OTUS

# PostgreSQL для администраторов баз данных и разработчиков

otus.ru


**Проверить, идет ли запись**

# Меня хорошо видно && слышно?

# Физический уровень PostgreSQL

**Антон Золотов**

Программист, разработчик баз данных.
Общий стаж разработки программного обеспечения более 35 лет, разработкой БД занимаюсь около 20 лет.

Предметные области:
•ERP;
•складская логистика;
•системы анализа медиаданных;
•системы поддержки принятия решений.

- сертифицированный тренер PostgreSQL
- преподаватель курса “PostgreSQL” в OTUS

# Правила вебинара

Активно участвуем

Задаем вопросы в чат

Вопросы вижу в чате,
могу ответить не сразу

## Условные обозначения

Индивидуально

Время, необходимое на активность

Пишем в чат

Говорим голосом

Документ

Ответьте себе или задайте вопрос

# Маршрут вебинара

# Цели вебинара

1. Получить представление об устройстве PostgreSQL
2. Узнать основные процессы PostgreSQL и структуру объектов в памяти
3. Понять, как PostgreSQL работает с данными на физическом уровне

# Подключение к PostgreSQL. Инструментарий.

# Инструментарий

PostgreSQL – серверная программа, не имеющая собственного UI. Что бы иметь возможность взаимодействовать с сервером – нужно клиентское приложение

- psql – «штатный» терминал для работы с PostgreSQL
- PgAdmin, DataGrip, Dbeaver – GUI-клиенты

# Клиент psql

Запуск:

```
psql -h <сервер> -d <база данных> -p <порт> -U <пользователь>
```

Выполнение команд:

1. Запросы SQL

```
SELECT * FROM some_table;
```

Заканчиваются символом «;»

2. Собственные команды psql:

```
\conninfo
```

Предваряются символом «\»

# Клиент psql

**Connection options:**

| | |
|---|---|
| -h, --host=HOSTNAME | database server host or socket directory |
| -p, --port=PORT | database server port |
| -U, --username=USERNAME | database user name |
| -w, --no-password | never prompt for password |
| -W, --password | force password prompt (should happen automatically) |

# Клиент psql

**General options:**

| | |
|---|---|
| -c, --command | =COMMAND<br>run only single command (SQL or internal) and exit |
| -d, --dbname | =DBNAME<br>database name to connect to |
| -f, --file | =FILENAME<br>execute commands from file, then exit |
| -l, --list | list available databases, then exit |
| -v, --set | =, --variable =NAME =VALUE<br>set psql variable NAME to VALUE<br>(e.g., -v ON_ERROR_STOP=1) |
| -V, --version | output version information, then exit |
| -X, --no-psqlrc | do not read startup file (~/.psqlrc) |

# Клиент psql

**Input and output options**:

| | |
|---|---|
| -a, --echo-all | echo all input from script |
| -b, --echo-errors | echo failed commands |
| -e, --echo-queries | echo commands sent to server |
| -E, --echo-hidden | display queries that internal commands generate |
| -L, --log-file | =FILENAME<br>send session log to file |
| -n, --no-readline | disable enhanced command line editing (readline) |
| -o, --output | =FILENAME send query results to file (or \|pipe) |
| -q, --quiet | run quietly (no messages, only query output) |
| -s, --single-step | single-step mode (confirm each query) |
| -S, --single-line | single-line mode (end of line terminates SQL command) |

# Клиент psql

**Output format options:**

| | |
|---|---|
| -A, --no-align | unaligned table output mode |
| --csv | CSV (Comma-Separated Values) table output mode |
| -F, --field-separator | =STRING<br>field separator for unaligned output (default: "\|") |
| -H, --html | HTML table output mode |
| -P, --pset | =VAR[=ARG]<br>set printing option VAR to ARG (see \pset command) |
| -R, --record-separator= | STRING<br>record separator for unaligned output (default: newline) |
| -t, --tuples-only | print rows only |
| -T, --table-attr | =TEXT<br>set HTML table tag attributes (e.g., width, border) |
| -x, --expanded | turn on expanded table output |
| -z, --field-separator-zero | set field separator for unaligned output to zero byte |
| -0, --record-separator-zero | set record separator for unaligned output to zero byte |

# Клиент psql

При запуске psql выполняются команды, записанные в 2-х файлах:

- системном - /usr/local/pgsql/etc/psqlrc
- пользовательском - .psqlrc из домашнего каталога пользователя. Расположение пользовательского файла можно изменить, задав переменную окружения PSQLRC

В эти файлы могут быть записаны команды настройки psql – например, можно изменить приглашение, включить вывод времени выполнения команд, и т.д.

