OTUS

Онлайн образование

otus.ru

Проверить, идет ли запись

Меня хорошо видно && слышно?

OTUS

Оптимизация PostgreSQL

Аристов Евгений

https://aristov.tech

Правила вебинара

Активно участвуем

Задаем вопрос в чат

Вопросы вижу в чате, могу ответить не сразу

Если остались вопросы/офтопик в канал Slack

Маршрут


1. Проблемы
2. Архитектурные пути решения
3. Мониторинг, профилирование
4. Варианты тюнинга
5. Стенд

Цель занятия

- варианты поиска «почему оно тупит»
- как «творчески» подходить к вопросу производительности
- как вносить точечные изменения в структуру БД или настроек СУБД для улучшения производительности
- как понять, когда это вопрос DBA, а когда разработчиков
- как учитывать все предыдущие вопросы при проектировании изначально

Проблемы

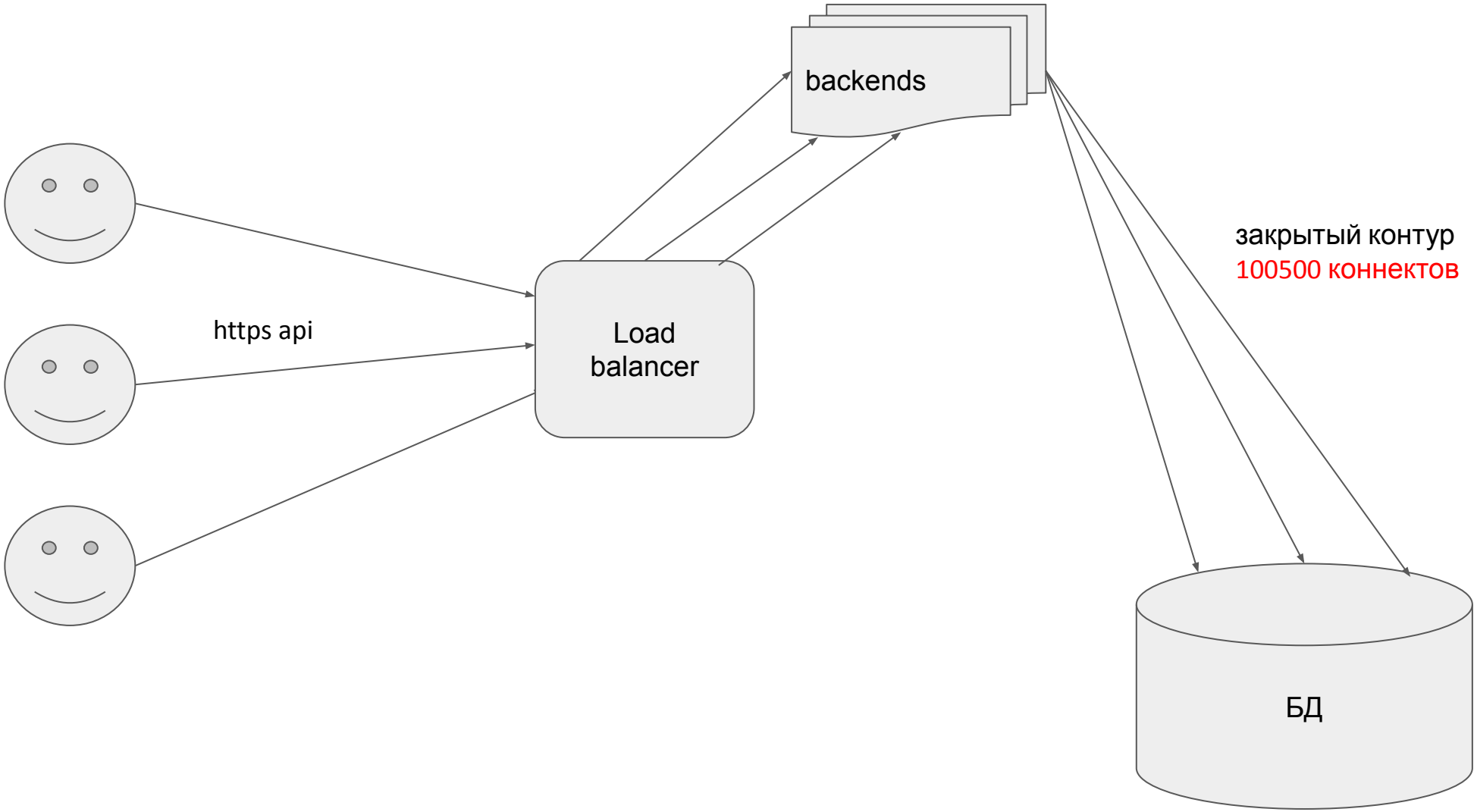
Проблемы

Что у нас есть на данный момент в большинстве проектов.
90% что был такой сценарий:

