

LK222134_DatabaseAssignment AbetaweEntertainment

by Liam Kelly

Submission date: 19-Jan-2023 11:39PM (UTC+0000)

Submission ID: 195671036

File name:

42197_Liam_Kelly_LK222134_DatabaseAssignmentAbetaweEntertainment_474125_1123568610.docx
(259.2K)

Word count: 3795

Character count: 22283

Data Storage Investigation Assignment

UWTSD - ACCA7002: Database Systems

Liam Kelly | 222134
2222134@student.uwtsd.ac.uk

Table of Contents

ABSTRACT	3
INTRODUCTION.....	3
NOSQL DATABASES	3
<i>Key-Value</i>	3
<i>Wide-Column</i>	4
<i>Graph</i>	5
<i>Document-oriented</i>	6
USE CASE OF SQL VS. NoSQL.....	6
CONCLUSION	9
WORKS CITED	10
BIBLIOGRAPHY	11

Word Count: 2856

Abstract

Storing data is an important consideration for any business. Traditionally, databasing operations have mainly been performed through the relational database model in combination with Structured Query Language (SQL). With the advances in storage technology over the last few decades and the rise of larger swathes of complex customer data, other options of data storage, once perhaps overlooked, are becoming viable for businesses to manage their applications and customer data. Not only Structured Query Language (NoSQL) databases can provide several solutions for companies like Abertawe Entertainment (AE), of which this report seeks to compare to one another and the more traditional relational model.

Introduction

When discussing data storage there are two main types of database solution to consider: relational databases with SQL (RDWSQL) and NoSQL. This report compares these within the commercial context of an expanding media entertainment company Abertawe Entertainment (AE). AE seeks to develop a platform for its customers as part of its expanding digital business model. This report will compare the four types of NoSQL databases – Key-Value, Graph, Wide-Column and Document (*Fig.1*) – alongside their use cases, advantages, and drawbacks. It will also compare the four against the traditional relational database model. This report will end with a broader overview and conclusive points of NoSQL vs. RDWSQL and their respective use case scenarios.

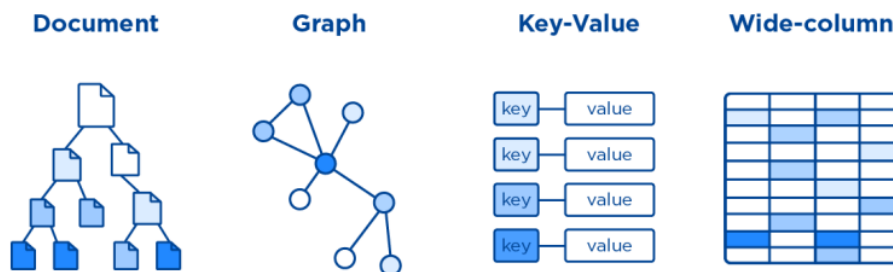


Figure 1: Basic structural illustration of each of the four types of NoSQL database discussed in this report [1].

NoSQL databases

Key-Value

A key-value database is a simple form of NoSQL database that's operation could be compared to a dictionary [2]. When provided with a data input or 'key', for example a String, it returns the data directly associated with it, known as the 'value' [2]. Key-value databases have the benefit of having a rapid access to specific data due to their key index, which points directly to the value associated [2] [3]. This makes key-value databases optimised for high-speed read and write operations. This simple structure helps busy web pages achieve a lower latency even when under high traffic when paired with the multi-nodal network that NoSQL lends itself to [2]. Furthermore, the data type of the value is not specified in this model unlike RDWSQL. Regardless of the data type provided, the system

stores the value as that type; string, binary large object (BLOB), etc. and is the same retrieved when prompted [2]. Due to this simple interface and the generality of its potential use, Key-Value can be used in a variety of systems from storing image files to webpages, shopping carts and user profiles [2] [4]. This simplicity also lends to higher scalability and reliability meaning that an organisation can adjust its database solution to its needed requirements. The interaction with a Key-Value database is also simple with only three operations ('put', 'get', and 'delete') to the key itself [5].

