Assignment 2

SARPREET SINGH BUTTAR<SB223CE@STUDENT.LNU.SE>

SONGHO LEE<SL222XK@STUDENT.LNU.SE>

Task 1

Our assumption of task 1 is none of relations are empty relation.

1. $|R \cup S|$: max = n+m, min = n = m
   maximum: where relation R and S do not have any duplicated tuples.
   minimum: where all tuples are duplicated in both relations.
2. $|R \bowtie S|$: max= $n \times m$, min= 0
   maximum: where all tuples of relation R and S have shared attribute.
   minimum: where no tuples of R and S have any shared attribute.
3. $|\sigma_c(R) \times S|$: max= $n \times m$, min = m
   maximum: where all tuples of relation R fulfils a condition C
   minimum: where no tuples of relation R fulfils a condition C
4. $|\pi_L(R) \setminus (S)|$: max =n, min = 0
   Assume that projection L has the same attributes as a relation S.
   maximum: where a relation S is different to projection of R
   minimum: where a projection of R equals to the relation S

Task 2

Let us examine the first case.

1. BCNF violations
   We found ABE as a candidate key and a super key because the closure of ABE fetches all the elements of the relation. ($ABE^+ = ABCDE$). None of the determinants of the given functional dependencies contains ABE, which is the candidate key. Therefore, it violates BCNF.

2. Decomposition of the first relation to satisfy BCNF
   First, we found our candidate key [ABE]. $ABE^+ = ABCDE$.
   Second, we chose one of the given functional dependencies, $AB \rightarrow C$.
   The determinant of a given FD is not the candidate key.
   Then, we find the closure of AB. $AB^+ = ABCD$. We take ABCD as a relation R1. Then, it covers other two given dependencies: $AB \rightarrow C$, $B \rightarrow D$ and it does not fulfil the BCNF yet. We will decompose this relation further to $R_1 = ABC$ ($\because AB^+$) and $R_2 = BD$($\because B^+$). Still, we have not covered $DE \rightarrow C$.
   Similarly, DE is not the candidate key, so we found the closure of the DE. $DE^+ = DEC$. We take 'DEC' as a relation. R_3(DEC) fulfils $DE \rightarrow C$, and it complies the BCNF for similar reason as mentioned above.
   In order to represent $R(ABCDE)$, we need two collections of R1 and R2.
   $R = R_1(ABC) \cup R_2(BD) \cup R_3(DEC)$
   $R_1 : AB \rightarrow C; R_2 : B \rightarrow D; R_3 : DE \rightarrow C$
3. 3NF violations
   In order to meet 3NF requirement, it should fulfil 2NF and non-key attributes are FD on the entire primary key. We found out it does not fulfil 2NF.
4. Decomposition of the first relation to satisfy 3NF
   $R = R_1(ABC) \cup R_2(BD) \cup R_3(DEC)$
   $R_1 : AB \rightarrow C; R_2 : B \rightarrow D; R_3 : DE \rightarrow C$
   According to the definition of BCNF, BCNF relation fulfils 3NF requirements.

Here we examine the second case.

1. BCNF violations
   We found AB, AC candidate keys because the closure of these fetches all the elements of the relation. ($\because AB^+ = AC^+ = ABCDE$). One determinant of the given functional dependencies contains AB, but the rest of these do not. Therefore, it violates BCNF as not all determinants are candidate keys in this case.
2. Decomposition of the first relation in BCNF
   Our candidate keys: [AB, AC]. $AB^+ = AC^+ = ABCDE$.

Second, we chose one of the given functional dependencies, $AB \rightarrow C$. The determinant is the candidate key. Therefore, there is no need to take a closure.

