Сравнение реляционных и нереляционных концепций на примерах MySQL и MongoDB

Студент 3 курса МОиАИС А.В. Лапшин

декабрь 2012

Содержание

1 О нереляционных СУБД

1.1 Нереляционная модель данных или NoSQL-подход

NoSQL (англ. not only SQL, не только SQL), термин, обозначающий ряд подходов или проектов, направленных на реализацию моделей баз данных, имеющих существенные отличия от используемых в традиционных реляционных СУБД с доступом к данным средствами языка SQL. Описание схемы данных в случае использования NoSQL-решений может осуществляться через использование различных структур данных: хеш-таблиц, деревьев и других.

Сторонниками концепции NoSQL подчёркивается, что она не является полным отрицанием языка SQL и реляционной модели, проект исходит из того, что SQL — это важный и весьма полезный инструмент, но при этом он не может считаться универсальным. Одной из проблем, которую указывают для классических реляционных БД, являются проблемы при работе с данными очень большого объема и в проектах с высокой нагрузкой. Основная цель подхода — расширить возможности БД там, где SQL недостаточно гибок, и не вытеснять его там, где он справляется со своими задачами.

По мнению сайта nosql-database.org, в основе идеи лежат:

1. Нереляционная модель данных;
2. Открытый исходный код;
3. Хорошая горизонтальная масштабируемость.

В качестве одного из методологических обоснований подхода NoSQL используется эвристический принцип, известный как теорема CAP (известная так же как теорема Брюера), утверждающий, что в распределённой системе невозможно одновременно обеспечить согласованность данных, доступность (в смысле наличия отклика по любому запросу) и устойчивость к расщеплению распределённой системы на изолированные части. Таким образом, при необходимости достижения высокой доступности и устойчивости к разделению предполагается не фокусироваться на средствах обеспечения согласованности данных, обеспечиваемых традиционными SQL-ориентированными СУБД с транзакционными механизмами на принципах ACID(**АСИД**: **А**томарность, **С**огласованность, **И**золированность, **Д**олговечность). Концепция NoSQL включает себя несколько реализаций: Поколоночные СУБД (Hadoop,Cassandra), Документоориентированные СУБД (CouchDB,MongoDB), Хранилища «ключ-значение», кортежные хранилища (Redis, MEMBASE), Базы данных на основе графов (OrientDB) и другие. [3]

1.2 Сравнение концепций SQL и NoSQL на примере

Сравниваться концепции будут на примерах реляционной и широко известной MySQL от Oracle и документоориентированной MongoDB, широко известной нереляционной современной СУБД от 10gen.
О самой СУБД MongoDB можно прочесть ниже, в разделе 2.1.

1. MongoDB — концептуально то же самое, что обычная, привычная нам база данных (или в терминологии Oracle — схема). Внутри MongoDB может быть ноль или более баз данных, каждая из которых является контейнером для прочих сущностей.
2. База данных может иметь ноль или более «коллекций». Коллекция настолько похожа на традиционную «таблицу», что можно смело считать их одним и тем же.
3. Коллекции состоят из нуля или более «документов». Опять же, документ можно рассматривать как «строку».
4. Документ состоит из одного или более «полей», которые — как можно догадаться — подобны «колонкам».

5. «Индексы» в MongoDB почти идентичны таковым в реляционных базах данных.

6. «Курсоры» отличаются от предыдущих пяти концепций, но они очень важны (хотя порой их обходят вниманием) и заслуживают отдельного обсуждения. Важно понимать, что когда мы запрашиваем у MongoDB какие-либо данные, то она возвращает курсор, с которыми мы можем делать все что угодно — подсчитывать, пропускать определённое число предшествующих записей — при этом не загружая сами данные.

