

Database Solutions for Pronto Parcels and Eco Bike Hubs

Abstract: This report examines the differences between SQL and NoSQL database solutions. It also explores Performance tuning techniques that can be implemented into an SQL database in the form of Secondary indexes and Transactions.

Table of Contents

Introduction	2
SQL vs NoSQL.....	2
Overview of SQL	2
Advantages of SQL:.....	2
Disadvantages of SQL:	3
Overview of NoSQL	3
Advantages of NoSQL:	3
Disadvantages of NoSQL:	3
Conclusion.....	3
Eco Bikes Hub	4
Performance Tuning	4
Transactions and Secondary Indexes	4
Transaction Optimisation.....	5
Overview of Transactions	5

Introduction

Databases are a crucial component within the functionality and scalability of applications. When considering an approach to what database to use, SQL or NoSQL, the impact to performance, development and maintainability are highly important. Correctly identifying which system will contribute to the overall success will require an in-depth analysis of the system requirements.

SQL (Structured Query Language) databases have been an industry standard for many crucial systems. They are valued for being a reliable and structured approach to managing data. They are utilised in financial and transactional systems [1].

NoSQL (Not Only SQL) has become an increasingly adopted approach in recent years. It excels in managing large unstructured datasets and rapidly evolving development cycles. Applications such as social media sites and real-time applications benefit significantly from the high traffic on these systems.

This report aims to highlight and compare the advantages and disadvantages of each system and its best use case. Additionally, this report will address some concerns raised by the company Eco Bikes, suggesting some solutions around performance tuning within their system and transactions to resolve problems currently encountered.

SQL vs NoSQL

Overview of SQL

SQL databases are widely considered traditional databases and use a relational model, relying on tables with predefined columns and relationships. Each table has a fixed structure and are connected through primary and foreign Keys. They follow a rigid predefined schema, which are documented using ERDs the tables, columns, primary and foreign keys, relationships and constraints. SQL has been relied on for data management by many industries for its reliability, consistency, and maintainability.

Advantages of SQL:

- **ACID Compliant:** SQL adhere to the ACID Principle, Atomicity, Consistency, Isolation and Durability. Following these principles provides confidence in the reliability of the data. These characteristics are why it has been a keystone in data storage throughout multiple industries. [2]
- **Standardisation:** SQL is a standard language which provides consistency across systems. However, there is a slight deviation between SQL systems such as MySQL [3], Postgres [4], and Microsoft SQL Server [5]. Each has its own merits and are best suited to different systems.
- **Support and Maintenance:** SQL benefits from decades of use. Systems have relied on SQL, which has bred a wide range of knowledge and detailed documentation. The tooling surrounding SQL is mature, robust, and highly reliable.

Disadvantages of SQL:

- **Scalability:** SQL suffers in terms of scalability. It is designed to be vertically scalable by adding more resources to a single server. This can be detrimental to the performance of high-traffic systems or in Vast data sets, where horizontal scaling would be beneficial. [2]
- **Unstructured Data:** SQL is designed with rigid predefined Schemes. When presented with data of this nature, it excels. When SQL is presented with Large Unstructured datasets, it becomes cumbersome, often requiring additional tools to manage this data successfully. [6]

Overview of NoSQL

NoSQL databases do not follow a relational model, providing highly flexible design and supporting data that is unstructured, semi-structured and structured. NoSQL supports a variety of data models such as Document, column-family, graph, and key-values. These characteristics make it ideal for rapidly evolving systems where data requirements are likely to change. Unlike SQL, it is considered schema-less, which benefits its ability to meet rapid change of requirements, however, each type has its own structure which determines how it stores data.

Advantages of NoSQL:

- **Flexibility:** NoSQL does not use rigid predefined schemes, which allows it to manage unstructured, semi-structured, and structured data. This is beneficial for evolving system requirements where data structures are likely to change [2].
- **Scalability:** Unlike SQL, NoSQL is designed with horizontal scalability in mind. The ability to distribute the data across multiple nodes means extensive storage capabilities and the ability to handle large amounts of traffic [7].
- **Rapid Development:** Developers benefit from NoSQL databases as they can quickly and easily iterate applications without the need to work with complex schemas. This reduces the time required to change requirements, reducing development challenges [8].

Disadvantages of NoSQL:

- **Limited Maturity:** NoSQL is considerably newer compared to SQL. Due to this there are potentially drawbacks when considering data reliability, such as ACID compliance. When considering NoSQL systems, they may also lack compatible tooling in less supported databases [9].
- **Standardisation:** NoSQL does not follow a standard language such as SQL. Due to this, there is less knowledge surrounding the various databases. A lack of consistency between the languages used by the systems can introduce unintended issues. Such as creating difficulties in working across databases or in recruitment of experienced developers [8].

