Реализация элементов логики приложения в MySQL: триггеры, хранимые процедуры, кэширование

Сергей Горшков, технический директор
Центра информационных технологий index.art

Когда нужны триггеры? Пример 1, складской учет методом FIFO (First In First Out)

Товар, пришедший на склад первым, первым уходит со склада.

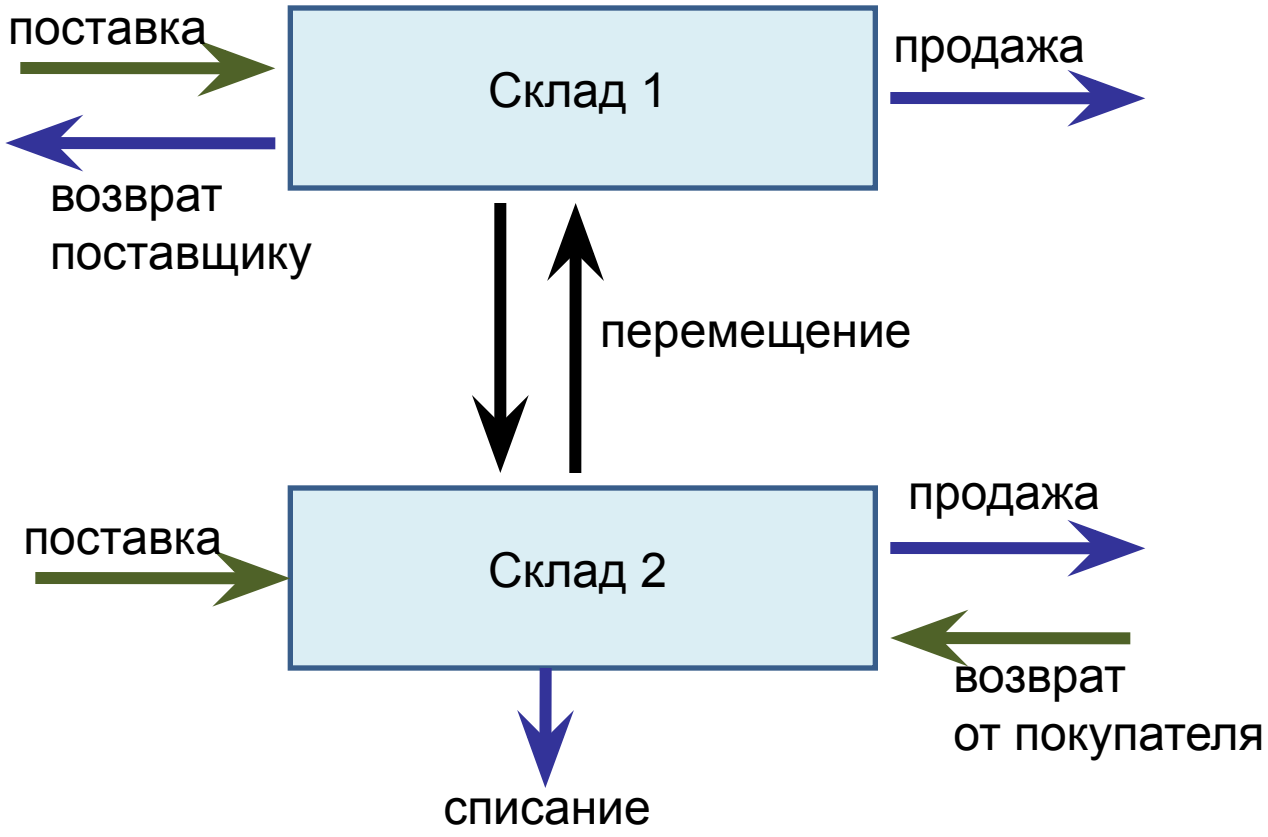
Пришло: 1 единица товара по 5 рублей
Пришло: 2 единицы товара по 10 рублей
Продано: 2 единицы товара по 15 рублей

Каков наш доход от продажи?