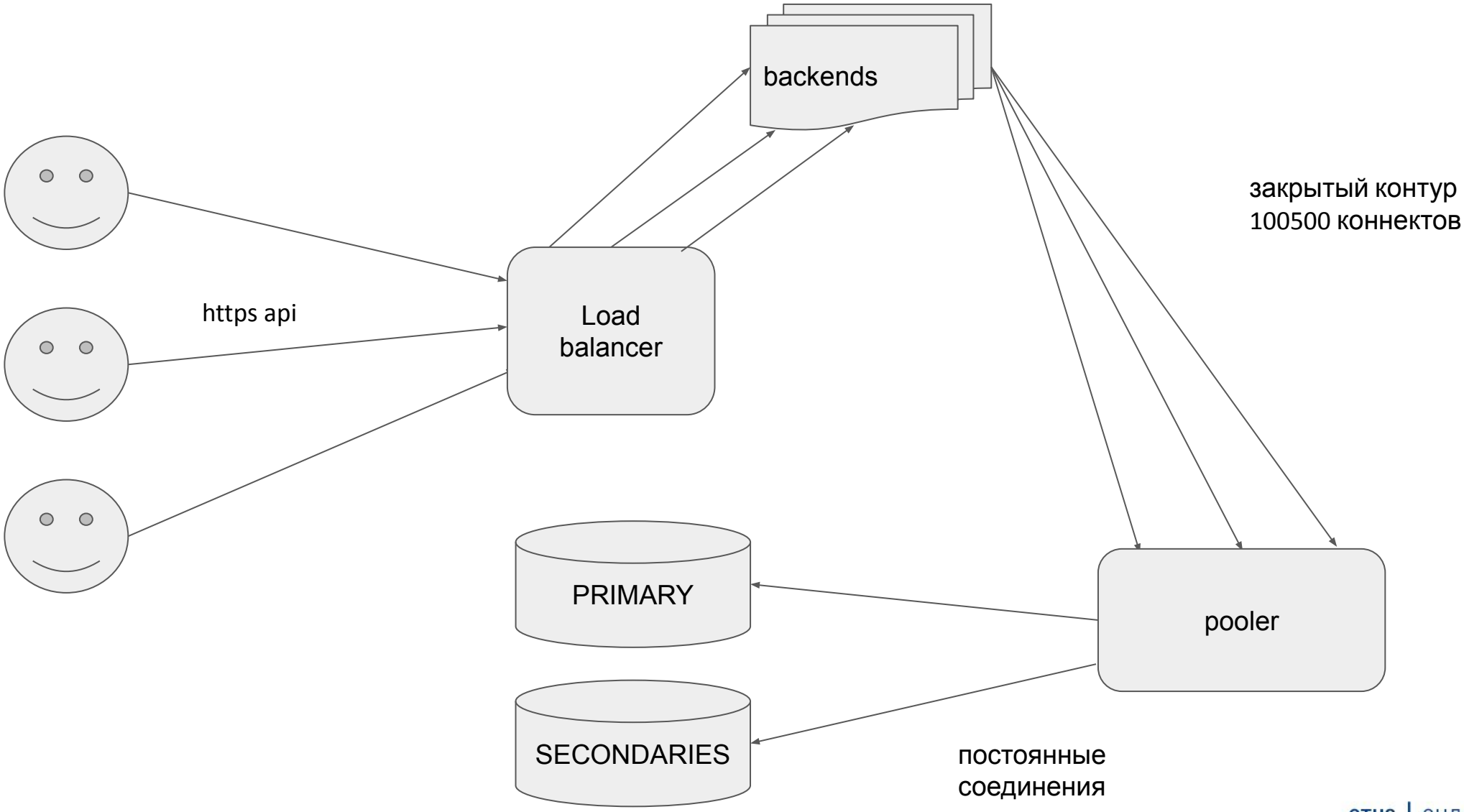
- У нас есть специалисты по Java/Ruby/etc
- ORM работает из коробки
- DBA нет или нет на это времени
- Постгрес выбрали, так как самый популярный, ну и не Оракл же
- Пока объемы были небольшие, все работало как часы
- Когда проект стал высоконагруженным начались проблемы:
  - коннектов не хватает
  - памяти не хватает
  - tps уперся в потолок в пик нагрузок

