Assignment 2

Assignment 2 focuses on relational algebra and SQL. For the SQL part, you will create a database to host data from Reddit. You can design the database however you like, with as few or as many tables you find necessary. Your main objective is to balance ease of implementation and performance; the database should perform as well as possible without sacrificing how understandable your design is.

You can use whatever relational database management system you prefer to solve the assignment. While it is technically not necessary to use any programming language, I recommend that you automate as much as you can, such as setting up tables, importing data, etc. You can use whatever programming language(s) you like for this task.

You are free to make additional assumptions, if you feel that some information is missing. Make sure to document all assumptions that you make.

All answers should be your own. You are allowed to work in groups of two. Make sure you include your names in the report when you submit.

Task 1: Relational algebra

Suppose relations *R* and *S* have *n* and *m* tuples, respectively. Give the minimum and maximum numbers of tuples that the results of the following expressions can have:

1. $R \cup S$
2. $R \bowtie S$
3. $\sigma_C(R) \times S$
4. $\pi_L(R) \setminus (S)$ (set difference)

Task 2: Normalization

For each of the following relational schemas and sets of functional dependencies:

1. $R(A, B, C, D, E)$ with functional dependencies $AB \to C$, $DE \to C$, and $B \to D$.
2. $R(A, B, C, D, E)$ with functional dependencies $AB \to C$, $C \to D$, $D \to B$, and $D \to E$.

do the following:

1. Indicate all the BCNF violations. Do not forget to consider dependencies that are not in the given set but follow from them.
2. Decompose the relations, as necessary, into collections of relations that are in BCNF.
3. Indicate all 3NF violations.
4. Decompose the relations, as necessary, into collections of relations that are in 3NF.

Task 3: Setting up the Reddit database

Reddit Data

The data from Reddit is distributed as Bzip2-compressed JSON records. Each file contains a single month of Reddit posts.

A JSON record is a set of key-value pairs, where the key is a string and the value can be of different types. For example, the key-value pair `"author":"adbac36"` has key `author` and the value is a text string that represents the author's username (`adbac36`). The Reddit JSON records are structured as follows:

```
{
    "parent_id":"t1_co779e1","id":"co77k46",
    "score_hidden":false, "gilded":0,
    "author_flair_text":null,"author_flair_css_class":null,
    "name":"t1_co77k46","body":"will keep you posted!",
    "created_utc":"1422748973","downs":0,
    "ups":-1,"distinguished":null,
    "controversiality":0,"link_id":"t3_2ucrbf",
    "subreddit_id":"t5_2uqch","score":-1,
    "archived":false,"subreddit":"AskNYC",
    "edited":false,"retrieved_on":1424281729,
    "author":"adbac36"
}
```

The following keys contain information that you will need to store in your database.

Key	Description
`id`	The id of a comment. The id is an integer value encoded using base36
`parent_id`	The id of the thing this comment is a reply to, either the link or a comment in it
`link_id`	The id of the link this comment is a reply to, can be same as `parent_id`
`name`	The name is a combination of a type prefix and the `id` of a post. The prefix `t1` indicates that it is a comment. You can find a full list of type prefixes at https://www.reddit.com/dev/api
`author`	The posters name
`body`	The comment's contents
`subreddit_id`	The id of the subreddit in base36
`subreddit`	The name of the subreddit
`score`	The combination of up and down votes. Note that `ups` and `downs` are not reliable in this dataset, `score` is often (always?) the same as `ups`
`created_utc`	When the comment was posted (UTC epoch-second format)

You can safely ignore any keys not mentioned in the table (but you are of course allowed to include and use them, if you find them useful). You can find a complete description of all keys at https://github.com/reddit/reddit/wiki/JSON.

There are three data files. You can use any combination of the files to create your database as long as you include `RC_2011-07.bz2` or `RC_2012-12.bz2`. Use `RC_2007-10.bz2` to test your database and queries, and then import either of the other two (or both, or all three) to run the queries to check performance.

File	Size
RC_2007–10.bz2	13MB
RC_2011–07.bz2	902MB
RC_2012–12.bz2	2GB

Provide an E/R diagram for your design as well as schemas with types.

Task 4: Importing data

Use the program or scripts that you created in the previous task to experiment with the cost of constraints (such as keys). Create two versions of your schema, one without any constraints (other than types) and one with "perfect" contains (e.g., not null, unique, primary keys, etc.). How do these constraints affect the import time. Measure and experiment with turning these on and off. Report measured times and an discuss why you think the constraints affected the times.

Would it be reasonable to import and turn on constraints after? When?

Task 5: Queries

Your database should be able to answer the following queries. Note that they are in no particular order. "Specific user" means that you pick one user, either manually or on random. The query should be easy to adapt to any user.

1. How many comments have a specific user posted?
2. How many comments does a specific subreddit get per day?
3. How many comments include the word 'lol'?
4. Users that commented on a specific link has also posted to which subreddits?
5. Which users have the highest and lowest combined scores? (combined as the sum of all scores)
6. Which subreddits have the highest and lowest scored comments?
7. Given a specific user, list all the users he or she has potentially interacted with (i.e., everyone who as commented on a link that the specific user has commented on).
8. Which users has only posted to a single subreddit?

Optionally, answer Which subreddits share no users, i.e., have no users that have posted to the others.

Create any indexes you think might help (and verify that they do!) and experiment with various SQL formulations. Report the queries you found to work best, together with a brief motivation for why you think that worked best. Optionally, report the difference in runtime between a naive query (i.e., simplest possible with no extra indexes) and your optimized query.

Submission and deadlines

Your submission should include solutions to all assignments above.

Submit a report in PDF format on Moodle. You can draw the E/R diagrams by hand and submit scanned versions or photos (as long as they are readable). If you want to include source code, you can, but please submit those files together with the report in a zip archive. Do *NOT* submit any database dumps; these will be far too large for Moodle to handle.

If you work together with someone, submit the assignment from one of your accounts and make sure to put both names in the report!

Deadline: End of day 2018–12–10.