**Project Group 08**

Kristin Swanson, Leonard Anderson, Tony Luo

CS419 Summer 2015

# 

# 

# 

# 

# 

# 

# 

# 

CS419 Final Report: Curse DB

# 

# 

# Table of Contents

[Table of Contents](#h.17jhtqqm96c)

[Introduction](#h.2x3ymci4posh)

[Executive Summary](#h.uw9quhjlt5gv)

[Importance of this Project](#h.1iur40czohbi)

[Client](#h.i7hiickt1ghn)

[Changes Made to Original Design](#h.lumomhly1uvx)

[Change in class structure](#h.1gtnoy5m316y)

[Feature Removal](#h.2l3elqfbu6je)

[Project Documentation](#h.crqcvu78ow3y)

[Curse DB In a Nutshell](#h.1va0zr55irhn)

[Structure](#h.viou6siiw6m0)

[Theory of Operation](#h.yhncxdwaduu)

[Hardware and Software Requirements](#h.3x07m64lwgp9)

[Installation](#h.x7y28n94c2ag)

[Execution](#h.stgsjcpat91q)

[HOWTO](#h.ecqc422ydaka)

[Learning New Technology](#h.eja51p9b0783)

[Helpful Websites](#h.og4jq9yag3qf)

[Helpful Reference Books](#h.t41assjf29kd)

[Campus Resources](#h.85jnv4jfy8if)

[Reflections](#h.bsq8kp37hoe7)

[Tony Luo](#h.27bbaa7idbd7)

[What technical information did you learn?](#h.86cl7er3zk2e)

[What non-technical information did you learn?](#h.v32aws8a0oz9)

[What have you learned about project work?](#h.frr9z656sqgd)

[What have you learned about project management?](#h.lhtm1j7xlkkp)

[What have you learned about working in teams?](#h.30icu2guhxxp)

[If you were to do it all over, what would you do differently?](#h.7kwozn3h5d9f)

[Leonard Anderson](#h.t7qjve4v1crj)

[What technical information did you learn?](#h.o1v991st4tvt)

[What non-technical information did you learn?](#h.n1aw39dnbw3l)

[What have you learned about project work?](#h.ak50tzf9jh9s)

[What have you learned about project management?](#h.mxnuj1bbe1uf)

[What have you learned about working in teams?](#h.tvew0l9ijom2)

[If you were to do it all over, what would you do differently?](#h.pzqv3j49kyvq)

[Kristin Swanson](#h.4s848xkmohz)

[What technical information did you learn?](#h.hivfcjz507nf)

[What non-technical information did you learn?](#h.a2kjy7t6ixnu)

[What have you learned about project work?](#h.50kdwhmrmkdk)

[What have you learned about project management?](#h.kjspmdmei679)

[What have you learned about working in teams?](#h.tswxta8das1q)

[If you were to do it all over, what would you do differently?](#h.rha2lkar2eqo)

[Appendix 1: Essential Code Listings](#h.9vu26vrdt19s)

[Appendix 2: Photos](#h.48krpxawyyhr)

# 

# 

# Introduction

## Executive Summary

Curse DB is an utility that provides an NCurses-based UI for users to interact with their MySQL and/or PostgreSQL databases.

## Importance of this Project

Curse DB fills a need in the database management tooling space. Currently there are utilities that are either web based, have a simple command line interface, or are heavyweight and have an arcane GUI that is not user friendly. Curse DB is designed to fill in the space between the simple command line interface and the heavyweight custom GUI solution.

## Client

Our client for this project is our instructor, Kevin D. McGrath. His role was to clear up any misconceptions that we had about the project requirements, things that we weren’t sure about. For example if our interpretation of the requirements were so different that we as a team could not come to an agreement.

# Changes Made to Original Design

We were not able to implement all of the features that we planned on implementing and had to make slight alterations to our original design.

## Change in class structure

We ended up changing some of the classes after starting to implement them. As we were building this system, we learned that there were simpler ways of organizing our code. We might have been optimizing too quickly. For example, the SaveDatabases and the ParseQuery classes could be condensed into a simpler Database class with some functions from the .

The UI classes (Menu, Inputbox, TabBar, ResultsPane) were all implemented, just with more methods than were mentioned in the original design. For instance, the Menu needed to keep track of which item was selected so that it could be returned at a time other than right after the item was clicked. A class for the Connect to Database screen should have been made, but once this began to be implemented, it was not clear whether there would be enough time to refactor that code into a proper class.

## Feature Removal

Initially, we also wanted to include a feature that would allow the user to search the database for a given keyword. While this sounded like a minor addition at the beginning, we ended up leaving it on the table for too long and not until very late in the game did we realize it actually had quite a bit of meat to it in terms of the implementation. We found that we would need to overhaul the UI for that particular page to provide a list of options to select a table, and then after that to dynamically generate a list of the fields for the chosen table that the user could then run their keyword search against.

In the end, we decided this feature was a bit more than we could handle at that point in time and decided to leave out the feature unless our progress drastically changed prior to the submission deadline. This wasn’t really detrimental to the project itself as the feature was really meant as more of an additional convenience item than anything since it really just took the place of a few successive queries and saved the user some time.

