Collaborative Project: Database Design and Architecture

Daniel Lopez, Mattia Russo, Sai Mummaneni

Our client EPL Fan Fever Ltd. is a retail business specialising in official Premier League merchandise, operating e-commerce and physical stores. Both channels generate continuous data on transactions, products, customers, and suppliers, as well as supporting inventory management, operations, and customer insights. As software consultants, the proposal presented in this essay seeks to offer our client a flexible and robust solution to provide a single source of operational and strategic decision-making.

Our client requires a reliable and scalable infrastructure capable of handling high-volume transactions and secure data management. For the e-commerce part, which is described in Figure 1, we proposed a solution that follows a traditional model. Online clients' requests pass through Cloudflare DNS before reaching an AWS EC2 server selected for its cost efficiency and future scalability. Running Ubuntu with Apache, the EC2 server hosts a Flask-based Python application connected with WSGI. Security is reinforced with Cloudflare WAF, UFW, and Apache Mod Security for web traffic filtering (Kleppmann, 2017). The application uses a local PostgreSQL for all the database-related needs. Proposed for its flexibility, reliability, and scalability, this database efficiently handles structured data via relational tables, indexes, and JSON fields (Reis and Housley, 2022).

On the other hand, the physical infrastructure consists of an ERP with a local database, connected for transactional CRUD operations and data transfers with the cloud database mentioned on the same EC2 instance. PostgreSQL also supports role-based access control and SSL/TLS encryption, allowing secure, differentiated multi-user access while ensuring GDPR compliance. Its open-source nature reduces licensing costs, while

automated backups, data replication, and robust recovery mechanisms ensure business continuity and data availability. With advanced SQL queries supporting analysis of sales, product performance, and customer behaviour, and Python integration allowing seamless data extraction, reporting, and dashboard generation, the database aligns smoothly with the overall infrastructure.

