404
GROUP

PostgreSQL — технологическая платформа сервиса Movebo.ru

Денис Милованов
movebo.ru

О сервисе

Сервис занимается продвижением сайтов в поисковых системах за счет улучшения поведенческих факторов.

- Десятки тысяч внешних исполнителей — “серферов”.
- Сотни тысяч сессий в сутки.
- Полтора десятка разных ограничений и условий по таргетингу.

Почему PostgreSQL?

- Отличная и честная функциональность (ACID и т.д.)

Чего нет в MySQL, но что крайне необходимо:

- транзакционный DDL, DEFERRABLE ключи,
  частичные индексы, строгая проверка типов.

В дополнение:

- отличная консоль psql :)

// Мы работаем с 3 базами на отдельных машинах
+ hot standby главной базы.

“Mission-critical” компоненты

Подготовка заданий — формирование актуального кеша заданий, которые могут выдаваться прямо сейчас.

Выдача заданий — подбор доступных заданий по запросу в зависимости от параметров.

“Mission-critical” компоненты

Обслуживание выданных заданий — снятие блокировок на выдачу по прошествии времени (“expiration”).

Модерация выданных заданий — распределение заданий между пулом модераторов.

Все организовано и отдается приложению в виде API из хранимых процедур.

Почему хранимые процедуры?

1. Система грантов позволяет закрыть доступ к важным данным. *SECURITY DEFINER-процедуры.*
2. Скрытие логики от разработчиков, занимающихся фронтом.
3. REPLACE, он же MERGE, он же UPSERT.
4. Транзакции по умолчанию.
5. Удобно оптимизировать и тестировать.

mbus — очередь для PostgreSQL

1. mbus = post + consume с API на хранимых процедурах.

2. Соответственно, транзакции и правильная выдача грантов “из коробки”.

3. Отложенная доставка, идеальна для организации “expiration”-джобов (в RabbitMQ, например, с этим не просто).

Миграции

Миграций, в классическом понимании, нет.

1. Добавление колонки в большую таблицу, перестроение её составного индекса и подобные тяжелые вещи всегда представляют собой спец-операцию, которую требуется делать вручную (и не в одной транзакции).

Пример к п.1

Как мы добавляем колонку к большой таблице:

```
BEGIN;

-- быстрое добавление колонки NULL
ALTER TABLE huge_table ADD COLUMN new_field integer NULL;

-- ставим дефолт, который будет применяться для всех новых строк
ALTER TABLE huge_table ALTER COLUMN new_field SET DEFAULT 1;

COMMIT;

for i in `seq 0 10000`; do psql -h localhost -p 5432 -U user-c "UPDATE huge_table SET
new_field = 1 WHERE id IN (SELECT id FROM huge_table AS q WHERE id BETWEEN $i * 10000
and $i * 10000 + 10000);" some_db; done;

ALTER TABLE huge_table ALTER COLUMN new_field SET NOT NULL;
```

Миграции

2. Поддержание down-кода имеет сомнительную полезность.

3. И, да, как писать up-down миграции на хранимые процедуры?

PostgreSQLDeployerGUI

Итак, миграции не всегда то, что хочется, а для хранимок они просто бессмысленны.

Хранимые процедуры — это код. А код — это гит.

Давайте раскатываться из гита.

// Тем более, что ещё до всяких хранимых процедур мы хранили там описание таблиц :)

PostgreSQLDeployerGUI

Идея:

1. Храним в базе актуальное описание объектов схемы.
2. Показываем те объекты в гите, код которых отличается от сохраненного.
3. Делаем возможным раскатать выбранные изменения в одной транзакции: DDL транзакционен.
4. По факту раскатки актуализируем описание в базе.

Организация репозитория

```
database-name.git/schemas/
    /main — схема
        /seeds — сиды (константы, словари и т.п.)
        /types — типы
        /functions — хранимые процедуры
        /tables — таблицы + индексы + констрейнты
    /other_schema — другая схема
        ...
```

Пример

database-name.git/schemas/main/users.sql

```
CREATE TABLE main.users ( … );
GRANT SELECT ON main.users TO …;
```

1 коммит

```
ALTER TABLE main.users
     ADD COLUMN deleted_at timestamptz NULL;
```

2 коммит

```
CREATE INDEX CONCURRENTLY
     ON main.users USING btree(...);
```

3 коммит

Что такое “раскатать” изменения?

