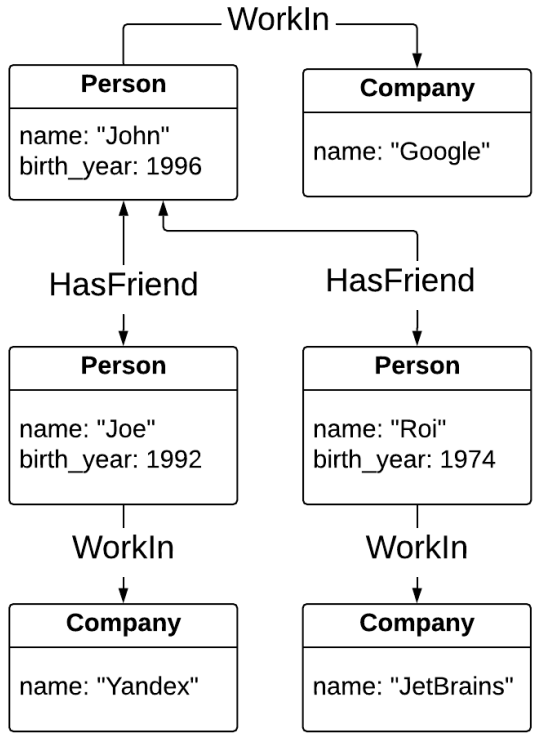
# Разработка матричного алгоритма поиска путей с контекстно-свободными ограничениями для RedisGraph

**Автор:** Терехов Арсений Константинович, 444 группа
**Научный руководитель:** доцент, к.ф.-м.н. Григорьев С. В.
**Рецензент:** инженер-программист ООО "ИнтеллиДжей Лабс", к.ф.-м.н. Березун Д. А.

Кафедра системного программирования

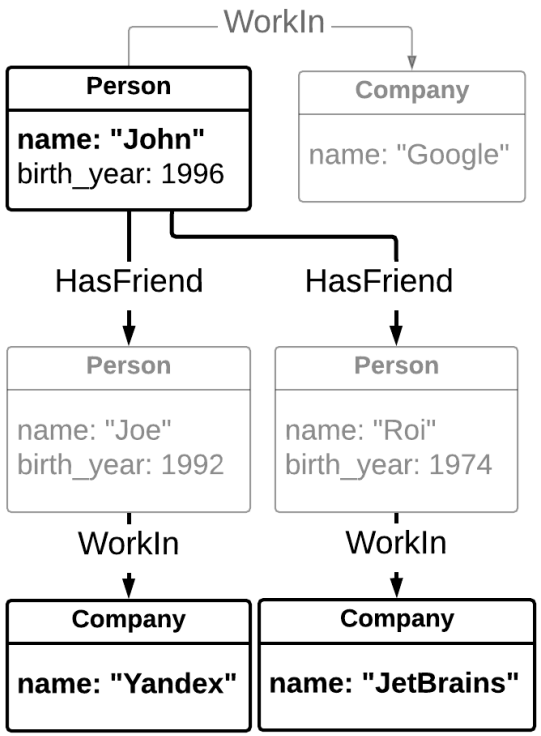
# Графовые базы данных

- Графовые базы данных
  - Удобный инструмент для работы с граф-структурированным данными
- Графовая модель представления данных
  - Основные сущности – вершины графа
  - Вершины могут иметь метки и свойства в формате ключ-значение
  - Отношения задаются ребрами графа
  - Типы отношений соответствуют меткам рёбер



# Графовые базы данных

- Язык запросов графовых баз данных
  - Является основным интерфейсом для работы с графами
  - Наиболее популярным является Cypher
- Запросы вида сопоставления с образцом
  - Задание интересующих путей или подграфов с помощью шаблонов
  - Извлечение информации из удачных сопоставлений




```
MATCH (person)-[:HasFriend]->()-[:WorkIn]->(company)
WHERE person.name = "John"
RETURN company.name
```

# Задача достижимости

## Задача достижимости на графе в терминах формальных языков

*Дано:*

- $(V, E)$ – *ориентированный граф,*
  $l : E \to \Sigma$ – *рёбра помечены символами алфавита* $\Sigma$
- $G$ – *формальная грамматика,* $L(G) \subset \Sigma^*$ – *язык ограничений*

