CS 121 23wi Final Project Proposal

Due: Saturday Feb. 18th, 11:30PM

For the final project, you may choose to work with up to one other person. For partner-finding, use the #partner-search Discord channel to share your interests, any ideas, and preferences for working with a partner. Tuesday's lecture (02/14) will be optional worktime to find partners and ask about any questions with El around; we encourage you to attend to aim for finalizing a project idea, a partner, and any dataset/scoping questions you have. You are welcome to remove any suggestions/notes from this proposal in your proposal, as long as you're required components are clearly in the order specified (we've highlighted text for you to answer in blue for ease).

*Note: If working with another student, only one of you needs to submit the required files to CodePost, but **both of you should have equal contributions; i.e. you should ideally work through this document together.** For ease of grading please have the other student submit a simple* **collaboration.txt** *file to their CodePost submission in the following format (failure to do so may result in deductions):*

```
Names: Chun-Fu (Jeff) Chen, Joshua Flashner

CodePost emails: cchen8@caltech.edu, jflashne@caltech.edu
```

For full credit, we are looking for a robust proposal demonstrating careful consideration of the motivation, design process, and implementation plan for your application; you should expect to spend 3-4 hours minimum on your proposal (finding a dataset may take the longest). For each part of this proposal that requires your response, you should have 2-3 sentences (bullet points are encouraged as well) justifying your answers. We strongly encourage you to include as much detail as you can in this proposal (this is your only assignment this week) and you can include any diagrams and SQL brainstorming (e.g. a start to your DDL) with your submission on CodePost. If you want to add any additional planning, you can include a diagram as a PDF/PNG/etc. and/or .sql files, though not required. You can also include a starter Python program with function stubs for a command-line implementation you will complete for the Final Project (without SQL integration).

Summary of Files to Submit (max 3MB, reach out to El if you have trouble with this limit):

- **proposal.pdf (a copy of this document filled out)**
- ***(optional) diagrams/sketches/brainstorming not otherwise required; you can include these at the bottom of your proposal document.***
- ***(optional) app.*.py funtion stubs***

The advantage of adding these in your proposal is to get you started in advance, and also to get feedback from the staff on your design and implementation so far.

Student name(s): Chun-Fu (Jeff) Chen, Joshua Flashner

Student email(s): cchen8@caltech.edu, jflashne@caltech.edu

Database/Application Overview

In this proposal, you will be "pitching" your project, in which you have some freedom in choosing the domain of with a structured set of requirements that bring together the course material in a single project, from design to implementation to tuning. Keep in mind that unlike CS 121 assignments, you are free to publish/share your project, which can be useful for internship or job applications. In terms of scope, you should shoot for 8-12 hours on this project, though you are free to go above and beyond if you choose.

First, what type of database application are you planning on designing? Remember that the focus of this project is the database DDL and DML, but there will be **a small Python command-line application** component to motivate its use and give you practice applying everything this term in an interactive program; you can find a template from last year here, which may be adjusted slightly before the Final Project is released, but gives you an idea of the breakdown. Don't worry too much about implementation at this step, and you can jot down a few ideas if you have more than one. Just think about something you would like to build "if you had the data" which could be simulated with a command-line program in Python. Your command-line program will start with a main menu with usage options for different features in your application. Staff are here to help you with scoping!

Here's a list of some application ideas to get you started. These encompass most of the applications within the scope of this project we anticipate students might be interested in, but if there's a different application that meets the requirements for your DB schema and implementation, you are welcome to ask!

Application ideas ***(you can remove this listing in your proposal)****:*

- Store simulations
    - Can auto-generate datasets, or find ones online (e.g. bookstore, Starbucks dataset, video games, Animal Crossing store dataset, Pokemart, etc.); we're looking for consideration of schema design, relational model constraints, and the client vs. database separation of concerns, but your data can be simulated of course.
    - Online delivery simulation
    - Tables could include products, purchases, users, employees, carts, specials/promos, loyalty programs, etc.
- Gameplay/leaderboard management
    - Trivia or Jeopardy! dataset (there are some good ones online)
    - Wordle leaderboard, possible basic implementation of game or some offshoot like Redactl (Kaggle dataset available)
    - Random Pokemon collecting/Pokedex management
        - Pokemon (Pokedex vs. collection), moves, types, collection, Pokemart (we have some datasets from previous projects)

    - Players, game entities, categories, etc. for a game you implement or a public game score database (22wi had a variety of sports-related projects; just be aware that many of these have redundant columns for stats, such as averages, which can be cleaned up similar to lecture's AirBNB demo)
- Budgeting/utility application
    - User configuration of income/expenses
    - Set monthly categories, goals, limits
    - Add some gamification/achievements
    - Tables may be harder for this one, unless you get particularly creative
- Messageboard, blog, or feed
    - Tables can include messages, users, admins, entry logs, event calendar, etc.
- Application system (e.g. applications for jobs, student clubs, research, adoption agency, course registration, etc.)
- Some creative idea with social media (most platforms have datasets published)
- Office Hour/other type of Queue simulator
- Stock market simulator
- Client applications for interacting with research data, climate data, etc.

