Data Management

University of Brighton
Richard Goodman
13842540

January 10, 2017

Abstract

This report will provide a detailed analysis of various DBMSs covering different areas, and will conclude the most appropriate DBMS for the proposed case study. Found in appendix A, an Entity Relationship diagram is supplied presenting a formalized idealistic database design and structure.

1. Introduction

There has been an extensive development in the database paradigm, and it's continuing to expand, with Relational DBMSs being the norm for a chosen database architecture. There are new emerging alternatives that differ to the traditions of a relational systems, and are becoming more and more popular.

The chosen DBMSs that will be compared for this report are:

Relational DBMS':

- Microsoft SQL Server 2012
- MySQL
- Oracle

Document based DBMS':

- Mongo-DB
- Couch-DB

2. Document base definition

Before any comparisons are given, a brief overview of what differs a No-SQL DBMS to a Relational DBMS will be presented as it is less familiarized. For starters, the term No-SQL generally means 'Non SQL'. What makes it different is that in most cases the system uses a document structure. How this compares to storing data in a relational table is quite the opposite. Aggregation and denormalization are a common construct "to simplify/optimize query processing" [1]. Additionally, document structures typically store the data using the language JSON, *(a visual representation of No-SQL can be seen in figure 1).*

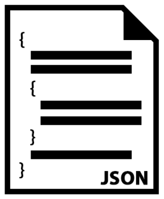


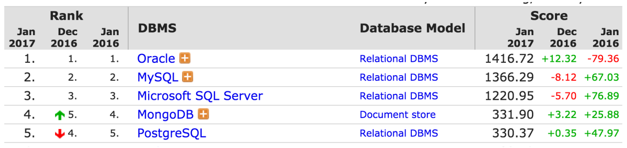
Figure 1: *A visual example of a document structure*

3. Variety of Languages & Currencies

Comparing the languages of different DBMSs produces an interesting topic. The first two DBMS's introduced feature the language (T-)SQL *((Transact-)Structured Querying Language)*, whereas Oracle uses PL/SQL *(Procedural Language/SQL)*. and No-SQL is a much more flexible environment.

3.1. SQL

SQL is the most known and in-demand programming language for databases, *(as seen in figure 2)*, by a website that provides a ranking system for database systems currently used in the professional environment [2].



Rank			DBMS	Database Model	Score		
Jan 2017	Dec 2016	Jan 2016			Jan 2017	Dec 2016	Jan 2016
1.	1.	1.	Oracle ➕	Relational DBMS	1416.72	+12.32	-79.36
2.	2.	2.	MySQL ➕	Relational DBMS	1366.29	-8.12	+67.03
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1220.95	-5.70	+76.89
4.	↑ 5.	4.	MongoDB ➕	Document store	331.90	+3.22	+25.88
5.	↓ 4.	5.	PostgreSQL	Relational DBMS	330.37	+0.35	+47.97

Figure 2: *DBMS Ranking for January 2017*

Although this is an advantage, the first two relational DBMS's are only restricted to SQL themselves. However, with Microsoft's SQL Server, a common conjunct tool used is Visual Studio, where programs are generally written in C# to extract data from the database for other external uses *(such as the web)*.

An obvious compatibility constraint to the first DBMS, is the product being proprietary. Because of this, the application itself can only be ran on a windows operating system. Whereas with MySQL can be ran on all 3 major operating systems, giving that wider flexibility in terms of compatibility.

Additionally, when it comes to extracting data, especially for web, a popular way is using PhP, although various other languages can be used. Moreover, it opens up the freedom of being used elsewhere; such as mobile application development.

3.2. Oracle

As stated above, Oracle slightly differs to the other two RDBMSs by having a slightly different language prefix. Although the system does use SQL, as it is a cloud based DBMS that also incorporates an object orientated style. One significant difference is that Oracle allows you to group stored procedures into 'packages'. Although there is that level of variation, as it is heavily SQL based a new domain knowledge is not overwhelming *(only the new features Oracle provide)*.

3.3. No-SQL

Most NoSQL systems "are based on storing simple key-value pairs on the premise that simplicity leads to speed" [3], because of this, NoSQL databases became the preferred currency for operating big data.

For both NoSQL databases chosen, the structure is written in JSON, *(as seen in figure 1)*. However, when it comes to extracting data, various languages can be used. Common languages used are JavaScript *(More specifically, Angular or Node JS)* and PhP. Although different languages are used to extract the data, the output returned is in JSON.

Similar to MySQL, the two chosen No-SQL systems chosen for this report are also supported on all 3 major operating systems. However, Linux is a preferred OS when using one of these systems.

