Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики

Курсовая работа

Применение графовых баз данных для анализа маршрутов

Работу выполнил
Безгласный П.А.

Научный руководитель:
с.н.с. лаборатории ОИТ факультета ВМК МГУ им. М.В.Ломоносова,
к.ф.-м.н. *Намиот Дмитрий Евгеньевич*

Москва
2016

Содержание

Введение.

С 1980-х годов реляционные системы управления базами данных (СУБД) стали занимать доминирующее положение среди средств хранения данных. Несмотря на то что реляционные хранилища обеспечивают наилучшее сочетание простоты, устойчивости, гибкости, производительности, масштабируемости и совместимости, их показатели по каждому из этих пунктов не обязательно выше, чем у аналогичных систем, ориентированных на какую-то одну особенность. Однако универсальность реляционных СУБД перевешивала какие-либо другие недостатки.

Сегодня ситуация несколько иная. Появившиеся в последние годы так называемые NoSQL (Not only SQL, не только SQL) хранилища реализуют модели данных, имеющие существенные отличия от традиционной реляционной модели. Основная их цель — расширить возможности баз данных (БД) в тех областях, где реляционная модель и SQL недостаточно гибки, и не вытеснять их там, где они справляются со своими задачами. Создатели таких БД среди множества преимуществ использования NoSQL-решений называют высокую производительность при использовании специфических моделей данных и легкость работы с ними.

Одним из наиболее популярных и актуальных подвидов нереляционных хранилищ являются графовые БД [1]. Как ясно из названия, основная модель данных в них — классический математический граф. Проекты в области графовых БД начали появляться с конца 1980-х годов, однако в большей степени носили академический характер [2, 3]. В последнее время наблюдается бурный рост интереса к графовым БД в связи с тем, что такая система представления данных оказалась естественной и востребованной в современном мире различных социальных связей (Интернет, социальные сети и т. д.).

К достоинствам графовых моделей БД по сравнению с традиционной реляционной моделью исследователи относят не только возможность естественной реализации графовых операций (поиска путей, выделения сообществ и т. п.), но и гибкую схему данных, позволяющую унифицировать хранение разнородных объектов [1, 4].

1. Теоретические основы баз данных и анализа маршрутов

1.1 Базы данных

База данных (БД) — именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области, или иначе БД — это совокупность взаимосвязанных данных при такой минимальной избыточности, которая допускает их использование оптимальным образом для одного или нескольких приложений в определенной предметной области. БД состоит из множества связанных файлов.

Система управления базами данных (СУБД) — совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями.

По степени универсальности различаются два класса СУБД — системы общего назначения и специализированные системы.

СУБД общего назначения не ориентированы на какую-либо конкретную предметную область или на информационные потребности конкретной группы пользователей. Каждая система такого рода реализуется как программный продукт, способный функционировать на некоторой модели ЭВМ в определенной операционной обстановке. СУБД общего назначения обладает средствами настройки на работу с конкретной БД в условиях конкретного применения.

В некоторых ситуациях СУБД общего назначения не позволяют добиться требуемых проектных и эксплуатационных характеристик (производительность, занимаемый объем памяти и прочее). Тем не менее создание специализированных СУБД весьма трудоемкий процесс и для того, чтобы его реализовать, нужны очень веские основания.

1.1.1 Функции СУБД

Управление данными во внешней памяти Данная функция предоставляет пользователям возможности выполнения самых основных операций, которые осуществляются с данными, — это сохранение, извлечение и обновление информации. Она включает в себя обеспечение необходимых структур внешней памяти как для хранения данных, непосредственно входящих в БД, так и для служебных целей, например, для ускорения доступа к данным.

Управление транзакциями

Транзакция — это последовательность операций над БД, рассматриваемых СУБД как единое целое. Транзакция представляет собой набор действий, выполняемых с целью доступа или изменения содержимого базы данных. Примерами простых транзакций может служить добавление, обновление или удаление в базе данных сведений о некоем объекте. Сложная же транзакция образуется в том случае, когда в базу данных требуется внести сразу несколько изменений. Инициализация транзакции может быть вызвана отдельным пользователем или прикладной программой.

Восстановление базы данных

Одним из основных требований к СУБД является надежность хранения данных во внешней памяти. Под надежностью хранения понимается то, что СУБД должна быть в состоянии восстановить последнее согласованное состояние БД после любого аппаратного или программного сбоя. Обычно рассматриваются два возможных вида аппаратных сбоев:

- мягкие сбои, которые можно трактовать как внезапную остановку работы компьютера (например, аварийное выключение питания);

- жесткие сбои, характеризуемые потерей информации на носителях внешней памяти.

