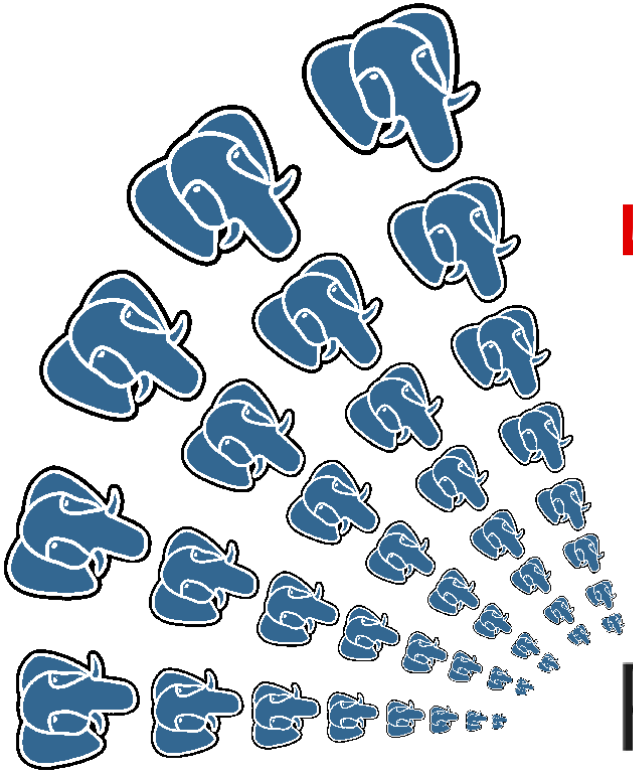
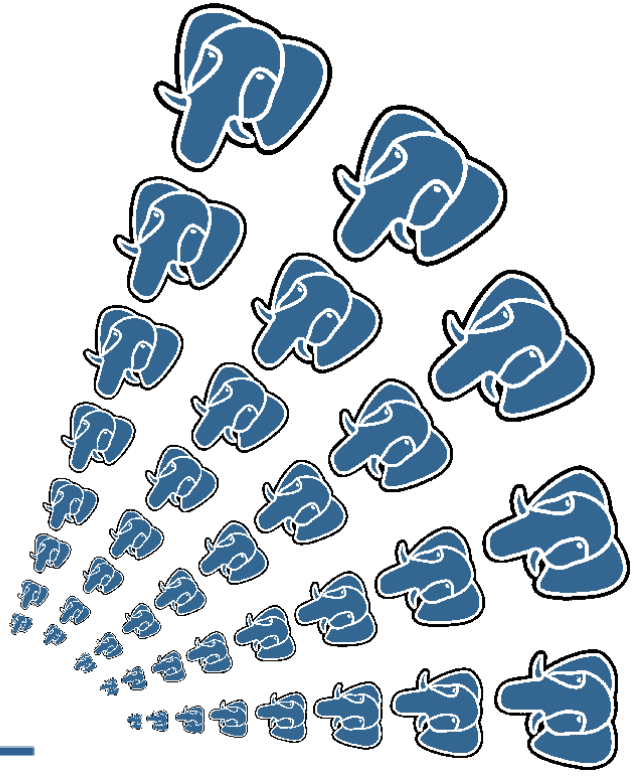


Как я перестал беспокоиться и мы перенесли 60K строк из 150 процедур PL/SQL в PostgreSQL