Неверно: 15 * 2 – 10 * 2 = 10 рублей

Верно: (5 * 1 + 10 * 1) – 10 * 2 = 5 рублей

Складской учет: схема процесса



Документы складского учета

Поставка
Дата
Склад
Поставщик
Кладовщик
Ревизор
…

Список товаров		
Товар	Цена	Количество

Продажа
Дата оформления
Менеджер
Склад
Покупатель
Дата отгрузки
…

Список товаров		
Товар	Цена	Количество

Перемещение
Дата отправки
Склад-отправитель
Склад-получатель
Дата прибытия
Ответственный
…

Список товаров	
Товар	Количество

Как узнать остаток товара???

Суммирование по спискам товаров из разных документов не подойдет:

- Много однотипных запросов к разным таблицам;
- Придется многое переписывать при добавлении новых типов документов;
- Операции получения остатка выполняются постоянно, а такой набор запросов будет выполняться довольно медленно;
- Хранить промежуточные остатки плохо, потому что данные часто меняются «задним числом».

Решение: создаем реестр операций складского учета

Дата	Склад	Товар	Кол-во	Цена
…				

Реестр операций складского учета

Дата	Склад	Товар	Кол-во	Цена
…				

поставка

Реестр операций складского учета

Дата	Склад	Товар	Кол-во	Цена
дата1	склад1	товар1	10	60
дата1	склад1	товар2	20	30
дата1	склад1	товар3	40	20
…				

поставка

Реестр операций складского учета

Дата	Склад	Товар	Кол-во	Цена
дата1	склад1	товар1	10	60
дата1	склад1	товар2	20	30
дата1	склад1	товар3	40	20
дата2	склад1	товар2	- 10	40
дата2	склад1	товар3	- 40	40
…				