*Хотим найти:*

- $\{(v, to) : \exists p = (e_1, ..., e_n) \in E^* : l(e1)...l(e_n) \in L(G),$
  $src(e_1) = v,\ dst(e_n) = to\}$

# Контекстно-свободные ограничения

- Контекстно-свободные ограничения
  - Позволяют выразить сложные отношения между вершинами
  - Имеют широкое применение в биоинформатике и при обработке rdf файлов
- Язык запросов графовых баз данных
  - Не поддерживает контекстно-свободных ограничений
  - Сильно ограничен
  - Для более сложных запросов приходится искать альтернативные решения
    - ★ Understanding Data Science Lifecycle Provenance via Graph Segmentation and Summarization, 2018, Hui Miao

# Цели и задачи

- Цель: разработать полную поддержку запросов с контекстно-свободными ограничениями для графовой базы данных
  - Выполнить обзор существующих реализаций поддержки запросов с контекстно-свободными ограничениями в графовых базах данных.
  - Интегрировать выбранный на этапе обзора алгоритм в подходящую для него графовую базу данных.
  - Расширить язык запросов выбранной базы данных конструкциями, необходимыми для выражения контекстно-свободных ограничений
  - Произвести замеры производительности полученного решения

# Существующие подходы и решения

- Сравнительный анализ времени работы алгоритмов, Йохем Куйперс и др., 2019
  - Реализованы наиболее известные алгоритмы контекстно-свободной достижимости
  - Алгоритмы запускались в контексте Neo4j
  - Время работы оказалось недостаточно приемлемым
- Расширение библиотеки Meerkat, 2018
  - Вместо языка запросов Cypher предоставляются парсер-комбинаторы
  - Использует Neo4j как хранилище графов
  - Медленно работает на больших графах

# Выбор базы данных и подходящего алгоритма

- Алгоритм, основанный на матричных операциях, Рустам Азимов, 2018
  - Работает с графами в виде разреженных матриц
  - На текущий момент самый быстрый на практике
- Графовая база данных RedisGraph
  - Использует GraphBlas для представления графов в виде матриц
  - Транслирует запросы в матричные выражения
  - Идеально подходит для матричного алгоритма

# Расширение языка запросов Cypher

Шаблоны путей:

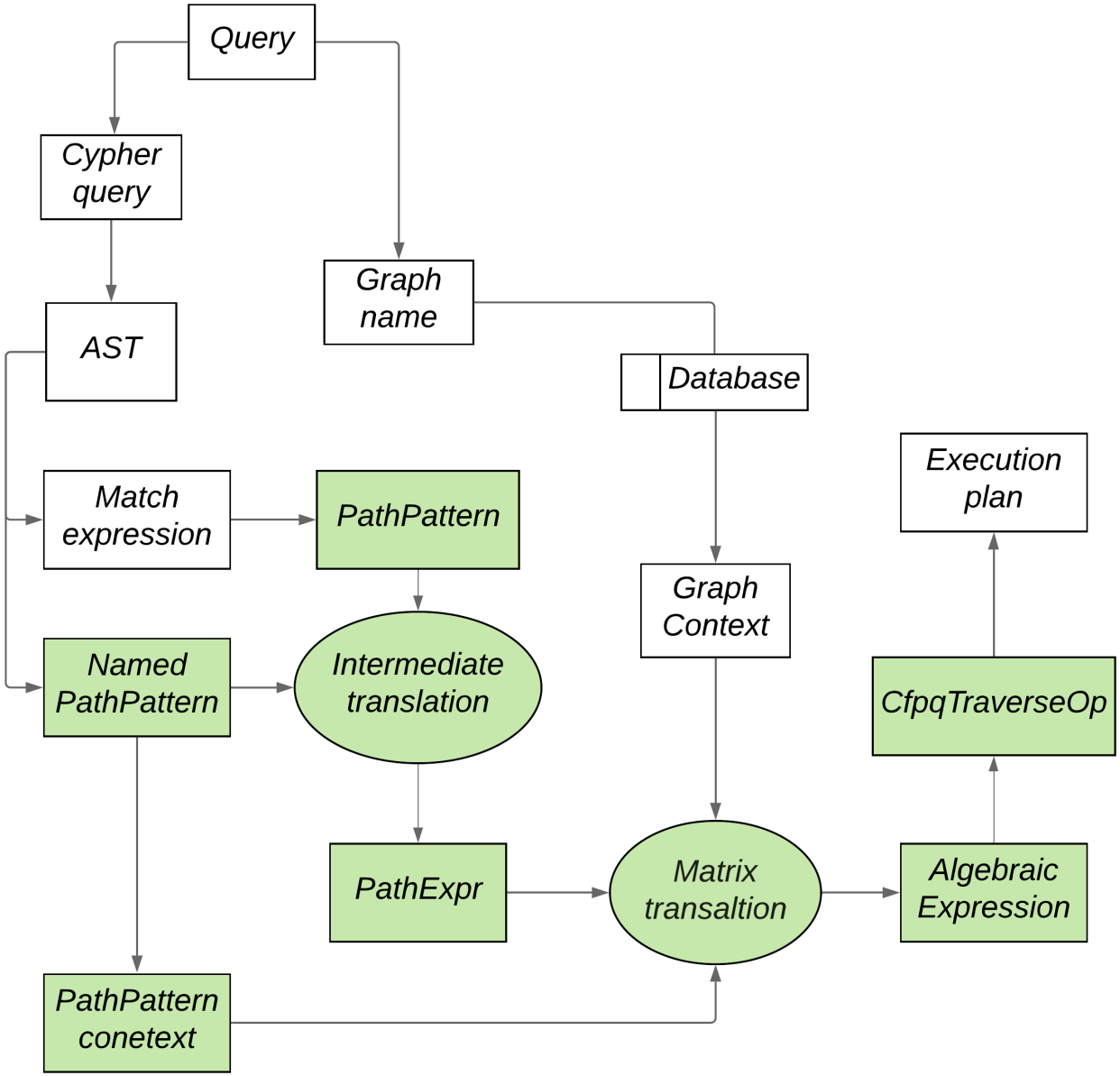
- Задают регулярные выражения над базовыми шаблонами
- Могут быть именованными
- Являются более мощной альтернативой шаблонам рёбер