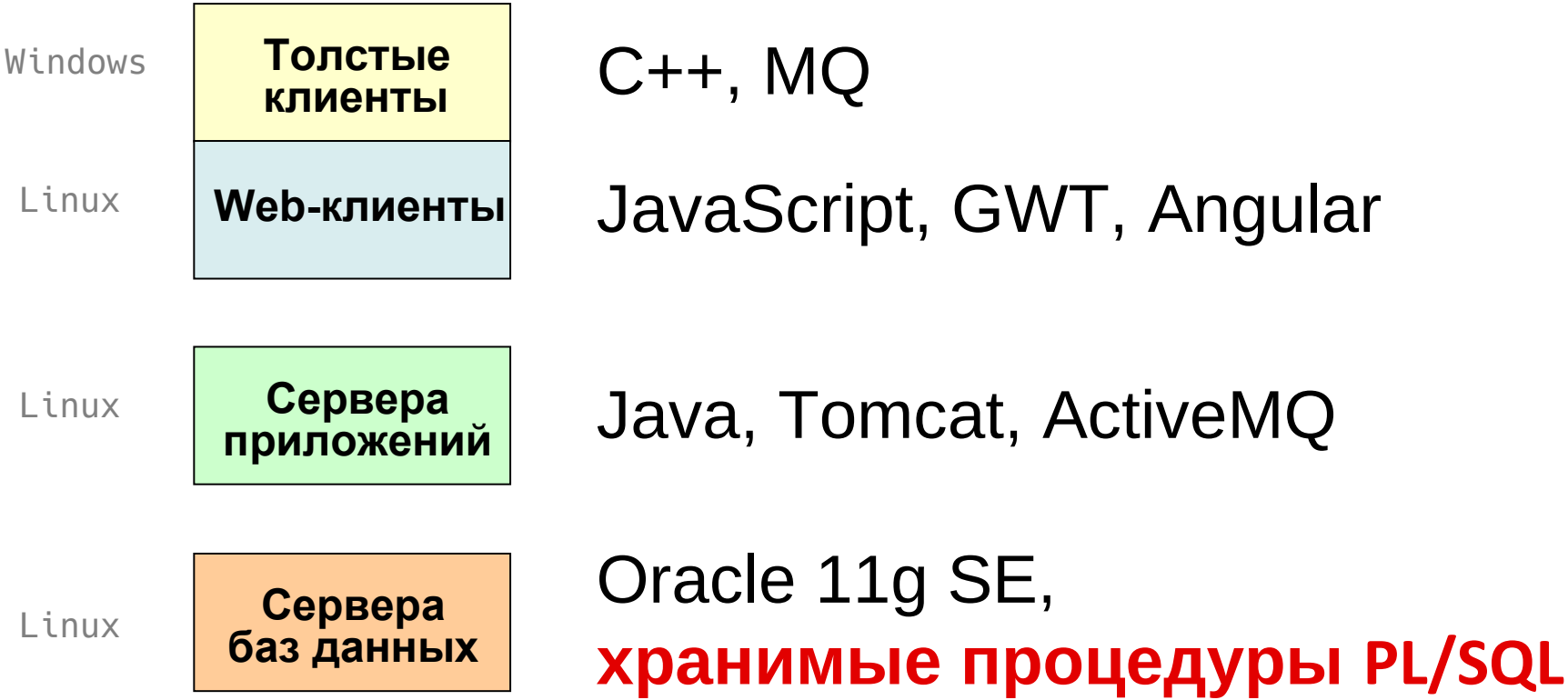


Анатолий Анфиногенов, АО “ВНИИЖТ”

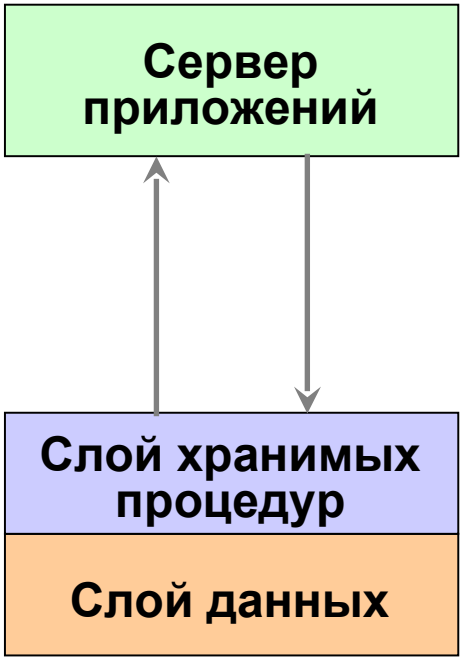
Пролог

- Распределенное технологическое приложение; работает на всех железных дорогах России от Калининграда до Хабаровска
- Oracle Database 11g Standard Edition
- Всего более 90 взаимодействующих серверов
- Более 40 серверов БД
- Эксплуатация 24/7; регулярные обновления серверных приложений 3-6 раз в год, включая БД

Трехзвенная архитектура



Почему хранимые процедуры?



