# Секционирование существующих таблиц с журналом Робота и Оркестратора для PostgreSQL[[1]](#footnote-1)

Для секционирования по месяцам нужно существующую таблицу подключить как раздел исторических данных.

Сначала для журнала робота. Создаём новую головную секционированную таблицу идентичной таблице Logs структуры под каким-нибудь новым именем, например, Logs2:

CREATE TABLE public."Logs2"

(

"Id" uuid NOT NULL,

"OrchTimestampUtc" timestamp without time zone NOT NULL,

"Event" integer,

"EntityId" text COLLATE pg\_catalog."default",

"UserId" text COLLATE pg\_catalog."default",

"TimestampUtc" timestamp without time zone,

"OperationKey" uuid,

"Signature" text COLLATE pg\_catalog."default",

"WorkerAdminName" text COLLATE pg\_catalog."default",

"Text" text COLLATE pg\_catalog."default",

"EventType" integer,

"IP" text COLLATE pg\_catalog."default",

"TenantId" text COLLATE pg\_catalog."default",

"AssignmentId" integer,

"RobotId" integer,

"RobotKey" text COLLATE pg\_catalog."default",

"ProjectId" integer,

"WorkerId" integer,

"Type" integer,

"ParsedDateEvent" timestamp without time zone,

"ParsedType" integer,

"ParsedElementName" text COLLATE pg\_catalog."default",

"ParsedElementId" text COLLATE pg\_catalog."default",

"ParsedElementClass" text COLLATE pg\_catalog."default",

"ParsedMessage" text COLLATE pg\_catalog."default",

"ScreenFilePath" text COLLATE pg\_catalog."default",

"OrchScreenFilePath" text COLLATE pg\_catalog."default",

"OrchScreenFilePathThumb" text COLLATE pg\_catalog."default",

"Node" text COLLATE pg\_catalog."default",

"ExtDataJson" text COLLATE pg\_catalog."default",

"RobotName" text COLLATE pg\_catalog."default",

"ProjectName" text COLLATE pg\_catalog."default",

"WorkerName" text COLLATE pg\_catalog."default",

"RdpUserName" text COLLATE pg\_catalog."default",

"LogType" integer NOT NULL

) partition by range ("OrchTimestampUtc");

Создаем секцию[[2]](#footnote-2) с данными на будущее (даты приведены для примера, нужно будет заменить на актуальные):

create table "Logs\_20230325" partition of "Logs2"

for values from ('2023-03-25') to ('2023-04-01');

Подготавливаем существующую таблицу с данными Logs – добавляем в неё ограничение на поле OrchTimestampUtc и проверяем это ограничение на данных таблицы:

begin;

set local statement\_timeout to '1s';

alter table "Logs" add constraint "Logs\_partbound\_check"

check ("OrchTimestampUtc" < '2023-03-25' and "OrchTimestampUtc" is not null) not valid;

commit;

alter table "Logs" validate constraint "Logs\_partbound\_check";

Если validate constraint выдаёт ошибки, то надо проверить данные/ограничение, и по возможности, провести корректировки (например, удалить некорректные данные).

Всю миграцию надо завершить до наступления даты 2023-04-01, либо удалить ограничение:

alter table "Logs" drop constraint "Logs\_partbound\_check";

и начать сначала.

Переименовываем таблицы и добавляем существующую таблицу (теперь она называется Logs\_archive) как секцию к головной секционированной таблице Logs:

begin;

set statement\_timeout to '1s';

alter table "Logs" rename to "Logs\_archive";

alter table "Logs2" rename to "Logs";

alter table "Logs" attach partition "Logs\_archive" for values from (MINVALUE) to ('2023-03-25');

commit;

Пока не настанет 2023-03-25, данные будут писаться в секцию Logs\_archive (в старую архивную таблицу). Начиная с 2023-03-25 – в секцию Logs\_20230325. Все запросы с фильтром по дате будут прозрачным для приложения образом идти через таблицу Logs.

