

INF1343, Winter 2011

Data Modeling and Database Design

Yuri Takhteyev
University of Toronto

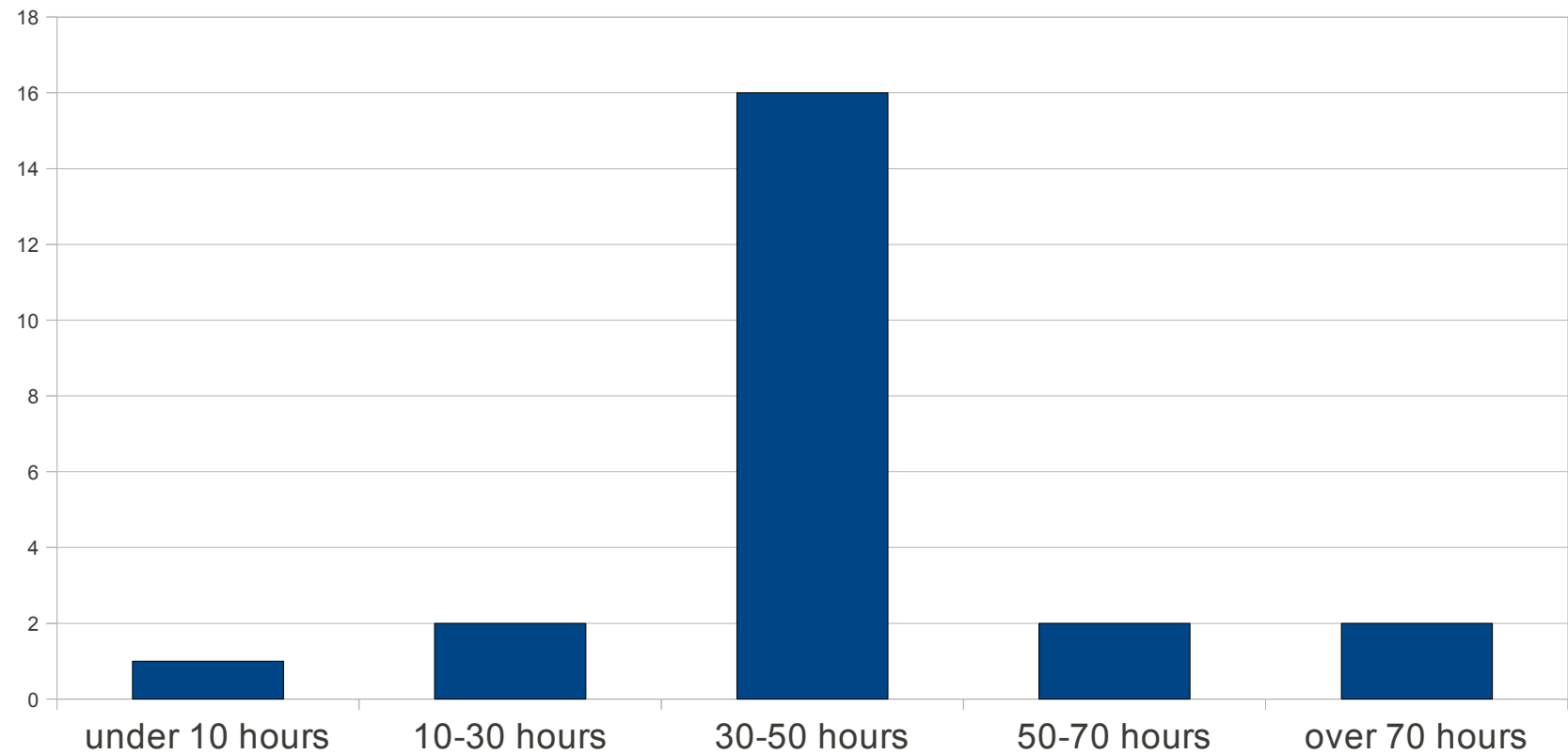


This presentation is licensed under Creative Commons Attribution License, v. 3.0. To view a copy of this license, visit <http://creativecommons.org/licenses/by/3.0/>. This presentation incorporates images from the Crystal Clear icon collection by Everaldo Coelho, available under LGPL from <http://everaldo.com/crystal/>.