- Отделяют (с помощью API) логику хранения от бизнес-логики приложения
- Позволяют вносить изменения в структуру БД без изменения сервера приложений (в пределах API) и облегчают обновление распределенного ПО (версионность API)
- Позволяют диагностировать, логировать, отлаживать и профилировать приложение без остановки сервиса

Структура и особенности базы данных

- Oracle Database 11g Standard Edition One
- Одна схема, порядка 250 таблиц, 50 Гб данных
- Приложение выполняет крупные (сотни Мб данных) и редкие транзакции (период в десятки секунд и минуты) плюс постоянно выполняются короткие мелкие транзакции (десятки раз в секунду)
- Более 200 хранимых процедур на PL/SQL (объем 60000 строк) в нескольких пакетах

Структура и особенности базы данных

- Редакция Oracle не включает standby; резервирование реализовано средствами приложения
- Использование автономных транзакций для логирования вызовов процедур и записи диагностической информации
- Использование временных таблиц для передачи большого объема данных (сотни Мб) при вызове хранимых процедур сервером приложений
- Широкое применение оператора MERGE в большинстве процедур, сохраняющих данные

ВНЕЗАПНО…

…ИМПОРТОЗАМЕЩЕНИЕ

Почему именно PostgreSQL?

- Наиболее близка по возможностям к Oracle Database 11g из всех СУБД, удовлетворяющим критериям импортозамещения
- Язык pl/pgSQL максимально близок к PL/SQL
- Есть замечательная команда **Postgres Professional** и большое и дружелюбное сообщество разработчиков
- Нет проблем с русскоязычной документацией, поддержкой, сертификацией и Реестром отечественного ПО

Что же нам предстояло сделать?

Поменять на ходу двигатель у локомотива так, чтобы пассажиры этого не заметили, то есть перенести таблицы, данные и хранимые процедуры так, чтобы все это продолжало работать уже на базе СУБД PostgreSQL прозрачно для приложения

Граничные условия

- Время – один год
- Силы – три человека: двое на PostgreSQL и один – на адаптацию сервера приложений
- Сервер приложений должен уметь работать и с версией для Oracle, и с версией для PostgreSQL
- Развитие функциональности системы продолжается и нужно выпускать новые версии для двух СУБД
- PostgreSQL – ванильный (sic!) версии 11.5

Кажется, что все еще не так уж и плохо

- За год до окончательного импортозамещения одна из вспомогательных подсистем (20 таблиц и 11 процедур) была нами перенесена на PostgreSQL 10.6
- Это позволило получить стартовый опыт в проектировании, разработке и администрировании PostgreSQL
- Переходом на PostgreSQL в своей предметной области мы занялись одними из первых, так что это – шанс сделать все самостоятельно и правильно (с нашей точки зрения)

ПЛАНИРУЕМ…

…СВОИ ДЕЙСТВИЯ

Что делать?

1) Разработать общую технологию миграции
2) Заняться самообразованием в части PostgreSQL
3) Разработать инфраструктурные workaround, без которых наше приложение не перенести
4) Перенести таблицы, VIEW, SEQUENCES и т.п.
5) Перенос и перепрограмирование хранимых процедур
6) Тестирование, отладка и оптимизация производительности

Что делать?

7) Разработать технологический процесс переключения

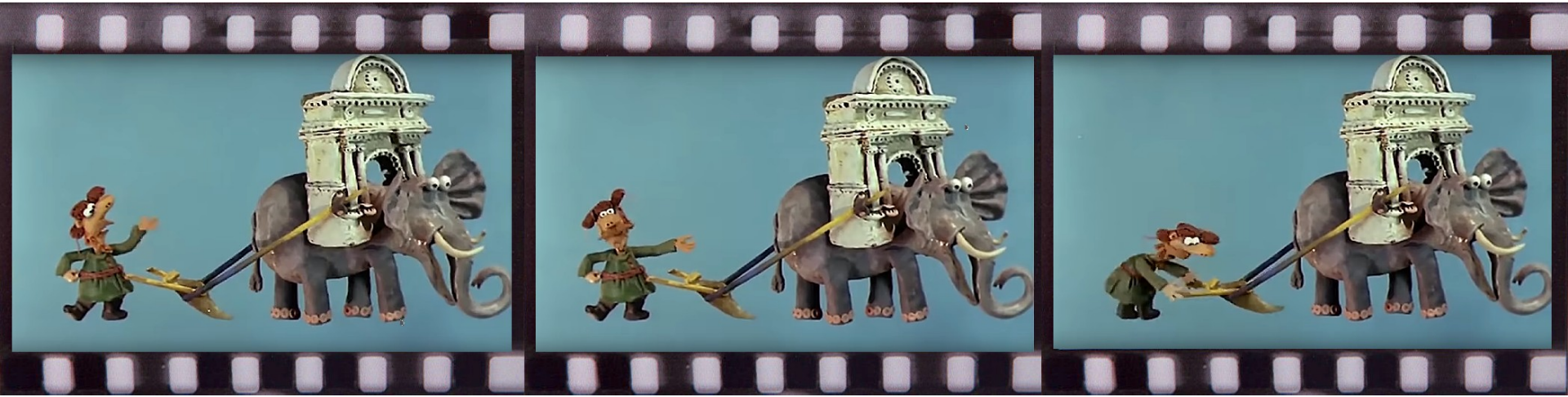
8) Разработать эксплуатационную документацию и инструкции для администраторов

