ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Technische Universität Berlin

Prof. Dr. Stefan Tai
Marco Peise

Your name:_________________

Enterprise Computing (WS 2016/17)

Exercise 5 (3 Portfoliopunkte)

Info:

- The solution to this exercise must be handed in by Wednesday, Dez 07th 2016, 12AM.
- Any written solution must be in an accessible PDF. Any source code in a separate ZIP File. All Files are uploaded in the Information System for Instructors and Students in a single ZIP File (https://isis.tu-berlin.de/course/view.php?id=8586).
- Please write your name on the solution sheet.

Task 1 – Cassandra Query Language (5+10+5+5+5+10+10 = 50%)

a) Keyspaces

Does the following CQL3 statement create a new keyspace that places one replica in each of two data centers?

```
cqlsh> CREATE KEYSPACE <ksname>
  WITH REPLICATION = {'class':'NetworkTopologyStrategy','DC1':1,'DC2':1};
```

b) Compound primary keys

Which statement creates a table "messages" in the keyspace "msgplatform" that is partitioned by "participant_id" across multiple servers and locally clustered by "posted_on" date and "posted_location" string?

c) Add a collection to a table

You want to alter an existing "customers" table schema with an additional column that can contain sets of "interests" (text strings). After you have altered the table, a query like this should work:

```
cqlsh:msgplatform> UPDATE customers SET interests = interests + {'climbing','baking'}
WHERE id = 202;
```

Please complete the following uncomplete statement using correct SQL syntax:

```
cqlsh:msgplatform> ALTER TABLE customers ADD ___________
```

d) TTL

Is it true or false? The following statement inserts an expiring column named "pw_reset_token" with value "fgh-lmn-456" into the "customers" table. The column "pw_reset_token" will be deleted automatically after 6000 seconds but the customer row with id 315 and name "karlmann" will still be present in the table after 6000 seconds.

```
cqlsh:msgplatform> INSERT INTO customers (id, name,
  pw_reset_token)
  VALUES (315, 'karlmann', 'fgh-lmn-456')
  USING TTL 12000;
```

e) Counter columns

You have created a table named "referendum" with two counter columns like this:

```
cqlsh:msgplatform> CREATE TABLE referendum (
  participant_id int,
  msg_created_time timeuuid,
  msg_created_location string,
  voteinfavor counter,
  voteoppose counter,
  PRIMARY KEY (participant_id, msg_created_time, )
  );
```

Complete the following query that increments the number of in favor votes for a referendum which are created in "germany", between the first and second of January 2016 (midnight-midnight) and from the participant with the number "519":

cqlsh:msgplatform> UPDATE referendum SET____

f) Lightweight transactions with Compare-and-Set (CAS)

Write two CQL queries that implement the following feature:

1. A user named Catarina (user id: “689”) wants to reset her password by creating an expiring token ("pw_reset_token") column in her user row in the “customers” table.

2. She gets a message with a link to a site where to reset her password. On that site she can enter a new password only if the expiring token (encoded as parameter into the link in her message) matches the expiring token in her user row and if the token has not expired, yet.

f) True or false?

Only one node in a P2P data cluster (such as a Cassandra cluster) maintains a sliding window of inter-arrival times of Gossip messages.

Task 2 – MongoDB – Modeling a Scenario (50%)

You are the Lead Developer for a new sub system to be implemented in MongoDB. The Platform as a Service should be able to create Invoices for Energy Customers in Germany. Therefore, the system should be able to store information about:

- Contracts made with Energy Customers (PPA - Power Purchase Agreements) [unique contract number, startDate, previousMeterNumber]
- Customers [unique customer number, company, firstName, lastName, email]
- Customers Contact Information [street, streetNumber, zip, city]
- Customers basic Bank Information [IBAN, BIC, BankName]
- Tariffs which are being used within PPA's [unique tariff name, netBasePrice, VAT, EEGLevy]
- Instalments which are being made by the Energy Customer up front [startDate, grossValue, payDayInMonth]
- Meter Information for every PPA [unique meter number]
- Meter Reading for every Meter [readingDate, meterValue]

A Customer can have many PPA's, many Addresses, many Bank Account Information.

One PPA must include at least one Customer, may include many Instalments being made from a Customer and must contain one Meter with maybe many Readings.

Download and install MongoDB from https://www.mongodb.com/download-center - community and run the according command lines in MongoShell (Database: "testing") to obtain the following JSON Structure in MongoDB:

```
{
  _id: <ObjectId01>,
  startDate: „<YYYY-mm-dd>“,
  previousMeterNumber: 123456789,
  instalments: [{
                    startDate: „<YYYY-mm-dd>“,
                    grossValue: 39
                  },
                  {
                    startDate: „<YYYY-mm-dd>“,
                    grossValue: 40
                  }
  ],
  meter: [{
             readingDate: „<YYYY-mm-dd>“,
             meterValue: 20
           },
           {
            readingDate: „<YYYY-mm-dd>“,
            meterValue: 40
           },
           {
            readingDate: „<YYYY-mm-dd>“,
            meterValue: 60
           }
  ],
  customer: {
                 company: „Smith Inc.“,
                 firstName: „John“,
                 lastName: „Smith“,
                 email: „john@smith.com“,
                 address: [{
                               street: „UperStreet“,
                               streetNumber: 15,
                               zip: 10411,
                               city: Berlin
                             },
                             {
                               street: „LowerStreet“,
                               streetNumber: 19,
                               zip: 10413,
                               city: Berlin
                             }
                 ],
                 bankAccounts: [{
                                     IBAN: 123456789,
                                     BIC: 112233445,
                                     bankName: BankOfEngland,
                                   },
                                   {
                                     IBAN: 123456799,
                                     BIC: 552233445,
                                     bankName: BankOfScottland,
                                   }
                 ]
  },
  tariff: {
            netBasePrice: 4.99,
            VAT: 19,
            EEGLevy: 6.354
  }
}
```

The only Commands you can use are

DB.<COLLECTION.**save**(...);

DB.<COLLECTION.**update**({...},{"**$set**"...});

DB.<COLLECTION.**update**({...},{"**$push**"...});

Fill in the commands which are missing to complete the provided JSON structure above. Please keep in Mind that the provided commands are in order executed:

1. db.testing.save({startDate:new Date()});
2. ...
3. db.testing.update({„_id“:ObjectId(„123“)},{„$set“: {„meter“: []}});
4. ...
5. ...
6. ...
7. ...
8. ...
9. ...
10. db.testing.update({„_id“:ObjectId(„123“)},{„$set“: {„customer“: {company: „Smith Inc.“,email: „john@smith.com“}}});
11. ...
12. ...
13. ...
14. db.testing.update({„_id“:ObjectId(„123“)},{„$set“: {„customer.bankAccounts“: []}});
15. ...
16. ...
17. db.testing.update({„_id“:ObjectId(„123“)},{„$set“: {„tariff“: {netBasePrice: 4.99,VAT:19,EEGLevy:6.354}}});