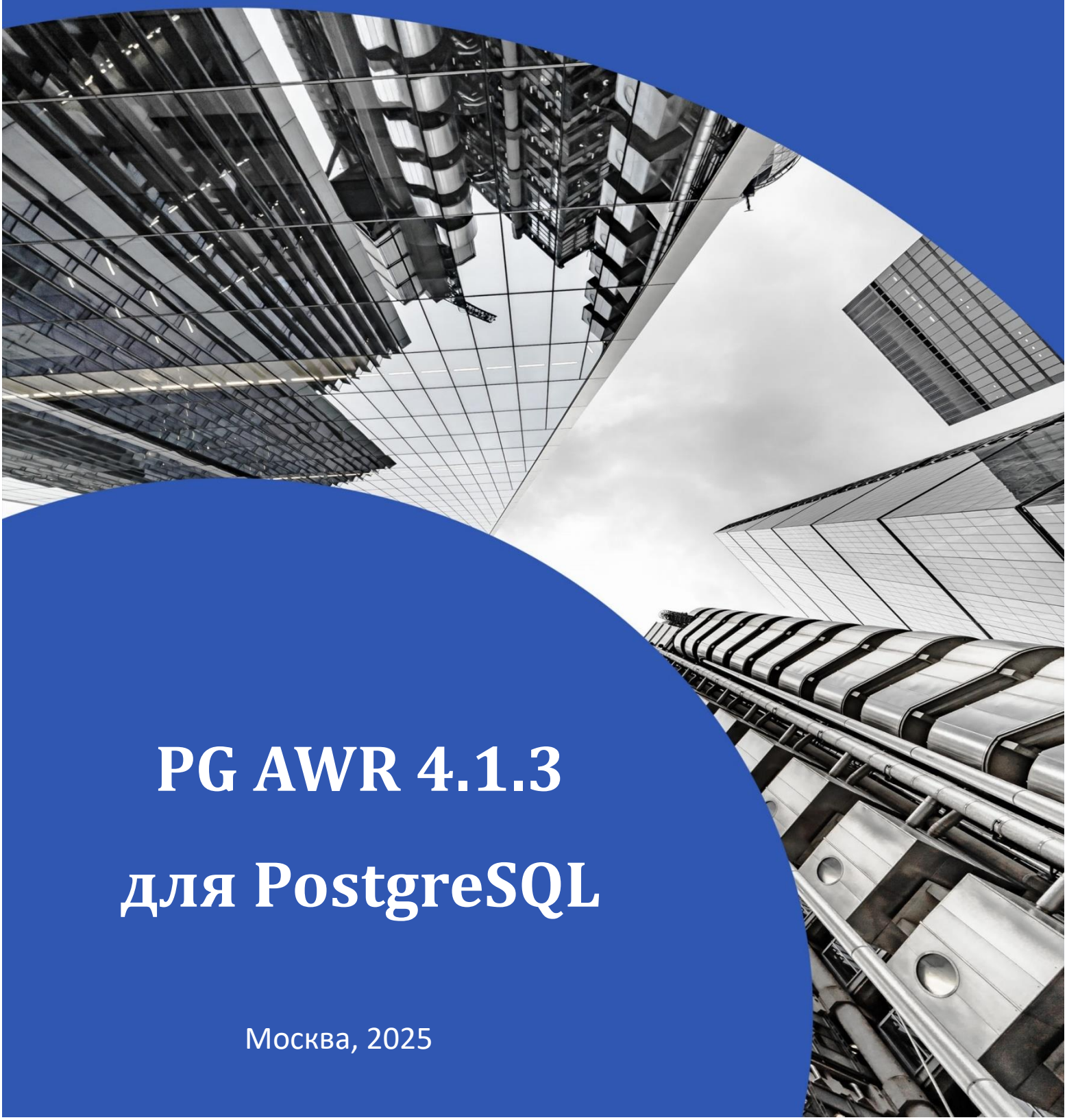
ПАО «Банк ВТБ»
Хаймин В.А.



PG AWR 4.1.3
для PostgreSQL

Москва, 2025

Справочное руководство по PG AWR 4.1.3 для PostgreSQL

1. Оглавление

2. Лицензирование. История версий

Лицензирование.

Проект PG AWR принадлежит ПАО «Банк ВТБ». По всем вопросам использования продукта вне банка вам нужно обратиться к руководству УАБД ПАО «Банк ВТБ»

Версия #	Дата	Авторы и контрибюторы	Комментарий
0.6	28.09.2022	Хаймин Владимир, Кашкаров Павел	Первый выпуск бекенда в промышленную эксплуатацию. В этой версии нет отчета AWR в виде html
0.11	16.11.2022	Хаймин Владимир, Кашкаров Павел, Шабурин Алексей	Релиз с добавленным AWR отчетом. Отчет подходить для бэкенда 0.6 версии
0.15.1	08.02.2023	Хаймин Владимир, Кашкаров Павел, Шабурин Алексей	Оптимизации для хранения pg_stat_statements, дополнительные секции в отчета AWR
0.20.3	19.07.2023	Хаймин Владимир, Кашкаров Павел, Шабурин Алексей, Маслов Михаил	Нормализация хранения pg_stat_statements, дополнительные секции в отчета AWR
1.0.3	05.04.2024	Хаймин Владимир, Кашкаров Павел, Шабурин Алексей, Маслов Михаил, Кузнецов Александр	Дополнение в бекенд анализа по статистике ожиданий, дополнения по активным сессиям, обновление отчета AWR
2.0.3	13.11.2024	Хаймин Владимир, Кашкаров Павел, Шабурин Алексей, Маслов Михаил, Кузнецов Александр, Камаев Алексей	Промежуточный релиз. Рефакторинг бекенда, дополнительные метрики по генерации WAL
3.0.2	10.12.2024	Хаймин Владимир, Кашкаров Павел, Шабурин Алексей, Маслов Михаил, Кузнецов Александр, Камаев Алексей	Реорганизация бекенда. Организация секционирования по датам. Дополнение в мониторинг прогресса процессов вакуум, реиндексации. Дополнения в отчет AWR.
4.1.3	н.в.	Хаймин Владимир, Кашкаров Павел, Шабурин Алексей, Маслов Михаил, Кузнецов Александр, Камаев Алексей, Дубровин Александр, Калинин Сергей	Реорганизация бекенда. Модульность, реализация Active Sessions History, дополнения и исправления в отчете AWR связанные прежде всего с секционированием. Автокастомизация истории pg_stat_statements, Анализ по родительским таблицам секций.

3. Основные ресурсы по поддержке и источники

Если у вас возникли вопросы и замечания по работе АВР вы можете их задать по ссылкам ниже. Также мы рады будем услышать ваше мнение по ошибкам и доработке этого решения.

Описание	**Где найти**
Основная страница по PG AWR, принцип работы, отчеты:	https://sfera.vtb.ru/knowledge/pages?id=108930
Как можно сформировать отчеты. Получение доступа в WatchDog 2.0:	https://sfera.vtb.ru/knowledge/pages?id=108983
Сервис WatchDog-2:	http://watchdog.vtb.ru/
Куда писать по вопросам анализа:	Поддержка PostgreSQL <postgresql@vtb.ru> или обращение в сфере на группу Postgresql-msk2
Дополнительно:	
Исходный проект pg_awr, который использовался в Alibaba Cloud, со значительными концептуальными изменениями.	https://juejin.cn/post/711129011861848067
Основы мониторинга PostgreSQL. Алексей Лесовский / Хабр (habr.com)	https://habr.com/ru/articles/486710/
PostgreSQL : Документация: 14: 28.2. Сборщик статистики : Компания Postgres Professional	https://postgrespro.ru/docs/postgresql/14/monitoring-stats#MONITORING-PG-STAT-SLRU-VIEW:
Как мы мигрировали из Oracle в PostgreSQL. Хаймин Владимир / Хабр (habr.com)	https://habr.com/ru/companies/vtb/articles/819133/
Секционирование в PostgreSQL. Архитектура корзинного хранения данных (Basket partitioning). Хаймин Владимир / Хабр (habr.com)	https://habr.com/ru/companies/vtb/articles/894950/

4. Вступление.

4.1. Причины создания PG AWR. Идеи реализации. Концепция.

Архитектурно **PG AWR** – система исторического мониторинга на основе создания снапшотов стандартных кумулятивных и мгновенных статистических представлений. Она позволяет получить информацию о профиле нагрузки системы, блокировках, и статистике в прошлом. Это бывает полезно для выявления и решения исторических проблем на БД а также для понимания причин их возникновения.

PG AWR – это система, написанная администраторами для администраторов. В ней данные хранятся точно так же как в системных представлениях, а не в каких-то сложных форматах. Вы можете использовать уже готовые запросы для штатных средств мониторинга немного их модифицировав. Как это можно сделать показано в Приложении Б.

PG AWR мы создали в какой-то степени от «безысходности», из-за невозможности отвечать на те вопросы, которые нам задают исходя из имеющихся штатных средств мониторинга PostgreeSQL. А также отсутствия внешних решений, в том числе коммерческих, способных обеспечить решение специфичных задач и особенностей нашей инфраструктуры PostgreSQL.

Но в этой «безысходности» есть и свои больше плюсы, здесь есть поле для деятельности и развития!

Надеюсь, что и вы найдете полезное для своих систем из нашего проекта.

В настоящее время в банке ВТБ эта система развернута на тысячах серверов PostgreSQL. И PG AWR является одной из базовых систем для анализа проблем на БД. При разборе почти любой проблемы всем нужен прежде всего отчет AWR. Фактически в своей инфраструктуре мы имеем уникальную ситуацию – практически все возможные профили нагрузки PostgreSQL. Без этой возможности проект PG AWR конечно не имел бы такого развития и реализации идей и наверняка остановился бы на первой промышленной версии 0.6.

Отчет AWR служит только для первичного анализа. В планарном тексте отчета нельзя много чего отобразить, например, динамику изменения какой-то метрики (хотя частично это функционал есть), или отображать данные многомерно. Более детально проблему можно исследовать прямыми запросами к репозиторию БД или применением других графических средств отображения.

В настоящее время базовые средства мониторинга PostgreSQL основаны на глобальных представлениях, или представлениях на основе расширений pg_stat_statements. Они позволяют видеть ситуацию в PostgreSQL или в моменте (pg_stat_activity, pg_locks), или накопительную статистику, начиная с последнего сброса статистики (pg_stat_statements, pg_stat_databases, pg_stat_all_tables...). Но не позволяют увидеть, что было когда-то на нужный период времени по хронологии нагрузки системы.

В сообществах существует достаточно много коммерческих и свободных проектов по организации исторического мониторинга на основе расширения pg_stat_statements и базовых глобальных представлений. Из некоммерческих наиболее известны pg_profile и pgpro_pwr, как развитие проекта

pg_profile применительно к дистрибутивам Postgres PRO. Из коммерческих я бы отметил Foglight for PostgreSQL.

За основу текущего проекта был взят ранний проект pg_awr, использовавшийся в Alibaba Cloud. Страница автора исходного проекта:

https://juejin.cn/post/7111290118618480670

Проект был переработан. В настоящее время от исходного проекта уже не осталось ничего кроме названия схем __rds_pg_stats__, названий некоторых таблиц, и главного цикла функции snap_global(). Внутри все изменено. В итоге в него заложены следующие основные принципы:

4.2. Концепция PG AWR.

1. Простота в установке, не нуждающаяся в правах суперпользователя операционной системы. Установка и настройка должна выполняться только от пользователя postgres.
2. Разделение backend и отчетной части PG_AWR.
3. Статистика должна хранится не централизованно, а в отдельной схеме в БД postgres. Это позволит избежать сложностей с чрезмерно большим единым хранилищем статистики для десятков тысяч систем суммарным размером десятки и сотни терабайт.
4. Исторические хранилища представления, по которым идет сбор статистики должны иметь все столбцы исходного представления.
5. Планировщик использует для запуска регламентных операций только штатный cron операционной системы под пользователем postgres. Сами регламентные операции выполнены в виде функций. Интервал работы cron является «тактом» для сбора разнопериодических метрик. (одни метрики собираются чаще, другие реже, но единицей измерения этих времен является именно этот «такт» планировщика)
6. Система использует только глобальные представления, без необходимости в использовании Discoverer для баз данных, и построения dblink внутри инстанса.
7. Система после снятия снимков представлений не должна делать сброса статистики (pg_stat_statements_reset()) для того, чтобы не вносить искажений в другие средства мониторинга. Но для освежения таблиц (прежде всего из-за вытеснения) всё же как минимум раз в сутки сбрасывать статистику рекомендуется. В настоящее время это делается автоматически при смене суток.
8. Рекомендуемый интервал сбора статистики 1-5 мин (это чаще чем в Foglight или pg_profile)
9. Представление pg_stat_statement сохраняется в истории не целиком. Но для большей компактности нормализуется по queryid. (В Foglight лимитируется по mean_exec_time с лимитом 1000)
10. В системе должен быть AWR отчет, аналогичный по структуре и оформлению AWR отчету Oracle, с меньшим объемом системной информации, и учитывать возможности PostgreSQL, которых нет в Oracle.

Архитектура бекенда и назначение:

- Структура исторической информации должна быть простой для самостоятельного написания запросов, с тем, чтобы администратор мог оперативно получать необходимую информацию.
- Проект является функциональной заменой в рамках импортозамещения коммерческой системы Foglight, имеющейся в эксплуатации.

5. Принцип работы Workload мониторинга

5.1. Какие есть штатные средства мониторинга PostgreSQL?

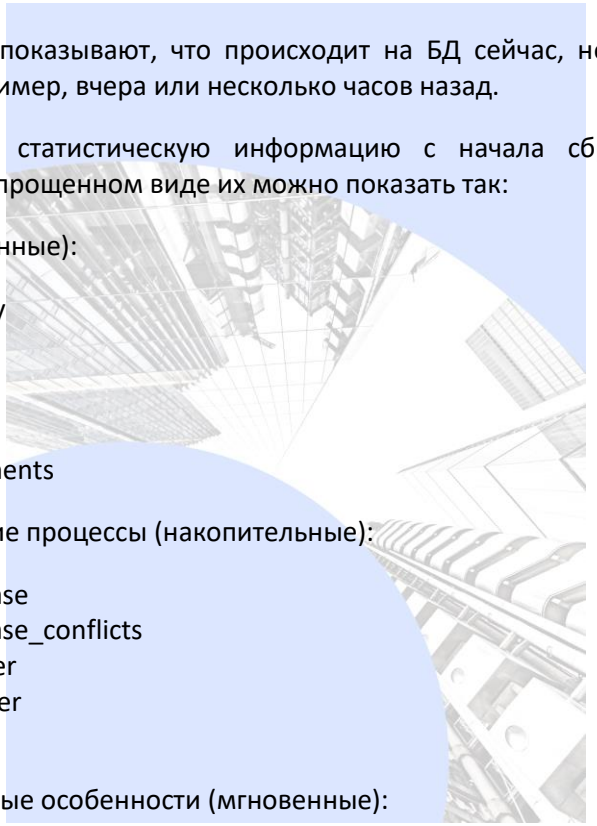
Это набор представлений и функций, позволяющих определять статистику по отдельным компонентам БД и ее функциональным блокам.

Это представления:

- Мгновенные
- Накопительные

Мгновенные представления показывают, что происходит на БД сейчас, но нельзя узнать, что было в системе когда-то ранее. Например, вчера или несколько часов назад.

Накопительные показывают статистическую информацию с начала сброса статистики. Например pg_stat_statement_reset(). В упрощенном виде их можно показать так:



Сессии и блокировки (мгновенные):

- pg_stat_activity
- pg_locks

Запросы (накопительные):

- pg_stat_statements

База данных и обслуживающие процессы (накопительные):

- pg_stat_database
- pg_stat_database_conflicts
- pg_stat_archiver
- pg_stat_bgwriter
- pg_stat_wal

Дополнительные архитектурные особенности (мгновенные):

- pg_stat_replication
- pg_stat_replication_slots
- pg_stat_subscription
- pg_stat_wal_receiver
- pg_stat_ssl
- pg_stat_gssapi

Объекты (накопительные):

- pg_stat_all_tables

- pg_stat_all_indexes
- pg_statio_all_tables
- pg_statio_all_indexes
- pg_statio_all_sequences
- pg_stat_user_functions

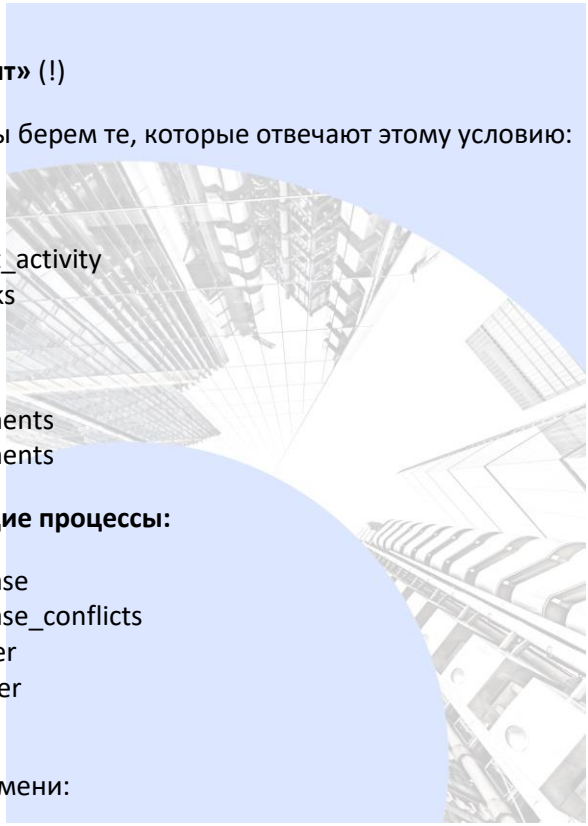
5.2. Принцип устройства backend в системе pg_awr.

Главный вопрос, на который должен отвечать WORKLOAD монитоинг:

«Что происходит в системе»

Но не **«почему это происходит»** (!)

Из штатных представлений мы берем те, которые отвечают этому условию:



Сессии и блокировки:

- pg_stat_activity
- pg_locks

Запросы:

- pg_stat_statements
- pg_stat_statements

База данных и обслуживающие процессы:

- pg_stat_database
- pg_stat_database_conflicts
- pg_stat_archiver
- pg_stat_bgwriter
- pg_stat_wal

И добавляем к ним метку времени:

Метка времени:

Для организации временных рядов представления дополняются информацией по времени данных.

- SNAP_ID – номер снимка
- SNAP_TIME – временная метка, которой соотвествует это время

Для представления pg_stat_database это можно сделать так. Берем абсолютно все поля, которые есть в представлении:

```
create table snap_pg_stat_database as
     select 1::int8 snap_id,
     now()::timestamptz snap_ts, * from pg_stat_database;
```

И создаем индекс для упрощения поиска по номеру снапшота:

```
CREATE INDEX snap_pg_stat_database_snap_id_idx ON snap_pg_stat_database USING btree
(snap_id);
```

Заполнение таблица происходит так (взята строка из задания cron):

```
-- Global, статистика базы данных, доля откатов, время чтения и записи данных с
коэффициентом попадания, взаимоблокировками, конфликтами и т.д.
insert into snap_pg_stat_database select snap_id, ts snap_ts, * from pg_stat_database;
```

Затем, для того чтобы увидеть, что из себя представляло это представление когда-то в прошлом нужно сделать такой запрос:

```
select * from snap_pg_stat_database where snap_id = <номер снапшота>;
```

5.3. Дифференциальная математика для извлечения данных

Чтобы получить статистику по какому-то накопительному представлению, например pg_stat_statements, pg_stat_database мы выполняем простой селект:

```
select * from snap_pg_stat_statements;

select * from snap_pg_stat_database;
```

Но так мы увидим значения только с начала сброса статистики. Как понять, что было именно в интервале времени между сбросом статистики и текущим временем, 1 час назад? Для этого применяется дифференциальный, а не интегральный принцип извлечения данных, когда каждый раз при создании снимка статистика сбрасывается, (например, в pg_profile).

Так как представление накопительное, его значения столбцов всегда увеличиваются. Значит для того чтобы понять, что было в интервале, нужно вычесть из более позднего снимка представления snap_pg_stat_database более ранний:

```
select
	sndb1.datname datname,
	(sndb2.xact_commit-sndb1.xact_commit) xact_commit,
	(sndb2.xact_rollback-sndb1.xact_rollback) xact_rollback,
	(sndb2.blks_read -sndb1.blks_read) blks_read,
	(sndb2.blks_hit -sndb1.blks_hit) blks_hit,
	(sndb2.tup_returned -sndb1.tup_returned) tup_returned,
	(sndb2.tup_fetched -sndb1.tup_fetched) tup_fetched,
	(sndb2.tup_inserted -sndb1.tup_inserted) tup_inserted,
	(sndb2.tup_updated -sndb1.tup_updated) tup_updated,
	(sndb2.tup_deleted -sndb1.tup_deleted) tup_deleted,
	(sndb2.blk_read_time -sndb1.blk_read_time)::int8 blk_read_time,
	(sndb2.blk_write_time -sndb1.blk_write_time)::int8 blk_write_time,
	(sndb2.temp_bytes -sndb1.temp_bytes)::int8 temp_bytes
from
	snap_pg_stat_database sndb1,
```

```
        snap_pg_stat_database sndb2
where
        sndb1.snap_id = [begin_id]          -- начальный снапшот
        and sndb2.snap_id = [end_id]        -- коненый снапшот
        and sndb1.datid = sndb2.datid       -- соединение по равеству БД
order by datname;
```

Для мгновенных представлений - берем их такими какие они есть на определенный момент времени.

Но не все метрики можно получить таким образом. Например, в представлении pg_stat_statements метрики экстремумов min_exec_time и max_exec_time вы можете получить только с начала сброса статистики. Более точно эти значения могут быть получены, если на начальном и конечном снимке pg_stat_statements эти значения изменились. Но такая ситуация возникает далеко не всегда. Тем не менее она хранится в репозитории, и нужно иметь ввиду, что эти метрики показаны с момента сброса статистики, то есть с начала текущих суток.

Значение mean_exec_time фактически пересчитывается по разнице total_exec_time и calls между двумя снимками в репозитории. То есть:

$$mean\_exec\_time = \frac{\Delta total\_exec\_time}{\Delta \text{calls}}$$

5.4. Пример решения проблемы с нагрузкой по ЦПУ.

Предположим к вам обратились с такой проблемой:

<u>ПРОБЛЕМА:</u> Во время нагрузочного тестирования наблюдалась высокая нагрузка ЦПУ. Эта проблема была позавчера во время нагрузочного тестирования с 11-45 до 12 часов. В настоящее время проблема не наблюдается.

<u>Решение.</u>

1. Строим отчет AWR в нужном интервале. Иногда желательно с запасом +- 5-10 мин, чтобы событие в него попало.

1.1. Получаем писок снапшотов:

```
psql -h pgc001lk.test.vtb.ru -U postgres -U postgres -c "select * from
__rds_pg_stats__.snap_list ORDER BY id DESC LIMIT 20 OFFSET 300"
```

```
 10384 | 2023-11-15 12:20:01.306535+03 | global
 10383 | 2023-11-15 12:15:01.824253+03 | global
 10382 | 2023-11-15 12:10:01.290993+03 | global
 10381 | 2023-11-15 12:05:01.74801+03  | global
 10380 | 2023-11-15 12:00:01.957001+03 | global
 10379 | 2023-11-15 11:55:01.446433+03 | global
 10378 | 2023-11-15 11:50:01.964052+03 | global
```

```
 10377 | 2023-11-15 11:45:01.470166+03 | global
 10376 | 2023-11-15 11:40:01.925693+03 | global
```

1.2. Генерируем отчет отчет.

```
psql -t -h pgc001lk.test.vtb.ru -U postgres -U postgres -c "select * from __rds_pg_stats__.snap_report_global(10377,10380)" > ./awr_13.html
```

Далее смотрим отчет в секции:

SQL ordered by CPU Time

- **DB name:** Name of the database sessions is connected to
- **Rolname:** User who executed the statement
- **CPU time,s:** CPU time spent executing the statement, in seconds
- **Executions:** Number of times the statement was executed
- **CPU,%:** CPU usage for this statement, in ptc
- **Avg time,s:** Mean time spent executing the statement, in seconds
- **SQL Id:** Hash code to identify identical normalized queries
- **SQL text:** Text of a representative statement limited by 30 chars

DB name	Rolename	CPU time, s	Executions	CPU,%	Avg time,s	SQL Id	SQL text
subofnsdb	usmzuser	1.421333	1	45	1.421333	-6552365863124210854	delete from public_subofnsdb.history_inc...
subofnsdb	usmzuser	0.388270	7388	12	0.000053	2418178176267029977	select clientmdmc0 .id as id1 6 , client...
subofnsdb	usmzuser	0.329952	7349	10	0.000045	-7508124494977091818	update public_subofnsdb.client_mdm_code ...
subofnsdb	usmzuser	0.193186	7388	6	0.000026	-4463258066288306748	select clientmdmc0_.id as id1_6_0_, clie...
subofnsdb	usmzuser	0.146801	3915	5	0.000037	-8827446471017739436	update public_subofnsdb.client_last_logo...
subofnsdb	usmzuser	0.169673	3980	5	0.000043	-3500619788571542750	select clientlast0_.channel as channel1_...

Как видим из отчета, в топе запрос:

```
delete from
      public_subofnsdb.history_income
where
      created_at=$1;
```

Строим план:

explain (generic_plan) delete from public_subofnsdb.history_income where created_at=$1;

```
Delete on history_income  (cost=0.00..282233.80 rows=0 width=0)
  ->  Seq Scan on history_income  (cost=0.00..282341.80 rows=1013311 width=6)
        Filter: (created_at = $1)
```

По плану видим seqscan по всей таблице history_income

Рекомендация:

Необходим индекс по полю created_at в таблице history_income.

6. Архитектура PG AWR

6.1. Установка PG AWR

Чтобы установить pg_awr воспользуйтесь либо скриптами автоматизации Ansible, либо это можно сделать вручную.

