A low-level approach for graph database query processor

**Graph database** is a database that uses graph structures for semantic queries with nodes, edges (i.e. relationships), and properties to represent and store data. The relationships allow data in the store to be linked together directly and, in many cases, retrieved with one operation.

**Cypher** is a declarative graph query language that allows for expressive and efficient data querying in a property graph. The language was designed with the power and capability of SQL in mind, but based on the components and needs of a graph database.

**Problem** Graph DBMS can run many kinds of queries, possibly in parallel, and sometimes with transaction isolation. Therefore it **may be slower** than specialized, one-time query written in C.

**Goal** Since graph data can be stored manifoldly, goal of this project is to **boost Cypher queries performance** assuming data storage as in one of the most popular graph database—Neo4j.

**Solution** Inspired by the lecture held by Tiark Rompf: “A PL & Compiler View on Data Management and ML Systems” and his presented paper [1], we will try to write **Cypher to C query compiler** for Neo4j data storage. In Neo4j, entities are stored on disk by leveraging linking [2] as shown on the graph on the right. Since entities there have fixed, maximum size, we can achieve similar performance boost as in [1] by utilizing pointers in C language and directly reference entities throughout memory or file. Below, two diagrams present Cypher Neo4j processor behaviour (left) and how we could interpret Cypher and generate C queries to get the same result, but faster (right).

### Cypher Neo4j processor

**MATCH (n: Person)-(KNOWS)->(m: Person)
WHERE n.name = 'Alice'
**

1. **NodeIndexSeek**
2. **ExpandAll**
3. **Filter**
4. **ProduceResults**
5. **Result**
6. **Execute Plan**

### Cypher queries in C

**MATCH (n: Person)-[KNOWS]->(m: Person)
WHERE n.name = 'Alice'
**

1. **Scala Cypher Interpreter**
2. **C language generator**
3. **Execute C**

![Diagram of Cypher Neo4j processor](image1)

![Diagram of Cypher queries in C](image2)

**References**


[2] “Understanding Neo4j’s data on disk”:

https://neo4j.com/developer/kb/understanding-data-on-disk/