# Project Documentation

## Curse DB In a Nutshell

Curse DB is conceptually separated into three main parts. The UI, Utilities, and the DB. This is analogous the to MVC pattern.

The UI can be thought of as the View, Utilities as the Controller, and DB as the Model. The View displays the current state of the program. It is also what the users are interacting with. When the system receives some kind of input, it will call make a call to the Controller, asking it to process that input. If data is needed or or if a query is being made, the Controller will communicate with the Model to get the required information. The Controller will then process the result and pass it to the View for rendering.

## Structure

We developed our program while keeping utilization and maintenance in mind. That being said, our software is broken into separate folders each containing scripts to handle that handle one of the following: Database functions, Interface functions, Settings and Utilities. The idea behind this was that all files were then located in a logical place within the codebase so that quickly referencing a specific file would be an easy task.

The Database folder’s primary purpose is to house the connect.py file which handled all database connectivity functionality. The Interface folder handles all functionality pertaining to using NCurses to create the UI itself. Next, the Settings folder contains configuration files that are dynamically updated and maintained by the program when connections are created or removed. And finally, our Utilities folder contains files that act as helper functions to make utilizing connect.py a simpler task.

## Theory of Operation



## Hardware and Software Requirements

* Must have python 2.x.x installed
  + Packages required:
    - psycopg2
    - pymysql
* Must have either MySQL and/or PostgreSQL installed
* Program must be ran in a 3rd party terminal like PUTTY or iTerm2
  + The terminal must be able to read mouse inputs

## Installation

In order to install the software, simply perform a Git Pull from our github repository.

Use the following command:

git clone https://github.com/LAnderson8899/CS419-Group8.git

## Execution

From your terminal of choice:

* Navigate to: path/to/curse\_db/interface
* Execute: python nav.py

## HOWTO

To see a series of screenshots, see Appendix 2.

You will start on the Main Menu where there are three options:

[ Connect to Database ]

[ Help ]

[ Quit ]

Notice that they are in brackets; this indicates that they are clickable (this is true throughout this program). Clicking Quit will end the program, clicking Help will bring up some information on using the main menu. Before you can use most of the features, you will need to connect to a database--so click that option.

Two panes will appear when you click Connect to Database. If you have connected to databases before with this program, their names (nicknames) will be displayed in the left-hand pane where you can select one and either connect to it or delete it from the list. The right-hand pane will have 5 input boxes, all of which are required. Input your database information by clicking on each box (pressing Enter to get out of each box). The nickname field is to name this connection for the previous database connection list on the left; it just needs not be the same as one of the previously named connections. Once you have filled out the input fields, there are two connect buttons: [sql connect] and [psql connect]. If your database is mySQL, click the sql button; if it is postgreSQL, click the second psql button.

If there was something wrong with you input, a message will appear near the bottom of the screen; if the database connection succeeded, then the other three tabs should appear at the top of the screen and the Connect to Database screen will disappear.

If you click on the Tables tab, you will see a screen with two panes. On the left, you can see the list of all the table names in this database. These names are clickable, and the one selected will have its information (field names and types) shown in the right-hand pane.

If you click on the Query tab, you will see a textbox where you can enter queries. Click on this box (which will expand) and type your query then press ctrl-g. (Note that to click on a tab you will also need to g-ctrl out of the textbox before your clicks will register outside of it). Once you submit your query with ctrl-g, the results will be displayed below. If there are too many to fit on one screen, a [Next->] button will appear in the bottom-right corner of the screen.

Finally, the Help tab provides help on each of the other tabs. Again, you can click on the bracketed names to select which help topic you would like to see.

# Learning New Technology

## Helpful Websites

* <https://docs.python.org/2/>
  + <https://docs.python.org/2/tutorial/index.html>
  + <https://docs.python.org/2/howto/curses.html>
* <http://stackoverflow.com/>
* <https://wiki.postgresql.org/wiki/Psycopg2_Tutorial>
* <https://dev.mysql.com/doc/refman/5.0/en/index.html>

## Helpful Reference Books

We were able to get all of our information from online resources. The Python has good documentation and we primarily used that as a language reference. Reference books were not necessary for this project.

## Campus Resources

We used the flip server as our development environment. We were able to handle most of the problems we encountered ourselves and did not seek an on-campus tutor.

# Reflections

## Tony Luo

### What technical information did you learn?

In this project I became more familiar with the Python programming language. I got to use some of the more advance features such as list comprehensions and lambda functions, which was very cool. I was able to save a lot of time by not having to write loops and to fiddle around with indexes. I also felt this style of programming is more expressive and clean. I think I will start to write more often in a functional style on a day to day basis.

Because we did not have the permissions on the FLIP servers, I volunteered to host, setup, and administer a VPS for our team so we can experiment, test, and use PostgreSQL.

### What non-technical information did you learn?

This project provided me an opportunity to practice using the OOP programming paradigm. I have been mostly programming in JavaScript these days OOP is not used very much. I forget how clean and logical OOP is.

