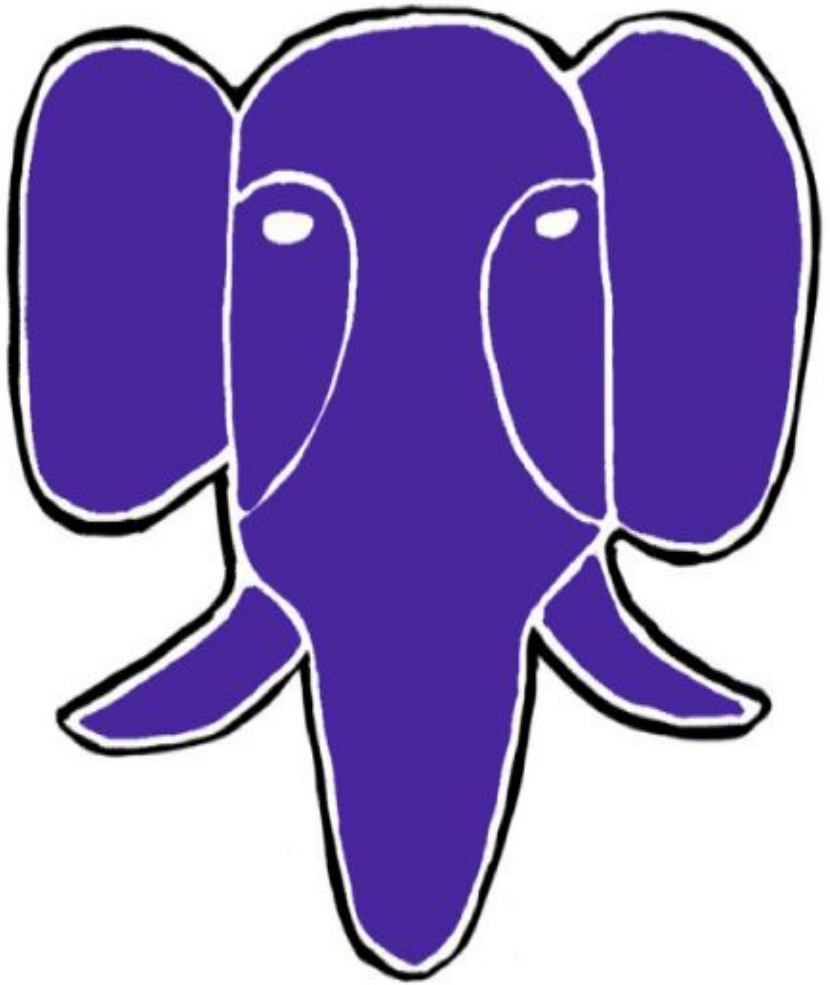