However, with this simplicity comes limitation. This limited number of operations does not permit a user to query the data as you can in other database types such as RDWSQL or document-oriented databases. It also does not allow analysis of the relationship between data points like a graph database. In continuation of this, operations need to be precise; each key must be distinct, which means that if you cannot identify a key-value pair you cannot return the result [2]. With a large and complex dataset (such as a multi-media platform containing assets, customer, and transaction data), the key-value database may not be the best option for AE. A key-value database could be utilised to store digital assets, however, this capability along with query searching is available in document-oriented stores [2].

Wide-Column

A wide-column database has some similarities with the row and table structure of a RDBMS and the key-notation index of the key-value database solution. Unlike relational tables, wide-column stores data only in columns [3] [6]. Consider this example in a traditional relational model: if one row field in a customer database requires the column "landline telephone number", for example, then all rows must have the column. This can be *null* in a RDWSQL, but it must still exist in the pre-defined schema. Wide-column compartmentalises each column of a data table so if a field does not require a specific column then that axis in the architecture is not generated. This may seem rudimentary, but when the database grows, having to generate each field can be resource- and time-consuming [7].

Not only this, but in wide-column databases, columns can be nested inside others to generate super columns which, in turn, can be grouped to form families [8]. In essence it creates a multi-dimensional data structure expanding the key-value store previously mentioned. Like key-value databases, wide-column can add new data entries easily and retrieve data quickly. Unlike RDWSQL, however, these operations do not require the full design of the database beforehand [9]. Much like the other forms of NoSQL database discussed in this report, wide-column shares the advantage of scalability and parallel processing. In addition, as the data is not stored in a rigid schema as required in RDWSQL the data is more readily compressed for storage.

Where the wide-column database falls short is when it is applied outside its scope of strength. In transactions for example, wide-column is best employed in the analysis of the transactions, not transactions themselves, as they do not update these in the most efficient way due to their architecture [2] [8]. NoSQLs lack of support for ACID transactions is a common drawback across all NoSQL databasing [6]. This alongside row-specific queries are drawbacks of wide-column.

Although the analysis of trade information would be useful, I believe a more generalised and capable database design is more suitable than a wide-column for expanding commercial businesses, such as AE.

Graph

A graph NoSQL database is comprised of *nodes* and their connections to one another through *relationships*. *Nodes* can also possess *properties* which leads to Graph NoSQL databases sometimes being referred to as 'triple stores' [2]. A mapping of these *nodes*, their *relationships*, and *properties*, creates a graphing network [10]. Graph stores are used when there are many data points which are related to one another in complex ways. Their common business application is for analysis between objects such as social networks, related products and finding patterns between data points (Fig.2) [2] [5].

Graph databases have the advantage over RDWSQL where the relationships between nodes are inexpensive regarding computing resourcing and do not need to be as rigidly defined as *joins* are in SQL [2] [11]. Where they fall short in comparison to their NoSQL counterparts is their difficulty in scaling horizontally due to their networked nature with interlinked data nodes [2]. This in addition to their emphasis on the connecting relational data *between* nodes means that graph databases are not only specialised but are also rarely used on their own in a business context [12]. With this in mind, a graph database would be useful for an advanced commercial web application. For example, recommending books and asset types that buyers of a similar purchase history have also bought and analysing similar relationships for commercial use. However, as a stand-alone application to manage transactional data, user information and cataloguing, graph stores are not suitable.