Подводя итог, MongoDB состоит из «баз данных», которые состоят из «коллекций». «Коллекции» состоят из «документов». Каждый «документ» состоит из «полей». «Коллекции» могут быть проиндексированы, что улучшает производительность выборки и сортировки. И наконец, получение данных из MongoDB сводится к получению «курсора», который отдаёт эти данные по мере надобности. Однако, хотя термины, хоть и близки своим «реляционным» аналогам, но не полностью идентичны им. Основное различие в том, что реляционные базы данных определяют «колонки» на уровне «таблицы», в то время как документ-ориентированные базы данных определяют «поля» на уровне «документа». Это значит, что любой документ внутри коллекции может иметь свой собственный уникальный набор полей. В этом смысле коллекция «глупее» чем таблица, тогда как документ имеет намного больше информации, чем строка. [1]

Небольшое сравнение синтаксиса двух языков (примечание: здесь в качестве языка MongoDB выступает нативный JavaScript). В MySQL **создание таблицы(схемы)** выглядит приблизительно так:

```
CREATE TABLE users (
    id MEDIUMINT NOT NULL
        AUTO_INCREMENT,
    user_id Varchar(30),
    age Number,
    status char(1),
    PRIMARY KEY (id)
)
```

А в MongoDB:

```
db.users.insert( {
    user_id: "abc123",
    age: 55,
    status: "A"
 } )
```

или

```
db.createCollection("users")
```

Притом, в случае с более длинным запросом, СУБД не просто создаст коллекцию, но также и документ с заполненными значениями полей.

Удаление схемы или коллекции

Mysql:

```
DROP TABLE users
```

MongoDB:

```
db.users.drop()
```

Запросы на выборку всех записей из users

Mysql:

```
SELECT *
FROM users
```

MongoDB:

```
db.users.find()
```

Запросы на выборку записей из users с определёнными колонками/полями.

Mysql:

```
SELECT id, user_id, status
FROM users
```

MongoDB:

```
db.users.find(
    { },
    { user_id: 1, status: 1 }
)
```

Запросы на выборку записей из users где статус равен A или возраст равен 50

Mysql:

```
SELECT *
FROM users
WHERE status = "A"
OR age = 50
```

MongoDB:

```
db.users.find(
    { $or: [ { status: "A" } ,
             { age: 50 } ] }
)
```

Запросы на выборку записей из users где статус равен A и упорядочиванием по убыванию индексов

Mysql:

```
SELECT *
FROM users
WHERE status = "A"
ORDER BY user_id DESC
```

MongoDB:

```
db.users.find( { status: "A" } ).sort( { user_id: -1 } )
```

Запросы на выборку записей из users, где возраст превышает 30

Mysql:

```
SELECT COUNT(*)
FROM users
WHERE age > 30
```

MongoDB:

```
db.users.count( { age: { $gt: 30 } } )
```

или:

```
db.users.find( { age: { $gt: 30 } } ).count()
```

Обновление записей, изменение статуса на C у тех записей, где возраст превышает 25
Mysql:

```
UPDATE users
SET status = "C"
WHERE age > 25
```

MongoDB:

```
db.users.update(
   { age: { $gt: 25 } },
   { $set: { status: "C" } },
   { multi: true }
)
```

Обновление записей, увеличение возраста на 3 где статус A
Mysql:

```
UPDATE users
SET age = age + 3
WHERE status = "A"
```

MongoDB:

```
db.users.update(
   { status: "A" } ,
   { $inc: { age: 3 } },
   { multi: true }
)
```

Удаление записей где статус D
Mysql:

```
DELETE FROM users
WHERE status = "D"
```

MongoDB:

```
db.users.remove( { status: "D" } )
```

Удаление записей
Mysql:

```
DELETE FROM users
```

MongoDB:

```
db.users.remove( )
```

Стоит также сказать несолько слов об индексации.

В данной СУБД, если не задано вручную, всем элементам коллекций автоматически присваивается уникальный 12-байтный идентификатор (идентификаторы являются **мультиключами**), который генерируется по принципу:

Байты 0-3: временная метка (секунд с начала эпохи);

Байты 4-6: id аппаратной платформы (компьютера);

Байты 7,8: id процесса;

Байты 9-11: счётчик.

Таким образом, диапазон адресов закончится очень не скоро.

2 СУБД 'MongoDB'

2.1 Описание СУБД

О разнице в терминологии между MySQL и MongoDB было сказано выше, в подразделе 1.2.

MongoDB - документо-ориентированная система управления базами данных с открытым исходным кодом, не требующая описания схемы таблиц. Она написана на языке C++ и распространяется в рамках лицензии Creative Commons. СУБД - разработка компании 10gen, которая в июне 2012 года начала длительное сотрудничество с корпорацией Microsoft, предоставив MongoDB ее облаку Windows Azure.

