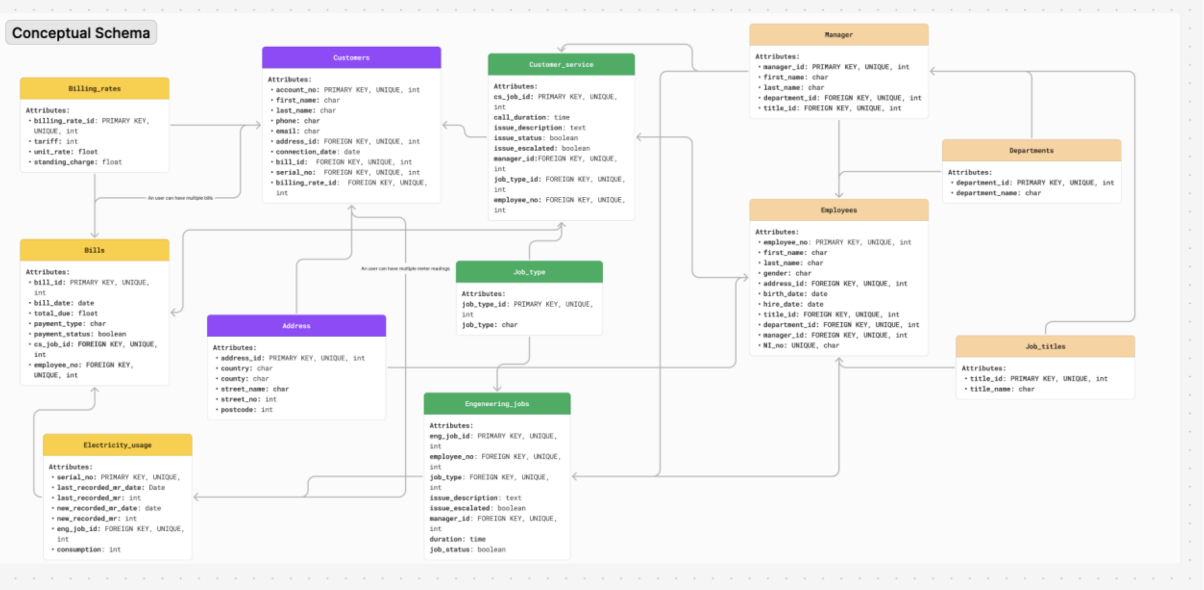
# Individual Project: Executive Summary

## Introduction

The proposal laid out previously in the Database Design: Project Report was for an Electricity Bill Management System. Initially discussing the logical design, there were some discussions about the DBMS, and the Data management pipeline. This Summary will have further discussions of the implementation of the design discussed, with more detail and critical discussion of the DBMS and the use of SQL vs NoSQL. We will explore concepts, and how we can be compliant with GDPR standards.



