NETFLIX USE CASE

What type of database is best suitable for the expansion of Netflix services to the rest of the world?

SUPERVISOR: Erwin Okken

Allan Akanyijuka and Martin Tonev
3-19-2018

Table of Contents

SYNOPSIS

The main question of this research is: *What type of database is best suitable for the expansion of Netflix services to the rest of the world?*

The main objecting of the research is: to concentrate on database management systems and analyze the current platforms used by Netflix, that is, Microsoft based platforms in comparison with NoSQL and come up with the most reliable and suitable solution for Netflix.

The primary phase of this research consists of building prototypes of three different database platforms and comparing different aspects which include but not limited to speed, scalability, ease of use and support. Literature review and desk research will be done, to explain the meaning of some concepts used in database design.

INTRODUCTION

This chapter introduces the background, reasons why this research was undertaken, objectives and the main concept

Reasons for research

A movie rental company, by the name of Netflix wish to maintain a smooth and operational database for their customers across the whole world. Concerns were raised by the company about extra capacity required to achieve the above-mentioned goals. With these concerns, the researchers have been motivated to undertake this research project and find the most suitable answer to the raised questions.

Research objective

The main objective of this research is to gain a better understanding of the differences, advantages and disadvantages among various database systems, to come up with an informed conclusion for the client.

The aim of the researchers will be to design and test three database systems, that is, ADO.NET, Entity Framework, and a non-relational database (NoSQL) and record the findings.

The client is Netflix and will use the findings to extensively understand the differences, advantages, and disadvantages of various large-scale database platforms in comparison with the current systems used by the organization. The client will also use the findings to solve the arising concerns about having a smooth and reliable movie rental database.

CONCEPT DEMARCATION

The research uses different concepts as explained below.

Relational databases:
According to the Oracle documentation, a relational database is a type of database that presents information in tables with rows and columns. A table is referred to as a relation in the sense that it is a collection of objects of the same type (rows). Data in a table can be related according to common keys or concepts, and the ability to retrieve related data from a table is the basis for the term relational database.
For an example of relationships in a relational database*, see Fig. 001* in the appendix

Non-Relation databases:
Commonly known as a NoSQL database provides a mechanism for storage and retrieval of data that is modeled. Developers are working with applications that create massive volumes of new, rapidly changing data types — structured, semi-structured, unstructured and polymorphic data, therefore most non-relational databases primarily fall into one of the following three categories:

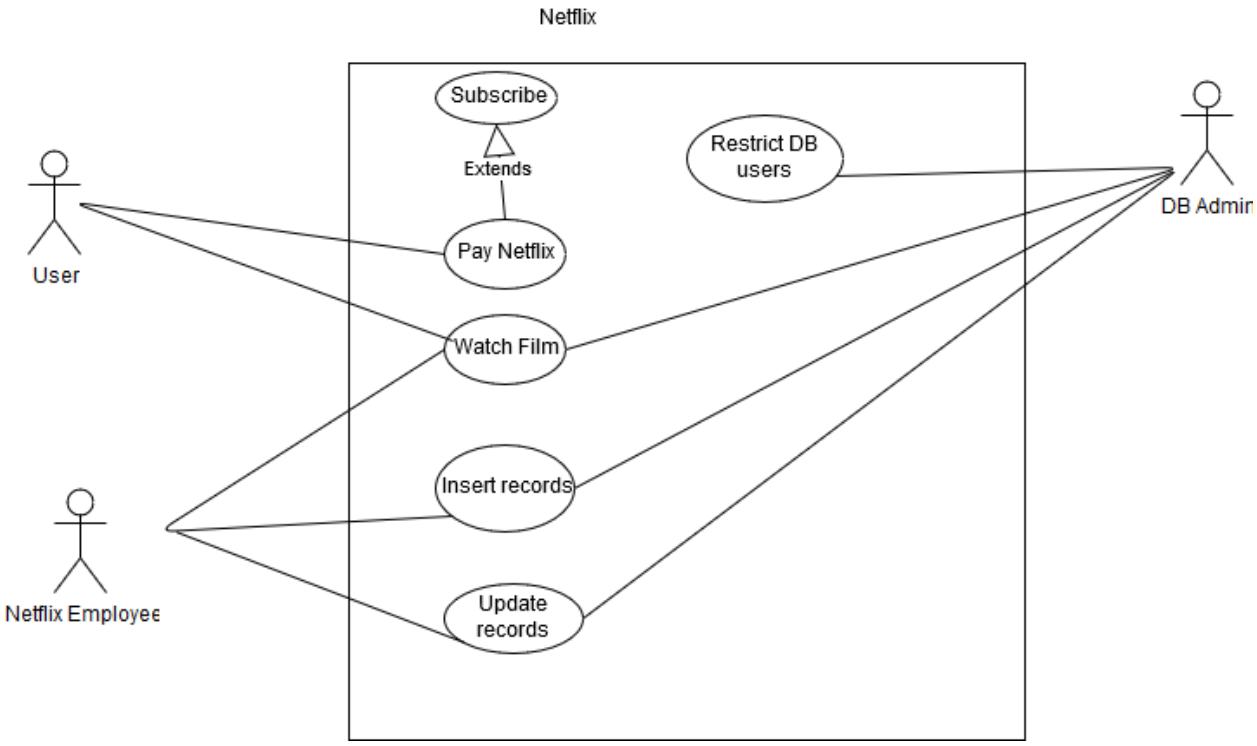
- Document Model; these store data in documents.
- Graph Model; graph databases use graph structures with nodes, edges and properties to represent data.
- Key-Value and Wide Column Models; every item in the database is stored as an attribute name, or key, together with its value.

