RDBS

Ahmet Oğuz Ergin

Introduction

This document provides an example of a basic relational database structure and its relationships, demonstrated through a code-stream project https://github.com/ahmettoguz/code-stream-docker-config. The goal is to explore and revisit key concepts that are often forgotten over time but are essential during the initial phases of a project.

Project Description

The code-stream project aims to deliver redemption codes to players for various games. For the purpose of this guide, the documentation covers one-to-one, one-to-many, and many-to-many relationships.

ERD

The following is the Entity Relationship Diagram (ERD) created with Chen Notation in pgAdmin. While the diagram primarily displays 1-N relationships, all relationship types are covered and explained in detail in the subsequent sections.

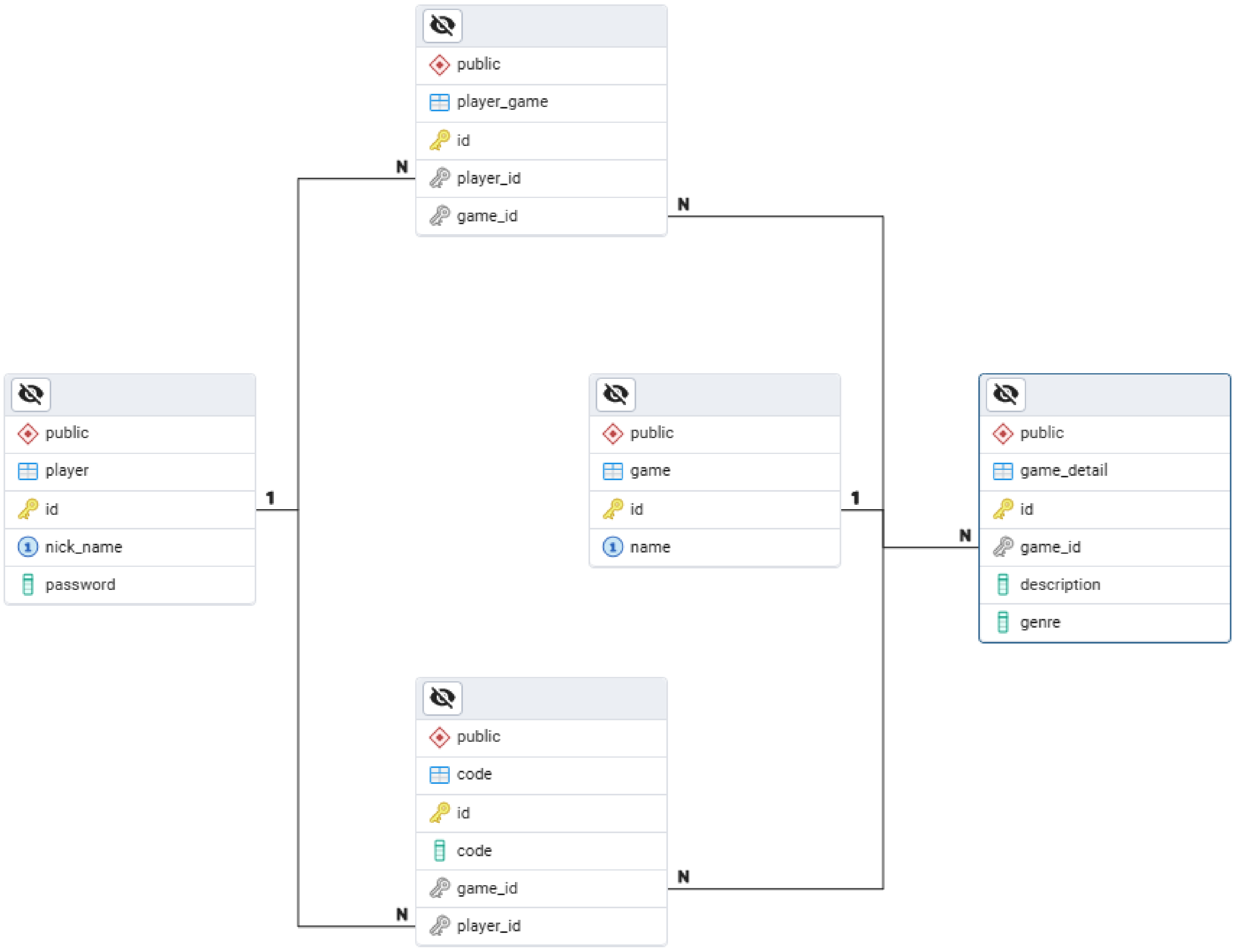


Figure 1: Entity Relationship Diagram of code-stream Project

Relationship Between Objects

To better understand the structure of the database, the relationships between objects are explained as follows.

Game and Game Detail Relation (One to One)

- A game can have only one game detail.
- A game detail can only belong to one game.

Game and Code Relation (One to Many)

- A game can have multiple redemption codes.
- A redemption code can only belong to one game.

Player and Code Relation (One to Many)

- A player can own multiple redemption codes.
- A redemption code can only belong to one player.

Player and Game Relation (Many to Many)

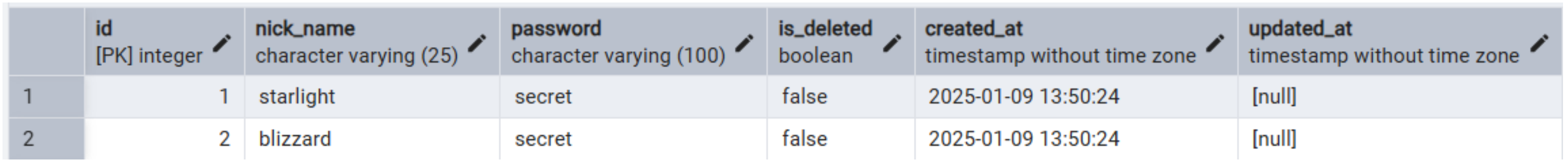