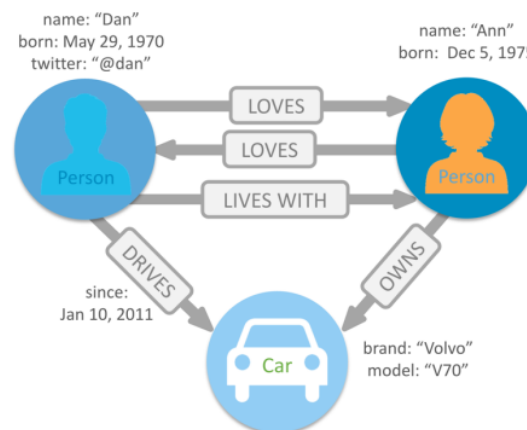


Figure 2: Illustration of a graph database [13]

Document-oriented

A document database stores data in document form in collections using Javascript Object Notation (JSON)-like documents or Extensible Markup Language (XML), as opposed to the tabular structure of RDWSQL [2] [14]. It can access information rapidly, like a key-value store. However, as the data is stored as documents, it provides the capacity for a more generalised structure that can be probed with advanced queries, unlike its other NoSQL counterparts [14]. Each document entry is an indexed object and within this object can be other nested objects. This allows for a very integrated and rich database structure.

With document databases, each object is indexed and is searchable. Query language is simple and does not require much knowledge of a structured syntax compared to that of SQL [3]. Not only this, but it is also incredibly scalable and, due to its lack of a rigid schema, requires less forward-planning in terms of storage design. It is also more responsive to commercial or technological change. When a new document is added to a collection, it is indexed automatically, which allows its associated values to be searched too. For complex datasets such as those provided by AE, similar data can be structured into different collections and 'joining' attributes can be referenced to one another. Combined with powerful applications like MongoDB, regular queries and indexing can be used to increase data accessibility and analysis. Not only this, but by data being associated to a document it removes the need for joins and complex functions used in SQL [3]. As with all NoSQL databases mentioned, this lack of joins and defined relationships allows document stores to scale horizontally and be more versatile to a business's needs.

Use case of SQL vs. NoSQL

RDWSQL are more suitable for structured data with uniform datatypes and attributes [3]. This is due to RDWSQLs requirement of having datatypes, table structures, and their relationships and schemas defined in advance [5]. This planning aspect, alongside the difficulties of changing the system readily, leads to RDWSQL being seen as obstinate to change with a large overhead in time, labour and technical execution before the data has even been processed [2].

In addition, SQL databasing has the limitation of scaling difficulty. SQL and the relational database model were developed in the 1970s. This was an era with expensive storage (~\$260/MB) compared to that of today (<\$0.001/MB) [15] and were developed in the era of localised data and designed to run on a single large server. RDWSQL scales vertically – when the database increases in size, the processing increases and, in turn, the hardware must improve with it – however, this gets expensive and is subject to the law of diminishing returns [6]. To minimise data repetition and accurate processing, RDWSQL requires the rigid schema, join and relationship design aforementioned. It is these requirements that prevent SQL data structures from being compartmentalised whilst maintaining these relationships. When data is provided that does not fit the relational model, architecture can become complex and resource consuming.

SQL is a clearly defined and documented query language which can help analyse and manipulate data [9]. SQL databases have been used as an industry-standard for decades and allow for the fast processing and retrieval of efficiently formed data [2]. SQL has a technical

investment for employees, as well as the required preparation of designing predefined schemas, constraints, joins and datatypes [3]. SQL databases' other strength and benefit over NoSQL is their support for transactions [3]. RDWSQL are ACID compliant [16]. This guarantees consistent data which is especially important when managing financial transactions. If the transaction is not completed, the structure will roll back to its previous state. When completed, it will commit. As mentioned NoSQL does not support this, with the recent exception of MongoDB [17]. As a result, if the use-case is managing financial or transactional information a RDWSQL has this advantage over NoSQL.

A	Atomicity – changes made are completed in full an 'all or nothing' concept. It also gives the ability for rollbacks and committing transactions.
C	Consistency – returns the database to its consistent state after processing the inconsistency of a transaction.
I	Isolation – transactions are isolated, restricted, and guided by a set of pre-determined rules of how data interacts with other data. This is achieved for example, by locking data and row versioning.
D	Durability – transactions are committed the data log before they are written to the data file. This prevents transactions reverting or changing state unexpectedly.

