



pgCodeKeeper

Инструмент для организации
современного процесса разработки БД

Сергей Мокеев
технический директор
Maxim technology

О нас



- Первые в России запустили цифровую платформу для связи пассажира и водителя такси
- Несколько миллионов заказов в сутки обрабатывает система
- 300+ IT-специалистов занимается разработкой, поддержкой и локализацией продуктов



- Основной заказчик
- В тройке лидеров среди сервисов заказа такси
- 20+ лет работы
- 120+ млн скачиваний приложения для заказа такси

О чем доклад

- Путь развития нашей системы
- Параллельный процесс развития разработки
- Инструмент pgCodeKeeper
- Тестирование функций БД
- CI/CD-процесс и автоматизация

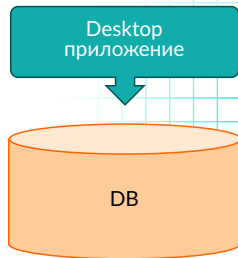


Мы не используем ORM

Развитие системы

С чего началось

- Бизнес-логика внутри БД
- **БД** как API
- Доступа у приложений и сервисов к таблицам нет
- Чтение данных через представления и функции
- Изменения данных через функции/хранимые процедуры
- Триггеры для бизнес-логики **не** используем
- Релизы приложения и БД **не** синхронизируем



Примеры изменений

- Учитываем обратную совместимость
- Значимое выполняем в 3 шага: создание новых функций, переключение на новые функции, удаление старых
- Выкатываем этапами: сначала изменение структуры, затем изменения функций/представлений
- Минимизируем изменения за раз

Развитие системы

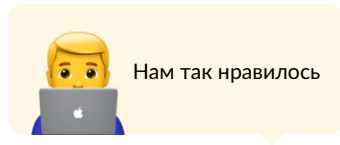
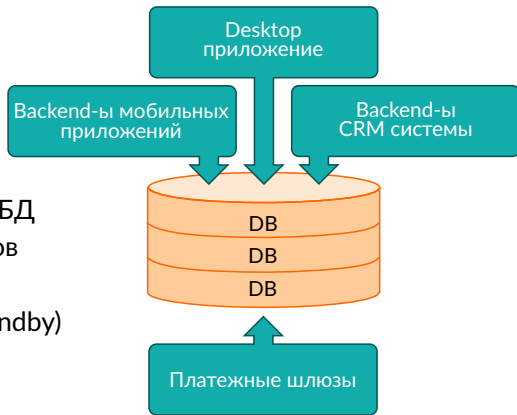
Как стало чуть позже

+ Плюсы

- Полный контроль над тем, что происходит в БД
- Широкие возможности оптимизации запросов
- Удобный деплой изменений
- Масштабирование средствами СУБД (hot-standby)

- Минусы

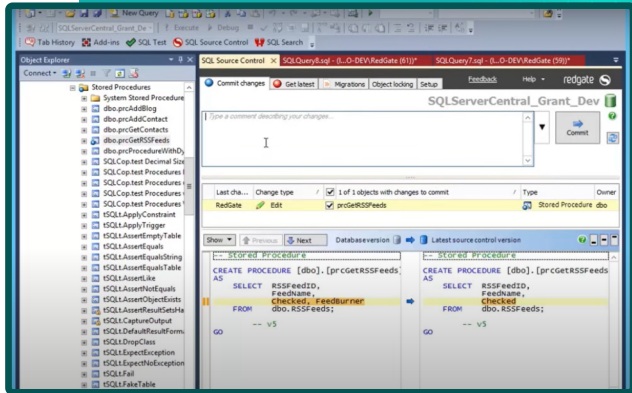
- БД может стать узким местом по производительности, сложно масштабировать
- «Размазывание» БЛ между БД и сервисами



Инструмент для MSSQL

Ключевые возможности

- Создание репозитория из БД
- Сравнение репозитория и БД и отображение различий
- Формирование скрипта миграции



Переезд на Postgres

- Перешли, чтобы не платить лицензии
- Нет аналогичного SC инструмента для работы с кодом
- Нет почти никакого инструментария, кроме pgAdmin3



Без аналога SC работа —
не работа(

pgCodeKeeper

- Создание проекта БД для хранения в системе контроля версий
- Сравнение БД и проекта с выводом расхождений
- Формирование скрипта миграции с учетом всех зависимостей
- Режим командной строки, есть safe-режим
- Работа с библиотеками
- Поддержка MSSQL



