people in this field frequently, so it can optimize for searches on that field. But this also uses more memory, since we're creating additional data structures to get higher performance. It also takes time to update those structures, so if we choose to index more than we actually use, we might lose performance. `UNIQUE` also indicates that all the values in the column must be unique, even if we don't plan to use it to identify rows. (`PRIMARY`, then, has the characteristics of both `INDEX` and `UNIQUE`.) `FULLTEXT` is just for data of any length.
- Finally, there are different storage engines for databases, which just means that the binary zeroes and ones are stored on the hard drive somewhat differently by each of these:

# InnoDB

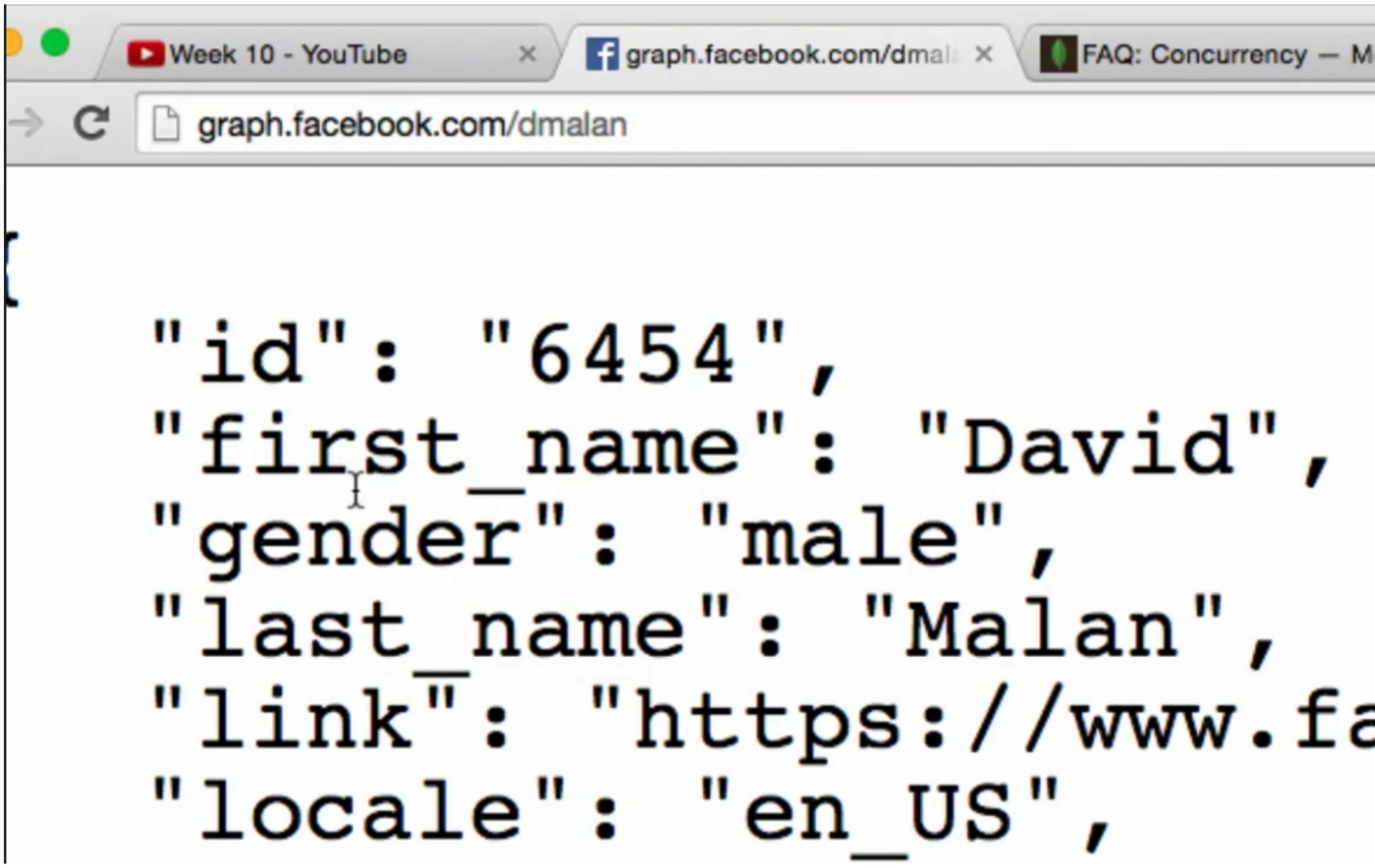
# MyISAM

# Archive

# Memory

…

- The Memory storage engine means that all of our data will be stored in RAM, making it good for performance and temporary storage, like for a cache.
- Archive means the data is pre-compressed, so we'd want to use it for data we write but not read too often, since it will take some time to decompress when we do but saves space on our hard drive.
- InnoDB and MyISAM also have different characteristics that give them advantages and disadvantages.
- If you have a Facebook username, you can see what your user `id` number on Facebook is by going to a URL like this (David is showing off how low his is):



```
{
  "id": "6454",
  "first_name": "David",
  "gender": "male",
  "last_name": "Malan",
  "link": "https://www.fa
  "locale": "en_US",
```

