торы считаются ведомыми - MRC (Media Redundancy Client). MRM коммутатор проверяет целостность кольца, посылая специальные кадры в одном направлении и принимая их с другого. В нормальном режиме - "замкнутое кольцо" (Ring-Closed) все кадры с противоположного направления отбрасываются MRM. При выходе из строя линии или коммутатора в кольце, происходит переключение в так режим "разомкнутого кольца" (Ring-Open), когда MRM начинает отправлять кадры в оба направления. Преимуществом кольцевого резервирования является стабильно низкое время переключения при отказе - до 200 мс. Однако для многих приложений может быть неприемлемым как кольцевая топология сети, так и даже настолько малое время восстановления.

Для параллельного резервирования между любыми двумя узлами сети создается два маршрута, идентичные кадры посылаются одновременно по каждому из маршрутов. В случае отказа одного маршрута кадры продолжают поступать через параллельный, вследствие этого отказ практически никак не сказывается на работе конечных приложений. Таким образом конечный узел в нормальном режиме всегда должен получать два одинаковых кадра, получение одного кадра сигнализирует об отказе одного из маршрутов. Для реализации параллельного резервирования разработан протокол PRP (Parallel Redundancy Protocol), стандартизированный как IEC 62439-3. Протокол PRP для работы фактически требует наличия двух параллельных Ethernet-сетей, поэтому для реализации общей сети, как правило, используется топология двойной звезды. В каждой отдельной сети может применяться дополнительное резервирование на основе протокола STP. Необходимо тщательное проектировании сети с резервированием PRP, так как любая связь между параллельными сетями может привести к мгновенному отказу обоих сетей вследствие образования петлевых маршрутов. К преимуществам PRP необходимо отнести отсутствие какой-либо специально поддержки протокола на промежуточных коммутаторах, что облегчает реализацию сетей с PRP. Поддержка PRP необходима только на конечных узлах, которые называют DANP (Double Attached Node for PRP), они должны иметь два сетевых подключения к каждой отдельной сети и проводить проверку дублирования поступающих кадров. Преимуществами протокола PRP являются6 нулевое время восстановления работоспособности при отказах, отсутствие каких-либо специальных требований к топологии. Главный недостаток состоит в необходимости удвоения ресурсов сети, что значительно увеличивает стоимость реализации.

Для кольцевой топологии существует адаптация протокола PRP - HSR (High -availability Seamless Redundancy). Как и в PRP, в HSR наличествует два параллельных маршрута, но реализованных в кольцевой топологии. По каждому кольцевому маршруту параллельно передаются идентичные кадры. Каждый HSR-коммутатор отбирает из кадры предназначенные для подключенных к нему узлов. Для отбрасывания дублированных кадров используется специальный заголовок, что требует специальной поддержки функций HSR на коммутаторах. К преимуществам HSR можно отнести более высокую скорость обработки кадров в сравнении с PRP. Недостатки HSR аналогичны недостаткам PRP, к которым добавляется необходимость поддержки функций HSR на коммутаторах.

На основе проведенного анализа можно сделать выводы о том, что пока существующие технологии традиционного, кольцевого и параллельного резервирования в сетях Ethernet далеки от идеала и не обладают достаточной гибкостью в сочетании с низкими затратами на реализацию. Необходимы дальнейшие исследования по их совершенствованию и поиску альтернативных решений, возможно совмещающих резервирование на втором и третьем уровнях модели OSI.

Литература

1. Стандарт IEEE 802.1D [Электронный ресурс]. Режим доступа: http://standards.ieee.org/getieee802/download/802.1D-2004.pdf

2. Стандарт IEC 62439-2:2010 / Industrial communication networks - High availability automation networks - Part 2: Media Redundancy Protocol (MRP) [Электронный ресурс]. Режим доступа: https://webstore.iec.ch/publication/7017

3. Стандарт IEC 62439-3:2010 / Industrial communication networks - High availability automation networks - Part 3: Parallel Redundancy Protocol (PRP) and High-availability Seamless Redundancy (HSR) [Электронный ресурс]. Режим доступа: https://webstore.iec.ch/publication/20490

М.В. Терёхин, Е.И. Чигарина

КЛАССИФИКАЦИЯ И АНАЛИЗ ИСПОЛЬЗОВАНИЯ ОКОННЫХ ФУНКЦИЙ В СИСТЕМАХ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ

(Самарский национальный исследовательский университет
имени академика С.П. Королёва)

В системах баз данных одной из основных задач является задача сокращения времени выполнения запросов при работе с данными с использованием минимального объема памяти, то есть задача эффективного выполнения запросов.

В настоящее время наибольшее распространение получили реляционные базы данных, для работы с которыми используется язык SQL (Structured Query Language – язык структурированных запросов) [1]. В языке SQL постоянно совершенствуются команды запросов для более эффективного их выполнения. В частности, в стандарте языка SQL для реализации аналитических задач стали использоваться многочисленные функции, оперирующие окнами.