ADO.NET

(Microsoft, 2017) explains the concept as, a set of classes that expose data access services for .NET Framework programmers. ADO.NET provides a rich set of components for creating distributed, data-sharing applications. It is an integral part of the .NET Framework, providing access to relational, XML, and application data. ADO.NET supports a variety of development needs, including the creation of front-end database clients and middle-tier business objects used by applications, tools, languages, or Internet browsers. under the same source, a different concept **(Entity Framework)** is explained as an object-relational mapper that enables .NET developers to work with relational data using domain-specific objects. It eliminates the need for most of the data-access code that developers usually need to write

With the main concepts explained, the research will further achieve final results creating prototypes using Visual Studio as the IDE and C# as the programing language. A console output will be created to show the speed of handling a query statement by each database types. This will be designed with the Netflix movie rental concept in mind

The Netflix concept can be further illustrated in the use-case diagram below;



CENTRAL QUESTIONS AND SUB-QUESTIONS

This chapter introduces the main question and its sub questions.

As stated in the introduction, the main objective of this research is to gain a better understanding of the differences, advantages and disadvantages among various database systems, to come up with an informed conclusion for the client.

Based on this objective, the main question of this research is:
What type of database is best suitable for the expansion of Netflix services to the rest of the world?

To realize the research goal, the main question is expanded in the following sub-questions:

- Should a different testing approach be used?
- which of the two database types (relational and non-relational) is most suitable for Netflix?
- Is there a different type of database with better performance?
- Are relationships important to Netflix?
- How does switching to a different platform affect Netflix?
- Is there a way to improve the performance without entirely changing the platform?

METHOD SECTION

Data collection method and description of fieldwork

Both Quantitative and qualitative data collection method will be used during this research, however, the researchers will focus majorly on observation. This will be done by viewing the results from the tests and analyzing them. This is due to the nature of the research which requires building of live prototypes.

Measurement instruments

Concepts will be transformed into answerable questions by examining and analyzing various elements of the three prototypes. These are;

- Speed; this is the amount of time taken by each one of the databases to insert, retrieve or update records inserted by a user.
- Ease of use; How much expertise is required to design and maintain the Netflix database?
- Scalability; the ability of a database to be expanded or shrank to any size without affecting performance, will be analyzed since the biggest concern with Netflix is expandability.
- Support; the availability of help from DBMS. This will be determined by reviewing literature.
- Cost; how much does it cost to design, expand, or de centralize a database? Which database requires the most expensive hardware?
- Consistency.

Analysis plan

All the data from the prototypes will be thoroughly sorted and analyzed to the best possible outcome.