Conclusion

The decision between SQL and NoSQL is heavily dependent on the requirements of the system being developed. Determining the best approach requires an

understanding of the data being provided, how it will be manipulated and the use case of the users.

- **SQL:** When requirements demand data consistency, structure and support, SQL remains the best solution. Financial systems, transactional systems and many legacy systems remain reliant on relational databases with structured data models.
- **NoSQL:** The flexibility offered to modern day development benefits from NoSQL. It is an ideal solution for mass data that is unstructured or rapidly evolving. Systems such as social media sites and analytics benefit in a NoSQL database. Furthermore, horizontal scaling of the database benefits these applications due to high demand, where a shared load reduces the risk of performance issues.

Eco Bikes Hub

Performance Tuning

Transactions and Secondary Indexes

The most common transactions that will be run against the database will include searches and inputs. Searching customer records will be essential towards invoicing or loyalty programs. Searching through the parts inventory will allow employees to ensure enough parts are available for repairs and sales. Additionally, it will allow staff to advise customers on alternative branches that currently have the parts available. With bike rentals being a key service of EcoBikes, searching the rental start dates will be vital in monitoring which bikes are currently in use or when they will be unavailable if already booked in the future. Using these scenarios, three secondary indexes can be identified, `customer_name`, `parts_name`, and `rental_start_date`.

- **Rental Start Date:** Indexing the rental creation will allow for optimised search queries [10], which will be essential in managing which bicycles are currently in use or going to be unavailable during specific date ranges. Additionally, the performance of these queries could drastically increase, especially if querying against all stores. Additional benefits from this indexing could also retrieving the peak rental periods, and in inputting new rentals where the transaction will be searching against other attributes within the database.
- **Part Name:** There are many queries that could benefit from indexing the part name. Primarily on search queries[11], which will affect the daily operation of their service centres and storefronts. Additionally, creating a composite index of `part_name` and `bicycle_id` would allow for tracking which bikes are stored and which parts require which. This would highly benefit the service and repair workflows to enable technicians to ensure they are using the correct parts when dealing with customers' bicycles. Indexing also offers significant benefits in scalability, with parts contributing as one of the more extensive data sets due to the requirement of different parts to maintain different bikes.
- **Customer Name:** Querying customers is essential throughout the services offered by EcoBikes. By indexing the `customer_name`, it would provide benefits to searching customer records when raising invoices and creating rentals or

loyalty schemes. Additionally, it would benefit the performance of customer lookups and filtering. This would be essential in-store for creating bookings or rentals on behalf of customers

Conclusion

Implementing secondary indexes will significantly enhance the system's efficiency and performance in data input and retrieval [9], [10]. Carefully designed indexes, such as `rental_start_dates`, `part_name`, and `customer_name`, will optimize transaction times, especially when handling large data sets. These performance improvements will streamline daily operations for staff, ensuring quicker searches, better inventory management, and enhancing customer experience. Additionally, secondary indexes contribute to maintaining accuracy and consistency across records, supporting the scalability and reliability of the system as EcoBikes continues to grow.

Transaction Optimisation

Overview of Transactions

Sometimes bicycle repair bookings are recorded but the details of the specific return date(s) as requested by customers are not registered in the system.

I have implemented a transaction within a trigger function that checks before creating an input. The transaction utilises two If statements. The first will check if the rental return date is null, if true, it will then create a new repair date, which takes the creation date and adds 7 days to the record. It will notify the user that a date is missing, and the default repair duration has been used. This allows the user to update the record with the correct date if required. Due to how I created this transaction and constrained my tables, there should not be a rollback event due to user error, only a system error.

