CS839 NoSQL Systems

Assginment 1

30 January 2023

Submitted By

Group 19

1. IMT2020502 - Monjoy Narayan Choudhury
2. IMT2020509 - Hardik Khandelwal
3. IMT2020548 - Tejas Sharma

Files Submitted

1. Report (this document)
2. Scripts in .sql format:
   (a) create_with_constraint.sql: Contains necessary DDL scripts with primary and foreign key constriants.
   (b) create_without_constraint.sql: Contains necessary DDL scripts without primary and foreign key constriants.
   (c) populate.sql which contains code to populate data (using COPY) from TSV to database.
   (d) queries.sql which contains solution to the problems asked in part 2.

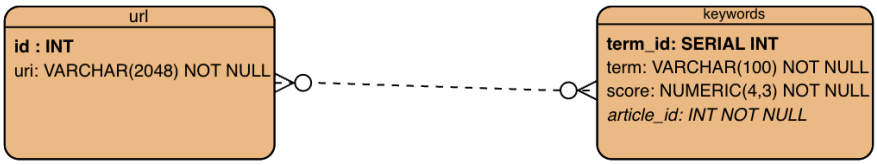
Bulkloading Data into a PostgreSQL Database

1

To create a new database instance we need to use the following query:-

```
CREATE DATABASE "wiki"
WITH OWNER "postgres"
ENCODING 'UTF8';
```

ENCODING option ensures that the database can store UTF-8 encoding, allowing us to preserve the encoding from the TSV files to the database. After this we need to use the databases. Postgres psql allows us to access it using the \c command. We have named our database "wiki" hence we will connect to it using the command \c wiki.
The schema followed by us can be understood with the following ER diagram:



The corresponding SQL code to create the same is as follows:

```
CREATE TABLE url(
id INT NOT NULL UNIQUE PRIMARY KEY,
uri VARCHAR(2048) NOT NULL
);
CREATE TABLE keywords(
term_id SERIAL PRIMARY KEY ,
term VARCHAR(100) NOT NULL,
score NUMERIC(4,3) NOT NULL,
article_id INT NOT NULL,
CONSTRAINT fk_articleid FOREIGN KEY(article_id) REFERENCES url(id)
);
```

Please note the following constraints:

1. For the table 'url': The id provided itself in the Wikipedia-EN-20120601_REVISION_URIS.TSV is made the primary key for the table url.
2. For the table 'keywords': We generate a surrogate key term_id using SERIAL (Similar to AUTO INCREMENT in MySQL) for each entry of the words which is also made as the primary key of the table.
3. The attribute article_id is a foreign key inside the keywords table that references the id from the url table.

2

The bulk-loading command can be done using the following:

```
COPY url(id,uri) FROM '/tmp/Wikipedia-EN-20120601_REVISION_URIS.TSV'
DELIMITER E '\t' ENCODING 'UTF8';
```

```
COPY keywords(article_id,term,score) FROM '/tmp/Wikipedia-EN-20120601_KEYWORDS.TSV'
DELIMITER E'\t' ENCODING 'UTF8';
```

Do note that the TSV files were stored in \tmp as it is a folder accessible to any user of the database easily. Also, we have already made our database accept UTF-8 Encoding as well as in COPY command also we stated the encoding to be UTF-8 which should ensure that UTF-8 encoding should be preserved.

3

To measure time \timing command is used in psql and all data obtained is based on that.

1. With the constraints part of create queries
   Time taken:
   (a) 32.717 ms to copy Wikipedia-EN-20120601_REVISION_URIS.TSV
   (b) 44495.191 ms (44.495 s) to copy Wikipedia-EN-20120601_KEYWORDS.TSV
2. Without constraints active
   Time taken:
   (a) 35.011 ms to copy Wikipedia-EN-20120601_REVISION_URIS.TSV
   (b) 9290.191 ms (9.290 s) to copy Wikipedia-EN-20120601_KEYWORDS.TSV

Running Keyword Queries over Wikipedia

1.

