SECŢIA CALCULATOARE, INFORMATICĂ ŞI MICROELECTRONICĂ

Subsecţia BAZE DE DATE

ОБЕСПЕЧЕНИЕ ВЫСОКОЙ ДОСТУПНОСТИ SQL SERVER

Виктор ЖДАНОВ

Технический университет Молдовы

Аннотация: *Статья посвящена анализу различных способов обеспечения высокой доступности SQL MS Server. Описана важность обеспечения высокой доступности в работе баз данных. Дано описание применения каждого из способов, описаны плюсы и минусы каждого.*

Ключевые слова*: СУБД, Microsoft SQL Server, отказоустойчивость, обеспечение высокой доступности.*

Введение

Обеспечение высокой доступности (обеспечение отказоустойчивости) SQL Server – очень важная задача. В любой системе иногда происходят сбои и простои. Цена простоя системы может варьироваться от нескольких тысяч до нескольких миллионов долларов, в зависимости от типа и продолжительности отключения, а также типа затронутой системы. Цель большинства организаций: обеспечить высокую максимальную доступность и свести простои – связанные, как и с плановым обслуживанием, так и с программными и аппаратными сбоями – к минимуму.

Обеспечение высокой доступности подразумевает поиск идеального баланса между затратами на оборудование, временем восстановления после сбоев или простоев системы и числом потенциального потерянных транзакций во время выключения. SQL Server предоставляет множество возможностей, например такие как зеркалирование, кластеризация, репликация, благодаря чему специалисты по IT-архитектуре могут создавать системы, идеально соответствующие потребностям и ресурсам своей организации. Сервер базы данных может перестать работать в результате любого из следующих факторов: среда (например, природный катаклизм), отказ оборудования, программная ошибка, разрыв сетевого соединения, человеческое вмешательство. Возможности SQL Server как раз и позволяют, чтобы при возникновении таких сбоев была возможность не потерять важные данные, и при этом быстро восстановить работу базы данных.

1. Способы обеспечения высокой доступности.

Кластеризация – это технология, позволяющая распределить базу данных на множество независимых узлов, чтобы до критического минимума уменьшить вероятность любых локальных сбоев и ошибок. Физически кластер – это два или более компьютеров (близких по производительности) подключённых к внешнему RAID-массиву, причём у внешнего RAID-контроллера должна быть общая шина SCSI которая обеспечивает подключение к нему более чем одного сервера одновременно. У каждого сервера есть своё имя, но пользователи при обращении к службе, работающей в кластере видят не одно из этих имён, а имя третьего компьютера – виртуального сервера, работу которого и защищает кластер.

Репликация (от лат. Replico – повторяю) – это тиражирование изменений данных с главного сервера БД на одном или нескольких зависимых серверах. Главный сервер будет называться мастером, а зависимые – репликами. Одним из плюсов является то, что репликация позволяет создать копию базы данных в географически удалённом пункте, например, в другом центре обработки данных, что, например, может помочь в случае если в районе где расположен один из серверов произошла авария.

Зеркальное отображение баз данных (database mirroring) - при использовании этого средства изменения, которые вносятся в БД на одном сервере, мгновенно (или с небольшой задержкой) отображаются в копии БД на другом сервере.

У зеркального отображения БД есть несколько преимуществ: зеркальное отображение баз данных не требует применения специального оборудования, серверы которые участвуют в зеркальном отображении могут находиться далеко друг от друга, в зеркальном отображении используются две отдельные копии БД, что повышает надёжность работы, а также нет необходимости вносить какие-либо изменения в сетевую инфраструктуру или настройки клиентов. Тем не менее, зеркалирование признано Microsoft устаревшей технологией для БД. Минусом зеркалирования является то, что требует дополнительной настройки клиентских приложений, для возможности использования этой технологии.

AlwaysOn - в версии Microsoft SQL Server 2012 были усовершенствованы механизмы аварийного восстановления и появились новые решения высокого уровня доступности, объединенные наименованием AlwaysOn. К ним относятся группы доступности AlwaysOn (Availibility group) и отказоустойчивые кластерные экземпляры (инстансы) AlwaysOn (Failover Cluster Instance).

Группы доступности AlwaysOn значительно расширяют функциональность зеркалирования баз данных и обеспечивают высоких уровень доступности баз данных приложений с нулевой потерей данных в случае отказа. Эта технология обеспечивает автоматический и ручной перевод базы данных или групп баз данных на резервный ресурс, поддерживает до четырех вторичных реплик и автоматическое восстановление страниц при ошибках. AlwaysOn распределяет нагрузку среди всех участников, при этом все участники должны быть по своими характеристикам максимально похожи между собой.

AlwaysOn поддерживает два режима работы, синхронный и асинхронный. Быстрота переключения в синхронном режиме практически мгновенна, и не требует вмешательства системного администратора, а асинхронном же режиме – всё зависит от текущего состояния БД-дублей, но обычно в среднем до пяти минут.

Именно AlwaysOn признана Microsoft рекомендуемой технологией для обеспечения высокой доступности БД.

Заключение

В любой системе иногда происходят сбои и простои, поэтому обеспечение высокой доступности играет очень важную роль в поддержание отказоустойчивости баз данных, особенно если эти данные очень важны. В SQL Server доступно очень много возможностей обеспечения высокой доступности. Какую именно использовать – зависит от сложности, внешних факторов, и типа базы данных, в которой необходимо достичь высокой доступности. У каждой технологии есть и плюсы, и минусы, и каждая из них может быть эффективна применена при разных условиях.

Библиография

1. Microsoft Docs. Обзор групп доступности AlwaysOn - [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/sql/database-engine/availability-groups/windows/overview-of-always-on-availability-groups-sql-server?view=sql-server-2017.
2. Джефри Р.Гарбус. Administering SQL Server. – 450 с.
3. Habrahabr. Реализация отказа в MS SQL Server 2017 Standard. - [Электронный ресурс]. – Режим доступа: https://habr.com/post/342248/
4. Сара Морган. Проектирование и оптимизация доступа к базам данных. – 500 с.
5. Алексей Вишневский. Microsoft SQL Server. Эффективная работа – 435 с.
6. Пол Нилсен. SQL Server 2005. Библия пользователя. – 1232 с.
7. Уильямс Станек. SQL Server 2005. Справочник администратора – 522 с.

ОБЛАЧНАЯ СУБД FIREBASE

Владимир РОМАН

Технический Университет Молдовы, Департамент Программной Инженерии и Автоматики

Аннотация: *Статья посвящена облачному сервису FireBase. Представлена информация о базе данных, об ее сущности и отличиях от других баз данных, основанных на SQL.*

Ключевые слова: *Firebase, NoSQL, database, облачный сервис.*

Введение

Firebase - это базы данных в режиме реального времени, с которой вы можете передавать данные непосредственно с клиента. Когда вы сохраняете JSON данные на Firebase, изменения отправляются мгновенно для всех клиентов, веб и мобильных устройств, который запросил их. Со встроенным статическим файл хостингом, пользовательским управлением и правилами безопасности, Firebase поможет вам построить современные приложения быстрее, чем когда-либо. Firebase использует модель, управляемую событиями, чтобы уведомить нас об изменениях в наших данных, а также предоставить данные, когда они впервые приходят. Вы должны быть знакомы с событийно - ориентированным программированием. В модели запрос-ответ, ваше приложение отправляет данные за пределы на ваш сервер. Сервер может взаимодействовать с SQL (или NoSQL) базами данных и, возможно, даже внешним API, прежде чем ответить на ваш клиент с запрошенными данными. После того, как ответ получен, обмен на этом заканчивается. Клиентское приложение всегда знает, когда он будет получать новые данные, потому что оно всегда должен явно запросить его, прежде чем что-либо будет получено.

1. Базы данных NoSQL

NoSQL (англ. not only *SQL*, не только *SQL*) — термин, обозначающий ряд подходов, направленных на реализацию хранилищ баз данных, имеющих существенные отличия от моделей, используемых в традиционных реляционных СУБД с доступом к данным средствами языка SQL. Применяется к базам данных, в которых делается попытка решить проблемы масштабируемости (англ. scalability) и доступности (англ. availability) за счёт атомарности (англ. atomicity) и согласованности данных (англ. consistency). Под термином NoSQL скрывается большое количество продуктов с абсолютно разными дизайнами и, иногда, при обсуждении разговор может идти о разных системах.

Между традиционными базами данных и NoSQL существует ряд отличий. Реляционные СУБД основаны на принципах ACID: Atomicity – атомарность, Consistency – согласованность, Isolation – изолированность, Durability - надежность.

NoSQL основаны на принципах BASE, данный термин был предложен Эриком Брюером: Basic Availability — базовая доступность — каждый запрос гарантированно завершается (успешно или безуспешно); Soft State - гибкое состояние — состояние системы может изменяться со временем, даже без ввода новых данных, для достижения согласования данных; Eventual Consistency - согласованность в конечном счёте — данные могут быть некоторое время рассогласованы, но приходят к согласованию через некоторое время.

2. Что представляет Firebase

Analytics — аналитика по приложению: размер аудитории, информация о пользователях, события в приложении и прочее. **Authentication** — пользователи могут привязать свои учетные записи к приложению, а к ним мы можем привязать любые данные. Из коробки поддерживаются следующие провайдеры авторизации: *Google, Facebook, Twitter, GitHub*, анонимный вход и имейл-пароль для своей регистрации **Realtime Database** — самая настоящая база данных, работает с живыми изменениями в реальном времени. **Storage** — хранилище для файлов пользователей, можно легко сделать персональное хранилище, а можно и делиться файлами. **Hosting** — тут просто моментальное развертывание веб-приложений и мобильных приложений с помощью безопасной глобальной сети доставки контента. **Test Lab for Android** — тестируй приложения *Android* на самых разных устройствах. **App Indexing** — свяжи информацию с веб-сайта с внутренними страницами

приложения, также есть возможность индексировать данные приложения и отображать их в результатах поиска на устройстве**. Crash Reporting** — сбор информации о сбоях в приложении. **Notifications** — уведомления, замена старым **Google Cloud Messaging**. **Remote Config** — способ менять поведение приложения прямо со своего сервера, изменяя нужные параметры. **Dynamic Links** — полезный способ прокинуть контекст в приложение (например, пользователь читал про аспирин на твоем сайте, перешел в маркет, установил приложение, и ему открылась страница с аспирином). **AdMob** — рекламный сервис с множеством форматов, по праву занимает лидирующие позиции в мобильной рекламе. У этой сети рекламы всегда много, и она модерируется.

Как общаться с Firebase. В Firebase поддержаны особенности интеграции с приложениями под операционные системы Android и iOS, реализовано API для приложений на JavaScript, Java, Objective-C и Node.js, также возможно работать напрямую с базой данных в стиле REST из ряда JavaScript-фреймворков, включая AngularJS, React, Vue.js, Ember.js и Backbone.js. Предусмотрено API для шифрования данных.

Интеграция с Google Cloud. Firebase является проектом Google, что является причиной полной поддержки интеграции с сервисами IT гиганта.Интеграция с Google Cloud проведена на очень глубоком уровне. Например, Firebase Analytics может экспортировать сырые данные в BiqQuery для дальнейшего анализа. Ещё одним хорошим примером интеграции стала привязка вашего аккаунта разработчика в Google Play к учётной записи в Firebase — таким образом консоль становится центром, в который стекаются данные обо всём: начиная с непладок у пользователей, покупок внутри вашего приложения и особенностями использования у различных групп пользователей, заканчивая финансовыми данными.

Ценовая Политика. Большая часть продуктов, включая **Analytics**, **Crash Reporting**, **Remote Config**, и **Dynamic Links** — полностью бесплатны и не имеют каких-либо ограничений. Платные же сервисы — **Test Lab**, **Storage**, **БДРВ** и **хостинг** — обзавелись упрощённой ценовой сеткой.

Заключение

Данная облачная система имеет богатый функционал, который находится в открытом доступе и предоставлен компанией Google. Так как данная база данных является облачной, это облегчает разработку приложений и программ. С другой стороны данное средство является не релевантным для компаний который не могут по каким-либо причинам раскрывать свои данные в облачные сервисы. Данная база данных является простой, удобной, изящной и бесплатной базой данных реального времени которая общается по средством JSON запросов и ответов и обязательно найдет спрос как у начинающих разработчиков так и у мировых гигантов IT индустрии.

Библиография

1. Официальный сайт сервиса Firebase. - [Электронный ресурс]. – Режим доступа: https://firebase.google.com/
2. Firebase. - [Электронный ресурс]. – Режим доступа: https://en.m.wikipedia.org/wiki/Firebase
3. Официальный твиттер сервиса Firebase. - [Электронный ресурс]. – Режим доступа: https://twitter.com/firebase
4. Объектно-ориентированные базы данных: достижения и проблемы. - [Электронный ресурс]. - Режим доступа: https://www.osp.ru/os/2004/03/184042/
5. М. Аткинсон и др. «Манифест систем объектно-ориентированных баз данных», // СУБД, №4, 1995.

GRAPHQL – ЯЗЫК ЗАПРОСОВ И МАНИПУЛИРОВАНИЯ ДАННЫМИ С ОТКРЫТЫМ ИСХОДНЫМ КОДОМ

Татьяна АГАКИ, Руслан ПАНЕВСКИЙ

Технический Университет Молдовы, Департамент Программной Инженерии и Автоматики

Аннотация: *Данная работа посвящена GraphQL. Она освещает историю этого популярного языка запросов, рассказывает о его преимуществах, недостатках и принципе работы. В статье будет рассмотрен пример, показывающий принцип работы GraphQL, а так же будет сделан вывод на основе рассмотренного материала.*

Ключевые слова: *GraphQL, язык запросов, данные, запрос, таблицы.*

Введение

GraphQL — это стандарт декларирования структуры данных и способов получения данных, который выступает дополнительным слоем между клиентом и сервером. GraphQL был разработан в большом старом Facebook, но даже гораздо более простые приложения могут столкнуться с ограничениями традиционных REST API интерфейсов.

Представьте, что необходимо отобразить список записей (posts), и под каждым опубликовать список лайков (likes), включая имена пользователей и аватары. На самом деле, это не сложно, вы просто измените API posts так, чтобы оно содержало массив likes, в котором будут объекты-пользователи. Но затем, при разработке мобильного приложения, оказалось, что из-за загрузки дополнительных данных приложение работает медленнее. Так что вам теперь нужно два endpoint, один возвращающий записи с лайками, а другой без них. Добавим ещё один фактор: оказывается, записи хранятся в базе данных MySQL, а лайки в Redis! Что же теперь делать?!

Facebook придумал концептуально простое решение: вместо того, чтобы иметь множество "глупых" endpoint, лучше иметь один "умный" endpoint, который будет способен работать со сложными запросами и придавать данным такую форму, какую запрашивает клиент.

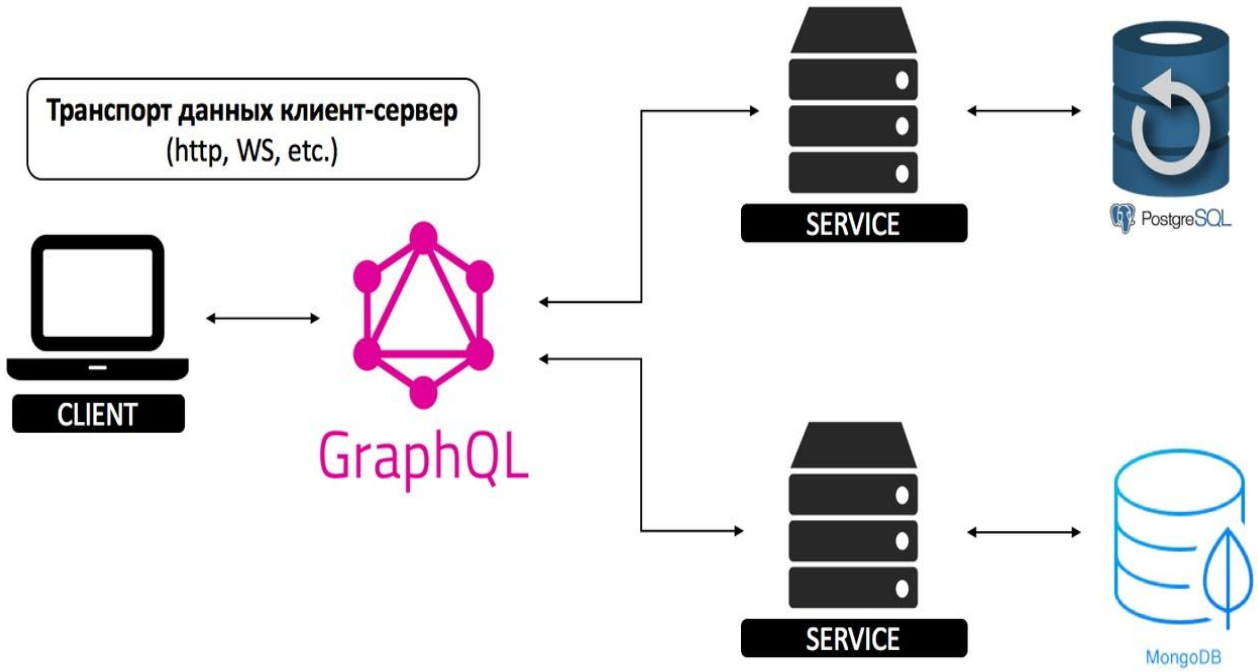


Рис. 1 Транспорт данных клиент-сервер.