При разработке авторы исходили из необходимости специализации баз данных, благодаря чему им удалось отойти от принципа «один размер подо всё». За счёт минимизации семантики для работы с транзакциями появляется возможность решения целого ряда проблем, связанных с недостатком производительности, причём горизонтальное масштабирование становится проще. Используемая модель документов хранения данных JSON/BSON (JavaScript Object Notation/Binary JavaScript Object Notation, форматы обмена данными) проще кодируется, проще управляется (в том числе за счёт применения бессхемного стиля, а внутренняя группировка релевантных данных обеспечивает дополнительный выигрыш в быстродействии. Нереляционный подход весьма удобен для создания баз данных, у которых горизонтальное масштабирование подразумевает разворачивание на множестве машин. Возможность обеспечивать наилучшую производительность должна существовать параллельно с поддержкой более обширной функциональности, чем это позволяет использование пар «ключ-значение» (в чистом виде). Технология баз данных должна работать везде, начиная с серверов пользователя и виртуальных машин и заканчивая облачными технологиями.

MongoDB, по мнению разработчиков, должна заполнить разрыв между простыми хранилищами данных типа «ключ-значение» (быстрыми и легко масштабируемыми) и большими РСУБД (со структурными схемами и мощными запросами).

СУБД управляет наборами JSON-подобных документов, хранимых в двоичном виде в формате BSON.

Основные возможности данной СУБД:

- Документо-ориентированное хранилище (простая и мощная JSON-подобная схема данных)
- Достаточно гибкий язык для формирования запросов
- Динамические запросы
- Полная поддержка индексов
- Профилирование запросов
- Быстрые обновления «на месте»
- Эффективное хранение двоичных данных больших объёмов, напр., фото и видео
- Журналирование операций, модифицирующих данные в БД
- Поддержка отказоустойчивости и масштабируемости: асинхронная репликация, набор реплик и шардинг
- Может работать в соответствии с парадигмой MapReduce

Имеется подробная и качественная документация, большое число примеров и драйверов под популярные языки Java, Node.js, C++, C#, PHP, Python, Perl, Ruby, Grails&Groovy

Первая релизная версия за номером 0.9.3 выпущена 25.09.2009г.[4]

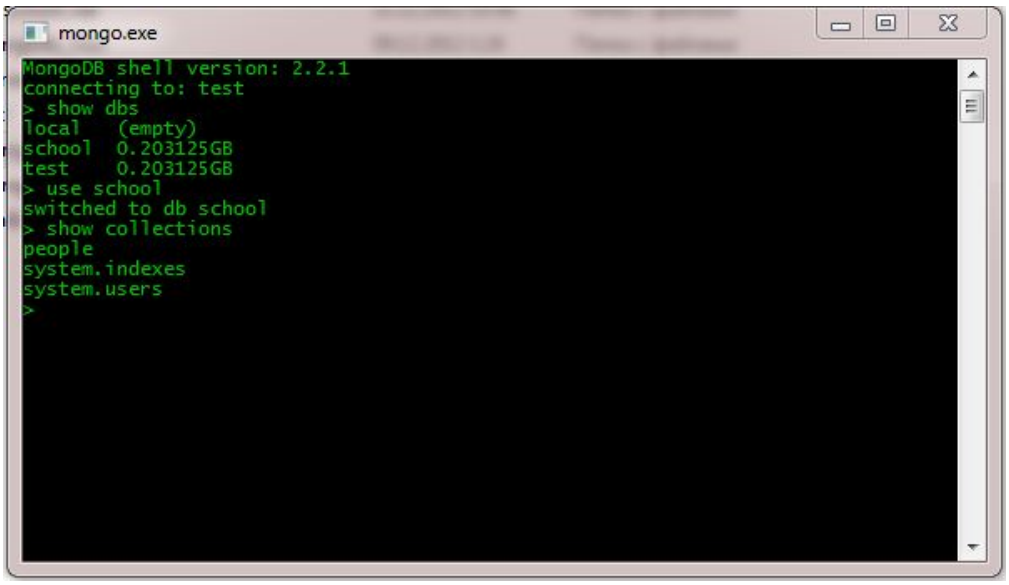


Рис. 1: консоль командной строки с нативным JS

2.2 Установка и настройка mongoDB

Разработка ведётся в ОС Windows 7, поэтому приведено описание настройки СУБД в в данном семействе ОС.

