Electricity Bill Management System

This proposal details the plan to create and implement a database for an Electricity Bill Management System. The primary objective is to develop a database that can effectively handle customer information, meter readings, billing, and payments. The design will cater to client requirements for data storage, user access, and efficient data manipulation and retrieval. We plan to use a relational model for this database design, given the structured nature of the customer data and the interrelationships between entities. Therefore, we need to select a suitable RDBMS. Potential options include Oracle, MySQL, and PostgreSQL.

There are 28 million electricity meter points in Great Britain (Ofgem, 2018). With one of the largest energy providers having 6.8 million customers as of April 2024 (Octopus Energy, N.D.). This gives us an idea of the size requirements of the database as to being small/medium.

Various business departments, including Finance, Sales, Accounting, Business Management, and Customer Service, will require different levels of user access to different parts of the database. The access levels and permissions for each department or user will be specified by the client based on the needs of the business.

An example of this would be an employee in the customer service role, where they would need reading privileges of all customer related entities and attributes, however would only have write permissions to change specific attributes such as in the Address table, or the “email” attribute in the Customers table. However, due to the flexible nature of the job roles the access permissions can be modified after creation according to the needs of the organisation.

Due to the size of the database, the cost of running Oracle as the DBMS cannot be justified. On the lower-cost solution end, PostgreSQL and MySQL are both suitable and have their own strengths and weaknesses. For the purpose of the Billing System, PostgreSQL is more suitable.

PostgreSQL is more robust and contains more features that can be useful for larger-scale applications. Additionally, PostgreSQL full-text search and geospatial data features could be beneficial in the long term (Amazon Web Services, N.D). When considering the scalability of the company and the scalability of the database, PostgreSQL seems more appropriate.

MySQL is also suitable for this use case. MySQL is preferred for read-only commands, which will be the primary purpose of the database when generating bills and being used by different departments (Smallcombe, 2023).

For this project, the following conceptual schema is proposed. The database can be split thematically in customers, billing, jobs, and company. Customers represent individual or corporate customers receiving energy services, billing involves the tables needed to manage the energy consumption and the payments, jobs is referring to the managing of customer service related issues, such as handling complaints, taking payments and engineering jobs, whereas the company entities will be used for the management of employees.

The attributes of the entities have been chosen based on the bare minimum needed to run a payment management system. Further end-user research is needed to assess whether the entities chosen are appropriate for the organisation, or if the attributes need to be modified. In the case in which the attributes need modifying, the tables, columns and data types can be changed using the ALTER TABLE function.

Most of the relationships between the entities are One-to-Many or Many-to-Many. A One-to-Many relationship means that a record in one table can be related to multiple records in another table, and a Many-To-Many relationship means that multiple records in a table are related to multiple records in another table (BATI, 2023).

An example of a One-to-Many relationship can be that a customer can have many bills, but a bill belongs to only one customer, whereas a many-to-many relationship can be customer-payment, where a customer can have many payments, and a payment can be associated with many customers (in the case of joint accounts or shared bills).

The data types selected for this database are char, int, boolean, text, time, decimal, and date and are represented in Figure 1. 'Char' data type has been chosen as opposed to the text data type in the cases of attributes such as first_name, last_name and gender because of the character size limit. Integer was chosen as opposed to bigint because it is unlikely that the energy consumed will surpass the signed range of 2147483647 (W3Schools, 2023). Decimal has been deemed as the most appropriate for the currency, as opposed to money data type, because the latter presents rounding errors during calculations (Factor, 2018).