Фактически, слой GraphQL находится между клиентом и одним или несколькими источниками данных; он принимает запросы клиентов и возвращает необходимые данные в соответствии с переданными инструкциями. Данный вид обработки данных показан на рисунке 1. GraphQL похож на личного помощника: вы можете передать ему адреса нескольких мест, а затем просто запрашивать то, что вам нужно и ждать их получения.

Одной из основных возможностей GraphQL является то, что структура и обьем данных определяется клиентским приложением. GraphQL — это стандарт декларирования структуры данных и способов получения данных, который выступает дополнительным слоем между клиентом и сервером. GraphQL является действительно хорошей альтернативой REST API, делает front-end программирование очень легким, так как позволяет использовать единственный endpoint.

1. Наиболее важные проблемы, которые решает GraphQL

Необходимость несколько раз обращаться за данными для рендеринга компонента. GraphQL позволяет получить все необходимые данные за один запрос к серверу.

Зависимость клиента от сервера. С помощью GraphQL клиент общается на универсальном языке запросов, который отменяет необходимость для сервера жестко задавать структуру или состав возвращаемых данных и не привязывает клиента к конкретному серверу.

Неэффективные способы разработки. На GraphQL разработчики описывают необходимые для интерфейса данные с помощью декларативного языка. Разработчики сосредоточены на том, что хотят получить, а не как это сделать. Данные, необходимые для UI, тесно связаны с тем, как эти же данные описываются в GraphQL.

Facebook использовал GraphQL с 2012 г., еще до перехода этого языка в опенсорс. Именно Facebook – та движущая сила, что отвечает за разработку спецификации GraphQL и ее справочной реализации на языке JavaScript. Итак, работая с GraphQL, вы уже стоите на плечах гигантов. Однако, и другие хорошо известные компании применяют этот язык в своих приложениях. Они инвестируют в экосистему GraphQL, поскольку современные приложения колоссально нуждаются именно в таком языке.

Заключение

В заключении перечислим некоторые возможности GraphQL, которые значительно облегчают работу с данными и их обработку:

- Несколько наборов данных в одном запросе;
- Фильтрация;
- Ограничение доступа;
- Авто генерируемая схема;
- Запросы (базовые, вложенные объекты, список из простых типов;
- Полнофункциональный парсер.

Если база данных представляла бы собой публичную библиотеку, то GraphQL стал бы библиотекарем. Он слушает Ваши запросы и определяет, что Вам нужно. Затем он идёт вдоль книжных полок и находит данные, которые Вы просили. Книжные полки в этой аналогии являются таблицами базы данных.

Библиография

1. A query language for your API. - [Электронный ресурс]. – Режим доступа: https://graphql.org
2. GraphQL API v4. - [Электронный ресурс]. – Режим доступа: https://developer.github.com/v4/
3. Что же такое этот GraphQL? - [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/326986/ - 2017
4. Алекс Банкс и Ив. Порселло. - Learning GraphQL: Declarative Data Fetching for Modern Web Apps – 2018.

AMAZON WEB SERVICES COGNITO

Анна ЗИНГАН, Максим КУКЛЕВ

Технический Университет Молдовы, Департамент Программной Инженерии и Автоматики

Аннотация: *В данной работе проводится описание возможностей и приницпа работы базы данных Amazon Cognito.*

Ключевые слова: *база данных, веб-сервисы Amazon, User Pools, Identity Pools.*

1. Назначение, компоненты и возможности Amazon Cognito

Amazon Cognito – это веб-сервис, предоставляемый компанией Amazon. Это простая в использовании, и вместе с тем, надёжная база данных, предназначенная для авторизации, регистрации и контроля доступа пользователей к веб-ресурсу или мобильному приложению.

Двумя основными компонентами Amazon Cognito являются сервисы User Pools и Identity Pools.

User Pools позволяют создавать профили и токены аутентификации для всех типов пользователей: как для тех, кто регистрируется в приложении напрямую, так и для тех, кто авторизируется через социальные или корпоративные сети. Данный сервис обладает способностью к масштабированию до десятков и даже сотен миллионов пользователей, а для его настройки нет необходимости прибегать к созданию специальной серверной инфраструктуры.

Сервис Identity Pools позволяет выдавать пользователям доступ к другим веб-сервисам Amazon. Он сохраняет информацию о том, какие платформы, девайсы и операционные системы используются для входа тем или иным пользователем. Данная информация синхронизируется с веб-сервисами Amazon каждый раз, когда пользователь находится в сети. Информация, предоставляемая сервисом Identity Pools необходима для определения и назначения ролей для пользователей.

Сервисы User Pools и Identity Pools можно использовать и вместе, и по отдельности. В качестве примера совместного использования можно рассмотреть типичный рабочий сценарий, целью является аутентификация юзера и выдача доступа к другому сервису AWS.

На первом шаге пользователь авторизируется через user pool и получает соответствующие токены после успешной аутентификации. Затем ваше приложение обменивает токены на креденшиалы AWS с использованием identity pool. Теперь пользователь может воспользоваться этими креденшиалами для получения доступа к другим сервисам AWS, к примеру, Amazon S3 и DynamoDB (рисунок 1).

Amazon Cognito позволяет своим пользователям пользоваться для авторизации такими распространёнными поставщиками социальных удостоверений, как Google, Facebook и Amazon, и поставщиками корпоративных удостоверений на основе SAML.

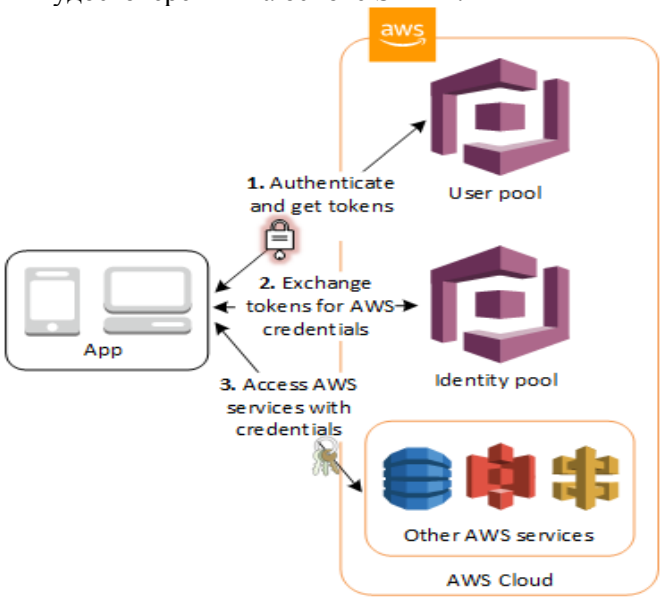


Рисунок 1 – Рабочий сценарий Amazon Cognito

Среди возможностей Amazon Cognito есть многофакторная аутентификация и шифрование данных во время их передачи и хранения. Обеспечиваемый ею уровень конфиденциальности подходит даже для организаций с самыми высокими требованиями к безопасности. Amazon Cognito соответствует требованиям HIPAA, PCI DSS, SOC, а также стандартам ISO/IEC 27001, ISO/EIC 27017, ISO/EIC 27018 и ISO 9001.

Дополняют список возможностей Amazon Cognito адаптивная аутентификация и защита от использования скомпрометированных учетных данных. Адаптивная аутентификация – это механизм, основанный на анализе риска и обеспечивающий безопасность как пользователей приложения, так и самого приложения в целом. Такие подозрительные действия, как попытка входа с не использовавшегося ранее устройства или из неизвестной локации, могут как принудить пользователя пройти дополнительную проверку, так и вовсе отказать ему в доступе.

Заключение

Сохранение и синхронизация пользовательских данных, а также встроенный настраиваемый интерфейс для регистрации и авторизации пользователей делают Amazon Cognito достойным подспорьем любому разработчику мобильных и веб-приложений, позволяя не беспокоиться об организации и поддержке back-end архитектуры приложения.

Библиография

1. What is Amazon Cognito? [Электронный ресурс]. – Режим доступа: https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html
2. Amazon Cognito – простая и безопасная регистрация и авторизация пользователей. - [Электронный ресурс]. – Режим доступа: https://aws.amazon.com/ru/cognito/
3. Amazon Cognito – вопросы и ответы. - [Электронный ресурс]. – Режим доступа: https://aws.amazon.com/ru/cognito/faqs/
4. AWS на понятном русском – шпаргалка по веб-сервисам Amazon. - [Электронный ресурс]. – Режим доступа: https://tproger.ru/translations/aws-in-plain-russian/
5. AWS представляет мобильные средства нового поколения [Электронный ресурс]. – Режим доступа: http://cloudzone.ru/community/blogs/amazon_com/358.html

SQL vs NoSQL

Максим КУКЛЕВ, Анна ЗИНГАН

Технический Университет Молдовы, Департамент Программной Инженерии и Автоматики

Аннотация: *В данной работе проводится сравнение использования реляционных и нереляционных баз данных, основанных на управлении SQL, либо NoSQL.*

Ключевые слова: *базы данных, реляционные базы данных, нереляционные базы данных, workflow, сравнение.*

Введение

Мир технологий баз данных держится на двух больших колоннах: реляционных и нереляционных баз данных, или SQL и NoSQL. Эти две структуры кардинально отличаются друг от друга в строении, замысле, проектировании, подходам к хранению информации. Однако какие из различий можно вывести на первый план, чтобы продемонстрировать разницу между этими двумя видами? Какую из моделей выбирать для своего проекта?

Говоря об общих сведениях, известно, что **реляционные БД** хранят структурированные данные, представляющие объекты реального мира. Например, информация о человеке, об ассортименте супермаркета и так далее. Всё это сгруппировано в таблицы, формат которых задаётся в процессе проектирования (рис. 1b).

Нереляционные, в свою очередь, выглядят несколько иначе. Базы с информацией о документах могут быть выстроены иерархически. Могут быть объекты с произвольным набором атрибутов, а то, что в реляционной БД разбивается на взаимосвязанные таблицы, здесь может содержаться в виде целостной сущности. (рис. 1a). Также известно, что нереляционные базы лучше поддаются масштабированию.

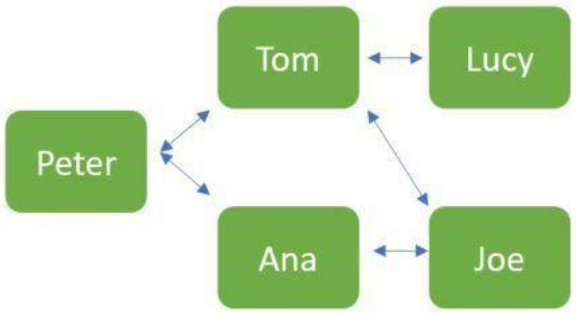
1. Преимущества SQL и NoSQL БД

Как правило, при выборе следует склониться в сторону SQL-баз, когда ваш проект подходит по двум следующим параметрам:

- База данных должна соответствовать требованиям ACID (атомарность, непротиворечивость, изолированность и долговечность), что позволяет уменьшить риск неожиданного поведения системы и обеспечить целостность базы. NoSQL, напротив, ставят в приоритет гибкость и скорость, нежели целостность хранимой информации;
- Данные должны быть структурированы, структура не подвержена частым изменениям. Если организация не находится в стадии экспоненциального роста, вероятно, не найдётся веских причин выбирать БД, вольно обращающуюся с данными и нацеленную на обработку огромных объёмов информации.

При использовании NoSQL-баз выделяют следующие преимущества:

- Хранение больших объёмов неструктурированной информации. База данных NoSQL не накладывает ограничений на введённые данные. Помимо этого, можно добавлять новые типы данных в процессе работы;
- Использование облачных вычислений и хранилищ. Данные должны легко распределяться между несколькими серверами для масштабирования;
- Быстрая разработка. Для agile-проектов не подходит реляционная модель, так как она замедляет работу.



(a)

NODE	FRIENDS
Peter	{Tom, Ana}
Tom	{Joe, Lucy, Peter}
Ana	{Joe, Peter}
Lucy	{Tom}
Joe	{Tom, Ana}

(b)

Рисунок 1 Представление данных в нереляционных (a) и реляционных (b) базах данных.

Как более продвинутый пример, для демонстрации того, когда SQL предпочтительнее NoSQL, рассмотрим особенности применения в NoSQL-базах алгоритмов уплотнения. Проблема заключается в том, что в некоторых NoSQL-базах (например, в CouchDB и HBase) постоянно приходится формировать так называемые sstables — строковые таблицы в формате ключ-значение, отсортированные по ключу. В такие таблицы, которые сохраняются на диск, данные попадают из таблиц, хранящихся в памяти, при их переполнении и в других ситуациях. При интенсивной работе с базой создание таблиц, со временем, приводит к тому, что подсистема ввода-вывода устройства хранения данных становится узким местом для операций чтения данных. Как результат, чтение в NoSQL-базе происходит медленнее, чем запись, что сводит на нет одно из главных преимуществ нереляционных баз данных. Именно для того, чтобы уменьшить этот эффект, системы NoSQL используют, в фоновом режиме, алгоритмы уплотнения данных, пытаясь объединить множество таблиц в одну. Но и сама по себе эта операция весьма ресурсоёмкая, система работает под повышенной нагрузкой.

Заключение

В итоге стоит сказать, что в современном мире нет противостояния между реляционными и нереляционными базами данных. Вместо этого стоит говорить об их совместном использовании для решения задач, на которых та или иная технология показывает себя лучше всего. Кроме того, всё сильнее наблюдается интеграция этих технологий друг в друга. Например, Microsoft, Oracle и Teradata сейчас предлагают некоторые формы интеграции с Hadoop для подключения аналитических инструментов, основанных на SQL, к миру неструктурированных больших данных.

Библиография

1. Alon Brody, *SQL или NoSQL — вот в чём вопрос*. - [Электронный ресурс]. – Режим доступа: https://habr.com/company/ruvds/blog/324936/
2. Alon Brody, *SQL vs NoSQL: The Differences Explained*. - [Электронный ресурс]. – Режим доступа: https://blog.panoply.io/sql-or-nosql-that-is-the-question
3. NoSQL. - [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/NoSQL

СИСТЕМА УПРАВЛЕНИЯ ОБЪЕКТНОЙ БАЗОЙ ДАННЫХ REALM

Вадим МОКАН

Технический Университет Молдовы, Департамент Программной Инженерии и Автоматики

Аннотация: *Данная работа посвящена базе данных Realm. Описаны особенности этой популярной базы данных, её принципы функционирования. Статья даёт ответы на множество вопросов, которыми задаются разработчики мобильных приложений при поиске и выборе оптимальной базы данных, в зависимости от поставленных задач.*

Ключевые слова: *база данных, реляционная, мобильная, React Native, объект, кроссплатформенность, ключ, синхронизация.*

Введение

Realm - это система управления объектной базой данных с открытым исходным кодом, первоначально созданная для мобильных платформ Android, iOS, а также доступна для таких платформ, как Xamarin или React Native, и др. Нативная NoSQL база даных обладает своим собственным ядром. Полностью заменяет старомодные основанные на SQLite базы данных которые являются альтернативой для разработчиков на сегодняшний день.

Стандартные ORM (Object-Relational Mapping) связанные базы данных не обходятся без копирования:

- Данные на диске;
- Данные считываются с диска;
- Копирование исходных данных в десериализованное промежуточное представление в памяти. (выделение памяти);
- Копирование промежуточного представления в языковой уровень в памяти;
- Возврат конечного объекта.

Realm обходится без копирования:

- Вычисляет смещение данных для чтения;
- Считывает из соответственного файла;
- Возврат конечного объекта.

1. Особенности Realm

Управление параллельным доступом с помощью многоверсионности - один из механизмов обеспечения параллельного доступа к БД, заключающийся в предоставлении каждому пользователю так называемого «снимка» БД, обладающего тем свойством, что вносимые пользователем изменения в БД невидимы другим пользователям до момента фиксации транзакции. Этот способ управления позволяет добиться того, что пишущие транзакции не блокируют читающих, и читающие транзакции не блокируют пишущих.

В упрощенном виде этот механизм можно представить следующим образом: все операции с данными можно условно разделить на чтение (select), вставку (insert), удаление (delete), обновление (update).

Realm поддерживает ACID транзакции:

Atomicity — транзакции атомарны, то есть либо все изменения транзакции фиксируются (commit), либо все откатываются (rollback);

Consistency — транзакции не нарушают согласованность данных, то есть они переводят базу данных из одного корректного состояния в другое. Тут можно упомянуть допустимые значения полей, внешние ключи и более сложные ограничения целостности;

Isolation — работающие одновременно транзакции не влияют друг на друга, то есть многопоточная обработка транзакций производится таким образом, чтобы результат их параллельного исполнения соответствовал результату их последовательного исполнения;

Durability — если транзакция была успешно завершена, никакое внешнее событие не должно привести к потере совершенных ей изменений.

