Графовая база Neo4j во внутреннем приложении 2ГИС

Антон Максимов, 2ГИС
a.maximov@2gis.ru, twitter.com/evilsyncope
19.06.2015

Про меня

- 7 лет в программировании.
- Писал на Java платёжные системы.
- Потом бэкенды для мобильных MMO.
- Нашёл себя в гео-информационных системах и пересел на .NET

Кровавый Enterprise

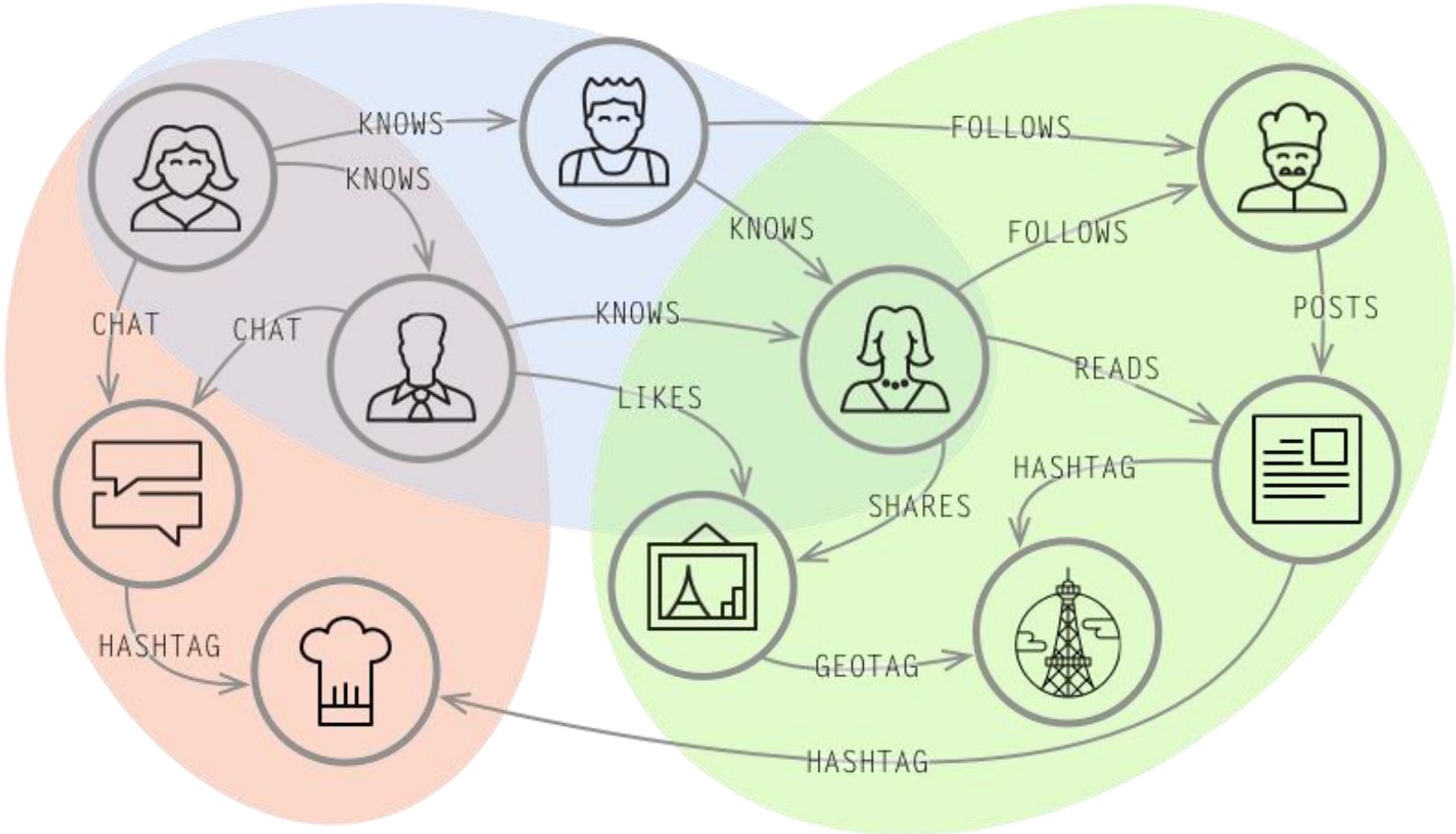
2ГИС - не только сайтики и мобилки

- Все данные (карты, справочники фирм и т.д.) собирают наши люди (специально обученные)
- Сами пишем инструменты для работы с данными.
- ~ 100 человек во внутренней разработке.

