Как прекратить решать проблемы хранилища и начать разрабатывать бизнес-логику

Денис Милованов

movebo.ru

Про Movebo

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.

Исторический экскурс

Первая итерация: **нечто на node.js.**

Вторая итерация — комм. успешная: **MySQL + Redis.**

Из плюсов:

- нормальная эксплуатация БД (насколько это возможно),
- Redis обеспечивает нужную функциональность
  и годится для чернового варианта (“потом перепишем”).

Спустя 2 года

- **С базой проблемы:**
  множество дедлоков, неявное поведение запросов
  (group by, etc.), отсутствие необходимой функциональности.

- **Redis оказывается не удачен** с точки зрения новых задач.

...это тормозит коммерческое развитие и всех нервирует.

Миграция на PostgreSQL

- **Миграция, собственно, базы MySQL:**

миграция схемы, MySQL-specific операторов, тестирование компонентов, аккуратный последовательный деплой всего.

- **Миграция сервиса выдачи (Redis)**

— далее.

Сервис выдачи

Абстрактно:

- набор задач, которые требуется раздавать пулу исполнителей,
- ограничения по числовым параметрам
  (напр., число выполненных в день задач),
- ограничения по временным параметрам
  (напр., нельзя выдавать выполненную задачу в течение дня).

Выдача на PHP + Redis

1. Задачи в виде списков внутри Redis.
2. Выдача на PHP: вычитывание списка, поиск в цикле.
3. Expirations встроены в Redis.

Проблемы:

1. Транзакционность?
2. Расширение функциональности?

```
while (!$exit) {
    if ($this->_redis-> setnx($lock_key, 1)) {
        $exit = true;
        $result = true;
        $this->_redis-> expire($lock_key, ...);
    } else {
        $this->_redis-> watch($lock_key);
        $c = $this->_redis-> get($lock_key);

        if ($c < $count) {
            $k = $this->_redis-> get($lock_key);
            $r = $this->_redis-> multi()
                                -> incr($lock_key)
                                -> exec();

            if ($r == FALSE) {
                $result = false;
                 $exit = true;
                $this->_redis-> unwatch();
            }
        else {
            $this->_redis-> expire($lock_key, ...);
            $exit = true;
            $result = true;
            $this->_redis-> unwatch();
        }
    } else {
        $exit = true;
        $result = false;
        $this->_redis-> unwatch();
    }
}
```

Выдача из PostgreSQL

- Ограничения и текущие значения — в разных таблицах (*limits / *counts).
- Также: выдачи, отчеты, локи.
- Максимум констрейнтов.
- API на хранимых процедурах.
- Expirations в виде системы джобов.

Foreign keys, checks, uniques

FK — целостность данных “из коробки”,

особое внимание обновлениям записей таблиц с FK (9.2),

checks — логические проверки на корректность данных в строках (если статус такой-то, то другое поле is not null),

unique-индексы, в т.ч. **условные** — ограничения типа “одна задача выполняется в данный момент одним пользователем”.

Foreign keys, checks, uniques

```
-- в каждый момент времени один серфер может выполнять (performed_at is null)
-- только одну задачу
CREATE UNIQUE INDEX serfers_tasks_locks_unique_task_on_the_run
    ON cache.serfers_tasks_locks
    USING BTREE (serfer_id)
    WHERE performed_at IS NULL;

-- у задания по которому принято решение есть время принятия решения
ALTER TABLE cache.moderation_queue
    ADD CONSTRAINT moderation_queue_logical_check2
    CHECK ( CASE WHEN status_id IN (2, 3) -- MODERATION.APPROVED, MODERATION.DECLINED
                    THEN moderated_at IS NOT NULL
                    ELSE TRUE
            END);
```

Stored functions

API выдачи (все запросы на модификацию данных и сложные выборки) — в хранимых процедурах:

- один коннект + одна транзакция,
- begin / exception (хотя и по минимуму),
- advisory locks (тоже; + словили advisory deadlock),
- позволяет разделить работу между сотрудниками + хранить логику в одном месте, когда git submodules уже поздно внедрять.

mbus

0. Задача: сделать аналог expirations в Redis.

1. mbus — асинхронная очередь (post / consume)
   с отложенной доставкой (delayed_until).

