Linnéuniversitetet
Kalmar Växjö

Linnaeus University

Faculty of Technology – Department of Computer Science

2DV513 – Database Theory

Students Name:

1. **Mohammadali Rashidfarokhi (mr223jp)**
2. **Amirhossein Soltaninejad (as225rr)**

Database Theory Assignment 2

Link to the presentation: https://youtu.be/I4GY3-gnMEI
Link to the Source code: https://gitlab.lnu.se/mr223jp/database-theory

1. Relational algebra.

- **General Justification:**

Although in the requirements, it has been pointed out that we are supposed to only write queries with relational algebra notation. We have decided also to carry it out in the MYSQL workbench to assure about our relations and support our logic. As a result, along with every relation, there is a figure attached under it for a better presentation.

- **What are the names of students enrolled in 1dv513?**

Projection (π) $_{name}$ (Selection (σ) code = "1dv513" (student $\bowtie$ enrolledIn)).

```
9  •  select name, enrolledin.code
10    from student
11    join enrolledIn on enrolledIn.id = student.id
12     where enrolledIn.code = '1dv513';
13
14
```

Result Grid | Filter Rows: | Export: | Wra

name	code
Ali	1dv513
Amir	1dv513

- ✓ **Justification:**

Based on the above relation and the picture, we will project a student's name by selecting the subject code that the student has been in enrolledIn. When it comes to joining, we have joined the student and enrolledIn relations. However, the condition is that the subject code should be equal to "1dv513".

- **What are the names of students in both 1dv513 and 2dv513?**

Projection (π) $_{name}$ (Selection (σ) code = "1dv513" (student $\bowtie$ enrolledIn)) $\cap$ Projection (π) $_{name}$ (Selection (σ) code = "2dv513" (student $\bowtie$ enrolledIn)).

```
6  select student.name
7  from student
8  join enrolledIn on student.id = enrolledIn.id
9  where enrolledIn.code = '1dv513'
10 and student.id in
11 (select student.id
12 from student join enrolledIn on student.id = enrolledIn.id
13 where enrolledIn.code = '2DV513');
14
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name
Ali
Amir

- ✓ **Justification:**

Regarding the above relation, the name (student) will be projected who is studying both 1dv513 and 2dv513 (intersection). As a result, the student and enrolledIn will be joined together once. Moving on, the condition is that the student should be enrolledIn "1dv513". Consequently, we will do the same but this time the condition is the student should be enrolledIn "2dv513". Lastly, an intersection will take a place that will enable us to find out the name of the students who are enrolledIn in both 1d513 and 2d513.

- ➢ **Who teaches 2dv610?**

Projection (π) $_{\text{lecturer}}$ (Selection (σ) code = "2dv610" (subject)).

```
8  select lecturer, code
9  from subject
10 where code = '2DV610';
11
12
13
14
```

Result Grid | Filter Rows: | Edit: | Export/Import:

lecturer	code
Jonas	2DV610
NULL	NULL

- ✓ **Justification:**

Opposite to the above relation and picture, there is no joining here. The reason is that we can project the teacher's name based on the subject code which is 2dv610. Also, the condition is that the teacher should be teaching in 2dv610 course.

- **Who teaches 1dv513 and 2dv513?**

Projection (π) $_{\text{lecturer}}$ (Selection (σ) code = “2dv513” ∩ code “1dv513” (Subject))

```
select sub1.lecturer
from subject as sub1 join
subject as sub2 on sub1.lecturer = sub2.lecturer
where sub1.code = '1dv513'
and sub2.code = '2DV513';
```

Result Grid | Filter Rows: | Export: | Wrap Cell C

lecturer
Ilir

- ✓ **Justification:**

Regarding the above relation, the intersection will take a place between two courses that are named 1dv513 and 2dv513. Moving on, the condition here is that the lecturer is required to be the teacher of the two mentioned courses. As a result, the name of the teacher will be projected at the end.

- **What are the names of students who are taking a subject not taught by Ilir?**

Projection (π) $_{\text{name}}$ (Selection (σ) lecturer != “Ilir” (student ⋈ enrolledIn ⋈ subject)).

```
select name , subject.lecturer
from student
  join enrolledIn on enrolledIn.id = student.id
  join subject on subject.code = enrolledIn.code
where subject.lecturer != 'ilir';
```

Result Grid | Filter Rows: | Export: | Wrap Cell

name	lecturer
Mohammad	Jonas
Reza	Jonas

