Linnéuniversitetet
Kalmar Växjö

Report

Databases Theory

Assignment 2

Författare: Ruth Dirnfeld
Helena Tevar
Handledare: Maria Ulan
Examinator: Illir Jusufi
Termin: 18HT
Kurskod: 1DV513

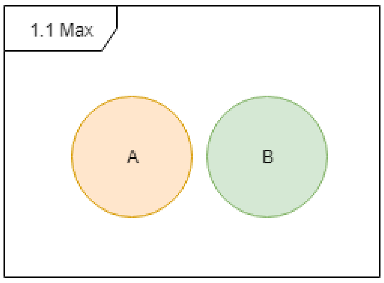
Linnéuniversitetet
Kalmar Växjö

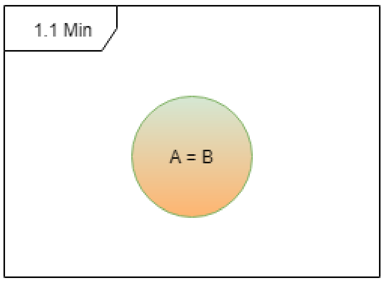
Innehållsförteckning

Task 1

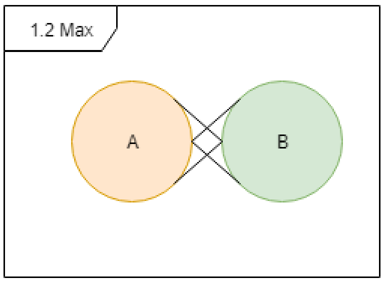
Suppose relations R and S have n and m tuples, respectively. Give the minimum and maximum number of tuples that the results of the following expressions can have.

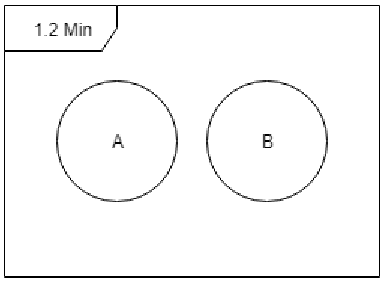
1. $R \cup S$
   a. Max: m + n (all tuples are unique)
   b. Min: m = n (all tuples are the same)



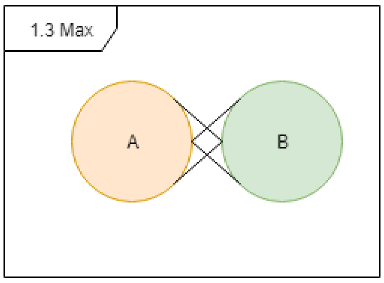


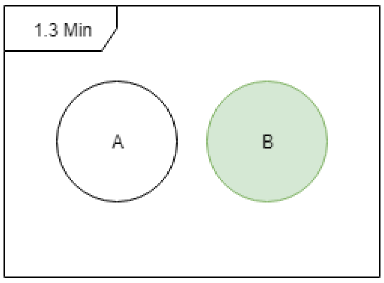
2. $R \bowtie S$
   a. Max: All the tuples have some shared attribute $\rightarrow$ $m \times n$
   b. Min: There is no tuples with shared attributes $\rightarrow 0$



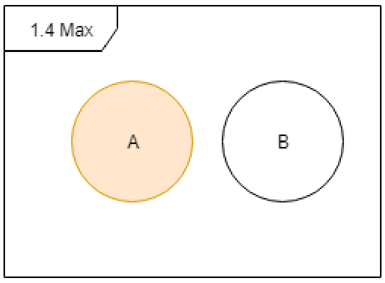


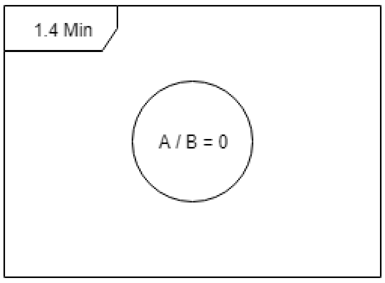
3. $\sigma_C (R) \times S$
   a. Max: All R follows C $\rightarrow$ $R \times S = m \times n$
   b. Min: No tuples to construct the cartesian product = 0





4. π_L (R) \ S
   a. Max: All n follows the projection of L and S does not shares with R = n.
   b. Min: There is no n in R that follows projection L = 0.
   c. Min part 2: Even if there was some n, R and S are the same = 0.





Task 2

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

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

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.

BCNF Violations

Relation 1

The FD of AB and the reflexive dependency of E to itself completes all the relations with all the attributes of this case. That means that sets A, B and E are keys.

$AB \cup E \rightarrow A,B,C,D,E = \{ABE\}$ key.

Any of the FDs follow the rule of BCNF of having the super key set represented in the left side of the dependency, so **this relationship do not follow the BCNF**.

Relation 2

By following the closure of $\{AB\}^+$ and $\{AC\}^+$ We could conclude that AB, AC are the keys of this relations because from that set, you could reach all the attributes. However, not al the FDs include the super key in their left side, so **this relationship do not follow the BCNF** neither.