1. Инициализация бекенда:
   ```
   ssh -oStrictHostKeyChecking=no 'TEST\VTB70148294'@lttsec-pg5005lp.test.vtb.ru
   'sudo -u postgres psql' < ./init_pg_awr.sql
   ```

2. Установка отчета:
   ```
   ssh -oStrictHostKeyChecking=no 'TEST\VTB70148294'@lttsec-pg5005lp.test.vtb.ru
   'sudo -u postgres psql' < ./pg_awr_html.sql
   ```

3. Задания cron должны иметь примерно такой вид:

```
# PG_AWR: Global Database snap_global for Stanalong
*/5 * * * * psql -c "select  __rds_pg_stats__.snap_global()" >> /pg_data/pg_awr.log 2>&1
# PG_AWR: Clean old snaps in d snap_delete_obsolete(7 - default) days
5 0 * * *  psql -c "select  __rds_pg_stats__.snap_delete_obsolete(7)" >> /pg_data/pg_awr.log
2>&1
# PG_AWR: date
59 23 * * *  date > /pg_data/pg_awr.log
```

Для автоматизации разворачивания на ПРОМ, ТЕСТ и Standalon вы можете воспользоваться скриптами из папки cron_jobs в дистрибутиве. Но предварительно поправьте в нем логин и пароль.

```
start_PROM.sh

start_STAND.sh

start_TEST.sh
```

Пример использования для Standalong:

```
./start_STAND.sh lttsec-pg5005lp.test.vtb.ru
```

Если у вас кластер, то хост нужно прогнать скрипты по обоим хостам кластера:

```
./start_PROM.sh p0emap-pgc001lk

./start_PROM.sh p0emap-pgc002lk
```

После разворачивания у вас должна появится схема в БД postgres:

```
postgres=# \dn
            List of schemas
       Name         |       Owner
--------------------+--------------------
 __rds_pg_stats__   | postgres
 public             | pg_database_owner
```

Проверьте, работает ли создание снимков:

```
postgres@lttsec-pg5005lp ~ $ psql  -c "select  __rds_pg_stats__.snap_global()" >>
/pg_audit/log/pg_awr/pg_awr.log
NOTICE:  Node role in cluster is replica?: f
NOTICE:  LEADER: start procedure.
NOTICE:  schema_to rds_data_20250410 :
NOTICE:  schema_nm <NULL> :
NOTICE:  schema rds_data_20250410 not found. Try to create objects...
NOTICE:  START: 2025-04-10 15:39:37.215249
NOTICE:  ENABLED: 0-snap_list: 2025-04-10 15:39:37.295982+03
NOTICE:  ENABLED: 4-snap_pg_stat_progress_create_index: 2025-04-10
15:39:37.296538+03
NOTICE:  ENABLED: 4-snap_pg_stat_progress_vacuum: 2025-04-10 15:39:37.297947+03
NOTICE:  ENABLED: 4-snap_pg_stat_progress_cluster: 2025-04-10 15:39:37.298588+03
NOTICE:  ENABLED: 2-snap_pg_db_role_setting: 2025-04-10 15:39:37.299274+03
NOTICE:  ENABLED: 2-snap_pg_tbs_size: 2025-04-10 15:39:37.299543+03
NOTICE:  ENABLED: 2-snap_pg_db_size: 2025-04-10 15:39:37.376459+03
NOTICE:  ENABLED: 4-snap_pg_host_info: 2025-04-10 15:39:37.451109+03
NOTICE:  ENABLED: 1-snap_pg_locks: 2025-04-10 15:39:37.460885+03
NOTICE:  ENABLED: 1-snap_pg_stat_activity: 2025-04-10 15:39:37.46431+03
NOTICE:  ENABLED: 2-snap_blocking_sessions: 2025-04-10 15:39:37.466051+03
NOTICE:  ENABLED: 4-snap_pg_stat_top: 2025-04-10 15:39:37.466985+03
NOTICE:  ENABLED: 4-snap_pg_stat_iotop: 2025-04-10 15:39:37.700703+03
NOTICE:  ENABLED: 2-snap_pg_role_conn_limit: 2025-04-10 15:39:37.771734+03
NOTICE:  ENABLED: 1-snap_pg_stat_database: 2025-04-10 15:39:37.773548+03
NOTICE:  ENABLED: 1-snap_pg_database: 2025-04-10 15:39:37.774516+03
NOTICE:  ENABLED: 1-snap_pg_stat_bgwriter: 2025-04-10 15:39:37.775058+03
NOTICE:  ENABLED: 1-snap_pg_stat_archiver: 2025-04-10 15:39:37.77558+03
NOTICE:  ENABLED: 2-snap_pg_database_age: 2025-04-10 15:39:37.77616+03
NOTICE:  ENABLED: 2-snap_pg_user_deadline: 2025-04-10 15:39:37.776862+03
NOTICE:  ENABLED: 3-snap_pg_stat_statements: 2025-04-10 15:39:37.777638+03
NOTICE:  ENABLED: 2-snap_pg_settings: 2025-04-10 15:39:37.782539+03
NOTICE:  END: 2025-04-10 15:39:37.7
```

После первого запуска должна появиться схема для текущей даты:

```
postgres=# \dn
            List of schemas
       Name         |       Owner
--------------------+-------------------
 __rds_pg_stats__   | postgres
 public             | pg_database_owner
 rds_data_20250410 | postgres
```

Если у вас система работает давно, то таких схем будет больше:

```
postgres=# \dn
            List of schemas
       Name         |       Owner
--------------------+-------------------
 __rds_pg_stats__   | postgres
 public             | pg_database_owner
 rds_data_20250402 | postgres
 rds_data_20250403 | postgres
 rds_data_20250404 | postgres
 rds_data_20250405 | postgres
 rds_data_20250406 | postgres
 rds_data_20250407 | postgres
 rds_data_20250408 | postgres
 rds_data_20250409 | postgres
 rds_data_20250410 | postgres
```

Вы можете удалить данные за конкретную дату через:

```
DROP SCHEMA rds_data_20250407 CASCADE;
```

Процедура snap_delete_obsolete(7) на самом деле по умолчании сохраняет данные за 9 дней. Если вы хотите изменить глубину хранения, по нужно поправить ее здесь в двух местах:

```
-- Удаление устаревших снапшотов старше d дней
CREATE OR REPLACE FUNCTION __rds_pg_stats__.snap_delete_obsolete(d int) RETURNS void as $$
declare
  r record;
  node_role bool;
  schema_nm varchar ='';
  schema_to varchar = '';
  r1 record;
  tmp1 record;
begin
  set search_path=__rds_pg_stats__,public,pg_catalog;
    select into node_role pg_is_in_recovery();
    raise notice 'Delete obsolete schemas.';
    raise notice '============================================';
    raise notice 'Node role in cluster is replica?: %', node_role;
    if node_role = false then
       raise notice 'LEADER: start procedure.';
           -- delete old schemas __rds_stats__, older then 9 days
           raise notice '--- Global schemas ---';
           raise notice 'Pg_sleep 30 sec....';
            select into tmp1 pg_sleep(30);
           select into schema_to '__rds_pg_stats__'||TO_CHAR (now()- interval '9 days',
'YYYYMMDD_HHMMSS');
           for r1 in
             select schema_name FROM information_schema.schemata WHERE schema_name like
'__rds_pg_stats__2%'
           loop
               raise notice '> % : ',r1.schema_name;
               if r1.schema_name < schema_to then
                    raise notice 'delete schema % : ',r1.schema_name;
                     execute 'drop schema ' || r1.schema_name || ' cascade;';
               end if;
           end loop;
           -- delete old schemas rds_data_, older then 9 days
           raise notice '--- Data schemas ---';
           select into schema_to 'rds_data_'||TO_CHAR (now()- interval '9 days',
'YYYYMMDD_HHMMSS');
           for r1 in
             select schema_name FROM information_schema.schemata WHERE schema_name like
'rds_data_2%'
           loop
               raise notice '> % : ',r1.schema_name;
               if r1.schema_name < schema_to then
                    raise notice 'delete schema % : ',r1.schema_name;
                     execute 'drop schema ' || r1.schema_name || ' cascade;';
               end if;
           end loop;
      else
             raise notice 'REPLICA: skip procedure.';
    end if;
```

```
    raise notice 'Delete obsolete schemas finished.';
    raise notice '============================================';
  reset search_path;
end;
$$ language plpgsql strict;
```

6.2. Структура бекенда PG AWR. Таблицы и представления.

6.2.1. Секционирование и схемы бекенда.

Бекенденд использует архитектуру «Корзинного секционирования». Более подробно вы можете ознакомиться с тем, как это работает здесь:

Секционирование в PostgreSQL. Архитектура корзинного хранения данных (Basket partitioning). Хаймин Владимир / Хабр (habr.com) https://habr.com/ru/companies/vtb/articles/894950/

С точки зрения использования, в БД postgres есть такие схемы:

```
postgres=# \dn
            List of schemas
       Name         |       Owner
--------------------+--------------------
 __rds_pg_stats__   | postgres
 public             | pg_database_owner
 rds_data_20250402  | postgres
 rds_data_20250403  | postgres
 rds_data_20250404  | postgres
 rds_data_20250405  | postgres
 rds_data_20250406  | postgres
 rds_data_20250407  | postgres
 rds_data_20250408  | postgres
 rds_data_20250409  | postgres
 rds_data_20250410  | postgres
```

Вы можете удалить данные за конкретную дату через команду:

```
DROP SCHEMA rds_data_20250407 CASCADE;
```

Эта операция происходит очень быстро, и не требует обслуживания. Данные в таблице секций всегда растут.

Но при этом обращаться к таблицам нужно через схему __rds_pg_stats__. Например, так можно получить список последних 20 снимков:

```
select * from __rds_pg_stats__.snap_list
       order by id desc
       limit 20;
```

6.2.2. snap_list

Имя таблицы или представления	Тип	Описание
snap_list	Секционированная таблица	Список снимков бекенда. В комментарии хранится версия pg_awr

Структура таблицы:

```
CREATE TABLE "__rds_pg_stats__".snap_list (
	id bigserial NOT NULL,
	snap_ts timestamptz NULL,
	snap_level text NULL
)
PARTITION BY RANGE (snap_ts);
CREATE INDEX snap_list_id_idx ON ONLY __rds_pg_stats__.snap_list USING btree (id);
```

Номера снимков этой таблицы id совпадают с номерами таблиц других таблиц по полю snap_id.

6.2.3. snap_pg_stat_database

Имя таблицы или представления	Тип	Описание
snap_pg_stat_database	Секционированная таблица	Снимки представления pg_stat_database

Структура таблицы:

```
CREATE TABLE "__rds_pg_stats__".snap_pg_database (
	snap_id int8 NULL,
	snap_ts timestamptz NULL,
	"oid" oid NULL,
	datname name NULL COLLATE "C",
	datdba oid NULL,
	"encoding" int4 NULL,
	datlocprovider char(1) NULL,
	datistemplate bool NULL,
	datallowconn bool NULL,
	datconnlimit int4 NULL,
	datfrozenxid xid NULL,
	datminmxid xid NULL,
	dattablespace oid NULL,
	datcollate text NULL,
	datctype text NULL,
	daticulocale text NULL,
	datcollversion text NULL,
	datacl _aclitem NULL
)
PARTITION BY RANGE (snap_ts);
CREATE INDEX snap_pg_database_snap_id_idx ON ONLY __rds_pg_stats__.snap_pg_database USING
btree (snap_id);
```

Именно с этой таблицей есть особенности при обновлении на PostgreSQL версии 16. См. Приложение А.

Соответствует описанию представления pg_stat_database в документации PostgreSQL.

Таблица динамичеd2ская, формируется со всеми столбцами представления, которые есть в вашей версии БД PostgreSQL

6.2.4. snap_pg_database_age

Имя таблицы или представления	Тип	Описание
snap_pg_stat_database_age	Секционированная таблица	Таблица, определяемая запросом. Возраст баз данных и номер транзакции в связке с временем.

Структура таблицы:

```
CREATE TABLE "__rds_pg_stats__".snap_pg_database_age (
	snap_id int8 NULL,
	snap_ts timestamptz NULL,
	txid_current int8 NULL,
	datname name NULL COLLATE "C",
	age int4 NULL,
	age_remain float8 NULL
)
PARTITION BY RANGE (snap_ts);
CREATE INDEX snap_pg_database_age_snap_id_idx ON ONLY __rds_pg_stats__.snap_pg_database_age
USING btree (snap_id);
```

Основное назначение этой таблицы – следить за динамикой возраста БД.

В таблице есть также дополнительное поле txid_current которое может быть полезно, например, для приблизительного определения времени создания конкретной строки в таблице по xmin.

Например, вам нужно узнать какие строки были созданы/изменены после времени, соответствующего номеру снимка snap_id = 11436:

Время, которому соответствует этот номер снимка:

```
select * from "__rds_pg_stats__".snap_list where id=11436;
```

11436 2025-04-09 00:20:01 global

Найдем номер транзакции, которому соответствует это время:

```
select txid_current from "__rds_pg_stats__".snap_pg_database_age spda where snap_id = 11436
limit 1;
```

870510790

Соответственно строки таблицы, которые были созданы/изменены после этого номера времени:

```
select xmin, * from table1 where xmin > 870510790;
```

Запрос, которым определяется таблица:

```
create table snap_pg_database_age as
select
	1::int8 snap_id,
	now()::timestamptz snap_ts,
	txid_current() txid_current, -- номер текущей транзакции
	datname,
	age(datfrozenxid),
	2^31-age(datfrozenxid) age_remain
from
	pg_database
order by
	age(datfrozenxid) desc;
```

6.2.5. snap_pg_db_role_setting

Имя таблицы или представления	Тип	Описание
snap_pg_db_role_setting	Секционированная таблица	Снимки представления pg_db_role_setting.

Структура таблицы:

```
CREATE TABLE "__rds_pg_stats__".snap_pg_db_role_setting (
	snap_id int8 NULL,
	snap_ts timestamptz NULL,
	setdatabase oid NULL,
	setrole oid NULL,
	setconfig _text NULL
)
PARTITION BY RANGE (snap_ts);
CREATE INDEX snap_pg_db_role_setting_snap_id_idx ON ONLY
__rds_pg_stats__.snap_pg_db_role_setting USING btree (snap_id);
```

Соответствует описанию представления pg_db_role_setting в документации PostgreSQL.

Таблица динамическая, формируется со всеми столбцами представления, которые есть в вашей версии БД PostgreSQL.

6.2.6. snap_pg_db_size

Имя таблицы или представления	Тип	Описание

snap_pg_db_size	Секционированная таблица	Таблица, определяемая запросом. Динамика размеров баз данных.

Структура таблицы:

```
CREATE TABLE "__rds_pg_stats__".snap_pg_db_size (
	snap_id int8 NULL,
	snap_ts timestamptz NULL,
	datname name NULL COLLATE "C",
	pg_size_pretty text NULL,
	pg_size int8 NULL
)
PARTITION BY RANGE (snap_ts);
CREATE INDEX snap_pg_db_size_snap_id_idx ON ONLY __rds_pg_stats__.snap_pg_db_size USING
btree (snap_id);
CREATE INDEX snap_pg_locks_snap_id_idx ON ONLY __rds_pg_stats__.snap_pg_db_size USING btree
(snap_id);
```

Запрос, которым определяется таблица:

```
create table snap_pg_db_size as
select
	1::int8 snap_id,
	now()::timestamptz snap_ts,
	datname,
	pg_size_pretty(pg_database_size(oid)),
	pg_database_size(oid) as pg_size
from
	pg_database
order by
	pg_database_size(oid) desc;
```

6.2.7. snap_blocking_sessions

Имя таблицы или представления	Тип	Описание
snap_pg_db_size	Секционированная таблица	Таблица, определяемая запросом. История блокировок.

Структура таблицы:

```
CREATE TABLE "__rds_pg_stats__".snap_blocking_sessions (
	snap_id int8 NULL,
	snap_ts timestamptz NULL,
	blocked_by _int4 NULL,
	pid int4 NULL,
	usename name NULL COLLATE "C",
	blocked_query text NULL
)
```

```
PARTITION BY RANGE (snap_ts);
CREATE INDEX snap_blocking_sessions_snap_id_idx ON ONLY
__rds_pg_stats__.snap_blocking_sessions USING btree (snap_id);
```

Запрос, которым определяется таблица:

```
create table snap_blocking_sessions as
select
	1::int8 snap_id,
	now()::timestamptz snap_ts,
	pg_blocking_pids(pid) as blocked_by,
	pid,
	usename,
	query as blocked_query
from
	pg_stat_activity
where
	cardinality(pg_blocking_pids(pid)) > 0;
```

6.2.8. snap_pg_locks

Имя таблицы или представления	Тип	Описание
snap_pg_locks	Секционированная таблица	Снимки представления pg_locks. Блокировки.

Структура таблицы:

```
CREATE TABLE "__rds_pg_stats__".snap_pg_locks (
	snap_id int8 NULL,
	snap_ts timestamptz NULL,
	locktype text NULL,
	"database" oid NULL,
	relation oid NULL,
	page int4 NULL,
	tuple int2 NULL,
	virtualxid text NULL,
	transactionid xid NULL,
	classid oid NULL,
	objid oid NULL,
	objsubid int2 NULL,
	virtualtransaction text NULL,
	pid int4 NULL,
	"mode" text NULL,
	"granted" bool NULL,
	fastpath bool NULL,
	waitstart timestamptz NULL
)
PARTITION BY RANGE (snap_ts);
```

Соответствует описанию представления pg_locks в документации PostgreSQL.

Таблица динамическая, формируется со всеми столбцами представления, которые есть в вашей версии БД PostgreSQL.

6.2.9. snap_pg_role_conn_limit

Имя таблицы или представления	Тип	Описание
snap_pg_role_conn_limit	Секционированная таблица	Таблица, определяемая запросом. Лимиты по ролям.

Структура таблицы:

```
CREATE TABLE "__rds_pg_stats__".snap_pg_role_conn_limit (
	snap_id int8 NULL,
	snap_ts timestamptz NULL,
	rolname name NULL COLLATE "C",
	rolconnlimit int4 NULL,
	connects int8 NULL
)
PARTITION BY RANGE (snap_ts);
CREATE INDEX snap_pg_role_conn_limit_snap_id_idx ON ONLY
__rds_pg_stats__.snap_pg_role_conn_limit USING btree (snap_id);
```

Запрос, которым определяется таблица:

```
create table snap_pg_role_conn_limit as
select
	1::int8 snap_id,
	now()::timestamptz snap_ts,
	a.rolname,
	a.rolconnlimit,
	b.connects
from
	pg_roles a,
	(
	select
		usename,
		count(*) connects
	from
		pg_stat_activity
	group by
		usename) b
where
	a.rolname = b.usename
order by
	b.connects desc;
```

6.2.10. snap_pg_settings

Имя таблицы или представления	Тип	Описание
snap_pg_settings	Секционированная таблица	Снимки представления pg_settings.

Структура таблицы:

```
CREATE TABLE "__rds_pg_stats__".snap_pg_settings (
	snap_id int8 NULL,
	snap_ts timestamptz NULL,
	"name" text NULL,
	setting text NULL,
	unit text NULL,
	category text NULL,
	short_desc text NULL,
	extra_desc text NULL,
	context text NULL,
	vartype text NULL,
	"source" text NULL,
	min_val text NULL,
	max_val text NULL,
	enumvals _text NULL,
	boot_val text NULL,
	reset_val text NULL,
	sourcefile text NULL,
	sourceline int4 NULL,
	pending_restart bool NULL
)
PARTITION BY RANGE (snap_ts);
CREATE INDEX snap_pg_settings_snap_id_idx ON ONLY __rds_pg_stats__.snap_pg_settings USING
btree (snap_id);
```

Соответствует описанию представления pg_settings в документации PostgreSQL.

Таблица динамическая, формируется со всеми столбцами представления, которые есть в вашей версии БД PostgreSQL.

6.2.11. snap_pg_stat_activity, snap_pg_stat_top, snap_pg_stat_iotop, snap_pg_stat_activity_v (компоненты Active Session History ASH)

Имя таблицы или представления	Тип	Описание
snap_pg_stat_activity	Секционированная таблица	Снимки представления pg_stat_activity.
snap_stat_top	Секционированная таблица	Результат выполнения команды top по пользователям postgres

snap_stat_iotop	Секционированная таблица	Результаь вывода по IO процессов postgres из /proc/[PID]/io
snap_pg_stat_activity	Представление	Объединение по snap_pg_stat_activity, snap_pg_stat_top, snap_pg_stat_iotop

Структура таблицы snap_pg_stat_activity:

```
CREATE TABLE "__rds_pg_stats__".snap_pg_stat_activity (
	snap_id int8 NULL,
	snap_ts timestamptz NULL,
	datid oid NULL,
	datname name NULL COLLATE "C",
	pid int4 NULL,
	leader_pid int4 NULL,
	usesysid oid NULL,
	usename name NULL COLLATE "C",
	application_name text NULL,
	client_addr inet NULL,
	client_hostname text NULL,
	client_port int4 NULL,
	backend_start timestamptz NULL,
	xact_start timestamptz NULL,
	query_start timestamptz NULL,
	state_change timestamptz NULL,
	wait_event_type text NULL,
	wait_event text NULL,
	state text NULL,
	backend_xid xid NULL,
	backend_xmin xid NULL,
	query_id int8 NULL,
	query text NULL,
	backend_type text NULL
)
PARTITION BY RANGE (snap_ts);
CREATE INDEX snap_pg_stat_activity_pid_snap_id_idx ON ONLY
__rds_pg_stats__.snap_pg_stat_activity USING btree (pid, snap_id);
CREATE INDEX snap_pg_stat_activity_snap_id_idx ON ONLY
__rds_pg_stats__.snap_pg_stat_activity USING btree (snap_id);
```

Соответствует описанию представления pg_stat_activity в документации PostgreSQL.

Таблица динамическая, формируется со всеми столбцами представления, которые есть в вашей версии БД PostgreSQL.

Структура таблицы snap_pg_stat_top:

```
CREATE TABLE "__rds_pg_stats__".snap_pg_stat_top (
	snap_id int8 NULL,
	snap_ts timestamptz NULL,
	pid bpchar(20) NULL,
	usr bpchar(40) NULL,
	pr int8 NULL,
	ni int8 NULL,
```

```
	virt bpchar(80) NULL,
	res bpchar(80) NULL,
	shr bpchar(80) NULL,
	s bpchar(2) NULL,
	cpu bpchar(10) NULL,
	mem bpchar(10) NULL,
	tme bpchar(10) NULL,
	cmd bpchar(200) NULL
)
PARTITION BY RANGE (snap_ts);
CREATE INDEX snap_pg_stat_top_cpu_snap_id_idx ON ONLY __rds_pg_stats__.snap_pg_stat_top
USING btree (((cpu)::double precision) DESC, snap_id);
CREATE INDEX snap_pg_stat_top_mem_snap_id_idx ON ONLY __rds_pg_stats__.snap_pg_stat_top
USING btree (((mem)::double precision) DESC, snap_id);
CREATE INDEX snap_pg_stat_top_pid_snap_id_idx ON ONLY __rds_pg_stats__.snap_pg_stat_top
USING btree (((pid)::integer), snap_id);
CREATE INDEX snap_pg_stat_top_snap_id_idx ON ONLY __rds_pg_stats__.snap_pg_stat_top USING
btree (snap_id);
```

Результат выполнения команды:

```
top -b -n 1| grep postgres|sed -e 's/^ *//' |tr -s ' '|sed -e 's/\s/|/g'
```

Структура таблицы snap_pg_stat_top:

```
CREATE TABLE "__rds_pg_stats__".snap_pg_stat_iotop (
	snap_id int8 NULL,
	snap_ts timestamptz NULL,
	pid bpchar(20) NULL,
	rchar int8 NULL,
	wchar int8 NULL,
	syscr int8 NULL,
	syscw int8 NULL,
	read_bytes int8 NULL,
	write_bytes int8 NULL,
	cancelled_write_bytes int8 NULL
)
PARTITION BY RANGE (snap_ts);
CREATE INDEX snap_pg_stat_iotop_pid_read_bytes_idx ON ONLY
__rds_pg_stats__.snap_pg_stat_iotop USING btree (read_bytes);
CREATE INDEX snap_pg_stat_iotop_pid_snap_id_idx ON ONLY __rds_pg_stats__.snap_pg_stat_iotop
USING btree (((pid)::integer), snap_id);
CREATE INDEX snap_pg_stat_iotop_pid_write_bytes_idx ON ONLY
__rds_pg_stats__.snap_pg_stat_iotop USING btree (write_bytes);
CREATE INDEX snap_pg_stat_iotop_snap_id_idx ON ONLY __rds_pg_stats__.snap_pg_stat_iotop USING
btree (snap_id);
```

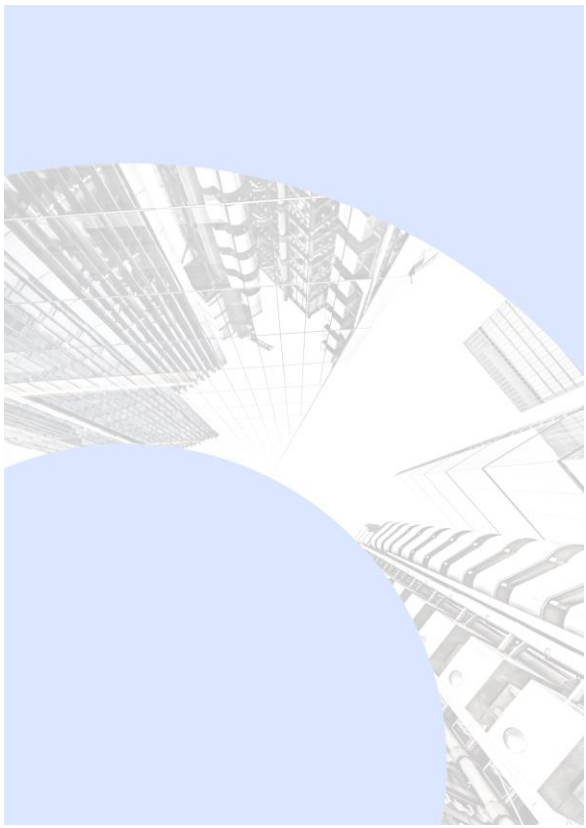
Результат выполнения команды:

```
ps -e | grep -w 'postgres\|postmaster'|awk '{print "+" $1 " "; system("cat /proc/"$1"/io")}'| tr -s "\n" "|" | tr -s
"+" "\n"| tr -d "[a-z],_,:, "
```

Структура представления snap_pg_stat_activity.

```
CREATE OR REPLACE VIEW "__rds_pg_stats__".snap_pg_stat_activity_v
```

```
AS SELECT pa.snap_id,
    pa.snap_ts,
    pa.datid,
    pa.datname,
    pa.pid,
    pa.leader_pid,
    pa.usesysid,
    pa.usename,
    pa.application_name,
    pa.client_addr,
    pa.client_hostname,
    pa.client_port,
    pa.backend_start,
    pa.xact_start,
    pa.query_start,
    pa.state_change,
    pa.wait_event_type,
    pa.wait_event,
    pa.state,
    pa.backend_xid,
    pa.backend_xmin,
    pa.query_id,
    pa.query,
    pa.backend_type,
    pt.usr,
    pt.pr,
    pt.ni,
    pt.virt,
    pt.res,
    pt.shr,
    pt.s,
    pt.cpu,
    pt.mem,
    pt.tme,
    pt.cmd,
    piot.rchar,
    piot.wchar,
    piot.syscr,
    piot.syscw,
    piot.read_bytes,
    piot.write_bytes,
    piot.cancelled_write_bytes
   FROM __rds_pg_stats__.snap_pg_stat_activity pa
     JOIN __rds_pg_stats__.snap_pg_stat_top pt ON pa.snap_id = pt.snap_id AND pa.pid =
pt.pid::integer
     JOIN __rds_pg_stats__.snap_pg_stat_iotop piot ON pa.snap_id = piot.snap_id AND pa.pid =
piot.pid::integer;
```



Фактически это представление – расширяет pg_stat_activity, добавляя в него инфомацию по ОЗУ, Памяти и IO процессом.

Снимки этих представлений в процедуре должны находится максимально близко друг к другу. Очень редко могут возникать ситуации, когда снимок по конкретному PID по pg_stat_activity есть, а по pg_stat_top и

pg_stat_iotop сняться уже не успел, т.к. процесса не существует. Но это специфика дискретности архитектуры при почти недискретной природе реальных процессов в системе.

6.2.12. snap_pg_stat_archiver

Имя таблицы или представления	Тип	Описание
snap_pg_stat_archiver	Секционированная таблица	Снимки представления pg_stat_archiver.

Структура таблицы:

```
CREATE TABLE "__rds_pg_stats__".snap_pg_stat_archiver (
	snap_id int8 NULL,
	snap_ts timestamptz NULL,
	now_insert_xlog_file text NULL,
	pg_current_wal_insert_lsn pg_lsn NULL,
	archived_count int8 NULL,
	last_archived_wal text NULL,
	last_archived_time timestamptz NULL,
	failed_count int8 NULL,
	last_failed_wal text NULL,
	last_failed_time timestamptz NULL,
	stats_reset timestamptz NULL
)
PARTITION BY RANGE (snap_ts);
CREATE INDEX snap_pg_stat_archiver_snap_id_idx ON ONLY
__rds_pg_stats__.snap_pg_stat_archiver USING btree (snap_id);
```

Соответствует описанию представления pg_stat_archiver в документации PostgreSQL.

Таблица динамическая, формируется со всеми столбцами представления, которые есть в вашей версии БД PostgreSQL.

6.2.13. snap_pg_stat_bgwriter

Имя таблицы или представления	Тип	Описание
snap_pg_stat_archiver	Секционированная таблица	Снимки представления pg_stat_bgwriter.

Структура таблицы:

```
CREATE TABLE "__rds_pg_stats__".snap_pg_stat_bgwriter (
	snap_id int8 NULL,
	snap_ts timestamptz NULL,
	checkpoints_timed int8 NULL,
	checkpoints_req int8 NULL,
	checkpoint_write_time float8 NULL,
```

```
	checkpoint_sync_time float8 NULL,
	buffers_checkpoint int8 NULL,
	buffers_clean int8 NULL,
	maxwritten_clean int8 NULL,
	buffers_backend int8 NULL,
	buffers_backend_fsync int8 NULL,
	buffers_alloc int8 NULL,
	stats_reset timestamptz NULL
)
PARTITION BY RANGE (snap_ts);
CREATE INDEX snap_pg_stat_bgwriter_snap_id_idx ON ONLY
__rds_pg_stats__.snap_pg_stat_bgwriter USING btree (snap_id);
```

Соответствует описанию представления pg_stat_bgwriter в документации PostgreSQL.

Таблица динамическая, формируется со всеми столбцами представления, которые есть в вашей версии БД PostgreSQL.

6.2.14. snap_pg_stat_database

Имя таблицы или представления	Тип	Описание
snap_pg_stat_database	Секционированная таблица	Снимки представления pg_stat_database.

Структура таблицы:

```
CREATE TABLE "__rds_pg_stats__".snap_pg_stat_database (
	snap_id int8 NULL,
	snap_ts timestamptz NULL,
	datid oid NULL,
	datname name NULL COLLATE "C",
	numbackends int4 NULL,
	xact_commit int8 NULL,
	xact_rollback int8 NULL,
	blks_read int8 NULL,
	blks_hit int8 NULL,
	tup_returned int8 NULL,
	tup_fetched int8 NULL,
	tup_inserted int8 NULL,
	tup_updated int8 NULL,
	tup_deleted int8 NULL,
	conflicts int8 NULL,
	temp_files int8 NULL,
	temp_bytes int8 NULL,
	deadlocks int8 NULL,
	checksum_failures int8 NULL,
	checksum_last_failure timestamptz NULL,
	blk_read_time float8 NULL,
	blk_write_time float8 NULL,
	session_time float8 NULL,
	active_time float8 NULL,
```

```
	idle_in_transaction_time float8 NULL,
	sessions int8 NULL,
	sessions_abandoned int8 NULL,
	sessions_fatal int8 NULL,
	sessions_killed int8 NULL,
	stats_reset timestamptz NULL
)
PARTITION BY RANGE (snap_ts);
CREATE INDEX snap_pg_stat_database_snap_id_idx ON ONLY
__rds_pg_stats__.snap_pg_stat_database USING btree (snap_id);
```

Соответствует описанию представления pg_stat_database в документации PostgreSQL.

Таблица динамическая, формируется со всеми столбцами представления, которые есть в вашей версии БД PostgreSQL.

6.2.15. snap_pg_stat_progress_cluster

Имя таблицы или представления	Тип	Описание
snap_ pg_stat_progress_cluster	Секционированная таблица	Снимки представления pg_stat_progress_cluster. Прогресс команд CLUSTER и VACUUM FULL в моменте.

