Database Scheme structure:

Database optimization performed:

Description of query:

what does it do, how you optimized the query and how the DB design supports this query

1. Best production company by genre
2. Best production company by language

What does the query do:

The user inputs genre/language name with the judging criteria: Movie rating ,vote count and difference between the movie revenue and budget. The query returns the best production company satisfying the category (genre, language) and criteria.

How we optimized it:

Use of indices in each of the relevant tables, clustering the data and achieving higher selection times. The data is joined from different tables, the selection is done in each table so that the join is performed on smaller relations.

How the database design support these queries:

Take query #1 (second one is of similar design):

Data is retrieved from genres and movies tables, the selection is done on each of these tables. Afterwords through moviegenres the data is joined from each side via movie\_id and genre\_id (this is where the indices optimize the query).

1. In Which movies did a given actor voice acted in

This is a full text query.

What does the query do:

How we optimized it:

How the database design support these queries:

1. In which movies did a given person have a particular job (Actor, director, camera handler…)

API description:

The Movie Database: we iteratively made calls to the API and got data about movies via their unique ID. This data was split into 5 main tables: movies, genres, production\_companies, spoken\_languages and production\_countries. with additional connection data put into movieprodcomp, moviegenres, movielanguages and movieprodcountry.

Using the unique movie\_id we made additional calls to get the actors and crew members or each movie, which were inserted into actors and crew, with the connection to movies table being made via movieactors and moviecrew.

Flow of the application: