Data model

A data model is the visual representation of the data elements and their relationships. It serves as a bridge between business requirements and technical implementation. It is not static but updated when the data sets and business needs change. Data models design the database at the conceptual, logical and physical levels based on the level of abstraction (the process of hiding unwanted details from end user).

In the conceptual data model, the representation of data is at a high level of abstraction (The higher the level, the less details. The lower the level, the more details). Here the entities, their attributes, and relationships are established. The focus is to represent data, in the simplest way. The design of conceptual data model does not require a high level of expertise and knowledge in databases to understand. This makes it the preferred option to use when working with non-technical stakeholders. This data model is designed by business stakeholders and data architects. One such popular model is the entity relationship model (ER model).

The logical data model refines the data elements introduced by a conceptual data model and forms the basis of physical data model. It is the first step in designing and building of the database. It builds upon the business facts and the appropriation of the business rules then forms relationships between the various data objects and entities but the the primary or secondary keys are not defined. The logical data model does not depend on specific database and data storage structures.

The Physical data model has the most details, and it is the last stage before the creation of the database. Here decisions about the implementation of the logical design on a specific database management system (DBMS) for example MySQL, Oracle, PostgreSQL are made. To design the physical data model requires in-depth knowledge of the specific DBMS used. For example, in a relational database, it is important to understand the implementation of the tables, columns, data types, and the relationships between tables and columns in the specific relational database product.

Data modelling

Data modelling is the process of creating a visual representation of data. Data modelling makes it easy for businesses to work with the data in their database and have a deep understanding of their data structure. It enforces the business rules, regulatory compliances, and government policies on the data. Data modelling techniques fall into two categories namely the entity relationship (ER) model and the Unified modelling language (UML).

We designed our Evans cycle logical database by employing the conceptual, logical and physical data models (entity relational modelling technique). The three data models represented the data, how it was stored in the database, the relationship between the data items and its implementation. Our conceptual data model defined what the system contained, organized the scope and defined the business concepts and rules. Our logical data modelling translated the conceptual model into detailed data structures, attributes and constraints without paying attention to specific functions and capabilities of the database management system (DBMS) that would store the data. Here we documented the comprehensive business information requirements in an accurate and consistent format. We defined the data items and the business rules that affect those data items. We also worked interactively with the end users as much as possible and used diagrams to represent as much of the logical data model as possible. Our process of data logical modeling acknowledges that our Evans cycle business data is a vital asset we ought to understand and carefully manage. (Sloan, 2004)

Implementation of Evans cycle Logical database

After completing our logical data modelling, I began with our physical design. Here I first made the decision on how to translate entities into tables, the instances into rows and the attributes into columns. Secondly, i decideded what indexes to define on the tables, which columns of the tables to define as keys, what views to define on the tables, how to de-normalize the tables and how to resolve the different types of relationships. I used a specific database management system MYSQL (Structured query language) to implement my Evans cycle logical database. MYSQL is a standard language for accessing (storing and retrieving) and manipulating databases. It turns raw data stored in a database into actionable insights.

I started first by setting up the MYSQL community server in my system to create the database schema. A database schema are the blue prints of databases. It shows a database design such as what tables are included in the database and any relationship btw its tables. It lets one know what data type each field can hold. A database schema is first created before the table is created.

To create the database schema, I right clicked on "database" on the left hand of MYSQL work bench and click on create schema. This created a new\_schema tab, I named the schema "Evans cycle" then I clicked on apply and finish. After which I clicked on the schema, clicked on tables, right clicked on tables and clicked on create table. A new table tab poped up which I named as customer, then I filled the attributes and the datatypes. I also selected the "customer\_id" to be the PRIMARY KEY to uniquely identify each row in the table.

Next, I added constraints to some of the attributes. These constraints specify rules for the data in the tables and controls what can be inputted into the DBMS. I added a NOT NULL constraint to the attributes first\_name, last \_name and phone number in my "customers" table so that the table will not accept a record where those attributes are set to "NULL". This is because the "customers" table would require a first name, last name and phone number. I also set email to be "UNQUE" to ensure every record must have a different value for this attribute.

After adding constraints, I established relationships between the tables by setting up foreign keys. I used the ALTER Table followed by the name of the participant table to update the table. Next, I used the ADD foreign key followed by the attribute to reference the Primary key.