2. Джобы на обработку заданий типа “снять лок через N минут”.

3. Воркеры на PHP.

https://code.google.com/p/mbus/

Deployment, 1

Одна база, таблицы и индексы вручную, остальное — автоматически на основании кода в репозитории:

db.schema.git/database/schema/

- /seeds — сиды
- /types — типы
- /functions — хранимые процедуры
- /tables — таблицы + индексы + констрейнты

Deployment, 2

Автоматическая инкрементальная раскатка PHP-скриптом.

- Инкрементальность на основании md5 от текста кода в репозитории.
- Дроп изменившихся типов с зависимостями (в том числе хранимки с переменными этого типа, на основании кода хранимки; drop type cascade не удаляет их).
- Одна транзакция.

```
---------------------------------
~/work/movebo/daemons/tools[feature/queries-stat-doubles] $ php tool-deploy-schema.php
Есть типы: /home/deni/work/movebo/db.schema/movebo_maindb/cache/types/.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/types/t_found_impracticable_task.sql.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/types/t_found_task.sql.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/types/t_log.sql.
                Выполняем:  /home/deni/work/movebo/db.schema/movebo_maindb/cache/types/t_serfer_info.sql.
                Дополнительно дропаем зависимость:  DROP FUNCTION cache.get_serfer_last_timeout(i_serfer_id bigint);.
                Дополнительно дропаем зависимость:  DROP FUNCTION cache.serfer_process_subject(i_serfer_id bigint, i_task_id bigint);.
                Дополнительно дропаем зависимость:  DROP FUNCTION cache.find_task(i_serfer_id bigint, i_type_id integer, i_geo_id integer, i_gi
ven_task_id bigint, i_kind_id integer);.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/types/t_serfer_session.sql.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/types/t_serfer_site_lock.sql.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/types/t_serfer_task_lock.sql.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/types/t_serfer_task_response.sql.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/types/t_serfer_task_types_info.sql.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/types/t_setting.sql.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/types/t_software_response.sql.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/types/t_task.sql.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/types/t_task_info.sql.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/types/t_tasks_cache.sql.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/types/t_type_id_geo_id.sql.
---------------------------------
Схема cache.
Есть сиды: /home/deni/work/movebo/db.schema/movebo_maindb/cache/seeds/.
                Выполняем:  /home/deni/work/movebo/db.schema/movebo_maindb/cache/seeds/manual_task_templates_bodies_after.sql.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/seeds/tasks_screenshots_check_statuses.sql.
                Выполняем:  /home/deni/work/movebo/db.schema/movebo_maindb/cache/seeds/observed_parameters.sql.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/seeds/manual_task_actions.sql.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/seeds/manual_task_transitions.sql.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/seeds/log_cases.sql.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/seeds/manual_task_templates_headers.sql.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/seeds/manual_task_templates_bodies_before.sql.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/seeds/manual_task_templates.sql.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/seeds/actions_types.sql.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/seeds/manual_task_templates_bodies.sql.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/seeds/system_settings.sql.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/seeds/cache_geo_group.sql.
Есть хранимые процедуры: /home/deni/work/movebo/db.schema/movebo_maindb/cache/functions/.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/functions/add_session_keys_by_uid.sql.
                ПРОПУСКАЕМ: /home/deni/work/movebo/db.schema/movebo_maindb/cache/functions/add_to_errors.sql.
```

logs / observations

- Все, что происходит значимого, логируется.
- Раз в минуту — снятие ключевых параметров.
- Ошибки PG перехватываются на PHP и тоже складываются в базу.

Интерфейс на все + стратегия очистки.

Польза от внедрения PG

- Все преимущества реляционной модели хранения данных (структурность, целостность, сложные аналитические выборки и т.п.).

- Быстрое внедрение новых возможностей (4 новые сложные “фичи” в месяц).

- Оптимизация быстродействия — быстрая выдача (поиск по таблице вместо последовательного php+redis поиска) и т.п.

Проблемы и пути решения

- Дедлоки — внимательное проектирование,
  advisory locks, 9.2 -> 9.3 (NO KEY UPDATE / KEY SHARE).

- Рост таблиц (выдачи, отчеты) — партиционирование,
  архивирование старых данных.

Вопросы?

;)