МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра математической теории упругости и биомеханики

Проектирование информационной системы транспортной компании

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 442 группы

направления 09.03.03 – Прикладная информатика

механико-математического факультета

Калашникова Сергея Игоревича

Научный руководитель
к.ю.н., доцент _____ Р.В. Амелин
подпись, дата

Зав. кафедрой
д.ф.-м.н., профессор _____ Л.Ю. Коссович
подпись, дата

Саратов 2021

Введение

На протяжении долгого промежутка времени одно оставалось неизменным - хранение данных в реляционных базах. Было несколько попыток изменить это положение, и некоторые из них достигли успеха в отдельных нишах рынка, но в целом хранение данных с точки зрения архитектуры всегда сводилось к использованию реляционных баз данных. Стабильность такой ситуации обеспечивает много преимущесrв. Данные организаций существуют намного дольше, чем программы, которые их обрабатывают. Это неплохо, что существуют стабильные хранилища данных, понятные и доступные для многих платформ прикладного программирования. SQLite напрямую не сопоставим с клиент-серверными СУБД SQL, такими как MySQL, Oracle, PostgreSQL или SQL Server, поскольку SQLite пытается решить другую проблему.

Клиент-серверные СУБД SQL стремятся реализовать общий репозиторий корпоративных данных. Они подчеркивают масштабируемость, параллелизм, централизацию и контроль. SQLite стремится обеспечить локальное хранение данных для отдельных приложений и устройств. SQLite подчеркивает экономичность, эффективность, надежность, независимость и простоту. Поскольку база данных SQLite не требует администрирования, она хорошо работает на устройствах, которые должны работать без экспертной поддержки человека. SQLite хорошо подходит для использования в мобильных телефонах, телеприставках, телевизорах, игровых консолях, камерах, часах, кухонной технике, термостатах, автомобилях, станках, самолетах, дистанционных датчиках, дронах, медицинских устройствах и роботах: "интернет вещей".

Основное содержание работы

В первом разделе рассказывается об основных способах использования SQLite. Это удобная встраиваемая база данных, которая доступна в исходных кодах. SQLite отлично работает в качестве движка базы данных для большинства веб-сайтов с низким и средним трафиком (то есть для большинства веб-сайтов). Объем веб-трафика, который может обрабатывать SQLite, зависит от того,

насколько интенсивно веб-сайт использует свою базу данных. Вообще говоря, любой сайт, который получает менее 100 тысяч просмотров в день, должен хорошо работать с SQLite. Цифра 100 тысяч просмотров в день-это консервативная оценка, а не жесткая верхняя граница. Было продемонстрировано, что SQLite работает с 10-кратным объемом трафика. SQLite успешно используется в качестве хранилища данных для серверных приложений, запущенных в центре обработки данных, или, другими словами, об использовании SQLite в качестве базового механизма хранения для сервера базы данных конкретного приложения.

С этим шаблоном общая система все еще клиент/сервер: клиенты отправляют запросы на сервер и получают ответные ответы по сети. Но вместо отправки общего SQL и получения сырого содержимого таблицы клиентские запросы и ответы сервера являются высокоуровневыми и специфичными для приложений. Сервер переводит запросы в несколько SQL-запросов, собирает результаты, выполняет постобработку, фильтрацию и анализ, а затем строит ответ высокого уровня, содержащий только необходимую информацию.

Считается, что SQLite часто быстрее, чем клиент/серверный компонент SQL database engine в этом сценарии. Запросы базы данных сериализуются сервером, поэтому параллелизм не является проблемой. Параллелизм также улучшается" сегментацией базы данных": использование отдельных файлов базы данных для разных поддоменов. Например, сервер может иметь отдельную базу данных SQLite для каждого пользователя, так что сервер может обрабатывать сотни или тысячи одновременных подключений, но каждая база данных SQLite используется только одним соединением.

