PostgresPro

Новые возможности СУБД Postgres Pro Enterprise 15

Марк Ривкин
Начальник отдела,
Postgres Professional

Упрощение миграции с Oracle

PostgresPro

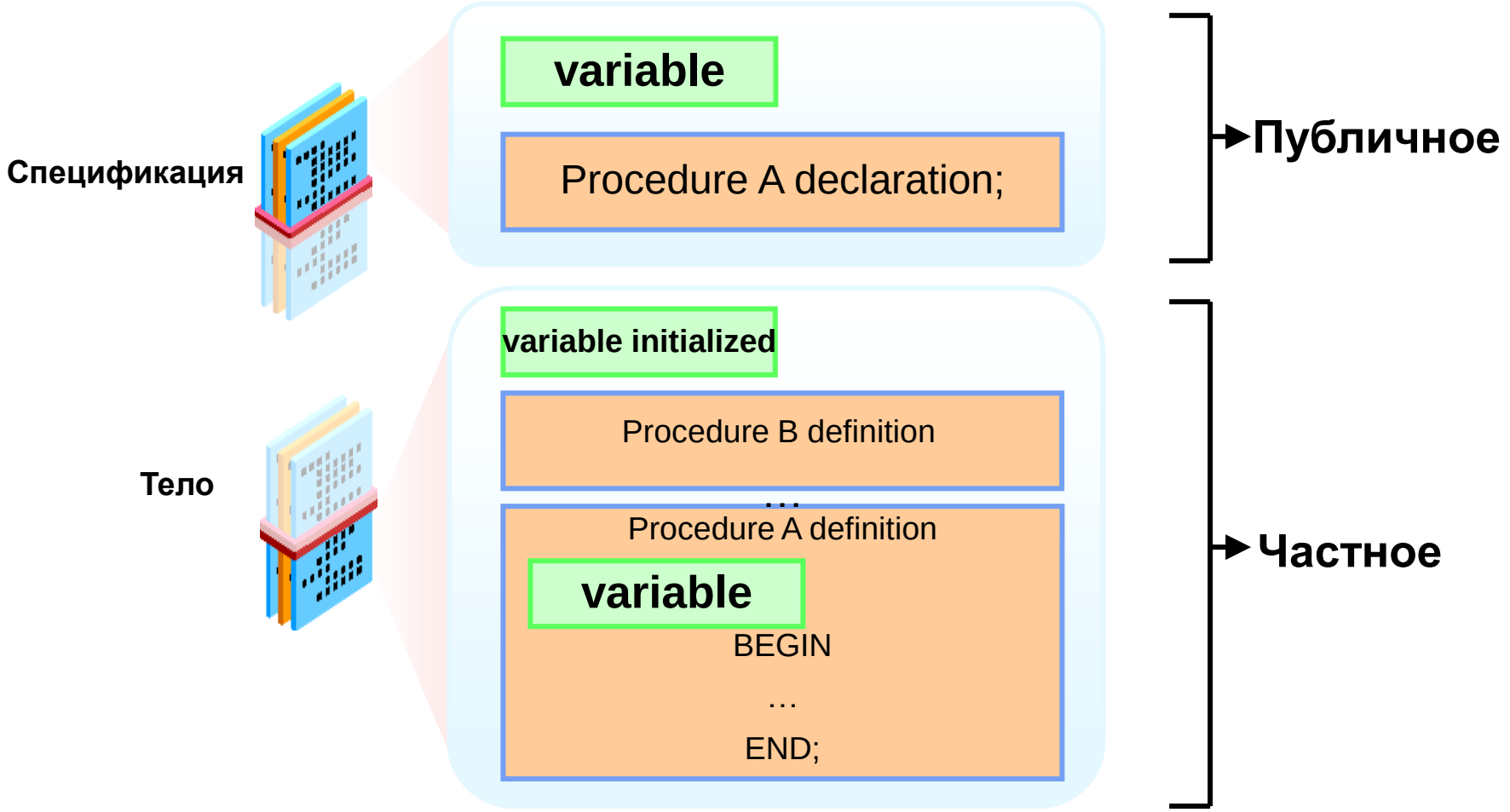
- Пакеты
- Ora2pgpro
- ORAFCE
- Параметры скриптов

Пакеты (Package) в Oracle

PostgresPro

- Группировка логически связанных процедур, функций, переменных, курсоров, констант, исключений, типов
- **Объект схемы**, компилируется и хранится в БД
- **2 части**: спецификация пакета и тело пакета. Тело можно дописать позже
- При вызове процедуры пакета **весь пакет грузится в память** и остальные процедуры работают быстрее
- Может быть область инициализации переменных Срабатывает при первом обращении к элементу пакета
- Не все **процедуры** пакета могут быть в спецификации – тогда они **видны только в пакете**
- Пример: DBMS_OUTPUT.PUT_LINE

Компоненты PL/SQL пакета (Oracle)