Текущая конфигурация

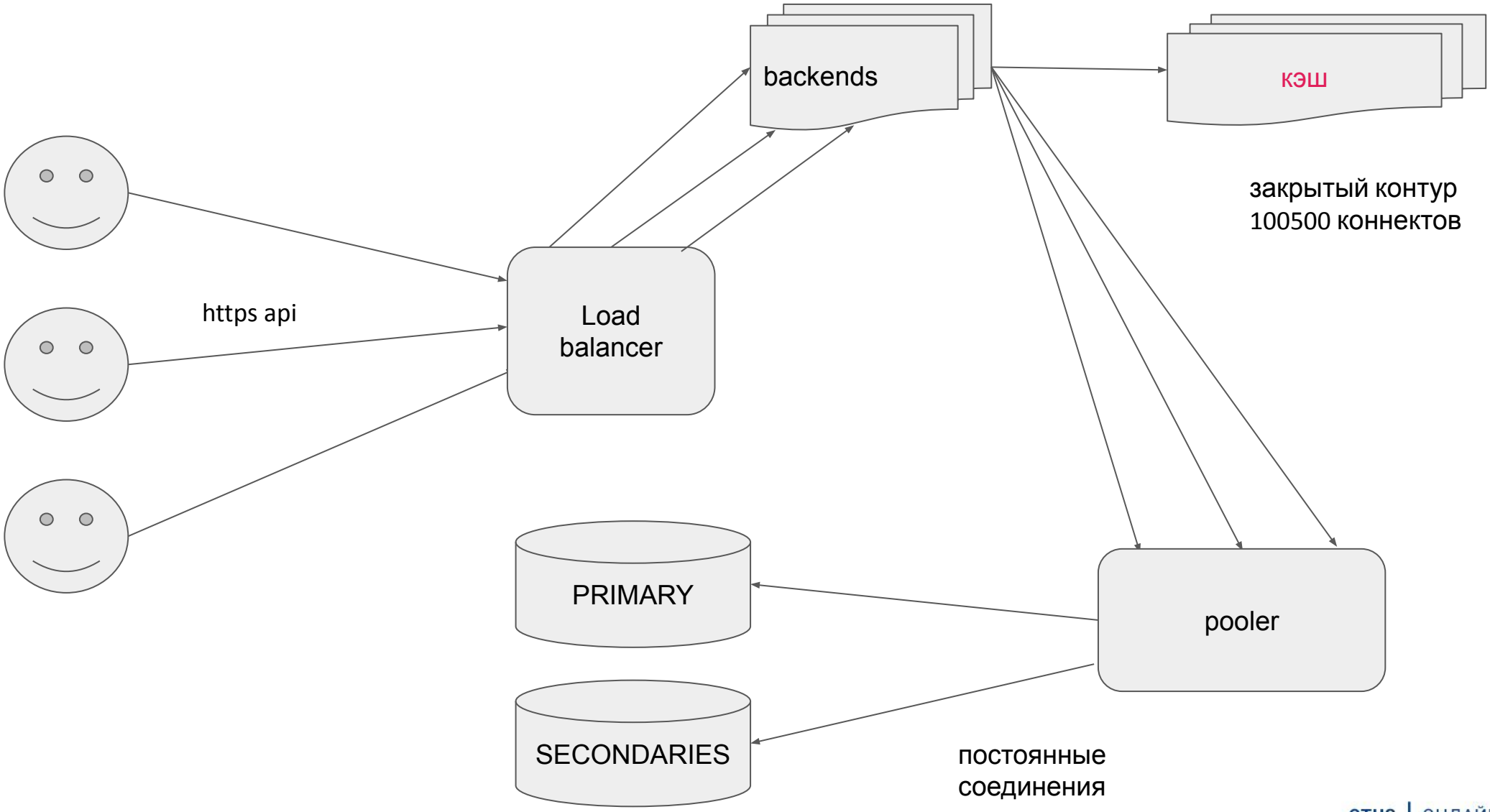
Чем плохо много коннектов?



Ну теперь то все хорошо?



Добавим кэш. Какие проблемы остались?



Какие проблемы при этом есть?

- **Если делать локальный кэш, то лоад балансер должен запросы с 1 фронта всегда отправлять на нужный бэкенд**
- **Если делать общий tarantool/redis (не совсем кэш), то сетевые задержки**

И общие проблемы:

- **TTL**
- **инвалидация кэша**

Кто же будет переключать secondary -> primary?

Предпочтительно использовать кластера

HA:

- Patroni
- Stolon
- Slony
- ClusterControl
- Kubegres

Параллельные кластера:

- Postgres-BDR
- CitusData
- Bucardo
- CockroachDB
- Yogabyte
- Greenplum

Опять же как развернуть.. on-premise, docker, k8sю…

А уж сколько облачных решений...

