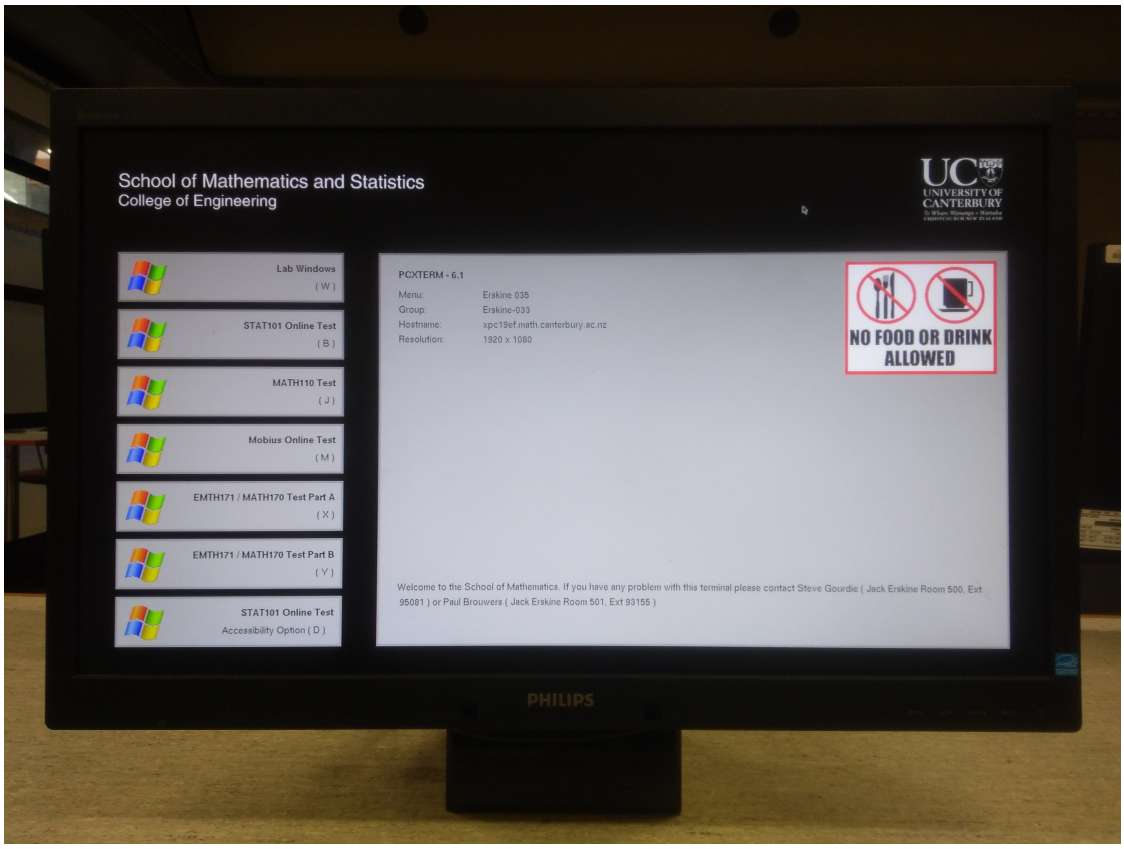
Course Blocks

INFO263-20S2 - Web Design and Development

Group Project Specifications

Background: Thin Clients, Event Scheduler, and the tserver Database

The UC School of Mathematics & Statistics hosts a number of computer labs in the Erskine building for UC-wide invigilated assessment, as well as a variety of computer tutorials for the Maths & Stats students. In these labs there around 180 active thin-client terminals. They display a menu at runtime that is supplied from a central server via a network daemon. Following is an example menu from one of the thin clients:



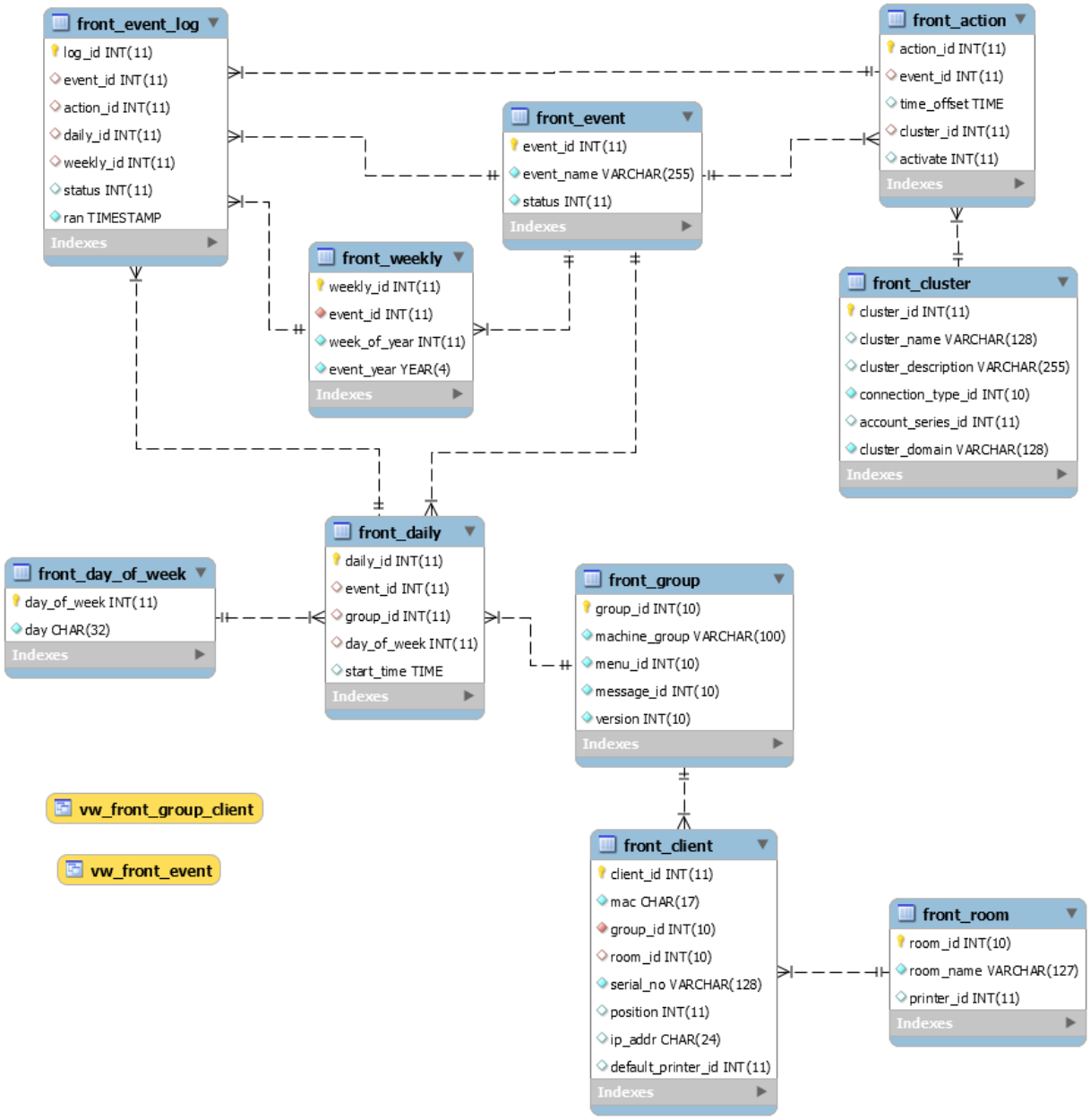
The network daemon runs the scheduler, where the schedule is defined in a database, tserver, which runs on MySQL server. There is no direct access to the database from the thin clients and the network daemon only accesses the database via stored procedures -- no direct access to the database tables allowed.

The scheduler is based around the concept of an event. The events are defined in the tserver database via direct interaction with the database, which is very laborious, error-prone and entirely user-unfriendly. The following screen cast demonstrates the process of defining an event, which happens to be an EMTH119 test.

tserver Event Definition

As demonstrated in the screen cast, an event has a name and an event_id and a name. Each event has a number of actions associated with it, which define what is done when the event occurs -- essentially these actions turn on or off menu items. Then the event details define what time and day of week this event should be activated for each action, as well as which weeks the event should occur.

The essential tables are front_event, front_actions, front_daily, front_weekly, front_event_log, front_groups, front_cluster. The vw_front_event view is the most relevant view as it can supply a summary of what happened/will happen over a specified time. Following is the ER diagram, showing the relationships between the various tables in the tserver database:



- front_event -- this table contains event name along with the event_id, which is used in other tables which define other event details.
- front_group -- this table defines a number of groups of clients (computers) which correspond to various rooms in the system.
- front_weekly -- this table defines which week(s) of the year the event will take place in.
- front_daily -- this table defines the days of a given week in the event schedule.
- front_cluster -- this table defines a set of various configurations which may be associated with an event; please note for clarity, that this table would be more appropriately called configuration type or something along those lines.
- front_actions -- this table defines which configurations need to be switched on or off in the event schedule.
- front_event_log -- this table serves as a record of triggered actions; however the table is modified by the network daemon, and not the tserver web interface.