```
select name , subject.lecturer
from student
  join enrolledIn on enrolledIn.id = student.id
  join subject on subject.code = enrolledIn.code
where subject.lecturer != 'Jonas';
```

Result Grid | Filter Rows: | Export: | Wrap Cell C

name	lecturer
Ali	Ilir
Amir	Ilir
Omid	Ilir
Javad	Ilir

- ✓ **Justification**

Considering the above figures and relation, we are selecting students who is their lecturer is not Ilir. As a result, we are joining student, enrolledIn, and subject together to be able to get the name of the students. The condition here is not having Ilir as the lecturer. Moving on, we have tested this based on the above figures. As it is clear, it will print two students whose lecturer is Jonas.

2. Relational algebra.

- Find functional dependencies.

1. Functional dependency:

(Manager, day - - > Room).

Justification:
Based on the above functional dependency, the manager is considered to use the same room in a single day.

2. Functional dependency:

(Manager, day, time - - > Applicant, room).

Justification:
Regarding the above functional dependency, it can be understood that one manager carries out an interview with one applicant.

3. Functional dependency:

(Room, time, day - - > Applicant, manager).

Justification:
Noticing the above functional dependency, a room with a particular time and day is designated for a manager along with a particular applicant.

4. Functional dependency:

(Day, applicant - - > Room, time, manager).

Justification:
An applicant is interviewed in one day. Therefore, the interview will be carried out in a designated room at a particular time along with a certain manager.

- **Find the keys of the relation.**

Based on the functional dependencies 3 and 4, there are two possible key combinations to the relation.

Keys:
1- **room**, **time**, **day**

2- **day**, **applicant**.

- **Show that the relation is in 3NF but not in BCNF.**

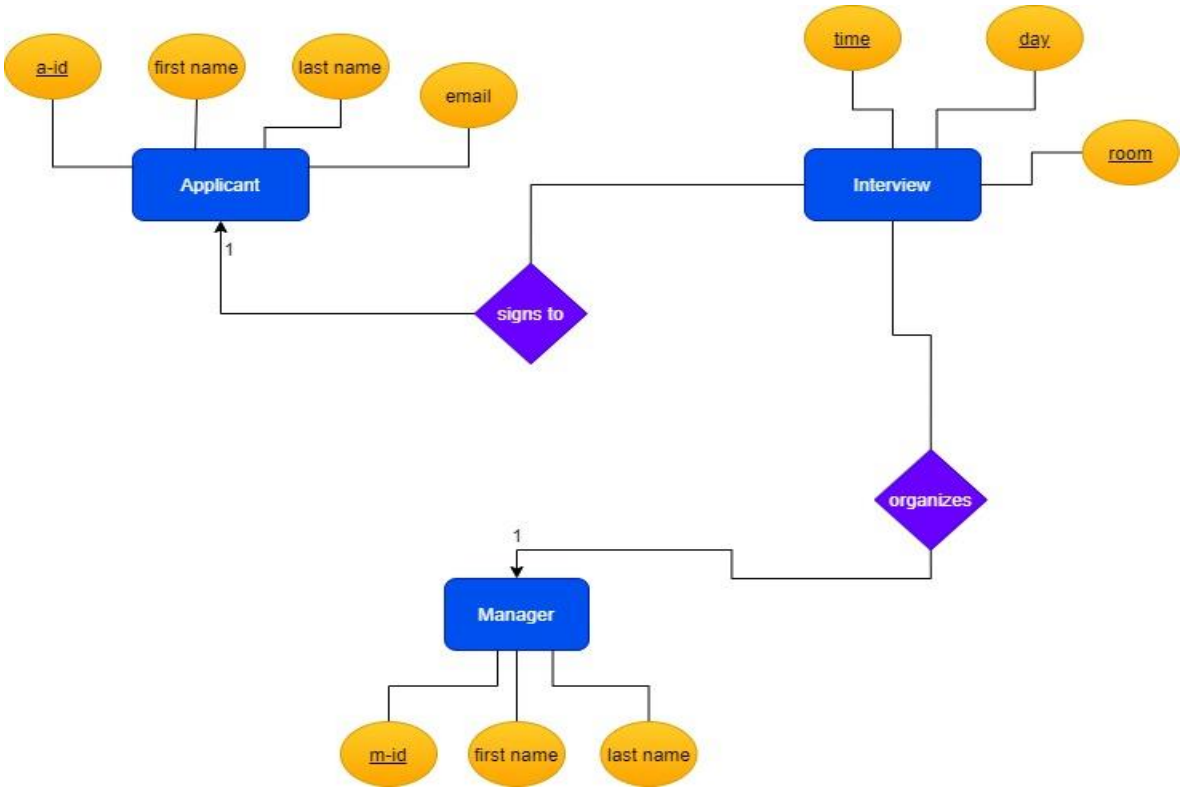
Obviously, the relation is not considered to be in BCNF. As a result, depending on the rule that the left side is required to be a super key. However, as it is clear in functional dependency 1 (manager and day), it is not a super key since it has been stated in the lectures if A is a member of some key, a relation R is in BCNF. Therefore, as it has been shown in functional dependency 1 when it comes to the room, it belongs to a key. Consequently, the relation is in 3NF rather than being in BCNF.

