TheFoodCourt

Software Documentation

Database Systems Course (2018-2019) – Final Project

Ram Shimon (203537246)

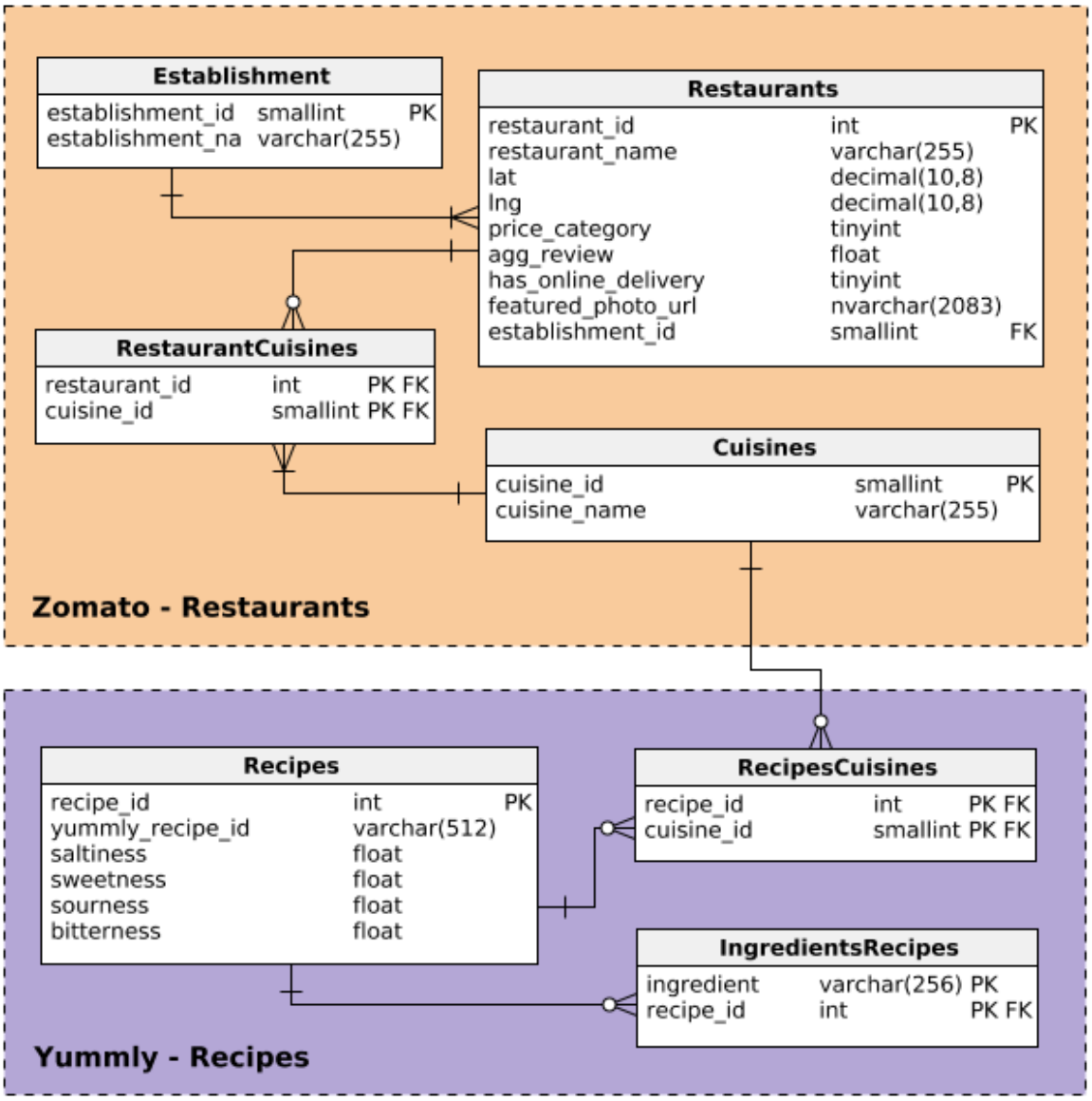
Yuval Weiss (204330740)

Nitzan Yogev (312433279)

Michael Khaitov (308401082)

DB Scheme structure

DB Scheme:



For more details, see `CREATE-DB-SCRIPTS.sql`.

Number of rows per table:

Table Name	Establishments	Restaurants	RestaurantsCuisines	Cuisines	Recipes	RecipesCuisines	IngredientsRecipes
# rows	34	5,039	9,274	138	18,734	21,042	164,127

TODO: ADD DB CHOICES

DB Optimizations Performed

Indexes:

When designing the table and indexes to use, we used one main assumption in our reasoning:

All online queries (queries performed by clients) are `SELECT` queries, and not `INSERT` or `UPDATE` queries.

This key assumption was crucial to our index choice. Since each added index increases insert and update time (due to modification needed after the query to update the index), the assumption above enabled us to add indexes without impacting online performance in terms of query time. We did keep in mind that still, indexes needed to be loaded into memory, thus we still needed to pick our indexes intelligently.

The indexes mentioned below are additional indexes to those created automatically for primary keys. Those are not mentioned here.

- **`flavor_index`:** the index improves query performance for the `restaurant_by_taste` query (see next chapter). The index is composed of four columns – the exact ones used in the query. Since the query searches up ranges of flavors (e.g. `sweetness BETWEEN X AND Y`) we used a BTREE index.
- **`restaurant_location_index`:** the index improves query performance for all restaurant queries using location filtering. We used a normal BTREE index (and not SPATIAL), since our location filtering for restaurant is simple – a square search around a given location, only needing range querying for the `lat,lng` columns, and not complex geometric queries.
- **`restaurant_review_index`:** the simple index is used to optimize restaurant filtering by user's review averages (the `agg_review` column). Since the query is using comparisons (e.g. `greater than` we used a BTREE for this index as well.
- **`restaurant_price_index`:** similar to the above, but for the `price_category` column