Conceptual Schema
Billing_rates
Attributes:
• billing_rate_id: PRIMARY KEY, UNIQUE, int
• tariff: int
• unit_rate: float
• standing_charge: float
An user can have multiple bills
Customers
Attributes:
• account_no: PRIMARY KEY, UNIQUE, int
• first_name: char
• last_name: char
• phone: char
• email: char
• address_id: FOREIGN KEY, UNIQUE, int
• connection_date: date
• bill_id: FOREIGN KEY, UNIQUE, int
• serial_no: FOREIGN KEY, UNIQUE, int
• billing_rate_id: FOREIGN KEY, UNIQUE, int
Customer_service
Attributes:
cs_job_id: PRIMARY KEY, UNIQUE, int
call_duration: time
issue_description: text
issue_status: boolean
issue_escalated: boolean
manager_id:FOREIGN KEY, UNIQUE, int
job_type_id: FOREIGN KEY, UNIQUE, int
employee_no: FOREIGN KEY, UNIQUE, int
Manager
Attributes:
• manager_id: PRIMARY KEY, UNIQUE, int
• first_name: char
• last_name: char
• department_id: FOREIGN KEY, UNIQUE, int
• title_id: FOREIGN KEY, UNIQUE, int
Departments
Attributes:
• department_id: PRIMARY KEY, UNIQUE, int
• department_name: char
Bills
Attributes:
• bill_id: PRIMARY KEY, UNIQUE, int
• bill_date: date
• total_due: float
• payment_type: char
• payment_status: boolean
• cs_job_id: FOREIGN KEY, UNIQUE, int
• employee_no: FOREIGN KEY, UNIQUE, int
An user can have multiple meter readings
Job_type
Attributes:
job_type_id: PRIMARY KEY, UNIQUE, int
job_type: char
Employees
Attributes:
• employee_no: PRIMARY KEY, UNIQUE, int
• first_name: char
• last_name: char
• gender: char
• address_id: FOREIGN KEY, UNIQUE, int
• birth_date: date
• hire_date: date
• title_id: FOREIGN KEY, UNIQUE, int
• department_id: FOREIGN KEY, UNIQUE, int
• manager_id: FOREIGN KEY, UNIQUE, int
• NI_no: UNIQUE, char
Address
Attributes:
• address_id: PRIMARY KEY, UNIQUE, int
• country: char
• county: char
• street_name: char
• street_no: int
• postcode: int
Job_titles
Attributes:
• title_id: PRIMARY KEY, UNIQUE, int
• title_name: char
Engeneering_jobs
Attributes:
eng_job_id: PRIMARY KEY, UNIQUE, int
employee_no: FOREIGN KEY, UNIQUE, int
job_type: FOREIGN KEY, UNIQUE, int
issue_description: text
issue_escalated: boolean
manager_id: FOREIGN KEY, UNIQUE, int
duration: time
job_status: boolean
Electricity_usage
Attributes:
• serial_no: PRIMARY KEY, UNIQUE,
• last_recorded_mr_date: Date
• last_recorded_mr: int
• new_recorded_mr_date: date
• new_recorded_mr: int
• eng_job_id: FOREIGN KEY, UNIQUE, int
• consumption: int

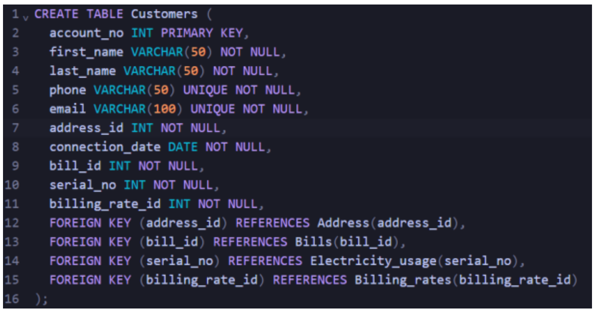
Figure 1: Conceptual Schema from Database Design: Project Report.

## Work Carried out

The purpose and needs of the database were outlined initially in the Database Design: Project Report with the Conceptual schema, also known as an entity-relationship diagram. This section will go through the work carried out from this point onwards with regards to creation of the database tables and establishing relationships for foreign keys.

### Building Tables with SQL

The CREATE TABLE statement was used to create each table outlined in the Conceptual schema of Figure 1.



```
CREATE TABLE Customers (
    account_no INT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    phone VARCHAR(50) UNIQUE NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    address_id INT NOT NULL,
    connection_date DATE NOT NULL,
    bill_id INT NOT NULL,
    serial_no INT NOT NULL,
    billing_rate_id INT NOT NULL,
    FOREIGN KEY (address_id) REFERENCES Address(address_id),
    FOREIGN KEY (bill_id) REFERENCES Bills(bill_id),
    FOREIGN KEY (serial_no) REFERENCES Electricity_usage(serial_no),
    FOREIGN KEY (billing_rate_id) REFERENCES Billing_rates(billing_rate_id)
);
```

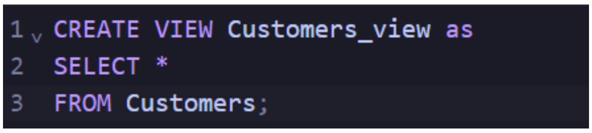
Figure 2: Creation of Customers table using SQL.

Figure 2 shows the use of the CREATE TABLE statement to create the Customers table, it outlines the attribute names, the data types and various constraints depending on the data each attribute represents.

Relationships between tables were defined through the use of foreign keys. It can be seen from the creation of the Customers table; the foreign key address_id references the Address table primary key and bill_id references the Bills table primary key. The linking of data points across tables using foreign keys helps to with referential integrity and preventing inconsistencies when updating and modifying data (Kolosky, 2024).