- A player can be registered for multiple games.
- A game can have multiple players.

Tables

Each table and field description is explained below, along with example data.

player

The simple player table is shown in the following figure.

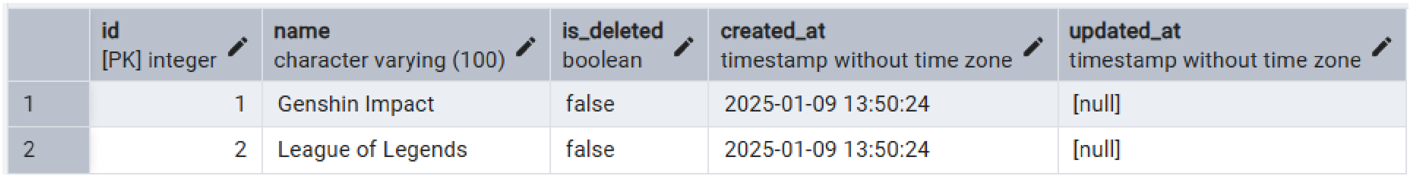


	id [PK] integer	nick_name character varying (25)	password character varying (100)	is_deleted boolean	created_at timestamp without time zone	updated_at timestamp without time zone
1	1	starlight	secret	false	2025-01-09 13:50:24	[null]
2	2	blizzard	secret	false	2025-01-09 13:50:24	[null]

Figure 2: player Table With Data

game

The simple player table is shown in the following figure.



	id [PK] integer	name character varying (100)	is_deleted boolean	created_at timestamp without time zone	updated_at timestamp without time zone
1	1	Genshin Impact	false	2025-01-09 13:50:24	[null]
2	2	League of Legends	false	2025-01-09 13:50:24	[null]

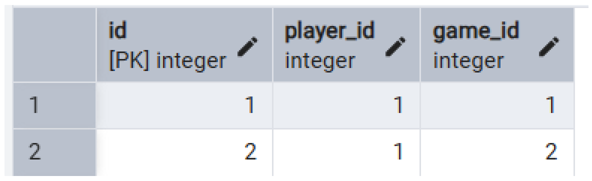
Figure 3: game Table With Data

player_game

Since there is a many-to-many relationship between players and games, an additional table is required to store their relationship. There are two options for defining the primary key:

- Composite Key (player_id and game_id): This approach works but has a drawback. If "ON DELETE SET NULL" is assigned, and you want to use it (for example, if Player 1 is removed from the player table but you don't want to lose related data), setting the player_id to NULL would not be possible because the primary key cannot contain NULL values.

