|  |  |  |
| --- | --- | --- |
| **Используемые объекты и технологии Oracle** |  | **Реализация в PostgreSQL** |
| Хранимые процедуры | + | Совместимые.  С примерами процедур можно ознакомиться на стенде. |
| Хранимые функции | + | Совместимые.  С примерами функций можно ознакомиться на стенде. |
| Пакеты(packages) | + | Начиная с версии 15 появилась иммитация пакетов - команда CREATE PACKAGE. Пакет — это по сути схема, которая помогает организовать взаимосвязанные именованные объекты, поэтому его также можно создать командой [CREATE SCHEMA](https://postgrespro.ru/docs/enterprise/15/sql-createschema), и с ним можно выполнять те же действия, что и с обычной схемой. Однако пакет может содержать только функции, процедуры и составные типы. На стенде имеется демонстрация того, как пакеты Oracle мигрировали в PostgreSQL. |
| Триггеры | + | Совместимые (функциональная часть триггера представляется тригерной функцией).  С примером триггера можно ознакомиться на стенде. |
| Объектные типы (object types) | +- | В PostgreSQL не поддерживается такой тип как object. Однако его можно реализовать имеющимся функционалом, используя различные подходы. В частности, объектны тип может быть представлен как схема, со всеми необходимыми методами и свойствами. Также подобная функциональность может быть разработана с помошью внешних языков и внедрена в БД. |
| Коллекции (collection type) | +- | Коллекции могут быть реализованы как с помощью массивов, так и в виде временных таблиц. Однако имеется ряд расширений (plv8), которые могут поддерживать аналогичную фунциональность как и для коллекций Oracle. Следует отметить, что все же они уступают в производительности. Имеются примеры на стенде. |
| Линки (database link) | + | Можно создавать линки как на дуругие БД PostgreSQL, так и на Oracle. Передача BLOB/CLOB (text/bitea) данных через dblink в PostgreSQL возможна. На стенде представлены примеры работы dblink. |
| Последовательности (sequences) | + | Совместимые.  С примерами последовательностей можно ознакомиться на стенде. |
| Индексы | + | Совместимые, но есть ряд других индексов, отсутствующих в Oracle. |
| Битмап индексы | + | В PostgreSQL нет bitmap индексов, как в Oracle, но GIN также очень хорошо справляется с дубликатами. Внутренняя структура индекса GIN на самом деле очень похожа на индекс растрового изображения. Индексы Bitmax используются для полей с низкой мощностью. Индексы GIN используются таким же образом. |
| Ограничения (constraints) | + | Совместимые. Имеются примеры на стенде. |
| Представления (views) | + | Совместимые. Имеются примеры на стенде. |
| Материализованные представления (materialized views) | +- | PostgreSQL имеет только одну опцию для создания materialized views как REFRESH COMPLETE ON DEMAND |
| Журнал материализованного представления (materialized views logs) | - | отсутствует |
| Планировщики заданий (scheduler jobs) | + | pgpro\_scheduler — это встроенное в Postgres Pro Enterprise расширение, позволяющее планировать и контролировать задания, а также управлять их выполнением в базе данных Postgres Pro Enterprise. pgpro\_scheduler отличается очень лёгкой реализацией, так как для планирования и контроля заданий, а также для управления ими он использует фоновые рабочие процессы. И при этом pgpro\_scheduler не задействует никакие клиентские подключения.  Для большей стабильности в каждой базе данных имеется собственный руководящий планировщик, а каждое запланированное задание выполняется в отдельном рабочем процессе. |
| Автономные транзакции | + | Автономные транзакции могут поддерживаться как новым синтаксисом, представленным в версии Pro (BEGIN AUTONOMOUS TRANSACTION), так и с помощью расширения dblink, позволяющего создать новое соединение к БД в рамках текущей транзакции.  Для нового синтаксиса следует выделить, что в рамках отдельной родительской транзакции можно выполнить несколько автономных, как последовательных, так и вложенных. По умолчанию Postgres Pro позволяет выполнить одновременно до 100 автономных транзакций во всех сеансах. Это ограничение можно увеличить, изменив параметр конфигурации **[max\_autonomous\_transactions](https://postgrespro.ru/docs/enterprise/15/runtime-config-resource" \l "GUC-MAX-AUTONOMOUS-TRANSACTIONS)**. Максимальный уровень вложенности ограничен значением 128 и не может быть изменён. |
