

Flight Booking System Using Cassandra

Table of Contents

Definition	2
Project Statement	2
Data Source.....	2
Data Description	3
Source Code Details	4
Software and Libraries.....	4
Environment Set-up	4
Development Environments	5
Methodology	6
Building and Running the Code	6
The Cassandra Keyspace Schema Description.....	7
The Java Code Design	8
Booking System Input and Report Output	9
DML	11
Keyspace snapshot.....	11

Definition

Project Statement

The aim of the project is to build a Cassandra solution for a simplified flight booking system. The solution should answer the following requirements:

- Load into Cassandra data relating to:
 - The airlines unique identifiers and their flight dates.
 - The flights departure/arrival and price.
- Provide a simple user interface to:
 - Allow the user to select a flight given the inbound/return destination, dates.
 - List the flights available for the selected input. The screen should show the departure/return dates/times and price for the destination for each travel legs.
 - Enable the user to select a flight from that list and book the return trip. The booking information requires the all the above information, the travellers' first names and last names and his/her credit card number.
 - For simplicity, it is assumed that:
 - Only one passenger can book a flight at a time.
 - Only one return flight can be booked at a time.
 - The system does not provide validity checks on data entered by the user in the user interface. This means that the user could potentially book a flight that does not exist.
 - All prices are quoted in a unique dummy currency.
 - All departures are only available from 'London Heathrow' and Arrivals are only available at 'Paris Orly'
 - Flight numbers are unique per date

Data Source

The CF1.xls / CF2.xls / CF3.xls / CF4.xls data source files can be found in the 'Flight_Booking_System.zip' file.

Data Description

File Name CF1.xls / CF2.xls / CF3.xls

Excel Column	Name	Description
A	DATE	Unique identifier for a flight
B	FLIGHT_1	The flight list associated to a date.
C	FLIGHT_2	
D	FLIGHT_3	
E	FLIGHT_4	
F	FLIGHT_5	
G	FLIGHT_6	

Sample

	A	B	C	D	E	F	G
1	18/11/2013	SG354	6E452	9W450	6Q452	9W652	
2	19/11/2013	SG355	6E453	9W451	6Q453	9W653	
3	20/11/2013	SG356	6E454	9W452	6Q454	9W654	8F142
4	21/11/2013	SG357	6E455	9W453	6Q455	9W655	8F143
5	22/11/2013	SG358	6E456	9W454	6Q456	9W656	8F144
6	23/11/2013	SG359	6E457	9W455	6Q457	9W657	8F145
7	24/11/2013	SG360	6E458	9W456	6Q458	9W658	8F146

File Name CF4.xls

Excel Column	Name	Description
A	FLIGHT_ID	Unique identifier for a flight
B	DEPARTURE_TIME	The flight departure time
C	ARRIVAL_TIME	The flight arrival time
D	PRICE	The price of flight in the fictitious currency 'XYZ'

Sample

A1					
	A	B	C	D	E
1	SG354	00:00	03:00	17,123	
2	SG355	01:00	04:00	17,123	
3	SG356	02:00	05:00	17,123	
4	SG357	03:00	06:00	17,123	
5	SG358	04:00	07:00	17,123	
6	SG359	05:00	08:00	17,123	
7	SG360	06:00	09:00	17,123	

Source Code Details

The code for this report can be found in the 'Flight_Booking_System.zip' file.

Software and Libraries

The following software need to be installed:

- Ubuntu 12.04
- Eclipse IDE for Java Developers Luna SR2 / JRE 1.7
- Maven Integration for Eclipse 1. 3.1
- Cassandra 1.2.11

Environment Set-up

The code will only work against the versions indicated below. It is not guaranteed to work with any other version of the software.

The VM installation set-up...