Базовые шаблоны:

- Шаблоны вершин и рёбер
- Ссылки на именованные шаблоны

| Шаблон пути | Язык ограничений |
|---|---|
| [:A :B] \| <[:C :D] | $\{AB,\ DC\}$ |
| [:A :B]* | $\{AB\}^* = \{AB,\ ABAB, ...\}$ |
| S = [:A ~S :B] \| () | $\{a^n b^n : n \geq 0\}$ |

# Реализация


Query
Cypher query
AST
Graph name
Database
Match expression
PathPattern
Named PathPattern
Intermediate translation
Graph Context
Execution plan
CfpqTraverseOp
PathExpr
Matrix transaltion
Algebraic Expression
PathPattern conetext

# Замеры производительности: сравнение с парсер-комбинаторами

| G | $\|V\|$ | $\|E\|$ | #result | Meerkat time (ms) | RedisGraph time (ms) |
|---|---|---|---|---|---|
| wine | 773 | 2450 | 66572 | 541 | 31 |
| pizza | 671 | 2604 | 56195 | 476 | 24 |
| measure-primitive | 341 | 771 | 15156 | 158 | 11 |
| funding | 778 | 1480 | 17634 | 99 | 14 |
| atom-primitive | 291 | 685 | 15454 | 102 | 10 |

Оборудование: Intel Core i7 4×1.8GHz, 8 GB RAM

# Замеры производительности: сравнение с neo4j

```
PATH PATTERN
 S = ()-/ [:brdrTransitive [~S | ()] <:brdrTransitive] /->()
MATCH (v)-/ ~S /->(to)
RETURN COUNT(*)
```

| G | $\|V\|$ | $\|E\|$ | #result | Neo4j best time (s) | RedisGraph time (s) |
|---|---|---|---|---|---|
| geospeices | 225 000 | 1 550 000 | 226 669 749 | 6 953.9 | 26.1 |

- Neo4j: Intel Xeon E5-4610 v2, 8×2.30GHz, 400 GB RAM
- RedisGraph: Intel Core i7-6700 CPU, 64 GB RAM 4×3.4GHz

# Результаты

- Выполнен обзор текущих решений поддержки запросов с контекстно-свободными ограничениями
- Интегрирован матричный алгоритм в RedisGraph
- Разработана поддержка расширения языка запросов Cypher для RedisGraph
- Произведены замеры производительности полученного решения
- Часть результатов работы изложены в статье, принятой на конференцию GRADES-NDA 2020