Другими особенностями Realm являются:

- Realm - это не единственная база данных в приложении. Обычно в приложении используется одна SQL база данных, тогда как можно создать множественное количество Realm-ов для работы с данными более эффективно, для их распределения и контроля.
- Realm - не таблица! Таблицы обычно хранят только один вид информации: записи пользователя, сообщения электронной почты и т.д. Realm может содержать несколько видов объектов.
- Realm не является хранилищем документов. Поскольку свойства объектов аналогичны парам ключ: значение, легко представить Realm как хранилище, но объекты имеют определенные схемы, которые поддерживают присвоение значений по умолчанию или их маркировку по мере необходимости.
- Как только объект был добавлен в Realm, все действия, связанные с ним в коде проводятся и в самом Realm автоматически, не нужно вызывать вспомогательный метод для обновления или лишнего добавления в Realm для проведения изменений. Они будут сделаны автоматически.
- Это касается и синхронных объектов Realm, не нужно прилагать никаких усилий для проведения изменений на серверной стороне. Изменяем объект и при дальнейшем подключении устройства к сети, изменения распространятся на сервер и на другие клиенты Realm, которые синхронизируются с этими объектами.

Заключение

Realm сложнее, чем кажется на первый взгляд. Однако все недостатки с лихвой покрываются его мощностью и удобством. Live объекты, нотификации и реактивность, удобный интерфейс и множество других вещей упрощают создание приложений. Полностью же Realm раскрывает себя при построении оффлайн приложений, когда все данные мы получаем из кэша и нам необходимы сложные выборки, а также постоянное обновление данных.

Realm лучше использовать по следующим причинам:

- Быстрые запросы - Realm предоставляет результаты относительно быстрее, чем SQLite, и имеет большую производительность;
- Лёгкость в использовании - Realm гораздо проще в использовании, благодаря тому, что он является объектной базой данных. Просто создав класс, который расширяет класс RealmObject, вы настраиваете абсолютно всё;
- Кроссплатформенность - Хорошая альтернатива SQLite (Android) и CoreData (IOS). Его можно использовать на всех платформах, таких как iOS, Xamarin и React Native;
- Простое создание и хранение данных.

Библиография

1. System Properties Comparison Firebase Realtime Database vs. Realm vs. SQLite. - [Электронный ресурс]. – Режим доступа: https://db-engines.com/en/system/Firebase+Realtime+Database%3BRealm%3BSQLite
2. Realm Databese. - [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Realm_(database)
3. Реалистичный Realm. - [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/328418/
4. Realm Sync Documentation. - [Электронный ресурс]. – Режим доступа: https://docs.realm.io/sync/

ОБЪЕКТНО - РЕЛЯЦИОННАЯ СУБД POSTGRE SQL

Руслан ПАНЕВСКИЙ, Татьяна АГАКИ

Технический Университет Молдовы, Департамент Программной Инженерии и Автоматики

Аннотация: *данная работа посвящена СУБД PostgreSQL. Она освящает историю этой популярной базы данных, повествует об истоках её возникновения и о пройденном ею пути. Статья даёт ответы на множество вопросов, которыми задаются все без исключения любители этой базы данных.*

Ключевые слова*: Postgre, SQL, архитектура, СУБД, ОРСУБД, POSTQUEL.*

Введение

Postgre SQL — это объектно-реляционная система управления базами данных (ОРСУБД, ORDBMS), основанная на POSTGRES, Version 4.2 — программе, разработанной на факультете компьютерных наук Калифорнийского университета в Беркли. В POSTGRES появилось множество новшеств, которые были реализованы в некоторых коммерческих СУБД гораздо позднее.

PostgreSQL — СУБД с открытым исходным кодом, основой которого был код, написанный в Беркли. Она поддерживает большую часть стандарта SQL и предлагает множество современных функций: сложные запросы, внешние ключи, триггеры , изменяемые представления, транзакционная целостность, многоверсионность.

Объектно-реляционная система управления базами данных, именуемая сегодня PostgreSQL, произошла от пакета POSTGRES, написанного в Беркли, Калифорнийском университете. После двух десятилетий разработки PostgreSQL стал самой развитой СУБД с открытым исходным кодом.

Стоунбрейкер и его студенты разрабатывали новую СУБД в течение восьми лет с 1986 по 1994 год. За этот период в синтаксис были введены процедуры, правила, пользовательские типы и другие компоненты. В 1995 году разработка снова разделилась: Стоунбрейкер использовал полученный опыт в создании коммерческой СУБД Illustra, продвигаемой его собственной одноимённой компанией (приобретённой впоследствии компанией Informix), а его студенты разработали новую версию Postgres — Postgres95, в которой язык запросов POSTQUEL — наследие Ingres — был заменен на SQL.

Разработка Postgres95 была выведена за пределы университета и передана команде энтузиастов. Новая СУБД получила имя, под которым она известна и развивается в текущий момент — PostgreSQL.

1. Поддержка стандартов, особенности

В PostgreSQL есть следующие параметры:

Максимальные размер базы данных - нет ограничений, Максимальный размер таблицы - 32 ТБайт, Максимальный размер записи 1,6 ТБайт, Максимальный размер поля - 1 Гбайт, Максимум записей в таблице - Нет ограничений, Максимум полей в записи - 250-1600, в зависимости от типа полей, Максимум индексов в таблице - нет ограничений

Сильными сторонами PostgreSQL считаются:

- высокопроизводительные и надёжные механизмы транзакций и репликации;
- расширяемая система встроенных языков программирования: в стандартной поставке поддерживаются PL/pgSQL, PL/Perl, PL/Python и PL/Tcl; дополнительно можно использвать PL/Java, PL/PHP, PL/Py, PL/R, PL/Ruby, PL/Scheme, PL/sh и PL/V8, а также имеется поддержка загрузки C-совместимых модулей;
- наследование;
- легкая расширяемость.

2. Основные возможности

PostgreSQL предоставляет множество различных возможностей, достаточно надежна и имеет хорошие характеристики по производительности. Она работает практически на всех UNIX-платформах, включая UNIX-подобные системы, такие как FreeBSD и Linux. Ее можно применять на Windows NT Server и Windows 2000 Server, а для разработки годятся даже такие системы Microsoft для рабочих станций, как ME. Кроме того, PostgreSQL имеет открытый исходный код.

PostgreSQL выгодно отличается от многих других СУБД. Она обладает практически всеми возможностями, которые есть в других базах данных (коммерческих или Open Source), а также некоторыми дополнительными.

Приведем перечень функциональных возможностей PostgreSQL: Транзакции, Вложенные запросы, Представления, Ссылочная целостность - внешние ключи, Сложные блокировки, Типы определяемые пользователем, Наследственность, Правила, Проверка совместимости версий

3. Архитектура PostgreSQL

Одной из сильных сторон PostgreSQL является ее архитектура. Как и многие коммерческие СУБД, PostgreSQL может применяться в среде клиент-сервер, что дает массу преимуществ как пользователям, так и разработчикам.

Основа PostgreSQL составляет серверный процесс базы данных. Он выполняется на одном сервере. (В этой СУБД еще не реализована технология высокой готовности, как в некоторых других коммерческих системах уровня предприятия, которые могут распределять нагрузку между несколькими серверами, добиваясь таким образом дополнительной масштабируемости и устойчивости к внешним воздействиям.

Доступ из приложений к данным базы осуществляется посредством процесса базы данных. Клиентские программы не могут получить доступ к данным самостоятельно, даже если они работают на том же компьютере, на котором выполняется серверный процесс.

Такое разделение клиентов и сервера позволяет построить распределенную систему. Можно отделить клиентов от сервера посредством сети и разрабатывать клиентские приложения в среде, удобной для пользователя. Например, можно реализовать базу данных под UNIX и создать клиентские приложения, которые будут работать в системе Microsoft Windows.

Несколько клиентов подсоединяются к серверу по сети. PostgreSQL ориентирован на протокол TCP/IP - это может быть локальная сеть или Интернет. Каждый клиент соединяется с основным серверным процессом базы данных (на схеме - Postmaster), который создает новый серверный процесс специально для обслуживания запросов на доступ к данным конкретного клиента.

Архитектура клиент-сервер делает возможным разделение труда. Машина-сервер хорошо подходит для хранения и управления доступом к большим объемам данных, она может использоваться как надежный репозиторий. Для клиентов могут быть разработаны сложные графические приложения. В качестве альтернативы можно создать внешний интерфейс на основе Интернета, который предоставлял бы доступ к данным и возвращал результат в виде веб-страниц в стандартный веб-браузер, при этом не требовалось бы никакого дополнительного клиентского программного обеспечения.

Заключение

У PostgreSQL множество возможностей. Созданный с использованием объектно-реляционной модели, она поддерживает сложные структуры и широкий спектр встроенных и определяемых пользователем типов данных. Она обеспечивает расширенную ёмкость данных и заслужила доверие бережным отношением к целостности данных. Возможно, вам не понадобятся все те продвинутые функции хранения данных, которые мы исследовали в этой статье, но, поскольку потребности могут быстро возрасти, есть несомненное преимущество в том, чтобы иметь всё это под рукой.

PostgreSQL чрезвычайно богат функциональными возможностями, с множеством встроенных “фич” и бесчисленным количеством способов их индивидуализации и расширения для удовлетворения ваших потребностей. Добавьте к этому общепризнанную надежность и зрелость, и станет ясно, почему это решение для баз данных стоит усилий любого крупного предприятия. При этом, Postgre SQL остаётся доступным и эффективным также и для небольших проектов.

Библиография

1. PostgreSQL.- [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/PostgreSQL
2. Что такое PostgreSQL? Плюсы и минусы бесплатной базы данных.- [Электронный ресурс]. – Режим доступа: https://oracle-patches.com/common/3214-%D1%87%D1%82%D0%BE-%D1%82%D0%B0%D0%BA%D0%BE%D0%B5-postgresql
3. Когда использовать неструктурированные типы данных в PostgreSQL? Сравнение Hstore vs. JSON vs. JSONB.- [Электронный ресурс]. – Режим доступа: ttps://habr.com/post/306602/comments/
4. Чем PostgreSQL лучше других SQL баз данных с открытым исходным кодом. Часть 2.- [Электронный ресурс]. – Режим доступа: https://habr.com/post/302160/

ЗАЩИТА ПЕРСОНАЛЬНЫХ ДАННЫХ В СУБД ORACLE

Анастасия СИДОРЕНКО

Технический Университет Молдовы, Департамент Программной Инженерии и Автоматики

Аннотация: *В данной статье раскрывается технология защиты персональных данных в одной из самых распространенных СУБД на сегодняшний день – Oracle Database. Приведен пример механизма шифрования, используемого в рассматриваемой СУБД – прозрачное шифрование данных.*

Ключевые слова: *база данных, прозрачное шифрование, конфиденциальность, Oracle, защита.*

Введение в понятие шифрования данных

В современном мире, где информация становится одним из основных ресурсов в экономике, становится все труднее обеспечивать защиту данных от злоумышленников. Заполняя анкету на Интернет-форуме или делая заказ в онлайн-магазине, личные данные пользователя попадают напрямую в базу данных приложения. Существуют различные меры предосторожности для сохранения информации в базе данных и обеспечения ее конфиденциальности и целостности, например, проектирование безопасной системы, шифрование конфиденциальных активов, а также создание брандмауэра вокруг серверов баз данных. Данные методы могут быть очень эффективны для достижения вышеназванной цели, но в одном случае они не смогут уберечь личные данные от кражи – в случае похищения физических носителей хранения базы данных. Тогда в ход вступает технология шифрования конфиденциальных данных и соответствующая защита ключей, используемых в криптографическом процессе, с помощью так называемых сертификатов.

1. Шифрование данных в СУБД Oracle Database

На сегодняшний день одной из самых распространенных и надежных систем управления базами данных является объектно-реляционная СУБД Oracle Database, созданная в 1979 году группой американских программистов: Ларри Эллисон, Боб Майнер и Эд Оутс. Своей надежностью она обязана поддержкой технологии прозрачного шифрования данных, которая в англоязычных источниках называется TDE, то есть Transparent Data Encryption. TDE входит в состав опции Oracle Advanced Security с версии 10g, выпущенной на пользовательский рынок в 2005 году. Эта технология позволяет «прозрачно» для приложений шифровать данные на уровне колонок таблиц или табличных пространств.

Прозрачное шифрование позволяет защитить различные конфиденциальные данные: от номеров кредитных карт и банковских реквизитов до номеров социального и медицинского страхования. Значительным преимуществом прозрачного шифрования является тот факт, что зашифрованные данные видны в своем первоначальном виде для разработчика базы данных или для пользователя, имеющего доступ к базе данных, что значительно упрощает манипуляцию данными для ИТ-специалиста.

Теперь стоит рассмотреть технологию прозрачного шифрования данных в рамках системы управления базы данных Oracle Database. Здесь используется аутентификация, авторизация и другие механизмы обеспечения безопасности данных непосредственно в базе данных, но не в файлах данных операционной системы, что и делает их наиболее уязвимыми для злоумышленников. Именно для защиты этих данных предусмотрена поддержка TDE в СУБД Oracle.

Как было отмечено ранее, прозрачное шифрование может применяться по отношению к столбцам таблиц: как к одному, так и ко всем, а также к табличным пространствам. Однако для понимания принципа прозрачного шифрование данных достаточно рассмотреть механизм TDE лишь на уровне колонок. Например, если таблица имеет четыре столбца, два из которых необходимо подвергнуть процессу шифрования то сервер Oracle Database генерирует один зашифрованный ключ. Таким образом, на диске значения обычных столбцов будут храниться в виде текста, а зашифрованные столбцы будут представлены в шифрованном формате. В случае, когда пользователь захочет выбрать зашифрованные столбцы, сервер Oracle прозрачно для приложения извлечет из словаря данных ключ шифрования таблицы, а главный ключ из так называемого «бумажника», который представляет собой файл, защищенный паролем, для хранения ключей шифрования, паролей, а также личных ключей, сертификатов, и находящийся в файловой системе сервера базы данных. На следующем шаге происходит расшифровка данных сервером и возвращение их

пользователю в виде обычного текста. Ключи, с помощью которых и происходит процесс шифрования персональных данных в технологии TDE хранятся в модуле защиты внешне по отношению к базе данных. Таким образом, даже в случае, когда злоумышленник получает доступ к базе данных, он не сможет провести криптоанализ и расшифровать данные без знания пароля бумажника. Этот факт обеспечивает еще один уровень защиты персональных данных в СУБД Oracle.

После рассмотрения технологии прозрачного шифрования данных с теоретической точки зрения стоит взглянуть, как данный механизм ведет себя на практике, а именно – как ИТ-специалист может настроить и установить TDE для своей базы данных в СУБД Oracle.

Первым шагом в этом процессе станет открытие упомянутого выше бумажника и определение его местоположения. Для удобства использования Oracle прописывает путь хранения бумажника в файле sqlnet.ora, который по умолчанию будет иметь значение $ORACLE_BASE/admin/$ORACLE_SID/wallet. Однако можно установить пользовательское значение, прописав в данном файле следующие строки:

ENCRYPTION_WALLET_LOCATION = (SOURCE (METHOD=file) (METHOD_DATA=(DIRECTORY=<пользовательское_значение_директории>)))

После успешного определения расположения бумажника можно приступить к его непосредственному созданию, путем выполнения следующего оператора, который не только создает бумажник в указанном каталоге, но и открывает его для хранения и извлечения главного ключа средствами TDE: alter system set encryption key authenticated by “remnant”; Чтобы шифровать столбцы таблиц, используя средства прозрачного шифрования, все, что достаточно сделать, это добавить к определениям самих столбцов дескриптор ENCRYPT. Однако до этого необходимо решить и установить какой именно тип шифрования будет использоваться и какой будет длина ключа. Стоит отметить, что по умолчанию СУБД Oracle использует алгоритм шифрования AES (Advanced Encryption Standard) с 192-битовым ключом.

Для рассмотрения примера использования шифрования в случае уже существующей таблицы, можно предположить, что в базе данных имеется отношение, определенное следующим образом:

ACCOUNT_NUMBER	NUMBER
ACCOUNT_NAME	VARCHAR(20)
SSN	VARCHAR(10)

При создании этой таблицы не было указано, стоит ли применять шифрование к одному или нескольким ее столбцам, поэтому данные в настоящий момент времени представлены в виде обычного текста. Можно предположить, что требуется применить алгоритм AES с ключом в 128 бит для шифрования столбца SSN. Тогда необходимо выполнить оператор alter table accounts modify (ssn encrypt using ‘AES128’); который не только создает ключ шифрования для указанной таблицы, но и преобразовывает все значения столбца SSN в формат, соответствующий алгоритму шифрования.

Заключение

Встроенные в СУБД Oracle Database механизмы шифрования конфиденциальных данных отвечают нуждам и требованиям современных разработчиков и клиентов к архитектуре и безопасности базы данных. Технология прозрачного шифрования данных полностью интегрирована в систему Oracle, что делает ее использование наиболее удобным для ИТ-специалиста. Но не только удобство и относительная простота использования делают данный механизм одним из самых распространенным в среде разработки и проектирования базы данных, ведь, как было сказано ранее, он обеспечивает очень высокий уровень безопасности и надежности хранения персональных данных пользователей, защищая их даже от угрозы физической кражи.