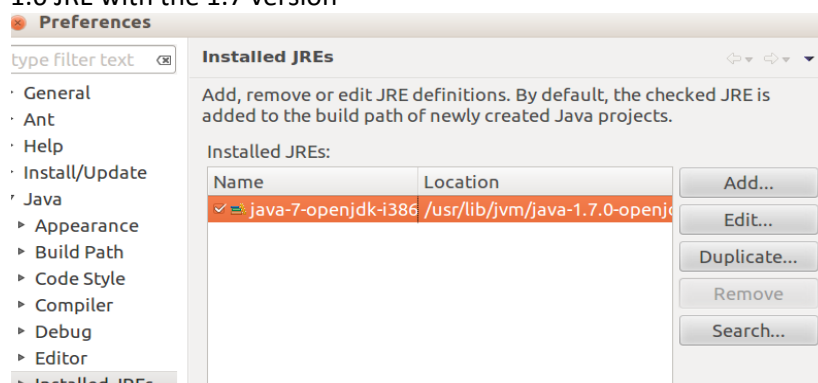
- Install VMware player (<https://www.vmware.com/tryvmware/?p=player>)
- Download Ubuntu 12.04 (<http://www.traffictool.net/vmware/ubuntu1204t.html>)
- Load Ubuntu in VMware (via 'Open a Virtual Machine')
- Change the settings on the Ubuntu and set the Memory = 4GB and Processors = 2 and start the VM

The first steps...

- In a Terminal window, execute the following commands:
 - `sudo apt-get update` (password for user: **password**)
 - `sudo apt-get install openjdk-7-jre-headless`
 - `Java -version` (should be 1.7.025 or above)
- Download Eclipse Luna SR2 (<http://www.eclipse.org/downloads/packages/release/luna/sr2>)
- Create a directory named 'MyProjects' under Home, and copy the Eclipse downloaded 'gz.tar' file into the 'MyProjects' directory

The Eclipse Installation...

- In Terminal window:
 - In the 'MyProjects' directory execute the command: `tar -xvzf Eclipse.tar.gz`
 - Execute the command: `cd Eclipse` and then call `eclipse`
 - The prompt will request for an update as the platform is not update to date. Follow the proposed solution display on the screen (e.g. `sudo apt-get update`)
 - This will take approximately 10-15mins and will turn the `Java -version` to version 1.6. This is expected.
 - Start the Eclipse IDE from a Terminal window -> the 'Indigo' Eclipse IDE version starts instead of 'Luna'. This is expected.
 - The Eclipse IDE loads but it refers the JRE 1.6 . This needs changing.
 - Open the Eclipse IDE
 - Go to Menu => Window => Preferences => Java (left pane) => Installed JREs and replace the 1.6 JRE with the 1.7 version



The Maven light installation ...

- Open Eclipse, go to the menu => Help => Install New Software...
 - In the 'Work With' box input <http://download.eclipse.org/technology/m2e/releases/1.3/1.3.1.20130219-1424>
 - Give it a unique name e.g. m2e
 - Then 'next' and 'finish'

Open the Eclipse project...

- Unzip the 'Flight_Booking_System.zip' into the 'MyProjects' directory.
- Open Eclipse => menu => File => Import => browse to the folder where the '.pom' file resides.
- No compilation error should appear.

Now the Cassandra install...

- Download Cassandra (<https://archive.apache.org/dist/Cassandra/1.2.11/>)
- Move it into the 'MyProjects' folder
- In Terminal window (from the 'MyProjects' folder):
 - `tar -xvzf apache-Cassandra-1.2.11-bin.tar.gz`
 - `mv apache-Cassandra-1.2.11 ~/Cassandra`
 - `sudo mkdir /var/lib/Cassandra`
 - `sudo mkdir /var/log/Cassandra`
 - `sudo chown -R $USER:$GROUP /var/lib/Cassandra`
 - `sudo chown -R $USER:$GROUP /var/log/Cassandra`
 - `export CASSANDRA_HOME=~/Cassandra`
 - `export PATH=$PATH:$CASSANDRA_HOME/bin`
 - `nano ~/Cassandra/conf/Cassandra-env.sh => find -Xss256k and replace to -Xss280k (save and exit)`
 - `sudo sh ~/Cassandra/bin/Cassandra`
 - Cassandra is working correctly if it shows the following:

```
INFO 09:58:13,642 Node localhost/127.0.0.1 state jump to normal
INFO 09:58:13,652 Startup completed! Now serving reads.
```