First, we need to filter the keywords table with rows having any of the 4 given words. The resultant table can then be used to create an aggregate statistic of article_id along with a count of keywords which would basically group an article id with the frequency of the keyword. After that, if the count for a given article_id is = 4 that means it contains all of the given keywords and we return that article_id for the final result which is a foreign key to the links tables, and eventually we can get the URLs corresponding to the article_id as the answer. The query is as follows:

```
SELECT uri as url from url where url.id in (SELECT temp.article_id from (SELECT article_id,count(article_id) from keywords where
(term = 'infantri' or term = 'reinforc' or term = 'brigad' or term = 'fire')  GROUP BY(article_id)) as temp where temp.count=4);
```

Please note we have assumed that we have to look for the words exactly mentioned in the doc and not look for words whose stemming gives the following outputs. If that is the case then the query would change as follows:

```
SELECT uri as url from url where url.id in (SELECT temp.article_id from (SELECT article_id,count(article_id) from keywords where
(term like '%infantri%' or term like '%reinforc%' or term like '%brigad%' or term like '%fire%')
GROUP BY(article_id)) as temp where temp.count=4);
```

2

Using a similar line of approach(of filtering rows with any of the 4 given words) with frequency as above. Note that for 'any of' we just need to have the count of words for a particular article_id to be greater than 0. This means that there are any of the following words present. The query for this is as follows:

```
SELECT uri as url from url where url.id in (SELECT temp.article_id from (SELECT article_id,count(article_id)
from keywords where (term = 'infantri' or term = 'reinforc' or term = 'brigad' or term = 'fire')
GROUP BY(article_id)) as temp where temp.count>0);
```

This query can further be simplified and we actually don't need the count comparison due to the framing of the where statement in the inner-most query.

```
SELECT uri as url from url where url.id in (SELECT article_id
from keywords where (term = 'infantri' or term = 'reinforc' or term = 'brigad' or term = 'fire'));
```

Again here we have implicitly assumed that the words given are to be searched exactly and the same assumption applies to all following questions.

3

First, we get all the rows from the keywords table which contains 'reinforc'. Along with that, we try to get rows from keywords that contain any one of the remaining 3 words. Then we remove those entries from the table containing ids with 'reinforc'. The result should give us rows that have ids of articles containing 'reinforc' but none of the other 3. The final query is as follows:

```
SELECT uri as url from url where url.id in (SELECT temp2.article_id from (SELECT temp1.article_id from
(SELECT article_id,count(article_id) from keywords where (term = 'reinforc')
GROUP BY(article_id)) as temp1 where temp1.count>0) as temp2 where article_id not in
(SELECT temp.article_id from
(SELECT article_id,count(article_id) from keywords where (term = 'infantri'  or term = 'brigad' or term = 'fire')
GROUP BY(article_id)) as temp where temp.count>0));
```

Again we can reduce the size of the query as the size condition is redundant. The smaller query is as follows:

```
/* Problem 3 - Shortened */
SELECT uri as url from url where url.id in
(SELECT article_id from keywords where term = 'reinforc' and article_id not in
(SELECT article_id from keywords where (term='infantri' or term='brigad' or term='fire')));
```

4.

The row retrieval logic remains the same as what we did in the first question but here once we get the id's then we need to do an aggregate query of that id and the sum of scores corresponding to the id (group by id) and order it in descending order. This sequence of id is then linked with the url table via the foreign key to giving the required URL. The query is as follows:

```
SELECT uri as url from url where url.id in (SELECT temp3.article_id from
(SELECT temp2.article_id,sum(temp2.score) from (SELECT article_id,score from keywords where
(article_id in (SELECT temp.article_id from (SELECT article_id,count(article_id) from keywords where
(term = 'infantri' or term = 'reinforc' or term = 'brigad' or term = 'fire')
GROUP BY(article_id)) as temp where temp.count=4) and (term = 'infantri' or term = 'reinforc' or term = 'brigad' or term = 'fire'))) as
temp2 GROUP BY(temp2.article_id) ORDER BY sum DESC) as temp3);
```

5.

The row retrieval logic remains the same as what we did in the second question but here once we get the id's then we also need to obtain the scores for all those keywords corresponding to the ids found. This is followed by an aggregate query of the sum of scores corresponding to the id (group by id) and scores now which we order in descending order. This sequence of id is then linked with the url table via the foreign key to giving the required URL. The query is as follows:

```
SELECT uri as url from url where url.id in (SELECT temp3.article_id from
(SELECT temp2.article_id,sum(temp2.score) from (SELECT article_id,score from keywords where
(article_id in (SELECT temp.article_id from (SELECT article_id,count(article_id) from keywords where
(term = 'infantri' or term = 'reinforc' or term = 'brigad' or term = 'fire')
GROUP BY(article_id)) as temp where temp.count>0) and (term = 'infantri' or term = 'reinforc' or term = 'brigad' or term = 'fire'))) as
temp2 GROUP BY(temp2.article_id) ORDER BY sum DESC) as temp3);
```

Again the same query can be shortened by removing the > 0 clause as it is redundant as shown in 2.