| Профили | + | Создание профиля в кластере баз данных Postgres Pro выполняется командой CREATE PROFILE. Профили Postgres Pro позволяют установить парольную политику для пользователей, которым они назначены. Профили определяются на уровне кластера баз данных, поэтому они распространяются на все базы в кластере. |
| Расширенная репликация (advanced replication) master-master | +- | multimaster — это расширение Postgres Pro Enterprise, которое в сочетании с набором доработок ядра превращает Postgres Pro Enterprise в синхронный кластер без разделения ресурсов, который обеспечивает масштабируемость OLTP для читающих транзакций, а также высокую степень доступности с автоматическим восстановлением после сбоев.  Расширение multimaster осуществляет репликацию данных полностью автоматическим образом и позволяет одновременно выполнять пишущие транзакции и работать с временными таблицами на любом узле кластера. **Однако при этом нужно учитывать имеющиеся ограничения репликации.** (см. документацию)  Следует так же отметить, что поскольку multimaster  позволяет одновременно менять одни и те же данные в разных узлах, 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. |
| Отправка почты напрямую из Oracle | + - | Не имеется нативного механизма отправки почты из процедур или триггеров, но это может быть решено с помошью расширения plv8, поддерживающего функции JavaScript. Ниже приведен пример кода реализации функции send\_mail с помощью plv8.  const result = plv8.execute(`select \* from http((  'POST',  'https://api.sendgrid.com/v3/mail/send',  ARRAY[http\_header('Authorization', $1)],  'application/json',  $2  )::http\_request);`, ['Bearer ' + apiKey, JSON.stringify(payload)]);  const { status, content } = result[0];  if (status !== 202) {  plv8.elog(ERROR, `Sending email failed, status: ${status}, content: ${content}`);  } |
| Процесс слежения за изменениями (Oracle Database Change Notification - OCN) | +  - | Возможное решение с помощью простых триггеров и метода pg\_notify, с помощью которого и будут поступать сообщения об изменениях в таблицах. Также необходимо будет создать пользовательский сервис-listener, который будет перехватывать все необходимые сообщения и сохранять их на диске. |
| Использование hints в запросах и операторах | + | В Postgres Pro Enterprise используется оптимизатор запросов, учитывающий стоимость их выполнения, и при этом он использует статистику данных, а не жёсткие правила. Планировщик (оптимизатор) оценивает стоимость всех возможных планов выполнения оператора SQL, и в итоге выбирает для выполнения план с наименьшей стоимостью. Планировщик делает всё возможное, чтобы выбрать лучший план выполнения, но он не идеален, так как он не учитывает некоторые свойства данных, например корреляции между столбцами.  Модуль **pg\_hint\_plan** позволяет корректировать планы выполнения, применяя так называемые «указания», записываемые в виде простых описаний в SQL-комментариях особого вида. |
| Принудительная блокировка таблицы | + | В PostreSQL поддерживаются следующие типы блокировок:   1. Блокировки на уровне таблицы 2. Блокировки на уровне страниц 3. Блокировки на уровне строк 4. Рекомендательные блокировки (Postgres Pro также имеет средства создания блокировок, смысл которых определяют сами приложения. Такие блокировки называются рекомендательными, так как система не форсирует их использование — правильно их использовать должно само приложение.)   C возможныим режимами для каждого типа блокировки можно ознакомиться в документации. |