Proposed Database and Application Program Answer (3-4 sentences) :
Pokemon Battle Simulator (excluding/including ability, item, etc)

- ***Create team with up to six pokemon***
- ***Design custom moveset***
- ***Swap between available pokemon***
- ***Calculate damage and apply effects (poison/badly poison, paralyze, sleep, burn, frozen,± (sp)atk/(sp)def/spd/acc/dodge)***
- ***Local multiplayer on same keyboard***
- ***I'm not a big fan of mega evolve and dynamax, so I'll probably just support pokemon til gen 4***

Next, where will you be getting your data? We will post a list of datasets to get you started, but you can find many here and here (look for ones in CSV file(s), which you will break into schemas similar to the Spotify assignment; we can also help students convert JSON to CSV if needed). You are also welcome to auto-generate your own datasets (e.g. with a Python script) and staff are more than happy to help you with the dataset-finding process, which can take longer than the rest of the starting design process.

Data set (general or specific) Answer:
Pokemon species strength: ***Pokemon with stats | Kaggle***
Pokemon type chart: Type - Bulbapedia, the community-driven Pokémon encyclopedia (bulbagarden.net)

Pokemon Nature: Nature - Bulbapedia, the community-driven Pokémon encyclopedia (bulbagarden.net)

Pokemon move pool: ***https://pokemondb.net/move/generation/3***

(Need to convert the text description into db)

Who is the intended user base and what is their role? You will need to have at least one client usertype, and one admin. These different users may have different permissions granted for tables or procedural SQL. Consider clients as having the ability to search ("query") data from the command line, submit requests to insert/update data, etc. In 22wi, some students had a separate client .py file and admin .py file, with admins having admin-related features, such as approving requests, inserting/updating/deleting information, querying, etc.

***Client user(s) Answer*:**

Client users are people who want to build teams and simulate battles.

Create an account with a username and password.

Build/Update teams, select pokemon/moves/nature/EVs

IVs will be randomly initialized by default but users can choose to set IVs for simulation.

Enter a battle

Admin user(s) Answer:

(actually I don't think this part is really necessary, I'd rather spend time on the other part.)

Build recommended teams

Insert new moves (and update "Knows" table)

Add new pokemon

Required Features

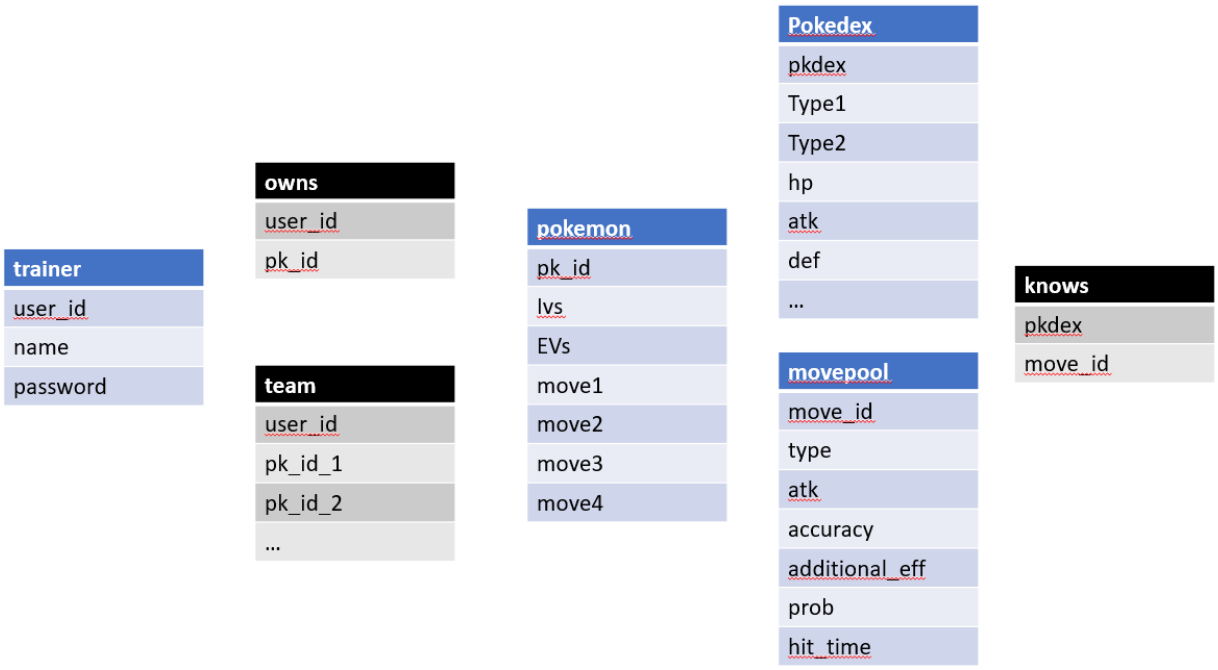
The full specification of the project will outline the requirements of the project, but you remember you should aim to spend 8-12 hours (it replaces the final exam and A8), depending on how far you want to go with it. The time spent on finding or creating a dataset will vary the most. For the proposal, you will need to brainstorm the following (you can change your decisions later if needed).

