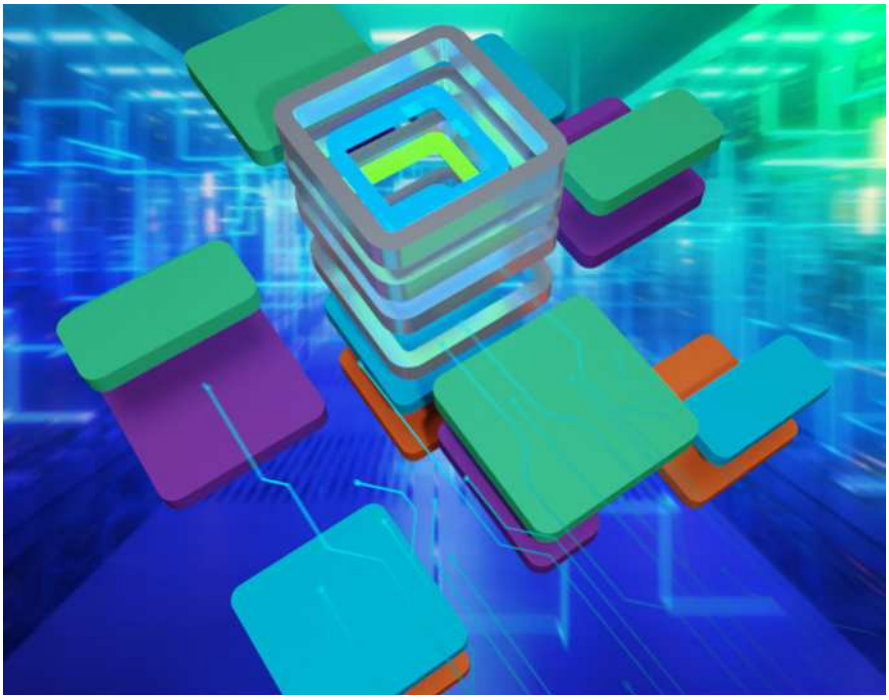
СУБД для высоконагруженных систем

В условиях, когда большинство западных производителей СУБД покинули российский рынок, перед властями и заказчиками в полный рост встал вопрос импортозамещения. СУБД Postgres Pro — один из вариантов перехода, миграции данных и приложений.

Ключевое слово: технологическая независимость
Keywords: PostgreSQL, Open Source, Technological independence



Марк Ривкин

Выбор российской СУБД, способной адекватно заменить западное решение в высоконагруженной, критичной бизнес-системе, весьма неоднозначен — требуется реляционная СУБД, реализующая большинство привычных пользователям объектов базы данных, обеспечивающая приемлемый уровень надежности, безопасности, масштабируемости, производительности, управляемости и т. д. Желательно, чтобы такая СУБД присутствовала в Реестре отечественного программного обеспечения и имела сертификат ФСТЭК. Несмотря на то, что в этом реестре можно найти много систем, помеченных тэгами «СУБД», «БД», «базы данных» и пр., большинство из них нельзя в полной мере отнести к СУБД. Основная часть подобных продуктов имеет ограниченную функциональность или опирается на узкоспециализированную модель данных, а сертификатом ФСТЭК обладает лишь СУБД «Линтер» [1], функциональности которой для применения в коммерческих системах недостаточно; несколько форков СУБД PostgreSQL, включая СУБД Postgres Pro [2]; форк FireBird/Interbase — «Ред База Данных» [3].

Наиболее вероятным и часто используемым кандидатом на замену высоконагруженных западных систем является отечественная СУБД Postgres Pro Enterprise, построенная на основе СУБД с открытым кодом PostgreSQL. Важно не путать эти две СУБД: PostgreSQL — реляционная СУБД с открытым кодом для работы в небольших и не критичных для бизнеса решениях, а для работы в высоконагруженных корпоративных системах, предъявляющих высокие требования к надежности и безопасности, предназначена отечественная коммерческая СУБД Postgres Pro Enterprise. Последняя имеет сертификат ФСТЭК по уровню доверия 4 (защита конфиденциальной информации), зарегистрирована в Реестре и при этом не отрывается от своего материнского продукта — PostgreSQL: ежегодно выходит новая версия PostgreSQL (сейчас это версия 15) — после чего сливаются старые и новые функции Postgres Pro Enterprise с кодом PostgreSQL. Таким образом, новая версия Postgres Pro Enterprise содержит все функции свежей версии PostgreSQL и ряд новых возможностей.

СУБД Postgres Pro Enterprise — один из немногих кандидатов на замену таких систем, как Microsoft SQL Server, Oracle, DB2 и MySQL. Она поддерживает привычную для пользователей этих западных СУБД функциональность, концепции и объекты реляционной базы данных (таблицы, секции, индексы, материализованные представления, последовательности, функции, процедуры, триггеры, роли, привилегии, ограничения целостности, LOB, оконные функции и т. д.). Например, в Postgres Pro Enterprise версии 15 реализована концепция пакетов (packages), хорошо знакомая пользователям Oracle; поддерживается работа с транзакциями (ACID), многоверсионность записей и стандарты языка SQL. Процедурный язык Postgres — PL/pgSQL очень похож на процедурные языки Oracle, Microsoft SQL Server и DB2. Все это облегчает миграцию на Postgres Pro Enterprise и снижает затраты на переучивание администраторов баз данных и разработчиков приложений.

Отличительная особенность СУБД PostgreSQL и ее форков — возможность добавления расширений (extention) для минимизации изменений в ядре системы. Этот механизм широко используется в Postgres Pro Enterprise, позволяя синхронизировать наработки компании Postgres Professional с новой версией PostgreSQL.