Development Environments

The CQL/CLI development is performed in both './cqsh' and './Cassandra-cli' command lines (as show in the below screen shot). The Java implementation is performed in the Eclipse environment. Eclipse IDE launcher is available in the Eclipse destination directory mentioned above.

```
user@ubuntu:~/cassandra/bin$ ls
cassandra          cassandra-shuffle.bat  nodetool.bat        sstablescrib
cassandra.bat      cqlsh                 sstable2json        sstablesplit
cassandra-cli      debug-cql             sstable2json.bat    sstableupgrade
cassandra-cli.bat  json2sstable          sstablekeys         stop-server
cassandra.in.sh    json2sstable.bat      sstablekeys.bat
cassandra-shuffle  nodetool              sstableloader
user@ubuntu:~/cassandra/bin$ ./cqlsh
Connected to Test Cluster at localhost:9160.
[cqlsh 3.1.7 | Cassandra 1.2.11 | CQL spec 3.0.0 | Thrift protocol 19.36.1]
Use HELP for help.
cqlsh>
```

Methodology

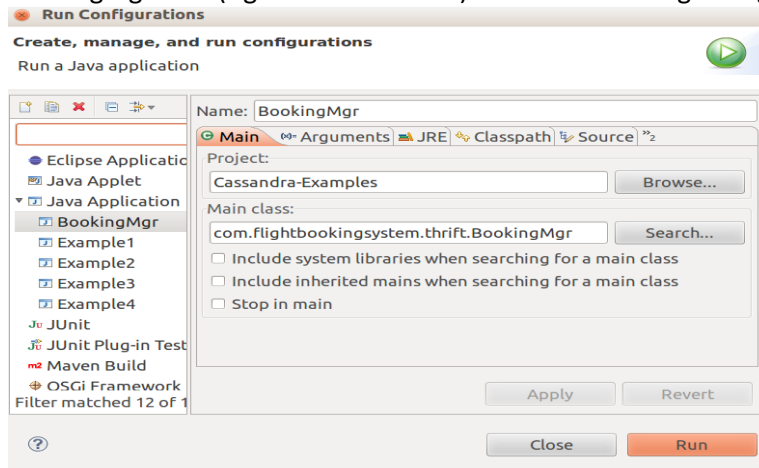
In this document, the terms database/keyspace, table/column family are used interchangeably.

Building and Running the Code

- Prior to running the application, the keyspace 'flight_booking_db' needs to be dropped.
 - Start a cqlsh command (./cqsh)
 - Execute 'drop keyspace flight_booking_db'. If the keyspace does not exist (s it is expected initially), then you will get the error message below. This is correct.

```
cqlsh:flight_booking_db> drop keyspace flight_booking_db;  
Bad Request: Cannot drop non existing keyspace 'flight_booking_db'.
```

- This step is required each time the user runs the application and exits it.
- On successful completion of the installation described in the section 'Environment Set-up', then start the BookingMgr.Java (right click -> Run As) with the following configuration.



The Cassandra Keyspace Schema Description

This schema does not contain 'supercolumn' on purpose (instead composite columns are used). Supercolumns do not perform well and they will be removed for Cassandra in a coming release.