# Подключение к PostgreSQL

Если PostgreSQL работает под управлением ОС из *nix – семейства, возможно подключение не только по TCP-IP, но и по unix-socket

Возможности подключения к PostgreSQL определяются настройками, сохраняемыми в конфигурационных файлах:

pg_hba.conf - описывает настройки подключения (“host-based authentication”)

postgresql.conf - содержит основные настройки, в т.ч. порт PostgreSQL и адрес(а) с которых разрешены подключения

Узнать расположение этих файлов можно, выполнив запросы

```
SHOW hba_file;
SHOW config_file;
```

# Базы данных кластера

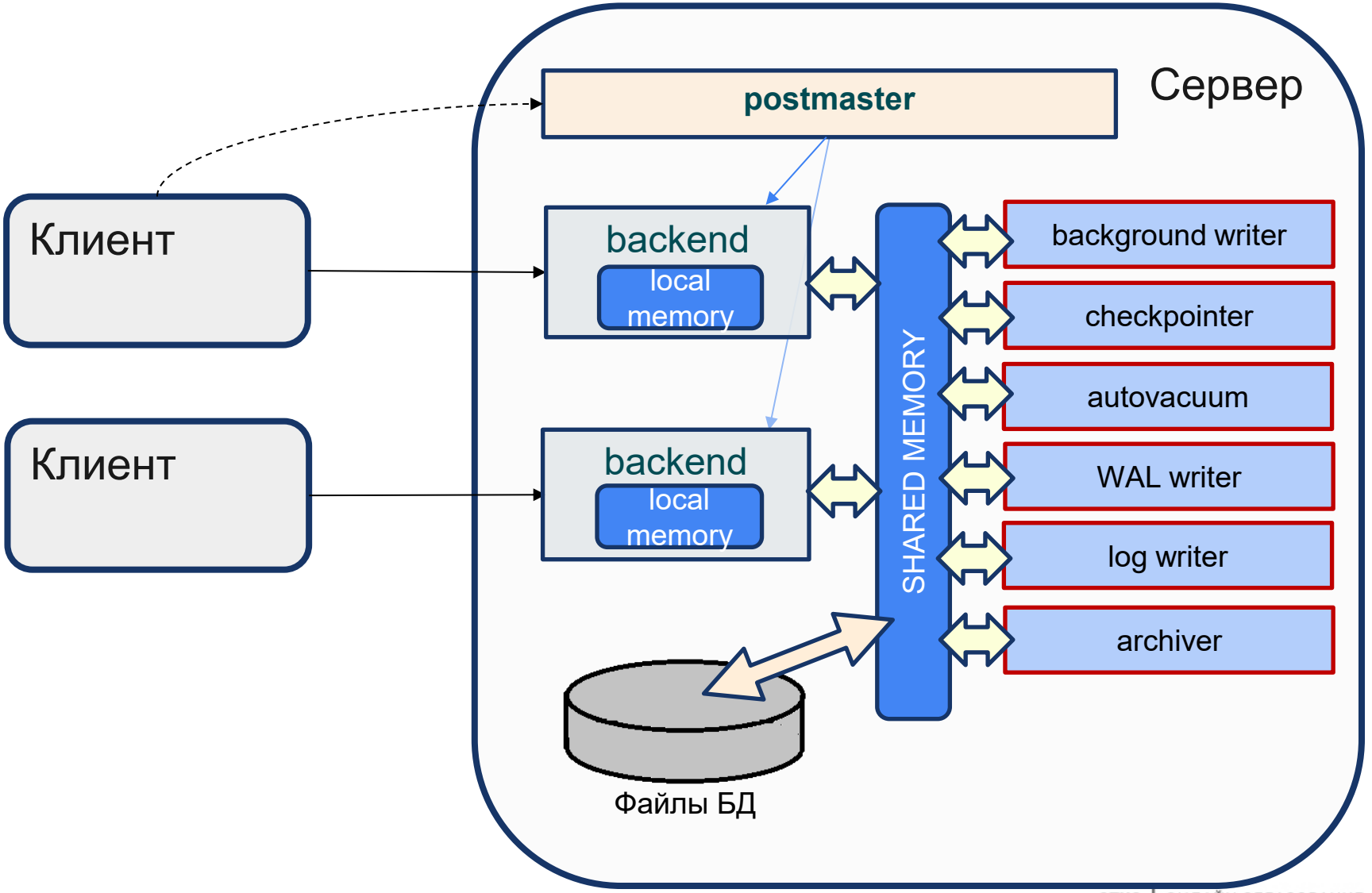
- template0 - неизменяемый шаблон
- template1 – изменяемый шаблон
- postgres – БД «для первого подключения»

# Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

# Взаимодействие клиента и сервера



# postmaster

**postgres server process**

- первый процесс postgres
- запускается при старте сервиса
- порождает все остальные процессы
- создает (выделяет) shared memory
- слушает TCP и/или unix socket

# backend processes

- создаются postmaster’ом
- обслуживают сессию
- время жизни = время сесии
- максимальное количество определяется параметром **max_connections** по умолчанию 100

# background procrsses

- запускаются postmaster’ом при старте сервиса
- у каждого – своя роль

| | |
|---|---|
| archiver | архивация wal |
| autovacuum launcher | запуск процессов очистки - autovacuum workers |
| bgwriter | превентивная запись грязных буферов |
| checkpointer | выполнение контрольных точек (принудительный сброс грязных буферов на диск) |
| logger | запись сообщений в журнал сервера |
| wal writer | запись wal на диск |

# Memory Architecture

## Локальная память

### 1. Память для внутренних операций (work_mem)

принадлежит backend процессу
размер определяется параметром work_mem (4 MB)
используется при выполнении запросов для создания хэш-таблиц, сортировки и т.д.

### 2. Память для операций обслуживания (maintenance_work_mem)

принадлежит backend процессу
используется служебными операциями VACUUM, REINDEX и т.д.
размер определяется параметром maintenance_work_mem (64 MB)
выделяется только при необходимости (при использовании соответствующих команд

### 3. Память для временных таблиц (temp_buffers)

размер определяется параметром temp_buffers (8 MB)
используется для хранения временных таблиц

# Memory Architecture

## Что не так с настройками?

**Памяти у инстанса 4 Gb (периодически приходил OOM killer)**

```
max_connections = 1000          # (change requires restart)
shared_buffers = 6GB            # min 128kB
work_mem = 16MB                 # min 64kB
maintenance_work_mem = 256MB    # min 1MB
```

# Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

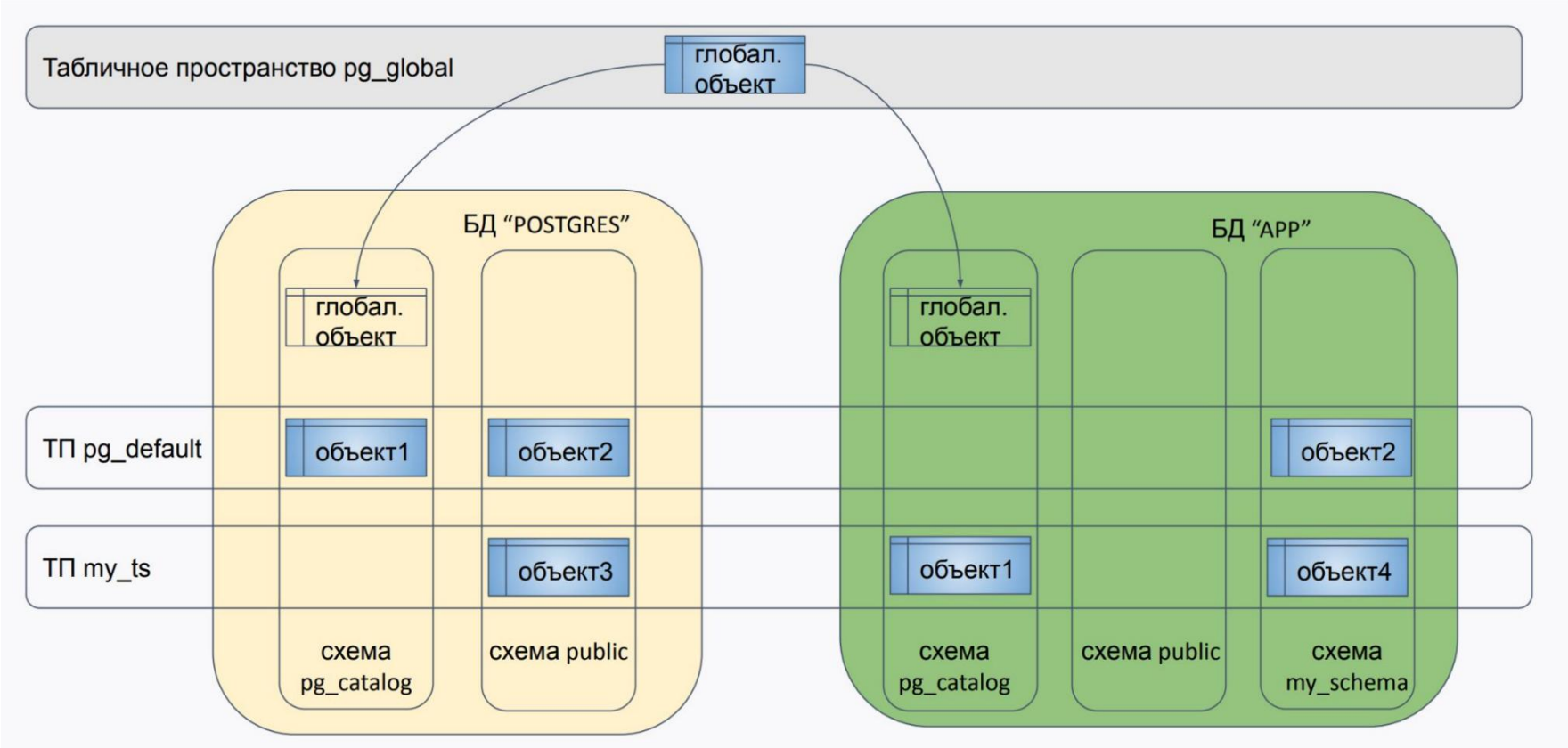
# Физическая структура данных

# Табличные пространства

- определяют место в ФС сервера (каталог) для объектов БД
- одно табличное пространство может содержать объекты различных БД
- по умолчанию создаются 2 табличных пространства pg_default и pg_global

**Общие рекомендации:**

- Часто используемые данные – на быстрые носители, архив – на медленные
- Разнести индексы и таблицы по разным носителям
- Выделить SSD для материальных представлений


Табличное пространство pg_global
глобал. объект
БД “POSTGRES”
БД “APP”
глобал. объект
глобал. объект
ТП pg_default
объект1
объект2
объект2
ТП my_ts
объект3
объект1
объект4
схема pg_catalog
схема public
схема pg_catalog
схема public
схема my_schema

