Data And Applications Project Phase 4

Group 14 - SQ(L)uad

December 2023

1 Phase 4 Highlights

We made a few changes to our database model to fit MySQL constraints. As MySQL doesn't dynamically update itself we had to make do with some tables. Also, we decided to get rid of some attributes as they were causing unnecessary redundancy in the database.

2 SQL Queries

2.1 Selection Queries

2.1.1 To retrieve a list of all opponents faced by a particular player

Query: SELECT p.PlayerID AS OpponentId, p.PlayerName AS OpponentName FROM Players_Attack_Log pal JOIN Player p ON pal.Defender_ID = p.PlayerId WHERE pal.Attacker_ID = %s UNION SELECT p.PlayerID AS OpponentId, p.PlayerName AS OpponentName FROM Players_Attack_Log pal JOIN Player p ON pal.Attacker_ID = p.PlayerID WHERE pal.Defender_ID = %s;

This query is useful when you want to retrieve a list of all opponents faced by a specific player in the game. It filters the attack logs to show only those entries where the specified player was the attacker. The DISTINCT keyword ensures that each opponent is listed only once, even if the player attacked the same opponent multiple times.

2.1.2 To retrieve a list of all defenses with the maximum hitpoints

Query: SELECT Name AS MaxHitpointsDefense FROM Defences ORDER BY Hitpoints DESC LIMIT 1;

This query is useful when you want to find all defenses with the maximum hitpoints in your game.

2.2 Projection Queries

2.2.1 To retrieve the names of all players in a league

Query: SELECT p.PlayerName FROM Player p, Leagues l WHERE l.Name = %s AND p.Trophies BETWEEN l.LowerLimitOfTrophyRange AND l.UpperLimitOfTrophyRange;

This query is useful when you want to retrieve the names of all players who are part of a specific league.

2.2.2 To retrieve the names of all members of a clan

Query: SELECT p.PlayerName FROM Player p JOIN Player_part_of_clan pc ON p.PlayerID = pc.PlayerID WHERE pc.Clan_ID = %s;

This query is useful when you want to retrieve the names of all players who are members of a specific clan.

2.3 Aggregation Queries

2.3.1 To calculate the highest win-lose ratio of a clan

Query: SELECT c.Name, COALESCE(c.Num_Wins / NULLIF(c.Num_Losses, 0), NULL) AS WinLoseRatio FROM Clan c ORDER BY WinLoseRatio DESC LIMIT 1;

This query is useful when you want to find the clan with the highest win-lose ratio. It calculates the win-lose ratio for each clan, filters out clans with no losses, and then orders the result set to find the clan with the highest ratio.

2.3.2 To calculate the average trophies of a clan

Query: SELECT c.Name, COALESCE(c.Medals/ NULLIF(c.NumberOfMembers, 0), NULL) AS Average_Medals FROM Clan c WHERE c.Clan_ID=%s;

This query is useful when you want to find the average number of trophies for each clan.

2.4 Search Queries

2.4.1 To search for clan names that contain string "War"

Query: SELECT Name FROM Clan WHERE Name LIKE '%War%';

This query is useful when you want to find clans with names containing the specific string "War." The LIKE operator with % is used for pattern matching, allowing flexibility in searching for substrings within the clan names.

2.4.2 To search for player names that start with alphabet "U"

Query: SELECT PlayerName FROM Player WHERE PlayerName LIKE 'U%';

This query is useful when you want to find players whose names start with the letter "U."

2.5 Insertion Queries

2.5.1 To add the details of a new player who joins the game

Query: INSERT INTO Player (PlayerID,PlayerName, Elixir, Gold, DarkElixir, Gems, Trophies, XPLevel, TownHallLevel) VALUES (%s, %s, 0, 0, 0, 0, 0, 0, 0);

This query is useful when a new player joins the game, and you need to add their details to the Player table. We have to ensure that the PlayerID is unique, as it is specified as the primary key. We may also need to adjust the values based on the actual structure and requirements of our database

2.5.2 To add the details of new troop added to the game

Query: INSERT INTO Troops (Name, Hitpoints, DamagePerSec, TrainingTime, DamageType, MovementSpeed, BarrackLevelUnlock, Currency, UpgradeTime, UpgradeCost, Targets, HousingSpace) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s);

This query is useful when a new troop is added to the game, and we need to add its details to the Troops table. We might have to update some constraints of some attributes while inserting some values.

