|  |
| --- |
| Advanced Databases |
| CA1 Group Project |
| Section A (Data Mart) and Section B (Report) |

|  |
| --- |
| Cree Gunning, Lee Harold  3/6/2021 |

**Table of Contents**

[1 Section A 2](#_Toc65888962)

[1.1 Industry Topic and Business Area 2](#_Toc65888963)

[2 Section B 2](#_Toc65888964)

[2.1 Introduction 2](#_Toc65888965)

[2.2 Analysis and comparison of NoSQL databases with relational databases 2](#_Toc65888966)

[2.2.1 Strengths and Weaknesses 2](#_Toc65888967)

[2.2.2 Future Prospects 2](#_Toc65888968)

[2.3 Motivation for the use of NoSQL databases 2](#_Toc65888969)

[2.3.1 Main characteristics of key-value store databases, column-oriented databases, graph databases and document store databases. 2](#_Toc65888970)

[2.4 How Redis Works (Odd) 2](#_Toc65888971)

[2.4.1 Key Features 2](#_Toc65888972)

[2.5 How Neo4j Works (Odd) 3](#_Toc65888973)

[2.5.1 Key Features 3](#_Toc65888974)

[2.6 How Cassandra Works (Even) 3](#_Toc65888975)

[2.6.1 Key Features 3](#_Toc65888976)

[2.7 How HBase Works (Even) 3](#_Toc65888977)

[2.7.1 Key Features 3](#_Toc65888978)

[2.8 Conclusion 3](#_Toc65888979)

[3 Bibliography 3](#_Toc65888980)

# Section A

## Industry Topic and Business Area

**Industry Topic: Sports**

**Business Area: Purchase Orders (PO – Items)**

**1. Business Requirements**

**2. Data Mart (Snowflake Schema)**

**3. SQL Script to create Fact table and Dimension tables**

**4. SQL script to load data into the Data Mart**

**5. Queries and Outputs**

# Section B

## Introduction

## Analysis and comparison of NoSQL databases with relational databases

NoSQL is a non-relational DMS, that does not require a fixed schema, avoids joins, and is easy to scale. NoSQL databases are used for distributed data stores with extremely large data storage needs. NoSQL is used for Big data and real-time web apps. For example, Facebook, Twitter, Google, etc., as these websites collect thousands of GB of data from users every day. (guru99, n.d.)

Relational databases define relationships in the form of tables. SQL (Structured Query Language) is the standard language for dealing with these relational databases. SQL programming can be adequately used to insert, search, update, delete database records. It can also be used to ensure efficient optimization and maintenance of databases. Relational databases like MySQL Database use SQL which enables the user to simply create multiple databases with different tables and attributes within each. It is quite straightforward to add data into a database using MySQL and overall, its easy to follow the process of how the data is stored within a relational database. (guru99, n.d.)

### Strengths and Weaknesses

* Relational databases are vertically scalable while NoSQL databases are horizontally scalable.
* Relational databases have a predefined schema whereas NoSQL databases use dynamic schema for unstructured data.
* Relational should be used over NoSQL when data validity is important for your database, whereas NoSQL should be used over Relational when want to prioritise fast data over correct data.
* Comparing NoSQL vs SQL performance, SQL requires specialized DB hardware for better performance while NoSQL uses commodity hardware.
* Relational databases should be used when you need to support dynamic queries, however NoSQL databases are used when you need to scale based on changing requirements.
* Relational databases offer strong consistency whereas only a few NoSQL databases offer strong consistency where the others only offer eventual consistency.
* NoSQL databases can be document based, graph databases, key-value pairs, whereas relational are only table based. So, depending on the type of database you want, the each have their advantages and disadvantages.

### Future Prospects

In terms of future prospects, it’s possible that NoSQL databases may be growing in popularity over relational. This seems to be the case from our research as we learned that many companies are expanding/thinking about expanding into the implementation of NoSQL databases over relational databases.

