## Introduction

Transport for London’s (TfL) public transport system generates significant volumes of data across services like stations and vehicles. Managing this data efficiently is crucial to ensure smooth operations and timely updates to passengers. Our objective was to design a secure and scalable database that can be used to manage the TfL’s transportation network data. Using a relational data model, the database was built to capture, clean, and store data from the TfL API in a normalised format to ensure efficiency and reduce redundancy.

This executive summary outlines the work carried out in this project, critically reviews the chosen database model, evaluates the database management system (DBMS) chosen, and assesses legal and compliance requirements for managing passenger and operational data, particularly GDPR compliance (European Union, 2016).

## Summary of Work Carried Out

### Overview of the Database Design

The database has been designed to manage core operational data for Transport for London (TfL), focusing on entities such as Stations, Routes, Vehicles, and Schedules. These entities are integral to TfL’s transportation network, and their relationships were defined using a relational model with primary and foreign key links (Elmasri and Navathe, 2016).

* **Stations:** Attributes include ***station\_id*** (primary key), ***station\_name***, ***latitude***, ***longitude***, and ***station\_type***.
* **Routes:** Attributes include ***route\_id*** (primary key), ***start\_station\_id*** (foreign key), ***end\_station\_id*** (foreign key), ***route\_name***, ***transport\_mode***.
* **Vehicles:** Attributes include ***vehicle\_id*** (primary key), ***route\_id*** (foreign key), ***vehicle\_type***, and ***capacity***.
* **Schedules:** Attributes include ***schedule\_id*** (primary key), ***route\_id*** (foreign key), ***station\_id*** (foreign key), ***arrival\_time***, ***departure\_time***, ***day\_of\_week***, and ***service\_frequency***.

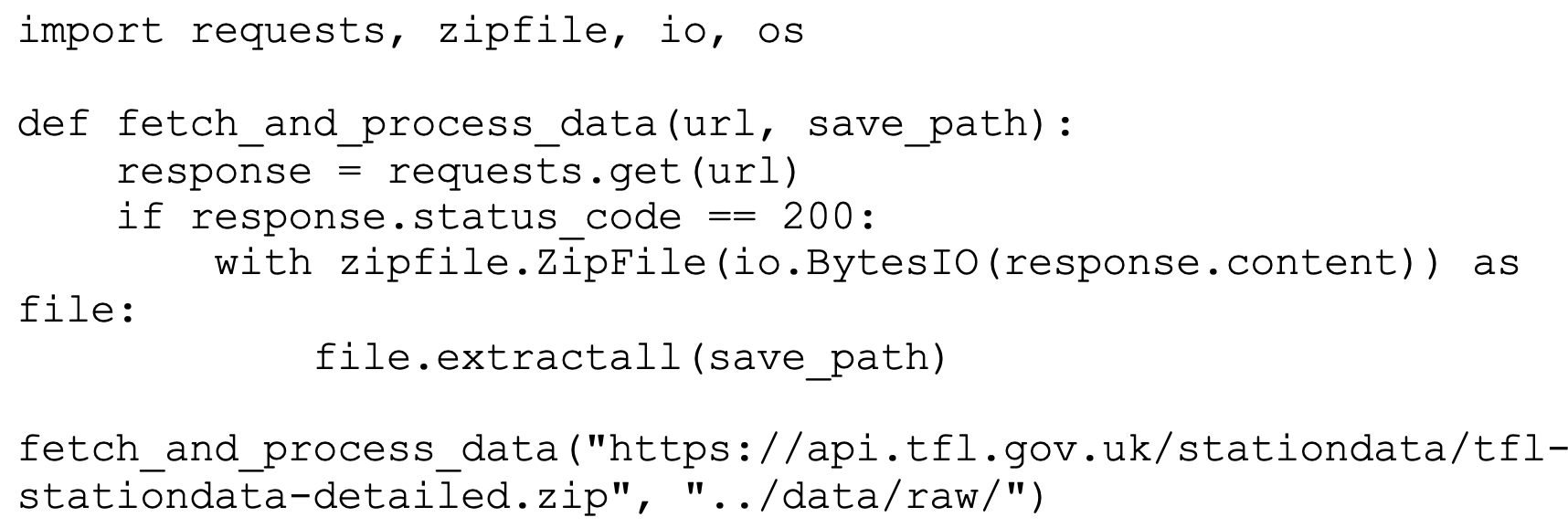
The relationships between each entity are shown below in the Entity-Relationship Diagram (ERD). The primary keys ensure unique identification of records; foreign keys maintain the integrity of the links between tables. For example, the Routes table uses the station\_id foreign key to link each route to its starting and ending stations, while the Vehicles table connects to the Routes table via route\_id (Kimball and Ross, 2013).