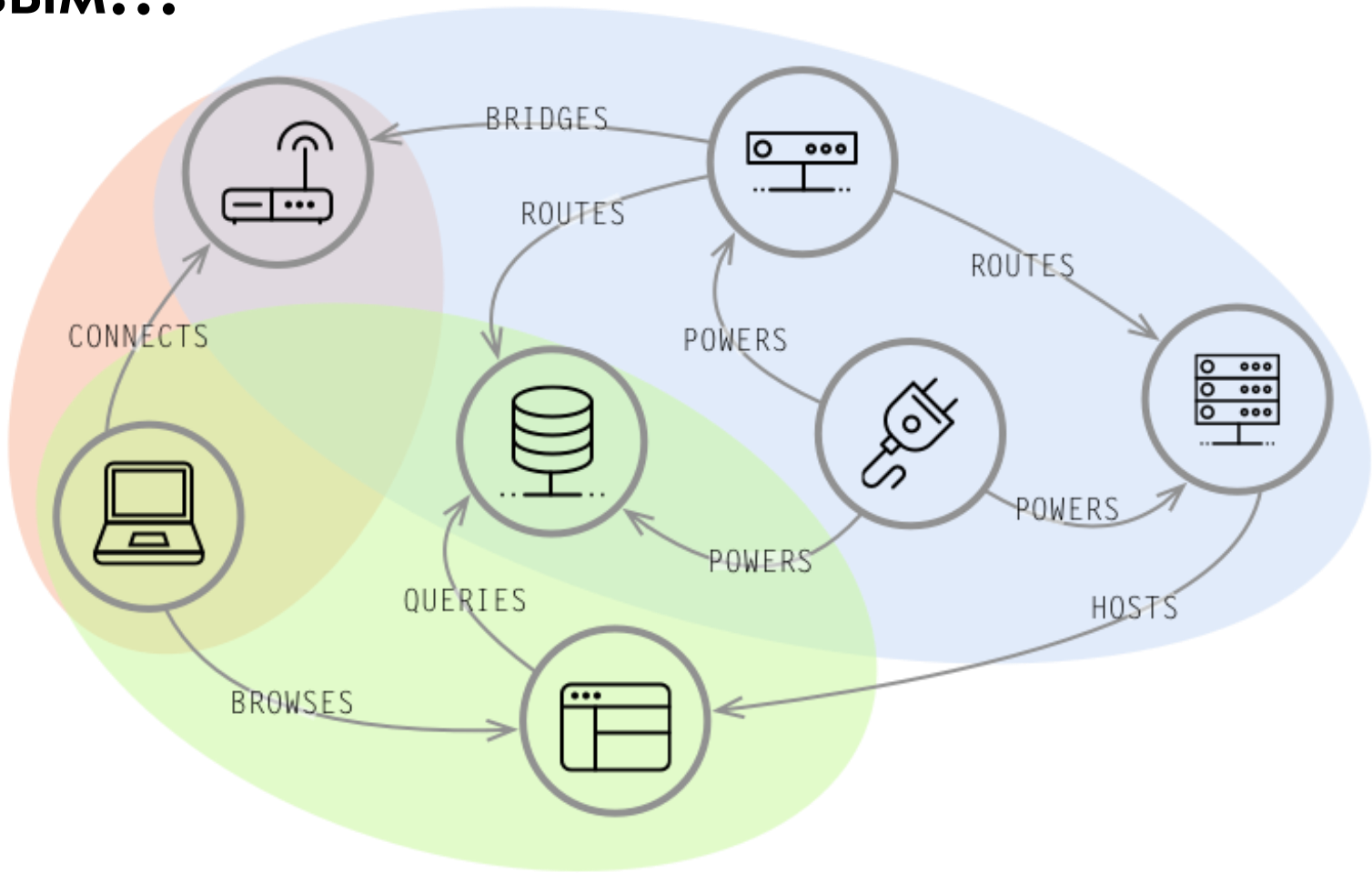
Что такое Fiji? Это карта!