Библиография

1. А. Нанда, Прозрачное шифрование данных – М.: Oracle Magazine – 2005
2. А. Л. Додохов, А. Г. Сабанов, Исследование применения СУБД Oracle для защиты персональных данных – 2011
3. ArcGIS, Прозрачное шифрование данных (TDE) для рабочей области Reviewer в Oracle. – [Электронный ресурс.]. – Режим доступа: https://desktop.arcgis.com/ru/arcmap/latest/extensions/data-reviewer-guide/admin-dr-oracle/transparent-data-encryption-tde-for-the-reviewer-workspace-in-oracle.htm
4. ISO27000, Обеспечение защиты персональных данных в СУБД Oracle. - [Электронный ресурс]. – Режим доступа: http://www.iso27000.ru/chitalnyi-zai/zaschita-personalnyh-dannyh/obespechenie-zaschity-personalnyh-dannyh-v-subd-oracle

РАЗВЕРТЫВАНИЕ SQL СЕРВЕРОВ НА ОПЕРАЦИОННЫХ СИСТЕМАХ LINUX И MacOS

Илья ЧУКИТУ

Технический Университет Молдовы, Департамент Программной Инженерии и Автоматики

Аннотация: *Статья посвящена развертыванию SQL серверов на ОС Linux и MacOS. Описана краткая характеристика SQL серверов и способы их установки, запуска и взаимодействия. Перечислены программы для более удобного взаимодействия пользователя с базой данных, дана их краткая характеристика.*

Ключевые слова*: терминал, установка, виртуализация, SQL Server, Server User Interface.*

1. Краткое описание и разбор установки SQL

Базы данных являются центральным ядром многих компаний, они хранят в себе платежи, личные данные и корпоративную информацию, без которой существование организации станет невозможным. Компании стремятся уменьшить риски потери или утечки информации. Поэтому очень важно, чтобы уже при развертывании SQL на сервер, пользователь четко понимал все необходимые операции и правильно их выполнил.

Установка многих SQL серверов хоть и является схожей, но таит в себе множество подводных камней, которые могут усложнить развертывание. Некоторые базы данных вообще не поддерживают изначально работу на некоторых операционных системах. Например, Microsoft SQL не поддерживает стандартную работу на MacOS, поэтому для работы на ней, либо придется ставить виртуальную машину и работать уже на ней, либо использовать альтернативное предложение от самого разработчика. Microsoft предлагает воспользоваться программой «Docker».

Docker — программное обеспечение для автоматизации развёртывания и управления приложениями в среде виртуализации на уровне операционной системы. Позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, который может быть перенесён на любую Linux-систему с поддержкой cgroups в ядре, а также предоставляет среду по управлению контейнерами. Изначально использовал возможности LXC, с 2015 года применял собственную библиотеку, абстрагирующую виртуализационные возможности ядра Linux — libcontainer. С появлением Open Container Initiative начался переход от монолитной к модульной архитектуре.

Эта программа, позволяет операционной системе запускать процессы в изолированном окружении на базе специально созданных образов. Несмотря на то, что технологии, лежащие в основа Докера появились до него, именно Докер произвел революцию в том, как сегодня создается инфраструктура проектов, собираются и запускаются сервисы. Docker позволяет выбрать нужное окружение (ОС) и запускать в этой среде необходимые программы. Докер позволяет запускать приложения одной командой, после все необходимые операции будут выполнятся на фоне автоматически. Это очень сильно экономит время и ресурсы.

На Linux же вся процедура проводится в несколько консольных команд.

- Импорт из публичного репозитория используя GPG ключ:

```
wget -qO- https://packages.microsoft.com/keys/microsoft.asc | sudo apt-key add
```

- Регистрация the Microsoft SQL Server Ubuntu репозитория:

```
sudo add-apt-repository "$(wget -qO- https://packages.microsoft.com/config/ubuntu/16.04/mssql-server-2017.list)»
```

- Запуск следующий команд для установки SQL:

```
sudo apt-get update
sudo apt-get install -y mssql-server
```

- После завершения установки пакета, запустите mssql-conf установкуи следуйте дальнейшим инструкциям для настройки:

```
sudo /opt/mssql/bin/mssql-conf setup
```

- Как только установка и настройка будет закончена проверьте все:

```
systemctl status mssql-server
```

2. Выбор SQL Server User Interface

Работа с серверами возможна и через терминал, но для обычных пользователей и даже для многих программистов это является не самым удобным вариантом для работы с БД.

Для эффективной работы необходима программа для повышения удобства работы с серверами. Интерфейс этих приложений представляет собой несколько окон для удобной работы с БД. Слева находится файловая система БД, в центре окно для презентации кода, диаграмм, скриптов и т.д. Справа находится панель инструментов и параметров для работы с БД (не всегда). Внизу же находится окно сообщений, с выводом таблиц, информации, ошибок, и других результатов запросов.

Был подобран некоторый список программ, подходящих для наших нужд:

SQL Operations Studio (SQLOPS) – создан компанией microsoft и доступен для Windows, macOS и Linux. Разрабатывался для работы с Microsoft Azure SQL Database, является бесплатным. Доступен на macOS, Linux и Windows (рис.1).

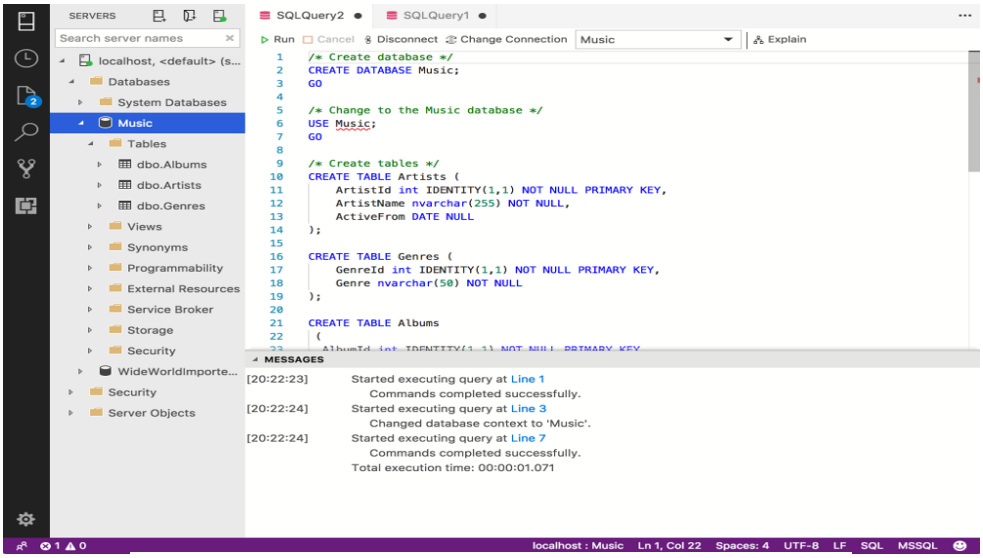


Рисунок 1 – Интерфейс программы SQL Operation Studio.

Microsoft Visual Code - редактор, разработанный Microsoft. Позиционируется как «лёгкий», кроссплатформенный и бесплатный. Доступен для macOS, Windows и Linux.

DataGrip - Программное обеспечение JetBrains DataGrip представляет собой IDE-инструмент для работы с базами данных. Доступна для Windows, macOS и Linux.

TablePlus - Современная и удобная база данных разработанная для работы с MySQL, PostgreSQL, SQLite и т.д.Работает на Windows, Linux и macOS.

Заключение

Тематика статьи не является, сложной, но вероятность совершить ошибку при выполнении простых действий не так уж и мала, а последствия неудачного развертывания могут отнять много времени на их устранение. Из нее также можно вычесть урок, что не все базы данных могут нативно работать на операционных системах MacOS и Linux, и потребуют для своей работы виртуальную машину или специальное ПО типа Docker. Поэтому, стоит учитывать выбор базы данных не только на личном опыте и предпочтениях к БД, но и ее поддержки на определенных операционных системах. Microsoft SQL Server предпочтительнее использовать на операционной системе Windows.

Библиография

1. Microsoft SQL documentation. – [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/en-us/sql/?view=sql-server-2017
2. Как установить MySQL сервер на Mac OS X. - [Электронный ресурс]. – Режим доступа: https://vladster.net/ru/как-установить-mysql-сервер-на-mac-os-x/
3. Лучшие БД для Linux. – [Электронный ресурс]. – Режим доступа: https://losst.ru/luchshie-bazy-dannyh-dlya-linux
4. Руководство по Ubuntu Server. – [Электронный ресурс]. – Режим доступа: https://help.ubuntu.ru/wiki/руководство_по_ubuntu_server/базы_данных

ОБЛАЧНАЯ НЕРЕЛЯЦИОННАЯ БАЗА ДАННЫХ AMAZON DYNAMO DB: ПРОСТО И УДОБНО

Максим ЧУКИТУ, Валерия КОРОСТИНСКАЯ

Технический Университет Молдовы, Департамент Программной Инженерии и Автоматики

Аннотация: *Данная работа посвящена нереляционной serverless базе данных Amazon DynamoDB, предоставляемой компанией Amazon Web Services, её ключевым особенностям, истории создания и становления одной из самых популярных serverless баз данных.*

Ключевые слова: *база данных, NoSQL, DynamoDB, Amazon, Web Services, облако, cloud, serverless, масштабируемость.*

Введение

В мире информационных технологий всегда существовала потребность в записи, хранении и чтении информации. Эта потребность привела к появлению и развитию множества различных типов физических накопителей, файловых систем и, конечно же, баз данных.

Существует множество различных решений по хранению информации на любой вкус и цвет: реляционные и нереляционные, SQL и NoSQL, но все они обладают одним существенным недостатком: необходимостью сервера для размещения, который также нуждается в обслуживании и улучшении при достижении пика производительности. Хотя, вернее сказать, обладали, ведь и эта проблема была решена с появлением облачных технологий. Но обо всём по порядку. И, пожалуй, стоит начать с того, что Amazon Web Services.

1. Amazon Web Services

Amazon Web Services (AWS) является дочерней компанией Amazon.com, предоставляющей платформу облачных вычислений частным лицам, компаниям и правительствам на платной подписке. Эта платформа предоставляет пользователям в своё распоряжение виртуальный кластер компьютеров, доступный постоянно через Интернет.

Amazon Web Services в значительной степени (наряду с Google Cloud Platform) повлияли на формирование концепции облачных вычислений в целом, и определили основные направления развития публичной модели развёртывания. Длительное время Amazon Web Services было крупнейшим в мире по выручке публичным облаком, во второй половине 2010-х годов уступив по этому показателю Azure от Microsoft, при этом сохраняя доминирование в сегментах инфраструктурных и платформенных услуг. По состоянию на 2017 год годовая выручка от услуг AWS превысила $20 млрд, что составило около 11,5 % доходов Amazon.

Ввиду огромного количества все сервисы разбиты на семейства сервисов. Одним из сервисов, предоставляемых платформой AWS, является Amazon DynamoDB.

2. Amazon DynamoDB

Нереляционная NoSQL база данных Amazon DynamoDB была представлена в середине января 2012 года. Ключевой особенность DynamoDB, делающей её столь привлекательной является serverless подход, который позволяет забыть о проблемах масштабирования базы данных, необходимости обслуживания и настройки сервера и о других неудобствах, присущих серверным базам данных.

Как заявляет Jeff Barr в статье посвящённой релизу новой базы данных, DynamoDB позволяет пользователю хранить такой объём данных, который ему необходим, и получать к ним доступ так часто, как ему необходимо, с предсказуемой производительностью, обеспечиваемой использованием твердотельного диска, более известного как SSD.

DynamoDB работает на основе “предоставленной пропускной способности” (provisioned throughput). Пользователю достаточно указать необходимую пропускную способность для записи и чтения (отдельно друг от друга) при создании таблицы DynamoDB. За кулисами же всё будет настроено таким образом, чтобы удовлетворить потребности пользователя, сохраняя при этом задержку в миллисекундах. Позже, при росте потребностей пользователя, он может просто увеличить выделенную пропускную способность (или наоборот уменьшить). Данное изменением можно

произвести онлайн, без простоев и без влияния на общую пропускную способность. Другими словами, пользователь имеет возможность изменять масштаб, даже когда база данных обрабатывает запросы.

Однако многие рабочие нагрузки имеют циклический характер или их сложно предсказать заранее. Например, приложение для социальных сетей, в котором большинство пользователей активны в дневное время. База данных должна быть в состоянии обрабатывать дневную активность, но нет необходимости в одинаковых уровнях пропускной способности ночью. Такие виды рабочих нагрузок часто требуют ручного вмешательства для увеличения или уменьшения ресурсов базы данных в ответ на различные уровни использования.

Сервис Amazon DynamoDB обладает опцией автоматического масштабирования для динамической настройки выделенной пропускной способности от вашего имени в ответ на фактические схемы трафика. Это позволяет увеличить свою подготовленную емкость для чтения и записи, чтобы обрабатывать внезапные увеличения трафика без регулирования. Когда рабочая нагрузка уменьшается, автоматическое масштабирование приложения уменьшает пропускную способность, поэтому пользователь не платит за неиспользованную выделенную емкость.

Другим немаловажным достоинством является то, что Amazon DynamoDB смехотворно проста в использовании. Только что созданные таблицы обычно готовы к использованию уже через одну или две минуты. Как только таблица будет готова, пользователь просто начинает хранить в ней необходимые ему данные в необходимом ему количестве, оплачивая при этом только используемое хранилище (нет необходимости предварительно выделять хранилище). За кадром же позаботятся о том, чтобы предоставить достаточно места для хранения.

Каждая таблица должна иметь первичный индекс. Существует два типа первичных ключей: простые хэш-ключи и составные хэш-ключи с диапазонными ключами.

Простые хэш-ключи создают абстракцию распределенной хэш-таблицы (Distributed Hash Table) и используются для индексации уникального ключа. Ключ хэшируется по нескольким разделам обработки и хранения для оптимального распределения рабочей нагрузки.

Составные хэш-ключи с диапазонными ключами предоставляют возможность создания первичного ключа, который состоит из двух атрибутов - атрибута hash и атрибута range. Когда вы выполняете запрос к ключу этого типа, атрибут hash должен соответствовать уникально, но для атрибута range можно указать диапазон (от низкого до высокого). Такой подход может быть использован для выполнения запросов, таких как «все заказы от Джеффа за последние 24 часа».

Каждый элемент в таблице DynamoDB состоит из набора пар ключ/значение. Каждое значение может быть строкой, числом, набором строк или набором чисел.

Выводы

Вне всяких сомнений Amazon DynamoDB является крайне привлекательным решением для хранения данных. Данный сервис невероятно прост в использовании, решает проблемы масштабирования и при этом является экономически выгодным для пользователя за счёт автоматического увеличения и уменьшения выделенной пропускной способности в зависимости от фактической нагрузки. Многие известные компании такие как Nike Digital, Samsung, Snap Inc., Netflix, Tinder и Amazon отдали предпочтение именно этому сервису. Возможно, и вам стоит попробовать?

Библиография

1. Jeff Barr, «Amazon DynamoDB – Internet-Scale Data Storage the NoSQL Way».- [Электронный ресурс]. – Режим доступа: https://aws.amazon.com/ru/blogs/aws/amazon-dynamodb-internet-scale-data-storage-the-nosql-way/
2. Amazon DynamoDB Developer Guide (API Version 2012-08-10), «Managing Throughput Capacity Automatically with DynamoDB Auto Scaling».- [Электронный ресурс]. – Режим доступа: https://docs.aws.amazon.com/en_us/amazondynamodb/latest/developerguide/AutoScaling.html
3. Amazon DynamoDB. Быстрый и гибкий сервис баз данных NoSQL для любого масштаба.- [Электронный ресурс]. – Режим доступа: https://aws.amazon.com/ru/dynamodb/
4. Jeff Barr, « Announcing BatchWriteItem for DynamoDB».- [Электронный ресурс]. – Режим доступа: https://aws.amazon.com/ru/blogs/aws/announcing-batchwriteitem-for-dynamodb/

ОБЛАЧНАЯ РЕЛЯЦИОННАЯ БАЗА ДАННЫХ AMAZON AURORA: ГРАЦИОЗНО И ЭЛЕГАНТНО

Максим ЧУКИТУ, Валерия КОРОСТИНСКАЯ

Технический Университет Молдовы, Департамент Программной Инженерии и Автоматики

Аннотация: *Данная работа посвящена реляционной serverless базе данных Amazon Aurora, предоставляемой компанией Amazon Web Services, её ключевым особенностям, истории развития и становления первой в мире serverless relational database service (RDS).*

Ключевые слова: *база данных, SQL, MySQL, PostgreSQL, Amazon Web Services, облако, cloud, serverless, масштабируемость, безопасность*.

Введение

В последние годы с развитием сетевых технологий всё большую популярность в мире приобретают бессерверные вычисления, которые предоставляют ряд преимуществ в сравнении с “классическими” серверными вычислениями. Многие известные компании такие как “The Coca-Cola Company”, “iRobot”, “Autodesk”, “MLB Advanced Media”, “SQUARE ENIX”, “Netflix”, а также многие другие фирмы, используют облачные вычисления для своих постоянных нужд.