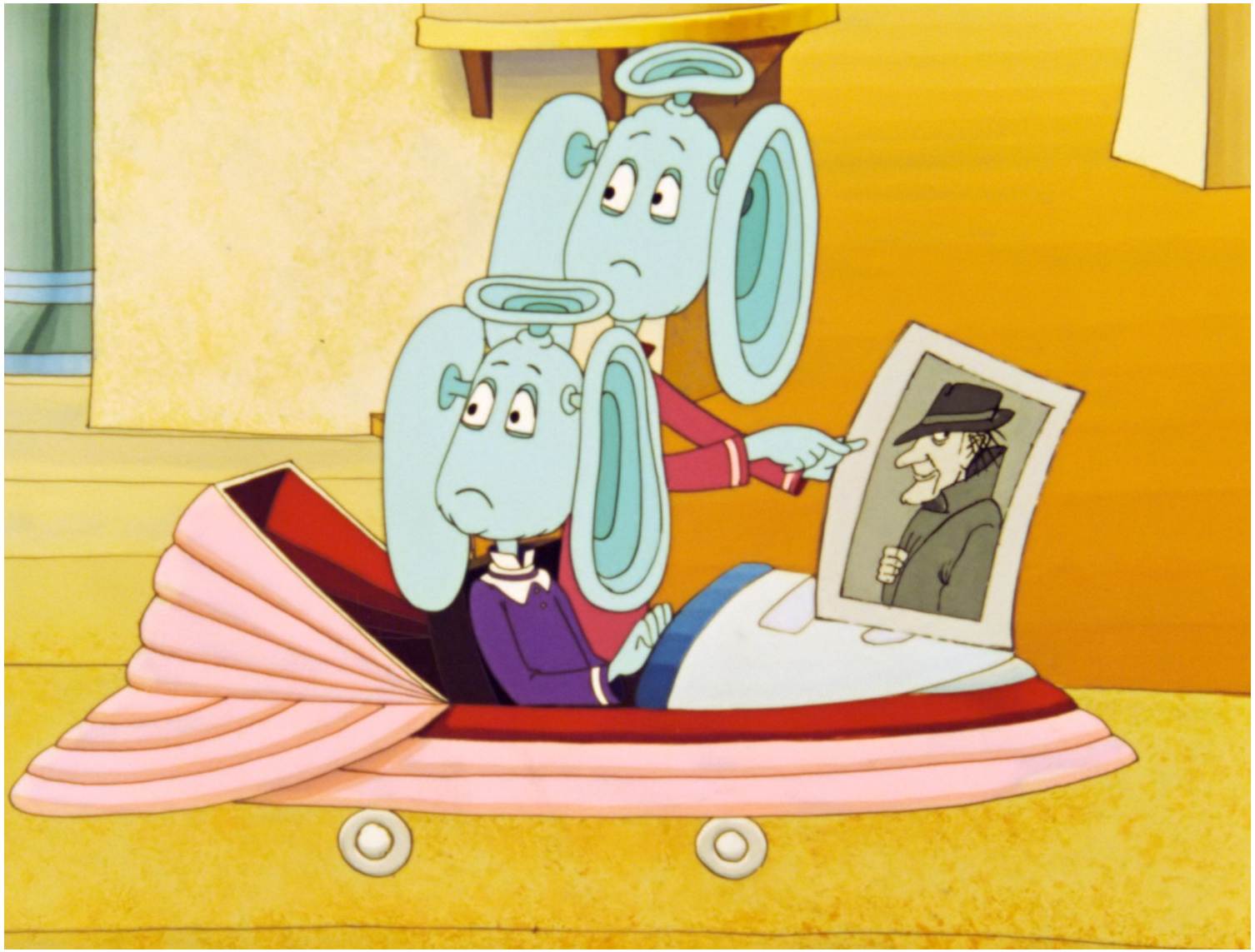
Евгений Аристов