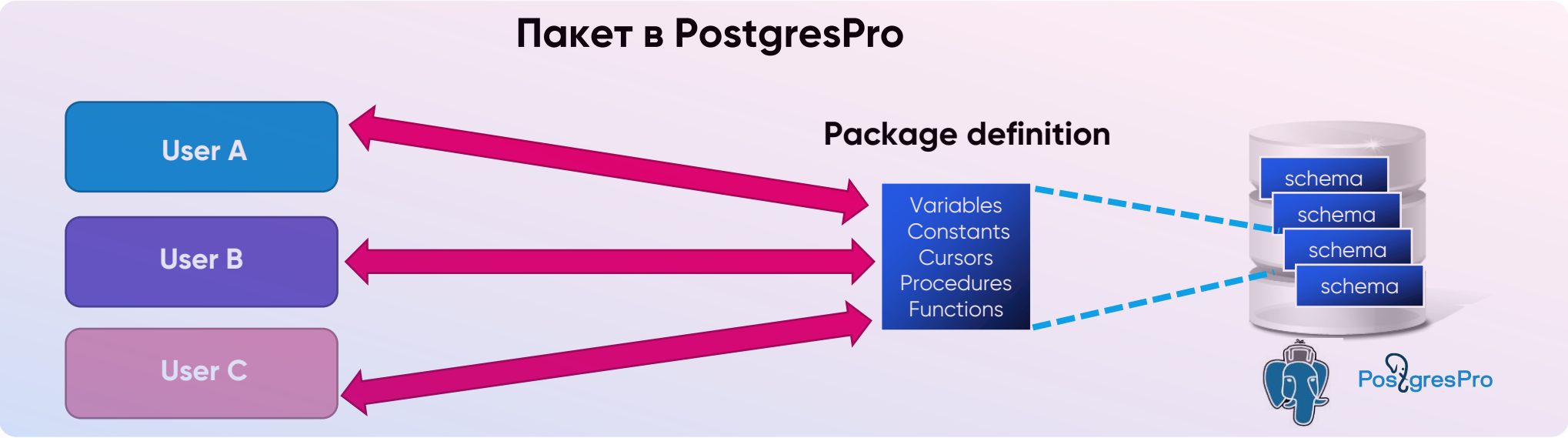


Поддержка пакетов PL/pgSQL в Postgres Pro 15 EE

PostgresPro

15.1

- Пакет является отдельной схемой
- Поддерживается функции инициализации – процедура с фиксированным именем "*__init__*" – вызывается в момент первого обращения к пакету
- Поддерживаются глобальные переменные, типы, курсоры – все объявленные в процедуре инициализации "*__init__*"
- Обращение к процедурам других пакетов (схем)
- Grant на схему (пакет)



Определение пакета в PostgresPro 15 EE

```
create or replace package htp
create function __init__() returns void as $$
declare
   htbuf   text[ ]; -- buffer
   rows_in int;    -- buffer size
begin
  htbuf   := '{}';
  rows_in := 0;
end;    $$

create procedure  p(v_pStr varchar)  as $$
begin
  htbuf = array_append(htp.htbuf, v_pStr::TEXT);
  rows_in := rows_in + 1;
end;    $$
;
```

- Преобразуется в: DROP SCHEMA IF EXISTS; CREATE SCHEMA …
- Секция инициализации
- Глобальные переменные пакета - объявлены в секции инициализации;
- Функция входит в пакет

Использование пакета в PostgresPro 15 EE

PostgresPro

```
do $$
begin
    call htp.p('<p>Hello</p>');
end;
$$;
```

• Использование процедуры из пакета

```
create or replace function qux.demo() returns int as $$
#import foo, bar
declare z int := foo.x + bar.y;
Y numeric;
begin
  if z > (foo.x + bar.y) then
    return 1;
  end if;
  return 1;
end  $$ language plpgsql;
```

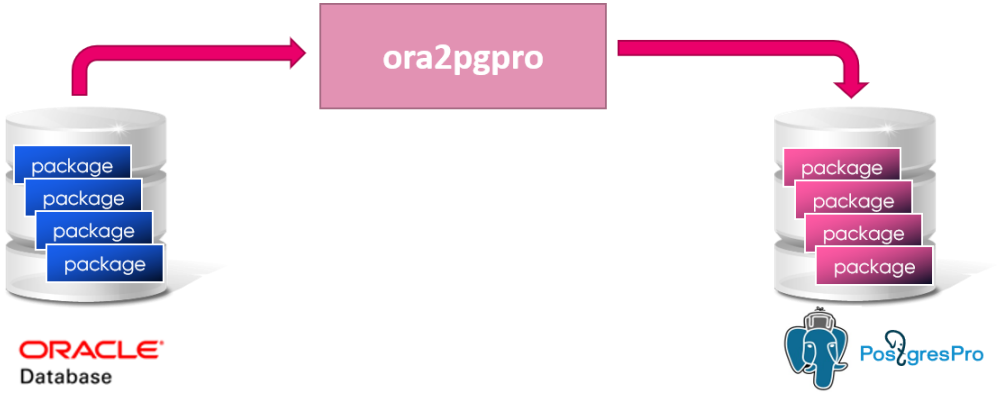
• Импорт глобальных переменных пакетов для использования

Ограничения в Postgres Pro 15.1 EE

- Имена пакетов в СУБД в всех схемах приложениях должны быть уникальными
- Не поддерживаются приватные переменные и методы пакета, - все являются публичными
- Пока отсутствует глобальная блокировка на изменение пакета, который в данный момент выполняется (аналог латча в СУБД Oracle);
- Пока отсутствует механизм отслеживания факта изменения пакета - аналог исключения *ORA-04068: existing state of packages has been discarded* в СУБД Oracle

Ora2pgpro и ORAFCE

- **Ora2Pgpro**
  - Анализирует исходный код PL/SQL в Oracle и формирует DDL-скрипты для PostgresPro 15 EE в синтаксисе PL/pgSQL
  - Транслятор кода пакетов Oracle в синтаксис PostgresPro 15 EE с добавлением прагм #import и #package
  - Генерация скриптов выдачи привилегий (grant)



- **ORAFCE**
  - dbms_application_info (% выполнения)
  - utl_mail
  - utl_htp (чтение по URL)
  - dbms_change_notification (подписка на инфо об изменении данных курсора)

Параметры при вызове скрипта в утилите psql

PostgresPro

15.1

- До PG Pro 15 вызов скрипта с параметрами в psql не поддерживался!