- The researchers will analyze the ease of use for each one of the database types during the design phase and make notes about it.
- The cost of setup, migration or expansion will be determined by both the prototypes and reading of literature review from the providers of the database systems.
- Support will also be determined by how much time the researchers used in getting help on how to use each one of the databases.
- Speed will be determined by comparing the execution time of queries among the three prototypes.

RESULTS

FINDINGS

Following the research done, through literature review, tests with prototypes and the whole database design process, the following findings were discovered. They will be used by Netflix to decide on which database platform is the best for the organization.

NOSQL

Ease of use:

- During the design phase, MongoDB was used and was found to be relatively easier to use since there was little time used in trying to critically create relationships between the data. NoSQL allows duplication of data, therefore there was no need to do normalization; which can be a time-consuming phase of database design.

- Little expertise and a basic understanding of databases was required to come up with a database and its content. All a user needs to do is download MongoDB from https://www.mongodb.com/download-center?jmp=nav and a well document set of installation instructions are provided by the website.

- Using MongoDB, there are endless possibilities on which IDE or development platform to use. JetBrains IDEs, Visual Studio, or a variety of IDEs provided by MongoDB can be used for development. This was found to be flexible especially for Netflix developers who are not willing to change their development environment.

- Data insertion and manipulation was found to be flexible and easy. In the Appendix of this document is the documented syntax of how to insert a new document into a NoSQL database using the visual studio IDE.

Support:

- A well document set of installation instructions with an example of how to use NoSQL are provided by the MongoDB website. This makes it easier for a complete noob to understand the installation process and the usage. In addition to this, a white paper is provided, on why NoSQL should be used, and this is critical in deciding on whether to use MongoDB or not.
- It was also discovered that the NoSQL expert community is still lesser than the relational database community therefore, there is a little lesser support from fellow users of the database.

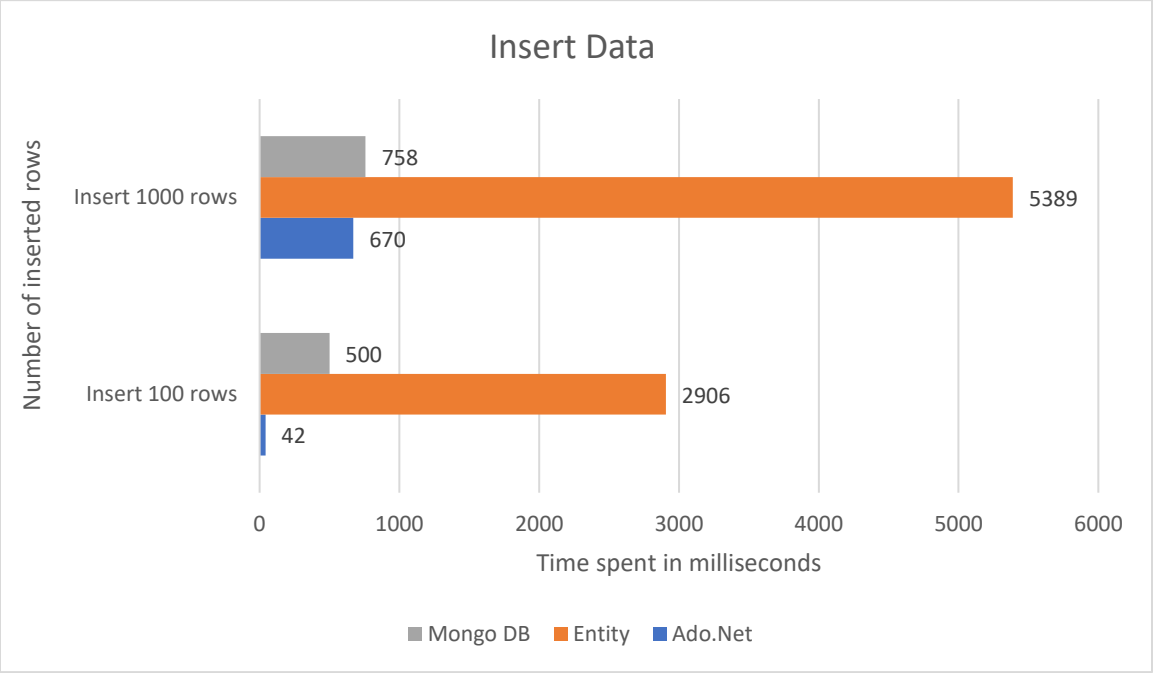
Consistency:

- Since some fields between documents can be duplicated, this showed little consistency while using this type of database.

