Hacettepe University
Department of Computer Engineering
BBM 471 Database Management Systems
Experiment

Subject	NoSQL Databases - MongoDB
Submission Date	13.04.2017
Due Date	04.05.2017 13:00
Programming Environment	Platform independent (Java/C#/C++)

Background

The relational database has been the foundation of enterprise applications for decades, and since MySQL's release in 1995 it has been a popular and inexpensive option. Yet with the explosion in the volume and variety of data in recent years, non-relational database technologies like MongoDB have emerged to address the requirements of new applications. MongoDB is used for new applications as well as to augment or replace existing relational infrastructure.

What is MySQL?

MySQL is a popular open-source relational database management system (RDBMS) that is developed, distributed and supported by Oracle Corporation. Like other relational systems, MySQL stores data in tables and uses structured query language (SQL) for database access. In MySQL, you pre-define your database schema based on your requirements and set up rules to govern the relationships between fields in your tables. In MySQL, related information may be stored in separate tables, but associated through the use of joins. In this way, data duplication is minimized.

What is MongoDB?

MongoDB is an open-source database developed by MongoDB, Inc. MongoDB stores data in JSON-like documents that can vary in structure. Related information is stored together for fast query access through the MongoDB query language. MongoDB uses dynamic schemas, meaning that you can create records without first defining the structure, such as the fields or the types of their values. You can change the structure of records (which we call documents) simply by adding new fields or deleting existing ones. This data model give you the ability to represent hierarchical relationships, to store arrays, and other more complex structures easily. Documents in a collection need not have an

identical set of fields and denormalization of data is common. MongoDB was also designed with high availability and scalability in mind, and includes out-of-the-box replication and auto-sharding.

Terminology and Concepts

Many concepts in MySQL have close analogs in MongoDB. This table outlines some of the common concepts in each system.

MySQL	MongoDB
Table	Collection
Row	Document
Column	Field
Joins	Embedded documents, linking

Feature Comparison

Like MySQL, MongoDB offers a rich set of features and functionality far beyond those offered in simple key-value stores. MongoDB has a query language, highly-functional secondary indexes (including text search and geospatial), a powerful aggregation framework for data analysis, and more. With MongoDB you can also make use of these features across more diverse data types than with a relational database, and at scale.

	MySQL	MongoDB
Rich Data Model	No	Yes
Dynamic Schema	No	Yes
Typed Data	Yes	Yes
Data Locality	No	Yes
Field Updates	Yes	Yes
Easy for Programmers	No	Yes

Complex Transactions	Yes	No
Auditing	Yes	Yes
Auto-Sharding	No	Yes

Query Language

Both MySQL and MongoDB have a rich query language. A comprehensive list of statements can be found in the MongoDB documentation.

MySQL	MongoDB
INSERT INTO users (user_id, age, status) VALUES ('bcd001', 45, 'A')	db.users.insert({ user_id: 'bcd001', age: 45, status: 'A' })
SELECT * FROM users	db.users.find()
UPDATE users SET status = 'C' WHERE age > 25	db.users.update({ age: { $gt: 25 } }, { $set: { status: 'C' } }, { multi: true })

Why use MongoDB instead of MySQL?

Organizations of all sizes are adopting MongoDB because it enables them to build applications faster, handle highly diverse data types, and manage applications more efficiently at scale.

Development is simplified as MongoDB documents map naturally to modern, object-oriented programming languages. Using MongoDB removes the complex object-relational mapping (ORM) layer that translates objects in code to relational tables.

MongoDB's flexible data model also means that your database schema can evolve with business requirements. For example, schema changes that took days of weeks in The Weather Channel's MySQL databases could be made in just hours with MongoDB.
MongoDB can also be scaled within and across multiple distributed data centers, providing new levels of availability and scalability previously unachievable with relational databases like MySQL. As your deployments grow in terms of data volume and throughput, MongoDB scales easily with no downtime, and without changing your application. In contrast, to achieve scale with MySQL often requires significant, custom engineering work. Baidu migrated from MySQL to MongoDB to support its rapidly growing business. The Chinese internet services giant now powers over 100 apps and manages in excess of 1PB of data with its MongoDB cluster.

