assignment-1.md

Cassandra Assignment 1

Zoltan Debre - 300360191

Original source code and git log: https://github.com/zoltan-nz/cassandra-exercise

Run everything with the following source command in cqlsh

```
source './reset.cql'; source './q4.cql'; source './q5.cql'; source './q6.cql';
```

Answers for Question 1, Question 2 and Question 3 in this file, CQL commands in `q4.cql`, `q5.cql` and `q6.cql`.

Assignment Questions

(Note for cross-references: The list in Question 1 is mainly about "updates", the list in Question 2 is mainly about "reads", for this reason the cross-reference is in the following format, ex. "Question 2, Read 1" or "Question 1, Update 3.2".)

Question 1.

List the database write and `update` requests the application requires using plain English. [10 marks]

1. Administrator creates a keyspace for the application. Keyspace name: `tranz_metro`.
2. Administrator creates accounts for drivers:

   2.1 Create a table for drivers if not exists. Table name: `driver`. Columns: `driver_name` (unique, if not exists, primary key), `password` (string), `mobile` (number), `current_position` (string), `skill` (`set` type with strings).

   2.2 Seed the initial drivers data.
3. Drivers can update:

   3.1 Drivers can change their password. They provide `old_password` and `new_password`. Update the driver's row with `new_password` only if the `old_password` equal with the stored `password`. If the conditions apply, `password` will be equal with `new_password`.

   3.2 Drivers can update their `current_position`: (with city name string) `'Wellington'` OR (with vehicle) `vehicle_id` OR (with not available string constant) `'not_available'`. The update process managed by the app, based on the driver's skill and the location of the train. See Question 2, Read 5.

   3.3 Drivers can add new skill to `skill` column. Skill column type is `SET<string>`.

   3.4 App updates a counter log table for payrol. See Question 2, Read 1.
4. Administrator initializes vehicles:

   4.1 Create a table for vehicles. Table name: `vehicle`. Columns: `vehicle_id` (string, unique, if not exists), `status` (string), type (string)

   4.2 Seed the initial vehicles data.
5. App automatically updates the `status` of a `vehicle`. Station name, like `Wellington` OR `in_use` OR `maintenance` OR `out_of_order`.

   5.1 Status will be updated based on timetable (departure). `Status` will be the departure station name. See Question 2, Read 4.

5.2 `Status` will be updated to `in_use`, when driver turns on the engine.

5.3 Status will be updated when the driver turns off the engine on the destination station. `status` will be the destination station name. A log event will be called also, see Question 1, Update 7.2.

6. Administrator initializes timetables:

   6.1 Create a table for timetables. Table name: `time_table`. Columns: `line_name` (unique, if not exists, string), `service_no` (number, asc within line_name), `station_name` (string), `latitude` (double), `longitude` (double), `time` (int), `distance` (double), Notes: time are departure times, except the last (destination) time, it is arrival time. Sorted `asc` by `time`.

   6.2 Seed `time_table`.

7. Recording the travelled distance of a vehicle.

   7.1 Need a `vehicle_usage` table for logging vehicle usage. Administrator can create this table with the following columns: `vehicle_id`, `total_distance` (counter).

   7.2 This log will run after the app updated the vehicle `status`. See Question 1, Update 5.3. Distance information comes from Question 2, Read 7.

8. Recording data points after the vehicle's engine started.

   8.1 Administrator create a table. Table name: `data_point`. Columns: `day` (int), `sequence` (timestamp), `latitude` (double), `longitude` (double), `speed` (double).

   8.2 The app creates a new log entry in this table in every 10 seconds, when the vehicle's engine is on.

REVIEW THIS: 9. Administrator create a neighbour reference table and seed with initial data. Table name: `station`, Columns: `name` (string), `latitude` (double), `north_neighbour` (string), `south_neighbour` (string)

Question 2.

List the `read` requests the application requires using plain English. [12 marks]

1. Read the number of working days of a driver. (Payroll will use this information.). App collects this information in a separate table. Table name: `driver_working_days`, Columns: `driver_name` (unique, string), `working_day` (counter). This is a counter table and the app will update the counter, when the driver starts to work.