- В Fiji картографы рисуют карты для конечных продуктов. Пришла на смену древней системе из прошлого века.
- Энтерпрайз, но весёлый.
- Сервисы на ASP.NET, rich client на WPF.
- В мае 2014 вышли в московском филиале.
- Спустя год - 80 пользователей из 37 городов

Граф бывает социальным...



Сетевым...


BRIDGES
ROUTES
ROUTES
CONNECTS
POWERS
POWERS
POWERS
QUERIES
HOSTS
BROWSES

И даже дорожным

Дорожный граф велик

- 560000 объектов-дорог в Москве.
- Более 2000000 дорог в 37 городах.
- Нужен для поиска проезда на личном транспорте.
- Мы отдаём данные, по которым в конечных продуктах работают хитрые алгоритмы поиска кратчайшего пути.
- Но нужна корректность графа — **проверяем её**.

А есть ещё и транспортный

Тоже немаленький

- 200000 звеньев.
- Поиск проезда на общественном транспорте.
- Также проверяем граф и ищем кратчайшие пути (маршруты)

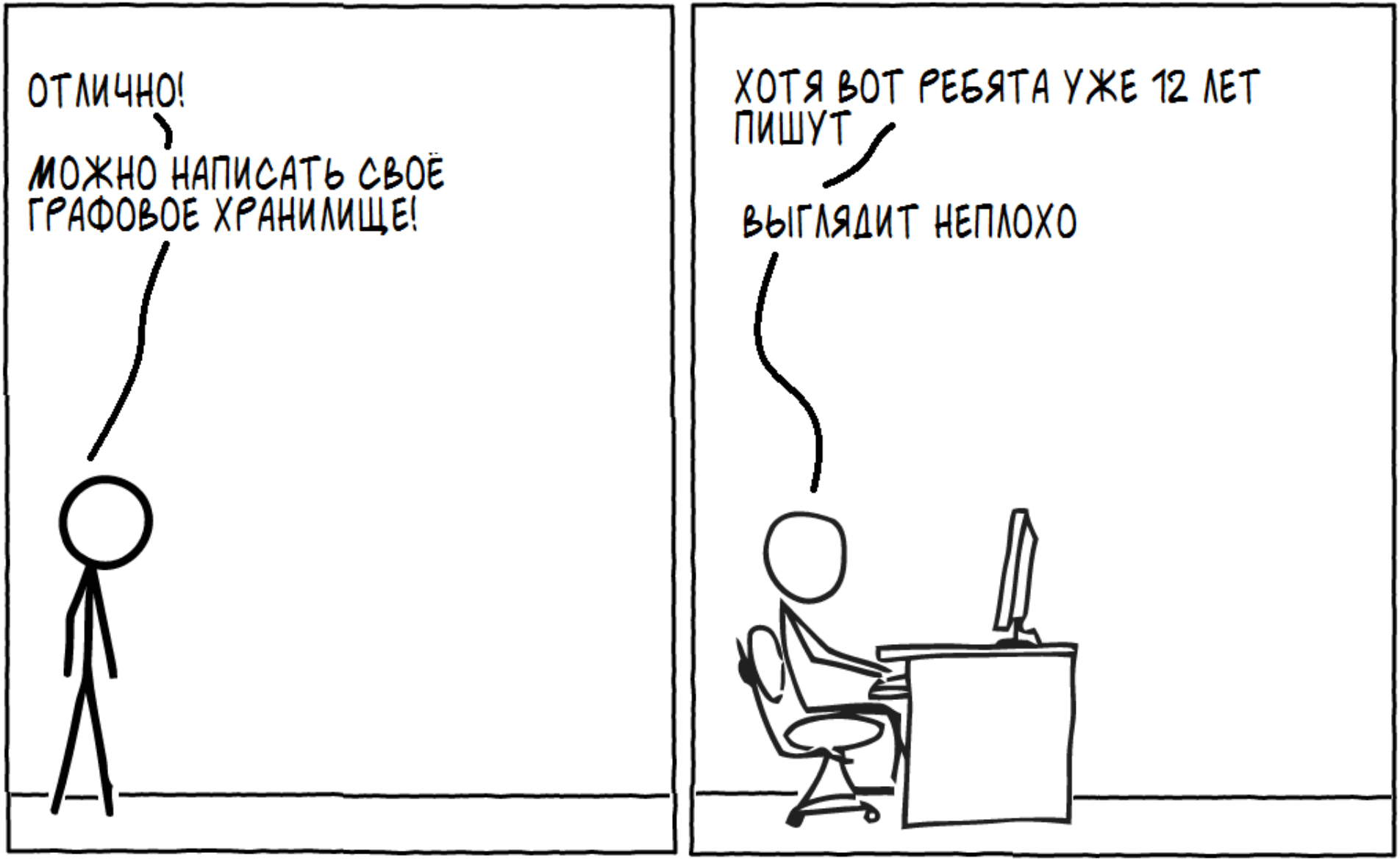

КОНЕЦ МЕСЯЦА
Я ТАМ ГРАФ МОСКВЫ ПОПРАВИЛ
МНЕ БЫ СВЯЗНОСТЬ ПРОВЕРИТЬ
ОК!
ЩА ЗАПУСТИМ
А СКОРО РЕЗУЛЬТАТ БУДЕТ?
КОНЕЧНО!
ЗАВТРА ПРИХОДИ

Как было

- Oldschool-style: храним всё в RDBMS, по запросу сооружаем граф и обходим его.
- Проверка связности в Москве занимает 6 часов.
- Поменял граф — корректность могла нарушиться — перепроверяй.
- Искать маршруты транспорта — тоже очень долго.

И как хочется

- Хорошо бы держать готовый граф и применять к нему изменения, ведь меняется небольшая часть – быстрое обновление.
- Изменения в графе желательно видеть почти сразу после завершения операции в мастер-системе.