В данном разделе описан способ, основанный на руководстве с официального сайта разработчика mongoDB, в разделе документации[2].

С официального репозитория проекта `http://www.mongodb.org/downloads` была скачана и распакована соответствующая версия mongoDB. Для дальнейшей разработки в папке СУБД были созданы поддиректории \data\db\, где будут располагаться все данные о базах данных. После этого, есть два способа запуска сервера.

1. Запуск через mongod.exe (командная строка/bat-файл)
   Например:

   ```
   d:\mongodb\bin\mongod.exe --dbpath d:\mongodb\data\db\
   ```

2. Настройка и запуск службы.

Рассмотрим подробнее второй случай.

Во-первых, для запуска службы необходима какая-нибудь созданная бд. Для этого придётся запустить единожды сервер mongod.exe (см. выше пункт 1), затем запустить консоль mongo.exe и добавить какие-нибудь значения, например:

```
> db.test.save( { a: 1 } )
```

После этого, закрываем консоль и сервер, создаём директорию logs в папке СУБД и открываем с правами администратора командную строку Windows. Пишем в файле mongod.cfg, который должен располагаться в корневом каталоге СУБД:

```
logpath=d:\mongodb\log\mongo.log
```

А затем, в консоли:

```
d:\mongodb\bin\mongod.exe --config d:\mongodb\mongod.cfg
--dbpath d:\mongodb\data\db --install
```

После этих операций запускаем службу через панель управления ->
администрирование-> службы-> MongoDB или командой в консоли:

```
net start mongodb
```

Примечение: В случае, когда служба настроена некорректно или требуется её удалить, требуется её сначала остановить (через службы или net stop) и использовать:

```
sc delete mongodb
```

или:

```
C:\mongodb\bin\mongod.exe --remove
```

2.3 Разграничение доступа

По умолчанию, MongoDB не использует какие-либо способы авторизации для доступа к базе данных. Сами разработчики MongoDB объясняют это тем, что всю логику должно содержать в себе приложение, а база должна делать то, для чего она лучше всего и предназначена – хранение и управление данными.

В SQL базах имеется возможность хранить множество пользователей, групп и схем для более тонкой настройки прав доступа к данным. Но если не рассматривать MongoDB только со стороны философии, которую они пропагандируют, то можно найти прекрасный механизм авторизации.

MongoDB позволяет управлять пользователями на уровне базы, но правда только в "read/write"или "readonly"режимах.
Создание пользователя с правами "read/write":

```
use mydatabase
db.addUser("admin", "Sup3rG00dP@azzword")
```

Создание пользователя с правами "readonly":

```
db.addUser("web", "prettyGoodPass", true)
```

Параметр true как раз и задает readonly права для создаваемого пользователя.
Для смены пароля необходимо еще раз вызвать addUser

```
db.addUser("web", "wayGooderPass", true)
```

Для удаления пользователя необходимо удалить соответсвующий документ в system.users коллекции

```
db.system.users.remove({"user" : "web"});
```

Если есть желание полностью закрыть неавторизованный доступ, то требуется изменение конфигурации MongoDB:

```
auth = true
```

После этих изменений требуется перезапуск демона.
При попытке получить доступ с данными несуществующего пользователя, то вы получите ошибку вида:

```
error: { ‘‘$err’’ : ‘‘unauthorized for db [mydatabase] lock type: -1 ’’ }
```

Также readonly пользователь при попытке записи получит ошибку:

```
unauthorized
```

Описание взято с источника [6].

3 Сравнение на примере баз данных

3.1 Модель Базы Данных

Для сравнения двух технологий были разработаны небольшие и в большинстве совпадающие модели баз данных на каждом из языков. Структуры представлены на рис.2 и рис.3 для MySQL и MongoDB соответственно. Предметная область – спортивная школа, объекты – тренеры-судья.