A screenshot of a computer

Description automatically generated

### Data Management Pipeline

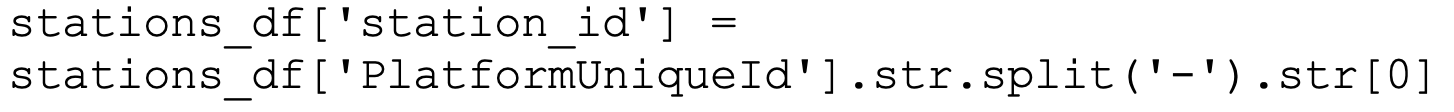
Data for this project was retrieved using the TfL Unified API (Transport for London, 2024), which provides real-time and static data on stations, routes, vehicles, and schedules. The data was fetched using Python’s requests library (McKinney, 2012), which allowed automated retrieval of data in both CSV and ZIP formats. Below is an example of how the data was fetched from the API:



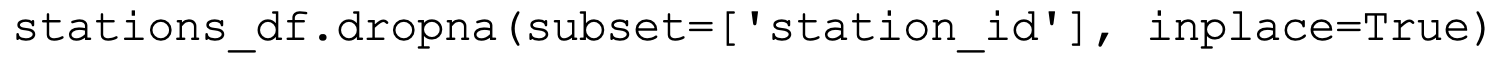
### Data Cleaning Process

Once the data was captured, it was processed to prepare it for storage in the database. This involved several cleaning steps to ensure data integrity and usability:

**Foreign Key Extraction:** In some cases, primary and foreign keys were embedded in composite fields. For example, the station\_id needed to be extracted from the PlatformUniqueId field. This was done using string operations in Pandas (McKinney, 2012):



**Handling Missing Data:** Missing values were handled in a case-by-case manner. Where critical data, such as station\_id, was missing, records were dropped. However, for non-crucial fields like certain location details, missing data was imputed where possible:



**Standardising Data Formats:** Data type consistency was ensured by converting numerical values (such as latitude and longitude) into the correct types (in this case; floats). This helped prevent issues when importing the data into the database (Kimball and Ross, 2013):

A black text on a white background

Description automatically generated**Removing Duplicates and Outliers:** Duplicate entries, such as station names and route information, were removed to ensure no redundant data was stored. Outliers, such as invalid vehicle capacities or incorrect station coordinates, were identified and adjusted:The cleaned data was then transformed into normalised tables (Codd, 1970), with each table reflecting a distinct entity, and foreign key relationships ensuring that data could be linked efficiently across tables.