DATABASE SCHEMA TABLE





Правила вебинара

Задаем вопрос в чат

Вопросы вижу, отвечу в момент логической паузы

Если есть вопрос голосом - поставьте знак ? в чат

Если остались вопросы, можно их задать на следующем занятии

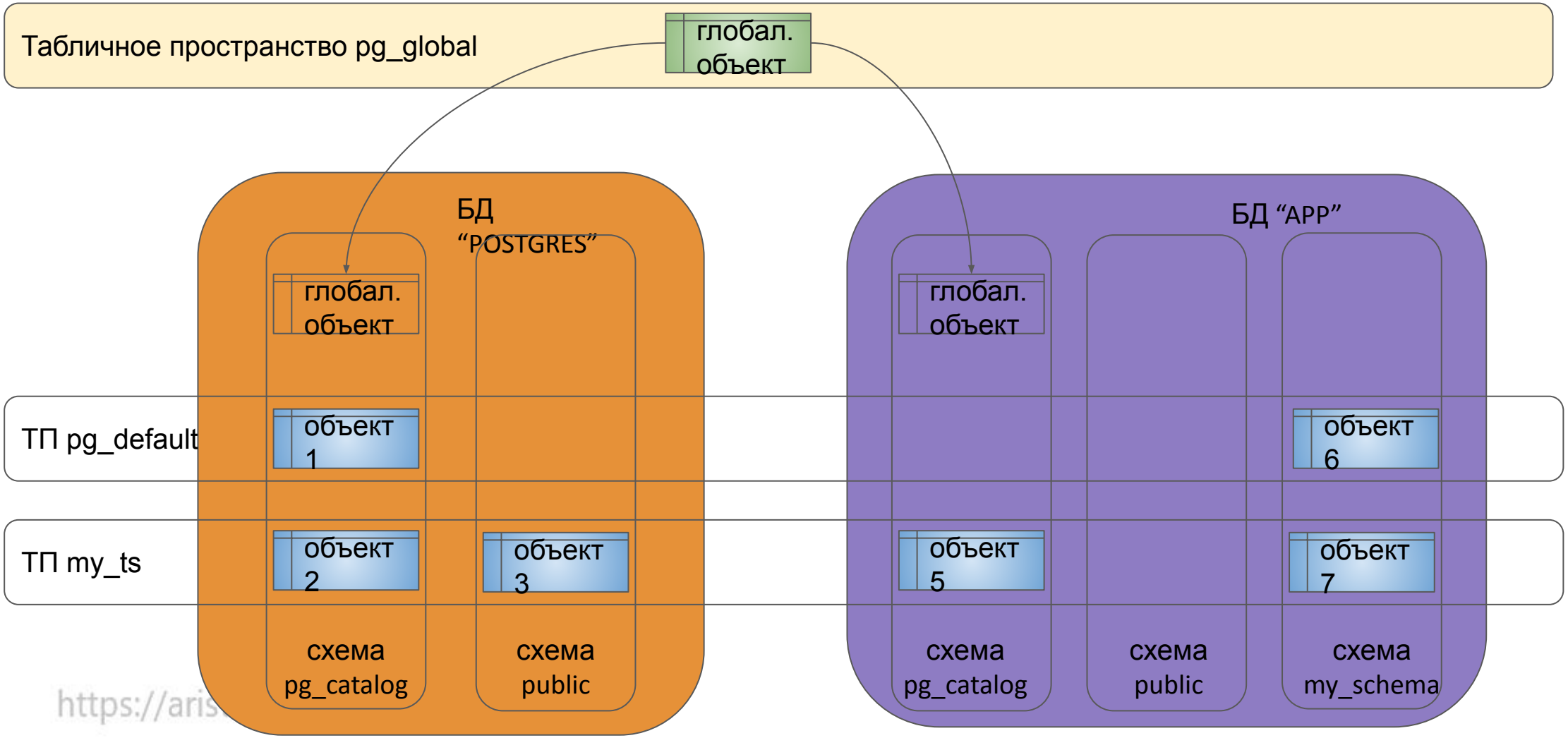
Маршрут вебинара

1. Табличное пространство
2. Уровни вложенности объектов
3. Database
4. Schema
5. Table
6. Constraint
7. Именование объектов
8. search_path
9. DEFAULT vs IDENTITY