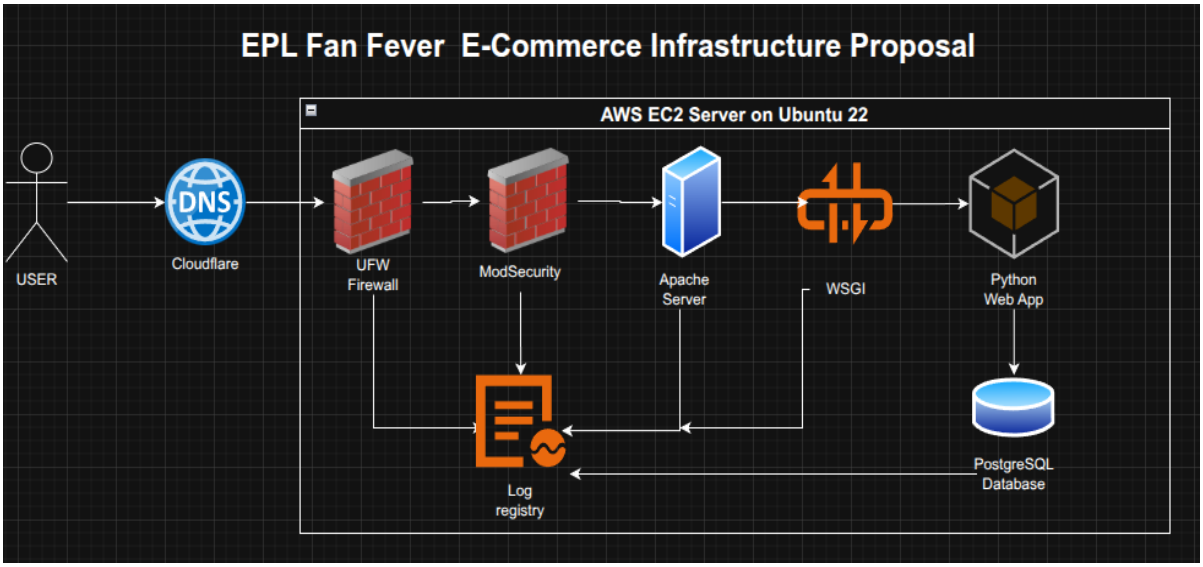

Figure 1- E-commerce Cloud Infrastructure schema

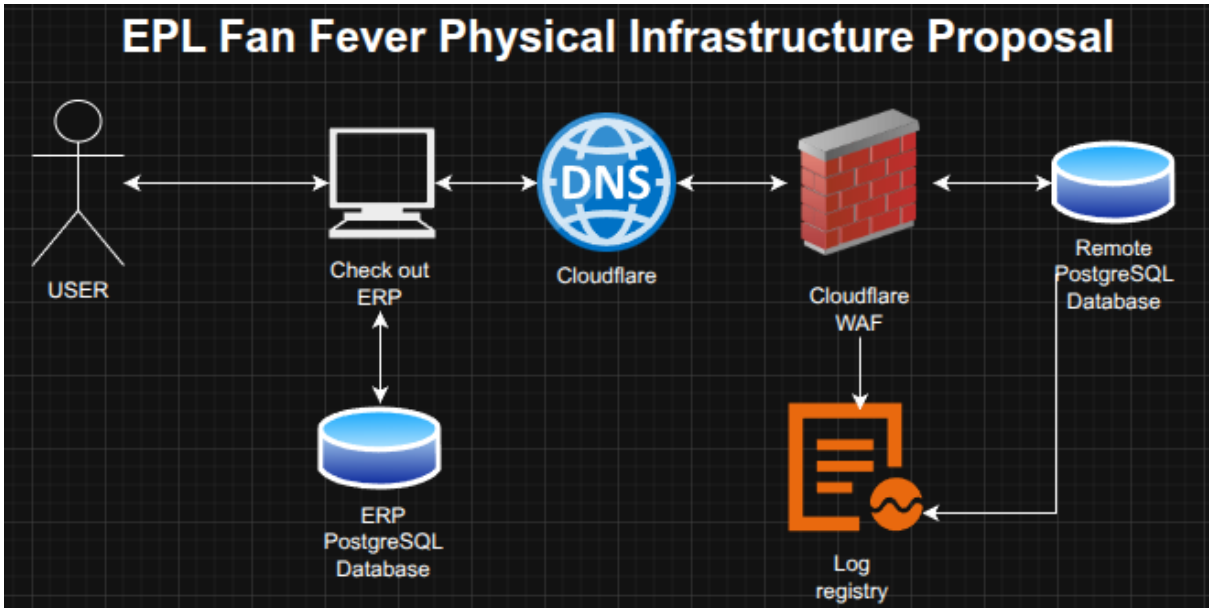

Figure 2- Physical Infrastructure schema

Our client relies on two primary data sources. The first one arises from user interactions with the e-commerce platform or physical stores, generating transactional, product, or customer data, including purchases and inventory checks. The second one comes from internal data changes, for instance, updates to the product catalogue, employee records, or supplier information.

Regarding data ingestion, we suggest a Python API running on FastAPI that will capture data from in-store POS and website transactions. This proposed API must validate file format, size, filename pattern, completeness, and data integrity. Successful files are logged for each structured table in the database (Segner, 2023).

During the staging phase, the system logs essential metadata such as store ID, timestamp, file type, row count, and error flags to maintain traceability across multiple retail locations. The pipeline implements a clear separation of concerns where validation and conversion occur during staging, errors are routed to dedicated exception tables, normalisation progresses during the transformation stage, and clean records are finally loaded into PostgreSQL through ACID-compliant transactions (Huxley, 2025).

ETL transformations are orchestrated with Apache Airflow to maintain proper sequencing, dependency management, and scheduling. AWS Lambda is considered as a potential alternative for faster-than-expected scaling scenarios. Cleaned data is loaded into a PostgreSQL database hosted on AWS EC2 using UPSERT operations. As illustrated in Figure 3, the mentioned ETL process follows a structured sequence to maintain data quality and integrity.

For raw data ingestion and staging, the files are mapped and loaded into staging tables with all columns stored as strings, preserving original snapshots for auditing purposes. During field validation and conversion, mandatory fields undergo validation, with conversions applied to timestamps, quantities, and prices. A Python library called Polars is used to parse data and correct format errors. To prevent duplication, each record is assigned a unique hash identifier, enabling efficient deduplication when duplicates arise (Patel and Shah, 2025). Speaking about referential integrity checks, foreign keys such as “Product_ID”, “Team_ID”, and “Supplier_ID” are validated against master tables. Near-misses, including SKU variations, are resolved while enforcing foreign-key constraints (Jebali, 2021). For anomaly and outlier detection, negative quantities, future timestamps, and inflated totals are flagged using statistical thresholds and z-scores to maintain data reliability. To apply normalisation, text fields, phone numbers, emails, and date/time formats are standardised to reduce variability prior to integration (Sankpal and Metre, 2020).