2. Read timetable data for showing timetable for passengers. Requested columns from `time_table` table: `line_name`, `station_name`, `time`.

3. Application can list `station_name`, `service_no`, `time` from `time_table`. `desc` sorted by `time`.

4. The iPhone app, which is on the train can read `station_name`, `time`, `line_name`, `service_no`. The iPhone app connected with a train with `line_name` and `service_no`. If the `line_name`, `service_no` and `time` matches, we can update the vehicle status. See Question 1, Update 5.1

5. The application runs a query to list trains on a station. The application reads from `driver` table driver's `current_position` and check their `skill` values. If the list of skills contains the `vehicle`'s `type`, the driver will get a text message and the driver will be allocated to this train. The driver's `current_postion` will be updated. See Question 1, Update 3.2.

6. The app authentication service reads from the database, from `driver` table, `password` column for checking password, which provided by the driver after she/he entered in the vehicle.

7. For logging `vehicle_usage`, the app has to be able to read distances from `time_table`. See Question 1, Update 7.2.

8. Readings from `data_point` table:

   8.1 Last entry of a service, based on `line_name` and `service_no`.

   8.2 List of entries in a time interval (`start_time`, `end_time`). List all the entries, where `sequence` between the given time intervals.

8.3 Find a data point in `data_point` table. It can provide a `time` and `latitude`. With this information find the previous and next station in `time_table`. List city names as `north_neighbour` and `south_neighbour`.

Question 3.

Consider Cassandra data model design guidelines we discussed in lectures and list names of database tables the application requires using plain English. Recall, Cassandra tables strongly depend on requested queries. If there is no queries needing a table, the table should not exist. (Don't invent queries to justify the existence of any tables.) After each table name, list those queries you identified in your answer to question 2 that use the table. [9 marks]

Tables.:

1. `driver` => Question 1 Update 2.1, 2.2, 3.1, 3.2, 3.3; Question 2 Read 5, 6;
2. `vehicle` => Question 1, Update 4.1, 4.2, 5.1, 5.2, 5.3, Question 2, Read 4.
3. `time_table` => Question 1, Update 6.1; Question 2, Read 2., 3., 7.
4. `vehicle_usage` => Question 1, Update 7.1
5. `data_point` => Question 1, Update 8.1, Question 2., Read 8.1, 8.2
6. `station` => Question 1, Update 9; Question 2, Read 9.1;
7. `driver_working_days` => Question 2, Read 1.

Question 4.

Create data model using CQL 3 statements that support the requirements. To answer questions, use Cassandra CCM. In your answers, copy your CCM and CQL commands. [20 marks]

A/ Create a cluster and a keyspace that will satisfy infrastructure and availability requirements above. [5 marks]

Create cluster with 6 node, using Cassandra v3.10 with `ccm`

Run the following command in terminal.

```
$ ccm create tranz-cluster -v 3.10 -n 6
```

Create keyspace

Switch to the created cluster and launch it:

```
$ ccm switch tranz-cluster
$ ccm start
```

Run `cqlsh` on `node1` with local timezone support:

```
$ TZ=Pacific/Auckland ccm node1 cqlsh
```

Notes:

- Cassandra stores timestamps in UTC format, `cqlsh` converts timezones only if `pytz` python package is installed.
- Please install with `pip install pytz`.
- Launch `cqlsh` with the following command: `TZ=Pacific/Auckland ccm node1 cqlsh`

Create the keyspace using `cql`:

```
CREATE KEYSPACE If NOT EXISTS tranz with replication = {'class': 'NetworkTopologyStrategy', 'datacenter1':
USE tranz;
```

B/ Define tables listed in your answer to question 3 above. For the table definitions include any non default property settings. Optimize your database solution just for iPhone application queries you identified in question 2 above. [15 marks]

Details in `q4.cql`.

Question 5.

Provide CQL3 statements to support each of the application write and update requests you specified in Question 1 above. Show the consistency level before each write and update statement. [20 marks]

Details in `q5.cql`.

Question 6.

Provide CQL3 statements to support each of the application read requests you specified in Question 2 above. Show the consistency level before each read statement. In your answer copy your CQL statement and the result produced by Cassandra from the screen. [29 marks]

Details in `q6.cql`.