Column Family Name	Description	Type
airports	A static table containing the airports details	
Column Name	Description	Type
iata	The airport unique identifier	varchar
city	The airport city	varchar
country	The airport country	varchar
Name	The airport long name	varchar

Column Family Name	Description	Type
bookings	A static table listing the bookings details	
Column Name	Description	Type
booking_id	The unique id for a passenger booking	varchar
group_id	The unique id when multiple passengers book on the same flight (not in use)	varchar
departure_details	The inbound flight information. A composite comma separated containing the departure airport information, the flight departure/arrival times and the price	varchar
arrival_details	The outbound flight information. A composite comma separated containing the departure airport information, the flight departure/arrival times and the price	varchar
passenger_details	A composite comma separated column containing the first and last name of the passenger	varchar
credit_card_details	The credit card details. A composite comma separated column containing the card type and the long number	varchar
price	The total price of each travel leg	varchar

Column Family Name	Description	Type
flight_details	A flexible table containing the flight departure/arrival date/times and the price.	
Column Name	Description	Type
key	The flight date	varchar
column1	The flight number	varchar
Value	A composite comma separated column containing the departure/arrival times and the travel leg price	varchar

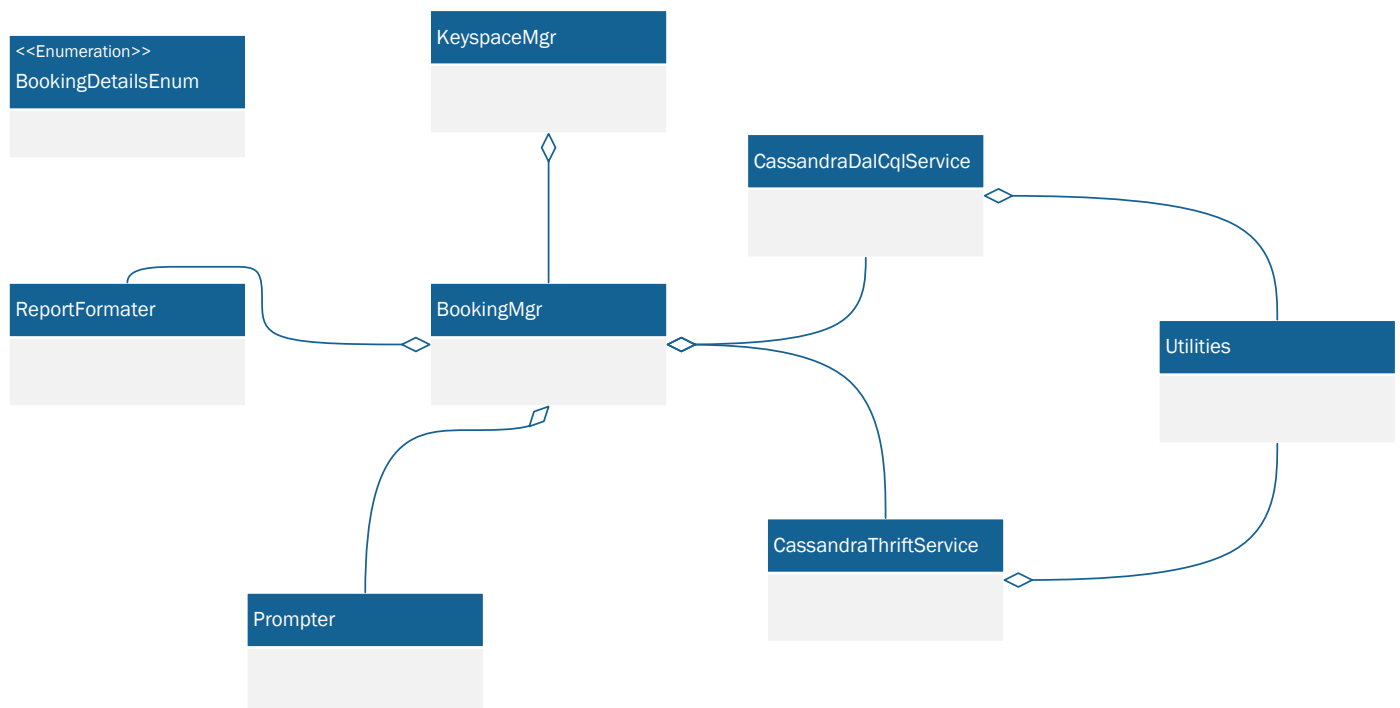
Indexes Names	Description
booking(group_id)	Secondary index required for where clause query on column group_id
booking(credit_card_details)	Secondary index required for where clause query on column credit_car_details