Because I was mainly working on the controller functions of our application, I had a chance to work on creating an API that would be used by my teammates. It was a lot of fun when people are using your code and to talk about the code. Here I learned more about encapsulation.

### What have you learned about project work?

We are no longer working in an isolated environment. The code that you change could end up breaking the application. The code that you write will be used by other people. I learned to be more mindful about that.

It is important to write clean code and have it well documented so that your team will be able to understand what you are working on without having to spend a lot of energy and effort trying to figure out what you are doing.

One thing that would have made this process a lot easier is if we had a good test suite and a good number of unit tests. We had a bunch of manual tests and it took a lot of time to see if any of our changes broke anything because we would have to run through the manual tests over again.

### What have you learned about project management?

Organization matters, a lot. I think it would have been easier if we spent a little more time to flush out ideas, e.g., design for the edge cases and handling errors. We mostly talked about and work on the average use case, and this ended up biting us in the back.

### What have you learned about working in teams?

It is important that everyone understands what they are supposed to be doing and the role that they play. It was a little shaky in the beginning, but as time passed, people began to naturally pick something and start building from there. It was not explicit and I think our experience would have been more pleasant if it was.

Something else that we should have done was have a style guide that we would follow. Not just syntax, but also design, e.g., procedural or OO? What kind of parameters should we taking and what will our methods be returning? This would have improved our consistency.

### If you were to do it all over, what would you do differently?

If I were to do this project again from the beginning. I would have us spend more time on the organization and planning. By that I mean, setting up style guide. I would also have spent more time testing.

## Leonard Anderson

### What technical information did you learn?

I became a lot more comfortable with the Python programming language and using it to interact with a MySQL database. I also learned how to work with new Python libraries that I wasn’t accustomed to and about how to work around import errors when using python in tandem with various file path structures.

### What non-technical information did you learn?

I had never really made a recorded presentation with a group spread so far apart, so it was interesting to explore the media options available and to make those choices and develop the presentation as a group.

### What have you learned about project work?

It’s really import to set a timeline and try as hard as you can to stick to this goal. Sometimes it may seem tempting to push a particular goal in a week into the coming sprint’s goals, but this just ends up pre-emptively bogging you down for the next week. I suppose the biggest take-away here is that time and resource management is key to the project’s success.

### What have you learned about project management?

Be very thorough in your evaluation process at the beginning of the project when you’re determining how to break up the work in an even manner. I often felt that the portion I had ended up with was, at times, underwhelming compared to the workloads of other members of the group.

### What have you learned about working in teams?

It’s important to keep in mind the requirements that others are working around when developing in a team environment. For example, we ran into an issue wherein another member of the group was developing functionality that would work with my code and that of my other partner’s code. Their code was assuming (as it should have), that my code and my partner’s code would both take in the same parameters, but unfortunately I had deviated and mine ended up using additional arguments. The take-away for me here was to always keep the big picture in mind and don’t forget about the code of other’s that you’re working with. At the end of the day, it all has to work together.

### If you were to do it all over, what would you do differently?

I would probably want to be more aggressive with my own time management of my deliverables. I let myself get bogged down on several occasions and that type of issue is detrimental both to my own work as well as to that of the team. Specifically, I wish I had budgeted more time to include the Search functionality. I think it would have been a really neat feature to showcase and would have given me a lot of practice in complex query interactions.

## Kristin Swanson

### What technical information did you learn?

A fair amount about how to use the Python programming language. Specifically, how to use Curses.

### What non-technical information did you learn?

I got a lot of practice using Github as that is what we were using for code integration.

The video demo was also interesting; listening to our recording, I realized that I say “um” a lot when doing presentations--I should work on that.

### What have you learned about project work?

It is far easier to do a little planning for making something into a class than it is to later go back and make a class out of a messy chunk of code. For many of the UI elements (input boxes, tab bar, menus), we decided that we should have classes. It did not take that long to make these classes, and they were quite convenient to use. However, the Connect to Database screen did not obviously lend itself as a class--certainly I was not thinking about doing it that way--so I just started working on it in my main script. But once I got a fair way into it, the code got very messy and interlocked, and I feared messing it up or running out of time. So I never got to refactoring it.

### 

### What have you learned about project management?

Project management has an extra layer of complexity when working in teams. You really have to plan ahead of time to make sure all the work gets done. I had a firm idea of what I needed to work on, but I was not always sure what my teammates were working on. Toward the end of the project, it seemed like there were several things that no one was working on!

### What have you learned about working in teams?

If you will be using other people’s functions, It is helpful to specify early on what you want those functions to take as parameters and what they return. I worked on the interface and needed my teammates back-end functions to hook into my code. But when my teammate told me what some of the functions needed (like an index), I had to tell him that all I had was a name string, so he had to update the function for me.

### If you were to do it all over, what would you do differently?

I would make sure everyone was clear about who is working on what. It would also be good to make sure everyone has a reasonable timeline (start early enough) for the tasks they are to work on. I would also want to design not just the classes, but also the functions calls that will be used in code that someone else wrote.