4. Security

Security is important, especially when storing personal data about people, as well as the key areas [4]:

- Theft & Fraud
- loss of confidentiality
- loss of privacy
- loss of integrity
- loss of availability

It's important to note that most data threats are internal than external. To prevent this a series of features provided by the DBMS can be implemented.

For relational systems, a big threat is SQL injections, "65 percent of organizations represented in this study experienced a SQL injection attack successfully" [5]. To prevent such a common threat, stored procedures can be used, this is because the code is stored in the database itself, and if using parameters, and languages such as PhP *(if using it for web based projects)*, constraints can be applied to check for any abnormalities in the parameters passed.

Additionally, the RDBMSs chosen feature authentication for users, and also views. Views are a great way of displaying data to those without technical knowledge of SQL and preventing any accidental loss of data/integrity.

Similarly, the NoSQL DBMSs chosen also feature authentication that can be applied to users *(for example, only allowing select privileges to users)*. As these systems are more simplistic in their approach to storing data, the main security issues are when using the data elsewhere. For example on the web, one authentication possibility would be JSON Web Tokens (JWT), which are used to "safely pass claims in a constrained environment" [6]. By storing them locally in a cookie to a authorised user once logged in, you can control what data they can view / edit at various levels.

5. Value

The chosen systems that are being compared all have their own economical factor. For example, using Microsoft's SQL Server requires not only a Microsoft OS Server which already adds a price of $6,155[1], but the licensing fee which is also $14,256[2].

Despite this cost, with it being a propriety software it offers extensive features, with the purchase of Microsoft Office Suite, users can import queries into other applications such as Microsoft Excel for any analytical purposes, but also Microsoft Access for users with low database knowledge to use as it provides a user friendly interface. Moreover, one significant feature of Microsoft SQL Server is the Analysis services data mining tool which is a popular tool for business analytics's.

Oracle also being a propriety software comes with a license fee, with the standard edition costing $17,500 and the enterprise version costing $47,500[3]. Although Oracle is considerably a lot more in price, the features it provides speak volumes. Nevertheless, they provide powerful features to support business intelligence, data warehousing, migration and much more.

Compared to the proprietary options that have been discussed, open-source services are typically free to obtain, however, hosting them is a different manner. MySQL only requires a server with any operating system, with the ability to host it on a UNIX server this reduces costs down significantly.

Hosting No-SQL servers is very similar to open source relational services such as MySQL. Although depending on the chosen No-SQL DBMS there can be a desired set-up to make full functionality of the service. For example, Mongo-DB suggests having several servers for optimal performance *(also known as horizontal scaling)* [7], even though the same data will be distributed over various servers, one server is purely dedicated in handling transactional queries such as: Creating, Updating and Deleting records, and the other servers are used to for the analytical queries *(Reading data)*.

As well as this, an additional server is required to act as a tie breaker, this is because if the transactional server fails, another server is chosen to take over. However, if there are an even number of analytical servers, then the tie breaker server will break the even vote. Even though this is the desired set-up for Mongo-DB, this can be hosted on one server, but isn't recommended.

Furthermore, there are hosting services dedicated to providing No-SQL systems, such as Amazon web services and Google. By providing a front end it makes it easier for management and overall performance statistics.

6. Transaction Speeds

Performance is a key topic when selecting an appropriate DBMS. The main transactions of any database system is CRUD: Create, Read, Update and Delete.

One paper compares these transactions over several DBMSs compromising of several NoSQL and Microsoft SQL Express. It's worth noting that although this paper uses tests to store key-value pairs which isn't RDBMSs strong point, the results prove controversial to this statement. Below is the graph and tables from this paper for each iteration of Crud [3]. *(To prevent confusion, the other databases have been removed).*

[1]`https://www.microsoft.com/en-gb/cloud-platform/windows-server-pricing`

[2]`https://www.microsoft.com/en-gb/sql-server/sql-server-2016-pricing`

[3]`http://www.oracle.com/us/corporate/pricing/technology-price-list-070617.pdf`

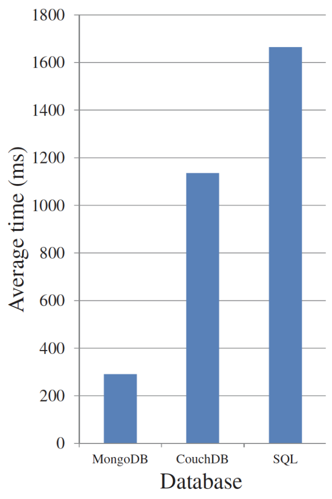


Figure 3: *Time for creating (MS)*

Database	Number of operations					
	10	50	100	1000	10000	100000
MongoDB	8	14	23	138	1085	10201
CouchDB	23	101	196	1819	19508	176098
MS SQL Express	13	23	46	277	1968	17214

Table 1: *Time for reading (MS)*

Database	Number of operations					
	10	50	100	1000	10000	100000
MongoDB	61	75	84	387	2693	23354
CouchDB	90	374	616	6211	67216	932038
MS SQL Express	30	94	129	1790	15588	216479

Table 2: *Time for updating (MS)*

Database	Number of operations					
	10	50	100	1000	10000	100000
MongoDB	4	15	29	235	2115	18688
CouchDB	71	260	597	5945	67952	705684
MS SQL Express	11	32	57	360	3571	32741

Table 3: *Time for deleting (MS)*

Reading is the most common transaction, Bassil uses this protocol to test the execution times on RDBMS', testing 10 select statements varying in complexity. His recordings can be found below [8].