- В PG Pro 15 Enterprise поддерживается передача именованных и позиционных параметров при вызове скрипта в утилите psql

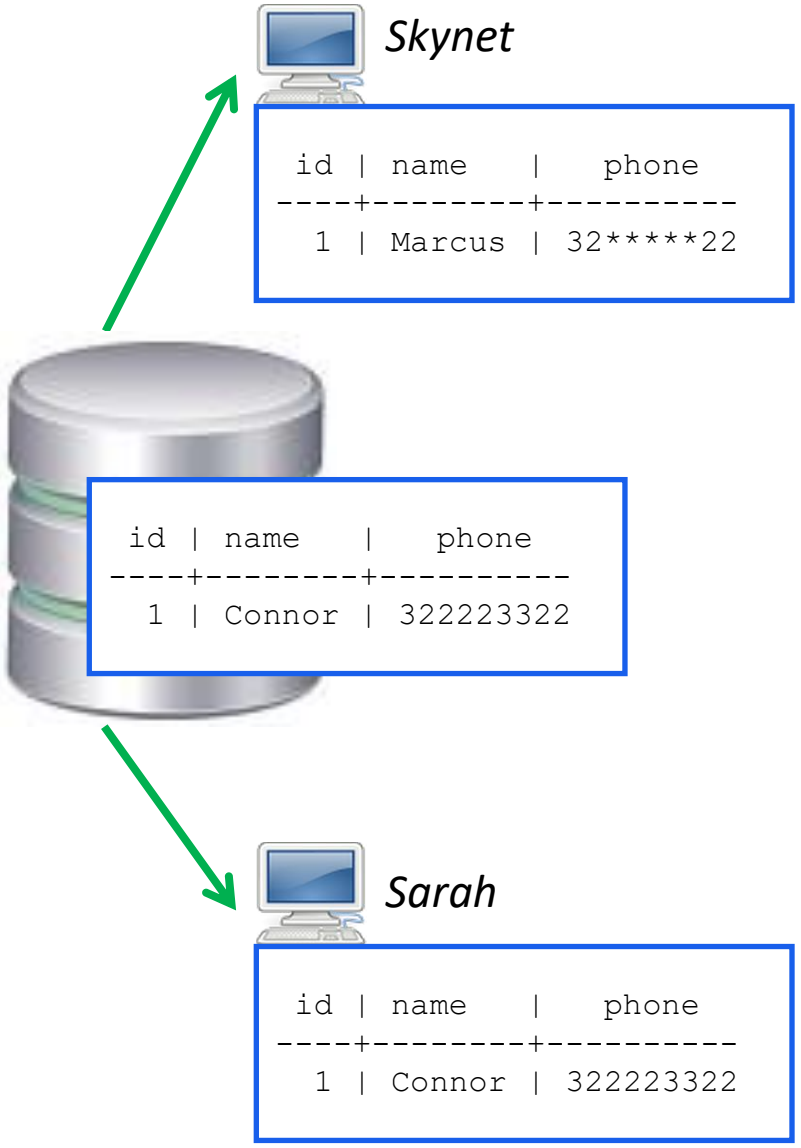
```
mydb=#  \i  expfile.sql   directory=PG_DUMP_DIR  dumpfile=myfile.txt  12
```

- Позиционные параметры получают имена PSQL_ARGn (где: n от 1 до 42)
- Использование параметров в скрипте psql

```
[student@localhost] more expfile.sql

\echo :dumpfile
\echo :directory
\echo :PSQL_ARG1
```

Маскирование данных



pgpro_anonymizer —расширение для маскирования или замены конфиденциальных данных непосредственно внутри экземпляра Postgres Pro:

- Динамическое маскирование: сокрытие данных только от недоверенных пользователей.
- Статическое маскирование: преобразование данных в БД в соответствии с правилами.
- Экспортирование замаскированных данных в файл SQL.

Правила маскирования задаются метками безопасности:

```
SELECT anon.start_dynamic_masking();

SECURITY LABEL FOR anon ON COLUMN people.name
   IS 'MASKED WITH FUNCTION anon.fake_first_name()';
SECURITY LABEL FOR anon ON COLUMN people.phone
   IS 'MASKED WITH FUNCTION anon.partial(phone,2,$$******$$,2)';

SECURITY LABEL FOR anon ON ROLE Skynet, Mark IS 'MASKED';
```

Маскирование данных

PostgresPro

Данные могут быть замаскированы несколькими способами:

- Удаление или статическая замена одним и тем же значением.
  Например: замена на значение «КОНФИДЕНЦИАЛЬНО».
- Частичное скрытие оставляет часть данных нетронутыми.
  Например: заменить часть номера кредитной карты на “X”.
- Фальсификация заменяет данные случайными, но правдоподобными значениями.
- Отклонение «сдвигает» даты и числовые значения.
  Например , применив отклонение на +/- 10% данные не потеряют смысл .
- Перестановка перемешивает значения в рамках столбца.
- Обобщение - замена диапазоном значений.
  метод полезен для аналитики, так как данные остаются верными
- Псевдонимизация защищает данные с помощью дополнительной информации
  Например: шифрование или хеширование.
- Пользовательские правила - предназначены для изменения данных в соответствии с особыми требованиями, чтобы данные оставались согласованными.

Расширение SR_PLAN – фиксация плана запроса

PostgresPro

Стабильность планов, ручная настройка

```
SELECT sr_id FROM sr_register_query('SELECT count(*)
FROM a
WHERE x = 1 OR (x > $2 AND x < $1) OR x = $1', 'int', 'int');

 sr_id
-------
     1
(1 row)
```

Execute a query with specific value of each parameter:

```
SELECT count(*) FROM a WHERE x = 1 OR (x > 11 AND x < 22) OR x = 22;
```

Addditional info in explain shows that this query is under control of sr_plan:

```
EXPLAIN SELECT count(*) FROM a WHERE x = 1 OR (x > 11 AND x < 22) OR x = 22;

 Custom Scan (SRScan)  (cost=0.00..0.00 rows=0 width=0)
   SR_PLAN: frozen plan
   ->  Aggregate  (cost=1.60..1.61 rows=1 width=8)
         ->  Seq Scan on a  (cost=0.00..1.60 rows=2 width=0)
               Filter: ((x = 1) OR ((x > $2) AND (x < $1)) OR (x = $1))
```

```
SET enable_seqscan = 'off';
```

```
 Custom Scan (SRScan)  (cost=0.00..0.00 rows=0 width=0)
   SR_PLAN: frozen plan
   ->  Aggregate  (cost=12.89..12.90 rows=1 width=8)
         ->  Index Only Scan using a_x_idx on a  (cost=0.14..12.89 rows=2 width=0)
               Filter: ((x = 1) OR ((x > $2) AND (x < $1)) OR (x = $1))
(5 rows)
```