к содержанию

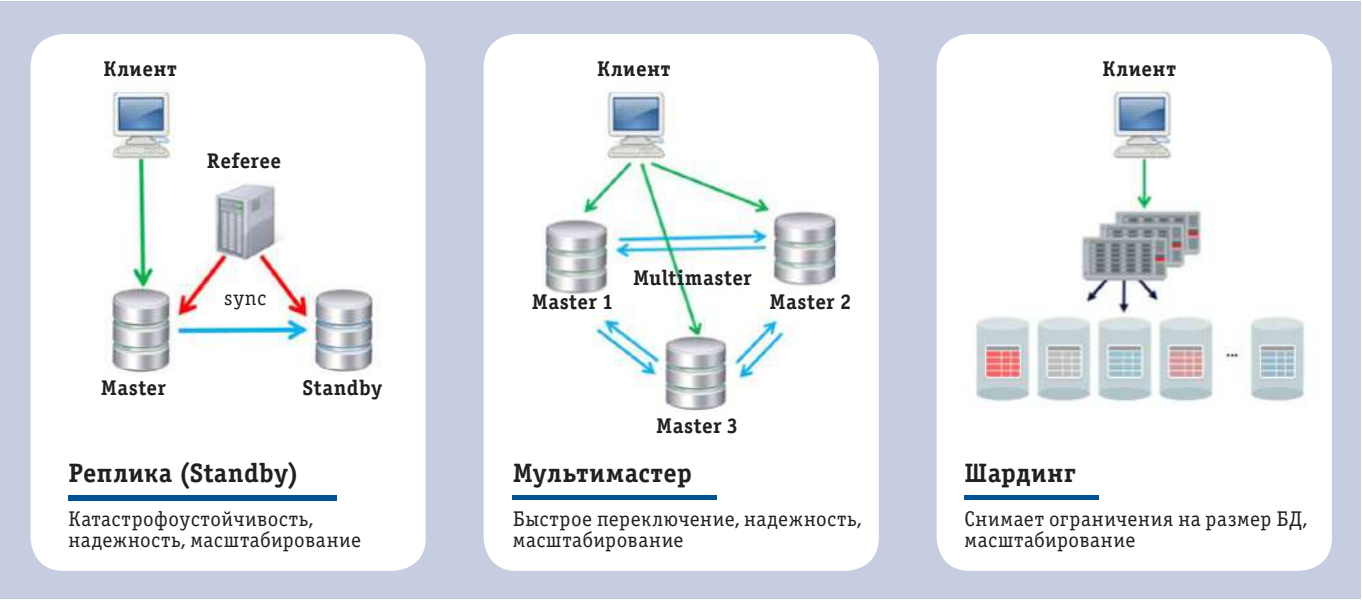


Рис. 1. Механизмы обеспечения архитектуры высокой надежности

Как и любая коммерческая СУБД, Postgres Pro Enterprise должна обеспечивать ряд возможностей, необходимых для промышленных систем:

- многоплатформность, включая и возможность работы в облачной инфраструктуре (DBaaS);
- поддержка множества различных типов данных (структурированных и неструктурированных;
- высокая надежность, доступность 24×7 и управляемость;
- поддержка многосерверных архитектур;
- безопасность и защита данных;
- масштабируемость;
- высокая производительность.

Postgres Pro Enterprise поддерживает более 30 платформ: семейство ОС Linux, включая и отечественные клоны Astra, Alt, Rosa, Red OS; ОС «Эльбрус»; ARM, RISC и мейнфреймы. Кроме этого, версия Postgres Pro Enterprise 14 работает и еще четыре года будет поддерживаться на платформе Windows. В режиме DBaaS возможна также работа в популярных облаках, например в Amazon. Сейчас разрабатывается собственная платформа DbaaS — выполнив, например, на ноутбуке отладку приложения, его вместе с базой можно перенести на более мощные конфигурации.

Postgres Pro Enterprise работает с различными типами структурированных и неструктурированных данных. Механизм расширений позволяет также создавать свои типы, описывать механизмы их хранения, индексирования, обработки и т. д. Среди структурированных данных, например, поддерживается множество числовых, денежные, символьные, двоичные, дата/время, логические, перечисления, сетевые адреса, битовые строки, UUID и т. д. В СУБД можно хранить видео, аудио, изображения и другие неструктурированные данные, причем много расширений было сделано для работы с геоинформацией, текстами и XML. Например, GIN- и RUM-индексы (generalized inverted index) значительно ускоряют работу с текстами. Много внимания сейчас уделяется работе с типами данных JSON и JSONB, для которых создаются специальные индексы, форматы хранения, а в версии Postgres Pro Enterprise 15 реализована поддержка стандарта SQL/JSON.

НАДЕЖНОСТЬ И ДОСТУПНОСТЬ

Надежность — один из главных критериев выбора СУБД для коммерческих применений. СУБД PostgreSQL — изначально достаточно надежная система, а в Postgres Pro Enterprise предусмотрено три механизма обеспечения надежности (рис. 1): мастер-реплика (standby); Мультимастер; шардинг. Кроме того, что подобные многосерверные архитектуры обеспечивают надежность, они упрощают масштабируемость системы и повышение производительности системы.

Архитектура мастер-реплика хорошо знакома пользователям СУБД Oracle, SQL Server и DB2 — имеется основная база (мастер) и ее копии, на которые синхронно или асинхронно передаются изменения основной базы. Эти изменения «накатываются» на реплику (резервную копию), и она постоянно «догоняет» основную базу данных, в случае сбоя или недоступности мастера пользователи вручную или автоматически переключаются на резервную базу и продолжают работу. В случае синхронной репликации допустимая потеря данных (recovery point objective, RPO) будет нулевой, однако работа замедлится, а при большом удалении реплики от мастера (она может быть, например, расположена в другом городе) задержка может быть ощутимой. Часто делают две реплики — первая (синхронная) располагается в том же ЦОДе, что и мастер, и страхует от поломки мастера, а вторая (асинхронная) располагается в удаленном ЦОДе и служит для защиты от катастроф (disaster recovery). Для контроля состояния узлов в такой архитектуре и автоматического переключения приложения на резервный узел при сбое или недоступности узла используются различные решения с открытым кодом: Patrony, Stolon, Corosync, а скоро в Postgres Pro Enterprise появится собственный встроенный отказоустойчивый кластер.