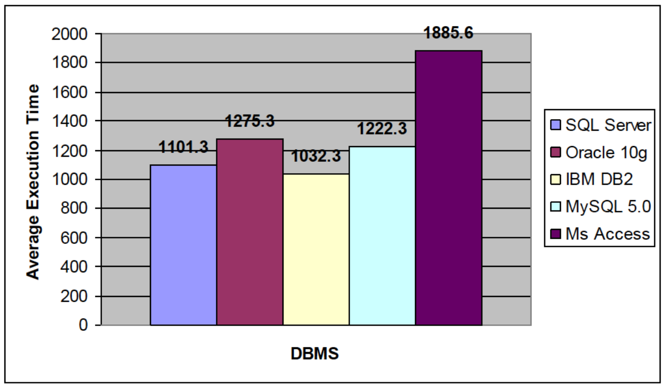


Figure 4: *Average execution time (MS)*

From these findings, it's interesting how the performance of each system is. Specifically how MS SQL Express outperformed CouchDB. Additionally from the second set of findings, Oracle on average performed slower, one could argue from looking at the statements it's because it was more focused towards SQL rather than PL/SQL.

7. Conclusion

In conclusion, the most appropriate DBMS selected for the case study is MySQL. The reason for this is due to the low cost of set-up and up-keep, but also the support for database connectivity, as well as query transactions using other technologies such as PhP or Javascript. Moreover, by joining such technologies, one can use hashing methods to encrypt and decrypt private information such as bank details and passwords.

Additionally, MySQL provides high security elements that can be implemented. For example, stored procedures to prevent SQL injection attacks, as well as authentication for users. In addition to the above results, MySQL sits comfortably in terms of average transaction speed.

One final point to make for this selection is the volume of knowledge known in this field already, with any new technology *(in this case NoSQL)*, the knowledge and requirements are more scarce, which can provide pressure in the business aspect.

In appendix A is the ER diagram suitable for the domain problem, followed by table declaration in appendix B, with the appropriate data types and any constraints that should be implemented.

References

[1] Katsov, I. (2012).
NoSQL data modeling techniques.
`https://highlyscalable.wordpress.com/2012/03/01/nosql-data-modeling-techniques/`
(Accessed: 7 January 2017).

[2] gmbh, solid I. (2017).
DB-Engines ranking - popularity ranking of database management systems.
`http://db-engines.com/en/ranking`
(Accessed: 7 January 2017).

[3] Li, Y., Manoharan, S., *Date unknown*.
A performance comparison of SQL and NoSQL databases.
Department of Computer Science, University of Aukland, New Zealand.

[4] Connolly, T. and Begg, C. (2014).
Database systems: A practical approach to design, implementation, and management: Global edition. 06th edn. Harlow, United Kingdom: Pearson Education.

[5] Ponemon institute LLC, (April 2014).
The SQL Injection Threat Study.
Sponsored by DB Networks