Now, freeze this plan. This statement will use Index Only Scan even you change the planner options:

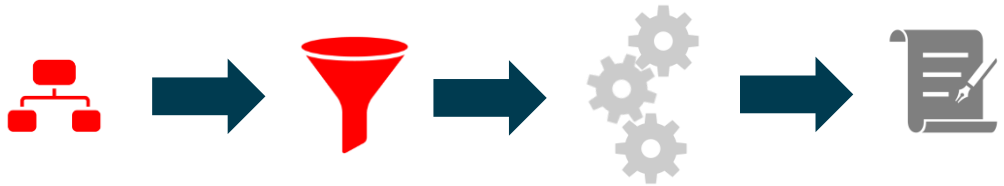
```
SELECT sr_plan_freeze(1);
RESET enable_seqscan;
```

Расширение PGPRO_STATS

PostgresPro

- Сбор дополнительных статистик

- **Session tracer** - часть pgpro_stats, позволит собирать статистику об отдельных запросах, подпадающих под гибкий набор условий. Например, долго выполняющихся, содержащих много дисковых операций, исходящих из определенного бэкенда..



Поддержка стандарта SQL/JSON

PostgresPro

15.1

```
CREATE TABLE cities ( js jsonb );
INSERT INTO cities VALUES ('
{ "города" : [
   { "название" : "Москва",
     "инфо" :
        { "код города" : "495"}     },
   { "название" : "Зеленоград",
     "инфо" :
       { "код города" : "49653",
         "часовой пояс" : "+3"}     },
   { "название" : "Зион"            }
 ] }    ');

SELECT jt.* FROM cities,
 JSON_TABLE ( js,
   '$.города?(@.название starts with "З")'
   COLUMNS (
     city  text PATH '$.название',
     phone text PATH '$.*."код города"' )
 ) AS jt;

   city          | phone
-----------------+-------------------
 Зеленоград  | 49653
 Зион           |
```

До 15 версии Postgres Pro поддерживал конструкцию JSONPATH для доступа к отдельным частям данных в формате JSON. В 15 добавлены стандартизированные функции для работы с JSON. Поддержка SQL/JSON позволяет обрабатывать данные JSON наряду с обычными данными SQL.

Функции SQL/JSON генерируют данные JSON из значений типов SQL или преобразуют JSON в типы SQL, обеспечивая встроенную поддержку типов данных JSON в среде SQL.

Функций SQL/JSON :

- функции-конструкторы (JSON_OBJECT , JSON_ARRAY, JSON_ARRAYAGG),
- функции проверки и сериализации (IS JSON OBJECT, IS JSON ARRAY),
- функции запросов (JSON_EXISTS, JSON_VALUE, JSON_QUERY) .

Функция JSON_TABLE обрабатывает данные JSON и выдаёт результаты в виде реляционного представления, к которому можно обращаться как к обычной таблице SQL.

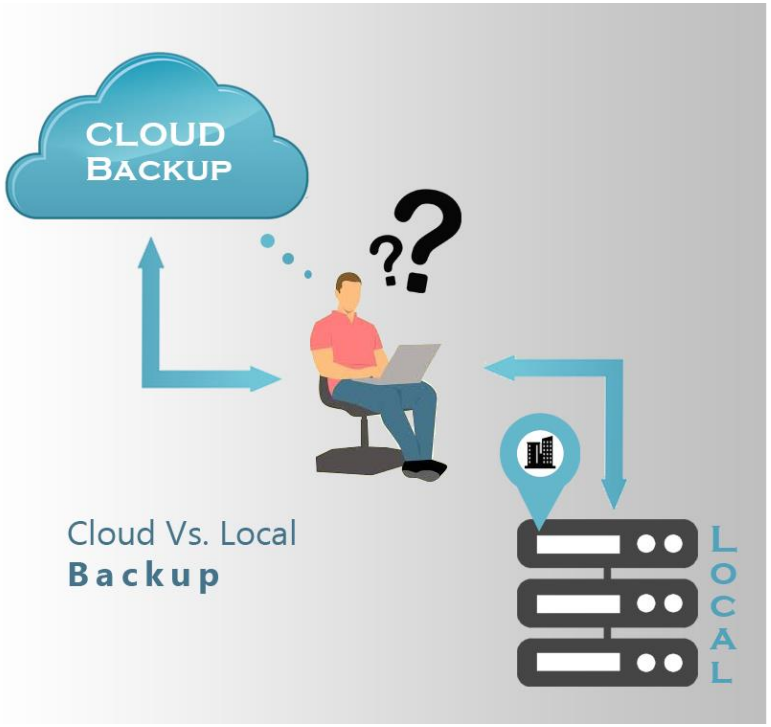
Функций SQL/JSON можно использовать также и для создания ограничений целостности для таблиц.

Pg_probackup

PostgresPro

15.1

- Pg_probackup - новая архитектура
- Pg_probackup - поддержка object storage S3
- Pg_probackup - поддержка PTRACK on CFS