- **Decompose the relation in relations that are in BCNF.**

There is a possibility to create it in BCNF by depending on functional dependency 1 that violates BCNF. Also, creating two relations by taking advantage of all attributes but the one which is on the right side of functional dependency 1 and making another relation with the functional dependency 1 attributes. Therefore, it will fulfill the BCNF conditions.

As a result:
Relation 1 (applicant, manager, day, and time).
Relation 2 (manager, day, and room).

- **Draw an E/R diagram that describes the system. Try to incorporate all dependencies.**



✓ **Justification**

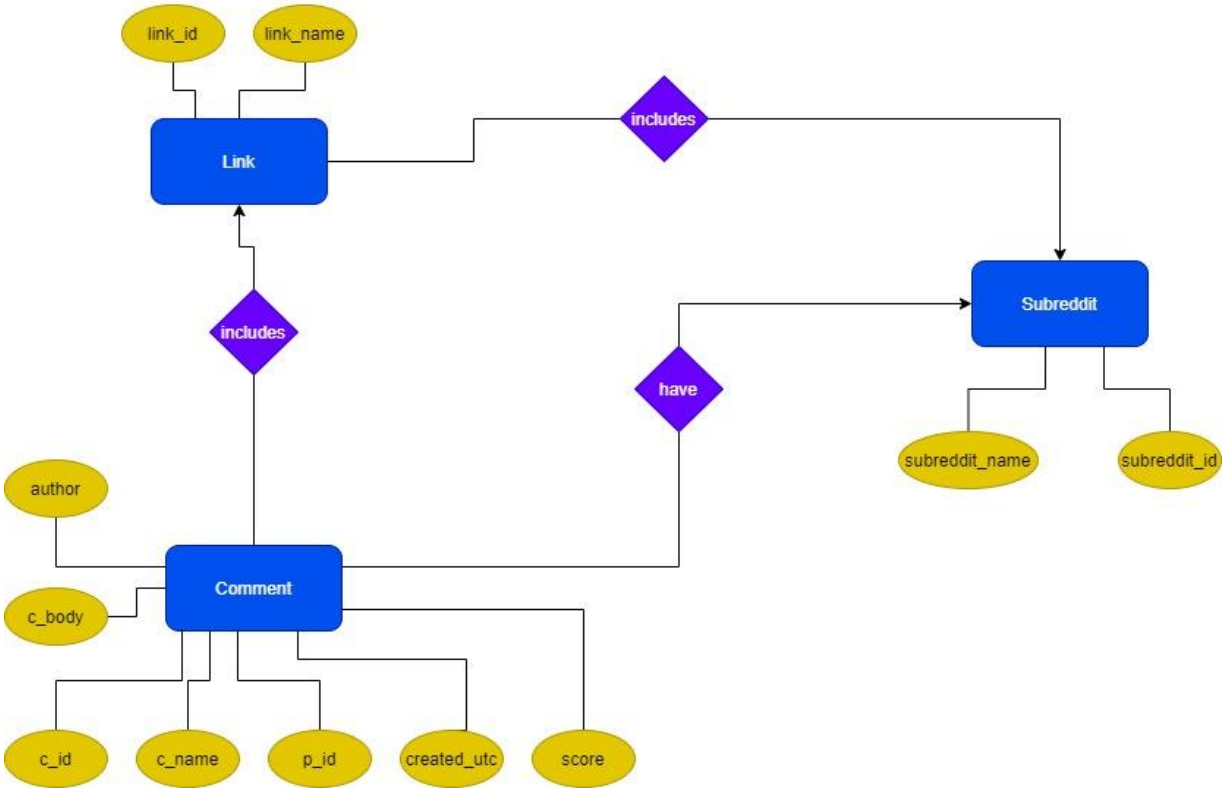
- Regarding the relation (signs to) between Interview and Applicant entity sets, it has been set as many to one. As a result, each applicant can sign to many interviews. However, each interview can be signed by at most one applicant.

- When it comes to the relation (organizes) between Manager and interview entity sets, each interview can be organized by at most one manager. On the other hand, each manager can organize many interviews. The reason is based on the following statements from the requirements where it has said that interviews with job applicants can be carried out by several managers. Also, there is a chance that each applicant is interviewed at different timings with dissimilar managers.

- Looking at the Interview entity set, this entity set has been assigned 3 attributes which are room, day, and time. As a result, by having these attributes, you can track the interview. However, we have set them as keys. The reason is that, during a day, there can be many interviews but when we have these 3 attributes as keys, we can find out about each specific interview.

3. Setting up the Reddit database.

◆ **E/R diagram:**



- ✓ **Justification:**

- Regarding the relation (includes) between link and Subreddit entity sets; it has been set as many to one. Each subreddit can include many links. However, links are included in one subreddits.