Decomposition

Relation 1
$R(A,B,C,D,E)$ with functional dependencies $AB \rightarrow C$, $DE \rightarrow C$ and $B \rightarrow D$.
Key $\{ABE\}$
$AB \rightarrow C$ is not BCNF. The closure of AB+ = ABC and create a new relation with it.
Solution: **R(A, B, C)**
$DE \rightarrow C$ is not BCNF. The closure of DE+ = CDE and create a new relation with it.
Solution: **R(C, D, E).**
$B \rightarrow D$ is not BCNF . Closure B+ = D Solution: **R(B,D)**.
So, Relation: $R(ABC) \cup R(BD) \cup R(DEC)$

Relation 2
$R(A,B,C,D,E)$ with functional dependencies $AB \rightarrow C$, $C \rightarrow D$, $D \rightarrow B$, $D \rightarrow E$
Key = AB, AC
$AB \rightarrow C$ is BCNF, we take the functional dependency so
Solution: **R(A,B,C)**
$C \rightarrow D$ with functional dependency we get
Solution: **R(C,D)**
Now we take:
$C \rightarrow D$ is not BCNF, so the closure of C = CDBE, which covers $D \rightarrow B$, $D \rightarrow E$
$D \rightarrow B$, $D \rightarrow E$ is not BCNF, so after decomposing the closure of D = DBE
So: **R(D,B,E)**
So, Relation: $R(ABC) \cup R(CD) \cup R(DBE)$

3NF Violations

Relation 1
Is not 3NF because the non key attributes should be dependent on the primary key.
$DE \rightarrow C$ violates 3NF
Relation 2
Is not 3NF because the non key attributes should be dependent on the primary key.
$AB \rightarrow C$, $D \rightarrow B$, $D \rightarrow E$ all violate 3NF

Decomposition 3NF

For this task we will decompose the dependencies as written below.

Relation 1
$AB \rightarrow C$, $DE \rightarrow C$, $B \rightarrow D$
We can remove redundant attributes as D in the second dependency because we can get to C through AB, and to D through B.
$AB \rightarrow C$, $E \rightarrow C$, $B \rightarrow D$

$R(ABC) \cup R(EC) \cup R(BD)$

Relation 2

$AB \rightarrow C,\ C \rightarrow D = AB \rightarrow C, D$

We stop here because the FD must be non trivial.

$D \rightarrow B,\ D \rightarrow E = D \rightarrow B, E$

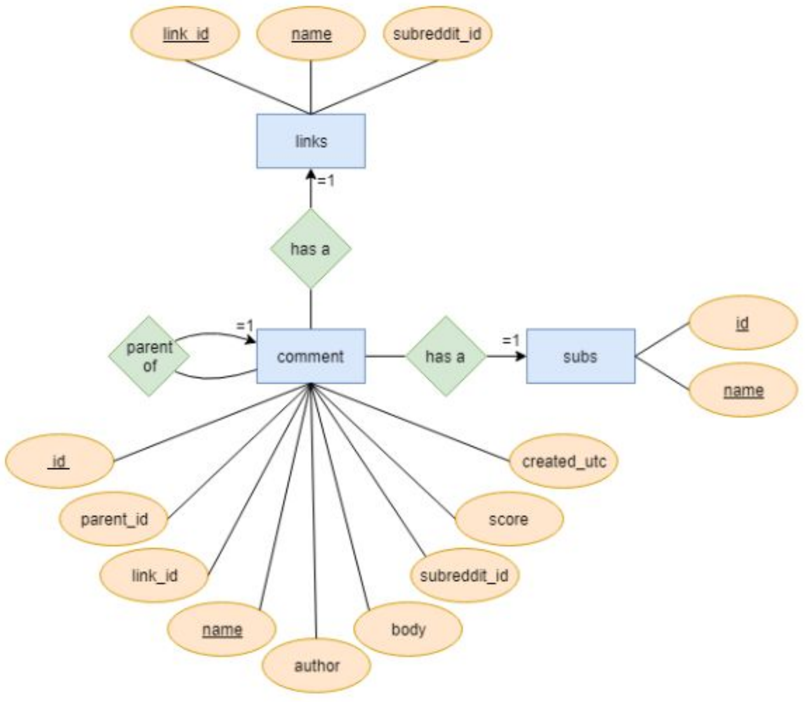
We cannot eliminate any of given dependencies. First dependency does not imply the second one. They are minimal basis, so we create for each functional dependency a schema of the relations.

$R(ABCD) \cup R(BDE)$

Task 3

Diagram

The diagram shows the relations between a reddit comment with a subreddit and links. Each comment has a group of attributes, including parent_id. The id of the parent showed in a comment points to the parent comment, which is a comment acting with a different role in the relationship. We can assume that a comment can be the parent of different posts, but for each comment there is only one parent.



Schema: reddit

Following the diagram, we created three schemas:

- ➔ comment(<u>id</u>, parent_id, link_id, <u>name</u>, author, body, subreddit_id, score, created_utc)
- ➔ subs(<u>id</u>, <u>name</u>)
- ➔ links(<u>id</u>, <u>name</u>, subreddit_id)