Speed:

- A test was initiated, to determine the speed it takes MongoDB to insert 100 rows and 1000 rows of data as compared to other database solutions.

Insert test:



As seen from the graph, Ado.Net takes the least time in both tests, while NoSQL takes a slightly longer time with 100 rows but there is a slight increment with time when it is 1000 rows. MongoDB (NoSQL) is faster with more data because there is only 258 milliseconds increment as compared to Ado.Net and Entity with 628 and 2483 milliseconds increment respectively.

Scalability:

- Looking at the NoSQL prototype, a NoSQL database with 100,000 table records takes up about 17MB of storage space. This should have been more, but NoSQL shrinks the data to fit into the least possible storage space without affecting performance.

Cost:

- NoSQL database operates at minimal hardware requirements and therefore the expansion costs are relatively cheaper than all other database platforms.

ADO.NET and Entity Framework

Ease of use:

- Being a relational database, the design process starts with identifying the right link between different data. These links are called relationships. Much as this process eliminates duplication and improves cohesion, it requires a lot of time and expertise for such a huge database like Netflix. During the creation of this database, it was discovered that it is relatively more difficult to setup and use than the NoSQL database.
- Users can access and manipulate the data using a strong query language that uses Join statements which can extract data from more than one table, but all this comes with the required expertise and understanding of how to use these tools.

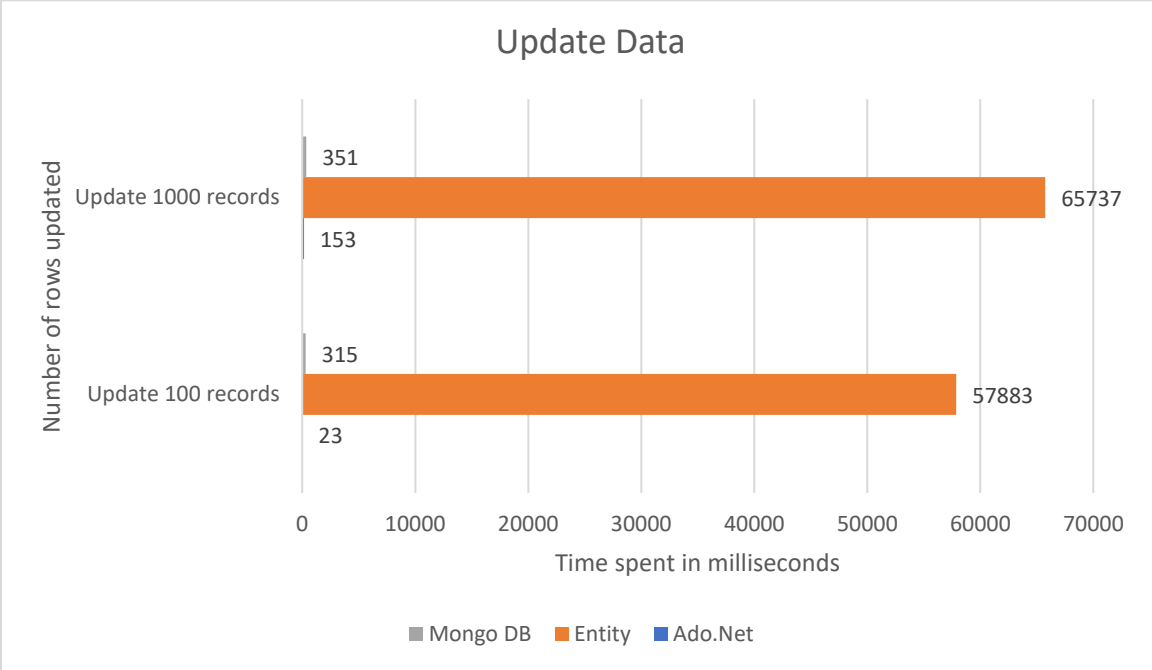
Support:

- Much as it is difficult to use, there is a large community of users offering support, in addition to well documented tutorials from Microsoft and organizations responsible for the maintenance of the frameworks.

