

# Автоматизация миграции SQL-логики в проектах импортозамещения

Васильев Максим Сергеевич  
Архитектор Данных

# Васильев Максим: ☺ спикере



Архитектура с 2017

Отчетность с 2009

@todmay

## Ключевая идея

- Массовая миграция SQL-процедур в рамках импортозамещения невозможна без автоматизации.
- Ни один инструмент не даёт готового результата.
- Автоматизация экономит 30–60% времени, но 20–40% кода требуют ручной переработки.

## Постановка задачи

- “Всё уже есть”.
- Нам “просто перенести”.
- Оставьте “как есть”.



# Реальность миграции

- Основной риск — не данные, а процедурная логика.
  - Более 50% хранимых процедур содержат бизнес-логику, а не просто выборки.
- Автоматическая конвертация покрывает синтаксис, но не архитектуру.
  - «перенести таблицы» ≠ «перенести систему»
- Чем старше система, тем выше доля динамического SQL.
- Ограничения безопасности: отсутствие прямого доступа.

Проблема — это не SQL, а накопленная архитектура.

## Миграция старых систем

- В старых системах 10–15 лет доля динамического SQL достигает 25–40%.
- Более 30% процедур используют временные таблицы.
- Часто отсутствует документация зависимостей.

# Ограничения PostgreSQL, влияющие на SQL-логику

- большое количество процедур и зависимостей
- нет cross-database в PostgreSQL
- файловые операции ≠ целевая архитектура
- нет global temp tables
- нет linked servers
- иная модель блокировок



# Подготовка: зависимости, схемы, окружение

- Что можно сделать ДО конвертации:
  - ручная рекурсивная сборка зависимостей (DDL)
  - перепроектирование:
    - файлов → таблицы
    - cross-database → схемы вида БД\_Схема
  - проверка работоспособности логики на исходной БД
- Ключевая мысль:
  - Авто-конвертер не понимает бизнес-контекст и архитектурные ограничения — это ответственность команды.

# Инструменты автоматизации конвертации

- Готовые инструменты:
  - Convertum
  - SQLines
  - ora2pg (для Oracle)
  - AWS SCT
  - pgloader (для данных)
- Платформенные решения:
  - Postgres Pro Migration Toolkit
  - Arenadata инструменты миграции
- LLM и Шаблонизаторы.



# Сравнение инструментов

Инструмент	DDL	Процедуры	Данные
Convertum	отлично	средне	нет
SQLines	средне	средне	нет
pgloader	нет	нет	отлично
AWS SCT	хорошо	хорошо	частично

# Автоматическая конвертация: ожидания vs реальность

- Почему выбрали автоматизацию:
  - Большой объём скриптов
  - Ручная миграция не масштабируется
  - Опыт использования Конвертума
- Два режима: файлы vs метаданные БД.
- Лучшие результаты при анализе метаданных.

Экономия времени есть, «готового результата» — нет.

# Типовые ошибки автоперевода

- cross-database → DBLINK (что не всегда допустимо)
- ошибки с датами и интервалами
- лишние кавычки в именах
- некорректные типы и длины
- динамический SQL
- collation и сортировки
- поведение TOP vs LIMIT



# Где ломается автоматическая конвертация

- Реальные зоны риска:
  - Cross-database запросы (в PostgreSQL отсутствуют)
  - Динамический SQL
  - Различия в типах DATE / TIMESTAMP / implicit cast
  - TRY/CATCH vs EXCEPTION
  - Поведение временных таблиц
  - Case sensitivity и кавычки
- Практический вывод: автоконвертер решает до 60% синтаксических проблем, но 30–40% логики требуют ручной переработки.

# Пример. Cross-database

```
SELECT *  
FROM db2.dbo.table t  
JOIN db3.dbo.dict d ON ...
```

- Рабочие варианты
  - FDW
  - Репликация
  - Консолидация в одну БД (через схемы)
- Наш выбор: схемы вида `db_schema` как компромисс между архитектурой и трудозатратами.

# Пример. Динамический SQL — 100% ручная зона

```
EXEC('SELECT ... WHERE id = ' + @id)
```

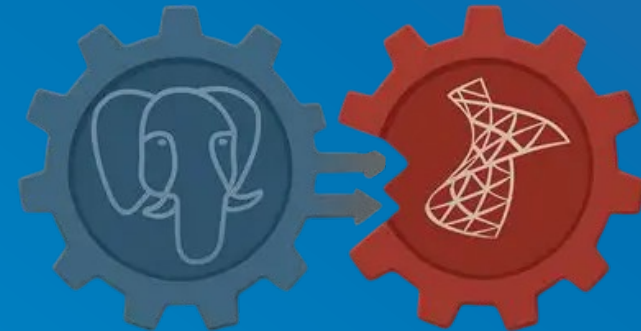
- PostgreSQL
  - EXECUTE через format()
  - другой контекст переменных
  - другая модель плана выполнения
- Факт: ни один инструмент не переносит динамический SQL без ручной переработки.
  - Если в системе >20% динамики — это уже не миграция, а рефакторинг.

# Итоговая схема процесса миграции

- Анализ и подготовка SQL-логики
- Архитектурные правки (схемы, файлы, зависимости)
- Классификация процедур по сложности
- Автоматическая конвертация
- Ручная доработка типовых проблем
- Тестирование
- Обратная связь в инструмент (кастомизация)

# Выводы и рекомендации

- Когда HE надо мигрировать “как есть”:
  - если >40% процедур — бизнес-логика
  - если heavy dynamic SQL
  - если cross-database архитектура
- Иногда дешевле:
  - вынести логику в приложение
  - переписать на ETL
  - внедрить DWH



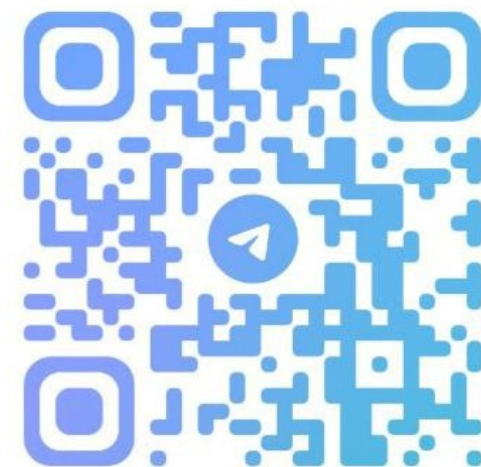
## Выводы и рекомендации

- Автоматизация миграции SQL — это усилитель инженерной команды, а не замена экспертизы.
- В импортозамещении выигрывают те, кто строит процесс, а не ищет идеальный инструмент.



@TODMAY

**Спасибо за внимание!**



@SELFDATAARCH

**Написать  
мне**

**Материалы  
доклада**