CT3532 Database Systems 2

Assignment 1

Name: Maxwell Maia Student ID: 21236277
Name: Alexis Shaju Student ID: 21516056

Schema and explanation

games

game_id	home_team_id	away_team_id	heap_file
(Primary Key)	(FK to player table)	(FK to player table)	

The games table stores a record for each match.

The heap file is linked to each game by way of a heap file attribute. This allows the file for each game to be retrieved.

team

team_id	team_name	home_venue	manager
(Primary Key)			

This table stores information on teams.

player

player_id	player_name	squad_number	team_id
(Primary Key)			(FK to team table)

This table stores information on the players. There is a foreign key to the team table to link the player to the team.

initial_players

initial_players_id	game_id	team_id	player_id
(Primary Key)	(FK to games table)	(FK to team table)	(FK to player table)

This table keeps track of the initial 11 players on the pitch for each team for each match.

substitution

substitution_id	game_id	player_id_in	player_id_out	time
(Primary Key)	(FK to games table)	(FK to player table)	(FK to player table)	

This table keeps track of the substitutions for each match as well as the time in the match that the substitution took place.
player_id_in is the player substituted in.
player_id_out is the player substituted out.

sent_off

sent_off_id	game_id	player_id	time
(Primary Key)	(FK to games table)	(FK to player table)	

This table stores the players that are sent off, the time they are sent off for each game.

goal

goal_id	game_id	player_id	team_id	time
(Primary Key)	(FK to games table)	(FK to player table)	(FK to team table)	

The requirements requested that the "number" of each scorer is recorded for each goal. I will instead store the player_id. The squad_number is determinable using the player_id in the player table.

After a record is inserted or a record is updated, the goals for each team for that match is summed up and the match_results table below is updated.

match_results

match_results_id	game_id	home_team_id	away_team_id	home_team_score	away_team_score
(Primary Key)	(FK to games table)	(FK to team table)	(FK to team table)		

This table contains redundant data. It contains information on the score for each team for each match but that is already stored in the database (because it can be calculated) from the goal table.

There is a danger that when a record in this table is updated, e.g. the score is changed, without first updating the goal table, then the score in the matches table could be different from what is stored in the goal table. This is an update anomaly. To prevent this, this table will block all update queries done by the user, giving the message: "Update query blocked. Please update the goal table.

After something is inserted into this table or the table is updated, the number of wins and draws for each team are calculated and the league_results table is updated.

league_results

league_points_id	team_id	wins	draws
(Primary Key)	(FK to team table)		

A table for storing the wins and draws for each game in the league.

De-normalization was implemented for more efficient queries. This table contains redundancy in that the number of wins and draws can be calculated from the match_results table and the goals table.

This will not cause any anomalies because we do not allow this table to be updated or anything inserted into it without updating the goal and league_results tables. A user attempting to update this table gives the message: "Query blocked. Please update the goal table."

After a record is inserted into this table, or a record is updated, the number of points is calculated for that team. This is possible because the team_id is known when this update takes place. The number of points is updated in the league_points table.

league_points

league_points_id	team_id	points
(Primary Key)	(FK to team table)	

We want to store the points for each team in the league because this will likely be accessed many times by apps that show the current league points. This is de-normalization for efficiency. This causes redundancy, since the league points can be calculated in a query using the number of wins and draws in the league_results table. This redundancy will not cause any anomalies because we will not allow any updates to this table without updating the goal table and subsequently the match_results and league_results table. A user attempting to update this table gives the message: “Query blocked. Please update the goal table.”

Assumptions

Squad number is referring to a number of the player on the team. The squad number is unique within a team. The number is not unique to the squad numbers of another team. For example: Team A has players with squad numbers 1-16 and Team B has players with squad numbers 1-16.

Substitution time is the time of the game where a player was substituted.

Queries

1. List all players playing for a given team.

```
SELECT player.name
FROM player
INNER JOIN team ON player.team_id = team.team_id
WHERE team.team_name = ‘Liverpool’;
```

2. List all players who have scored in a given game.

```
SELECT DISTINCT player.name
FROM player
INNER JOIN goal ON player.player_id = goal.player_id
WHERE goal.game_id = 21;
```

3. List the top five goal scorers in the league.

```
SELECT player.name, COUNT(*) AS goalsScored
FROM player
INNER JOIN goal ON player.player_id = goal.player_id
ORDER BY goalsScored
LIMIT 5;
```

4. List all teams and the amount of points they have so far.

```
SELECT team.team_name, league_points.points
FROM team
INNER JOIN league_points ON team.team_id = league_points.team_id;
```

Discussion of design choices

The games table:
The information of where the game is being played is stored because we are storing the team_id's as either home team or away team. We also have the home venue stored in the team table so the location of the match can be determined. This is suitable because it is not necessary to query the location of a match very often but when it is required, it is possible with relatively minimal computational power.

The initial_players table:
There is a requirement to know all players playing. The initial_players table stores the initial 11 players on the pitch for each team for each match. Using this table in conjunction with the substitution table, we are keeping enough information to calculate which players are on the pitch at any given time.

Every time a goal is scored:
A record is inserted into the goals table. The total goals for that match is calculated from the goals table and the match_results table is updated. It is possible to calculate the total goals for each team for only that one match because game_id is known when a goal is scored. This means that all the matches are not updated every time a goal is stored; only the one match. This minimizes the computations necessary when a goal is scored.

Then since that updates the match_results table:
The number of wins and draws for each team are calculated and the league_results table is updated.

Then since that updates the league_results table:
The number of points for each team are calculated and the league_points table is updated.

My design has a lot of processing requirements every time a goal is scored because after the goal is inserted into the goals table, the match_results, league_results and league_points tables need to be updated too.

The advantages of this outweigh the disadvantages. This design choice allows the following queries to be very efficient.
1. the score of each match
2. the total wins of each team
3. the current points of each team in the league.

This database is being used for a soccer league. There will be more queries requesting the above 3 pieces of information than there will be updates to the goals. We don’t want the database to slow down when a game is on and everyone is checking the score at the same time. Using my design the bulk of that information is calculated once, as soon as a goal is scored. This means that users will get the information they want to see quickly and the database will run smoothly even during live games.