9) Провести обучение администраторов

10) Организовать сопровождение, включая разработку технологии выполнения обновлений

11) Выполнить сдачу системы в эксплуатацию

ПРИСТУПАЕМ…



…К РАБОТЕ

1) Создаем общую технологию миграции

- Оцениваем примерный объем работ
(ура! – из 200 процедур переносить надо только 150!)
- Выбираем версию и редакцию PostgreSQL
(ответ: 11.5, ванильная)
- Придумываем и согласовываем с Заказчиком этапы и порядок работ в ходе миграции
- Планируем вычислительные ресурсы и заблаговременно организуем их выделение
- Продумываем порядок переноса данных

2) Самообразование в части PostgreSQL

- PostgreSQL – это другая СУБД! Это – не Oracle, хотя местами и похоже. Как следствие, pl/pgSQL – это не PL/SQL!
- Учимся правильно работать именно с PostgreSQL, не пытаясь механически воспроизвести рецепты из Oracle
- Узнаем об EXSTENSIONS; выбираем необходимые для проекта; учимся устанавливать и настраивать ОС, СУБД и расширения
- Выбор инструментария для работы с СУБД PostgreSQL (купили лицензии на EMS SQLManager for PostgreSQL)

СМОТРИМ…

…ПОД КАПОТ

3) Инфраструктурные workarounds - 1

- Для анализа работы приложения наши хранимые процедуры пишут логи в лог-таблицы независимо от успешности транзакции – нужны автономные транзакции.
- Ванильный PostgreSQL не поддерживает автономные транзакции
- Эмуляция автономных транзакций через dblink работает медленно, т.к. создание соединения – дорогая операция
- Нас спасает расширение pg_variables; запоминаем в переменной открытое соединение и логи начинают писаться с приемлемой скоростью

3) Инфраструктурные workarounds - 2

- Для получения больших объемов данных в виде входной информации для работы хранимых процедур нужны временные таблицы (в стиле Oracle).
- Ванильный PostgreSQL не поддерживает временные таблицы, сохраняющиеся в словаре данных СУБД по завершению сессии
- Эмуляция временных таблиц в стиле Oracle выполнялась с помощью UNLOGGED-таблиц, в которых задается уникальный идентификатор (GUID) слоя данных
- Проблемы: выбор GUID, очистка, производительность

3) Инфраструктурные workarounds - 3

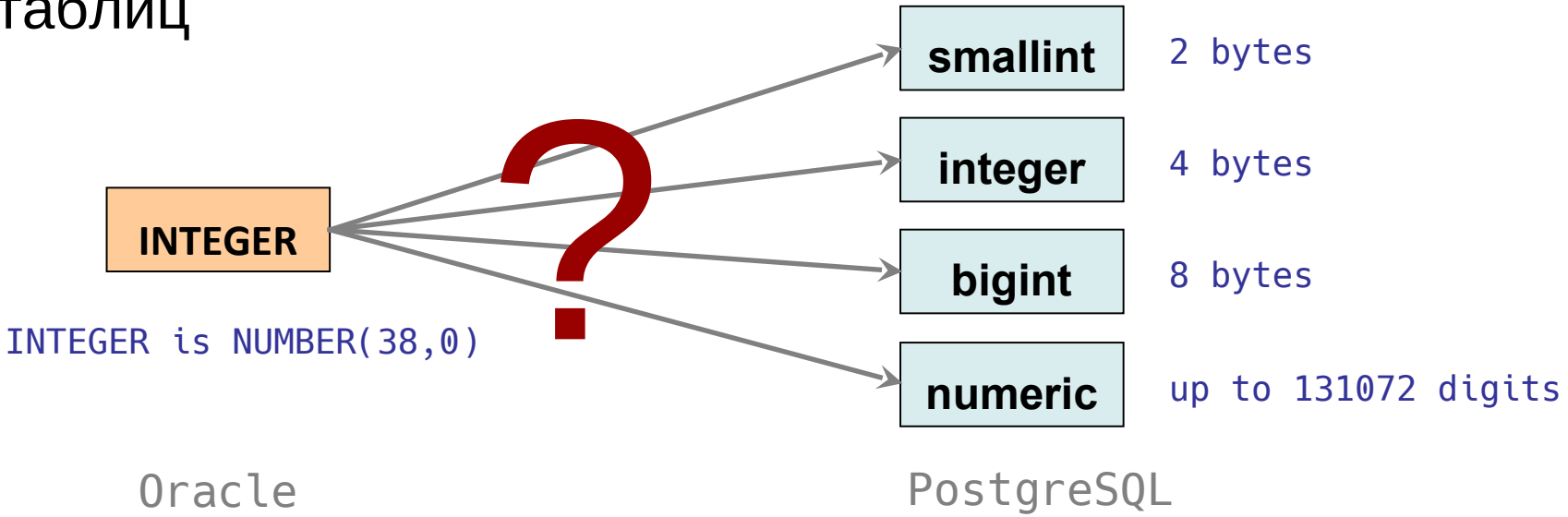
- Для выполнения задач архивирования устаревших данных, техобслуживания БД, диагностики и проактивного мониторинга мы использовали пакетные задания (JOB), которые вызывали процедуры с оператором COMMIT внутри
- Ванильный PostgreSQL не поддерживает JOB
- Для запуска процедур, реализующих эти функции, было создано Java-приложение jobrunner, исполняемое в Apache Tomcat. Плюсы: оно гибче и может выполнять не только задания БД. Минусы: более сложная конструкция