Поддержание надежности хранения данных в БД требует избыточности хранения данных, причем та часть данных, которая используется для восстановления, должна храниться особо надежно. Наиболее распространенным методом поддержания такой избыточной информации является ведение журнала изменений БД.

Поддержка языков

БД Для работы с базами данных используются специальные языки, называемые языками баз данных.

В современных СУБД обычно поддерживается единый интегрированный язык, содержащий все необходимые средства для работы с БД, начиная от ее создания, и обеспечивающий базовый пользовательский интерфейс с базами данных. Стандартным языком наиболее распространенных в настоящее время реляционных СУБД является язык SQL (Structured Query Language — язык

структурированных запросов). Язык SQL позволяет определять схему реляционной БД и манипулировать данными.

Словарь данных

Одной из основополагающих идей рассмотренной выше трехуровневой архитектуры является наличие интегрированного системного каталога с данными о схемах, пользователях, приложениях и т. д. Системный каталог, который еще называют словарем данных, является, таким образом, хранилищем информации, описывающей данные в базе данных. Предполагается, что каталог доступен как пользователям, так и функциям СУБД. Обычно в словаре данных: содержится следующая информация:

- имена, типы и размеры элементов данных;
- имена связей;
- накладываемые на данные ограничения поддержки целостности;
- имена пользователей, которым предоставлено право доступа к данным;
- внешняя, концептуальная и внутренняя схемы и отображения между ними;
- статистические данные, например, частота транзакций и счетчики обращений к объектам базы данных.

Управление параллельным доступом

Одна из основных целей создания и использования СУБД заключается в том, чтобы множество пользователей могло осуществлять параллельный доступ к совместно обрабатываемым данным. Параллельный доступ сравнительно просто организовать, если все пользователи выполняют только чтение данных, поскольку в этом случае они не могут помешать друг другу. Однако, когда два или больше пользователей одновременно получают доступ к базе данных, конфликт с нежелательными последствиями легко может возникнуть, например, если хотя бы один из них попытается обновить данные.

СУБД должна гарантировать, что при одновременном доступе к базе данных многих пользователей подобных конфликтов не произойдет.

Управление буферами оперативной памяти

СУБД обычно работают с БД значительного размера. Понятно, что если при обращении к любому элементу данных будет производиться обмен с

внешней памятью, то вся система будет работать со скоростью устройства внешней памяти. Практически единственным способом реального увеличения этой скорости является буферизация данных в оперативной памяти. В развитых СУБД поддерживается собственный набор буферов оперативной памяти с собственной дисциплиной замены буферов.

Контроль доступа к данным

СУБД должна иметь механизм, гарантирующий возможность доступа к базе данных только санкционированных пользователей и защищающий ее от любого несанкционированного доступа. В современных СУБД поддерживается один из двух широко распространенных подходов к вопросу обеспечения безопасности данных: избирательный подход или обязательный подход. В большинстве современных систем предусматривается избирательный подход, при котором некий пользователь обладает различными правами при работе с разными объектами. Значительно реже применяется альтернативный, обязательный подход, где каждому объекту данных присваивается некоторый классификационный уровень, а каждый пользователь обладает некоторым уровнем допуска.

Поддержка целостности данных

Термин целостность используется для описания корректности и непротиворечивости хранимых в БД данных. Реализация поддержки целостности данных предполагает, что СУБД должна содержать сведения о тех правилах, которые нельзя нарушать при работе с данными, и обладать инструментами контроля за тем, чтобы данные и их изменения соответствовали заданным правилам.

1.1.2 Языки баз данных

В СУБД поддерживается несколько специализированных по своим функциям подъязыков. Их можно разбить на две категории: *f*

язык определения данных БД — ЯОД {DDL — Data Definition Language);

язык манипулирования данными— ЯМД (DML — Data Manipulation, Language).

Язык определения данных

Язык определения данных — описательный язык, с помощью которого описывается предметная область: именуются объекты, определяются их свойства и связи между объектами. Он используется главным образом для определения логической структуры БД.

Схема базы данных, выраженная в терминах специального языка определения данных, состоит из набора определений.

Язык ЯОД используется как для определения новой схемы, так и для модификации уже существующей. Результатом компиляции ЯОД — операторов является набор таблиц, хранимый в системном каталоге, в котором содержатся метаданные — т. е. данные, которые включают определения записей, элементов данных, а также другие объекты, представляющие интерес для пользователей или необходимые для работы СУБД. Перед доступом к реальным данным СУБД обычно обращается к системному каталогу.

Языки манипулирования данными

Язык манипулирования данными содержит набор операторов манипулирования данными, т. е. операторов, позволяющих заносить данные в БД, удалять, модифицировать или выбирать существующие данные.