- Concerning the relation (includes) between link and comment entity sets, it is many to one Therefore, and each link can include many comments. However, comments are included in one link.

- Considering Comment and subreddit entity sets, the relation (have) is many to one. Consequently, each subreddit can include many comments. However, comments are included in one subreddit.

◆ Schemas:

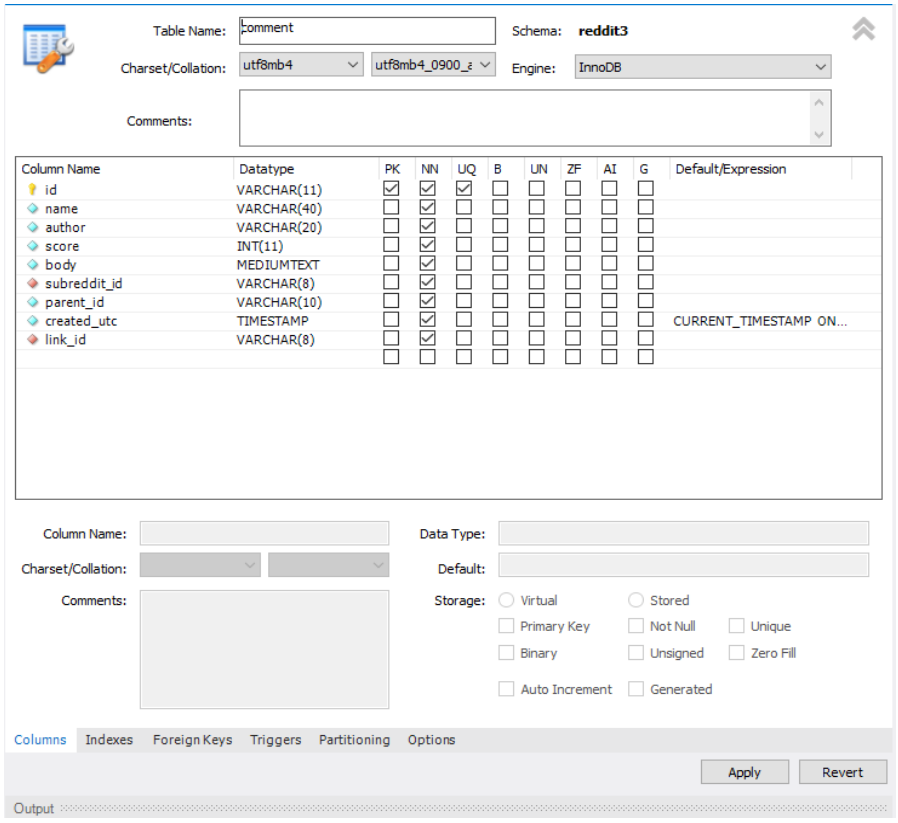


Table Name: comment
Schema: reddit3
Charset/Collation: utf8mb4 utf8mb4_0900_a
Engine: InnoDB
Comments:
Column Name | Datatype | PK | NN | UQ | B | UN | ZF | AI | G | Default/Expression
id | VARCHAR(11)
name | VARCHAR(40)
author | VARCHAR(20)
score | INT(11)
body | MEDIUMTEXT
subreddit_id | VARCHAR(8)
parent_id | VARCHAR(10)
created_utc | TIMESTAMP | CURRENT_TIMESTAMP ON...
link_id | VARCHAR(8)
Column Name:
Data Type:
Charset/Collation:
Default:
Comments:
Storage: Virtual Stored
Primary Key Not Null Unique
Binary Unsigned Zero Fill
Auto Increment Generated
Columns Indexes Foreign Keys Triggers Partitioning Options
Apply Revert
Output

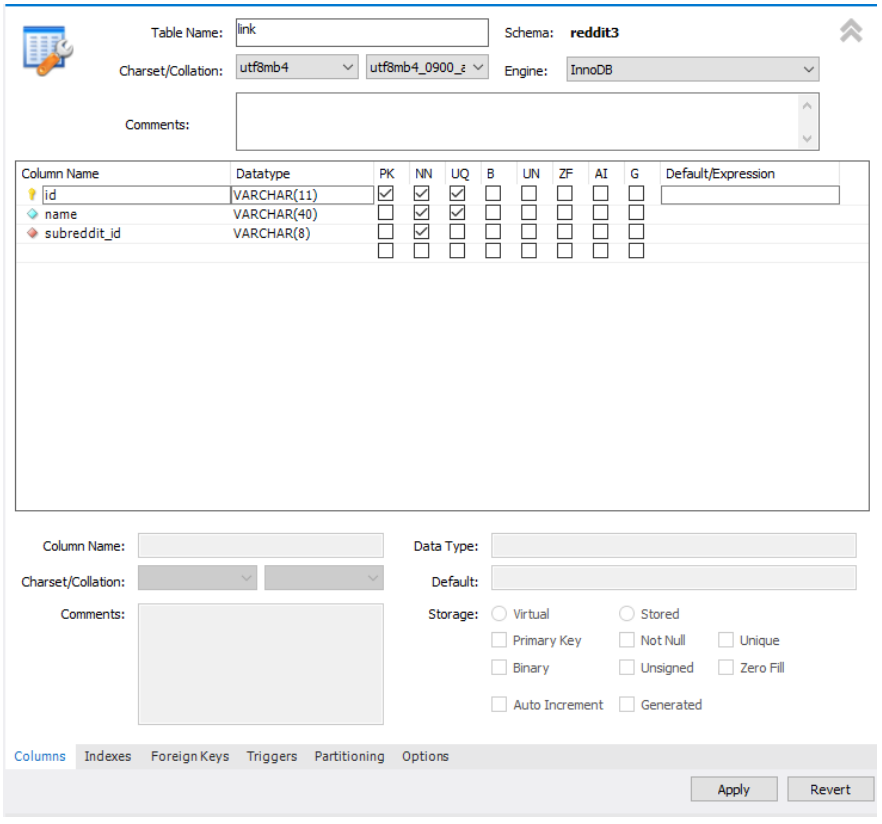
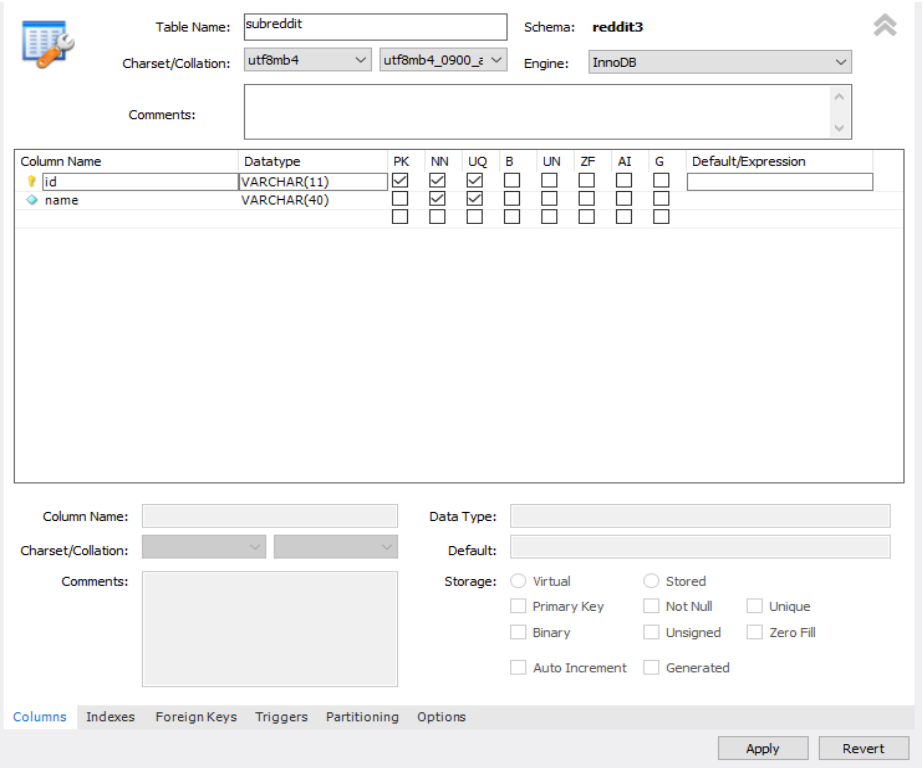


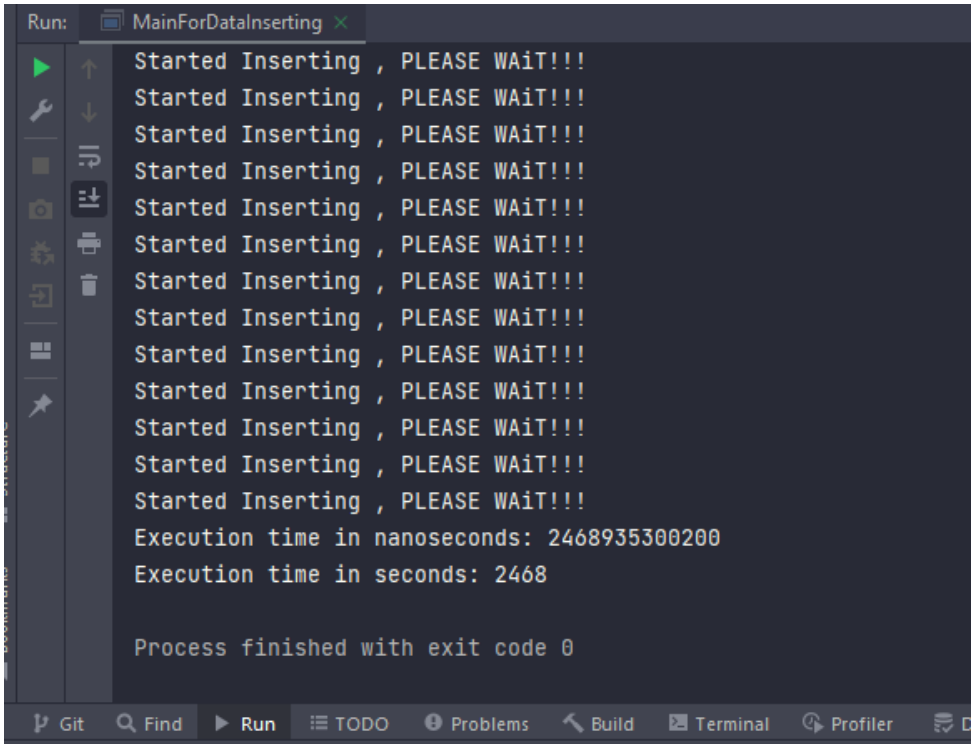
Table Name: link
Schema: reddit3
Charset/Collation: utf8mb4 utf8mb4_0900_a
Engine: InnoDB
Comments:
Column Name | Datatype | PK | NN | UQ | B | UN | ZF | AI | G | Default/Expression
id | VARCHAR(11)
name | VARCHAR(40)
subreddit_id | VARCHAR(8)
Column Name:
Data Type:
Charset/Collation:
Default:
Comments:
Storage: Virtual Stored
Primary Key Not Null Unique
Binary Unsigned Zero Fill
Auto Increment Generated
Columns Indexes Foreign Keys Triggers Partitioning Options
Apply Revert



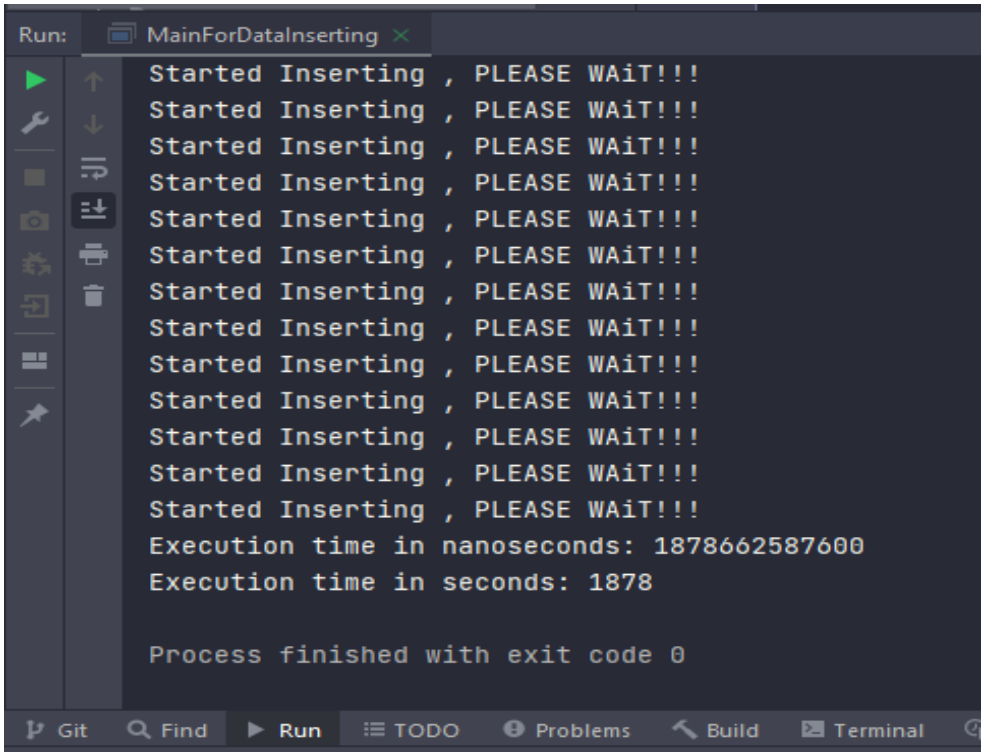
4. Importing the data.

- PC1

Running time with constraint:

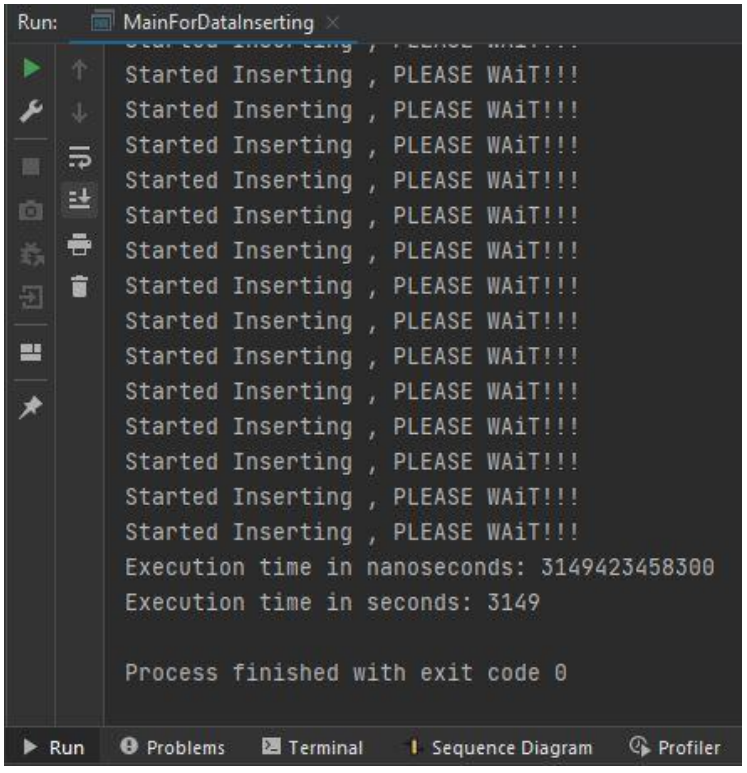


Running time without constraint:

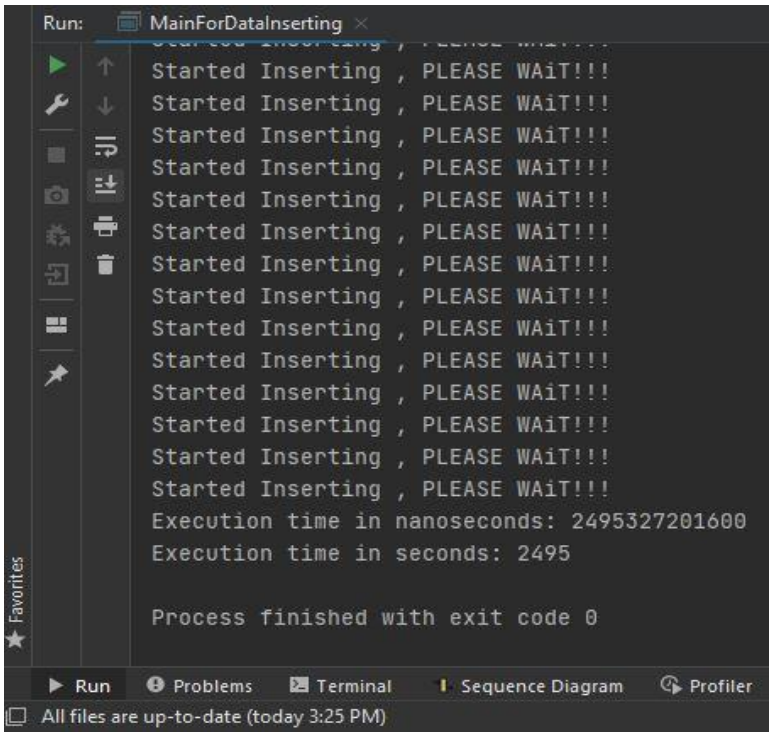


- PC2

Running time with constraint:



Running time without constraint:



- **Justification:**

