Миграция баз данных на потоке

Роман Друзягин
технический директор, 404 Group
roman.druzyagin@404-group.com

Причины миграции?

- Ситуация технологического кризиса в проекте.
- Прихоти людей на управляющих позициях в техотделе.
- Сторонние факторы (экономические, политические и т.д.)

Типичная история

- Регулярно случается простой сервиса по причине базы.
- Нет бэкапа/high-avaliable решения (почему-то).
- Документация минимальна.
- Есть локальный гуру, который торжественно и таинственно умеет базу чинить.
- Для бизнеса простой становится нормой.

Когда локальный гуру ушёл
(или его “ушли”)

- Система жизнеобеспечения оставшегося в наследство решения устроена крайне неочевидно.
- Никто не имеет представления, с какой стороны можно подступиться, не поломав все напрочь.
- Орех занимает не менее 80% рабочего времени техотдела.
- Масштабирование команды не приносит должного эффекта, соизмеримого с затратами.
- На Сарех ресурсов практически не остается.

Приходит понимание, что так жить нельзя

- На данном этапе желание изменить ситуацию обычно высказывает бизнес.

- Если бизнес не знает (или не хочет знать) состояние дел в техничке, прецедент надо создать.

Когда делать миграцию противопоказано?

- Не исчерпаны все способы решить проблемы в текущих условиях.
- Проект новый и не изучен настолько, чтобы иметь возможность с какой-то долей уверенности оценить работу после миграции.

Планирование миграции

- Выделяемое бизнесом время на проведение работ.
- Количество человек, которые будут заниматься миграцией.
- Будет ли параллельно вестись разработка новой функциональности?
- Наличие у коллектива требуемых для осуществления миграции компетенций.
- Оценка объема и сложности кода, который надо будет переработать и протестить для миграции.
- Оценка способностей предполагаемого storage, исходя из нагруженности системы (крайне актуально для “highload” проектов).

Выбор технологии
или почему PostgreSQL?

- Адекватная, “честная” база, которая решает широкий спектр задач: “хайлоад”, хранение и подсчет денег, “наукоемкие” вычисления, аналитические инструменты и проч.
- Предостаточно средств для разработки и эксплуатации.
- Отличное “коммьюнити” и документация.
- Накопившийся за годы практики большой опыт в коллективе.
- Компетентные консалтеры-dba с 24/7 саппортом.

На практике

При переезде с любой некоммерческой базы на PostgreSQL
жить становится резко лучше.

// этого слайда не было на highload ;)

Проработка деталей

- Устранение bloat и лишней информации из СУБД.
- Миграция схемы и данных.
- Переработка кода, тщательное тестирование.
- Отработка процедуры миграции, формализация плана (важно!).
- Оценка количества узлов, которые будут недоступны во время миграции и связанных с этим рисков/потерь для бизнеса.
- Обсуждение процесса миграции с нетехническим персоналом.
- Закладывание времени, требуемого на проведение миграции.
- Проработка плана Б.
- Закладывание времени и трудовых ресурсов на постмиграционный период.

На практике

- Переработка кода всегда займет больше времени, чем планировалось.
- Обязательно устраните bloat!
- Будьте готовы не взлететь с первого раза (и готовьте бизнес).

Как освоить технологию командой?

- Сильно зависит от отношений в коллективе и с техлидом.

- Не затевайте миграцию, если в коллективе нет хотя бы одного компетентного лица, владеющего технологией!
  - Рискуете платить всю жизнь деньги сторонней конторе, которая вам смигрирует и потом будет поддерживать базу ;)

Как мы обучаем?

- Правила разработки и guidelines.
- Учебные семинары (раз в полгода).
- Постоянное ревью кода и “разбор полетов”.
- Дотошность к деталям и микроменеджмент разработки.

Как организовать работу команды над миграцией?

- Заранее проработайте все типовые задачи и составьте базу знаний.
- Чётко обозначьте, “что такое хорошо, а что такое плохо”.
- Выработанным правилам работы со старым и новым кодом должна подчиняться вся команда, без исключения.
- Организуйте “горячую линию” связи с компетентным специалистом, владеющим технологией.

Как смигрировать данные?

- Схема мигрируется только вручную.
- Чем проще, тем лучше.
- Не изобретайте универсальных велосипедов для решения одноразовой задачи!
- Всеми силами избегайте переработки бизнес-логики.

На практике

- Велосипед строился полтора года, колеса получились квадратные, далеко он не уехал.
- Ручная переработка схемы – 1 день.
- Написание “наколенной” тулзы – 1 день.
- Неделя на обработку в тулзе исключительных ситуаций.

“Best practices” по тестированию

- Тестирование должно быть ручным и выполняться профессиональным тестировщиком.
- Новую функциональность надо сразу “мигрировать” и независимо проверять.
- Компоненты с нетривиальной бизнес-логикой (биллинг, парсеры, асинхронная обработка и т.п.) надо перерабатывать в первую очередь и тестировать на протяжении длительного времени.
- Процесс миграции данных надо тестировать как **backup/recovery-решение**

На практике

- Тестируйте много и часто.
- Делайте по несколько подходов к тестированию, будут всплывать новые проблемы.
- Обязательно проверьте, как система запустится сразу после даунтайма!
- Проведите тестовые миграции!

Час Ч

- Все действующие лица должны быть выспавшиеся, бодрые, трезвые.
- Должно быть обеспечено наличие у всех нормального интернета и резервных каналов, проведена плановая проверка рабочих станций.
- Порядок действий от первого до последнего пункта должен быть расписан с максимальной точностью на бумаге/в документе.
- “Штатные” внештатные ситуации тоже должны быть запланированы и расписаны.
- Правильно составленный план позволяет “разыграть” миграцию словно по нотам.

Как удостовериться, что миграция удалась?

- Миграция не потребовала большого количества дополнительного времени.
- Вам не пришлось подпирать “костылями” ключевые компоненты системы.
- Вам не потребуется еще две недели простоя бизнеса на то, чтобы “зафиксить все баги”.
- Неожиданные проблемы – повод откатиться!

Светлое будущее: наступило или нет?

- Количество бизнес-хотелок после миграции удвоится.
- До повторной миграции лучше дело не доводить.
- Новую базу надо теперь уметь воспитывать.

Первый раз – он самый трудный!

- Около десятка MySQL разной степени запущенности.
- 3 MongoDB.
- 2 Sphinx.
- Несколько Redis.
- Самописные NoSQL, придуманные еще до того, как хипстеры придумали слово NoSQL.

to be continued...

404
GROUP

Вопросы?

Роман Друзягин, 404 Group
roman.druzyagin@404-group.com
skype: roman.druzyagin