Table 1: The ACID properties [16], self-generated graphic

In summary, RDWSQL databases are best suited to manage uniform, structured datasets where there are relationships between the data. SQL has strengths in the power of its querying language as well as its ACID support of transactions. The limitations of its architecture however, mean that RDWSQL is best suited to data requirements which are slow to change or unchanging.

NoSQL, on the other hand is suitable for unstructured, semi-structured and structured data [4] [18]. It has the advantage of having many types of NoSQL databases to accommodate the data scenario, of which, have been discussed in this report. NoSQL systems sought to overcome expensive vertical scaling by sharding the database from a single node to a cluster of less expensive hardware nodes operating as a network, also known as 'horizontal scaling' [6]. This allows NoSQL to distribute data processing across multiple nodes when being queried, managing resources in high traffic [3]. The lack of prior architectural definition in a NoSQL database allows it to manage and accept a variety of data types from BLOBs and documents to varied character and integer data. The lack of prior labour also means the technical bar is arguably lower than that of SQL [17]. This lower technical requirement, lack of rigid schema and defined relationships also results in NoSQL executing queries faster than SQL requests [3] [18]. However, this processing power is at the sacrifice of data consistency, which is argued to not be as important in the vast expanse of 'Big Data' [6].

Since the turn of the 21st century, the nature of data has changed. The integration of web applications, social media and cloud computing into the commercial and public space has led to large, non-uniform, volumes of varied and continuously changing data. The birth and growth of 'Big Data' has in many respects led to the out-growing of the restraints of the relational database. Servers are required to process large numbers of concurrent users at all times of the day with varied forms of data which are not necessarily related. In order to process this data; the relational model, its pre-requirements and its ACID compliance are at the expense of performance and availability in NoSQL.

Due to its unrestricted nature, NoSQL is flexible to change. It is not defined or restricted by a rigid schema so it can accommodate the characteristic nature of the ever-changing technology sector. Also, at a company level, should the local requirements of a business or the data it wishes to store change, NoSQL has an unmatched adaptability. On the other hand, RDWSQL would require a refactoring from the schema upward, which would have implications for a company in regard to technical cost, labour, and time. This flexibility of NoSQL increases its accessibility to its users, however, with the lack of a standardised query language like SQL this could yield potential challenges especially when moving from one NoSQL system to another. As mentioned, NoSQL sacrifices the principles of ACID, so is not specialised for transactional information. Nevertheless, it follows the principles of BASE [2].

B	<i>Basically Available</i> – a guarantee to the availability of data
A	
S	<i>Soft state</i> – ability for the system to change over time
E	<i>Eventual consistency</i> – the system eventually is consistent when it stops receiving input

Table 2: The BASE properties [2], self-generated graphic

The BASE principle states that although the data may be inconsistent temporarily, it will return to consistency at rest [2]. This poses some challenges to NoSQL databasing and transactions, especially when compared to the consistency of RDWSQL. However, software such as MongoDB now allows for NoSQL systems to support ACID transactions [17]. As discussed, NoSQL databases process operations quicker than RDWSQL [5]. Li investigated the performance of both SQL and NoSQL databases and concluded that there was little correlation between performance and the used data model, nor between SQL vs NoSQL [18]. Although Li did find that the fastest performing database was a NoSQL database [18]. However, this performance testing was only completed on simple read, write and delete protocols for key-value pair database examples. This lends to the idea if performance difference is negligible then the querying power of complex SQL queries exceeds the use capability of NoSQL, aided in large part by SQL's standardised language [3].

Conclusion

The idea that SQL and NoSQL data structures are in competition with one another is a narrow view of the potential of both systems. Many commercial systems integrate both approaches for defined circumstances, and in the rapidly evolving environment of business and technology, these territories change continuously.