Specifically in my code, I would plan out encapsulation for complex screens, in this case our “Connect to Database” screen. My code got very messy; half the handling was going on in my main script and the other half in a large function in another file. I really should have made a class for it--but we did not include it in our design document as one of the classes we needed. The other problem was that I was working on it toward the end, but in retrospect, it was one of the more complex areas of our program so I should have started on it sooner.

# 

# 

# Appendix 1: Essential Code Listings

Main loop (in nav.py) once a database connection has been established:

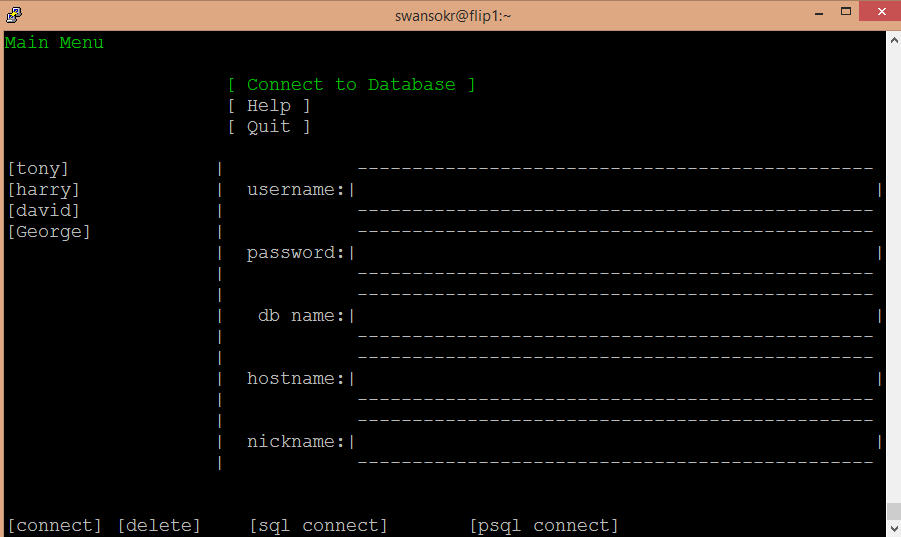
|  |  |
| --- | --- |
|  |  |
|  | while not quit: |
|  | if switchTab == True: |
|  | switchTab = False |
|  | scr.clear() |
|  | scr.refresh() |
|  | tabs.selectOnlyTab(inTab) |
|  | mainPane.reset() |
|  | resultsPane.reset() |
|  | hide([mainMenu, tableMenu, inputBox]) |
|  | if inTab == "Main Menu": |
|  | mainMenu.unhide() |
|  | elif inTab == "Tables": |
|  | tableNames = db.getTables() |
|  | tableMenu.setItems(tableNames) |
|  | tableMenu.unhide() |
|  | if len(tableNames) > 0: |
|  | tableMenu.selectItem(tableNames[0]) |
|  | mainPane.setResults(db.getTableInfo(tableMenu.getSelected())) |
|  | mainPane.showResults(mainPane.getPageNum()) |
|  |  |
|  | elif inTab == "Query": |
|  | inputBox.unhide() |
|  | elif inTab == "Search": |
|  | inputBox.unhide() |
|  | elif inTab == "Help": |
|  | for b in helpButtons: |
|  | b.unhide() |
|  | b.selectOnlyItem(helpTab) |
|  | helpWin.addstr(0, 0, getHelp(helpTab)) |
|  | helpWin.refresh() |
|  |  |
|  | # Be ready to capture a mouse click |
|  | if mouseClick(): |
|  | (mid, x, y, z, s) = curses.getmouse() |
|  |  |
|  | # if the click in the tab bar at the top of the screen |
|  | if y == 0: |
|  | # select the tab at the x-position of the click |
|  | newTab = tabs.selectTabAt(x) |
|  | if newTab and newTab != inTab: |
|  | inTab = newTab |
|  | switchTab = True |
|  |  |
|  | # Click anywhere on Main Menu page |
|  | elif inTab == "Main Menu": |
|  | response = mainMenuClick(x, y, mainMenu, mainMenuNames, databaseInputBoxes, prevDatabasesMenu, prevConnect, prevDelete, newsqlConnect, newpsqlConnect, helpWin) |
|  | if response == "quit": |
|  | quit = True |
|  | elif response == "failedConnect": |
|  | scr.addstr(curses.LINES - 2, databaseInputBoxesX, "Must provide all 5 fields.") |
|  | scr.refresh() |
|  | elif response == "failedOldConnect": |
|  | scr.addstr(curses.LINES - 2, 0, "Problem connecting to database.") |
|  | scr.refresh() |
|  | elif isinstance(response, str): |
|  | # clear failed connect space |
|  | scr.addstr(curses.LINES - 2, 0, " "); |
|  | scr.addstr(6, 0, response) |
|  | scr.refresh() |
|  | elif response: |
|  | db = response |
|  | connected = True |
|  |  |
|  | # Click in Tables menu |
|  | elif inTab == "Tables" and tableMenu.itemAt(y, x): |
|  | tableMenu.selectOnlyItem(tableMenu.itemAt(y, x)) |
|  |  |
|  | # if the click was in the Query or Search input box |
|  | if not inputBox.isHidden() and (y >= inputY and y <= inputYmax and x >= inputX): |
|  | # Get whatever the user enters into the textbox |
|  | queryInput = inputBox.edit() |
|  |  |
|  | #Display "results" |
|  | resultsPane.reset() |
|  | queryResults = db.executeQuery(queryInput) |
|  | if queryResults != False: |
|  | resultsPane.setResults(queryResults) |
|  | else: |
|  | resultsPane.setResults(["Error in query"]) |
|  | resultsPane.showResults(resultsPane.getPageNum()) |
|  |  |
|  | # Make the textbox clear when next clicked on |
|  | inputBox.clear() |
|  |  |
|  | # If the click is at the bottom of the Query or Seach screen |
|  | elif y == curses.LINES - 1 and (inTab == "Query" or inTab == "Search"): |
|  | if resultsPane.atNext(x): |
|  | resultsPane.showResults(resultsPane.getPageNum() + 1) |
|  | elif resultsPane.atPrev(x): |
|  | resultsPane.showResults(resultsPane.getPageNum() - 1) |
|  | elif inTab == "Help": |
|  | for b in helpButtons: |
|  | if b.itemAt(y, x): |
|  | helpTab = b.itemAt(y, x) |
|  | switchTab = True |
|  |  |