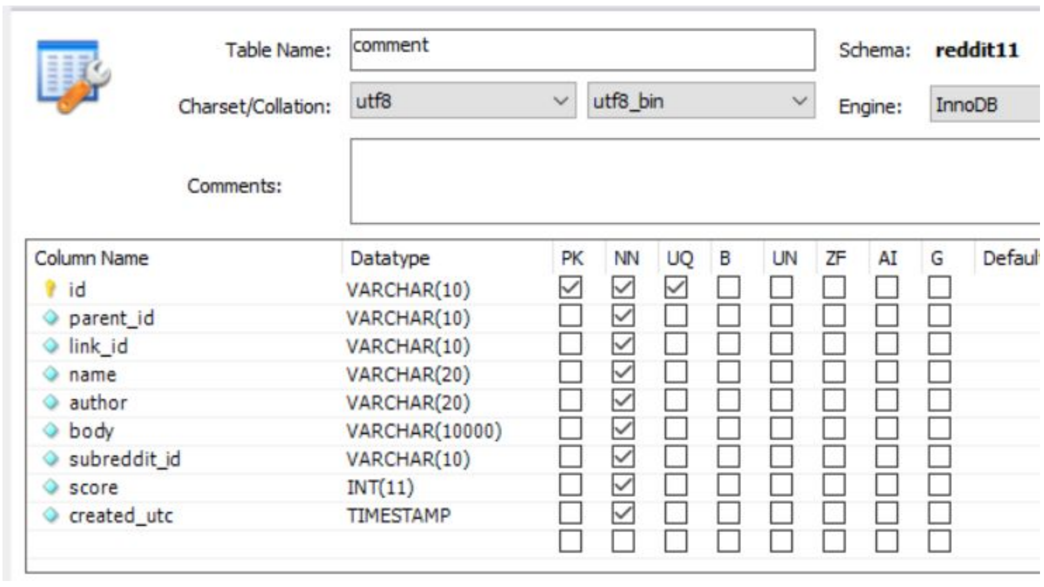


Table Name: comment Schema: reddit11
Charset/Collation: utf8 utf8_bin Engine: InnoDB
Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Defaul
id	VARCHAR(10)	✓	✓	✓						
parent_id	VARCHAR(10)		✓							
link_id	VARCHAR(10)		✓							
name	VARCHAR(20)		✓							
author	VARCHAR(20)		✓							
body	VARCHAR(10000)		✓							
subreddit_id	VARCHAR(10)		✓							
score	INT(11)		✓							
created_utc	TIMESTAMP		✓							

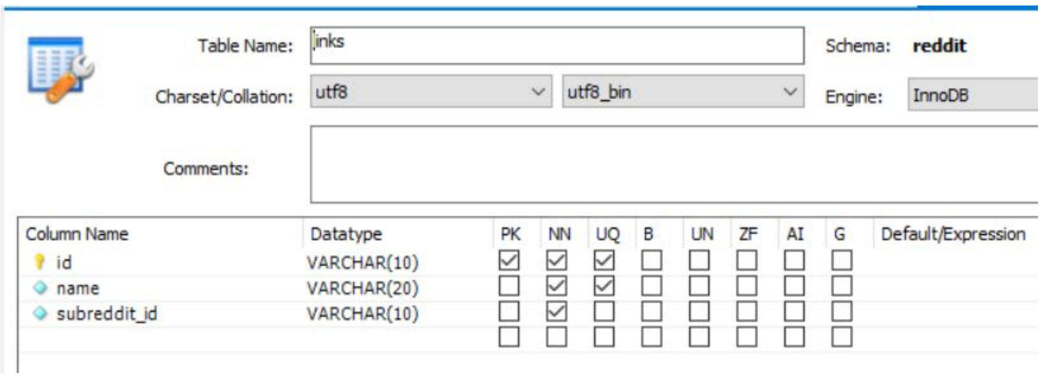


Table Name: links Schema: reddit
Charset/Collation: utf8 utf8_bin Engine: InnoDB
Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	VARCHAR(10)	✓	✓	✓						
name	VARCHAR(20)		✓	✓						
subreddit_id	VARCHAR(10)		✓							

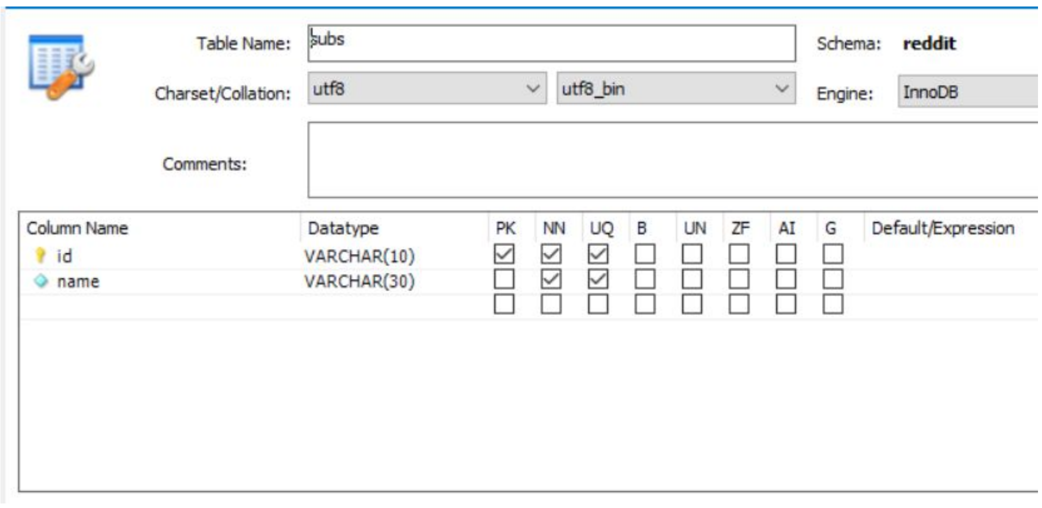


Table Name: subs Schema: reddit
Charset/Collation: utf8 utf8_bin Engine: InnoDB
Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	VARCHAR(10)	✓	✓	✓						
name	VARCHAR(30)		✓	✓						

Task 4

For testing our database we imported first the RC_2007-10 and later for all official tests the RC_2011-07 file, which are also included in the results sections in the following Tasks.

Our program reads the json file line by line and populates the tables in MySQL. First we populate the subs table and insert values for: subreddit id and subreddit name. Next we populate and insert values to the links table for: link id, link name and subreddit_id. And last, we populate the comment table and insert values for: id, parent_id, link_id, name, author, body, subreddit_id, score and created_utc.

First we populated the tables **without constraints**, which means that we have not assigned any primary keys, nor unique / foreign keys. For reusability reasons and possible change of mind, we do not populate the tables in parallel, rather we chose in code which table to populate. As for time measurements: the importing of all values into the tables without any constraints took in total 441 seconds when testing on the provided RC_2011-07 json file.

Next, we tested the insertions on the same file, but this time **with constraints**, such as primary keys and unique / foreign keys. As for time measurements: the insertion of all values into the tables with constraints took in total 553 seconds when testing on the provided RC_2011-07 json file.

Comparing the results of inserting the data with and without constraints, we can see that the insertion without constraints is faster. This has been tested on both files, RC_2007-10 and also on RC_2011-07. On the smaller the importing without constraints was 3 seconds faster than with constraints, and on the bigger file the difference was 112 seconds. In both cases the importing without constraints was faster. Nevertheless, even though it is faster to populate the tables without constraints, it is not a good choice, because once we populate without constraints, we cannot add them afterwards (when considering the populating the tables with much larger files). Furthermore, without constraints we insert duplicates as well.

Task 5

Queries

In order to elaborate this task, we will show how theoretically the queries are formed. Afterwards, we will try to adapt those queries to a programming scenario by using MySQL Workbench. The result will be added at the end of the explanation.

Note: Special thanks to www.w3schools.com for the help provided for this Task in this assignment.

1. **How many comments have a specific user posted?**

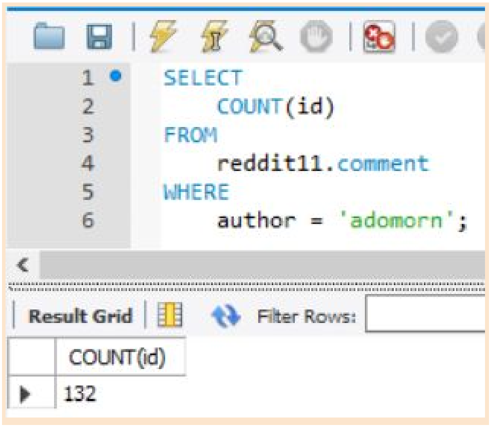
In order to get the number of comments for a specific user, we could solve it by selecting all the tuples from comments with the condition of a username.

SELECT COUNT *
FROM comment
WHERE author = ‘a_username’;

The solution by this query will count all the tuples from comment whereas the author is a specific username. We do not need to count from a table created with all the attributes, so a better query would be:

SELECT COUNT id
FROM comment
WHERE author = ‘a_username’;

Actual query:

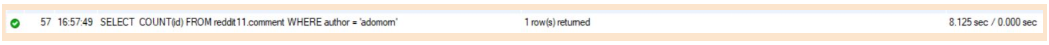


```
SELECT
    COUNT(id)
FROM
    reddit11.comment
WHERE
    author = 'adomorn';
```

Result Grid | Filter Rows:

COUNT(id)
132

Time taken: 8.125 sec



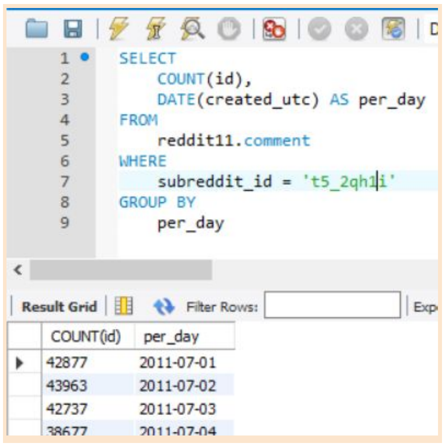
```
57  16:57:49  SELECT COUNT(id) FROM reddit11.comment WHERE author = 'adomorn'    1 row(s) returned    8.125 sec / 0.000 sec
```

2. **How many comments does a specific subreddit get per day?**

SELECT COUNT id, date
FROM comment
WHERE subreddit_id = ‘a_reddit_id’
GROUP BY date;

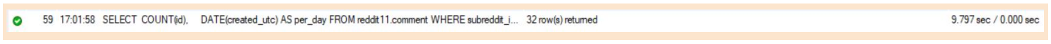
The tuples selected from the table comment with a given reddit id will be grouped by date and will be counted to give the solution.

Actual query:


```
SELECT
    COUNT(id),
    DATE(created_utc) AS per_day
FROM
    reddit11.comment
WHERE
    subreddit_id = 't5_2qh1i'
GROUP BY
    per_day
```

```
COUNT(id)   per_day
42877       2011-07-01
43963       2011-07-02
42737       2011-07-03
38677       2011-07-04
```

Time taken: 9.797 sec

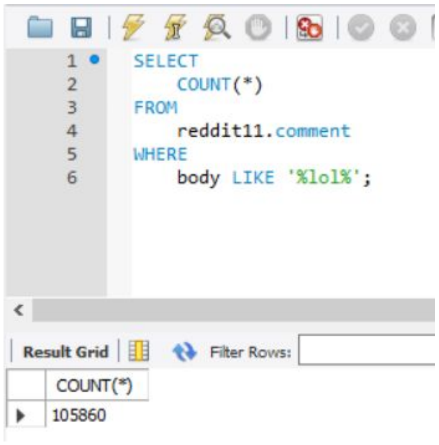

```
59  17:01:58  SELECT COUNT(id),   DATE(created_utc) AS per_day FROM reddit11.comment WHERE subreddit_i...  32 row(s) returned    9.797 sec / 0.000 sec
```

3. How many comments include the word 'lol'?

SELECT COUNT *
FROM comment
WHERE body LIKE '%lol%';

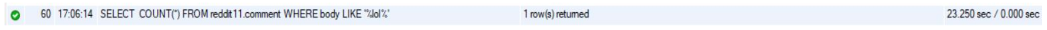
In this case, we want to count the comments where the string 'lol' is included. It may be in any place of the body, so we have to ensure with the sign '%' that there may be more characters before and after our string. Appreciate that this search will count words as 'lollipop', but we will assume that this is correct.

Actual query:


```
SELECT
    COUNT(*)
FROM
    reddit11.comment
WHERE
    body LIKE '%lol%';
```

```
COUNT(*)
105860
```

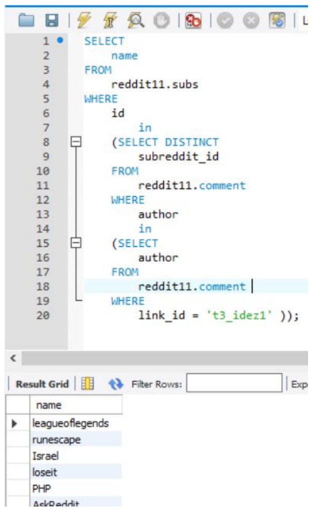
Time taken: 23.250 sec


```
60  17:06:14  SELECT COUNT(*) FROM reddit11.comment WHERE body LIKE '%lol%'    1 row(s) returned    23.250 sec / 0.000 sec
```

4. Users that commented on a specific link has also posted to which subreddits?
For what we understand, we have to pick all the subreddits that a specific user from a specific link has written. We use the link to get the author name, we select all subreddit_id from that author and then we select the name of that collection of subreddit_id. Multiple comments may have the same subreddit id so we have to make a DISTINCT action to prevent that. In this case the query will show all the information that follows our condition in the subreddit table, but we can constrict it to only the name or id of the subreddit if needed.

```
SELECT name
FROM subs
WHERE id IN (
        SELECT DISTINCT subreddit_id
        FROM comment
        WHERE author IN (
                SELECT author
                FROM comment
                WHERE link_id = 'a_specific_link' ))
```

Actual query:



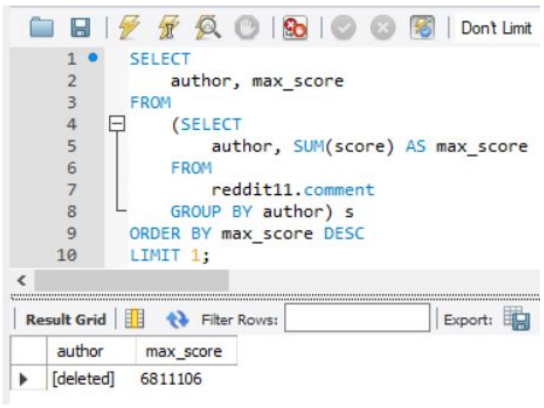
```
SELECT
    name
FROM
    reddit11.subs
WHERE
    id
        in
    (SELECT DISTINCT
        subreddit_id
    FROM
        reddit11.comment
    WHERE
        author
        in
    (SELECT
        author
    FROM
        reddit11.comment
    WHERE
        link_id = 't3_idez1' ));
```

```
Result Grid | Filter Rows: | Exp
name
leagueoflegends
runescape
Israel
loseit
PHP
AskReddit
```

5. Which users have the highest and lowest combined scores? (combined as the sum of all scores)
We need a relation that is defined in terms of the contents of other tables. We have to take care of not repeating the authors of the comments when making aggregations by using DISTINCT. Our second column will be the sum of the score attribute. The next queries are to select the maximum and minimum scores.

```
SELECT author, max_score
FROM (SELECT author,
        SUM(score) AS max_score
        FROM comment
        GROUP BY author)
ORDER BY max_score DESC
LIMIT1;
```

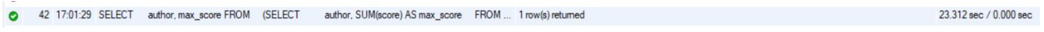
Actual sum of MAX query:



```
SELECT
    author, max_score
FROM
    (SELECT
        author, SUM(score) AS max_score
    FROM
        reddit11.comment
    GROUP BY author) s
ORDER BY max_score DESC
LIMIT 1;
```