Повторяем индексы на вновь созданной секции таблицы Logs\_20230325[[3]](#footnote-3):

CREATE INDEX "IX\_Logs\_20230325\_EntityId"

ON public."Logs\_20230325" USING btree

("EntityId" COLLATE pg\_catalog."default" ASC NULLS LAST)

TABLESPACE pg\_default;

CREATE INDEX "IX\_Logs\_20230325\_Event"

ON public."Logs\_20230325" USING btree

("Event" ASC NULLS LAST)

TABLESPACE pg\_default;

CREATE INDEX "IX\_Logs\_20230325\_OperationKey"

ON public."Logs\_20230325" USING btree

("OperationKey" ASC NULLS LAST)

TABLESPACE pg\_default;

CREATE INDEX "IX\_Logs\_20230325\_ProjectId"

ON public."Logs\_20230325" USING btree

("ProjectId" ASC NULLS LAST)

TABLESPACE pg\_default;

CREATE INDEX "IX\_Logs\_20230325\_RobotId"

ON public."Logs\_20230325" USING btree

("RobotId" ASC NULLS LAST)

TABLESPACE pg\_default;

CREATE INDEX "IX\_Logs\_20230325\_RobotId\_OperationKey\_TimestampUtc"

ON public."Logs\_20230325" USING btree

("RobotId" ASC NULLS LAST, "OperationKey" ASC NULLS LAST, "TimestampUtc" DESC NULLS LAST)

INCLUDE("Type", "LogType")

TABLESPACE pg\_default;

CREATE INDEX "IX\_Logs\_20230325\_RobotKey"

ON public."Logs\_20230325" USING btree

("RobotKey" COLLATE pg\_catalog."default" ASC NULLS LAST)

TABLESPACE pg\_default;

CREATE INDEX "IX\_Logs\_20230325\_UserId"

ON public."Logs\_20230325" USING btree

("UserId" COLLATE pg\_catalog."default" ASC NULLS LAST)

TABLESPACE pg\_default;

CREATE INDEX "IX\_Logs\_20230325\_WorkerId"

ON public."Logs\_20230325" USING btree

("WorkerId" ASC NULLS LAST)

TABLESPACE pg\_default;

Повторяем аналогичные действия для журнала оркестратора:

CREATE TABLE public."OrchEvents2"

(

"Id" uuid NOT NULL,

"OrchTimestampUtc" timestamp without time zone NOT NULL,

"Event" integer NOT NULL,

"EntityId" text COLLATE pg\_catalog."default",

"UserId" text COLLATE pg\_catalog."default",

"OperationKey" uuid,

"Signature" text COLLATE pg\_catalog."default",

"WorkerAdminName" text COLLATE pg\_catalog."default",

"Text" text COLLATE pg\_catalog."default",

"EventType" integer,

"IP" text COLLATE pg\_catalog."default",

"TenantId" text COLLATE pg\_catalog."default",

"AssignmentId" integer,

"LogType" integer NOT NULL DEFAULT 1

) partition by range ("OrchTimestampUtc");

create table "OrchEvents\_20230325" partition of "OrchEvents2"

for values from ('2023-03-25') to ('2023-04-01');

begin;

set local statement\_timeout to '1s';

alter table "OrchEvents" add constraint "OrchEvents\_partbound\_check"

check ("OrchTimestampUtc" < '2023-03-25' and "OrchTimestampUtc" is not null) not valid;

commit;

alter table "OrchEvents" validate constraint "OrchEvents\_partbound\_check";

begin;

set statement\_timeout to '1s';

alter table "OrchEvents" rename to "OrchEvents\_archive";

alter table "OrchEvents2" rename to "OrchEvents";

alter table "OrchEvents" attach partition "OrchEvents\_archive" for values from (MINVALUE) to ('2023-03-25');

commit;

CREATE INDEX "IX\_OrchEvents\_20230325\_AllOrchTimestampUtc"

ON public."OrchEvents\_20230325" USING btree

("TenantId" COLLATE pg\_catalog."default" ASC NULLS LAST, "Event" ASC NULLS LAST, "UserId" COLLATE pg\_catalog."default" ASC NULLS LAST, "EventType" ASC NULLS LAST, "OrchTimestampUtc" DESC NULLS LAST)

INCLUDE("EntityId", "OperationKey", "Signature", "WorkerAdminName", "Text", "IP", "AssignmentId", "LogType")

TABLESPACE pg\_default;

CREATE INDEX "IX\_OrchEvents\_20230325\_EntityId\_OrchTimestampUtc"

