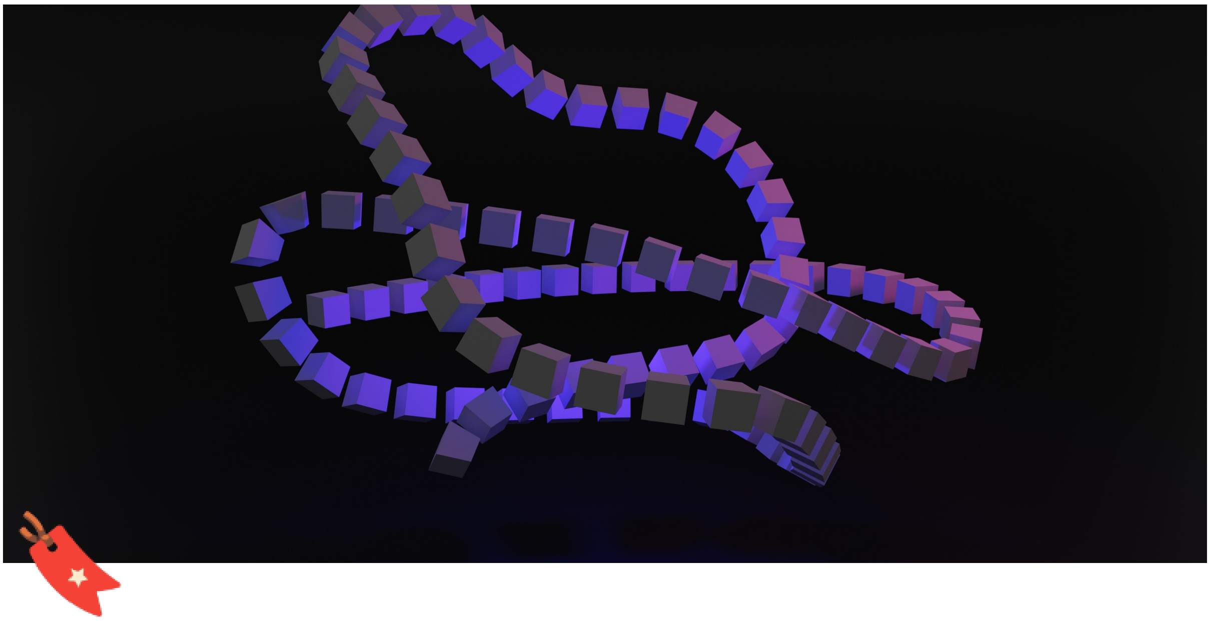


Database Management Project using MySQL on phpAdmin

Phase 1: Designing a database for VictoryRide Transport Services

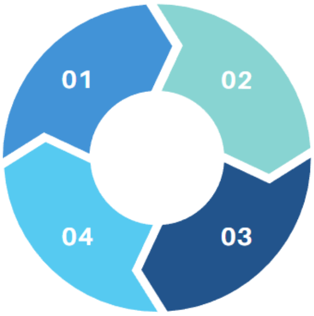
A. Why do we need to design the database before implementation?

Before implementing a new database, database administrators need to create a solid foundation for storing and managing data effectively. Going through the designing steps, such as designing an ERD and performing normalization, is crucial before implementation.

B. Techniques to design a database

Designing an ERD and performing normalization are typically **iterative** processes that complement each other. While there is no strict order in which they must be performed, it is often beneficial to start with an initial ERD design and then proceed with normalization, revisiting and refining the ERD as needed.

In this project I designed the databse for VictoryRide Transport Services step-by-step as follows:



Stage 1: Initial ERD Design

Captures the entities, attributes, and relationships based on my understanding of the requirements.

Stage 2: Normalization Analysis

Analyze ERD and perform normalization techniques (final goal is Third Normal Form) to eliminate data redundancies or anomalies.

Stage 3: Evaluate and Refine ERD

Revisit the ERD to ensure that it accurately reflects the normalized structure. Make any necessary adjustments.