# Appendix 2: Photos

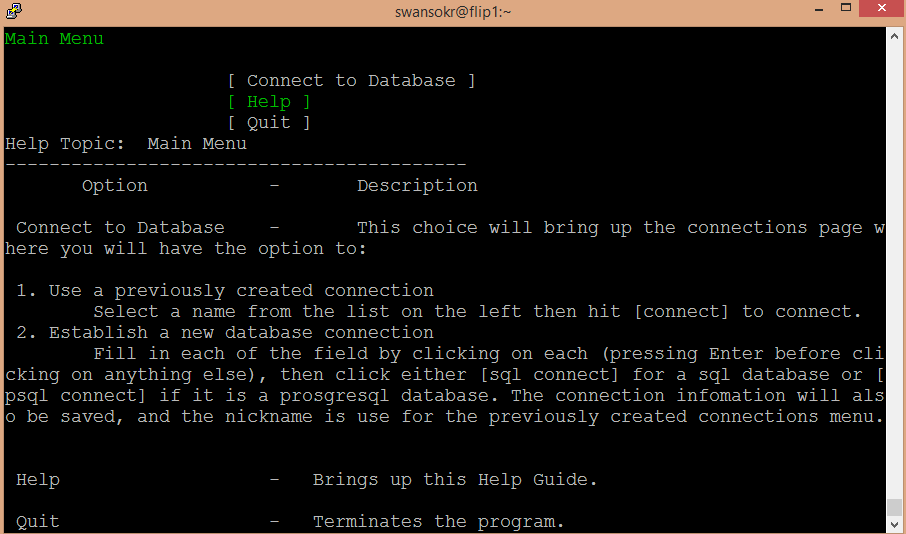
First screen the user sees. There are no other tabs until a database connection is established.



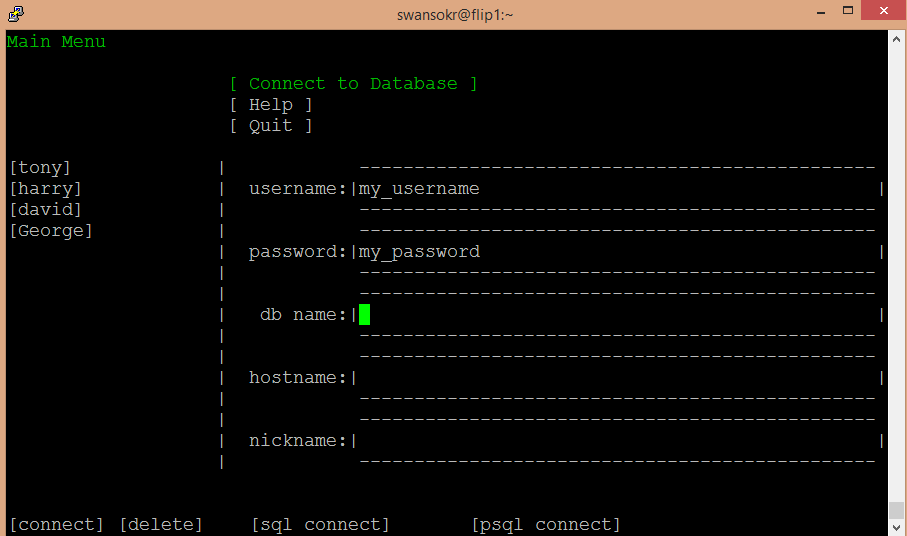
Connect to Database screen:



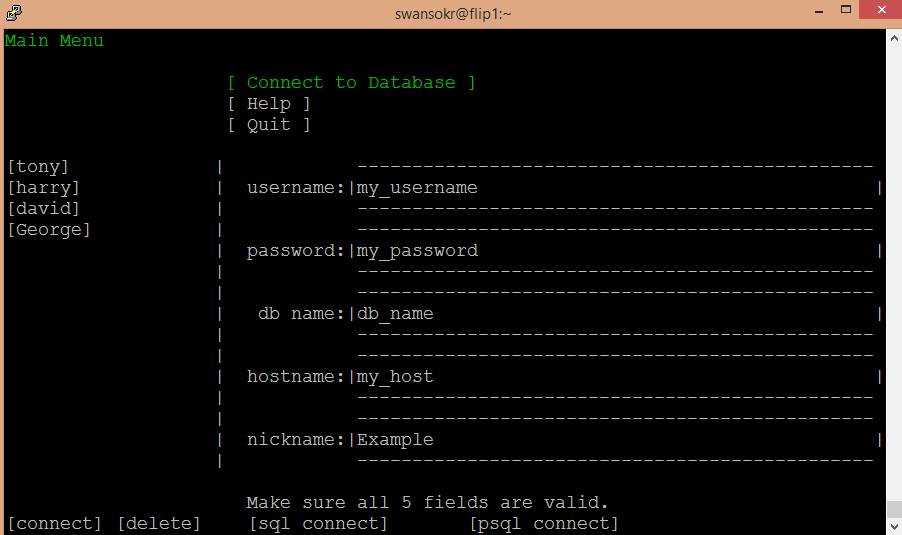
Main Menu help:



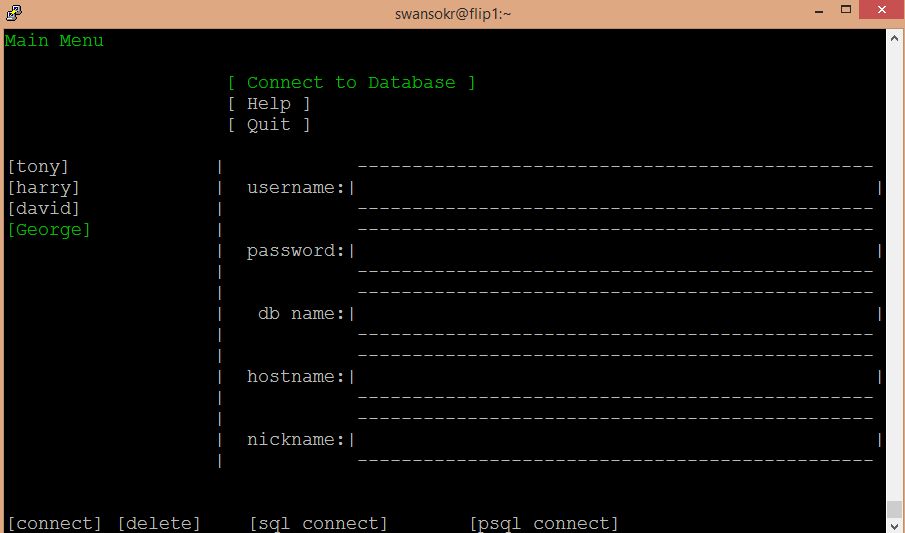
Inputting information for a new connection:



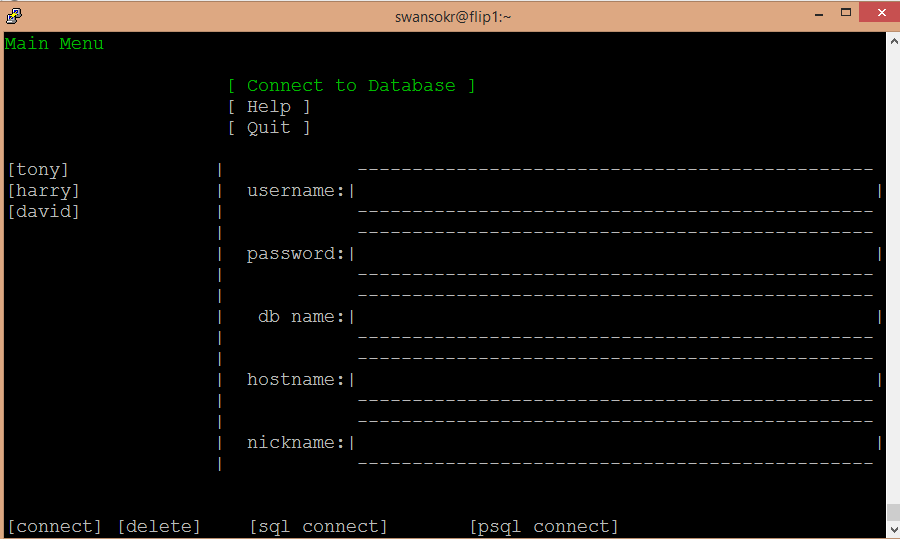
Message for bad connection input (information was not saved as a previous database connection):