Based on the above pictures, we have captured the running time with constraints and without constraints in two different PC. The reason we have decided to take advantage of two PCS is that we wanted to figure out how the performance will be when using two different systems (PC) with different configurations especially the memory.

1. **How do these constraints affect the import time?**

When it comes to inserting data to the database with full constraints, it has taken a long time to insert the data into the tables; on the other hand, when inserting data to the database without constraints, it has taken a shorter time inserting compared to the time when we have used constraints.

2. **Report measured times and discuss why you think the constraints affected the times?**

Based on the results that were captured by PC1, the running time with constraints has been set as 2468 seconds. Moving on, it took 1878 seconds for PC1 to capture the running time without constraints. As it has been mentioned before we have captured the times by two different systems. consequently, PC2 has provided us with times 3149 seconds for constraints and 2495 seconds for without constraints.

The constraints have affected the times anyway. Although we have used the HashSet to store the duplicate entries and double-check them with the new entries and the reason for that was to prevent the duplication of the entries. However, the time of inserting data without constraints is still faster since it will insert all data with or without duplication and without checking the data.

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

Due to the reason that there are no constraints in our database, an enormous number of duplications and flaws data can be inserted into the database. By omitting the data that were considered to have duplications and flaws it will be time-consuming. It will be costly when it comes to correcting the flaws and checking for duplications. Apart from that, by inserting flaws and duplication, it can occupy a huge amount of space in our database. As a result, it will lower the efficiency of the system and waste a great deal of space while that space can be used for other data in the database. Therefore, it is surely more proper to have the constraints turned on.

5. SQL queries.

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

