NDBI040: PRACTICAL CLASS 4

APACHE CASSANDRA

Based on NDBI040 practical class materials created by Martin Svoboda; Tutor: Pavel Čontoš; November 11th 2020

(RECOMMENDED) REQUIREMENTS

- Database concepts
- macOS / Linux command line or PuTTy / WinSCP on Windows

SERVER ACCESS

CONNECT TO NOSQL SERVER

- `ssh` on macOS / Linux
- `PuTTy` on Windows

- `nosql.ms.mff.cuni.cz:42222`
- Login and password send by e-mail
- Change your initial password (if not yet changed) by `passwd`

TRANSFER FILES

- `scp` on macOS / Linux
- `WinSCP` on Windows

DATA MODEL

- `Instance` → `keyspaces` → `tables` → `rows` → `columns`

- **Keyspace**
- **Table (column family)**
  - `Collection of (similar) rows`
  - `Table schema must be specified`, yet can be modified later on
- **Row**
  - `Collection of columns`
  - Rows in a table do not need to have the same columns
  - Each row is `uniquely identified by a primary key`
- **Column**
  - `Name-value pair` + additional data

DATA MODEL: COLUMN VALUES

EMPTY VALUE

- null

ATOMIC VALUE

- `Native data types` such as texts, integers, dates, ...
- `Tuples`
  - Tuple of anonymous field, each of any type (even different)
- `User defined types` (UDT)
  - Set of named fields of any type

COLLECTIONS

- `Lists`, `sets`, and `maps`
  - Nested tuples, UDTs, or collections are allowed, but currently only in frozen mode (such elements are serialized when stored)

CASSANDRA QUERY LANGUAGE (CQL)

DDL STATEMENTS

- `CREATE KEYSPACE` - creates a new keyspace
- `CREATE TABLE` - creates a new table
- ...

DML STATEMENTS

- `SELECT` - selects and projects rows from a single table
- `INSERT` - inserts rows into a table
- `UPDATE` - updates columns of rows in a table
- `DELETE` - removes rows from a table
- ...

CQLSH: FIRST STEPS

START CQLSH SHELL

- `cqlsh`

BASIC USEFUL COMMANDS

- `CLEAR`
  - Clears the terminal window contents
- `EXIT`
- `QUIT`
  - Terminates the current database connection

KEYSPACE

CREATE YOUR PERSONAL KEYSPACE

- `CREATE KEYSPACE login WITH replication = {'class' : 'SimpleStrategy', 'replication_factor' : 3}`
  - Use your login name as a name of your keyspace
    - E.g.: m201_student

LIST ALL EXISTING KEYSPACES

- `DESCRIBE KEYSPACES`

SWITCH TO YOUR KEYSPACE

- `USE login`

EXERCISE 1: TABLES

CREATE A NEW TABLE FOR USERS

- Columns: integer identifier, first name, last name

LIST ALL EXISTING TABLES AND VIEW TABLE DEFINITION

- `DESCRIBE TABLES`
- `DESCRIBE TABLE users`

INSERT NEW USERS INTO THE TABLE OF USERS

- `1, Irena Holubova`
- `2, Pavel Contos`

BROWSE EXISTING USERS

- Find all users
- Find a specific user with identifier `1`

EXERCISE 2: FILTERING

- Try finding a particular user according to their last name
  - `lname = 'Holubova'`
- Try finding a particular user once again
  - Enable filtering
- Create a secondary index for last names
  - `CREATE INDEX ON ...`

EXERCISE 3: DATA TYPES

CREATE A USER-DEFINED TYPE FOR NAMES OF PEOPLE

- `CREATE TYPE ...`
- Fields: first, last

CREATE A NEW TABLE FOR CONTACTS

- Columns
  - `id`: integer identifier
  - `name`: first and last name
  - `address`: triple containing street, city and ZIP code
  - `emails`: set of e-mail addresses
  - `apps`: list of the preferred messenger applications
  - `phones`: map of phone numbers (work, home, ...)

EXERCISE 4: INSERTION

INSERT NEW RECORDS INTO THE TABLE OF CONTACTS

- 1

  Irena Holubova

  Malostranske namesti, Praha, 11800

  holubova@ksi.mff.cuni.cz

  WhatsApp, Messenger

  work +420951554316

- 2

  Pavel Contos

  contos@ksi.mff.cuni.cz, pavel.contos@eli-beams.eu

  iMessage

  work +420999999999, fax +420999333999

EXERCISE 5: UPDATE

MODIFY EXISTING CONTACT RECORDS

- Replace columns of a person with id 1
  - Replace address: Malostranske namesti 25, Praha, 11800
  - Replace applications: Hangouts
- Modify columns of a person with id 1
  - Add new e-mail address: holubova@ksi.mff.cuni.cz
  - Add new applications: Messenger and WhatsApp
  - Add new phone number: home +420123456789
- Modify columns of a person with id 1
  - Remove e-mail address: irena.holubova@mff.cuni.cz
  - Remove applications: Hangouts and Messenger
  - Remove phone number: home

EXERCISE 6: DELETION

- Modify columns of existing contact records
  - Remove / update columns of a person with id 1
    - Remove address column
    - Remove the first application
    - Remove phone number to work

EXERCISE 7: AGGREGATION AND ORDERING

- Create a new table for messages
  - Columns
    - `sender`: integer identifier of a sender
    - `app`: name of a messenger application used
    - `date`: date a given message was sent
    - `time`: time a given message was sent
    - `recipient`: integer identifier of a recipient
    - `message`: message text
  - Primary key involves the following columns
    - `sender`, `app`, `date`, and `time`
  - Columns `sender` and `app` are considered to be partitioning

AGGREGATION AND ORDERING

- Insert the following rows into the table of messages

```
INSERT INTO messages(sender, app, date, time, recipient, message) VALUES (2, 'WhatsApp', '2020-11-10', '10:00:00', 1, 'Hi Irena');

INSERT INTO messages(sender, app, date, time, recipient, message) VALUES (2, 'WhatsApp', '2020-11-10', '10:15:00', 1, 'Are you there?');

INSERT INTO messages(sender, app, date, time, recipient, message) VALUES (2, 'Messenger', '2020-11-10', '11:30:00', 1, 'Are you there?');

INSERT INTO messages(sender, app, date, time, recipient, message) VALUES (1, 'Messenger', '2020-11-10', '11:32:00', 2, 'Yes, I am');

INSERT INTO messages(sender, app, date, time, recipient, message) VALUES (1, 'Messenger', '2020-11-10', '11:33:00', 2, 'How are you?');

INSERT INTO messages(sender, app, date, time, recipient, message) VALUES (2, 'iMessage', '2020-11-11', '11:59:00', 1, 'I am fine');

INSERT INTO messages(sender, app, date, time, recipient, message) VALUES (2, 'iMessage', '2020-11-11', '12:00:00', 1, 'And you?');
```

EXERCISE 8: AGGREGATION AND ORDERING

- Find all messages of a user with id 2 sent using iMessage
  - Order the rows according to dates and times, both in descending order

- Aggregate messages sent by a particular user with id 2
  - Return the overall number of sent messages for each combination of an application name and message date

REFERENCES

- CQL - Cassandra Query Language
  - http://cassandra.apache.org/doc/latest/cql/