B4M36DS2, BE4M36DS2: **Database Systems 2**

https://cw.fel.cvut.cz/b231/courses/b4m36ds2/

Practical Class 7

Cassandra

Yuliia Prokop
prokoyul@fel.cvut.cz

6. 11. 2022

Author: Martin Svoboda
(martin.svoboda@matfyz.cuni.cz)

Czech Technical University in Prague, Faculty of Electrical Engineering

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.felk.cvut.cz

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.: f221_student

```
cqlsh> CREATE KEYSPACE f221_student
   ... WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 3};
```

Keyspace

List all existing keyspaces

- `DESCRIBE KEYSPACES`

```
cqlsh> DESCRIBE KEYSPACES

f221_student  system_auth         system_schema  system_views          yuliia
system        system_distributed  system_traces  system_virtual_schema
```

Switch to your keyspace

- `USE login`

```
cqlsh> USE f221_student;
```

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`

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`

```
cqlsh:f221_student> CREATE TABLE users(
                ... id INT PRIMARY KEY,
                ... fname TEXT,
                ... lname TEXT
                ... );
cqlsh:f221_student> DESCRIBE TABLES

users
```

Tables

View table definition

- `DESCRIBE TABLE users`

```
cqlsh:f221_student> DESCRIBE TABLE users

CREATE TABLE f221_student.users (
    id int PRIMARY KEY,
    fname text,
    lname text
) WITH additional_write_policy = '99p'
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND cdc = false
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy'
, 'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '16', 'class': 'org.apache.cassandra.io.compress.LZ
4Compressor'}
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    AND extensions = {}
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair = 'BLOCKING'
    AND speculative_retry = '99p';
```

Tables

Insert new users into the table of users

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

Tables

Insert new users into the table of users

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

```
cqlsh:f221_student> INSERT INTO users (id, fname, lname)
                ... VALUES (1, 'Irena', 'Holubova');
cqlsh:f221_student> INSERT INTO users (fname, lname, id)
                ... VALUES ('Martin', 'Svoboda', 2);
```

Tables

Browse existing users

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

Tables

Browse existing users

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

```
cqlsh:f221_student> SELECT * FROM users;

 id | fname   | lname
----+---------+----------
  1 |  Irena | Holubova
  2 | Martin |  Svoboda

(2 rows)
cqlsh:f221_student> SELECT * FROM users WHERE (id = 1);

 id | fname | lname
----+-------+----------
  1 | Irena | Holubova

(1 rows)
```

Filtering

Try to find a particular user according to their last name

- `lname = 'Holubova'`

Try to find a particular user once again

- Enable filtering

Create a secondary index for last names

- `CREATE INDEX ON …`

Filtering

Try to find a particular user according to their last name

- `lname = 'Holubova'`

```
cqlsh:f221_student> SELECT * FROM users
                ... WHERE (lname = 'Holubova');
InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this query
as it might involve data filtering and thus may have unpredictable performance. If you want to e
xecute this query despite the performance unpredictability, use ALLOW FILTERING"
```

```
cqlsh:f221_student> SELECT * FROM users
                ... WHERE (lname = 'Holubova')
                ... ALLOW FILTERING;

 id | fname | lname
----+-------+----------
  1 | Irena | Holubova

(1 rows)
```

Filtering

Create a secondary index for last names

- `CREATE INDEX ON ...`

```
cqlsh:f221_student> CREATE INDEX ON users (lname);
```

Data Types

Create a user-defined type for names of people

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

```
cqlsh:f221_student> CREATE TYPE person(
                   ... first TEXT,
                   ... last TEXT
                   ... );
```

Data Types

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, …)

```
cqlsh:f221_student> CREATE TABLE contacts(
                ... id INT,
                ... name person,
                ... address TUPLE<TEXT, TEXT, INT>,
                ... emails SET<TEXT>,
                ... apps LIST<TEXT>,
                ... phones MAP<TEXT, TEXT>,
                ... PRIMARY KEY (id)
                ... );
```

Data Types

```
cqlsh:f221_student> DESCRIBE TABLE contacts

CREATE TABLE f221_student.contacts (
    id int PRIMARY KEY,
    address frozen<tuple<text, text, int>>,
    apps list<text>,
    emails set<text>,
    name person,
    phones map<text, text>
) WITH additional_write_policy = '99p'
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND cdc = false
    AND comment = ''
    AND compaction = {'class':
'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy',
'max_threshold': '32', 'min_threshold': '4'}
    AND compression = {'chunk_length_in_kb': '16', 'class':
'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    AND extensions = {}
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair = 'BLOCKING'
    AND speculative_retry = '99p';
```

Cassandra is not able to directly access the individual components of thing which is wrapped by frozen