Множество операций над данными можно классифицировать следующим образом:

1. операции селекции;

2. действия над данными:

включение — ввод экземпляра записи в БД с установкой его связей; *f*

удаление — исключение экземпляра записи из БД с установкой новых связей;

модификация — изменение содержимого экземпляра записи и коррекция связей при необходимости.

Языки манипулирования данными делятся на два типа. Это разделение обусловлено коренным различием в подходах к работе с данными, а следовательно, различием в базовых конструкциях в работе с данными.

Первый тип — это процедурный ЯМД.

Второй тип — это декларативный (непроцедурный) ЯМД.

К процедурным языкам манипулирования данными относятся и языки, поддерживающие операции реляционной алгебры, которую основоположник теории реляционных баз данных Э. Ф. Кодд ввел для управления реляционной базой данных. Реляционная алгебра — это процедурный язык обработки реляционных таблиц, где в качестве операндов выступают таблицы в целом.

Декларативные языки предоставляют пользователю средства, позволяющие указать лишь то, какие данные требуются. Решение вопроса о том, как их следует извлекать, берет на себя процессор данного языка, работающий с целыми наборами записей.

Реляционные СУБД обычно включают поддержку непроцедурных языков манипулирования данными — чаще всего это бывает язык структурированных запросов SQL или язык запросов по образцу QBE.

В настоящее время нормой является поддержка декларативного языка SQL, в основе которого лежит реляционное исчисление, также введенное Э Коддом. Этот язык стал стандартом для языков реляционных баз данных, что позволяет использовать один и тот же синтаксис и структуру команд при переходе от одной СУБД к другой.

Следует отметить, что язык SQL имеет сразу два компонента: язык DDL (ЯОД) для описания структуры базы данных, и язык DML (ЯМД) для выборки и обновления данных.

Другим широко используемым языком обработки данных является язык QBE, который заслужил репутацию одного из самых простых способов извлечения информации из базы данных. Особенно это ценно для пользователей, не являющихся профессионалами в этой области Язык предоставляет графические средства создания запросов на выборку данных с использованием шаблонов Ответ на запрос также представляет собой графическую информацию.

Часть непроцедурного языка ЯМД, которая отвечает за извлечение данных, называется языком запросов Язык запросов можно определить как высокоуровневый узкоспециализированный язык, предназначенный для удовлетворения различных требований по выборке информации из базы данных.[5]

1.2 Теоретические основы теории графов

Графом G называется совокупность из некоторого (обычно конечного) множества V , элементы которого называются вершинами, и некоторого выделенного подмножества E множества V 2 пар элементов множества V (называемых ребрами). Обычно подразумевается, что пары вершин неупорядочены (граф неориентированный) и элементы в каждой паре различны (нет петель). Если рассматриваются упорядоченные пары, граф называется ориентированным (или орграфом). [6]

1.2.1 Основные алгоритмы теори графов.

Поиск в глубину

Поиск в глубину (англ. Depth-first search, DFS) — один из методов обхода графа. Стратегия поиска в глубину, как и следует из названия, состоит в том, чтобы идти «вглубь» графа, насколько это возможно. Алгоритм поиска описывается рекурсивно: перебираем все исходящие из рассматриваемой вершины рёбра. Если ребро ведёт в вершину, которая не была рассмотрена ранее, то запускаем алгоритм от этой нерассмотренной вершины, а после возвращаемся и продолжаем перебирать рёбра. Возврат происходит в том случае, если в рассматриваемой вершине не осталось рёбер, которые ведут в нерассмотренную вершину. Если после завершения алгоритма не все вершины были рассмотрены, то необходимо запустить алгоритм от одной из нерассмотренных вершин.

Поиск в ширину

Поиск в ширину (обход по уровням) – один из алгоритмов обхода графа. Метод лежит в основе некоторых других алгоритмов близкой тематики. Поиск в ширину подразумевает поуровневое исследование графа: вначале посещается корень – произвольно выбранный узел, затем – все потомки данного узла, после этого посещаются потомки потомков и т.д. Вершины просматриваются в порядке возрастания их расстояния от корня.

Пусть задан граф G=(V, E) и корень s, с которого начинается обход. После посещения узла s, следующими за ним будут посещены смежные с s узлы (множество смежных с s узлов обозначим как q; очевидно, что $q \subseteq V$, то есть q – некоторое подмножество V). Далее, эта процедура повториться для вершин смежных с вершинами из множества q, за исключением вершины s, т. к. она

уже была посещена. Так, продолжая обходить уровень за уровнем, алгоритм обойдет все доступные из s вершины множества V. Алгоритм прекращает свою работу после обхода всех вершин графа, либо в случае выполнения наличествующего условия. [7]

Алгоритм Ли

Алгоритм работает на дискретном рабочем поле (ДРП), представляющем собой ограниченную замкнутой линией фигуру, не обязательно прямоугольную, разбитую на прямоугольные ячейки, в частном случае — квадратные. Множество всех ячеек ДРП разбивается на подмножества: «проходимые» (свободные), т. е при поиске пути их можно проходить, «непроходимые» (препятствия), путь через эту ячейку запрещён, стартовая ячейка (источник) и финишная (приемник). Назначение стартовой и финишной ячеек условно, достаточно — указание пары ячеек, между которыми нужно найти кратчайший путь.