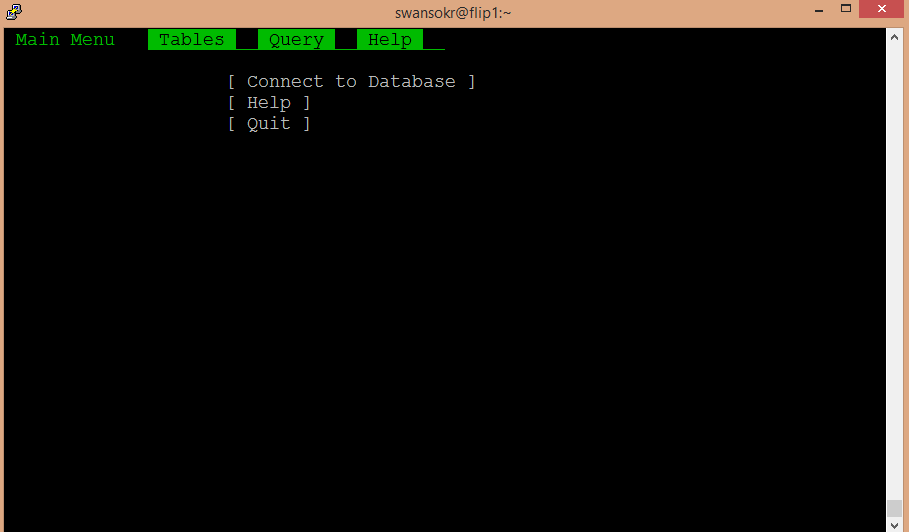
Selecting a previous database:



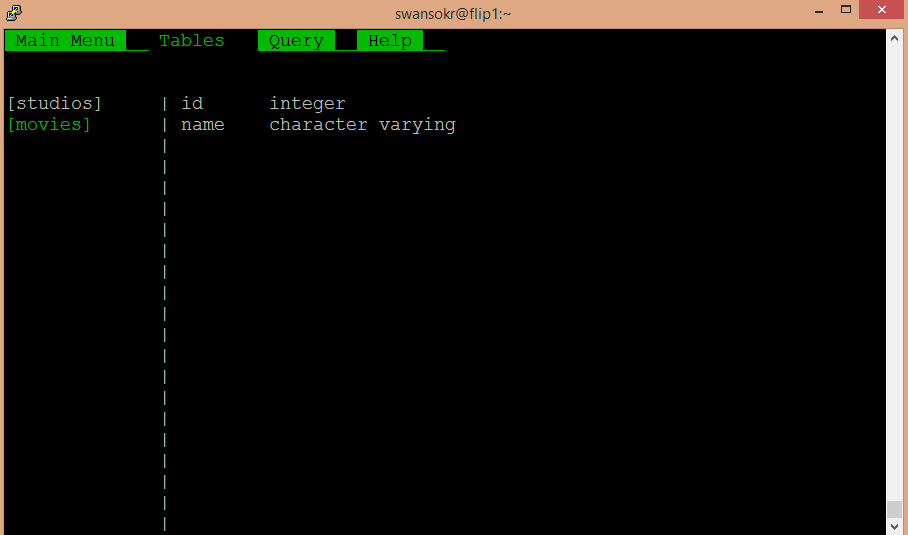
After clicking [delete] with [George] selected (George is removed from the previously used database connection list):



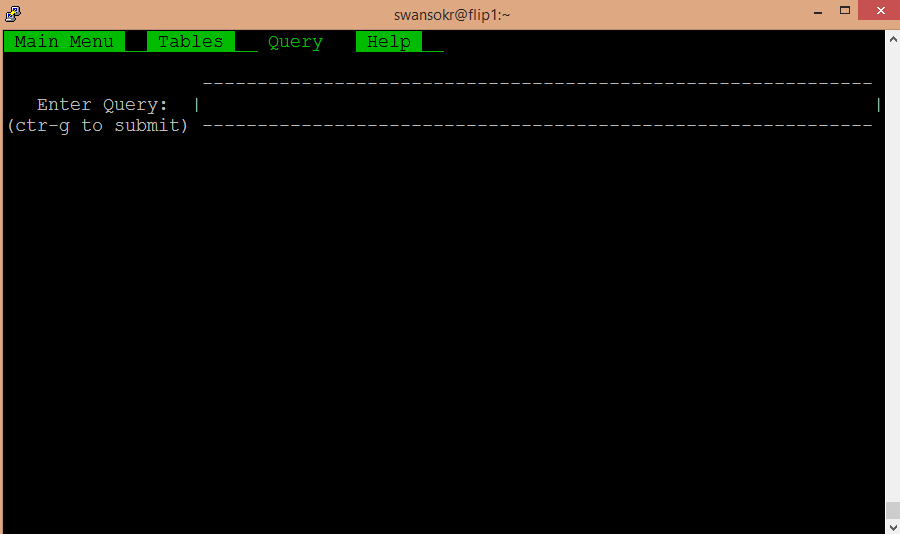
After selecting [Tony] and clicking [connect], a connection is established so the other tabs are visible:



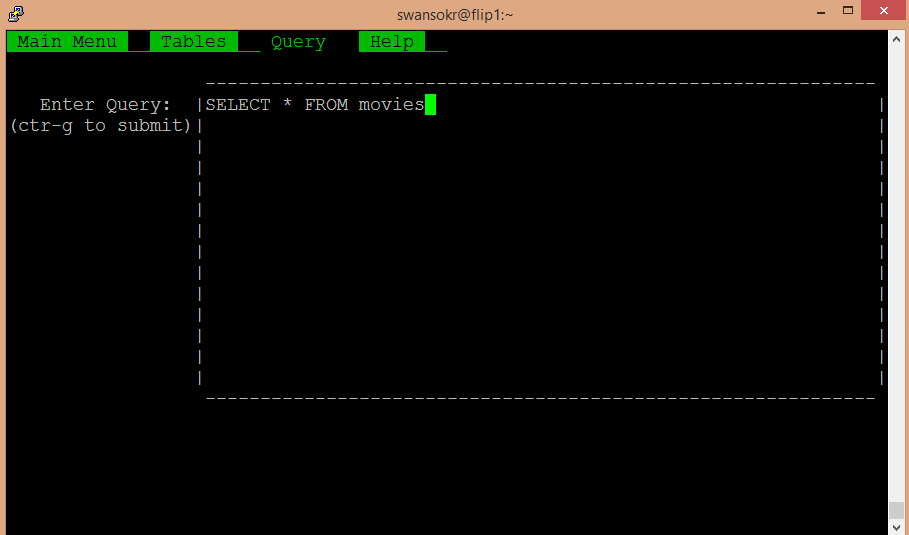
Tables tab with the movies table selected:



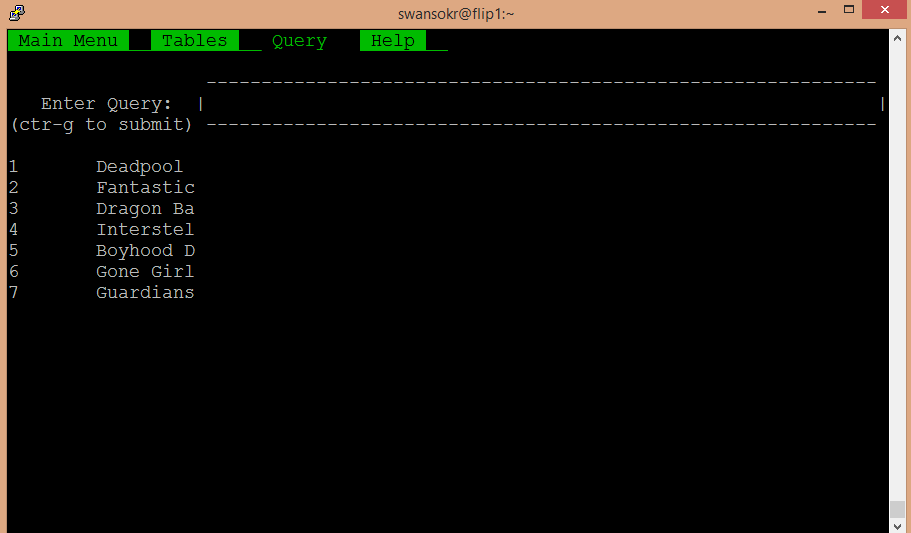
Query tab before selecting the query textbox:



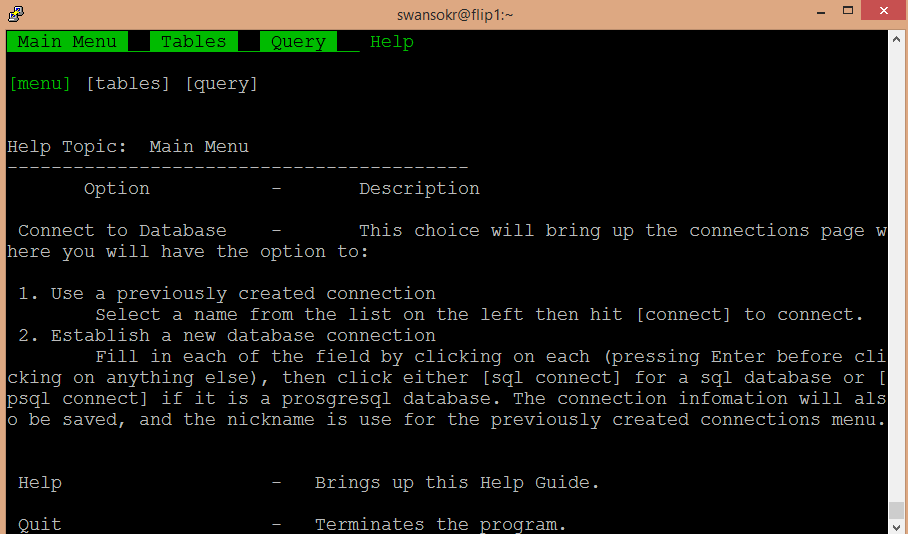
After clicking on the query textbox and typing a query in the textbox:



The results are shown after submitting a valid query (and the textbox shrink back to its original size):



Help tab. When the user first clicks it, it selects the Main Menu help:



After clicking on [tables] to get help for the Tables tab:



And finally, clicking on [query] to get help on the Query tab:

