SOEN 363 - Database Systems for Software Engineers
Report - Project Phase 2

Ali Hanni - 40157164
Marie-Eve Hazari - 40156408
Beshoy Soliman - 40047115
Alireza Ziarizi - 40027914

Gina Cody School of Computer Science and Software Engineering
Concordia University, Montreal, QC, Canada

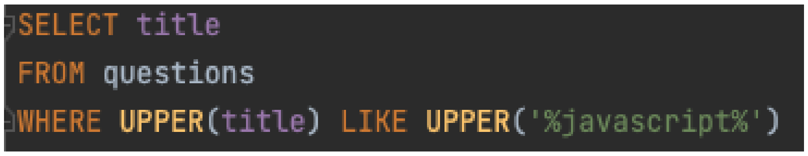
Winter 2022

Analyzing Big Data Using SQL and RDBMS Systems

Query 1

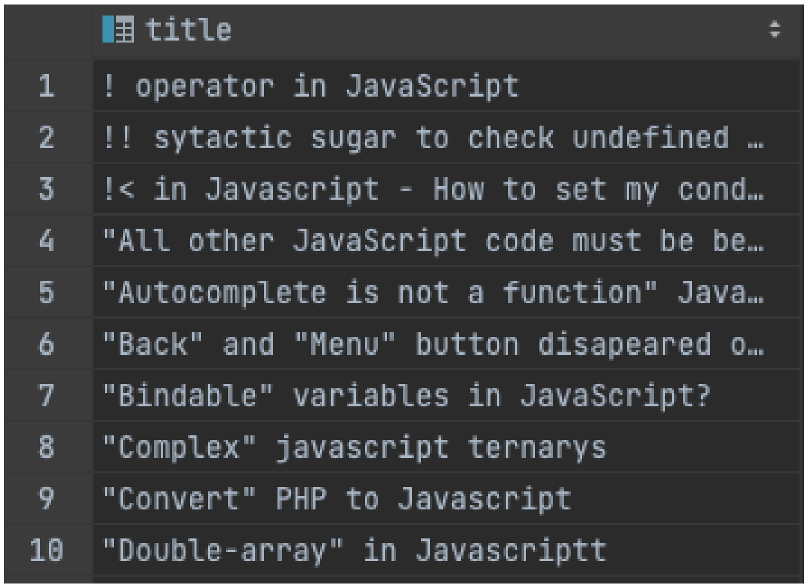
For the first query, we wanted to take a look at the kind of questions involving JavaScript in order to try and catch any trend. To do that, we displayed all questions which contained 'JavaScript' (not case sensitive).

Query:



```sql
SELECT title
FROM questions
WHERE UPPER(title) LIKE UPPER('%javascript%')
```

Result:



	title
1	! operator in JavaScript
2	!! sytactic sugar to check undefined …
3	!< in Javascript - How to set my cond…
4	"All other JavaScript code must be be…
5	"Autocomplete is not a function" Java…
6	"Back" and "Menu" button disapeared o…
7	"Bindable" variables in JavaScript?
8	"Complex" javascript ternarys
9	"Convert" PHP to Javascript
10	"Double-array" in Javascriptt

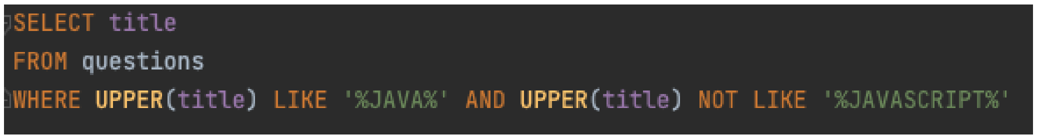
Indexing

Without any index, the execution time averaged 461.784ms. We added an index for `title`. After adding that index, the average execution time for the query decreased to 367.477ms. The execution time decreased by 20.42% after adding the index.