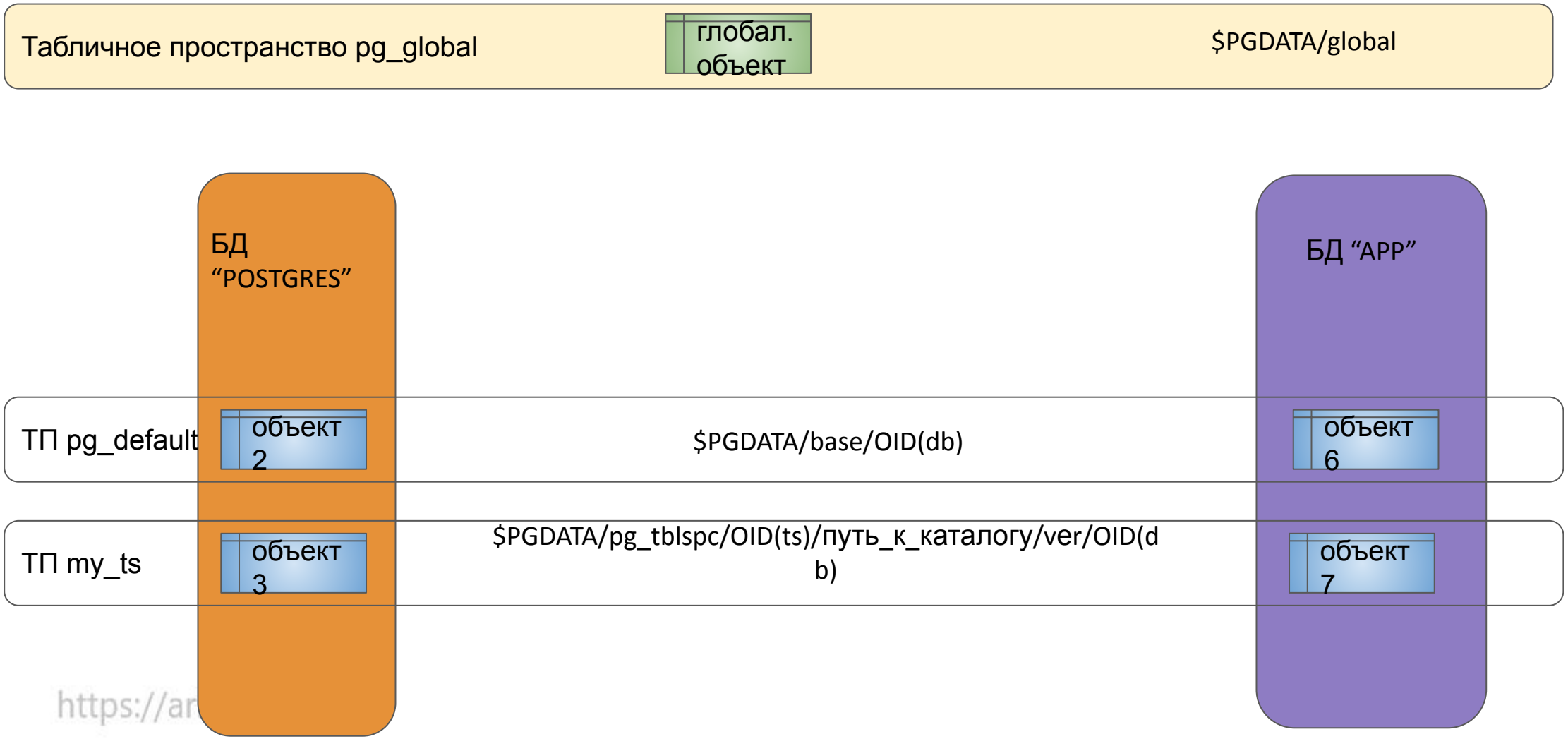
Табличные пространства

https://aristov.tech

Устройство ТП



Табличное пространство. Где файлы?