Общие проблемы

- **бэкап снимаем с secondary**
- **возможно есть смысл в каскадной репликации при проблемах с производительностью сети**
- **геораспределение нагрузки**
- **мониторим пустое место !!!**
- **строим планы развития системы (краткосрочный, долгосрочный) - прогноз обязателен!!!**

Как думаете, на это все?)

Варианты тюнинга

Мониторинг

Мониторинг

Срочно что-то посмотреть:

- htop
- atop
- iotop
- iostat
- **pgtop**

Мониторинг

Стандартные пути решения:

- прометей экспортер + графана
- https://okmeter.io/
- https://www.zabbix.com/ru - железо, хотя некоторые умудряются и Постгрес
- **Percona monitoring and management (PMM)**
  https://www.percona.com/doc/percona-monitoring-and-management/2.x/index.html

Мониторинг

Посмотрим внутрь БД через запросы на практике

Мониторинг

Что еще:

- pgHero
  https://habr.com/ru/company/domclick/blog/546910/
- pgWatch2
  https://github.com/cybertec-postgresql/pgwatch2

Анализ исторической нагрузки

- pgBadger https://github.com/darold/pgbadger
- pg_profile https://github.com/zubkov-andrei/pg_profile
- PoWA https://powa.readthedocs.io/en/latest/
- PoWA like https://habr.com/ru/post/345370/
- pgFouine https://highload.today/profilirovanie-v-postgresql/

Профилирование запросов

Профилирование запросов

- Для pl/pgsql есть расширение https://github.com/bigsql/plprofiler
- для остальных запросов используется журнал сообщений сервера
    - **log_min_duration_statements = 0** — время и текст всех запросов, при увеличении порога срабатывания - только “толстые” запросы
    - **log_line_prefix** — идентифицирующая информация
- включается для всего сервера
- большой объем
- анализ внешними средствами, например **pgBadger** **https://github.com/darold/pgbadger**
- для отслеживаются вложенных запросов можно использовать расширение **auto_explain** **https://www.postgresql.org/docs/14/auto-explain.html**

Практика

Профилирование запросов

Что еще:

- pg_profile https://github.com/zubkov-andrei/pg_profile
  Анализ исторической нагрузки PostgreSQL
- PoWA (PostgreSQL Workload Analyzer)
  https://powa.readthedocs.io/en/latest/
- PoWA like
  https://habr.com/ru/post/345370/
- pgFouine
  https://highload.today/profilirovanie-v-postgresql/

Анализ структуры запросов

Запрос ORM №1


SELECT DISTINCT f1.`name`,
f1.`value`
FROM (
SELECT f.name,
f.`value`,
c1.name AS cname
FROM (
SELECT DISTINCT fltr[0] AS name,
fltr[1] AS `value`
FROM `mes-orders` AS o
UNNEST o.items AS item

Запрос ORM №2

```
SELECT RAW `Extent1` FROM [redacted] as `Extent1` WHERE ((TRUE AND (`Extent1`.`documentType` =
[redacted] AND (TRUE AND (`Extent1`.`station` IN (557, 81, 82, 553, 75, 76,
70)) AND ((`Extent1`.`orderNumber` IS NULL) OR TRUE))))
```

Рекомендации по оптимизации запросов

- делать меньше запросов
- читать меньше данных
- обновлять меньше данных (несколько update insert в одну транзакцию)
- уменьшать обработку на лету
- проанализировать передачу данных
  - сколько данных было просканировано - сколько отослано
  - сколько было отослано - сколько использовано приложением