Query 2

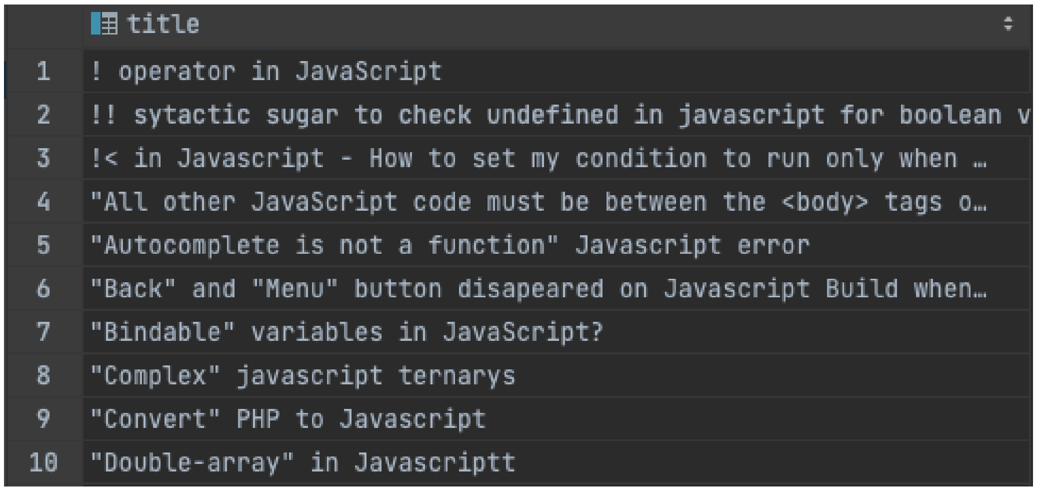
The second query aims to find all questions whose title contains Java, but not JavaScript. A title that contains both must be excluded from the result.

Query:



```sql
SELECT title
FROM questions
WHERE UPPER(title) LIKE '%JAVA%' AND UPPER(title) NOT LIKE '%JAVASCRIPT%'
```

Result:



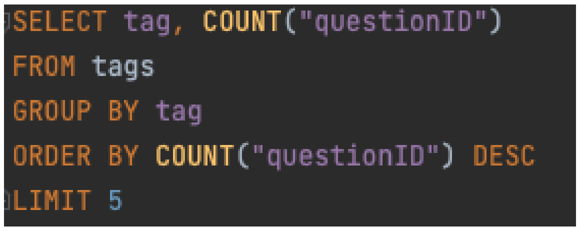
	title
1	! operator in JavaScript
2	!! sytactic sugar to check undefined in javascript for boolean v
3	!< in Javascript - How to set my condition to run only when …
4	"All other JavaScript code must be between the <body> tags o…
5	"Autocomplete is not a function" Javascript error
6	"Back" and "Menu" button disapeared on Javascript Build when…
7	"Bindable" variables in JavaScript?
8	"Complex" javascript ternarys
9	"Convert" PHP to Javascript
10	"Double-array" in Javascriptt

Indexing
Without any index, the execution time averaged 439.408ms. We added an index for `title`. After adding that index, the average execution time for the query decreased to 402.354ms. The execution time decreased by 8.43% after adding the index.

Query 3

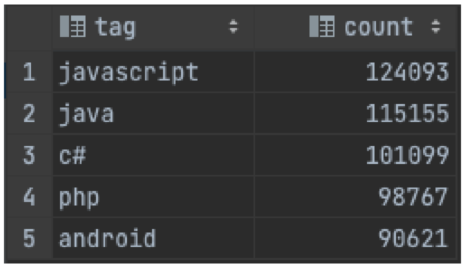
The third query aims to find the 5 most used tags across all questions. We want to see if either JavaScript or Java appears among them. We also want to have a better idea of what other popular tags are.

Query:



```
SELECT tag, COUNT("questionID")
FROM tags
GROUP BY tag
ORDER BY COUNT("questionID") DESC
LIMIT 5
```

Result:



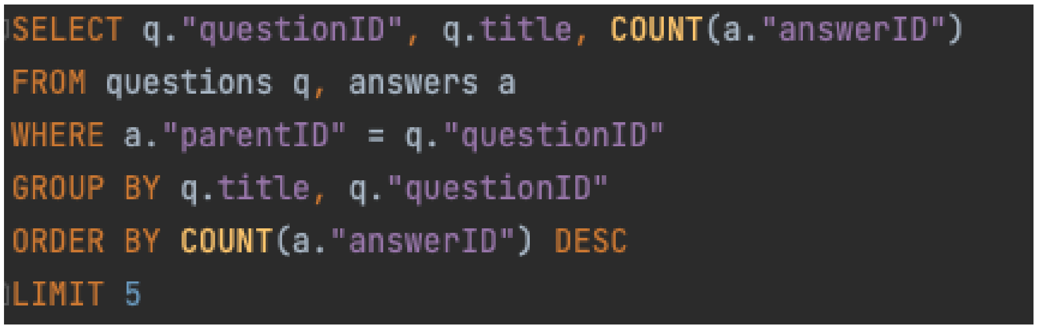
	tag	count
1	javascript	124093
2	java	115155
3	c#	101099
4	php	98767
5	android	90621

Indexing
Without any index, the execution time averaged 875.623ms. We added an index for `questionID`. After adding that index, the average execution time for the query decreased to 824.371ms. The execution time decreased by 5.85% after adding the index. Although the decrease may seem very small, especially if we only look at execution time, a 5.85% decrease in a very long query can save a lot of time.

Query 4

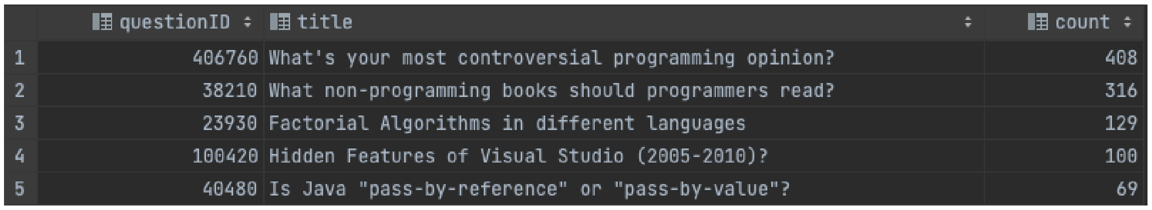
Finally, as we all know, StackOverflow is one if not the biggest programming forum. A lot of programmers use it to find an answer to their questions everyday. Similarly, a lot of highly competent programmers answer questions and give their input about different subjects. It would be interesting to know what are the questions that generate the most interest among the community. To do so, we write a query to find the 5 questions who generated the most attention on the platform. We measure the popularity of a question by the number of answers it got.

Query:



```
SELECT q."questionID", q.title, COUNT(a."answerID")
FROM questions q, answers a
WHERE a."parentID" = q."questionID"
GROUP BY q.title, q."questionID"
ORDER BY COUNT(a."answerID") DESC
LIMIT 5
```

Result:

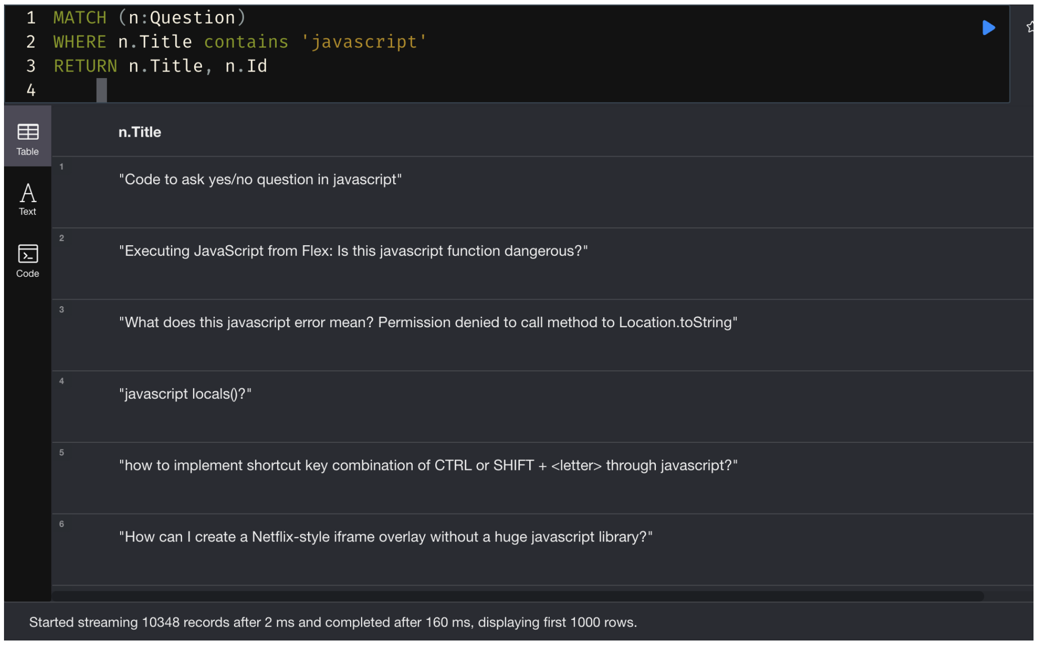


	questionID	title	count
1	406760	What's your most controversial programming opinion?	408
2	38210	What non-programming books should programmers read?	316
3	23930	Factorial Algorithms in different languages	129
4	100420	Hidden Features of Visual Studio (2005-2010)?	100
5	40480	Is Java "pass-by-reference" or "pass-by-value"?	69

Indexing
Without any index, the average execution time for this query was around 4992.462ms. We added 3 indexes,one for `questionID`, one for `answerID`, and one for `parentID`. After adding the indexes, the average execution time decreased to 4040.185ms. The execution time decrease by 19.07%.

Analyzing Big Data Using NoSQL systems

Query 1



```
MATCH (n:Question)
WHERE n.Title contains 'javascript'
RETURN n.Title, n.Id
```

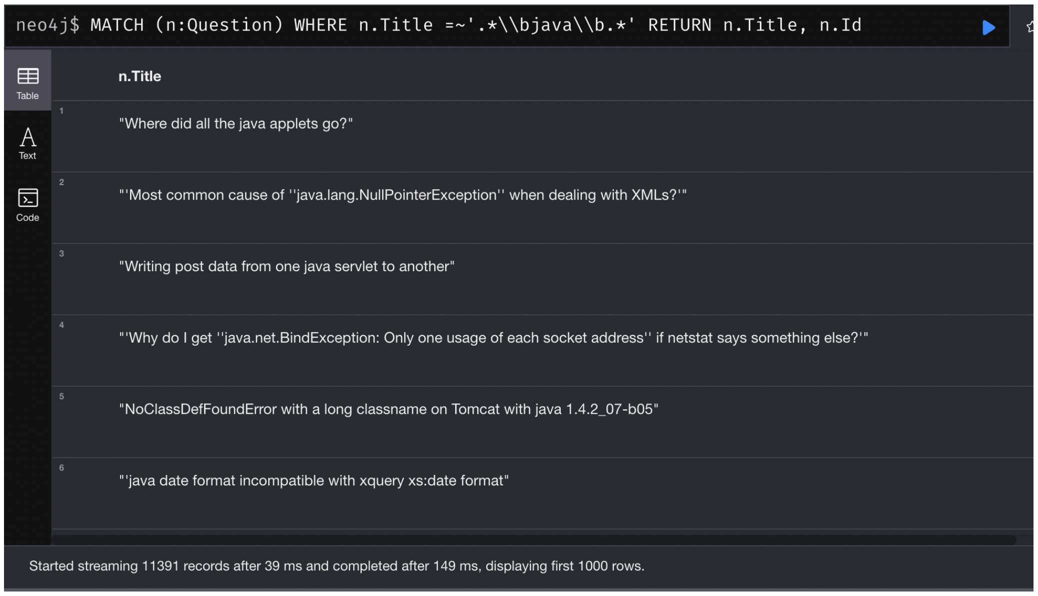
	n.Title
1	"Code to ask yes/no question in javascript"
2	"Executing JavaScript from Flex: Is this javascript function dangerous?"
3	"What does this javascript error mean? Permission denied to call method to Location.toString"
4	"javascript locals()?"
5	"how to implement shortcut key combination of CTRL or SHIFT + <letter> through javascript?"
6	"How can I create a Netflix-style iframe overlay without a huge javascript library?"

Started streaming 10348 records after 2 ms and completed after 160 ms, displaying first 1000 rows.

Indexing

Without any index, the execution time averaged 160ms. After adding that index, the average execution time for the query decreased to 13ms. The execution time decreased by 91.88% after adding the index. This is the most important decrease in execution time observed.

Query 2



```
neo4j$ MATCH (n:Question) WHERE n.Title =~'.*\\bjava\\b.*' RETURN n.Title, n.Id
```

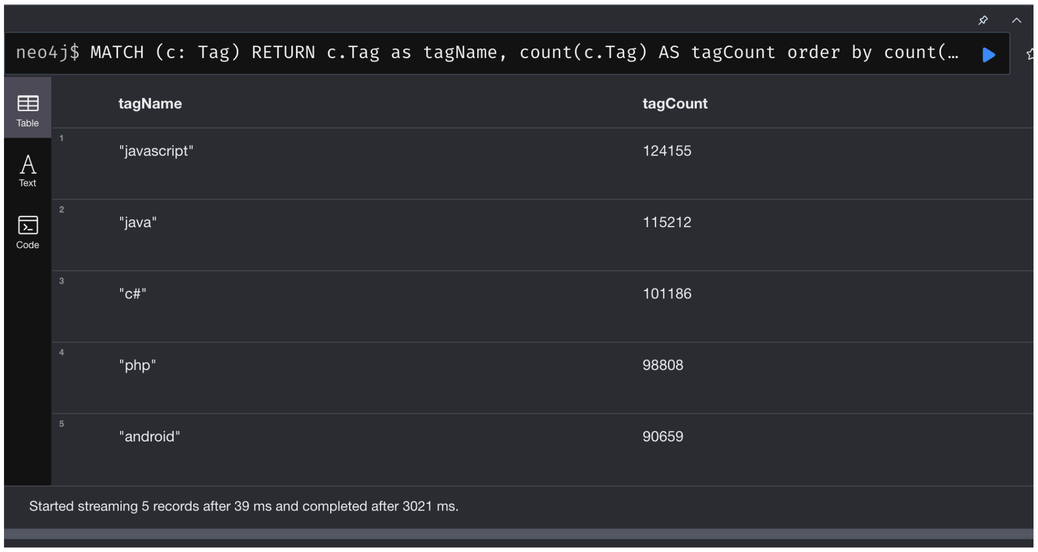
	n.Title
1	"Where did all the java applets go?"
2	"'Most common cause of ''java.lang.NullPointerException'' when dealing with XMLs?'"
3	"Writing post data from one java servlet to another"
4	"'Why do I get ''java.net.BindException: Only one usage of each socket address'' if netstat says something else?'"
5	"NoClassDefFoundError with a long classname on Tomcat with java 1.4.2_07-b05"
6	"'java date format incompatible with xquery xs:date format"

Started streaming 11391 records after 39 ms and completed after 149 ms, displaying first 1000 rows.

Indexing

Without any index, the execution time averaged 149ms. After adding that index, the average execution time for the query decreased to 74ms. The execution time decreased by 50.34% after adding the index. Again, this is the second most important decrease in execution time observed.

Query 3



```
neo4j$ MATCH (c: Tag) RETURN c.Tag as tagName, count(c.Tag) AS tagCount order by count(…
```