```
141     -- Question 1
142     SELECT COUNT(name) AS "Number of Comments"
143     FROM Comment
144     WHERE author = "gigaquack";
```

✓ **Justification:**

Based on the above query, the calculated number of comments from a specific author will be chosen by counting the names.

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

```
143     -- Question2
144 •   SELECT
145     DATE(created_utc) as Date_Of_Comments,
146     COUNT(id) as Number_Of_Comments
147     FROM Comment
148     WHERE subreddit_id = "t5_2cneq"
149     GROUP BY Date_Of_Comments;
```

Result Grid | Filter Rows: | Export:

Date_Of_Comments	Number_Of_Comments
2007-10-14	747
2007-10-15	1524
2007-10-16	1992
2007-10-17	2205
2007-10-18	1942
2007-10-19	1628
2007-10-20	1988
2007-10-21	2446
2007-10-22	2269
2007-10-23	2302
2007-10-24	3045
2007-10-25	2409

Result 55 ×

Output

✓ **Justification:**

Based on the above query, the selection will take a place from date from the column “created_utc” and we will calculate the number of the comments based on the ids. Moving on, we will group them based on the date of the comments for a specific “subreddit_id”.

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

```
107
108     -- Question3
109 •   SELECT
110     COUNT(*) AS "Number of lol's"
111     FROM comment
112     WHERE body LIKE "%lol%";
113
```

Result Grid | Filter Rows:

Number of lol's
759

✓ **Justification:**

The above query will demonstrate the process of counting all the comments where in their body it the word “lol” can be detected.

4. Users that commented on a specific link has also posted to which subreddits?