Рекомендации по оптимизации запросов

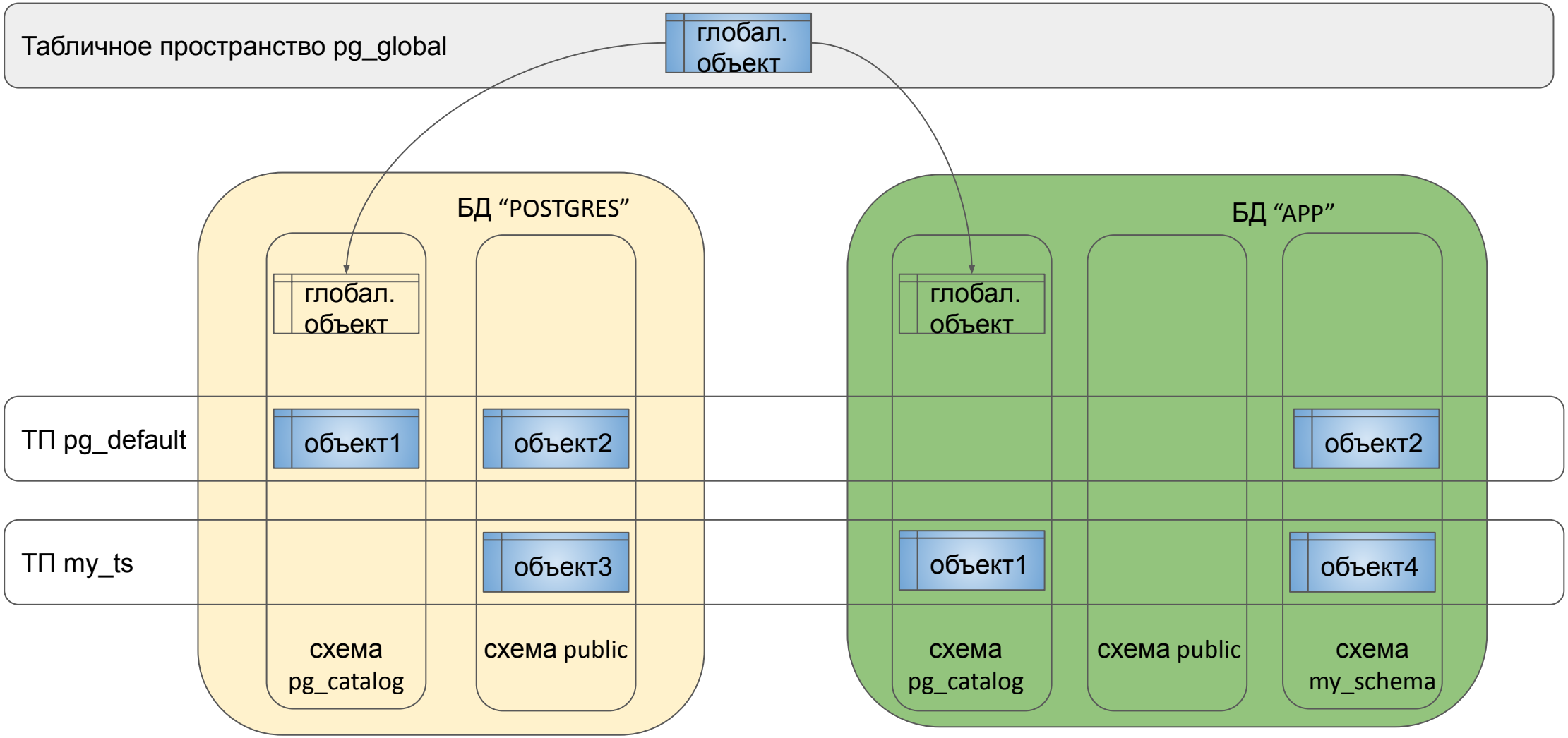
- Не использовать UNION, когда можно UNION ALL
- Использовать в выборке только нужные столбцы select *
- Избегать distinct
- При использовании сложных индексов учитывать наличие в запросе первого столбца индекса
- Избегать декартова произведения в джойнах
- При использовании нескольких таблиц давать имена столбцам с алиасами таблиц
- Используйте читабельный синтаксис SQL
- Используйте понятные названия и camelCase - при этом mysql автоматом в нижний регистр переводит
- Не используем пробелы в именах полей
- Не используем русский язык в именах полей
- **Пишите комментарии**

Рекомендации по оптимизации запросов

- Используйте EXISTS и IN там, где возможно
- Для небольших таблиц полное чтение FULL SCAN оптимальнее
- Индекс на маленькую таблицу не делаем, она скорее всего и так будет в 1 странице памяти
- Использование функций над индексными столбцами отключает использование индекса

Табличное пространство

Табличное пространство (ТП)



Tablespaces

Варианты оптимизации производительности:

- самые часто используемые данные на самые быстрые носители
- редко используемые архивы на медленные
- распараллелить нагрузку по разным рейд массивам
- локальный ссд без рейда для материальных представлений
- ну и самый быстрый вариант - **монтируем память в дисковую подсистему** и подключаем самые нагруженные данные (например stats_temp_directory - постоянно происходит доступ и при потере этого каталога Постгрес максимум будет строить неоптимальные планы, но работать продолжит)
- Естественно для чувствительных данных должна быть какая то асинхронная таблица на диске для постоянного хранения

TOAST

Версия строки должна помещаться на одну страницу 8кб

- можно сжать часть атрибутов,
- или часть вынести в отдельную TOAST-таблицу,
- или сжать и вынести одновременно

TOAST-таблица

- поддержана собственным индексом
- читается только при обращении к «длинному» атрибуту
- собственная версионность (если при обновлении toast часть не меняется, то и не будет создана новая версия toast части)
- работает прозрачно для приложения
- *стоит задуматься, когда пишем select **

https://habr.com/ru/company/oleg-bunin/blog/597187/

минитест

https://forms.gle/YNTiRR13Jf7tr3r5A

2-3 минуты

Тюнинг параметров

Параметры

По умолчанию Постгрес настроен отвратительно!