| Аудит | + | Расширение pg\_proaudit позволяет регистрировать различные события, связанные с безопасностью.  pg\_proaudit работает параллельно со стандартными средствами протоколирования PostgreSQL (сборщиком сообщений) и не зависит от них. Журнал событий безопасности расширения pg\_proaudit сохраняется отдельно от журнала работы сервера. При запуске Postgres Pro Enterprise расширение pg\_proaudit запускает специальный фоновый процесс, который записывает в журнал информацию о событиях безопасности. |
| Хранение изображений в BLOB | + | Вендор рекомендует складывать изобажения в определённый каталог, а в базе данных хранить только информацию о путях к файлам, так как использование BLOB-данных может привести к увеличению размера базы данных и снижению производительности  Если вопрос хранения изображений в БД критически важный, то имеются два подхода:   1. Небольшой файл то можно сохранить в поле типа bytea. Тип данных BYTEA позволяет хранить двоичные данные в таблицах базы данных (максимальный объем записи 1 Гб).   create table images (id int, image bytea);  insert into images values (1, pg\_read\_file('/path/to/image.jpg')::bytea);   1. Но надо учесть, что при операциях с полем image как bytea, PostgreSQL будет загружать все его содержимое в память. Поэтому лучше использовать другой способ - использование типа данных oid (Object Identifier), который позволяет хранить ссылку на внешний файл с бинарными данными. Данные на диске хранятся отдельно от таблицы, и доступ к ним осуществляется по идентификатору oid. При этом подходк возможно чтение порциями (chunk) для уменьшения требований к оперативной памяти сервера.   create table images (id int, image oid);  insert into images values (1, lo\_import('/path/to/image.jpg')); |
| Использование механизма RLS (Row Level Security) | + | В PostgreSQL имеется возможность в дополнение к стандартной [системе прав](https://postgrespro.ru/docs/enterprise/15/ddl-priv) SQL определять на уровне таблиц политики защиты строк, ограничивающие для пользователей наборы строк, которые могут быть возвращены обычными запросами или добавлены, изменены и удалены командами, изменяющими данные (RLS, Row-Level Security).  В отличие от Oracle, политики в нем компилируемые. Это упрощает разработку и устраняет проблемы с кешами, но дает чуть меньше возможностей. |
| Работа с транзакциями (чистка, обновления, потенциальная блокировка БД) | +- | Могут возникать проблемы с VACUUM.  Результат обычного VACUUM может быть неудовлетворительным, когда вследствие массового изменения или удаления в таблице оказывается много мёртвых версий строк. Для физического освобождения лишнего пространства, которое она занимает, необходимо использовать команду VACUUM FULL или, в качестве альтернативы, [CLUSTER](https://postgrespro.ru/docs/postgresql/15/sql-cluster) или один из вариантов [ALTER TABLE](https://postgrespro.ru/docs/postgresql/15/sql-altertable), выполняющий перезапись таблицы. Эти команды записывают абсолютно новую копию таблицы и строят для неё индексы. Все эти варианты требуют блокировки в режиме ACCESS EXCLUSIVE. Кроме того они также на время требуют дополнительного пространства на диске в объёме, приблизительно равном размеру таблицы, поскольку старые копии таблицы и индексов нельзя удалить до завершения создания новых копий. |
| Database Back Up | + | Для создания копий БД в PostgreSQL могут использоваться три программы: pg\_probackup, pg\_basebackup  и pg\_dump(pg\_dumpall).  pg\_probackup — это утилита для управления резервным копированием и восстановлением кластеров баз данных Postgres Pro. Она предназначена для регулярного создания резервных копий экземпляра Postgres Pro, позволяющих восстанавливать сервер в случае необходимости.  Программа pg\_basebackup предназначена для создания базовых копий работающего кластера баз данных Postgres Pro. Процедура создания копии не влияет на работу других клиентов базы. Полученные копии могут использоваться и для восстановления на момент времени, и в качестве базового состояния ведомого сервера при реализации трансляции журнала или потоковой репликации. pg\_dump — используется, чтобы выгрузить базу данных Postgres Pro в виде скрипта или в архивном формате. С деталями работы каждой из них можно ознакомиться в документации. |