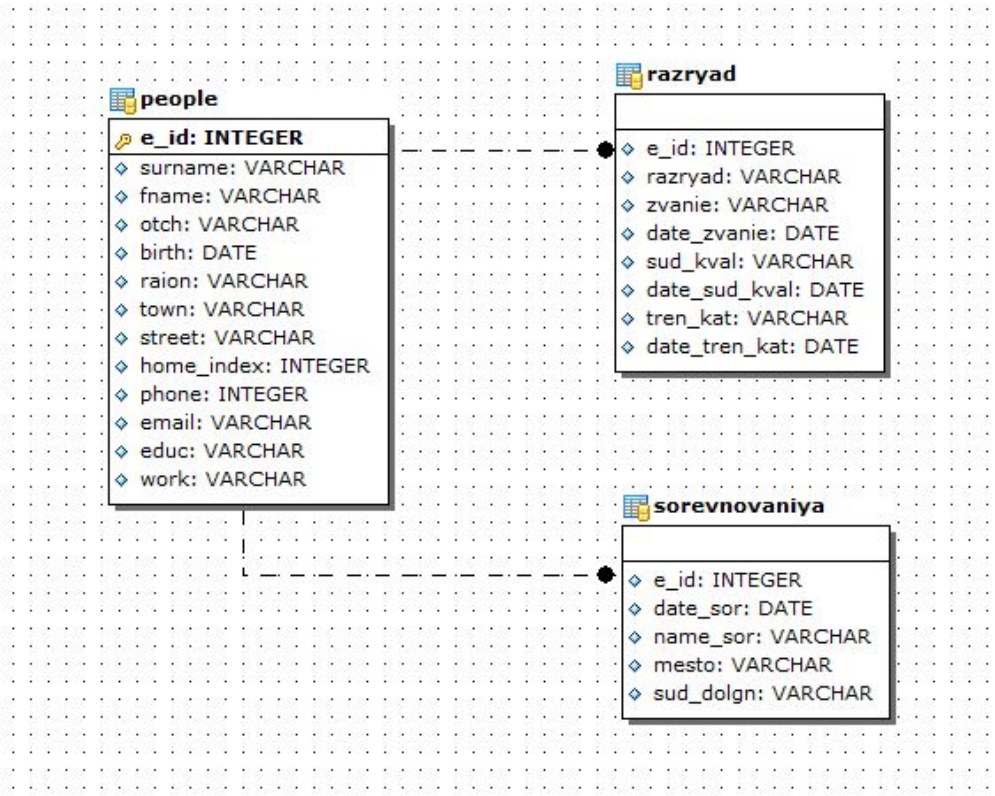


Рис. 2: модель базы данных на языке MySQL

Различие моделей в том, что MySQL не поддерживает структуру, подобную MongoDB, в связи с чем возникла необходимость разбивать общую структуру на три таблицы.

Примечание: такие модели для соответствующих структур взяты не случайно, именно возможности или недостатки определённой концепции дают те тестовые данные, которые необходимы для этого сравнения, а не идентичность структур. Хотя и эти показатели могут быть протестированы, но не здесь.

Общим ключом для обеих таблиц в базе данных на основе MySQL является первое поле таблицы *people*. В базе данных на основе MongoDB единый идентификатор – автоматически сгенерированный id, а роль дополнительных таблиц-коллекций выполняют поля, содержащие в себе сущности, похожие на коллекции, только без идентификаторов, за счёт чего планируется достигнуть бóльшей производительности, хотя и сама MongoDB позиционируется как СУБД с лучшей производительностью и скоростью работы, нежели классические реляционные СУБД.

Разработка базы данных MySQL проходила на версии сервера 5.5, MongoDB – 2.2.2

3.2 Разработка GUI для баз данных

В качестве основного инструмента разработки оболочек был выбран язык C#, в качестве рабочей среды - Microsoft Visual Studio C# 2010 Express. Для взаимодействия с базами данных были также задействованы некоторые драйверы:

```
people:
{
  "_id" ,
  "fname" ,
  "sname" ,
  "DateOfBirth" ,
  "town" ,
  "street" ",
  "education" ,
  "qualifications" : [{
      "category" ,
      "category_date" ,
      "judge_category" ,
      "judge_category_date" ,
      "trainer_category" ,
      "trainer_category_date"
    }],
  "_events" : [{
      "title" ,
      "place" ,
      "date" ,
      "judge_post"
    }]
}
```

Рис. 3: модель базы данных в виде JSON

- MySQL .Net Connector, входящий в комплект установщика сервера MySQL;
- Драйвер для более удобной работы с базой данных MySQL, взятый с http://kbss.ru/blog/lang_c_sharp/96.html;
- CSharpDriver 1.7 для MongoDB.