Алгоритм предназначен для поиска кратчайшего пути от стартовой ячейки к конечной ячейке, если это возможно, либо, при отсутствии пути, выдать сообщение о непроходимости.

Работа алгоритма включает в себя три этапа: инициализацию, распространение волны и восстановление пути.

Во время инициализации строится образ множества ячеек обрабатываемого поля, каждой ячейке приписываются атрибуты проходимости/непроходимости, запоминаются стартовая и финишная ячейки.

Далее, от стартовой ячейки порождается шаг в соседнюю ячейку, при этом проверяется, проходима ли она, и не принадлежит ли ранее меченной в пути ячейке.

Соседние ячейки принято классифицировать двояко: в смысле окрестности Мура и окрестности фон Неймана, отличающийся тем, что в окрестности фон Неймана соседними ячейками считаются только 4 ячейки по вертикали и горизонтали, в окрестности Мура — все 8 ячеек, включая диагональные.

При выполнении условий проходимости и непринадлежности её к ранее помеченным в пути ячейкам, в атрибут ячейки записывается число, равное количеству шагов от стартовой ячейки, от стартовой ячейки на первом шаге это будет 1. Каждая ячейка, меченая числом шагов от стартовой ячейки становится

стартовой и из неё порождаются очередные шаги в соседние ячейки. Очевидно, что при таком переборе будет найден путь от начальной ячейки к конечной, либо очередной шаг из любой порождённой в пути ячейки будет невозможен.

Восстановление кратчайшего пути происходит в обратном направлении: при выборе ячейки от финишной ячейки к стартовой на каждом шаге выбирается ячейка, имеющая атрибут расстояния от стартовой на единицу меньше текущей ячейки. Очевидно, что таким образом находится кратчайший путь между парой заданных ячеек. Трасс с минимальной числовой длиной пути, как при поиске пути в окрестностях Мура, так и фон Неймана может существовать несколько. Выбор окончательного пути в приложениях диктуется другими соображениями, находящимися вне этого алгоритма. Например, при трассировке печатных плат — минимумом линейной длины проложенного проводника.

Алгоритм Дейкстры

Алгоритм Дейкстры (англ. Dijkstra’s algorithm) — алгоритм на графах, изобретённый нидерландским учёным Эдсгером Дейкстрой в 1959 году. Находит кратчайшие пути от одной из вершин графа до всех остальных. Алгоритм работает только для графов без рёбер отрицательного веса. Алгоритм широко применяется в программировании и технологиях, например, его используют протоколы маршрутизации OSPF и IS-IS.

Каждой вершине из V сопоставим метку — минимальное известное расстояние от этой вершины до a. Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки. Работа алгоритма завершается, когда все вершины посещены.

Инициализация. Метка самой вершины a полагается равной 0, метки остальных вершин — бесконечности. Это отражает то, что расстояния от a до других вершин пока неизвестны. Все вершины графа помечаются как непосещённые.

Шаг алгоритма. Если все вершины посещены, алгоритм завершается. В противном случае, из ещё не посещённых вершин выбирается вершина u, имеющая минимальную метку. Мы рассматриваем всевозможные маршруты, в которых u является предпоследним пунктом. Вершины, в которые ведут рёбра из u, назовём соседями этой вершины. Для каждого соседа вершины u, кроме

отмеченных как посещённые, рассмотрим новую длину пути, равную сумме значений текущей метки u и длины ребра, соединяющего u с этим соседом. Если полученное значение длины меньше значения метки соседа, заменим значение метки полученным значением длины. Рассмотрев всех соседей, пометим вершину u как посещённую и повторим шаг алгоритма. [8]

1.3 Пространственный анализ

К средствам пространственного анализа относятся различные процедуры манипулирования пространственными и атрибутивными данными, выполняемые при обработке запросов пользователя. (Например, операции наложения графических объектов, средства анализа сетевых структур или выделения объектов по заданным признакам).

Для каждого ГИС-пакета характерен свой набор средств пространственного анализа, обеспечивающий решение специфических задач пользователя, в тоже время можно выделить ряд основных функций, свойственных практически каждому ГИС-пакету. Это, прежде всего, организация выбора и объединения объектов в соответствии с заданными условиями, реализация операций вычислительной геометрии, анализ наложений, построение буферных зон, сетевой анализ.