В качестве языка программирования в данной работе был использован Go (часто также встречается название Golang). Go – компилируемый, многопоточный, строготипизированный язык программирования, разработанный в компании Google и представленный в 2009 году. По духу Go подобен языку C – это компактный и эффективный язык программирования с низкоуровневыми возможностями, такими как указатели. Однако Go обладает множеством особенностей, характерных для очень высокоуровневых языков, таких как

поддержка Unicode строк, высокоуровневые структуры данных, автоматическая сборка мусора и высокоуровневая поддержка взаимодействий между потоками, основанная на обмене сообщениями, а не на блокировках и разделяемых данных. Кроме того, язык Go имеет обширную и всестороннюю стандартную библиотеку. Go проектировался с прицелом на эффективное масштабирование, благодаря чему его можно использовать для создания очень больших приложений и компиляции даже очень больших программ за секунды на единственном компьютере.

Молниеносная скорость компиляции обеспечивается отчасти простотой синтаксического анализа программ на этом языке, но главным образом благодаря особенностям управления зависимостями. Например, если файл main.go зависит от файла pkg1.go, который, в свою очередь, зависит от файла pkg2.go, в обычных компилирующих языках для компиляции файла main.go необходимо иметь объектные модули, полученные в результате компиляции обоих файлов, pkg1.go и pkg2.go. Но в Go все, что экспортирует pkg2.go, включено в объектный модуль для файла pkg1.go, поэтому для компиляции main.go достаточно иметь только объектный модуль для файла pkg1.go. Для случая с тремя файлами это едва ли имеет большое значение, но в огромных приложениях с большим количеством зависимостей эта особенность дает весьма значительный прирост скорости компиляции. Благодаря высокой скорости компиляции программ на языке Go появляется возможность использовать этот язык в областях, где обычно применяются интерпретируемые языки. 15 Языки программирования, такие как C и C++, требуют от программистов выполнения массы работы, когда дело доходит до управления памятью, которую можно переложить на плечи компьютера, особенно в многопоточных приложениях, где учет использования динамической памяти может оказаться невероятно сложной задачей. В последние годы ситуация в этой области в языке C++ намного улучшилась благодаря появлению «интеллектуальных» указателей, но он пока не способен догнать язык Java с его библиотекой поддержки многопоточной модели выполнения. Язык Java освобождает программиста от бремени управления памятью с помощью механизма сборки мусора.

Поддержка многопоточной модели выполнения в языке C++ в настоящее время включена в состав стандартной библиотеки, однако в языке C она реализована только в виде сторонних библиотек. Но, несмотря на все это, создание многопоточных программ на языке C, C++ или Java требует от программиста немалых усилий, чтобы обеспечить своевременное приобретение и освобождение ресурсов. Все сложности, связанные с учетом ресурсов в языке Go, берут на себя компилятор и среда выполнения. Для управления памятью в Go имеется механизм сборки мусора, что избавляет от необходимости использовать «интеллектуальные» указатели или освобождать память вручную. А поддержка параллелизма в языке Go реализована в форме механизма взаимодействующих последовательных процессов (Communicating Sequential Processes, CSP), основанного на идеях специалиста в области теории вычислительных машин и систем Чарльза Энтони Ричарда Хоара (C. A. R. Hoare), благодаря которому во многих многопоточных программах на языке Go вообще отпадает необходимость блокировать доступ к ресурсам. Кроме того, в языке Go имеются так называемые go-подпрограммы (goroutines) – очень легковесные процессы, которых можно создать великое множество. Выполнение этих процессов автоматически будет распределяться по доступным процессорам и ядрам, что обеспечивает возможность более тонкого деления программ на параллельно выполняющиеся задачи, чем это позволяют другие языки программирования, основанные на потоках выполнения. Фактически поддержка параллелизма в языке Go реализована настолько просто и естественно, что при переносе однопоточных программ на язык Go часто обнаруживается возможность параллельного выполнения нескольких задач, ведущая к увеличению скорости выполнения и более оптимальному использованию машинных ресурсов. Go – практичный язык, где во главу угла поставлены эффективность программ и удобство программиста. Например, встроенные и определяемые пользователем типы данных в языке Go существенно отличаются – операции с первыми из них могут быть значительно оптимизированы, что невозможно для последних. В Go имеются также два встроенных фундаментальных типа коллекций: срезы (slices) (фактически ссылки

на массивы переменной длины) и отображения (maps) (словари, или хеши пар ключ/значение). Коллекции этих типов высокооптимизированы и с успехом могут использоваться для решения самых разных задач. В языке Go также поддерживаются указатели (это действительно компилируемый язык программирования – в нем отсутствует какая-либо виртуальная машина, снижающая производительность), что позволяет с непринужденностью создавать собственные, весьма сложные типы данных, такие как сбалансированные двоичные деревья.