2.6 Updation Queries

2.6.1 To update the Town Hall of a player who got upgraded

Query: UPDATE Player SET TownHallLevel = %s WHERE PlayerID = %s;

This query is useful when a player upgrades their Town Hall, and you need to update the TownHallLevel in the database.

2.6.2 To update the Trophy Count of a player after a battle

Query1: INSERT INTO Players_Attack_Log (Attacker_ID, Defender_ID, Result, Trophy_Transaction, Battle_Time) VALUES (%s, %s, %s, %s, %s);
Query2: UPDATE Player SET Trophies=Trophies+%s WHERE PlayerID = %s;
Query3: UPDATE Player SET Trophies=Trophies-%s WHERE PlayerID = %s;

This query is useful when you want to update a player's trophy count after a battle.

2.7 Deletion Queries

2.7.1 To delete the details of a troop who has been removed from the game

Query: DELETE FROM Troops WHERE Name = %s;

This query is useful when you want to remove the details of a troop that is no longer part of the game. CASCADE is used wherever needed.

2.7.2 To delete the details of a clan whose member count is 0

Query: DELETE FROM Clan WHERE NumberOfMembers = 0;

This query is useful when you want to remove the details of a clan that has no members.

2.8 Analysis Queries

2.8.1 To investigate how the attack timing are affected at a particular Town Hall

Query: SELECT p.TownHallLevel, SEC_TO_TIME(AVG(TIME_TO_SEC(pa.Battle_Time))) AS AvgAttackTiming FROM Players_Attack_Log pa JOIN Player p ON pa.Attacker_ID = p.PlayerID WHERE p.TownHallLevel = %s GROUP BY p.TownHallLevel;

This query is useful when you want to investigate how attack timings are affected at a particular Town Hall level. We average out all the attack timing corresponding to a particular townhall level and then display it. This useful for people who would want to get such analysis to improve their performance in comparision to other people of a similar level.

2.8.2 To find the most popular troop, spell and hero

Query for spell: SELECT Spells.Name AS MostPopularSpell FROM Troops_Spells_Heroes_part_of_Army_composition LEFT JOIN Spells ON Troops_Spells_Heroes_part_of_Army_composition.SpellName = Spells.Name GROUP BY Spells.Name ORDER BY COUNT(*) DESC LIMIT 1;

Query for troop: SELECT Troops.Name AS MostPopularTroop FROM Troops_Spells_Heroes_part_of_Army_composition LEFT JOIN Troops ON Troops_Spells_Heroes_part_of_Army_composition.TroopName = Troops.Name GROUP BY Troops.Name ORDER BY COUNT(*) DESC LIMIT 1;
Query for hero: SELECT Heroes.Name AS MostPopularHero FROM Troops_Spells_Heroes_part_of_Army_composition LEFT JOIN Heroes ON Troops_Spells_Heroes_part_of_Army_composition.HeroName = Heroes.Name GROUP BY Heroes.Name ORDER BY COUNT(*) DESC LIMIT 1;

This query is useful when you want to find the most popular troop, spell or hero based on its usage in various army compositions. Here we count the usage of all the troops in different army compositions and then we display the troop that is involved in most army compositions. The same process is followed for spell and hero. This query could be useful way for Clash of Clans fans to see if their favourite troop, spell or hero is the most popular.

Important Note

Before running the python queries with the the populated database, we need to run another .sql file. This file creates a new user and grants all permissions to this new user over the SQL database. This is to bypass the sudo access that the MySQL on our device needs.