Структура таблицы:

```
CREATE TABLE "__rds_pg_stats__".snap_pg_stat_progress_cluster (
	snap_id int8 NULL,
	snap_ts timestamptz NULL,
	pid int4 NULL,
	datid oid NULL,
	datname name NULL COLLATE "C",
	relid oid NULL,
	command text NULL,
	phase text NULL,
	cluster_index_relid oid NULL,
	heap_tuples_scanned int8 NULL,
	heap_tuples_written int8 NULL,
	heap_blks_total int8 NULL,
	heap_blks_scanned int8 NULL,
	index_rebuild_count int8 NULL
)
PARTITION BY RANGE (snap_ts);
CREATE INDEX snap_pg_stat_progress_cluster_idx ON ONLY
__rds_pg_stats__.snap_pg_stat_progress_cluster USING btree (snap_id);
```

Соответствует описанию представления pg_stat_progress_cluster в документации PostgreSQL.

Таблица динамическая, формируется со всеми столбцами представления, которые есть в вашей версии БД PostgreSQL.

6.2.16. snap_pg_stat_progress_create_index

Имя таблицы или представления	Тип	Описание
snap_ pg_stat_progress_create_index	Секционированная таблица	Снимки представления pg_stat_progress_create_index. Прогресс команд CREATE INDEX ...

Структура таблицы:

```
CREATE TABLE "__rds_pg_stats__".snap_pg_stat_progress_create_index (
	snap_id int8 NULL,
	snap_ts timestamptz NULL,
	pid int4 NULL,
	datid oid NULL,
	datname name NULL COLLATE "C",
	relid oid NULL,
	index_relid oid NULL,
	command text NULL,
	phase text NULL,
	lockers_total int8 NULL,
	lockers_done int8 NULL,
	current_locker_pid int8 NULL,
	blocks_total int8 NULL,
	blocks_done int8 NULL,
	tuples_total int8 NULL,
	tuples_done int8 NULL,
	partitions_total int8 NULL,
	partitions_done int8 NULL
)
PARTITION BY RANGE (snap_ts);
```

Соответствует описанию представления pg_stat_progress_create_index в документации PostgreSQL.

Таблица динамическая, формируется со всеми столбцами представления, которые есть в вашей версии БД PostgreSQL.

6.2.17. snap_pg_stat_progress_vacuum

Имя таблицы или представления	Тип	Описание
snap_ pg_stat_progress_vacuum	Секционированная таблица	Снимки представления pg_stat_progress_vacuum. Прогресс команд VACUUM...

Структура таблицы:

```
CREATE TABLE "__rds_pg_stats__".snap_pg_stat_progress_vacuum (
	snap_id int8 NULL,
	snap_ts timestamptz NULL,
```

```
	pid int4 NULL,
	datid oid NULL,
	datname name NULL COLLATE "C",
	relid oid NULL,
	phase text NULL,
	heap_blks_total int8 NULL,
	heap_blks_scanned int8 NULL,
	heap_blks_vacuumed int8 NULL,
	index_vacuum_count int8 NULL,
	max_dead_tuples int8 NULL,
	num_dead_tuples int8 NULL
)
PARTITION BY RANGE (snap_ts);
CREATE INDEX snap_pg_stat_progress_vacuum_idx ON ONLY
__rds_pg_stats__.snap_pg_stat_progress_vacuum USING btree (snap_id);
```

Соответствует описанию представления pg_stat_progress_vacuum в документации PostgreSQL.

Таблица динамическая, формируется со всеми столбцами представления, которые есть в вашей версии БД PostgreSQL.

6.2.18. snap_pg_tbs_size

Имя таблицы или представления	Тип	Описание
snap_pg_tbs_size	Секционированная таблица	Таблица, определяемая запросом. Размеры табличных пространств.

Структура таблицы:

```
CREATE TABLE "__rds_pg_stats__".snap_pg_tbs_size (
	snap_id int8 NULL,
	snap_ts timestamptz NULL,
	spcname name NULL COLLATE "C",
	pg_tablespace_location text NULL,
	pg_size_pretty text NULL,
	pg_tablespace_size int8 NULL
)
PARTITION BY RANGE (snap_ts);
CREATE INDEX snap_pg_tbs_size_snap_id_idx ON ONLY __rds_pg_stats__.snap_pg_tbs_size USING
btree (snap_id);
```

Запрос, которым определяется таблица:

```
create table snap_pg_tbs_size as
select
	1::int8 snap_id,
	now()::timestamptz snap_ts,
	spcname,
	pg_tablespace_location(oid),
	pg_size_pretty(pg_tablespace_size(oid)),
	pg_tablespace_size(oid) as pg_tablespace_size
```

```
from
	pg_tablespace
where
	spcname <> 'pg_global'
order by
	pg_tablespace_size(oid) desc;
```

6.2.19. snap_pg_user_deadline

Имя таблицы или представления	Тип	Описание
snap_pg_user_deadline	Секционированная таблица	Таблица, определяемая запросом. Ограничение по ролям.

Структура таблицы:

```
CREATE TABLE "__rds_pg_stats__".snap_pg_user_deadline (
	snap_id int8 NULL,
	snap_ts timestamptz NULL,
	rolname name NULL COLLATE "C",
	rolvaliduntil timestamptz NULL
)
PARTITION BY RANGE (snap_ts);
CREATE INDEX snap_pg_user_deadline_snap_id_idx ON ONLY
__rds_pg_stats__.snap_pg_user_deadline USING btree (snap_id);
```

Запрос, которым определяется таблица:

```
create table snap_pg_user_deadline as
select
	1::int8 snap_id,
	now()::timestamptz snap_ts,
	rolname,
	rolvaliduntil
from
	pg_roles
order by
	rolvaliduntil;
```

6.2.20. snap_pg_stat_statements_short, snap_pg_stat_sqltext, snap_pg_stat_statemetns

Имя таблицы или представления	Тип	Описание
snap_pg_stat_statements_short	Секционированная таблица	Таблица, определяемая запросом. Мгновенный снимок представления pg_stat_statements без поля query (sqltext)

snap_pg_stat_sqltext	Секционированная таблица	Таблица, определяемая запросом. Выделенные значения с queryid и query (sqltext). Если в комментарии есть текст Custom , это говорит о том, что бекенд сформирован из init_pg_awr_custom.sql
snap_pg_stat_statements	Представление	Объединение таблиц snap_pg_stat_statements_short и snap_pg_stat_sqltext

Для более компактного хранения снимков pg_stat_statements текст запроса выделен в отдельную таблицу, содержащую queryid и query (sqltext). Текст запроса находится в поле типа text, имеющего ограничение 1 Гб. Поэтому данные pg_stat_statements в планарном виде могут занимать существенные объемы вплоть до до сброса статистики или вытеснения, и может сохранятся значительное время. Кроме того, значение по умолчании параметра pg_stat_statements.max = 5000. То есть теоретически представление может вместить в себя около 5 Тб. И все эти снимки мы должны сохранять в Бекенде PG AWR. Выделение текста запроса в отдельную таблицу уменьшает вероятность переполнения статистических таблиц, но эта вероятность все же сохраняется. Одним из дополнительных методов по уменьшению размеров данных может быть урезание текстов запроса по длине. Именно для этого в частности нужен init_pg_awr_custom.sql

Функционале назначение объектов:

- snap_pg_stat_statements_short – это содержимое представления pg_stat_statements со всеми полями, но без query
- snap_pg_stat_sqltext – содержит в уникальном виде queryid и query.
- snap_pg_stat_statemetns – представление, которое объединяет snap_pg_stat_statements_short, snap_pg_stat_sqltext по query id, таким образом, чтобы одновременно был вид вместе с текстом запроса всего представления pg_stat_statements в определенный момент времени.

Структура таблицы snap_pg-stat-statements_short для PostgreSQL 15:

```
CREATE TABLE "__rds_pg_stats__".snap_pg_stat_statements_short (
	snap_id int8 NULL,
	snap_ts timestamptz NULL,
	userid oid NULL,
	dbid oid NULL,
	toplevel bool NULL,
	queryid int8 NULL,
	"plans" int8 NULL,
	total_plan_time float8 NULL,
	min_plan_time float8 NULL,
	max_plan_time float8 NULL,
	mean_plan_time float8 NULL,
	stddev_plan_time float8 NULL,
	calls int8 NULL,
	total_exec_time float8 NULL,
	min_exec_time float8 NULL,
	max_exec_time float8 NULL,
	mean_exec_time float8 NULL,
	stddev_exec_time float8 NULL,
	"rows" int8 NULL,
	shared_blks_hit int8 NULL,
```

```
	shared_blks_read int8 NULL,
	shared_blks_dirtied int8 NULL,
	shared_blks_written int8 NULL,
	local_blks_hit int8 NULL,
	local_blks_read int8 NULL,
	local_blks_dirtied int8 NULL,
	local_blks_written int8 NULL,
	temp_blks_read int8 NULL,
	temp_blks_written int8 NULL,
	blk_read_time float8 NULL,
	blk_write_time float8 NULL,
	temp_blk_read_time float8 NULL,
	temp_blk_write_time float8 NULL,
	wal_records int8 NULL,
	wal_fpi int8 NULL,
	wal_bytes numeric NULL,
	jit_functions int8 NULL,
	jit_generation_time float8 NULL,
	jit_inlining_count int8 NULL,
	jit_inlining_time float8 NULL,
	jit_optimization_count int8 NULL,
	jit_optimization_time float8 NULL,
	jit_emission_count int8 NULL,
	jit_emission_time float8 NULL
)
PARTITION BY RANGE (snap_ts);
CREATE INDEX snap_pg_stat_statements_short_snap_id_idx ON ONLY
__rds_pg_stats__.snap_pg_stat_statements_short USING btree (snap_id);
CREATE	INDEX	snap_pg_stat_statements_short_userid_dbid_queryid_idx	ON	ONLY
__rds_pg_stats__.snap_pg_stat_statements_short USING btree (queryid);
```

Структура таблицы snap_pg_stat_sqltext:

```
CREATE TABLE "__rds_pg_stats__".snap_pg_stat_sqltext (
	snap_id int8 NULL,
	snap_ts timestamptz NULL,
	queryid int8 NULL,
	query text NULL
)
PARTITION BY RANGE (snap_ts);
```

Структура представления snap_pg_stat_statements:

```
CREATE OR REPLACE VIEW "__rds_pg_stats__".snap_pg_stat_statements
AS SELECT DISTINCT stst.snap_id,
    stst.snap_ts,
    stst.userid,
    stst.dbid,
    stst.toplevel,
    stst.queryid,
    stst.plans,
    stst.total_plan_time,
    stst.min_plan_time,
    stst.max_plan_time,
    stst.mean_plan_time,
```

```
    stst.stddev_plan_time,
    stst.calls,
    stst.total_exec_time,
    stst.min_exec_time,
    stst.max_exec_time,
    stst.mean_exec_time,
    stst.stddev_exec_time,
    stst.rows,
    stst.shared_blks_hit,
    stst.shared_blks_read,
    stst.shared_blks_dirtied,
    stst.shared_blks_written,
    stst.local_blks_hit,
    stst.local_blks_read,
    stst.local_blks_dirtied,
    stst.local_blks_written,
    stst.temp_blks_read,
    stst.temp_blks_written,
    stst.blk_read_time,
    stst.blk_write_time,
    stst.temp_blk_read_time,
    stst.temp_blk_write_time,
    stst.wal_records,
    stst.wal_fpi,
    stst.wal_bytes,
    stst.jit_functions,
    stst.jit_generation_time,
    stst.jit_inlining_count,
    stst.jit_inlining_time,
    stst.jit_optimization_count,
    stst.jit_optimization_time,
    stst.jit_emission_count,
    stst.jit_emission_time,
    stxt.query
   FROM __rds_pg_stats__.snap_pg_stat_statements_short stst,
    __rds_pg_stats__.snap_pg_stat_sqltext stxt
  WHERE stxt.queryid = stst.queryid;
```

6.2.21. snap_pg_host_info

Имя таблицы или представления	Тип	Описание
snap_pg_host_info	Секционированная таблица	Таблица, определяемая запросом. Информация о хосте.

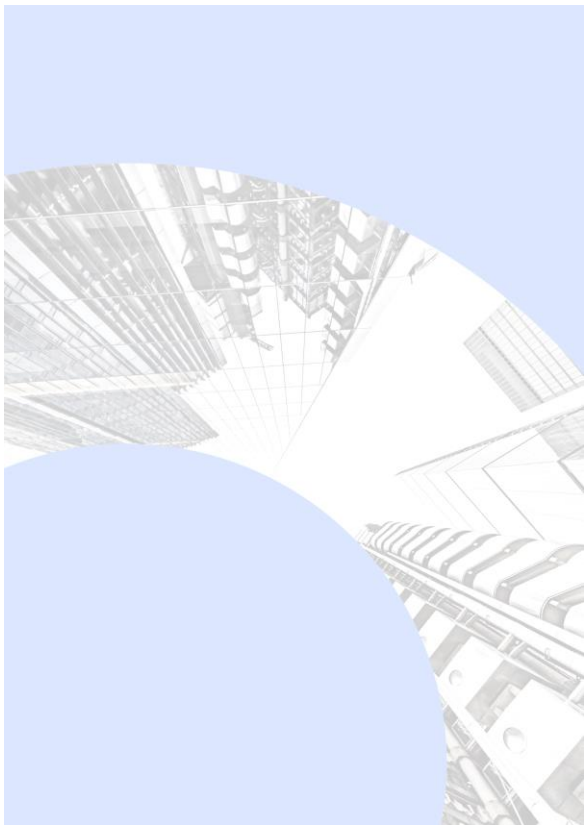
Структура таблицы:

```
CREATE TABLE "__rds_pg_stats__".snap_pg_host_info (
       snap_id int8 NULL,
       snap_ts timestamptz NULL,
       hostname bpchar(40) NULL,
       platform bpchar(40) NULL,
```

```
        cpu numeric NULL,
        mem_mb numeric NULL
)
PARTITION BY RANGE (snap_ts);
CREATE INDEX snap_pg_host_info_snap_id_idx ON ONLY __rds_pg_stats__.snap_pg_host_info USING
btree (snap_id);
```

Запрос, которым определяется таблица:

```
copy pgawr_cmd_srvinfo
from
program 'hostname; uname -por; nproc; grep MemTotal /proc/meminfo | awk ''{print $2 /
1024}''';
```



6.3. Структура бекенда PG AWR. Функции.

6.3.1. snap_global (void, erase_stats boolean DEFAULT true)

Создание снапшота в текущее время по всем модулям, определенным в snap_modules.

Как правило эта функция выпускается через cron, но возможен запуск и вручную или из скрипта.

Пример использования:

```
psql  -c "select  __rds_pg_stats__.snap_global()" >>
/pg_audit/log/pg_awr/pg_awr.log
```

Вывод:

```
NOTICE:  Node role in cluster is replica?: f
NOTICE:  LEADER: start procedure.
NOTICE:  schema_to rds_data_20250417 :
NOTICE:  schema_nm rds_data_20250417 :
NOTICE:  schema rds_data_20250417 allredy exists. Exit..
NOTICE:  START: 2025-04-17 14:25:24.649481
NOTICE:  ENABLED: 0-snap_list: 2025-04-17 14:25:24.652394+03
NOTICE:  ENABLED: 4-snap_pg_stat_progress_create_index: 2025-04-17
14:25:24.653601+03
NOTICE:  ENABLED: 4-snap_pg_stat_progress_vacuum: 2025-04-17 14:25:24.655833+03
NOTICE:  ENABLED: 4-snap_pg_stat_progress_cluster: 2025-04-17 14:25:24.656647+03
NOTICE:  DISABLED: 2-snap_pg_db_role_setting: 2025-04-17 14:25:24.657563+03
NOTICE:  DISABLED: 2-snap_pg_tbs_size: 2025-04-17 14:25:24.657675+03
NOTICE:  ENABLED: 2-snap_pg_db_size: 2025-04-17 14:25:24.657778+03
NOTICE:  DISABLED: 4-snap_pg_host_info: 2025-04-17 14:25:24.745595+03
NOTICE:  ENABLED: 1-snap_pg_locks: 2025-04-17 14:25:24.745709+03
NOTICE:  ENABLED: 1-snap_pg_stat_activity: 2025-04-17 14:25:24.750261+03
NOTICE:  ENABLED: 2-snap_blocking_sessions: 2025-04-17 14:25:24.753726+03
NOTICE:  ENABLED: 4-snap_pg_stat_top: 2025-04-17 14:25:24.754979+03
NOTICE:  ENABLED: 4-snap_pg_stat_iotop: 2025-04-17 14:25:25.002475+03
NOTICE:  DISABLED: 2-snap_pg_role_conn_limit: 2025-04-17 14:25:25.230471+03
NOTICE:  ENABLED: 1-snap_pg_stat_database: 2025-04-17 14:25:25.230596+03
NOTICE:  ENABLED: 1-snap_pg_database: 2025-04-17 14:25:25.231811+03
NOTICE:  ENABLED: 1-snap_pg_stat_bgwriter: 2025-04-17 14:25:25.232451+03
NOTICE:  ENABLED: 1-snap_pg_stat_archiver: 2025-04-17 14:25:25.233171+03
NOTICE:  ENABLED: 2-snap_pg_database_age: 2025-04-17 14:25:25.233862+03
NOTICE:  DISABLED: 2-snap_pg_user_deadline: 2025-04-17 14:25:25.234762+03
NOTICE:  ENABLED: 3-snap_pg_stat_statements: 2025-04-17 14:25:25.234885+03
NOTICE:  DISABLED: 2-snap_pg_settings: 2025-04-17 14:25:25.24133+03
NOTICE:  END: 2025-04-17 14:25:25.241363
NOTICE:  Reset_stats_flag: 0
```

В результате работы функции должен сформироваться новый снимок. Проверить это можно так:

```
psql -c 'select * from __rds_pg_stats__.snap_list order by id desc limit 10;'
  id  |            snap_ts             | snap_level
------+--------------------------------+------------
 5713 | 2025-04-17 16:31:01.621567+03 | global
 5712 | 2025-04-17 16:30:01.985267+03 | global
 5711 | 2025-04-17 16:29:01.349656+03 | global
 5710 | 2025-04-17 16:28:01.7112+03   | global
 5709 | 2025-04-17 16:27:01.070933+03 | global
 5708 | 2025-04-17 16:26:01.434623+03 | global
 5707 | 2025-04-17 16:25:01.799245+03 | global
 5706 | 2025-04-17 16:24:01.167308+03 | global
```

```
 5705 | 2025-04-17 16:23:01.529779+03 | global
 5704 | 2025-04-17 16:22:01.885519+03 | global
```

6.3.2. snap_report_global (SETOF text, i_begin_id bigint, i_end_id bigint, i_level text DEFAULT 'global'::text | func)

Генерация AWR отчета между указанными номерами снапшотов.

Пример использования:

```
psql -p 6432 -t -c "select * from __rds_pg_stats__.snap_report_global(54446,54449)"
> awr_13.html
```

Вывод:

Будет сформирован файл – отчет awr_13.html в интервале снимков 54446 – 54449

6.3.3. snap_detete_obsolete (bigint)

Процедура удаления схем со старыми данными.

Фактически она выполняет такую команду для схем старше N дней:

```
DROP SCHEMA rds_data_20250407 CASCADE;
```

где 20250407 – данные за указанную дату.

Процедура snap_delete_obsolete(7) на самом деле по умолчании сохраняет данные за 9 дней. Если вы хотите изменить глубину хранения, по нужно поправить ее здесь в двух местах:

```
-- Удаление устаревших снапшотов старше d дней
CREATE OR REPLACE FUNCTION __rds_pg_stats__.snap_delete_obsolete(d int) RETURNS void as $$
declare
  r record;
  node_role bool;
  schema_nm varchar ='';
  schema_to varchar = '';
  r1 record;
  tmp1 record;
begin
  set search_path=__rds_pg_stats__,public,pg_catalog;
    select into node_role pg_is_in_recovery();
    raise notice 'Delete obsolete schemas.';
    raise notice '============================================';
    raise notice 'Node role in cluster is replica?: %', node_role;
    if node_role = false then
       raise notice 'LEADER: start procedure.';
           -- delete old schemas __rds_stats__, older then 9 days
           raise notice '--- Global schemas ---';
           raise notice 'Pg_sleep 30 sec....';
```

```
            select into tmp1 pg_sleep(30);
          select into schema_to '__rds_pg_stats__'||TO_CHAR (now()- interval '9 days',
'YYYYMMDD_HHMMSS');
          for r1 in
            select schema_name FROM information_schema.schemata WHERE schema_name like
'__rds_pg_stats__2%'
          loop
              raise notice '> % : ',r1.schema_name;
              if r1.schema_name < schema_to then
                  raise notice 'delete schema % : ',r1.schema_name;
                    execute 'drop schema ' || r1.schema_name || ' cascade;';
              end if;
          end loop;
          -- delete old schemas rds_data_, older then 9 days
          raise notice '--- Data schemas ---';
          select into schema_to 'rds_data_'||TO_CHAR (now()- interval '9 days',
'YYYYMMDD_HHMMSS');
          for r1 in
            select schema_name FROM information_schema.schemata WHERE schema_name like
'rds_data_2%'
          loop
              raise notice '> % : ',r1.schema_name;
              if r1.schema_name < schema_to then
                  raise notice 'delete schema % : ',r1.schema_name;
                    execute 'drop schema ' || r1.schema_name || ' cascade;';
              end if;
          end loop;
      else
            raise notice 'REPLICA: skip procedure.';
    end if
```

6.4. Таблица snap_modules. Разнопериодические сборщики метрик, и «тактирование»

Метрики по разным представлениям могут собираться с разной периодичностью. Модули можно отключать, например, на авариях на очень высоких нагрузках.

Настройки регулируются через таблицу snap_modules:

id	mod_name	collect	description	type	time_w
0	0-snap_list	[v]	Snapshot list	Global	1
1	1-snap_pg_database	[v]	pg_database	PG view	1
2	1-snap_pg_locks	[v]	pg_locks	PG View	1
3	1-snap_pg_stat_activity	[v]	pg_stat_activity	PG View	1
4	1-snap_pg_stat_archiver	[v]	pg_stat_archiver	PG View	1
5	1-snap_pg_stat_bgwriter	[v]	pg_stat_bgwriter	PG View	1
6	1-snap_pg_stat_database	[v]	pg_stat_database	PG View	1
7	2-snap_blocking_sessions	[v]	Blocking sessions	Query	1
8	2-snap_pg_database_age	[v]	Database age and txid info	Query	1
9	2-snap_pg_db_role_setting	[v]	Role settings	PG View	12
10	2-snap_pg_db_size	[v]	Database size	Query	1
11	2-snap_pg_role_conn_limit	[v]	Role connection limits	Query	12
12	2-snap_pg_settings	[v]	pg_settings	PG View	12
13	2-snap_pg_tbs_size	[v]	Tablespace size	Query	12
14	2-snap_pg_user_deadline	[v]	User deadline	Query	12
15	3-snap_pg_stat_statements	[v]	pg_stat_statements	Extention	1
16	4-snap_pg_host_info	[v]	UNIX commad of hostname information	System	12
17	4-snap_pg_stat_iotop	[v]	UNIX command of /proc/[nproc]/io	System	1
18	4-snap_pg_stat_progress_cluster	[v]	pg_stat_progress_cluster	PG View	1
19	4-snap_pg_stat_progress_create_index	[v]	pg_stat_progress_create_index	PG View	1
20	4-snap_pg_stat_progress_vacuum	[v]	pg_stat_progress_vacuum	PG View	1
21	4-snap_pg_stat_top	[v]	UNIX top command	System	1

Поле collect – булево. True - метрика собирается, False – метрика не собирается.

Time_w – целочисленное, временной вес метрики. То есть если значение 1, то метрика собирается на каждом снимке, 5 – на каждом пятом снимке + 1 (то есть 1, 6, 11, 16 и т.д.), 12 – на каждом 12 снимке + 1 (1, 13, 25 и т.д.)

Единица измерения – это интервал сборки CRON. Именно по нему осуществляется «тактирование» сборки метрик. Ниже – 5 мин:

```
*/5 * * * * psql -p 6432 -c "select  __rds_pg_stats__.snap_global()" >>
/pg_audit/log/pg_awr/pg_awr.log 2>&1
```

Или минимально возможный интервал – 1 мин:

```
*/1 * * * * psql -p 6432 -c "select  __rds_pg_stats__.snap_global()" >>
/pg_audit/log/pg_awr/pg_awr.log 2>&1
```

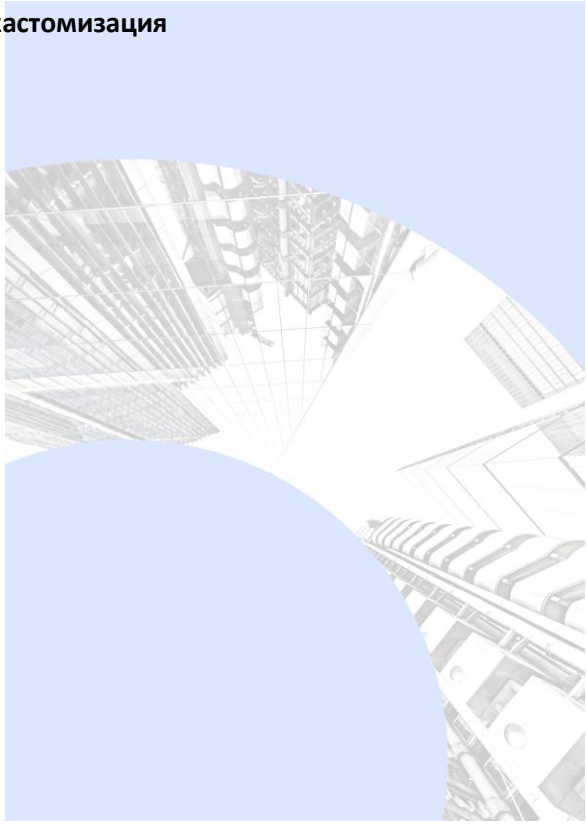
Для этого варианта временные веса можно поменять.

В минимальной конфигурации должны быть включены модули с номерами 0 и 1. При этом АВР отчет будет показывать минимальную информацию. При этом снимки будут формироваться максимально быстро. Но, разумеется, информации по историческим данным будет меньшее. В любом случае нужно понимать, какие именно параметры вам могут понадобиться в минимальной кастомизации.

123 id	ABC mod_name	collect
0	0-snap_list	[v]
1	1-snap_pg_database	[v]
2	1-snap_pg_locks	[v]
3	1-snap_pg_stat_activity	[v]
4	1-snap_pg_stat_archiver	[v]
5	1-snap_pg_stat_bgwriter	[v]
6	1-snap_pg_stat_database	[v]

Отключенные модули в АВР отчете отображаются пустыми.

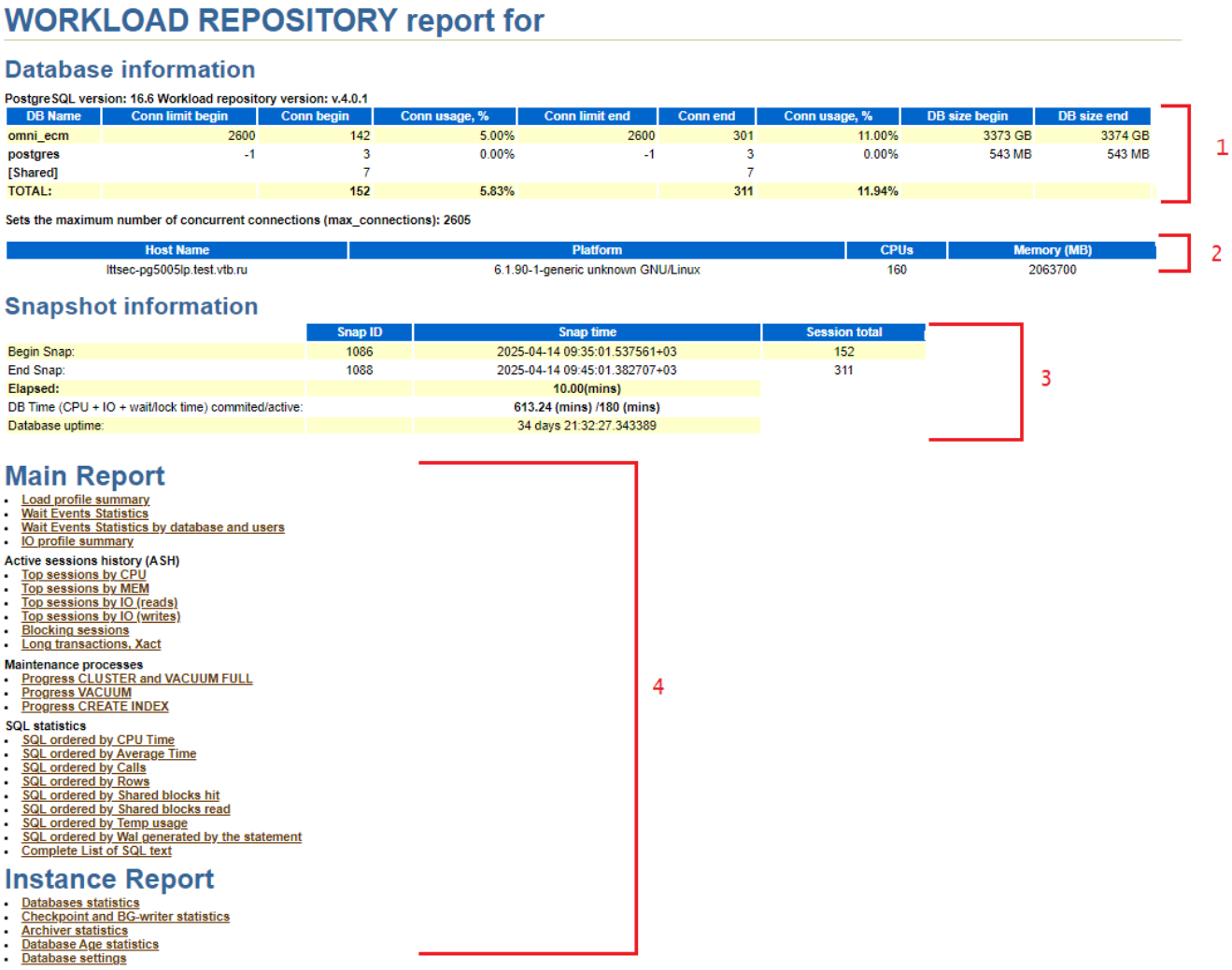
Более подробную информацию по кастомизации модулей PG AWR вы можете в разделе **Приложение Г. Кастомизация PG AWR. Автокастомизация**



7. Структура отчета PG AWR

7.1. Общий обзор.

