

## 1 Preliminaries

Before starting on this assignment, please be sure to read the General Instructions that are on Piazza (under Resources->General Resources). If you did Lab1, you should already know how to log in to the class PostgreSQL server. You'll get help on Lab2 in your Lab Section, not the Lectures, so *be sure to attend Lab Sections*.

## 2 Goal

The goal of the second assignment is to create a PostgreSQL data schema with 7 Adventure tables that are very similar to the tables that you created in Lab1. The tables have the same names, attributes and data types as the tables of Lab1, and the same Primary Keys and Foreign Keys, but which has some additional UNIQUE constraints and restrictions on NULL that are described below.

After you create the data schema with the 7 tables, you must write five SQL statements that use those tables. Under Resources→Lab2, we have provided you with data that you can load into your tables so that you can test the results of your queries. Testing can prove that a query is wrong, but not that it is right, so be careful. We will not give you with the results of these queries on the load data; you should be able to figure out the results on that data yourselves. You can also test your queries on your own data. In the “real world”, you have to make and check your own tests.

Lab2 is due in two weeks, so you will have an opportunity to discuss the assignment during the Discussion Section in the first week of the assignment, and to discuss issues you have had in writing a solution to the assignment during the Discussion Section of the second week. Instructions for submitting the assignment appear at the end of this document.

### 3 Lab2 Description

#### 3.1 Create PostgreSQL Schema Lab2

You will create a Lab2 schema to set apart the database tables created in this lab from the tables you will create in future, as well as from tables (and other objects) in the default (public) schema. We'll refer to the Adventure database throughout the Lab Assignments, but your schemas will be called Lab1, Lab2, etc. In PostgreSQL, a database can have more than one schema; see [here](#) for more details on PostgreSQL schemas. You create the Lab2 schema as follows:

```
CREATE SCHEMA Lab2;
```

Now that you have created the schema, you want to set Lab2 to be your default schema when you use psql. If you do not set Lab2 as the default schema, then you will have to qualify your table names with the schema name (e.g., Lab2.Authors). To set the default schema, you modify your search path. (For more details, see [here](#).)

```
ALTER ROLE username SET SEARCH_PATH to Lab2;
```

You will need to log out and log back in to the server for this default schema change to take effect. (Students often forget to do this.)

You do not have to include the CREATE SCHEMA or ALTER ROLE statements in your solution.

#### 3.2 Create tables

You will create tables in schema Lab2 for Members, Rooms, Roles, Characters, Things, Monsters and Battles. The attributes for these 7 tables are the same as for the tables of Lab1. Moreover, the data types for the attributes in these tables are the same as the ones specified for the tables of Lab1, and the Primary Keys and other Foreign Keys are also the same. You may use the *create\_lab1.sql* file solution that we provided on Piazza for Lab2 as the basis for the *create\_lab2.sql* file that you include in your Lab2 solution (although you don't have to include our comments). However, the tables must have additional constraints, which are described in the next section.

##### 3.2.1 Constraints

The following attributes cannot be NULL. All other attributes can be NULL ... but remember that attributes in Primary Keys also cannot be NULL.

- In Rooms: roomDescription
- In Things: cost
- In Characters: roomID

Also, the following must be unique for the specified table. That is, there cannot be identical rows in that table that have exactly the same (non-NULL) values for all of those attributes (composite unique constraint).

- In Characters: the attribute name
- In Members: the 2 attributes name and address
- In Monsters: the 2 attributes monsterKind and name

For example, the first constraint says that there can't be two rows in Characters that have the same non-NULL name (although there could be two rows in Characters in which name is NULL). And the second constraint says that there can't be two rows in Members that have the same values for name and address (if both of those attributes are not NULL). Think of this as saying that there can't be two members who have both the same name and the same address (if both of those attributes are not NULL).

You will write a CREATE TABLE command for each of the 7 tables that has these additional constraints. Save the commands in the file *create\_lab2.sql*

## 4 SQL Queries

Below are English descriptions of the five SQL queries that you need to write for this assignment, which you will include in files queryX.sql, where X is the number of the query, e.g., your SQL statement for Query 1 will be in the file query1.sql, and so forth. Follow the directions as given. You will lose points if you give extra tuples or attributes in your results, if you give attributes in with the wrong names or in the wrong order, or if you have missing or wrong results. You will also lose points if your queries are unnecessarily complex, even if they are correct. Grading is based on correctness of queries on all data, not just the load data that we have provided.

Remember the Referential Integrity constraints from Lab1, which you should retain for Lab2. For example, if an initialRoomID appears in a Things tuple, then there must be a tuple in Rooms that has that initialRoomID value as its roomID.

Attributes should have their original names in the results of your queries, unless an alias is requested. And if a query asks that several attributes appear in the result, the first attribute mentioned should appear first, the second attribute mentioned should appear second, etc.

