Mobile Legends Database

Database Design Document (DDD)

Version *1.0.1*

Prepared by: ***Tristan Jay M. Sevilla***

Revision History

Date	Version	Description	Author

Table of Contents

1 Introduction

This section introduces the Mobile Legends Database

1.1 Document Objectives

This document aims to provide a comprehensive design of the Mobile Legends database. It details the structure of the database, including tables, fields, data types, and relationships between entities. The design supports the game's features such as user management, hero and skin customization, and in-game transactions. This document serves as a blueprint for developers and database administrators to understand and implement the database system.

1.2 Intended Audiences

The intended audiences for this document are:

- **Database developers and administrators** responsible for implementing and maintaining the database.
- **Game developers** who need to integrate the database with the game's frontend and logic.
- **Project managers and stakeholders** who require an understanding of the data organization and flow within the game.
- **Users or Players** who wants to know the whole process of the game and its database.

1.3 References

Notion – The all-in-one workspace for your notes, tasks, wikis, and databases. (n.d.). Notion. https://believed-bongo-319.notion.site/CTINFMGL-Project-Specifications-19296450aad180aea9ebf27987415f4c

2 Detailed Database Design

This section describes the actual design of the Mobile Legends database at varying levels of abstraction: the conceptual, logical, and physical levels. Each subsection provides a detailed view of the database structure, from high-level entity relationships to specific implementation details in MySQL.

2.1.1 Data dictionary

2.1.1.1 Data dictionary for Element: *Admin*

Name	Data Type	Constrain	Description
Username	VARCHAR(20)	NOT NULL	The unique username for the admin account.
Password	VARCHAR(40)	DEFAULT NULL	The password for the admin account.
AccountCreated	TIMESTAMP	NOT NULL, DEFAULT current_timestamp(), ON UPDATE current_timestamp()	The timestamp when the account was created or last updated.

2.1.1.2 Data dictionary for Element: *Player*

Name	Data Type	Constrain	Description
PlayerID	INT	AUTO_INCREMENT, PRIMARY KEY	A unique identifier for each player, auto-incremented.
IngameName	VARCHAR(16)	NOT NULL	The player's in-game username.
GameAccount	VARCHAR(100)	NOT NULL, UNIQUE	The player's unique game account/email identifier.
PlayerPassword	VARCHAR(30)	NOT NULL	The player's password for the game account.

2.1.1.3 Data dictionary for Element: *Hero*

Name	Data Type	Constrain	Description
HeroID	INT	NOT NULL, PRIMARY KEY	A unique identifier for each hero.
HeroName	VARCHAR(30)	NOT NULL	The name of the hero.
HeroRole	VARCHAR(30)	NOT NULL	The role or category of the hero (e.g., tank, support)

2.1.1.4 Data dictionary for Element: *Skin*

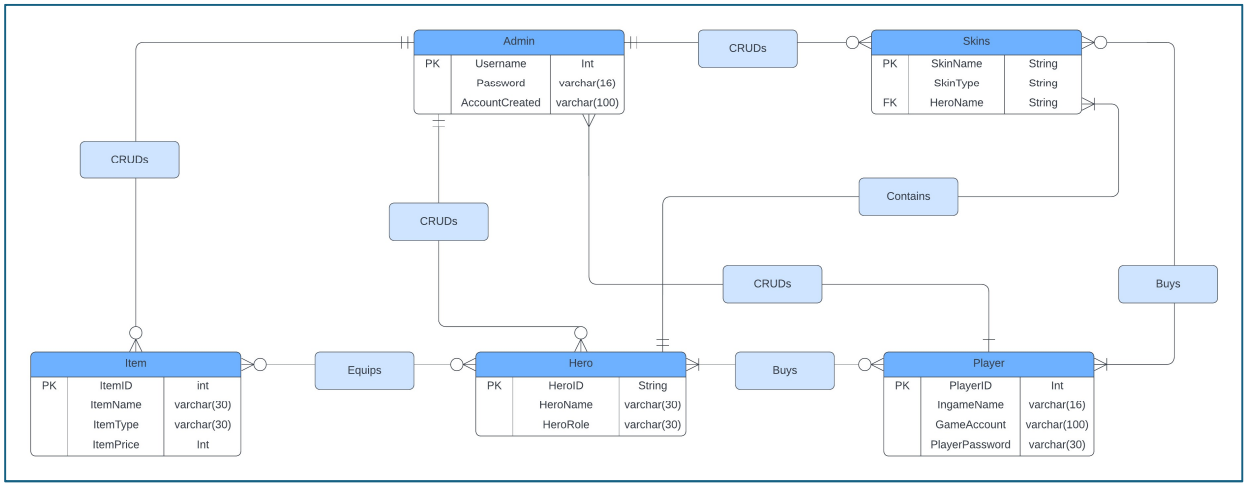
Name	Data Type	Constrain	Description
SkinID	INT	NOT NULL, PRIMARY KEY	A unique identifier for each skin.
SkinName	VARCHAR(30)	NOT NULL, UNIQUE	The unique name of the skin.
SkinType	VARCHAR(30)	NOT NULL	The type or category of the skin (e.g., epic, rare).
HeroID	INT	NOT NULL, FOREIGN KEY REFERENCES hero(HeroID) ON DELETE CASCADE ON UPDATE CASCADE	The ID of the hero this skin belongs to, linked to the hero table.