6.

To retrieve the rows, we first select all those articles that have the keyword 'reinforc' that don't all of 'fire', 'brigad' and 'infantri' in them. Now that we have the articles required, we have to order them based on their scores. It is given that if the sum of 'fire', 'brigad' and 'infantri' is less than 'reinforc' then articles with higher 'reinforc' values are ranked higher.

Method 1 Here we assume that the 'reinforc' being larger than the sum of the other three would mean that we would take the value of reinforc to order our results. Similarly if the sum of the other three keywords was higher, we would take that.

Based on this, we require the sum of 'reinforc' and the sum of the other 3. The sums can be found by obtaining the scores of all those keywords corresponding to the ids found. Now that we have 2 tables - one containing sums of all 3 and one containing the sum of 'reinforc', we take an outer join and take the greatest value for each row. Then the resulting table is ordered in descending order. The corresponding URLs are obtained by a simple join operation.

```
SELECT uri as url from url where url.id in
(SELECT temp10.article_id from (SELECT temp7.article_id, greatest(temp7.scoreone, temp7.scorefour) AS maxx from
(SELECT temp9.article_id, temp8.scorefour, temp9.scoreone from ((SELECT temp2.article_id, sum(temp2.score) as "scorefour" from
(SELECT article_id, score from keywords where (article_id in (SELECT temp2.article_id from (SELECT temp1.article_id from
(SELECT article_id,count(article_id) from keywords where (term = 'reinforc') GROUP BY(article_id)) as temp1 where temp1.count>0)
as temp2 where article_id not in
(SELECT temp.article_id from (SELECT article_id, count(article_id) from keywords where
(term = 'brigad' or term = 'infantri' or term = 'fire') GROUP BY(article_id)) as temp where temp.count=3))
and (term = 'brigad' or term = 'infantri' or term = 'fire'))) as temp2 GROUP BY(article_id)) as temp8
FULL OUTER JOIN
(SELECT temp4.article_id, sum(temp4.score) as "scoreone" from (SELECT article_id, score from keywords where
(article_id in (SELECT temp4.article_id from (SELECT temp5.article_id from (SELECT article_id,count(article_id) from keywords where
(term = 'reinforc') GROUP BY(article_id)) as temp5 where temp5.count>0)
as temp4 where article_id not in
(SELECT temp6.article_id from (SELECT article_id, count(article_id) from keywords where
(term = 'brigad' or term = 'infantri' or term = 'fire') GROUP BY(article_id)) as temp6 where temp6.count=3)) and
(term = 'reinforc'))) as temp4 GROUP BY(article_id)) as temp9 on temp8.article_id = temp9.article_id))
as temp7 ORDER BY maxx DESC) as temp10);
```

Method 2 Here we assume that if 'reinforc' for a given article is higher than the sum of the other three for other articles then it would be ranked higher than the other articles. This means that we should order the results based on the total score as well as maintaining order between those articles where the sum

of the other three keywords is less than 'reinforc'.
To do this, in the join of the 2 (The score of 'reinforc' and the sum of the scores of the other keywords) sums we just order the results by the decreasing order of the total sum first (articles which have at least one of the 3 keywords will be ranked higher as per this logic) and then in the decreasing order of the score of 'reinforc'. This is done by an order by clause with priority given to the total sum, then the reinforc score. After this we do a join with the url table to get the final result.

```
SELECT uri from url INNER JOIN
(SELECT *,  (temp1.sumone + temp2.sumfour) as f_score from
(SELECT article_id as id1, sum(score) as sumone  from keywords where term = 'reinforc'  GROUP BY(id1)) as temp1
LEFT JOIN
(SELECT article_id as id2, count(article_id) as count, sum(score) as sumfour from keywords where
term = 'brigad' OR term = 'fire'  OR term = 'infantri' GROUP BY (id2)) as temp2 ON temp1.id1 = temp2.id2
where temp2.count IS NULL OR temp2.count != 3
)as comparision
ON url.id = comparision.id1
ORDER BY comparision.f_score DESC, comparision.sumone DESC;
```