Несмотря на то что с точки зрения оформления отчет похож на аналогичный отчет Оракл, по структуре он отличается от него. В нем может не быть каких-то секций, и присутствовать дополнительные, специфичные для PostgreSQL. Выглядит от так:



WORKLOAD REPOSITORY report for

Database information

PostgreSQL version: 16.6 Workload repository version: v.4.0.1

DB Name	Conn limit begin	Conn begin	Conn usage, %	Conn limit end	Conn end	Conn usage, %	DB size begin	DB size end
omni_ecm	2600	142	5.00%	2600	301	11.00%	3373 GB	3374 GB
postgres	-1	3	0.00%	-1	3	0.00%	543 MB	543 MB
[Shared]		7			7			
TOTAL:		152	5.83%		311	11.94%		

1

Sets the maximum number of concurrent connections (max_connections): 2605

Host Name	Platform	CPUs	Memory (MB)
lttsec-pg5005lp.test.vtb.ru	6.1.90-1-generic unknown GNU/Linux	160	2063700

2

Snapshot information

	Snap ID	Snap time	Session total
Begin Snap:	1086	2025-04-14 09:35:01.537561+03	152
End Snap:	1088	2025-04-14 09:45:01.382707+03	311
Elapsed:		10.00(mins)	
DB Time (CPU + IO + wait/lock time) commited/active:		613.24 (mins) /180 (mins)	
Database uptime:		34 days 21:32:27.343389	

3

Main Report

- Load profile summary
- Wait Events Statistics
- Wait Events Statistics by database and users
- IO profile summary

Active sessions history (ASH)

- Top sessions by CPU
- Top sessions by MEM
- Top sessions by IO (reads)
- Top sessions by IO (writes)
- Blocking sessions
- Long transactions, Xact

Maintenance processes

- Progress CLUSTER and VACUUM FULL
- Progress VACUUM
- Progress CREATE INDEX

SQL statistics

- SQL ordered by CPU Time
- SQL ordered by Average Time
- SQL ordered by Calls
- SQL ordered by Rows
- SQL ordered by Shared blocks hit
- SQL ordered by Shared blocks read
- SQL ordered by Temp usage
- SQL ordered by Wal generated by the statement
- Complete List of SQL text

Instance Report

- Databases statistics
- Checkpoint and BG-writer statistics
- Archiver statistics
- Database Age statistics
- Database settings

4

1. Заголовок отчета, содержащий общую информацию.
2. Информация о хосте БД.
3. Информация о снимках и нагрузке.
4. Разделы отчета.

Рассмотрим структуру отчета подробнее.

7.2. Заголовок отчета AWR

WORKLOAD REPOSITORY report for

Database information

PostgreSQL version: 15.6 (Debian 15.6-1.pgdg100+1) Workload repository version: v.4.0.1

DB Name	Conn limit begin	Conn begin	Conn usage, %	Conn limit end	Conn end	Conn usage, %	DB size begin	DB size end
atrium	-1	49	0.00%	-1	53	0.00%	834 GB	830 GB
atrium_refresh	100	1	1.00%	100	1	1.00%	550 GB	550 GB
modullar	100	14	14.00%	100	14	14.00%	153 GB	153 GB
modullar_receipt_manager	100	6	6.00%	100	4	4.00%	1011 MB	1018 MB
postgres	-1	4	0.00%	-1	4	0.00%	4140 MB	4478 MB
[Shared]		6			6			
TOTAL:		80	5.33%		82	5.47%		

Sets the maximum number of concurrent connections (max_connections): 1500

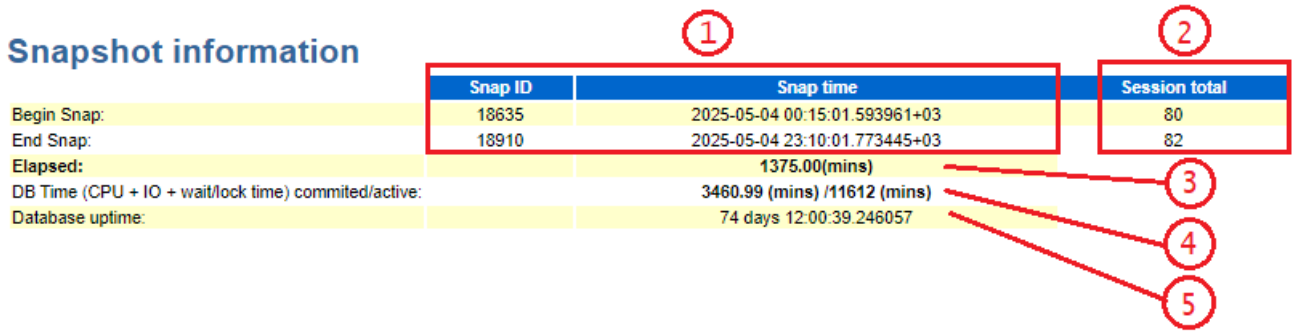
1. Информация о версии БД и о платформе БД
2. Версия Бэкенда Pg AWR
3. Список баз и служебные строки
4. Информация о лимитах соединений, утилизации соединений и проценте утилизации на начальном снимке каждой из БД.
5. Информация о лимитах соединений, утилизации соединений и проценте утилизации на конечном снимке каждой из БД.
   Здесь наиболее важно посмотреть если число соединений значительно выросло или упало в начале и конце отчета. Но динамику этого изменения лучше смотреть в графических средствах мониторинга или прямым запросом к Бекенду по таблицам snap_pg_database и snap_pg_stat_activity.
6. Размеры БД на начальном и конечном снимке отчета AWR. Также бывает ситуация когда размеры БД меняются аномально. Динамику также лучше смотреть в графических средствах мониторинга или прямым запросом к таблице snap_pg_db_size.
7. Информация по max_connection. Она дублируется с секцией database settings, просто размещена здесь для удобства.

7.3. Информация о хосте БД

Host Name	Platform	CPUs	Memory (MB)
rrfori-pg5001lp.test.vtb.ru	5.15.0-111-lowlatency unknown GNU/Linux	64	515620

1. Полное имя хоста БД
2. Версия ядра и платформа Операционной системы, на которой запущена БД PostgreSQL
3. Число ядер/потоков на сервере Базы данных (nproc)
4. Размер оперативной памяти в мегабайтах.

7.4. Информация о снимках и нагрузке



Snapshot information

	Snap ID	Snap time	Session total
Begin Snap:	18635	2025-05-04 00:15:01.593961+03	80
End Snap:	18910	2025-05-04 23:10:01.773445+03	82
Elapsed:		**1375.00(mins)**	
DB Time (CPU + IO + wait/lock time) commited/active:		**3460.99 (mins) /11612 (mins)**	
Database uptime:		74 days 12:00:39.246057	

1. Номера и метки времени начального и конечного снимков отчета PG AWR

2. Количество сессий на начальном и конечном снимке

3. Elapsed – интервал между снапшотами.

4. DB Time (CPU + IO + wait/lock time) commited/active: время БД сессий, которые находились в любых состояниях кроме Idle. Это время рассчитывается по двум разным критериям: по ASH, то есть по сессиям, находящимся в активном состоянии, но возможно еще не завершенным, и второе время commited – по завершенным запросам.

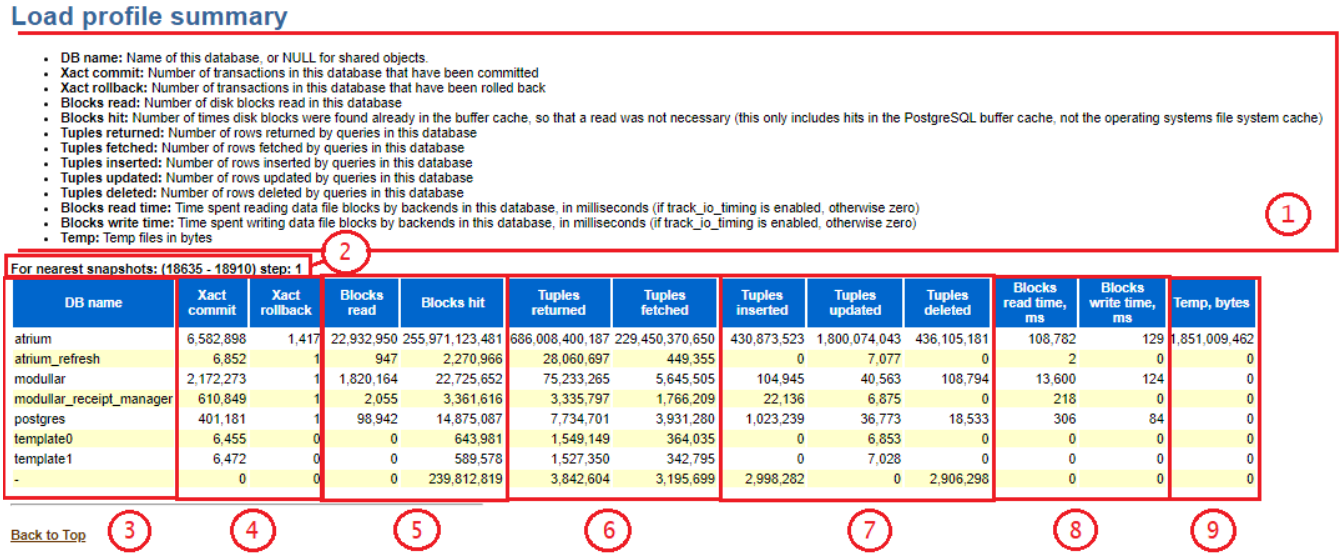
   DB Time ASH – берется из представления pg_stat_activity с дискретизацией сбора снимков. Чем меньше интервал сбора снимков, тем точнее это значение. Она учитывает откаченные и незавершенные сессии.
   DB Time commited - берется из представления pg_stat_statements, и **не учитывает** откаченные или незавершенные сессии. Дискретизация сбора метрик здесь значения не имеет в силу специфики работы pg_stat-statements.

   DB Time по метрике ASH и Commited может быть больше интервала между снимками отчета по понятным причинам, не IDLE сессий может быть много, и это время суммируется. То есть, например, если в интервале отчетов 1 мин было 10 активных сессий, но которые продолжали работу и выполняли 1 запрос в каждой из них, то DB Time ASH будет 10 мин. Если запросы при этом не завершились или были откачены, то DB Time commited будет равно 0.

   В целом соотношение между DB Time ASH и DB Time commited характеризует профиль нагрузки. При высоком значении DB Time ASH и меньшем DB Time commited – соответствует OLAP нагрузке. Если наоборот, то OLTP. Эта метрика полезна для того чтобы сравнить профили нагрузки системы в разных интервалах времени и для анализа отклонений. Например, когда система декларирована архитектурой как OLTP, но ведет себя как OLAP – это может говорить о наличии проблем, длинных запросов, и может потребовать их оптимизации.

5. Database uptime – время работы БД PostgreSQL с момента перезагрузки или переключения ролей в кластере.

7.5. Load profile summary



Load profile summary

- **DB name:** Name of this database, or NULL for shared objects.
- **Xact commit:** Number of transactions in this database that have been committed
- **Xact rollback:** Number of transactions in this database that have been rolled back
- **Blocks read:** Number of disk blocks read in this database
- **Blocks hit:** Number of times disk blocks were found already in the buffer cache, so that a read was not necessary (this only includes hits in the PostgreSQL buffer cache, not the operating systems file system cache)
- **Tuples returned:** Number of rows returned by queries in this database
- **Tuples fetched:** Number of rows fetched by queries in this database
- **Tuples inserted:** Number of rows inserted by queries in this database
- **Tuples updated:** Number of rows updated by queries in this database
- **Tuples deleted:** Number of rows deleted by queries in this database
- **Blocks read time:** Time spent reading data file blocks by backends in this database, in milliseconds (if track_io_timing is enabled, otherwise zero)
- **Blocks write time:** Time spent writing data file blocks by backends in this database, in milliseconds (if track_io_timing is enabled, otherwise zero)
- **Temp:** Temp files in bytes

1

For nearest snapshots: (18635 - 18910) step: 1

2

DB name	Xact commit	Xact rollback	Blocks read	Blocks hit	Tuples returned	Tuples fetched	Tuples inserted	Tuples updated	Tuples deleted	Blocks read time, ms	Blocks write time, ms	Temp, bytes
atrium	6,582,898	1,417	22,932,950	255,971,123,481	686,008,400,187	229,450,370,650	430,873,523	1,800,074,043	436,105,181	108,782	129	1,851,009,462
atrium_refresh	6,852	1	947	2,270,966	28,060,697	449,355	0	7,077	0	2	0	0
modullar	2,172,273	1	1,820,164	22,725,652	75,233,265	5,645,505	104,945	40,563	108,794	13,600	124	0
modullar_receipt_manager	610,849	1	2,055	3,361,616	3,335,797	1,766,209	22,136	6,875	0	218	0	0
postgres	401,181	1	98,942	14,875,087	7,734,701	3,931,280	1,023,239	36,773	18,533	306	84	0
template0	6,455	0	0	643,981	1,549,149	364,035	0	6,853	0	0	0	0
template1	6,472	0	0	589,578	1,527,350	342,795	0	7,028	0	0	0	0
-	0	0	0	239,812,819	3,842,604	3,195,699	2,998,282	0	2,906,298	0	0	0

Back to Top 3 4 5 6 7 8 9

Эта секция показывает общий профиль нагрузки системы с разбавкой по Базам данных по разнице в представлении pg_stat_statements

1. Заголовок секции с описанием столбцов.
2. For nearest snapshots – показывает ближайшие снимки в случае если начальный или конечный снимок не попали в дискретную выборку. Например если вы задали шаг дискретизации 5, (то номера снимков будут 1, 5, 11,…) а строите отчет например между снимком 3 и 10, до ближайшими снимки будут иметь номера 1 и 5.
3. Список Баз данных.
4. Xact commit – завершенные транзакции в интервале отчета, Xact rollback – откаченные.
5. Blocks read, Blocks hit – число блоков считанных с диска и ОЗУ. Отношение одного к другому – это очень важный параметр Shared Blocks hit или Hit ratio%, он дублируется в секции Database statistics.
6. Чисто строк, которые извлечено и было возвращено клиентам в интервале отчета.
7. Динамика вставок, обновлений и удалений. Именно эти параметры больше всего влияют на генерацию WAL.
8. Общее время чтения и записи блоков. Увеличение этих значений по сравнению с другими интервалами говорит о проблемах с IO
9. Динамика работы с temp данными

Для возврата в меню нажмите Back to top

7.6. Wait event statistic

Wait Events Statistics

- **Wait event:** Wait event name if backend is currently waiting, otherwise NULL
- **Wait event type:** The type of event for which the backend is waiting, if any; otherwise NULL
- **State:** Current overall state of backend.
- **Backend type:** Type of backend.
- **Event count:**The count of curent event.
- **Event avg:**The average count of curent event.
- **Event %:**The % of curent event.

Wait event	Wait event type	State	Backend type	Event count	Event count avg	Event %
-	-	active	client backend	1,946	7.08	7.244
ClientRead	Client	active	client backend	199	0.72	0.741
SubtransSLRU	LWLock	active	client backend	79	0.29	0.294
-	-	active	autovacuum worker	46	0.17	0.171
VacuumDelay	Timeout	active	autovacuum worker	22	0.08	0.082
VacuumTruncate	Timeout	active	autovacuum worker	6	0.02	0.022
transactionid	Lock	active	client backend	5	0.02	0.019
WALSync	IO	active	client backend	4	0.02	0.015
SLRURead	IO	active	client backend	3	0.01	0.011
relation	Lock	active	client backend	3	0.01	0.011
DataFileRead	IO	active	client backend	1	0.00	0.004
WALWrite	IO	active	client backend	1	0.00	0.004
WALInitSync	IO	active	client backend	1	0.00	0.004
tuple	Lock	active	client backend	1	0.00	0.004

Back to Top

Секция – компонент ASH. В этой секции показаны статистика ожидания сессий по их типу, состоянию и типу бекенда.

Изменения этой секции, связанные с ожиданиями IO, IPC и т.д. говорит о том, что система испытывает сложности с вводом-выводом или борьбой за контекст.

Для возврата в меню нажмите Back to top

7.7. Wait Events Statistics by Database & Users

Wait Events Statistics by Database & Users

- **DB Name:** Name of the database backend connected to
- **User Name:** Name of the user logged into backend
- **Backend type:** Type of backend.
- **Wait event:** Wait event name if backend is currently waiting, otherwise NULL
- **Wait event type:** The type of event for which the backend is waiting, if any; otherwise NULL
- **State:** Current overall state of backend.
- **Count:**Total count of curent event in the time interval
- **Count avg:**Average count of curent event in the time interval

DB name	User name	Backend type	Wait event type	Wait event	State	Count	Count avg
atrium	atrium	client backend	-	-	active	1,458	5.283
postgres	postgres	client backend	-	-	active	276	1.000
atrium	limit	client backend	-	-	active	211	0.764
atrium	atrium	client backend	Client	ClientRead	active	165	0.598
atrium	atrium	client backend	LWLock	SubtransSLRU	active	71	0.257
atrium	-	autovacuum worker	-	-	active	45	0.163
atrium	limit	client backend	Client	ClientRead	active	21	0.076
atrium	-	autovacuum worker	Timeout	VacuumDelay	active	20	0.072
modullar	modullar	client backend	Client	ClientRead	active	13	0.047
atrium	limit	client backend	LWLock	SubtransSLRU	active	8	0.029
atrium	-	autovacuum worker	Timeout	VacuumTruncate	active	6	0.022
atrium	atrium	client backend	IO	WALSync	active	4	0.014
atrium	atrium	client backend	IO	SLRURead	active	3	0.011
atrium	atrium	client backend	Lock	transactionid	active	3	0.011
atrium	limit	client backend	Lock	relation	active	3	0.011
atrium	limit	client backend	Lock	transactionid	active	2	0.007
modullar	-	autovacuum worker	Timeout	VacuumDelay	active	2	0.007
atrium	atrium	client backend	IO	WALWrite	active	1	0.004
atrium	atrium	client backend	IO	WALInitSync	active	1	0.004
atrium	atrium	client backend	Lock	tuple	active	1	0.004
modullar	modullar	client backend	IO	DataFileRead	active	1	0.004
modullar	modullar	client backend	-	-	active	1	0.004
modullar	-	autovacuum worker	-	-	active	1	0.004

Back to Top

Секция – компонент ASH. В этой секции показаны статистика ожидания сессий по их типу, состоянию и типу бекенда. В отличие от предыдущего раздела добавлена деталировка по базам данных и пользователям.

Изменения этой секции, связанные с ожиданиями IO, IPC и т.д. говорит о том, что система испытывает сложности в вводом-выводом или борьбой за контекст.

Для возврата в меню нажмите Back to top

7.8. IO profile summary

IO profile summary

- **Backend type:** Type of backend.
- **rchar:** Characters read. The number of bytes returned by successful read(2) and similar system calls.
- **wchar:** Characters written. The number of bytes returned by successful write(2) and similar system calls.
- **syscr:** Read syscalls. The number of "file read" system calls-those from the read(2) family, sendfile(2), copy_file_range(2), and ioctl(2) (including when invoked by the kernel as part of other syscalls).
- **syscw:** Write syscalls. The number of "file write" system calls-those from the write(2) family, sendfile(2), copy_file_range(2), and ioctl(2) (including when invoked by the kernel as part of other syscalls).
- **Read:** Bytes read. The number of bytes really fetched from the storage layer. This is accurate for block-backed filesystems.
- **Write:** Bytes written. The number of bytes really sent to the storage layer.
- **Canceled writes:** The above statistics fail to account for truncation: if a process writes 1 MB to a regular file and then removes it, said 1 MB will not be written, but will have nevertheless been accounted as a 1 MB write.
  This field represents the number of bytes "saved" from I/O writeback. This can yield to having done negative I/O if caches dirtied by another process are truncated. cancelled_write_bytes applies to I/O already accounted-for in write_bytes.
- **IO total:**Total IO utilisation by Read + Write values really sent or read to/from the storage layer.

Backend type	rchar	wchar	syscr	syscw	Read, bytes	Write, bytes	Canceled writes, bytes	IO total, bytes
client backend	13,081,370,866,943	394,699,721,128	1,585,666,443	31,340,225	431,828,992	309,419,597,824	69,686,165,504	309,851,426,816
archiver	192,788,830,835	192,653,506,240	1,745,396	1,481,307	0	192,652,771,328	0	192,652,771,328
checkpointer	0	153,984,407,750	0	18,655,094	0	153,975,402,496	0	153,975,402,496
walwriter	21,170,304	56,847,958,016	165,393	476,836	0	56,847,458,304	0	56,847,458,304
background writer	2,516,224	22,020,096	19,658	2,688	0	22,020,096	0	22,020,096
logical replication launcher	18,307,712	0	143,005	0	0	0	0	0
autovacuum launcher	20,477,184	0	159,954	0	0	0	0	0
autovacuum worker	0	0	0	0	0	0	0	0

Back to Top

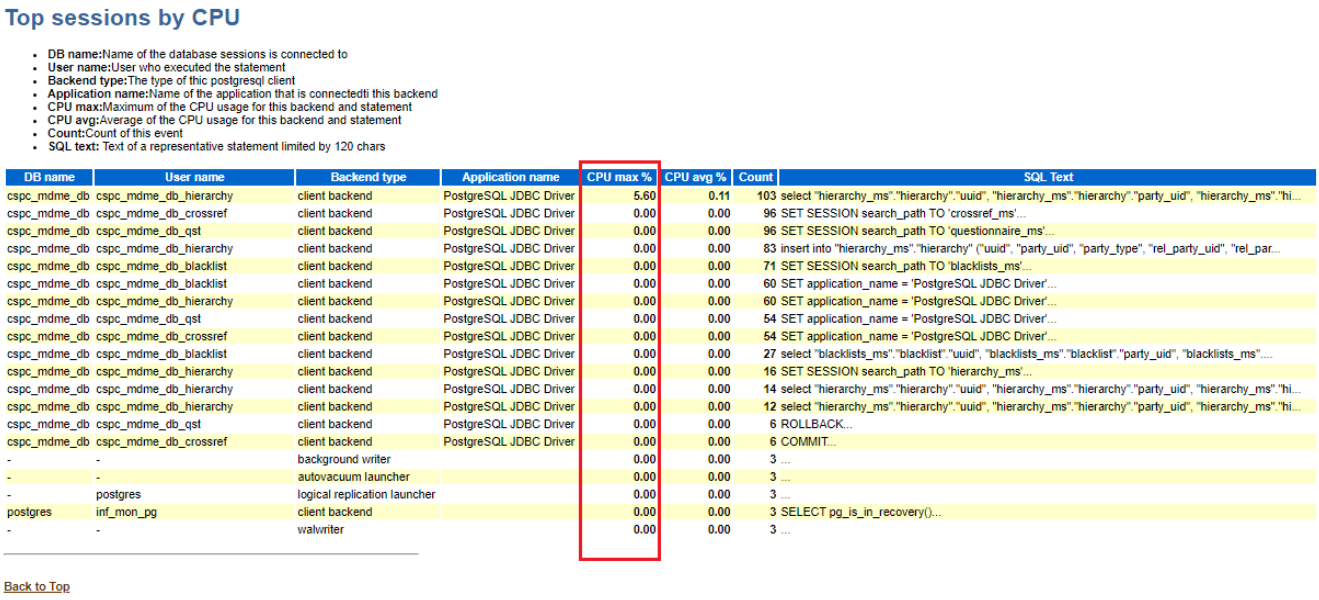
Секция – компонент ASH. Данная секция показывает информацию по вводу-выводу с разбивкой по типам бекенда. Данные берутся из связки snap_pg_stat_activity + snap_pg_iotop. Информация по вводу выводу конкретных процессов по PID берется из ядра Linix из /proc/[номер процесса]/io. В таблице показаны абсолютно все параметры этой метрики, но нам более интересны:

1. Действительные чтения и запись на устройства ввода-вывода без учета кэша операционной системы в байтах.
2. Суммарное количество байт по вводу выводу.

Здесь вы можете видеть топ по IO даже по сессиям, которые еще не завершились или были откачены.

Для возврата в меню нажмите Back to top

7.9. Top sessions by CPU



Top sessions by CPU

- **DB name:**Name of the database sessions is connected to
- **User name:**User who executed the statement
- **Backend type:**The type of thic postgresql client
- **Application name:**Name of the application that is connectedti this backend
- **CPU max:**Maximum of the CPU usage for this backend and statement
- **CPU avg:**Average of the CPU usage for this backend and statement
- **Count:**Count of this event
- **SQL text:** Text of a representative statement limited by 120 chars

DB name	User name	Backend type	Application name	CPU max %	CPU avg %	Count	SQL Text
cspc_mdme_db	cspc_mdme_db_hierarchy	client backend	PostgreSQL JDBC Driver	5.60	0.11	103	select "hierarchy_ms"."hierarchy"."uuid", "hierarchy_ms"."hierarchy"."party_uid", "hierarchy_ms"."hi...
cspc_mdme_db	cspc_mdme_db_crossref	client backend	PostgreSQL JDBC Driver	0.00	0.00	96	SET SESSION search_path TO 'crossref_ms'...
cspc_mdme_db	cspc_mdme_db_qst	client backend	PostgreSQL JDBC Driver	0.00	0.00	96	SET SESSION search_path TO 'questionnaire_ms'...
cspc_mdme_db	cspc_mdme_db_hierarchy	client backend	PostgreSQL JDBC Driver	0.00	0.00	83	insert into "hierarchy_ms"."hierarchy" ("uuid", "party_uid", "party_type", "rel_party_uid", "rel_par...
cspc_mdme_db	cspc_mdme_db_blacklist	client backend	PostgreSQL JDBC Driver	0.00	0.00	71	SET SESSION search_path TO 'blacklists_ms'...
cspc_mdme_db	cspc_mdme_db_blacklist	client backend	PostgreSQL JDBC Driver	0.00	0.00	60	SET application_name = 'PostgreSQL JDBC Driver'...
cspc_mdme_db	cspc_mdme_db_hierarchy	client backend	PostgreSQL JDBC Driver	0.00	0.00	60	SET application_name = 'PostgreSQL JDBC Driver'...
cspc_mdme_db	cspc_mdme_db_qst	client backend	PostgreSQL JDBC Driver	0.00	0.00	54	SET application_name = 'PostgreSQL JDBC Driver'...
cspc_mdme_db	cspc_mdme_db_crossref	client backend	PostgreSQL JDBC Driver	0.00	0.00	54	SET application_name = 'PostgreSQL JDBC Driver'...
cspc_mdme_db	cspc_mdme_db_blacklist	client backend	PostgreSQL JDBC Driver	0.00	0.00	27	select "blacklists_ms"."blacklist"."uuid", "blacklists_ms"."blacklist"."party_uid", "blacklists_ms"....
cspc_mdme_db	cspc_mdme_db_hierarchy	client backend	PostgreSQL JDBC Driver	0.00	0.00	16	SET SESSION search_path TO 'hierarchy_ms'...
cspc_mdme_db	cspc_mdme_db_hierarchy	client backend	PostgreSQL JDBC Driver	0.00	0.00	14	select "hierarchy_ms"."hierarchy"."uuid", "hierarchy_ms"."hierarchy"."party_uid", "hierarchy_ms"."hi...
cspc_mdme_db	cspc_mdme_db_hierarchy	client backend	PostgreSQL JDBC Driver	0.00	0.00	12	select "hierarchy_ms"."hierarchy"."uuid", "hierarchy_ms"."hierarchy"."party_uid", "hierarchy_ms"."hi...
cspc_mdme_db	cspc_mdme_db_qst	client backend	PostgreSQL JDBC Driver	0.00	0.00	6	ROLLBACK...
cspc_mdme_db	cspc_mdme_db_crossref	client backend	PostgreSQL JDBC Driver	0.00	0.00	6	COMMIT...
-	-	background writer		0.00	0.00	3	...
-	-	autovacuum launcher		0.00	0.00	3	...
-	postgres	logical replication launcher		0.00	0.00	3	...
postgres	inf_mon_pg	client backend		0.00	0.00	3	SELECT pg_is_in_recovery()...
-	-	walwriter		0.00	0.00	3	...

