Мотивация

Кэширование необходимо для улучшения производительности, снижения задержек и уменьшения нагрузки на базу данных. Основные проблемы, которые оно решает:

1. **Задержки в загрузке первой страницы MES**: Ускорение доступа к frequently used данным, таким как список заказов.
2. **Высокая нагрузка на базу данных**: Уменьшение частоты обращений к базе данных для неизменяющихся или редко изменяющихся данных.

#### **Элементы системы для кеширования**

1. **Список заказов для операторов**: Часто запрашиваемая информация, особенно фильтры и статусные списки.
2. **Данные 3D-моделей**: Эти данные могут быть длительными в обработке и редко изменяются после загрузки.

Предлагаемое решение

#### **Выбор типа кеширования**

* **Серверное кеширование**: Выбрано из-за возможности более гибкого управления данными и разгрузки серверов баз данных. Оно позволяет централизовано управлять кешем и эффективно обслуживать все клиенты независимо от их характеристик.

### 

### **Паттерн кеширования**

#### **Cache-Aside**

* **Почему Cache-Aside**:
  + Позволяет контролировать, когда данные загружаются в кэш.
  + Хорошо подходит для данных, которые читаются часто, но изменяются редко.
  + Упрощает управление состоянием кеша вручную.
* **Почему не Write-Through или Refresh-Ahead**:
  + **Write-Through**: Подходит для частых изменений данных, что не является приоритетом. Внесение всех изменений сначала в кэш добавляет накладные расходы.
  + **Refresh-Ahead**: Создает излишнюю сложность и дополнительную нагрузку, не всегда давая экономию за счет проактивного обновления.

### **Операции чтения и записи**

#### **Чтение списка заказов**

1. **Запрос из приложения**: Идёт сначала в кэш.
2. **Если данные есть (хит)**: Возвращаем данные из кеша.
3. **Если данных нет (мисс)**: Запрос в базу данных → данные возвращаются клиенту и записываются в кэш.

#### **Запись об изменении статуса заказа**

1. **Изменение статуса**: Операция обновляется в базе данных.
2. **Инвалидация кеша**: Программная — удаление устаревших данных из кеша, обновление будет на следующем чтении.

### 

### **Стратегия инвалидации**

* **Программная инвалидация**: В случае изменения данных, таких как статус заказа, удобна, так как изменения контролируются серверной логикой, а не временем.
* **Почему не временная или по ключу**:
  + **Временная**: Может привести к устаревшим данным.
  + **По ключу**: Не всегда предсказуема при сложных связях данных.

### **Сравнительный анализ**

| **Критерий** | **Cache-Aside** | **Write-Through** | **Refresh-Ahead** |
| --- | --- | --- | --- |
| **Подходит для** | Редко изменяемых данных | Часто изменяемых данных | Периодически изменяемых |
| **Сложность** | Средняя | Высокая | Высокая |
| **Нагрузка на кеш** | Оптимальная | Большая | Большая |
| **Устаревание данных** | Контролируется вручную | Минимальная | Возможное |

### 

### 

### **Диаграмма**

1. **Чтение списка заказов:**
   * Оператор запрашивает через MES список заказов по статусам.
   * MES обращается через MES API к кешу.
   * При отсутствии данных в кеше, MES API запрашивает CRM, которая извлекает данные из базы. Полученные данные кешируются для последующих запросов.
2. **Изменение статуса заказа:**
   * Оператор обновляет статус через MES.
   * MES через MES API отправляет сообщение в очередь.
   * CRM обрабатывает сообщение, обновляя статус в базе данных.
3. **Пример изменения статуса:**
   * Переход заказов от "MANUFACTURING\_STARTED" к "MANUFACTURING\_COMPLETED" и уведомление об этом через MES API и очередь.

Эта модель иллюстрирует использование кеша через MES API и взаимодействие через очереди для обновления статусов.