Конечно же, облачные технологии не могли обойти стороной и базы данных. В результате возникло множество различных решений бессерверных хранилищ на все случаи жизни. Подробное описание и сравнение всех облачных баз данных достойно полноценной диссертации. Поэтому в данной статье будет рассмотрена лишь одна из них – бессерверная реляционная база данных Amazon Aurora, входящая в семейство Amazon Web Services.

1. Amazon Web Services

Amazon Web Services (AWS) является дочерней компанией Amazon.com, предоставляющей платформу облачных вычислений частным лицам, компаниям и правительствам на платной подписке. Эта платформа предоставляет пользователям в своё распоряжение виртуальный кластер компьютеров, доступный постоянно через Интернет.

Amazon Web Services в значительной степени (наряду с Google Cloud Platform) повлияли на формирование концепции облачных вычислений в целом, и определили основные направления развития публичной модели развёртывания. Длительное время Amazon Web Services было крупнейшим в мире по выручке публичным облаком, во второй половине 2010-х годов уступив по этому показателю Azure от Microsoft, при этом сохраняя доминирование в сегментах инфраструктурных и платформенных услуг.

Ввиду огромного количества все сервисы разбиты на семейства сервисов. Одним из таких семейств является Amazon Relational Database Service.

2. Amazon Relational Database Service

Amazon Relational Database Service (или Amazon RDS) - это сервис распределенных реляционных баз данных, разработанный Amazon Web Services (AWS) и запущенный в 2009 году. Это веб-сервис, предназначенный для упрощения настройки, эксплуатации и масштабирования реляционной базы данных для использования в приложениях. Процессы администрирования, такие как исправление программного обеспечения базы данных, резервное копирование баз данных и включение восстановления на определенный момент времени, управляются автоматически, что в значительной степени упрощает работу с базой данных.

Amazon Relational Database Service предоставляет широкий выбор ядер баз данных в зависимости от предпочтений пользователя: MySQL, MSSQL, PostgreSQL, MariaDB, Oracle и Amazon Aurora.

3. Amazon Aurora

Облачная реляционная база данных Amazon Aurora была запущена в конце 2014-ого года и изначально поддерживала только MySQL в качестве ядра базы данных пользователя. В 2017-ом году Amazon Aurora также приобрела совместимость с PostgreSQL.

Aurora обладает следующими достоинствами: уменьшение времени на управление и настройку базы данных и в результате увеличение времени на создание самого приложения и развития бизнеса пользователя; по мере роста последнего Amazon Aurora будет расширяться автоматически.

Нет необходимости переводить приложение в автономный режим, чтобы добавить хранилище. Вместо этого Amazon Aurora будет добавлять хранилище с шагом 10 ГБ по мере необходимости, вплоть до 64 ТБ. Базовая производительность хранения является быстрой, надежной и предсказуемой - она линейно масштабируется по мере роста объёма хранимых данных, и позволяет в случае необходимости повышать скорость. Пользователь может масштабировать размер экземпляра в считанные минуты и добавлять копии всего парой кликов.

Тем не менее, не смотря на все достоинства, Amazon Aurora, оставалась всё ещё “облачной” базой данных, но не бессерверной. Но всё изменилось в августе 2018 года.

4. Amazon Aurora Serverless

В конце ноября 2017 года в ходе ежегодной конференции AWS re:Invent, посвящённой новейшим разработкам и достижениям Amazon Web Services, была анонсирована Amazon Aurora Serverless – первая в мире бессерверная реляционная база данных. А в августе 2018 года Aurora Serverless стала общедоступна на ядре MySQL версии 5.6.

Aurora Serverless автоматически приспосабливается к текущей нагрузке с чрезвычайно детальными приращениями – почти полностью соответствуя кривой спроса. В процессе разработки пользователь может значительно сэкономить на расходах, автоматически останавливая кластер (масштабируя его до нуля), когда он не используется.

При создании нового кластера Aurora Serverless пользователь может указать минимальное и максимальное количество единиц мощности, используемых Авророй в зависимости от степени текущей нагрузки на кластер. Количество единиц мощности прямо пропорционально стоимости Aurora Serverless, что позволяет выбрать конфигурацию с ценой оптимальной для пользователя.

Правила и показатели для автоматического масштабирования будут автоматически созданы Aurora Serverless и будут включать использование ЦП и количество подключений. Когда Aurora Serverless обнаруживает, что кластеру требуется дополнительная емкость, он получает емкость из “горячего” пула ресурсов для удовлетворения потребностей. Эта новая вычислительная мощность может начать обслуживать трафик в считанные секунды из-за разделения уровня вычислений и уровня хранения, присущих конструкции Aurora.

Кластер может даже автоматически уменьшаться до нуля, если он не видит никаких действий. Это идеально подходит для разработки приложений, которая может занимать длительное время без частого обращения к базе данных. Когда кластер приостановлен, с пользователя взимается плата только за базовое хранилище.

Выводы

Amazon Aurora Serverless стала первой в мире облачной бессерверной реляционной базой данных и сделала большой шаг в развитии и росте популярности облачных бессерверных технологий. Новая база данных не только значительно упрощает разработку и обслуживание базы данных, но и позволяет сэкономить деньги на всех этапах от разработки до постоянного обслуживания приложения. Благодаря Amazon Aurora ещё одна мечта об облачных вычислениях стала реальностью.

Библиография

1. Jeff Barr, «Amazon Aurora – New Cost-Effective MySQL-Compatible Database Engine for Amazon RDS».- [Электронный ресурс]. – Режим доступа: https://aws.amazon.com/blogs/aws/highly-scalable-mysql-compat-rds-db-engine/
2. Jeff Barr, «Introducing Amazon RDS – The Amazon Relational Database Service».- [Электронный ресурс]. – Режим доступа: https://aws.amazon.com/blogs/aws/introducing-rds-the-amazon-relational-database-service/
3. Randall Hunt, «Aurora Serverless MySQL Generally Available».- [Электронный ресурс]. – Режим доступа: https://aws.amazon.com/ru/blogs/aws/aurora-serverless-ga/
4. Jeff Barr, «Sign up Today – Preview of Amazon Aurora with PostgreSQL Compatibility».- [Электронный ресурс]. – Режим доступа: https://aws.amazon.com/blogs/aws/sign-up-today-preview-of-amazon-aurora-with-postgresql-compatibility/

ОБЪЕКТНО-ОРИЕНТИРОВАННЫЕ БАЗЫ ДАННЫХ

Ирина ЧЕРНЕЙ

Технический Университет Молдовы, Департамент Программной Инженерии и Автоматики

Аннотация: *Статья посвящена объектно-ориентированным базам данных. Описана краткая характеристика ООБД и объектно-ориентированной модели данных. Дано описание фундаментальных особенностей объектно-ориентированных баз данных и их преимущества.*

Ключевые слова: *ООБД, объект, класс, наследование, инкапсуляция, ассоциация.*

Введение

Объектно-ориентированные базы данных (OODB) представляют данные в виде объектов и классов. В объектно-ориентированной терминологии *объект* - это сущность реального мира, а ***класс*** - это совокупность объектов. Объектно-ориентированные базы данных следуют фундаментальным принципам объектно-ориентированного программирования (ООП). Сочетание свойств реляционной модели (параллелизм, транзакция и восстановление) с объектно-ориентированными принципами приводит к объектно-ориентированной модели базы данных. Объектно-ориентированная модель базы данных (OODBM) является альтернативной реализацией, что и в реляционной модели. Объектно-ориентированная база данных в принципе похожа на объектно-ориентированный язык программирования. Объектно-ориентированная система управления базами данных - это гибридное приложение, в котором для обработки данных используется комбинация принципов объектно-ориентированной и реляционной базы данных. Тем не менее, мы можем использовать следующую формулу для описания OODBM: На рисунке 1 показана объектно-ориентированная модель базы данных, а также ее принципы и возможности.

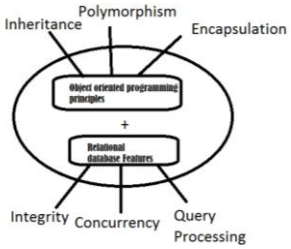


Рисунок 1- Объектно-ориентированная модель базы данных

1. Фундаментальные особенности объектно-ориентированных баз данных

Объекты и Классы. Объектно-ориентированный подход рассматривает все объекты как объекты, которые обладают свойствами (состояния) и методы (поведение). Каждый объект идентифицируется с использованием уникального идентификатора объекта. Например, рассмотрим сущность реального мира под названием «Student». У студента есть состояния или свойства, такие как имя, номер USN, дата рождения, адрес. Аналогично, у студента есть поведение или методы, включающие посещение занятий, письменные экзамены, оплату сборов. Класс представляет собой совокупность подобных объектов.

На рисунке 2 показано, как объект Student может быть представлен.

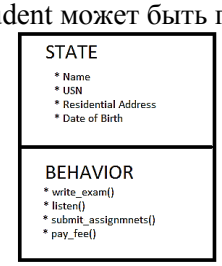


Рисунок 2 - Представление объекта Student.

Инкапсуляция является важной объектно-ориентированной функцией, она скрывает детали реализации от конечных пользователей и отображает только необходимые описания.

Наследование считается важным в объектно-ориентированном проектировании, поскольку оно позволяет использовать его повторно. Он определяется как метод создания новых классов из

существующих классов. Новые классы не только наследуют свойства своего родительского класса, но также имеют свои уникальные свойства.

Ассоциация относится к связям между различными объектами приложения. В объектно-ориентированной базе данных ассоциация обозначается как ссылки между различными объектами.

Концепция *сложных объектов* основана на применении конструкторов к простым объектам. Сложные объекты - это такие элементы, как карты, наборы, списки, кортежи или коллекции многих примитивных объектов.

Простые объекты -это, в основном, такие элементы, как целые числа, байтовые строки и символы.

2. Преимущества объектно-ориентированной базы данных

Существует много преимуществ использования объектно-ориентированных баз данных для создания приложений и управления данными. Некоторые из них следующие:

- Они позволяют легко обмениваться данными, программными компонентами, информацией, вычислительными средами и продуктами;
- Они позволяют интегрировать базы данных, операционные системы, электронные таблицы, языки, системы искусственного интеллекта, текстовые процессоры и другие объекты или приложения;
- Они дают возможность совместного использования продуктов и приложений, что достигается путем наследования и идентификации объекта. Примером этого могут быть гиперссылки, используемые при навигации между различными сайтами.

Заключение

Направление объектно-ориентированных баз данных (ООБД) возникло в середине 1980-х. Однако наиболее активно это направление развивается в последние годы. С каждым годом увеличивается число публикаций и реализованных коммерческих и экспериментальных систем.

Возникновение направления ООБД определяется прежде всего потребностями практики: необходимостью разработки сложных информационных прикладных систем, для которых технология предшествующих систем БД не была вполне удовлетворительной.

Объектно-ориентированные базы данных, которые часто используются инжиниринговыми компаниями и научными лабораториями, могут содержать более сложные данные, чем инструменты реляционных баз данных, для которых требуется таблица с данными.

Библиография

1. OODB (Object-oriented Database). - [Электронный ресурс]. – Режим доступа: https://ru.bmstu.wiki/OODB_(Object-oriented_Database)
2. М.Н. Гринев, С.Д. Кузнецов. Управление данными: достижения и проблемы. - [Электронный ресурс]. – Режим доступа: http://www.ict.edu.ru/ft/005644/62321e1-st08.pdf
3. Сергей Кузнецов. Объектно-ориентированные базы данных - основные концепции, организация и управление: краткий обзор). - [Электронный ресурс]. – Режим доступа:
4. Сиха Багуи. Объектно-ориентированные базы данных: достижения и проблемы. - [Электронный ресурс]. – Режим доступа: https://www.osp.ru/os/2004/03/184042/ http://citforum.ru/database/articles/art_24.shtml
5. Избачков Юрий Сергеевич, Петров Владимир Николаевич, Васильев Александр Алексеевич, Телина Ирина Сергеевна. Информационные системы: Учебник для вузов. 3-е изд. Питер, 2011г., 544 с.

ФУНКЦИОНАЛЬНЫЕ ПРЕИМУЩЕСТВА И НЕДОСТАТКИ AZURE DATA STUDIO

Андрей МАЛЫХИН

Технический Университет Молдовы, Департамент Программной Инженерии и Автоматики

Аннотация: *Статья посвящена рассмотрению возможностей Azure Data Studio. В статье приведены основные сведения, официально предоставляемые Microsoft о данном продукте. Описаны его основные возможности, достоинства и недостатки.*

Ключевые слова: *Azure Data Studio, SQL, кроссплатформенность, open source, keyboard-centric, текстовый редактор, Git Hub.*

Введение

Azure Data Studio, в прошлом SQL Operations Studio - это небольшой бесплатный инструмент для разработки и эксплуатации современных баз данных. Он поддерживает SQL Server для Windows, Linux и Docker, а также базу данных SQL Azure и хранилище данных SQL Azure для Windows, Mac и Linux.

Данная программа построена вокруг редактирования запросов к БД, в то время как SQL Server Management Studio предоставляет более широкий спектр административных функций, оставаясь флагманом в управлении БД.

1. Преимущества Azure Data Studio

Бесплатность и Open Source

Azure Data Studio распространяется бесплатно. Код лежит в открытом доступе на сервисе GitHub, где довольно активно развивается компанией Microsoft и Open Source сообществом. Microsoft заявляет о выпуске новых версий ежемесячно.

Кросплатформенность

Поскольку редактор разработан на базе фреймворка .NET Core, он является кросплатформенным. На данный момент существуют версии для всех основных ОС для персональных компьютеров (Windows, Linux, MacOS). Также, Microsoft предоставляет готовый Docker образ для создания своих контейнеров, содержащих продукцию Microsoft.

Расширяемость

Azure Data Studio предоставляет несколько механизмов, позволяющих расширить функциональность данного приложения. Поскольку оно было создано на основе Visual Studio Code, написание расширений для Azure Data Studio не отличается от создания расширений для VS Code (для справки, расширения для VS Code создаются на языке JavaScript с использованием среды времени исполнения Node.js). Также, программа предоставляет API для управления соединением с хранилищем данных, и проводником объектов.

Малое потребление ресурсов

Azure Data Studio по природе своей является всего лишь редактором запросов к БД, поэтому и потребление ресурсов соответствующее.

IntelliSense

IntelliSense – технология авто дополнения Microsoft, наиболее известная в Visual Studio. Кроме этого, IntelliSense используется для доступа к документации и устранения неоднозначности в именах переменных, функций и методов, используя средства рефлексии. Также, редактор предоставляет умные сниппеты для генерации кода для создания БД, таблиц, представлений, хранимых процедур, пользователей, ролей и т.д. и позволяет создавать свои сниппеты (полезные куски кода).

Встроенный контроль версий

Azure Data Studio предоставляет функции контроля версий прямо внутри редактора. Редактор на данный момент поддерживает GitHub, который должен быть установлен на машине.

Keyboard-centric

Работа в данном редакторе построена вокруг работы с клавиатурой. Azure Data Studio предоставляет большое количество горячих клавиш для того, чтобы можно было работать не отрывая рук от клавиатуры.

2. Недостатки Azure Data Studio

В общем-то все недостатки исходят из того, что Azure Data Studio, по сути, просто редактор в отличии от, скажем, SQL Server Management Studio. Azure Data Studio не предоставляет интергрированной среды для управления SQL инфраструктурой, от SQL-сервера до SQL-базы данных. Простой редактор не даст вам возможности для конфигурации, мониторинга и администрирования SQL сервера.

Заключение

Azure Data Studio является кроссплатформенным инструментом, позволяющим создавать и редактировать запросы к базам данных. Вместе с малым потреблением ресурсов и простотой развёртывания, данный редактор имеет возможность быть использованным в любом современном окружении. Несмотря на малый набор функциональности, по сравнению с SQL Server Management Studio, Azure Data Studio позволяет легко увеличить этот с помощью написания расширений на JavaScript, являющимся одним из наиболее популярных языков в мире. Данный продукт призван не заменить SQL Server Management Studio, а лишь облегчить часть работы, связанную с редактированием запросов.

Библиография

1. Docs.microsoft.com. What is Azure Data Studio? – [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/en-us/sql/azure-data-studio/what-is?view=sql-server-2017
2. Vicky Harp. Azure Data Studio for SQL Server. – [Электронный ресурс]. – Режим доступа: https://cloudblogs.microsoft.com/sqlserver/2018/09/25/azure-data-studio-for-sql-server/
3. Azure Data Studio Git Hub Readme. – [Электронный ресурс]. – Режим доступа: https://github.com/Microsoft/ Azure Data Studio
4. Docs.microsoft.com. Azure Data Studio FAQ. – [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/en-us/sql/azure-data-studio/faq?view=sql-server-2017

ВЫБОР ПРАВИЛЬНОЙ БАЗЫ ДАННЫХ ДЛЯ БОЛЬШИХ ДАННЫХ

Ион ДОГА

Технический Университет Молдовы, Департамент Программной Инженерии и Автоматики

Аннотация: *В данной статье раскрывается базовые принципы по которым стоит выбирать базу данных для больших данных. Технология 3V является сегодня одной из самых актуальных. Приведены примеры баз данных.*

Ключевые слова: *база данных, большие данные, конфиденциальность, big data.*

1. Введение в понятие BigData

В текущее время объемы информации растут экспоненциально. Для того чтобы быстрее реагировать на изменения рынка, получить конкурентные преимущества, повысить эффективность производства нужно получить, обработать и проанализировать огромное количество данных. Для работы с такими объемами информации инженеры были вынуждены модернизировать инструменты для работы над анализом всех данных. Так в 2000-х годах сформировалось понятие BigData, которое было интересно лишь узкому кругу специалистов. Сейчас это слово на слуху у любого, кто интересуется сферой информационных технологий. И это определение, а точнее направление развития ИТ, становится крайне популярным и стратегически важным в последнее время.