Примечание: В отличии от драйверов MySQL, CSharpDriver не является ddl-драйвером, а предоставляет полноценную возможность использовать язык, отличный от нативного, что, как предполагается, позволяет теснее взаимодействовать с СУБД и самой базой данных, а также должно улучшать производительность. Как показали более поздние сравнительные тесты, это на самом деле частично зависит от подобной интеграции.

В конечном итоге, разработанные оболочки позволяют добавлять, удалять, обновлять записи, осуществлять поиск по некоторым ключевым полям, есть возможность экспорта в электронные таблицы Microsoft Office Excel, а также добавлен некоторый тестовый алгоритм, позволяющий провести серии тестов с определённым числом проходов и возвращающий в итоге некоторое время.

3.3 Сравнение времени тестов

Для сравнения времени выполнения подключения, добавления, удаления и вывода информации были разработаны серии тестов, позволяющие производить эти операции с замером времени выполнения.

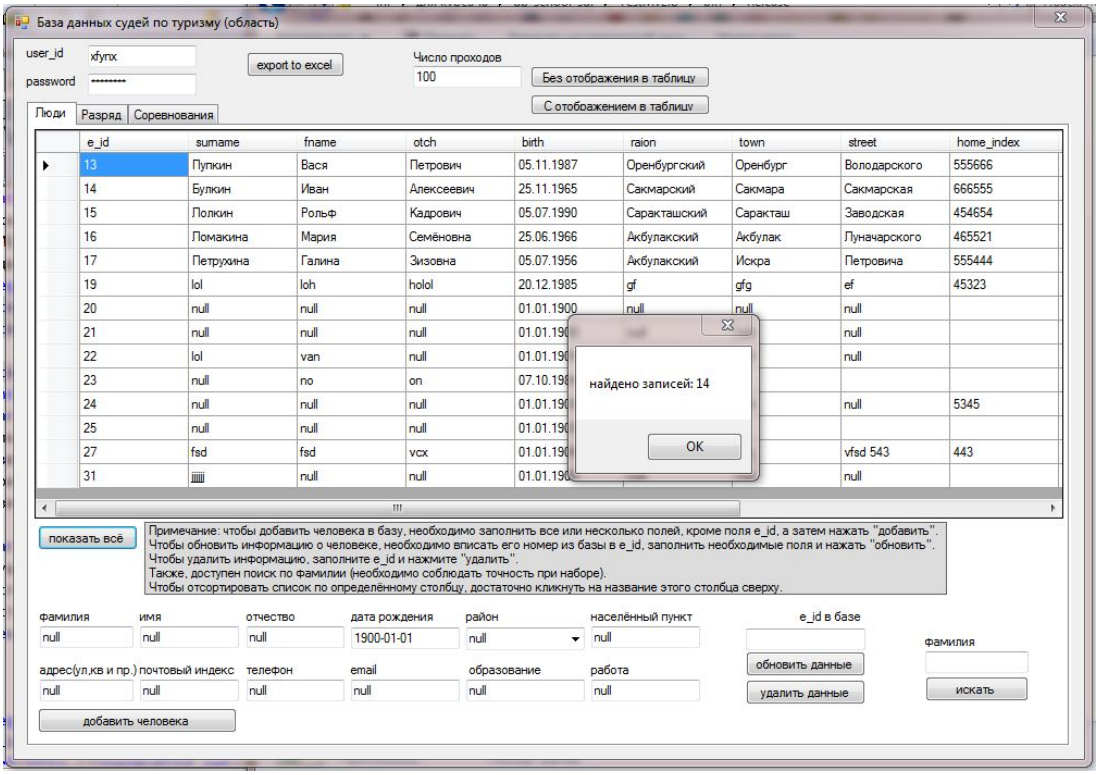


Рис. 4: Оболочка базы данных, использующей СУБД MySQL

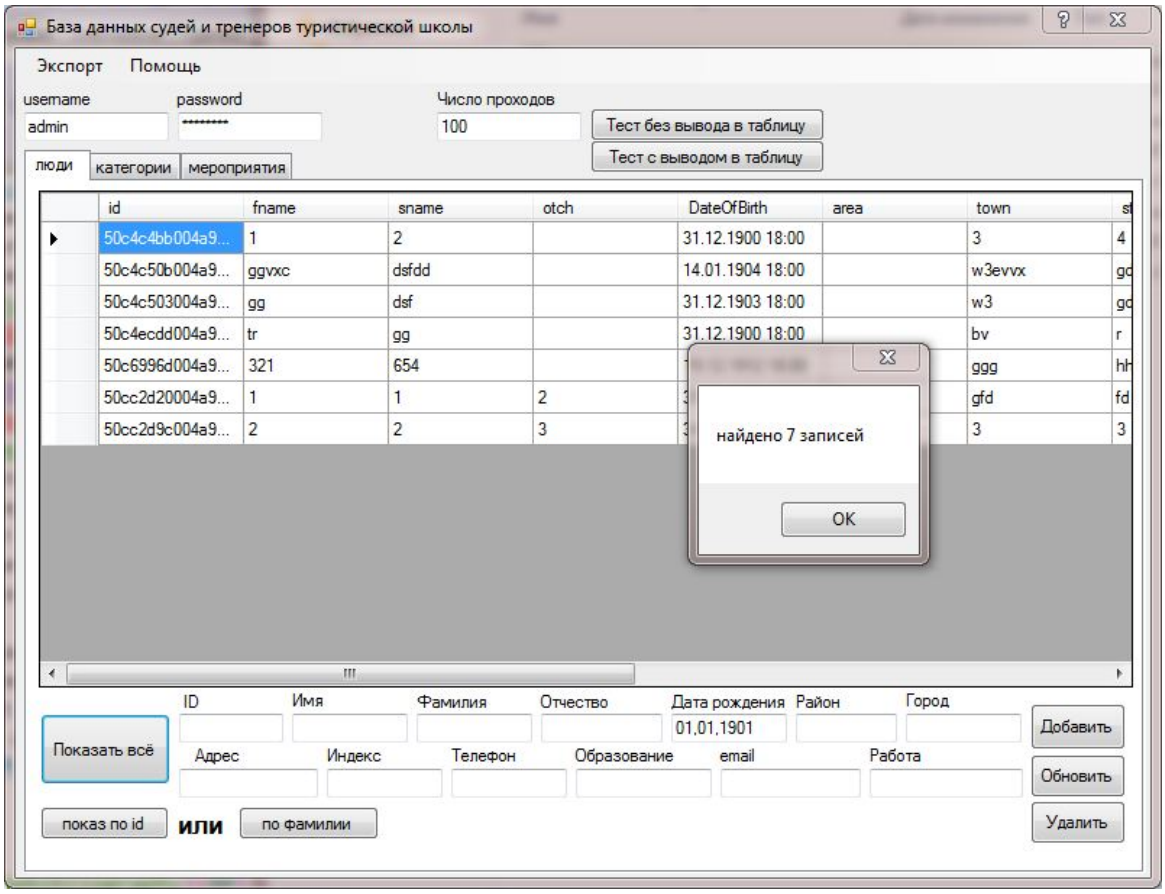
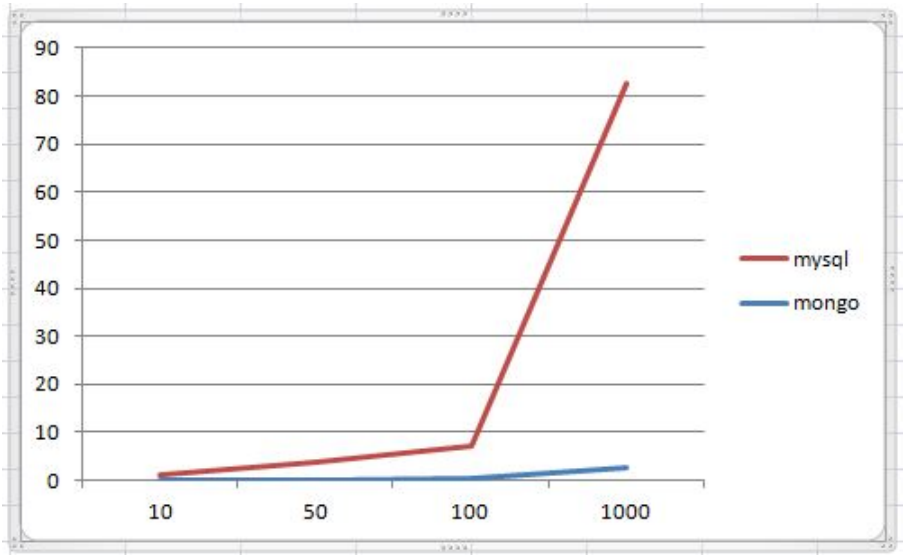
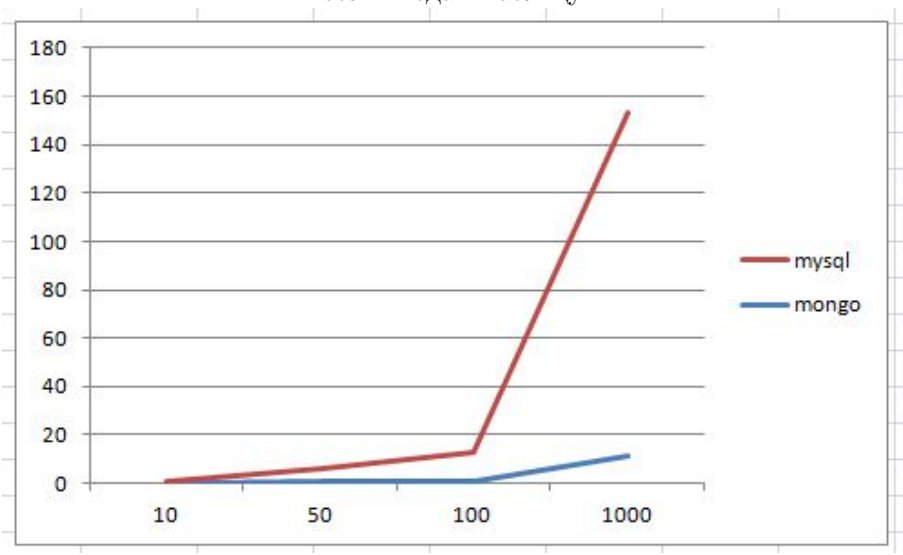