3) Инфраструктурные workarounds - 4

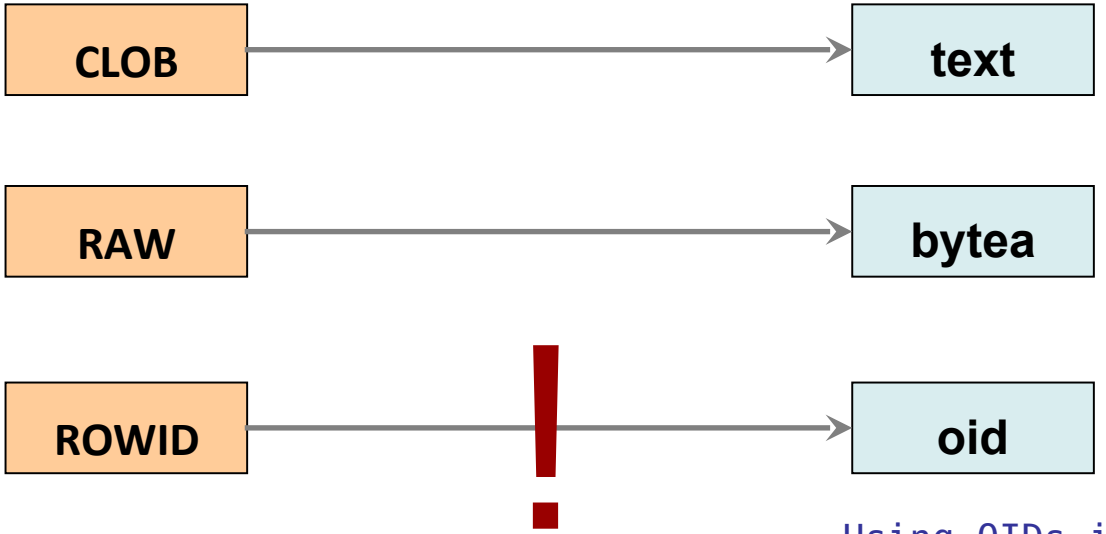
- Какие EXTENSIONS понадобились нам?
  1. Штатное расширение **dblink**
  2. Штатное расширение **postgres_fdw**
  3. Штатное расширение **pgcrypto**
  4. Штатное расширение **pg_repack**
  5. Нештатное расширение **pg_variables** (штатное для PostgresPro)
  6. Нештатное расширение **oracle_fdw**
- Нештатные расширения затрудняют обновление СУБД!

4) Перенос таблиц, VIEW, SEQUENCES...

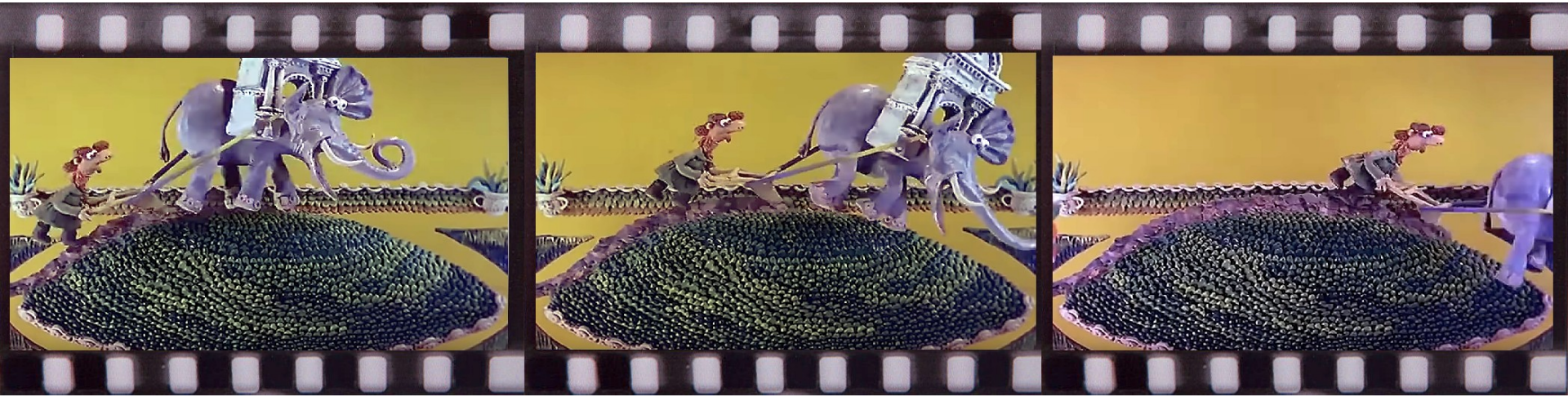
- Главный вопрос при переносе таблиц – сопоставление типов данных Oracle и PostgreSQL. Этот вопрос не решается механически для заметного числа полей таблиц



4) Перенос таблиц, VIEW, SEQUENCES...



ПРОДОЛЖАЕМ…



…ПАХАТЬ

5) Перенос хранимых процедур – 1

- Что делать с оператором MERGE? В большинстве случаев годится INSERT(…) ON CONFLICT DO UPDATE (…)
- Как быть с отсутствием PACKAGES? Нужно создать схему с названием пакета
- Не создавайте свои функции в схеме public: в ней EXTENSIONS создают свои функции; возможен конфликт имен!
- Операторы DDL не вызывают автоматический COMMIT! Их можно вызывать внутри функций, а также не забывать фиксировать транзакцию после создания функции

5) Перенос хранимых процедур – 2

- Функции в PostgreSQL могут иметь одно имя, но разный набор аргументов (зачастую ничтожно отличающийся по типу). Это – потенциальный источник больших проблем. Обеспечьте уникальность имен с помощью CONSTRAINTS
- Обработка EXCEPTIONS отличается. Используйте свою систему именованных ERRCODE, например:

```
RAISE EXCEPTION '%', v_ErrorMessage
        USING ERRCODE = 'EL023', HINT = 'ARCHIVE data not found!';
```

- Будьте осторожны с CTE – они материализуются! В новых версиях PostgreSQL это управляемо

5) Перенос хранимых процедур – 3

- В Oracle пустые строки и NULL – одно и то же; в PostgreSQL – нет!

```
IF ('' IS NULL) THEN => TRUE
```
Oracle

```
IF ('' IS NULL) THEN => FALSE
```
PostgreSQL

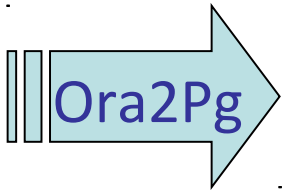
- Как силами двух человек за полгода перенести 150 хранимых процедур из Oracle в PostgreSQL? Без автоматизации – никак!
- Спасение: Ora2Pg! Спасибо, Gilles Darold!

5) Перенос хранимых процедур – 4

- Как правильно использовать Ora2Pg?

```
IF ( A = B ) THEN
    do_smth_1();
ELSIF (C IS NULL) THEN
    IF (NVL(D,0) = 0) THEN
       do_smth_2();
    ELSE
       do_smth_3();
    END IF;
ELSE
    RETURN 0;
END IF;
```

Oracle




```
IF ( A = B ) THEN
     PERFORM do_smth_1();
ELSIF (C IS NULL) THEN
    IF (COALESCE(D,0) = 0) THEN
        PERFORM do_smth_2();
    ELSE
        PERFORM  do_smth_3();
    END IF;
ELSE
    RETURN 0;
END IF;
```

PostgreSQL

- Результат трансляции – только канва (помним про MERGE, NVL2() и многое другое); правим руками

МЫ УЖЕ ПОБЕДИЛИ?...