I created tables for the other entities (orders, staff, stores and order\_items) in my Evans cycle logical database. Now that I have all of my tables created in a fully functioning relational database, I populated the tables with data by using the INSERT INTO statement with the table name followed by the data I want in parentheses. The syntax is "INSERT INTO table VALUES (data I want to insert, separated by commas)". I retrieved the data from the database using the SELECT statement. To retrieve all the columns from the database, I used SELECT with an asterisk while to select only some columns, I used SELECT statement followed by the names of the columns I want separated by a comma.

Below is my “customers table” syntax:

CREATE TABLE `customers` (

`id` int NOT NULL,

`first\_name` varchar(45) NOT NULL,

`last\_name` varchar(45) NOT NULL,

`phone\_number` varchar(45) NOT NULL,

`email` varchar(45) NOT NULL,

`city` varchar(45) DEFAULT NULL,

`street` varchar(45) DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `email\_UNIQUE` (`email`)

);

DESC customers;

INSERT INTO customers VALUES(1,'PRINCE','OKOLIE','07036126160','PINO@YAHOO.CA','JOS','KASHIM IBRAHIM');

SELECT \* FROM customers;

In summary, the primary purpose of data modeling is **not** to design a database; rather, it is to **describe a business**. It provides the conceptual understanding and clarity needed to describe a business, while database design translates that understanding into a concrete database structure. Data models serve as a bridge between business requirements and technical implementation. The primary purpose of designing a database is to produce physical and logical models of designs for the proposed database system. Database design focuses on creating the actual structure of a database system. It involves defining tables, relationships, constraints, and indexes. Database design is dependent on the specific **DBMS (Database Management System)** and the hardware it runs on. [Both are essential for effective data management and successful business operations](https://techmoodly.com/database/data-model-vs-database/" \t "_blank).

General database protection regulation (GDPR)

With the advent of data privacy regulations and the ever-growing threat of data breaches, I identified and included all the necessary components for regulatory compliance requirements into our database infrastructure during my data modelling. One of the regulatory compliance requirements I included was the general database protection regulation (GDPR) enforced by the European Union. GDPR mandates strict controls on the processing and storage of personal data. The law regulates the export of personal data outside the EU and aims to give EU citizens and residents control over their personal data. This law includes articles that regulate data breaches, the right for citizens to access their personal data, the right for citizens to request the erasure of personal data, and data portability. Meeting database regulatory compliance and security requirements was a top priority during our data modelling, this is because I recognize that a business that fails to meet regulatory requirements may find it difficult to remain in business or may go into bankruptcy due to substantial fines and penalties by regulators.

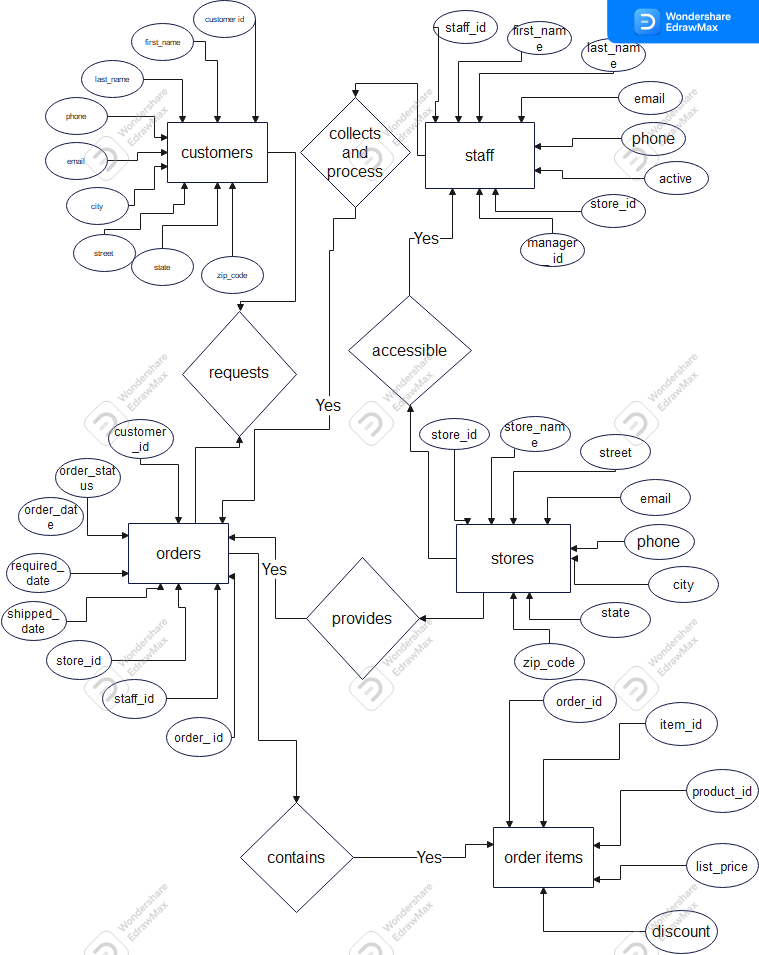
My conceptual model provided a visual representation of the database structure, which would make it easy for stakeholders to understand, make decisions and provide feedback. In my conceptual model, I represented entities by a rectangle, attributes by eclipse and relationships by a rectangle. It was flexible, easy to design and easily modified to accommodate changes in the database design. It created databases of varying sizes and it provided useful documentation for future reference. My conceptual models helped minimize data redundancy by identifying and eliminating duplicate data entries.