В настоящее время не существует четкой классификации оконных функций в зависимости от их назначения и использования, а также не проводился анализ эффективности выполнения запросов с использованием оконных функций по сравнению с традиционными запросами без них. В реляционных базах данных традиционно используются скалярные и агрегатные функции. Агрегат-

ные функции выполняют вычисления над группой значений столбца и всегда возвращают одно значение результата этих вычислений. Скалярные функции – это функции, которые возвращают одно значение, работая с одним значением или вообще без входных данных. В отличие от этих функций оконные функции берут в качестве аргумента таблицу, представляющую логический промежуточный результат обработки SQL-оператора, где использовано обращение к такой функции, и возвращают в качестве своего результата обычно тоже таблицу [2]. Другими словами, одно из главных отличий оконных функций от обычных, скалярных, заключается в том, что этот класс операторов работает с готовой выборкой. Именно поэтому эти функции указываются в списке выборки или в условии сортировки [3].

Цель введения оконных функций – дать лаконичную формулировку и увеличить скорость выполнения запросов к БД, имеющих смыслом выявление внутренних соотношений и зависимостей между данными.

Оконные функции принимают в качестве аргумента столбец промежуточного результата вычисления SQL-предложения и возвращают тоже столбец. Поэтому местом их использования в SQL-предложении могут быть только фразы ORDER BY и SELECT, выполняющие завершающую обработку логического промежуточного результата. В результате анализа применения оконных функций выделены их варианты использования в сочетании с функциями ранжирования, функциями подсчета долей, оконными функциями общего назначения, агрегатными функциями.

Результат классификации оконных функций приведен на рисунке 1.

Функции ранжирования позволяют «раздать» строкам «места» в зависимости от имеющихся в них значениях. Эти функции возвращают ранг каждой записи внутри «окна». В общем случае рангом является некое число отражающее положение или «вес» записи относительно других записей в том же наборе. Формируется «окно» с помощью группировки. Однако, поскольку результат работы функций ранжирования зависит от порядка обработки записей, то обязательно должен быть указан порядок записей внутри «окна» посредством конструкции ORDER BY. В зависимости от используемой функции некоторые записи могут получать один и тот же ранг. Функции ранжирования являются недетерминированными, то есть при одних и тех же входных значениях они могут возвращать разный результат [3].

Функции подсчета долей позволяют одной SQL-операцией получить для каждой строки ее «вес» в таблице в соответствии со значениями.

Функции LAG и LEAD обращаются к данным из предыдущей и из последующей (соответственно) строки того же результирующего набора без использования самосоединения. Функция LAG обеспечивает доступ к строке с заданным физическим смещением перед началом текущей строки, а функция LEAD – после текущей строки [4].

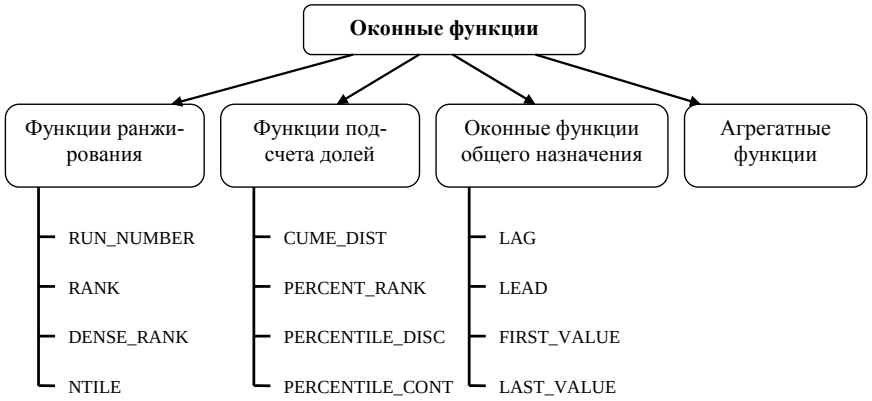

Рисунок 1 – Классификация оконных функций

Кроме того, к функциям общего назначения можно отнести функции FIRST_VALUE и LAST_VALUE, которые возвращают первое или последнее значение из упорядоченного набора значений соответственно.

Поскольку аналитическая функция возвращает агрегированный результат обработки записей, попавших в «окно», то самые обычные агрегатные функции имеют возможность выступить в качестве оконных. Отличие состоит в том, что «обычные» агрегаты уменьшают степень детализации результирующего набора, а в аналитическом варианте степень детализации не уменьшается [5].

Агрегатные функции работают как оконные, только когда за их вызовом следует предложение OVER, в противном случае они останутся обычными агрегатными.

Для выполнения анализа использования оконных функций в запросах определен перечень параметров, влияющих на их использование. К таким параметрам относятся:

- время выполнения запроса;
- количество логических и физических чтений при обращении к данным;
- структура данных, включая различные виды соединений реляционных множеств при организации запросов, а также особенности взаимосвязей между таблицами.

В качестве альтернативы использования оконных функций могут выступать запросы, использующие самосоединение, а также хранимые процедуры и функции, реализующие действия оконных функций.