ОТЛИЧНО!
МОЖНО НАПИСАТЬ СВОЁ ГРАФОВОЕ ХРАНИЛИЩЕ!
ХОТЯ ВОТ РЕБЯТА УЖЕ 12 ЛЕТ ПИШУТ
ВЫГЛЯДИТ НЕПЛОХО

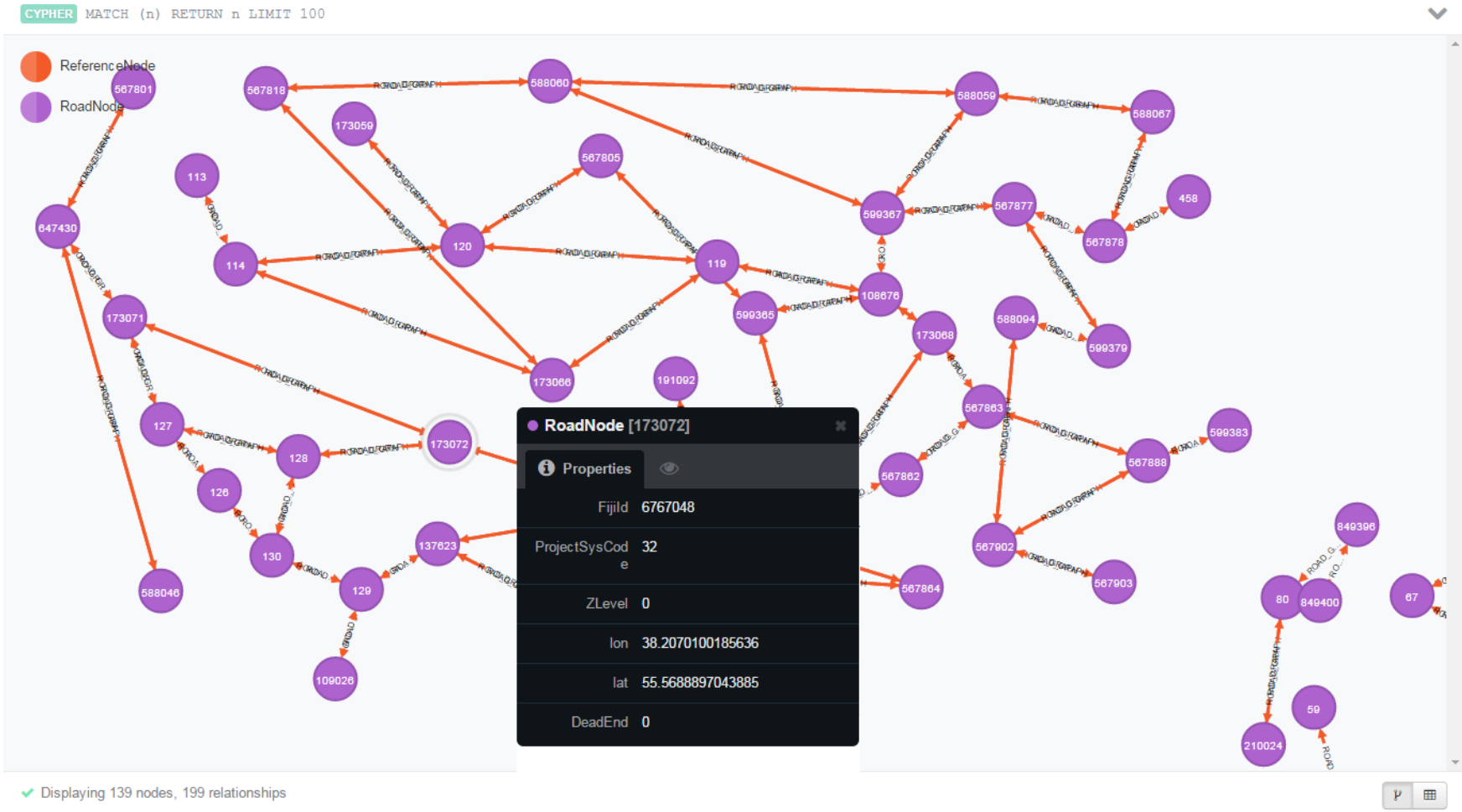
Neo4j Graph DB

- Работает на JVM, написана на Java и Scala.
- Community и Enterprise edition.
- Полностью опенсорс. Даже платная часть. https://github.com/neo4j
- Кроссплатформенная. Embeddable.
- 2 типа объектов: Node и Relationship.
- REST API.
- Запросы, индексы, транзакции - всё умеет. Cypher query language — киллер-фича.