Всего 350+

Основные:

- shared_buffers
- max_connections
- effective_cache_size
- work_mem
- maintenance_work_mem
- wal_buffers
- max_wal_size
- checkpoint_timeout
- max_parallel_workers

описание параметров

Параметры

Варианты тюнинга:

http://pgconfigurator.cybertec.at/ - продвинутый конфигуратор от Cybertec.

https://pgtune.leopard.in.ua/ - онлайн-версия классического конфигуратора pgtune.

Вроде все? или нет?

Параметры

У кого на проекте разрабы несколько раз в секунду или при каждом запросе делают:

SELECT 1 ???

Еще варианты

Отключаем автовакуум.. или нет?

Позволяет убрать нагрузку с дисков, но есть несколько моментов, которые в долгосрочной перспективе выйдут боком и сделают только хуже.

- Самый простой и очевидный (для DBA) момент – это то, что статистика планировщика перестает собираться. Потому что автовакуум не только чистит таблицы, а еще и собирает статистику о распределении данных. И эта статистика используется планировщиком для того, чтобы строить оптимальные планы для запросов.
- Как только мы выключаем вакуум, таблицы и индексы перестают чиститься. И они начинают распухать. В них появляются мусорные строки.
- Следствием этого является то, что область **shared buffers**, где размещаются все оперативные данные для работы базы данных (это таблицы, индексы), начинается использоваться неэффективно. В ней находятся те самые мусорные строки. И чтобы запросу прочитать какие-то данные, постгресу нужно загружать страницу с мусорными данными и помещать ее в shared buffers.

Все это очень плохо с точки зрения общей производительности. Производительность запросов снижается. И в долгосрочной перспективе отключение вакуума – это гарантированно плохие спецэффекты в БД.

https://habr.com/ru/post/501516/

https://github.com/lesovsky/ConferenceStuff/tree/master/2016.highload

На что еще обратить внимание

- Добавление/изменение/удаление большого объема пишем пакетно!!
- WAL - тюнить. Например синхронная/асинхронная запись на диск - разница от 2 до 30 раз!!! Может есть смысл писать пакетами, например по 5 изменений или **выделить для этого отдельный диск**
- Индексы!!! **Удалить ненужные, добавить нужные.** Индексы далеко не бесплатны на вставку, обновление или удаление. Составные, функциональные! REINDEX CONCURRENTLY - не забываем анализировать и перестраивать
- Джойны - ORM неважно это делает
- Структура таблиц - аналогично
- Анализ/профилирование медленных/частых запросов/Статистика
- Секционирование
- Полнотекстовый поиск
- Тюнинг линукс (huge pages, swappiness, etc..)
- Бэкапы со слейва!
- Не забываем включать pg_rewind !!!

Практика

Тюнинг Автовакуума

https://aws.amazon.com/ru/blogs/database/a-case-study-of-tuning-autovacuum-in-amazon-rds-for-postgresql/

Общая рекомендация - тюним максимально агрессивно

Использование доп.объектов

промежуточные агрегаты, избыточность

материализованные представления

... может быть использование Хранимых Процедур?

... message broker для накопления данных и их ночная обработка?

... может быть OLAP (Pentaho)?

... может есть смысл fdw? cstore? pgmemcache? timescaledb?