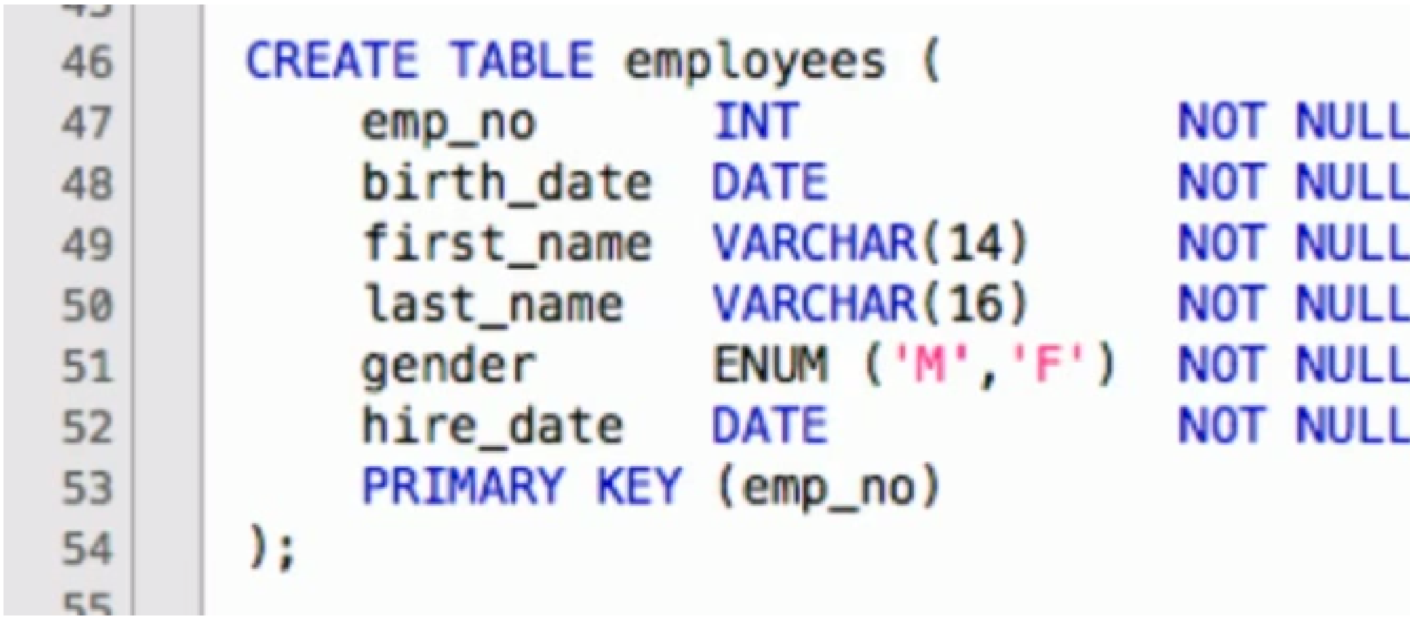
- Suppose you and your roommate share a fridge, and you both like milk. So you come home one day, and your roommate is still in class, and you see that there's no more milk in the fridge. So you leave for CVS to buy some milk. Meanwhile, your roommate gets home and realizes that there's no milk too, and happens to go to the nearby supermarket, and by the time you both return, you have two gallons milk instead of one. This was because you and your roommate didn't communicate with each other (texting didn't exist back in David's time, so this story made more sense then).
- This problem also comes from the fact that there is a lag between the time you see there is no milk and the time you get back with it.
- In the database world, operations that happen simultaneously with no lag are called atomic, so the milk replacement algorithm isn't atomic.
- One of you could have marked in the fridge that you were going to get milk, and analogously in the database world we "lock" a database when we want to do something to it without the state being changed by someone else.
- In SQL there is syntax like this (which we won't worry about the details of):

```
INSERT INTO table (id, symbol, shares) VALUES(7, "GOOG", 10)
    ON DUPLICATE KEY UPDATE shares = shares + VALUES(shares)
```

  # What we're doing here is adding some stocks to a user's account, and `ON DUPLICATE KEY UPDATE` is telling our database to update a row's values, instead of creating a new one, if one with the same key exists already.
- We could also do what's called a transaction:

```
START TRANSACTION;
UPDATE account SET balance = balance - 1000 WHERE number = 2;
UPDATE account SET balance = balance + 1000 WHERE number = 1;
COMMIT;
```