ON public."OrchEvents\_20230325" USING btree

("EntityId" COLLATE pg\_catalog."default" ASC NULLS LAST, "OrchTimestampUtc" ASC NULLS LAST)

TABLESPACE pg\_default;

CREATE INDEX "IX\_OrchEvents\_20230325\_Event\_OrchTimestampUtc"

ON public."OrchEvents\_20230325" USING btree

("Event" ASC NULLS LAST, "OrchTimestampUtc" ASC NULLS LAST)

TABLESPACE pg\_default;

CREATE INDEX "IX\_OrchEvents\_20230325\_UserId\_OrchTimestampUtc"

ON public."OrchEvents\_20230325" USING btree

("UserId" COLLATE pg\_catalog."default" ASC NULLS LAST, "OrchTimestampUtc" ASC NULLS LAST)

TABLESPACE pg\_default;

Секций с новыми данными сразу можно добавить с запасом несколько. Дальнейшее их создание и удаление старых секций можно автоматизировать на основе задачи по расписанию.

## Автоматическое создание новых секций

Устанавливаем расширение pg\_cron[[4]](#footnote-4)

Копируем пакет с pg\_cron в расшаренную папку, из которой будет производиться его установка. Переходим в эту папку:

# cd /srv/samba/shared/install/postgresql-13

Устанавливаем пакет:

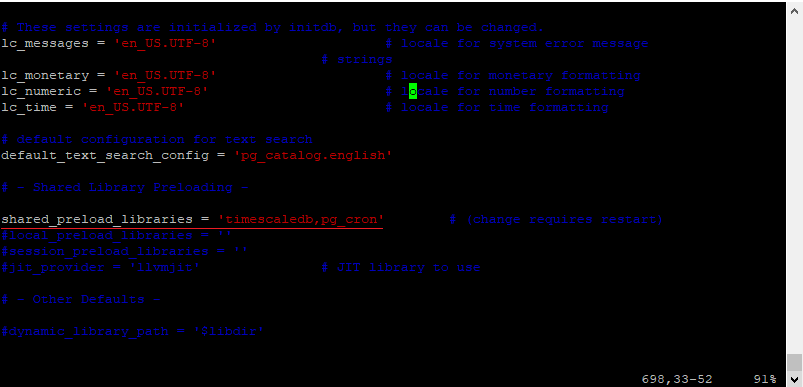
# dnf -y install 'pg\_cron\_13-1.3.0-1.rhel8.x86\_64.rpm'

Вносим изменения в файл postgresql.conf:

Устанавливаем (или дописываем) 'pg\_cron' в shared\_preload\_libraries:

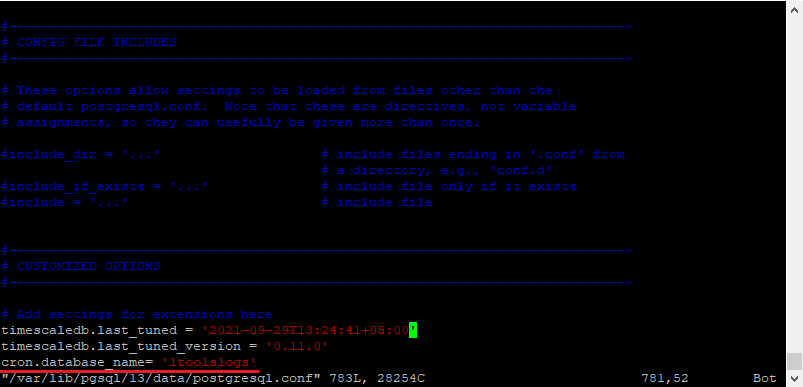
# vim /var/lib/pgsql/13/data/postgresql.conf

shared\_preload\_libraries = 'pg\_cron'



Добавляем в конец файла строку:

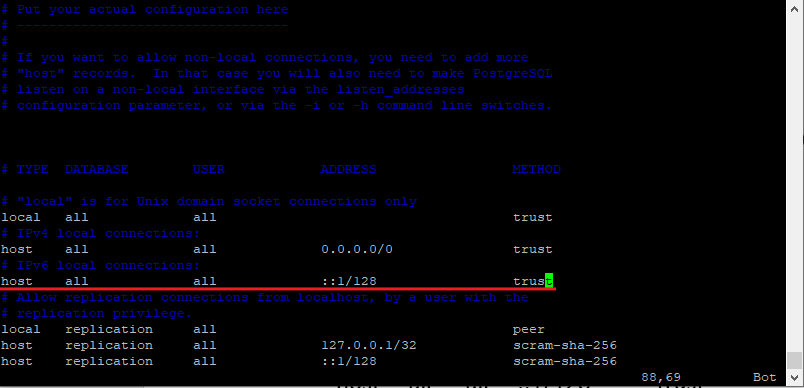
cron.database\_name= 'ltoolslogs'



Даем возможность локально подключаться к БД без пароля будущему заданию – вносим изменение в pg\_hba.conf:

# vim /var/lib/pgsql/13/data/pg\_hba.conf

host all all ::1/128 trust



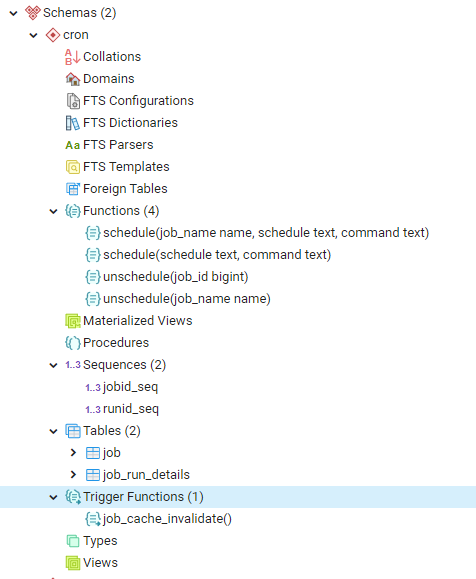
Перезапускаем службу:

# systemctl restart postgresql-13

Создаем в БД ltoolslogs расширение pg\_cron. Можно через какой-то менеджер для работы с PostgreSQL, например, pgAdmin4. Или из командной строки:

# sudo -i -u postgres psql -d ltoolslogs -c 'CREATE EXTENSION pg\_cron'

После этого проверим, что в БД ltoolslogs появились схема cron и необходимые для заданий объекты БД:



В БД ltoolslogs создаем процедуру для создания секций:

CREATE OR REPLACE PROCEDURE public.create\_logs\_part(

)

LANGUAGE 'plpgsql'

AS $BODY$

DECLARE

y integer;

m integer;

d1 timestamp;

d2 timestamp;

part varchar(100);

BEGIN

y := date\_part('year', CURRENT\_DATE);

m := date\_part('month', CURRENT\_DATE) + 1;

d1 = make\_date(y, m, 1);

IF m = 12 THEN

d2 = make\_date(y + 1, 1, 1);

ELSE

d2 = make\_date(y, m + 1, 1);

END IF;

part := format('%s%s', y, CASE WHEN m > 9 THEN format('%s', m) ELSE format('0%s', m) END);

if NOT EXISTS (

SELECT FROM pg\_catalog.pg\_class c

JOIN pg\_catalog.pg\_namespace n ON n.oid = c.relnamespace

WHERE c.relname = format('Logs\_%s', part)) then

EXECUTE format('create table "Logs\_%s" partition of "Logs" for values from (''%s'') to (''%s'')',

part, to\_char(d1, 'yyyy-mm-dd'), to\_char(d2, 'yyyy-mm-dd'));

EXECUTE format('CREATE INDEX "IX\_Logs\_%1$s\_EntityId"

ON "Logs\_%1$s" USING btree

("EntityId" COLLATE pg\_catalog."default" ASC NULLS LAST)

TABLESPACE pg\_default;

CREATE INDEX "IX\_Logs\_%1$s\_Event"

ON public."Logs\_%1$s" USING btree

("Event" ASC NULLS LAST)

TABLESPACE pg\_default;

CREATE INDEX "IX\_Logs\_%1$s\_OperationKey"

ON "Logs\_%1$s" USING btree

("OperationKey" ASC NULLS LAST)

TABLESPACE pg\_default;

CREATE INDEX "IX\_Logs\_%1$s\_ProjectId"