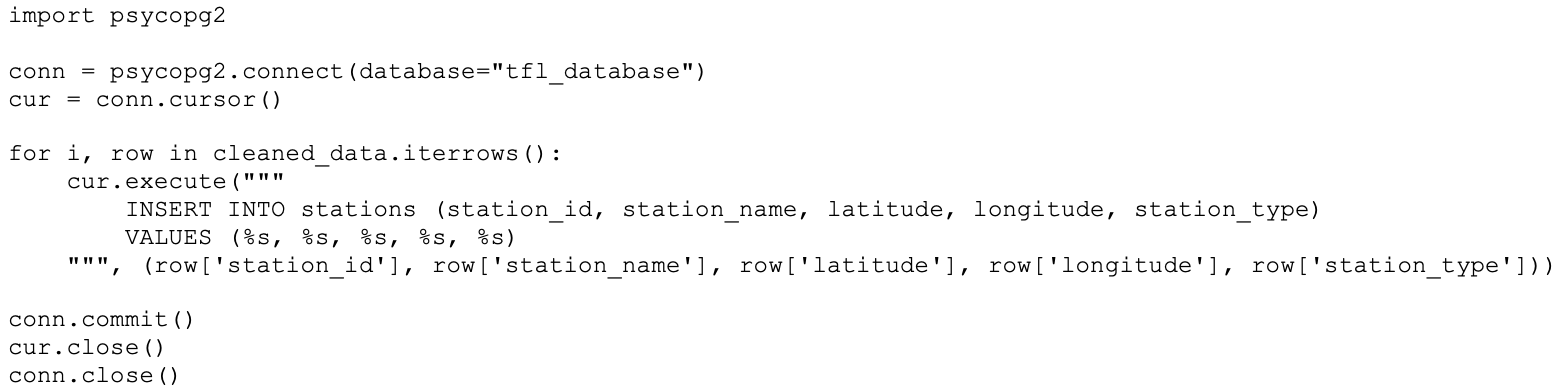
### Data Storage Preparation

After cleaning, the data was prepared for insertion into a PostgreSQL database. SQL queries were generated to create tables for each entity, ensuring proper data types and constraints for each column. For instance, the Stations table was created as follows:

A close-up of a text

Description automatically generated

The cleaned data was inserted into these tables through Python scripts using the Psycopg2 library for PostgreSQL, ensuring that data integrity was maintained during the import process:



### Challenges Faced

Several challenges were encountered during the project, such as missing and inconsistent data, which were resolved through cleaning and standardisation. The API’s rate limits were handled by splitting data retrieval into smaller batches and retrying failed requests. Outliers, like incorrect vehicle capacities, were corrected or removed during the cleaning process.

Despite these challenges, the data was successfully cleaned, transformed, and stored in a structured and efficient manner within the database, ensuring that TfL’s operational data could be managed effectively.

## Critical Review of Database Modelling Concepts

### Relational Data Model

### The TfL database uses a relational data model, which is well-suited for managing structured data with clear relationships between entities. Data is organised into tables, and primary and foreign keys ensure data integrity. The relational model also supports ACID properties (Atomicity, Consistency, Isolation, Durability) to guarantee transaction reliability (Elmasri and Navathe, 2016). For example, the Routes table links to the Stations table via station\_id, enabling TfL to track routes and vehicles efficiently.

### Strengths of the Relational Model

Support for data integrity and consistency through ACID properties (Codd, 1970) is one of the key strengths of the relational model. This ensures that transactions, such as updating or inserting data, are processed reliably. For example, if a new station is added or a route is updated, ACID compliance guarantees that these changes are either fully committed or fully rolled back if an error occurs. Additionally, the relational model allows for complex queries using SQL, which is crucial for TfL’s operations, where large datasets need to be queried for specific information, such as vehicle schedules or station locations.

The use of normalisation in relational databases minimises redundancy and prevents data anomalies by dividing the data into several related tables. In this project, each entity (Stations, Routes, Vehicles, and Schedules) was stored separately, ensuring that updates in one table would not result in inconsistencies across the database (Kimball and Ross, 2013).

### Weaknesses of the Relational Model

Despite its strengths, the relational model does have some limitations. One such limitation is scalability. As TfL’s operations expand, the volume of data generated is likely to increase significantly. Relational databases, while robust, can experience performance bottlenecks when handling extremely large datasets or complex queries. Additionally, the relational model’s strict schema can make it less flexible when dealing with unstructured or semi-structured data. This is becoming increasingly common in large-scale organisations like TfL.

### Comparison with NoSQL

In contrast, NoSQL databases, such as MongoDB, offer more flexibility in handling large volumes of unstructured data. They allow for horizontal scaling, which enables the database to grow as data increases. However, NoSQL databases sacrifice the ACID properties and data consistency that relational databases offer, which is a key requirement for TfL’s operational data. Given TfL’s need for consistency and structured data management, the relational model is better suited for this project (Narizhnykh, 2023).

## Analysis of the DBMS Choice

### SQL vs NoSQL

For this project, a relational database management system (RDBMS) was chosen due to the structured nature of TfL’s operational data. TfL’s data—such as stations, routes, vehicles, and schedules—is inherently structured and contains well-defined relationships, making a relational DBMS like PostgreSQL ideal.