Прочее в Postgres Pro Enterprise 15

PostgresPro

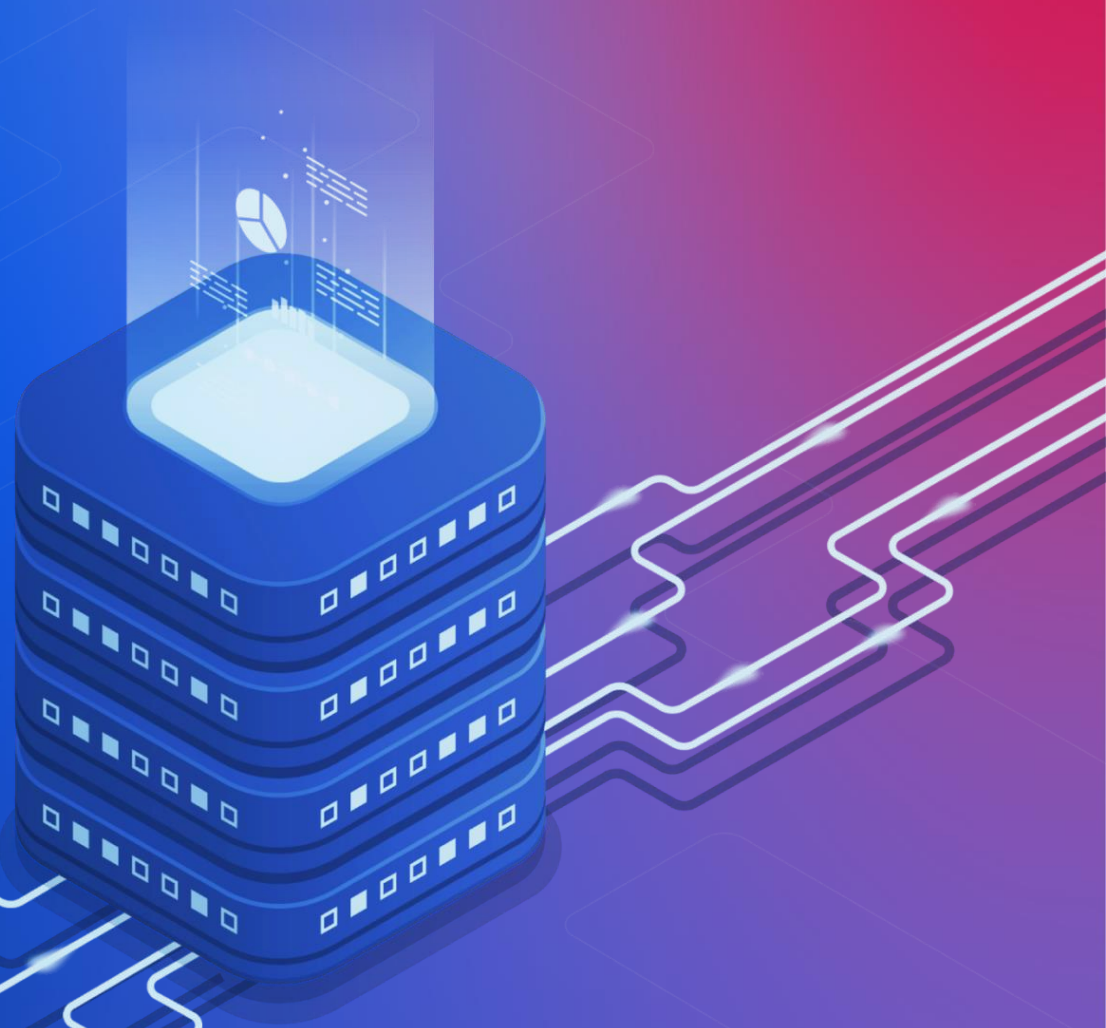
15.1

- Прекращение поддержки MS Windows (будет поддерживаться для версии 14 до окончания ее поддержки в 2026)
- Совместимость с расширением TimescaleDB (временные ряды) *
- Поддержка платформ ARM и Эльбрус
  - ALT 9/10 для e2kv3/e2kv4, ALT 8.2 SP для e2kv3/e2kv4
  - Astra Linux Leningrad 8.1
- Ускорение выпуска новых мажорных версий
  - Новая технология
  - Postgres Pro Standard 15 вышел через месяц после PostgreSQL 15
  - Postgres Pro Enterprise 15 тоже уже вышел (на 3 месяца быстрее, чем 14)

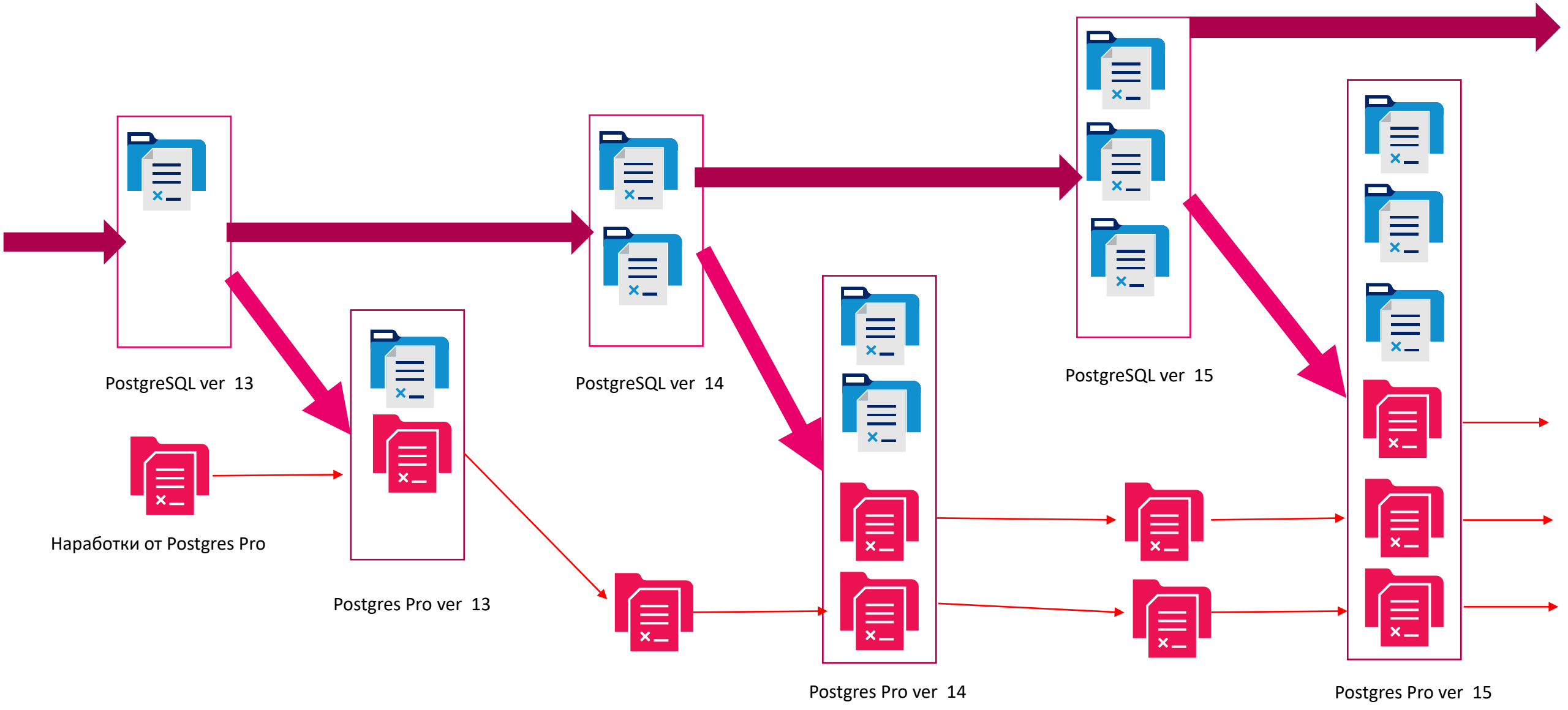
Из PostgreSQL 15

https://postgrespro.ru/docs/postgresql/15/release-15.html#id-1.11.6.6.5

Более 150 изменений



Слияние версий PostgreSQL и Postgres Pro Enterprise
(версии 11, 12, 13 ….)



Поддержка оператора MERGE

PostgresPro

15.1

EmployeeSource

ID	Name
1	Иван Иванов
2	Петров

EmployeeTarget

ID	Name
1	Иванов
3	Сидоров

```
MERGE INTO EmployeeTarget T
USING EmployeeSource  S
ON T.ID = S.ID
WHEN MATCHED THEN
  UPDATE SET NAME = S.NAME
WHEN NOT MATCHED THEN
  INSERT (ID,NAME) VALUES (S.ID, S.NAME);
```

EmployeeTarget

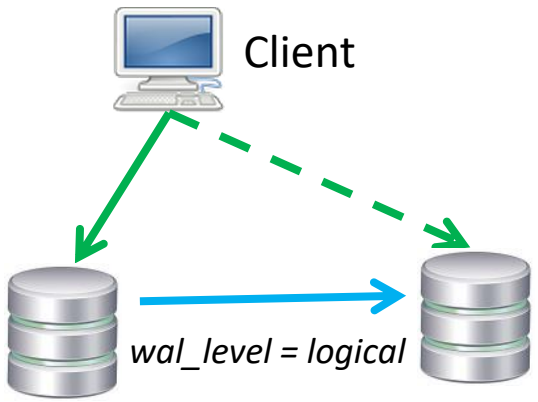
ID	Name
1	Иван Иванов
2	Петров
3	Сидоров

- Преимущества перед CONFLICT UPDATE ON:
  - Ориентирован на пакетную обработку в ETL
  - Позволяет добавлять доп. условия в предложение MATCHED/NOT MATCHED

Улучшения в логической репликации

PostgresPro

15.1



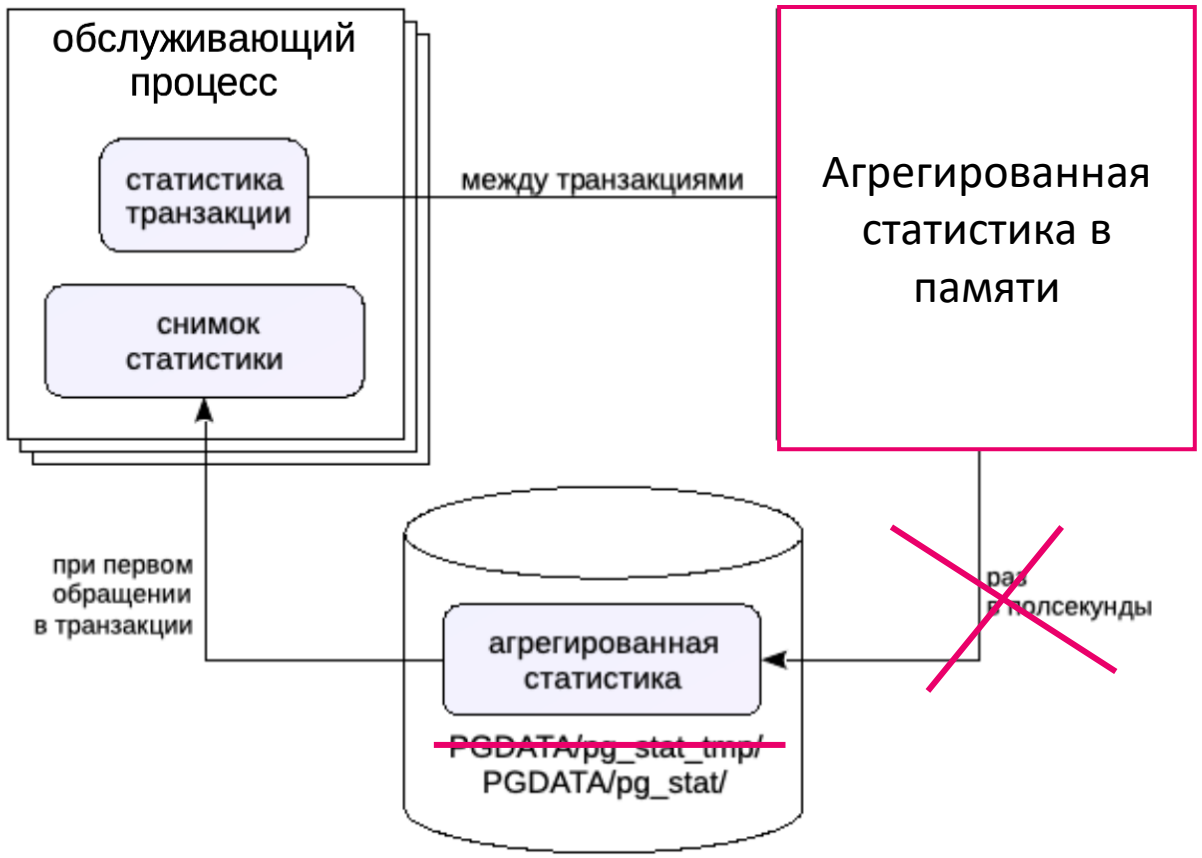
publication subscription

- публикация всех таблиц в схеме
  CREATE | ALTER PUBLICATION … FOR TABLES IN SCHEMA …
  (раньше только For ALL Tables БД)
- фильтрация содержимого публикации, строки, не удовлетворяющие условию WHERE, не будут публиковаться.
- ограничение содержимого публикаций определёнными столбцами
- остановка применения изменений логической репликации в случае ошибки (disable_on_error) на стороне подписчика
  (позволяет избежать бесконечного зацикливания при ошибке)
- Пропуск транзакции для разрешения конфликта: в команде ALTER SUBSCRIPTION появился параметр SKIP, где можно указать LSN команды завершающей конфликтующую транзакцию (LSN смотри в журнале сервера) (толкнуть дальше)

Накопление статистики в памяти

PostgresPro

15.1



Мониторинг – разное в 15

PostgresPro

15.1

- **Журналирование работы процесса startup** – новый параметр *log_startup_progress_interval* позволит увидеть в журнале, чем занимается startup

- **Журнал сервера в формате JSON** – полезно для сторонних систем анализа логов

- **Информация о JIT в pg_stat_statements** – проще оценивать влияние JIT на выполнение запросов

- **Статистика ввода/вывода по работе с временными файлами** – более информативный вывод команды explain (например, для функции)

Владелец базы - владелец схемы public

PostgresPro

15.1

По умолчанию при создании объектов БД без указания имени схемы они автоматически помещаются в схему «public». Ранее схемой «public» владел пользователь, выполнявший начальную инициализацию, и пользователи-владельцы баз данных, не имеющие прав суперпользователя, не могли ничего с этим сделать.

Начиная с 15-й версии каждый владелец базы данных теперь будет иметь права владельца схемы public в своей базе данных.

```
postgres=# \dn+
                                  Список схем
  Имя   |      Владелец      |             Права доступа              |        Описание
--------+--------------------+----------------------------------------+------------------------
 public | pg_database_owner | pg_database_owner=UC/pg_database_owner+| standard public schema
        |                    | =U/pg_database_owner                   |
(1 строка)
```

Обратите внимание, что обновление с предыдущей версии до 15-й версии (с использованием pg_dump или pg_upgrade) сохраняются исходные привилегии. Чтобы исправить обновленные базы данных, выполните следующие команды:

ALTER SCHEMA public OWNER TO pg_database_owner;

REVOKE CREATE ON SCHEMA public FROM PUBLIC;

Безопасность, psql

PostgresPro

15.1

- Улучшения в безопасности
  - Grant на изменение параметров alter system для не суперюзера,
  - Роли pg_checkpoint, pg_write_server_files (backup)
  - View с правами вызывающего

- Команда \dconfig[+] [шаблон] в psql (поиск по маске параметров конфигурации)

  \dconfig   listn*

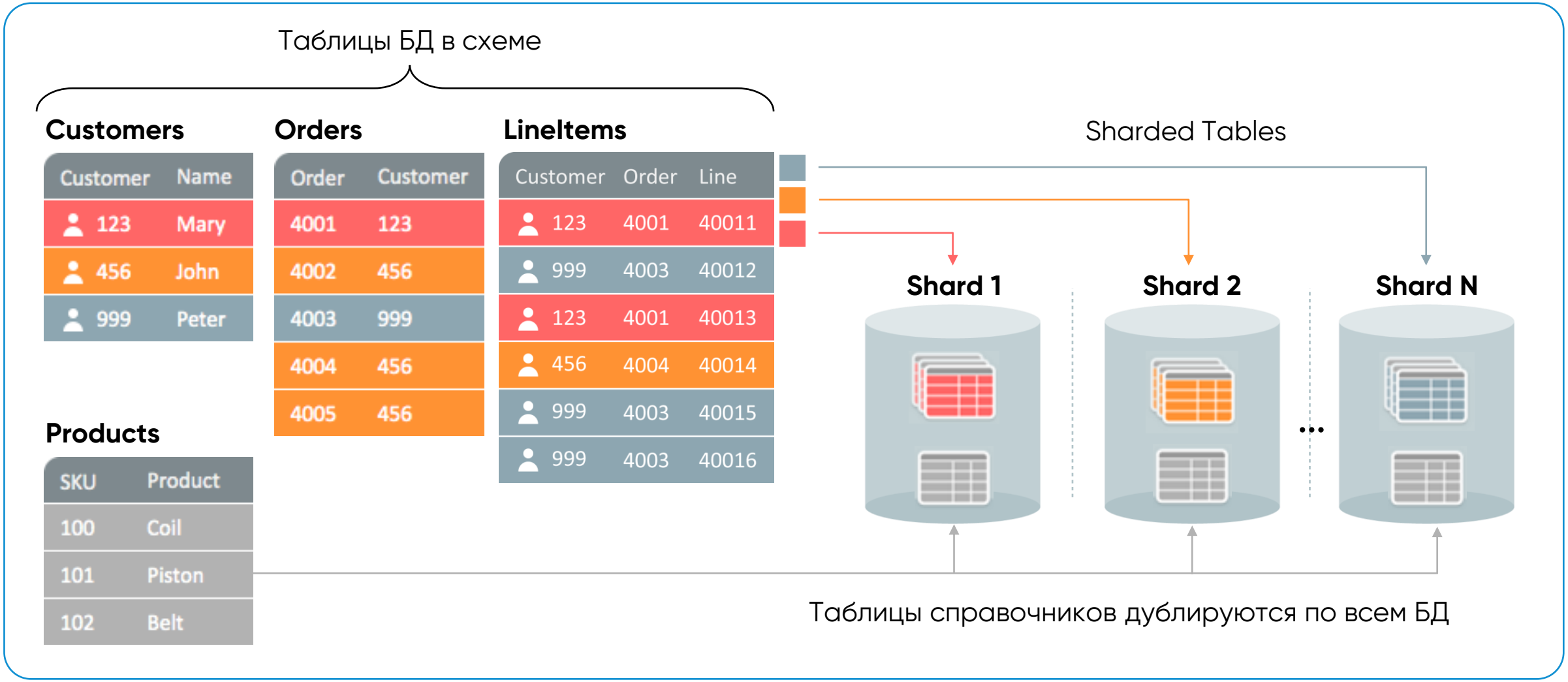
  ? – любой символ; * - любой набор символов

Прочее...