Как мы понимаем, серебряной пули, к сожалению, не существует(

Что еще

CTE - обобщенные табличные выражения

CTE

CTE

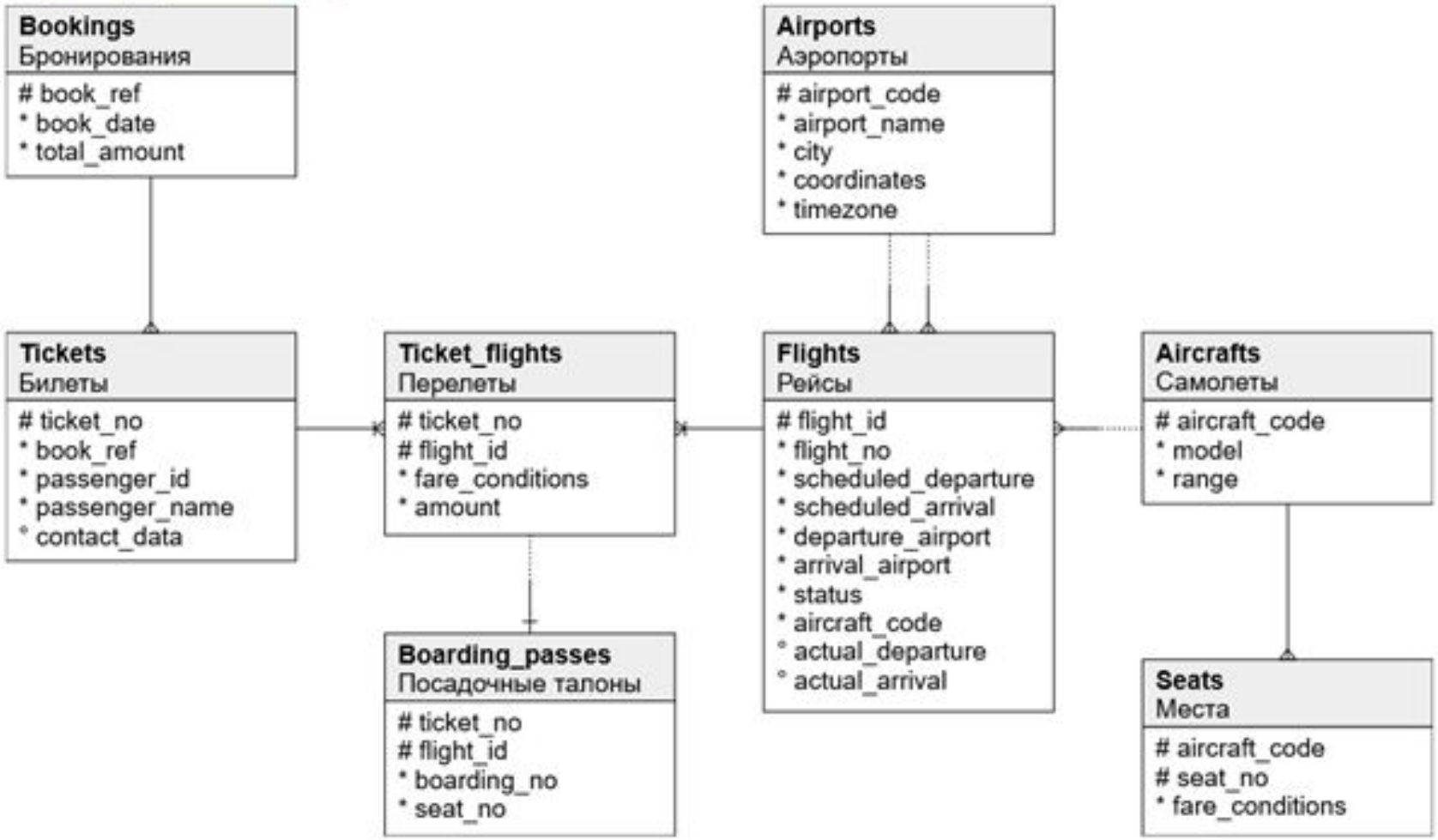
Используем небольшую БД Bookings на 2 млн записей

https://postgrespro.ru/docs/postgrespro/14/demodb-bookings

Используется для облегчения чтения и ускорения работы запросов

CTE

Рисунок К.2. Диаграмма схемы Bookings



CTE

Используем небольшую БД Bookings на 2 млн записей

https://postgrespro.ru/docs/postgrespro/14/demodb-bookings

Наша задача: написать запрос с выборкой всех рейсов и посчитаем количество свободных мест на этих рейсах.

Данный запрос будет соединять несколько таблиц и включать множество условий.

В такой ситуации я рекомендую идти поступательно от простого к сложному и разбить задачу на подзадачи.

Переходим к практике

CTE

Общие рекомендации:

- CTE
- EXPLAIN
- сначала посмотреть, что с 1 джойном и потихоньку усложнять -> мат представление (возможно промежуточное)
- еще варианты: вставлять также уже в общую таблицу
- триггер на обновление данных и формирование промежуточной таблицы
- work_mem на конкретную сессию

EXPLAIN

Визуализация планов:
https://explain.tensor.ru/
https://gitlab.com/postgres-ai/joe
https://tatiyants.com/pev (VPN)
https://explain.depesz.com/ (VPN)

Обратите внимание на Nested Loops & Seq Scan.
join_collapse_limit – сколько перестановок имеет смысл делать для поиска оптимального плана запроса

default_statistics_target - число записей просматриваемых при сборе статистики по таблицам. Чем больше тем тяжелее собрать статистику. Статистика нужна, к примеру для определения «плотности» данных.