Back to Top

Секция – компонент ASH. Данная секция показывает информацию по утилизации ЦПУ с разбивкой по типам бекенда, базе данных, пользователю и тексту запроса. Данные берутся из связки snap_pg_stat_activity + snap_pg_top. Последний является снимком команды Linux - top по пользователю postgres.

Здесь вы можете видеть топ по ЦПУ даже по сессиям, которые еще не завершились или были откачены. То есть то, чего нет в pg_stat_statements.

Для возврата в меню нажмите Back to top

7.10. Top sessions by MEM

Top sessions by MEM

- **DB name:**Name of the database sessions is connected to
- **User name:**User who executed the statement
- **Backend type:**The type of thic postgresql client
- **Application name:**Name of the application that is connectedti this backend
- **MEM max:**Maximum of the MEMORY usage for this backend and statement
- **MEM avg:**Average of the MEMORY usage for this backend and statement
- **Count:**Count of this event
- **SQL text:** Text of a representative statement limited by 120 chars

DB name	User name	Backend type	Application name	MEM max %	MEM avg %	Count	SQL Text
-	-	checkpointer		2.70	2.70	3	...
cspc_mdme_db	cspc_mdme_db_hierarchy	client backend	PostgreSQL JDBC Driver	0.70	0.44	14	select "hierarchy_ms"."hierarchy"."uuid", "hierarchy_ms"."hierarchy"."party_uid", "hierarchy_ms"."hi...
cspc_mdme_db	cspc_mdme_db_hierarchy	client backend	PostgreSQL JDBC Driver	0.70	0.48	12	select "hierarchy_ms"."hierarchy"."uuid", "hierarchy_ms"."hierarchy"."party_uid", "hierarchy_ms"."hi...
cspc_mdme_db	cspc_mdme_db_hierarchy	client backend	PostgreSQL JDBC Driver	0.50	0.35	103	select "hierarchy_ms"."hierarchy"."uuid", "hierarchy_ms"."hierarchy"."party_uid", "hierarchy_ms"."hi...
cspc_mdme_db	cspc_mdme_db_hierarchy	client backend	PostgreSQL JDBC Driver	0.50	0.35	83	insert into "hierarchy_ms"."hierarchy" ("uuid", "party_uid", "party_type", "rel_party_uid", "rel_par...
-	-	background writer		0.30	0.30	3	...
cspc_mdme_db	cspc_mdme_db_hierarchy	client backend	PostgreSQL JDBC Driver	0.30	0.30	2	update "hierarchy_ms"."hierarchy" set "uuid" = cast($1 as uuid), "party_uid" = $2, "party_type" = $3...
cspc_mdme_db	cspc_mdme_db_crossref	client backend	PostgreSQL JDBC Driver	0.20	0.20	6	COMMIT...
postgres	postgres	client backend	psql	0.20	0.13	3	select __rds_pg_stats__.snap_global()...
cspc_mdme_db	cspc_mdme_db_hierarchy	client backend	PostgreSQL JDBC Driver	0.20	0.20	2	select "hierarchy_ms"."hierarchy"."uuid", "hierarchy_ms"."hierarchy"."party_uid", "hierarchy_ms"."hi...
cspc_mdme_db	cspc_mdme_db_qst	client backend	PostgreSQL JDBC Driver	0.10	0.10	96	SET SESSION search_path TO 'questionnaire_ms'...
cspc_mdme_db	cspc_mdme_db_crossref	client backend	PostgreSQL JDBC Driver	0.10	0.10	96	SET SESSION search_path TO 'crossref_ms'...
cspc_mdme_db	cspc_mdme_db_blacklist	client backend	PostgreSQL JDBC Driver	0.10	0.10	71	SET SESSION search_path TO 'blacklists_ms'...
cspc_mdme_db	cspc_mdme_db_blacklist	client backend	PostgreSQL JDBC Driver	0.10	0.10	60	SET application_name = 'PostgreSQL JDBC Driver'...
cspc_mdme_db	cspc_mdme_db_hierarchy	client backend	PostgreSQL JDBC Driver	0.10	0.10	60	SET application_name = 'PostgreSQL JDBC Driver'...
cspc_mdme_db	cspc_mdme_db_qst	client backend	PostgreSQL JDBC Driver	0.10	0.10	54	SET application_name = 'PostgreSQL JDBC Driver'...
cspc_mdme_db	cspc_mdme_db_crossref	client backend	PostgreSQL JDBC Driver	0.10	0.10	54	SET application_name = 'PostgreSQL JDBC Driver'...
cspc_mdme_db	cspc_mdme_db_blacklist	client backend	PostgreSQL JDBC Driver	0.10	0.10	27	select "blacklists_ms"."blacklist"."uuid", "blacklists_ms"."blacklist"."party_uid", "blacklists_ms"....
cspc_mdme_db	cspc_mdme_db_hierarchy	client backend	PostgreSQL JDBC Driver	0.10	0.10	16	SET SESSION search_path TO 'hierarchy_ms'...
cspc_mdme_db	cspc_mdme_db_qst	client backend	PostgreSQL JDBC Driver	0.10	0.10	6	ROLLBACK...

Back to Top

Секция – компонент ASH. Данная секция показывает информацию по утилизации Памяти с разбивкой по типам бекенда, базе данных, пользователю и тексту запроса. Данные берутся из связки snap_pg_stat_activity + snap_pg_top. Последний является снимком команды Linux - top по пользователю postgres.

Здесь вы можете видеть топ по Памяти даже по сессиям, которые еще не завершились или были откачены. То есть то, чего нет в pg_stat_statements.

Для возврата в меню нажмите Back to top

7.11. Top sessions by IO (reads)

Top sessions by IO (reads)

- **PID:** Process IDentifier
- **DB name:** Name of the database sessions is connected to
- **User name:** User who executed the statement
- **Backend type:** The type of thic postgresql client
- **Application name:** Name of the application that is connectedti this backend
- **Query start:** Time when the currently active query was started, or if state is not active, when the last query was started
- **Xact start:** Time when this process current transaction was started, or null if no transaction is active. If the current query is the first of its transaction, this column is equal to the query_start column.
- **PID lifetime:** Liftime of the current PID (TIME+ by UNIX top utility implementation: HH:MM:SS.SS)
- **Snap min:** The minimal snapshot number, where PID is exists (Begin SNAP ID)
- **Snap max:** The maximal snapshot number, where PID is exists (End SNAP ID)
- **Read bytes:** The number of bytes really fetched from the storage layer. This is accurate for block-backed filesystems.
- **SQL text:** Text of a representative statement limited by 1000 chars

PID	Database name	User name	Backend type	Application name	Query start	Xact start	PID lifetime	Snap min (18635)	Snap max (18910)	Read bytes	SQL text
799334	atrium	atrium	client backend	Systematica DB Adapter atrium	2025-05-04 08:06:13.959845+03	2025-05-04 08:06:13.959363+03	4:49.61	18670	18730	343 MB	select * from dbo."ExportClient_GetEvent"($1,$2,$3)...
1231111	modullar	modullar	client backend	Systematica DB Adapter modullar-qm	2025-05-04 23:10:01.693154+03	-	1755:27	18635	18910	44 MB	select query_get_resultsets() as results_A53DEF1535814D9C8D792B99CF169F79; ...
779930	atrium	limit	client backend	Limit	2025-05-04 05:59:04.907949+03	-	1:28.68	18668	18704	6768 kB	select 1...
701582	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 07:16:21.993997+03	2025-05-04 07:16:21.993997+03	168:03.17	18659	18731	5064 kB	CALL lim."LimitAPI_ImportBalanceCover"('BALANCE.LIMITAPI.DISP')...
1084029	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 08:55:00.770834+03	-	59:06.24	18704	18739	4528 kB	CALL dbo."Position_ProcessEvents" ('POSITION.PROCESS.EVENTS.ERROR')...
470345	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 03:16:21.15385+03	2025-05-04 03:16:21.15385+03	105:37.52	18635	18683	1520 kB	CALL lim."LimitAPI_ImportBalanceCover"('BALANCE.LIMITAPI.DISP')...
1106975	modullar_receipt_manager	modullar	client backend	Systematica DB Adapter ReceiptManager	2025-05-04 23:09:44.880173+03	-	3583:34	18635	18910	1396 kB	select query_get_resultsets() as results_77C5040B441541CD8878C6ACCAC3EBC5; ...
2668955	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 23:10:01.724544+03	-	53:06.28	18892	18910	904 kB	CALL dbo."MessageBusPublish_Process"('MessageBusPublish')...
2668973	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 23:10:00.077911+03	2025-05-04 23:10:01.692728+03	25:19.30	18892	18910	904 kB	CALL dbo."Position_ProcessEvents"('POSITION.PROCESS.EVENTS')...
734319	modullar	modullar	client backend	Systematica DB Adapter modullar-qm	2025-05-04 23:10:01.595435+03	-	651:59.68	18635	18910	812 kB	select query_get_resultsets() as results_9112D98A967C4C25AD9A23FBEEB897CC; ...
1035004	modullar_receipt_manager	modullar	client backend	Systematica DB Adapter ReceiptManager	2025-05-04 23:09:36.87515+03	-	1478:11	18635	18910	716 kB	select query_get_resultsets() as results_4A1C9A6D69DF4CD5AAAA991CD51D2895; ...
2650045	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 23:10:01.216094+03	-	51:07.66	18890	18910	416 kB	CALL dbo."Position_ProcessEvents" ('POSITION.PROCESS.EVENTS.ERROR')...
355564	atrium	atrium	client backend	Systematica DB Adapter atrium	2025-05-04 12:49:02.33783+03	-	4:03.07	18635	18786	284 kB	select 1...
979031	modullar_receipt_manager	modullar	client backend	Systematica DB Adapter ReceiptManager	2025-05-04 23:09:11.875842+03	-	2:19.85	18635	18910	236 kB	select query_get_resultsets() as results_EF61C595F3134C3AAF70C7AE0A8F3E41; ...
1589353	atrium	atrium	client backend	Systematica DB Adapter atrium	2025-05-04 11:04:10.923717+03	2025-05-04 11:04:10.923699+03	1:56.50	18764	18765	232 kB	do $__CMD__$ declare _Item_Id numeric(18,0); _ErrMsg text; _ErrCode int; _FetchedDate timestamp; _cOut refcursor; begin open _cOut for select 'N' AS "__UNIQUE_FIELDS", 'G' AS "__DATA_MODEL_MODE", '0' as "__INDEX_AUTO_ON"; perform query_save_resultset(_cOut); perform custom."VTBrpt_UAIB_CurrencyCBR_Cpty_1000" ('2025-04-16 00:00:00.000000'); end $__CMD__$; ...

① ② ③ ④

Секция – компонент ASH. Данная секция показывает информацию по Чтениям с устройств ввода-вывода с разбивкой по базе данных, пользователю и тексту запроса. Данные берутся из связки snap_pg_stat_activity + snap_pg_iotop. Последний является снимком /proc/[Номер процесса]/io по PID конкретного процесса.

Здесь вы можете видеть топ по вводу-выводу даже по сессиям, которые еще не завершились или были откачены. То есть то, чего нет в pg_stat_statements.

Более подробная информация по чтению этого раздела находиться в Приложение Г. Как работает ASH (Active Session History) и как читать разделы таблиц, относящиеся к нему.

Основные столбцы раздела:

1. PID процесса.
2. Информация по времени жизни процесса между снимками отчета.
3. Утилизация по чтению с устройств ввода-вывода данным процессом.
4. Текст последнего запроса в данной сессии в интервале отчета.

Для возврата в меню нажмите Back to top

7.12. Top sessions by IO (writes)

Top sessions by IO (writes)

- **PID:** Process IDentifier
- **DB name:** Name of the database sessions is connected to
- **User name:** User who executed the statement
- **Backend type:** The type of thic postgresql client
- **Application name:** Name of the application that is connectedti this backend
- **Query start:** Time when the currently active query was started, or if state is not active, when the last query was started
- **Xact start:** Time when this process current transaction was started, or null if no transaction is active. If the current query is the first of its transaction, this column is equal to the query_start column.
- **PID lifetime:** Liftime of the current PID (TIME+ by UNIX top utility implementation: HH:MM:SS.SS)
- **Snap min:** The minimal snapshot number, where PID is exists (Begin SNAP ID)
- **Snap max:** The maximal snapshot number, where PID is exists (End SNAP ID)
- **Write bytes:** The number of bytes really sent to the storage layer.
- **SQL text:** Text of a representative statement limited by 1000 chars

PID	Database name	User name	Backend type	Application name	Query start	Xact start	PID lifetime	Snap min (18635)	Snap max (18910)	Write bytes	SQL text
1012846	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 11:14:59.930435+03	-	201:10.28	18696	18767	**18 GB**	CALL dbo."BumpingCmdQueue_Process" ('BUMPINGCMDQUEUE.PROCESS')...
806955	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 09:05:01.602517+03	2025-05-04 09:05:01.693062+03	131:16.67	18671	18741	**16 GB**	CALL dbo."BumpingCmdQueue_Process" ('BUMPINGCMDQUEUE.PROCESS')...
701582	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 07:16:21.993997+03	2025-05-04 07:16:21.993997+03	168:03.17	18659	18731	**13 GB**	CALL lim."LimitAPI_ImportBalanceCover" ('BALANCE.LIMITAPI.DISP')...
990041	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 08:16:22.210678+03	2025-05-04 08:16:22.210678+03	140:45.73	18693	18743	**11 GB**	CALL lim."LimitAPI_ImportBalanceCover" ('BALANCE.LIMITAPI.DISP')...
1084018	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 10:16:22.690789+03	2025-05-04 10:16:22.690789+03	175:25.97	18704	18767	**10160 MB**	CALL lim."LimitAPI_ImportBalanceCover" ('BALANCE.LIMITAPI.DISP')...
2042751	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 18:16:24.498826+03	2025-05-04 18:16:24.498826+03	165:06.62	18818	18863	**8589 MB**	CALL lim."LimitAPI_ImportBalanceCover" ('BALANCE.LIMITAPI.DISP')...
710020	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 06:16:21.783941+03	2025-05-04 06:16:21.783941+03	102:33.29	18660	18719	**7870 MB**	CALL lim."LimitAPI_ImportBalanceCover" ('BALANCE.LIMITAPI.DISP')...
470347	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 04:16:21.361363+03	2025-05-04 04:16:21.361363+03	118:37.58	18635	18695	**7566 MB**	CALL lim."LimitAPI_ImportBalanceCover" ('BALANCE.LIMITAPI.DISP')...
470345	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 03:16:21.15385+03	2025-05-04 03:16:21.15385+03	105:37.52	18635	18683	**7127 MB**	CALL lim."LimitAPI_ImportBalanceCover" ('BALANCE.LIMITAPI.DISP')...
2518052	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 22:16:25.416228+03	2025-05-04 22:16:25.416228+03	134:04.18	18874	18910	**6944 MB**	CALL lim."LimitAPI_ImportBalanceCover" ('BALANCE.LIMITAPI.DISP')...
1790603	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 14:45:01.205584+03	2025-05-04 14:45:01.240371+03	66:54.63	18788	18809	**6455 MB**	CALL dbo."PricingCmdQueue_Process" ('PRICINGCMDQUEUE.PROCESS')...

1 2 3 4

Секция – компонент ASH. Данная секция показывает информацию по Записи на устройства ввода-вывода с разбивкой по базе данных, пользователю и тексту запроса. Данные берутся из связки snap_pg_stat_activity + snap_pg_iotop. Последний является снимком /proc/[Номер процесса]/io по PID конкретного процесса.

Здесь вы можете видеть топ по вводу-выводу даже по сессиям, которые еще не завершились или были откачены. То есть то, чего нет в pg_stat_statements.

Более подробная информация по чтению этого раздела находиться в Приложение Г. Как работает ASH (Active Session History) и как читать разделы таблиц, относящиеся к нему.

Основные столбцы раздела:

1. PID процесса.
2. Информация по времени жизни процесса между снимками отчета.
3. Утилизация по записи на устройства ввода-вывода данным процессом.
4. Текст последнего запроса в данной сессии в интервале отчета.

Для возврата в меню нажмите Back to top

7.13. Blocking sessions

Blocking sessions

- **Snap ID:** Snapshot ID
- **Snap time:** Snapshot timestamp
- **Blocking PID:** PID that is blocking query
- **User name:** Username of blocking process
- **Wait event:** Wait event which is blocking query
- **Wait event type:** Wait event type which is blocking query
- **State:** State of blocking query
- **Change age:** Change state age of blocking query
- **The Last query of the blocking session:** The last query text which is blocking session
- **Blocked PID:** PID of blocked query
- **User name:** Username of blocked process
- **Wait event:** Wait event which is blocked query
- **Wait event type:** Wait event type which is blocked query
- **State:** State of blocked query
- **Blocked query:** SQL text of the blocked query

1 2

Snap info		Blocking session							Blocked session					
Snap ID	Snap time	Blocking PID	User name	Wait event	Wait event type	State	Change age	The Last query of the blocking session	Blocked PID	User name	Wait event	Wait event type	State	Blocked query
29	2025-06-17 18:50:01.641478+03	2370458	limit	SubtransSLRU	LWLock	active	00:00:02.701	call "Limit_OL_Filter"(_LimitEventType_Id =>$1,_LimitCalc_Id =>$2,_LimitRisk_Id =>$3,_LimitRiskEvent...	2342866	limit	transactionid	Lock	active	call "Limit_OL_Filter"(_LimitEventType_Id =>$1,_LimitCalc_Id =>$2,_LimitRisk_Id =>$3,_LimitRiskEvent...
57	2025-06-17 21:10:01.614982+03	3086652	limit			active	00:00:00.844	select * from "Limit_RevalidateEventResults"()...	2370458	limit	transactionid	Lock	active	do $$ begin reset all; perform pg_advisory_unlock_all(); discard temp; end; $$...
80	2025-06-17 23:05:01.846392+03	2370458	limit	transactionid	Lock	active	00:02:39.743	call "Limit_OL_Fetch2Process"(_ProcessSingleErrorEvent =>$1,_RunTriolFilter =>$2,_RunRecalcPositionI...	3457844	atrium	transactionid	Lock	active	CALL lim."Limit_ExportClientEvent_Proceed"($1)...
80	2025-06-17 23:05:01.846392+03	3085064	atrium			active	00:53:45.071	CALL lim."LimitAPI_ImportBalanceCover"($1)...	2370458	limit	transactionid	Lock	active	call "Limit_OL_Fetch2Process"(_ProcessSingleErrorEvent =>$1,_RunTriolFilter =>$2,_RunRecalcPositionI...
81	2025-06-17 23:10:01.133214+03	2370458	limit	transactionid	Lock	active	00:07:39.029	call "Limit_OL_Fetch2Process"(_ProcessSingleErrorEvent =>$1,_RunTriolFilter =>$2,_RunRecalcPositionI...	3457844	atrium	transactionid	Lock	active	CALL lim."Limit_ExportClientEvent_Proceed"($1)...
81	2025-06-17 23:10:01.133214+03	3085064	atrium			active	00:58:44.358	CALL lim."LimitAPI_ImportBalanceCover"($1)...	2370458	limit	transactionid	Lock	active	call "Limit_OL_Fetch2Process"(_ProcessSingleErrorEvent =>$1,_RunTriolFilter =>$2,_RunRecalcPositionI...
88	2025-06-17 23:45:02.045874+03	3860344	atrium			active	00:26:52.772	CALL lim."LimitAPI_ImportBalanceCover"($1)...	2219081	limit	relation	Lock	active	select * from "Limit_B_Plugin"($1,$2,$3,$4)...
89	2025-06-17 23:50:01.134783+03	3860344	atrium			active	00:31:51.861	CALL lim."LimitAPI_ImportBalanceCover"($1)...	2219081	limit	relation	Lock	active	select * from "Limit_B_Plugin"($1,$2,$3,$4)...
90	2025-06-17 23:55:01.228129+03	3860344	atrium	BufFileRead	IO	active	00:36:51.955	CALL lim."LimitAPI_ImportBalanceCover"($1)...	3998403	limit	relation	Lock	active	select * from "Limit_B_Plugin"($1,$2,$3,$4)...

3

Back to Top

Секция истории блокировок. Здесь можно видеть структуру блокировок, которые были когда-то в прошлом с деталировкой интервала сбора статистики.

Основные столбцы раздела:

1. Информация о блокирующей сессии.
2. Информация о блокируемой сессии.

Коневые в дереве блокировок сессии отмечены красным цветом.

На примере, выделенном 3, для снимка с номером 80 (Snap ID) мы видим 2 сессии с PID 2370458,3085064. Сессия с номером 3085064 выделена красным цветом, значит она является корневой в данном дереве блокировок. Она блокирует сессию 2370458, которая в свою очередь блокирует сессию 3457844.

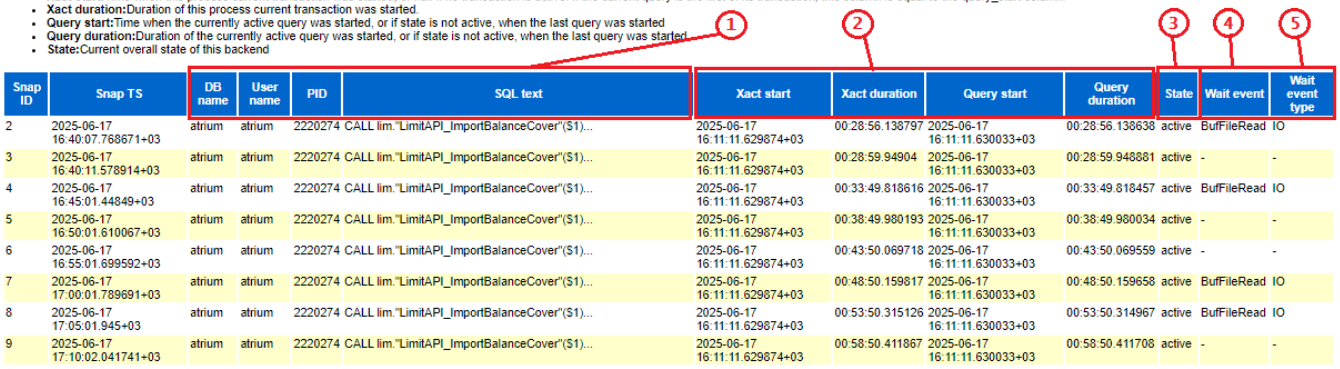
Запрос, который показан в отчете – не обязательно именно тот который создал блокировку. Внутри транзакции до него могут быть и другие запросы. Фактически нужно это читать так что сессия с таким-то номером PID заблокировала другую сессия, последним запросом которой был тот что указан в отчете.

Для возврата в меню нажмите Back to top

7.14. Long transactions, Xact

Long transactions, Xact

- **Snap TS:** Snap timestamp
- **DB name:**Name of the database this backend is connected to
- **User name:**Name of the user logged into this backend
- **PID:**Process ID of this backend
- **SQL Text:**Text of this backends most recent query. If state is active this field shows the currently executing query. In all other states, it shows the last query that was executed. By default the query text is truncated at 1024 bytes; this value can be cha
- **Xact start:**Time when this process current transaction was started, or null if no transaction is active. If the current query is the first of its transaction, this column is equal to the query_start column.
- **Xact duration:**Duration of this process current transaction was started.
- **Query start:**Time when the currently active query was started, or if state is not active, when the last query was started
- **Query duration:**Duration of the currently active query was started, or if state is not active, when the last query was started
- **State:**Current overall state of this backend



Snap ID	Snap TS	DB name	User name	PID	SQL text	Xact start	Xact duration	Query start	Query duration	State	Wait event	Wait event type
2	2025-06-17 16:40:07.768671+03	atrium	atrium	2220274	CALL lim."LimitAPI_ImportBalanceCover"($1)...	2025-06-17 16:11:11.629874+03	00:28:56.138797	2025-06-17 16:11:11.630033+03	00:28:56.138638	active	BufFileRead	IO
3	2025-06-17 16:40:11.578914+03	atrium	atrium	2220274	CALL lim."LimitAPI_ImportBalanceCover"($1)...	2025-06-17 16:11:11.629874+03	00:28:59.94904	2025-06-17 16:11:11.630033+03	00:28:59.948881	active	-	-
4	2025-06-17 16:45:01.44849+03	atrium	atrium	2220274	CALL lim."LimitAPI_ImportBalanceCover"($1)...	2025-06-17 16:11:11.629874+03	00:33:49.818616	2025-06-17 16:11:11.630033+03	00:33:49.818457	active	BufFileRead	IO
5	2025-06-17 16:50:01.610067+03	atrium	atrium	2220274	CALL lim."LimitAPI_ImportBalanceCover"($1)...	2025-06-17 16:11:11.629874+03	00:38:49.980193	2025-06-17 16:11:11.630033+03	00:38:49.980034	active	-	-
6	2025-06-17 16:55:01.699592+03	atrium	atrium	2220274	CALL lim."LimitAPI_ImportBalanceCover"($1)...	2025-06-17 16:11:11.629874+03	00:43:50.069718	2025-06-17 16:11:11.630033+03	00:43:50.069559	active	-	-
7	2025-06-17 17:00:01.789691+03	atrium	atrium	2220274	CALL lim."LimitAPI_ImportBalanceCover"($1)...	2025-06-17 16:11:11.629874+03	00:48:50.159817	2025-06-17 16:11:11.630033+03	00:48:50.159658	active	BufFileRead	IO
8	2025-06-17 17:05:01.945+03	atrium	atrium	2220274	CALL lim."LimitAPI_ImportBalanceCover"($1)...	2025-06-17 16:11:11.629874+03	00:53:50.315126	2025-06-17 16:11:11.630033+03	00:53:50.314967	active	BufFileRead	IO
9	2025-06-17 17:10:02.041741+03	atrium	atrium	2220274	CALL lim."LimitAPI_ImportBalanceCover"($1)...	2025-06-17 16:11:11.629874+03	00:58:50.411867	2025-06-17 16:11:11.630033+03	00:58:50.411708	active	-	-

Длинные транзакции. Раздел показывает транзакции, длительность которых превысила 5 мин.

Основные столбцы раздела:

1. Информация о процессе и его запрос, который выполняется на текущий момент.
2. Длительность запроса и информация о том, когда сессия меняла статус.
3. Состояние сессии (активна или нет)
4. Ожидание, в котором находится сессия.
5. Тип ожидания, в котором находится сессия.

Для возврата в меню нажмите Back to top

7.15. Секции процессов обслуживания: Progress CLUSTER and VACUUM FULL, Progress VACUUM, Progress CREATE INDEX

Progress VACUUM

- **Snap TS:** Snapshot ID
- **Snap TS:** Snapshot timestamp
- **DB Name:** Name of this database
- **PID:**Process ID of this backend
- **Phase:** Current processing phase.
- **Heap blks total:** Total number of heap blocks in the table. This number is reported as of the beginning of the scan; blocks added later will not be (and need not be) visited by this VACUUM.
- **Heap blks vacuumed:** Number of heap blocks vacuumed. Unless the table has no indexes, this counter only advances when the phase is vacuuming heap. Blocks that contain no dead tuples are skipped, so the
- **Num dead tuples:** Number of dead tuples collected since the last index vacuum cycle.

