CS 121 Final Project Reflection

Due: March 16th, 11:59PM PST

This reflection document will include written portions of the Final Project. Submit this as `reflection.pdf` with the rest of the required files for your submission on CodePost.

For ease, we have highlighted any parts which require answers in **blue**.

Student name(s): Halle Blend, Eli Seiner

Student email(s): hblend@caltech.edu, eseiner@caltech.edu

Part L0. Introduction

Answer the following questions to introduce us to your database and application; you may pull anything relevant from your Project Proposal and/or README.

Database/Application Overview

What application did you design and implement? What was the motivation for your application? What was the dataset and rationale behind finding your dataset?

Database and Application Overview Answer (3-4 sentences) :

***Our proposed database is an NFL historical player stats database.* We found this database from https://www.kaggle.com/kendallgillies/nflstatistics?select=Basic_Stats.csv, which stores several csv files regarding historical NFL stats for players from 1924 to 2016. We are selecting five csv files to focus on in an overall category of offense in football: Basic Stats (essentially player information), Career Passing Stats, Career Rushing Stats, Career Receiving Stats, and Career Defensive Stats. We can apply this data by using our created dataset to create an application that makes it easier to compare different players across different eras in order to decide which players have strong enough stats to be considered for the hall of fame, an extremely prestigious award granted to only the most elite players .**

Data set (general or specific) Answer:

Kaggle: NFL Statistics

https://www.kaggle.com/kendallgillies/nflstatistics/version/1?select=Basic_Stats.csv

- **Basic Stats**
- **Career Stats Passing**
- **Career Stats Receiving**
- **Career Stats Rushing**
- **Career Stats Defensive**

Client user(s) Answer:

The intended client user base are fans of football, people who need some database to store historical information about football (records, history, etc.), people who serve as scouts that need reliable

information about player history, and people who serve on the hall of fame voting committee who need access to a database to easily compare players.

Admin user(s) Answer:

The intended admin users are NFL statisticians, who have access to reliable NFL statistics and can continually update the tables, creating an updated database of all player's career statistics.

Part A. ER Diagrams

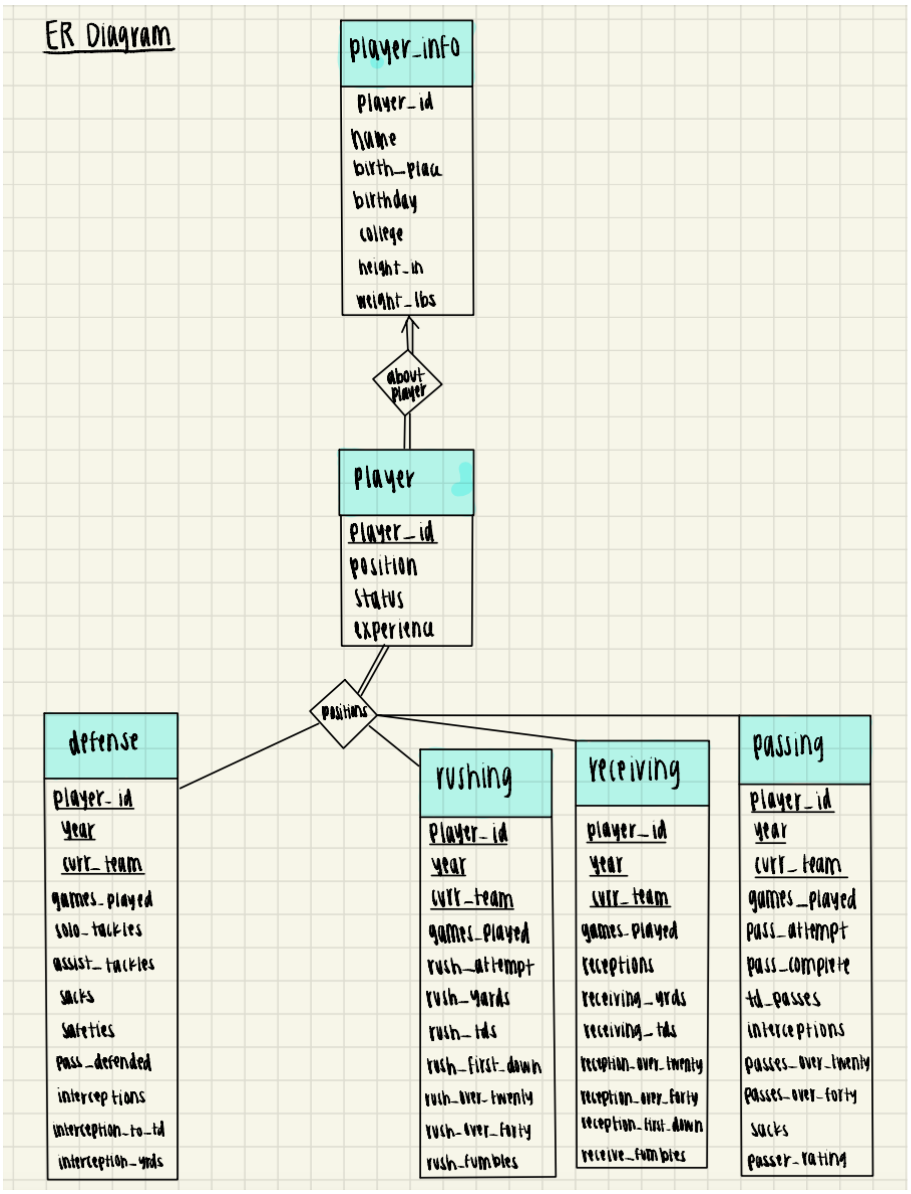
As we've practiced these past few weeks, the ER model is essential for designing your database application, and we expect you to iterate upon your design as you work through the ER and implementation steps. In this answer, you should provide a full ER diagram of your system. Your grade will be based on correct representation of the ER model as well as readability, consistency, and organization.