Once all the tables and relationships were created, the Database is ready population of data using the INSERT statement.

Views of the data held in tables can be created using the CREATE VIEW statement, and example is shown in Figure 3.



```
CREATE VIEW Customers_view as
SELECT *
FROM Customers;
```

Figure 3: Creating a view of the Customers table called Customers_view.

These views can be accessible to people who only need to read the data and can be used to restrict viewing access of users and help with security.

## Review of concepts

The Database model is how a database is implemented, there are many types of models including Hierarchical, Relational, Document and Network models (Lucidchart, 2017).

This implementation of the database uses the Relational model. The relational model is great when dealing with interrelated data securely and consistently. Rules and constraints implemented by this model can prevent data anomalies from modification of the database. The Relational model is also good for data security, this is based on the definition of user roles, permission and encryption (linkedin, N.D.).

Some weaknesses of the Relational model are that they aren’t so good at representing complex data types, having some information split so the attributes have atomic values can create the need for complex queries. It is also only useful when dealing with structured data. One other downside is that SQL is not a computationally complete language, meaning it supports a limited number of operations that can cause difficulty for domain specific constraints (UWE Bristol, N.D.).

Here is a review of some of the concepts when developing a database model:

### Normalisation

Normalisation involves organising data into a relational database in a way that minimises data redundancy and data anomalies, this ensures consistency and accuracy of the data (Chen, 2023).

### Data Modelling and entity relationship model

A conceptual representation of the data and its relationships normally represented using an Entity-Relationship (ER) Diagram. This represents the structure of the database and can be used as a blueprint for the implementation.

### From Data Models to DB design

- Map entities and relationships to tables in the schema.
- Each entity becomes a table.
- Each relationship becomes a foreign key.
- Define indexes and constraints on the tables. Constraints ensure consistency and validity of data and indexes improve performance of queries.

### Views

Predefined views can be created for various end-users of the database in the business. These views and queries can permit specific users to access data in a customised way, it can also be a good security mechanism by giving partial access to certain users dependent on their needs (UoEO, 2024).

### Transaction Management

A transaction can transform the database from one consistent state to another. A transaction can result in Success or Failure depending on whether it violates the consistency of the data. In addition, top transaction management, concurrency control needs to be established to prevent simultaneous transactions interfering with each other and causing inconsistencies.

## Analysis of DBMS

DBMS environment Involves considerations of Hardware, Software and Data used. For this implementation we have established a SQL database would be more suitable due to the structured nature of the data and the relational model outlined. We will however analyse the benefits of SQL vs NoSQL for this use case (Anderson and Nicholson, 2022).

### SQL

- Vertical scalability, meaning that to increase server capacity, physical hardware needs to be increased.
- The database schema is good for tabular and relational data. This is good for structured data applications however means that the data needs more rigorous organising before implementing the Database.
- SQL has been implemented for a long while, the first commercially available implementation being in 1979 (Oracle, 2024). There is a lot of support and knowledge in the ecosystem.

### NoSQL

- Horizontal scalability, meaning additional servers/nodes can be added to existing hardware setup to increase capacity.
- Not relational and can hold multiple formats of data in addition to structured tabular data. It is good for use cases that use multiple data types and formats.

- NoSQL is relatively new, has smaller user communities and therefore there is less support available.

Considering the points outlined above, SQL has a less flexible and more expensive infrastructure when compared to NoSQL's horizontal scalability, this being due to the costs of hardware as well as upgradeability due to increased data loads. NoSQL is good to use for big Data and constantly changing data sets as well as shifting demands (GeeksforGeeks, 2018).

While NoSQL can be used for structured data, the use case of an Electricity Billing Management System fits better with a SQL Database, this is due to the data being more relational, structured and predictable as well as the importance of data validity and integrity (Babitz, 2023).

## Meeting Current Standards and GDPR

It is important to comply with standards and regulations when handling personal data of customers and employees, these standards are laid out under the General Data Protection Regulation (GDPR) when operating in Europe, UK GDPR and the Data Protection Act 2018 when operating in the United Kingdom. As this business will be primarily operating in the UK and with personal data in the UK the data will need to be compliant with the latter two regulations.