The Java Code Design

The below UML diagram shows the class diagram for the flight booking system. At the center, the *BookingMgr* class. It is responsible for creating the cluster, the keyspace and managing the flight booking process/reporting by delegating into the other specialised class in the system.

- The *BookingMgr* creates new keyspace definition via the *KeyspaceMgr* object.
- The *Prompter* and *ReportFormatter* objects respectively focus on the user interface query/response and the formatting of Cassandra data into the command window.
- The *BookingMgr* communicates with the Cassandra database via the *CassandraDalCqlService* and *CassandraDalThriftService* services. These services specialise in storing/retrieving data from the Cassandra database.
 - The *CassandraDalCqlService* service generates Java embedded DML/DLL for CRUD operations. The constructed queries are executed by the *CqlQuery* object (e.g. see the method *saveBookings()*). It also generates secondary indexes for predicate search (e.g. *where* clause). An example can be found in the method *createBookingIndexes()*.
 - The *CassandraDalThriftService* uses *mutators* to insert data into the Cassandra database (see the method *saveFlightDetails()*). It uses *slice predicates* to retrieve data (see the method *getFlightDetails()*).

There is no real need to have two data access layer types, rather than for testing the technology. In a real life project, it is advised to use one methodology over the other to reduce code complexity.



Booking System Input and Report Output

The booking system enables return flights booking as well as querying existing bookings. The user interface is very simplistic (command line based). The system prompt is displayed with a black font, while the user input is in green. The upgrade of the system to a more modern user interface will be part of another exercise.

Step 1 – On start

The first few lines log the insertion of the initial data in the system. It is then followed by a prompt asking the user to make a choice, as shown below.

```
Start -> createBookingsTable
End -> createBookingsTable
Start -> createBookingsIndexes
End -> createBookingsIndexes
Start -> createAirportsTable
End -> createAirportsTable
Start -> saveBookingsTable
End -> saveBookingsTable
Start -> saveAirportsTable
End -> saveAirportsTable
Start -> saveFlightDetails
End -> saveFlightDetails
*****
Welcome to the Casssandra Flight Booking System!
1. Make a Booking / 2. Check Bookings / 9. Exit Booking (Enter [1] [2] or [9])
*****
1|
```

Step 2 – Make a Booking

The user decides to make a booking (enter [1] in the prompt). The user is then prompted to provide inbound departure date. The system returns the list of available flights for that date. The user makes a selection.

```
*****  
Inbound Date:  
*****  
01-Dec-2013  
*****  
This is the list of available Inbound times & prices:  
*****  
  
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>  
SG367: 13:00,16:00,17123  
9W665: 14:00,15:00,25998  
9W463: 10:00,11:00,25998  
8F153: 13:00,12:35,25998  
6Q465: 13:00,15:00,25998  
6E465: ,,  
  
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>  
  
*****  
Select Flight Details (Inbound)  
*****  
8F153: 13:00,12:35,25998|
```

The same set of questions are then repeated for the return leg of the flight, as shown below:

```
*****  
Outbound Date:  
*****  
03-Dec-2013  
*****  
This is the list of available Outbound times & prices:  
*****  
  
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>  
SG369: 15:00,18:00,17123  
9W667: 16:00,17:00,25998  
9W465: 12:00,13:00,25998  
8F155: 15:00,14:35,25998  
6Q467: 15:00,17:00,25998  
6E467: ,,  
  
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>  
  
*****  
Select Flight Details (Outbound)  
*****  
9W465: 12:00,13:00,25998
```