Реплики во всех СУБД, построенных на основе PostgreSQL, всегда открыты на чтение, поэтому часть рабочей нагрузки (создание резервной копии, построение отчетов, аналитика) можно выносить на реплики. В случае порчи блоков базы данных или журнала мастер-узла в Postgres Pro Enterprise, реплика позволяет их автоматически починить. Время простоя при переключении приложения на реплику обычно не превышает пяти секунд.

Кроме быстрой физической репликации, между мастером и репликой используется и логическая репликация, которая может быть задана для всей базы данных, для схемы, для отдельных таблиц или их частей. Изменения преобразуются в команды SQL, которые применяются на

к содержанию

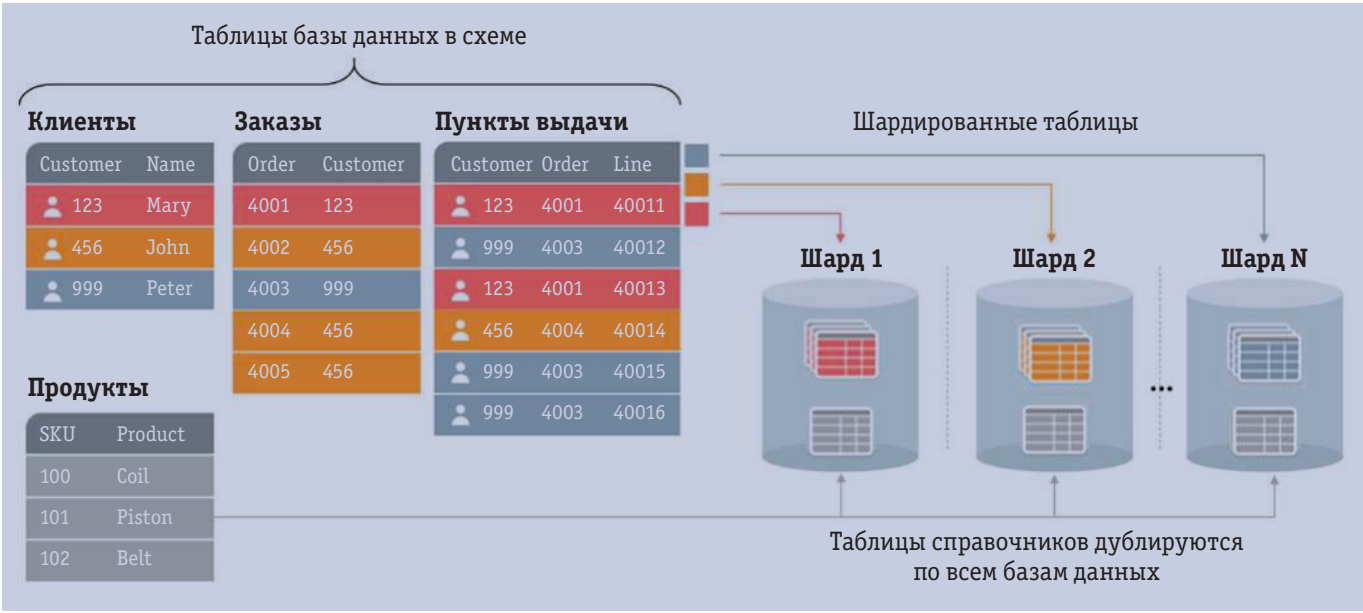


Рис. 2. Пример шардирования семейства таблиц базы данных

реплике. При этом она может быть более старой версии, чем мастер.

Архитектура мастер-реплика получила широкое распространение, однако она ограничивает возможности масштабирования — при физической репликации реплика открыта только на чтение, а логическая репликация медленная и имеет много ограничений — отсутствие двунаправленной репликации, репликации DDL, последовательностей, LOB, view, materialized view и т. д. Этих недостатков лишена архитектура Мультимастер, в которой реализована концепция active-active. Конфигурация состоит из набора одинаковых узлов с Postgres Pro Enterprise, причем все базы на всех узлах открыты на чтение и на запись — пользователь может подключиться к любому узлу и полноценно работать, а все изменения сразу отображаются в базах данных остальных узлов. Для этого используется специальный алгоритм трехфазной фиксации — транзакция завершится только тогда, когда ее изменения появятся во всех узлах Мультимастера. Конечно, это замедляет работу приложений, но в большинстве случаев такое масштабирование полезно. Например, если надо не только читать данные, но и писать небольшие объемы данных, не конфликтующие с изменениями других узлов. Часто, прежде чем построить отчет (чтение), надо записать в таблицы параметры и граничные условия отчета (запись).

Поскольку Мультимастер позволяет одновременно менять одни и те же данные в разных узлах, то может возникнуть конфликт обновлений или проблема взаимоблокировки транзакций. Несмотря на то, что существуют механизмы выявления и разрешения таких ситуаций, желательно так проектировать приложения, чтобы они минимизировали возможность возникновения конфликтов (например, каждый узел меняет только свое подмножество данных), и тогда снижение производительности будет минимальным.