Explain. Caveats

Время выполнения, измеренное с помощью EXPLAIN ANALYZE , может отличаться от нормального выполнения одного и того же запроса двумя важными способами:

- поскольку никакие выходные строки не доставляются клиенту, затраты на передачу по сети и затраты на преобразование ввода-вывода не включаются
- накладные расходы на измерения, добавленные EXPLAIN ANALYZE

Explain. Caveats

Также результаты EXPLAIN не следует экстраполировать на ситуации, сильно отличающиеся от той, которую вы фактически тестируете; например, нельзя предполагать, что результаты для таблицы размером с 1000 строк применимы к большим таблицам.

Смета расходов планировщика не является линейной, поэтому он может выбрать другой план для большего или меньшего набора данных. Ярким примером является то, что для таблицы, занимающей только одну страницу на диске, вы почти всегда получаете план последовательного сканирования независимо от того, доступны ли индексы или нет. Планировщик понимает, что для обработки таблицы в любом случае потребуется чтение одной страницы с диска, поэтому нет смысла расходовать дополнительные чтения страницы для просмотра индекса

Explain. Caveats

Бывают случаи, когда фактические и расчетные значения не совпадают, но на самом деле все в порядке. Один из таких случаев происходит, когда выполнение узла плана прекращается из-за LIMIT или аналогичного эффекта. В реальности узел с Limit перестает запрашивать строки после того,как их стало требуемое количество и время выполнения будет меньше, чем можно предположить по расчетной стоимости.

Explain. Типы запросов и соединений

Объясняя необъяснимое. Часть 3 / Хабр
https://habr.com/ru/post/560834/
https://habr.com/ru/company/postgrespro/blog/696680/
https://habr.com/ru/company/postgrespro/blog/412605/
https://habr.com/ru/company/postgrespro/blog/683020/

https://postgrespro.ru/docs/postgresql/15/using-explain

Варианты повлиять на планировщик:
https://postgrespro.ru/docs/postgresql/15/runtime-config-query#RUNTIME-CONFIG-QUERY-ENABLE

Explain. Хинты планировщику

https://github.com/ossc-db/pg_hint_plan

Еще архитектурные решения

Что в итоге

- на живую не меняем БД!!! скрипты миграции
- обязательны тестовые стенды !!!
- insert -> copy, потом создавать индексы и поменять местами таблицы
- update -> мат представление содержит данные из основной таблицы-> delete & copy -> REFRESH CONCURRENTLY
- однозначно тяжелый одинаковый запрос каждый раз считать не надо -> MAT VIEW
- как добавить колонку с дефолтным значением, когда сотни миллионов строк:
    - добавляем колонку без дефолтного значение
    - батчами по 10-100 тысяч заполняем, не забываем про вакуум
    - потом добавляем дефолтное значение
    - в постгресе не нужно)
    - аккуратно в случае timestamp

насчет удаления 100 миллионов также - сначала думаем)

Что думают об этом в интернете

https://qastack.ru/programming/621884/database-development-mistakes-made-by-application-developers

Практика

ДЗ

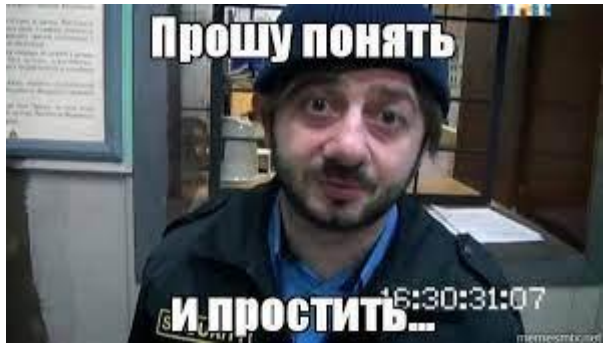
Изменение расписания

К сожалению форсмажорчик(

Бэкапы пройдут 16 в 12:00

Патрони1 18 в 20:00

Патрони2 21 в 20:00



Домашнее задание

Развернуть Постгрес на ВМ

Протестировать pg_bench

Выставить оптимальные настройки

Проверить насколько выросла производительность

Настроить кластер на оптимальную производительность не обращая внимания на стабильность БД

ДЗ сдаем в виде миниотчета в markdown и гите

Заполните, пожалуйста,
опрос о занятии по ссылке в чате
https://otus.ru/polls/72695/

Спасибо за внимание!
Приходите на следующие вебинары

Аристов Евгений