While NoSQL databases like MongoDB offer advantages in terms of horizontal scalability and flexibility in handling unstructured data, these advantages were less relevant for this project. TfL’s need for structured queries, data integrity, and reliable transactions made SQL databases a better fit (Narizhnykh, 2023).

### Why PostgreSQL?

PostgreSQL was selected due to several key features:

**ACID Compliance**: PostgreSQL supports ACID properties (Atomicity, Consistency, Isolation, and Durability), which is crucial for ensuring integrity in TfL’s operational data, such as schedules and routes (Sheldon, 2023).

**Scalability:** While relational databases generally face challenges in horizontal scalability, PostgreSQL includes features like table partitioning and replication that allow it to handle large datasets more efficiently (PostgreSQL Documentation, n.d.).

**Complex Queries:** PostgreSQL is well-suited for complex queries, essential for querying TfL’s data on stations, routes, and schedules.

**Open-Source and Extensible:** PostgreSQL is open-source, making it cost-effective. It also supports extensions like PostGIS for handling spatial data, which is useful for managing station locations and vehicle routes (Narizhnykh, 2023).

### Alternative DBMS Considerations

MySQL is another popular RDBMS, but it was not selected due to its more limited handling of complex queries and data types. While NoSQL databases, such as MongoDB, provide better horizontal scalability, they sacrifice data integrity and consistency, making them less suitable for TfL’s structured data and operational needs (Smallcombe, 2023).

### DBMS Analysis Conclusion

In conclusion, PostgreSQL was chosen for its strong support of ACID transactions, complex queries, and data integrity. While NoSQL systems offer scalability benefits, the structured nature of TfL’s data and the need for reliable relationships between entities made PostgreSQL the ideal solution.

## Legal and Compliance Considerations

### GDPR Compliance

Handling operational data for TfL requires compliance with the General Data Protection Regulation (GDPR), ensuring proper data handling and privacy (European Union, 2016). To meet these standards, the following steps were implemented:

* **Data Minimisation:** Only essential operational data is collected (e.g., stations, routes), avoiding unnecessary personal data, thus reducing risk.
* **Data Encryption:** Sensitive information, such as vehicle or station data, is encrypted both at rest and during transmission to prevent unauthorised access.
* **User Rights:** Although this project focuses on operational data, the database design supports GDPR rights such as data access, rectification, and deletion if personal data is added in the future.
* **Data Retention:** Policies are established to make sure data is kept only for as long as necessary, and deleted securely once it is no longer required.

### Data Security and Access Control

Data security is essential for protecting TfL’s operational data from breaches or unauthorised access. Key security features include:

* **Role-Based Access Control (RBAC):** Access to data is restricted based on user roles. For example, only authorised personnel can access sensitive vehicle schedules or station operations.
* **Database Encryption:** PostgreSQL offers encryption at rest and in transit, securing operational data throughout its lifecycle (PostgreSQL Documentation, n.d.).
* **Audit Logs:** Detailed audit logs are maintained to monitor all database access and changes, providing transparency and ensuring accountability.

### Compliance with Data Protection Laws

In addition to GDPR, the database aligns with the UK Data Protection Act 2018, ensuring that any future integration of personal data (e.g., passenger information) complies with UK laws. Although this project focuses on operational data, protecting this data is equally important to prevent service disruptions and unauthorised changes that could affect public safety.

By implementing GDPR compliance measures, including data minimisation, encryption, and access control, TfL’s database is designed to handle both operational and personal data securely. These practices ensure compliance with legal requirements and protect TfL against potential data security risks.

## Conclusions and Recommendations

### Conclusions

The design and implementation of the TfL database focused on managing core operational data, such as stations, routes, vehicles, and schedules. By using a relational database model (PostgreSQL), the project ensured that data integrity, complex querying, and scalability were addressed effectively. Key conclusions from the project include:

* **Effective Data Management:** The use of PostgreSQL enabled TfL to maintain relationships between its various data entities, ensuring consistent and accurate data retrieval through ACID compliance and foreign key constraints.
* **Data Security and Compliance:** The project integrated GDPR principles, such as data minimisation, encryption, and role-based access control (RBAC), ensuring that TfL’s operational data remains secure and compliant with legal requirements.
* **Handling Data Challenges:** Through effective data cleaning and normalisation, the database successfully addressed challenges such as missing data, inconsistent data types, and API rate limits. This ensured that the data was structured for efficient querying and minimal redundancy.