Обычно число узлов Мультимастера не превышает 3–5, а если приложению требуется больше, то надо тестировать конфигурацию под нагрузкой и разносить ее. Поскольку все узлы Мультимастера одинаковы и «разогреты», то переключение на другой узел при сбое происходит быстро, а при выходе из строя одного из узлов Мультимастер сам обрабатывает ситуацию, изолируя «погибший» или недоступный узел. После ремонта или замены узел подключается к работе и сам догоняет остальные узлы.

Дальнейшее масштабирование и повышение производительности при работе с очень большими базами данных обеспечивает шардирование, а именно — специальная редакция Postgres Pro Enterprise — Шардман. Шардирование является отличным решением, если есть возможность разделить данные (связанные таблицы) на части по ключу шардирования (рис. 2). Например, у трех таблиц: Заказчики (Customers), Заказы (Orders) и Пункты заказов (LineItems) есть общий ключ (или часть ключа) — Customer и можно разделить данные этих связанных таблиц на части, соответствующие значению поля Заказчик (это и будет ключ шардирования). Все записи трех таблиц, относящиеся к заказчику 123 (помечены красным цветом), попадут в шард 123, а все записи трех таблиц, относящиеся к заказчику 999 (серый), попадут в шард 999. Каждый шард хранится в отдельной базе данных, открытой и на чтение, и на запись. Иначе говоря, в этом примере основная база была «размазана» по трем узлам и когда приходит запрос с ключом шарда 123, он будет перенаправлен на узел, где хранится шард 123, там выполнится, а результат вернется заказчику. Это обеспечивает быструю параллельную работу с несколькими шардами. Если запрос затрагивает несколько шардов (кроссшардовый запрос), то результаты подзапросов с шардов вернутся на узел-координатор (куда подключена сессия) и там сформируется результат запроса. Если в системе имеются таблицы-справочники, которые нужны для выполнения частей запроса на разных шардах, они просто дублируются на этих шардах.

Шардман обеспечивает синхронное резервное копирование/восстановление шардов, перемещение подзапросов на шарды и сбор результата; согласованность справочников; решардинг (перемещение частей шардов (микрошардов) на другие узлы при их разрастании); надежность работы системы при выходе шарда из строя. Для каждого шарда создается реплика (как в архитектуре мастер-реплика).

Реализация шардирования в Шардмане отличается от традиционной, принятой в Oracle, Microsoft SQL Server и Google, — каждый узел Шардмана видит шардированную таблицу целиком, хотя некоторые

к содержанию

ее части физически размещаются в других шардах, поэтому пользователь может обращаться к любому узлу Шардмана, который сам определит, на каких узлах будут обрабатываться подзапросы.

Если приложение хорошо поддается шардированию (можно выделить ключ шардирования), то Шардман обеспечивает высокую масштабируемость, производительность и надежность, поскольку даже при выходе из строя базы данных шарда и базы реплики шарда запросы, не затрагивающие этот шард, продолжат работу.

Кроме архитектуры максимальной доступности требуется обеспечить надежность работы и минимизацию простоев отдельного узла СУБД Postgres Pro Enterprise. Для этого многие операции администрирования позволено выполнять без остановки СУБД: перестраивать индексы, менять структуру таблиц, перемещать таблицы в другое табличное пространство, работать с партициями, устанавливать обновления. Для защиты от сбоя базы конкретного узла обычно используется резервное копирование. В Postgres Pro Enterprise для этого можно как создавать дамп-файл с операторами SQL для воссоздания объектов базы и данных, так и утилиту pg_probackup физического резервного копирования/восстановления, позволяющую быстро создавать полную и инкрементальную резервную копию, объединять инкрементальные бэкапы с полным (быстро получать новый полный бэкап, не отвлекая СУБД от выполнения продуктивной работы), выполнять резервное копирование/восстановление параллельно (в несколько потоков). Бэкап можно сжимать. Восстановление базы данных возможно пробное и на конкретный момент времени. Возможно управление политиками хранения резервных копий и удаления ненужных файлов. В версии 15 возможно резервное копирование не только на диски, но и в облачные системы объектного хранения формата S3.

БЕЗОПАСНОСТЬ И ЗАЩИТА ДАННЫХ

СУБД Postgres Pro Enterprise имеет сертификат ФСТЭК 4-го уровня доверия, что выше, чем было у Oracle и Microsoft SQL Server, причем сертификаты получены не только для мажорных (выходят раз в год и содержат много новых возможностей, причем иногда изменения затрагивают структуру словаря данных), но и для минорных версий (выходят раз в квартал и в основном содержат исправления и улучшение некоторых алгоритмов). Постоянно осуществляются контроль безопасности исходного кода, статистический анализ (coverity scan static analysis) и функциональное тестирование кода. Планируется добавить в Postgres Pro Enterprise режим прозрачного шифрования (Transparent Data Encription), после чего продукт будет соответствовать всем требованиям стандарта PCI DSS.

Наличие различных средств авторизации и аутентификации позволяет выбрать приемлемый вариант, а кроме традиционной защиты идентификации и авторизации средствами ОС или СУБД можно также использовать внешние системы: Kerberos, LDAP, Active Directory, Radius, PAM и peer/ident. Допускается два алгоритма защиты пароля — md5 и scram-sha-256. Передача данных по сети шифруется с помощью SSL. Что касается шифрования данных в таблицах, то с учетом требований российского законодательства используется решение Аладдин «Крипто БД» — сертифицированная по классам КС1, КС2 и КС3 система предотвращения утечки информации из СУБД, использующая российские алгоритмы шифрования.

В случае применения парольной защиты пользователь может управлять паролями: задавать их сложность, запрещать быстрое повторное использование, устанавливать срок действия и предупреждения о необходимости смены пароля и пр. Для управления правами доступа используется стандартный механизм привилегий и ролей, соответствующий стандарту SQL. Привилегии могут даваться на объекты, операции или работу с системой. Процедурный код и представления могут выполняться как с привилегией создателя, так и с привилегией текущего пользователя. В СУБД на основе PostgreSQL роли — это глобальное понятие, и они относятся не к отдельной базе данных, а к ее кластеру.