1.3.1 Основные функции пространственного анализа данных

Выбор объектов по запросу: самой простой формой запроса является получение характеристик объекта указанного курсором на экране и обратная операция, когда изображаются объекты с заданными атрибутами. Более сложные запросы позволяют выбирать объекты по нескольким признакам, например по признаку удаленности одних объектов от других, совпадающие объекты, но расположенные в разных слоях и т. д.

Для выбора данных в соответствии с определенными условиями используются SQL- запросы. Для выполнения запросов разной сложности реализованы возможности использования при составлении запросов математических и статистических функций, а также географических операторов, позволяющих выбирать объекты на основании их взаимного расположения в пространстве

(например, находится ли анализируемый объект внутри другого объекта или пересекается с ним).

Обобщение данных может проводиться по равенству значений определенного атрибута, в частности для зонирования территории. Еще один способ группировки - объединение объектов одного тематического слоя в соответствии с их размещением внутри полигональных объектов других тематических слоев.

Геометрические функции: к ним относят расчеты геометрических характеристик объектов или их взаимного положения в пространстве, при этом используются формулы аналитической геометрии на плоскости и в пространстве. Так для площадных объектов вычисляются занимаемые ими площади или периметры границ, для линейных - длины, а также расстояния между объектами и т.д.

Оверлейные операции (топологическое наложение слоев) являются одними из самых распространенных и эффективных средств. В результате наложения двух тематических слоев образуется другой дополнительный слой в виде графической композиции исходных слоев. Учитывая, что анализируемые объекты могут относиться к разным типам (точка, линия, полигон), возможны разные формы анализа: точка на точку, точка на полигон и т.д. Наиболее часто анализируется совмещение полигонов.

Построение буферных зон. Одним из средств анализа близости объектов является построение буферных зон. Буферные зоны - это районы (полигоны), граница которых отстоит на заданном расстоянии от границы исходного объекта. Границы таких зон вычисляются на основе анализа соответствующих атрибутивных характеристик. При этом ширина буферной зоны может быть как постоянной, так и переменной. Например, буферная зона вокруг источника электромагнитного излучения, будет иметь форму круга, а зона загрязнения от дымовой трубы завода с учетом розы ветров будет иметь форму близкую к эллипсу.

Сетевой анализ позволяет пользователю проанализировать пространственные сети связных линейных объектов (дороги, линии электропередач и т. д.). Обычно сетевой анализ служит для задач определения ближайшего, наибо-

лее выгодного пути, определения уровня нагрузки на сеть, определение адреса объекта или маршрута по заданному адресу и другие задачи.

1.3.2 Анализ пространственного распределения объектов

Анализ пространственного распределения объектов. Фактически во многих случаях необходимо знать не только объем пространства, занимаемый объектами, но и расположение объектов в пространстве, которое может характеризоваться количеством объектов в определенной области, например, распределение численности населения. Наиболее распространены методы анализа распределения точечных объектов. Мерой точечного распределения служит плотность. Она определяется как результат деления числа точек на значение площади территории, на которой они расположены. Кроме плотности распределения можно оценить форму распределения. Точечные распределения встречаются в одном из четырёх возможных вариантов: равномерном (если число точек в каждой малой подобласти такое же, как и в любой другой подобласти), регулярном (если точки, разделённые одинаковыми интервалами по всей области, расположены в узлах сетки), случайном, кластерном (если точки собраны в тесные группы).

Точечные распределения могут описываться не только количеством точек в пределах подобластей. Часто анализируются локальные отношения внутри пар точек. Вычисление этого статистического показателя включает определение среднего расстояния до ближайшей соседней точки среди всех возможных пар ближайших точек. Данный метод позволяет оценить меру разреженности точек в распределении.

Распределение линий также оценивается по плотности. Обычно вычисления выполняются для сравнения разных географических областей, например по густоте гидрографической сети. Линии могут также оцениваться по близости и возможным пересечениям. Другими важными характеристиками являются ориентация, направленность и связанность.

Анализ распределения полигонов подобен анализу распределения точек, однако при оценке плотности определяют не количество полигонов на единицу площади, а относительную долю площади, занимаемой полигоном. [9]

2. Практическая часть

2.1 Обзор графовых баз данных.

Sones GraphDB.