В OpenSource с 2017-го

Так выглядит pgCodeKeeper

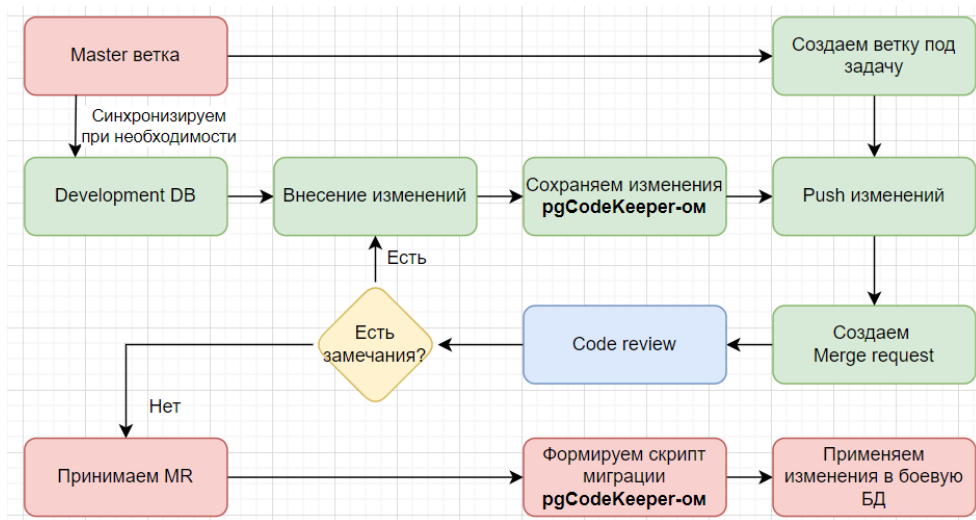
The screenshot displays the pgCodeKeeper application interface, which is used for managing PostgreSQL databases. The interface is divided into several panes:

- Project Explorer:** Shows the project structure, including the 'test project' and its sub-entities like 'pgCodeKeeper Project', 'Referenced Libraries', 'EXTENSION', 'SCHEMA', 'public', 'AGGREGATE', 'DOMAIN', 'FUNCTION', 'OPERATOR', 'TABLE', 'TYPE', 'VIEW', and 'public.sql'.
- test project - postgres_12:** A table listing the objects in the database. The table has columns for 'Object Type', 'Change Type for project', and 'Object Name'. The 'TABLE' object 't1' is selected.
- Progress / pg Dependencies:** A diagram showing the dependencies between database objects. It includes 'TBL t1', 'FUNC serv', and 'public'.
- Database / Project:** A SQL editor showing the SQL code for the selected object. The code is as follows:

```
1 CREATE TABLE public.t1 (  
2   c1 integer,  
3   c2 integer,  
4   c3 integer,  
5   c4 text  
6 );  
7  
8 CREATE INDEX t1_c1_idx ON public.t1 USING btree
```

The status bar at the bottom indicates the application is running on a 440M of 835M memory.