Технологии BigData позволяют обработать большой объем неструктурированных данных, систематизировать их, проанализировать и выявить закономерности там, где человеческий мозг никогда бы их не заметил. Это открывает совершенно новые возможности по использованию данных.

Само понятие BigData означает не просто большие пласты данных. Это огромные хранимые и обрабатываемые массивы из сотен гигабайт, и даже петабайт данных. Данных, которые можно обработать и извлечь из них некоторое количество полезной информации. Говоря коротко, можно определить BigData как совокупность технологий обработки информации для получения информации

2. Основные принципы и сферы применения BigData

Большие данные (big data) — обозначение структурированных и неструктурированных данных огромных объёмов и значительного многообразия, эффективно обрабатываемых горизонтально масштабируемыми программными инструментами, появившимися в конце 2000-х годов и альтернативных традиционным системам управления базами данных и решениям класса Business Intelligence. В широком смысле о «больших данных» говорят как о социально-экономическом феномене, связанном с появлением технологических возможностей анализировать огромные массивы данных, в некоторых проблемных областях — весь мировой объём данных, и вытекающих из этого трансформационных последствий. В качестве определяющих характеристик для больших данных традиционно выделяют «три V»: объём (англ. volume, в смысле величины физического объёма), скорость (velocity в смыслах как скорости прироста, так и необходимости высокоскоростной обработки и получения результатов), многообразие (variety, в смысле возможности одновременной обработки различных типов структурированных и полуструктурированных данных. В дальнейшем возникли различные вариации и интерпретации этого признака. С точки зрения информационных технологий в совокупность подходов и инструментов изначально включались средства массово-параллельной обработки неопределённо структурированных данных, прежде всего, системами управления базами данных категории NoSQL, алгоритмами MapReduce и реализующими их программными каркасами, и библиотеками проекта Hadoop. В дальнейшем, к серии технологий больших данных стали относить разнообразные информационно-технологические решения, в той или иной степени обеспечивающие сходные по характеристикам возможности по обработке сверхбольших массивов данных.

Важно заметить, объемы обрабатываемых через BigData данных постоянно растут, также, как и растет скорость ее обработки. Развитие этого направления вполне соответствует современному миру, стремительному и инновационному. С развитием BigData развивались и технологии, и наоборот. На текущий момент, BigData удел не только гигантов IT мира. Это направление, благодаря таким решениям как Hadoop от Apache Software Foundation, набору облачных сервисов от IBM, Amazon, Google, становится доступным практически любым компаниям, работающим в сфере ИТ. А такие

решения как Clickhouse, Cassandra, InfluxDB позволяют войти в сферу работы с BigData даже отдельным персонам.

Использование BigData на сегодняшний день становится обязательным условием для развития крупных ИТ компаний. Без анализа поведения своих пользователей, без возможности прогнозирования, руководствуясь только опытом и интуицией, уже крайне сложно оставаться конкурентоспособным. Настроенная и работающая система BigData способна в секунды предоставить ценнейшую информацию, полученную из анализа миллиардов действий клиентов компании. В текущем бизнесе уже зародилось понятие Data Driven Managment, которое подразумевает управление компанией исходя строго из анализа данных. И такие способы управления показывают блестящие результаты. Facebook, Google, Mail.ru, Яндекс уже давно используют аналитику для принятия решений. На сегодняшний момент в BigData заинтересован и традиционный бизнес, представители которого нуждаются в новых инструментах повышения эффективности.

Ниже представлены основные принципы работы с BigData.

- Горизонтальная масштабируемость: так как данных может быть много, то и система, в которой они хранятся должна быть расширяемой. Если объем данных вырос в 2 раза, то и количество кластеров увеличивается в 2 раза.
- Отказоустойчивость: горизонтальная масштабируемость подразумевает тот факт, что машин в кластере большое количество. И естественно эти машины будут по тем или иным причинам выходить из строя. К примеру, Hadoop-кластер Yahoo насчитывает более 42000 машин. Методы работы с BigData должны учитывать этот фактор и продолжать работать без видимых потерь.
- Локальность данных: в больших системах данные распределены на большом количестве машин. Если данные находятся на одной машине, а обрабатываются на другой, то расходы на передачу этих данных могут и вовсе превысить расходы на обработку. Поэтому важным вопросом в проектировании BigData стоит принцип локальности данных, обработке информации там же, где она хранится.

Сфера использования технологий BigData обширна. Так, с помощью BigData можно узнать о предпочтениях клиентов, об эффективности маркетинговых кампаний или провести анализ рисков. Ниже представлены результаты опроса IBM Institute, о направлениях использования BigData в компаниях.

Большинство компаний используют BigData в сфере клиентского сервиса, второе по популярности направление — операционная эффективность, в сфере управления рисками BigData менее распространены на текущий момент. Следует также отметить, что BigData являются одной из самых быстрорастущих сфер информационных технологий, согласно статистике, общий объем получаемых и хранимых данных удваивается каждые 1–2 года. За период с 2012 по 2014 год количество данных, ежемесячно передаваемых мобильными сетями, выросло на 81 %. По оценкам Cisco, в 2014 году объем мобильного трафика составил 2,5 эксабайта (единица измерения количества информации, равная 10^18 стандартным байтам) в месяц, а уже в 2019 году он будет равен 24,3 эксабайтам. Таким образом, BigData — это уже устоявшаяся сфера технологий, даже несмотря на относительно молодой ее возраст, получившая распространение во многих сферах бизнеса и играющая немаловажную роль в развитии компаний.

3. Основные решения

Технологии BigData, используемые для сбора и обработки BigData, можно разделить на 3 группы: программное обеспечение; оборудование; сервисные услуги. К наиболее распространенным подходам обработки данных относятся:

- SQL — язык структурированных запросов, позволяющий работать с базами данных. С помощью SQL можно создавать и модифицировать данные, а управлением массива данных занимается соответствующая система управления базами данных.
- NoSQL — термин расшифровывается как Not Only SQL (не только SQL). Включает в себя ряд подходов, направленных на реализацию базы данных, имеющих отличия от моделей, используемых в традиционных, реляционных СУБД. Их удобно использовать при постоянно меняющейся структуре данных. Например, для сбора и хранения информации в социальных сетях.
- MapReduce — модель распределения вычислений. Используется для параллельных вычислений над очень большими наборами данных (петабайты* и более). В программном

интерфейсе не данные передаются на обработку программе, а программа — данным. Таким образом запрос представляет собой отдельную программу. Принцип работы заключается в последовательной обработке данных двумя методами Map и Reduce. Map выбирает предварительные данные, Reduce агрегирует их.

- Hadoop — используется для реализации поисковых и контекстных механизмов высоконагруженных сайтов — Facebook, eBay, Amazon и др. Отличительной особенностью является то, что система защищена от выхода из строя любого из узлов кластера, так как каждый блок имеет, как минимум, одну копию данных на другом узле.
- SAP HANA — высокопроизводительная NewSQL платформа для хранения и обработки данных. Обеспечивает высокую скорость обработки запросов. Еще одним отличительным признаком является то, что SAP HANA упрощает системный ландшафт, уменьшая затраты на поддержку аналитических систем.

4. Проблемы Bigdata

Проблемы системы BigData можно свести к трем основным группам: объем, скорость обработки, неструктурированность. Это три V–Volume, Velocity и Variety. Хранение больших объемов информации требует специальных условий, и это вопрос пространства и возможностей. Скорость связана не только с возможным замедлением и «торможением», вызываемым старыми методами обработок, это еще и вопрос интерактивности: чем быстрее процесс, тем больше отдача, тем продуктивнее результат. Проблема неоднородности и неструктурированности возникает по причине разрозненности источников, форматов и качества. Чтобы объединить данные и эффективно их обрабатывать, требуется не только работа по приведению их в пригодный для работы вид, но и определенные аналитические инструменты (системы). Но это еще не все. Существует проблема предела «величины» данных. Ее трудно установить, а значит трудно предугадать, какие технологии и сколько финансовых вливаний потребуется для дальнейших разработок. Ресурсы не бесконечны, хранение всех возможных данных в какой-то момент становится нецелесообразным. И встает необходимость отказа от части данных.

Собственно, это и является главной причиной отсрочки внедрения в компании проектов BigData (если не брать во внимание еще один фактор — довольно высокую стоимость). Подбор данных для обработки и алгоритм анализа может стать не меньшей проблемой, так как отсутствует понимание, какие данные следует собирать и хранить, а какие можно игнорировать. Становится очевидной еще одна «болевая точка» отрасли — нехватка профессиональных специалистов, которым можно было бы доверить глубинный анализ, создание отчетов для решения бизнес-задач и как следствие извлечение прибыли (возврат инвестиций) из BigData. Еще одна проблема BigData носит этический характер. А именно: чем сбор данных (особенно без ведома пользователя) отличается от нарушения границ частной жизни? Так, информация, сохраняемая в поисковых системах Google и Яндекс, позволяет им постоянно дорабатывать свои сервисы, делать их удобными для пользователей и создавать новые интерактивные программы. Поисковики записывают каждый клик пользователя в Интернете, им известен его IP-адрес, геолокация, интересы, онлайн-покупки, личные данные, почтовые сообщения и прочее, что, к примеру, позволяет демонстрировать контекстную рекламу в соответствии с поведением пользователя в Интернете. При этом согласия на это не спрашивается, а возможности выбора, какие сведения о себе предоставлять, не дается. То есть, по умолчанию в BigData собирается все, что затем будет храниться на серверах данных сайтов. Здесь можно затронуть другую проблему — обеспечение безопасности хранения и использования данных. Например, сведения о возможных покупателях и их история переходов на сайтах интернет-магазинов однозначно применимы для решения многих бизнес-задач. Но безопасна ли аналитическая платформа, которой потребители в автоматическом режиме (просто потому, что зашли на сайт) передают свои данные, — это вызывает множество споров. Современную вирусную активность и хакерские атаки не сдерживают даже супер-защищенные серверы правительственных спецслужб.

Заключение

BigData открывает перед нами новые горизонты в планировании производства, образовании, здравоохранении и других отраслях. Если их развитие будет продолжаться, то технологии BigData могут поднять информацию, как фактор производства, на совершенно новый качественный уровень.

Информация станет не только равноценна труду и капиталу, но и возможно станет наиважнейшим ресурсом современной экономики.

Библиография

1. Веретенников А. В. BigData: анализ больших данных сегодня // Молодой ученый. — 2017. — №32. — С. 9-12. — URL https://moluch.ru/archive/166/45354/
2. Работа с Big Data: основные области и возможности . – [Электронный ресурс]. – Режим доступа: https://www.marketing.spb.ru/lib-around/stat/Big_Data.htm
3. Большие данные. – [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/%D0%91%D0%BE%D0%BB%D1%8C%D1%88%D0%B8%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5

СИЛЬНЫЕ СТОРОНЫ MySQL ДЛЯ ВЫСОКОНАГРУЖЕННЫХ ПРОЕКТОВ

Дмитрий ДРЕГЛЯ

Технический Университет Молдовы, Департамент Программной Инженерии и Автоматики

Аннотация: *Статья посвящена системе управления базами данных MySQL. Описаны ее сильные стороны в контексте сравнения с PostgreSQL. Рассмотрены механизмы репликации и компрессии данных.*

Ключевые слова*: база данных, СУБД, MySQL, PostgreSQL, репликация, компрессия данных.*

Введение

Базы данных - это специально разработанное хранилище для различных типов данных. Каждая база данных имеет определённую модель (реляционная, документно-ориентированная), которая обеспечивает удобный доступ к данным. Системы управления базами данных (СУБД) – это специальные приложения (или библиотеки) для управления базами данных различных размеров и форм. СУБД – это, наверное, самый сложный класс программ.

В статье пойдет речь о двух СУБД проектах - *MySQL и PostgreSQL,* с более чем 20-ти летней историей, с огромным набором функционала, и в таком случае детали - это главная проблема. Ни в один доклад эта информация не влезет, однако мы постараемся построить общую картину.

Речь пойдет о камне преткновения при сравнении этих двух СУБД, а именно о репликации, а также о компрессии данных.

1. Репликация

Репликация – это основа любых нагруженных проектов. Для таких проектов очень важна высокая доступность данных, а также их сохранность. Репликация позволяет разгрузить один сервер и распределять запросы между множеством серверов. Это требуется по ряду причин:

- Если мастер сервер становится недоступен, то мы переключаем клиентов на резервные сервера;
- Географическая распределёность клиентов – если у нас большой, масштабный проект, и клиент находится в США, а дата-центр в Азии , то работать им будет не очень комфортно просто в силу законов физики.

Существует репликация двух видов: физическая и логическая.

Физическая – самый простой вид репликации, она работает на низком уровне и передает на реплику точную побайтовую копию всех данных. Физическая репликация отсутствует в MySQL, но встроена в PostgreSQL с 2010 года.

Логическая – передает только изменение данных. К примеру, можно передать SQL запрос на slave-сервер и там проигрывать их заново. В PostgreSQL все аналоги решений не покрывают функционал логической репликации в MySQL.

У логической репликации следующие плюсы:

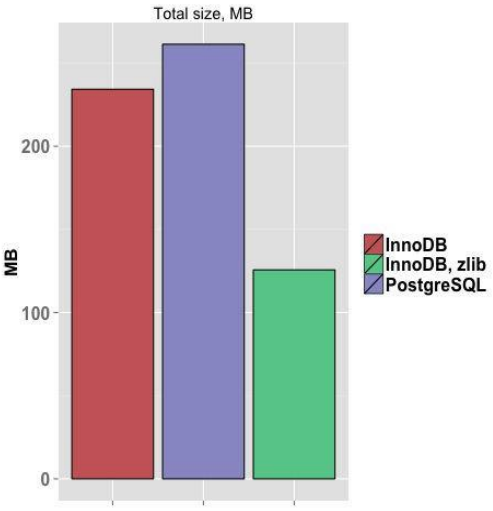
- Не создает точную побайтовую копию данных, следственно если мы на реплике перестроим таблицу, перестроим данные и индексы, то она продолжит работать, физическая нет;
- Позволяет иметь различную структуру данных на мастере и на реплике. Можно, например, добавить колонку, назначить для нее дефолтное значение и логическая репликация продолжит работать, несмотря на то, что на мастере этой колонки ещё нет. Это свойство как раз используется в высоконагруженных проектах, когда нужно изменить схему в большом распределённом кластере, сначала мы делаем каскадный апгрейд. Изменяем схемы на каждой реплике, переключаем клиентов на одну из реплик и старый мастер тоже апгрейдим;
- При логической репликации нет ограничения на чтение, с физической репликацией есть некоторые ограничения, связанные с работой механизма мульти-версионности.
- Можно делать частичную репликацию, то есть, передавать на slave не весь набор данных, а только его часть, это также используется в высоконагруженных проектах;
- Компактность – обычно набор данных передаваемых по сети намного меньше чем при физической репликации. Пример с индексами, если у нас есть много индексов в таблице, и мы изменяем какую-то строчку, то при физической репликации на реплику пойдет изменение не только данных, но и всех индексов, а при логической только данных.

2. Компрессия данных

Компрессия данных тоже достаточно важная функциональность для высоко нагруженных проектов. Любой проект, который работает с каким-то не тривиальным объемом данных заинтересован в том, чтобы размер этих данных на диске уменьшить. Для интернет гигантов вроде Facebook – это вопрос выживаемости.

InnoDB – это одна из подсистем низкого уровня для СУБД MySQL. InnoDB поддерживает постраничную компрессию данных, т.е. компрессируются как данные, так и индексы, и результаты компрессии и декомпрессии кэшируются, то есть в памяти хранятся как компрессированные, так и декомпрессированные образы страниц.

В PostgreSQL компрессии нет как таковой, есть нечто под названием TOAST, но компрессируются только отдельные записи. причем не все, а только те, которые длиннее 2 КБ.



Компрессия работает только для полей переменной длины.

Компрессируются только данные, индексы не компрессируются и результат компрессии и декомпрессии никак не кэшируется. То есть, если мы будем тысячу раз читать одну и ту же запись, и она скомпрессирована, то PosgtreSQL будет тысячу раз её декомпрессировать.

К примеру, была создана некая таблица, туда было загружено миллион записей и красным мы видим объем данных InnoDB без компрессии, синий это PostgreSQL, а зеленый это InnoDB с компрессией.

Для этой таблицы TOAST компрессия даже не включилась, т.к. средний размер записи был около 180 байт, а PostgreSQL требует 2 КБ, и то, для того чтобы решить компрессировать её или нет.