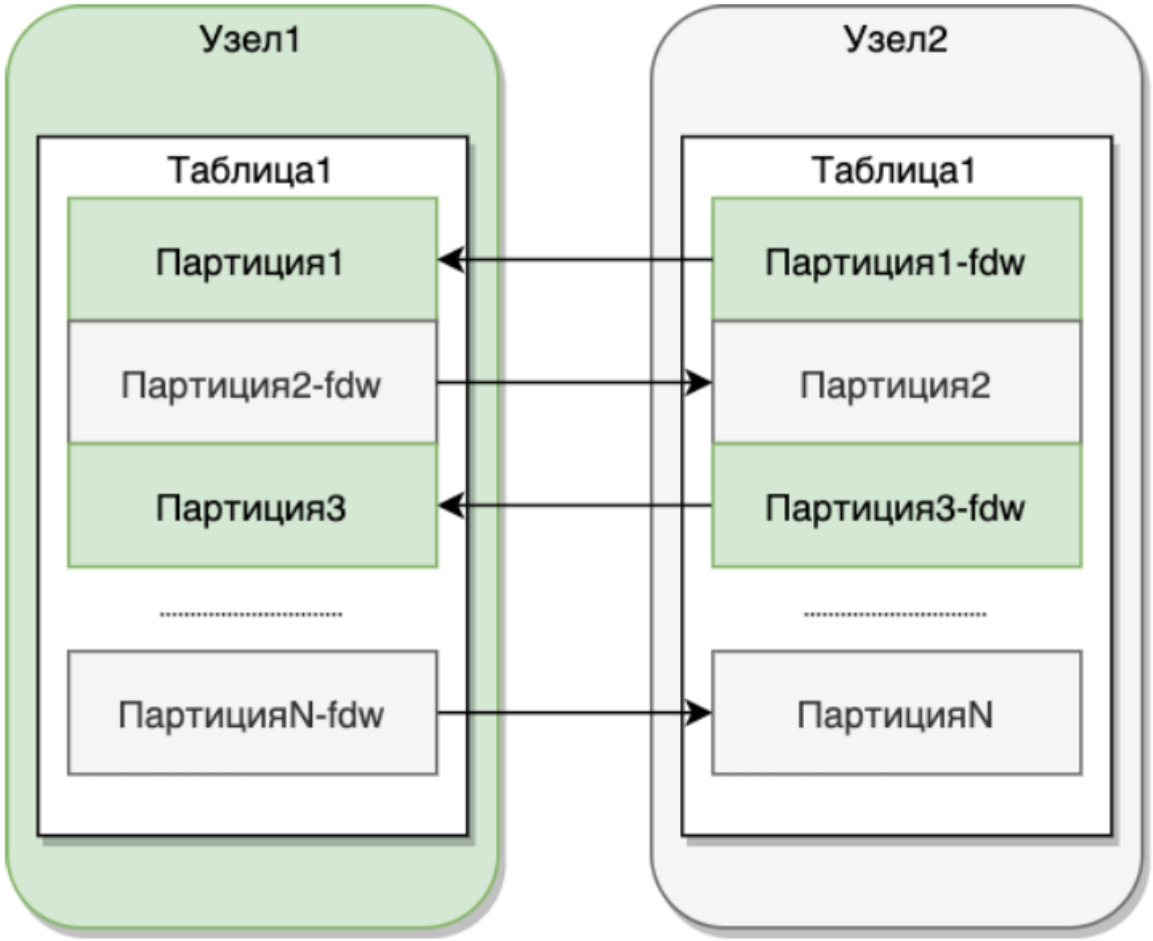
Шардирование

PostgresPro

Шардирование по хэш функции снимает ограничения на размер БД



Шардман – масштабируемость и надежность



Распределенная таблица - это партицированная таблица, состоящая из N секций. Количество партиций указывается при создании распределенной таблицы. Части таблицы находятся на всех узлах кластера. Партиции связаны между собой через механизм postgres_fdw. Таблицы доступны на запись на всех узлах кластера. Каждый шард - отказоустойчив, в случае выхода одного из узлов шарда из строя, данные не будут потеряны.

Соразмещенная таблица - это партицированная таблица, состоящая из N секций, каждая секция которой соразмещена с распределенной таблицей.

Глобальная таблица - это таблица, размещенная на всех узлах кластера, содержащая идентичные данные.

Шардман

PostgresPro

- **Перевод Шардман в production**

- **Загрузчик шардов Шаман**

Postgres Pro Enterprise 15 ++

Планы разработки

Планы на будущее

PostgresPro

- Свой встроенный HA кластер
- PPEM - Enterprise Manager
- DBaaS
- Resource manager (управление и приоритизация ресурсов)
- Анонимные блоки с параметрами
- Рекурсивные запросы (connect by)
- Pluggable TOAST
- PL/pgSQL wrapping (шифрование кода)
- Ограничение доступа DBA к данным
- Подключение чужих систем шифрования
- Transparent Data Encryption
- Интеграция Pg_probackup с другими СРК
- Baseline (управление планами запросов)
- Интеграция Pg_probackup с Шардман и EM
- Адаптивный экзекутор (несколько join в плане)

PostgresPro

Q&A