A Performance Comparison between Graph Databases

A Degree Project about the comparison between Neo4j, GraphDB and OrientDB on different operations

Introduction

- Lavdim Imeri
- Robert Alm
- We aim to find out how Graph Databases perform on CRUD operations and why.
- CRUD means “Create, Read, Update and Delete”.

What are Graph Databases?

Databases with Nodes that are connected by edges.

Sounds like something that utilizes Tree data-structures.

In reality, they are more like a predefined level 3-Normalization.

What are the research Questions?

Question 1: What is the complexity on CRUD operations of Neo4j, GraphDB and OrientDB Databases.

Question 2: How do Neo4j, GraphDB and OrientDB perform on their CRUD operations in terms of time.

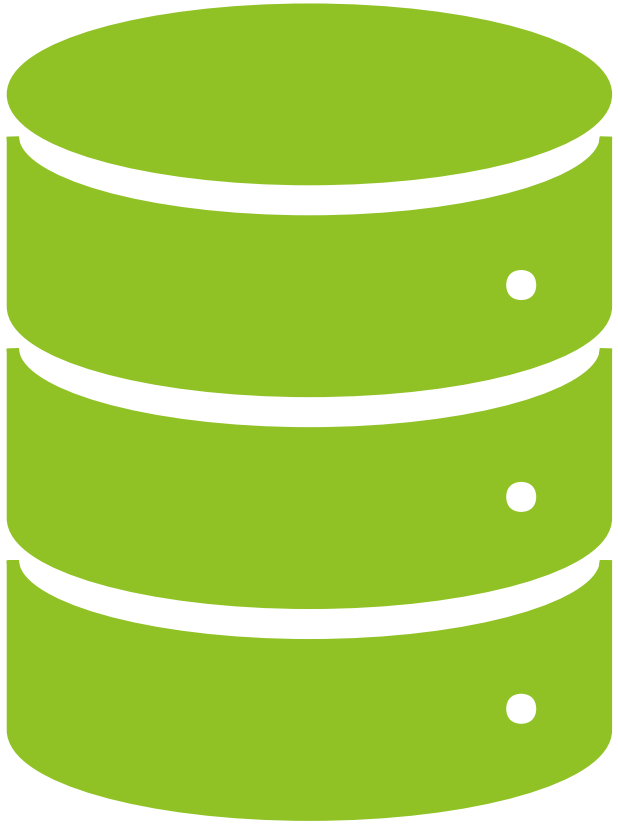
Question 3: How complexity can affect the performance of Neo4j, GraphDB and OrientDB. Why do those 3 Databases act that way.

Our Methodology

- Literature Review (for the research questions 1 and 3), which means reviewing the documentation of the selected Graph Databases.
- Profiling, (for the research question 2).
- Profiling means recording the duration of an operation.
- Analysis, (for the questions 1, 2 and 3), to draw useful conclusions.

The three Graph Databases

- Neo4J from Neo4J inc.
- GraphDB from Ontotext.
- OrientDB from SAP.