Эта графовая БД разработана компанией Sones в 2009 г. и поддерживалась до конца 2011 г., после чего компания обанкротилась. За время поддержки была выпущена версия 2.1, доступная в двух вариантах: версия Community распространяется по лицензии AGPL v3, для коммерческих же проектов необходимо приобретать версию Enterprise. Кроме того, важное отличие платной версии от бесплатной состоит в наличии возможности устойчивого хранения базы на жестком диске. Бесплатная же версия позволяет хранить данные только в оперативной памяти (in-memory). Поскольку компания-разработчик прекратила свое существование, единственная версия БД, которую можно протестировать, — бесплатная версия Community, т. е. тестирование будет ограничено небольшими объемами данных, целиком помещающимися в оперативную память. Соответственно, работать с такими данными можно только в случае либо их маленького объема, либо постоянной очистки текущего хранилища от старых данных. Sones не имеет встроенной поддержки алгоритмов обхода графов, в связи с чем для тестирования задачи поиска соседей был реализован поиск в ширину [9]. Кроме того, для тестирования поиска пересечений множеств соседних вершин была реализована работа с множествами найденных вершин.

Neo4J GraphDB.

Эта графовая БД разработана компанией Neo Technology в 2009 г. Она доступна в трех вариантах: Community, Advanced и Enterprise. Версия Community распространяется по лицензии AGPL v3, для коммерческих же проектов необходимо приобретать версию Advanced, включающую дополнительные возможности мониторинга состояния базы. Версия Enterprise, кроме того, поддерживает резервное копирование и масштабируемость. На сегодняшний день Neo4J является наиболее популярной графовой БД, в первую очередь ввиду того, что бесплатная версия предлагает все необходимые инструменты для полноценной работы с графами. Neo4J, в отличие от графовой БД Sones, может устойчиво хранить данные на жестком диске. Следовательно, объем хранящейся информации ограничен лишь объемами жесткого диска. Для дос-

тижения максимальной производительности в Neo4J существует два типа кэширования: файловый кэш (file buffer cache) и объектный кэш (object cache). Первый кэширует данные с жесткого диска, целью чего является увеличение скорости чтения/записи на жесткий диск. Второй кэш хранит в себе различные объекты графа: вершины, ребра и свойства в специальном оптимизированном формате для увеличения производительности обходов графа. Neo4J обладает режимом импорта большого объема данных BatchInserter [10]. В этом режиме происходит отключение транзакций в БД, следствием чего является резкое увеличение скорости импорта. Кроме того, в Neo4J поддерживаются алгоритмы обхода графов, в том числе можно сделать обход в ширину до заданного уровня, что как раз и необходимо для решения задачи поиска соседних вершин. Для пересечения двух множеств, как и в случае с базой данных Sones, была реализована работа с множествами.

DEX GraphDB.

Эта база разработана компанией Sparsity Technologies в 2008 г. Существует три типа лицензий, по которым предоставляется право пользования DEX GraphDB: Community, Commercial и Education. При наличии лицензии Community в базе суммарно может содержаться не более 1 млн вершин и разрешен доступ к чтению данных только одному потоку. По лицензии Commercial отсутствуют какие-либо ограничения, но годовая стоимость владения лицензией зависит от суммарного количества объектов в базе и количества потоков, имеющих доступ к чтению данных. Лицензия Education предназначена для бесплатного предоставления исследовательским и учебным некоммерческим проектам. DEX GraphDB, как и Neo4J, имеет полноценную поддержку устойчивого хранения данных. Но в отличие от Neo4J ядро DEX написано на C++, что очень хорошо сказывается на производительности. Графовая БД DEX имеет только один объектный кэш, который держит в оперативной памяти все часто используемые объекты хранилища [11]. БД DEX имеет широкие возможности по обходу графов. Она предоставляет множество встроенных алгоритмов обхода графов, таких как поиск в глубину, поиск в ширину, Дейкстры и поиска компо- нентов сильной связности. Кроме того, существует встроенная поддержка работы с множествами. Таким образом, обе поставленные задачи ана-

лиза биллинговой информации можно решить, используя только встроенные средства БД.

В дальнейшам для анализа будет использоваться графовая БД Neo4J.

2.2 Анализ использования графовой БД Neo4J в геолокационных приложениях.

Язык запросов Cypher.

Для работы с базой данной neo4j используется специальный язык Cypher. Cypher является декларативным языком запросов для графов, который позволяет выразительно и эффективно выполненить запросы и обновленить базу графов. Cypher является относительно простым, но также мощным языком. Сложные запросы к базе данных легко могут быть выражены через Cypher.

Cypher разработан как человекочитаемый язык запросов, подходящий как для разработчиков, так и специалистов аналитиков. Задачей языка состоит в том, чтобы сделать простые вещи легко, и сложные вещи возможными. Его конструкции основаны на английской языке и иконографии, которые помогают сделать запросы более понятными. Язык старается быть удобным как для чтения, так и для записи.

Cypher заимствует свою структуру из SQL - запросы строятся с использованием различных операций.

