Embedded SQL

- Purpose is to allow you to run SQL commands from within a C program

Step 1: Skeleton File

- Create an empty file named test.sqc (not .c)
- Include the following lines at the top of the file:

```
#include <stdio.h>
#include "util.h"
EXEC SQL INCLUDE SQLCA;
```

- You can get util.h and util.c from /u/cs448/public

Step 2: Declaring variables to return data from queries

- To get data back from your SQL commands, you need to declare variables in which the results are stored.

```
EXEC SQL BEGIN DECLARE SECTION;
     char db[6] = "cs448";
     char title[72], pubid[9];
     short dollars;
EXEC SQL END DECLARE SECTION;
```

- This declaration block should be placed in the same function that the query will be called from

Step 3: Connect to the CS 448 DB and Error Handling

- To connect to the CS 448 DB, you need to type the following:

```
EXEC SQL CONNECT TO :db;
```

- The simplest way to handle errors is to type the following:

```
EXEC SQL WHENEVER SQLERROR  GO TO error;
```

- Then, later in the code, insert the label `error:` which is followed by the error handling code (more on this later)

Step 4a: Simple retrieval

- This method will retrieve a single tuple from a single query

```
    strncpy(pubid,argv[1],8);
    EXEC SQL WHENEVER NOT FOUND GO TO nope;
    EXEC SQL SELECT title INTO :title
             FROM   publication
             WHERE  pubid = :pubid;
    printf("%10s: %s\n",pubid,title); goto done

nope: printf("%10s: *** not found *** \n",pubid);
done:
```

- Note that whenever you are using C variables, you need to place a colon in front of them.

Step 4b: Using cursors

- This method will retrieve **multiple** tuples from a single query

```
strncpy(apat,argv[1],8);

EXEC SQL DECLARE auth CURSOR FOR
          SELECT name, title
          FROM author , wrote, publication
          WHERE name LIKE :apat
            AND aid=author
            AND pubid=publication;

EXEC SQL OPEN auth;
EXEC SQL WHENEVER NOT FOUND GO TO end;
for (;;) {
  EXEC SQL FETCH auth INTO :name, :title;
  printf("%10s -> %20s: %s\n",apat,name,title);
};
end:
```

- The cursor must be declared first, then opened before being used
- Indicate where control should flow when there are no more tuples
- Create an infinite loop that fetches a new tuple in each iteration

Step 5: Cleaning up

- After all your database use is complete, you should include the following two commands:

```
EXEC SQL COMMIT;
EXEC SQL CONNECT RESET;
```

Step 6: Error Handling

- We included a command earlier that causes the code to jump to the error label if there is ever an error performing an SQL action
- Your error handling code should look as follows:

```
error:
   check_error("My error",&sqlca);
   EXEC SQL WHENEVER SQLERROR CONTINUE;

   EXEC SQL ROLLBACK;
   EXEC SQL CONNECT reset;
   exit(1);
```

Step 7: Makefile & Other Information

- Before your .sqc file can be compiled, it must be preprocessed and turned into a .c file.
- A Makefile is available in /u/cs448/public/Makefile that should do all the work of preprocessing and compiling your code. You need only change the first line which indicates the name of your .sqc file
- There are four example .sqc files available in /u/cs448/public/sample[1-4].sqc
- Finally, there is an example demonstrating query 2 of assignment 1 located at /u/cs448/public/q2.sqc