We take the FD as a relation $R_1(ABC)$

We go further to the rest of FD. We take FD $C \rightarrow D$. This FD does not fulfil BCNF because the determinant is not the candidate key of the relation ABCDE. Therefore, we find the closure of C. $C^+ = CDBE$. We take CDBE as a relation R_2. Then, it covers other two given dependencies: $D \rightarrow B$, $D \rightarrow E$ but it does not fulfil the BCNF because the determinant (D) is not the key. We need to decompose further. We take a closure of $D^+ = DBE$. In this case, all the determinant of FD $D \rightarrow B, D \rightarrow E$ is the key element. It fulfils BCNF.

We take the closure into a relation $R_2(DBE)$, and we take FD: $C \rightarrow D$ into another relation $R_3(CD)$.

In order to represent $R(ABCDE)$, we need two collections of R1 and R2.

$R = R_1(ABC) \cup R_2(DBE) \cup R_3(CD)$

$R_1 : AB \rightarrow C;\ R_2 : D \rightarrow B,\ D \rightarrow E; R_3 : C \rightarrow D$

3. All 3NF violations

   The model does not fulfil the 2NF because "all non-key attributes should functionally depend on the entire primary key", but the attributes do not dependent on the super key. As 3NF requires 2NF, the relation does not meet the 3NF. Furthermore, there are transitive relations such as $AB \rightarrow C$, $C \rightarrow D$ and $C \rightarrow D$, $D \rightarrow B$

4. Decomposition of the first relation in 3NF

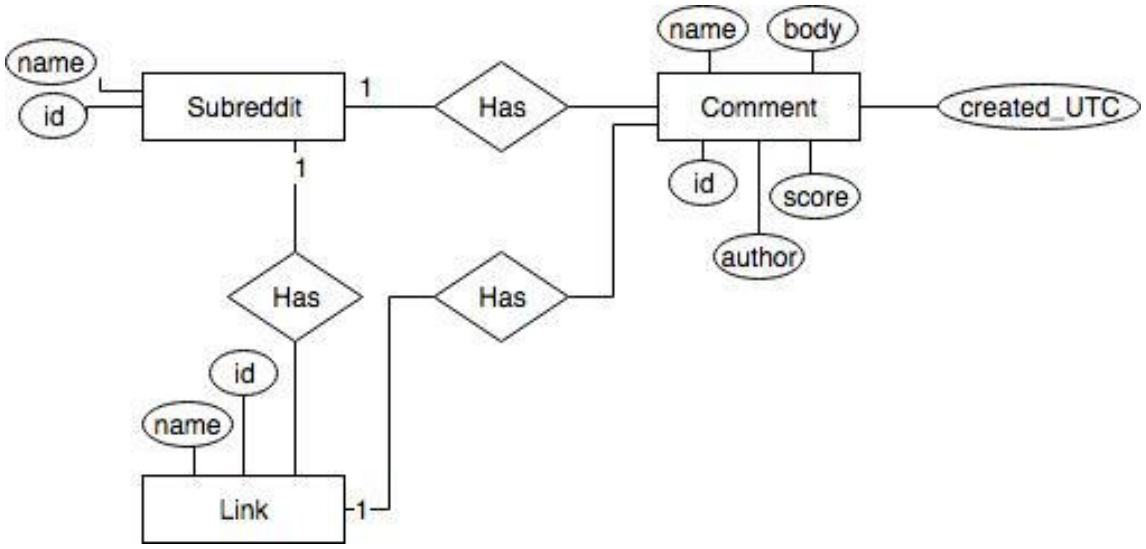
   $R = R_1(ABC) \cup R_2(DBE) \cup R_3(CD)$

   $R_1 : AB \rightarrow C;\ R_2 : D \rightarrow B,\ D \rightarrow E; R_3 : C \rightarrow D$

   According to the definition of BCNF, BCNF relation fulfils 3NF requirements.

Task3

Below is the E/R diagram of the Reddit.