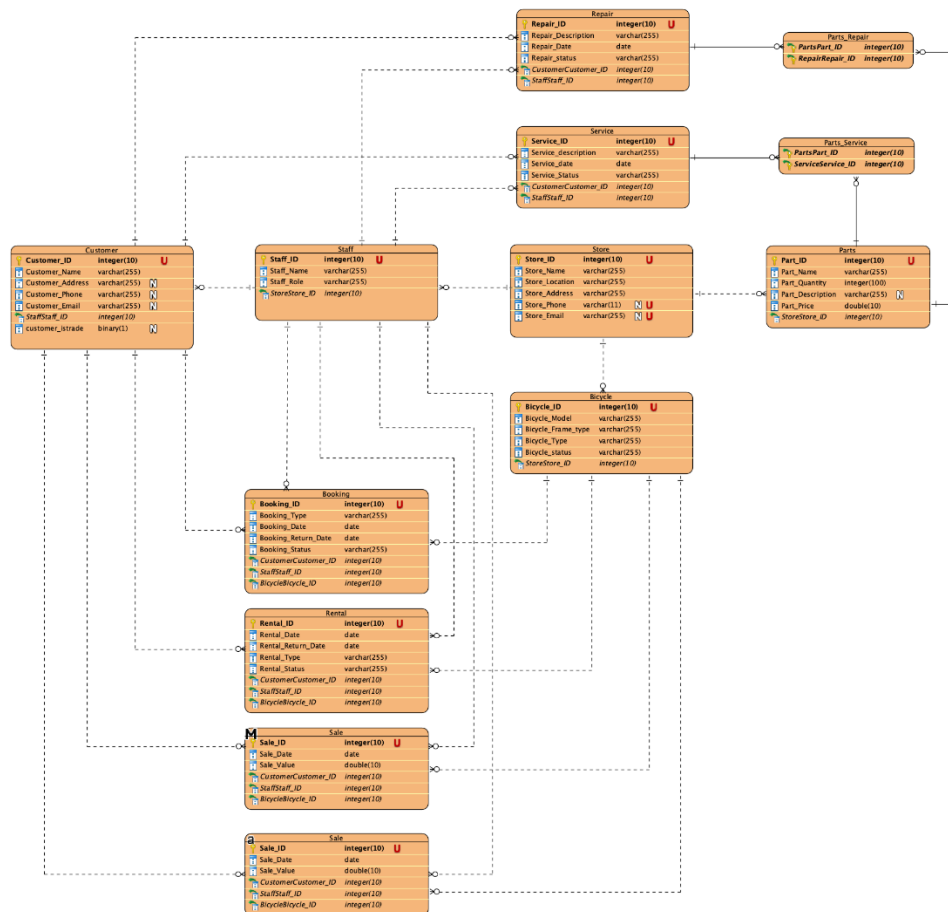
Customers are able to book bicycle hire times that are not registered as possible schedules available.

I have implemented a transaction within a trigger function that executes before an insert is made. The transaction selects all records into a temporary table that matches the bicycle ID input, where the input creation date and input return date are between the stored dates. I then use an if statement if a record is found. It raises an exception and notifies the user that the input bicycle is already booked between the found conflicting dates and rolls back.

Due to the transactions occurring within trigger functions, Postgres automatically handles the rollback or commit in compliance with ACID.

Appendix

1. EcoBikesHub ERD



2. [mockaroo. (n.d.). <https://www.mockaroo.com/>.

mockaroo was used to generate the mock data required for some tables within my database. Only the data generated was used and assigned to my written insert statements.

Reference List

- [1] Y. Xu, 'Research on the Application of SQL in Corporate Finance', *Frontiers in Business, Economics and Management*, vol. 16, no. 2, pp. 1–7, Aug. 2024, doi: 10.54097/dj8pnp70.
- [2] S. Binani, A. Gutti, and S. Upadhyay, 'SQL vs. NoSQL vs. NewSQL- A Comparative Study', *Communications on Applied Electronics*, vol. 6, no. 1, pp. 43–46, Oct. 2016, doi: 10.5120/cae2016652418.
- [3] 'MySQL', <https://www.mysql.com/>.
- [4] 'PostgreSQL', <https://www.postgresql.org/>.
- [5] 'Microsoft SQL Server', <https://www.microsoft.com/en-us/sql-server/sql-server-downloads>.
- [6] W. Khan, T. Kumar, C. Zhang, K. Raj, A. M. Roy, and B. Luo, 'SQL and NoSQL Database Software Architecture Performance Analysis and Assessments—A Systematic Literature Review', Jun. 01, 2023, *MDPI*. doi: 10.3390/bdcc7020097.
- [7] J. E. Pagán, J. S. Cuadrado, and J. G. Molina, 'A repository for scalable model management', *Softw Syst Model*, vol. 14, no. 1, pp. 219–239, Feb. 2015, doi: 10.1007/s10270-013-0326-8.
- [8] J. Li and J. Li, 'Research on NoSQL Database Technology', in *Proceedings of the 2018 2nd International Conference on Management, Education and Social Science (ICMESS 2018)*, Paris, France: Atlantis Press, 2018. doi: 10.2991/icmess-18.2018.252.
- [9] S. Ramzan, I. S. Bajwa, R. Kazmi, and Amna, 'Challenges in NoSQL-Based Distributed Data Storage: A Systematic Literature Review', *Electronics (Basel)*, vol. 8, no. 5, p. 488, Apr. 2019, doi: 10.3390/electronics8050488.
- [10] M. ANCA, P. CERASELA, and C. Etegan, 'INDEXING STRATEGIES FOR OPTIMIZING QUERIES ON MYSQL', *Annals of the University of Petrosani : Economics*, vol. X, Jan. 2010.
- [11] T. Oktavia and S. Sujarwo, 'Evaluation of Sub Query Performance in SQL Server', doi: 10.1051/C.
- [12] mockaroo. (n.d.). <https://www.Mockaroo.Com/>.