Механизм защиты на уровне строк (Row Level Security, RLS) позволяет разным пользователям ограничить доступ к различным наборам строк — при выполнении одного и того же запроса к таблице срабатывает процедура политики безопасности для данной таблицы, которая на ходу модифицирует запрос. В результате разные пользователи видят разный набор строк таблицы.

Еще одним важным аспектом обеспечения безопасности является аудит действий пользователей. В Postgres Pro Enterprise реализованы расширенные функции аудита выполнения команд языка SQL: DDL и DML, Select, Grant, Execute, connect/disconnect. Аудит позволяет, например, выявить пользователя, изменившего данные, или пользователя, подбирающего пароль к системе, и т. д. Администратор безопасности может включать аудит для отдельных пользователей или отдельных объектов, причем расширение pg_proaudit позволяет «на лету» менять настройки аудита. Результаты аудита записываются в журналы СУБД, что, однако, может затормозить работу, поэтому обычно используется гранулярный аудит наиболее критичных пользователей или объектов.

База данных может содержать конфиденциальные данные: номера кредитных карт, сведения о зарплате, названия подразделений и т. д. Чтобы защитить эту информацию от пользователей, которым доступ к ней запрещен, или от сотрудников внешних организаций, которым база передается для разработки, тестирования и пр., используется механизм маскирования. В версии Postgres Pro Enterprise 15 за эту функцию отвечает расширение pgpro_anonymizer. Для передачи базы с конфиденциальными сведениями осуществляется статическое маскирование — информация маскируется в самой передаваемой базе. Если же надо, чтобы часть пользователей видела данные в незамаскированном виде, а другая часть — в замаскированном, то используется динамическое маскирование. В этом случае маскирование выполняется «на лету» и только для части пользователей. Например, в номере телефона оставляется код города и страны, а остальные цифры забиваются звездочками. Маскирование данных производится на уровне колонок таблицы за счет вызова функций маскирования, которые могут быть пользовательскими или стандартными (например, перемешивание, удаление, статическая замена, частичное маскирование, фальсификация, отклонение, обобщение, шифрование, хэширование).

МАСШТАБИРОВАНИЕ

СУБД PostgreSQL обычно используется для работы с базами относительно небольшого объема (менее 10 Тбайт), а в Postgres Pro Enterprise для эффективного использования всех ресурсов компьютера, поддержки больших баз и высоких нагрузок разработаны собственные дополнительные механизмы поддержки крупных баз. Кроме механизмов мастер-реплика, Мультимастер и шардинг, позволяющих построить многосерверную конфигурацию и распределить нагрузку по нескольким компьютерам, имеется еще один механизм для работы с большими таблицами —

секционирование (partitioning). В СУБД Postgres Pro имеется собственный механизм секционирования, а в PostgreSQL — свой, и пользователи могут выбрать любой из них. Секционирование от Postgres Professional позволяет реализовать интервальное секционирование и по синтаксису напоминает команды Oracle.

При секционировании многие операции с секциями можно выполнять онлайн, без остановки СУБД. Доступны hash-, range-, list- и композитное секционирование (например, range + hash). Оптимизатор запросов учитывает секционирование и при соединении одинаково секционированных таблиц соединяет секции параллельно (partition wise join). Секционирование не только ускоряет выполнение запросов, но и упрощает администрирование больших таблиц.

Другой механизм ускорения работы запросов — распараллеливание. Запросы разных сессий в PostgreSQL выполняют разные бэкенд-процессы, работающие параллельно на разных ядрах — этот параллелизм обеспечивается самой архитектурой СУБД, а для долго выполняющихся запросов одной сессии можно использовать встроенный механизм параллельного выполнения отдельного запроса. При этом процесс-координатор запустит несколько рабочих процессов, которые параллельно выполнят части запроса и вернут результат координатору. Распараллеливаться могут операции сканирования таблиц и индексов, соединения и агрегирования. Администратор может выдавать подсказки оптимизатору запросов на включение/выключение параллельного выполнения запроса.

Для больших баз одним из важнейших механизмов снижения затрат на хранение данных и ускорение выполнения запросов является сжатие данных. В Postgres Pro Enterprise встроен механизм CFS (Compression File System) — для табличного пространства можно задать степень и алгоритм сжатия (lz4, zstd, zlib, pqlz) для всех таблиц и индексов из этого табличного пространства. Обычно степень сжатия зависит от данных, но на практике можно говорить про сжатие в два-три, а иногда и в пять раз. И этот эффект мультиплексируется — сжимается не только основная база, но и реплики, шарды, узлы Мультимастера, база разработки и тестирования. Сжимать можно и резервные копии и файлы дампа. Мало того, сжимать можно и неструктурированные данные, хранящиеся в специальных TOAST-таблицах. Хорошо поддаются компрессии тексты, файлы XML, тексты в формате JSON, а использование формата JSONB позволяет компактно хранить бинарные оптимизированные представления JSON-документов. Как показывает опыт, при использовании дешевых, медленных дисков для хранения базы данных сжатие в Postgres Pro Enterprise в разы обеспечивает ускорение выполнения запросов.

При увеличении нагрузки удобно переносить обработку на другие узлы, как это предусмотрено в Мультимастере, или выполнять распределенные запросы. Можно, например, выносить некоторые таблицы в другие базы данных или в файлы ОС — для этого в Postgres Pro Enterprise используется механизм оберток FDW (Foreign Data Wrapper), позволяющий хранить часть данных в таблицах другой базы PostgreSQL или другой базы данных (Oracle, Microsoft SQL Server и т. д.) либо вообще в плоских файлах на диске, работая с этим множеством таблиц как с единым целым. При этом подзапросы «проталкиваются» в другие базы для параллельного исполнения. Имеются стандартные обертки к популярным СУБД и форматам файлов, но можно написать и собственные. Подобный механизм позволяет выносить часть таблиц на компьютеры и базы, где они будут обрабатываться быстрее.

Встроенные механизмы Postgres Pro Enterprise предназначены в первую очередь для ускорения выполнения операций, но они важны и для масштабирования, позволяя обеспечить приемлемую скорость работы с большими таблицами и под большой нагрузкой. Сюда можно отнести и поддержку материализованных представлений и поддержку огромного числа индексов разных видов, механизмы эффективной реализации многоверсионности (читатели и писатели никогда не блокируют друг друга), ускоренный инкрементальный бэкап и т. д.

Работу с высоконагруженными системами ускоряет поддержка в Postgres Pro Enterprise 64-битного счетчика транзакций. В СУБД на основе PostgreSQL существует проблема 32-битного счетчика транзакций. Дело в том, что для того чтобы читатели и писатели не блокировали друг друга (это снижает производительность), используется механизм многоверсионности записи таблицы MVCC (multiversion concurrency control), основанный на использовании номеров транзакций. Однако 32-битный счетчик может адресовать только чуть больше четырех миллиардов транзакций, а затем обнуляется, поэтому номера старых и новых транзакций могут совпасть. Чтобы этого избежать, при высокой интенсивности работы, типичной для высоконагруженных систем, периодически производится «заморозка» счетчика — деактуализация старых номеров транзакций. Это приводит к временному, но существенному падению производительности. Счетчик 64 бит может адресовать около 18 квантилионов транзакций — вероятность заморозки ничтожно мала.

Вообще говоря, у Postgres Pro Enterprise мало ограничений — объем базы данных, количество записей в таблице, количество индексов не ограничены; максимальный размер одной таблицы — 32 Тбайт, размер первичного ключа — 32 колонки, а максимальная длина записи — 400 Гбайт.

ПРОИЗВОДИТЕЛЬНОСТЬ

При выборе СУБД и выполнении проектов миграции решающее значение очень часто имеют скорость выполнения отдельных запросов и скорость работы в многопользовательском режиме при росте нагрузки. Имеется много стандартных тестов [4] (TPC, HummerDB, тест Гилева для «1С», Pyrabench и т. д.), но результаты нагрузочного тестирования сильно зависят от множества факторов: типа и объема данных, вида нагрузки, опыта тестировщика, выбора и настройки тестов и т. д. Только полноценное нагрузочное тестирование поможет определить СУБД, обеспечивающую наибольшую производительность для реальной нагрузки и конкретного приложения.

Вместе с тем надо анализировать, какие средства можно использовать для ускорения работы приложения: сжатие данных; секционирование; вынесение части нагрузки на другие узлы; шарды; отключение некоторых механизмов (сбор статистики, немедленный commit — фиксация транзакции); 64-битный счетчик транзакций; увеличение числа ядер и кэшей памяти и т. д. Один из главных факторов, определяющих производительность СУБД, — качество работы оптимизатора запросов, от которого зависит, насколько быстро будет работать даже неаккуратно написанный код.

Работу оптимизатора можно корректировать с помощью подсказок (hint) и механизма заморозки хороших планов запросов (расширение sr_plan) [5]. Оптимальные планы выполнения запросов строятся и оцениваются на основе статистики, собранной при работе СУБД. Если она устарела, то и план будет неоптимальным. В Postgres Pro Enterprise реализовано расширение AQO (адаптивная оптимизация запросов), которое сохраняет статистику шагов реального выполнения запроса и может использоваться при сле-