What are common use cases for MongoDB?

MongoDB is a general purpose database that is used for a variety of use cases. The most common use cases for MongoDB include Single View, Internet of Things, Mobile, Real-Time Analytics, Personalization, Catalog, and Content Management.

When would MySQL be a better fit?

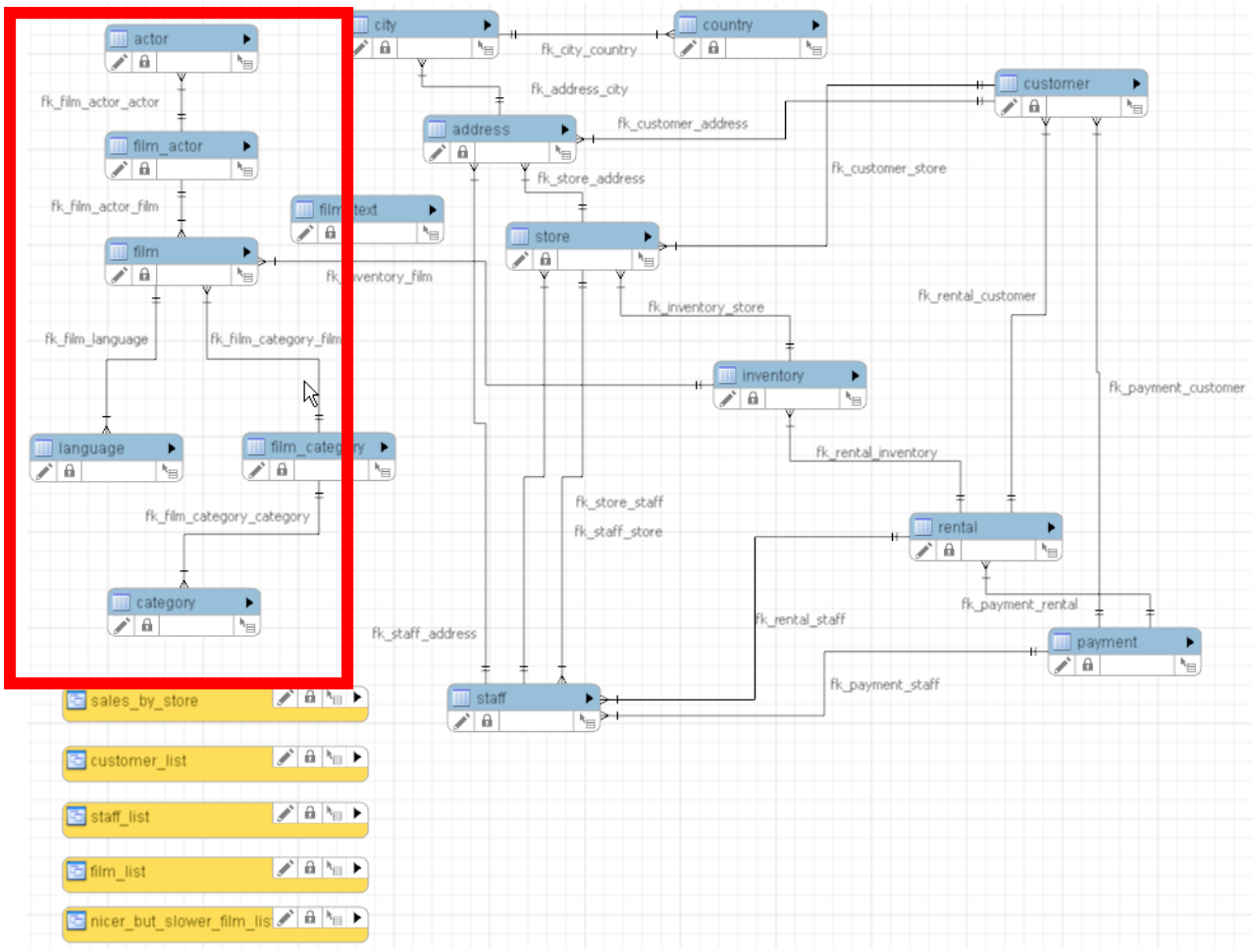
While most modern applications require a flexible, scalable system like MongoDB, there are use cases for which a relational database like MySQL would be better suited. Applications that require complex, multi-row transactions (e.g., a double-entry bookkeeping system) would be good examples. MongoDB is not a drop-in replacement for legacy applications built around the relational data model and SQL.