For example, MongoDB, a NoSQL document database, is now able to support ACID transactions [17]. Silva [19] also argues that SQL and its relational architecture can be combined with 'Big Data' systems into NewSQL [6], a system with the scalability of NoSQL but the query power and ACID compliance of SQL.

There are use cases for either system. Considering the reflections of this report, AE, running a digital platform where customers can browse catalogues of assets and process purchases could be achieved using either a SQL or NoSQL document database. The deciding factor, I believe, is AE's intention for the future.

Points to consider would be its future scalability, response to a changing business environment, and its in-house technical capacity. If AE anticipates little change in its market focus and business model, I would conclude a SQL database model may be more suitable with its query capacity, ad hoc usage by users and its transaction-oriented system. However, if AE hopes to expand, venturing into other digital media provisions or extending its digital platform, a NoSQL document store may be more beneficial.

Regarding in-house technical capability, I would argue that a NoSQL database in combination with MongoDB could be a viable option. Due to its lower technical requirement, it would allow developers, for example, to adapt the system as they see fit. The challenge with this would come with system changes. NoSQL query language is not ubiquitous and moving from one system to another would take reskilling. If the business need thinks its viable to have a database engineer, perhaps SQL would be more appropriate.

Provided along with this report, both SQL and NoSQL designs for AE's database have been implemented for consideration.

Works Cited

- [1] "NoSQL Technologies," abcloudz, 2022. [Online]. Available: <https://abcloudz.com/services/nosql-databases/>. [Accessed 18 January 2023].
- [2] A. Kelly and D. McCreary, *Making Sense of NoSQL*, Shelter Island, NY: Manning Publications, 2014.
- [3] N. Leavitt, "Will NoSQL databases live up to their promise?," *Computer: IEEE Xplore*, vol. 43, no. 2, pp. 12-14, 2010.
- [4] A. C. Goje and S. Mohammad, "SQL and NOSQL: Which is Better," *SSRN Electronic Journal*, vol. 2, no. 8, pp. 3277-3284, 2015.
- [5] R. Čerešňák and M. Kvet, "Comparison of query performance in relational a non-relation databases," in *13th International Scientific Conference on Sustainable, Modern and Safe Transport (TRANSCOM 2019)*, High Tatras, Novy Smokovec, 2019.
- [6] A. Pavlo and M. Aslett, "What's Really New with NewSQL?," *SIGMOD Record*, vol. 45, no. 2, pp. 45-55, 2016.
- [7] V. Abramova, J. Bernardino and P. Furtado, "Experimental Evaluation of NoSQL Databases," *International Journal of Database Management Systems*, vol. 6, no. 3, pp. 1-16, 2014.
- [8] ScyllaDB, "Wide-Column Database," ScyllaDB, 2022. [Online]. Available: <https://www.scylladb.com/glossary/wide-column-database/>. [Accessed 4 January 2023].
- [9] A. Beaulieu, *Learning SQL*, Sebastopol, CA: O'Reilly Media, Inc., 2020.
- [10] Amazon Web Services, "What Is a Graph Database?," Amazon Web Services, 2022. [Online]. Available: <https://aws.amazon.com/nosql/graph/>. [Accessed 12 January 2023].
- [11] A. Williams, "NoSQL database types explained: Column-oriented databases," TechTarget, 22 September 2021. [Online]. Available: <https://www.techtarget.com/searchdatamanagement/tip/NoSQL-database-types-explained-Column-oriented-databases>. [Accessed 5 January 2023].
- [12] S. Paul, A. Mitra and C. Koner, "A Review on Graph Database and its representation," in *2019 International Conference on Recent Advances in Energy-efficient Computing and Communication (ICRAECC)*, Nagercoil, 2019.
- [13] Neo4j, "What is a Graph Database?," Neo4j, 2021. [Online]. Available: <https://neo4j.com/developer/graph-database/>. [Accessed 19 January 2023].
- [14] MongoDB, "What is a Document Database?," MongoDB, 2022. [Online]. Available: <https://www.mongodb.com/document-databases>. [Accessed 4 January 2023].
- [15] Our World in Data, "Historical Cost of Computer Memory and Storage," Our World in Data, 2022. [Online]. Available: https://ourworldindata.org/grapher/historical-cost-of-computer-memory-and-storage?country=~OWID_WRL. [Accessed 8 January 2023].
- [16] A. Jorgensen, J. Segarra, P. LeBlanc, J. Chinchilla and A. Nelson, *Microsoft SQL Server 2012 Bible*, Indianapolis: John Wiley & Sons Inc., 2012.
- [17] MongoDB, "NoSQL vs. SQL Databases," MongoDB, 2021. [Online]. Available: <https://www.mongodb.com/nosql-explained/nosql-vs-sql>. [Accessed 12 January 2023].
- [18] Y. Li and S. Manoharan, "A performance comparison of SQL and NoSQL databases," in *Communications, Computers and Signal Processing (PACRIM), 2013 IEEE Pacific Rim Conference*, Auckland, 2013.
- [19] Y. Silva, I. Almeida and M. Queiroz, "SQL: From Traditional Databases to Big Data," in *47th ACM Technical Symposium*, Memphis, 2016.