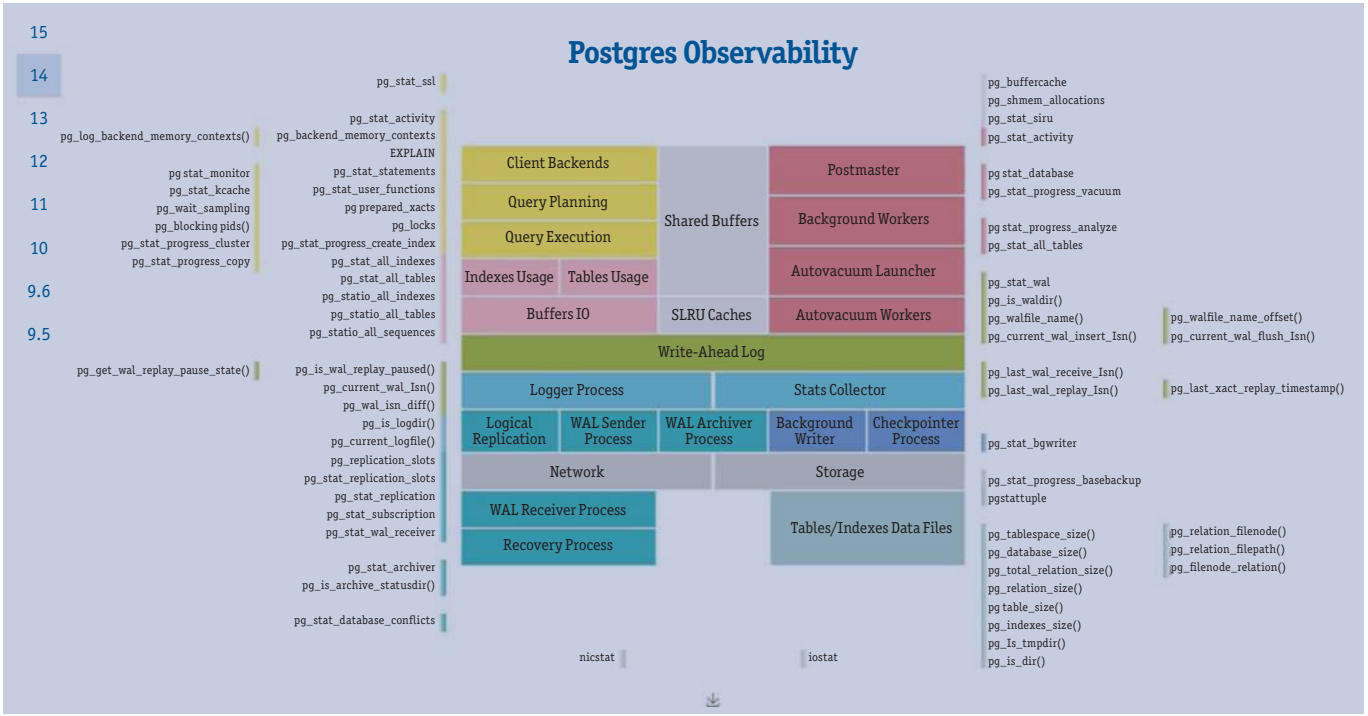


Рис. 3. Инструмент Postgres Observability

дующем выполнении этого запроса, корректируя ошибки оптимизатора запросов.

Существенное влияние на скорость исполнения запросов оказывают индексы — одна лишь замена сканирования данных на выборку по индексу дает значительное ускорение работы в OLTP-системах. Postgres Pro Enterprise поддерживает использование множества типов индексов — это и традиционные B-деревья, и хэш-индексы (хороши при выборке по значению ключа), и специфические индексы.

Индексы GIST (generalized search tree) очень эффективны при работе с текстами, геоинформацией и рисунками, позволяя разбить индексируемое пространство на квадраты и индексировать их с учетом функции соответствия (например, по близости, вхождению, пересечению, нахождению правее/левее, яркости, цвету, KNN и т. д.). Индексы SP-GIST (space partitioning GIST) развивают алгоритм GIST, но делят пространство на непересекающиеся квадраты. Индексы GIN — обратные индексы часто используются при полнотекстовом поиске и позволяют быстро найти в тексте нужные элементы. GIN напоминают глоссарий в печатных книгах, по которым сразу можно понять, на каких страницах используется конкретный термин. Индексы RUM — развитие GIN и также удобны при поиске релевантных документов в полнотекстовом поиске, но, в отличие от GIN, такой индекс знает позицию слова на странице и может, например, решить задачу быстрого поиска текстов, где заданные слова отстоят друг от друга на x слов. Они также удобны для определения «похожести» документов. Индексы BRIN (block range index) позволяют исключать из просмотра часть данных, разделяя отсортированные индексируемые данные на блоки, для каждого из которых известно минимальное и максимальное значение индексируемого элемента. После этого просто определять, какие блоки уже не надо просматривать при поиске. Также в Postgres Pro Enterprise можно использовать покрывающие индексы (данные результата берутся из индекса, без обращения к таблице), индексы по выражению, триграммы и т. д. Если надо индексировать для поиска на равенство значений большое количество колонок таблицы, то очень удобны Bloom-индексы на основе bloom-фильтров. Один bloom-индекс может заменить множество индексов B-tree.

При частом выполнении запроса много времени расходуется на его разбор и построение плана выполнения. Эту работу можно ускорить, используя заранее подготовленные запросы или включив режим автоматической подготовки часто используемых запросов. При работе с JSON-документами использование формата JSONB (бинарный JSON) позволяет заранее подготовить бинарное оптимизированное представление документа, что сильно ускорит работу. Использование компилятора JIT (Just in time) также ускоряет работу. Компиляция «точно в нужное время» — производимая во время выполнения процедура преобразования интерпретируемого варианта исполнения программы в программу на языке процессора.

УПРАВЛЯЕМОСТЬ

Серьезная проблема современных отечественных СУБД — отсутствие эффективной унифицированной графической системы управления, мониторинга и администрирования. Обычно для этих целей используется либо набор команд/функций, либо множество разнородных графических приложений с открытым кодом. Хорошая система управления должна, как минимум, выполнять следующие задачи:

- управление объектами базы данных и кластера (просмотр иерархии и характеристик объектов, их создание/изменение/удаление, администрирование базы, например, управление табличными пространствами, пользователями, ролями, журналами, экземплярами базы и т. д.);
- мониторинг поведения базы данных (сбор метрик, графики их изменения со временем, дашборды, богатый набор шаблонов мониторинга, тенденции использования ресурсов, проактивный мониторинг и т. д.);
- конфигурирование архитектур (кластера, базы данных, реплик, узлов, внешних таблиц, резервного копирования, восстановления и т. д.);

- управление безопасностью (привилегии, пароли, аудит, RLS, маскирование и т. д.);
- диагностика состояния базы данных и запросов (поиск, выявление, предсказание проблем);
- настройка базы данных и запросов.

В Postgres Pro Enterprise удобные графические инструменты имеются пока для двух первых задач, а остальные выполняются администратором путем вызова соответствующих функций базы и с помощью таблиц/представлений словаря базы. В администрировании очень помогает инструмент Postgres Observability, который для каждой проблемы конкретной версии СУБД показывает набор функций и таблиц словаря, требуемых для диагностики, настройки, конфигурирования (рис. 3).

Для управления объектами базы и администрирования СУБД имеется множество бесплатных графических инструментов: pgAdmin, Dbeaver, Valentina Studio, Navicat и пр. Для мониторинга обычно используется Zabbix или Prometheus, причем для Zabbix компания Postgres Professional предлагает расширенный агент мониторинга mamonsu, который работает быстрее штатного, собирает больше статистики и предоставляет расширенный набор шаблонов. Запланирован выпуск единого графического инструмента управления — Enterprise Manager, который позволит использовать GUI для решения всех задач управления и мониторинга.