E-R Model: There will be an ER model component, but it is not required for the proposal.

DDL: What are possible tables you would have for your database? Remember to think about your application (e.g. command-line functions for user prompts to select, add, update, and delete data, either as a client or an admin user). To make this step easier, thinking about the menu options your application provides, as well as the queries you might want to support, is helpful to narrow down the information you'll want to model. At minimum, you must provide a general listing of attributes

and possible types; you can provide these on a high-level (e.g. bullet points) or actual DDL with CREATE TABLE statements in an optional **`setup_proposal.sql`** file attached in your submission. The more you provide, the more we can help provide feedback. Include at least 4 tables in your response, as well as any design decisions you may run into in your schema breakdown.

Answer: (see sql for more detail)



Queries: Part of the application will involve querying data in a way that would be useful for a client (e.g. searching for "Puzzle" games made in the last 5 years, ordered by year and price in a Video Game Store database, or finding all applicants who are pending for an adoption agency). Identify at least 3 queries that would make sense in a simple command-line application. In your answers, provide a brief description of the query or pseudocode, as well as the purpose in your application. These will likely be implemented as SQL queries within Python functions, wrapping your SQL queries (the methods of which will be taught in class). You are welcome (and encouraged) to add more, though not required.

Answers:

1. Show PKMs based on user's filter (used when building a team)

Select pkdex, pkm_name FROM pokedex

WHERE …

2. Show all moves that a pkm can learn (used when building a team)

SELECT move_name FROM movepool, know

WHERE pkdex=.....

3. Show pkm and opponent pkm info (hp, status, moves) when in a battle

SELECT status, hp, atk (-6~+6), …, mv_id_1, mv_id_2, … FROM pokemon, fighting_status

WHERE pkm_id=.....

For opponent, move id won't be shown (you don't know opponent's move)

Procedural SQL: You will need to implement at least 1 each of a UDF, procedure, and trigger in your project (we haven't seen triggers yet, so you don't need to provide one in your proposal). Otherwise, identify at least one UDF and one procedure here. For each:

1. What is the name of the function/procedure?
2. Why is it motivated in your database? Consider the differences discussed in Lecture 10 for UDFs, procedures, queries, temporary tables, and views. In your Final Project, we'll be looking for you to demonstrate an understanding of appropriate design decisions here (and we're happy to help discuss any trade-offs)

Consider some examples in class/HW, such as logging DML queries, adding an extra layer of constraint-handling (e.g. the overdraft example from lecture), etc. For procedures, remember that these can be called in an application program written in a language like Python or Node.js, so these are especially useful to avoid writing queries in such application programs, *especially* SQL that performs **`INSERT/DELETE/UPDATE`** which you do not want to leave to an application user. Remember that you can also set permissions for different users to access tables and procedures. In your Python program, you can connect to your database as different users defined in your database (similar to the Node.js program El demo'd a while back).