Snap ID	Snap TS	DB Name	PID	Rel ID	Phase	Heap blks total	Heap blks vacuumed	Num dead tuples
2	2025-06-17 16:40:07.768671+03	atrium	2687400	473032494	scanning heap	983,715	0	0
7	2025-06-17 17:00:01.789691+03	atrium	2746632	473032771	scanning heap	3,412	0	21
8	2025-06-17 17:05:01.945+03	atrium	2761641	473032530	scanning heap	355,577	0	3,121
9	2025-06-17 17:10:02.041741+03	atrium	2779440	473032530	scanning heap	355,577	0	0
9	2025-06-17 17:10:02.041741+03	atrium	2777325	1237508799	cleaning up indexes	132,476	132,476	0
10	2025-06-17 17:15:01.176489+03	atrium	2794566	473032530	truncating heap	355,577	355,577	0
16	2025-06-17 17:45:01.301481+03	atrium	2884932	1247	scanning heap	38,656	0	110
20	2025-06-17 18:05:01.343038+03	atrium	2943616	473032530	truncating heap	355,577	355,577	4,155
31	2025-06-17 19:00:01.949882+03	rubrica	3115503	1214	vacuuming indexes	5,872	0	177
39	2025-06-17 19:40:01.702856+03	atrium	3241831	473032559	truncating heap	4,806	4,806	0
40	2025-06-17 19:45:01.818461+03	atrium	3256526	473032530	scanning heap	380,864	0	0
41	2025-06-17 19:50:01.914309+03	atrium	3272579	1249	truncating heap	330,853	330,853	39,661
42	2025-06-17 19:55:01.266498+03	atrium	3286321	2610	cleaning up indexes	12,039	12,039	97
43	2025-06-17 20:00:01.360355+03	atrium	3301822	473032530	scanning heap	380,864	0	0
44	2025-06-17 20:05:01.670123+03	atrium	3316592	473032494	cleaning up indexes	1,051,757	1,051,757	0
45	2025-06-17 20:10:01.752974+03	atrium	3334326	473032494	scanning heap	1,051,757	0	0
46	2025-06-17 20:15:01.138959+03	atrium	3349343	473032139	truncating heap	10,990	10,990	0

В целом это похожие секции, которые показывают в моменте состояние представлений:

pg_stat_progress_cluser – для команд VACUUM FULL, CLUSTER

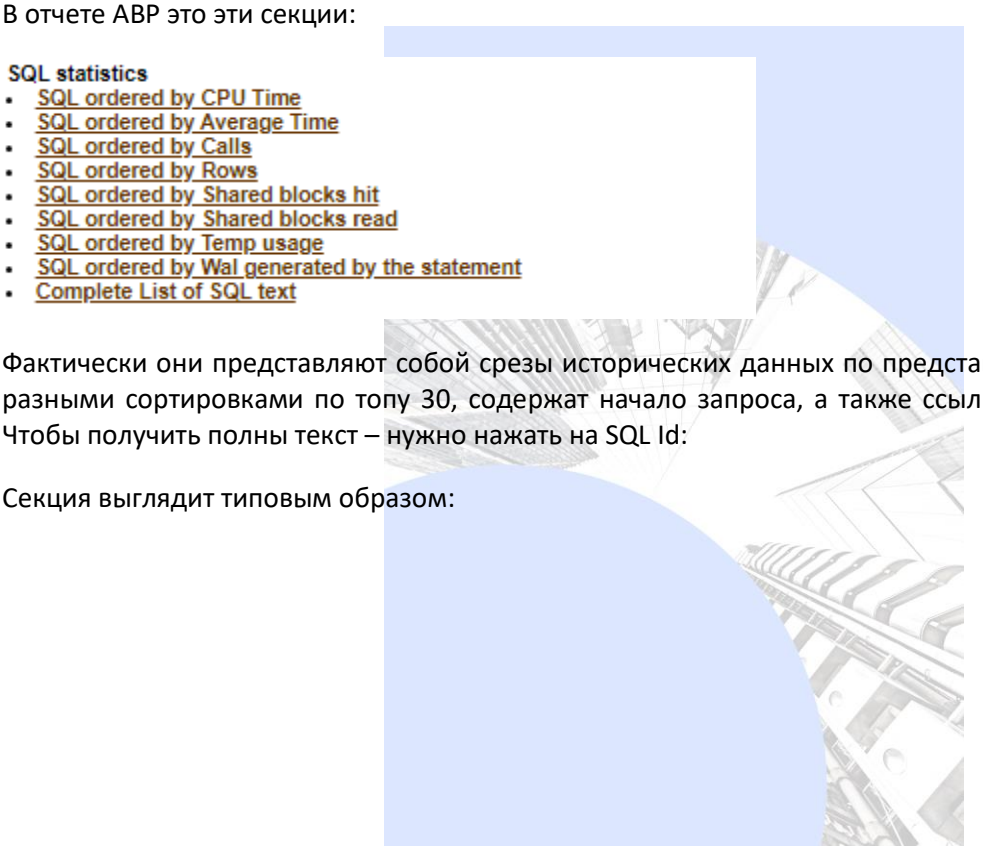
pg_stat_progress_vacuum – для команды VACUUM (как ручного так и автовакуума)

pg_stat_progress_create_index – для CREATE INDEX

Для возврата в меню нажмите Back to top

7.16. Секции, относящиеся к расширению pg_stat_statements

В отчете АВР это эти секции:



Фактически они представляют собой срезы исторических данных по представлению pg_stat_statements с разными сортировками по топу 30, содержат начало запроса, а также ссылку на полный текст запроса. Чтобы получить полны текст – нужно нажать на SQL Id:

Секция выглядит типовым образом:

SQL ordered by Shared blocks hit

- **DB name:** Name of the database sessions is connected to
- **Relname:** User who executed the statement
- **Executions:** Number of times the statement was executed
- **Avg time,s:** Mean time spent executing the statement, in seconds
- **Shared blocks hit:** Total number of shared block cache hits by the statement
- **Shared blocks read:** Total number of shared blocks read by the statement
- **Shared blocks dirted:** Total number of shared blocks dirtied by the statement
- **Shared blocks written:** Total number of shared blocks written by the statement
- **SQL Id:** Hash code to identify identical normalized queries
- **SQL text:** Text of a representative statement limited by 30 chars

DB name	Rolename	Executions	Avg time,s	Shared blocks hit,B	Shared blocks read,B	Shared blocks dirted,B	Shared blocks written,B	SQL Id	SQL text
solar	solar	491,831	0.001398	2070 GB	0 bytes	24 kB	0 bytes	-4578705062319191240	select txndetails0_.id as id1_169_, txnd...
solar	solar	491,831	0.415644	814 GB	1263 GB	3129 MB	1480 kB	-4476328401965952657	select txndetails0_.id as id1_169_, txnd...
solar	solar	4,944	0.018123	651 GB	0 bytes	38 MB	0 bytes	-1461966443531979458	delete from bch_row_record_ext where id ...
solar	solar	987,109	0.000187	195 GB	445 MB	1802 MB	1282 MB	-4364651833713025060	update bo_txn set attributes=$1, auth_co...
solar	solar	1,549,867	0.000068	185 GB	28 MB	444 MB	398 MB	-2289085836370352814	insert into slr_external_event (audit_us...
solar	solar	1,549,867	0.000030	85 GB	296 kB	50 MB	45 MB	7877834853203744612	update slr_external_event set audit_user...
solar	solar	494,176	0.000053	75 GB	87 MB	875 MB	732 MB	-8109425927812857433	update bch_row_record_ext set file_job_i...
solar	solar	1,958,184	0.000019	61 GB	0 bytes	0 bytes	0 bytes	3649895757175629O5	select actualclas0_.classifier_type_id a...
solar	solar	1,549,867	0.000023	59 GB	0 bytes	0 bytes	0 bytes	-9087154760435265251	select externalev0_.id as id1_351_0_, ex...
solar	solar	494,176	0.002444	57 GB	6396 MB	4179 MB	54 MB	4245331739792856610	insert into bo_txn (attributes, auth_cod...
solar	solar	494,176	0.000033	57 GB	62 MB	119 MB	21 MB	2651609915885916681	update bch_row_record set file_job_id=$1...
solar	solar	2,450,067	0.000014	56 GB	0 bytes	0 bytes	0 bytes	7390475153234154445	select client0_.id as id1_213_, client0_...
solar	solar	988,132	0.000043	49 GB	83 MB	48 kB	0 bytes	-7898326968188797656	select agreementa0_.id as id1_32_, agree...
solar	solar	1,970,583	0.000031	45 GB	200 MB	16 kB	0 bytes	-3434963163368159171	select agreement0_.id as id1_40_0_, agre...
solar	solar	491,844	0.000311	39 GB	1011 MB	16 kB	0 bytes	-6748093069744348336	select actualclas0_.classifier_type_id a...
solar	solar	982,645	0.000026	38 GB	56 kB	136 kB	0 bytes	2602989581417718764	select txnsummary0_.id as id1_169_, txns...
solar	solar	983,758	0.000024	38 GB	14 MB	456 kB	0 bytes	-9445542832536577l5	select originaltx0_.id as id1_150_, orig...
solar	solar	16,355	0.000797	35 GB	43 MB	20 MB	0 bytes	5645243277756146740	select txndetails0_.id as id1_169_, txnd...
solar	solar	990,330	0.000018	34 GB	7184 kB	3792 kB	0 bytes	-4050249023372938491	select txndetails0_.id as id1_169_0_, tx...
solar	solar	155,500	0.000357	31 GB	12 MB	0 bytes	0 bytes	5519208069475639067	select mcclist0_.mcc_group_id as mcc_gro...
solar	solar	491,844	0.001963	30 GB	8752 MB	6412 MB	142 MB	4969859029184282277	insert into bo_stmt_entry_agmt (show_in_...
solar	solar	493,572	0.000050	28 GB	14 MB	109 MB	91 MB	-1144107239984566566	insert into slr_external_event_key (exte...
solar	solar	490,794	0.000030	27 GB	112 kB	129 MB	129 MB	3703230090249604141	update bo_original_txn_data set serializ...
solar	solar	494,176	0.000037	27 GB	89 MB	858 MB	770 MB	6906287231492975727	insert into bch_row_record_ext (file_job...
solar	solar	3,418,076	0.000015	26 GB	0 bytes	0 bytes	0 bytes	6551810431058284830	select bankingdat0_.id as id1_66_, banki...
solar	batm_bd_zsm	249	0.162228	24 GB	1853 MB	104 kB	0 bytes	2594132888391834162	select s1.job_type as jt, coalesce(count...
solar	solar	492,900	0.000036	23 GB	0 bytes	8328 kB	0 bytes	-4997115545354335450	select txndetails0_.id as id1_169_0_, tx...
solar	batm_bd_zsm	249	0.185023	21 GB	1684 MB	24 kB	0 bytes	-1392148142064273671	select start_date::date as only_date, to...
solar	solar	491,844	0.000033	20 GB	0 bytes	88 kB	0 bytes	7183802244779073973	SELECT orig_txn_rule_id, rcvr_txn_rule_i...
solar	solar	491,844	0.000030	19 GB	0 bytes	8192 bytes	0 bytes	5738946674149317803	select txndetails0_.id as id1_169_, txnd...

Back to Top

Далее чуть подробнее по конкретным секциям:

SQL ordered by CPU Time – pg_stat_statements отсортированный по CPU Time. Это искусственная метрика, которая показывает долю процессорного времени, занятую данным запросом по отношению ко всем запросам, выполнявшихся в том же интервале времени. То есть 100% - это время выполнения всех запросов. Полученное значение не является тем же самым, что и нагрузка на ЦПУ в данном интервале времени. Это нужно иметь ввиду.

SQL ordered by Average Time – pg_stat_statements, отсортированный по среднему времени выполнения запроса. Это значение пересчитывается из Tolal Time в интервале (общее время выполнения этого запроса) и Calls (количество выполнений). В этой секции также представлены min_exec_time и max_exec_time. Но нужно понимать, что при помощи дифференциальной математики в интервале их вычислить нельзя. Поэтому они представлены с момента сброса статистики. Это начало суток. Данные значения могут быть применимы в интервале, только если в нем изменились.

SQL ordered by Calls – pg_stat_statements, отсортированный по количеству выполнений. Это разница по полю calls между конечным и начальным снимками. Если это значение не изменялось, то и этот запрос в интервале не выполнялся. Поэтому данный критерий факта выполнения запроса является основным и для других сортировок.

SQL ordered by Rows – pg_stat_statements, отсортированный по количеству затронутых строк.

SQL ordered by Shared blocks hit – pg_stat_statements, отсортированный по объёму данных, прочитанных запросом из Shared Buffers или ОЗУ. Как правило высокие значения чтений по этому параметру также вызывают высокие нагрузки в том числе по ЦПУ.

SQL ordered by Shared blocks read – pg_stat_statements, отсортированный по чтениям с устройства ввода вывода. Чем больше доля чтений с диска, тем хуже попадания блоков данных в ОЗУ, это в свою очередь значительно замедляет время выполнения запросов. Нужно стремиться к тому, чтобы чтения с дисков были минимальны.

SQL ordered by Temp usage – pg_stat_statements, отсортированный по утилизации TEMP таблиц. Как правило если недостаточно work_mem для выполнения каких-то сортировок – БД вынуждена использовать систему хранения на дисках. Это в свою очередь также замедляет время выполнения запросов.

SQL ordered by Wal generated by the statement – pg_stat_statements, отсортированный по объему сгенерированных WAL файлов. Но нужно понимать, что в представление pg_stat_statements попадают только успешно завершенные запросы. Если они еще продолжают выполняться, или были откачены, то лучше смотреть в секцию Active Session history, такую как **Top sessions by IO (writes).** Полную информацию по динамике объёма сгенерированных WAL файлов, правда без разбивки по запросам сы можете увидеть в секции **Archiver statistics**.

Complete List of SQL text – в этой секции содержится полный текст запросов из секций выше при нажатии на SQL Id.

7.17. Databases statistics

Databases statistics

- **DB name:**Name of this database, or NULL for shared objects.
- **Snapshot ID:**The ID of the snapshot
- **Snapshot timestamp:**The time of the snapshot
- **Rollback ratio:**Number of transactions in this database that have been rolled back
- **Hit ratio:**Number of times disk blocks were found already in the buffer cache, so that a read was not necessary (this only includes hits in the PostgreSQL buffer cache, not the operating systems file system c
- **Block read time,ms:**Number of disk blocks read in this database
- **Block write time,ms:**Number of disk blocks write in this database
- **Conflicts:**Number of queries canceled due to conflicts with recovery in this database. (Conflicts occur only on standby servers; see pg_stat_database_conflicts for details.)
- **Deadlocks:**Number of deadlocks detected in this database
- **Stats reset:**Time, when pg_stat_reset() was executed

DB name	Snapshot ID	Snapshot timestamp	Rollback ratio, %	Hit ratio, %	Block read time,ms	Block write time,ms	Conflicts	Deadlocks	Stats reset
atrium	2	2025-06-17 16:40:07.768671+03	0.77	99.97	183,682,085.429	292,864.637	0	7,345	-
atrium	3	2025-06-17 16:40:11.578914+03	0.77	99.97	183,682,089.180	292,864.637	0	7,345	-
atrium	4	2025-06-17 16:45:01.44849+03	0.77	99.97	183,682,352.935	292,864.637	0	7,345	-
atrium	5	2025-06-17 16:50:01.610067+03	0.77	99.97	183,682,825.911	292,864.637	0	7,345	-
atrium	6	2025-06-17 16:55:01.699592+03	0.77	99.97	183,683,228.188	292,864.637	0	7,345	-
atrium	7	2025-06-17 17:00:01.789691+03	0.77	99.97	183,683,559.064	292,864.637	0	7,345	-
atrium	8	2025-06-17 17:05:01.945+03	0.77	99.97	183,686,014.567	292,864.637	0	7,345	-
atrium	9	2025-06-17 17:10:02.041741+03	0.77	99.97	183,686,215.999	292,864.637	0	7,345	-
atrium	10	2025-06-17 17:15:01.176489+03	0.77	99.97	183,688,493.622	292,864.637	0	7,345	-
atrium	11	2025-06-17 17:20:01.237107+03	0.77	99.97	183,688,799.027	292,864.637	0	7,345	-
atrium	12	2025-06-17 17:25:01.202288+03	0.77	99.97	183,689,032.576	292,864.637	0	7,345	-
atrium	13	2025-06-17 17:30:01.282668+03	0.77	99.97	183,689,293.191	292,864.637	0	7,345	-
atrium	14	2025-06-17 17:35:01.115052+03	0.77	99.97	183,689,743.297	292,864.637	0	7,345	-
atrium	15	2025-06-17 17:40:01.198919+03	0.77	99.97	183,689,946.320	292,864.653	0	7,345	-
atrium	16	2025-06-17 17:45:01.301481+03	0.77	99.97	183,690,092.521	292,864.653	0	7,345	-
atrium	17	2025-06-17 17:50:01.434294+03	0.77	99.97	183,690,283.572	292,864.653	0	7,345	-

Секция содержит в себе дополнительную информацию по статистике БД, не вошедшую в секцию Load Profile Summary. Но содержит ее не в дифференциальном, а в абсолютном виде. То есть фактически нужно смотреть на изменение параметра, чтобы говорить, что он был изменен в данном интервале снимков.

7.18. Checkpoint and BG-writer statistics

Checkpoint and BG-writer statistics

- **Snap ID:** Snapshot ID
- **Snap TS:** Snapshot timestamp
- **Checkpoints timed:** Summ of the Number of scheduled checkpoints that have been performed
- **Checkpoints req:** Number of requested checkpoints that have been performed
- **Checkpoint write time:** Total amount of time that has been spent in the portion of checkpoint processing where files are written to disk, in milliseconds
- **Checkpoint sync time:** Total amount of time that has been spent in the portion of checkpoint processing where files are synchronized to disk, in milliseconds
- **Buffers checkpoint:** Number of buffers written by the background writer
- **Buffers clean:** Number of buffers written by the background writer
- **Maxwritten clean:** Number of times the background writer stopped a cleaning scan because it had written too many buffers
- **Buffers backend:** Number of buffers written directly by a backend
- **Buffers backend fsync:** Number of times a backend had to execute its own fsync call (normally the background writer handles those even when the backend does its own write)
- **Buffers alloc:** Number of buffers allocated

For nearest snapshots: (2 - 90) step: 1

Snap ID	Snap TS	Checkpoints timed	Checkpoints req	Checkpoint write time	Checkpoint sync time	Buffers checkpoint	Buffers clean	Maxwritten clean	Buffers backend	Buffers backend fsync	Buffers alloc
2	2025-06-17 16:40:07.768671+03	0	0	0	0	2615	0	0	212	0	279
3	2025-06-17 16:40:11.578914+03	0	0	0	0	1531	0	0	131	0	143
4	2025-06-17 16:45:01.44849+03	1	0	269918	25	91699	0	0	1088	0	2725
5	2025-06-17 16:50:01.610067+03	1	0	269875	17	98121	0	0	2749	0	5187
6	2025-06-17 16:55:01.699592+03	1	0	269931	16	124388	0	0	6707	0	7609
7	2025-06-17 17:00:01.789691+03	1	0	269946	58	121196	0	0	1864	0	6078
8	2025-06-17 17:05:01.945+03	1	0	269844	20	110421	0	0	8362	0	18397
9	2025-06-17 17:10:02.041741+03	1	0	269942	47	127295	0	0	35137	0	36008
10	2025-06-17 17:15:01.176489+03	0	1	402158	85	1113046	0	0	32670	0	34095
11	2025-06-17 17:20:01.237107+03	0	2	156950	32	1110420	0	0	2481	0	3164
12	2025-06-17 17:25:01.202288+03	0	1	140617	13	205880	0	0	6410	0	6845
13	2025-06-17 17:30:01.282668+03	1	0	269694	20	165600	0	0	2999	0	3741
14	2025-06-17 17:35:01.115052+03	1	0	269873	14	137740	0	0	2808	0	4921
15	2025-06-17 17:40:01.198919+03	1	0	269967	19	109532	0	0	2925	0	4731
16	2025-06-17 17:45:01.301481+03	1	0	269951	7	67084	0	0	3784	0	4319
17	2025-06-17 17:50:01.434294+03	1	0	269843	8	58924	0	0	1089	0	2008
18	2025-06-17 17:55:01.049192+03	1	0	270011	5	70750	0	0	6241	0	6556
19	2025-06-17 18:00:01.125693+03	1	0	269951	9	90509	0	0	2034	0	3352
20	2025-06-17 18:05:01.343038+03	1	0	269954	36	153253	692	3	26824	0	86625

Секция содержит статистику по работе CheckPoint и BG-Writer. Данные показаны в дифференциальном виде. Поэтому аномалии по частоте или объемам данных видны в табличном виде.

7.19. Archiver statistics

Archiver statistics

- **Snap ID:** Snapshot ID
- **Snap TS:** Snapshot timestamp
- **Number of archived WAL:** Number of WAL files that have been successfully archived
- **Last archived wal:** Name of the last WAL file successfully archived
- **Last archived time:** Time of the last successful archive operation
- **Failed count:** Number of failed attempts for archiving WAL files
- **Last failed wal:** Name of the WAL file of the last failed archival operation
- **Last failed time:** Time of the last failed archival operation
- **Now insert xlog file:** Last inserted xlog file
- **Current lsn:** Current lsn sequence
- **WAL generated:** WAL generated in current snapshot

For nearest snapshots: (80 - 90) step: 1

Snap ID	Snap TS	Number of archived WAL ①	Last archived wal	Last archived time	Failed count ②	Last failed wal ②	Last failed time ②	Now insert xlog file	Current lsn ③	WAL generated ④
80	2025-06-17 23:05:01.846392+03	212	000000010000399600000082	2025-06-17 23:05:01.60117+03	0	-	-	000000010000399600000084	3996/84098398	3403 MB
81	2025-06-17 23:10:01.133214+03	196	000000010000399700000046	2025-06-17 23:10:00.86884+03	0	-	-	000000010000399700000047	3997/47BDF608	3131 MB
82	2025-06-17 23:15:01.372363+03	494	000000010000399900000034	2025-06-17 23:15:01.776269+03	0	-	-	000000010000399900000035	3999/35948FD8	7901 MB
83	2025-06-17 23:20:01.620833+03	446	000000010000399A000000F2	2025-06-17 23:20:01.597825+03	0	-	-	000000010000399A000000F3	399A/F359E820	7132 MB
84	2025-06-17 23:25:01.4934+03	277	000000010000399C00000007	2025-06-17 23:25:01.784569+03	0	-	-	000000010000399C00000008	399C/8C81BA0	4439 MB
85	2025-06-17 23:30:01.692716+03	257	000000010000399D00000008	2025-06-17 23:30:00.95422+03	0	-	-	000000010000399D00000009	399D/9EA2DC8	4114 MB
86	2025-06-17 23:35:01.892883+03	146	000000010000399D0000009A	2025-06-17 23:35:00.703221+03	0	-	-	000000010000399D0000009B	399D/9B913E90	2330 MB
87	2025-06-17 23:40:01.946904+03	82	000000010000399D000000EC	2025-06-17 23:39:58.530959+03	0	-	-	000000010000399D000000EE	399D/EE0AB2F0	1320 MB
88	2025-06-17 23:45:02.045874+03	109	000000010000399E00000059	2025-06-17 23:45:01.374629+03	0	-	-	000000010000399E0000005A	399E/5A918058	1736 MB
89	2025-06-17 23:50:01.134783+03	70	000000010000399E0000009F	2025-06-17 23:49:59.230207+03	0	-	-	000000010000399E000000A0	399E/A0B58CB8	1122 MB
90	2025-06-17 23:55:01.228129+03	75	000000010000399E000000EA	2025-06-17 23:54:59.92778+03	0	-	-	000000010000399E000000EB	399E/EBF027C0	1204 MB
TOTAL:	-	2152	-	-	0	-	-	-	-	34 GB

Back to Top

Секция содержит информацию по работе архиватора WAL логов (если archive_mode = off эта секция пуста), а также информацию по объему сгенерированных WAL файлов.

Наиболее важные столбцы секции:

1 – количество архивированных файлов в интервале между снимками.

2 – ошибки архиватора. Они могут быть, например, когда закончилось место в разделе /pg_walarchive

3 – Текущий номер LSN для конкретного номера снимка snap_id

4 – Объём сгенерированных WAL файлов в интервале между снимками. Внизу – общее количество.

7.20. Database Age statistics

Database Age statistics

- **DB Name:** Name of this database
- **Snap TS:** Snapshot ID
- **Snap TS:** Snapshot timestamp
- **Age:** Age of database for this snapshot
- **Age remain:** Remain Age of database for this snapshot

① ②

DB Name	Snap ID	Snap TS	TXID current	Age	Age remain
atrium	80	2025-06-17 23:05:01.846392+03	2399277637	199,844,979	1,947,638,669
atrium	81	2025-06-17 23:10:01.133214+03	2399332261	199,899,603	1,947,584,045
atrium	82	2025-06-17 23:15:01.372363+03	2399393464	199,960,806	1,947,522,842
atrium	83	2025-06-17 23:20:01.620833+03	2399458142	200,001,715	1,947,481,933
atrium	84	2025-06-17 23:25:01.4934+03	2399502905	198,462,339	1,949,021,309
atrium	85	2025-06-17 23:30:01.692716+03	2399529480	198,488,914	1,948,994,734
atrium	86	2025-06-17 23:35:01.892883+03	2399644831	198,604,265	1,948,879,383
atrium	87	2025-06-17 23:40:01.946904+03	2399674543	198,633,977	1,948,849,671
atrium	88	2025-06-17 23:45:02.045874+03	2399703453	198,662,887	1,948,820,761
atrium	89	2025-06-17 23:50:01.134783+03	2399733658	198,693,092	1,948,790,556
atrium	90	2025-06-17 23:55:01.228129+03	2399759191	198,718,625	1,948,765,023
atrium_refresh	80	2025-06-17 23:05:01.846392+03	2399277637	199,249,128	1,948,234,520
atrium_refresh	81	2025-06-17 23:10:01.133214+03	2399332261	199,303,752	1,948,179,896
atrium_refresh	82	2025-06-17 23:15:01.372363+03	2399393464	199,364,955	1,948,118,693
atrium_refresh	83	2025-06-17 23:20:01.620833+03	2399458142	199,429,633	1,948,054,015
atrium_refresh	84	2025-06-17 23:25:01.4934+03	2399502905	199,474,396	1,948,009,252
atrium_refresh	85	2025-06-17 23:30:01.692716+03	2399529480	199,500,971	1,947,982,677
atrium_refresh	86	2025-06-17 23:35:01.892883+03	2399644831	199,616,322	1,947,867,326
atrium_refresh	87	2025-06-17 23:40:01.946904+03	2399674543	199,646,034	1,947,837,614
atrium_refresh	88	2025-06-17 23:45:02.045874+03	2399703453	199,674,944	1,947,808,704
atrium_refresh	89	2025-06-17 23:50:01.134783+03	2399733658	199,705,149	1,947,778,499
atrium_refresh	90	2025-06-17 23:55:01.228129+03	2399759191	199,730,682	1,947,752,966

Статистика по возрасту БД.

1 – Номер TXID. По нему м некоторых случаях можно установить время обновления конкретной строки по значению xmin.

2 – Счетчики возраста БД. По ним можно понять, требуется ли принудительно выполнять VACUUM FREEZE, если по каким-то причинам он не выполнился.

7.21. Database settings

Database settings

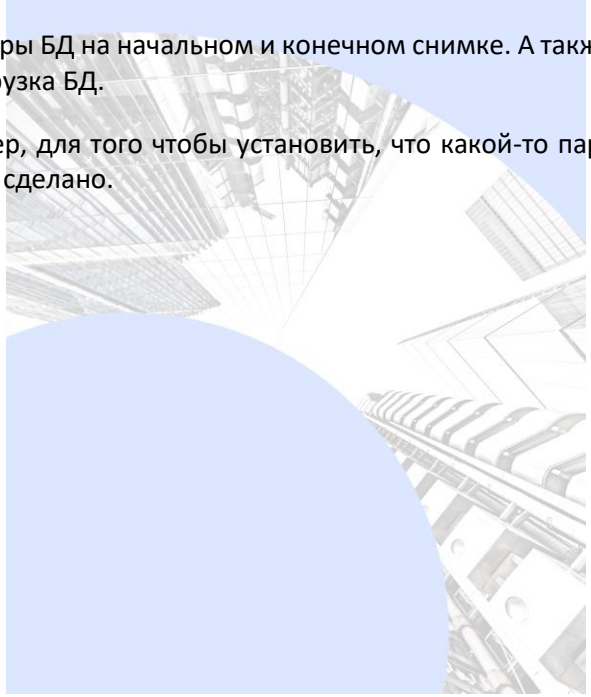
- **Name:**Run-time configuration parameter name
- **Setting begin:**Current value of the parameter
- **Setting end:**Current value of the parameter
- **Unit:**Implicit unit of the parameter
- **Pending restart begin:**Pending testart on the starting snapshot
- **Pending restart begin:**Pending testart on the ending snapshot
- **Description:**A brief description of the parameter

For nearest snapshots: (73 - 85) step: 12

Name	Setting begin	Setting end	Unit	Pending restart begin	Pending restart end	Description
allow_in_place_tablespaces	off	off	-	no	no	Allows tablespaces directly inside pg_tblspc, for testing.
allow_system_table_mods	off	off	-	no	no	Allows modifications of the structure of system tables.
application_name	psql	psql	-	no	no	Sets the application name to be reported in statistics and logs.
archive_cleanup_command			-	no	no	Sets the shell command that will be executed at every restart point.
archive_command	test ! -f /pg_walarchive/%f && cp %p /pg_walarchive/%f	test ! -f /pg_walarchive/%f && cp %p /pg_walarchive/%f	-	no	no	Sets the shell command that will be called to archive a WAL file.
archive_library			-	no	no	Sets the library that will be called to archive a WAL file.
archive_mode	on	on	-	no	no	Allows archiving of WAL files using archive_command.
archive_timeout	0	0	s	no	no	Sets the amount of time to wait before forcing a switch to the next WAL file.
array_nulls	on	on	-	no	no	Enable input of NULL elements in arrays.
authentication_timeout	60	60	s	no	no	Sets the maximum allowed time to complete client authentication.
auto_explain.log_analyze	off	off	-	no	no	Use EXPLAIN ANALYZE for plan logging.
auto_explain.log_buffers	off	off	-	no	no	Log buffers usage.
auto_explain.log_format	text	text	-	no	no	EXPLAIN format to be used for plan logging.
auto_explain.log_level	log	log	-	no	no	Log level for the plan.
auto_explain.log_min_duration	-1	-1	ms	no	no	Sets the minimum execution time above which plans will be logged.
auto_explain.log_nested_statements	off	off	-	no	no	Log nested statements.
auto_explain.log_settings	off	off	-	no	no	Log modified configuration parameters affecting query planning.
auto_explain.log_timing	on	on	-	no	no	Collect timing data, not just row counts.
auto_explain.log_triggers	off	off	-	no	no	Include trigger statistics in plans.

Секция содержит все параметры БД на начальном и конечном снимке. А также информацию, что параметр изменен и требуется перезагрузка БД.

Это бывает полезно, например, для того чтобы установить, что какой-то параметр был изменен и нужно понять время, когда это было сделано.