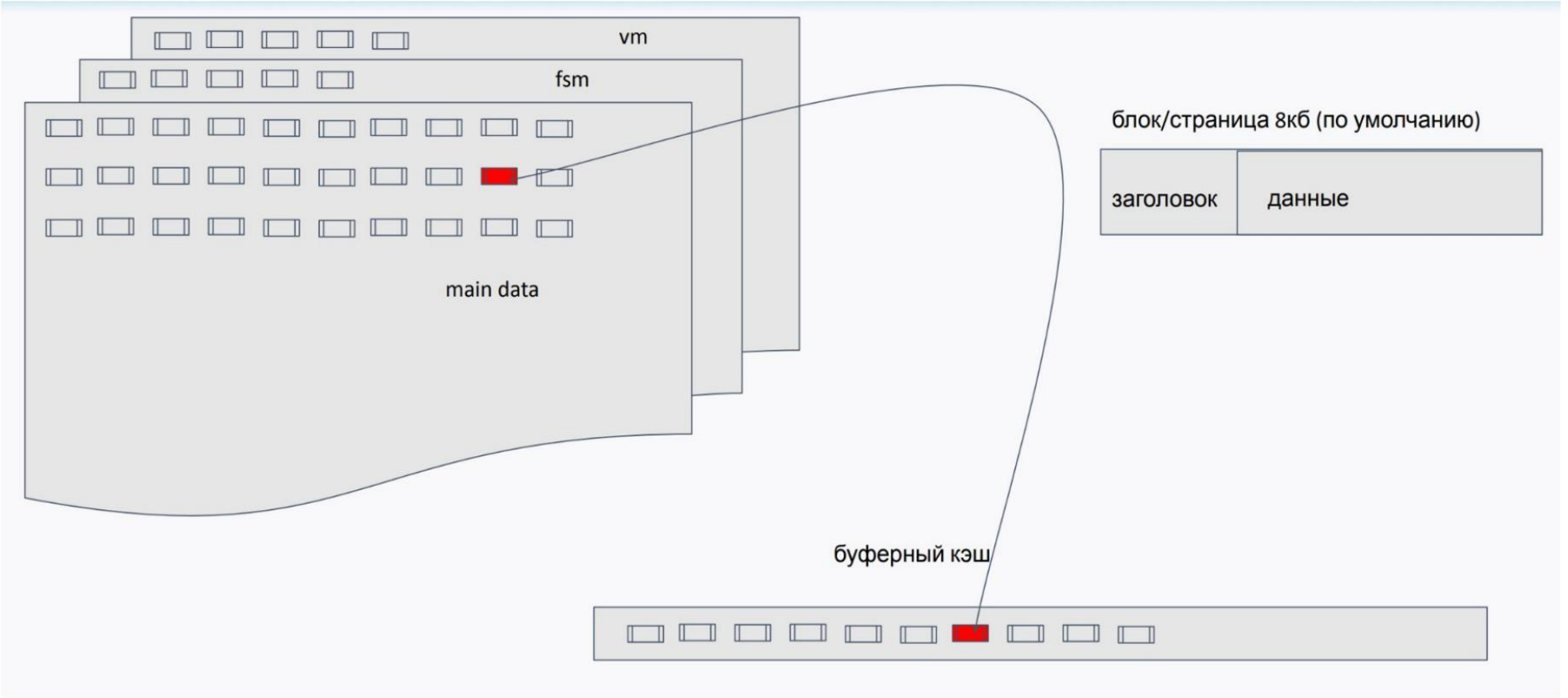
# Таблицы

Таблицы хранятся в файлах, размер каждого файла (сегмента) <= 1G.

Для каждой таблицы может существовать несколько «слоёв»

- Слой данных – содержит данные таблицы
- Карта свободного пространства (free space map, fsm) – файл(ы) с отметками свободных областей, появляющихся после очистки
- Карта видимости (visibility map, vm) – файл(ы) с отметкой страниц, на которых все версии строк видны во всех снимках

# Слои табличных файлов



# TOAST – The Oversize Attributes Storage Technique

**Размер табличной страницы – 8K, ограничение на размер столбца – 1G ???**

**Стратегии хранения:**

- **EXTENDED**. Допускается сжатие и перенос в таблицу TOAST.
  Для типов, поддерживающих перенос в TOAST, обычно используется
  эта стратегия.
- **EXTERNAL**. Допускается только перенос в таблицу TOAST, но не сжатие. Если загружаемые данные уже сжаты, имеет смысл выбрать стратегию **EXTERNAL** для экономии затрат на малоэффективное повторное сжатие.
- **MAIN**. Поля с этой стратегией сжимаются только в том случае, если обработки полей EXTENDED и EXTERNAL оказалось недостаточно, чтобы уместить версию строки на странице. Перенос в таблицу TOAST тоже возможен, но только как крайняя мера.
- **PLAIN**. Хранение только в основной таблице. Используется для типов, значения которых имеют небольшой размер.

Стратегию хранения и метод сжатия можно задать на уровне столбца таблицы

# TOAST – The Oversize Attributes Storage Technique

## TOAST – таблица

- размещается в схеме pg_toast;
- поддерживается собственным индексом;
- читается только при обращении к «длинному» атрибуту (не надо без повода писать SELECT * ...);
- доступ к значениям TOAST всегда осуществляется по индексу - самый быстрый способ получить все фрагменты одного значения, но он заведомо неэффективен для доступу ко всем значениям;
- имеет собственную версионность (если при UPDATE «длинный» атрибут не затрагивается, новая версия строки toast таблицы не создается);
- работает прозрачно для приложения;
- «порог срабатывания» определяется параметром TOAST_TUPLE_THRESHOLD (2 KB, не изменяется)

# Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

# Рефлексия