

CMSC 424: Application Development Project: Vital Statistics Data Management - Fall 2009

1 Overview

In last few years, miniaturization of sensing devices like GPS, heart rate monitors (HRMs) etc., has changed the way athletes do training or log their workouts. Specifically it has become feasible to accurately track the vital statistics (like heart rate, blood pressure, VO2) quite precisely and very easily (by having them connect (wirelessly) to a cellular device like an iPhone). More generally being able to track the vital statistics continuously could help in early diagnosis of serious ailments.

You are to write a web-based application toolkit, called VSD, to allow users to collect, manage, and visualize (to some degree) such data. Specifically, users will have a collection of sensing devices, e.g. GPS (for location), heart rate, blood pressure, VO2 etc (but the users may want to use other sensing devices as well). The users may not record data all the time (e.g. data is recorded maybe only during workouts), and not all sensors will be active at all times. The users may want to browse historical data, look for patterns, or visualize or otherwise manipulate the data (more details below). The primary care physicians of the users also have access to their data.

You are free to change the details, as long as the main functions described here are supported. In case of drastic changes, you should talk to the TA or me first. The project description is intentionally somewhat vague; you are free to decide what kind of interfaces to support etc.

2 Data

A brief description of the data that needs to be stored in the database follows.

- **Subscribers:** Information about the subscribers (names, addresses, age), what sensors they use etc.
- **Sensors:** Different types of sensors that may be used by the subscribers, and any other pertinent information about them. Sensors should be grouped by the type, e.g., *heart rate* sensors (of which there are many).
- **Physicians:** Contact information about the physicians, and the subscribers for whom they are the primary physician.
- **Sensor data logs:** The actual sensor readings, the times they were collected etc.
- **Alerts:** See below.
- **Events:** Users may define significant events, each associated with a time duration. E.g. an event may be “training ride” or “marathon”, which would be associated with a time duration of the event. See below for more details.

3 Tasks & Queries

There are two sets of tasks/queries that VSD needs to support. The main tasks are provided below. You may choose to add and/or substitute other tasks depending on the emphasis of your project.

3.1 Stand-alone, Continuously Running Programs

Some tasks are better done using stand alone programs that sit atop the database system.

- **Data Ingest:** It is not feasible to expect the sensing devices or the mobile phones to use SQL to insert the sensed data into the system. Instead we will have a stand-alone program that is listening on some port, that accepts XML messages, and inserts appropriate tuples into the database. Assume that all the devices follow a simple format for sending their information.

```
<sensor-reading>
  <subscriber> the subscriber id </subscriber>
  <time> ... </time>
  <sensor> the id of the sensor </sensor>
  <reading> the sensor reading </reading>
</sensor-reading>
```

For instance, a specific message, for user “s100” for sensor “hr1” might look like:

```
<sensor-reading>
  <subscriber>s100</subscriber>
  <time>8/10/2009:13:30:45</time>
  <sensor>hr1</sensor>
  <reading>120</reading>
</sensor-reading>
```

- **Simple Alerts:** The data ingest program will also provide support for some simple alert monitoring. Specifically, an alert should be generated if: for a user, the reading of a sensor is consistently low or high for a period of time. In case this is observed, then the physician should be alerted (by email – for now, you can output a message to a log file or to standard output).

An alert is defined by four parameters: (subscriber-id, sensor-id, threshold-predicate, duration). E.g. an alert ('s100', 'hr1', '> 170', 3600) asks us to monitor the heart rate sensor hr1 of subscriber s100, and if the predicate '> 170' is true consistently for a duration of 3600 seconds, then an alert should be generated.

It is unclear how exactly and when to check for alerts, especially since we allow for sensor readings to be missing. The simplest way to check for alerts is when a new message arrives. At that time, we could check whether all the readings in the last hour satisfy the predicate. However, if there hasn't been any recorded reading within last hour, then this may lead to too many alerts and the alerts may not be useful. You are free to decide how to handle this – list this out clearly in report.