Notes: For this section **only**, we will allow (and encourage) students to share their diagrams on Discord (**#er-diagram-feedback**) to get feedback from other students on their ER diagrams given a brief summary of your dataset and domain requirements. This is offered as an opportunity to test your ER diagrams for accuracy and robustness, as another pair of eyes can sometimes catch constraints that are not satisfied or which are inconsistent with your specified domain requirements.

Requirements:

- Entity sets, relationship sets, and weak entity sets should be properly represented (also, do not use ER symbols not taught in class)
- Mapping cardinalities should be appropriate for your database schema, and in sync with your DDL
- Participation constraints should be appropriate and in sync with your DDL (total, partial, numeric)
- Use specialization where appropriate (e.g. *purchasers* and *travelers* inheriting from a *customers* specialization in A6)
- Do not use degrees greater than 3 in your relationships, do not use more than one arrow in ternary relationships.
- Use descriptive attributes appropriately
- Underline primary keys and dotted-underline discriminators
- Expectations from A6 still apply here
- Note: You do not need ER diagrams for views

ER Diagrams:



ER Diagram
player_info
player_id
name
birth_place
birthday
college
height_in
weight_lbs
about player
player
player_id
position
status
experience
positions
defense
player_id
year
curr_team
games_played
solo_tackles
assist_tackles
sacks
safeties
pass_defended
interceptions
interception_to_td
interception_yrds
rushing
player_id
year
curr_team
games_played
rush_attempt
rush_yards
rush_tds
rush_first_down
rush_over_twenty
rush_over_forty
rush_fumbles
receiving
player_id
year
curr_team
games_played
receptions
receiving_yrds
receiving_tds
reception_over_twenty
reception_over_forty
reception_first_down
receive_fumbles
passing
player_id
year
curr_team
games_played
pass_attempt
pass_complete
td_passes
interceptions
passes_over_twenty
passes_over_forty
sacks
passer_rating

Part B. DDL (Indexes)

As mentioned in Part B, you will need to add at least one index at the bottom of your **`setup.sql`** and show that it makes a performance benefit for some query(s).

Here, describe your process for choosing your index(es) and show that it is used by at least one query, which speeds up the performance of the same query on a version of the same table without that index. You may find lec14-analysis.sql and Lecture 14 slides on indexes useful for strategies to choose and test your indexes. **Remember that indexes are already created in MySQL for PKs and FKs, so you should not be recreating these.**

Index(es):

This index makes it faster to look up touchdown passes.

CREATE INDEX idx_nfl_passing ON passing (td_passes);

This index makes looking up player's former colleges faster.

