Как правило, алгоритм принадлежит классу методов Монте-Карло для марковскихцепей (MarkovChainMonteCarlo, MCMC). Примером такого алгоритма является сэмплирование по Гиббсу, которое состоит в том, чтобы на каждом шаге фиксироватьвсе переменные, кроме одной, и выбирать оставшуюся переменную согласно распределению вероятности этой переменной при условии всех остальных. Методы второй группы вариационные алгоритмы. В них сначала задается параметризованное семейство распределений над скрытыми переменными, а затем с помощью EM-алгоритма ищется распределение из этого семейства, наиболее близкое к исходному апостериорному распределению.

В качестве языка программирования, на котором выполнялась реализация практических исследований, был выбран язык Java. Это объектно-ориентированный язык, который хорошо подходит для прикладных задач. Кроме того, программы, написанные и скомпилированные на Java можно запускать на любой операционной системе, где поддерживается запуск виртуальной машины Java. При проверке орфографии на этапе предобработки данных использовалась библиотека Snowball и MyStem. Для оценки интерпретируемости с помощью Google использовалась разрабатываемая в ИСП РАН утилита для скачивания веб-страниц из сети Интернет. Обе библиотеки реализованы на Java, что также является доводов в пользу данного языка программирования.

Использовались готовые реализации тематических моделей на языке C. Выбор этихреализаций обусловлен тем, что они принадлежат авторам исследуемых методов тематического моделирования. Кроме того, язык C хорошо подходит для таких задач, где производится большое количество вычислений.

Заключение

В процессе выполнения работы были исследованы методы определения тематической направленности текстового содержимого микроблогов и реализован алгоритм автоматической оценки интерпретируемости результатов тематического моделирования текстов микроблогов. Так же были исследованы существующие методы тематического моделирования и способы оценки их качества. Выполнена экспериментальная оценка интерпретируемости методов тематического моделирования текстов микроблогов с использованием разработанных методов. Разработаны и реализованы методы автоматической оценки интерпретируемости результатов тематического моделирования по ключевым словам тем. А также были проведены расчёты популярности сообщений (твиттов) при помощи метрик, на основании которых был осуществлён прогноз тенденции популярности.

ПРИНЯТИЕ РЕШЕНИЯ О ДЕНОРМАЛИЗАЦИИ БАЗЫ ДАННЫХ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Чочиева А.С.

Калугина М. А. - к.ф-м.н., доцент

В ходе проектирования баз данных может возникнуть вопрос: “Необходима ли её большая нормализация и если да, то до какой степени?” Большинство источников указывают, что всегда нужно стремиться к нормализации базы данных - “больше таблиц, меньше столбцов”, к большему разделению на сущности. Нормализация является важным процессом проектирования базы данных. Но могут ли у неё иметься недостатки?

Нормализация базы данных — это процесс реструктурирования реляционной базы данных в соответствии с нормальными формами отношения с целью снижения избыточности данных и обеспечения их целостности. Денормализация — обратный этому процесс.

Сравним эти два процесса для выделения очевидных аргументов в пользу денормализации.

Нормализация экономит место, но это место стоит дешевле [1]. В денормализованной базе данных может находиться 10000 записей, например, названия страны, что занимает гораздо больше места, чем просто внешний целочисленный ключ на одну строку в другой таблице. Но терабайтные носители сейчас дешевле, чем ранее.

Нормализация упрощает изменение строки, но чтение обычно происходит чаще, а нормализация замедляет чтение. Таким образом, если данные редко меняются, имеется смысл задуматься о денормализации.

Расчетные данные [3]. Существуют ситуации, когда есть запросы с долгими, сложными вычислениями, потребляющими много ресурсов. В таких случаях можно выделить столбцы, в которых будут храниться часто используемые расчётные значения, чтобы избежать потери ресурсов и времени на их повторное вычисление.

Нормализация снижает производительность. Это и есть наиболее часто упоминаемая причина денормализации. Сильная нормализация может привести к большому количеству соединений таблиц, что выливается в объемные составные SELECT-запросы с большим количеством JOIN операций- это будет замедлять работу базы данных. В источниках также упоминается [1], что нормализация часто становится причиной неполадок в работе программного обеспечения из-за ошибок в сложных запросах. В некоторых случаях целесообразно хранить редко изменяемые данные просто в текстовом файле, загружая их по мере необходимости.

