

NDBI040: Modern Database Concepts

<http://www.ksi.mff.cuni.cz/~svoboda/courses/191-NDBI040/>

Practical Class 7

Cassandra

Martin Svoboda

svoboda@ksi.mff.cuni.cz

26. 11. 2019

Charles University, Faculty of Mathematics and Physics

Data Model

Database system structure

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 fields, 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)

Query Language

CQL = Cassandra Query Language

- **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
- ...

First Steps

Connect to our NoSQL server

- SSH / SFTP and PuTTY / WinSCP
- **nosql.ms.mff.cuni.cz:42222**

Start CQLSH shell

- `cqlsh`

Basic useful commands

- **CLEAR**
 - Clear the shell terminal window
- **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

List all existing keyspaces

- DESCRIBE KEYSACES

Switch to your keyspace

- USE `login`

Tables

Create a new table for users

- Columns: integer identifier, first name, last name

View table definition

- `DESCRIBE TABLE users`

Insert new users into the table of users

- 1, Irena, Holubova
- 2, Martin, Svoboda

Browse existing users

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

Filtering

Try to find a particular user according to their last name

- `lname = 'Holubova'`

Create a secondary index for last names

- `CREATE INDEX ON ...`

Try to find a particular user once again

- Enable filtering

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 of street, city and ZIP code
 - emails: set of e-mail addresses
 - apps: list of preferred messenger applications
 - phones: map of phone numbers (work, home, ...)

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
Martin Svoboda
svoboda@ksi.mff.cuni.cz, martin.svoboda@mff.cuni.cz
Viber, WhatsApp
work +420951554250, fax +420951554323

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

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

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', '2017-11-27', '10:00:00', 1, 'Hi Irena');
INSERT INTO messages (sender, app, date, time, recipient, message)
  VALUES (2, 'WhatsApp', '2017-11-27', '10:15:00', 1, 'Are you there?');
INSERT INTO messages (sender, app, date, time, recipient, message)
  VALUES (2, 'Messenger', '2017-11-27', '10:30:00', 1, 'Are you there?');
INSERT INTO messages (sender, app, date, time, recipient, message)
  VALUES (1, 'WhatsApp', '2017-11-27', '10:45:00', 2, 'Yes, I am');
```

Aggregation and Ordering

Find all messages of a user with id 2 sent using *whatsapp*

- 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/>