CREATE INDEX idx_nfl_college ON player_info (college);

Justification and Performance Testing Results:

One of our queries was looking at how many quarterbacks that played for the Dallas Cowboys that had at least 10 touchdowns in a season. We thought an index would be useful because touchdown passes are a common stat in football that would be looked at often. We made an index to make this query run faster when looking at the number of touchdown passes. Our index makes a quick search table to cut down what it searches for to make the query run faster. Before the index it took 0.0041 seconds to run and after the index it took 0.0022 seconds to run.

Another query we had was looking at the colleges of each defensive player. We made an index to make this query run faster when looking at a player's former colleges. Our index makes a quick search table to cut down what it searches for to make the query run faster. Before the index it took 0.052 seconds to run and after the index it took 0.041 seconds to run.

Part C. Functional Dependencies and Normal Forms

Requirements (from Final Project specification):

- Identify *at least 2 functional dependencies* in your database
- Choose and justify your decision for the normal form(s) used in your database for at least 4 tables (if you have more, we will not require extra work, but will be more lenient with small errors). BCNF and 3NF will be the more common NF's expected, 4NF is also fine (but not 1NF).
    - Your justification will be strengthened with a discussion of your dataset breakdown, which we expect you to run into trade-offs of redundancy and performance.
- For two of your relations having at least 3 attributes (each) and at least one functional dependency, prove that they are in your chosen NF, using similar methods from A7.
    - If you have identified functional dependencies which are not preserved under a BCNF decomposition, this is fine
- Expectations from A7 still apply here.

Functional Dependencies:

1. {player_id} → {name, birth_place, birthday, college, height_in, weight_lbs}
    a. In the player_info table, we have Player_info (player_id, name, birth_place, birthday, college, height_in, weight_lbs). We see that each player_id goes to exactly one name, birth place, birthday, college, height, and weight.
2. {player_id, year, curr_team} → {games_played, solo_tackles, assist_tackles, sacks, safeties, pass_defended, interceptions, interception_to_td, interception_yrds}
    a. From the defense table, we have (player_id, year, curr_team, games_played, solo_tackles, assist_tackles, sacks, safeties, pass_defended, interceptions, interception_to_td, interception_yrds) as our attributes. We see that every individual set of player_id, year, and curr_team corresponds to a specific set of defense stats for a specific player for that given year and team the player is on.

Normal Forms Used (with Justifications):

We used the BCNF normal form for our player, player_info, defense, passing, rushing, and receiving tables.

In the player and player_info table, all attributes depend on the player_id.

In the player table, we see that player_id is a superkey because each player_id goes to exactly one position, status, and experience for each player. In this table we decided to include information about a player related to football like his position on the team, if the player is retired or active, and how much experience or years in the league this player has. We have all of this information for each unique player_id. We kept this table smaller with less attributes to include the most important information when looking at a player in general, not looking at specific stats.

In the player_info table, we see that player_id is a superkey because each player_id goes to exactly one name, birth place, birthday, college, height, and weight for each player. In this table we decided to include more personal background information about the player, which is why it is separate from the player table. We have specific background information such as the birthday and college of every player for a unique player_id, which is not important in the player's football stats.

In the defense, passing, rushing, and receiving tables, all attributes depend on player_id, year, and curr_team.

We broke the positions of players into four tables. While there is some redundancy between the four tables for primary keys (player_id, year, and curr_team), we thought this was necessary for overall performance because players can play for different teams in a year (if they were traded to another team), so we could have multiple rows for one player in a given year but with different teams. If we made player_id, year, and curr_team separate we could potentially lose rows of data. This means we must include the primary keys in every table instead of taking them out separately.

In the defense table, we see that (player_id, year, curr_team) is a superkey because each set of player_id, year, and current team go to exactly one set of statistics for each player on that team for the given year including games_played, solo_tackles, assist_tackles, sacks, safeties, pass_defended, interceptions, interception_to_td, and interception_yrds. The defense table focuses on specific stats for defensive players.

In the passing table, we see that (player_id, year, curr_team) is a superkey because each set of player_id, year, and current team go to exactly one set of statistics for each player on that team for the given year including games_played, pass_attempt, pass_complete, td_passes, interceptions, passes_over_twenty, passes_over_forty, sacks, and passer_rating. The passing table focuses on specific stats for passers (typically quarterbacks).