Neo4j

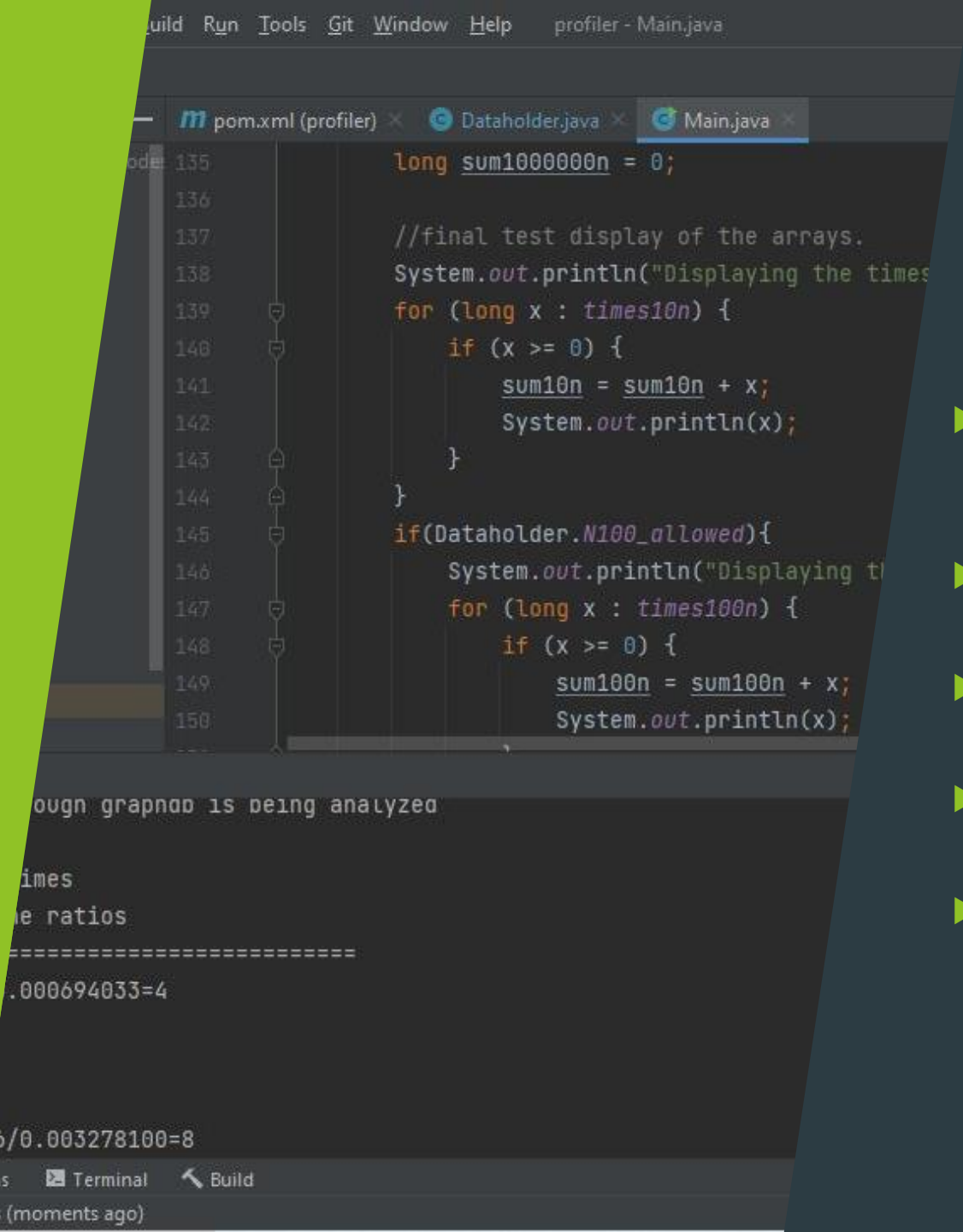
- The most widely used Graph Database right now.
- Fast, Robust, and Reliable.
- It uses the Cypher query language.
- The Abbreviation means Network Exploration and Optimization "for" Java.
- Stores data in files that hold together all the similar items, (all nodes, all relationships, etc).

GraphDB

- Utilizes web protocols like HTTP to work.
- It works based on RDF protocol and stores its data in a triadic way.
- RDF protocol uses RDF language and its abbreviation means "Resource Description Framework“.
- Uses SPARQL language to do its queries which means "SPARQL Protocol and RDF Query Language“.

OrientDB

- Tries to implement a multi-model approach.
- Encapsulates SQL inside a layer of Object Oriented Programming.
- Tries to mimic Tree Data Structures.
- Uses the memory of the computer and stores its data through series of buffers.
- Its development seems to be on hold for now.