A concrete example would be the booking engine behind a travel reservation system, which also typically involves complex transactions. While the core booking engine might run on MySQL, those parts of the app that engage with users – serving up content, integrating with social networks, managing sessions – would be better placed in MongoDB

Experiment

In this experiment you are given a sample database Sakila depicted in Figure. The Sakila sample database is designed to represent a DVD rental store. The Sakila[1] sample database still borrows film and actor names from the Dell sample database. You can download Sakila DB from https://github.com/datacharmer/test_db/tree/master/sakila . In this database you will use these tables: **Actor, FilmActor, Film, Language, FilmCategory, Category.**

MySQL Sakila Sample DB Structure



In this experiment, you are expected to develop a desktop application accessing to a relational database system (You are free to choose a RDBMS in this list: MySQL, SQL Server, PostgreSQL) and MongoDB database system working on this sample database

[1] https://dev.mysql.com/doc/sakila/en/

with the following requirements. Please read them carefully before design and implementation.

- Film table holds 1000 records. First of all, you must increase the row number to more than 5.000.000 records by randomly picking the field values and appending to table again by considering unique **film_id** key. You should also do this step for the other tables. (Hint: Develop an application to achieve this) Do not duplicate the same rows. Instead, prefer an original random selection method for generating new records.
- Then, according to NoSQL database paradigm you must create your MongoDB database. This stage requires your research about NoSQL and MongoDB. Note that NoSQL databases are differing than relational ones in terms of database design and usage.
- You must **develop a visual application for querying by film.title, film.release_year, actor.first_name, actor.last_name, category.name.** Users can select arbitrary numbers of filter. In other words, user can filter records by all or only one filter. However, field name will be used only once.

  Your application must enable users to connect relational DB and Mongo DB and user should have the ability to define query terms via text boxes and drop down lists (Dropdown lists will hold field names). Your application should connect to RDBMS and run SQL query to retrieve the results in tabular format, then your application must connect to MongoDB server and do the similar operations. Required time (as milliseconds) to complete the querying operations must be written on screen (a label can be used) upon querying. By this way, we can understand the difference of amount of time in querying between relational and NoSQL DB systems for same result set.

- After **all** fields must shown to user in a tabular format. At UI side, you can use *Grid* controls for tabular viewing. Even a large multiline text box for result set viewing is allowed.
- **At implementation stage of your program you are free to select either Java or C#. Microsoft Visual C++ environment is also accepted.** Due to selection of your environment you must download and use required MongoDB driver and API. Without an appropriate driver you cannot access MongoDB. This information is also valid for your relational database server access. For instance, if you are using MySQL and C# then you must obtain appropriate drivers.

GENERAL RULES

- First of all, you must learn the theoretical background of NoSQL database paradigm and MongoDB
- **You are free to complete your homework with these defined relational databases (MySQL, SQL Server, PostgreSQL, Oracle) and those defined programming environments (Java, C#, Visual C++). Due to your selection you**

must use appropriate driver that can be freely downloaded from MongoDB web site.

- **Do not duplicate your original database contents for increasing the number of records. Use random selection and assignment.**
- **You have to write report and report language must be in English.**
- **Your report should cover**
    - **Explanation about NoSQL**
    - **Comparison about NoSQL vs RDBMS**

NOTES:

- SAVE all your work until the experiment is graded.
- The assignment must be original, INDIVIDUAL work.
- You will work as groups. Each group will contain 2 people.

REFERENCES:

1- https://www.mongodb.com/compare/mongodb-mysql
2- https://dev.mysql.com/doc/sakila/en/