Week 7

Embedded SQL

The Survey



The Final Project

(See the handout on the website)

Facebook runs on MySQL

facebook

Email

Keep me logged in

Password

Login

[Forgot your password?](#)

Sign Up

MySQL at Facebook is on Facebook

Sign up for Facebook to connect with MySQL at Facebook.



Devoted to making MySQL better. Written by people who work at Facebook.

Information

Founded:
2009

42,703 People Like This



Nesrine
Messai



Krishna
Chaitanya



Martin
Prayoga

MySQL at Facebook



Wall

Info

Notes

Discussions

Links

MySQL at Facebook + Others

Just MySQL at Facebook

Just Others



MySQL at Facebook I just registered for PgWest 2010. I look forward to learning more about current work on PostgreSQL.



PgWest 2010 | The PostgreSQL Conference
www.postgresqlconference.org

Thursday at 12:18pm · Comment · Like

Paul Saab, Bill Blum, Chetan Tanna and 51 others like this.

View all 7 comments



MySQL at Facebook I think they both have awesome technology and continue to improve.
Yesterday at 9:24am · 2 people · Flag



Ricky Elrod Indeed they do, I was just pulling your leg. Have fun at the conference :)
Yesterday at 9:26am · Flag

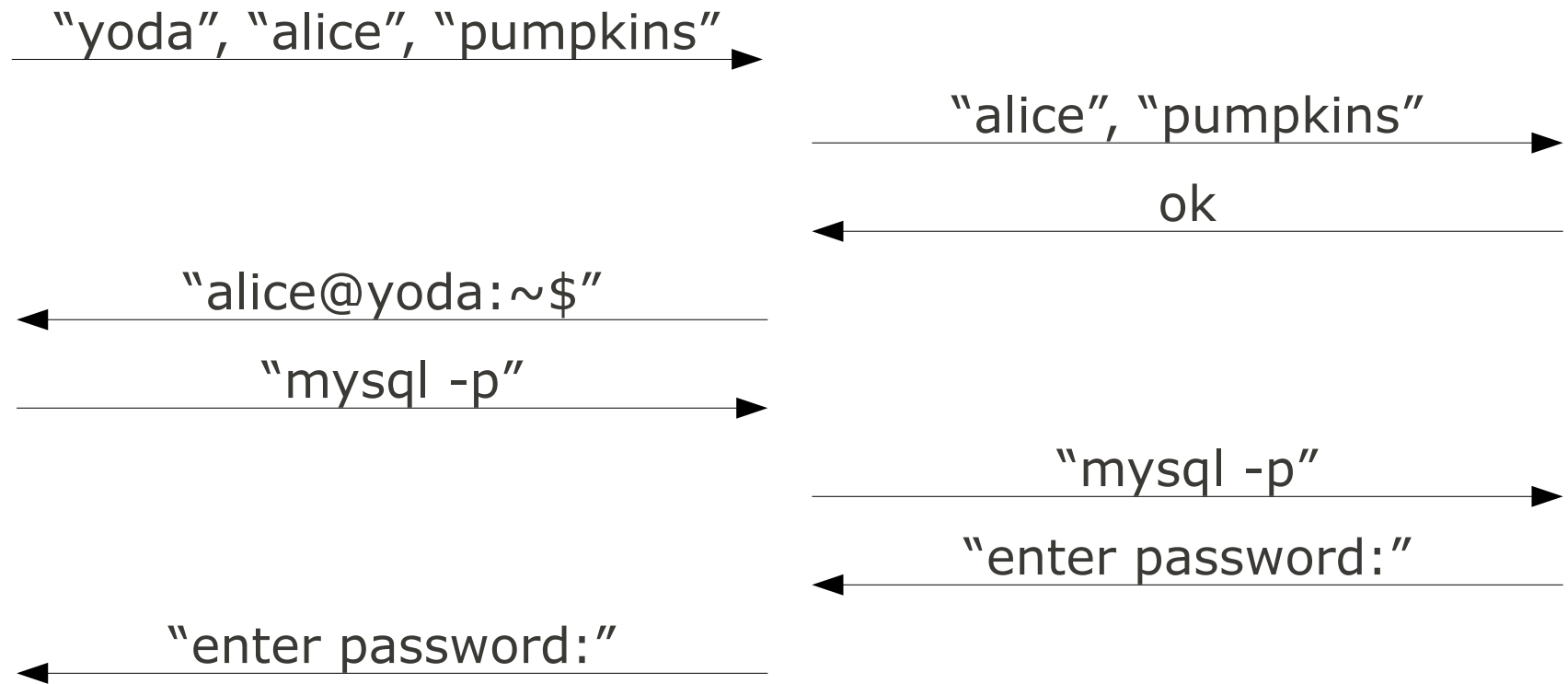


MySQL at Facebook Kristian Nielsen had a very interesting presentation on making replication better in MariaDB. Perhaps one day he will share that (hint, hint).

