электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.665

Сравнение возможностей по созданию постреляционных баз данных в средах PostgreSQL и MS SQL Server

***Медведев А.Э.**, магистр*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»

Научный руководитель: Виноградова М.В., к.т.н, доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»
vinogradova.m@bmstu.ru

Введение

Реляционные базы данных (БД) и системы управления реляционными базами данных (РСУБД) в настоящее время широко распространены, но, несмотря на это, они обладают своими плюсами и минусами. В реляционной базе данных все данные представлены в виде таблиц, а операции над ними реализованы как операции над этими таблицами [1].

Главным преимуществом реляционных БД является способность к нормализации, которая позволяет избежать избыточности данных, а также аномалий удаления и изменения записей [2].

Главным недостатком реляционных БД является ограниченность использования в областях, в которых требуются достаточно сложные структуры данных [3]. Атомарность данных, которые хранятся на пересечении строк и столбцов таблицы – один из основных принципов реляционных баз данных, который также ограничивает область применения реляционных БД. Помимо этого, особенность реализации реляционной модели не дает возможности адекватно отражать существующие связи между объектами в предметной области. Эти недостатки не позволяют эффективно реализовать современные приложения, для которых необходимы другие подходы к организации данных.

Постреляционная модель представления данных дает возможность устранить ограничение атомарности данных, что является более эффективным способом хранения данных по сравнению с реляционной моделью.

Постреляционные СУБД дают возможность хранить в полях отношений данные типов, которые определяет сам пользователь. Это позволяет хранить объекты и массивы

данных. Внедрение объектного-ориентированного подхода в традиционную реляционную модель дало повод к появлению объектно-реляционных СУБД.

В связи с тем, что постреляционная модель использует многомерные структуры хранения данных, т.е. в полях таблицы могут храниться другие таблицы, ее также называют «не первой нормальной формой» или «многомерной базой данных». В данной модели используется расширенный язык запросов SQL, который дает возможность получить сложные объекты из таблицы, не используя операции соединения. Учитывая все, что говорилось выше, можно выделить основное отличие постреляционных СУБД от реляционных:

- отказ от атомарности;
- отказ от нормализации;
- возможность создания пользовательских типов данных;
- использование расширенного языка запросов SQL.

Далее эти отличия рассмотрены более подробно.

Основные особенности постреляционных СУБД

Постреляционные СУБД обладают рядом особенностей [4, 5], которые можно разделить на несколько групп:

1) управление объектами и правилами;

2) увеличение функциональных возможностей;

3) открытость и доступность системы.

К первой группе можно отнести следующие качества:

- разнообразные конструкторы типов (массив, последовательность, запись, множество, функция, объединение);
- наследование, в том числе множественное;
- функции, процедуры и методы баз данных;
- уникальный идентификатор записей (UID);
- триггеры (правила).

Особенностями второй группы являются:

- использование языков высокого уровня;
- спецификации коллекций;
- использование представлений;
- возможность кеширования данных.

Третья группа представляет собой изменения, связанные с интерфейсом:

- использование различных API;
- использование расширенного языка запросов SQL.

Существует несколько направлений развития постреляционых БД, например, объектно-реляционные, NoSQL [6] или темпоральные [7]. Также применяют расширение возможностей реляционных СУБД посредством создания дополнительных оболочек к ним для отдельных предметных областей, например, для работы с темпоральными данными [8] или функциональными зависимостями [9].

Поскольку объектно-реляционные СУБД обладают обратной совместимостью, универсальны и широко распространены, то сфокусируем взгляд на постреляционных возможностях основных представителей этого класса.

Дополнительным преимуществом указанной категории СУБД является поддержка нормализации, которая может быть выполнена автоматически на основе алгоритма [12].

Постановка задачи исследования

Необходимо проанализировать и сравнить основные возможности и преимущества двух СУБД: MS SQL Server и PostgreSQL. Анализ провести в разрезе основных особенностей постреляционных СУБД.

PostgreSQL является свободной объектно-реляционной системой управления базами данных, которая написана на языке C и SQL. Разработчик данной системы - сообщество PostgreSQL [10].

Microsoft SQL Server представляет собой систему управления реляционными базами данных, разработчиком которой является корпорация Microsoft [11]. Основной используемый язык - язык запросов Transact-SQL, являющийся расширенным языком SQL.

Сравнительный обзор возможностей MS SQL и PostgreSQL

Наследование

Наследование позволяет создавать иерархии типов, когда производные типы расширяют или ограничивают свойства базового типа.

Множественное наследование – свойство, когда класс имеет более одного родителя. Это означает, что при создании нового класса, которому необходимы свойства от двух и более различных уже созданных классов, можно включить их свойства в создаваемый класс. Таким образом, это помогает убрать дублирование данных. Но как бы там ни было, множественное наследование имеет ряд своих недостатков.

Microsoft SQL и PostgreSQL поддерживают как одиночное, так и множественное наследование.

Пользовательские функции баз данных

В процессе разработки базы данных на языке SQL может потребоваться создание сложных алгоритмов, которые будут использоваться более одного раза. Именно для этого созданы пользовательские функции, когда пользователь сам создает функцию и в процессе разработки базы данных в любой момент может ею воспользоваться.

В MS SQL пользовательские функции представлены в виде самостоятельных объектов БД (процедуры, триггеры). Функции расположены в конкретной базе данных, и доступ осуществляется только в ее контексте.

В MS SQL Server выделены следующие типы функций, которые возвращают:

- скалярные значения;
- набор данных, представленный в виде таблицы;
- набор данных также представленный в виде таблицы, но с выполнением дополнительных команд SQL.

Пользовательские функции имеют сходства с процедурами, но также используются в запросах. Помимо этого, они могут стать заменой предствлениям, так как имеют возможность включать дополнительные выражения, которые позволяют создавать более сложные конструкции.

В PostgreSQL выделены отличающиеся от MS SQL виды функций:

- SQL-функции;
- функции на языке C;
- функции на процедурных языках.

SQL-функции состоят из одного или более SQL-запросов, при этом в выводе будет отображаться результат последнего запроса. Помимо этого, если возвращаемый значение имеет тип не void, то разрешается использование INSERT, UPDATE, или DELETE с конструкцией RETURNING.

Функции на языке C имеют два варианта: статически и динамически загружаемые. Статически загружаемые формируются на сервере при инициализации кластера базы данных, динамически загружаемые — подгружаются сервером по требованию.

Функции на процедурных языках требуют создания соответствующих расширений, причем некоторые из языков могут быть двух видов — доверенные и не доверенные. В базовой поставке PostgreSQL идут plpgsql, pltcl, plperl и plpython.

Уникальный идентификатор записей

GUID (англ. Globally Unique Identifier) – универсальный идентификатор записей. Основной особенностью является присвоение универсальных ключей, вероятность совпадения которых крайне мала. Это позволяет избежать совпадения по ключам.

В Microsoft SQL для этих целей используется тип данных uniqueidentifier, который содержит 16-байтовые двоичные значения.

Значение идентификатора создается одним из нескольких способов:

- во встроенном языке запросов Transact-SQL, пакете или сценарии с помощью использования функции NEWID;
- в коде приложения путем вызова функции API-интерфейса или метода, возвращающего идентификатор GUID.

Эти методы создают новые значения, используя при этом MAC-адрес сетевой платы и уникальное значение счетчика тактового генератора ЦП. При использовании первого способа берется MAC-адрес сетевой платы сервера, во втором случае – сетевой платы клиента.

Данный тип определяется не как константа, но есть возможность создать как константу, записав его в одном из форматов: символьном или двоичном.

В отличии от MS SQL, PostgreSQL использует тип данных uuid – 128-битный идентификатор, который использует алгоритм, позволяющий создавать максимально возможную уникальность идентификатора.

В PostgreSQL uuid обычно имеет вид последовательности шестнадцатеричных цифр в нижнем регистре, но также имеется возможность использования альтернативных форм ввода: использовать цифры в верхнем регистре, стандартный формат заключается в фигурные скобки, позволяя опускать все или некоторые переносы, добавлять переносы после любой группы из 4-х разрядов. Не смотря на все это, значения данного типа выводятся только в стандартной форме.

Триггеры

Триггеры – это инструмент SQL-сервера, который используется для поддержания целостности данных, реализации различных по сложности алгоритмов проверки данных, а также контроля изменения связанных данных при необходимости. Триггеры представляют собой фильтры, которые начинают работать после выполнения операций в соответствии с правилами, стандартными значениями и т.д.