Табличное пространство. Оптимизация

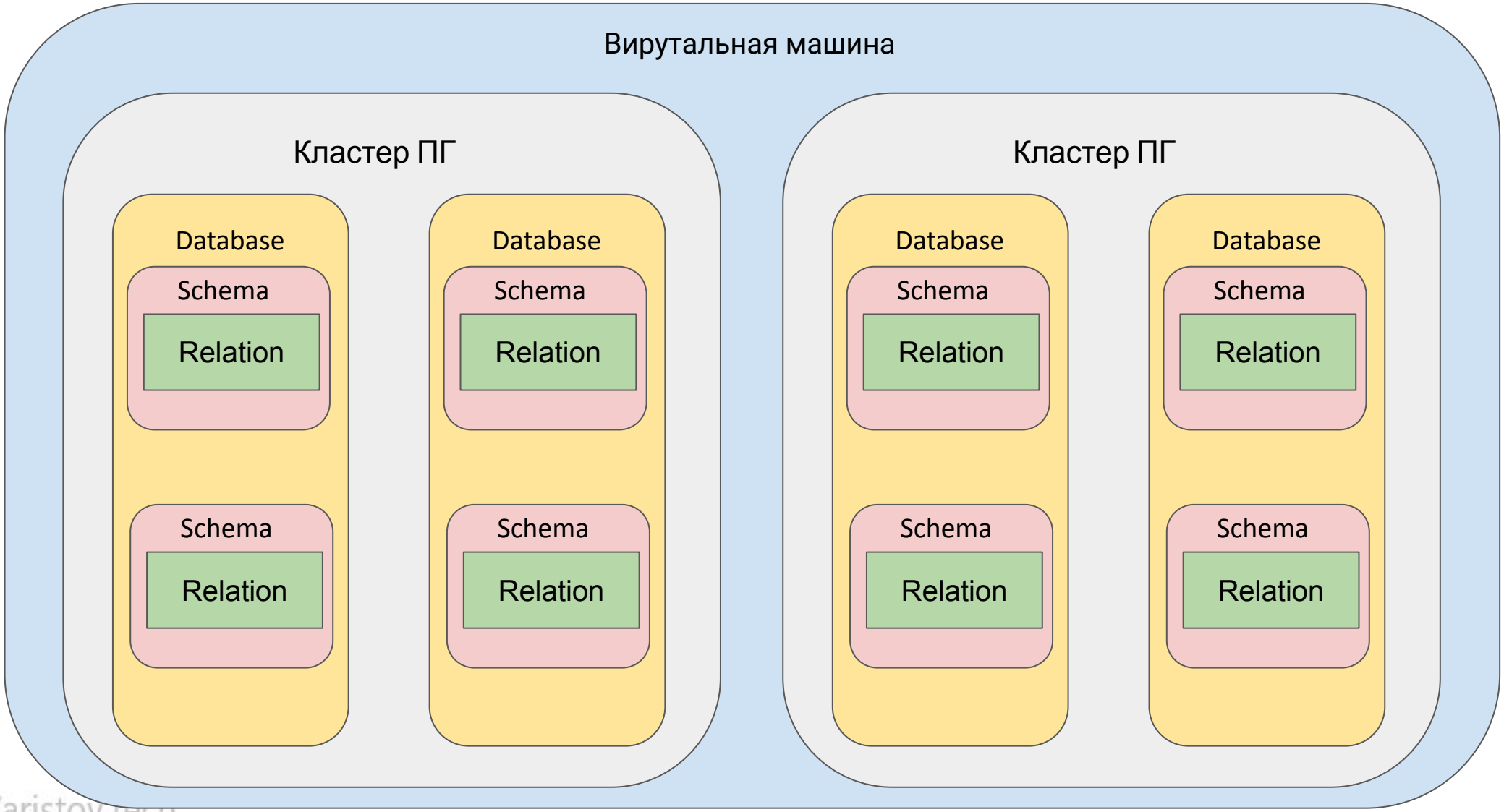
Варианты:

- отдельные объекты (БД, таблицы) размещаем в разных ТП, расположенных на разных дисках, дисковых массивах - таким образом распараллеливаем нагрузку
- можно из оперативной памяти часть смонтировать как файловую систему, создать ТП и держать там, например, материализованные представления, временные таблицы, индексы
- вариант установить локально на сервер ssd диск под такие же некритичные данные