```
92    -- Question4
93    SELECT name AS Subreddit_Name FROM subreddit
94    WHERE id IN ( SELECT subreddit_id
95    FROM comment WHERE author IN ( SELECT author FROM
96    comment WHERE link_id = "5ybb9"));
```

Result Grid | Filter Rows: | Export: | Wrap Cell Conten

Subreddit_Name
ads
bugs
business
de
entertainment
features
fr
freeculture
gadgets
gaming
id
it

subreddit 58

✓ **Justification:**

Based on the link “5ybb9”, initially the query will choose the author’s comment based on the mentioned link. Moving on, the next selection will be based on the “subreddit_id” where the comments of the authors can be distinguished. Lastly, the names of the subreddit based on the subreddit table where it takes advantage of its id will be returned.

5. **Which users have the highest and lowest combined scores? (Combined as the sum of all scores)**

```
71    -- Question5
72    CREATE VIEW Total_scores AS
73    SELECT Distinct SUM(score) AS Sum_Score , author
74    FROM comment
75    GROUP BY author;
76    SELECT MIN(Sum_Score) AS Lowest_Score,
77    MAX(Sum_Score) AS Highest_Score
78    FROM Total_scores;
79    DROP VIEW Total_scores;
80
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Lowest_Score	Highest_Score
-369	109468