2.1.1.5 Data dictionary for Element: *Item*

Name	Data Type	Constrain	Description
ItemID	INT	NOT NULL, PRIMARY KEY	A unique identifier for each item.
ItemName	VARCHAR(30)	NOT NULL, UNIQUE	The unique name of the item.
ItemType	VARCHAR(30)	NOT NULL	The type or category of the item (e.g., attack, magic).
ItemPrice	INT	NOT NULL	The price of the item in the game currency.

2.2 MySQL database design (Relational database)

2.2.1 Conceptual diagram



2.2.2 Description

The conceptual diagram represents a database schema or entity-relationship model designed for a gaming or e-commerce platform. It manages user accounts, items, heroes, skins, and transactions through a structured system of tables. The diagram uses blue rectangular boxes (entity) containing fields (attributes) with specified data types. Some fields are marked as primary keys (PK) or foreign keys (FK) to establish relationships. Light blue rectangles with lines and arrows connect the tables, indicating actions or associations such as "CRUDs (Create, Read, Update, and Delete)," "Equips," "Buys," and "Contains." This schema supports a dynamic environment where users can create and manage content, players can acquire heroes, and heroes can be customize with items and skins.

The system appears to facilitate a website or game with features like user authentication, gameplay mechanics (e.g., equipping items), and transactional operations (e.g., buying heroes). Below is a detailed description of each table and its purpose, followed by an explanation of the relationships between them.

2.2.3 Purpose of Tables

2.2.3.1 Purpose of Admin Table

This table manages user accounts for administrators. It stores authentication details (username and password) and account creation information, serving as the central hub for user management and access control within the system.

2.2.3.2 Purpose of Player Table

This table focuses on player-specific data, managing in-game identities and authentication. It is similar to the User Admin table but is dedicated to players, supporting their profiles and login credentials.

2.2.3.3 Purpose of Hero Table

This table manages hero characters, which are likely central to gameplay. It stores their identities and roles, providing the foundation for hero selection, customization, and interactions with skins.

2.2.3.4 Purpose of Skin Table

This table handles cosmetic items or skins, which enhance the appearance of heroes or items in the game. The foreign key connection ensures skins are tie to specific heroes, enabling hero-specific customization.

2.2.3.5 Purpose of Item Table

This table stores details about items that can be purchase or equip in the game. It supports an in-game economy or inventory system by tracking item names, types, and prices.

2.2.4 Relations

From Table	To Table	Relation
Admin	Player	An admin can **CRUD** zero to many player.
Admin	Hero	An admin can **CRUD** zero to many heroes.
Admin	Skin	An admin can **CRUD** zero to many skins.
Admin	Item	An admin can **CRUD** zero to many items.
Player	Admin	A player can be **CRUD** by one and only one admin.
Player	Skin	A player can **Buy** zero to many skins.
Player	Hero	A player can **Buy** one to many heroes.
Skin	Admin	A skin can be **CRUD** by one and only one admin.
Skin	Player	A skin can be **Bought** by one to many players.
Skin	Hero	A skin **Contains** by one and only one hero.
Hero	Admin	A hero can be **CRUD** by one and only one admin.
Hero	Player	A hero can be **Bought** by zero to many player
Hero	Skin	A hero can **Contain** one to many skins.
Hero	Item	A hero can **Equip** zero to many items.
Item	Admin	An item can be **CRUD** by one and only one admin.
Item	Hero	An item can be **Equip** by zero to many hero

3 References

Notion – The all-in-one workspace for your notes, tasks, wikis, and databases. (n.d.). Notion. https://believed-bongo-319.notion.site/CTINFMGL-Project-Specifications-19296450aad180aea9ebf27987415f4c