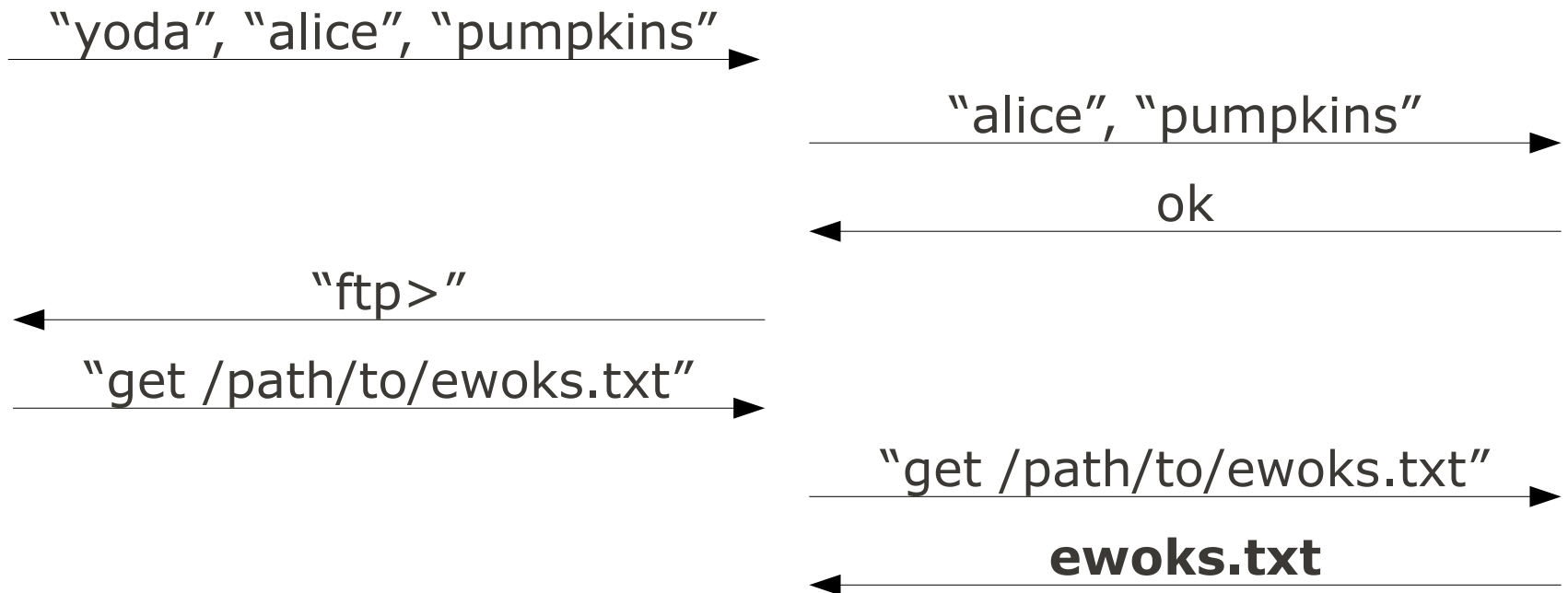
October 11 at 8:21am · Comment · Like

Kian Sin Teo, Donald G. Lane Jr., Roja Rani Penmetsa and 38 others like this.