поставка

продажа

Реестр операций складского учета

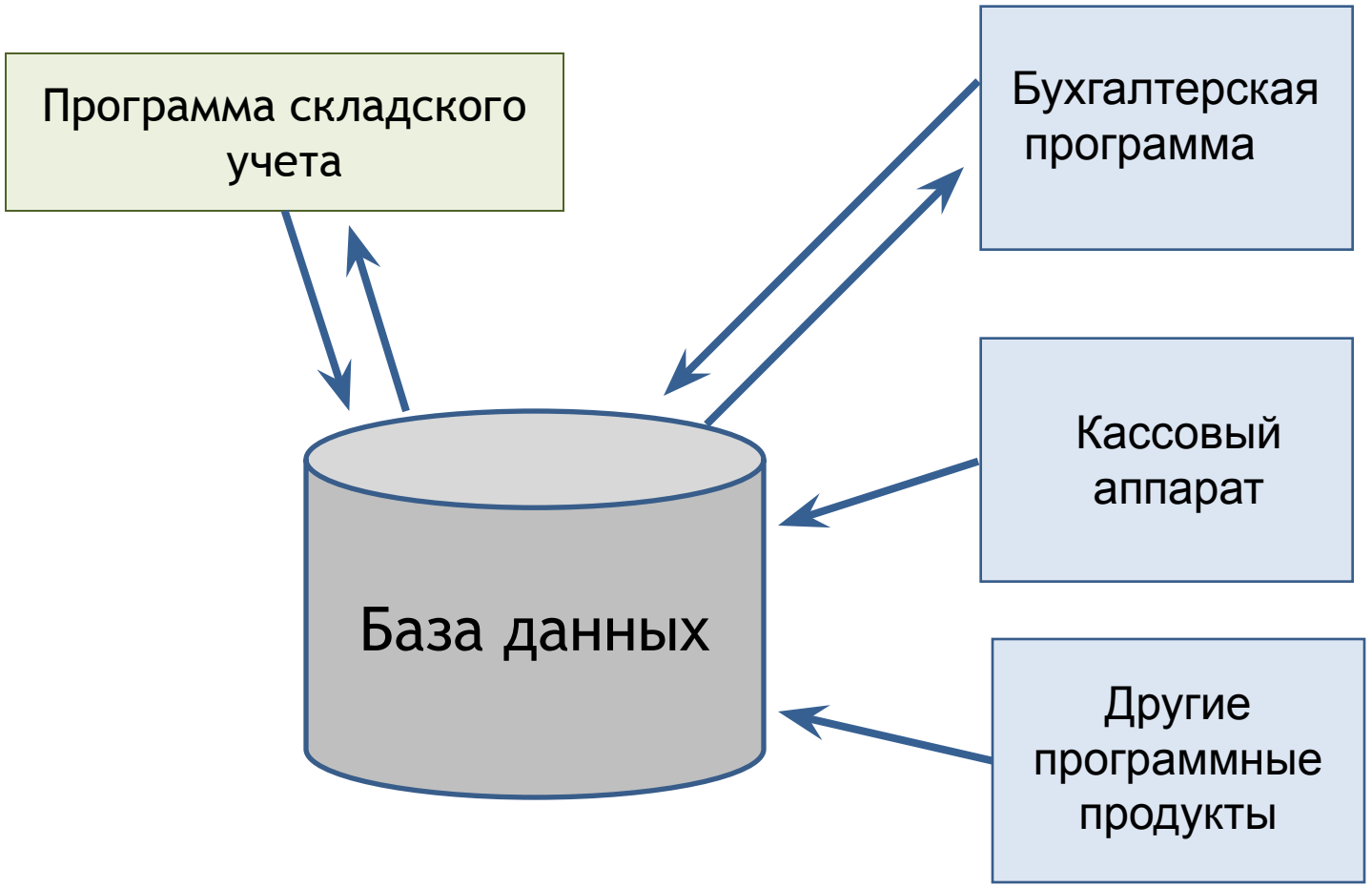
Дата	Склад	Товар	Кол-во	Цена
дата1	склад1	товар1	10	60
дата1	склад1	товар2	20	30
дата1	склад1	товар3	40	20
дата2	склад1	товар2	- 10	40
дата2	склад1	товар3	- 40	40
…			Σ	

поставка

продажа

суммирование по этому столбцу дает остаток товара на любую дату

Как поддерживать актуальность данных в реестре?


Программа складского учета
Бухгалтерская программа
Кассовый аппарат
База данных
Другие программные продукты

Триггеры для построения реестра

Дата	Склад	Товар	Кол-во	Цена
…				

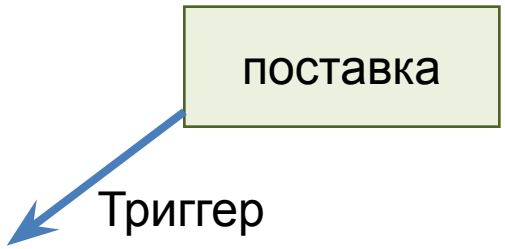
Триггеры для построения реестра

Дата	Склад	Товар	Кол-во	Цена
…				

поставка

Триггеры для построения реестра

Дата	Склад	Товар	Кол-во	Цена
…				



Триггеры для построения реестра

Дата	Склад	Товар	Кол-во	Цена
дата1	склад1	товар1	10	60
дата1	склад1	товар2	20	30
дата1	склад1	товар3	40	20
…				

поставка

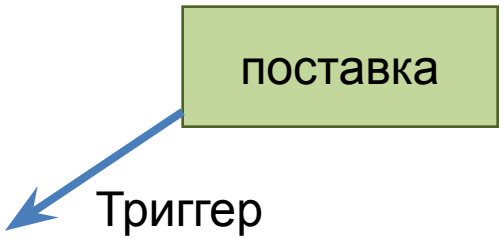
Триггеры для построения реестра

Дата	Склад	Товар	Кол-во	Цена
дата1	склад1	товар1	10	60
дата1	склад1	товар2	20	30
дата1	склад1	товар3	40	20
…				