“According to a data and analytics survey Forrester conducted in 2016, 29% of tech decision makers said they have already implemented NoSQL technology, or are currently implementing it. Another 12% said they were expanding their implementations.” (little\_fish (admin), 2020)

Some of the reasons why these companies are considering the implementation of NoSQL databases over relational are because it enables elastic scale to support dynamic workloads, it supports flexible data models, it delivers impressive read and write capabilities for demanding customer apps, it simplifies data management for any type of application, and it also lowers data management cost. (little\_fish (admin), 2020)

Its not clear that NoSQL databases will take over in the future as it has both strengths and weaknesses as explained above, however it does look promising overall.

## Motivation for the use of NoSQL databases

Why should we use NoSQL databases?

* **“The pace of development with NoSQL databases can be much faster than with a SQL database.”**

Development speed is increased as structure of data is controlled by developers which increases efficiency among development with code pushes being less of a hassle and a positive evolution alongside more modern Agile database development.

* **“The structure of many different forms of data is more easily handled and evolved with a NoSQL database.”**

NoSQL is better suited across storing all types of databases from completely structured databases down to fully unstructured databases.

* **“The amount of data in many applications cannot be served affordably by a SQL database.”**

NoSQL databases are built on the foundation of handling big data, unlike SQL databases, which needs additional work in handling web-scale applications. Scalability is cheaper than SQL in large volumes of data and is more straightforward.

* **Traffic, downtime and paradigms.**

SQL databases cannot match the traffic and zero downtime associated with NoSQL databases alongside its scalability.

NoSQL can be both a transactional and analytical system while SQL databases are restricted. As NoSQL came to fruition during the era of cloud computing, is has adapted quickly in working alongside real time services

### Main characteristics of key-value store databases, column-oriented databases, graph databases and document store databases.

## How Redis Works (Odd)

Redis (**Re**mote **Di**ctionary **S**erver) is an efficient, open-source, in-memory data store for use as a database, cache, message broker, and queue. This project began when Salvatore Sanfilippo, the original developer, was trying to improve the scalability and flexibility of his startup. Redis now offers sub-millisecond response times which enables an extremely large number of requests per second for real-time applications within Healthcare, Financial Services, Gaming, etc. Redis is one of the most popular choices for procedures such as caching, real-time analytics, session management, gaming, leaderboards, etc. (AWS, n.d.)

All Redis data resides in-memory. This contrasts with databases that store data on disk or SSDs. By eliminating the need to access disks/hard drives, this gives in-memory data stores like Redis an advantage over others as it can avoid search time delays and can access data in almost instantly. Redis features versatile data structures, high availability, geospatial, Lua scripting, transactions, on-disk persistence, and cluster support making it simpler to build real-time internet scale apps. (AWS, n.d.)

### Key Features

**Transactions**

A transaction in Redis is unlike a transaction in a SQL database. It consists of a block of commands placed between MULTI and EXEC (or DISCARD for rollback). Once a MULTI has been found, the commands on that connection are not executed, instead they are queued. When an EXEC is found, they are all applied in a single unit (i.e. without other connections getting time between operations). If a DISCARD is seen instead of an EXEC, everything is removed. Because the commands inside the transaction are queued, you can’t make decisions inside the transaction. (StackExchange.Redis, n.d.)

There is also a WATCH command within Redis which tells the program that the user is interested in a specific key in the database. This is essentially how you can check different types of data and then make the appropriate changes after if needed. You can use UNWATCH when you’ve finished as this just forgets about the specified keys that were being watched.

**Recoverability**

Redis does support recoverability of your database if it somehow gets corrupted or your cluster fails. The steps to do so are quite simple and easy to follow:

1. If the cluster failed, then you must recover the cluster.

2. Identify your recoverable databases.

3. Restore the database with its previous configuration and data.

4. And finally verify that your databases are active.

(RedisLabs, n.d.)

How to do all of this is easily explained in more detail over on their ‘Recovering a failed database’ webpage, but overall, it is quite simple to follow. Redis does also offer ways to back up your databases in the event of the corruption of data.