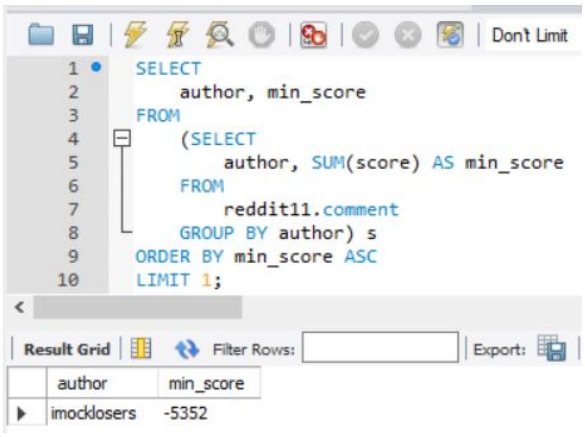
author	max_score
[deleted]	6811106

Time taken: 23.312 sec



```
42  17:01:29  SELECT  author, max_score FROM  (SELECT  author, SUM(score) AS max_score  FROM ...  1 row(s) returned  23.312 sec / 0.000 sec
```

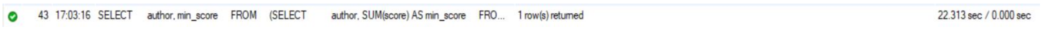
Actual sum of MIN query:



```
SELECT
    author, min_score
FROM
    (SELECT
        author, SUM(score) AS min_score
    FROM
        reddit11.comment
    GROUP BY author) s
ORDER BY min_score ASC
LIMIT 1;
```

author	min_score
imocklosers	-5352

Time taken: 22.313 sec



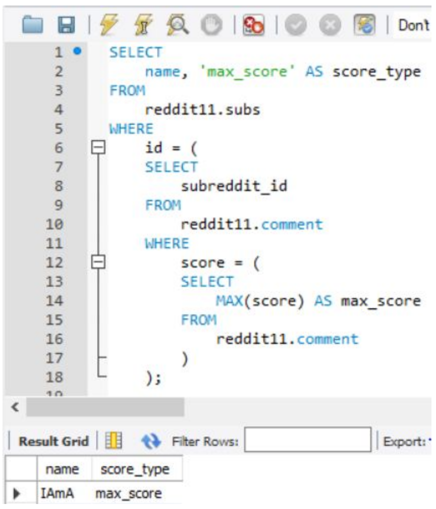
```
43  17:03:16  SELECT  author, min_score  FROM  (SELECT  author, SUM(score) AS min_score  FRO...  1 row(s) returned  22.313 sec / 0.000 sec
```

6. Which subreddits have the highest and lowest scored comments?

From bottom to top, the process is looking for the maximum score from comments, select the subreddit id of that result, select the name of that subreddit.

```
SELECT name, score_type
FROM subs
WHERE id = (
            SELECT subreddit_id
            FROM comment
            WHERE score = (
                        SELECT MAX(score)
                        FROM comment )
            );
```

Actual query:



```
SELECT
    name, 'max_score' AS score_type
FROM
    reddit11.subs
WHERE
    id = (
    SELECT
        subreddit_id
    FROM
        reddit11.comment
    WHERE
        score = (
        SELECT
            MAX(score) AS max_score
        FROM
            reddit11.comment
        )
    );
```

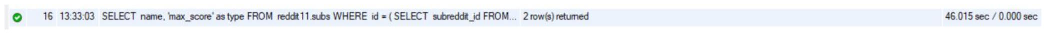
Result Grid | Filter Rows: | Export:

	name	score_type
▶	IAmA	max_score

In the case of the lowest score, it is the same pattern but looking for the minimum.

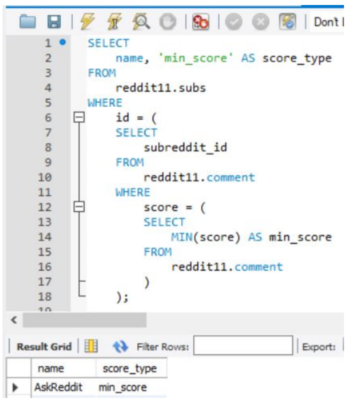
```
SELECT name
FROM subs
WHERE id = (
            SELECT subreddit_id
            FROM comment
            WHERE score = (
                        SELECT MIN(Score)
                        FROM comment)
            );
```

Time taken: 46.015 sec



```
16  13:33:03  SELECT name, 'max_score' as type FROM reddit11.subs WHERE id = ( SELECT subreddit_id FROM...  2 row(s) returned  46.015 sec / 0.000 sec
```

Actual query:



```
SELECT
    name, 'min_score' AS score_type
FROM
    reddit11.subs
WHERE
    id = (
    SELECT
        subreddit_id
    FROM
        reddit11.comment
    WHERE
        score = (
        SELECT
            MIN(score) AS min_score
        FROM
            reddit11.comment
        )
    );
```

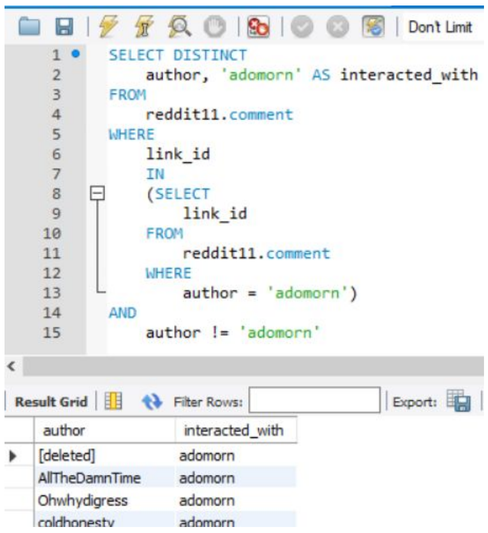
name	score_type
AskReddit	min_score

7. Given a specific user, list all the users he or she has potentially interacted with (i.e., everyone who has commented on a link that the specific user has commented on).