поставка

Триггеры для построения реестра

Дата	Склад	Товар	Кол-во	Цена
дата1	склад1	товар1	10	60
дата1	склад1	товар2	20	30
дата1	склад1	товар3	40	20
…				



Триггеры для построения реестра

Дата	Склад	Товар	Кол-во	Цена
дата3	склад1	товар1	10	60
дата3	склад1	товар2	20	15
дата3	склад1	товар3	40	20
…				

поставка

Триггеры для построения реестра

Дата	Склад	Товар	Кол-во	Цена
дата3	склад1	товар1	10	60
дата3	склад1	товар2	20	15
дата3	склад1	товар3	40	20
…				

поставка

Нужна процедура полного или частичного пересчета реестра!

Хранимые функции

getGoodsRemainder (товар, дата, склад)
- возвращает остаток товара на заданную дату на конкретном складе

getSaleIncome (продажа)
– доход от одной конкретной продажи

getGoodsIncome (товар, склад, период)
– доход, полученный от реализации конкретного товара за указанный период

getSelfCost (товар, склад, период)
– себестоимость партии товара, приобретенной за указанный период

Оценим решение?

Плюсы:

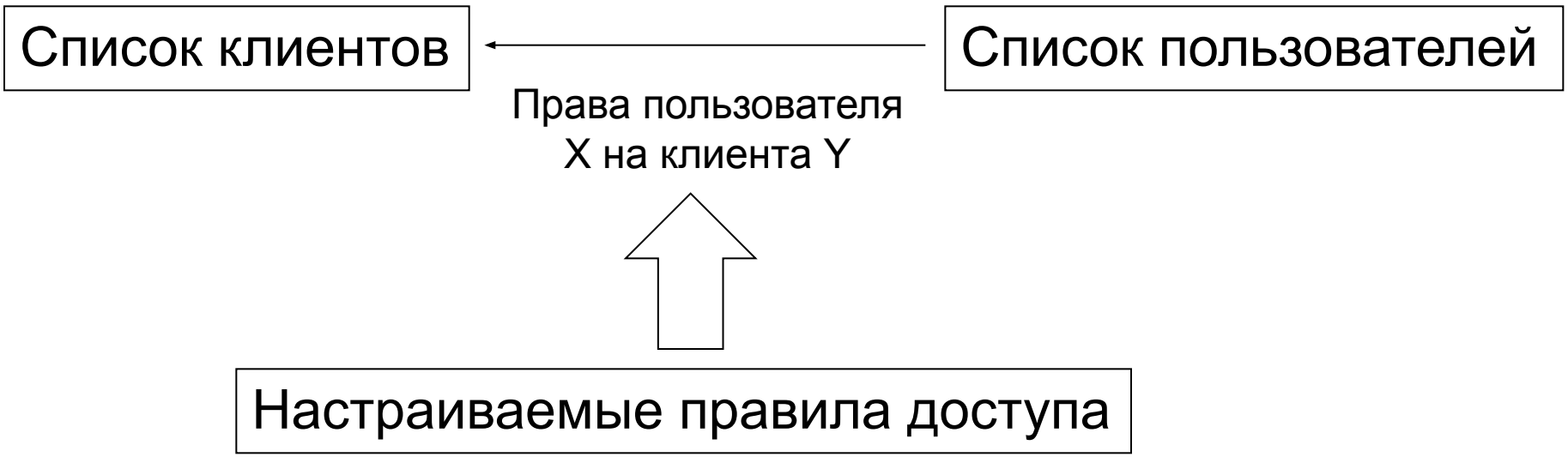
+ все работает очень быстро!
+ легко добавлять новые типы документов;
+ сторонние приложения могут добавлять/удалять документы, не заботясь о реестре складского учета;
+ для выполнения часто требуемых операций создан удобный набор хранимых процедур.

Оценим решение?

