INFO-H-415 : Advanced Database
NoSQL Databases (1)

Lecturer : Esteban Zimanyi
Teaching-Assistants : François Picalausa, Serge Boucher
http://cs.ulb.ac.be/public/teaching/infoh415

Academic Year 2010–2011

Queries

Following is a database structure for a message board:

```
users: [ {
  nickname: string,
  email: string
} ]
threads: [ {
  topic: string,
  board: string,
  size: integer,
  messages: [ {
      author: string (refers to an user email),
      content: string
  } ]
} ]
```

Explain the following queries:

- `db.users.findOne("nickname": "esteban", "email": 1)`
- ```
  db.users.find().map(function (user) {
            var numPosts = db.threads.find({"messages.author": user.email}).count();
            return {name: user.nickname, num: numPosts};
        }).filter(function (user) {
            return user.num > 1;
        }).map(function (user) {
            return user.name;
        })
  ```

Write the following queries:

- Find every registered user.
- Find the thread with a topic of "INFO-H-415" in the "Polytech" board.
- Find the topic of the threads in which "ezimanyi@ulb.ac.be" posted, ordered by number of messages.
- Find the threads with at least one reply, and where no message has been posted by "ezimanyi@ulb.ac.be"
- Find the number of registered users.
  ```

- Find the name of boards containing threads with more than 2 messages.
- **Supplemental:** Find the topic of each thread, along with the nickname of each poster.
- Register a new user: "Serge Boucher" (sboucher@ulb.ac.be). He will be a super user, denoted by the addition of a "superuser" field.
- Add a new message to the "Polytech board", in the "INFO-H-415" thread, by Serge Boucher saying "Hello NoSQL World!". Increment the size field at the same time.

Design considerations

Consider the following alternative structures for the data of exercise 1:

- ```
  threads: [ {
      topic,
      board,
      size,
      messages: [ {
        author: {\bf {nickname, email}},
        content
      } ]
  } ]
  ```
- ```
  users: [ { nickname,  email } ]
  boards: [ {
      name,
      threads: [ {
          topic,
          size,
          messages: [ {
                  author [refers to users.email]
                  content
          } ]
      } ]
  } ]
  ```
- ```
  users: [ {  nickname, email } ]
  boards: [ { name } ]
  threads: [ {
      board [refers to boards._id],
      topic
  } ]
  messages: [ {
      thread [refers to threads._id],
      date,
      author,
      content
  } ]
  ```
- What is the granularity of each of these models (i.e. what kind of objects can be retrieved) ?
- What are the pros and cons of these data models (1) in general, and (2) for the queries of Exercise 1?
  ```

Solutions

Queries

```
function myprint(ex, query) {
    print("Result of exercise " + ex + "\n");

    if (query.forEach) {
        query.forEach(function (q) { print(tojson(q)); });
    }
    else print(tojson(query));
}

myprint(1, db.users.find());
myprint(2, db.threads.findOne({topic: "INFO-H-415", board: "Polytech"}));
myprint(3, db.threads.find({ "messages.author": "ezimanyi@ulb.ac.be" },
                           { topic: 1 }).sort({size: 1}));
myprint(4, db.threads.find({ "messages.author": {$nin: ["ezimanyi@ulb.ac.be"]},
                             "size": {$gte: 2}}).sort({size: 1}));
myprint(5, db.users.count());
myprint(6, db.threads.distinct( "board" , { size: { $gt: 2 }}));

function posterNick(email) {
    var user = db.users.findOne({"email": email}, {"nickname": 1});
    return user ? user.nickname : email;
}
function q7(item) {
    var poster = item.messages[0].author;
    /* Note that posterNick will be invoked for each thread,
     * effectively performing a join */
    return {topic: item.topic, poster: posterNick(poster)};
}
myprint(7, db.threads.find(null, {"topic": 1, "messages": 1}).map(q7));

/* Updates: note that the schema is not predefined */
db.users.insert({nickname: "Serge Boucher",
    email: "sboucher@ulb.ac.be", superuser: true});
db.threads.update({topic: "INFO-H-415", board: "Polytech"},
    { $inc: {"size": 1},
      $push: { messages: { author: "sboucher@ulb.ulb.ac.be",
                           content: "Hello NoSQL World!"}}});
```

Design Considerations

For each model, only the top level elements can be retrieved (although projections can be used to filter out information). Note that specific items from arrays cannot be retrieved.

To check whether a query matches or not a document, it may be necessary to load the whole document. Hence storing every threads inside of board objects may be expensive. Also, it prevents indexing individual threads on, e.g. their title. On the other hand, keeping only smaller documents creates a need for additional joins, which must be performed outside the dbms and can also be expensive. The first alternative schema also proposes to store each user information multiple time. While this may lead to inconsistencies, there is also a gain from not having to retrieve additional information, e.g. to display a thread.

As a conclusion, the design should focus on answering efficiently the most common queries. I.e., returning the exact amount of information needed, with indexes on the appropriate search fields.