✓ **Justification:**

Based on the above query, initially, the selection will take a place based on the distinct sum of each score and the author itself. Moving on, storing of the results will be set as "Total_scores". Consequently, based on the "Total_scores", we will choose the lowest and highest scores. Lastly, by the time of finishing the query, results will be omitted.

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

```
45    -- Question6
46    SELECT "Lowest Score" AS result , name
47    FROM Subreddit
48    WHERE id = ( SELECT subreddit_id FROM comment WHERE score =
49    ( SELECT MIN(score) FROM comment))
50    UNION
51    SELECT "Highest Score" AS result, name
52    FROM Subreddit
53    WHERE id = ( SELECT subreddit_id FROM comment WHERE score =
54    ( SELECT MAX(score) FROM comment));
55
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

result	name
Lowest Score	politics
Highest Score	reddit.com

✓ **Justification:**

The above query will result in finding out the highest and lowest scored comments and the union between the highest and lowest scores will be returned eventually.

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).

```
25    -- Question 7
26    SELECT DISTINCT author
27    FROM comment
28    WHERE link_id IN ( SELECT
29    link_id FROM comment
30     WHERE author = "gigaquack")
31     AND author!= "gigaquack";
32
```

Result Grid | Filter Rows: | Export:

author
bostich
igiveyoumylife
Arve
[deleted]
Percept
mcm69
deki
raldi
jezmck
sickofthisshit
hfaber
paternoster
zoomzoom83
tashbarg
newton_dave
lyon

comment 41 ×

Output

✓ **Justification:**

Based on the above query, the selection will cover all the "link_id" from a specific author. Moving on, it will choose the rest of others based on the "link_id" excluding the given user.

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

```
6
7   -- Question 8
8   SELECT
9   COUNT(DISTINCT subreddit_id),
10  author
11  FROM comment
12  GROUP BY author
13  HAVING COUNT(subreddit_id) = 1;
14
```

Result Grid | Filter Rows: | Export: | Wrap Cell Con

COUNT(DISTINCT subreddit_id)	author
1	--GOD--
1	00sweeney
1	00u6166
1	02J
1	0xdefec8
1	128bit
1	13aces
1	16pac
1	1776patriot
1	1800123456
1	1812overture
1	1amgreat
1	1way2find
1	1weblord
1	2-pac

Result 39

✓ **Justification:**

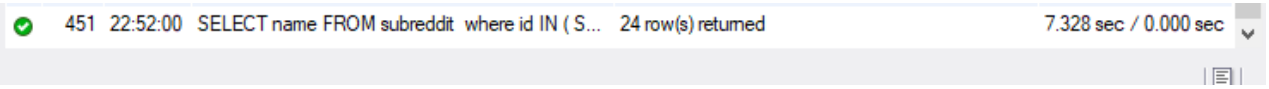
The above query will indicate that the initial selection is based on count-specific "subreddit_id" and the author. The return will take a place where the distinct "subreddit_id" is one.

◆ **Report the queries you found to work best, together with a brief motivation for why you think that worked best:**

Regarding the queries that perform the best, we have noticed that most of the queries took less than 500ms to run. The reason is that they were required less combination at the time of creating the query such as query number 3 where it took 0.391ms.

490 16:44:32 SELECT COUNT(*) AS "Number of lol's" FRO... 1 row(s) returned 0.391 sec / 0.000 sec

On the other hand, query number 4 has taken a long time to run with an execution time of 7.328 seconds. The reason that it has taken a long time to be executed is that the query should traverse the table's columns two times. Therefore, it is required to iterate over the whole entries. That is why it is time-consuming compared to the others. There is a possibility of reducing the query time by altering the query to work based on the indexes.



Assignment 2 Run instructions:

1. Please put the Java codes, db.properties and (desired JSON files) into the project named "Assignment2". Java codes and db.properties are available in the src folder.

2. Run the SQL file named "Create database (Initial setup).sql" available on the SQL folder in any MySQL space like workbench or PHPMyAdmin to create the initial database.

3. Add the libraries provided in the "Libraries" folder to the library class path of your IDE.
`

4. Two Java "Main" file is available for creating the database with Constraint and without Constraint.

5. run one of the above mentioned Java Files ("MainForCreatingDatabaseWithConstraint","MainForCreatingDatabaseWithoutConstraint") to create the database with the needed tables.

6. To insert data please run the Java file named "MainForDataInserting", there are three JSON files available which can be chosen for inserting by commenting or uncommenting them.
Note: the "RC_2007-10.json" is initially uncommented so if you like to insert the other files, please uncomment them.