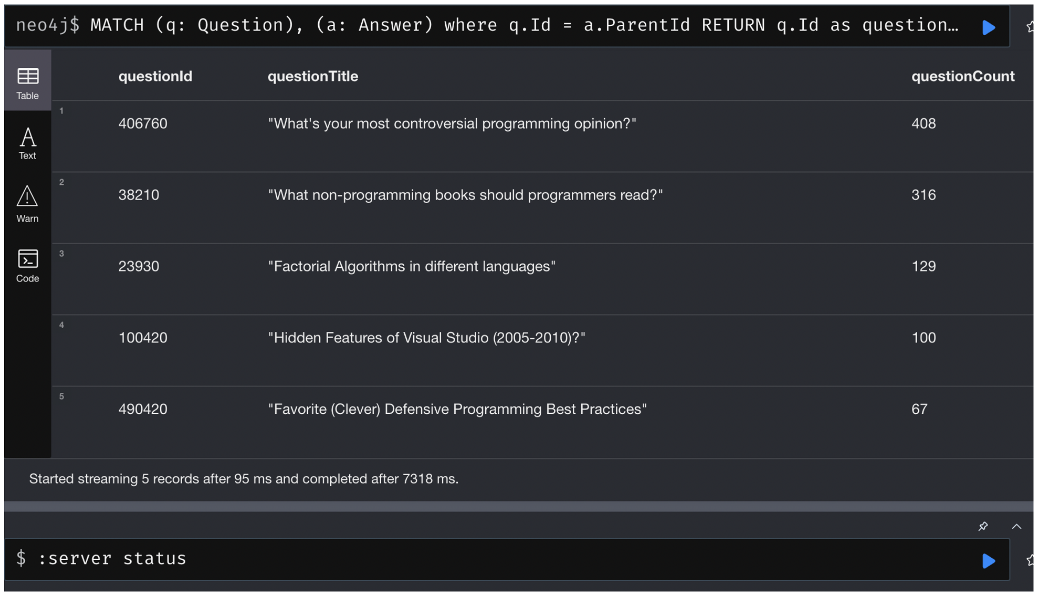
	tagName	tagCount
1	"javascript"	124155
2	"java"	115212
3	"c#"	101186
4	"php"	98808
5	"android"	90659

```
Started streaming 5 records after 39 ms and completed after 3021 ms.
```

Indexing

Without any index, the execution time averaged 3021ms. After adding that index, the average execution time for the query decreased to 2697ms. The execution time decreased by 10.72% after adding the index. It is possible to notice that the decrease in execution time caused by indexing is insignificant here compared to the previous two queries.

Query 4



```
neo4j$ MATCH (q: Question), (a: Answer) where q.Id = a.ParentId RETURN q.Id as question…
```

```
  questionId   questionTitle                                                questionCount
1 406760       "What's your most controversial programming opinion?"        408
2 38210        "What non-programming books should programmers read?"        316
3 23930        "Factorial Algorithms in different languages"                129
4 100420       "Hidden Features of Visual Studio (2005-2010)?"              100
5 490420       "Favorite (Clever) Defensive Programming Best Practices"     67

Started streaming 5 records after 95 ms and completed after 7318 ms.
```

```
$ :server status
```

Indexing

Without any index, the execution time averaged 7318ms. After adding that index, the average execution time for the query decreased to 6989ms. The execution time decreased by 4.50% after adding the index. Once again, the reduction in execution time here is very small compared to the first two queries.

Analysis

First, we realize that the two first queries have an execution time that is way shorter when using Neo4j compared to PostgreSQL. We also notice that the decrease in execution time induced by indexes is significantly more important for Neo4j queries compared to PostgreSQL queries, particularly for the first two queries.
This trend do not hold to the last two queries where Neo4j execution time is generally higher than the execution time of the same queries using PostgreSQL.
It is also possible to notice that the decrease in execution time due to indexing is smaller but more homogeneous for PostgreSQL systems with an average decrease of 20.42%, 8.43%, 5.85%, and 19.07% for all four queries respectively. This is not the case for Neo4j queries where major discrepancies can be observed. Indeed, while the first two queries' execution time decrease by 91.88% and 50.34% respectively, a decrease of only 10.62% and 4.50% is observed for the last two queries.
This can be explained by the nature of the queries as well as the nature of the needed indexes. For example, we know that PostgreSQL and most Database management systems automatically create an index for the primary key and constraints. Because of that and because of the nature of our analysis, we will note notice any tangible change in execution time if the index we manually added concerns the primary key of the table (for example questionID).