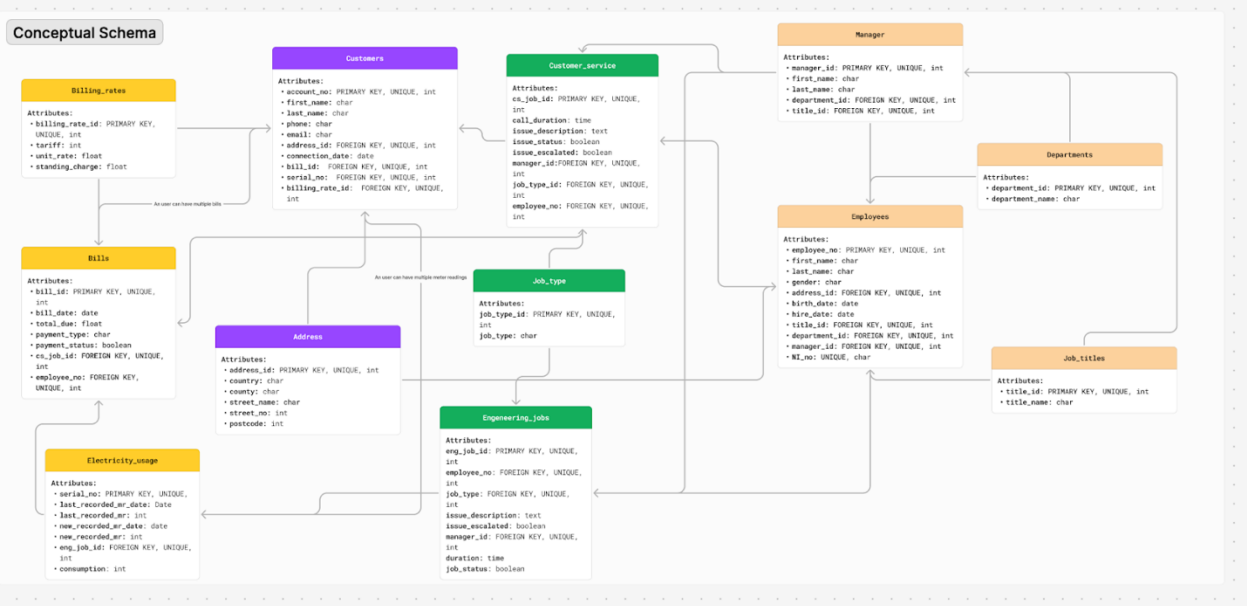

Figure 1: Conceptual Schema

Primary data will be captured via direct, real-time inputs such as electricity consumption from smart meters, manual meter readings, or customer-submitted information (Rind et al., 2023). Secondary data encompasses historical billing records, aggregated consumption patterns, and tariff information, which have already been processed or collected for prior uses but are repurposed for analytics or forecasting within the system.

Database cleaning and normalisation are fundamental processes that ensure the integrity, accuracy, and efficiency of any data-driven system, including electric billing systems. These processes help eliminate data redundancy, correct inconsistencies, and optimise data management.

The data management process starts by importing raw data into staging tables, where it undergoes preliminary examination (Chu, 2019). Next, validation rules are applied to ensure the data's integrity. The data cleaning process begins with detecting outliers and identifying data points that deviate significantly from the norm, such as unusually high meter readings, which may indicate errors. Next, data quality rules are enforced, such as transforming the data to ensure consistency across the dataset (Chu, 2019). Finally, deduplication is carried out to eliminate redundant records, such as consolidating multiple entries for the same meter reading.

Once the data is clean, it is merged into the main database tables and normalisation is applied to structure the data efficiently. Normalisation involves organising the data into different tables and defining relationships between the data (Kumar & Azad, 2017). Each entity is stored only once, with relevant data linked through primary keys and foreign keys. In an electric billing system, this involves creating separate tables for customers, meters, meter readings, bills, payments, and tariffs. Normalisation proceeds through normal forms: First (1NF) ensures atomic values, Second (2NF) ensures non-key attributes depend on the primary key, and Third (3NF) eliminates transitive dependencies (Domingues, 2024). This relational structure facilitates easy updates and improves query performance, making the system more efficient and accurate.

References:

Ofgem. (2018) Record number of customers with small and medium sized suppliers. Available from: https://www.ofgem.gov.uk/press-release/record-number-customers-small-and-medium-sized-suppliers#:~:text=There%20are%20around%2028%20million [Accessed 8 September 2024].

Octopus Energy. (N.D.) Octopus Energy leads retail market growth, confirmed as UK's largest electricity supplier by Ofgem. Available from: https://octopus.energy/press/largest-electricity-supplier-market-share/#:~:text=London%2C%20April%2029th%202024%20%2D%20Octopus [Accessed 8 September 2024].

Amazon Web Services, Inc. (N.D.). MySQL vs. PostgreSQL - Comparing Relational Database Management Systems (RDBMS) - AWS. Available from: https://aws.amazon.com/compare/the-difference-between-mysql-vs-postgresql/ [Accessed 7 September 2024].

Smallcombe, M. (2023) PostgreSQL vs MySQL: The Critical Differences. Available from: https://www.integrate.io/blog/postgresql-vs-mysql-which-one-is-better-for-your-use-case/ [Accessed 8 September 2024].

BATI, O.K. (2023) Understanding Database Relationships: Connecting Data for Better Insights. Available from: https://medium.com/@oguzkaganbati/understanding-database-relationships-connecting-data-for-better-insights-d0f5f0a0e4c4 [Accessed 6 September 2024].

Factor, P. (2018). Avoid use of the MONEY and SMALLMONEY datatypes (BP022) | Redgate.Redgate. Available from: https://www.red-gate.com/hub/product-learning/sql-prompt/avoid-use-money-smallmoney-datatypes#:~:text=Use%20of%20the%20MONEY%20and%20SMALLMONEY%20datatypes%20can%20lead%20to [Accessed 6 Sep. 2024].

W3Schools. (2023) SQL Data Types for MySQL, SQL Server, and MS Access. W3schools. Available from: https://www.w3schools.com/sql/sql_datatypes.asp [Accessed 6 September 2024].

Chu, X. (2019) Data Cleaning. Available from: https://bpb-us-w2.wpmucdn.com/sites.gatech.edu/dist/b/1653/files/2020/10/data-cleaning-book-chapter.pdf [Accessed 5 September 2024].

Domingues, I. (2024) 'Design and Implementation of an Introductory Course on Clinical Databases', *2024 IEEE Global Engineering Education Conference (EDUCON)*, Kos Island, Greece, pp. 1-5. DOI: 10.1109/EDUCON60312.2024.10578794.

Kumar, K. & Azad, S.K. (2017) *Database normalization design pattern.* In: 2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON). Mathura, India, 2017, pp. 318-322. DOI: 10.1109/UPCON.2017.8251067.

Rind, Y., Raza, M., Zubair, M. & Mehmood, M.Q. (2023) 'Smart Energy Meters for Smart Grids, an Internet of Things Perspective', Energies, 16, p. 1974. DOI: 10.3390/en16041974.