In the rushing table, we see that (player_id, year, curr_team) is a superkey because each set of player_id, year, and current team go to exactly one set of statistics for each player on that team for the given year including games_played, rush_attempt, rush_yards, rush_tds, rush_first_down, rush_over_twenty, rush_over_forty, and rush_fumbles. The rushing table focuses on specific stats for players who run the ball (typically running backs).

In the receiving table, we see that (player_id, year, curr_team) is a superkey because each set of player_id, year, and current team go to exactly one set of statistics for each player on that team for the given year including games_played, receptions, receiving_yrds, receiving_tds, reception_over_twenty, reception_over_forty, reception_first_down, and receive_fumbles. The defense table focuses on specific stats for players who catch or receive the ball (typically wide receivers).

NF Proof 1: Proving normal form for player_info table.

We see that the normal form for the player_info table is BCNF because given (<u>player_id</u>, name, birth_place, birthday, college, height_in, weight_lbs), we have:

R(<u>player_id</u>, name, birth_place, birthday, college, height_in, weight_lbs)

R1 = (<u>player_id</u>, name, birth_place, birthday) where player_id is the primary key. We know that R1 is in BCNF because player_id is the primary key.

R2 = (R - {name, birth_place, birthday}) = (<u>player_id</u>, college, height_in, weight_lbs). We know that R2 is in BCNF because player_id is the primary key.

We see that this breakdown makes sense because it is practical to enforce that every unique player_id would map to an unique name, birth_place, and birthday for every player, while also enforcing that every unique player_id would also map to an unique college, height, and weight for every player. We see that these schema definitions are practical and efficient while also correctly enforcing the required dependencies.

NF Proof 2: Proving normal form for defense table.

We see that the normal form for the defense table is BCNF because given (player_id, year, curr_team, games_played, solo_tackles, assist_tackles, sacks, safeties, pass_defended, interceptions, interception_to_td, interception_yrds), we have:

R(player_id, year, curr_team, games_played, solo_tackles, assist_tackles, sacks, safeties, pass_defended, interceptions, interception_to_td, interception_yrds). We know that R1 is in BCNF because player_id, year, and curr_team are primary keys.

R1 = (player_id, year, curr_team, games_played, solo_tackles, assist_tackles, sacks, safeties)

R2 = (R - [games_played, solo_tackles, assist_tackles, sacks, safeties}) = (player_id, year, curr_team, pass_defended, interceptions, interception_to_td, interception_yrds). We know that R2 is in BCNF because player_id, year, and curr_team are primary keys.

We see that this breakdown makes sense because it is practical to enforce that every unique setof player_id, year, and curr_team would map to an unique number of games played, solo tackles, assisted tackles, sacks, and safeties, while also enforcing that every unique set of player_id, year, and curr_team would also map to an unique number of passes defended, interceptions, interceptions to touchdowns, and total interception yards for each player on a specific team during that year. We see that these schema definitions are practical and efficient while also correctly enforcing the required dependencies.

Part G. Relational Algebra

Requirements (from Final Project specification, Part G):

- Minimum of 3 non-trivial queries (e.g. no queries simply in the form **`SELECT <x> FROM <y>`**)
- At least 1 group by with aggregation
- At least 3 joins (across a minimum of 2 queries)
- At least 1 update, insert, and/or delete
    - This may be equivalent to said SQL statements elsewhere (e.g. queries or procedural code), but are not required to be; in other words, you can write these independent of other sections
- Appropriate projection/extended projection use
- Computed attributes should be renamed appropriately
- Part of your grade will come from overall demonstration of relational algebra in the context of your schemas; obviously minimal effort will be ineligible for full credit; it is difficult to formally define "obviously minimal", but refer to A1 and the midterm for examples of what we're looking for
- Above each query, briefly describe what it is computing; we will use this to grade for correctness based on what the query is supposed to compute; lack of descriptions will result in deductions, since we have no idea otherwise of what the query is intended to do.

Below, provide each of your RA queries following their respective description.