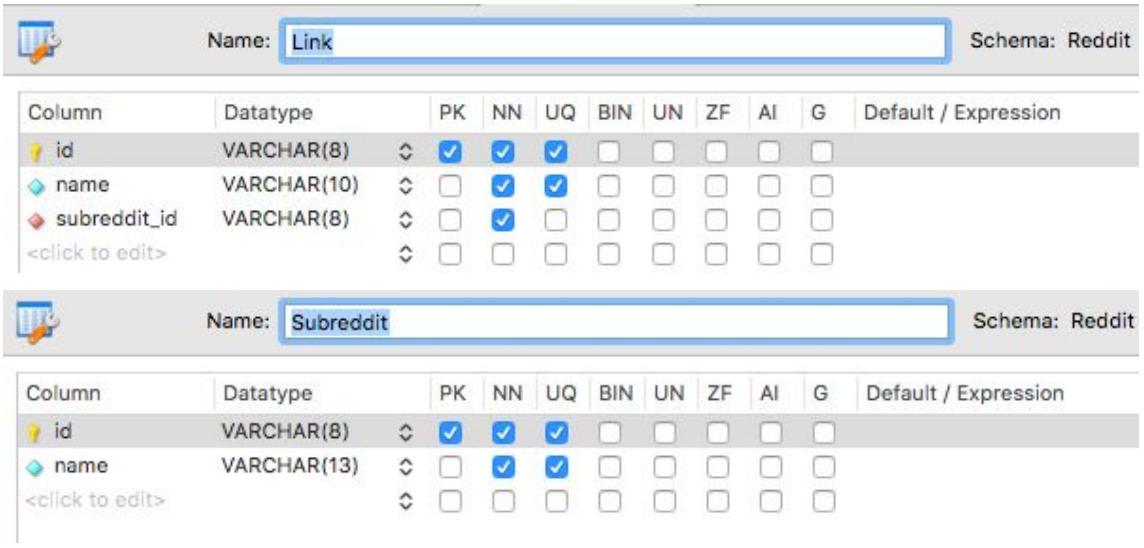
We convert the above diagram to the tables. Types of these are Strings, unless specified.

Link(id, name, subreddit_id*)
Comment(id, name, author, score:int, body, subreddit_id*, parent_id, link_id* created_UTC:TimeStamp)
Subreddit(id, name)

We present the schema of all tables below.

Name: Comment Schema: Reddit

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
id	VARCHAR(7)	☑	☑	☑	☐	☐	☐	☐	☐	
name	VARCHAR(10)	☐	☑	☐	☐	☐	☐	☐	☐	
author	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	
score	INT(11)	☐	☑	☐	☐	☐	☐	☐	☐	
body	VARCHAR(9900)	☐	☑	☐	☐	☐	☐	☐	☐	
subreddit_id	VARCHAR(8)	☐	☑	☐	☐	☐	☐	☐	☐	
parent_id	VARCHAR(10)	☐	☑	☐	☐	☐	☐	☐	☐	
created_utc	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
link_id	VARCHAR(8)	☐	☑	☐	☐	☐	☐	☐	☐	
<click to edit>		☐	☐	☐	☐	☐	☐	☐	☐	


Name: Link
Schema: Reddit
Column Datatype PK NN UQ BIN UN ZF AI G Default / Expression
id VARCHAR(8)
name VARCHAR(10)
subreddit_id VARCHAR(8)
<click to edit>
Name: Subreddit
Schema: Reddit
Column Datatype PK NN UQ BIN UN ZF AI G Default / Expression
id VARCHAR(8)
name VARCHAR(13)
<click to edit>

Task 4

According to the Reddit API, 'name' field of each data represents a type of the object, by including 't1_' or 't2_', and such.
However, as all entries' name field only begins with 't1' in the provided data file, we can conclude that the files only contain 'Comment' data type.

Therefore, we had to derive subreddit and link from comment object's properties using our programme.
Here is a short description of what the programme does. It reads the data line by line and stores subreddit names and ids in a subreddit table in a database, link_id and link name, subreddit_id entries to a link table and entries with all the fields excluding the subreddit name to a comment table.