### 4.1 Query 1

A thing is a scroll if its thingKind is 'sc'. Find all the rooms and scrolls where the room is the scroll's initial room (initialRoomID), that initial room's description has the lowercase letter 'w' in its second position, and the cost of the scroll is less than 12. The attributes in your result should be roomID, initialRoomDescription, thingID and cost.

The tuples in your result should appear in alphabetical order based on room description. Two result tuples that have the same room description should appear in decreasing order of cost. No duplicates should appear in your result.

### 4.2 Query 2

Find every monster whose wasDefeated value is FALSE, but who was in a battle in which their monsterBattlePoints was less than the characterBattlePoints of the character they fought in that battle. The attribute in your result should be the name of the monster, which should appear as monsterName.

No duplicates should appear in your result.

### 4.3 Query 3

A member may have more than one character, as long as those character roles are different. Some characters may own things; some characters may not own any things.

Find the name and joinDate of every member whose joinDate isn't NULL and who has no character that owns a thing. (Yes, a member who has no characters should appear in your answer.) The attributes in your result should appear as memberName and memberJoinDate.

No duplicates should appear in your result.

#### 4.4 Query 4

A monster is a basilisk if its monsterKind is 'ba'. A character is a knight if its role is 'knight'. Find all the battles in which:

- the monster is a basilisk,
- the character is not a knight,
- the member associated with the character has an expiration date that is on or after December 10, 2022, and
- the current money that the character has is more than the initial money associated with that character's role.

The attributes which should appear in your result are the name of the monster in the battle, the name of the character in the battle, and the role of that character. These attributes should appear in your result as theMonster, theCharacter and theRole.

No duplicates should appear in your result.

#### 4.5 Query 5

We'll say that two rooms are east-west adjacent if the first is east of the second, and the second is west of the first. That is, the first room is the eastRoom of the second, and the second Room is the westRoom of the first. (Room directions in Adventure might not be logical; even though the first is east of the second, the second might not be west of the first.)

Find each pair of rooms that are east-west adjacent, output the roomID of the first, the room description of the first, the roomID of the second and the room description of the second. The attributes in your result should appear as eastRoomID, eastRoomDescription, westRoomID and westRoomDescription.

No duplicates should appear in your result.

### 5 Testing

While your solution is still a work in progress, it is a good idea to drop all objects from the database every time you run the script, so you can start fresh. Of course, dropping each object may be tedious, and sometimes there may be a particular order in which objects must be dropped. The following commands (which you can put at the top of *create\_lab2.sql* if you want, but you don't have to), will drop your Lab2 schema (and all objects within it), and then create the (empty) schema again:

```
DROP SCHEMA Lab2 CASCADE;  
CREATE SCHEMA Lab2;
```

Before you submit, login to your database via psql and execute your script. As you've learned already, the command to execute a script is: \i <filename>.

Under Resources→Lab2 on Piazza, we soon will provide a load script named *load\_lab2.sql* that loads data into the 6 tables of the database. You can execute that script with the command:

```
\i lab2_load_data.sql
```

You can test your 5 queries using that data, but you will have to figure out on your own whether your query results are correct. We won't provide answers, and students should not share answers with other students. Also, your queries must be correct on any database instance, not just on the data that we provide. You may want to test your SQL statements on your own data as well.

## 6 Submitting

1. Save your scripts for table creations and query statements as *create\_lab2.sql* and *query1.sql* through *query5.sql*. You may add informative comments inside your scripts if you want (the server interprets lines that start with two hyphens as comment lines).
2. Zip the file(s) to a single file with name Lab2\_XXXXXXX.zip where XXXXXXX is your 7-digit student ID. For example, if a student's ID is 1234567, then the file that this student submits for Lab2 should be named Lab2\_1234567.zip

To generate the zip file you can use the Unix command:

```
zip Lab2_1234567 create_lab2.sql query1.sql query2.sql query3.sql query4.sql query5.sql
```

(Of course, you use your own student ID, not 1234567.)

3. You should already know how to transfer the files from the UNIX timeshare to your local machine before submitting to Canvas. If you are still not familiar with the process, use the instructions we provided at the Lab1 assignment.
4. Lab2 is due by 11:59pm on Sunday, January 30. Late submissions will not be accepted, and there will be no make-up Lab assignments.
5. You can always get rid of duplicates by using DISTINCT in your SELECT. In CSE 180, we deduct points if students use DISTINCT and it wasn't necessary because even without DISTINCT, there couldn't be duplicates. We will also deduct if you were told not to eliminate duplicates but you did.
6. Be sure to follow directions about Academic Integrity that are in the Syllabus. If you have any questions about those directions, please speak to the instructor as soon as possible.