Hyper Text Markup Language (HTML) — язык разметки гипертекста – предназначен для написания гипертекстовых документов, публикуемых в World Wide Web. Гипертекстовый документ — это текстовый файл, имеющий специальные метки, называемые тегами, которые впоследствии опознаются браузером и используются им для отображения содержимого файла па экране компьютера. С помощью этих меток можно выделять заголовки документа, изменять цвет, размер и начертание букв, вставлять графические изображения и таблицы. Но основным преимуществом гипертекста перед обычным текстом является возможность добавления к содержимому документа гиперссылок — специальных конструкций языка HTML, которые позволяют щелчком мыши перейти к просмотру другого документа. HTML-документ состоит из двух частей: собственно, текста, т. е. данных, составляющих содержимое документа, и тегов — специальных конструкций языка HTML, используемых для разметки документа и управляющих его отображением. Теги языка HTML определяют, в каком виде будет представлен текст, какие его компоненты будут исполнять роль гипертекстовых ссылок, какие графические или мультимедийные объекты должны быть включены в документ. Графическая и звуковая информация, включаемая в HTML-документ, хранится в отдельных файлах. Программы просмотра HTML-документов (браузеры) интерпретируют флаги разметки и располагают текст и графику на экране соответствующим образом. Для файлов, содержащих HTMLдокументы приняты расширения .htm или .html. В большинстве случаев теги используются парами. Пара состоит из открывающего <имя_тега> и

закрывающего тегов. Действие любого парного тега начинается с того места, где встретился открывающий тег, и заканчивается при встрече соответствующего закрывающего тега. Часто пару, состоящую из открывающего и закрывающего тегов, называют контейнером, а часть текста, окаймленную открывающим и закрывающим тегом, — элементом. Последовательность символов, составляющая текст может состоять из пробелов, табуляций, символов перехода на новую строку, символов возврата каретки, букв, знаков препинания, цифр, и специальных символов (например, #, +, $, @), за исключением следующих четырех символов, имеющих в HTML специальный смысл: <(меньше), >(больше), &(амперсанд) и "(двойная кавычка). Если необходимо включить в текст какой-либо из этих символов, то следует закодировать его особой последовательностью символов.

Самым главным из тегов HTML является одноименный тег . Он всегда открывает документ, так же, как тег должен непременно стоять в последней его строке. Эти теги обозначают, что находящиеся между ними строки представляют единый гипертекстовый документ. Без этих тегов браузер или другая программа просмотра не в состоянии идентифицировать формат документа и правильно его интерпретировать. HTML-документ состоит из двух частей: заголовок (head) и тела (body), расположенных в следующем порядке: Заголовок документа Тело документа Чаще всего в заголовок документа включают парный тег , определяющий название документа. Многие программы просмотра используют его как заголовок окна, в котором выводят документ. Программы, индексирующие документы в сети Интернет, используют название для идентификации страницы.

Хорошее название должно быть достаточно длинным для того, чтобы можно было корректно указать соответствующую страницу, и в то же время оно должно помещаться в заголовке окна. Название документа вписывается между открывающим и закрывающим тегами. Тело документа является обязательным элементом, так как в нем располагается весь материал документа. Тело документа размещается между тегами. Все, что размещено между этими тегами, интерпретируется браузером в соответствии с правилами языка HTML позволяющими корректно отображать страницу на экране монитора. Текст в

HTML разделяется на абзацы при помощи тега <p>. Он размещается в начале каждого абзаца, и программа просмотра, встречая его, отделяет абзацы друг от друга пустой строкой. Использование закрывающего тега необязательно. Если требуется «разорвать» текст, перенеся его остаток на новую строку, при этом, не выделяя нового абзаца, используется тег разрыва строки. Он заставляет программу просмотра выводить стоящие после него символы с новой строки. В отличие от тега абзаца, тег не добавляет пустую строку. У этого тега нет парного закрывающего тега.