Speed:

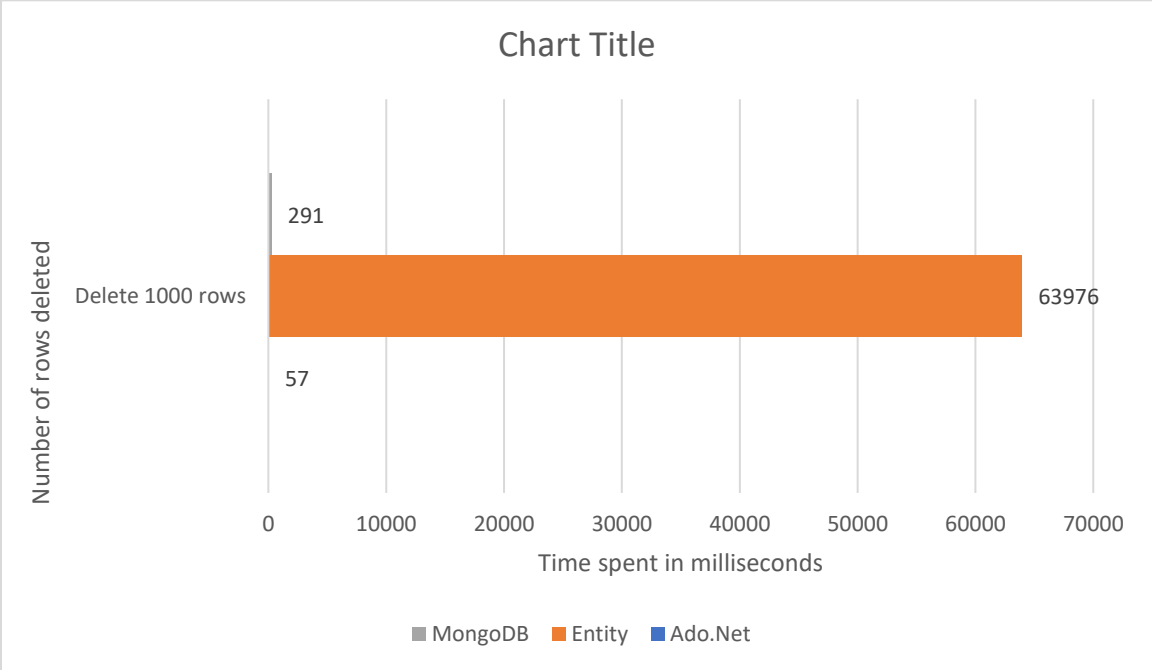
- A test was initiated, to determine the speed it takes all the 3 typed of databases to update and delete 1000 records of data.

Update test:



- Ado.Net spend the least time updating batches of 100 and 1000 records of data at 23 and 153 milliseconds respectively. There is 130 milliseconds increment from 100 to 1000 records.
- Entity framework requires the most time to do a similar task, which is 57883 and 65737 milliseconds on 100 records and 1000 records respectively. That is a 7854-milliseconds increment
- Mongo DB takes a slightly higher time than Ado.Net for both tasks and 351 and 315 milliseconds, but the difference between 100 and 1000 records is lower than the rest at just 36 milliseconds. This means, scaling the database has the least negative impact on NoSQL databases.

Delete test:



- Ado.Net deletes 1000 records in 57 milliseconds while using the same hardware infrastructure, entity framework needs a whopping 63976 milliseconds. MongoDB uses 291 milliseconds to complete a similar task.

Consistency:

- By eliminating data redundancy through carefully created relationships, Ado.net and entity framework ensured strong consistency among data.
- SQL JOINS ensure that data is manipulated and accessed in any manner the user intends to. This requires consistency and Ado.Net and entity framework were discovered to provide that.
- However, as seen from the tables, Ado.Net looks to have dropped 1 record from the 10,000 supposedly inserted records through a loop while entity framework maintained the exact number of rows inserted. Therefore, there is lower consistency while inserting a huge amount of data with Ado.Net.