Устройство ПГ

https://aristov.tech

Общее логическое устройство PostgreSQL



Database

- DDL
- Является контейнером самого верхнего уровня
- По умолчанию в любом кластере есть как минимум 3 БД:
  - postgres
  - template0
  - template1
- Присутствует на логическом и физическом уровне

template0

- для восстановления из резервной копии
- по умолчанию даже нет прав на connect
- лучше всего не создавать в ней никаких объектов

template1

- используется как шаблон для создания новых баз данных
- в нем имеет смысл делать некие действия, которые не хочется делать каждый раз при создании новых баз данных
- например create extension или create schema
- но (как мне кажется) лучше не создавать объектов, так как для других пользователей это будет неочевидно
- лучше сделать свой шаблон для создания других БД

https://aristov.tech

postgres

- первая база данных для регулярной работы
- создается по умолчанию
- хорошая практика - также не использовать, но и не удалять - иногда нужна для различных утилит

Create Database

https://www.postgresql.org/docs/current/sql-createdatabase.html

https://aristov.tech

Create Schema

Контейнер 2 уровня.

CREATE SCHEMA **IF NOT EXISTS** *имя_схемы* [AUTHORIZATION *указание_роли*]

https://www.postgresql.org/docs/current/sql-createschema.html

По умолчанию используется схема PUBLIC

Create Table

Контейнер 3 уровня.

CREATE [{ TEMPORARY | TEMP } | UNLOGGED] TABLE [IF NOT EXISTS] *имя_таблицы* ([
{ *имя_столбца тип_данных* [*ограничение_столбца* [...]]

https://www.postgresql.org/docs/current/sql-createtable.html

Для изменения DDL есть команда ALTER - разберём на следующем занятии

Ни в коем случае нельзя делать DROP -> CREATE !!!