**ACID/BASE Properties**

Redis is single threaded which means it allows itself to be ACID compliant. Other NoSQL databases generally don’t provide ACID compliance (they often use BASE properties instead), or they only provide it partially. Having ACID properties is a huge advantage for a database as it means that the data within remains accurate, authentic, and reliable.

**Concurrency**

Redis provides concurrency at an I/O level by using an I/O multiplexing mechanism and an event loop. The fact that Redis operations are atomic is simply an effect of the single-threaded event loop. The interesting point is atomicity is included at no extra cost (no synchronization required). This can be exploited by the user to implement assured locking and other patterns without having to pay for the synchronization expense.

**Scalability**

When you need to adjust your cluster size, various options are available within Redis to scale up and/or scale in or out. This allows your cluster to easily grow with your demands. This is a huge advantage overall as it provides consistent performance and reliability while still being able scaling your database in whatever way you like.

## How Neo4j Works

The most widely used type of graph database is Neo4j, which is a service implemented in Java. It was first released in February 2010 and is now under two types of license: a commercial license and also a general public license. It is developed by a company called Neo Technology, which is a Swedish start-up based in San Francisco. Companies such as eBay, Walmart, etc., have used Neo4j to improve their services.

Neo4j uses graphs to represent data and the relationships between them. Several different types of graphs can be used to display this data, such as:

– *Undirected graphs:* the nodes and relationships are interchangeable, their relationship can be interpreted in any way, e.g., Friend relationships on Facebook work in this way.

– *Directed graphs:* nodes and relationships are not bidirectional by default. Twitter relationships are a type of directed graph. For example, a user is able follow other users/profiles on Twitter without them following him/her back.

– *Property graphs:* a type of weighted graph with labels which allows you to assign attributes to both nodes and relationships (e.g., name, age, nationality, etc.). This is the most complex yet most useful graph.

Neo4j uses property graphs to extract added value of data of any company with great performance and in an agile, flexible, and scalable way. (BBVA API MARKET, 2015)

### Key Features

**Transactions**

Neo4j uses DBMS transactions. Starting a transaction while connected to a DBMS will start a DBMS-level transaction. This is a container for database transactions. A database transaction has begun when the first query to a specific database is issued. Database transactions opened within a DBMS-level transaction are committed/rolled back when the DBMS-level transaction is committed/rolled back.

**Recoverability**

If a data center somehow becomes ‘lost’, for example, you can’t add instances to the current data center, and can only use the current read-only cluster, then Neo4j has you covered. There are 2 options available for recovery depending on the situation you are in; both are easy to follow and will enable you to restore your database swiftly and easily. Neo4j also provides support for replication for data safety and reliability (make a backup of your database in the case of any problems).

**ACID/BASE Properties**

Neo4j supports full ACID transactions. This is the best way to maintain the accuracy and reliability of a database. This ultimately allows the authenticity of data to be done so that it doesn’t become corrupted in any way. You can also plan your database around BASE (Basic Availability, Soft-State, Eventual Consistency) properties; however, this limitation is probably a big disadvantage when compared to the simplicity, reliability, and consistency of ACID transactions.

**Concurrency**

Neo4j does have concurrency control. It uses locks for transactions which may lead to deadlock. If a deadlock is detected, then the transaction is marked for rollback. The transaction may be reinitialized if needed by the user. This decision to retry is left up to the user.

**Scalability**

You can scale the database by increasing the number of reads and/or writes, and the volume without effecting the query processing speed and data integrity. This is a huge advantage because when needs increase, the possibilities of adding more nodes and relationships to an existing graph will not hinder or compromise the performance and structure of the database. Neo4j also provides support for replication for data safety and reliability.