Bibliography

A. Beaulieu, Learning SQL, Sebastopol, CA: O'Reilly Media, Inc., 2020.

Abcloudz, "NoSQL Technologies," abcloudz, 2022. [Online]. Available: <https://abcloudz.com/services/nosql-databases/>. [Accessed 18 January 2023].

A. C. Goje and S. Mohammad, "SQL and NOSQL: Which is Better," *SSRN Electronic Journal*, vol. 2, no. 8, pp. 3277-3284, 2015.

A. Jorgensen, J. Segarra, P. LeBlanc, J. Chinchilla and A. Nelson, Microsoft SQL Server 2012 Bible, Indianapolis: John Wiley & Sons Inc., 2012.

A. Kelly and D. McCreary, Making Sense of NoSQL, Shelter Island, NY: Manning Publications, 2014.

A. Pavlo and M. Aslett, "What's Really New with NewSQL?," *SIGMOD Record*, vol. 45, no. 2, pp. 45-55, 2016.

Amazon Web Services, "What Is a Graph Database?," Amazon Web Services, 2022. [Online]. Available: <https://aws.amazon.com/nosql/graph/>. [Accessed 12 January 2023].

A. Williams, "NoSQL database types explained: Column-oriented databases," TechTarget, 22 September 2021. [Online]. Available: <https://www.techtarget.com/searchdatamanagement/tip/NoSQL-database-types-explained-Column-oriented-databases>. [Accessed 5 January 2023].

D. Bartholomew, "SQL vs. NoSQL," LinuxJournal, 1 September 2010. [Online]. Available: <https://www.linuxjournal.com/article/10770>. [Accessed 15 January 2023].

CodeCademy, "Learn MongoDB", CodeCademy, 2022. [Online]. Available: <https://www.codecademy.com/courses/learn-mongodb/>. [Accessed 21 December 2022].

"Mockaroo.com," [Online]. Available: <https://www.mockaroo.com/>. [Accessed From December 2022].

MongoDB, "Database Scaling," MongoDB, 2022. [Online]. Available: <https://www.mongodb.com/databases/scaling#:~:text=Vertical%20scaling%20refers%20to%20increasing,maximum%20processing%20power%20and%20throughput..> [Accessed 28 December 2022].

MongoDB, "MongoDB Documentation," MongoDB, 2021. [Online]. Available: <https://www.mongodb.com/docs/>. [Accessed 12 January 2023].

MongoDB, "NoSQL vs. SQL Databases," MongoDB, 2021. [Online]. Available: <https://www.mongodb.com/nosql-explained/nosql-vs-sql>. [Accessed 12 January 2023].

MongoDB, "Understanding the Different Types of NoSQL Databases," MongoDB, 2022. [Online]. Available: <https://www.mongodb.com/scale/types-of-nosql-databases>. [Accessed 2 January 2023].