At the end, the name and credit card of the passenger are requested and the booking is made.

```
*****
Passenger [First Name,Last Name]
*****
John, Papoi
*****
Credit Card Details [Type,Long Number]
*****
AMEX-123456789
Start -> saveBookingsTable
End -> saveBookingsTable
*****
```

The booking is complete... The prompt returns to the main menu

```
*****
Welcome to the Casssandra Flight Booking System!
1. Make a Booking / 2. Check Bookings / 9. Exit Booking (Enter [1] [2] or [9])
*****
```

Step 3 – Viewing Bookings

The user can verify a booking by selection option [2] on the main menu. The user can select between viewing all bookings or filtering them by *credit card details*, *booking id* or *group id*. The below example shows the *booking id* filtering. The same principle applies for the other filters.

```
*****
1. View all bookings / 2. View filtered Bookings (Enter [1] or [2])
*****
2
*****
Filter by: 1. Credit Card Details / 2. Booking Id / 3. Group Id (Enter [1], [2] or [3])
*****
2
*****
Booking Id:
*****
9fd429d2-0b24-4fec-a896-4a4900c26ad2
```

The result appears in the screen (it is not possible to take a full screen shot of the result)

[illegible]

The user can also view all bookings in the system (selecting option [1]). The result is shown below. The first row of the result shows the booking that was made via the prompt. The 3 other bookings come from the preloaded phase (on program start up).

```
*****  
1. View all bookings / 2. View filtered Bookings (Enter [1] or [2])  
*****  
  
1  
  
>>>>>>>>Start Query Result>>>>>>>>>>>>>  
booking_id: 9fd429d2-0b24-4fec-a896-4a4900c26ad2 / group_id: abef1c1e-14ca-412f-8d08-24e3a01d294a / departure_details:  
booking_id: 3 / group_id: 1 / departure_details: ORY,Paris-Orly,Paris,France,28-Aug-2016 18:00:00 / arrival_details: LI  
booking_id: 2 / group_id: 1 / departure_details: ORY,Paris-Orly,Paris,France,28-Aug-2016 18:00:00 / arrival_details: LI  
booking_id: 1 / group_id: 1 / departure_details: ORY,Paris-Orly,Paris,France,28-Aug-2016 18:00:00 / arrival_details: LI  
>>>>>>>>End Querv Result>>>>>>>>>>>>>
```

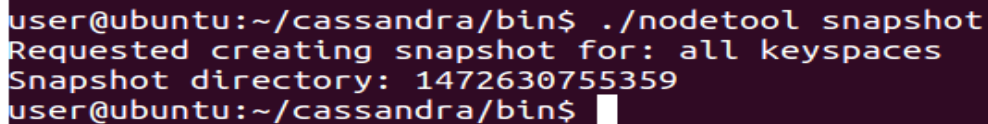
DML

The below DML shows how to perform a batch query against the flight booking keyspace.

```
USE flight_booking_db;
BEGIN BATCH
INSERT INTO bookings (booking_id,group_id, departure_details, arrival_details,
passenger_details,credit_card_details, price) VALUES ('10', '10', 'ORY,Paris-Orly,Paris,France,28-Aug-2016 18:00:00',
'LHR,London Heathrow,London,United Kingdom,30-Aug-2016 19:00:00','John,McKay','AMEX,1234', '100.00')
DELETE FROM bookings WHERE booking_id = '10';
APPLY BATCH;
```

Keyspace snapshot

The flight_booking_db snapshot (for backup) was created with following command line.



```
user@ubuntu:~/cassandra/bin$ ./nodetool snapshot
Requested creating snapshot for: all keyspaces
Snapshot directory: 1472630755359
user@ubuntu:~/cassandra/bin$
```