

1 Preliminaries

In this lab, you will work with a database schema similar to the schema that you used in Lab2. We've provided a lab3_create.sql script for you to use, so that everyone can start from the same place. Please remember to DROP and CREATE the lab3 schema before running that script (as you did in previous labs), and also execute:

```
ALTER ROLE yourlogin SET SEARCH_PATH TO lab3;
```

so that you'll always be using the lab3 schema without having to mention it whenever you refer to a table. We've also provided a lab3_data_loading.sql script that will load data into your tables. You'll need to run that before executing Lab3.

You will be required to combine new data (as explained below) into one of the tables. You will also need to add some new constraints to the database and do some unit testing to see that the constraints are followed. You will also create and query a view, and create an index.

New goals for Lab3:

1. Perform SQL to "combine data" from two tables
2. Add foreign key constraints
3. Add general constraints
4. Write unit tests for constraints
5. Create and query a view
6. Create an index

There are lots of parts in this assignment, but only a few of them should be difficult. Lab3 will be discussed in the Lab Sections during the 3 weeks before its due date, Sunday, May 20. You have an extra week to do this Lab because of the Midterm, which is on Wednesday, May 9.

2. Description

2.1 Tables with Primary Keys for Lab3

The primary key for each table is underlined.

```
ChirpUsers(userID, userPassword, userName, joinDate, address, education, income, spouseID, active)
```

```
ChirpPosts(posterID, postNum, thePost, censored, postDate)
```

```
ChirpFollowers(userID, followerID, followStartDate)
```

```
ChirpReads(posterID, postNum, postReader, timesRead, latestReadDate)
```

```
NewReads(posterID, postNum, postReader, readDate)
```

In the `lab3_create.sql` file that we've provided under Resources→Lab3, the first 4 tables are the same as they were in our Lab2 solution, including NULL and UNIQUE constraints. Note that there is an additional table, `NewReads` that has the same key as `ChirpReads`. As the table name suggests, each of its tuples records a new time that a post (which is identified by `posterID` and `postNum`) has been read by a user (who is identified by `postReader`). We say more about `NewReads` (which is only used in Section 2.2) below.

As in previous assignments, to avoid mentioning the schema every time you refer to the tables, you can make **Lab3** the default schema in the search path by issuing the following command:

```
ALTER ROLE your_user_id SET SEARCH_PATH TO Lab3;
```

In practice, primary keys and unique constraints are almost always entered when tables are created, not added later, and `lab3_create.sql` handles those constraints for you. However, we will be adding some additional constraints to these tables, as described below.

Under Resources→Lab3, you've also been given a load script named `lab3_data_loading.sql` that loads tuples into the tables of the schema. You must run both `lab3_create.sql` and `lab3_data_loading.sql` before you run the parts of Lab3 that are described below.

2.2 Combine Data

Write a file, *combine.sql* (which may have multiple sql statements in it in a serializable transaction) that will do the following. For each “new read” tuple *t* that’s in NewReads, either a) there isn’t a tuple in ChirpReads that has the same primary key, or b) there is a tuple in ChirpReads with the same primary key.

- a) If there isn’t already a tuple in ChirpReads that has the same primary key, then insert a tuple into the ChirpReads table corresponding to that tuple in the NewReads table, with timesRead set to 1 and latestReadDate set to the readDate in the NewReads tuple.
- b) If there is already a tuple in ChirpReads that has the same primary key, then update ChirpReads, adding 1 to timesRead and setting latestReadDate to be the readDate in the NewReads tuple. (We’re assuming that NewReads always have a later value for readDate.)

Your transaction may have multiple statements in it. The SQL constructs that we’ve already discussed in class are sufficient for you to do this part (which is probably the hardest part of Lab3). A helpful hint is provided in the initial Lab3 announcement posted on Piazza.

2.3 Add Foreign Key Constraints

Important: Before running Sections 2.3, 2.4 and 2.5, recreate the Lab3 schema using the *lab3_create.sql* script, and load the data using the script *lab3_data_loading.sql*. That way, any database changes that you’ve done for Combine won’t propagate to these other parts of Lab3.

Here’s a description of the Foreign Keys that you need to add for this assignment. The default for referential integrity should be used in all cases. The data that you’re provided with should not cause any errors.

- a) The posterID field in ChirpPosts should reference the userID primary key in ChirpUsers.
- b) The userD field in ChirpFollowers should reference the userID primary key in ChirpUsers.
- c) The followerID field in ChirpFollowers should reference the userID primary key in ChirpUsers.

Write commands to add foreign key constraints in the same order that the foreign keys are described above. Save your commands to the file *foreign.sql*

Note: We also could have requested that the (posterID, postNum) fields in ChirpReads reference the (posterID, postNum) primary key in ChirpPosts. However, we’re not asking you to do that in Lab3, so please don’t do it. But our solution will show you how it could have been done.