Триггер – это процедура, которая запускается сервером автоматически при попытке изменения данных в таблицах. Каждый триггер привязывается к определенной таблице. Все изменения рассматриваются как одна транзакция, и если на каком-нибудь

этапе работы триггера возникает ошибка или нарушается целостность данных, происходит откат этой транзакции, также отменяются все предыдущие транзакции, выполненные триггером, и пользователь получает отказ в изменении данных.

С помощью триггеров достигаются следующие цели:

- проверка правильности введенных данных, а также сложных ограничений целостности;
- вывод предупреждений, которые уведомляют о необходимости выполнения определенных действий при обновлении таблицы;
- сохранение данных о внесенных изменениях и пользователе, который вносил их;
- поддержка репликации.

В MS SQL триггер создается только в текущей базе данных, но имеется возможность обращения внутри триггера к другим базам данных, в том числе и расположенным на удаленном сервере. Имя триггера должно быть уникальным в пределах базы данных. Для создания используется одна из команд CREATE TRIGGER или ALTER TRIGGER. Поддерживается возможность создания триггера, срабатывающего при изменении структуры данных.

В SQL Server существует два параметра, которые определяют поведение триггеров:

- AFTER - триггер выполняется после успешного выполнения вызвавших его команд, при неуспешном выполнении – триггер не выполняется;
- INSTEAD OF - триггер вызывается вместо выполнения команд, в отличии от первого типа, может быть определен как для таблицы, так и для просмотра.

По умолчанию все триггеры имеют параметр AFTER. Для каждой операции INSERT, UPDATE, DELETE можно определить только один INSTEAD OF-триггер.

В зависимости от команд, на которые реагируют триггеры, различают три типа триггеров:

- при попытке вставки данных с помощью команды INSERT.
- при попытке изменения данных с помощью команды UPDATE.
- при попытке удаления данных с помощью команды DELETE.

В PostgreSQL триггеры создаются на основании уже существующих функций. Сначала командой CREATE FUNCTION определяется триггерная функция, затем на ее основе командой CREATE TRIGGER определяется собственно триггер.

Компонентами описания триггера являются:

- CREATE TRIGGER «название триггера». Имя может дублироваться, но с условием: что одноименный триггер установлен для другой таблицы;
- { BEFORE | AFTER } – первый вариант означает, что функция должна вызываться перед попыткой выполнения операции, второй вариант – после ее завершения;
- далее следуют события SQL: INSERT, UPDATE или DELETE. При использовании нескольких событий они разделяются ключевым словом OR.
- ON «имя таблицы» - при изменении данных какой таблицы будет вызван триггер, если выполнятся условия срабатывания;
- FOR EACH { ROW | STATEMENT } – определяет количество вызовов функции: ROW означает, что функция будет вызываться для каждой изменяемой записи, STATEMENT – один раз для всей команды;
- EXECUTE PROCEDURE «название функции».

Представления

Представления – это виртуальные таблицы, являющиеся объектами базы данных, информация в которых формируется благодаря запросу во время обращения. Использование представлений дает возможность разработчику базы данный предоставить пользователям наиболее подходящие способы работы с данными, благодаря чему решается проблема простоты их использования и безопасности. Представление выглядит как таблица, которая создается на основании данных других таблиц в соответствии с запросом. У пользователя создается впечатление, что он работает с реально существующей таблицей.

Создание и изменение представлений в стандарте языка и реализация в MS SQL Server совпадают и представлены командами CREATE VIEW или ALTER VIEW.

Представление в PostgreSQL создается командой CREATE VIEW.

API

В многих СУБД при разработке приложений используются библиотеки функций, которые представляют собой интерфейс между прикладными программами и СУБД (англ. Application Program Interface – API). В настоящее время существует много новых API для доступа к БД. Это ODBC, JDBC, OLE DB.

OLE DB – является самым быстрым и гибким API. OLE DB можно использовать для разработки на языках программирования, поддерживающих прямую работу с COM. По существу, это ограничивает его применение языками C и C++. Можно попытаться применять OLE DB на Delphi, но надо учитывать, что файлы описания для Delphi

недоступны. Для OLE DB в библиотеках C++, таких как ATL и MFC, имеются удобные оболочки.