Cost:

- The set-up cost is relatively cheap since most of the tools used by Ado.Net and entity framework are offered free of charge from Microsoft. But since the main concern with Netflix was expansion, the cost of expansion was discovered to be expensive. This was determined by the computing power required to process a huge number of queries. Inserting 100 Million records, which is about the current amount of Netflix subscribers, required a bigger storage space and faster processors. Expansion would require Netflix to purchase bigger isolated data houses, which could be a bit costly.
- The cost was discovered to affect scalability since new hardware is required to scale higher.

Scalability:

- As mentioned in cost, inserting 100 Million records required a bigger storage space and faster processors. This means, both Ado.Net and Entity framework do not scale well if 100 million users accessed the database at the same time. There was no significant test made, but the requirement of new hardware shows how Ado.Net and entity framework are good for smaller applications and would choke without any additional physical infrastructure in place.

CONCLUSION

This chapter includes the conclusions that have been made through the research that was conducted.

Based on the central question, it can be concluded that the best database for Netflix expansion would be NoSQL. This conclusion considers various aspects like scalability, ease of use, cost and speed.

Much as Ado.Net and Entity framework provide a more consistent approach, NoSQL provides more benefits than just consistency. With expansion, Netflix would like to save time and cost involved in maintaining their database and NoSQL provides the best solution.

Based on the findings, NoSQL is relatively faster than both Ado.Net and entity framework because it scales the space according to the amount of data inserted. It manages to insert and update many records with just a slight change in performance as compared to the other two frameworks. Therefore, this is good for expansion of the Netflix database.

In addition to speed, NoSQL scales up and down to any size without needing extra hardware and it can be concluded that this answers the central question.

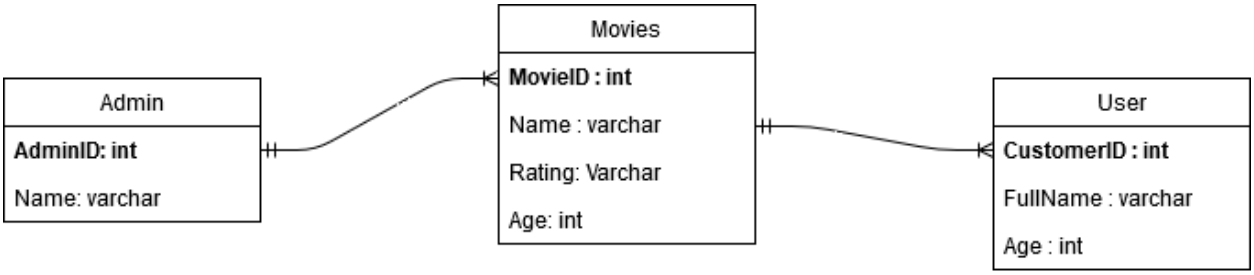
As mentioned in the findings, expansion comes with added cost. Therefore, Netflix would like to have the best and least costly option. NoSQL has proven to provide every necessary tool to achieving this goal

APPENDICES

NoSQL syntax

```
        private static IEnumerable<BsonDocument> createNewMovies()
        {
       var movie1 = new BsonDocument //a new document movie1 is initialized as a variable

{
       /* Data is then inserted by typing the data id in quotation marks and its value
next, all enclosed in parenthesis. */

      {"id", 1},
      {"name", "Davinci code"},
      {"age", 15}
    };

            var movie2 = new BsonDocument //another document movie2 is initialized
    {
       {"id", 2},
      {"name", "The God Father"},
      {"age", 13}
    };

            var movie3 = new BsonDocument //and another document is also initialized

/* Notice Data is inserted in movie3 with the same movie id as movie1. NoSQL treats each
document independently and therefore creates an auto generated ID for each document to
uniquely identify them. */

{
       {"id", 1},
      {"name", "The Incredible Hulk"},
      {"age", 18}
    };
       /* All the documents are then added to the collection */
            var newMovies = new List<BsonDocument>();
            newMovies.Add(movie1);
            newMovies.Add(movie2);
            newMovies.Add(movie3);

            return newMovies;
        }
    }
```

Fig 001.


Admin
AdminID: int
Name: varchar
Movies
MovieID : int
Name : varchar
Rating: Varchar
Age: int
User
CustomerID : int
FullName : varchar
Age : int

REFERENCES

Microsoft. (2017, 03 30). *ADO.NET Overview*. Retrieved from docs.microsoft.com: https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/