...ЕЩЕ НЕТ

6) Настройка производительности – 1

- Сравним максимальные времена выполнения одной из ключевых процедур на одних и тех же данных:

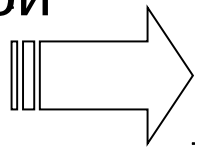
10 минут
Oracle

19 часов
PostgreSQL

- Еще рано впадать в отчаяние – попробуем заняться оптимизацией
- Вот теперь-то нам и пригодилось встроенное логирование и профилирование хранимых процедур

Как это выглядит:

- Для настройки производительности используем встроенные в приложение механизмы логирования и профилирования этапов выполнения процедур
- Пример ускорения одной из ключевых функций; очередная итерация



Ускорение	877%	
Время выполнения, мс:	Старая 171275	Новая 19526

Ускорение	Шаг	T_ms
3,00	2)	6
	2)	2
1,30	2.4)	13
	2.4)	10
1,40	3.1.2)	2309
	3.1.2)	1648
17,09	3.2.S)	48468
	3.2.S)	2836
1,04	3.3.1)	55
	3.3.1)	53
0,89	3.3.3.1)	24
	3.3.3.1)	27
	4.0)	20
497,54	4.1)	97517
	4.1.1)	67
	4.1.3)	82
200,00	5.1)	2
	5.1)	0
	5.2)	0
	5.2)	0
1,00	6.1)	6
	6.1)	6
0,83	7)	10
	7)	12
1,00	8)	18
	8)	18
1,55	9)	22812
	9)	14733
1,38	10)	11
	10)	8

6) Настройка производительности – 2

- За счет чего можно повышать производительность?
- Выявляем причины низкой производительности и принимаем меры по их устранению:

  1) Использование UNLOGGED-таблиц вместо временных таблиц – удобно и похоже на то, как было в Oracle, но отсутствие индексов и статистики замедляет работу.

     Решение: использовать CTE в запросах с UNLOGGED-таблицами

     Результат: ускорение в 3 раза
     (но 6 часов вместо 10 минут все равно плохо)

6) Настройка производительности – 3

2) Попробуем использовать UNLOGGED-таблицы только для приема данных от сервера приложений, а далее создать временные таблицы PostgreSQL внутри функции и работать уже с ними.

Решение:
```
CREATE TEMPORARY TABLE TMP_Input
    ON COMMIT DROP AS SELECT * FROM UL_Input;
```

Результат: ускорение в 600 раз

(стало 15 минут вместо 10 минут – почти паритет)

6) Настройка производительности – 4

3) А если еще поднажать? Вспоминаем, что в PostgreSQL внутри функций можно не только создавать таблицы, но и собирать статистику.

Решение: `ANALYZE TMP_InputData;`

Результат: ускорение в 1.5-3 раза

(стало 6 минут вместо 10 минут – УРА!!!)

- В рассматриваемой функции резервы повышения производительности практически исчерпаны. Как повышать производительность в функциях другого типа?

6) Настройка производительности – 5

4) Оказывается, использование специфических только для PostgreSQL нестандартных форм операторов UPDATE и DELETE дает прирост скорости до 40% по сравнению с использованием подзапросов вида

... WHERE EXISTS (SELECT ...) или ... WHERE X IN (SELECT ...)

Примеры:

```
UPDATE IDs  SET ID = Q.NEW_ID
FROM   (SELECT NEW_ID, ID FROM TMP_IDs) AS Q
WHERE (Q.ID = IDs.ID);
```

```
DELETE FROM IDs
USING  TMP_IDs
WHERE (IDs.ID=TMP_IDs.ID);
```

ГОТОВИМСЯ…

…К ПЕРЕКЛЮЧЕНИЮ

7) Разработка технологии переключения

- Как перенести данные из Oracle в PostgreSQL?
- Как обеспечить переходный период – параллельную эксплуатацию приложений для Oracle и PostgreSQL с возможностью быстрого возвращения на Oracle в случае обнаружения сбоев или крупных и неприятных сюрпризов?
- Как поддерживать синхронность данных в БД Oracle и PostgreSQL в течении переходного периода?