### Recommendations

Moving forward, several recommendations can be made to further enhance the database and its capabilities:

1. **Integration of Personal Data:** As the database currently focuses on operational data, future integration of passenger data may be necessary. In such cases, additional security measures, including enhanced encryption and advanced access controls, should be implemented to protect sensitive personal information.
2. **Scalability Improvements:** While PostgreSQL provides solid scalability features, as TfL’s operations grow, further scalability strategies, such as sharding or horizontal partitioning, may be required to handle increasing data volumes and maintain performance during peak operations.
3. **Enhanced Reporting and Analytics:** Incorporating business intelligence (BI) tools such as Power BI or Tableau could improve TfL’s ability to analyse data and generate operational insights, helping the organisation optimise routes, vehicle scheduling, and overall service efficiency.
4. **Monitoring and Auditing:** Regular audits of the database, focusing on data access and usage, will ensure that security measures are continually evaluated and updated. Automated monitoring tools can be deployed to detect anomalies or potential security breaches in real-time.
5. **Further Expansion of Spatial Data:** With the current use of PostGIS for managing station locations, the system can be expanded to include more advanced spatial queries for route optimisation or traffic analysis. This would enable TfL to improve its decision-making processes, especially in areas like vehicle routing and infrastructure planning.

## Final Conclusion

In summary, the TfL database project successfully implemented a secure, scalable, and efficient relational database system using PostgreSQL. By addressing data management, compliance, and security, TfL can continue to rely on this database to manage its operational data effectively. Future improvements, such as integrating personal data and enhancing scalability, will ensure that the system evolves alongside TfL’s expanding operations.

# References

Codd, E.F., 1970. A Relational Model of Data for Large Shared Data Banks. Communications of the ACM, [online] Available at: <https://dl.acm.org/doi/10.1145/362384.362685> [Accessed 3 October 2024].

Elmasri, R. and Navathe, S.B., 2016. Fundamentals of Database Systems. 7th ed. Boston: Pearson. Available at: <https://www.auhd.edu.ye/upfiles/elibrary/Azal2020-01-22-12-28-11-76901.pdf> [Accessed 3 October 2024].

European Union, 2016. General Data Protection Regulation (GDPR). [online] Available at: <https://eur-lex.europa.eu/eli/reg/2016/679/oj> [Accessed 3 October 2024].

Kimball, R. and Ross, M., 2013. The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. 3rd ed. Indianapolis: Wiley. Available at: <https://ia801609.us.archive.org/14/items/the-data-warehouse-toolkit-kimball/The%20Data%20Warehouse%20Toolkit%20-%20Kimball.pdf> [Accessed 30 September 2024].

McKinney, W., 2012. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. Sebastopol: O'Reilly Media. Available at: <https://nibmehub.com/opac-service/pdf/read/Python%20for%20Data%20Analysis%20_%20data%20wrangling%20with%20Pandas-%20NumPy-%20and%20IPython.pdf> [Accessed 5 October 2024].

Narizhnykh, D., 2023. MySQL vs PostgreSQL in 2023. [online] Available at: <https://dbconvert.com/blog/mysql-vs-postgresql/> [Accessed 22 September 2024].

PostgreSQL Documentation, n.d. Chapter 24: Data Encryption. [online] Available at: <https://www.postgresql.org/docs/current/encryption-options.html> [Accessed 5 October 2024].

Sheldon, R., 2023. ACID (Atomicity, Consistency, Isolation, and Durability). [online] Available at: <https://www.techtarget.com/searchdatamanagement/definition/ACID> [Accessed 21 September 2024].

Smallcombe, M., 2023. PostgreSQL vs MySQL: The Critical Differences. [online] Available at: <https://www.integrate.io/blog/postgresql-vs-mysql-which-one-is-better-for-your-use-case/> [Accessed 21 September 2024].

Transport for London, 2024. TfL Open Data. [online] Available at: <https://api-portal.tfl.gov.uk/apis> [Accessed 2 October 2024].