A critical operational component involves generating structured error reports that are delivered to store managers, enabling them to address data quality issues at the source. Recurring data quality problems trigger improvements to POS procedures and inform targeted staff training initiatives, creating a continuous feedback loop that enhances data integrity across the retail network.

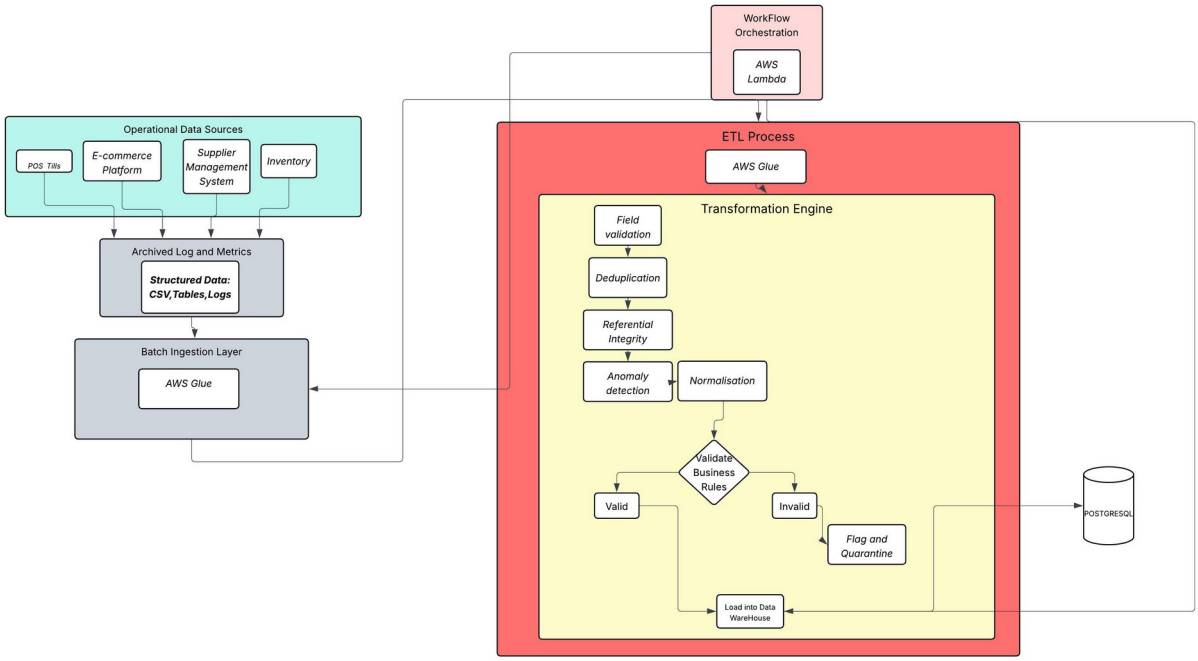

Figure 3- ETL Pipeline Schema

Regarding the database schema, a star schema will be used, with the fact table capturing transactional data and linking to multiple dimension tables for descriptive attributes. We suggest this schema design, as it ensures data integrity, reduces redundancy, and enables efficient analytics.

Figure 4 illustrates the database schema, which is structured around the central table, “OrderItems”. This table contains detailed customer purchase orders organised by product level and links the “Orders” and “Products” tables. We suggest making “OrderItems” the core of the schema, as the company’s primary transactional process is the recording of sales, and this table effectively manages the relationship between orders and products. Supporting tables include business-related tables, “Product,” “Customer,” and “Order”, and operational tables, “Team,” “Inventory,” “Employee,” “Supplier,” and “Store.”

