Database Systems Assessment 2 Report

NoSQL

"Describe and critique what is meant by the NoSQL term 'Dynamic Schema', and what benefits could the use of a Dynamic Schema have over relational database Fixed Schemas, such as that currently used in your SQL database solution?"

The difference between SQL fixed schemas and NoSQL 'Dynamic Schemas' are described as "a (logical) schema is fixed if it is defined before a program is written, but dynamic if it is defined by the program or data itself." (dbms2, 2011). This in itself gives only a partial explanation; the bigger difference is that a dynamic schema will grow with the business instead of being fixed with a highly structured table organisation which my SQL database solution is currently using. This is fine for the current format with the same rigidly-defined data formats and record structure, especially as mine only current holds 3 records for each table but a larger database if the business was real and grew to develop more services and other dimensions then this fixed schema would be unlikely to be sustainable. A company simply selling t-shirts as their only stock would be able to sustain this type of schema but a large department store with multiple different tables with different stock and column numbers would require a dynamic schema to be able to go through their data using different types of JOINs as a fixed schema would increase in complexity and computing resource power and be unsustainable. Another problem that a business using a fixed-schema might run into is frequent schema changes, "At a time when business agility is at a premium, requests for changes are more often than not put off by DBAs because the schema of relational databases isn't designed for frequent changes and pivots. Common schema changes indicate that the data or requirements are rapidly evolving, calling for a more flexible model." (neo4j, 2015). This would be detrimental to an SME or large business and their database efficiency as a change to the schema would require considerable work if it was a frequent change.

The benefits to my own database system would allow me to create new table if the business grew to sell or provide different services and then I could split the services and stock to different tables to separate them and make it easier to access them with more effective column names. A NoSQL approach such as MongoDB would allow me to turn the data into documents with a specific id given to them, this would be put into a collection and used to scale up with the number of servers to save money. The speed of using the BASE principles rather than the ACID principles are also an advantage as it's much quicker and more efficient even if it is losing out on some reliability that ACID can give you. This is evident in the story of Flexcoin losing 896 BTC (equivalent to £500,000) because of the limited security that the NoSQL offers and not being transaction critical. (hacking distributed, 2014). However, for my database it would suit a fixed-schema until the size of the business grew to a level where the number of products and services outweighed the capabilities of a fixed-schema solution.

Database Security

"Using your own relational database solution as context, describe and critique what an SQL Injection Attack is, and what steps you can take to protect your database solution from such an attack?"

An SQL injection is the process of unlawfully gaining access to a database through a login system comprising of an email and password for example and using simple commands to obtain any of the information in the database as you please. An official example is shown here "An SQL injection is a computer attack in which malicious code is embedded in a poorly-designed application and then passed to the backend database. The malicious data then produces database query results or actions that should never have been executed. (Techopedia, 2019). In relation to my database, it could easily be accessed through a login system as it currently has no security to stop unlawful access, this could be changed by separating the code from the data using prepared statements which are effectively setting up simple 'prepare' commands and 'set' commands which mean that the code for several queries and the data that is entered by the user on the login screen are entirely separate and means they can only access the data which the specific email and password can access. These changes in code are very little and can make a huge impact to the overall security of the entire system. However, they can't provide complete security as users can bypass these 'prepared' statements, therefore, you have to use other methods such as stored procedures, these are defined as "A stored procedure is a set of Structured Query Language (SQL) statements with an assigned name, which are stored in a relational database management system as a group, so it can be reused and shared by multiple programs." (SearchOracle, 2017). The use of this means that you are only allowed to execute statements but not modify them as they don't work like SQL queries and can bypass the ability to access the query code and then steal data. These can be more time consuming to first program, but they can be reused in the code multiple times so can save you time overall in the end. Due to my system being relatively simple, the use of stored procedures would be ideal especially if the business was to grow as then it would mean I could reuse the stored procedures and not have to have extensive code files. I would need to combine this with 'prepared' statements as well to achieve a comprehensive security.

References:

1) Terminology: Dynamic- vs. fixed-schema databases
Available at: www.dbms2.com/2011/07/31/dynamic-fixed-schema-databases/
(Accessed: 06/01/19)

2) 5 Sure signs it's time to give up your relation database
Available at: https://neo4j.com/blog/five-signs-to-give-up-relational-database/
(Accessed: 07/01/19)

3) NoSQL meets Bitcoin and brings down two exchanges: The story of Flexcoin and Poloniex
Available at: http://hackingdistributed.com/2014/04/06/another-one-bites-the-dust-flexcoin/ (Accessed at: 07/01/19)

4) Definition – What does SQL injection mean?
Available at: https://www.techopedia.com/definition/4126/sql-injection
(Accessed: 07/01/19)

5) Definition: Stored Procedure
Available at: https://searchoracle.techtarget.com/definition/stored-procedure
(Accessed: 07/01/19)