- Using an id field as the primary key This approach is simpler, easier to use, and more straightforward to understand.



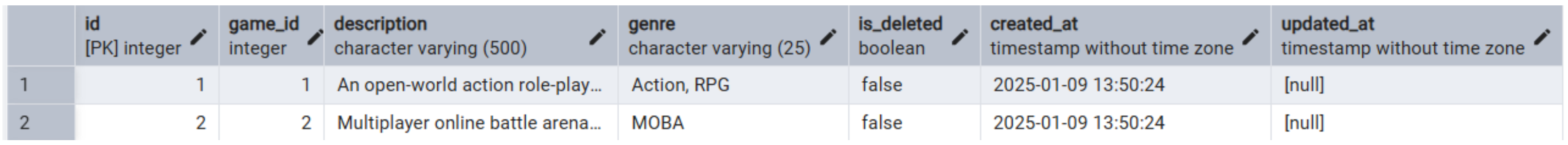
	id [PK] integer	player_id integer	game_id integer
1	1	1	1
2	2	1	2

Figure 4: player_game Table With Data

game_detail

The game detail table stores additional data for each game. There is a one-to-one relationship between the game and its details, with the game_id field acting as a foreign key referencing the id field in the game table. There are two options for defining the primary key:

- Using the game_id field as both the primary and foreign key: This approach works, but as mentioned earlier, if you want to set the game_id to NULL upon the deletion of a game, it would not be possible because the primary key cannot contain NULL values.
- Using a id field as the primary key: This is a simpler and more intuitive approach. In this case, the game_id field is UNIQUE, as there is a one-to-one relationship between the game and its details. The only distinction between a one-to-many and a one-to-one relationship is that the foreign key in a one-to-one relationship must be UNIQUE. It's that simple.

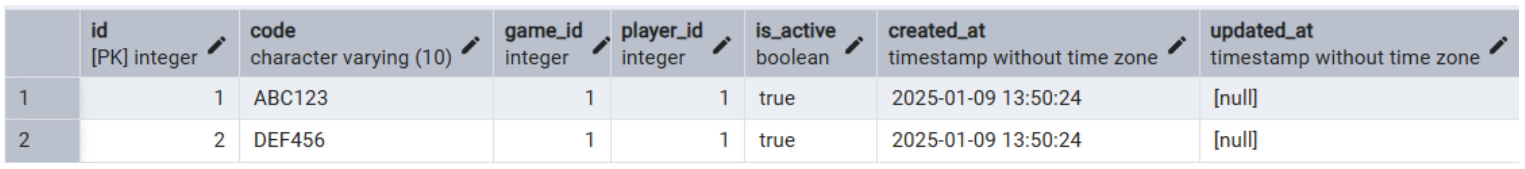


	id [PK] integer	game_id integer	description character varying (500)	genre character varying (25)	is_deleted boolean	created_at timestamp without time zone	updated_at timestamp without time zone
1	1	1	An open-world action role-play...	Action, RPG	false	2025-01-09 13:50:24	[null]
2	2	2	Multiplayer online battle arena...	MOBA	false	2025-01-09 13:50:24	[null]

Figure 5: game_detail Table With Data

code

This table has two relationships, both of which are one-to-many.



	id [PK] integer	code character varying (10)	game_id integer	player_id integer	is_active boolean	created_at timestamp without time zone	updated_at timestamp without time zone
1	1	ABC123	1	1	true	2025-01-09 13:50:24	[null]
2	2	DEF456	1	1	true	2025-01-09 13:50:24	[null]

Figure 6: code Table With Data