Insertion

Insert new records into the table of contacts

- 1
  Irena Holubova
  Malostranske namesti, Praha, 11800
  holubova@felk.cvut.cz
  WhatsApp, Messenger
  work +420951554316
- 2
  Martin Svoboda
  svoboda@ felk.cvut.cz, martin.svoboda@mff.cuni.cz
  Viber, WhatsApp
  work +420951554250, fax +420951554323

Insertion

```
cqlsh:f221_student> INSERT INTO contacts (id, name, address, emails,
apps, phones)
        ... VALUES (
        ... 1,
        ... {first: 'Irena', last: 'Holubova'},
        ... ('Malostranske namesti', 'Praha', 11800),
        ... {'holubova@felk.cvut.cz'},
        ... ['WhatsApp', 'Messenger'],
        ... {'work' : '+420951554316'}
        ... );
cqlsh:f221_student> INSERT INTO contacts (id, name, emails, apps, phones)
        ... VALUES (
        ... 2,
        ... {first: 'Martin', last: 'Svoboda'},
        ... {'svoboda@felk.cvut.cz', 'martin.svoboda@mff.cuni.cz'},
        ... ['Viber', 'WhatsApp'],
        ... {'work' : '+420951554250', 'fax' : '+420951554323'}
        ... );
```

INT
name (UDT)
TUPLE
SET
LIST
MAP

Update

Modify existing contact records

- Replace columns of a person with id *1*
  - Replace address: Malostranske namesti 25, Praha, 11800
  - Replace applications: Hangouts

Update

Modify existing contact records

- Replace columns of a person with id *1*
  - Replace address: Malostranske namesti 25, Praha, 11800
  - Replace applications: Hangouts

```
cqlsh:f221_student> UPDATE contacts
        ... SET
        ... address = ('Malostranske namesti 25', 'Praha', 11800),
        ... apps = ['Hangouts']
        ... WHERE (id = 1);
```

```
 id | address                                            | apps                  |
 emails                                                       | name
                  | phones
----+--------------------------------------------------+----------------------+
---------------------------------------------------------------+------------------
-----------------+---------------------------------------------------------
  1 | ('Malostranske namesti 25', 'Praha', 11800) |          ['Hangouts'] |
                                    {'holubova@felk.cvut.cz'} | {first: 'Irena',
last: 'Holubova'} |                           {'work': '+420951554316'}
```

Update

Modify existing contact records

- 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

Update

Modify existing contact records

- 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

```
cqlsh:f221_student> UPDATE contacts
          ... SET
          ... emails = emails + {'holubova@ksi.mff.cuni.cz'},
          ... apps = ['Messenger', 'WatsApp'] + apps,
          ... phones = phones + {'home' : '+420123456789'}
          ... WHERE (id = 1);
```

```
 id | address                                          | apps                                        |
 emails                                                       | name
   | phones
----+--------------------------------------------------+---------------------------------------------+
--------------------------------------------------------------+------------------------------------------
---+--------------------------------------------------------
  1 | ('Malostranske namesti 25', 'Praha', 11800) | ['Messenger', 'WatsApp', 'Hangouts'] |
  {'holubova@felk.cvut.cz', 'holubova@ksi.mff.cuni.cz'} | {first: 'Irena', last: 'Holubova
'} | {'home': '+420123456789', 'work': '+420951554316'}
```

Update

Modify existing contact records

- Replace columns of a person with id *1*
- Modify columns of a person with id *1*
  - Remove e-mail address: irena.holubova@felk.cvut.cz
  - Remove applications: Hangouts and Messenger
  - Remove phone number: home

Update

Modify existing contact records

- Replace columns of a person with id *1*
- Modify columns of a person with id *1*
  - Remove e-mail address: irena.holubova@felk.cvut.cz
  - Remove applications: Hangouts and Messenger
  - Remove phone number: home

```
cqlsh:f221_student> UPDATE contacts
                ... SET
                ... emails = emails - {'irena.holubova@felk.cvut.cz'},
                ... apps = apps - ['Hangouts', 'Messenger'],
                ... phones = phones - {'home'}
                ... WHERE (id = 1);
```

```
 id | address                                          | apps        | emails
                            | name                                    | phones
----+--------------------------------------------------+-------------+-----------------------
----------------------------+-----------------------------------------+-----------------------
-----
  1 | ('Malostranske namesti 25', 'Praha', 11800) | ['WatsApp'] | {'holubova@felk.cvut.cz'
, 'holubova@ksi.mff.cuni.cz'} | {first: 'Irena', last: 'Holubova'} | {'work': '+4209515543
16'}
```