Заключение

Есть много причин использовать MySQL в высоко нагруженных проектах. Некоторые важные для крупных проектов возможности отсутствуют в PostgreSQL и не реализованы до сих пор. Однако, это не значит, что MySQL должна использоваться абсолютно везде.

Выбор между данными СУБД сложный вопрос, если кто-то вам предлагает простой ответ на этот вопрос, то этот человек либо не объективен, либо не компетентен.

Библиография

1. Алексей Копытов. Сильные стороны MySQL, OSSDEVCONF-2016. – [Электронный ресурс]. – Режим доступа: http://catcut.net/zNzA
2. Сергей Яковлев. MySQL и PostgreSQL. Сравнительный анализ. – [Электронный ресурс]. – Режим доступа: https://www.ibm.com/developerworks/ru/library/os-mysql-postgresql/01/index.html
3. Devacademy.ru. SQLite vs MySQL vs PostgreSQL: сравнение систем управления базами данных. – [Электронный ресурс]. – Режим доступа: http://devacademy.ru/posts/sqlite-vs-mysql-vs-postgresql/

РЕЛЯЦИОННАЯ СУБД MariaDB

Александр ЖУКОВ

Технический Университет Молдовы, Департамент Программной Инженерии и Автоматики

Аннотация: *Статья посвящена реляционной СУБД MariaDB. Описана краткая характеристика СУБД. Дано описание достоинств и недостатков MariaDB.*

Ключевые слова*: MariaDb, реляционная, СУБД, My SQL.*

1. Введение

Сервер MariaDB является одним из самых популярных серверов баз данных в мире. Разработана MariaDB разработчиками MySQL. Весь исходный код MariaDB распространяется под лицензиями GPL, LPGL или BSD. MariaDB не содержит закрытых модулей или компонентов, на подобие тех, что содержатся в MySQL Enterprise Edition. Однако, это не влияет на доступный функционал MariaDB. Все технологии, существующие в закрытой версии MySQL 5.5 Enterprise Edition, в полном объеме представлены и в MariaDB.

Известные пользователи включают Википедию, WordPress.com и Google.

MariaDB используется в широком спектре приложений, от банковского до веб-сайтов. Это расширенная замена для MySQL. MariaDB используется потому, что она быстрая, масштабируемая и надежная, с богатой экосистемой систем хранения, плагинами и многими другими инструментами что делает ее универсальной для самых разных случаев использования. MariaDB разработана как программное обеспечение с открытым исходным кодом и как реляционная база данных, она предоставляет интерфейс SQL для доступа к данным. В последних версиях MariaDB также есть функции GIS и JSON.

Еще больше механизмов хранения данных

В дополнении к стандартным механизмам хранения данных, - MyISAM, Blackhole, CSV, Memory, и Archive, РСУБД - MariaDB содержит следующие способы хранения данных:

- Aria;
- XtraDB (прозрачная замена InnoDB);
- PBXT (Доступно в MariaDB 5.1, 5.2 и 5.3. Не поддерживается с версии 5.5);
- FederatedX (прозрачная замена Federated);
- OQGRAPH - с версии 5.2;
- SphinxSE - с версии 5.2;
- IBMDB2I. Компания Oracle убрала поддержку этого механизма хранения данных, начиная с версии MySQL 5.1.55, но исходный код сохранен в MariaDB до версии 5.5;
- TokuDB;
- Cassandra (MariaDB 10.0);
- CONNECT (MariaDB 10.0);
- SEQUENCE (MariaDB 10.0);
- Spider (MariaDB 10.0).

2. Сравнение показателей MariaDB и MySQL

MariaDB имеет несколько оптимизаций, которые, как правило, повышают производительность по сравнению с MySQL. Фактически, это было именно то, что было в виду, когда MariaDB был запущен Майклом Видениусом, оригинальным основателем как MySQL, так и MariaDB.

Представления базы данных. Например, существует огромная оптимизация производительности в отношении представлений базы данных. «Представления» - это, по сути, виртуальные таблицы базы данных, которые могут запрашиваться как обычные таблицы базы данных. В MySQL, когда вы запрашиваете представление, запрашиваются все таблицы, связанные с представлением, независимо от того, что запрос может не потребовать некоторых из представлений. Это было оптимизировано в MariaDB, где запрашиваются только те таблицы, которые требуются по запросу.

ColumnStore. В качестве еще одного примера, MariaDB обеспечивает еще одно мощное повышение производительности в виде «ColumnStore», которое представляет собой распределенную

архитектуру данных, которая позволяет значительно масштабировать MariaDB. Она может масштабироваться линейно, чтобы хранить петабайты данных на разных серверах в кластере базы данных.

Лучшая производительность во флэш-накопителе. MariaDB также предоставляет механизм хранения MyRocks, который добавляет к нему базу данных RocksDB. RocksDB - это база данных, которая была разработана для лучшей производительности во флэш-памяти, обеспечивая более высокий уровень сжатия данных.

Сегментированный ключевой кеш. MariaDB представляет еще одно улучшение производительности в виде кэша сегментированных ключей. В типичном кэше различные потоки конкурируют за блокировку записи в кэше. Эти замки называются мьютексами. Когда несколько потоков конкурируют за мьютекс, только один из них может получить его, в то время как другие должны дождаться освобождения блокировки до выполнения операции. Это приводит к задержкам выполнения этих потоков, замедляя производительность базы данных. В случае Segmented Key Cache поток не должен блокировать всю страницу, но он может блокировать только тот сегмент, к которому принадлежит страница. Это помогает нескольким потокам работать параллельно, тем самым увеличивая параллельность в приложении, что приводит к повышению производительности базы данных.

Виртуальные столбцы. Интересной особенностью, поддерживаемой MariaDB, является функция виртуальных столбцов. Эти столбцы способны выполнять вычисления на уровне базы данных. Это очень полезно, когда многие приложения обращаются к одному столбцу, поэтому нет необходимости писать вычисления в каждом приложении - база данных может это сделать для вас. Эта функция недоступна в MySQL.

Параллельное выполнение запросов. Одна из последних версий MariaDB - 10.0 - позволяет параллельно выполнять несколько запросов. Идея состоит в том, что некоторые запросы от Master могут быть реплицированы в подчиненном устройстве и поэтому могут выполняться параллельно. Этот параллелизм в выполнении запроса, безусловно, предоставляет MariaDB преимущество над MySQL.

Объединение потоков. MariaDB также представляет новую концепцию «Объединение потоков». Раньше, когда требовалось несколько подключений к базе данных, для каждого соединения был открыт поток, который приводил к архитектуре, основанной на «одном потоке на одно соединение». С «Объединением потоков» будет существовать пул открытых потоков, который новое соединение может получить и запросить базу данных. Таким образом, новый поток не нужно открывать для каждого нового запроса на соединение, что приводит к более быстрым результатам запроса. Эта функция доступна в версии Enterprise для MySQL, но, к сожалению, недоступна в версии сообщества.

Заключение

MariaDB, несомненно, достаточно мощная СУБД, и предоставляет множество функций, которые чрезвычайно полезны и не поддерживаются в MySQL. Такие функции действительно делают MariaDB выгодным выбором для использования в качестве основной базы данных. Вообще говоря, организации, которые уже приобрели лицензии для Oracle, не должны инвестировать в MariaDB. Тем не менее, те, кто начинает заново и хотят решить, какую базу данных использовать, несомненно, MariaDB - лучший выбор.

Библиография

1. The MariaDB Foundation – Supporting continuity and open collaboration in the MariaDB ecosystem. – [Электронный ресурс]. – Режим доступа: https://mariadb.org/
2. Wikipedia. MariaDB. – [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/MariaDB
3. MariaDB: drop-in replacement for MySQL. – [Электронный ресурс]. – Режим доступа: https://github.com/MariaDB/server

РАСПРЕДЕЛЕННЫЕ И ПАРАЛЛЕЛЬНЫЕ СИСТЕМЫ БАЗ ДАННЫХ

Владислав КУШНИР

Технический Университет Молдовы, Департамент Программной Инженерии и Автоматики

Аннотация: *В статье представлен обзор технологий распределенных и параллельных СУБД, выделены их отличительные черты, отмечены схожие признаки. Цель статьи – помочь в осмыслении уникальной роли систем каждого из этих двух типов и их взаимодополняемости в решении задач управления данными.*

Ключевые слова: *архитектура, клиент-сервер, без разделяемых ресурсов, обработка, запрос, оптимизация, декомпозиция, локализация данных, пространство поиска, глобальная оптимизация.*

Введение

Становление систем управления базами данных (СУБД) совпало по времени со значительными успехами в развитии технологий распределенных вычислений и параллельной обработки. В результате возникли *распределенные системы управления базами данных и параллельные системы управления базами данных*. Именно эти системы становятся доминирующими инструментами для создания приложений интенсивной обработки данных. Благодаря интеграции рабочих станций в распределенную среду становится возможным более эффективное распределение функций в ней, когда прикладные программы выполняются на рабочих станциях, называемых *серверами приложений,* а базы данных обслуживаются выделенными компьютерами, называемыми *серверами баз данных*. Это служит источником развития таких распределенных архитектур, где в роли узлов выступают не просто компьютеры общего назначения, а специализированные серверы.

1. Архитектурные проблемы

Существует множество альтернатив распределенной обработки. Наиболее популярна в настоящее время *архитектура клиент-сервер*, когда множество машин-клиентов осуществляют доступ к одному серверу баз данных. В таких системах, которые можно определить как системы типа *много-клиентов/один-сервер*, проблемы управления базой данных решаются относительно просто, поскольку вся она хранится на одном сервере. Задачи, с которыми приходится здесь сталкиваться, – это управление буферами клиентов, кэширование данных и, возможно, блокировки. Управление данными реализуется *централизованно* на одном сервере. Более распределенной и более гибкой является архитектура типа *много-клиентов/много-серверов*, когда база данных размещена на нескольких серверах, которым, для того чтобы вычислить результат пользовательского запроса или выполнить транзакцию, необходимо взаимодействовать друг с другом. Каждая клиентская машина имеет свой "домашний" сервер; ему она направляет пользовательские запросы. Взаимодействие серверов друг с другом прозрачно для пользователей. В большинстве существующих СУБД реализован один из этих двух типов архитектуры клиент-сервер. В истинно распределенной СУБД клиентские и серверные машины не различаются. В идеале каждый узел может выступать и как клиент, и как сервер. Такие архитектуры, тип которых определяют как *равный-к-равному*, требуют сложных протоколов управления данными, распределенными по нескольким узлам. Предложение продуктов такого вида задерживается из-за сложности необходимого для реализации их программного обеспечения. Архитектуры параллельных систем варьируются между двумя крайними точками, называемыми *архитектура без разделяемых ресурсов* и *архитектура с разделяемой памятью***.** Промежуточную позицию занимает *архитектура с разделяемыми дисками.*

2. Обработка и оптимизация запросов

Обработка запроса – это процесс трансляции декларативного определения запроса в операции манипулирования данными низкого уровня. *Оптимизация запроса* – это процедура выбора "наилучшей" стратегии выполнения запроса из множества альтернатив.

Для централизованной СУБД весь процесс состоит обычно из двух шагов: *декомпозиции запроса* и *оптимизации запроса*. Декомпозиция запроса – это трансляция его с языка SQL в выражение реляционной алгебры. В ходе декомпозиции запрос подвергается семантическому анализу; при этом некорректные запросы отвергаются, а корректные упрощаются. Для заданного SQL-запроса существует более чем одно алгебраическое представление, причем некоторые из них могут быть "лучше" других. "Качество" алгебраического выражения определяется исходя из объема

затрат, необходимых для его вычисления. В распределенной СУБД между шагами декомпозиции и оптимизации запроса включаются еще два действия: *локализация данных* и *глобальная оптимизация запроса*. Исходной информацией для локализации данных служит исходное алгебраическое выражение, полученное на шаге декомпозиции запроса. Основная роль локализации данных заключается в том, чтобы локализовать участвующие в запросе данные, используя информацию об их распределении. На этом шаге выявляются фрагменты, реально участвующие в запросе, и запрос преобразуется к форме, где операции применяются уже не к глобальным отношениям, а к фрагментам. Распределенные отношения реконструируются путем применения инверсии правил фрагментации. Это называется *программой локализации*. Программа локализации для горизонтально (вертикально) фрагментированного отношения представляет собой объединение (union) (соединение (join)) соответствующих фрагментов. Таким образом, на шаге локализации данных каждое глобальное отношение запрос заменяется его программой локализации, а затем результирующий фрагментный запрос упрощается и реструктурируется с целью получения другого "хорошего" запроса. Цель глобальной оптимизации – найти стратегию выполнения запроса, близкую к оптимальной. Стратегию выполнения распределенного запроса можно выразить в терминах *операций реляционной алгебры* и *коммуникационных примитивов*, используемых для пересылки данных между узлами. На предыдущих шагах запрос уже был в определенной мере оптимизирован, в частности, за счет удаления избыточных выражений. Однако проведенная оптимизация не зависела от характеристик фрагментов, например, их мощности. Оптимизация запроса заключается в нахождении "наилучшего" плана из множества возможных планов, исследуемых оптимизатором.

Оптимизатор запросов обычно представляется в виде трех компонентов: пространство поиска, модель стоимости и стратегия поиска. *Пространство поиска* – это множество альтернативных планов выполнения исходного запроса. Эти планы эквивалентны в том смысле, что они дают один и тот же результат, но различаются порядком и способами выполнения отдельных операций. *Модель стоимости* – это способ оценить стоимость данного плана выполнения запроса. Для достижения точности модель стоимости должна основываться на точных знаниях о среде параллельных вычислений. *Стратегия поиска* – это способ обхода пространства поиска и выбора наилучшего плана.

Выводы

За последние несколько лет распределенные и параллельные СУБД стали реальностью. Они предоставляют функциональность централизованных СУБД, но в такой среде, где данные распределены между компьютерами, связанными сетью, или между узлами многопроцессорной системы. Распределенные СУБД допускают естественный рост и расширение баз данных путем простого добавления в сеть дополнительных машин. Подобные системы обладают более привлекательными характеристиками "цена/производительность", благодаря современным прогрессивным сетевым технологиям. Параллельные СУБД – это, пожалуй, единственный реалистичный подход для удовлетворения потребностей многих важных прикладных областей, которым необходима исключительно высокая пропускная способность баз данных. Поэтому при проектировании параллельных и распределенных СУБД следует предусмотреть в них соответствующие протоколы и стратегии обработки, направленные на достижение высокой производительности.

Библиография

1. Распределенные и параллельные системы управления базами данных. – [Электронный ресурс]. – Режим доступа: http://unesco.kemsu.ru/study_work/method/bd/umk/book/gl_1.html
2. Параллельные базы данных. – [Электронный ресурс]. – Режим доступа: https://knowledge.allbest.ru/programming/2c0a65635b3ac68a4d53b89421306c36_0.html
3. М.Тамер Оззу, Патрик Валдуриз. Распределенные и параллельные системы баз данных. Журнал системы управления базами данных № 4/1996, издательский дом “Открытые системы”. – [Электронный ресурс]. – Режим доступа: - http://citforum.ru/database/classics/distr_and_paral_sdb/

MICROSOFT SQL SERVER: СРАВНЕНИЕ ВЕРСИЙ

Максим ТИМОФЕЕВ

Технический Университет Молдовы, Департамент Программной Инженерии и Автоматики

***Аннотация**: Статья посвящена системе управления базами данных Microsoft SQL Server. Произведен сравнительный анализ версий Microsoft SQL Server, указаны нововведения различных версий.*

***Ключевые слова:** СУБД, Microsoft SQL Server, сравнение версий.*

Введение

Microsoft SQL Server — система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов SQL с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

Реляционная СУБД отвечает за поддержку структуры базы данных и решает следующие задачи:

- поддерживает связи между данными в базе;
- гарантирует корректное хранение данных и выполнение правил, регламентирующих связи между ними;
- восстанавливает данные после аварии системы, переводя их в согласованное состояние, зафиксированное до сбоя.

1. Сравнение версий Microsoft SQL Server

SQL Server 2000

Версия, вышедшая уже довольно давно. Надежная и проверенный временем стабильная версия сервера. SQL Server 2000 предоставляет много возможностей. SQL Server 2000 обладал полностью переписанным движком, поддержкой новых хранимых структур, методов доступа к данным, технологией блокировки записей, алгоритмов восстановления, новой архитектурой логирования транзакций, новой архитектурой памяти и оптимизатором. SQL Server 2000 обладал многочисленными языковыми улучшениями, равно как и серьёзными изменениями в представленных ранее объектах, таких как например, табличные ограничения, представления и триггеры, в которых нуждались все разработчики и большинство администраторов БД.

SQL Server 2005

Чтобы соответствовать возросшим потребностям всех категорий пользователей, корпорация Microsoft выпустила семейство продуктов SQL Server 2005. Будучи недорогой и популярной СУБД, SQL Server 2005 предоставляет беспрецедентную стоимость и функциональность, сравнимую с решениями ведущих мировых производителей. Снижение времени простоя приложений, быстрая масштабируемость, высокая производительность и надежное управление безопасностью SQL Server 2005 - огромный шаг вперед в поддержке наиболее востребованных систем масштаба предприятия в мире. Поскольку SQL Server является составной частью Windows Server System, заказчики получают дополнительные преимущества: уменьшение совокупной стоимости владения, сокращение времени разработки, возросшую управляемость и интегрируемость, как результат общей стратегии разработки, реализованной во всех продуктах Windows Server System.