Alerts may be defined by the physicians or the subscribers using the web interface (see below).

3.2 Web Interface

The following tasks should be provided using a web interface.

- **Subscriber and Physician log-ins:** There will be an account for each subscriber, and also for each physician. The accounts must be password protected.

- **Insert/update Sensors, Subscribers, Physicians etc.:** These web-based interfaces will allow addition/updates of new subscribers, new physicians, information about who is the primary care physician etc.
- **Allow defining alerts:** An intuitive and easy to use interface would be desirable.
- **Browsing historical sensor data:** Users or physicians should be able to browse through the historical data. Browsing by dates, by sensors etc. should be allowed. Ideally plots should be shown, but for the purposes of the project, lists of readings would be fine.

Typically more sophisticated analysis of the data would be desired as well. You should support one such basic functionality: a user (or his physician) should be able to choose a sensor, and for that sensor, you should generate a table, with columns indicating the dates, and rows corresponding to the hours (so 24 rows). Each cell in the table should contain the average reading for that hour, for that day.

- **Associating events with sensor logs:** Users should be allowed to associate events with time durations with the sensed data. E.g. a user may have logged all their data during a training bike ride (this may happen automatically). After logging in and having browsed the data, the user may associate the readings with an event "bike ride", to make it more informative for the user or the physician.

3.3 Populating the tables

You should populate the tables manually with sufficient tuples to illustrate the key functionality (10-100 each for smaller tables, a few thousands for the sensor readings). You should also set up a program for automatically sending sensor reading updates to VSD, in order to demonstrate the stand alone programs. Feel free to use made-up data.

4 Rules of the game

- **Groups:** The project is to be done in groups of 2 students. The groups are "self-policing" (e.g., each group is responsible for its own division of labor, scheduling, etc.). *Note: If an unreconcilable problem arises in your group, it is your responsibility to contact both me and the TA as soon as possible.*
- **Assumptions:** In cases where you have questions on the above description, it is acceptable to make assumptions about the application providing that: 1) they are explicitly stated in the report, 2) they don't terribly conflict with any of the requirements specified above, and 3) they are "reasonable". If you have a question about the acceptability of any of your assumptions, check with the TA or the professor.
- **Reports:** A report should be handed in at the end of each phase (Due dates below).
- **Implementation:** The final phase of the project requires a working implementation of the system to be built, tested, and demonstrated. A large part of the project grade depends on the quality of this implementation. The implementation will be done as a client-server system in which a web server runs on your cluster unix account, accepts web queries, and connects to the Oracle DBMS to retrieve the data (you are free to use other tools).

5 Project Phases

Phase	Phase Name	Due Date
<i>0</i>	Group names to the TA	Sept 22, 2009
<i>I</i>	System Analysis and Specification, Conceptual Modeling, Toy Web Application	Oct 22, 2009
<i>II</i>	Implementation, Testing, and Demo	Nov 24, 2009

6 Reports

The Phase I report must contain:

1. Assumptions you have made the enterprise, or any additions/changes made to the specs.
2. The graphical schema using the E-R model, including a list of the attributes for each entity and relationship.
3. The relational schema obtained by converting E/R to relations, and their BCNF or 3NF forms with keys.
4. A description of how you plan to populate the tables (including details of how you plan to parse the XML updates).
5. A toy end-to-end application that uses the database. The sole purpose of this is to familiarize yourself with how to build such an application. E.g. you could populate the database with the Users data from the SQL assignment, and the webpage could provide a text box for typing in a user name (like Google search box), and when the “submit” button is clicked, the results of “select * from Users where name like '%ABC%’” would be displayed (where ABC is the name entered in the text box).

You can either send me and the TA an email with the website address (preferred), or show it to the TA in person. More details will be announced later.

The Phase II report must contain:

1. Phase I report with corrections addressing TA’s feedback.
2. Any revisions made to the relational schema definition from Phase II,
3. A brief summary of your implementation efforts, the tools used etc.
4. The key deliverable here is the demo of the system in person. Details will be announced later.