Виды отношений в БД

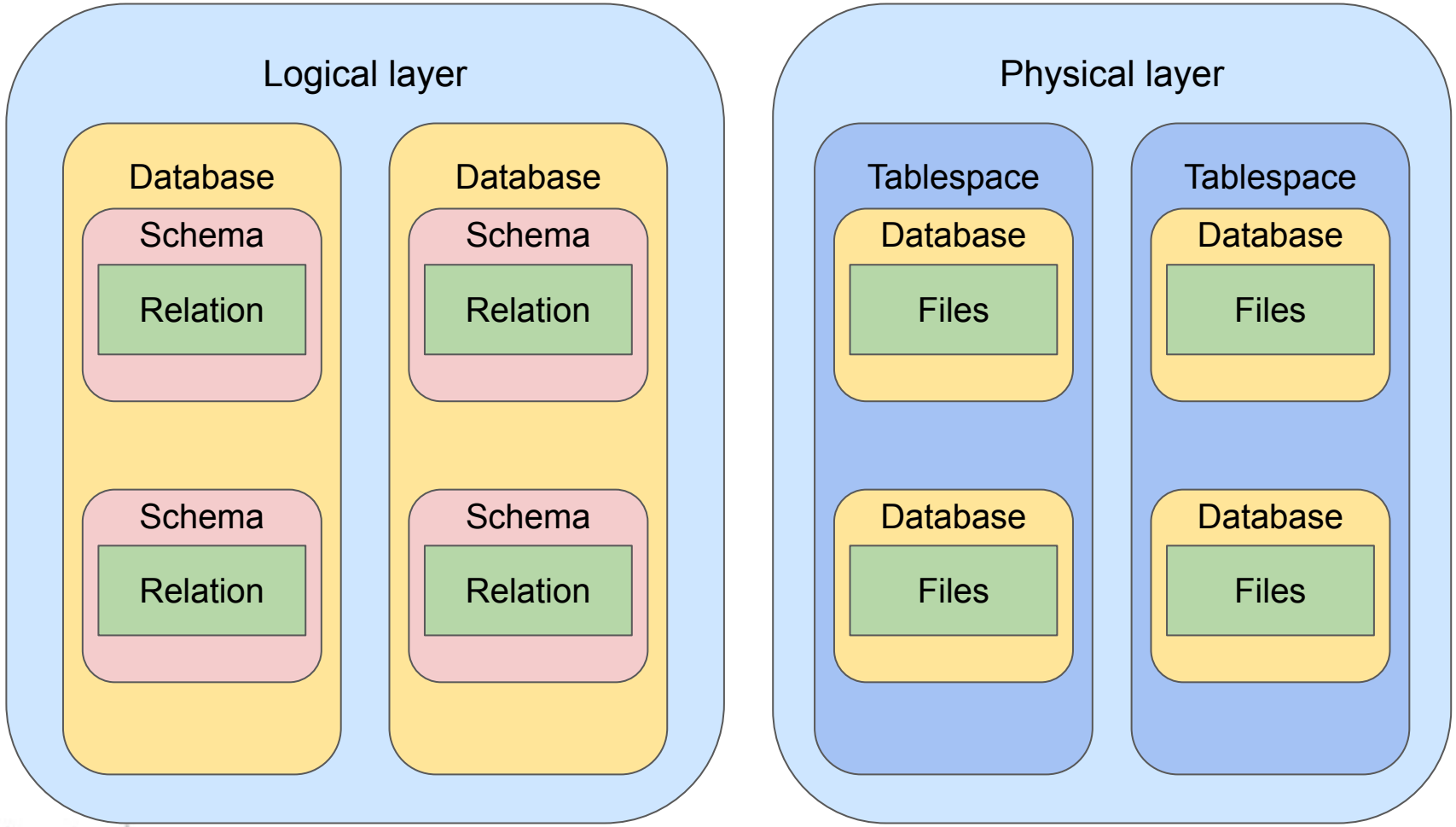
r = ordinary table

Кроме простых таблиц, также существуют и другие отношения:
i = index,
S = sequence,
v = view,
m = materialized view,
c = composite type,
t = TOAST table,
f = foreign table,

https://www.postgresql.org/docs/15/catalog-pg-class.html

https://aristov.tech

Соответствие физического и логического уровней PostgreSQL



Constraint

https://aristov.tech

Constraint/Ограничения

https://www.postgresql.org/docs/current/ddl-constraints.html

CHECK

NOT NULL

UNIQUE

PRIMARY KEY

FOREIGN KEY - каскадное удаление, запрет

Именование объектов

https://aristov.tech

Именование объектов

PostgreSQL автоматически текст приводит к нижнему регистру.

2 популярных варианта:

camelCase -> camelcase

snake_case -> snake_case

Можно использовать кавычки - тогда имя будет сохранено как в оригинале:

"camelCase" -> "camelCase"

“русское имя”

search_path

search_path

Без указания схемы у таблицы, как определить в какой схеме создавать объекты или в какой схеме искать?

Используется переменная search_path:

SHOW search_path;

Посмотрим на практике.

Хорошая практика указывать схему, чтобы не было разночтений.

DEFAULT vs IDENTITY

https://aristov.tech

DEFAULT vs IDENTITY

Классический SERIAL -> INT NOT NULL DEFAULT(nextval('test_i_seq'))

```
CREATE TABLE color (
    color_id SERIAL,
    color_name VARCHAR NOT NULL
);
```

с 10 версии рекомендация использовать IDENTITY

```
CREATE TABLE color (
    color_id INT GENERATED ALWAYS AS IDENTITY,
    color_name VARCHAR NOT NULL
);
```

По факту разница одна - IDENTITY только для 1 таблицы, SERIAL можем использовать сквозную нумерацию в разных таблицах.

Посмотрим на практике.

Итоги

https://aristov.tech

Итоги

Остались ли вопросы?

Увидимся на следующем занятии

https://aristov.tech

Спасибо за внимание!

Когда дальше и куда?
В чате напишу
материалы для бесплатного доступа будут появляться на ютубе

Аристов Евгений