2.4 Add General Constraints

General constraints are:

1. In ChirpReads, timesRead must be positive. Please give a name to this constraint when you create it. We recommend that you use the name `positive_reads`, but you may use another name. (The other constraints don't need to have names.)
2. In ChirpUsers, userID and spouseID must not be the same.
3. In ChirpUsers, if joinDate is NULL, then active must also be NULL.

Write commands to add general constraints in the same order that the constraints are described above, and save your commands to the file *general.sql*. (Reminder: UNKNOWN for a Check constraint is okay.)

2.5 Write unit tests

Unit tests are important for verifying that your constraints are working as you expect. We will write just a few for the common cases, but there are many more possible tests we could write.

For each of the 3 foreign key constraints specified in section 2.3, write one unit test:

- An INSERT command that violates the foreign key constraint (and elicits an error).

Also, for each of the 3 general constraints, write 2 unit tests:

- An UPDATE command that meets the constraint.
- An UPDATE command that violates the constraint (and elicits an error).

Save these $3 + 6 = 9$ unit tests, in the order given above, in the file *unittests.sql*.

2.6 Working with a view

2.6.1 Create a view

Create a view named `ManyFollowers`. For each user in `ChirpUsers`, this view should provide `userID` and the number of followers that the user has. Don't count followers if their name is 'Voldemort' or 'Malfoy'. In your result, the second attribute should be called `numberOfFollowers`. But only include a tuple for a user if that user has at least 3 followers (who aren't named 'Voldemort' or 'Malfoy').

Save the script for creating this view in a file called *createview.sql*.

2.6.2 Query a view

If there's a tuple for a `userID` in `ManyFollowers`, then we'll say that the user has many followers. Write a query over the `ManyFollowers` view (and the `ChirpPosts` table) to execute the following "Post Statistics" query:

For users that have many followers, output that user's `userID`, the date of that user's earliest post, the date of that user's latest post, and that user's `numberOfFollowers` (as it appears in `ManyFollowers`). The attributes in your result should be called `userID`, `earliestPostDate`, `latestPostDate` and `numberOfFollowers`.

Important: Before running this query, recreate the Lab3 schema using the *lab3_create.sql* script, and load the data using the script *lab3_data_loading.sql*. That way, any changes that you've done for previous parts of Lab3 (e.g., Unit Test) won't affect the result of this query. Then write the results of that query in a comment.

Next, write commands that delete just the tuples with the following primary keys from the `ChirpFollowers` table:

(120, 105)

(103, 111)

Run the "Post Statistics" query once again after those deletions. Write the output of the query in a second comment. Do you get a different answer?

You need to submit a script named *queryview.sql* containing your query on the view. In that file you must also include the comment with the output of the query on the provided data before the deletions, the SQL statements that delete the two tuples indicated above, and a second comment with the second output of the same query after the deletions. You do not need to replicate the query twice in the *queryview.sql* file (but you will not be penalized if you do).

2.7 Create an index

Indexes are data structures used by the database to improve query performance. Locating all the ChirpPosts for made by a particular user on or after a particular date may be slow if the database system has to search the entire ChirpPosts table. To speed up that search, create an index named LookUpPosts over the posterID and postDate columns (in that order) of the ChirpPosts table. Save the command in the file *createindex.sql*.

Note that you can run the same SQL statements whether or not this index exists; having indexes just changes the performance of SQL statements.

For this assignment, you need not do any searches that use the index, but if you're interested, you might want to do searches with and without the index, and look at query plans using EXPLAIN to see how queries are executed. Please refer to the documentation of PostgreSQL on EXPLAIN [here](#).

3 Testing

Before you submit, login to your database via psql and execute the provided database creation and load scripts, and then test your seven scripts (combine.sql foreign.sql general.sql unittests.sql createview.sql queryview.sql createindex.sql). Make sure that you run the query on the view make sure you recreate the schema and reload the data, as the updates you choose to do for testing the general constraints may change the initial data in a way that changes the output of the query on the view. The command to execute a script is: \i <filename>.

4 Submitting

1. Save your scripts indicated above as combine.sql foreign.sql general.sql unittests.sql createview.sql queryview.sql createindex.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 files to a single file with name Lab3_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 Lab3 should be named Lab3_1234567.zip To create the zip file you can use the Unix command:

```
zip Lab3_1234567 combine.sql foreign.sql general.sql unittests.sql createview.sql queryview.sql  
createindex.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.
4. Lab3 is due on Canvas by 11:59pm on Sunday, May 20. Late submissions will not be accepted, and there will be no make-up Lab assignments.