SQL Server 2014

SQL Server 2014 разработан для использования в гибридной среде. Версия реляционной СУБД Microsoft SQL Server 2014 обеспечила значительный прирост производительности в рамках систем онлайн транзакций. Данная версия СУБД имела встроенный in-memory OLTP «движок», построенный на технологиях Heckaton, новой, в то время, разработке Microsoft Research. Также эта версия позволила снизить издержки на оборудование, поскольку требования к необходимому для

работы СУБД «железу» стали более демократичными в части вычислительной мощности и количества серверов.

Платформа SQL Server 2014 обеспечила комплексный подход к управлению и анализу данных, в частности:

- обработку вычислений в оперативной памяти (in-memory OLTP), увеличивающую производительность в среднем в 10 – 30 раз;
- более высокую производительность обработки запросов и скорость загрузки данных;
- инструменты для бизнес-аналитики как для частных, так и для корпоративных пользователей;
- поддержку критически важных приложений, соответствующую требованиям производительности, безопасности, масштабируемости;
- мощные аналитические инструменты, удобные в использовании и имеющие привычный интерфейс.

SQL Server 2016

Благодаря SQL Server 2016 можно создавать важные аналитические приложения с помощью масштабируемой гибридной платформы баз данных, в которую уже встроено все необходимое — от возможностей эффективной обработки в памяти и повышенной безопасности до функций аналитики в базе данных. В выпуске SQL Server 2016 были добавлены новые средства безопасности, возможности выполнения запросов, интеграции Hadoop, облачной интеграции, аналитики R, а также множество других усовершенствований. Преимуществами SQL Server 2016:

- Query Data Store работает как рекордер информации для базы данных, обеспечивая полную историю исполнения запросов, так что DBA может отслеживать ресурсоёмкие запросы и оптимизировать их;
- Предлагаются усовершенствованные возможности управления сервером для Master Data Services;
- Новая технология Always Encrypted (всегда зашифровано) помогает защитить данные при хранении и при перемещении, в локальных системах и в облаке, с помощью основных ключей, расположенных в приложении, без внесения изменений в приложение;
- Технология Stretch Database (расширение базы данных) позволяет держать под рукой большую часть данных клиентских журналов, перемещая данные OLTP в Microsoft Azure безопасным способом без необходимости внесения изменений в приложение;
- Повышение эффективности обработки в памяти, при которой транзакции обрабатываются в 30 раз, а запросы — в 100 раз быстрее, чем в дисковых реляционных базах данных и системах оперативной аналитики в реальном времени;
- Изучение бизнес-данных посредством визуализации на мобильных устройствах с использованием собственных приложений для Windows, iOS и Android;
- Быстрые гибридные резервные копии, высокий уровень доступности и сценарии аварийного восстановления для резервного копирования и восстановления локальных баз данных в Microsoft Azure и размещение вторичных реплик SQL Server AlwaysOn в Azure.

SQL Server 2017

Новая версия SQL Server выпускается в тех же редакциях, как предыдущая, Microsoft SQL Server 2016.

Новые возможности SQL Server 2017:

- Поддержка платформы LINUX. SQL Server 2017 теперь можно установить на операционную систему Linux. Microsoft обещает в скором времени поддерживать почти все дистрибутивы;
- Поддержка языка Python. Напомню, в предыдущей версии SQL Server 2016 был интегрирован язык R, Microsoft решила пойти дальше, и уже сейчас в SQL Server 2017 был интегрирован язык Python, который можно использовать для аналитики, создавая интеллектуальные приложения в базе данных SQL Server. Другими словами, пользовательское приложение может просто вызывать хранимую процедуру на SQL сервере, в которой будет исполняться код R или Python, анализируя при этом данные в БД, не передавая их пользовательскому приложению;

- Адаптивная обработка запросов. В SQL Server 2017 появилось новое поколение функций обработки запросов, позволяющие улучшить производительность запросов в SQL Server путем адаптации к среде выполнения рабочих нагрузок приложений;
- Поддержка графовых данных.

Заключение

Microsoft SQL Server - представляет собой комплексную платформу для работы с данными, которая способна расти вместе с компанией. Высокая производительность SQL Server позволяет соответствовать растущим потребностям приложений баз данных и ИТ-инфраструктуры. Технологии поддержки новейшего аппаратного обеспечения, включенные в состав SQL Server, помогают в полной мере использовать преимущества современных серверных платформ и повышать производительность Microsoft SQL Server соразмерно развитию предприятия. SQL Server способна поддерживать огромные базы данных, эффективно распределяя нагрузку и информацию. На базе Microsoft SQL Server могут быть построены решения для компаний малого, среднего и крупного бизнеса.

Библиография

1. Codd E.F. Relational completeness of data base sublanguages. IBM Research Laboratory, San Jose, California. KO 987 (#170041), March 6, 1972, Computer Sciences.
2. Энтони Молинаро. SQL. Сборник рецептов. – Пер. с англ. – СПб: Символ Плюс, 2009. – 672 с.
3. Wikipedia. Microsoft SQL Server. – [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Microsoft_SQL_Server
4. Microsoft SQL Server. – [Электронный ресурс]. – Режим доступа: https://www.microsoft.com/ru-ru/sql-server/sql-server-downloads
5. Wikipedia. Список версий Microsoft SQL Server. – [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D0%B9_Microsoft_SQL_Server
6. Как определить версию, выпуск и уровень обновления системы SQL Server и ее компонентов. – [Электронный ресурс]. – Режим доступа: https://support.microsoft.com/ru-kz/help/321185/how-to-determine-the-version-edition-and-update-level-of-sql-server-an
7. Microsoft SQL ServerVersions List. – [Электронный ресурс]. – Режим доступа: https://sqlserverbuilds.blogspot.com/

БАЗА ДАННЫХ MICROSOFT SYSTEM CENTER CONFIGURATION MANAGER

Станислав ФРИДМАН

Технический Университет Молдовы, Департамент Программной Инженерии и Автоматики

Аннотация: *в данной работе проводится обзор базы данных Microsoft System Center Configuration Manager. В статье выявлена необходимость использования Microsoft SCCM в управлении IT инфраструктурой, кратко описан функционал и принцип работы. В работе перечислены и описаны группы представлений базы данных.*

Ключевые слова: *база данных, Microsoft, System Center, администрирование систем, программное обеспечение, группы представлений, SQL.*

Введение. Необходимость использования Microsoft SCCM

Как известно, любая IT инфраструктура – совокупность компьютеров, серверов, принтеров, IP-телефонов, видеокамер. Как правило, больше всего компьютеров. В крупных IT компаниях их количество может достигать сотни или тысячи единиц. Возникает логически обоснованный вопрос: как управлять этим большим количеством оборудования? Как устанавливать программное обеспечение или отследить компьютер, зараженный вирусом? Если установка приложения с внешнего хранилища или из интернета займет в среднем 5 минут и это без возможных возникающих проблем – тогда нужно несколько рабочих дней или даже недель, чтобы установить один продукт на все компьютеры.

Чтобы облегчить жизнь системным инженерам и администраторам, компания Microsoft разработала продукт для централизованного администрирования компьютерами – Microsoft System Center Configuration Manager. С помощью этого продукта можно удаленно управлять компьютерами, при этом пользователи даже не будут замечать происходящих процессов (установка, удаление ПО, обновлений и т.д.)

1. База данных в Microsoft System Center Configuration Manager

Microsoft System Center Configuration Manager (сокр. SCCM) – продукт для управления IT инфраструктурой на семействе ОС Windows. SCCM предоставляет следующий функционал:

- управление обновлениями;
- развертывание ПО и операционных систем;
- мониторинг использования ПО;
- инвентаризация оборудования;
- удаленное управление;
- управление виртуальными машинами и мобильными устройствами на ОС Windows.

С логической точки зрения все управляемые системы объединяются в сайты (площадки).

Сайты содержат в себе: серверы сайта, системы сайта, выполняющие определенные роли по управлению инфраструктурой, собственно, управляемые клиенты.

Каждый из серверов сайта должен иметь доступ к базе данных Microsoft SQL Server. Сайты могут подразделяться на административные (Administrative Site), основные (Primary Site) и дополнительные (Secondary Site), в SCCM 2012 все сайты имеют собственную базу данных.

Сайты также образуют иерархию родительских (Parent Site) и дочерних (Child Site). Каждый дочерний сайт имеет только один родительский сайт. Дочерние сайты, в свою очередь, могут иметь свои дочерние сайты и так далее.

База данных SCCM - система сайта SCCM, предназначенная для хранения данных сайта: настроек сайта в целом и отдельных его компонентов, клиентских политик, данных, собранных на клиентах, информации о ресурсах сети предприятия и сообщений о состоянии.

Текущая версия базы SCCM (build 1810) содержит 1623 таблицы и столько же представлений, большинство имеют понятные названия, поэтому легко можно понять их назначение (содержимое). Просмотреть можно с помощью SQL Server Management Studio. Таблицы имеют такие названия: DEVICE_CAMERA_DATA, Network_DATA, Disk_DATA. Исходя из опыта, можно сказать, что с таблицами приходится работать крайне редко, а вот с представлениями - регулярно. Представления и, соответственно, таблицы можно разделить на следующие группы:

- V_* - данные по умолчанию;
- V_R_* - представления, содержащие данные обнаружения для данных, содержащихся в скалярных свойствах WMI (Windows Management Instrumentation);
- V_RA_* - представления, содержащие данные обнаружения для данных, содержащихся в свойствах массива WMI;
- V_G_* - данные инвентаризации для пользовательских архитектур;
- V_GS_* - текущие данные инвентаризации;
- V_HS_* - история данных инвентаризации;
- V_AI_* - данные Asset Intelligence. Содержат информацию об установленном ПО и их лицензировании;
- V_LU_* - данные Localized Unit;
- V_CI_* - данные элементов конфигурации операционной системы, приложений, общих настроек, обновлений ПО;
- V_CH_* - данные Client Health. Содержат информацию о состоянии установленных SCCM клиентов на оборудовании пользователей.

На сайте Microsoft размещен xlsx файл со всеми атрибутами всех таблиц базы данных SCCM.

Как в любой базе данных, в БД SCCM можно узнать данные инвентаризации или обновлять таблицы с помощью SQL запросов. Для этого необходимо обладать знаниями SQL и иметь представление о работе с SQL Server и SQL Server Management Studio. Гораздо удобнее делать это с помощью оснастки SCCM – пользовательский интерфейс позволяет управлять БД более интуитивно.

Заключение

SCCM позволяет системным администраторам управлять большим количеством оборудования. По сути – это большая база данных, где указываются настройки для групп или конкретных пользователей. Как было написано ранее, существуют 2 способа управления этой БД – с помощью SQL и с помощью оснастки SCCM. Как правило, администраторы и системные инженеры пользуются преимущественно оснасткой.

Перед настройкой такого решения по управлению IT инфраструктурой, как SCCM, необходимо учитывать несколько параметров. В первую очередь, следует задаться вопросом: а стоит ли использовать этот продукт в зависимости от количества объектов IT инфраструктуры? Стоимость лицензии высока ($3607 на 2 года), а управление программным обеспечением требует знаний и опыта, чтобы избежать неприятных ситуаций. Однако, существует пробная версия на 180 дней, позволяющая оценить возможности продукта.

Библиография

1. Microsoft System Center Configuration Manager. - [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Microsoft_System_Center_Configuration_Manager
2. Описание базы данных SCCM. - [Электронный ресурс]. – Режим доступа: http://corpadmin.pp.ua/blog/sccm/post/79-opisanie-bazy-dannyh-sccm
3. SCCM Different Types of Views. - [Электронный ресурс]. – Режим доступа: http://www.mssccm.com/sccm-super-flows/sccm-different-types-of-views/

РЕЛЯЦИОННАЯ АЛГЕБРА И РЕЛЯЦИОННОЕ ИСЧИСЛЕНИЕ

Алиса РЯБИНИНА

Технический Университет Молдовы, Департамент Программной Инженерии и Автоматики

Аннотация: *Статья посвящена реляционной алгебре и реляционному исчислению. Описано как они могут быть применены. Представлено сравнение между реляционной алгеброй и реляционным исчислением.*

Ключевые слова: *реляционная, алгебра, исчисление, операция, присвоение, выборка, проекция, пересечение, соединение, деление, объединение, разность, произведение, квантор всеобщности, квантор существования.*

1. Краткая характеристика понятий

Реляционная алгебра представляет собой абстрактный процедурный язык запросов для обработки реляционных таблиц (процедурный язык – язык, обеспечивающий пошаговое решение задач). Реляционная алгебра состоит из набора операций над таблицами: объединение, пересечение, разность, декартово произведение, присвоение, выборка (селекция), проекция, соединение, деление.

Реляционное исчисление – это непроцедурный язык создания запросов (непроцедурный язык – язык, позволяющий формулировать, что нужно сделать, а не как этого добиться).

В реляционном исчислении запрос состоит из целевого списка и определяющего выражения, разделённых двоеточием. Целевой список – список в выражении реляционного исчисления, определяющий атрибуты таблицы решения. Определяющее выражение – условие в выражении реляционного исчисления, ограничивающее вхождение элементов (строк) в таблицу решения.

Результат запроса реляционной алгебры и реляционного исчислении есть реляционная таблица.

Команды SQL построены в основном на реляционном исчислении, кроме случаев использования команд, реализующих непосредственно функции реляционной алгебры (реляционно-полный язык – язык, имеющий такую же логическую мощность, что и реляционная алгебра или реляционное исчисление).

2. Примеры запросов

Предположим, что необходимо найти ФИО работников, подчинённых бригадиру с ID 1520?

Пример абстрактной записи запроса в реляционном исчислении будет:

{r.ФИО: r IN РАБОТНИКИ AND r.ID_БРИГ='1520'}.

Тот же запрос в реляционной алгебра: $\pi_{ФИО}(\sigma_{ID\_БРИГ='1520'}(РАБОТНИКИ))$.

И наконец, SQL запрос: SELECT ФИО FROM РАБОТНИКИ WHERE ID_БРИГ='1520';

Данный запрос выполняется в следующем порядке: система просматривает строки таблицы РАБОТНИКИ одну за другой. Первой строке временно присваивается имя r и проверяется истинность определяющего выражения. В этом случае, поскольку определяющее выражение r.ID_БРИГ=’1520’ ложно, то строка r.ФИО: Иванов И.И. не помещается в таблицу решения. Затем система переходит к следующей строке, даёт ей имя r и снова проверяет истинность определяющего выражения. На этот раз выражение истинно, поэтому Петров П.П. помещается в таблицу решения. Затем процесс повторяется для каждой из остальных строк таблицы РАБОТНИКИ.

Целевой список может состоять из нескольких атрибутов:

{r.ФИО, r.ID_РАБ, r.ПОЧАСОВАЯ_РАСЦЕНКА, r.СПЕЦИАЛЬНОСТЬ: r IN РАБОТНИКИ AND r.ID_БРИГ='1520'}.

В выражениях реляционного исчисления применяются кванторы существования и всеобщности. Понятие квантор обозначает количество чего-либо.

Квантор существования - выражение реляционного исчисления, означающее существование хотя бы одной строки, удовлетворяющей условию.

Квантор всеобщности - выражение реляционного исчисления, обозначающее, что некоторое условие применяется к каждой строке определённого типа. Квантор всеобщности используется аналогично операции деления в реляционной алгебре для выполнения запросов со словами все и каждый.

Заключение

Выражение реляционной алгебры определяет последовательность операций над одним или несколькими отношениями и порядок их выполнения.

Реляционное исчисление представляет адаптацию исчисления предикатов применительно к области баз данных. Оно освобождает пользователя от необходимости определения того как достичь результата.

Реляционная алгебра и реляционное исчисление логически эквивалентны, то есть любой запрос выполненный в одном из этих языков, выполняется и в другом.

Таким образом, операции выборки и проецирования поддерживаются в реляционном исчислении. Объединение, пересечение, разность и декартово произведение также можно легко вывести из конструкций реляционного исчисления. Поскольку в реляционном исчислении не используются пошаговые процедуры алгебры, операция присвоения не нужна. Для выполнения некоторых операций реляционной алгебры в реляционном исчислении требуются кванторы: квантор существования (для соединения) и квантор всеобщности (для деления).

Библиография

1. Реляционная алгебра и реляционное исчисление. – [Электронный ресурс]. – Режим доступа: http://bgtu-ief.com/index.php?option=com_content&view=article&id=574:-7-&catid=44:2011-11-25-19-30-30&Itemid=64
2. Основы реляционной алгебры. – [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/145381/
3. Реляционное исчисление. – [Электронный ресурс]. – Режим доступа: http://www.mstu.edu.ru/study/materials/zelenkov/ch_4_5.html
4. И.Ф. Астахова , А.П. Толстобров, В.М. Мельников. SQL в примерах и задачах Учебное пособие, Воронеж, 2001.