StackSample

Stackoverflow Questions & Answers

SOEN 363 - Data Systems for Software Engineers

Presented by

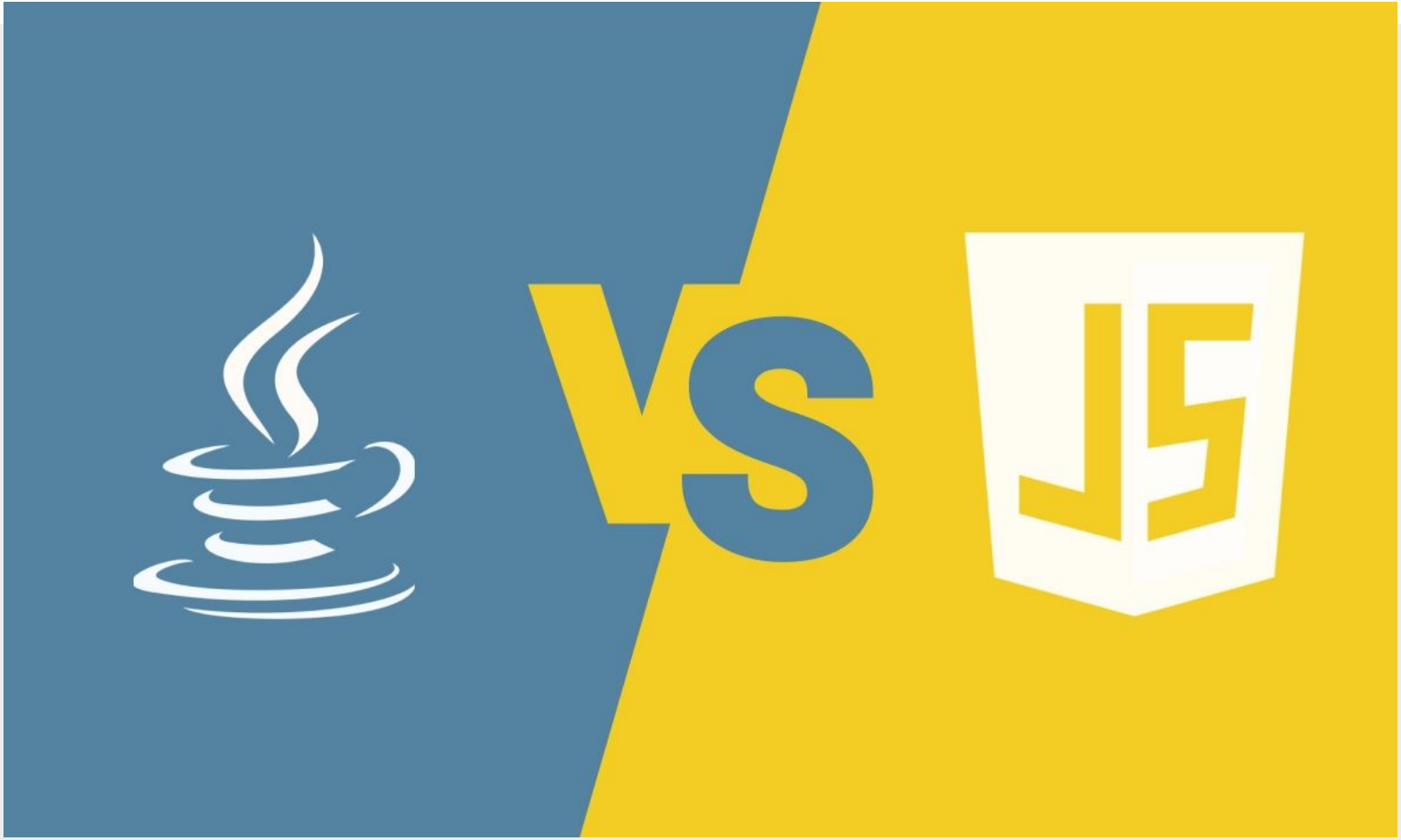
Ali Hanni - 40157164
Marie-Eve Hazari - 40156408
Beshoy Soliman - 40047115
Alireza Ziarizi - 40027914

About the Dataset

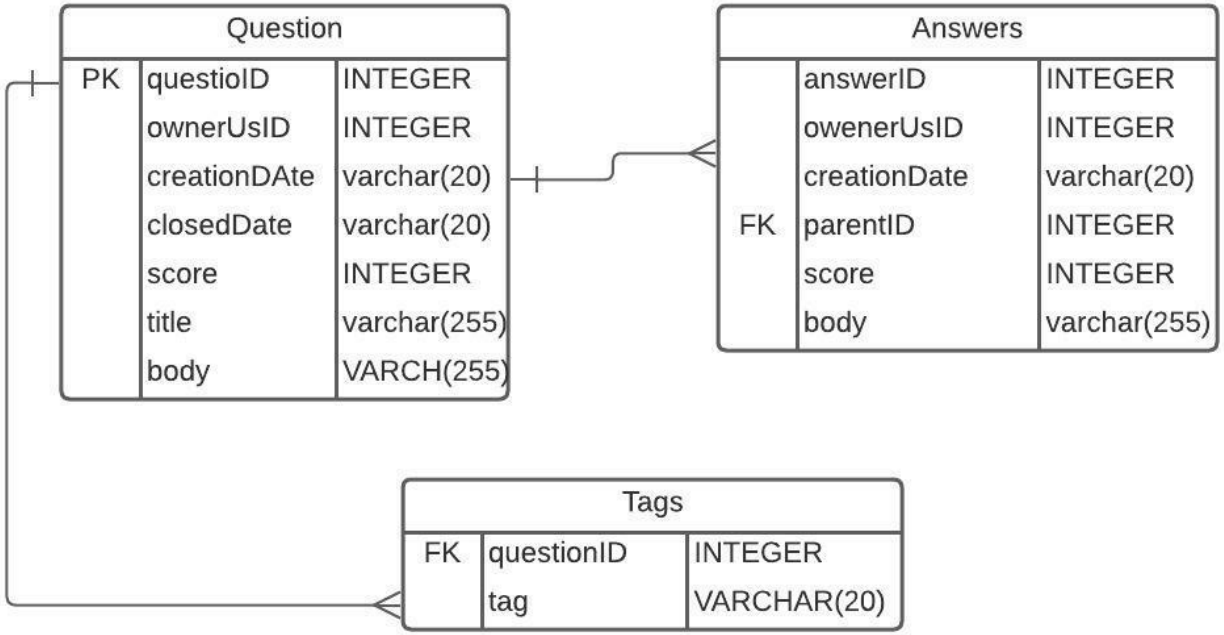
- Contains 10% of all programming questions and their answers found on Stackoverflow.
- 3 files :
    - ➢ Questions.csv (1.92GB)
    - ➢ Answers.csv (1.61GB)
    - ➢ Tags.csv (65.5MB)
- We use PostgreSQL as RDBMS and Neo4J for NoSQL system

Data loading

- Postgres:
  - Create schema
  - directly import csv no issue was found
- Neo4J:
  - Has to refactor the csv files
  - Changed “ to ‘


VS
JS

ER Diagram


Question
PK
questioID
INTEGER
ownerUsID
INTEGER
creationDAte
varchar(20)
closedDate
varchar(20)
score
INTEGER
title
varchar(255)
body
VARCH(255)
Answers
answerID
INTEGER
owenerUsID
INTEGER
creationDate
varchar(20)
FK
parentID
INTEGER
score
INTEGER
body
varchar(255)
Tags
FK
questionID
INTEGER
tag
VARCHAR(20)

PostgreSQL

Creating Tables

```
CREATE TABLE Tags(
    "questionID" INTEGER,
    "tag" VARCHAR(20),
    FOREIGN KEY ("questionID") REFERENCES questions ("questionID")
);
```

```
CREATE TABLE answers(
 "answerID" INTEGER,
 "ownerUsID" INTEGER,
 "creationDate" VARCHAR(20),
 "parentID" INTEGER,
 "score" INTEGER,
 "body" VARCHAR(255),
 FOREIGN KEY ("parentID") REFERENCES questions ("questionID")
);
```

```
CREATE TABLE questions (
     "questionID" INTEGER,
     "ownerUsID" INTEGER,
     "creationDate" VARCHAR(20),
     "closedDate" VARCHAR(20),
     "score" INTEGER,
     "title" VARCHAR(255),
     "body" VARCHAR(255),
      PRIMARY KEY ("questionID")
)
```

Query 1

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

- First, 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)
- Index created for ‘title’ field.
- Execution time without index: 461.784ms.
- Execution time with index: 367.477ms.
- Decreased execution time of 20.42%.

Query 2

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

- Searching for questions whose title contains ‘Java’ but not ‘Javascript’
- Index created for ‘title’ field.
- Execution time without index: 439.408ms.
- Execution time with index: 402.354ms.
- Decreased execution time of 8.43%.

Query 3

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

- Query to get the 5 tags that are most often used.

- Index created for ‘questionID’ field.
- Execution time without index: 875.623 ms.
- Execution time with index: 824.371 ms.
- Decreased execution time of 5.85%.

Query 4

```
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
```

- Query to get the 5 questions who generated the most attention (highest number of answers).

- Index created for question table's 'questionID' field and answers table's 'answerID' and 'parentID' fields.
- Execution time without index: 4992.462 ms.
- Execution time with index: 4040.185 ms.
- Decreased execution time of 19.07%.

Neo4j

Creating Tables

Create Tags Table

```
load csv with headers from "file:///Tags.csv" as row create (t:Tag) set
t.Tag=row.Tag, t.Id=toInteger(row.Id)
```

Added 3750994 labels, created 3750994 nodes, set 7501988 properties, completed after 18058 ms.

Create Answers Table

```
load csv with headers from "file:///Answers.csv" as row create (a:Answer) set
a.ParentId=toInteger(row.ParentId), a.CreationDate=row.CreationDate,
a.Score=toInteger(row.Score), a.Id=toInteger(row.Id)
```

Added 1048575 labels, created 1048575 nodes, set 4194300 properties, completed after 6912 ms.