Язык HTML поддерживает логическое и физическое форматирование содержимого документа. Логическое форматирование указывает на назначение данного фрагмента текста, а физическое форматирование задает его внешний вид. При использовании логического форматирования текста браузером выделяются различные части текста в соответствии со структурой документа. Чтобы отобразить название, используется один из тегов заголовка. Заголовки в типичном документе разделяются по уровням. Язык HTML позволяет задать шесть уровней заголовков: h1 (заголовок первого уровня), h2, h3, h4, h5 и h6. Заголовок первого уровня имеет обычно больший размер и насыщенность по сравнению с заголовком второго уровня. Пример использования тегов заголовков: l. Название главы.

Стили позволяют одним действием применить сразу всю группу атрибутов форматирования. Они, как правило, записываются в отдельный файл и 25 подключаются к нужному документу, то есть к каркасу всех страниц сайта. Основная задача стилей в том, чтобы отделить дизайн документа от его содержимого, то есть внешний вид от логической структуры. Разделение такого плана позволяет уменьшить сложность и повторяемость в структурном содержимом страницы, а, следовательно, упростит написание любого проекта. В случае необходимости изменения стилевого оформления сайта или полной его замены, достаточно просто заменить файл стилей, без необходимости обращения к каждой странице сайта. Это значительно упростило создание единого стиля для web-сайта. Основными принципами CSS являются: - Наследование - Каскадирование При указании стиля для элемента часть свойств CSS,

объявленные для элементов-предков, наследуются элементами потомками. Этот эффект называется наследованием CSS. Наследуемые свойства можно переопределить, задав индивидуальные свойства стилей для нужного элемента.

Каскадирование это одна из важнейших особенностей CSS, с помощью которой браузер определяет значения каких свойств стилей будут применены к тому или иному элементу при возникновении конфликта свойств. Конфликт свойств может возникнуть, когда для одного элемента определено более одного правила с одинаковым приоритетом, и они содержат одинаковые свойства, но с разными значениями, то есть, когда происходит конфликт значений этих правил. Чтобы разрешить такие конфликты, вводятся правила приоритета. Процесс создания дизайна на сайте называется версткой web-страниц. На сегодняшний день выделяют два основных способа верстки: - Табличная - Блочная Табличная верстка сайтов уже значительно устарела и редко когда используется. Главным недостатком табличной верстки является то, что таблица не будет показана на экране до тех пор, пока она не будет полностью загружена браузером. В таком случае если весь сайт сделан в таблице, то он появится на дисплее лишь после того, как будет полностью загружен весь HTML код страницы, а значит, дизайн сверстанный блоками быстрее загружается, ведь содержимое блоков, в отличии от содержимого ячеек таблиц отображается по мере загрузки. Блочная верстка предоставляет больше возможностей для форматирования и дизайна. Этот метод построен на использовании обтекаемых блоков, с их помощью можно произвольно располагать элементы на web- странице, то есть вы создаете такое количество блоков, которые как слои накладываются друг на друга. Часть тэгов HTML такие как , <p>, <h>- <h6>, , <ol>, <ul>, <li> являются блочными изначально. Остальные можно сделать блочными посредством стилей CSS (имеется в виду свойство display). Другими словами работа с дизайном web-страницы, это в основном работа с блочными элементами. Основа блочной системы - это тег <\div>, который является контейнером для контента. Внутри него также могут содержаться другие контейнеры. Каждый блок содержит в себе информационную часть (контент). Это может быть изображение, текст или

еще что-то. Благодаря этому HTML-код распадается на ряд чётких наглядных блоков, и при этом получается более компактным, чем при табличной вёрстке. Все блочные элементы имеют одну и ту же структуру. Блоки можно размещать на web-странице с точностью до пикселя. Положение блока можно задать двумя координатами относительно родительского элемента, любого угла окна браузера или документа. Для упрощения размещения элементов на странице, блоки можно накладывать друг на друга, перемещать, прятать и показывать без перезагрузки всей страницы. Применяя блочную верстку, вместе со стилями CSS, можно создавать различные эффекты, наподобие движущихся объектов, выпадающих элементов меню, всплывающих подсказок, при этом нет необходимости обновления или повторной загрузки страницы. Эти эффекты выполняются без какой-либо задержки со стороны браузера клиента. Использование подобных приемов при верстке значительно повышает привлекательность webсайта для конечного пользователя. В заключение данного раздела можно сделать вывод, что CSS представляют собой мощную систему для разработчиков сайтов, расширяя их возможности по дизайну и верстке веб-страниц. Современный, интерактивный и привлекательный web-сайт не может быть выполнен без использования CSS и одного из методов верстки.

