Division System Concept

Division		Intended Trueskill Rating (μ-3sigma)	Point maximum	Division index	Subdivision Index
Grandmaster		2000+	100	6	1
Master	I	1915+	10	5	5
	II	1830+	10	5	4
	III	1745+	10	5	3
	IV	1660+	10	5	2
	V	1575+	10	5	1
Diamond	I	1490+	10	4	5
	II	1405+	10	4	4
	III	1320+	10	4	3
	IV	1235+	10	4	2
	V	1150+	10	4	1
Gold	I	1065+	10	3	5
	II	980+	10	3	4
	III	895+	10	3	3
	IV	810+	10	3	2
	V	725+	10	3	1
Silver	I	640+	10	2	5
	II	555+	10	2	4
	III	470+	10	2	3
	IV	385+	10	2	2
	V	300+	10	2	1
Bronze	I	215+	10	1	5
	II	130+	10	1	4
	III	45+	10	1	3
	IV	-40+	10	1	2
	V	-40 and lower	10	1	1

Goals/Requirements:
create incentive to play
make trueskill rating less visible (people try to get highest rating possible, but that is not the purpose of trueskill rating)
easy to understand rules
no influence to trueskill rating
resetable to sort out inactive players

Challenges:
progress in the divisions wanted to motivate players
most players have established trueskill already -> winrate close to 50%
system should also be working for new players with undetermined rating

Features/how it works:
matchmaking still based on trueskill for fair games
trueskill is still calculated in the background as normal
subdivisions behave like divisions, this is just cosmetic

win: gain 1 point
loss: lose 1 point
draw: no point change
rank up to next (sub)division if reaching point minimum of next division
start with 2 score in the next division to prevent instant derank
derank if falling under point minimum of your division
start with 2 points under division point maximum to prevent instant uprank

placement phase (10 games) before placing the player in a division (gives trueskill some time to determine rating for new players)
starting points based on relative position in division (many if close to rank ceiling, less if closer to rank floor of division) e.g. 8 rating ~ 1 point
place the player based on trueskill rating, but 100 points lower than intended rating

while player rating is higher than intended rating in his division give him a boost:
one more point for winning than for losing (2/-1 on win/loss is preferred over 1/0 because that could make the players believe the game wasn't counted)
-> with 50% winrate a player needs 18 games to climb one division

remove the boost, when the player reaches the division for his rating
this will probably mean that the player doesn't climb anymore unless he improves his skill
-> players' divisions settle around the intended rating, preventing division inflation

a negative boost can be applied should a player rank up too much (1/-2)

seasons last 3 months

highest division:
have boost only end once player has points that equal his rating to help high rated players float to the top

Advantages:

- rewards possible based on division
- negative trueskill rating in the first games of bad/unlucky players is not visible (possible demotivation prevented)
- placement phase makes players tolerate unbalanced games in the beginning more
- harder to derank in the beginning (because boost drags you up)
- first upranks almost guaranteed -> incentive to play
- satisfies players desire to improve displayed „skill“
- high skill players have to play more games to show up high in the leaderboard

Database Format

A `league` is simply a common name for several `league_seasons`. A `league_season` will subscribe to a rating type and will then update players' scores with every game of that rating type, while the season is active.
Hence `league_season`s reference their parent `league`'s id, a `leaderboard` id, and a start date, as well as an end date.
The scores in a league season are split into divisions, which are themselves split into subdivisions.
Since a league might change its division layout (count of divisions, rating ranges of subdivisions, etc.) between seasons, every `league_season_division` is associated with a fixed season.
A `league_season_division` stores the id of the season it belongs to (which in turn stores the id of the league it belongs to), as well as a `division_index` used to order divisions inside a season from lowest to highest by ascending index. It's encouraged to use 1, 2, 3, ... as indices, although that is not enforced server-side.
The actual configuration is stored in the individual `league_season_division_subdivision`s. They store the id of the parent `league_season_division` (which in turn stores the id of its season, which in turn stores the id of its league. So to find out which league a subdivision belongs to you need to join four tables.) Like divisions they store a `subdivision_index` used to orderthem from lowest to highest in ascending order of index inside their division. 1, 2, 3, ... is encouraged. Additionally, subdivisions store a `min_rating` and `max_rating` (both mandatory) to specify the intended rating ranges for players placed in this subdivision, used for "boosting" decisions. The guaranteed boundaries allow a more fine-tuned placement of the players at the start of a season. This is especially important for the highest division because there will be players with a wide range of ratings. To properly calculate the later boosting it is important that the division ratings cover the whole player rating range. Hence the overall lowest/highest subdivision should have appropriately low/high values entered as min/max rating, currently -1000 and +3000. Each subdivision in addition stores a `highest_score`, e.g. 10 indicating that points within the division range from 0 to

`highest_score`. If a player would go higher than `highest_score` points they should hence be promoted to the next higher subdivision, if the fall lower than 0 they should be demoted.

A player's rating is saved as `league_season_score`, associated to a `login_id` and a `league_season_id`. Hence players will get a fresh rating object for every season and the latest rating of the previous season persists. A history gets stored in `league_score_journal`.
The `league_season_score` containes the id of the `league_season_division_subdivison` that the player is currently placed in, as well as the `score` (which should be between 0 and the `highest_score` of that subdivision) that they currently hold inside that subdivision.
In addition we store a `game_count` to count the games played during this `league_season`, used to defer placement until an appropriate amount of games has been played.

All of `league`, `league_season`, `league_season_division`, and `league_season_division_subdivision` in addition store a `name_key` and a `description_key` used for internationalization.
For example, the `name_key` of a division should translate to "Bronze" and the `name_key` of a subdivision should translate to the suffix to identify the subdivision inside the division, e.g. "III".
Description keys are mostly a convenience feature for the client, giving a place for additional info. This might translate to a description of rating type and start/end dates for league seasons, or some full name like "Ladder League Bronze Division III" for divisions, should this be needed.

TODO: The `league_season_id` of `league_season_score` is not a foreign key. Why?

Full description of the database tables is here:
https://faforever.github.io/faf-league-service/relationships.html