Минусы:

- данные о каждом перемещении хранятся в базе дважды;
- при создании этой системы пришлось много думать ☺

Когда нужны триггеры? Пример 2, права доступа и построение списка записей



Задача: Показать пользователю список клиентов с постраничной навигацией

Для этого надо:

1. Получить общее число клиентов,
доступных пользователю;

2. Выстроить их в определенном порядке;

3. Рассчитать номера клиентов,
которые окажутся на определенной странице.

Как реализуем? Вариант 1

Создать в PHP функцию
GetRights (user, record, module), которая:

А) Построит список правил, применимых в данном случае,

Б) По каждому правилу сформулирует условие
и проверит его выполнение,

В) Выберет минимальный результат из всех,
которые дают правила.

Оценим? Вариант 1

\+ Для определения доступа к одной записи – просто и надежно.

\- С большим списком записей будет работать очень медленно, даже если кэшировать результат шага A (список применимых правил).

Можем работать только со списками в сотни записей.

Как реализуем? Вариант 2

Кэшировать результат расчета прав в БД.

Пересчитывать фрагмент кэша из PHP каждый раз при изменении правил, свойств пользователя, свойств записи.

+ Выборка из кэша будет работать очень быстро.

- Кэш будет пересчитываться очень медленно, особенно после выполнения операций с группами записей.

Можем работать со списками в тысячи записей.

Как реализуем? Вариант 3

Создадим хранимую функцию, которая будет вычислять и возвращать права доступа для любой пары пользователь-клиент.

А) Текст функции генерируем из PHP
при изменении правил доступа,

Б) Функцию можно включить в SQL-запрос как условие:

```
SELECT * FROM clients WHERE GetAccess(clients.id,users.id)>0
```

Оценим? Вариант 3

\+ Быстрее, чем считать права в PHP.

\- Сложная генерация синтаксиса функции.
\- С большими списками все равно работает медленно.

Можем работать со списками в тысячи записей

Как реализуем? Вариант 4

Объединим варианты 2 и 3.

А) Реализуем специальную таблицу – кэш в БД.