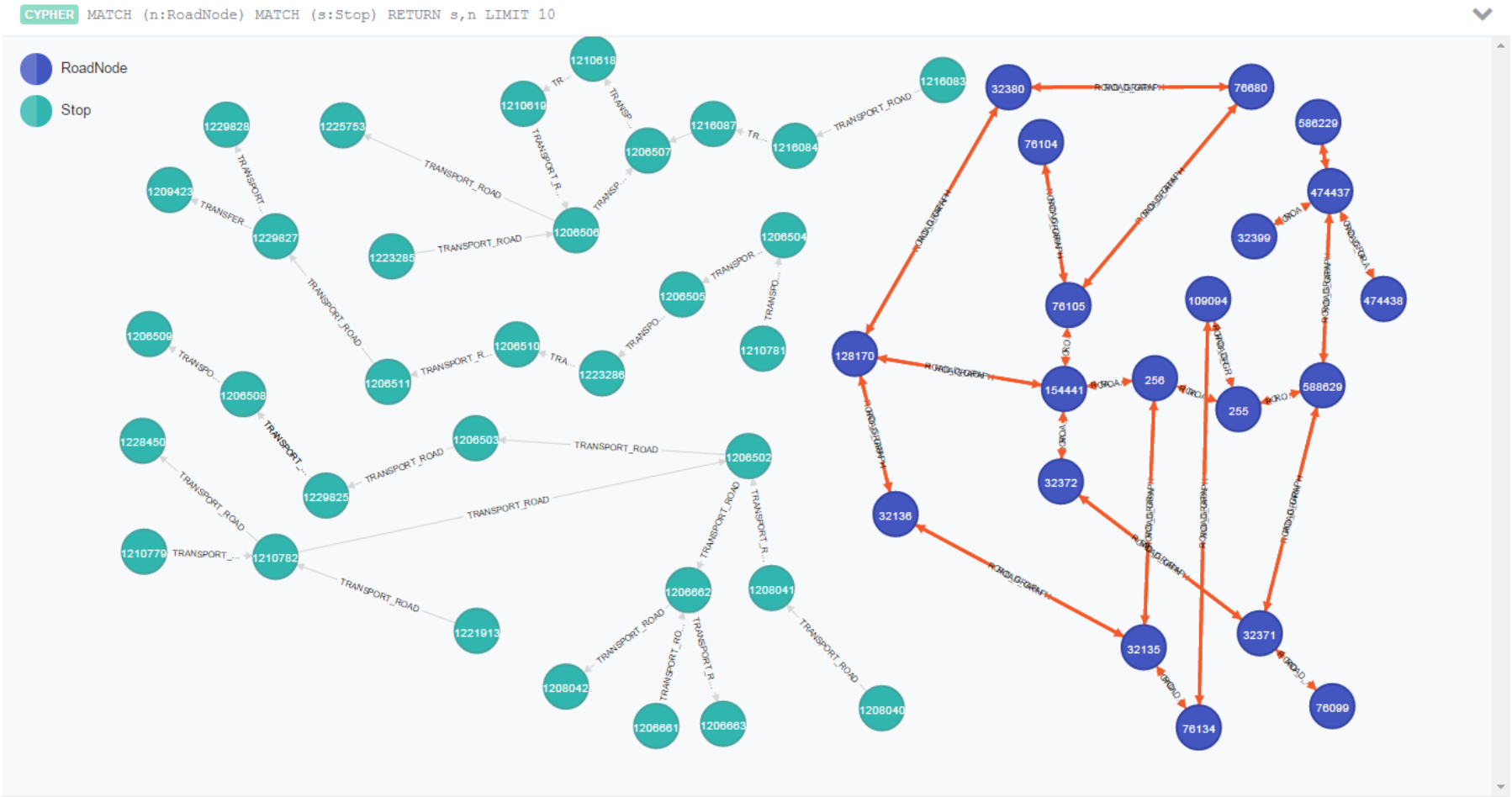
Просто вершины и рёбра?

