

CSE 180 - Lab 4

Stored Functions and C Application Programming

Dev
Nov 22 2023, Section 8



Logistics

- Lab 4 Due 11:59 PM, Tuesday December 5



Getting Started

- We provide you the create file
- New data loading file
- Goals
 1. Write a C program that interfaces with PostgreSQL
 2. Write 3 functions: `countCoincidentSubscriptions`, `increaseSomeRates`, `ChangeAddresses`
 3. Write a Stored Function `increaseSomeRatesFunction` on the database

What to turn in?

- `runNewspaperApplication.c`
- `increaseSomeRatesFunction.pgsql`

Functions in runNewspaperApplication

- `countCoincidentSubscriptions(PGconn *conn, int theSubscriberPhone)`
 - Returns
 - $s1's \text{ start date} \leq s2's \text{ start date, and}$
 $s2's \text{ start date} \leq s1's \text{ end date.}$
 - Number of coincident subscriptions
 - -1 if the subscriber doesn't exist
 - Arguments: `int theSubscriberPhone`
 - Checks the number of coincident subscriptions

Functions in runNewspaperApplication

- `changeAddress(PGconn *conn, char *oldAddress, char *newAddress) :`
 - Returns:
 - Number of updates if updated
 - 0 if not updated
 - -1 if invalid
 - Updates the Address field

Functions in runNewspaperApplication

- `increaseSomeRates(PGconn *conn, int maxTotalRateIncrease) :`
 - Returns : The value returned by stored function
`increaseSomeRatesFunction`
 - Arguments: `int maxTotalRateIncrease`
 - Calls the stored function

Stored Function

`increaseSomeRatesFunction(maxTotalRateIncrease INTEGER):`

iterates through the subscriptionKinds for rate increase

- increase 10 if popularity ≥ 5
- increase 5 if $3 \leq \text{popularity} < 5$
- increase 3 if $2 \leq \text{popularity} < 3$
- increase 0 if $0 \leq \text{popularity} < 2$
- total increase cannot exceed maxTotalRateIncrease

Updates only if the total does not exceed maximum allowed. Returns total increase.

In the main()

- Set up the database connection
- Use username and password to connect to the database
- Run the tests mentioned in Lab4 over your 3 functions
- A template has been provided

libpq

- PostgreSQL library that we use in C
- Same library as psql
- Make sure you follow the compilation instructions for unix.ucsc.edu
- `gcc -L/usr/include -lpq -o runNewspaperApplication runNewspaperApplication.c`
- `./runNewspaperApplication <your_userid> <your_password>`
- On Mac, if using Homebrew:
 - `gcc -L/usr/local/opt/libpq/lib -I/usr/local/opt/libpq/include/ -lpq -o runNewspaperApplication runNewspaperApplication.c`

Execution Functions

- `PGresult *PQexec(PGconn *conn, const char *command);`
- The connection will be the first parameter in any of your functions
- You can use String concatenation commands to generate the second parameter

```
PGresult *Result = PQexec(conn, "SELECT * FROM Conferences");
```

<https://www.postgresql.org/docs/current/libpq-exec.html>

DB Connection functions: Template

Getting Results

- `int PQntuples(const PGresult *res);`
 - Returns the number of tuples returned by each query
- `int PQnfields(const PGresult *res);`
 - Returns the number of fields, or columns returned by the query
- `char *PQgetvalue(const PGresult *res, int row_num, int column_num);`
 - Returns the value of result at row number, column number

Simple C example

```
int getSubscribers(PGconn *conn){  
  
    char stmt[30] = "SELECT * FROM Subscribers";  
    PGresult *res = PQexec(conn, stmt);  
    if (PQresultStatus(res) != PGRES_TUPLES_OK)  
    {  
        fprintf(stderr, "SELECT failed: %s", PQerrorMessage(conn));  
        PQclear(res);  
        bad_exit(conn);  
    }  
  
}
```

Simple C example

```
int n = PQntuples(res);
for (int j = 0; j < n; j++)
    printf("Number %d has %s Name\n",
           atoi( PQgetvalue(res, j, 0) ),
           PQgetvalue(res, j, 1) );

printf("\n");
PQclear(res);
return n;
}
```