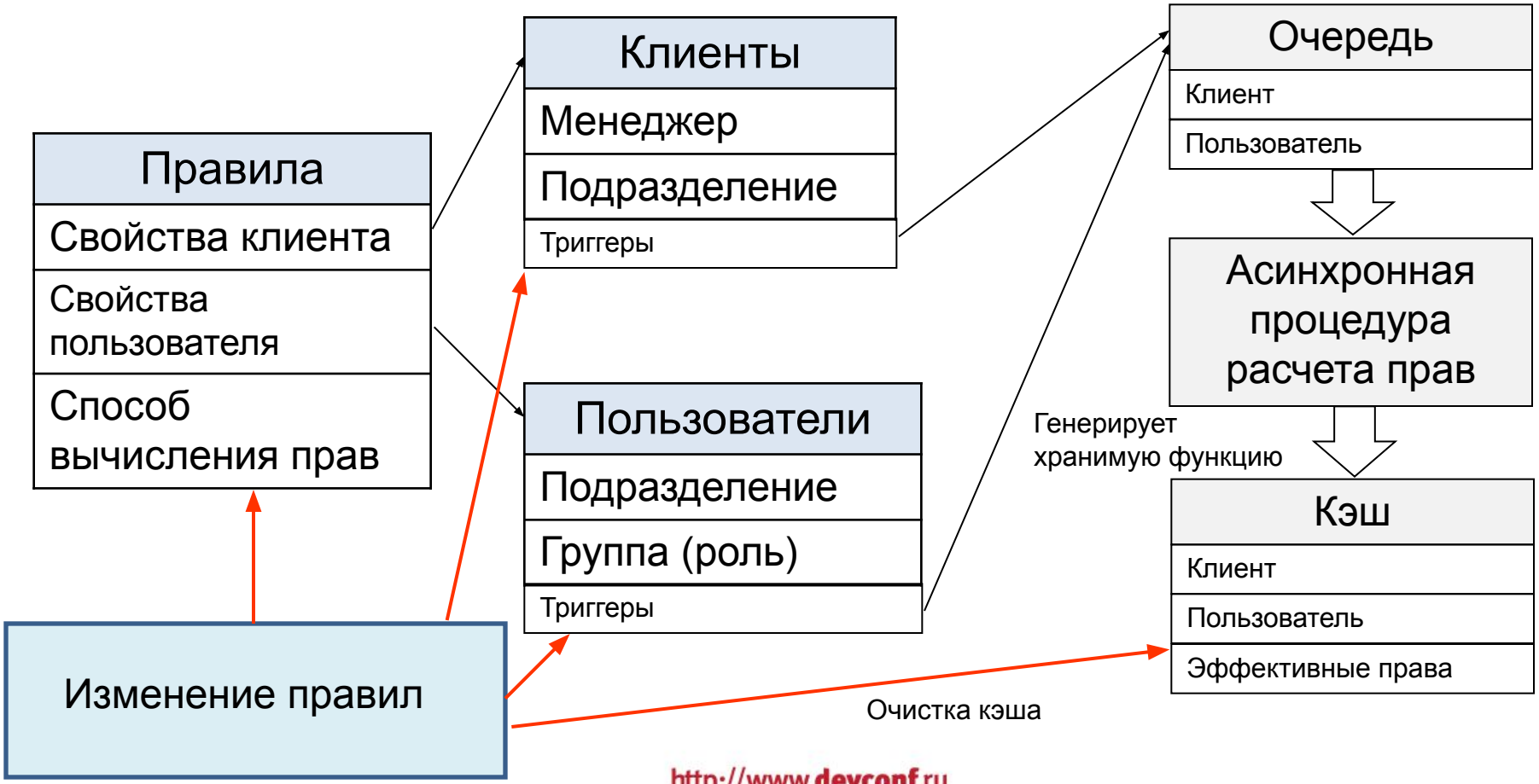
Б) Заполнять ее будем при помощи хранимой функции.

В) Вызывать пересчет кэша будем при помощи триггеров на таблицах «клиенты» и «пользователи».

Г) Триггеры перегенерируем при изменении правил. Триггеры ставят фрагменты кэша в очередь на пересчет при изменении свойств клиента и пользователя.

Д) Хранимая функция создается в момент пересчета.

Схема базы данных



Оценим? Вариант 4

\+ Очень быстро!

\- Сложная генерация синтаксиса функции.
\- Иногда права вычисляются не сразу.

Можем работать со списками в сотни тысяч записей.

Как реализуем? Вариант 5

Усовершенствуем механизм.

А) Не будем хранить в кэше нулевые значения.

Б) Не будем создавать кэш, если правила не зависят от свойств клиента и пользователя.

Г) Создадим удобные функции для работы с правами из PHP.

Оценим? Вариант 5

\+ Еще быстрее!

\- Усложнился PHP-код системы прав доступа.

Можем работать со списками в сотни тысяч записей.

Результаты

Обеспечена работоспособность списка записей, содержащего сотни тысяч значений.

Сохранена гибкость системы – администратор имеет возможность создавать любые правила доступа, зависящие от свойств клиента и пользователя.

Создан программный интерфейс, позволяющий максимально просто проверять права доступа как к набору записей, так и к отдельным клиентам, не задумываясь о физических механизмах реализации контроля прав.

Выводы!

1. При помощи MySQL можно решать сложные вычислительные задачи, возникающие при создании бизнес-приложений.

2. Создание массивов избыточных данных (кэшей) в базе способно увеличить скорость работы приложения в сотни или тысячи раз.

3. Наиболее естественный и удобный способ формирования кэшей в базе данных состоит в использовании триггеров и хранимых процедур.

Любите триггеры и хранимые процедуры! ☺

Спасибо за внимание!

Обсудить можно здесь:

http://serge-index.livejournal.com

Вопросы?