ODBC - является низкоуровневым API, но его можно напрямую использовать из большего количества языков программирования. Главным требованием здесь является поддержка языком динамически подключаемых библиотек (DLL).

ADO - является совместимой с OLE Automation надстройкой над OLE DB, обеспечивает самый простой способ доступа из высокоуровневых языков программирования (MS Visual Basic, MS Visual J++, Delphi) и из скриптовых языков (VBScript, JavaScript). ADO применим и в C/C++, но здесь его преимущества не так заметны по сравнению с использованием OLE DB.

MS SQL Server поддерживает способы доступа: OLE DB, TDS (Tabular Data Stream), ADO.NET, JDBC, ODBC.

PostgreSQL поддерживает способы доступа: встроенная библиотека C, потоковые API для больших объектов, ADO.NET, JDBC, ODBC.

Кэширование

Кэширование – использование нестатических данных, для обработки которых необходимо время, занесение их в память, для быстрого обращения к ним.

Варианты решения Microsoft SQL:

- Таблица для хранения этих данных, процедура обновления по расписанию, либо другие более удобные формы использования или извлечения этих данных. Плюсами является удобство и быстрота использования данных, минусами – дополнительные конструкции и проблемы при кэшировании данных;
- Таблица для хранения данных, процедура, извлекающая эти данные. Процедура проверяет наличие нужных данных в таблице, если есть – отдает, нет – обращается к сервису, получает данные, после чего записывает в таблицу и отдает. Плюсы: компактное решение, минусы - неудобное получение данных.

В PostgreSQL нет возможности чтения напрямую с диска, а также записи на диск. Данные грузятся в общий буфер сервера, который находится в разделяемой памяти, серверные процессы читают и пишут блоки в этом буфере, а после изменения сбрасываются на диск. Если процессу требуется получить доступ к таблице, то он сначала ищет необходимые блоки в общем буфере.

Если блоки присутствуют, то он может продолжать работу, если нет -- делается системный вызов для их загрузки. Загружаться блоки могут как из файлового кэша

операционной системы, так и с диска, и эта операция может оказаться весьма требовательной.

Если объёма буфера недостаточно для хранения часто используемых рабочих данных, то они будут постоянно писаться и читаться из кэша операционной системы или с диска, что сильно повлияет на производительность.

Языки программирования

Языки программирования используются для создания различных пользовательских приложений, используя базы данных.

Можно выделить следующие языки программирования, которые поддерживаются данными системами:

- MS SQL: .Net, Java, PHP, Python, Ruby, Visual Basic;
- PostgreSQL: .Net, C, C++, Java, Perl, Python, Tcl.

Репликация

Репликация представляет собой свойство передачи баз данных между различными пользователями, а также между базами данных. Репликация создана для восстановления уже существующих баз данных, а также создания копий и новых баз данных на основе уже имеющихся.

Microsoft SQL поддерживает репликацию, но вся функциональность зависима от издания SQL-Server.

В PostgreSQL репликация на уровне разработчика. Виды другой репликации доступны только с расширениями.

Кроссплатформенность

Кроссплатформенность – возможность базы данных быть установленной на различные операционные системы.

Для MS SQL единственной системой является Windows.

Для PostgreSQL: FreeBSD, HP-UX, Linux, NetBSD, OpenBSD, OS X, Solaris, Unix, Windows.

Подведение итогов

Поскольку при создании программных приложений важную роль играет правильность выбора технологии, модели и средств реализации [13], то проектировщикам и разработчикам информационных систем необходимо учитывать особенности используемых СУБД.

В помощь экспертам и консультантам [14] для повышения качества их оценок предлагается сводная таблица постреляционных возможностей сравниваемых СУБД.

Учитывая то, о чем говорилось выше можно составить итоговую таблицу.

Параметр	Microsoft SQL Server	PostgreSQL
Разработчик	Microsoft	Сообщество PostgreSQL
Язык СУБД	C++	C
Язык запросов	Transact-SQL и .NET	SQL
Стоимость	коммерческая	бесплатная
Кроссплатформенность	Windows	FreeBSD, HP-UX, Linux, NetBSD, OpenBSD, OS X, Solaris, Unix, Windows.
Наследование	+	+
API	OLE DB, TDS (Tabular Data Stream), ADO.NET, JDBC, ODBC	ADO.NET, JDBC, ODBC
Поддерживаемые языки программирования	.Net, Java, PHP, Python, Ruby, Visual Basic	.Net, C, C++, Java, Perl, Python, Tcl
Триггеры	+	+
Кэширование	+	+
Представления	+	+

Заключение

Можно сделать вывод, что каждая из рассмотренных систем обладает рядом достоинств и недостатков.

