To begin working with the anther's ladder database dump, first acquire the sql dump file from the #datamine channel (here saved as dumpfile.sql), then import it to mysql. If the dump begins with something like:

```
CREATE DATABASE `anthers_02_02_2016`
USE `anthers_02_02_2016`
```

Then you are good to simply import it directly like

```
$ mysql < dumpfile.sql
```

If there are no CREATE or USE directives, you will need to create an empty database first, then import it to that database:

```
$ mysql
> CREATE DATABASE anthers_02_02_2016;
> USE anthers_02_02_2016;
> SOURCE dumpfile.sql;
> quit
```

Alternately, after creating the database, you can import it from the command line with:

```
$ mysql anthers_02_02_2016 < dumpfile.sql
```

After initially creating the database, you may note that the stages table is missing. Get an export of that table and save it as stagedb.txt. Import it to your database with:

```
$ mysql anthers_02_02_2016 < stagedb.txt
```

1 Tables Overview

Note that I will describe foreign key - primary key type relations between tables, but those constraints will not have been actually programmed in to the database. Nevertheless, they can be used as such.

In this section, I'll describe the contents of each table, starting from most human readable, establishing relations as we go. Not every field will be listed below, only those most relevant to the types of analyses we will undertake.

1.1 games

This table's primary purpose is to map games.name as strings (melee, wii u, 3ds) to games.id as integers (2, 4, 3 respectively).

1.2 game_rooms

This table is generally not necessary for our analyses. We can already get game to id mappings from the games table. This does have some convenience fields like active_season, but it can generally be disregarded.

1.3 game_seasons

Provides a mapping from titles (e.g. melee or wii u) to a unique season ID number.
game_seasons.ladder_id is a foreign key to games.id.

1.4 characters

Provides a mapping from character names as strings (characters.name) to an integer id (characters.id). The ID is unique to a character/game combo.
characters.game_id is a foreign key to games.id.

1.5 stages

Provides a mapping from stage names as strings (stages.name) to an integer id (stages.id). The ID is unique to a stage/game combo.
stages.game_id is a foreign key to games.id.

1.6 season_characters

Provides a way to track which characters were permitted for use in which season. Useful to track things like DLC release. Can imagine using down the line to track usage shares corrected by number of available characters.
season_characters.season_id foreign key to game_seasons.id and season_characters.character_id foreign key to characters.id.
season_characters.active only takes values of 0 (not permitted) or 1 (permitted).

1.7 season_stages

Provides a way to track which stages were permitted and what their status was in each season.
season_stages.season_id foreign key to game_seasons.id and season_stages.stage_id foreign key to stages.id.
season_stages.stage_type takes on values of 0 (banned), 1 (starter), 2 (counterpick), and 3 (dlc [just dreamland on wii u and 3ds - probably just to check if user has it available]).

1.8 player_ladder_stats

This gives us information on the performance of each player, most notably their rating, which is the best variable to represent player skill (check in the #datamine chat to see how the ratings translate to classes).
player_id doesn't strictly have a foreign key but it points to numerous other user identifiers in later tables in pretty intuitive ways.
player_ladder_stats.ladder_id is a foreign key to games.id, and player_ladder_stats.season_id is a foreign key to game_seasons.id.

Table 1: Summary of fields in ladder_matches table

Field	Foreign Key?	Description
id	pointed to by others	new unique ID for each set
search_user_id	same player_id as elsewhere	unique ID for player
reply_user_id	same player_id as elsewhere	unique ID for player
accepted	no	0/1
rejected	no	0/1
match_count	no	NULL/0/1/3/5
results_finalized	no	0/1/2/3/4
type	no	1/2
ladder_id	games.id	which smash title?
season_id	game_seasons.id	game/time combo

1.9 player_ladder_stat_characters

Gives season summary information on character usage. For each season, it breaks down each user's wins and losses by character.
player_ladder_stat_characters.player_id maps to player_ladder_stats.player_id.
player_ladder_stat_characters.season_id is a foreign key to game_seasons.id, and
player_ladder_stat_characters.character_id is a foreign key to characters.id.

1.10 ladder_matches

Now we're getting into the real meat of the database. On the most basic level, it assigns a unique id to each set which you can assign games to. It stores information about the participants, the game type, season, and status on the outcome of the game itself. The information is summarized in table 1.

My best guesses for the cryptically coded fields are:

- match_count: 3 and 5 are best of 3 and 5 respectively (regular ladder matches), 1 is a rarely used option only used for one day on PM and Brawl ladders, so it doesn't concern me. When it was null, all results_finalized = 4 and all type = 1, and no game records exist. It seems like 0 means friendlies (unlimited match count) also don't have game records (characters used/stages), these have mixed results_finalized with all type = 1.
- results_finalized: 1 = p1 win, 2 = p2 win, 3 = set cancelled, 4 = endless friendlies ended
- type: 1 = unranked, 2 = ranked

In this dump, player/team 1 is always the search user, and player/team 2 is always the reply user. This way change in the future.

1.11 match_games

Similar to the previous table, this table's basic function is to assign a unique ID to each game played (game here, meaning game-set terminology) and to track information about

Table 2: Summary of fields in match_games table

Field	Foreign Key?	Description
id	no	new unique ID for each game
match_id	ladder_matches.id	unique ID for set
game_number	no	1/2/3/4/5
stage_pick	stages.id	ID for stage
search_user_character	characters.id	ID for P1 char
reply_user_character	characters.id	ID for P2 char
search_user_result	no	NULL/1/2/3/5
reply_user_result	no	NULL/1/2/3/5
final_result	no	NULL/1/2/3

each game (characters used, stage selected etc). We summarize the information in table 2.