Telnet (1969)



FTP (1971)



the file is saved locally

Anonymous FTP



"yoda", "/path/to/ewoks.txt" →

→ "get /path/to/ewoks.txt" →

← **ewoks.txt** ←

the file is saved locally

`ftp://yoda/path/to/ewoks.txt`

HTTP (1991)



"yoda", "/path/to/ewoks.html" →

"GET /path/to/ewoks.html" →

← the **content** of ewoks.html

← **ewoks.html** is displayed

`http://yoda/path/to/ewoks.html`

HTML

```
<html xmlns="http://www.w3.org/1999/xhtml">  
  <head>  
    <title>  
      Ewoks  
    </title>  
  </head>  
  <body>  
    <h1>  
      The List of Ewoks  
    </h1>  
    ...
```

See: <http://www.w3schools.com/html/default.asp>

On Yoda

1. Put the file in `~/public_html/`

2. Let the server see it:

`chmod a+r ewoks.html`

Locally:

```
scp ewoks.html kenobio1@yoda.ischool.utoronto.ca:~/public_html/
```

Remotely:

```
cd ~/public_html/  
chmod a+r ewoks.html
```

CGI



"yoda", "/path/to/ewoks.cgi" →

"GET /path/to/ewoks.cgi" →

← **the html output of ewoks.cgi**

← **html output** is displayed

ewoks.cgi contains **code**, but **returns** HTML

ewoks.cgi



ewoks.cgi

```
#!/usr/bin/python

print "Content-Type: text/html"
print
print "<h1>Ewoks</h1>"
```

Set permissions to `a+rx`

Remotely:

```
cd ~/public_html
mkdir cgi-bin
chmod a+x cgi-bin
cd cgi-bin
cp /play/ewoks.cgi .
chmod a+rx ewoks.cgi
```

Python Console

On Yoda:

type "python"

Locally:

Install from <http://python.org/>

Use Python **2.x**, not Python **3** ("2000").

Elements of Python

Constants and Expressions:

```
5
```

```
"Ewoks"
```

Variables:

```
count = 5
```

```
count = count + 1
```

```
species = "Ewoks"
```

```
"Hello, " + species + "!"
```


Elements of Python

Print:

```
print count  
print species, count
```

Functions:

```
print str(count)  
print str(count) + " " + species
```

Functions in Libraries/Modules:

```
import math  
print math.sqrt(2)
```

Python is “Procedural”

(We describe the steps rather than the results.)

ewoks2.cgi

```
#!/usr/bin/python
import cgi
cgi.enable()
print "Content-Type: text/html"
print

species = "Ewoks"
count = 5
print "<h1>", species, "</h1>"
print count, species, "<br/>"
count = count + 1
print count, species, "<br/>"
```

Elements of Python

Lists:

```
names = ["Chirpa", "Warok",  
———"Wicket"]  
print names[1]  
print len(names)
```

Loops:

```
for name in names:  
——print name
```

ewoks3.cgi

```
#!/usr/bin/python
import cgi
cgi.enable()
print "Content-Type: text/html"
print

names = ["Chirpa", "Warok", "Wicket"]
species = "Ewoks"

print "<h1>", len(names), species
print "</h1>"
for name in names:
    print name, "<br/>"
```

Always the Same?

Parameters

- additional data from the user

Database

- information that we store

MySQLdb

```
import MySQLdb

config = "/home/kenobio1/.my.cnf"
db = MySQLdb.connect(db="starwars",
                    read_default_file=config)
print db

cursor=db.cursor()
query = "select * from persona";
cursor.execute(query);
rows = cursor.fetchall()
print len(rows)
print rows[0]
```

Elements of Python

Objects:

```
db = MySQLdb.connect(...)
```

Methods (Objects' Functions):

```
cursor = db.cursor()  
cursor.execute(query)  
cursor.fetchall()
```


Loops Revisited

A Loop over the Rows:

```
for row in rows:  
— print row[0]
```

Nested Loops (Rows, Records):

```
for row in rows:  
— print "<tr>"  
— for x in [0,1,2]:  
— print "<td>", row[x], "</td>"  
— print "</tr>"
```

MySQLdb

```
import MySQLdb

config = "/home/kenobi01/.my.cnf"
db = MySQLdb.connect(db="starwars",
                     read_default_file=config)

cursor=db.cursor()
query = "select * from persona";
cursor.execute(query);
rows = cursor.fetchall()
print "<table>"
for row in rows:
    print "<tr>"
    for x in [0,1,2]:
        print "<td>", row[x], "</td>"
    print "</tr>"
print "</table>"
```

personas.cgi

```
#!/usr/bin/python
import cgi
cgi.enable()
print "Content-Type: text/html;charset=utf-8"
print

import MySQLdb

config = "/home/kenobi01/.my.cnf"
db = MySQLdb.connect(db="starwars",
                    read_default_file=config)

cursor=db.cursor()
query = "select * from persona";
cursor.execute(query);
rows = cursor.fetchall()
print "<table>"
for row in rows:
    print "<tr>"
    for x in [0,1,2]:
        print "<td>", row[x], "</td>"
    print "</tr>"
print "</table>"
```