В случае отсутствия хорошего графического инструмента, для управления можно использовать команды ОС и функции базы данных. СУБД Postgres Pro Enterprise постоянно собирает статистику о своей работе. Эту статистику можно разделить на две части: статистика для оптимизатора запросов (селективность, распределение данных по колонкам, средняя длина строки, кардинальность, размер таблицы и т. д.); статистика для администраторов базы данных и мониторинга ее состояния (события ожиданий, блокировки, запросы и планы их выполнения, количество прочитанных строк и т. д.). За сбор статистики для оптимизатора отвечает процесс Vacuum. Статистику для мониторинга собирают фоновые процессы. На основе собранной статистики можно выполнять диагностику состояния базы, выявлять и устранять проблемы. Расширение pgpro_stats выполняет мониторинг работы SQL-операций. Функции explain и explain analyze позволяют просматривать план выполнения запроса и принимать решение о его оптимизации или фиксации (расширение sr_plan).

Администраторам базы данных особенно полезен будет отчет PWR (Postgres workload report) — аналог отчета AWR в СУБД Oracle, без которого невозможен анализ состояния и проблем. Этот HTML-отчет позволяет выявить «тяжелые» SQL-запросы, проанализировать статистику их выполнения, оценить нагрузку на таблицы, сеансы, базу в целом и т. д. Можно сравнить статистику за разные периоды времени, чтобы узнать, например, почему раньше СУБД работала лучше.

МИГРАЦИЯ НА POSTGRES PRO ENTERPRISE

Сегодня многие отечественные организации переносят свои приложения с СУБД Oracle, Microsoft SQL Server и DB2. Каждый такой проект индивидуален и обычно требует больших ресурсов — нет механизмов автоматической миграции, однако любой проект миграции содержит следующие шаги:

- миграция структуры данных;
- миграция данных;
- миграция программного кода (функции, процедуры, триггеры, пакеты);
- функциональное и нагрузочное тестирование;
- обеспечение требуемого уровня надежности, безопасности, масштабируемости, производительности).

Наибольших ресурсов требуют последние два пункта. Миграция структуры и данных обычно не вызывает проблем (на этом этапе решаются вопросы несоответствия типов данных, зависимости объектов, скорости перекачки и докачки больших объемов данных и пр.), но миграция кода программных объектов базы слабо автоматизирована и требует его переписывания, а также, возможно, изменения логики работы программ. Некоторых объектов, привычных пользователям западных СУБД, в PostgreSQL может просто не быть либо алгоритмы и API работы с ними сильно отличаются.

Для выполнения миграции в PostgreSQL существует множество инструментов: бесплатные утилиты ora2pg (миграция с Oracle), sqlserver2pgsql (миграция в Microsoft SQL Server), Inspirer, Pentaho Kettle, пакет от Diasoft, Cybertech Migrator. Поскольку в PostgreSQL отсутствуют некоторые системные пакеты и функции Oracle (например, sysdate, dbms_output и utl_file), для эмуляции работы с ними имеется специальное расширение orafce.

Для упрощения миграции с СУБД Oracle в Postgres Pro Enterprise в версии 15 реализована поддержка пакетов, добавлены системные пакеты в расширение orafce, обеспечена поддержка скриптов с параметрами, модифицирована утилита миграции ora2pgpro.

Миграцию с Oracle сильно упрощает связка Postgres Pro Enterprise + ora2pgpro + orafce. Кстати, в некоторых утилитах миграции имеется возможность оценки сложности и трудоемкости проекта миграции, что может служить основой для планирования работ.

***

СУБД Postgres Pro Enterprise — хороший кандидат для импортозамещения, обеспечивающий привычный для разработчиков и администраторов инструмент работы с данными и предоставляющий средства оперативной миграции высоконагруженных приложений. Данная СУБД позволяет обеспечить необходимый уровень надежности, безопасности и производительности приложений. Однако надо помнить — нет средств автоматической миграции кода западных СУБД в Postgres Pro Enterprise, а сама миграция может оказаться долгим и трудоемким процессом, требующим тщательного предварительного планирования. ■

ЛИТЕРАТУРА

1. Виталий Максимов и др. Защищенная реляционная СУБД Линтер // Открытые системы.СУБД. — 1999. — № 11–12. — С. 69–79. URL: https://www.osp.ru/os/1999/11-12/177904 (дата обращения: 21.05.2023).

2. Дмитрий Волков. Сила в сообществе // Открытые системы.СУБД. — 2016. — № 2. — С. 38–41. URL: https://www.osp.ru/os/2016/02/13049332 (дата обращения: 21.05.2023).

3. Сергей Муравьев, Сергей Дворянкин, Игорь Насенков. СУБД: проблема выбора // Открытые системы.СУБД. — 2015. — № 1. — С. 22–24. URL: https://www.osp.ru/os/2015/01/13045322 (дата обращения: 21.05.2023).

4. Андрей Николаенко. Эталонные тесты СУБД: что было, что стало, что будет // Открытые системы.СУБД. — 2017. — № 2. — С. 35–39. URL: https://www.osp.ru/os/2017/02/13052225 (дата обращения: 21.05.2023).

5. Сергей Кузнецов. Оптимизация запросов: вечнозеленая область // Открытые системы.СУБД. — 2003. — № 4. — С. 69–71. URL: https://www.osp.ru/os/2003/04/182939 (дата обращения: 21.05.2023).

Марк Ривкин (m.rivkin@postgrespro.ru) — руководитель отдела технических консультантов, Postgres Professional (Москва).