Flow разработки

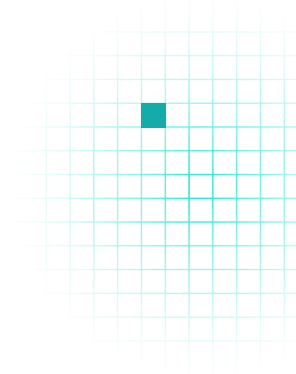


Тестирование

Для unit-тестов используем pgTap

Нужна БД с эталонными данными

Копии боевых БД, Dev БД и подобные **НЕ ПОДХОДЯТ**



Пример структуры репозитория

```
graph LR
    Root[ ] --- EXTENSION
    Root --- SCHEMA
    Root --- UPDATES
    Root --- dot_dependencies[".dependencies"]
    Root --- build_properties["build.properties"]
    Root --- build_xml["build.xml"]
    Root --- readme_md["readme.md"]
    Root --- tests
    Root --- city_tests["city"]
    Root --- setup
    Root --- data
    Root --- city_t_country_sql["city.t_country.sql"]

    SCHEMA --- city_schema["city"]
    city_schema --- FUNCTION
    FUNCTION --- get_country_sql["get_country.sql"]
    city_schema --- VIEW
    VIEW --- v_country_sql["v_country.sql"]
    city_schema --- TABLE
    TABLE --- t_country_sql["t_country.sql"]
    city_schema --- city_sql["city.sql"]

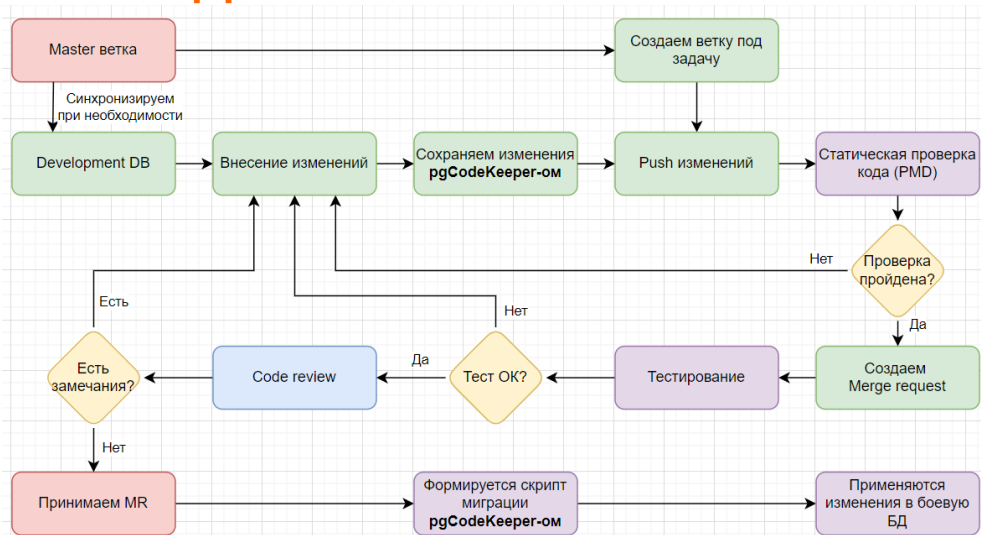
    UPDATES --- city_updates["city"]
    city_updates --- test_get_country_sql_updates["test_get_country.sql"]

    tests --- city_tests
    city_tests --- test_get_country_sql_tests["test_get_country.sql"]

    setup --- data
    data --- city_t_country_sql
```

- EXTENSION
 - postgis.sql
- SCHEMA
 - city
 - FUNCTION
 - get_country.sql
 - VIEW
 - v_country.sql
 - TABLE
 - t_country.sql
 - city.sql
- UPDATES
 - city
 - test_get_country.sql
- .dependencies
- build.properties
- build.xml
- readme.md
- tests
 - city
 - test_get_country.sql
- setup
 - data
 - city.t_country.sql

Итоговый pipeline



Создание экземпляра БД для тестов

1. Клонировем репозиторий БД, сливаем целевую ветку с мастером
2. Запускается docker-контейнер с предустановленным Postgres
3. Консольным pgCodeKeeper-ом создаем структуру БД
4. Накатываются на БД библиотеки для тестирования
5. Наливаем тестовые данные
6. Запускаем тесты



Раннер-утилита для запуска unit-тестов

- Умеет параллелить запуски тестов между несколькими БД
- Возвращает результаты запуска тестов в формате отчета junit
- Выполняет дополнительные проверки

