National Junior Basketball Association Application

Group 5

Matthew Frazier

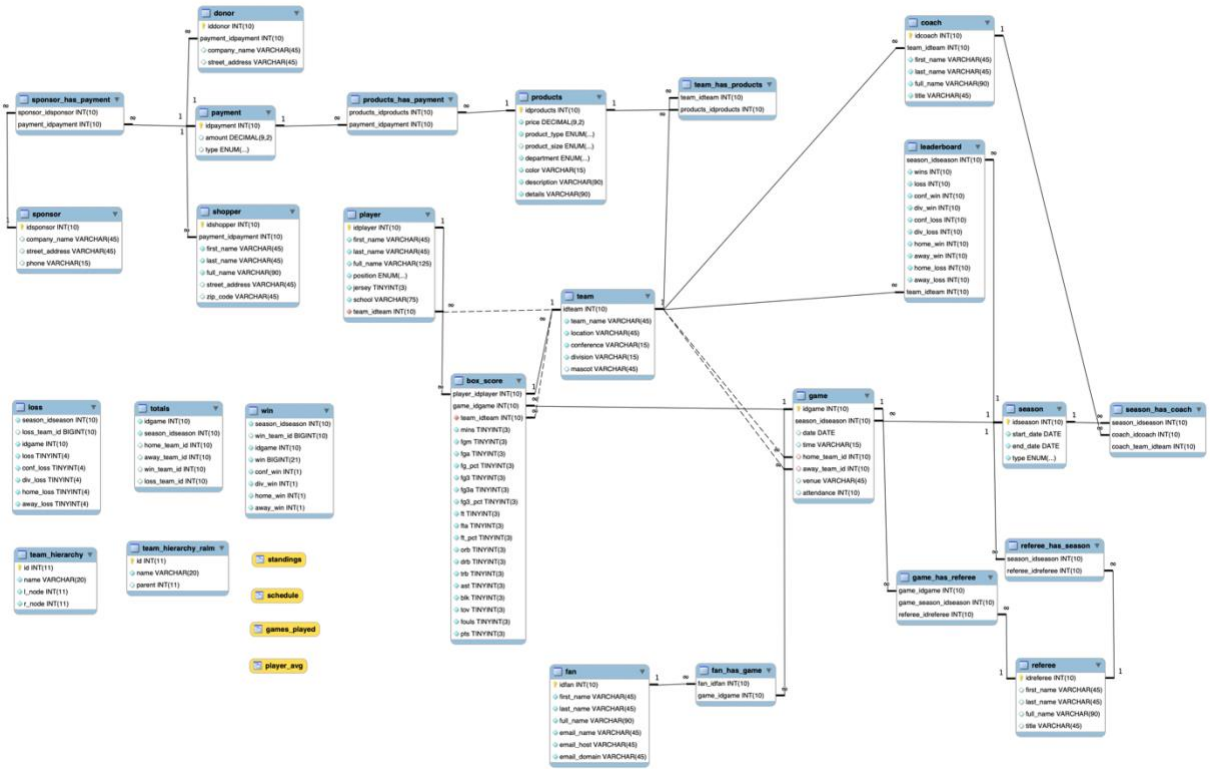
11/21/19
In-class Presentation
2:30 PM - 3:00 PM

Outline

- Database Design Phase – Models (EER and ERD)
- Normalization
- Data generation
- Advanced SQL
- Functions
    - Single Point of Access
    - Transaction Management
- Stored Procedure
- Query Processing
    - Optimizations
    - Reporting & Analytics
- Database Security
    - AWS Infrastructure
    - EC2 and RDS Endpoints
    - VPC, Security Group, & Subnets
- Concurrency Control & Recovery
    - DB size (~4.5 GB of data)
    - Failover
    - Read Replica - t2.micro and t2.medium
- Web Application
    - PHP & MySQL connection

NJBA Database

- Database Design Phase – Models (EER and ERD)
    - During the database design phase, I prepared an ERD of the conceptual and logical design models. The logical model was converted into a EER using the MySQL Workbench client. Refer to the documents in njba/docs/models for images. The end result physical design is below (post normalization).



- Normalization
    - Going through the full triage process of designing the NJBA database was beneficial. I preformed 3rd Normal Form (3NF) normalization on the database by removing redundancy in the schema architecture in some areas while adding complexity in others. For example, in my logical design, both the relations, game and referee, had no formal notion of time. In the transformation to the physical design EER, the relations referee_has_season and game_has_referee were created to normalize the data consistency and accuracy which follows the ACID principles in database modeling.