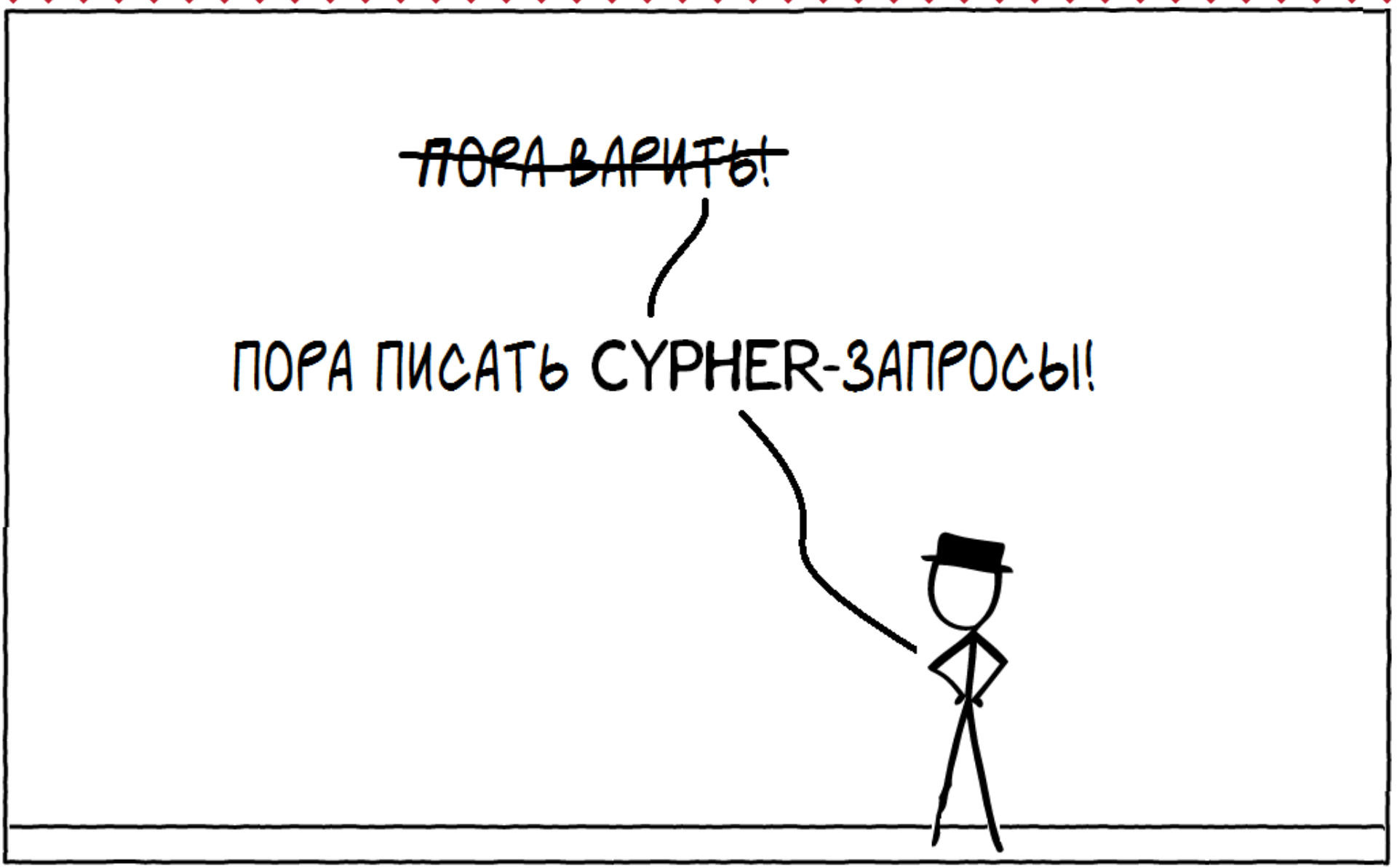
- У рёбер есть тип. Например, ROAD_EDGE или TRANSPORT_EDGE.
- У вершин тоже есть тип (Label), может быть, и не один.
- Плоские свойства (property), ключ-значение: int, long, String и т.д.

График, у меня есть график



Графы мирно сосуществуют




~~ПОРА ВАРИТЬ!~~
ПОРА ПИСАТЬ CYPHER-ЗАПРОСЫ!

Возможности Cypher (CQL)

- Старались сделать похожим на SQL.
- Create, Update, Merge (Upsert), Delete.
- Load CSV Using Periodic Commit для обновлений большими пачками.
- Match ~ Join.
- Where == Where. And, Or, In.
- Return, Limit, Order by.
- Множество встроенных функций: математика, строки, коллекции.

Попробуем написать Cypher-запрос

```
01. match (stop :Stop )-[r :TRANSPORT_ROAD ]-()
02. where  not  ()-[r:TRANSPORT_ROAD] -> (stop)
03.   or not  (stop)-[r:TRANSPORT_ROAD]->()
04.   and  stop.CityId={CityId}
05. return distinct  r.FijiId
```

Обновление/удаление данных

- Процесс-синхронизатор читает пачки изменений из RDBMS.
- Инкрементально - не используем Update, только Insert и Delete.
- Немного медленнее, но всегда можно быстро восстановить из ночного бэкапа.

