Advanced Databases Coursework 1

October 20, 2017

1 Introduction

This excercise is meant to familiarize you with an object-relational modeling framework called ODB (see `http://www.codesynthesis.com/products/odb/doc/manual.xhtml` for the documentation). As well as the creation of columnar indices (basically copies of columns in DSM format) in SQL Server.

1.1 Setup

For your labwork, we have set up a virtual machine template with a Microsoft SQL server instance that is preloaded with the Yelp dataset (find the details of the dataset at `https://www.yelp.com/dataset/documentation/sql`). We have slightly altered the schema in that we created id columns where necessary for the coursework. You will find the altered schema at the end of this document.

To set up the environment, log in to the CSG cloudstack environment (`http://cloudstack.doc.ic.ac.uk`) with your imperial account name, password and the domain `doc`. Under `Instances`, add an instance in the `doc-46` zone. Select the `advanceddb-yelp` template in the list of featured images. Select a custom compute offering (from the list) and configure it with 2 CPU cores of 2000Mhz and 4096 MB of memory. Leave the rest of the settings on default (i.e., simply click next throughout the dialog) and launch the vm. After a couple of minutes waiting, you will see the IP address of the newly created VM in the `Instances` overview page.

Log in to the VM using ssh and execute the following script to check out the project template:

```
git clone https://github.com/little-big-h/AdvancedDB17CW1.git
cd AdvancedDB17CW1
mkdir build
cd build
cmake ..
```

In the project, you will find three files: `CMakeLists.txt`, `ADBCoursework1.cpp` and `Yelp.hpp`. The first is part of the build-system and need not concern you (though we encourage you to learn about CMake). The other two need to be modified to implement your solution. The `Yelp.hpp` file will contain your class structure for the ORM part of the assignment. `ADBCoursework1.cpp` will contain the active parts of your solution. All three tasks need modification of this file.

To build the project execute:

```
cmake --build ~/AdvancedDB17CW1/build
```

To run it execute

```
~/AdvancedDB17CW1/build/ADBCoursework1
```

1.2 Exercise

There are three parts to the excercise:

- The first is to find the opening hours of all restaurants a user has rated. You need to provide the implementation of the function `findHours` in `ADBCoursework1.cpp` that accepts a single string parameter that designates a username and returns the hours as an array of strings. You are expected to implement this query entirely using the ODB object relational mapper and the C++ language (no SQL).
- The second query involves the calculation of aggregates: write a query to calculate a histogram of all reviews (number of stars) for all businesses within an area (defined by latitude and longitude). Implement your solution in the function `countStars`. The query needs to be wrapped in what ODB calls a view: a query with an output schema that is reflected in a C++ class. This class (`StarCount`) is given as part of the assignment and included in the `Yelp.hpp` header file. However, you will need to make sure that the query you write exactly matches the schema of the `StarCount` class: `stars, count`, both as integers.
- The third task is the creation of what Microsoft calls columnstore indices. These are, ipn essence, a DSM representation of selected columns of an single table (base tables are always represented in NSM format in SQL Server). Create a columnstore index to support the `countStars` query you've developed in the previous task (see `https://docs.microsoft.com/en-us/sql/t-sql/statements/create-columnstore-index-transact-sql` for the technical documentstion). To fulfill this part of the assignment, implement the function `createIndex` and `dropIndex`. Compile and run the program, observe the output (you should see at least 50 percent performance improvement) and explain the performance difference.

1.3 Submission

- You need to submit two files through cate: the yelp.hpp file containing the class model you developed based on the schema of the yelp database. In addition, you need to submit the ADBCoursework1.cpp file with the implementation of the four functions for the assignments. In addition, submit a copy of the output of the program as well as a short (one or two paragraphs) explanation of the performance before and after creating the columnstore index.

2 Hints

2.1 Anticipated Pitfalls

When creating the relationships in the class structure in ODB, one has to be careful to prevent ownership cycles (which can lead to infinite loops when loading the data). This is, unfortunately, not very prominently explained in the ODB documentation.