## How Cassandra Works (Even)

Cassandra is a NoSQL database. It is a free, open-source database management system (DMS). The DMS is designed and used in the management of large amounts of data across multiple servers. Cassandra is widely used among some of the biggest companies and brand in the world such as Apple, Spotify, Uber, McDonald’s, Microsoft and many more, “Cassandra is used by 40% of the Fortune 100”. (Apache Software Foundation, 2016)

### Key Features

**Types of Data –**

Cassandra supports multiple data types, such as:

* **Built-in Data Types –** Also known as “primitive data types”. They can be directly referred to as they are pre-defined, including but not limited to, Boolean, decimal, double and int.
* **Collection Data Types –** These are for collection of data through storing multiple values in one unit. Cassandra supports three types of collection:
  + **Maps** - storing data with key names for convenience
  + **Sets -** multiple unique values stored, but not in order
  + **Lists** - can store duplicate values and multiple values in specific order.
* **User-Defined Data Types –** Creating own data type around what’s needed. Multiple data fields of any type in one column. Data fields can be changed or removed as needed.

(TutorialsPoint, 2021)

**Query Language –** Similar to SQL with data stored in rows and columns within tables. Cassandra’s equivalent to SQL is known as Cassandra Query Language or CQL. It is used within the Cassandra database in creating, inserting and managing data within tables.

**DBaaS (Database as a Service)**. - Although available elsewhere I will be using DataStax for an example. DataStax Astra provides “cloud-native” application development for Cassandra to help simplify and encourage the process of application development alongside providing automatic backups and cloud storage to make development easier, accessible and scalable for companies.

(DataStax, 2021)

**Cost –** Cassandra is free and open-source.

**Security –** Cassandra provides three types of security:

* **TLS/SSL Encryption** – Secure communication between client and a database cluster (multiple servers connected to one database) as well as secure communication between nodes within the cluster.
* **Client Authentication** – Accessing a server securely using a password/login.
* **Authorisation** – Such as setting permission and restrictions for users, similar to an admin or teacher in a school setting.

(Apache Software Foundation, 2016)

## How HBase Works (Even)

HBase is an open-source non-relational database. It is written in Java and modelled with direct inspiration after Google’s Bigtable (Google’s data storage system). HBase does not support a query language such as SQL. HBase is column-oriented; attributes are grouped together into column families. This is different from relational databases that are row-oriented, as the columns of a row are all stored together.

### Key Features

**Types of Data –** HBase’s data types are saved in bytes and bits, with data types ranging from a byte of a variable length (OrderedBlob) to RawDoubles and RawIntegers.

**Query Language –** As HBase is a non-relational data store, HBase “does not support a structured query language such as SQL”, unlike Cassandra which is very similar to MySQL. HBase is written in Java.

**DBaaS –** HBase lacks in this department, especially in comparison with another databases such as MongoDB which offer very comprehensive DBaaS service for their databases or even Cassandra that has the Astra DBaas.

**Cost –** HBase is freeand open-source.

**Security –** HBase has a good standard of security, as previously stated with Cassandra, HBase has similar security levels with User Authentication, secure communication and user authorisation through granting and revoke permissions.

## Conclusion

# Bibliography

Neo4j

<https://www.tutorialspoint.com/neo4j/neo4j_overview.htm#:~:text=ACID%20properties%20%E2%88%92%20Neo4j%20supports%20full,processing%20speed%20and%20data%20integrity>.

<https://www.bbvaapimarket.com/en/api-world/neo4j-what-graph-database-and-what-it-used/>

<https://neo4j.com/docs/cypher-manual/current/introduction/transactions/>

<https://neo4j.com/docs/operations-manual/current/clustering-advanced/multi-data-center/disaster-recovery/>

Redis

<https://aws.amazon.com/redis/>

<https://dalibornasevic.com/posts/60-redis-in-the-nosql-ecosystem>

<https://stackexchange.github.io/StackExchange.Redis/Transactions.html>

<https://stackoverflow.com/questions/10489298/redis-is-single-threaded-then-how-does-it-do-concurrent-i-o>

Part 1.

<https://www.guru99.com/sql-vs-nosql.html>

https://forum.huawei.com/enterprise/en/sql-vs-nosql-databases-is-nosql-the-future-of-databases/thread/660677-899#:~:text=Relational%20databases%20are%20often%20referred%20to%20as%20SQL%20systems.&text=For%20general%20research%2C%20NoSQL%20databases,to%20their%20speed%20and%20flexibility.