MongoDB, "What are ACID Properties in Database Management Systems?," MongoDB, 2022. [Online]. Available: <https://www.mongodb.com/basics/acid-transactions>. [Accessed January 12 2023].

MongoDB, "What is a Document Database?," MongoDB, 2022. [Online]. Available: <https://www.mongodb.com/document-databases>. [Accessed 4 January 2023].

N. Leavitt, "Will NoSQL databases live up to their promise?," *Computer: IEEE Xplore*, vol. 43, no. 2, pp. 12-14, 2010.

Neo4j, "What is a Graph Database?," Neo4j, 2021. [Online]. Available: <https://neo4j.com/developer/graph-database/>. [Accessed 19 January 2023].

Our World in Data, "Historical Cost of Computer Memory and Storage," Our World in Data, 2022. [Online]. Available: https://ourworldindata.org/grapher/historical-cost-of-computer-memory-and-storage?country=~OWID_WRL. [Accessed 8 January 2023].

"PostgreSQLTutorial.com," [Online]. Available: <https://www.postgresqltutorial.com/>. [Accessed December 2022].

R. Čerešňák and M. Kvet, "Comparison of query performance in relational a non-relation databases," in *13th International Scientific Conference on Sustainable, Modern and Safe Transport (TRANSCOM 2019)*, High Tatras, Novy Smokovec, 2019.

R. Sheldon, "How to choose between SQL and NoSQL databases," Red-Gate, 13 April 2021. [Online]. Available: <https://www.red-gate.com/simple-talk/databases/nosql/how-to-choose-between-sql-and-nosql-databases/>. [Accessed 4 January 2023].

ScyllaDB, "Wide-Column Database," ScyllaDB, 2022. [Online]. Available: <https://www.scylladb.com/glossary/wide-column-database/>. [Accessed 4 January 2023].

S. Paul, A. Mitra and C. Koner, "A Review on Graph Database and its representation," in *2019 International Conference on Recent Advances in Energy-efficient Computing and Communication (ICRAECC)*, Nagercoil, 2019.

V. Abramova, J. Bernardino and P. Furtado, "Experimental Evaluation of NoSQL Databases," *International Journal of Database Management Systems*, vol. 6, no. 3, pp. 1-16, 2014.

Y. Li and S. Manoharan, "A performance comparison of SQL and NoSQL databases," in *Communications, Computers and Signal Processing (PACRIM), 2013 IEEE Pacific Rim Conference*, Auckland, 2013.

Y. Silva, I. Almeida and M. Queiroz, "SQL: From Traditional Databases to Big Data," in *47th ACM Technical Symposium*, Memphis, 2016.

FINAL GRADE

GENERAL COMMENTS

Instructor

86/100

Generic Assessment Criteria	Marks available	Marks awarded
1. Engagement with Literature Skills	20	
Engagement with a wide range of research-informed literature, including sources retrieved independently. Selection of relevant and credible sources. Very good use of referencing, with no/very few inaccuracies or inconsistencies.	15	
2. Knowledge and Understanding Skills	20	

Fantastic demonstration of knowledge and an interesting read	18	
3. Cognitive and Intellectual Skills	15	
Exceptionally designed SQL	14	
Perhaps a lot of redundancy within the NoSQL approach, not that it's a terrible thing but perhaps some of these collections were not required a		
An approach I would have taken after looking at yours, is to remove the assets/books and have them nested under each supplier and then referencing which customer owned each asset under their accounts to remove the need for a transaction?		
But this works and It is clear why you made these design choices and I am not holding them against you, just discussing my thoughts on it		

4. Practical Application Skills	35	
Good clean code that achieved everything requested and more	30	
5. Transferable Skills for Life and Professional Practice	10	
Great report, very well written and great flow	9	

Assessment Mark (Assessment marks are subject to ratification at the Exam Board. These comments and marks are to give feedback on module work and are for guidance only until they are confirmed.)

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12