Рис. 5: Оболочка базы данных, использующей СУБД MongoDB

Время выполнения тестов приведено на рис.6. Графики отчетливо показывают, насколько работа



без вывода в таблицу



с выводом в таблицу

Рис. 6: Сравнение скорости выполнения базовых операций

mongo быстрее (по вертикальным шкалам приведено время в секундах), в среднем, в 10 раз. На мой взгляд, это обусловлено более совершенным механизмом самой СУБД, лучшим взаимодействием с языком разработки и с тем, что не требуется каждый раз подключаться/отключаться, что наблюдается в MySQL и SQL-СУБД.

4 Заключение

В результате, MongoDB — это высокопроизводительная система, которая отлично подходит для проектов с высокой динамикой обновления базы. MongoDB в большинстве случаев способна стать заменой реляционной базе данных. Она намного проще и понятнее, быстрее работает и имеет меньше ограничений для разработчиков приложений, однако и она имеет ряд ограничений, поэтому к выбору СУБД нужно подходить исходя из требований разрабатываемого проекта.

В отличии от классических реляционных подходов, занесённых в стандарт и широко зарекомендовавших себя в массе проектов в течении многих лет, NoSQL – современный, динамичный и только-только развивающийся концепт, создающийся не только из необходимости, но ещё и интереса к поиску новых подходов. Возможно, в результате этого развития, СУБД и вся концепция хранилищ данных будет обновлена и развита лучше, нежели есть сейчас, но пока об этом сложно судить.

Список литературы

[1] Seguin K. The Little MongoDB Book [Electronic resource] — 2011 — Access mode: http://openmymind.net/2011/3/28/The-Little-MongoDB-Book/

[2] 10gen Inc. install mongodb on Windows [Electronic resource] — 2012 — Access mode: http://docs.mongodb.org/manual/tutorial/install-mongodb-on-windows/

[3] Wikipedia NoSQL [Electronic resource] — 13.12.2012 — Access mode: http://ru.wikipedia.org/wiki/NoSQL

[4] Wikipedia NoSQL [Electronic resource] — 10.12.2012 — Access mode: http://ru.wikipedia.org/wiki/MongoDB

[5] Redmond E., Wilson J.R. Seven databases in seven weeks — USA, Pragmatic Programmers, LLC., 2012 — P. 135-165.

[6] it addict Авторизация в MongoDB [Электронный ресурс] — 15.12.2012 — Режим доступа: http://www.itaddict.ru/2011/03/mongodb.html