Without Constraints

First, we created the tables without any constraints; no primary key and no foreign keys exist there. Since many comments belong to the same subreddits and the links, the subreddit and link tables end up having many duplicates. (I.e. all the tables have same amount of entries)

Our programme iterates over the n-entries from the data file and adds subreddits, link, comments in parallel. It took 10 minutes to import data from a file named 2007.

With Constraints

Meanwhile, once there are constraints such as foreign keys, an entry cannot be imported if foreign keys do not exist in their respective tables. For example, a comment cannot be imported if its parents like subreddit and link do not exist. So, we added subreddit first, then link, comments as the last in the same iteration. It is notable that now the import cannot be done concurrently, as the order matters; unlike the previous case, only single thread is used. It took 17 minutes in total.

Although it is faster to import data without having constraints, as it allows having multiple thread to process the import, it is not reasonable choice to apply constraints afterwards, because having no constraint may allow, for example, duplicated data, and the constraints cannot be added afterwards.
However, we can import the data and apply the constraints if integrity of the data dump is guaranteed.

Task 5

We provide MySQL queries to respective questions in this section.

1.

```
SELECT COUNT(id) FROM Reddit.Comment where author =
'paternoster'
```

This query select the given author and count its number of comments.

2.

```
SELECT
      COUNT(id) as Comment_count,
      DATE(created_utc) as Comment_day
FROM Comment
WHERE subreddit_id = 't5_6'
GROUP BY Comment_day
```

This query select all the comments of given subreddit_id and counts the total comments on each specific date.

3.

```
SELECT COUNT(*) FROM Reddit.Commentwhere body LIKE '%lol%';
```

This query select all the comments whose body contains the given word (lol).

4.

```
SELECT name from Subreddit where id in (SELECT DISTINCT
subreddit_id from Comment where author in (SELECT author from
Comment where link_id = '5yba3'))
```

First, the query selects the authors who are commenting on a link ‘5yba3’. Then, it selects subreddit_id which the authors have commented on, but it takes only unique subreddit_id. Afterwards, it fetches the name of subreddit from subreddit table using its id.

5.

```
CREATE VIEW scores AS
    SELECT DISTINCT author, SUM(SCORE) AS ScoreSum from
Comment group by author;
SELECT MAX(ScoreSum) As MaxScore,MIN(ScoreSum) As MinScore
from scores;
DROP VIEW scores;
```

This query first selects the distinct author and sum of each author's score. Afterwards, it temporarily stores the result set as scores. Afterwards, we Select the highest and lowest score from the resultset. We remove the resultset after finishing our query.

6.

```
select name, 'Highest' as type from Subreddit
where id = (select subreddit_id from comment where score =
(SELECT MAX(score) as Maximum from Comment))
UNION
select name, 'Lowest' as type from Subreddit
where id = (select subreddit_id from comment where score =
(SELECT MIN(score) as Minimum from Comment))
```

This query find the max and min score and return their union.

7.

```
select DISTINCT author from Comment where link_id in (select
link_id from comment where author = 'newhen') and author
!='newhen'
```

This query select all the link_id of given author and then select all the others who have also contains founded link_id except given user.

8.

```
SELECT author, COUNT(DISTINCT subreddit_id) AS SubredditCount
    from COMMENT GROUP BY author HAVING COUNT(SubredditCount)
= 1
```

This query select the author, count unique subreddit_id and then return where count is one.

Indexing

The query we mentioned in number four, was takes in average 6.342 seconds, which is the most time consuming of all the queries. This query is fetching subreddit_id and 'author' column two times.

Once we apply a multicolumn index on subreddit_id(#1) and author(#2) from the Comment table, the query time reduces to 0.01425 second. Meanwhile, applying index on author(#1) and subreddit_id(#2) didn't improve any performance. Our motivation to this phenomenon is that the above query executes fetching subreddit_id first and queries about author. If our query follows the order of index, it skips manually iterating over the whole entries, and therefore it reduces query time significantly.