Some more detail on the fields:

- game_number: just tracks place in BO3 of BO5. intuitive.
- search_user_result: from this user's perspective did they win (2) or lose (1). others tend to lack a final result, indicating that the set wasn't played to completion.
- reply_user_result: same as search user, but from reply user's perspective
- final_result: 1 = team 1 (search user) won, 2 = team 2 (reply user) won, 3 is disputed

In general, for the previous two tables, we'll probably want to filter on "well-behaved" sets, so where ladder_matches = 1 or 2, and match_games.final_result = 1 or 2.

1.12 match_players

This table will become more important in future dumps as the search user and reply user will not always be assigned as 1 and 2, and you will need to check this table to find their team assignments. The change field also serves as a convenient check to verify who the winner or loser of a set was, based on their rating change. A result of 1 indicates a loss (negative rating change) and a result of 2 indicates a win (positive rating change).
match_players.match_id is a foreign key to ladder_matches.id

1.13 match_point_change_log

Still a bit unclear on what this is. In many cases, there is only one entry for a match_id, so it feels like it cannot be tracking ratings for each participant, and player_ladder_stat_id doesn't seem to map to any other player IDs directly.

1.14 game_stage_strikes

This is exactly what you think it is. game_id is a foreign key to match_games.id, user_id is the player_id from elsewhere, and stage_id is a foreign key to stages.id.

2 Graphical representation

In figure 1 I attempt to map out some of the more important relations.

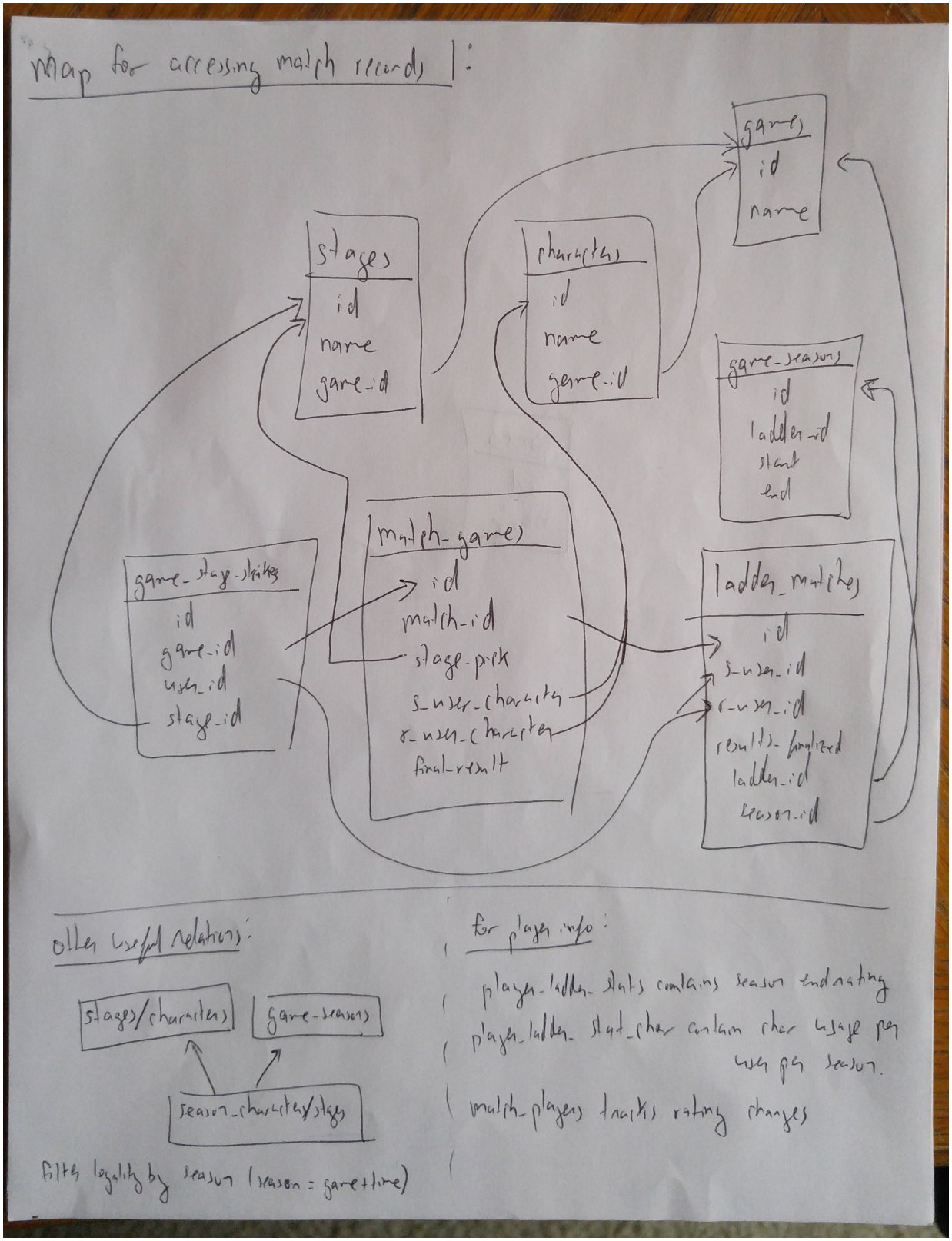

Figure 1: Some of the important relations to query game info from the anther's ladder db