COMP260 – Distributed programming Worksheet 8

Data Persistence with SQL

Introduction

The goal of this worksheet is to explore data persistence in the SUD you developed at the beginning of the course as a starting for assignment 2. To support this, there is a refresher on SQL with Python's SQL functionality

Python SQL 101

As you may remember from COMP130, Python supports sqlite through the sqlite3 library, which can be directly imported into your applications with the statement `import sqlite3`. To get or refresh your understanding of sqlite, build a simple Python application that will let you explore the lifecycle of data through CRUDS (create, retrieve, update, delete and search), by making a simple telephone book application. The app will require names and phone numbers to be added to a database which can then be searched for, updated and deleted.

Like the original SUD application, use command line processing to determine what the user wants to do and process their requests accordingly. https://www.pythoncentral.io/introduction-to-sqlite-in-python/ will be helpful for this.

To create a new address book, use the following SQL and wrap it into an appropriate sqlite command:

```
create table table_phonenumbers (name varchar(20), number varchar(20) )
```

To insert a new entry:

```
insert into table_phonenumbers (name, number) values(?,?)
```

SQL SUD

Returning to your earlier SUD work, determine a suitable SQL table for managing the dungeon as a table of rooms. The room record will need to contain all the attributes from the room class (description, exits and so on), these should be stored as varchars. Then refactor the dungeon accessing code so that the table of rooms is used to control your character around the dungeon.

Persistent SUD

Add a `table_dungeoneer` that will store your player and the room they are currently in, rather than using the data in the dungeon. Again, as your character moves around the dungeon, the table should be updated. At this point, stopping and restarting the SUD application should result in your character remaining in the correct room on restart – persistence has been achieved.

Next Steps

Apply this to your MUD application, remembering that sockets are not persistent (by design) and should be stored in a dictionary rather than in sqlite. You should now have a persistent MUD.

To add objects to your MUD, remember that each item will need a unique identifier so that it can be selected from the `table_objects`. Rather than store a player's possessions in a list or an array within `table_dungeoneer`, it will make sense to give objects an owner in their table.