  # And this just guarantees that both of our queries will complete together, without interruption.
- MongoDB uses locking, and SQL supports both locking at the table level and the row level, so using transactions that only lock certain rows makes for better performance since other users can update the other rows.
- What you'll do tonight is to download a nice big database, with files that have queries that look like this:

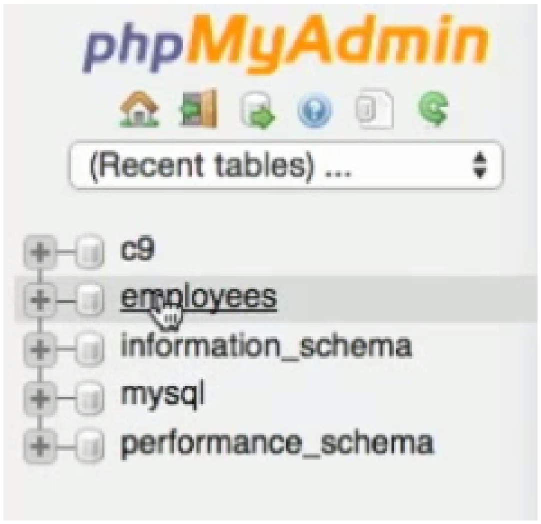


```
CREATE TABLE employees (
    emp_no      INT             NOT NULL
    birth_date  DATE            NOT NULL
    first_name  VARCHAR(14)     NOT NULL
    last_name   VARCHAR(16)     NOT NULL
    gender      ENUM ('M','F')  NOT NULL
    hire_date   DATE            NOT NULL
    PRIMARY KEY (emp_no)
);
```

- # It looks like this chunk is creating a table called `employees` with fields like `emp_no` and their types, and also a `NOT NULL` option that require the fields to have some value in every row.

- # `emp_no` is an `INT`, since we probably don't expect to have billions of employees, `birth_date` is a `DATE`, which makes sense, and `first_name` is `VARCHAR(14)`, which might seem short but being a sample database, the information probably won't be changed, so we know in advance what the longest name is. `gender` is `ENUM`, or enumeration, which means it will have the values of either `M` or `F`. And finally, `emp_no` seems to be made the primary key, which also makes sense.

- After we import all this data into Cloud9, we'll use a tool called phpMyAdmin. We see that our databases are on the left, and most of them already existed, except for the one called `employees`:



- Now if we click on it, we see lots of information:

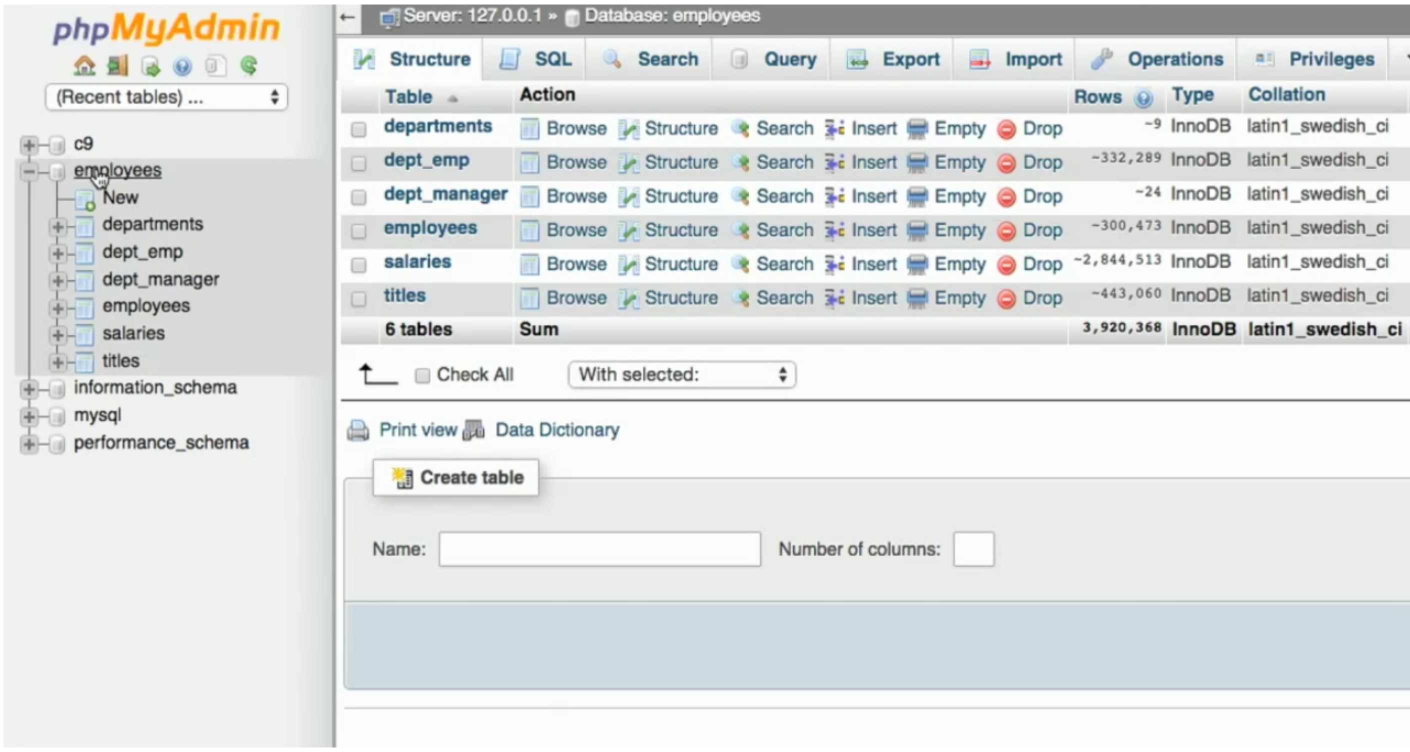
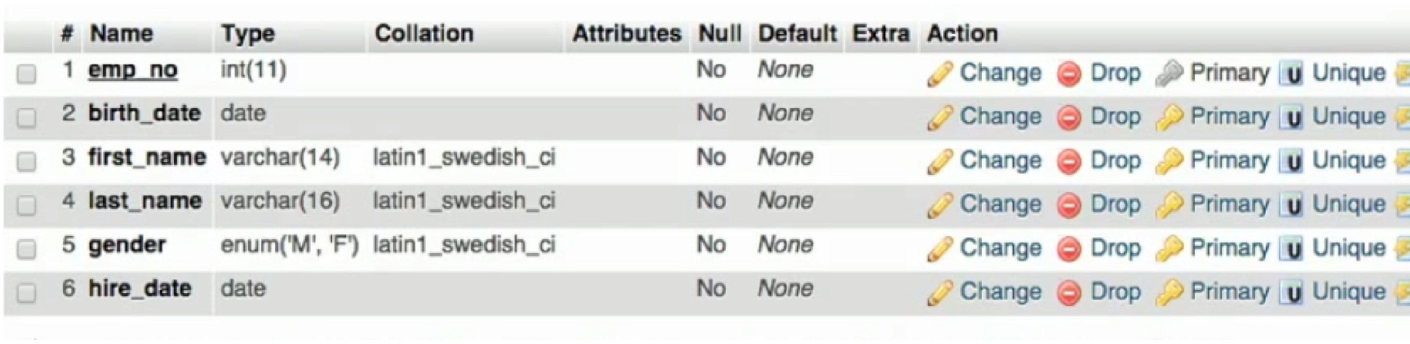


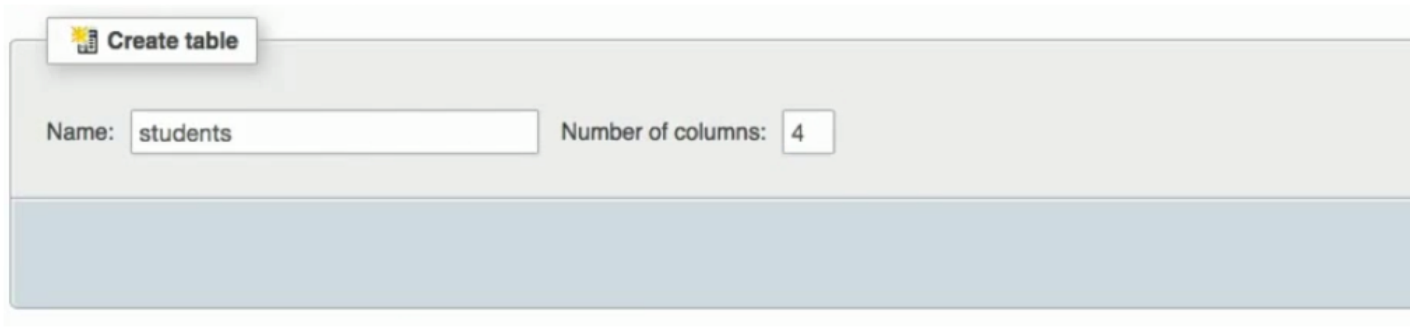
Table	Action						Rows	Type	Collation
departments	Browse	Structure	Search	Insert	Empty	Drop	~9	InnoDB	latin1_swedish_ci
dept_emp	Browse	Structure	Search	Insert	Empty	Drop	~332,289	InnoDB	latin1_swedish_ci
dept_manager	Browse	Structure	Search	Insert	Empty	Drop	~24	InnoDB	latin1_swedish_ci
employees	Browse	Structure	Search	Insert	Empty	Drop	~300,473	InnoDB	latin1_swedish_ci
salaries	Browse	Structure	Search	Insert	Empty	Drop	~2,844,513	InnoDB	latin1_swedish_ci
titles	Browse	Structure	Search	Insert	Empty	Drop	~443,060	InnoDB	latin1_swedish_ci
6 tables	Sum						3,920,368	InnoDB	latin1_swedish_ci

- If we click around some more, we see a graphical version of what we just saw:



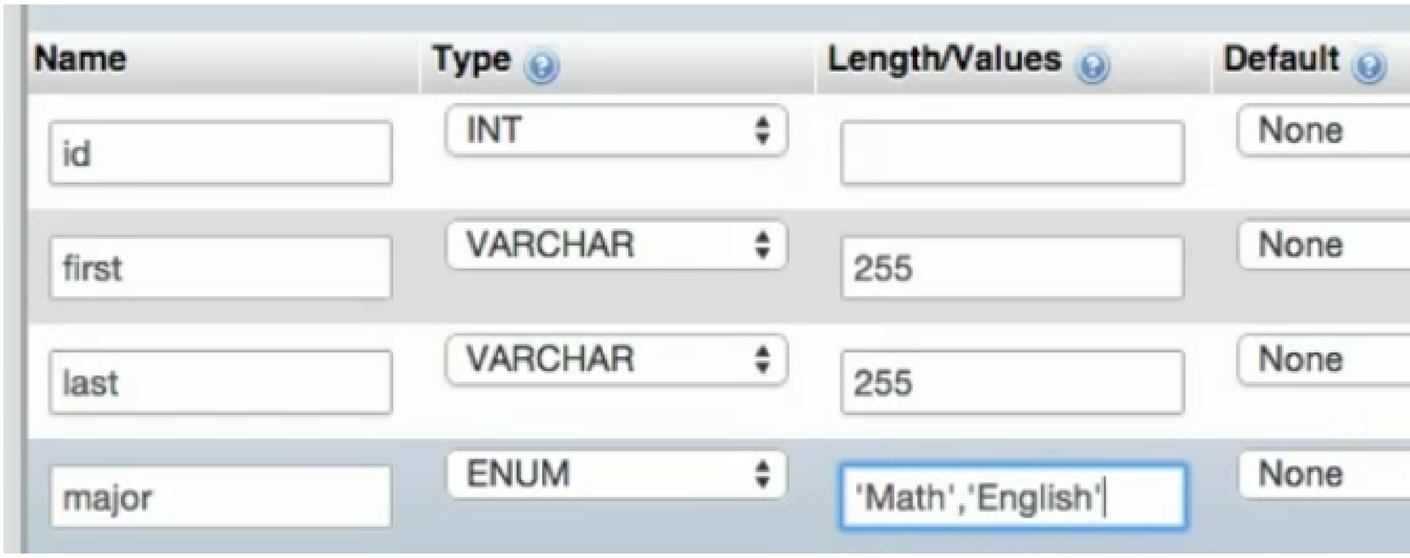
	#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
	1	emp_no	int(11)			No	None		Change Drop Primary Unique
	2	birth_date	date			No	None		Change Drop Primary Unique
	3	first_name	varchar(14)	latin1_swedish_ci		No	None		Change Drop Primary Unique
	4	last_name	varchar(16)	latin1_swedish_ci		No	None		Change Drop Primary Unique
	5	gender	enum('M', 'F')	latin1_swedish_ci		No	None		Change Drop Primary Unique
	6	hire_date	date			No	None		Change Drop Primary Unique

- It's also easy to create a table:



Create table

Name: students Number of columns: 4

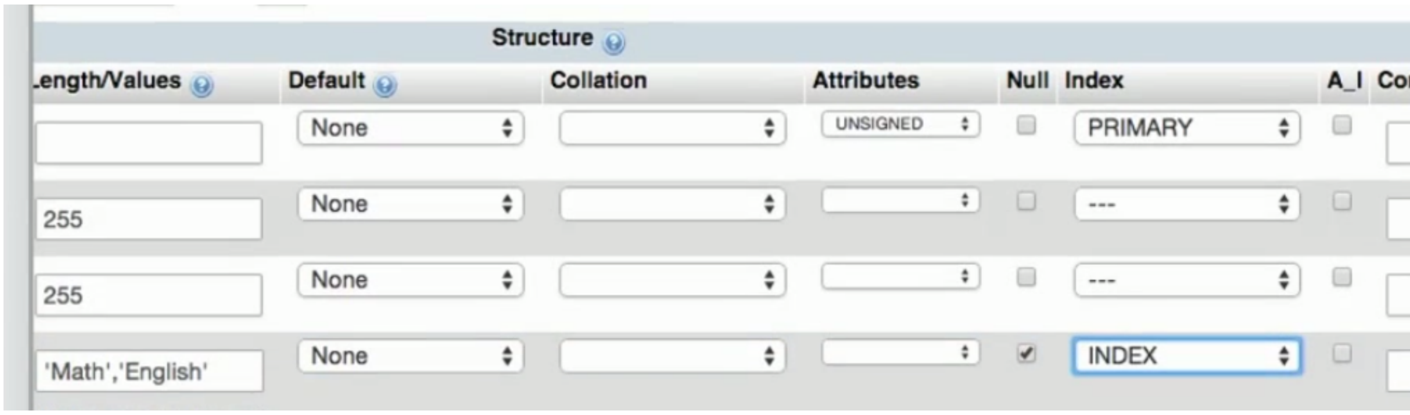


Name	Type	Length/Values	Default
id	INT		None
first	VARCHAR	255	None
last	VARCHAR	255	None
major	ENUM	'Math','English'	None

  # We've created some fields and set their data types as appropriate.

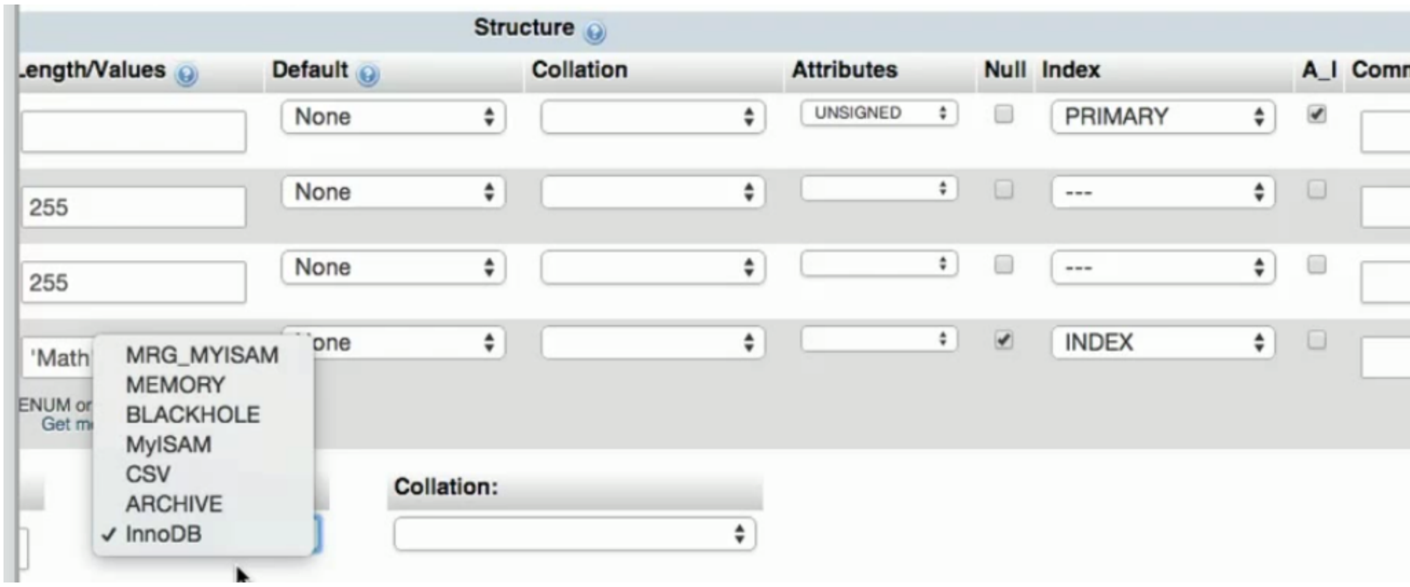
  # `TEXT` is another data type, for paragraphs and essays and longer text, but tends to be on the slow side. And there are lots more fancy ones, even for coordinates.

- There are more options for our fields: `COLLATION` being the encoding, like ASCII or others; `ATTRIBUTE` with an option of `UNSIGNED`, meaning that numbers can't be negative; `NULL` if we want any to be optionally empty, which we select for `major`; and `INDEX`, so we'll have `id` be our `PRIMARY` index, and `major` be `INDEX` too, so we can search on it more quickly:



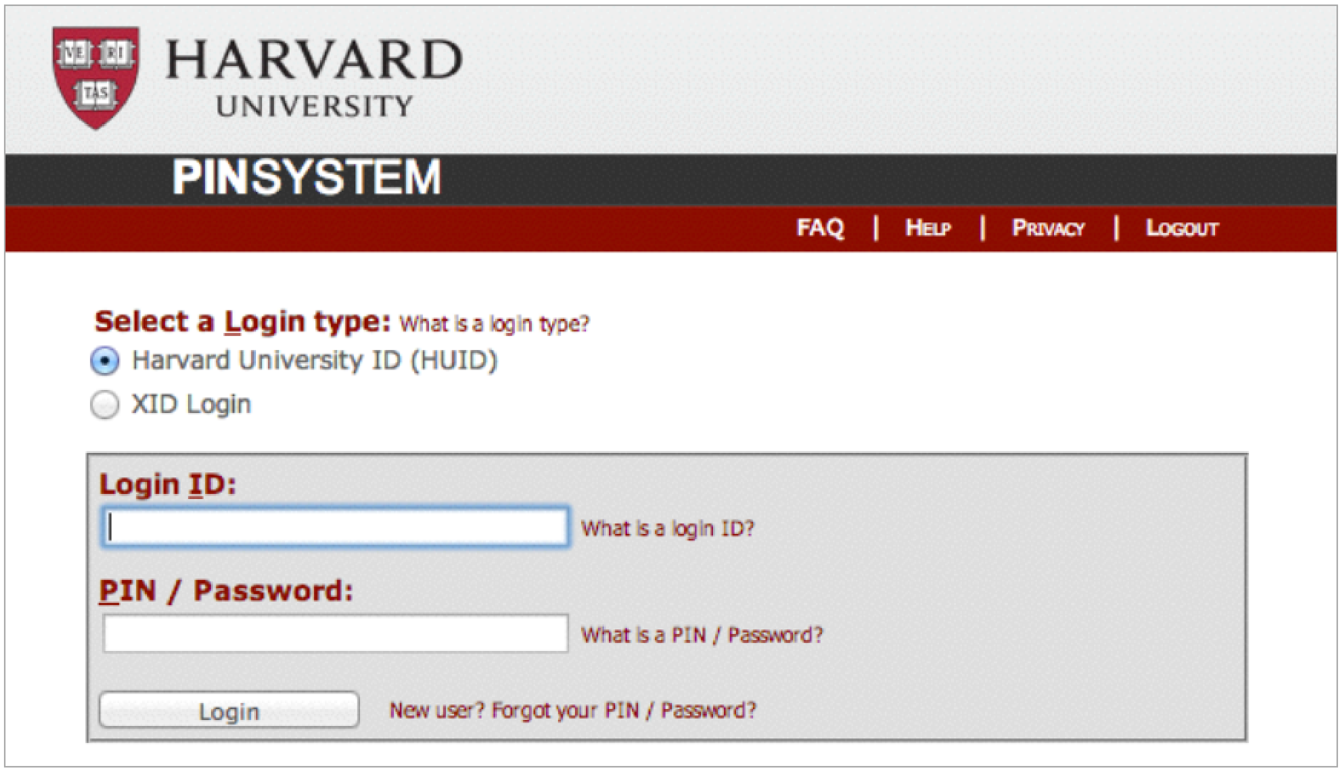
Length/Values	Default	Collation	Attributes	Null	Index	A_I
	None		UNSIGNED	☐	PRIMARY	☐
255	None			☐	---	☐
255	None			☐	---	☐
'Math','English'	None			✓	INDEX	☐

- `A_I` is short for `AUTO_INCREMENT`, which means that the database will automatically increment the field every time a new row is added. This is useful for ID numbers, when we don't care what it actually is, but that it's unique for every user. And we can also select the storage engine at the bottom:



Length/Values	Default	Collation	Attributes	Null	Index	A_I
	None		UNSIGNED	☐	PRIMARY	✓
255	None			☐	---	☐
255	None			☐	---	☐
'Math				✓	INDEX	☐