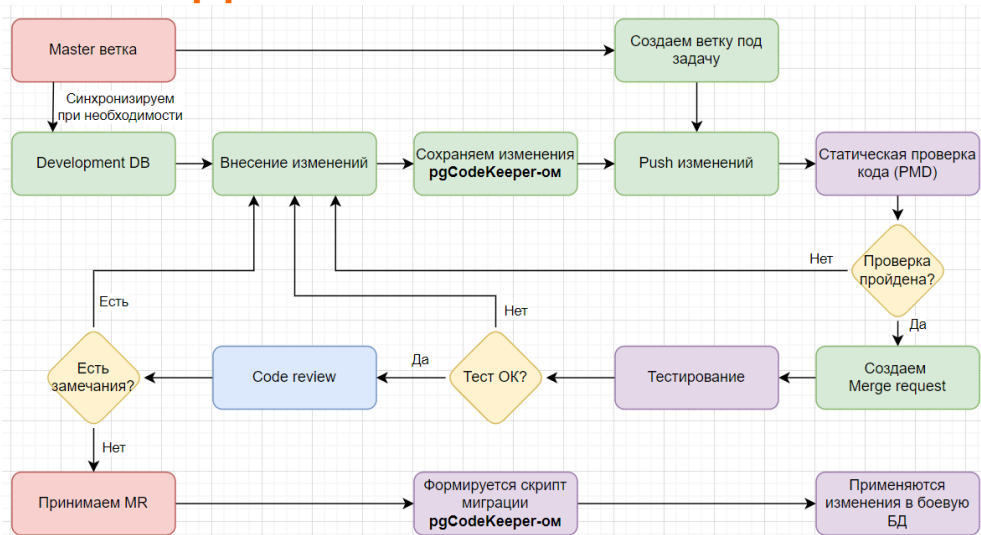
Функциональные тесты

Типовой алгоритм теста

- 1 Вызвали функцию в БД
- 2 Вызвали select-запрос для получения результата работы функции
- 3 Сравнили с ожидаемыми данными

Разработали специальный формат для описания функциональных тестов и утилиту для запуска.

Итоговый pipeline



Правила для PMD

- Ограничения
 - ExcessiveFunctionLengthRule — ограничение на размер функции
 - ExcessiveParameterListRule — ограничение на количество входных параметров
 - StdCyclomaticComplexityRule — ограничение на сложность и вложенность
 - NcssMethodCountRule - ограничение на количество выражений в функции
- Базовые правила
 - PublicInGrant — правило проверки на группу прав
 - CaseWithoutElse — правило блока case
 - CheckTempTable — правило создания таблицы
 - CyclomaticComplexity — проверка сложности кода
 - ExcessiveFunctionLengthRule — проверка сложности функции
 - ExcessiveParameterListRule — проверка на число входящих параметров
- Стилистические правила
 - CheckIndent — стилистическая проверка отступов
 - CheckSpaceOnMath — постановка пробелов
 - SpaceAfterIf — пробел после if
 - QuotesInTableColumn — кавычки в имени столбца

Метки на merge request в GitLab

Draft: D-1325 (4) удаление driver_registration_find_result

!31985 · created [REDACTED] codestyle-ok diff-ok transl-ok unit-test-ok

Draft: D-1381 Джоб violation_lifting_sanctions_job переписан на процедуру

!31960 · created [REDACTED] diff-ok transl-ok unit-test-error

Draft: Переключение вызовов на sync_data

!31958 · created [REDACTED] codestyle-ok diff-ok transl-ok unit-test-ok

Draft: tst 023

!31832 · created [REDACTED] diff-ok transl-ok unit-test-ok

Draft: Удалить временные джобы по удалению лишних данных

!31763 · created [REDACTED] diff-ok transl-ok unit-test-ok

Draft: D-1089 (2) set ф-я, тарифы смен водителя - поле расширить на подразделения

!31721 · created [REDACTED] codestyle-error diff-ok transl-ok unit-test-error

Draft: D-1089 (4) ф-и тарифы смен водителя - поле расширить на подразделения

!31720 · created [REDACTED] codestyle-error diff-ok transl-ok unit-test-error

Комментарии Jenkins в GitLab



Jenkins for Tests @jenk_maindb added **transl-ok** label 1 month ago



Jenkins for Tests @jenk_maindb added **diff-ok** label 1 month ago



Jenkins for Tests @jenk_maindb added **unit-test-error** label 1 month ago



Jenkins for Tests @jenk_maindb · 1 month ago

Maintainer



Test db-maindb-pipeline **unit ERROR** [Результат теста](#)



Jenkins for Tests @jenk_maindb added **codestyle-error** label 1 month ago



Jenkins for Tests @jenk_maindb · 1 month ago

Maintainer



Test db-maindb-pipeline **codestyle ERROR** [Результат теста](#)

Результаты тестов в Jenkins

Test Result

6 failures (+6)

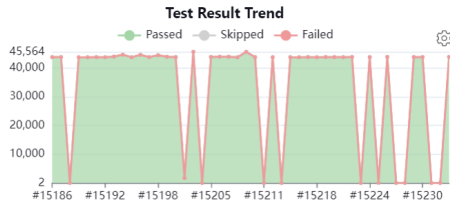
12,312 tests (+16)

Took 0 ms.