In this case we have to be careful to not duplicate the authors our user has interacted with. That is the reason why we have to use DISTINCT, so we only get single names of authors.

```
SELECT DISTINCT author, interacted_with
FROM comment
WHERE link_id in (
        SELECT link_id
        FROM comment
        WHERE author =  ‘a_username’)
AND != ‘a_username’;
```

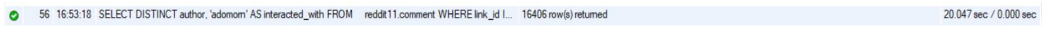
Actual query:



```
SELECT DISTINCT
    author, 'adomorn' AS interacted_with
FROM
    reddit11.comment
WHERE
    link_id
    IN
    (SELECT
        link_id
    FROM
        reddit11.comment
    WHERE
        author = 'adomorn')
AND
    author != 'adomorn'
```

author	interacted_with
[deleted]	adomorn
AllTheDamnTime	adomorn
Ohwhydigress	adomorn
coldhonesty	adomorn

Time taken: 20.047 sec



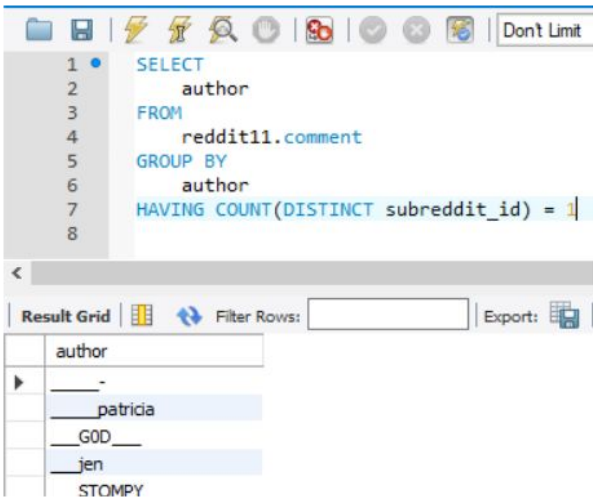
```
56  16:53:18  SELECT DISTINCT author, 'adomorn' AS interacted_with FROM    reddit11.comment WHERE link_id I...  16406 row(s) returned    20.047 sec / 0.000 sec
```

8. Which users has only posted to a single subreddit?

In this case we need to select all the authors that commented in exactly one subreddit. In this case we can group the authors and add the condition of having on count of distinct subreddits id.

SELECT author
FROM comment
GROUP BY author
HAVING COUNT (DISTINCT subrredit_id) = 1

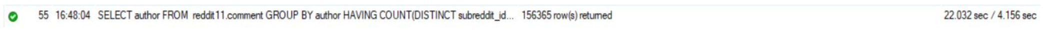
Actual query:



```
SELECT
    author
FROM
    reddit11.comment
GROUP BY
    author
HAVING COUNT(DISTINCT subreddit_id) = 1
```

```
author
_____-
_____patricia
___GOD___
___jen
STOMPY
```

Time taken: 22.032 sec



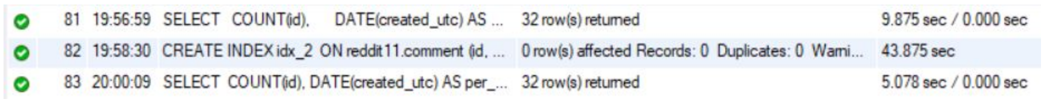
```
55  16:48:04  SELECT author FROM reddit11.comment GROUP BY author HAVING COUNT(DISTINCT subreddit_id...  156365 row(s) returned        22.032 sec / 4.156 sec
```

Indexes

We created Indexes for our Queries 2, 5.1, 5.2, 6, 7 and 8. We chose these queries because when compared from all together, these were a bit slower when running the query without indexing. In the following section, we listed the comparisons of running the queries before and after indexing, and at the end of this section, there is an explanation why the indexing improves the performance of the queries.

Index for Query 2:
CREATE INDEX idx_2
ON reddit11.comment (id, subreddit_id, created_utc);

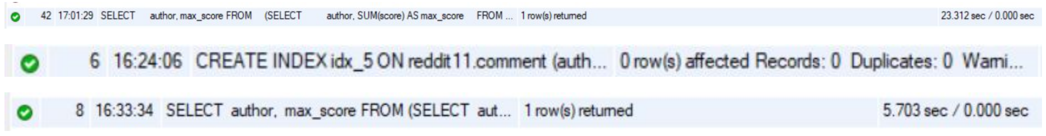
Output before Indexing was 9.875 and output after Indexing is more than 4 seconds faster. Output below..



```
81  19:56:59  SELECT  COUNT(id),    DATE(created_utc) AS ...   32 row(s) returned                                9.875 sec / 0.000 sec
82  19:58:30  CREATE INDEX idx_2  ON reddit11.comment (id, ...  0 row(s) affected Records: 0  Duplicates: 0  Warni...  43.875 sec
83  20:00:09  SELECT  COUNT(id), DATE(created_utc) AS per_...   32 row(s) returned                                5.078 sec / 0.000 sec
```