MS SQL Server – мощная система управления базами данных, которая обладает широким функционалом, графическим интерфейсом, но поддерживается только на операционной системе Windows.

PostgreSQL обладает полностью текстовым интерфейсом, также она имеет широкий функционал и может использоваться на различных платформах. Плюсом также является свободное распространение данной СУБД.

Подводя итоги, можно сказать, что MS SQL Server позволяет быстро создавать типовые приложения как для персональных, так и для крупных баз данных масштаба

предприятия. PostgreSQL рекомендуется использовать для программирования на стороне сервера сложных систем, например учета, онлайн проектов или персональных баз данных.

Список литературы

1. Ревунков Г.И. Проектирование баз данных: учебное пособие по курсу «Банки данных». МГТУ им. Н. Э. Баумана. М.: Изд-во МГТУ им. Н. Э. Баумана, 2009.
2. Виноградова М.В., Игушев Э.Г. Конструктор баз данных на основе сущностей и их реквизитов с возможностью нормализации // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2012. № 1. Режим доступа: http://technomag.edu.ru/doc/242645.html (дата обращения 20.03.2015). DOI: 77-30569/242645.
3. Черненький В. М., Ерцев П. Г. Объектно-ориентированное описание предметной области при моделировании процессов ликвидации последствий аварийных разливов нефти // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2011. № 11. Режим доступа: http://technomag.bmstu.ru/doc/292475.html (дата обращения 20.01.2015). DOI: 77-30569/292475.
4. Когаловский М. Р. Системы баз данных третьего поколения: Манифест // Системы Управления Базами Данных. № 2. 1995. Режим доступа: http://citforum.oldbank.com/database/classics/manifest/ (дата обращения 24.03.2015).
5. Аткинсон М., Бансилон Ф. Манифест систем объектно-ориентированных баз данных. Свойства ООБД. Режим доступа: http://www.osp.ru/dbms/1995/04/13031448 (дата обращения 25.03.2015).
6. Григорьев Ю.А. Анализ свойств баз данных NoSQL // Информатика и системы управления. 2013. № 2 (36). С. 3 -13.
7. Тоноян С.А., Сараев Д.В. Темпоральные модели базы данных и их свойства // Инженерный журнал: наука и инновации. 2014. № 12 (36). С. 15. Режим доступа: http://engjournal.ru/catalog/it/hidden/1333.html (дата обращения 27.03.2015).
8. Балдин А.В., Елисеев Д.В., Агаян К.Г. Обзор способов построения темпоральных систем на основе реляционной базы данных // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2012. № 8. Режим доступа: http://technomag.bmstu.ru/doc/441884.html (дата обращения 20.01.2015). DOI: 10.7463/0812.0441884.

9. Виноградова М. В. Методика создания мультиаспектной информационной системы с алгоритмоориентированной структурой данных: автореф. дис... канд. техн. наук. М., 2005. 16 с.
10. Документация MS SQL. Режим доступа: https://msdn.microsoft.com/ru-ru/library/ms130214(v=sql.100).aspx (дата обращения 26.03.2015).
11. Документация PostgreSQL. Режим доступа: http://postgresql.ru.net/docs.html (дата обращения 26.03.2015).
12. Григорьев Ю.А. Алгоритм синтеза частично оптимальной схемы реляционной базы данных // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2012. № 1. Режим доступа: http://technomag.bmstu.ru/doc/441884.html (дата обращения 20.01.2015). DOI: 77-30569/294486.
13. Яроц Е.В., Гапанюк Ю.Е. Система управления знаниями - путь к успеху организации // Славянский форум (Святой Влас, Болгария, 14-23 мая 2013 г.): тез. докл. Святой Влас, 2013. С. 291-297.
14. Постников В.М. Анализ подходов к формированию состава экспертной группы, ориентированной на подготовку и принятие решений // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2012. № 5. Режим доступа: http://technomag.bmstu.ru/doc/441884.html (дата обращения 20.01.2015). DOI: 10.7463/0512.0360720.