Stage 4: Iterative Process

Repeat steps 2 and 3 as needed. Identify opportunities for optimization or discover new dependencies that require modifications.

C. Outputs for design phase

Entity Relationship Diagram (ERD)

The following ERD helps in understanding the conceptual structure of the database by identifying entities & their attributes:

- Primary key (PK): An attribute that uniquely identify all other attributes of an entity;
- Foreign key (FK): Values match a PK in a referenced (parent) table;
- Composite primary key (PK,FK): Attributes which act as PK in 2 or more tables;
- Super key: Any key that uniquely identifies each row in an entity.

The diagram also illustrates the relationships between

- Strong relationships: When the PK of the related entity contains a PK component of the parent entity (solid lines)
- Weak relationships: When the PK of the related entity DOES NOT contains a PK component of the parent entity (dash lines)

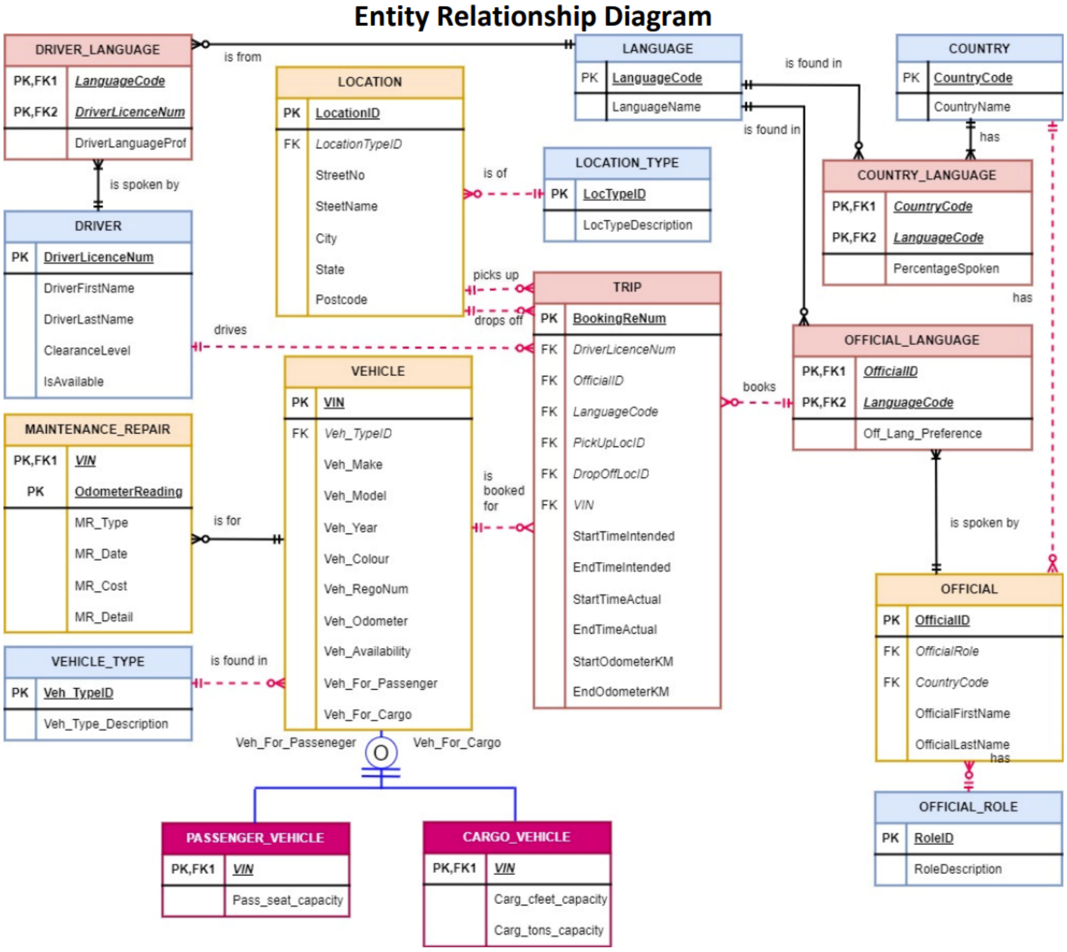
The ERD also defines and represents the connectivity, participation, and constraint types between entities accurately with Crow's Foot notations:

- Connectivity types: 1-to-1, 1-to-Many, Many-to-Many;

- Participation types: Optinal = 0, Mandatory = 1;
- Subtypes:
    - Disjoint (d) → contains unique subset of supertype entity set (e.g. if the vehicle can ONLY register as EITHER cargo OR passenger)
    - Overlapping (o) → contains NON-unique subset of supertype entity set (e.g. if the vehicle can register as BOTH cargo & passenger)
- Subtype constraints:
    - Partial (-) → some records in supertype entity aren't members of any subtype
    - Complete (=) → each record in supertype entity must be a member of at least 1 subtype

Noted that the ERD is designed to ensure ***entity integrity*** and ***referential integrity***:

- Entity integrity: PK in a table must be unique & no part of the key is NULL;
- Referential integrity: FK in a table must be either a valid PK value coming from another table or be NULL.



▼ *Dependency diagram in Third Normalisation Form*

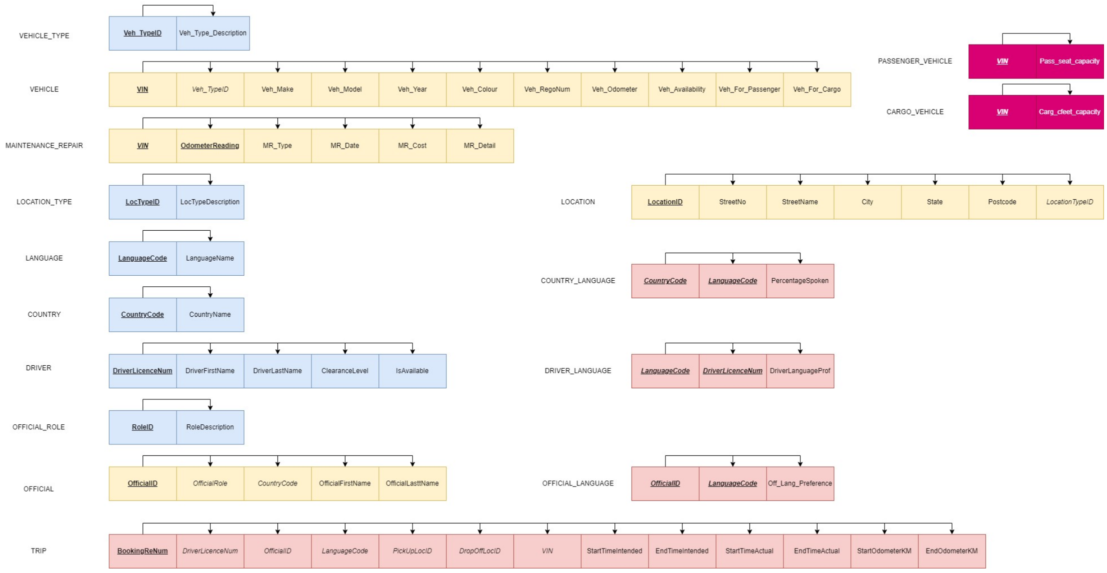
Normalization is the process of organizing the data model to eliminate data redundancy, inconsistency and prevent update/deletion/insertion anomalies.

There are 4 types of functional dependency - total dependency, full dependency, partial dependency, and transitive dependency. In this project, the diagram is designed to show full, partial, and transitive dependencies among entities and their attributes.