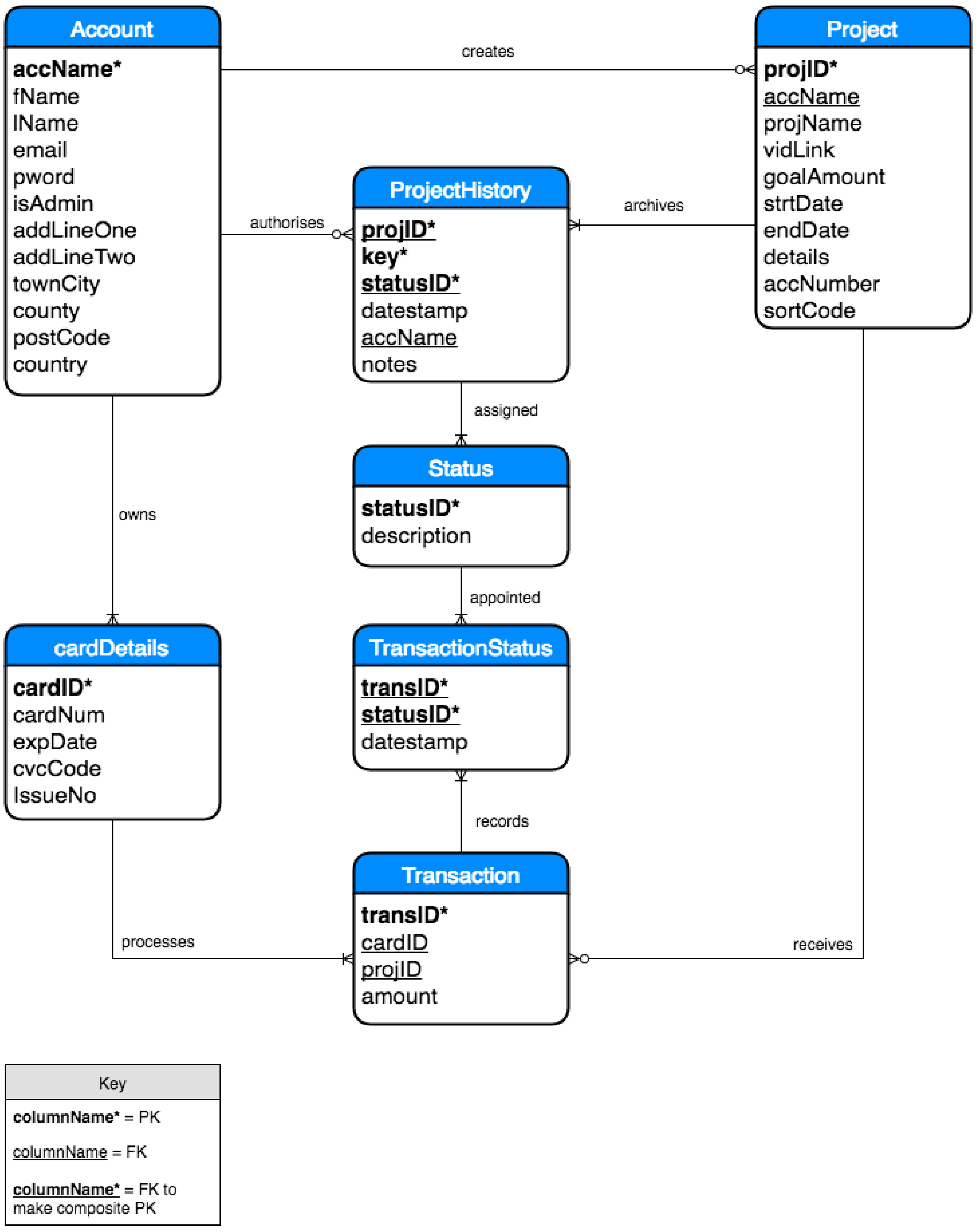
[6] Peyrott, S., (2016).
The JWT Handbook Version 0.9.2
Auth0 Inc.

[7] Unknown.
MongoDB Architecture.
`https://www.mongodb.com/mongodb-architecture`
(Accessed: 7 January 2017).

[8] Bassil, T., (February 2012).
A comparative Study on the Performance of the Top DBMS Systems.
LACSC - Lebanese Association for Computational Sciences
Registered under No. 957,2011, Beirut, Lebanon.

Appendix A

Entity relationship diagram for system requirements for the case study.



Appendix B

Table definition providing data types and any constraints needed.

	Not null	Data type	Constraint	Information
Account				
accName*	X	varchar(25)		
fName	X	varchar(25)		
lName	X	varchar(50)		
email	X	varchar(100)		
pword	X	varchar(255)		must be hashed
isAdmin	X	bit		
addLineOne		varchar(100)		
addLineTwo		varchar(100)		
townCity		varchar(50)		
county		varchar(50)		
postCode		varchar(10)		
country		varchar(30)		
Project				
projID*	X	int		
accName	X	varchar(25)		
projName	X	varchar(50)		
vidLink	X	varchar(100)		
goalAmount	X	double(14,5)		
strtDate	X	date		
endDate	X	date	endDate can't be before strtDate	
details	X	varchar(max)		
accNumber		varchar(16)		must be hashed
sortCode		varchar(8)		must be hashed
ProjectHistory				
projID*	X	int		
key*	X	int		
statusID*	X	int		
datestamp		datetime		
accName		varchar(25)	isAdmin must be 1	
notes		varchar(max)		
Status				
statusID*	X	int		
description	X	varchar(35)		
TransactionStatus				
transID*	X	int		
statusID*	X	int		
datestamp	X	datetime		
Transaction				
transID*	X	int		
cardID	X	int		
projID	X	int		
amount	X	double(14,5)		
cardDetails				
cardID*	X	int		
cardNum	X	varchar(25)		
expDate	X	date		
IssueNo		int		
accName	X	varchar(25)		