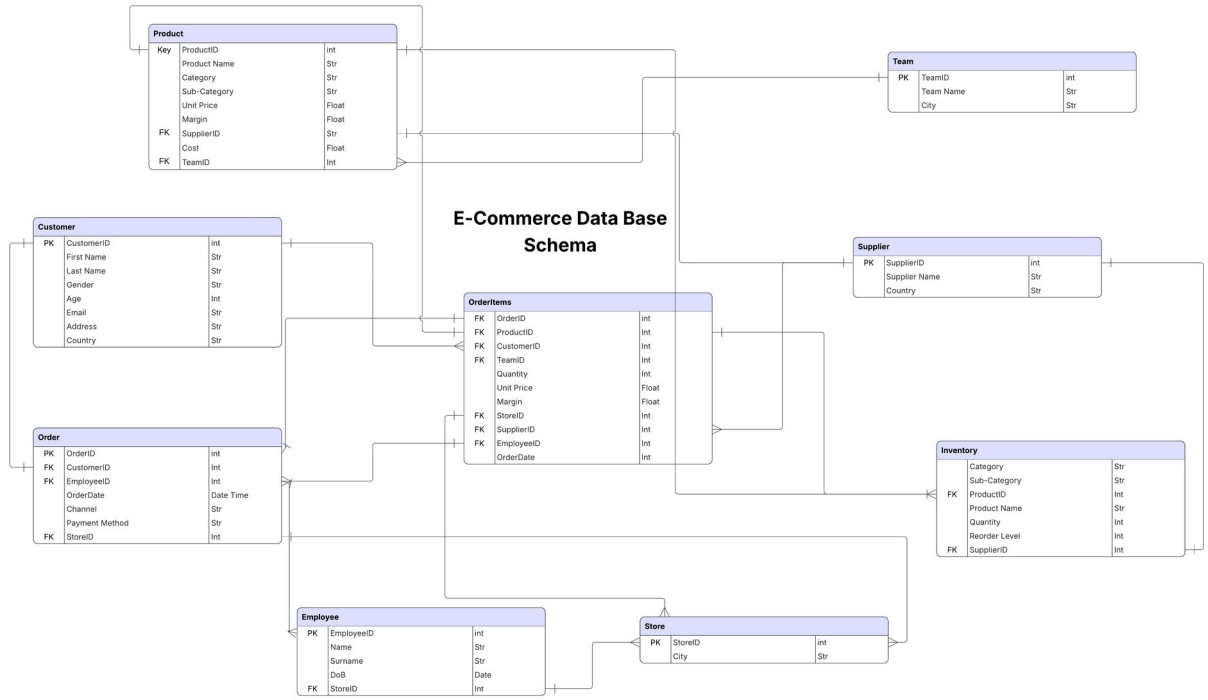

Figure 4- Database Schema

Conclusion

The proposed solution addresses EPL Fan Fever Ltd.'s core requirements through deliberate architectural choices. The ETL pipeline's multi-stage validation approach (hashing for deduplication, z-score anomaly detection, referential integrity checks) prioritises data quality over processing speed, appropriate for a retail context where accuracy outweighs real-time requirements. However, this design assumes data volumes remain within single-server PostgreSQL capacity; future horizontal scaling would require architectural revision.

Critical next steps include implementing automated schema migration tools, establishing data quality KPIs (target: <0.1% validation failures), and conducting penetration testing on the API layer. The system provides a solid foundation for EPL Fan Fever Ltd.'s current operational scale with clear pathways for incremental enhancement.

References

Huxley, K. (2025) *Data Cleaning*. London: SAGE Publications Ltd. Available at: https://doi.org/10.4135/9781526421036842861.

Jebali, A. (2021) *Access Control Policies Verification Over Distributed Queries*. Ph.D. Thesis. Faculté des sciences de Tunis. Available at: https://hal.science/tel-03535655.

Kleppmann, M. (2017) *Designing Data-Intensive Applications*. Sebastopol: O'Reilly Media, Inc.

Patel, P. and Shah, D. (2025) *Cryptographic Wallet Security and Key Management in Blockchain Financial Systems: A Systematic Literature Review*. Preprint/Working Paper. Waterloo, Ontario, Canada: Wilfrid Laurier University. Available at: Contact authors for preprint access.

Reis, J. and Housley, M. (2022) 'Designing Good Data Architecture', in *Fundamentals of Data Engineering*. First Edition. Sebastopol, CA: O'Reilly Media, Inc., p. Chapter 3. Available at: https://www.oreilly.com/library/view/fundamentals-of-data/9781098108298/.

Sankpal, K.A. and Metre, K.V. (2020) 'A Review on Data Normalization Techniques', *IJERT*, 9(6), pp. 1438–1441.

Segner, M. (2023) *Data Validation Testing: Techniques, Examples, & Tools*. Monte Carlo Data. Available at: https://www.montecarlodata.com/blog-data-validation-testing/ (Accessed: 28 November 2025).