Drawing my conceptual model came with some limitations. It had limited attribute representation, which led to loss of important information; it also had limited relationship representation since it did not indicate primary and foreign keys. It was difficult to scale to accommodate larger datasets. My conceptual model did not have representation for data manipulation commands like (insert, update, delete) and it did not provide support for business rules which makes it difficult to ensure data integrity. My conceptual model did not support abstraction, it had data inconsistency and it was difficult to modify once created.

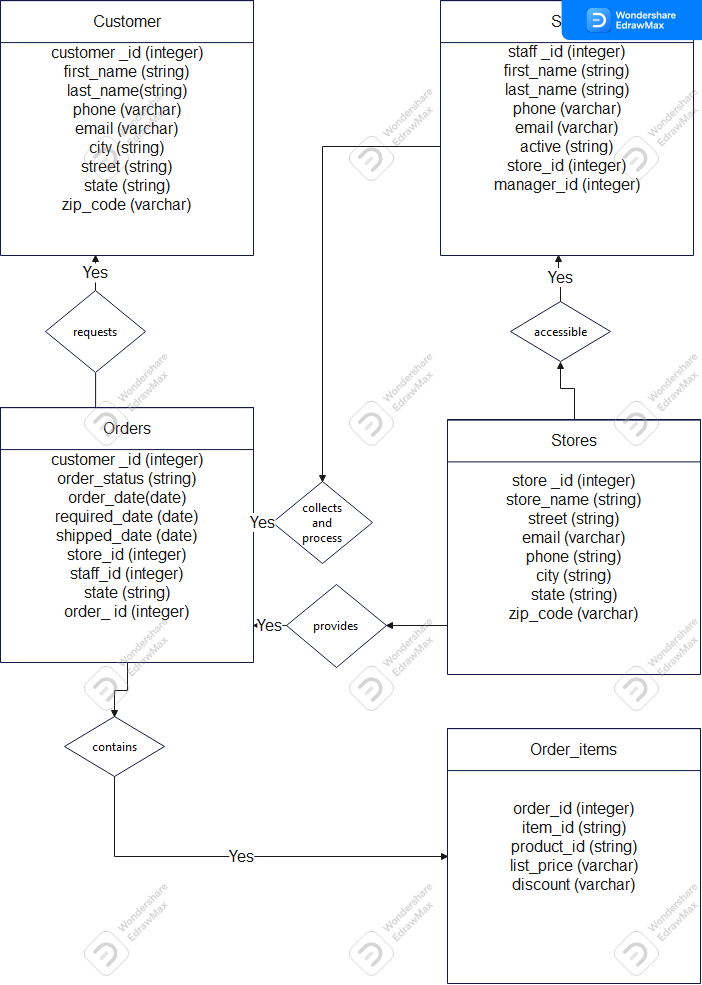
My logical data model was stable and highly conducive to data re-use and physical data sharing leading to reduced storage of redundant data. It helped identified areas for business process improvement, provided a basis for future models, reduced costs and increased efficiency. It helped articulate causal linkages, built consensus, and identified what to evaluate.

My logical model focused on expected outcomes and did not pay attention to alternative or unexpected outcomes (positive, negative, or neutral). It depicted assumed causal connections, not direct cause-and-effect relationships. It did not address the question: “Am I doing the right thing?” It was only as good as our understanding of the situation, the environment, the theory we expressed, and our assumptions.