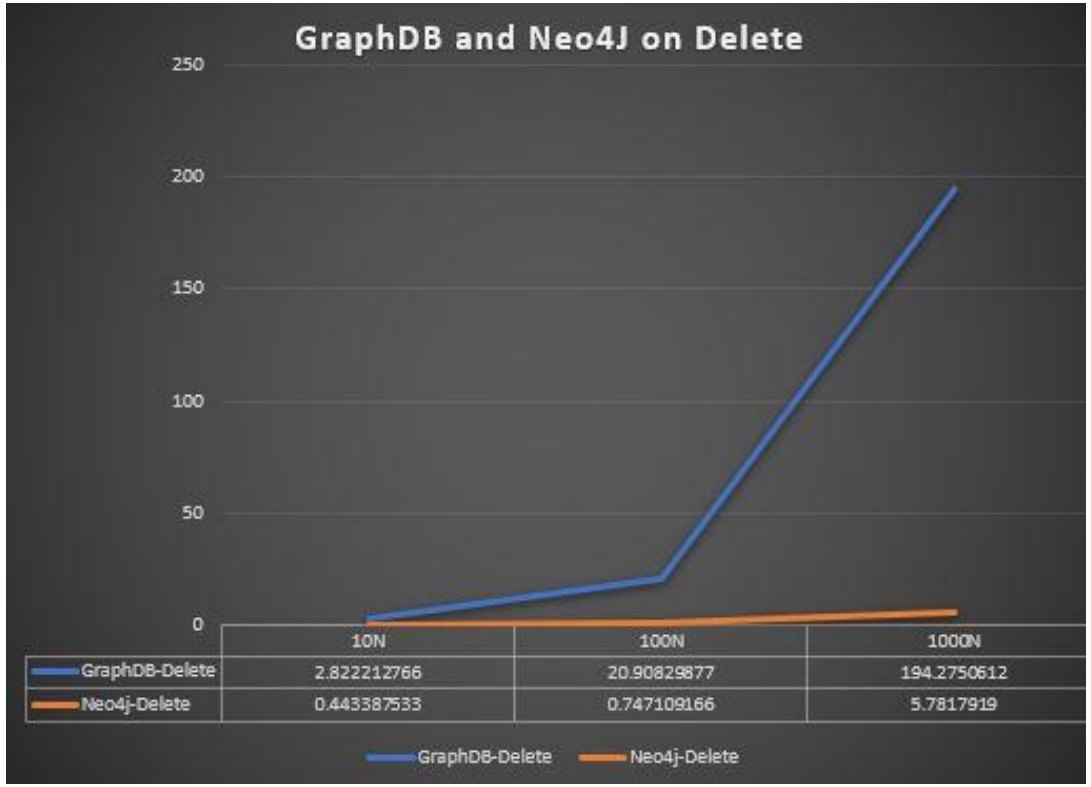


Profiler

- A software that is developed by us to profile CRUD operation on Graph Databases.
- It performs many sets of different sizes with single CRUD operations.
- Records the time for the completion on each set.
- It does that multiple times for accuracy.
- It has an analyzer functionality that performs a simple algorithmic analysis on each set to find its performance.

Example of Results

- DELETE can be sometimes the most complex operation
- OrientDB failed to deliver so it is excluded.
- We compare GraphDB with the second profiling of Neo4J
- Neo4J has a better performance

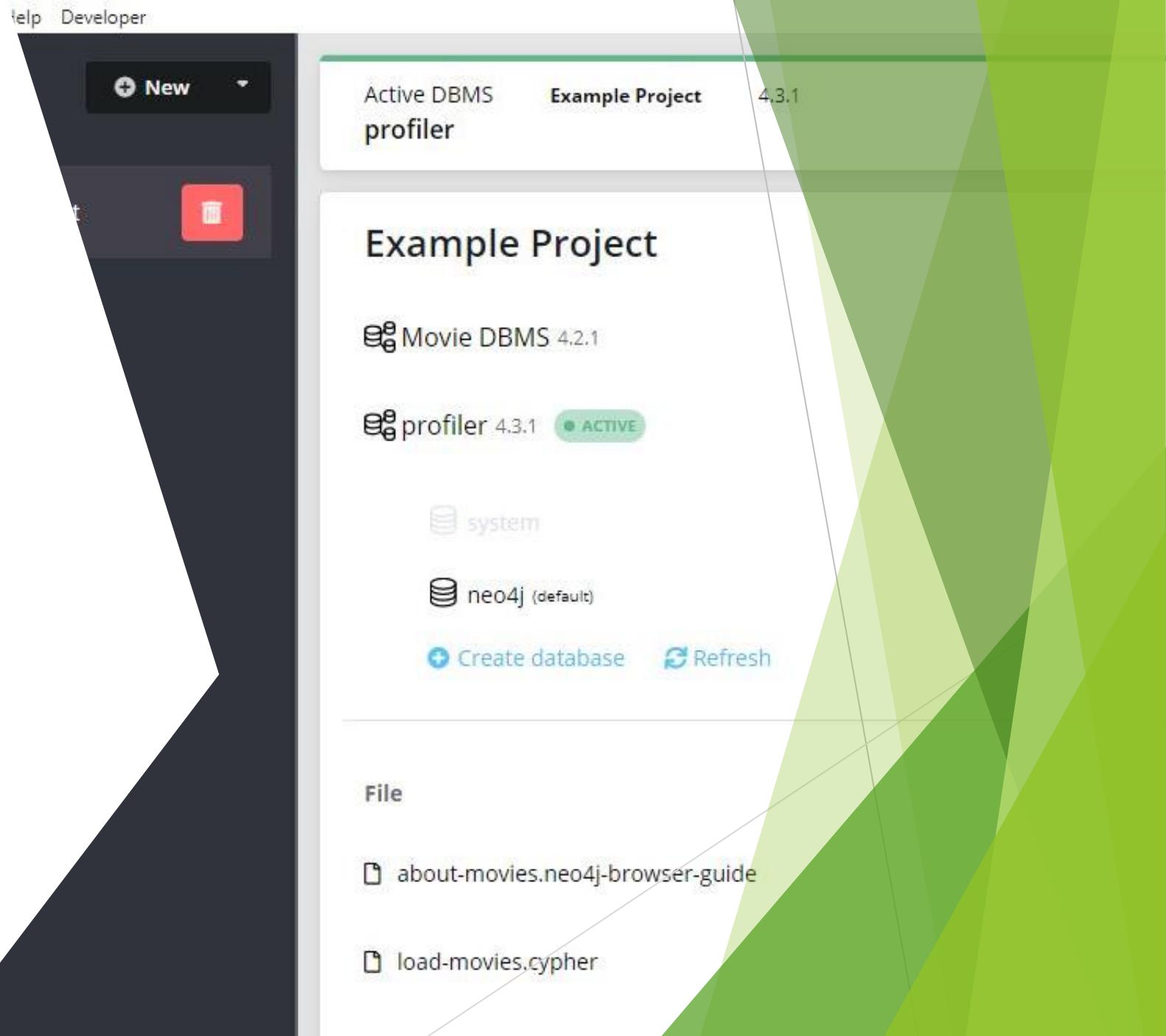


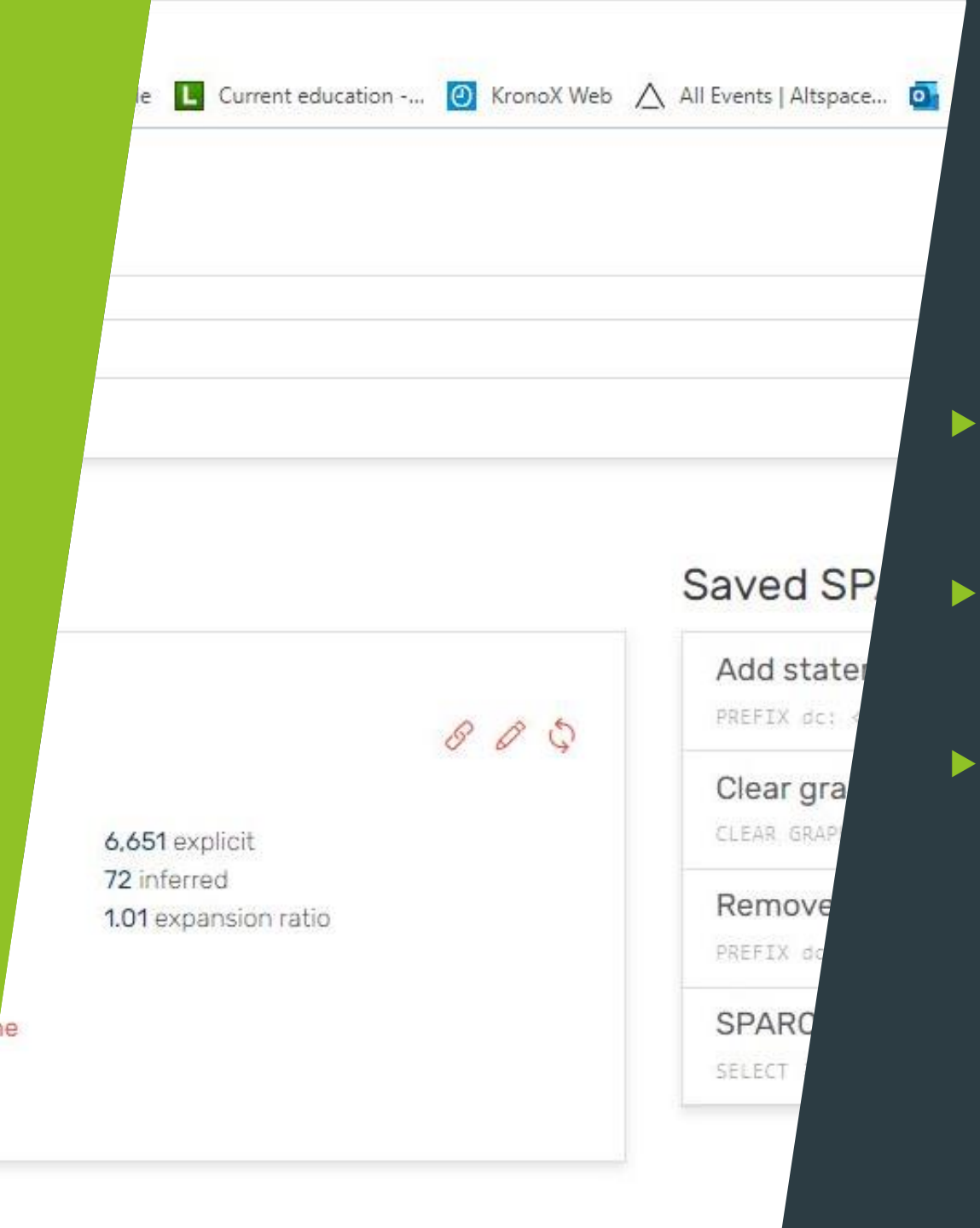
Results

- Neo4J has the best performance.
- GraphDB takes the second place and performs well.
- OrientDB fails to perform its requested operations.
- Neo4J performance got accidentally affected when its database was filled with records, and we decided to empty the database and try again with the empty database.

Discussion: Neo4J

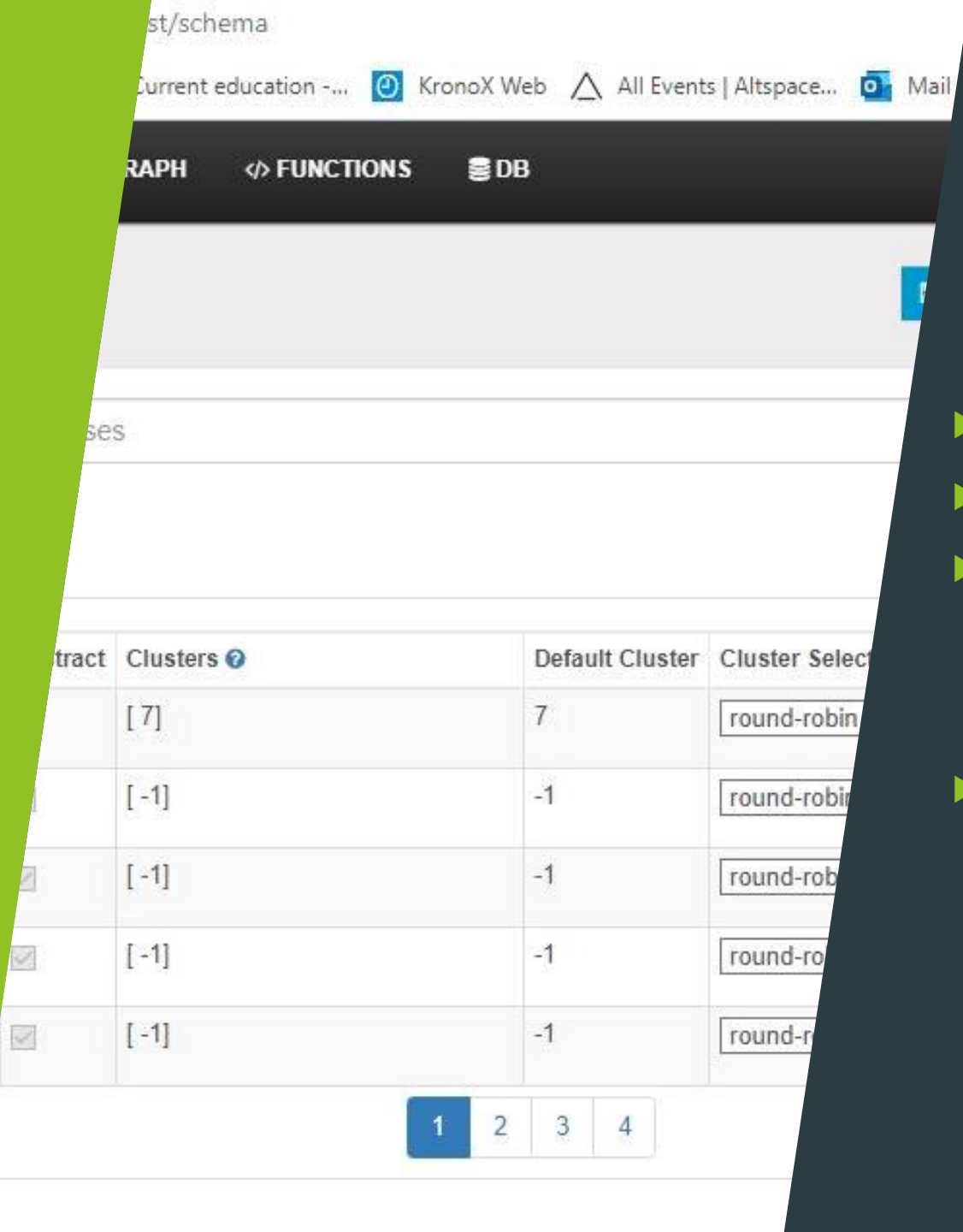
- The Profiling operation was a bit unfair for Neo4J as single operations are not the strong point of Neo4j, (because of its reliability factor).
- Neo4j is the "go to" solution for a stand-alone database server.
- The structure of its files gives it a great scalability, especially when it handles multiple operation at the same time.
- Cypher reduces human errors due to its simplicity and enforces normalization, making the databases more efficient.





Discussion: GraphDB

- The only database that could be turned to an internet of databases.
- The triadic way GraphDB stores its data allow complex queries on distributed systems.
- Not the best choice for a stand-alone database but it works decently.



Discussion: OrientDB

- Huge potential.
- Sadly, is given up and broken.
- Having a layer of OOP and using the memory of the host for a better performance is a great idea.
- Unnecessarily complex structure.

Conclusions

- All databases hold an O(N) average-case complexity, and we assume an O(N^2) worst-case complexity.
- We confirm that OrientDb currently does not working.
- Neo4J is a better stand-alone solution while GraphDB, theoretically, can work better for distributed systems.
- None of the Databases had Tree Data-structure properties like "Push/Pull" or "Enqueue/Dequeue" and we believe that they should have that.

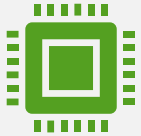
Answering Question 1: What is the complexity on CRUD operations of Neo4j, GraphDB and OrientDB Databases.

- The theoretical complexity of OrientDB remains theoretical because it didn't work.
- All the Databases, theoretically, have an average O(N) and a worst-case O(N^2).
- The average complexity is explained due to the large overhead that takes care everything around an operation.
- The worst case can be assumed due to unforeseen factors

Answering Question 2: How do Neo4j, GraphDB and OrientDB perform on their CRUD operations in terms of time.

- The best performance belongs to Neo4J.
- GraphDB takes the second place.
- Both databases display a very stable O(N) complexity.
- OrientDB failed to deliver.

Answering Question 3: How complexity can affect the performance of Neo4j, GraphDB and OrientDB .
Why do those 3 Databases act that way.



Theoretically, OrientDB takes advantage of the memory of the host to have a better performance.

GraphDB works on the top of web protocols, and it can work better on distributed systems of a large scale.

Neo4J is a robust stand-alone solution, but it has an impressive scalability due to its file structure.

We thank you for your time!