Операции могут быть соединены друг с другом, и они распределяют промежуточные результаты между собой. Например, совпадающие переменные из одного пункта MATCH имеют один контекст между операциями.

Язык запросов состоит из нескольких различных операций:

- MATCH: Шаблон графа для поиска. Наиболее распространенный способ получения данных из графа.
- WHERE: Используется как часть MATCH, OPTIONAL MATCH и WITH. Добавляет ограничения на шаблон, или фильтрует промежуточные результаты проходящий через WITH.
- RETURN: Что будет возвращено.

Использование neo4j в приложении.

Для текущего проекта будет использоваться следующая модель данных, представленная на рисунке 3.1

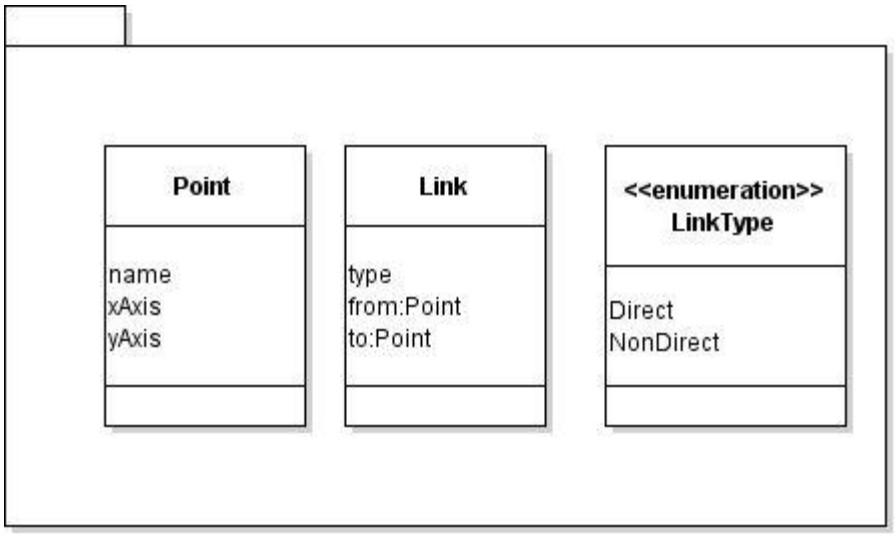


Рис.3.1 Модель данных приложения

Для создания в базе данных вершин графа используется следующий запрос:

```
CREATE (p:Point {name:'First', xAsis:15.15, yAx-
                is:15.15})
  CREATE (p:Point {name:'Second', xAsis:20.20, yAx-
                is:20.20})
```

Для создания ребер графа используется следующий запрос:

```
MATCH (f:Point {name:'First'}) MATCH
(s:Point{name:'Second'}) CREATE (f)-[d:DIRECT]->(s)
```

Для анализа работы с алгоритмами будет использоваться следующий граф.

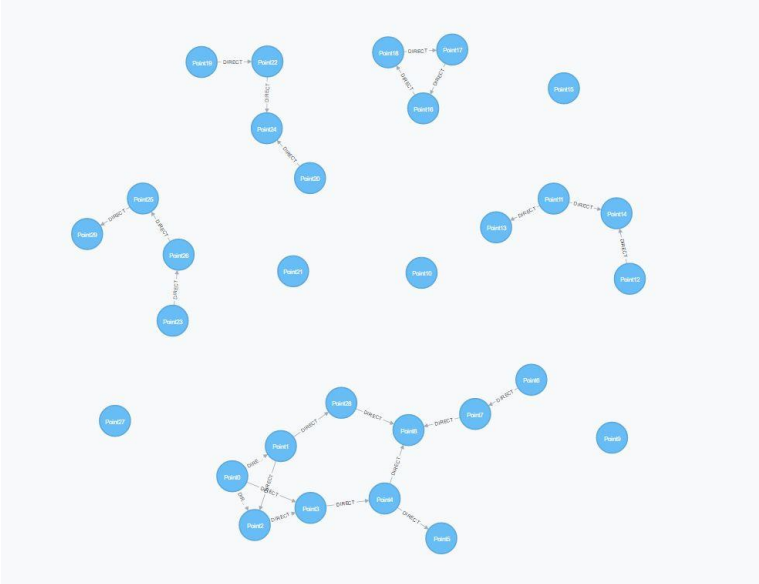


Рис. 3.2 – Тестовый граф

В базе данных neo4j также присутствуют реализации алгоритмов работы с графами, таких как поиск связанности вершин.