All Failed Tests

Test Name	Duration	Age
+ billing/test_account_debtor_cancel_check.test_account_debtor_cancel_check_3_active_ok [1]: Параметры вызова account_debtor_cancel	0 ms	1
+ billing/test_account_debtor_pay_job.Unexpected exception in tests.test_account_debtor_pay_job_0_ok()	0 ms	1
+ billing/test_account_debtor_pay_job.Unexpected exception in tests.test_account_debtor_pay_job_1_error()	0 ms	1
+ billing/test_account_debtor_pay_job.Unexpected exception in tests.test_account_debtor_pay_job_4()	0 ms	1
+ billing/test_account_debtor_pay_job.Unexpected exception in tests.test_account_debtor_pay_job_5_other_cluster()	0 ms	1
+ billing/test_account_debtor_pay_job.Unexpected exception in tests.test_account_debtor_pay_job_6_work_time()	0 ms	1

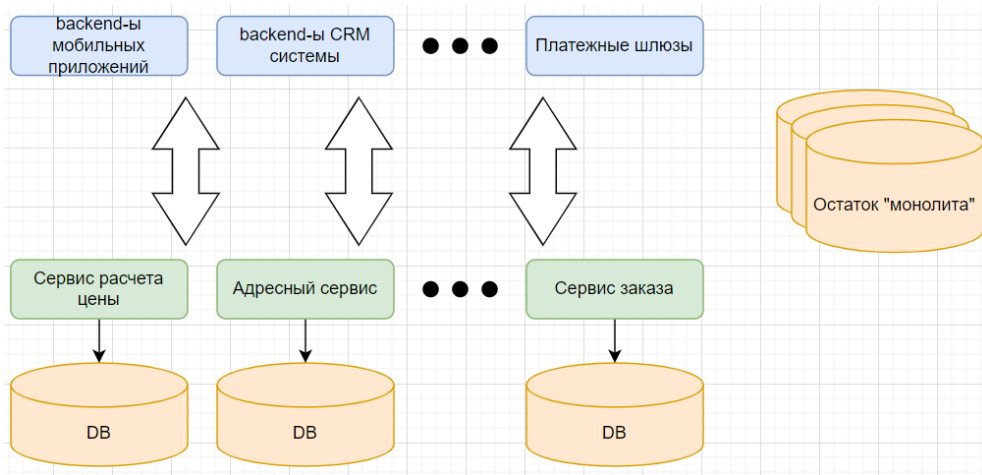
Страница статуса запуска тестов



Stage View

	scm	get test utils	pull images	build_db	Diff check	Translate test	prepare env	Unit tests	codestyle
Average stage times: (Average full run time: ~9min 35s)	2min 8s	4s	1s	51s	2min 9s	4s	17s	3min 54s	1s
#15235 Mar 22 15:27 No Changes	2min 10s	4s	1s	51s	8ms	4s	18s	0ms (paused for 0ms)	
#15234 Mar 22 15:24 No Changes	2min 7s	4s	1s	50s	3min 15s	4s	15s	5min 51s	1s

Сервисная архитектура



Немного (много) цифр

29 проектов
в разработке

119 инстансов баз
в эксплуатации

15 инстансов с **3 500**
транзакций в секунду —
самый нагруженный проект

4 дата-центра
по миру

В самом крупном проекте

970+ таблиц

1630 представлений

4280 функций

4,8 Tb объем БД

44 000 тест-кейсов, выполнение
укладывается в **6** минут

29 разработчиков
баз данных

3 DevOps
инженера

2 QA-инженера

643 принятых merge request-а
за прошедший месяц

Выводы



Такой подход позволяет

- ✓ эксплуатировать множество высоконагруженных БД силами небольшого количества людей
 - ✓ быстро разрабатывать новую функциональность
 - ✓ быстро доставлять до прода
 - ✓ иметь уверенность, что ничего не сломается после наката
 - ✓ эффективно мониторить и оптимизировать
- 

Теперь — вопросы

Ссылки, где есть я



📍 @sergusm

✉ s_mokeev@mail.ru

Ссылки, где есть gCodeKeeper

Официальный сайт

pgcodekeeper.org

Исходный код

github.com/pgcodekeeper/pgcodekeeper

Поддержка проекта в «Телеграме»

[@pgcodekeeper](https://t.me/pgcodekeeper)