Parameters

- additional data from the user

Database

- information that we store

Passing Parameters

Option 1, "GET" (in the URL)

<http://yoda.ischool.utoronto.ca/~kenobio1/cgi-bin/personas.cgi?species=Human>

`.../personas.cgi?species=Human`

Option 2, "POST" (invisibly)

(we'll get back to this)

Accessing Parameters

```
#!/usr/bin/python
import cgi
cgi.enable()
print "Content-Type: text/html;charset=utf-8"
print

import cgi
form = cgi.FieldStorage()
print form["species"].value
```

1. Save in test_form.cgi. (Set permissions.)
2. Test with .../test_form.cgi?species=Human

Handling an Error

Try just "test_form.cgi" (no parameters).
Oops.

```
#!/usr/bin/python
import cgi
cgi.enable()
print "Content-Type: text/html;charset=utf-8"
print

import cgi
form = cgi.FieldStorage()
if form.has_key("species") :
    print form["species"].value
else :
    print "No species specified"
```

Dynamic SQL (Wrong)

```
import cgi
form = cgi.FieldStorage()
if form.has_key("species") :
    cursor=db.cursor()
    species = form["species"].value
    query = ( "select * from persona"
            + " where species="
            + "'" + species + "'" )
    print query
    cursor.execute(query);
    rows = cursor.fetchall()
    print "<table>"
    for row in rows:
        print "<tr>"
        for x in [0,1,2]:
            print "<td>", row[x], "</td>"
        print "</tr>"
    print "</table>"
```


Python Templates

```
TEMPLATE = """
select * from persona
where species='%s';
"""

query = TEMPLATE % species
```

Dynamic SQL (Wrong)

```
TEMPLATE = """select * from persona
where species='%s';"""
import cgi
form = cgi.FieldStorage()
if form.has_key("species") :
    cursor=db.cursor()
    species = form["species"].value
    query = TEMPLATE % species
    print query
    cursor.execute(query);
    rows = cursor.fetchall()
    print "<table>"
    for row in rows:
        print "<tr>"
        for x in [0,1,2]:
            print "<td>", row[x], "</td>"
        print "</tr>"
    print "</table>"
```

Don't do this!

Here Is Why

```
http://yoda.ischool.utoronto.ca/~keno  
bio1/cgi-bin/personas2.cgi?  
species=Human'%3Binsert+into+pers  
ona+(name,species)+values+  
( 'X', 'Human' )  
'%3Bselect+*+from+persona+where+'
```

Dynamic SQL - Right

```
TEMPLATE = """select * from persona
where species='%s';"""
import cgi
form = cgi.FieldStorage()
if form.has_key("species") :
    cursor=db.cursor()
    species = form["species"].value
    query = TEMPLATE % db.escape_string(species)
    print query
    cursor.execute(query);
    rows = cursor.fetchall()
    print "<table>"
    for row in rows:
        print "<tr>"
        for x in [0,1,2]:
            print "<td>", row[x], "</td>"
        print "</tr>"
    print "</table>"
```

HTML Forms

```
<form action="/~kenobio1/cgi-bin/personas3.cgi"
      method="get">
  Please select a species:
  <input type="text" name="species" />
  <input type="submit" />
</form>
```

HTML Forms

```
<form action="/~kenobio1/cgi-bin/personas3.cgi"
      method="get">
  Please select a species:
  <select name="species">
    <option value="Human">Human</option>
    <option value="Ewok">Ewok</option>
    <option value="Wookiee">Wookiee</option>
  </select>
  <input type="submit" />
</form>
```

HTML Forms: POST

```
<form action="/~kenobio1/cgi-bin/personas3.cgi"
      method="post">
  Please select a species:
  <select name="species">
    <option value="Human">Human</option>
    <option value="Ewok">Ewok</option>
    <option value="Wookiee">Wookiee</option>
  </select>
  <input type="submit" />
</form>
```

Questions?