- Data generation
    - The programming language used to generate the fictious data was Python 3.7 and BASH Scripts.
    - With a healthy combination of an iterative processing and normalization, I have generated a full 20 GB of data. The code provided in the NJBA repository shows the full dataset generation process. However, the NJBA database only contains ~4.5 GB. This is for a few reasons which will be discussed in coming DB Size section. The only outside resource used while creating this dataset was for the game scheduler.

      The online game scheduler tool used https://www.hometeamsonline.com/. This website was needed because I wanted to focus specifically on the database and not get troubled with scheduling optimization. The field of scheduling optimization is a complex task within itself and this tool was leveraged to solve that problem. The game schedule settings were as follows:
        - Pre Season Scheduling:
            - 30 different teams
            - Each team plays at least 2 games between time series of 2 – 3 weeks (depending on the year
        - Regular Season Scheduling:
            - 30 different teams

        - Each team plays at least 200 games between time series of approximately 7 months (depending on the start date)
    - Post season Scheduling:
        - Top 16 teams of standings from the *Regular Season* results only
        - Each team plays at least 2 games between time series of 2 – 3 weeks (depending on the end date)
- Advanced SQL
    - The version of MySQL used in the NJBA database is MySQL 5.7. This is an older version of MySQL as compared to its latest version of MySQL 8.0. With that, there are some features that are not compatible between versions such as Common Table Expressions (CTE) and Recursion. Although these nuances mean that the NJBA database does not support them, they can be simulated in other ways as shown in the njba/sql/optimizations.sql and njba/sql/reporting-and-analytics.sql files. Using nested subqueries, all CTE queries can be converted to fit into MySQL 5.7 version support. Recursion is also simulated using 2 different patterns: Adjacency List Model and Nested Set Model.
- Functions
    - The functions in the NJBA serve 2 purposes:
        - Single Point of Access
            - Acting as a single place to update the start_date and end_date when running the stored procedure is key. This is essential because SELECT queries cannot use variables to dynamically set the value of a stored procedure over a given period. To mitigate this, a function can be used in the SELECT query instead which returns a dynamic attribute.
        - Transaction Management
            - When processing a transaction, the series of intermediary steps that ensure ACID principles of the transaction occur in the function. For example, as shown in the NJBA database, the function add_fan simulates a Fan purchasing a ticket and coming to a game. This transaction affects multiple relations in the database including the fan, fan_has_game, and game tables.
- Stored Procedure
    - During the data generation process, I simulated multiple people purchasing between 1 and 6 items from the NJBA store. The NJBA store is not in the MVP but it does relate to the stored procedure. Without cross-referencing the dataset in the products relation or team_has_products relation, instead, there stored procedure that sums all the products in a given payment_id.
    - During the analytics and optimizations phase, I created a stored procedure that updates the standings of the database. This is a fictious dataset but if this were an active basketball league that had data being added to it daily, this stored procedure could be called by a cron job or other daily task management system which would automatically update the leaderboard of the NJBA league.

- Query Processing
    - Optimizations
        - As a result of the limitations of this project (with a relative maximum of 20 tables) and the combination of using MySQL 5.7, the optimizations performed on the NJBA database include creating additional indices, removing order by clauses when using a group by clause to reduce duplicate sorting calls, among other optimizations as described in the nbja/docs/Reporting-and-Analytics/Optimizations.pdf document.
    - Reporting & Analytics
        - The reporting measures for the NJBA database include aggregate summaries of the statistics of the box_score table. There are no sales data reports in the MVP.

- Database Security
    - AWS Infrastructure
        - The Amazon Web Services (AWS) infrastructure provides excellent support for cloud based applications. Given that the MVP for this project was a web application, I decided to leverage AWS. The initial scope of this project were to try to run a serverless architecture but while going to implement this project, I was not familiar with that process so I decided to use an Apache MySQL web server on an Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type.
    - EC2 and RDS Endpoints
        - The database is hosted on an AWS MySQL RDS t2.micro instance. This is a free tier image and only supports a maximum of 20 GB of data.
    - VPC, Security Group, & Subnets
        - The NJBA database is hosted under a private VPC, Security Group, and Subnets in AWS VPC.

- Concurrency Control & Recovery

    - DB size (~4.5 GB of data)
    - Failover
        - The RDS instance is hosted in Multi-Availability Zones for failover recovery protection.
    - Read Replica - t2.micro and t2.medium
        - I discovered during the query optimization phase that there are some high CPU Utilization issues at times which lead me to create a larger read-only replica instance.

- Web Application
    - PHP & MySQL connection
        - The website application includes embedded SQL calls to the NJBA read-only t2.medium instance for performance enhancement.