The supervisory body for the UK GDPR and enforcing its standards is the Information Commissioner's Office (ICO) (Greenberg, 2023). Some of the main principles outlined are:

- Data minimisation - This means only collecting and holding personal data that is necessary for the business operation.
- Integrity and Confidentiality (security) – This states that data should be protected against unauthorised and unlawful processing, as well as accidental loss, destruction or damage.
- Accountability – This involves documenting how data is handled, and he steps taken to ensure it is only accessed by the relevant people (Burgess, 2020).

As this Database will hold customers personal information, such as their addresses, the data will need to be processed and stored in a compliant way. To help keep the data stored secure, we can implement, pseudonymisation of the personal data stored, have clearly outlined processes for handling and processing of the data and minimising the processing of the data (ICO, 2023).

Data processing is minimised by different end users having varying levels of user access for the information and parts of the database. These levels of access will be determined by the relevant individuals of the company. Database views can be used to control access to sensitive data restricting users to specific parts of the data, as well as simplifying the data access by providing a single interface through data abstraction (Dremio, N.D.).

## Conclusion

To summarise, the use of the relational model for an Electricity Bill Management system is suitable due to the nature structured and interrelated nature of the data. While the model database can define user access and help ensure data validity and integrity. Processes need to be implemented by the company to ensure compliance with UK GDPR standards on the correct handling of the customer data stored in the database.

## References

Kolosky, C. (2024). How to Build a Relational Database: A Complete Guide. [online] Knack: No-Code Application Development Platform. Available from: https://www.knack.com/blog/how-to-design-an-effective-relational-database/ [Accessed on 14 October 2024].

Lucidchart (2017). What is a Database Model | Lucidchart. Available from: https://www.lucidchart.com/pages/database-diagram/database-models [Accessed on 14 October 2024].

linkedin (N.D.). What are the benefits and drawbacks of using a relational database? Available from: https://www.linkedin.com/advice/3/what-benefits-drawbacks-using-relational-database-skills-databases [Accessed on 14 October 2024].

UWE Bristol. (N.D.). What is a Datanbase. Available from: https://www.cems.uwe.ac.uk/~pchatter/resources/html/rdb_strengths_weaknesses.html [Accessed on 14 October 2024].

Chen, W (2023). Database Design and Implementation. Online Research Commons @ ATU. Available from: https://orc.library.atu.edu/atu_oer/2/?utm_source=orc.library.atu.edu%2Fatu_oer%2F2&utm_medium=PDF&utm_campaign=PDFCoverPages [Accessed on 13 October 2024].

UoEO (2024). Database Models and Systems. Available from: https://www.my-course.co.uk/mod/scorm/view.php?id=1025048 [Accessed on 13 October 2024].

Oracle (2024). History of SQL. Available from: https://docs.oracle.com/cd/B12037_01/server.101/b10759/intro001.htm [Accessed on 13 October 2024].

Anderson, B. and Nicholson, B. (2022). SQL vs. NoSQL Databases: What's the Difference? | IBM. Available at: https://www.ibm.com/think/topics/sql-vs-nosql [Accessed on 13 October 2024].

GeeksforGeeks (2018). *Difference between SQL and NoSQL - GeeksforGeeks*. GeeksforGeeks. Available from: https://www.geeksforgeeks.org/difference-between-sql-and-nosql/ [Accessed on 13 October 2024].

Babitz, K. (2023). *SQL vs NoSQL Databases: Key Differences and Practical Insights*. Datacamp.com. Available from: https://www.datacamp.com/blog/sql-vs-nosql-databases [Accessed on 13 October 2024].

Greenberg, E. (2023). 10 Differences Between UK GDPR and EU GDPR. [online] Papaya Global. Available from: https://www.papayaglobal.com/blog/10-differences-between-uk-gdpr-and-eu-gdpr/ [Accessed on 13 October 2024].

Burgess, M. (2020). What is GDPR? The summary guide to GDPR compliance in the UK. Wired. Available from: https://www.wired.com/story/what-is-gdpr-uk-eu-legislation-compliance-summary-fines-2018/ [Accessed on 13 October 2024].

ICO (2023). Data protection by design and default. ico.org.uk. Available from: https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/accountability-and-governance/guide-to-accountability-and-governance/accountability-and-governance/data-protection-by-design-and-default/ [Accessed on 13 October 2024].

Dremio (N.D.). Database View. Available from: https://www.dremio.com/wiki/database-view/ [Accessed on 13 October 2024].