С учетом проведенного анализа, денормализацию не рекомендуется применять в следующих случаях:

- когда присутствуют отношения “один-ко-многим” (например, у пользователя есть возможность ввести несколько адресов, поэтому имеет смысл отделить адреса в отдельную таблицу);
- когда есть частая необходимость создать уникальный список из этих данных (например, для выпадающего меню, из которого будет выбирать пользователь);
- когда данные часто обновляются.

С целью оптимизации работы с данными принимать решение о денормализации рекомендуется лишь после проектирования нормализованной базы данных [2,3]. Необходимо проанализировать характер и частоту запросов и только после этого, при необходимости, аккуратно, с учётом сохранения целостности данных, произвести денормализацию [2, 3]. В таком случае её побочные эффекты снижаются до минимума, при этом обеспечивая увеличение быстродействия базы данных.

К недостаткам денормализации можно отнести:

1. большее потребление места на диске (из-за дублирования данных);
2. возможное замедление операций вставки, обновления и удаления данных;
3. аномалии данных (нарушение целостности данных): в какой-то момент данные могут измениться в нескольких местах, в таком случае нужно обеспечить корректное обновление их копий [3] (или в случае с расчётными данными – пересчитать расчётные значения, полученные из изменённых данных);
4. документация: денормализованную базу данных сложнее поддерживать, поэтому денормализацию базы данных следует подробно документировать (если возникнет необходимость поменять структуру базы данных, то нужно будет учитывать все предыдущие изменения) [3];
5. денормализация может потребовать написания большего количества кода [2], например, триггеров и процедур, обеспечивающих целостность данных, а если она проводится на существующей рабочей базе данных, то может понадобиться изменение существующих запросов и написание кода для обновления уже имеющихся записей [3].

Таким образом, залог быстрой и исправно работающей базы данных – это нахождение оптимального равновесия между нормализацией и денормализацией. Но началом и основой проектирования базы данных остаётся нормализация.

Список использованных источников:

1.Scott Selikoff , «Why too much Database Normalization can be a Bad Thing», November 19th, 2008.

2.Michelle A. Poolet, «Responsible Denormalization: How to break the rules and get away with it», 2002 Penton Media, Inc.

3.Блог компании Латера Софтвер, «Зачем нужна денормализация баз данных, и когда ее использовать», 9 апреля 2016

CHLORINE ENGINE И МИНИ-ИГРА НА НЕМ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Шпаков Н.И.

Жвакина А.В. – к.т.н., доцент

Игры все больше и больше заполоняют наш мир. Кто-то считает это лишь простой забавой, кто-то бизнесом, а для кого-то это жизнь или искусство. В любом случае, нельзя отрицать большого влияния этой отрасли на IT в целом. Развиваются технологии для обработки графики, звука, появляются новые технологии анимации и рендера, увеличивается мощность железа. Так или иначе, игры являются двигателем прогресса в мире компьютерных технологий.

В данной работе представлена игра, написанная на движке Clorineengine и сам движок, написанный с использованием библиотек SDL2 (окно и обработка событий), GLEW (для вытягивания функций OpenGL), OpenAL (библиотека для стереозвука), GLM (математическая библиотека), picoPNG (декодирование PNG файлов), freetype (преобразование текста в битмапы). В качестве языка программирования выбран C++ 11. Для сборки проекта используется технология CMake. Готовый проект с открытым кодом можно найти по ссылке на github: [https://github.com/ShpakovNikita/Chlorine-5].

Движок умеет обрабатывать пользовательский ввод, рисовать квадраты на поверхности (под углом также), проигрывать звуки с учетом модели дистанции, делать рэйкасты, проверять коллизии разными способами, отрисовывать простейший свет, рисовать GUI, динамический текст по шрифту, работать с анимациями спрайтов и многое другое. Отрисовка происходит посредством передачи данных о каждом полигоне в вершинный шейдер, а затем передачи его во фрагментный шейдер для дальнейшего вывода цвета каждого пикселя на экран. Звуки реализованы через модель источников и слушателя, где для каждого динамического объекта создается свой источник и для слушателя, «привязанного» к камере, идет просчет доходящего звука от источников. Игра же, по сути, в игровом цикле делает все эти действия последовательно, с учетом ООП модели.

Игра является относительно простым примером работы с движком, задействуя абсолютно все его возможности и реализуя множество алгоритмов совместно с функциями движка. Окружение в игре генерируется случайным образом с помощью алгоритма процедурной генерации подземелья, и динамически подбирает тайлы для создания изометрического эффекта. Для изометрического эффекта используется также zbuffer. Реализован также искусственный интеллект, который представлен системой конечных автоматов и