Для оценки эффективности работы функций, необходимо использовать средства, которые позволяют определить план и время выполнения запроса. В качестве СУБД для анализа работы функций использованы СУБД Oracle, MS SQL Server и PostgreSQL, имеющие необходимые инструменты для решения

этой задачи. В СУБД Oracle для определения плана и времени выполнения запроса используется специальная системная таблица PLAN_TABLE, а также средство трассировки, которое выдает информацию о производительности по индивидуальным предложениям SQL. В SQL Server Management Studio имеется встроенный планировщик запросов и инструмент «статистика клиента», с помощью которого можно просмотреть сведения об операции: временные характеристики, а также объемы отправленных и полученных пакетов данных выполненного запроса. В PostgreSQL также есть специальная подсистема – сборщик статистики – которая в реальном времени собирает данные об активности сервера. Данные, полученные сборщиком статистики, доступны через специальные системные представления.

Выполняется анализ по использованию оконных функций или традиционных способов по работе с объектами базы данных с учетом перечисленных параметров для выбранных систем управления базами данных.

Литература

1. Особенности реляционных баз данных [Электронный ресурс]. – http://ldalab.ru/post/pervichnij_kluch_bd.html

2. Аналитические функции в Oracle [Электронный ресурс]. – http://www.interface.ru/home.asp?artId=1774#04

3. MS SQL 2005: оконные функции. Еще одно расширение T-SQL [Электронный ресурс]. – http://rsdn.ru/?article/db/WindowFunctions.xml

4. Функции аналитики (Transact-SQL). [Электронный справочник Microsoft]. – https://msdn.microsoft.com/ru-ru/library/hh213234(v=sql.120).aspx

5. Оконные функции [Электронная документация PostgreSQL]. – http://postgrespro.ru/doc/functions-window.html

Д.А. Царёв, Ю.С. Артамонов

СРАВНЕНИЕ ОСНОВНЫХ ВОЗМОЖНОСТЕЙ И КЛАССИФИКАЦИЯ ОБЛАЧНЫХ ИНСТРУМЕНТОВ РАЗРАБОТКИ

(Самарский национальный исследовательский университет имени академика С.П. Королёва имени академика С.П. Королёва (национальный исследовательский университет))

Современная разработка программного обеспечения требует гибких инструментов и тесного взаимодействия между разработчиками [1] для достижения наилучшего результата и увеличения шансов проекта на успех. Эти требования поспособствовали появлению облачных сред разработки (Cloud IDE), основной упор в которых сделан на работе из браузера без необходимости установки средств разработки на машине пользователя [2]. Такие инструменты позволяют создавать ПО вне зависимости от технических характеристик машины разработчика, от неё требуется лишь наличие браузера и доступ к интернету.

Основные классы облачных средств разработки

К облачным инструментам разработки можно отнести облачные редакторы для прототипирования – “песочницы” (Sandbox) и полноценные рабочие места для разработки приложений – облачные IDE. Основное отличие редакторов для прототипирования от IDE заключается в том, что редактор для прототипирования в первую очередь нацелен на быструю апробацию примеров и простых набросков программных решений. Кроме того, IDE включают поддержку конкретных библиотек и дополнительных инструментов, таких как системы контроля версий [3].

Наиболее характерными представителями семейства облачных сред разработки являются CEclipse, Koding.com, Eclipse Che, которые относятся к классу Cloud IDE, а также Cpp.sh, JsFiddle и Orion, представляющие класс редакторов для прототипирования.

CEclipse позволяет разрабатывать приложения на Java в облаке, используя не только классическую функциональность Eclipse, такую как подсветка и анализ кода, выявление ошибок и автодополнение, но и дополнительные возможности по интеграции с системами управления задачами и онлайн тестирования [4].

Koding.com предоставляет разработчику выделенную виртуальную машину и веб-интерфейс для работы над приложения. Если необходимо из облака перейти в локальное рабочее окружение, то можно установить клиент Koding и начать использовать привычную IDE, редактор или терминал.

Eclipse Che – облачная IDE, которую может развернуть у себя любой желающий. Поддерживает самые популярные языки программирования: Java, C++, JS, Python, PHP и Ruby. Изолированное окружение для работы выделяется при помощи Docker контейнеризации [5].

Cpp.sh – легковесный редактор кода на C++. Компилирует и исполняет код на сервере при помощи компилятора GCC. Работать можно только с одним файлом, а сам редактор призван облегчить изучение языка C++.

jsFiddle.net – проект, с помощью который позволяет быстро создать тестовую веб-страницу для проверки работы HTML, CSS и JavaScript. Сервис имеет функцию совместной работы над исходным кодом и позволяет поделиться своей “песочницей”.

Orion – редактор для веб-приложений на HTML, CSS и JavaScript. Позволяет организовать совместную работу с ограничением на одновременное редактирование одного файла. Поддерживает систему контроля версий Git.

Помимо редакторов и облачных сред разработки можно выделить инструменты для апробации новых языков программирования, таких как Scala и Kotlin. Их облачные редакторы компилируют код в JavaScript и исполняют его в браузере пользователя, возможностей JavaScript хватает для того, чтобы продемонстрировать возможности языка и его основных концепций. А сам транспайлер Scala в JavaScript является отдельным проектом, призванным обеспечить интероперабельность Scala и JavaScript [6].