ТЗ: Разработка агента синхронизации данных (Sync Agent)

ТЗ: Разработка агента синхронизации данных (Sync Agent)

Источник файл Ожерельев В.А. ТЗ: Разработка агента синхронизации.md

Дата: 03.05.2026

---

1. Назначение

Агент синхронизации обеспечивает загрузку и актуализацию данных из Grace CRM в:

- PostgreSQL (операционный слой)
- OpenSearch (поисковый слой)

Система должна поддерживать:

- near real-time обновление (webhooks)
- инкрементальную синхронизацию
- полную переиндексацию

---

2. Состав синхронизируемых сущностей

Обязательные сущности:

Сущность	Описание
Projects	Сделки
Clients	Клиенты
Objects	Объекты клиента (ключевая связь клиент -> актив)
Activities	Активности
Tasks	Задачи
Comments	Комментарии
Calculations	Расчёты / КП
Orders	Заказы
Users	Пользователи

---

3. API интеграция с CRM

Основные endpoints

```
GET /api/projects
GET /api/projects/{id}
GET /api/projects?updated_since=timestamp

GET /api/activities
GET /api/activities?project_id=
GET /api/activities?updated_since=timestamp

GET /api/clients
GET /api/clients/{id}

GET /api/tasks
GET /api/tasks?project_id=

GET /api/comments
GET /api/comments?entity_type=&entity_id=

GET /api/calculations
```

```
GET /api/calculations?project_id=

GET /api/orders
GET /api/orders?project_id=

GET /api/users
```

3.1 Объекты (обязательное расширение CRM)

⚠ В рамках проекта требуется наличие сущности:

Objects (объекты клиента)

Если отсутствует -- требуется добавить API:

```
GET /api/objects
GET /api/objects/{id}
GET /api/objects?client_id=
GET /api/objects?updated_since=timestamp
```

Назначение:

- связь **Client -> Object -> Project**
- понимание контекста сделки

Пример:

- клиент: застройщик
- объект: ЖК "Северный"
- проекты: сделки по этому ЖК

---

4. Архитектура Sync Agent

Компоненты:

- API Connector
- Sync Scheduler
- Webhook Listener
- Data Transformer
- Indexer (OpenSearch)
- Storage Writer (PostgreSQL)

---

5. Mapping схемы OpenSearch (JSON)

5.1 crm-projects

```
{
  "mappings": {
    "properties": {
      "project_id": {"type": "keyword"},
      "client_id": {"type": "keyword"},
      "object_id": {"type": "keyword"},
      "name": {"type": "text"},
      "status": {"type": "keyword"},
      "stage": {"type": "keyword"},
      "amount": {"type": "double"},
      "manager_id": {"type": "keyword"},
      "created_at": {"type": "date"},
      "updated_at": {"type": "date"},
      "last_activity_at": {"type": "date"},
      "activities_count": {"type": "integer"}
    }
  }
}
```

5.2 crm-objects (новый индекс)

```
{
  "mappings": {
    "properties": {
      "object_id": {"type": "keyword"},
      "client_id": {"type": "keyword"},
      "name": {"type": "text"},
      "type": {"type": "keyword"},
      "status": {"type": "keyword"},
      "location": {"type": "text"},
      "created_at": {"type": "date"},
      "updated_at": {"type": "date"}
    }
  }
}
```

5.3 crm-clients

```
{
  "mappings": {
    "properties": {
      "client_id": {"type": "keyword"},
      "name": {"type": "text"},
      "industry": {"type": "keyword"},
      "segment": {"type": "keyword"},
      "created_at": {"type": "date"},
      "updated_at": {"type": "date"}
    }
  }
}
```

5.4 crm-activities

```
{
  "mappings": {
    "properties": {
      "activity_id": {"type": "keyword"},
      "project_id": {"type": "keyword"},
      "type": {"type": "keyword"},
      "description": {"type": "text"},
      "result": {"type": "text"},
      "created_at": {"type": "date"}
    }
  }
}
```

5.5 crm-tasks

```
{
  "mappings": {
    "properties": {
      "task_id": {"type": "keyword"},
      "project_id": {"type": "keyword"},
      "assignee_id": {"type": "keyword"},
      "status": {"type": "keyword"},
      "due_date": {"type": "date"}
    }
  }
}
```

5.6 crm-comments

```
{
  "mappings": {
```

```
    "properties": {
      "comment_id": {"type": "keyword"},
      "entity_type": {"type": "keyword"},
      "entity_id": {"type": "keyword"},
      "text": {"type": "text"},
      "author_id": {"type": "keyword"},
      "created_at": {"type": "date"}
    }
  }
}
```

```
{
  "mappings": {
    "properties": {
      "calculation_id": {"type": "keyword"},
      "project_id": {"type": "keyword"},
      "amount": {"type": "double"},
      "version": {"type": "integer"},
      "created_at": {"type": "date"}
    }
  }
}
```

5.8 crm-orders

```
{
  "mappings": {
    "properties": {
      "order_id": {"type": "keyword"},
      "project_id": {"type": "keyword"},
      "amount": {"type": "double"},
      "status": {"type": "keyword"},
      "created_at": {"type": "date"}
    }
  }
}
```

---

6. Распределение данных между PostgreSQL и OpenSearch

Данный раздел определяет, **какие данные и в каком виде сохраняются** в:

- PostgreSQL (операционный слой)
- OpenSearch (витрины / search layer)

Ключевой принцип:

- PostgreSQL -> нормализованные данные и состояние системы
- OpenSearch -> денормализованные витрины для поиска и аналитики

6.0 Диаграмма последовательности обмена данными

```
sequenceDiagram
    participant CRM
    participant Webhook
    participant SyncAgent
    participant PostgreSQL
    participant OpenSearch

    CRM->>Webhook: project-updated
    Webhook->>SyncAgent: событие

    SyncAgent->>CRM: GET /projects/{id}
    CRM-->>SyncAgent: project data

    SyncAgent->>SyncAgent: transform + normalize
```

```
    SyncAgent->>PostgreSQL: upsert
    SyncAgent->>OpenSearch: bulk index

    Note over SyncAgent: update last_sync_timestamp
```

6.1 Таблица распределения данных

Сущность	PostgreSQL (что сохраняется)	OpenSearch (что индексируется)
Projects (сделки)	Полная структура проекта (raw JSON + нормализованные поля), статус синхронизации, технические поля (sync_state, timestamps)	Денормализованная витрина: project_id, client_id, object_id, стадия, статус, сумма, менеджер, last_activity_at, activities_count
Clients (клиенты)	Полная карточка клиента, связи, служебные поля	Упрощённая витрина: client_id, название, отрасль, сегмент
Objects (объекты)	Полная модель объекта, связь с клиентом	Витрина: object_id, client_id, тип, статус, локация
Activities (активности)	Полные записи активностей (raw), связь с проектом	Индекс коммуникаций: тип, текст, результат, дата
Tasks (задачи)	Полная структура задач, статусы, SLA	Витрина: task_id, project_id, исполнитель, статус, due_date
Comments (комментарии)	Полные тексты, связь с entity	Индекс текстов для поиска: entity_type, entity_id, текст
Calculations (расчёты)	Полные расчёты, версии, параметры	Витрина: сумма, версия, привязка к проекту
Orders (заказы)	Полная структура заказов	Витрина: статус, сумма, дата
Users (пользователи)	Полная модель пользователей и ролей	Витрина: user_id, роль, команда
Sync metadata	offset, updated_since, last_sync_time, retry state	не индексируется
Ошибки / логи sync	Полный лог операций	не индексируется

6.2 Принципы формирования витрин OpenSearch

1. Денормализация

В OpenSearch данные агрегируются:

Пример:

- project + client + object -> единый документ
- activities -> агрегаты (count, last_activity_date)

---

2. Обогащение

При индексации добавляются:

- derived fields
- агрегаты
- вычисленные признаки

Пример:

- `activities_count`
- `days_without_activity`
- `is_overdue`

---

3. Оптимизация под сценарии поиска

OpenSearch индекс строится под:

- фильтрацию
- агрегации
- полнотекстовый поиск

НЕ под:

- транзакции
- сложные связи

6.3 Поток записи данных

```
CRM → SyncAgent → PostgreSQL (raw + normalized)
                          ↓
                    Transformer
                          ↓
                   OpenSearch (витрины)
```

6.4 Ключевые различия слоёв

Критерий	PostgreSQL	OpenSearch
Тип данных	нормализованные	денормализованные
Назначение	состояние системы	поиск и аналитика
Обновление	строгое (upsert)	bulk indexing
Источник истины	да (для AI слоя)	нет
Использование	Sync, Orchestrator	Recommendation, Analytics

6.5 Важное архитектурное требование

- OpenSearch **не должен использоваться как источник истины**
- Все изменения идут через:
  - CRM -> SyncAgent -> PostgreSQL -> OpenSearch
- Разделение позволяет:
- избежать перегрузки PostgreSQL аналитикой
- обеспечить быстрый поиск и рекомендации
- гарантировать консистентность системы

7. Модель синхронизации: когда, что и как обновляется

Синхронизация реализуется через **3 параллельных механизма**, каждый из которых решает свою задачу:

- Webhooks -> реакция в реальном времени
- Polling -> гарантированная консистентность
- Full Sync -> восстановление и выравнивание

7.1 Webhooks (near real-time синхронизация)

Назначение

Мгновенная реакция на изменения в CRM.

Когда срабатывает

При событиях в CRM:

- создание проекта
- обновление проекта
- создание активности
- (опционально) изменение задачи

Поток

```
CRM → webhook → SyncAgent → точечный fetch → update
```

Какие данные синхронизируются

Сущность	Что делаем
Projects	загружаем 1 проект по ID
Activities	загружаем активность или проект целиком
Tasks	загружаем задачу
Comments	при наличии webhook
Calculations	при изменении проекта

Особенность

Webhook **не доверяем полностью** -> всегда делаем **дочитывание через API**

SLA

- задержка: **1–5 секунд**
- режим: near real-time

7.2 Polling (инкрементальная синхронизация)

Назначение

Гарантия, что:

- ничего не потерялось
- данные актуальны

Механика

Используется:

```
GET /api/*?updated_since=timestamp
```

Частота по сущностям

Сущность	Частота	Причина
Projects	каждые 5 мин	ключевая сущность
Activities	каждые 5 мин	динамика общения
Tasks	5–10 мин	SLA
Comments	5–10 мин	контекст
Objects	10–15 мин	реже меняются
Clients	15–30 мин	редко меняются
Calculations	10–15 мин	средняя динамика
Orders	10–15 мин	финансовые события
Users	1 раз/час	почти статично

Поток

```
Scheduler → SyncAgent → API (bulk) → batch processing → upsert → index
```

Особенности реализации

- используется `updated_at`
- хранится `last_sync_timestamp`
- обрабатываются батчи (100–1000 записей)

SLA

- задержка актуализации: **до 5–10 минут**

7.3 Full Sync (полная синхронизация)

Назначение

- восстановление консистентности
- устранение ошибок sync
- перерасчёт витрин

Когда выполняется

- 1 раз в сутки (ночью)
- вручную (по триггеру)

Что синхронизируется

Сущность	Поведение
Все	полная выгрузка
Projects	пересборка агрегатов
Activities	полная история
Objects	восстановление связей
Clients	обновление справочника

Поток

```
Full load → overwrite / reindex → rebuild OpenSearch
```

SLA

- допускается длительность: **до нескольких часов**
- выполняется вне рабочего времени

7.4 Приоритет механизмов

Механизм	Приоритет	Роль
Webhooks	высокий	скорость
Polling	средний	надёжность
Full Sync	низкий	восстановление

7.5 Конфликты и консистентность

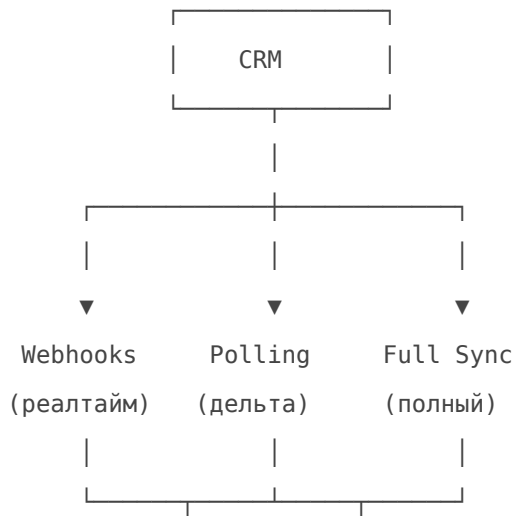
Правило:

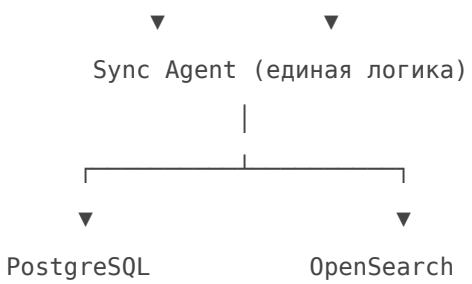
```
последнее изменение по updated_at побеждает
```

Дополнительно:

- idempotent операции
- защита от дублей
- retry (3+ попытки)

7.6 Сводная схема





7.7 Ключевой принцип (важно для разработки)

Нельзя полагаться только на один механизм:

- Webhooks -> могут теряться
- Polling -> не realtime
- Full Sync -> тяжёлый

➡ Только вместе они дают:

- **актуальность**
- **надёжность**
- **консистентность**

Итог

Тип данных	Как обновляется
Критичные (проекты, активности)	webhook + polling
Средние (задачи, расчёты)	polling + webhook (если есть)
Справочники (клиенты, users)	редкий polling
Связи (objects)	polling + full sync

---

8. Требования к реализации

Обязательные:

- idempotency (повторяемость без дубликатов)
- поддержка bulk операций
- retry (3 попытки минимум)
- логирование ошибок
- контроль `updated_at`

Производительность:

- batch загрузка (100–1000 записей)
- bulk indexing OpenSearch

---

9. Ключевые особенности

- поддержка связи:
  - Client -> Object -> Project
- денормализация данных в OpenSearch
- готовность к аналитике и AI

---

10. Результат

После реализации:

- все CRM-данные доступны в OpenSearch
- данные связаны через Object layer
- AI-агенты получают полноценный контекст

Revision #1
Created 2026-06-22 03:22:48 UTC by Claude Ops
Updated 2026-06-22 03:22:48 UTC by Claude Ops