Answers

UDF(s):

1. **Calculate damage**

input : pkm_id, opp_id, move_id

Output: damage

Factor = same_type (1.5 or 1) x effective (0, 0.25, 0.5, 1, 2, 4) x BURN x random_factor (0.85~1)

Damage = [(2*lv + 10)/250 x atk/def x move_power + 2] x factor

If damage<1.0: damage = 1

Else: damage = floor(damage)

2. **Compare speed**

Input: pkm_id_0, pkm_id_1, move_id_0, move_id_1:

output : tiny_int (0 or 1)

If move_id_0.priority > move_id_1.priority: return 0

Elif move_id_0.priority < move_id_1.priority: return 1

ELSE: compare speed of pkm (considering paralysis)

3. **Is_hit (considering accuracy rate), reference:** 命中判定 - 神奇寶貝百科，關於寶可夢的百科全書 **(52poke.com)**

Input: pkm_id, opp_id, move_id:

Output tinyint (0 is miss, 1 is hit)

Factor = pkm.acc - opp.evasion (-12~+12)

If factor>6: factor=6

Elif factor<-6: factor = -6

If factor>=0:

    Factor = (3+factor)/3

Else:

    Factor = 3/(3+factor)

A = floor(255*move.acc) * Factor /255

B = RAND(0,99)

If B<A: return 1 (hit)

Else: return 0 (miss)

Procedure(s):

1. **Register / Create team (select pkm, select moves)**

2. **Use a move (update hp and status of a pkm in battle)**

   If Freeze/sleeping/paralysis: can't use move with some probability

   elif miss: do nothing

   Elif protected: do nothing

   else:

      Hp -= damage()

      If status is NULL: update status with probability based on move_id

      If special status is NULL: Confusion with certain probability

3. **Handle one turn (More details are available here: Battle Mechanics | The Cave of Dragonflies (dragonflycave.com))**

   Check run/surrender

   Check switch

   Handle sleep/freeze (wake up / unfroze)

   Compare speed

   Use_move_1

   Use_move_2

   Handle burn/poison

~~**Trigger(s).**~~ ***(not required for 23wi proposal)***

~~**Database Performance**~~**:** ***(not required for 23wi proposal, but left in for a preview of the corresponding Final Project component)*** *At this point, you should have a rough feel for the shape and use of your queries and schemas. In the final project, you will need to add at least one index and show that it makes a performance benefit for some query(s). You don't need to identify what index(es) you choose right now (that comes with tuning) but you will need to briefly describe how and when you would go about choosing an index(es). Refer to the material on indexes if needed.*

~~**Performance Tuning Brainstorming:**~~

"Stretch goals"

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.

Answer:

1. Pokemon's Ability

My favorite pkm is Lanturn and the main reason is because of its super useful ability ***Volt Absorb*** (In gen3, the AI always use electric type move against *Lanturn because Lanturn is also water type*), it would be so cool to implement this part.

2. Some moves that hard to fit in the movepool Table:

One-Hit-Ko, self-demage move (Self-Destruct, Jump Kick), self-heal, field, mist, etc.

3. Items

Potential roadblocks

List any problems (at least one) you think could come up in the design and implementation of your database/application.

Answer:

- ***How to handle the battle mechanics of a turn (Keeping track of status effects)***
- ***Keeping track of health of pokemon when swapping.***
- ***Resetting pokemon status when swapping.***

Collaboration

For projects with partners, this section is required (if not, you can leave it out, or note that you are decided yet). Both students should provide a brief summary of their planned workload distribution, method(s) of collaboration, and 1-2 points you are most interested to in the project. You may also clarify any concerns/confidence for your partner work here. Feel free to provide a paragraph or bullet points; we're looking for you to have thought this out and discussed your plan for collaboration at this step.

[Chun-Fu (Jeff) Chen] Answer:

- ***Create tables/schema. Set types and constraints for each column***
- ***Calculate damage***
- ***Use a move, deal damage and apply effects to itself/opponent***
- ***Handle one turn (Battle Mechanics)***
- ***Python code for battle phase UI***
- ***Admin user to add new pkm/moves (UI)***

- ***interested in building pkm game by myself and learn more about pkm game mechanics (like how damage calculation works)***

[Joshua Flashner] Answer:

- ***Populate and clean database with online data.***
  - ***Possibly also with the database that El mentioned they had.***
- ***Python code for the UI to create an account and start battles.***
- ***Register/Create team (Python interface and SQL commands)***
- ***Compare Speed function to calculate which pokemon attacks first.***
- ***Is_hit function to calculate whether a move hits or not.***
- ***Python code for tracking health and status effects.***
- ***I'm most interested in learning about how we can implement SQL functionality in python.***

Our preferred method of collaboration is Discord for writing code and planning.

Open Questions

Is there something you would like to learn how to implement in lecture? Any other questions or concerns? Is there anything the course staff can do to help accommodate these concerns?

Answer:
How to handle user passwords in SQL.

Have fun!

Optional Brainstorming/Sketches/Other Notes

This is an optional section you can provide any other brainstorming notes here that may help in the design phase of your database project (you may find it helpful to refer to the early design phase in your Final Project Reflection).