Скорость записи

- Невысока, максимум ~4-5к записей в секунду на наших данных (Native API, single thread, batch import)
- В 2.2.0 (2015-03-25) стало быстрее для параллельных одиночных запросов (обещают до 100x).
- При импорте без транзакций – до 1M/sec, но только в пустую базу.
- Разработчики утверждают, что сейчас это наиболее быстрая графовая база.

Сколько весит?

- Вся база на диске ~2.8 Gb.
- В памяти ~10 Gb.
- ~2.4M вершин, 7.3M рёбер, 20.3M свойств

Когда Cypher бессилен

- Некоторые алгоритмы уже реализованы (Дейкстра).
- Проверка связности по алгоритму Косарайю - нет :(
- Не получается декларативно? Можно императивно!
- Самый низкоуровневый способ: Native API, работа с графом напрямую.
- Либо более удобный Traversal API — указываем способ обхода и действия на каждом шаге.
- **В любом случае, придётся писать *java-plugin*.**

Schema в моём NoSQL?

- Появилась в 2014 году, в версии 2.0.
- На вершины можно навешивать ярлыки (Label). Можно по несколько ярлыков на одну.
- Схема нежёсткая. Ярлыки добавляются и удаляются динамически.
- У нас всего два ярлыка - RoadNode и Stop.
- Но у вас может быть больше. Это удобно и позволяет искать быстрее.

Индексы

- Только для вершин.
- Строятся по паре ярлык-свойство.
- create index on :Stop(Fijild)
- Используются в условиях внутри Where.
- Если вам нужны индексы по рёбрам - скорее всего, что-то делаете не так.
- Есть и unique constraints, но ощутимо замедляют обновления - не используем.

Spatial index

- Подключается плагином, в основе - R-Tree.
- Дерево реализовано прямо в базе, т.е. специальными вершинами и рёбрами.
- Пробовали в 2014-м, на миллионах вершин потребление памяти было слишком высоким (бесконечный GC).
- Отказались.

Транзакции есть!

- ACID.
- Но MVCC всё ещё нет. Никаких вам снапшотов!
- Данные сливаются на диск отложенно.
- Много маленьких транзакций - нагрузка на диск, слишком большие - OutOfMemoryException.
- Batch insert - в пустую базу можно очень быстро писать без транзакций.

Получилось?

- Связность проверяется 30 секунд для графа Москвы (**в 720 раз быстрее**, если не вспоминать, что граф не пересоздаётся)
- Изменения вносить просто - добавил новый тип звеньев, старые алгоритмы не сломались.
- Либо проставил новое свойство только в некоторые рёбра/вершины.
- Java-код плагина легко тестировать с помощью embedded Neo4j.

Подводные грабли

- Потребление памяти и неожиданный GC.
- Один раз повредился индекс при остановке windows-службы (*делайте бэкапы*).
- Медленные запросы нельзя мониторить и отменять, только выставить общий таймаут.
- Репликация (master-slave), онлайн-бэкапы – только в Enterprise edition. Либо AGPL, либо за большие деньги.

Чего лучше не делать

- Загружать граф, который не влезает в память одной машины.
- Использовать под большой write-нагрузкой и запускать обширные обходы. Либо самому рулить блокировками.
- Использовать под Windows, потому что медленнее и ненадёжно.
- Использовать старые Lucene-индексы - тормозят.

Общие впечатления

- Заточена под чтение - обходы очень быстры.
- Данные пока не теряли.
- Неплохая документация.
- Популярная - много статей, презентаций, тем на StackOverflow.
- База хороша, но подойдёт только если вам действительно нужно обходить граф.

Спасибо! Вопросы?

Антон Максимов, 2ГИС
a.maximov@2gis.ru,
twitter.com/evilsyncope