Given the model above, and the screen cast, a definition of the event is expressed as follows:

```
insert into front_event (event_name, satus) values ('EMTH119-20S2 Tuesday', 1); -- 173
select group_id from front_group where machine_group in ('Erskine-033', 'Erskine-035', 'Erskine-036', 'Erskine-038'); --
5, 6, 22, 7
select cluster_id from front_cluster where cluster_name = "MapleTA"; -- 4
insert into front_weekly (event_id, week_of_year, event_year) values (173, 34, 2020);
insert into front_daily (event_id, group_id, day_of_week, start_time) values (173, 5, 2, '18:00:00');
insert into front_daily (event_id, group_id, day_of_week, start_time) values (173, 6, 2, '18:00:00');
insert into front_daily (event_id, group_id, day_of_week, start_time) values (173, 22, 2, '18:00:00');
insert into front_daily (event_id, group_id, day_of_week, start_time) values (173, 7, 2, '18:00:00');
insert into front_action (event_id, time_offset, cluster_id, activate) values (173, '-00:05:00', 3, 0);
insert into front_action (event_id, time_offset, cluster_id, activate) values (173, '-00:05:00', 4, 1);
insert into front_action (event_id, time_offset, cluster_id, activate) values (173, '01:00:00', 4, 0);
insert into front_action (event_id, time_offset, cluster_id, activate) values (173, '01:00:00', 3, 1);

-- All done!

select * from vw_front_event where date = '2020-08-18' order by time, cluster_id, group_id;
```

Requirements: tserver Web Interface

The task of your group is to develop a web interface to the tserver database. Following are the features, specifications for the web interface:

- The interface must be implemented in PHP, preferably using the same version we use in the labs, 7.2 or higher.
- The interface must allow user authentication -- you may need to create an additional table for this purpose.
- We need be able to display a list of currently defined events and for each event display its properties, such as dates, times, group and action.
- We need to be able to see a current state which displays what actions have happened, what is currently happening and what will happen in the near future. This should allow you to define the time periods to display.
- We need to be able to define a new event.
  - We need to be able to associate actions with it, preferable with pull-down items for time and cluster selection.
  - We need to be able to define sets of time, day of week and group of machines where this event will occur.
  - We need to be able to define the week of the year and year where these events will take place.
- The interface must have a search feature, where the user would get a dynamically populated pull-down menu with a list of matching events as soon as the user types at least three characters of the event name.
- The interface must have basic navigation features between the various tasks/pages.
- The solution must include a two-to-three page documentation for your solution, providing a description of your implementation, logic, and decision-making process.

Please note that the task of scheduling a event may be implemented as a series of linked steps, with the corresponding pages, or a single page with more sophisticated design.

A good solution use AJAX (Asynchronous JavaScript and XML) to retrieve the necessary details from the database during event scheduling, say to populate drop-down menus. Similarly, a good solution will use CSS libraries to create a suitable layout and a pleasing user experience. For example, it is advisable to use such libraries as jQuery and Bootstrap -- those will be briefly introduced later in the course; however, you may use other JavaScript and CSS libraries, if necessary. You may use images where suitable, for page backgrounds, but note the image license terms, if you source the image from the web.

As mentioned earlier, there is no direct access to the database tables allowed. Your implementation is to follow the same style -- using using stored procedures rather than direct queries. So, you may write your own stored procedures to add to the schema, as well as create additional views.

Getting Started

The first thing to do is to import the tserver.sql SQL dump into your INFO263_xyz123_tserver database (on the INFO263 MySQL server used the labs, where xyz123 is your user name), or else you can create an empty tserver database on your own computer, and import the tserver.sql SQL dump. Then it is recommended that your watch the tserver Event Definition screen cast until you understand what is going on. A clear understanding of this screen cast will ensure you start on track, and remain on track with your solution implementation.

Then it is recommended that you examine the tserver database with all it's tables, views, and stored procedures. Please keep in mind that tserver.sql SQL dump contains only a trimmed version of the actual tserver database -- the actual database contains a lot more details that go beyond the scope of this assignment.

Of course, you are encouraged to post questions about this via our INFO263 Discussion Forum. It is only natural that further clarifications will be required.

To begin with it may be necessary to have a few group meetings to divide tasks and establish your group collaboration approach. It may be useful to look into the use of version control software, but make sure you use private source code repositories to protect your group's solution.

Marking Schedule

- User authentication; at a bare minimum a user must be capable of signing in and signing out, using a user code and password — 5%
- Navigation; being able to get around the various pages of the site — 5%
- Displaying a list of events with the relevant details; this is to show the current state of the database/event list, which may be limited to a date range — 5%
- Core functionality, defining new events; this may be done in a single page, or in multiple steps — 25%
- Event search; given a string or (optionally) a date range, return the list of events; these features may be integrated into the event list display page, with the appropriate use of Ajax — 5%
- Other extra features, which may include event editing, user profile editing and more, up to your discretion — 5%
- Code style; use and declaration of the necessary stored procedures in MySQL, appropriate structure of the code with functionality decomposition into functions/classes/methods and files — 15%
- Ease of use, not overtaxing user's memory, simplicity of work flow, design considerations, such as visual appeal — 15%
- Documentation; the description of your specific implementation, additional components/libraries mention, and code provenance mention, if you were using some other code, other than from some common well-know libraries and such — 20%

Last modified: Monday, 5 October 2020, 3:52 PM

◄ INFO263-20S2 Midterm Test

Jump to...

Group Project Submission ►