Изменились **сиды:**

удалить старые и накатить новые.

Все ссылки на них
DEFERRABLE.

main/seeds/languages.sql >

```
BEGIN;

DELETE FROM main.languages;

INSERT INTO main.languages
    SELECT  1, 'Русский', 'RU';

+INSERT INTO main.languages
+    SELECT  2, 'Английский', 'EN';

COMMIT;
```

Что такое “раскатать” изменения?

Изменилась **процедура**:

1. удалить процедуру с таким именем, если поменялась сигнатура,
2. выполнить запросы из файла (код + гранты).

`main/functions/get_id.sql >`

```
CREATE OR REPLACE FUNCTION main.get_id (
    s_dict_name varchar,
    s_index varchar
)
+RETURNS integer AS
-RETURNS bigint AS
$BODY$
DECLARE
    i_id integer;
BEGIN
    ...
```

Что такое “раскатать” изменения?

Изменился **тип данных**:

1. удалить зависимые хранимые процедуры,
2. удалить тип,
3. создать его и процедуры заново.

`main/types/t_user_info.sql >`

```
CREATE TYPE main.t_user_info AS (
    id bigint,
    uid varchar,
-   email varchar
+   email varchar,
+   phone varchar
);
```

Что такое “раскатать” изменения?

Изменилось **описание таблицы:**
проверить, можно ли просто выполнить diff как набор запросов?

```
+ALTER TABLE main.users
+   ADD COLUMN phone varchar(20) NULL;
```

Можно

```
-update_at timestamp NULL,
+updated_at timestamptz NOT NULL,
```

Нельзя

Но конкретно вот так никто не делает

Что такое “раскатать” изменения?

Если можно (большинство случаев), то предложить этот вариант в интерфейсе.

Если нельзя, то сказать об этом.

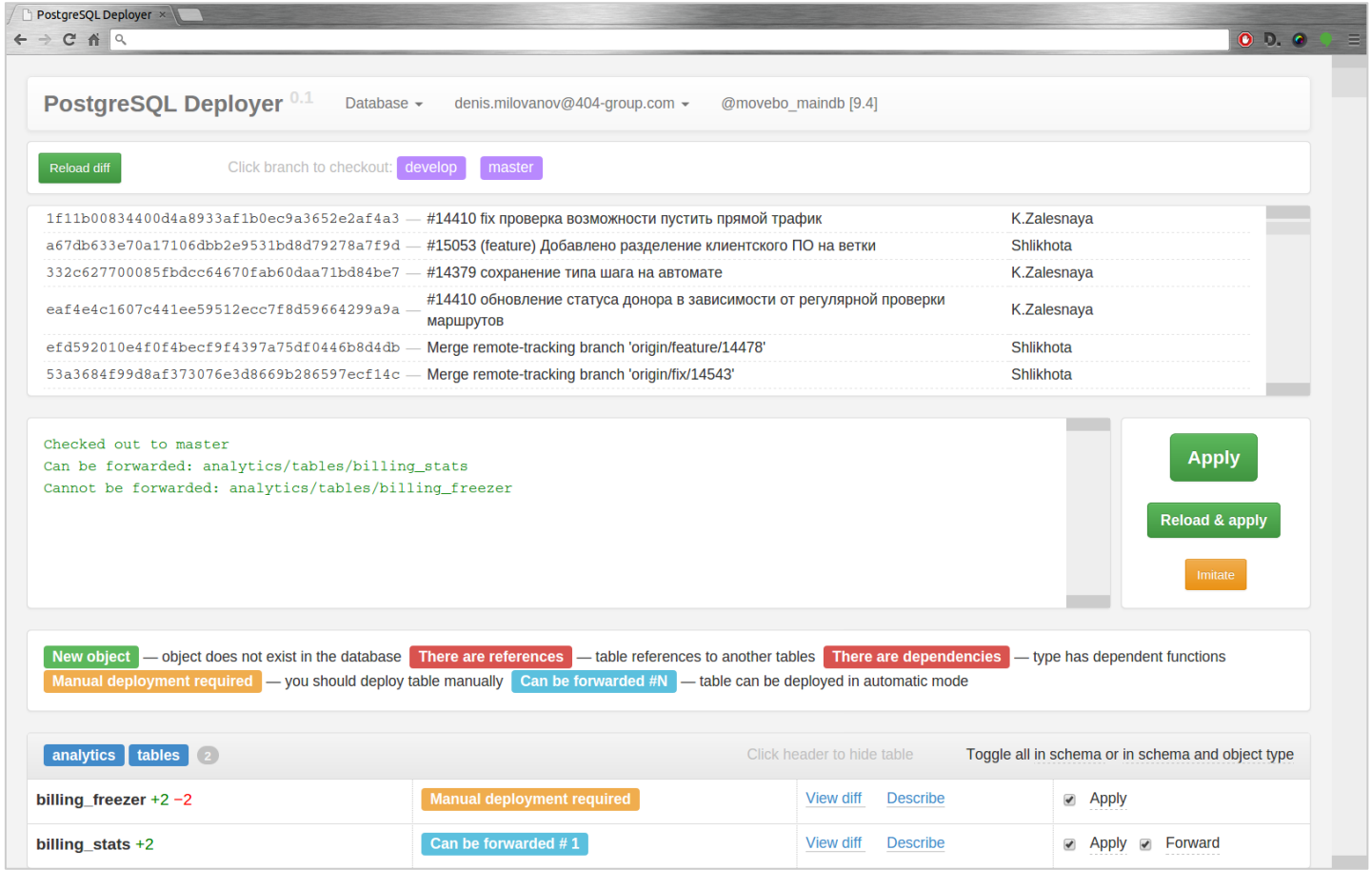
Когда нельзя? Например, при наличии циклических ссылок.