Create Questions Table

```
load csv with headers from "file:///Questions.csv" as row create (q:Question
set q.CreationDate=row.CreationDate, q.Score=toInteger(row.score),
q.OwnerUserId=toInteger(row.OwnerUserId), q.Title=row.Title,
q.Id=toInteger(row.Id), q.ClosedDate=row.ClosedDate
```

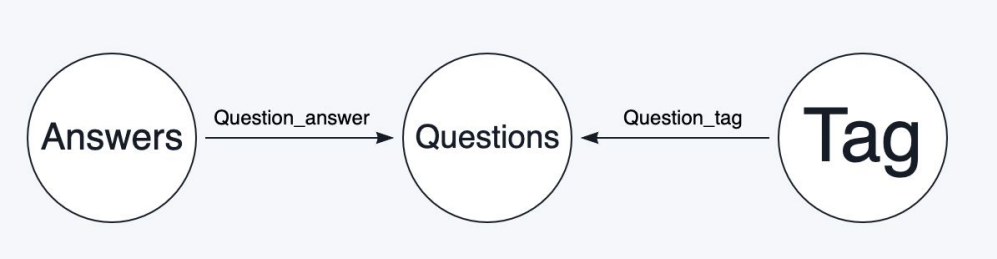
Added 1049308 labels, created 1049308 nodes, set 6295848 properties, completed after 12336 ms.

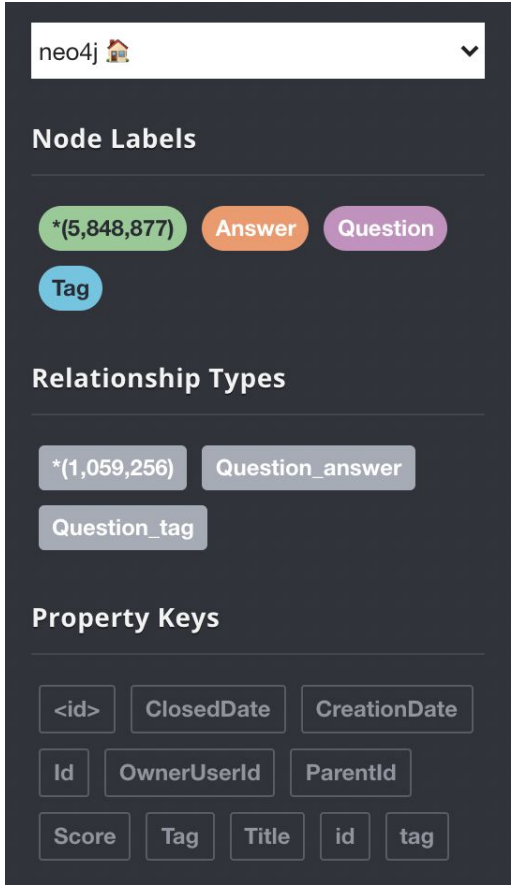
Create Relationship between Questions and Answers

```
load csv with headers from "file:///Answers.csv" as row MATCH (a: Answer { Id:
toInteger(row.Id) })
MATCH (q: Question { Id: toInteger(row.ParentId) })
MERGE (a)-[r: Question_answer]→(q);
```

Created 1048479 relationships, completed after 21289 ms.

Data Model


Answers
Question_answer
Questions
Question_tag
Tag


neo4j
Node Labels
*(5,848,877)
Answer
Question
Tag
Relationship Types
*(1,059,256)
Question_answer
Question_tag
Property Keys
<id>
ClosedDate
CreationDate
Id
OwnerUserId
ParentId
Score
Tag
Title
id
tag

Consistency vs Availability

- As a NoSQL system, Neo4j is committed to offer basic availability.
- However, a system of pagecache and transaction log (writing out changes to the graph) allow for a strong consistency.
- Neo4j handles causal-consistency and promises eventual-consistency when reading local writes.
- Causal-consistency guarantees that all processes agree on the order of causally-related operations.
- Two operations are considered causally-related when, for example, one of them is the cause for the other.

Indexing

- In RDBMS indexing can speed up joins
- However in neo4j they work a little bit different
  - Indexes are used to find starting point of a graph
  - Graph traversal benefits from “index free adjacency”
  - Indexes are only created on properties you’re searching in

Query 1

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

Table | Text | Code

	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.

- Index created for ‘questionID’ field.
- Execution time without index: 160 ms.
- Execution time with index: 13 ms.
- Decreased execution time of 91.88%.

Query 2

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

Table | Text | Code

	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.

- Index created for ‘questionID’ field.
- Execution time without index: 149 ms.
- Execution time with index: 74 ms.
- Decreased execution time of 50.34%.

Query 3

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

Table | Text | Code

	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.

- Index created for ‘questionID’ field.
- Execution time without index: 3021 ms.
- Execution time with index: 2697 ms.
- Decreased execution time of 10.72%.

Query 4

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

Table | Text | Warn | Code

	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
```

- Index created for question table's 'questionID' field and answers table's 'answerID' and 'parentID' fields.
- Execution time without index: 7318 ms.
- Execution time with index: 6989 ms.
- Decreased execution time of 4.50%.

Questions