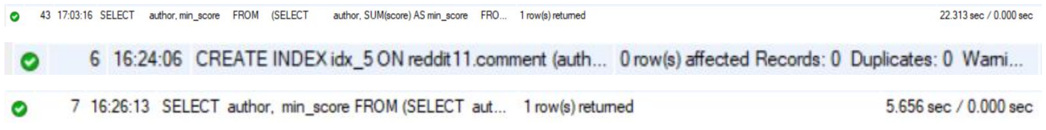
Index for Query 5:
CREATE INDEX idx_5
ON reddit11.comment (author, score);

5.1 before: 23.312 sec --- after: 5.703 sec



```
42  17:01:29  SELECT   author, max_score FROM   (SELECT    author, SUM(score) AS max_score   FROM ...  1 row(s) returned    23.312 sec / 0.000 sec
6  16:24:06  CREATE INDEX idx_5 ON reddit11.comment (auth...  0 row(s) affected Records: 0  Duplicates: 0  Warni...
8  16:33:34  SELECT  author,  max_score FROM (SELECT  aut...  1 row(s) returned    5.703 sec / 0.000 sec
```

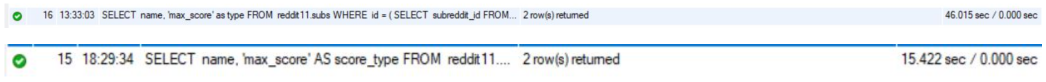
5.2 before: 22.313 sec --- after: 5.656 sec



```
43  17:03:16  SELECT   author, min_score   FROM   (SELECT    author, SUM(score) AS min_score   FRO...  1 row(s) returned    22.313 sec / 0.000 sec
6  16:24:06  CREATE INDEX idx_5 ON reddit11.comment (auth...  0 row(s) affected Records: 0  Duplicates: 0  Warni...
7  16:26:13  SELECT  author,  min_score FROM (SELECT  aut...  1 row(s) returned    5.656 sec / 0.000 sec
```

Index for Query 6:
CREATE INDEX idx_6
ON reddit11.subs (id, name);
+
CREATE INDEX idx_6_1
ON reddit11.comment (id, subreddit_id, score);

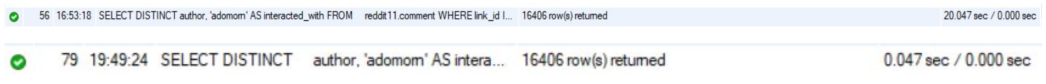
before: 46.015 sec --- after: 15.422 sec



```
16  13:33:03  SELECT  name, 'max_score' as type FROM  reddit11.subs WHERE  id = ( SELECT  subreddit_id FROM...  2 row(s) returned    46.015 sec / 0.000 sec
15  18:29:34  SELECT  name, 'max_score' AS score_type FROM  reddit11....  2 row(s) returned    15.422 sec / 0.000 sec
```

Index for Query 7:
CREATE INDEX idx_7
ON reddit11.comment (link_id, author, subreddit_id);

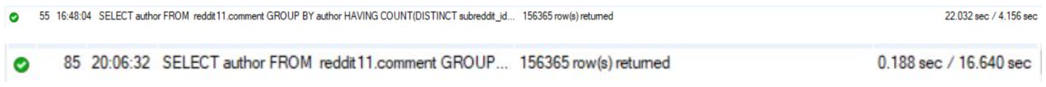
before: 20.047 sec --- after: 0.047 sec



```
56  16:53:18  SELECT DISTINCT author, 'adomom' AS interacted_with FROM    reddit11.comment WHERE link_id I...  16406 row(s) returned    20.047 sec / 0.000 sec
79  19:49:24  SELECT DISTINCT    author, 'adomom' AS intera...  16406 row(s) returned    0.047 sec / 0.000 sec
```

Index for Query 8:
CREATE INDEX idx_8
ON reddit11.comment (author, subreddit_id);

before: 22.032 sec --- after: 0.188 sec



```
55  16:48:04  SELECT author FROM  reddit11.comment GROUP BY author HAVING COUNT(DISTINCT subreddit_id...  156365 row(s) returned    22.032 sec / 4.156 sec
85  20:06:32  SELECT author FROM  reddit11.comment GROUP...  156365 row(s) returned    0.188 sec / 16.640 sec
```

Query vs Query with extra Indexes

It is not always a good idea to create Indexes, because as from our above examples, we can see that the execution of an Index takes some time as well. For example in the case of Query 2, executing the Index took roughly 43.5 seconds. Which means that it would only be smart to create indexes on those columns that will actually be frequently searched against. Indexes are very beneficial in retrieving data from the specific database very fast, and they speed up the process of searching and querying.

But for speeding up the process, also reading the values without obtaining the whole row is beneficial: SELECT x ... WHERE x = 2; since the index here is smaller then also less bytes are read, which improves the performance as well.

So it is not only the ”CREATE INDEX” as in the INDEX section, but we optimized the Queries in our Query section as well in order to improve the read performance. We documented the most optimal queries in the Query section of this document.