8) Разработка эксплуатационной документации

- Разработка подробных пошаговых инструкций на русском языке по установке системы, включая установку и настройку ОС, СУБД, EXTENSIONS, импорта стартовой БД и всех объектов, а также настройки приложения БД
- Разработка подробного руководства администратора системы на русском языке, включая администрирование БД (резервное копирование и восстановление, горячее резервирование, настройка параметров, мониторинг, техобслуживание и т.п.)

9) Обучение администраторов

- Практическая отработка подробных пошаговых инструкций на русском языке из п.8) с внесением в них изменений и уточнений
- Проведение очного и дистанционного обучения
- Проведение семинаров на базе ВНИИЖТ

10) Поддержка и обновление ПО – 1

- В части обновления прикладного ПО БД PostgreSQL обладает огромным преимуществом перед Oracle: операторы DDL не вызывают автоматический COMMIT
- Обновление прикладного ПО БД в Oracle: восстановление из резервной копии после первого же оператора ALTER TABLE … в случае сбоя при обновлении
- Обновление прикладного ПО БД в PostgreSQL: в случае сбоя при обновлении восстановление к исходному состоянию производится одной командой ROLLBACK;

10) Поддержка и обновление ПО – 2

- В части обновления ОС и СУБД ванильный PostgreSQL в нашей конфигурации обладает серьезным недостатком: некоторые расширения (pg_variables, oracle_fdw) скомпилированы из исходных кодов, поэтому обновляться командой yum update нельзя
- Выход – использование PostgresPro

ПОДЧИЩАЕМ ХВОСТЫ…

…И СДАЕМ ЗАКАЗЧИКУ

11) Ввод системы в эксплуатацию

- С октября 2019 года система на базе СУБД PostgreSQL работает в режиме постоянной эксплуатации на одной из 16 железных дорог России
- Конечные пользователи ничего (плохого) не заметили; быстродействие на тех же вычислительных мощностях не уступает исходному
- На оставшихся 15 дорогах система на базе СУБД PostgreSQL работает в качестве горячего резерва; к 01.07.2020 ожидается окончательный вывод системы на базе СУБД Oracle из эксплуатации

ПОДВОДИМ…

…ИТОГИ

Эпилог

- Подведем итоги – миграция на СУБД PostgreSQL завершилась в целом успешно
- Один из дальнейших вопросов – это вопрос сопровождения приложения и управления быстродействием как его, так и БД: перестроение индексов, оптимизация системы хранения и т.п.
- Другой вопрос – организация обновлений до следующей мажорной версии с учетом зависимости от EXTENSIONS и прекращения поддержки OID
- Да, самое главное – наш заказчик начинает получать выгоду от импортозамещения!

МАЛОВАТО, ПОНИМАЕШЬ…

…МАЛОВАТО

Что бы хотелось иметь в PostgreSQL

- Блокировки строк с заданным таймаутом (a-la Oracle):

  ```
  SELECT ID FROM IDs FOR UPDATE WAIT 60;
  ```

- Ослабление чрезмерной строгости в указании типов для параметров хранимых функций при их вызове.

  Необходимость писать f ('A'::VARCHAR) в качестве аргумента функции вместо f ('A') на мой взгляд – перебор

- Иметь возможность запросить план выполнения только что выполненного SQL-оператора, находясь внутри встроенной функции, для логирования и последующего анализа производительности

Что бы хотелось иметь в PostgreSQL

- Усеченный функционал автономных транзакций хотя бы для записи логов
- В утилите pg_dump иметь возможность:
  - брать параметры не из командной строки, а из файла
  - задавать дополнительные фильтры для данных, извлекаемых из отдельных таблиц наподобие `--sqlfilter=IDS:"ID<10“`
  - возможность сортировать объекты (хотя бы по именам) для того, чтобы иметь возможность сравнивать две БД при выгрузке в файл только метаданных

СПАСИБО ЗА...

…ВНИМАНИЕ

О докладчике

АО «ВНИИЖТ» (АО «Научно-исследовательский институт железнодорожного транспорта»)

Анфиногенов Анатолий Юрьевич

Заместитель директора научного центра – начальник отдела разработки ПО, кандидат физико-математических наук.

anfinogenov.anatoly@vniizht.ru

+7 (499) 262-45-06

Работаю уже два десятка лет во ВНИИЖТ над задачами имитационного моделирования и оптимизации железнодорожных перевозок.

Проектировал, разрабатывал и сопровождал серверное ПО для этих задач (Oracle, C++, Python), чем и продолжаю заниматься.

Третий год погружаюсь в мир PostgreSQL и мне это все больше начинает нравиться.