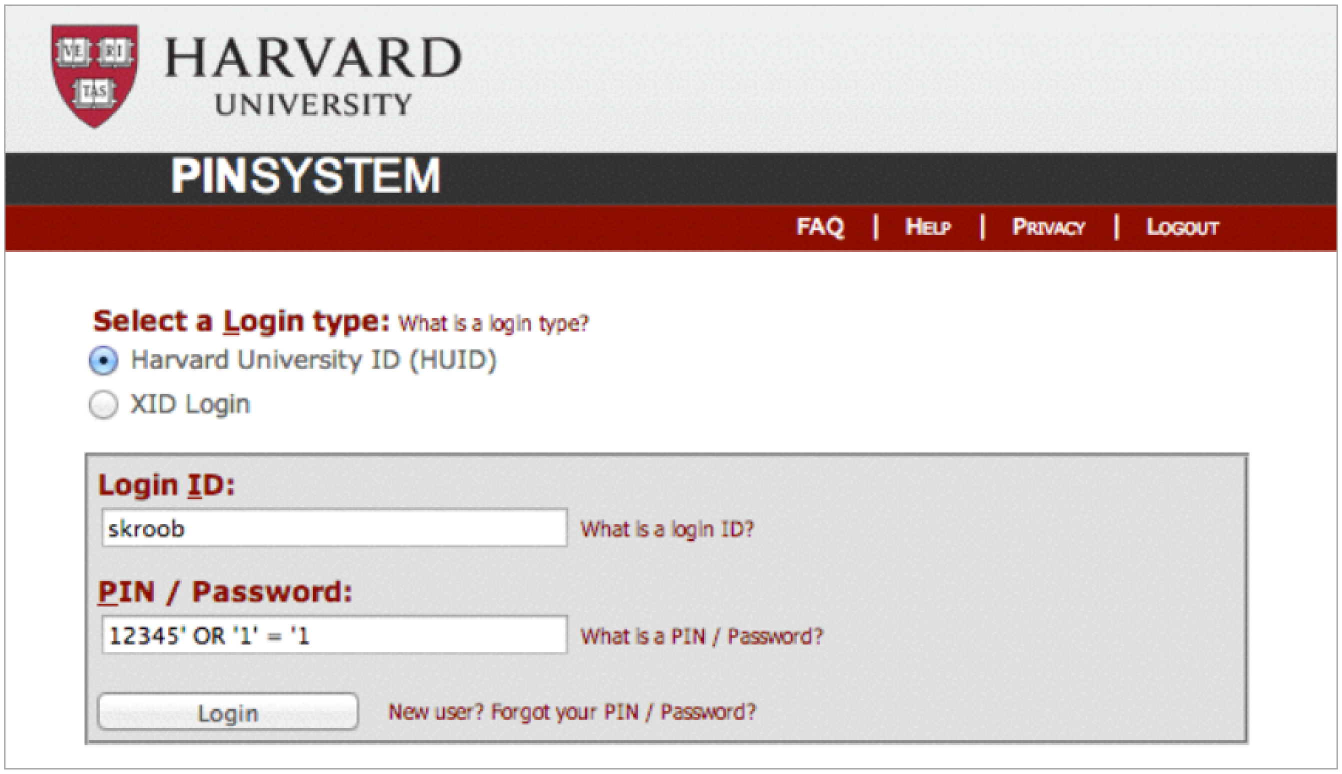
- So tonight we'll play around with databases and not get to learn every little detail, but just get a better understanding of how they work.
- **SQL injection attacks** are also a concern. Though the familiar Harvard PIN system isn't vulnerable to this attack, we'll use it for the sake of discussion.
- The PIN login screen looks like this:


HARVARD
UNIVERSITY
PINSYSTEM
FAQ | Help | Privacy | Logout
Select a Login type: What is a login type?
Harvard University ID (HUID)
XID Login
Login ID:
What is a login ID?
PIN / Password:
What is a PIN / Password?
Login
New user? Forgot your PIN / Password?

- Suppose that the code, somewhere behind this page, includes bits like this:

```
$username = $_POST["username"];
$password = $_POST["password"];
query("SELECT * FROM users WHERE username='{$username}' AND
 password='{$password}'");
```

- First we get the variables `$username` and `$password` from some form, and substitute them into our query, which looks (and is) correct.
- But a user could input something like this: (David changed the PIN field's usual bullets to actual text to reveal what the user has typed.)


HARVARD
UNIVERSITY
PINSYSTEM
FAQ | Help | Privacy | Logout

Select a Login type: What is a login type?
[✓ Harvard University ID (HUID)]
XID Login

Login ID: skroob	What is a login ID?
PIN / Password: 12345' OR '1' = '1	What is a PIN / Password?

Login
New user? Forgot your PIN / Password?

# `skroob` is the username (a reference to Spaceballs), but the password field is a bit fishy with `12345' OR '1' = '1`, especially with the single quotes.

- If we substitute that info into the form, the query will now look like this (note that `skroob` has left off single quotes on the outside of his password, assuming that the code for the query will, which it does):

```
$username = $_POST["username"];
$password = $_POST["password"];
query("SELECT * FROM users WHERE username='skroob' AND password='12345' OR
 '1' = '1'");
```

# So now we're selecting everything from the `users` table, `WHERE` username is `skroob` `AND` password is `12345`, `OR` `1 = 1`. Since `1 = 1` is always true, every row from the table will be selected.

- The takeaway here is that programmers should code defensively and anticipate users placing evil syntax that changes what they're trying to do.

- One fix is to disallow certain characters in the input, that we check for before we substitute it into our query, but this keeps us from choosing certain characters in our passwords or other input.
- The fix is to escape input, or convert the characters to something that looks like this:

```
$username = $_POST["username"];
$password = $_POST["password"];
query("SELECT * FROM users WHERE username='skroob' AND password='12345\'
 OR \'1\' = \'1'");
```

  # The single quotes from input are marked by a `\` in front of it, so it doesn't end the string but tells us it's part of the string. We added the `\` ourselves after getting the input, and before substituting it in our query.

- So a SQL injection attack can happen when a programmer places strings into queries carelessly. But there are libraries with functions that escape input that we can just use pretty straightforwardly.
- And now you can understand this xkcd[6].

  # We see that Robert's name has a single quote, ending the query, which is okay, but then we have another command called `DROP TABLE` that deletes a table.

[6] http://xkcd.com/327/