ON "Logs\_%1$s" USING btree

("ProjectId" ASC NULLS LAST)

TABLESPACE pg\_default;

CREATE INDEX "IX\_Logs\_%1$s\_RobotId"

ON "Logs\_%1$s" USING btree

("RobotId" ASC NULLS LAST)

TABLESPACE pg\_default;

CREATE INDEX "IX\_Logs\_%1$s\_RobotId\_OperationKey\_TimestampUtc"

ON "Logs\_%1$s" USING btree

("RobotId" ASC NULLS LAST, "OperationKey" ASC NULLS LAST, "TimestampUtc" DESC NULLS LAST)

INCLUDE("Type", "LogType")

TABLESPACE pg\_default;

CREATE INDEX "IX\_Logs\_%1$s\_RobotKey"

ON "Logs\_%1$s" USING btree

("RobotKey" COLLATE pg\_catalog."default" ASC NULLS LAST)

TABLESPACE pg\_default;

CREATE INDEX "IX\_Logs\_%1$s\_UserId"

ON "Logs\_%1$s" USING btree

("UserId" COLLATE pg\_catalog."default" ASC NULLS LAST)

TABLESPACE pg\_default;

CREATE INDEX "IX\_Logs\_%1$s\_WorkerId"

ON "Logs\_%1$s" USING btree

("WorkerId" ASC NULLS LAST)

TABLESPACE pg\_default;', part);

end if;

if NOT EXISTS (

SELECT FROM pg\_catalog.pg\_class c

JOIN pg\_catalog.pg\_namespace n ON n.oid = c.relnamespace

WHERE c.relname = format('OrchEvents\_%s', part)) then

EXECUTE format('create table "OrchEvents\_%s" partition of "OrchEvents" for values from (''%s'') to (''%s'')',

part, to\_char(d1, 'yyyy-mm-dd'), to\_char(d2, 'yyyy-mm-dd'));

EXECUTE format('CREATE INDEX "IX\_OrchEvents\_%1$s\_AllOrchTimestampUtc"

ON "OrchEvents\_%1$s" USING btree

("TenantId" COLLATE pg\_catalog."default" ASC NULLS LAST, "Event" ASC NULLS LAST, "UserId" COLLATE pg\_catalog."default" ASC NULLS LAST, "EventType" ASC NULLS LAST, "OrchTimestampUtc" DESC NULLS LAST)

INCLUDE("EntityId", "OperationKey", "Signature", "WorkerAdminName", "Text", "IP", "AssignmentId", "LogType")

TABLESPACE pg\_default;

CREATE INDEX "IX\_OrchEvents\_%1$s\_EntityId\_OrchTimestampUtc"

ON "OrchEvents\_%1$s" USING btree

("EntityId" COLLATE pg\_catalog."default" ASC NULLS LAST, "OrchTimestampUtc" ASC NULLS LAST)

TABLESPACE pg\_default;

CREATE INDEX "IX\_OrchEvents\_%1$s\_Event\_OrchTimestampUtc"

ON "OrchEvents\_%1$s" USING btree

("Event" ASC NULLS LAST, "OrchTimestampUtc" ASC NULLS LAST)

TABLESPACE pg\_default;

CREATE INDEX "IX\_OrchEvents\_%1$s\_UserId\_OrchTimestampUtc"

ON "OrchEvents\_%1$s" USING btree

("UserId" COLLATE pg\_catalog."default" ASC NULLS LAST, "OrchTimestampUtc" ASC NULLS LAST)

TABLESPACE pg\_default;', part);

end if;

END

$BODY$;

В БД ltoolslogs создаем задание, которое будет каждые 30 минут запускать созданную процедуру:

select cron.schedule('\*/30 \* \* \* \*','CALL public.create\_logs\_part ();');

Для пересоздания задания нужно удалить запись о нем из таблицы cron."job" и создать задание заново.

1. https://postgrespro.ru/docs/postgresql/10/ddl-partitioning [↑](#footnote-ref-1)
2. Можно сразу несколько секций создать, с запасом [↑](#footnote-ref-2)
3. Если несколько секция, то для каждой секции [↑](#footnote-ref-3)
4. Установка может отличаться для разных дистрибутивов Linux [↑](#footnote-ref-4)