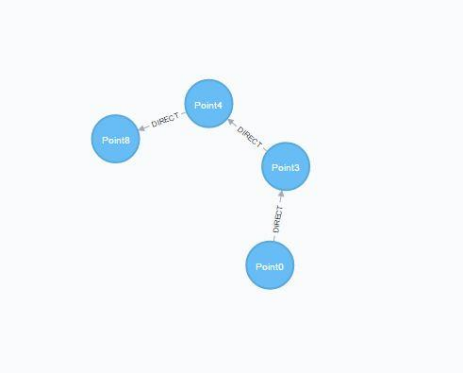
```
MATCH (f:Point{name:'Point0'})-[*]-
>(s:Point{name:'Point8'}) RETURN count(*)>0
```

Данный запрос вернет булево значение в зависимости связаны ли 2 вершины.

Другим встроенным алгоритмом является нахождение кратчайшего пути

```
MATCH (f:Point { name:"Point0" }),(s:Point {
name:"Point8" }), p = shortestPath((f)-[*]-(s)) RETURN p
```

Результатом выполнения будет являться подграф являющимся кратчайшим путем для данного графа.



3.3 – Результат нахождения кратчайшего пути для тетсового графа

Для примения пространственного анализа Neo4J в java-приложения имеется библиотека Neo4j Spatial. Данная библиотека позволяет выполнять следующие функции:

- Импорт данных из файлов ESRI Shapefile и Open Street Map

- Поддержка разных геометрических фигур

- Реализации структуры данных R-дерево для поиска пространственных данных

- Поддержка топологических операций поиска (содержит, в пределах, пересекается, охватывает, не пересекаются, и т.д.)

- Возможность применения пространственных операций на любом графе

Для выполнения поиска точек в пределах определенной геометрической фигуры может примениться следующий запрос, выполненный через библиотеку Neo4j Spatial:

```
      GraphDatabaseService database = new GraphDatabaseFacto-
      ry().newEmbeddedDatabase(storeDir);
    try {
        SpatialDatabaseService spatialService = new SpatialDatabaseService(database);
        Layer layer = spatialService.getLayer("layer_roads");
        SpatialIndexReader spatialIndex = layer.getIndex();

        Search searchQuery = new SearchIntersectWindow(new Envelope(xmin, xmax, ymin,
ymax));
        spatialIndex.executeSearch(searchQuery);
    List<SpatialDatabaseRecord> results = searchQuery.getResults();
    } finally {
    database.shutdown();
    }
```

Результатом выполнения будут вершины графа, координаты которого будут располагаться внутри параметров xmin, xmax, ymin, ymax.

Заключение

В статье проведено исследование графовых БД с целью определения возможности их применения в геолакационных приложениях. Из полученных результатов можно сделать вывод о том, что все они вполне жизнеспособны и показывают неплохие результаты. БД Neo4J может быть использована в качестве хранилища для обработки небольших подграфов, объемом до 1,5 млн объектов, поскольку с такими графами она работает очень быстро. Кроме того, Neo4J, являясь наиболее популярной графовой БД, обладает огромным набором различных API для разных языков программирования и множеством дополнительных функций, что делает ее наиболее простой в работе.

Литература

[1] Dominguez-Sal D., Urbon-Bayes P., Gimenez-Vano A., Gomez-Villamor S., Martınez-Bazan N., Larriba-Pey J.L. Survey of graph database performance on the HPC scalable graph analysis benchmark. Proceedings of the 2010 int. conf. on Web-age information management (WAIM'10). Berlin, Heidelberg, Springer-Verlag, 2010, pp. 37–48.

[2] Angles R., Gutierrez C. Survey of Graph Database Models. ACM Computing Surveys, 2008, vol. 40 (1), pp. 1:1–1:39.

[3] Angles R. A comparison of current graph database models. Proceedings of the 2012 IEEE 28th Int. Conf. on Data Engineering Workshops, (ICDEW'12). Wash., IEEE Computer Society, 2012, pp. 171–177.

[4] Vicknair C., Macias M., Zhao Z., Nan X., Chen Y., Wilkins D. A comparison of a graph database and a relational database: a data provenance perspective. Proceedings of the 48th Annual South-East Regional Conf. (ACM SE'10). N.Y., ACM, 2010, pp. 42:1–42:6.

[5] Головчинер М.Н. Базы данных Основные понятия, модели данных, процесс проектирования Курс лекций

[6] Селезнева С.Н. Лекции по курсу «Дискретные модели». Москва

[7] http://kvodo.ru/search-width.html

[8] http://suvitruf.ru/2012/05/13/1176/

[9] Gis-lab Пространственный анализ растровых данных

[10] Neo4J BatchInserter. URL: http://docs.neo4j.org/chunked/milestone/batchinsert.html.

[11] DEX. Cache configuration. URL: http://www.sparsity-technologies.com/dex_tutorials5?name=Configuration

[12] Graphs and Graph Algorithms in T-SQL. URL: http://hansolav.net/sql/graphs.html