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

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

```
cqlsh:f221_student> DELETE
                ... address,
                ... apps[0],
                ... phones['work']
                ... FROM contacts
                ... WHERE (id = 1);
```

```
 id | address | apps | emails
| name                                | phones
----+---------+------+-------------------------------------------------
+-----------------------------------+--------
  1 |    null | null | {'holubova@felk.cvut.cz', 'holubova@ksi.mff.cuni.cz'}
| {first: 'Irena', last: 'Holubova'} |   null
```

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

```
cqlsh:f221_student> CREATE TABLE messages (
                ... sender INT,
                ... app TEXT,
                ... date DATE,
                ... time TIME,
                ... recipient INT,
                ... message TEXT,
                ... PRIMARY KEY ((sender, app), date, time)
                ... );
```

Partition key (these two will be used when determining particular nodes in our cluster that will be responsible for storing the individual replicas)

Clustering columns

Order is important

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, 'Messenger', '2017-11-27', '10:45:00', 2, 'Yes, I am');
INSERT INTO messages (sender, app, date, time, recipient, message)
  VALUES (1, 'Messenger', '2017-11-27', '10:50:00', 2, 'How are you?');
 INSERT INTO messages (sender, app, date, time, recipient, message)
   VALUES (2, 'Viber', '2017-11-28', '18:01:00', 1, 'I am fine');
 INSERT INTO messages (sender, app, date, time, recipient, message)
   VALUES (2, 'Viber', '2017-11-28', '18:02:00', 1, 'And you?');
```

Aggregation and Ordering

```
cqlsh:f221_student> INSERT INTO messages (sender, app, date, time, recipient, message)
                ... VALUES (2, 'WhatsApp', '2017-11-27', '10:00:00', 1, 'Hi Irena');
cqlsh:f221_student> INSERT INTO messages (sender, app, date, time, recipient, message)
                ... VALUES (2, 'WhatsApp', '2017-11-27', '10:15:00', 1, 'Are you there?');
cqlsh:f221_student> INSERT INTO messages (sender, app, date, time, recipient, message)
                ... VALUES (2, 'Messenger', '2017-11-27', '10:30:00', 1, 'Are you there?');
cqlsh:f221_student> INSERT INTO messages (sender, app, date, time, recipient, message)
                ... VALUES (1, 'Messenger', '2017-11-27', '10:45:00', 2, 'Yes, I am');
cqlsh:f221_student> INSERT INTO messages (sender, app, date, time, recipient, message)
                ... VALUES (1, 'Messenger', '2017-11-27', '10:50:00', 2, 'How are you?');
cqlsh:f221_student> INSERT INTO messages (sender, app, date, time, recipient, message)
                ... VALUES (2, 'Viber', '2017-11-28', '18:01:00', 1, 'I am fine');
cqlsh:f221_student> INSERT INTO messages (sender, app, date, time, recipient, message)
                ... VALUES (2, 'Viber', '2017-11-28', '18:02:00', 1, 'And you?');
cqlsh:f221_student> SELECT * FROM messages;

 sender | app       | date       | time                 | message         | recipient
--------+-----------+------------+----------------------+-----------------+-----------
      2 |  WhatsApp | 2017-11-27 | 10:00:00.000000000 |       Hi Irena |         1
      2 |  WhatsApp | 2017-11-27 | 10:15:00.000000000 | Are you there? |         1
      1 | Messenger | 2017-11-27 | 10:45:00.000000000 |      Yes, I am |         2
      1 | Messenger | 2017-11-27 | 10:50:00.000000000 |   How are you? |         2
      2 | Messenger | 2017-11-27 | 10:30:00.000000000 | Are you there? |         1
      2 |     Viber | 2017-11-28 | 18:01:00.000000000 |      I am fine |         1
      2 |     Viber | 2017-11-28 | 18:02:00.000000000 |       And you? |         1
```

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

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

```
cqlsh:f221_student> SELECT *
                ... FROM messages
                ... WHERE sender = 2 AND app = 'WhatsApp'
                ... ORDER BY date DESC, time DESC;
```

Partition key

Only **clastering keys** can be used for sorting according to their order

sender	app	date	time	message	recipient
2	WhatsApp	2017-11-27	10:15:00.000000000	Are you there?	1
2	WhatsApp	2017-11-27	10:00:00.000000000	Hi Irena	1

Aggregation and Ordering

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

Aggregation and Ordering

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

```
cqlsh:f221_student> SELECT sender, app, date, COUNT(*)
                ... FROM messages
                ... WHERE sender = 2
                ... GROUP BY sender, app, date
                ... ALLOW FILTERING;
```

sender	app	date	count
2	WhatsApp	2017-11-27	2
2	Messenger	2017-11-27	1
2	Viber	2017-11-28	2

References

CQL – Cassandra Query Language

- http://cassandra.apache.org/doc/latest/cql/