In conclusion, we built our Evans cycle logical database using the entity relational(ER) modelling technique. We started with the conceptual data model, where the understanding of the problem domain occurs and the identification of entities and relationships takes place. Next, we used the logical data model to translate the conceptual model into a more detailed representation, specifying attributes, keys, and relationships. Finally, the physical data model transforms the logical data model into a concrete database schema, making decisions about data types, storage, and security measures. By following this structured approach to design our Evans cycle logical database, Evans cycle can ensure that their databases meet their data management and governance needs effectively and efficiently. In addition, Evans cycle can harness the power of their data, ensuring it remains organized, accessible, and valuable.



Conceptual Data Model



Logical data model

**References:**

Binus (2021) Expand the enterprise logical data model. Available at: [Expand the enterprise logical data model – School of Information Systems (binus.ac.id)](https://sis.binus.ac.id/2021/03/24/expand-the-enterprise-logical-data-model/) (Accessed: 25 January 2024).

Course Hero (2021) Fundamental of databases- data models. Available at: [Fundamentals of Databases Unit 3 Data Model.ppt - UNIT 3 Data Models NORTHERN CARIBBEAN UNIVERSITY College of Natural & Applied Sciences Allied | Course Hero](https://www.coursehero.com/file/107470739/Fundamentals-of-Databases-Unit-3-Data-Modelppt/)(Accessed: 28 January 2024).

Erwin (2024) Logical data modelling. Available at: [Logical Data Modeling | erwin, Inc.](https://www.erwin.com/solutions/data-modeling/logical.aspx) (Accessed: 27 January 2024).

Hart, A (2023) what is data modelling? Definition and examples. Available at: [What is Data Modeling? Definition & Examples (datamation.com)](https://www.datamation.com/applications/what-is-data-modeling/) (Accessed: 22 January 2024).

Kabir, A (2020) Database design step by step. Available at: [Database Design Step by Step – techforum (wordpress.com)](https://techforumugm.wordpress.com/2020/02/15/database-design-step-by-step/) (Accessed: 20 January 2024).

Luquibase (2018) Database compliance and security. Available at: [Database Compliance & Security: What You Need to Know | Liquibase](https://www.liquibase.com/blog/database-compliance-security) (Accessed: 4 February 2024).

Maria DB (2021) Database design: Logical and physical design. Available at: [Database Design Phase 2: Logical and Physical Design - MariaDB Knowledge Base](https://mariadb.com/kb/en/database-design-phase-2-logical-and-physical-design/) (Accessed: 2 February 2024).

Mullins, C (2023) Data privacy regulations and database compliance. Available at: [Data Privacy Regulations and Database Compliance: An Overview - elnion.com](https://elnion.com/2023/11/17/data-privacy-regulations-and-database-compliance-an-overview/#:~:text=Ensuring%20database%20compliance%20with%20data%20privacy%20regulations%20is,is%20vital%20for%20compliance%20and%20security.%20More%20items) (Accessed: 10 February 2024).

Sherman, R (2015) Foundational data modelling. Available at: [Foundational Data Modeling | Semantic Scholar](https://www.semanticscholar.org/paper/Foundational-Data-Modeling-Sherman/f1a494973e6c8520873e92079d853b09a2fc2c3f) (Accessed: 29 January 2024).

Taylor, D (2023) what is data modelling? Types(Conceptual, logical and physical) Available at: [What is Data Modelling? Types (Conceptual, Logical, Physical) (guru99.com)](https://www.guru99.com/data-modelling-conceptual-logical.html) (Accessed: 24 January 2024).

Tibco (2024) what is logical data model. Available at: [What is a Logical Data Model? | TIBCO](https://www.tibco.com/glossary/what-is-a-logical-data-model) (Accessed: 6 February 2024).

Visual Paradigm (2023) Navigating the Three Levels of Database Design: Conceptual, Logical, and Physical. Available at: [Navigating the Three Levels of Database Design: Conceptual, Logical, and Physical - Visual Paradigm Guides (visual-paradigm.com)](https://guides.visual-paradigm.com/navigating-the-three-levels-of-database-design-conceptual-logical-and-physical/) (Accessed: 28 January 2024).

Wise Edu (2024) Limitations of Logic models. Available at: [6.8: Limitations of Logic Models – Enhancing Program Performance with Logic Models (wisc.edu)](https://logicmodel.extension.wisc.edu/introduction-overview/section-6-how-good-is-my-logic-model/6-9-limitations-of-logic-models/) (Accessed: 28 January 2024).