Решение о способе деплоя и собственно деплой — на совести ответственного сотрудника.

Как откатиться?

Откат таблиц — исключительно вручную.

Откат хранимых процедур — выбор стабильного коммита и применение кода из него.


PostgreSQL Deployer
PostgreSQL Deployer 0.1
Database
denis.milovanov@404-group.com
@movebo_maindb [9.4]
Reload diff
Click branch to checkout: develop master
1f11b00834400d4a8933af1b0ec9a3652e2af4a3 — #14410 fix проверка возможности пустить прямой трафик K.Zalesnaya
a67db633e70a17106dbb2e9531bd8d79278a7f9d — #15053 (feature) Добавлено разделение клиентского ПО на ветки Shlikhota
332c627700085fbdcc64670fab60daa71bd84be7 — #14379 сохранение типа шага на автомате K.Zalesnaya
eaf4e4c1607c441ee59512ecc7f8d59664299a9a — #14410 обновление статуса донора в зависимости от регулярной проверки маршрутов K.Zalesnaya
efd592010e4f0f4becf9f4397a75df0446b8d4db — Merge remote-tracking branch 'origin/feature/14478' Shlikhota
53a3684f99d8af373076e3d8669b286597ecf14c — Merge remote-tracking branch 'origin/fix/14543' Shlikhota
Checked out to master
Can be forwarded: analytics/tables/billing_stats
Cannot be forwarded: analytics/tables/billing_freezer
Apply
Reload & apply
Imitate
New object — object does not exist in the database
There are references — table references to another tables
There are dependencies — type has dependent functions
Manual deployment required — you should deploy table manually
Can be forwarded #N — table can be deployed in automatic mode
analytics tables 2
Click header to hide table
Toggle all in schema or in schema and object type
billing_freezer +2 −2
Manual deployment required
View diff
Describe
Apply
billing_stats +2
Can be forwarded # 1
View diff
Describe
Apply
Forward

Еще полезные возможности

1. Просмотр diff’ов.

2. Возможность показать то, что есть в базе, но нет в гите.

   Плюс можно получить описание через psql (\dt+, \sf), схему через pg_dump (--schema-only) или удалить.

Еще полезные возможности

3. Проверка на наличие логических ошибок в процедурах с помощью расширения plpgsql_check.
   https://github.com/okbob/plpgsql_check

4. “Reload and deploy” — перечитать диффы и раскатать все изменения сразу. Полезно для разработки.

https://github.com/denismilovanov/PostgresqlDeployerGUI

404
GROUP

Вопросы?

Денис Милованов

movebo.ru
denis.milovanov@404-group.com