1) Get a list of all colleges and the number of defensive players that have attended the corresponding college.

player_colleges $\leftarrow {}_{\text{college}}G_{\text{count-distinct(player\_id) as nfl\_players}}(\Pi_{\text{player\_id, college}}(\text{defense} \bowtie \text{player\_info}))$

2) In football, a 2-way player is a player who plays on both offense and defense. Thus, these types of players have both offensive and defensive stats. Write a query that returns the names of NFL players that have both receiving statistics *and* defensive statistics.

two_way_players $\leftarrow \Pi_{\text{player\_id, name, year}}(\text{defense} \bowtie \text{receiving} \bowtie \text{player\_info})$

two_way_names $\leftarrow \Pi_{\text{name}}\ ({}_{\text{player\_id, name}}\ G\ _{\text{count(year) as seasons}}(\text{two\_way\_players}))$

3) We want to update the player table to only include players who are actively in the NFL, and the player_info table to include only players taller than 68 inches (under some imaginary height quota).

player $\leftarrow$ player - $\sigma_{\text{status = 'Retired'}}$ (player)

player_info $\leftarrow$ player_info - $\sigma_{\text{height\_in <= 68}}$ (player_info)

Part L1. Written Reflection Responses

Challenges and Limitations

List any problems (at least one) that came up in the design and implementation of your database/application (minimum 2-3 sentences)

Answer:
After receiving feedback on our proposal we had to think about how to organize our tables better and make the names of the attributes clearer. We also had some trouble properly implementing our UDF and Procedure that we had to work together to fix. We also had some trouble with the Python Command-Line Application, as this is something that neither of us have much experience doing prior.

Future work

If you are particularly eager for a certain application (have your own start-up in mind?), it is easy to over-scope a final project, especially one that isn't a term-long project. You can list any stretch goals you might have "if you had the time" which staff can help give feedback on prioritizing (2-3 sentences).

Answer:
One stretch goal could be to extend the application to creating a Fantasy Football league. This could include drafting players for teams and comparing them in different weeks to have a team win with the most collective points. Also, if we have enough time, we could add information about a player's success (how often they won the Super Bowl), since this is obviously correlated to how a player is perceived.

Collaboration (Required for Partners)

This section is required for projects which involved partner work. Each partner should include 2-3 sentences identifying the amount of time they spent working on the project, as well as their specific responsibilities and overall experience working with a partner in this project.

Partner 1 Name: **Halle Blend**
Partner 1 Responsibilities and Reflection:
Responsibilities: Changed table setup as specified in our proposal feedback (changed names, primary keys, and attributes), ER Diagrams, DDL, Functional Dependency Theory, load-data SQL file, MySQL Users and Permissions, Procedural SQL, Command-Line Python Application, Rest of Reflection.
Reflection: At least from my perspective, I thought that the time estimates were low for some parts. It took us longer to do some parts of the project because we didn't always implement what we are doing correctly the first time and ran into some errors, so I found myself taking longer than was

projected. I also had to go back and read through the lecture notes in some parts to double check my work, which took some extra time. We each checked what each other did and met to review it after we finished each part and then again at the end before we submitted the project. Overall I spent around 12 hours on the project. I worked really well with Eli. We split up some parts of the project but also worked on other parts together. We were able to problem solve some of our errors together and put together a great project.

Partner 2 Name: **Eli Seiner**
Partner 2 Responsibilities and Reflection:
Responsibilities: Changed some of the dataset as specified in our proposal feedback (made player_id only include the specific number and took out the name, added NULL values to the table where there was a lack of data from the dataset), Data Parsing, Password Management, Relational Algebra, SQL Queries, Procedural SQL, Command-Line Python Application, README. We each checked what each other did and met to review it after we finished each part and then again at the end before we submitted the project.
Reflection: I thought this project was a good chance to explore a project on our own, and it was good to be able to create a project for a personal portfolio. With that said, I think because we were trying to create such a grand project, some of the sections took significantly longer than expected, especially when we needed to correct or debug something that we had worked on earlier. In general, I also spent around 12 hours total working on this project, though a lot of it was debugging and data parsing. Because we were able to work in groups, and we met to discuss our progress, I was actually able to get a better understanding of some of the topics that were not as intuitive to me, like the ER diagram. Ultimately, I think I worked well with Halle, as we were able to work both efficiently and intelligently, making sure that both of us understood why we were making a particular design decision or implementation choice.

Other comments

This is the first time CS 121 has had a Final Project, and we would appreciate your feedback on whether you would recommend this in future terms, as well as what you found most helpful, and what you might find helpful to change.

Answer:
We think that the project was comprehensive of the material taught throughout the whole term. It was fun to be able to have our own project and application of what we learned this term. However, a big part of the project was the Python aspect, which hadn't been brought up much in term, so it was difficult to work through that and understand exactly how to implement that part.