8. Приложение 1. Особенности обновления при переходе на PostgreSQL 16

В PostgreSQL 16 изменился формат типа aclitem. Обновление через pg_upgrage при этом корректно проходит обновление для системных данных и представлений, но некорректно для пользовательских данных. Именно такой тип встречается в таблице snap_pg_database – снимки системного представления pg_database. Поэтому при мажорном обновлении до 16 нужно выполнить следующие действия от уз postgresql на лидере:

1. Перед pg_upgrade нужно выполнить:

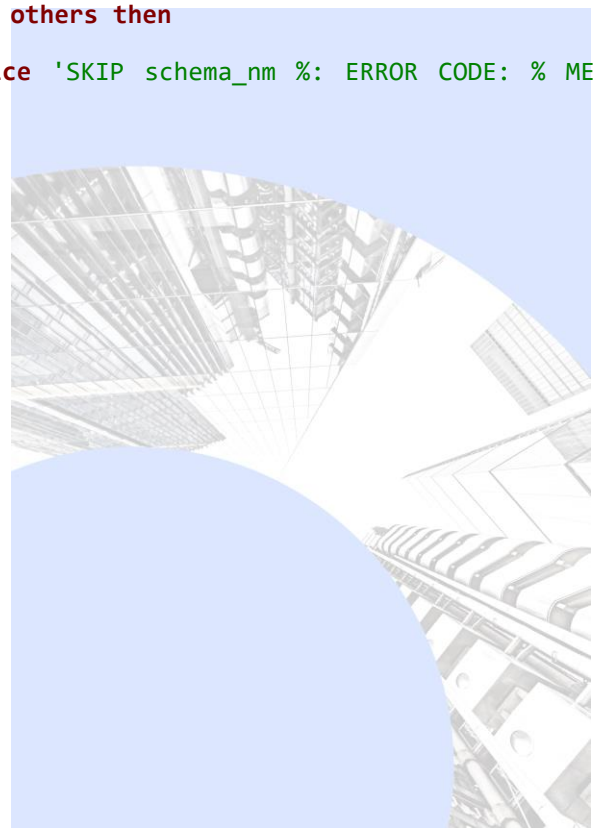
```
do $$
declare
  r record;
begin
for r in
       select * FROM information_schema.schemata WHERE schema_name like '__rds%'
       loop
             raise notice 'FIX schema_nm % : ',r.schema_name;
             begin
              execute 'ALTER TABLE ' || r.schema_name ||'.snap_pg_database ALTER COLUMN datacl
SET DATA TYPE text[];';
           exception when others then
               raise notice 'SKIP schema_nm %: ERROR CODE: % MESSAGE : %',r.schema_name,
SQLSTATE, SQLERRM;
           end;
       end loop;
end $$ ;
```

2. Проводим pg_upgrade

3. После pg_upgrade

```
do $$
declare
  r record;
```

```
begin
for r in
       select * FROM information_schema.schemata WHERE schema_name like '__rds%'
       loop
             raise notice 'FIX schema_nm % : ',r.schema_name;
             begin
              execute 'ALTER TABLE ' || r.schema_name ||'.snap_pg_database ALTER COLUMN datacl
SET DATA TYPE aclitem [] USING  datacl::aclitem[];';
           exception when others then
               raise notice 'SKIP schema_nm %: ERROR CODE: % MESSAGE : %',r.schema_name,
SQLSTATE, SQLERRM;
           end;
       end loop;
end $$ ;
```



9. Приложение 2. Пример модификации запроса для бекенда PG AWR

У вас может быть репозиторий скриптов, при помощи которых вы могли оперативно анализировать то или иное событие. Например, деревья блокировок. Один из примеров такого запроса, показывающий дерево блокировок в моменте:

```
WITH RECURSIVE l AS (
  SELECT pid, locktype, mode, granted,
ROW(locktype,database,relation,page,tuple,virtualxid,transactionid,classid,objid,objsubid)
obj
  FROM pg_locks
), pairs AS (
  SELECT w.pid waiter, l.pid locker, l.obj, l.mode
  FROM l w
  JOIN l ON l.obj IS NOT DISTINCT FROM w.obj AND l.locktype=w.locktype AND NOT l.pid=w.pid
AND l.granted
  WHERE NOT w.granted
), tree AS (
  SELECT l.locker pid, l.locker root, NULL::record obj, NULL AS mode, 0 lvl, locker::text
path, array_agg(l.locker) OVER () all_pids
  FROM ( SELECT DISTINCT locker FROM pairs l WHERE NOT EXISTS (SELECT 1 FROM pairs WHERE
waiter=l.locker) ) l
  UNION ALL
  SELECT w.waiter pid, tree.root, w.obj, w.mode, tree.lvl+1, tree.path||'.'||w.waiter,
all_pids || array_agg(w.waiter) OVER ()
  FROM tree JOIN pairs w ON tree.pid=w.locker AND NOT w.waiter = ANY ( all_pids )
)
SELECT (clock_timestamp() - a.xact_start)::interval(3) AS ts_age,
       replace(a.state, 'idle in transaction', 'idletx') state,
       (clock_timestamp() - state_change)::interval(3) AS change_age,
       a.datname,tree.pid,a.usename,a.client_addr,lvl,
       (SELECT count(*) FROM tree p WHERE p.path ~ ('^'||tree.path) AND NOT
p.path=tree.path) blocked,
       repeat(' .', lvl)||' '||left(regexp_replace(query, '\s+', ' ', 'g'),100) query
FROM tree
JOIN pg_stat_activity a USING (pid)
ORDER BY path;
```

Запрос использует два системных представления pg_locks и pg_stat_activity. Но снимки именно этих же системных представлений есть ре репозитории PG AWR – snap_pg_locks, snap_pg_stat_activity. Для того чтобы получить дерево блокировок когда то в прошлом, вам нужно построить похожий запрос но по конкретному номеру снимка snap_id. Модифицируем исходный запрос следующим образом для номера снимка 635:

```
---For pg_awr:
-- change snap 635

WITH RECURSIVE l AS (
  SELECT pid, locktype, mode, granted,
ROW(locktype,database,relation,page,tuple,virtualxid,transactionid,classid,objid,objsubid)
obj
  FROM "__rds_pg_stats__".snap_pg_locks spl where spl.snap_id = 635
```

```
), pairs AS (
  SELECT w.pid waiter, l.pid locker, l.obj, l.mode
  FROM l w
  JOIN l ON l.obj IS NOT DISTINCT FROM w.obj AND l.locktype=w.locktype AND NOT l.pid=w.pid
AND l.granted
  WHERE NOT w.granted
), tree AS (
  SELECT l.locker pid, l.locker root, NULL::record obj, NULL AS mode, 0 lvl, locker::text
path, array_agg(l.locker) OVER () all_pids
  FROM ( SELECT DISTINCT locker FROM pairs l WHERE NOT EXISTS (SELECT 1 FROM pairs WHERE
waiter=l.locker) ) l
  UNION ALL
  SELECT w.waiter pid, tree.root, w.obj, w.mode, tree.lvl+1, tree.path||'.'||w.waiter,
all_pids || array_agg(w.waiter) OVER ()
  FROM tree JOIN pairs w ON tree.pid=w.locker AND NOT w.waiter = ANY ( all_pids )
)
SELECT spsa.snap_id,spsa.snap_ts,
            (spsa.snap_ts - spsa.xact_start)::interval(3) AS ts_age,
       replace(spsa.state, 'idle in transaction', 'idletx') state,
       (spsa.snap_ts - state_change)::interval(3) AS change_age,
       spsa.datname,tree.pid,spsa.usename,spsa.client_addr,lvl,
       (SELECT count(*) FROM tree p WHERE p.path ~ ('^'||tree.path) AND NOT
p.path=tree.path) blocked,
       repeat(' -', lvl)||'> '||left(regexp_replace(query, '\s+', ' ', 'g'),100) query
FROM tree
JOIN "__rds_pg_stats__".snap_pg_stat_activity spsa USING (pid) where spsa.snap_id = 635
ORDER BY snap_id,path;
```

Если вы хотите увидеть дерево блокировок для другого снимка, то просто измените в двух местах номер снимка snap_id на нужный.

9.1. Замечание по оптимизации.

Для того чтобы ваши ручные запросы работали наиболее быстро, мы рекомендуем предварительно изменить search_path для нужной даты. И исключить из запроса схему "__rds_pg_stats__".

Например, если вы хотите выполнить запрос за 20 июня 2025 года, то перед его запуском выполните:

```
set search_path = 'rds_data_20250620';
```

В общем случае это может быть и так:

```
set search_path = rds_data_20250620,__rds_pg_stats__,public,pg_catalog;
```

Если вы понимаете, что запросы строятся в широком диапазоне и объем данных небольшой, то допустимо использовать:

```
set search_path = rds_pg_stats__,public,pg_catalog;
```

10. Приложение 3. Как работает ASH (Active Session History) и как читать разделы таблиц, относящиеся к нему.

Данный раздел относится в большей степени к разделам отчета:

Top sessions by IO (reads)

Top sessions by IO (writes)

В отличии от работы pg_stat_statements, в ASH могут попасть сессии, которые не завершены, или были прерваны, но утилизировали ресурсы. Но при этом нужно понимать, что в него не попадут сессии, которые существовали менее частоты дискретизации сбора снимков. Если такие сессии успешно завершились, то лучше их смотреть в разделах, относящихся к pg_stat_statements. Поэтому важен разумный баланс между этими двумя подходами, и нужно понимать, что и где нам лучше искать. Запросы, относящиеся к профилю OLAP лучше видны в ASH, а к OLTP – в pg_stat_statements.

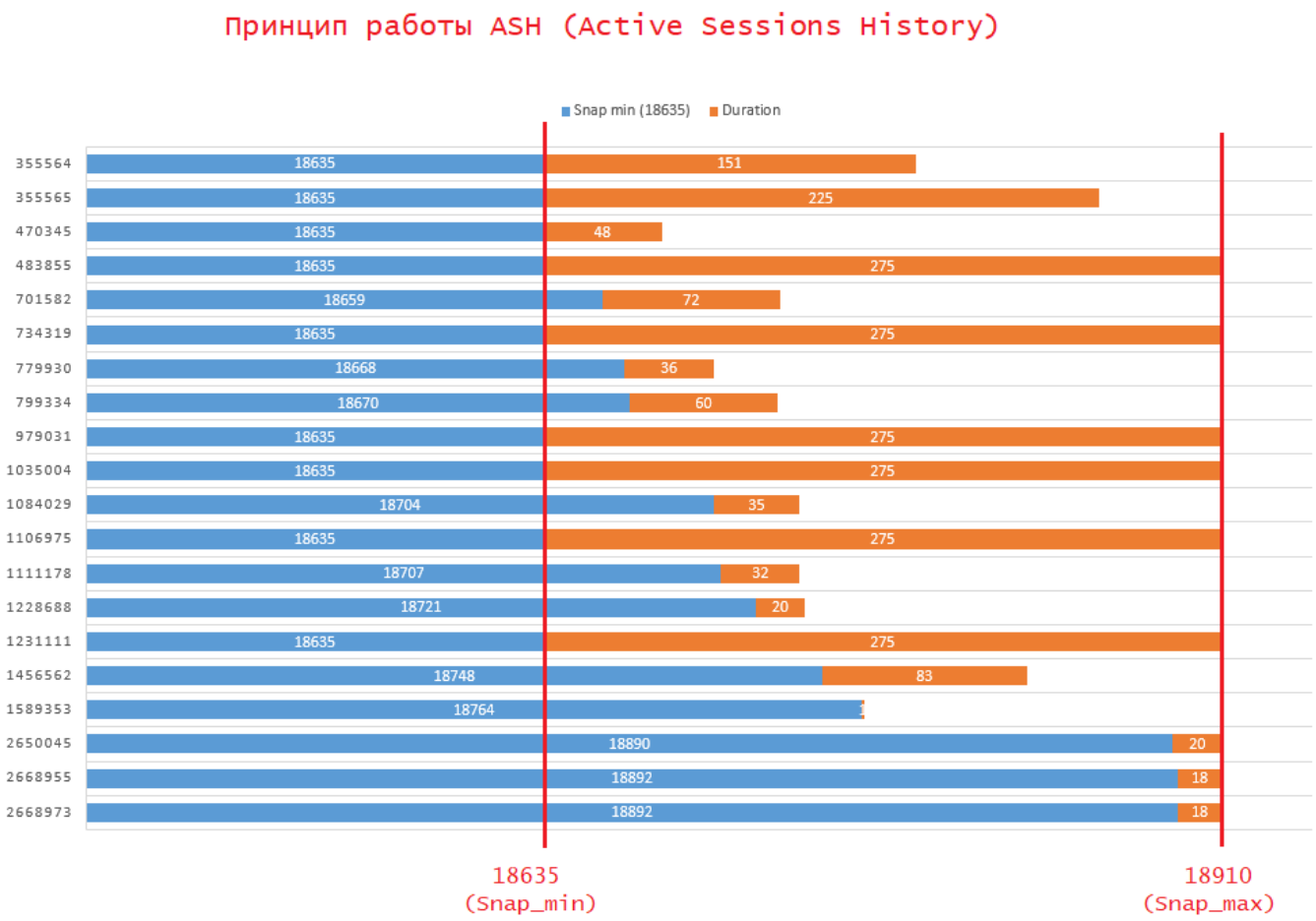
Пример исследования ввода-вывода по длинным сессиям. Мы построили отчет AWR в интервале snap_id с 18635 до 18910. Ниже я выделил нужные для демонстрации столбцы, также дополнительно добавил столбец Duration, который вычисляется как:

$$Duration = Snap\_\max - Snap\_min$$

PID	Snap min (18635)	Snap max (18910)	Duration
355564	18635	18786	151
355565	18635	18860	225
470345	18635	18683	48
483855	18635	18910	275
701582	18659	18731	72
734319	18635	18910	275
779930	18668	18704	36
799334	18670	18730	60
979031	18635	18910	275
1035004	18635	18910	275
1084029	18704	18739	35
1106975	18635	18910	275
1111178	18707	18739	32
1228688	18721	18741	20
1231111	18635	18910	275
1456562	18748	18831	83
1589353	18764	18765	1
2650045	18890	18910	20
2668955	18892	18910	18
2668973	18892	18910	18

Мы видим таблицу процессов с номерами PID, которые наблюдались между снимками 18635 и 18910 с дискретизацией – времени сбора метрики (в данном случае 1 мин, то есть 1 единица Duration равна 1 мин)

Графически это можно представить в виде диаграммы Ганта:



Первая сессия 355564 точно была на начальном снимке с номером 18635 но не дожила до конечного снимка 18910, просуществовав 151 мин. Только лишь по номерам снимков мы не знаем, когда она возникла. Но начало ее создания можно узнать по UNIX TIME (PID Lifetime) в представлении snap_pg_top (снимок команды UNIX top)

Четвертая сессия 483855 точно была на начальном снимке с номером 18635 и дожила до конечного снимка 18910. То есть существовала на протяжении всего интервала отчета. Сколько она существовал далее нам в данный момент не известно. Начало жизни сессии можно узнать так же по UNIX TIME (PID Lifetime).

Сессия, четвертая снизу 1589353 просуществовала всего один интервал дискретизации. Началась на снимке с номером 18764, завершилась на снимке 18765. В действительности она началась чуть раньше снимка 18764, и завершилась позже снимка 18765. Но мы точно знаем, что на снимках 18763 и 18766 этой сессии не было.

Последняя сессия 2668973 на момент снимка 18635 еще не существовала. Она возникла на снимке с номером 18892, и просуществовала до конечного снимка 18910, и возможно больше. Точное время возникновения сессии можно узнать по UNUX TIME (PID Lifetime).

Вот так это выглядит в отчете AWR. Попробуйте самостоятельно объяснить время жизни сессий:

Top sessions by IO (writes)

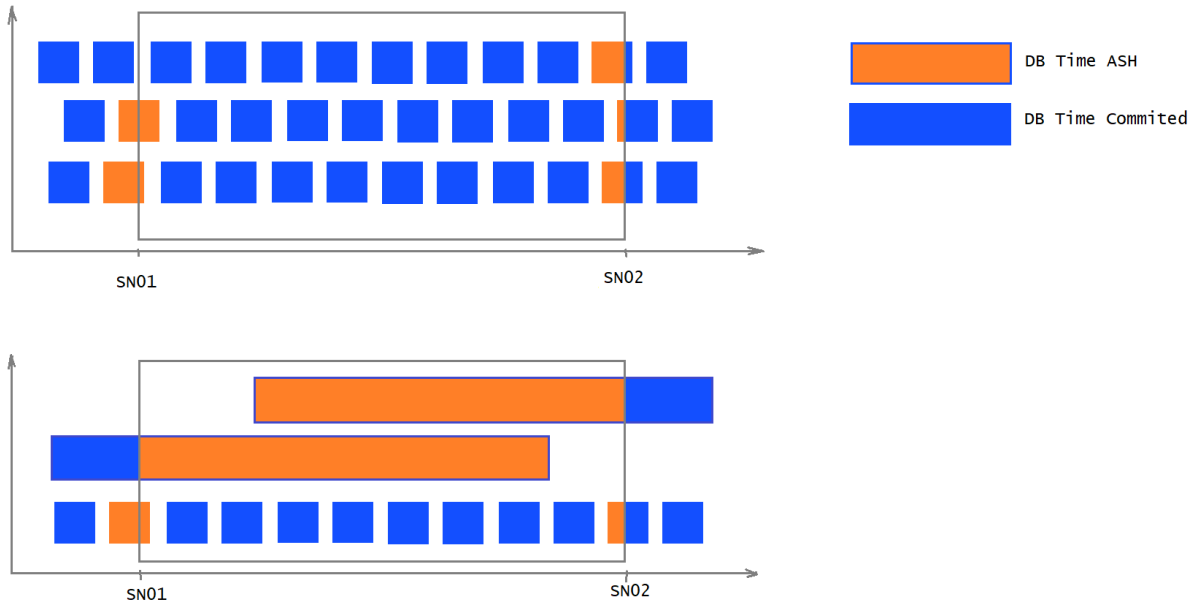
- **PID**: Process IDentifier
- **DB name**: Name of the database sessions is connected to
- **User name**: User who executed the statement
- **Backend type**: The type of thic postgresql client
- **Application name**: Name of the application that is connectedti this backend
- **Query start**: Time when the currently active query was started, or if state is not active, when the last query was started
- **Xact start**: Time when this process current transaction was started, or null if no transaction is active. If the current query is the first of its transaction, this column is equal to the query_start column.
- **PID lifetime**: Liftime of the current PID (TIME+ by UNIX top utility implementation: HH:MM:SS.SS)
- **Snap min**: The minimal snapshot number, where PID is exists (Begin SNAP ID)
- **Snap max**: The maximal snapshot number, where PID is exists (End SNAP ID)
- **Write bytes**: The number of bytes really sent to the storage layer.
- **SQL text**: Text of a representative statement limited by 1000 chars

PID	Database name	User name	Backend type	Application name	Query start	Xact start	PID lifetime	Snap min (18635)	Snap max (18910)	Write bytes	SQL text
1012846	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 11:14:59.930435+03	-	201:10.28	18696	18767	**18 GB**	CALL dbo."BumpingCmdQueue_Process" ('BUMPINGCMDQUEUE.PROCESS')...
806955	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 09:05:01.602517+03	2025-05-04 09:05:01.693062+03	131:16.67	18671	18741	**16 GB**	CALL dbo."BumpingCmdQueue_Process" ('BUMPINGCMDQUEUE.PROCESS')...
701582	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 07:16:21.993997+03	2025-05-04 07:16:21.993997+03	168:03.17	18659	18731	**13 GB**	CALL lim."LimitAPI_ImportBalanceCover" ('BALANCE.LIMITAPI.DISP')...
990041	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 08:16:22.210678+03	2025-05-04 08:16:22.210678+03	140:45.73	18693	18743	**11 GB**	CALL lim."LimitAPI_ImportBalanceCover" ('BALANCE.LIMITAPI.DISP')...
1084018	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 10:16:22.690789+03	2025-05-04 10:16:22.690789+03	175:25.97	18704	18767	**10160 MB**	CALL lim."LimitAPI_ImportBalanceCover" ('BALANCE.LIMITAPI.DISP')...
2042751	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 18:16:24.498826+03	2025-05-04 18:16:24.498826+03	165:06.62	18818	18863	**8589 MB**	CALL lim."LimitAPI_ImportBalanceCover" ('BALANCE.LIMITAPI.DISP')...
710020	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 06:16:21.783941+03	2025-05-04 06:16:21.783941+03	102:33.29	18660	18719	**7870 MB**	CALL lim."LimitAPI_ImportBalanceCover" ('BALANCE.LIMITAPI.DISP')...
470347	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 04:16:21.361363+03	2025-05-04 04:16:21.361363+03	118:37.58	18635	18695	**7566 MB**	CALL lim."LimitAPI_ImportBalanceCover" ('BALANCE.LIMITAPI.DISP')...
470345	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 03:16:21.15385+03	2025-05-04 03:16:21.15385+03	105:37.52	18635	18683	**7127 MB**	CALL lim."LimitAPI_ImportBalanceCover" ('BALANCE.LIMITAPI.DISP')...
2518052	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 22:16:25.416228+03	2025-05-04 22:16:25.416228+03	134:04.18	18874	18910	**6944 MB**	CALL lim."LimitAPI_ImportBalanceCover" ('BALANCE.LIMITAPI.DISP')...
1790603	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 14:45:01.205584+03	2025-05-04 14:45:01.240371+03	66:54.63	18788	18809	**6455 MB**	CALL dbo."PricingCmdQueue_Process" ('PRICINGCMDQUEUE.PROCESS')...
2302806	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 20:05:01.685248+03	2025-05-04 20:05:01.685248+03	76:11.70	18849	18873	**6341 MB**	CALL dbo."BumpingCmdQueue_Process" ('BUMPINGCMDQUEUE.PROCESS')...
587926	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 05:16:21.570341+03	2025-05-04 05:16:21.570341+03	118:05.26	18646	18707	**6322 MB**	CALL lim."LimitAPI_ImportBalanceCover" ('BALANCE.LIMITAPI.DISP')...
1558378	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 12:35:01.168512+03	2025-05-04 12:35:01.168512+03	58:16.79	18760	18783	**6146 MB**	CALL dbo."PricingCmdQueue_Process" ('PRICINGCMDQUEUE.PROCESS')...
1942911	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 16:39:00.027565+03	2025-05-04 16:45:01.304566+03	90:39.31	18806	18833	**5792 MB**	CALL dbo."Position_ProcessEvents" ('POSITION.PROCESS.EVENTS')...
1587335	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 12:45:00.831586+03	-	65:00.90	18764	18785	**5473 MB**	CALL dbo."PricingCmdQueue_Process" ('PRICINGCMDQUEUE.PROCESS')...
2650045	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 23:10:01.216094+03	-	51:07.66	18890	18910	**4965 MB**	CALL dbo."Position_ProcessEvents" ('POSITION.PROCESS.EVENTS.ERROR')...
2616625	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 23:09:59.193286+03	2025-05-04 23:09:59.19743+03	78:44.58	18886	18910	**4868 MB**	CALL dbo."BumpingCmdQueue_Process" ('BUMPINGCMDQUEUE.PROCESS')...
2668955	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 23:10:01.724544+03	-	53:06.28	18892	18910	**4597 MB**	CALL dbo."MessageBusPublish_Process"('MessageBusPublish')...
1689891	atrium	atrium	client backend	Systematica Event Manager	2025-05-04 13:16:23.36314+03	2025-05-04 13:16:23.36314+03	100:53.30	18776	18803	**4304 MB**	CALL lim."LimitAPI_ImportBalanceCover" ('BALANCE.LIMITAPI.DISP')...

Back to Top

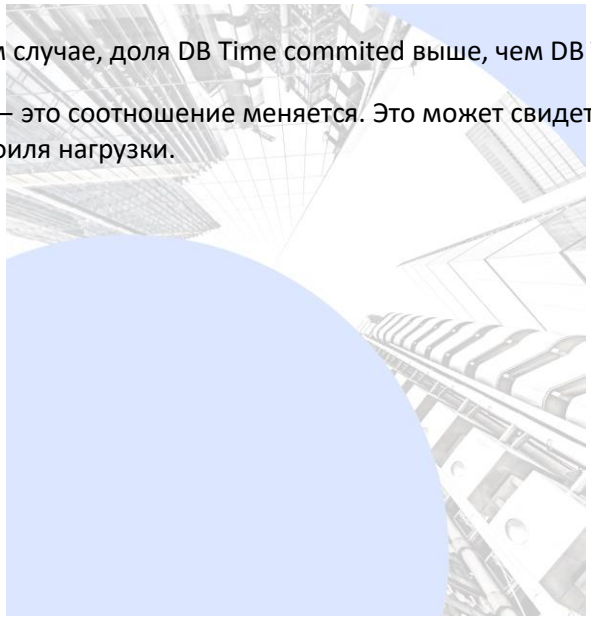
Данные представление в этой таблице были основной для диаграммы Ганта выше.

Более понятно, как это выглядит с точки зрения профиля нагрузок:



Здесь мы видим, что в первом случае, доля DB Time commited выше, чем DB Time active.

В случае смещения профиля — это соотношение меняется. Это может свидетельствовать о возникновении проблем или изменении профиля нагрузки.



11. Приложение 4. Визуализация ASH (Active Session History)

В репозитории PG AWR есть история представления pg_stat_activity в семплированном виде. Типовое время семплирования 5 мин с глубиной хранения в неизменном виде 9 дней (без свёртки да более длинные семплы).

В Оракл по умолчанию семплирование делается с частотой 1 секунда с последующей сверткой данных за 1 час через каждые 12 часов.

Как делать лучше вопрос дискуссионный, но не важный в данном контексте, т.к. для более коротких запросов, которые выполняются меньше времени семплирования мы используем расширение pg_stat_statements. В Оракле также для более коротких запросов есть его аналог DBA_HIST_SQLSTAT, но в него попадают не все запросы, а только ТОП в зависимости от statistic_level.

Что определяет характер нагрузки сессии?

В представлении pg_stat_activity характер работы сессии и ее влияние (CPU/IO/WAITS) можно однозначно определить по трем полям: state + wait_event + wait_event_type. Сделаем такой объединенный критерий sess_state.

Пример:

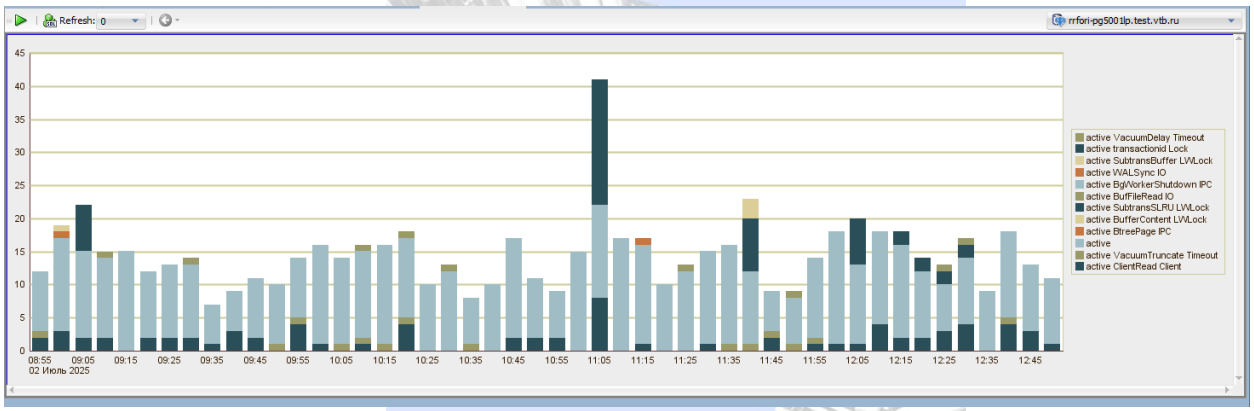
```
select
       snap_ts,
       -- статус сессий по утилизапции ресурсов и ожиданий.
       pa.state || ' ' || coalesce(wait_event, ' ')|| ' ' || coalesce(wait_event_type, ' ')
sess_state,
       count(*) cnt
from
       "__rds_pg_stats__".snap_pg_stat_activity pa
where
    -- выборка от текущего времени
       pa.snap_ts < now()
       -- за последние 4 часа
       and pa.snap_ts > now() - interval '240 min'
       and pa.state <> 'idle'
group by
       state, wait_event, wait_event_type, snap_ts
order by snap_ts, state desc, wait_event, wait_event_type;
```

Результат в табличном виде с разбивкой по метке семпла:

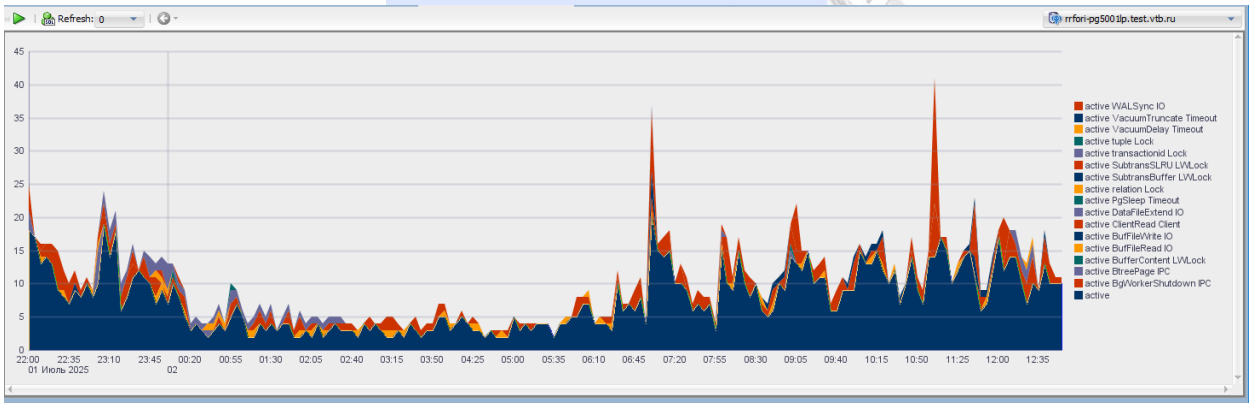
snap_ts	sess_state	count
2025-07-02 08:50:01.148831	active VacuumTruncate Timeout	1
2025-07-02 08:50:01.148831	active	10
2025-07-02 08:55:01.469529	active ClientRead Client	2
2025-07-02 08:55:01.469529	active VacuumTruncate Timeout	1
2025-07-02 08:55:01.469529	active	9
2025-07-02 09:00:01.733498	active BtreePage IPC	1
2025-07-02 09:00:01.733498	active BufferContent LWLock	1
2025-07-02 09:00:01.733498	active ClientRead Client	3
2025-07-02 09:00:01.733498	active	14

2025-07-02 09:05:01.201914	active ClientRead Client	2
2025-07-02 09:05:01.201914	active SubtransSLRU LWLock	7
2025-07-02 09:05:01.201914	active	13
2025-07-02 09:10:01.507053	active BufFileRead IO	1
2025-07-02 09:10:01.507053	active ClientRead Client	2
2025-07-02 09:10:01.507053	active	12
2025-07-02 09:15:01.893423	active	15
2025-07-02 09:20:01.197854	active ClientRead Client	2
2025-07-02 09:20:01.197854	active	10
2025-07-02 09:25:01.526333	active ClientRead Client	2
2025-07-02 09:25:01.526333	active	11
2025-07-02 09:30:01.912488	active BufFileRead IO	1
2025-07-02 09:30:01.912488	active ClientRead Client	2
и так далее		

В графическом виде это выглядит так:



Либо если это сделать в более похожем виде на Enterprise Manager через Bars stacked, то это выглядит так:



Все три поля сессии указывают не только чем они занимались (CPU/IO/Waits) но и какого типа эти ожидания. Таким образом состояние системы можно понять более точно.