There are three major stages of normalisation, conditions for each stage are briefly described as follows:

- 1NF: All dependencies (full, partial & transitive) are indentified and a PK has been identified for each row;
- 2NF: Eliminating all partial dependencies;
- 3NF: Removing all transitive dependencies.

Third Normal Form (3NF) is an important level of normalization that provides further improvements in data structure compared to lower normalization levels. By removing transitive dependencies, we ensure that changes in non-key attributes do not affect other non-key attributes, maintaining data consistency and reducing data redundancy.

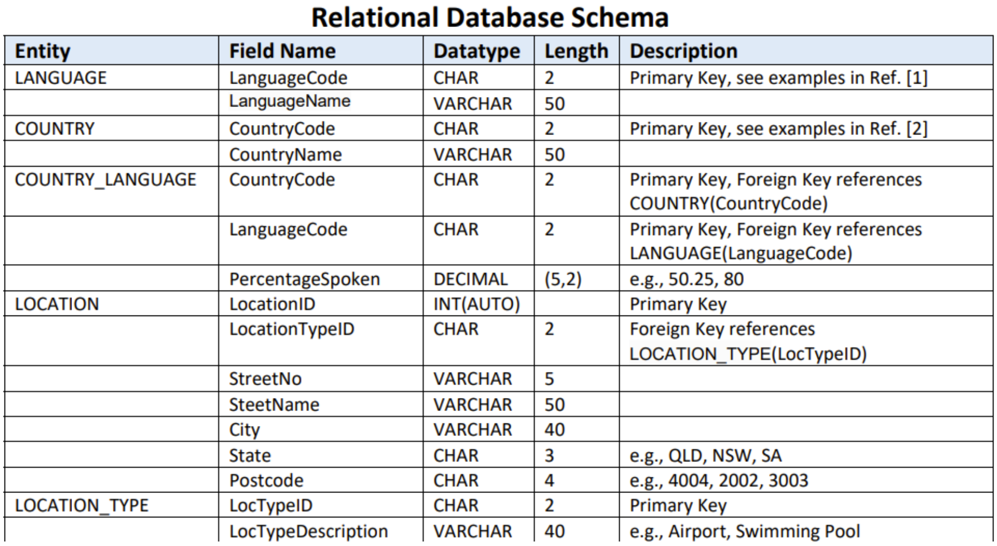


▼ *Relational Database Schema*

The final output of the design phase is the relational database schema. It is a logical blueprint that defines the structure, organization, and relationships of the data within a relational database.

The schema documentation serves as a reference for maintaining, updating, and evolving the database over time. It helps administrators and developers understand the database design, making it easier to make modifications, add new features, or troubleshoot issues.

Presented below is a concise depiction of the relational database schema pertaining to VictoryRide Transport Services:



Relational Database Schema

Entity	Field Name	Datatype	Length	Description
LANGUAGE	LanguageCode	CHAR	2	Primary Key, see examples in Ref. [1]
	LanguageName	VARCHAR	50	
COUNTRY	CountryCode	CHAR	2	Primary Key, see examples in Ref. [2]
	CountryName	VARCHAR	50	
COUNTRY_LANGUAGE	CountryCode	CHAR	2	Primary Key, Foreign Key references COUNTRY(CountryCode)
	LanguageCode	CHAR	2	Primary Key, Foreign Key references LANGUAGE(LanguageCode)
	PercentageSpoken	DECIMAL	(5,2)	e.g., 50.25, 80
LOCATION	LocationID	INT(AUTO)		Primary Key
	LocationTypeID	CHAR	2	Foreign Key references LOCATION_TYPE(LocTypeID)
	StreetNo	VARCHAR	5	
	SteetName	VARCHAR	50	
	City	VARCHAR	40	
	State	CHAR	3	e.g., QLD, NSW, SA
	Postcode	CHAR	4	e.g., 4004, 2002, 3003
LOCATION_TYPE	LocTypeID	CHAR	2	Primary Key
	LocTypeDescription	VARCHAR	40	e.g., Airport, Swimming Pool