Section 6.2 starts with the paragraph

> In bidirectional relationships we are interested in navigating from object to object in both directions. As a result, each object class in a relationship contains a pointer to the other object. If smart pointers are used, then a weak pointer should be used as one of the pointers to avoid owndership cycles.

Also consider using Lazy Pointers as described in Section 6.4 of the documentation.

2.2 Database schema

For your convenience, here is the modified schema of the yelp database.

```
SET QUOTED_IDENTIFIER ON
CREATE TABLE "attribute" (
  "business_id" varchar(22) NOT NULL,
  "name" varchar(255) DEFAULT NULL,
  "value" text,
);
CREATE TABLE "business" (
  "id" varchar(22) NOT NULL,
  "name" varchar(255) DEFAULT NULL,
  "neighborhood" varchar(255) DEFAULT NULL,
  "address" varchar(255) DEFAULT NULL,
  "city" varchar(255) DEFAULT NULL,
  "state" varchar(255) DEFAULT NULL,
  "postal_code" varchar(255) DEFAULT NULL,
  "latitude" float DEFAULT NULL,
  "longitude" float DEFAULT NULL,
  "stars" float DEFAULT NULL,
  "review_count" int DEFAULT NULL,
  "is_open" tinyint DEFAULT NULL,
  PRIMARY KEY ("id")
);
CREATE TABLE "category" (
  "business_id" varchar(22) NOT NULL,
  "category" varchar(255) DEFAULT NULL,
);
CREATE TABLE "checkin" (
  "business_id" varchar(22) NOT NULL,
  "date" varchar(255) DEFAULT NULL,
  "count" int DEFAULT NULL,
);
CREATE TABLE "elite_years" (
  "user_id" varchar(22) NOT NULL,
  "year" char(4) DEFAULT NULL,
);
CREATE TABLE "friend" (
  "user_id" varchar(22) NOT NULL,
  "friend_id" varchar(22) DEFAULT NULL,
);
CREATE TABLE "hours" (
  "id" int not null,
  "hours" varchar(255) DEFAULT NULL,
  "business_id" varchar(22) NOT NULL,
  PRIMARY KEY ("id")
);
CREATE TABLE "photo" (
  "id" varchar(22) NOT NULL,
  "business_id" varchar(22) NOT NULL,
  "caption" varchar(255) DEFAULT NULL,
  "label" varchar(255) DEFAULT NULL,
  PRIMARY KEY ("id"),
);
CREATE TABLE "review" (
  "id" varchar(22) NOT NULL ,
  "stars" int DEFAULT NULL,
  "date" datetime DEFAULT NULL,
  "text" text,
  "useful" int DEFAULT NULL,
  "funny" int DEFAULT NULL,
```

```
  "cool" int DEFAULT NULL,
  "business_id" varchar(22) NOT NULL,
  "user_id" varchar(22) NOT NULL ,
  PRIMARY KEY ("id"),
);
CREATE TABLE "tip" (
  "user_id" varchar(22) NOT NULL,
  "business_id" varchar(22) NOT NULL,
  "text" text,
  "date" datetime DEFAULT NULL,
  "likes" int DEFAULT NULL,
);
CREATE TABLE "user" (
  "id" varchar(22) NOT NULL,
  "name" varchar(255) DEFAULT NULL,
  "review_count" int DEFAULT NULL,
  "yelping_since" datetime DEFAULT NULL,
  "useful" int DEFAULT NULL,
  "funny" int DEFAULT NULL,
  "cool" int DEFAULT NULL,
  "fans" int DEFAULT NULL,
  "average_stars" float DEFAULT NULL,
  "compliment_hot" int DEFAULT NULL,
  "compliment_more" int DEFAULT NULL,
  "compliment_profile" int DEFAULT NULL,
  "compliment_cute" int DEFAULT NULL,
  "compliment_list" int DEFAULT NULL,
  "compliment_note" int DEFAULT NULL,
  "compliment_plain" int DEFAULT NULL,
  "compliment_cool" int DEFAULT NULL,
  "compliment_funny" int DEFAULT NULL,
  "compliment_writer" int DEFAULT NULL,
  "compliment_photos" int DEFAULT NULL,
  PRIMARY KEY ("id")
);
```