В рамках данной работы было разработано небольшое web-приложение, получившее название «Менеджер грузоперевозок». Используемый стек технологий: - СУБД MongoDB - Сервер приложения на языке Golang, с применением MVCфреймворка Revel - Клиентская часть – пользовательский веб интерфейс, основанный на html, css, javascript, с применением виджетного UI фреймворка Webix. Таким образом, приложение построено на так называемой трехуровневой архитектуре. Трёхуровневая архитектура – архитектурная модель программного комплекса, предполагающая наличие в нём трёх компонентов: клиента, сервера приложений (к которому подключено клиентское приложение) и сервера баз данных (с которым работает сервер приложений). В качестве драйвера для взаимодействия с MongoDB был использован mgo, ввиду своего очень богатого функционала и достаточной простоты. Подключение к базе данных

организовано при инициализации модели из контроллера, который в свою очередь инициализируется при соответствующем HTTP запросе на него.

В современном мире, все больше и больше используются IT-технологии. Они значительно упрощают жизнь не только в общем плане, но и в работе. Так и в логистике все чаще используются различные компьютерные информационные системы и программные продукты, а если быть точнее, то без них уже трудно представить современную логистику. Уже имеется достаточно много разных систем и продуктов, которые успешно функционируют, и используются во многих компаниях. Перспективным направлением развития логистических информационно-компьютерных технологий (в частности, при транспортировке грузов) является использование глобальной некоммерческой сети интернет. За рубежом известно достаточно большое количество логистических информационных систем, использующих возможности интернет. В нашей стране сейчас ведутся активные разработки различных логистических интернет-приложений, в частности, на транспорте.

Таким образом, транспортная логистика немыслима без активного использования информационных технологий. Трудно себе представить формирование и организацию работы цепей доставки товаров без интенсивного оперативного обмена информацией между участниками транспортного процесса, без возможностей быстрого реагирования на потребности рынка транспортных услуг. Сегодня практически невозможно обеспечить требуемое потребителями качество обслуживания и эффективность транспортных операций без применения информационных систем и программных комплексов для анализа, планирования и поддержки принятия коммерческих решений. Более того, именно благодаря развитию информационных систем и технологий, обеспечившему возможность автоматизации типовых операций в транспортных процессах, логистика стала доминирующей формой организации товародвижения на технологически высококонкурентном рынке транспортных услуг. Обеспечение качества и доступности необходимой информации для специалистов, 33 возможность её удобного представления и использования для решения различных

производственных задач имеют сегодня главенствующий приоритет. Огромное количество различных специалистов (менеджеров, аналитиков и др.) организуются лишь для того, чтобы принять решение. Но, так как у каждого в этой группе свои интересы, часто несовпадающие между собой, то возникают противоречия, выход из которых на первый взгляд, можно было бы найти, выработав единый язык или документ для всей системы, но исторический опыт человечества показывает, что это не реально. А вот выработать единые правила, то есть организовать соответствующую информационную технологию для обмена данными между участниками управления транспортным потоком вполне приемлемо. Весь вопрос только в инструменте, который бы позволил работать разным специалистам одновременно, с учетом собственных интересов.

Как и любая технология, информационная требует соответствия между способами, методами, методологией ее использования и объектом управления. Начиная с элементарных учетных систем и заканчивая сложными многофакторными системами статистической обработки, проблемы возникали практически только на уровне мощности технического оснащения (например, мощности процессора, объема памяти и т.д.). Сами алгоритмы работы с данными особо не изменились. И пока количество людей, участвующих в управлении информационными технологиями, не превышало определенного уровня, своевременность ответной реакции информационных систем удовлетворяла лиц, принимающих решение. Но с появлением логистики в транспортном потоке должны быть отслежены, проанализированы, оценены и одобрены ответственными лицами в приемлемое время). Оказалось, что соответствующих алгоритмов, отвечающих этим требованиям, практически нет.

Заключение

В результате проделанной работы была изучена реляционная СУБД и, что самое главное, её реальное применение в условиях создания учебного приложения. Данная работа служит ярким примером того, что это очень удобная СУБД, что, впрочем, можно сказать практически обо всех решениях в области хранения данных, не имеющих жесткой структуры.