Например,

active без каких-то ожиданий – это CPU. Как правило работа с shared buffers, work_mem.

active ClientRead Client – чтение данных от клиента, нагрузка по сети

active BufFileRead IO – ожидания по IO при чтении из файлового буфера.

И т.д.

Аналогично можно сделать и таблицу ниже графиков с разбивкой по запросам и ожиданиям.

Запрос для Bars Stacked чуть сложнее:

```
select
    tme.snap_ts,
    tme.lvl,
    coalesce (t1.cnt, 0)
from (select * from (select snap_ts from
    "__rds_pg_stats__".snap_pg_stat_activity pa where
        pa.snap_ts < now()
        and
        pa.snap_ts > now() - interval '900 min'
    group by snap_ts) as foo,
    (select
        pa.state ||' '|| coalesce(wait_event,' ')|| ' ' || coalesce(wait_event_type, ' ')
lvl
    from "__rds_pg_stats__".snap_pg_stat_activity pa where
        pa.snap_ts < now()
        and
        pa.snap_ts > now() - interval '900 min'
        and pa.state <> 'idle'
        group by state, wait_event, wait_event_type) as tmp01
    order by snap_ts) as tme
    full outer join
    (select snap_ts,
       pa.state ||' '|| coalesce(wait_event,' ')|| ' ' || coalesce(wait_event_type, ' ')
lvl,
       count(*) cnt
    from "__rds_pg_stats__".snap_pg_stat_activity pa where
        pa.snap_ts < now()
        and
        pa.snap_ts > now() - interval '900 min'
        and pa.state <> 'idle'
    group by state, wait_event, wait_event_type,snap_ts
    order by snap_ts, state desc, wait_event, wait_event_type) as t1 on
        t1.snap_ts = tme.snap_ts and
        t1.lvl = tme.lvl;
```

12. Приложение 5. Кастомизация PG AWR. Автокастомизация

Кастомизировать работу PG AWR можно через таблицу snap_modules. В типовом виде она выглядит следующим образом:

id	mod_name	collect	description	type	time_w
0	0-snap_list	[v]	Snapshot list	Global	1
1	1-snap_pg_database	[v]	pg_database	PG view	1
2	1-snap_pg_locks	[v]	pg_locks	PG View	1
3	1-snap_pg_stat_activity	[v]	pg_stat_activity	PG View	1
4	1-snap_pg_stat_archiver	[v]	pg_stat_archiver	PG View	1
5	1-snap_pg_stat_bgwriter	[v]	pg_stat_bgwriter	PG View	1
6	1-snap_pg_stat_database	[v]	pg_stat_database	PG View	1
7	2-snap_blocking_sessions	[v]	Blocking sessions	Query	1
8	2-snap_pg_database_age	[v]	Database age and txid info	Query	1
9	2-snap_pg_db_role_setting	[v]	Role settings	PG View	12
10	2-snap_pg_db_size	[v]	Database size	Query	1
11	2-snap_pg_role_conn_limit	[v]	Role connection limits	Query	12
12	2-snap_pg_settings	[v]	pg_settings	PG View	12
13	2-snap_pg_tbs_size	[v]	Tablespace size	Query	12
14	2-snap_pg_user_deadline	[v]	User deadline	Query	12
15	3-snap_pg_stat_statements	[v]	pg_stat_statements	Extention	1
16	4-snap_pg_host_info	[v]	UNIX commad of hostname information	System	12
17	4-snap_pg_stat_iotop	[v]	UNIX command of /proc/[nproc]/io	System	1
18	4-snap_pg_stat_progress_cluster	[v]	pg_stat_progress_cluster	PG View	1
19	4-snap_pg_stat_progress_create_index	[v]	pg_stat_progress_create_index	PG View	1
20	4-snap_pg_stat_progress_vacuum	[v]	pg_stat_progress_vacuum	PG View	1
21	4-snap_pg_stat_top	[v]	UNIX top command	System	1
22	3-pg_stat_statements_query_length	[v]	Query length for pg_stat_statements storage	Extention	50 000
23	5-analyze_global_tables	[v]	Analyze global tables	Maintenance	144

Здесь вы можете включать и выключать сборщики метрик, становив значение поля collect в true или false, изменять периодичность сборки метрик time_w.

Единицей измерения time_w является интервал запуска функции snap_global(). То есть если он установлен на 5 мин:

*/5 * * * * psql -p 6432 -c "select __rds_pg_stats__.snap_global()" >> /pg_audit/log/pg_awr/pg_awr.log 2>&1

То при значении time_w = 1 метрика будет собираться при каждом запуске snap_global().

При time_w = 2 – каждый второй снимок, time_w = 3 – каждый третий и т.д.

Например для параметра id=23 (5-analyze_global_tables) time_w = 144. Так как интервал сбора 5 мин, в часе 12 интервалов по 5 мин, то 12 * 12 = 144 означает, что Анализ глобальных таблиц будет запускаться раз в 12 часов.

Исключением является параметр 3-pg_stat_statements_query_length, который фиксирует максимальную длину текста запросов в истории представления pg_stat_statements (snap_ pg_stat_statements + окружение представлений)

12.1. Типовой сценарий 5 мин.

При сборе статистики 5 мин рекомендуются такие значения таблицы snap_modules:

```
-- For crontab 5 min
TRUNCATE __rds_pg_stats__.snap_modules;
INSERT INTO __rds_pg_stats__.snap_modules (id,mod_name,"collect",description,"type",time_w)
VALUES
        (0,'0-snap_list',true,'Snapshot list','Global',1),
        (1,'1-snap_pg_database',true,'pg_database','PG view',1),
        (2,'1-snap_pg_locks',true,'pg_locks','PG View',1),
        (3,'1-snap_pg_stat_activity',true,'pg_stat_activity','PG View',1),
        (4,'1-snap_pg_stat_archiver',true,'pg_stat_archiver','PG View',1),
        (5,'1-snap_pg_stat_bgwriter',true,'pg_stat_bgwriter','PG View',1),
        (6,'1-snap_pg_stat_database',true,'pg_stat_database','PG View',1),
        (7,'2-snap_blocking_sessions',true,'Blocking sessions','Query',1),
        (8,'2-snap_pg_database_age',true,'Database age and txid info','Query',1),
        (9,'2-snap_pg_db_role_setting',true,'Role settings','PG View',12),
        (10,'2-snap_pg_db_size',true,'Database size','Query',1),
        (11,'2-snap_pg_role_conn_limit',true,'Role connection limits','Query',12),
        (12,'2-snap_pg_settings',true,'pg_settings','PG View',12),
        (13,'2-snap_pg_tbs_size',true,'Tablespace size','Query',12),
        (14,'2-snap_pg_user_deadline',true,'User deadline','Query',12),
        (15,'3-snap_pg_stat_statements',true,'pg_stat_statements','Extention',1),
        (16,'4-snap_pg_host_info',true,'UNIX commad of hostname information','System',12),
        (17,'4-snap_pg_stat_iotop',true,'UNIX command of /proc/[nproc]/io','System',1),
        (18,'4-snap_pg_stat_progress_cluster',true,'pg_stat_progress_cluster','PG View',1),
        (19,'4-snap_pg_stat_progress_create_index',true,'pg_stat_progress_create_index','PG
View',1),
        (20,'4-snap_pg_stat_progress_vacuum',true,'pg_stat_progress_vacuum','PG View',1),
        (21,'4-snap_pg_stat_top',true,'UNIX top command','System',1),
        (22,'3-pg_stat_statements_query_length',true,'Query length for pg_stat_statements
storage','Extention',10000),
        (23,'5-analyze_global_tables',true,'Analyze global tables','Maintenance',144);
```

12.2. Типовой сценарий 1 мин.

При сборе статистики 1 мин рекомендуются такие значения таблицы snap_modules, здесь значения увеличены:

```
-- For crontab 1 min
TRUNCATE __rds_pg_stats__.snap_modules;
INSERT INTO __rds_pg_stats__.snap_modules (id,mod_name,"collect",description,"type",time_w)
VALUES
        (0,'0-snap_list',true,'Snapshot list','Global',1),
        (2,'1-snap_pg_locks',true,'pg_locks','PG View',1),
        (3,'1-snap_pg_stat_activity',true,'pg_stat_activity','PG View',1),
```

```
        (7,'2-snap_blocking_sessions',true,'Blocking sessions','Query',1),
        (17,'4-snap_pg_stat_iotop',true,'UNIX command of /proc/[nproc]/io','System',1),
        (18,'4-snap_pg_stat_progress_cluster',true,'pg_stat_progress_cluster','PG View',1),
        (19,'4-snap_pg_stat_progress_create_index',true,'pg_stat_progress_create_index','PG
View',1),
        (20,'4-snap_pg_stat_progress_vacuum',true,'pg_stat_progress_vacuum','PG View',1),
        (21,'4-snap_pg_stat_top',true,'UNIX top command','System',1),
        (1,'1-snap_pg_database',true,'pg_database','PG view',5),
        (4,'1-snap_pg_stat_archiver',true,'pg_stat_archiver','PG View',5),
        (5,'1-snap_pg_stat_bgwriter',true,'pg_stat_bgwriter','PG View',5),
        (6,'1-snap_pg_stat_database',true,'pg_stat_database','PG View',1),
        (8,'2-snap_pg_database_age',true,'Database age and txid info','Query',5),
        (9,'2-snap_pg_db_role_setting',true,'Role settings','PG View',60),
        (10,'2-snap_pg_db_size',true,'Database size','Query',5),
        (11,'2-snap_pg_role_conn_limit',true,'Role connection limits','Query',60),
        (12,'2-snap_pg_settings',true,'pg_settings','PG View',60),
        (13,'2-snap_pg_tbs_size',true,'Tablespace size','Query',60),
        (14,'2-snap_pg_user_deadline',true,'User deadline','Query',60),
        (15,'3-snap_pg_stat_statements',true,'pg_stat_statements','Extention',5),
        (16,'4-snap_pg_host_info',true,'UNIX commad of hostname information','System',60),
        (22,'3-pg_stat_statements_query_length',true,'Query length for pg_stat_statements
storage','Extention',10000),
        (23,'5-analyze_global_tables',true,'Analyze global tables','Maintenance',720);
```

12.3. Минимальный сценарий

В некоторых случаях, когда нагрузка БД экстремально велика, но нужно получать хотя бы общие параметры нагрузки, вы можете применить такие настройки в минимально возможной конфигурации.

```
-- Custom minimal
update
        __rds_pg_stats__.snap_modules
set
        "collect" = true
where
        mod_name like '0%'
        or mod_name like '1%';

update
        __rds_pg_stats__.snap_modules
set
        "collect" = false
where
        mod_name not like '0%'
        and mod_name not like '1%';
```

В этом случае будут собираться только метрики 0 и 1 уровня. При этом сценарии снимок создается примерно в 20 раз быстрее.

id	mod_name	collect	description	type	time_w
0	0-snap_list	[v]	Snapshot list	Global	1
1	1-snap_pg_database	[v]	pg_database	PG view	1
2	1-snap_pg_locks	[v]	pg_locks	PG View	1
3	1-snap_pg_stat_activity	[v]	pg_stat_activity	PG View	1
4	1-snap_pg_stat_archiver	[v]	pg_stat_archiver	PG View	1
5	1-snap_pg_stat_bgwriter	[v]	pg_stat_bgwriter	PG View	1
6	1-snap_pg_stat_database	[v]	pg_stat_database	PG View	1
7	2-snap_blocking_sessions	[v]	Blocking sessions	Query	1
8	2-snap_pg_database_age	[v]	Database age and txid info	Query	1
9	2-snap_pg_db_role_setting	[v]	Role settings	PG View	12
10	2-snap_pg_db_size	[v]	Database size	Query	1
11	2-snap_pg_role_conn_limit	[v]	Role connection limits	Query	12
12	2-snap_pg_settings	[v]	pg_settings	PG View	12
13	2-snap_pg_tbs_size	[v]	Tablespace size	Query	12
14	2-snap_pg_user_deadline	[v]	User deadline	Query	12
15	3-snap_pg_stat_statements	[v]	pg_stat_statements	Extention	1
16	4-snap_pg_host_info	[v]	UNIX commad of hostname information	System	12
17	4-snap_pg_stat_iotop	[v]	UNIX command of /proc/[nproc]/io	System	1
18	4-snap_pg_stat_progress_cluster	[v]	pg_stat_progress_cluster	PG View	1
19	4-snap_pg_stat_progress_create_index	[v]	pg_stat_progress_create_index	PG View	1
20	4-snap_pg_stat_progress_vacuum	[v]	pg_stat_progress_vacuum	PG View	1
21	4-snap_pg_stat_top	[v]	UNIX top command	System	1
22	3-pg_stat_statements_query_length	[v]	Query length for pg_stat_statements storage	Extention	50 000
23	5-analyze_global_tables	[v]	Analyze global tables	Maintenance	144

12.4. Автокастомизация

При сложных профилях нагрузки и при высоком вытеснении запросов из представления pg_stat_statements возникают ситуации, когда объёмы исторических данных экстремально велики. В этом случае нужно уменьшить длину запроса. При увеличении размера БД выше 15 ГБ автоматически изменяется значение кастомной метрики:

3-pg_stat_statements_query_length

до длины 2000 символов. Если вы считаете, что это много – вы можете изменить его в большую сторону вручную. Однако на практике для сложных профилей даже 5000 – это много. В любом случае вы можете поэкспериментировать с этим значением.

Автокастомизация срабатывает при смене суток, когда создаётся секционная корзина таблиц для нового дня в формате:

rds_data_YYYYMMDD

Пример работы автокастомизации:

DB grouth FILL (50000)	25											
DB grouth custom (500)	0,8											
Days	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
rds_data_DAY1	25	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8
rds_data_DAY2		25	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8
rds_data_DAY3			25	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8
rds_data_DAY4				25	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8
rds_data_DAY5					25	0,8	0,8	0,8	0,8	0,8	0,8	0,8
rds_data_DAY6						25	0,8	0,8	0,8	0,8	0,8	0,8
rds_data_DAY7							25	0,8	0,8	0,8	0,8	0,8
rds_data_DAY8								25	0,8	0,8	0,8	0,8
rds_data_DAY9									25	0,8	0,8	0,8
TOTAL	25	25,8	26,6	27,4	28,2	29	29,8	30,6	31,4	7,2	7,2	7,2
	Превысили порог 15 Гб в первый же день									Номализация		

Как видим, нормализация размера БД происходит не сразу, а по мере вытеснения корзин данных с избыточным размером.

Если говорить о размерах статистических данных, мы считаем, что нормально, когда статистические данные занимают 1-5 Гб, в крайнем случае 10Гб. Дальнейшее увеличение размера статистических данных может усложнить обслуживание статистических таблиц.

13. Приложение 6. Ручное формирование отчетов через SSH

Вы можете формировать отчеты АВР как из интерфейса WatcDOG/PgAM так и вручную.

1. Получить список последних 20 снимков:

```
sshpass -p '********' ssh -oStrictHostKeyChecking=no 'CDE\VTB70148294'@p0acqb-
pg2002lk.cde.vtb "sudo -u postgres psql -p 6432 -c 'select * from
__rds_pg_stats__.snap_list ORDER BY id DESC LIMIT 20;'"

id |            snap_ts             | snap_level
----+--------------------------------+------------
 76 | 2025-06-18 22:45:01.727566+03  | global
 75 | 2025-06-18 22:40:01.142464+03  | global
 74 | 2025-06-18 22:35:01.605527+03  | global
 73 | 2025-06-18 22:30:01.95234+03   | global
 72 | 2025-06-18 22:25:01.440351+03  | global
 71 | 2025-06-18 22:20:01.857947+03  | global
 70 | 2025-06-18 22:15:01.360298+03  | global
 69 | 2025-06-18 22:10:01.788351+03  | global
 68 | 2025-06-18 22:05:01.29899+03   | global
 67 | 2025-06-18 22:00:01.696723+03  | global
 66 | 2025-06-18 21:55:01.151465+03  | global
 65 | 2025-06-18 21:50:01.543821+03  | global
 64 | 2025-06-18 21:45:01.897988+03  | global
 63 | 2025-06-18 21:40:01.272141+03  | global
 62 | 2025-06-18 21:35:01.582261+03  | global
 61 | 2025-06-18 21:30:01.963225+03  | global
 60 | 2025-06-18 21:25:01.451624+03  | global
 59 | 2025-06-18 21:20:01.941738+03  | global
 58 | 2025-06-18 21:15:01.315942+03  | global
 57 | 2025-06-18 21:10:01.589381+03  | global
(20 rows)
```

2. Формирование отчета между снимками 57 и 76:

```
sshpass -p '******' ssh -oStrictHostKeyChecking=no 'CDE\VTB70148294'@p0acqb-
pg2002lk.cde.vtb 'sudo -u postgres -i psql -p 6432 -t -c "select * from
__rds_pg_stats__.snap_report_global(57,76)"' > awr_13.html
```

Файл awr_13.html – нужный вам файл отчета

14. Приложение 7. Добавление роли для формирования отчета AWR или для выполнения прямых запросов к статистическим данным.

По умолчании не предусмотрено дополнительных ролей для формирования отчета AWR или для выполнения прямых запросов к статистическим схемам бекенда, т.к. предполагается что отчёты прежде всего нужны администраторам БД. Однако вы можете создать специальную роль, которая может выполнять такие функции.

Пример такой роли с достаточными грантами и включение УЗ в роль ниже.

```
-- Роль для использования бекенда репозитория AWR
CREATE ROLE awr_role;
GRANT pg_monitor TO awr_role;
GRANT USAGE ON SCHEMA public TO awr_role;
GRANT USAGE ON SCHEMA __rds_pg_stats__ TO awr_role;
GRANT SELECT ON ALL TABLES IN SCHEMA __rds_pg_stats__ TO awr_role;
GRANT SELECT ON TABLE pg_authid TO awr_role;
GRANT SELECT ON TABLE pg_database TO awr_role;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA __rds_pg_stats__ TO awr_role;
GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA __rds_pg_stats__ TO awr_role;
GRANT EXECUTE ON FUNCTION "__rds_pg_stats__".snap_global(bool) TO awr_role;
--GRANT EXECUTE ON FUNCTION "__rds_pg_stats__".snap_report_global(int8,int8,text) TO
awr_role;
--GRANT EXECUTE ON FUNCTION "__rds_pg_stats__".snap_delete_obsolete(int) TO awr_role;

-- Включение УЗ мониторинга в роль awr_role
CREATE ROLE vtb1234567 PASSWORD '1234567';
ALTER ROLE vtb1234567 WITH LOGIN;
GRANT awr_role TO vtb1234567;
```

Пример формирования отчета специальной ролью:

1. Получение списка снимков:

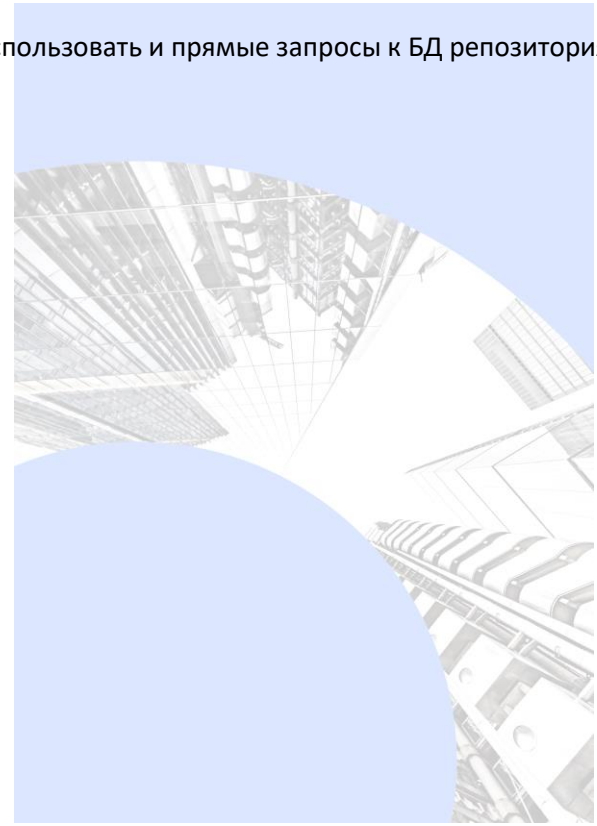
```
psql -h `hostname` -d postgres -U vtb1234567 -c 'select * from
__rds_pg_stats__.snap_list ORDER BY id DESC LIMIT 20;'
Password for user vtb1234567:
  id  |            snap_ts            | snap_level
------+-------------------------------+------------
 2815 | 2025-06-27 10:56:35.880194+03 | global
 2814 | 2025-06-27 10:55:01.739411+03 | global
 2813 | 2025-06-27 10:50:01.545831+03 | global
 2812 | 2025-06-27 10:45:01.35835+03  | global
 2811 | 2025-06-27 10:40:01.235391+03 | global
 2810 | 2025-06-27 10:35:01.072181+03 | global
 2809 | 2025-06-27 10:30:01.564101+03 | global
 2808 | 2025-06-27 10:25:01.379506+03 | global
 2807 | 2025-06-27 10:20:01.890768+03 | global
 2806 | 2025-06-27 10:15:01.323764+03 | global
 2805 | 2025-06-27 10:10:01.490051+03 | global
 2804 | 2025-06-27 10:05:01.743736+03 | global
```

```
 2803 | 2025-06-27 10:00:01.980723+03 | global
 2802 | 2025-06-27 09:55:01.507706+03 | global
 2801 | 2025-06-27 09:50:01.980873+03 | global
 2800 | 2025-06-27 09:45:01.529868+03 | global
 2799 | 2025-06-27 09:40:01.991138+03 | global
 2798 | 2025-06-27 09:35:01.452846+03 | global
 2797 | 2025-06-27 09:30:01.719427+03 | global
 2796 | 2025-06-27 09:25:01.213865+03 | global
(20 rows)
```

2. Формирование отчета:

```
psql -t -h `hostname` -d postgres -U vtb1234567 -c "select
__rds_pg_stats__.snap_report_global(2793,2812)" > awr_13.html
Password for user vtb1234567:
```

Под этой же ролью можно использовать и прямые запросы к БД репозитория в схеме __rds_pg_stats__.



15. Приложение 8. Планы запросов отчета PG AWR

Для прямых запросов планы могут быть построены как в GENERIC виде, так и через EXPLAIN ANALYZE.

Пример generic плана:

```
set plan_cache_mode = force_generic_plan;
```

Подготовка параметризованного запроса:

```
prepare query_01 as
select
	distinct "task"."question"."id", -- и т.д., пекст запроса в параметризованном виде
	...
		and "subQuestion"."type" in ($1, $2))))
order by
	"task"."question"."id" desc offset $3 rows fetch next $4 rows only;
```

Извлечение плана запроса:

```
explain execute query_01(null,null,null,null, null,null); -- столько NULL, сколько параметров в запросе.
```

Деалокация подготовленного запроса для построения нового.

```
deallocate query_01;
```

Начиная с 16 версии PostgreSQL появился более простой способ построения GENERIC планов параметризированных запросов.

```
explain (generic_plan) select
	distinct "task"."question"."id", -- и т.д., пекст запроса в параметризованном виде
	...
		and "subQuestion"."type" in ($1, $2))))
order by
	"task"."question"."id" desc offset $3 rows fetch next $4 rows only;
```

Для тестирования функций применяется трассирование через расширение auto_explain. Непосредственно план сохраняется в лог БД.

```
LOAD 'auto_explain';
SET auto_explain.log_min_duration = 0; -- чтобы логировать всё в данной сессии
SET auto_explain.log_nested_statements = on; -- для того чтобы логировать то что внутри процедуры.
```

После этого выполняем функцию и сохраняем лог БД.

16. Приложение 9. Методика тестирования PG AWR

Так как большинство проблем, связанных с формированием отчетов или извлечения данных из репозитория AWR связаны с ситуацией, когда размеры статистических данных имеют большой размер, то существует порог, при котором ситуация считается аварийной и мы не можем гарантировать извлечения данных за приемлемое время. Порог, при котором срабатывает автокастомизация – размер БД postres более 15Гб, то аварийный порог – считаем, как удвоенное значение, то есть 30Гб.

16.1. Нормализация размера БД статистики бекенда PG AWR

В методике тестирования вбираются тестовые системы с новой версией АВР или изменениями которые тестируются с размером БД близком к 15Гб или выше, но до порогового значения, которое считается аварийным.

Список серверов, попадающих в такой критерий:

HOST	STAT_SIZE	Type	PG AWR Version	PG AWR Version new
rrrpac-pgc006lk.test.vtb.ru	51.5 GiB	Cluster	3.0.2	4.1.3
rrrpac-pgc005lk.test.vtb.ru	51.5 GiB	Cluster	3.0.2	4.1.3
ltpcom-pgc004lk.test.vtb.ru	36.7 GiB	Cluster	3.0.2	4.1.3
ltpcom-pgc003lk.test.vtb.ru	36.7 GiB	Cluster	3.0.2	4.1.3
ifpcom-pgc001lk.test.vtb.ru	23.6 GiB	Standalon	3.0.2	4.1.3
ltrkkx-pg5221lp.test.vtb.ru	22.1 GiB	Cluster	3.0.2	4.1.3
ltrkkx-pg5222lp.test.vtb.ru	22.1 GiB	Cluster	3.0.2	4.1.3
ltcspc-pg5007lp.test.vtb.ru	21.2 GiB	Cluster	1.0.3	4.1.3
ltcspc-pg5006lp.test.vtb.ru	21.2 GiB	Cluster	1.0.3	4.1.3
ifaxfx-pgc004lk.test.vtb.ru	19.4 GiB	Cluster	3.0.2	4.1.3
ifaxfx-pgc005lk.test.vtb.ru	19.4 GiB	Cluster	3.0.2	4.1.3
hfibbs-pgc001lk.test.vtb.ru	17.4 GiB	Standalon	1.0.3	4.1.3
ltpgsq-pgc006lk.test.vtb.ru	15.4 GiB	Cluster	3.0.2	4.1.3
ltpgsq-pgc005lk.test.vtb.ru	15.4 GiB	Cluster	3.0.2	4.1.3
ltrkkx-pg5225lp.test.vtb.ru	15.2 GiB	Cluster	3.0.2	4.1.3
ltrkkx-pg5224lp.test.vtb.ru	15.2 GiB	Cluster	3.0.2	4.1.3

Здесь:

Столбец	Значение
HOST	Имя хоста БД
STAT_SIZE	Размер БД статистических данных
Type	Тип хоста БД. Cluster – член кластера или Satndalon – одиственный хост БД
PG AWR Version	Предыдущая версия PG AWR
PG AWR Version new	Тестируемая версия PG AWR

В результате, если параметры автокастомизации заданы верно, то по истечении 9 дней размер БД должен быть менее 15Гб, или меньше того что был ранее в диапазоне 15-30Гб.

16.2. Критерии успешности тестирования

Успешность тестирования определяется прежде всего размером БД статистических данных бекенда PG AWR.

Размер БД	**Значение**
1 - 5 Гб	Нормальный размер для большинства систем
5 - 15 Гб	Средний
15 – 30 Гб	Высокий, но допустимый размер
>30 Гб	Аварийный или кастомизированный. В случае когда владельцам системы объективно нужно иметь большие размеры статистических данных.

Также критерием успешности считается достаточно быстрое формирование типовых отчетов AWR на обычных нагрузках системы.

Время формирования отчета AWR	**Значение**
1 -15 сек	Нормальное
15 - 30 сек	Среднее
30 сек – 2 мин	Высокое, в обычной ситуации - аварийное
>2 мин	Высокое, аварийное, но возможное при высоких нагрузка на систему в случае аварий.