ORES
Научно-практический электронный журнал
Оригинальные исследования (ОРИС), *ISSN 2222-0402*
www.ores.su
original-research.ru

УДК 004

ИНТЕГРАЦИЯ CLICKHOUSE В SPRING FRAMEWORK

Ладыгин Сергей Алексеевич,
магистрант, кафедра ИиППО, Институт информационных технологий, РТУ МИРЭА, Москва.
Email: s.a.ladygin@yandex.ru

Плотников Сергей Борисович,
канд. техн. наук, доцент, кафедра ИиППО, Институт информационных технологий, РТУ МИРЭА, Москва.
Email: plotnikovsb@mail.ru

Аннотация

В статье рассматриваются задачи интеграции колоночной СУБД ClickHouse в экосистему Spring Framework. Проведен анализ существующих подходов взаимодействия с ClickHouse в Java-среде, таких как использование JDBC и официального асинхронного клиента. Обоснована необходимость создания абстракции, аналогичной Spring Data JPA. Представлен разработанный программный модуль, реализующий интеграцию ClickHouse с использованием аннотаций, автоконфигурации и декларативной модели работы с данными. Приведены примеры использования. Описаны перспективы дальнейших исследований.

Ключевые слова: ClickHouse, Spring, интеграция, БД, база данных, Java

INTEGRATION OF CLICKHOUSE INTO SPRING FRAMEWORK

Ladygin Sergei Alekseebvich,
Master's student, Department of Intelligent and Industrial Software, Institute of Information Technologies, RTU MIREA, Moscow.
Email: s.a.ladygin@yandex.ru

Sergey Borisovich Plotnikov,
Ph.D. in Engineering, Associate Professor, Department of Intelligent and Industrial Software, Institute of Information Technologies, RTU MIREA, Moscow.
Email: plotnikovsb@mail.ru

ABSTRACT

The article explores the challenges of integrating the column-oriented DBMS ClickHouse into the Spring Framework ecosystem. It analyzes existing approaches in Java, such as the JDBC driver and official async client. The necessity of creating an abstraction similar to Spring Data JPA

is substantiated. A software module is presented using annotations, auto-configuration, and a declarative data access model. Usage examples are shown, and future directions are discussed.

Keywords: ClickHouse, Spring, integration, DB, data base, Java

1 Актуальность

Современные распределённые системы обработки и анализа данных предъявляют всё более жёсткие требования к производительности, масштабируемости и гибкости архитектурных решений. С ростом объёмов данных, необходимость в быстрых OLAP-запросах становится критической для бизнеса, ориентированного на аналитику в реальном времени. ClickHouse, как одна из лидирующих колоночных СУБД, демонстрирует выдающиеся результаты в этой области благодаря архитектуре, оптимизированной под чтение больших массивов данных [3]. Однако, несмотря на технологическую зрелость ClickHouse, её интеграция в экосистему Java, и, в частности, Spring Framework, всё ещё вызывает затруднения у разработчиков, особенно привыкших к удобствам, предоставляемым Spring Data JPA. В отличие от классических реляционных СУБД, таких как PostgreSQL или MySQL, где Spring Data обеспечивает декларативный, типобезопасный и расширяемый подход к работе с базой данных [2], при взаимодействии с ClickHouse приходится прибегать к низкоуровневым средствам. Это увеличивает порог вхождения, усложняет сопровождение проектов и снижает скорость разработки.

Анализ существующих решений

На сегодняшний день в Java-среде для взаимодействия с ClickHouse используются в основном два подхода: через JDBC-драйвер или через официальный асинхронный клиент ClickHouse Client. Каждый из этих вариантов имеет как достоинства, так и ограничения, обусловленные архитектурными особенностями и целевыми сценариями применения.

ClickHouse JDBC:

Этот драйвер реализует стандартный интерфейс JDBC, что позволяет использовать ClickHouse как обычную реляционную СУБД с большинством Java-инструментов, поддерживающих работу с JDBC. Он обеспечивает базовый уровень совместимости с JDBC-ориентированными библиотеками и инструментами. Преимуществом такого подхода является простота подключения и минимальные требования к изменению существующего кода.

Тем не менее, JDBC-драйвер ClickHouse не предоставляет никаких средств интеграции со Spring или Spring Boot: отсутствует поддержка автоконфигурации, нет аннотаций или механизмов для генерации репозиториев, не реализован подход декларативного доступа к данным. Кроме того, архитектура JDBC ориентирована на синхронную модель, что снижает масштабируемость в условиях интенсивной аналитической нагрузки. Использование Spring Data с ClickHouse JDBC требует значительных усилий по написанию обёрток, а также не позволяет реализовать все преимущества инфраструктуры Spring Data, такие как автоматическая генерация репозиториев или семантический анализ сигнатур методов.

ClickHouse Client:

Официальный Java-клиент от разработчиков ClickHouse предлагает современный подход к работе с СУБД, предоставляя асинхронный, неблокирующий API, поддержку соединений через HTTP и TCP, настройку пулов и продвинутую работу с типами [1]. Он позволяет получить существенную производительность при прямом программировании.

Однако, в отличие от JDBC, ClickHouse Java-клиент не интегрируется напрямую с Spring Boot. Нет механизма автоконфигурации, отсутствуют аннотации или интерфейсы

для декларативной работы с сущностями, требуется вручную управлять жизненным циклом клиента и писать шаблонный код. Всё это делает применение ClickHouse Client в Spring-приложениях менее удобным и требует дополнительной обёртки.

Обоснование выбора направления

Модель Spring Data JPA на протяжении многих лет демонстрирует высокую эффективность при построении корпоративных приложений. Её достоинства заключаются в следующем:

Автоматизация работы с репозиториями: не требуется вручную реализовывать CRUD-интерфейсы.

Семантический анализ сигнатур методов: findByEmailAndStatus автоматически трансформируется в SQL-запрос без необходимости явно его описывать [5].

Гибкость конфигурации: через аннотации и свойства можно централизованно управлять источниками данных, профилями и миграциями.

Исходя из этих преимуществ, была определена задача – перенести аналогичный уровень удобства и абстракции на работу с ClickHouse, минимизировав различия в подходах при использовании этой СУБД в Spring-приложениях.

Примеры использования разработанного программного модуля

Разработанная библиотека представляет собой обёртку над ClickHouse Client, интегрированную с Spring Framework и реализующую подход, аналогичный Spring Data JPA. Рассмотрим примеры использования, демонстрирующие основные преимущества данного решения.

Подключение и настройка

Для начала работы с модулем требуется добавить необходимую зависимость io.clickhouse.springdata:clickhouse-spring-boot-starter:1.1.0 в сборщик проектов (например, Maven или Gradle), а также указать соответствующие переменные среды для подключения к ClickHouse в файле конфигурации application.yaml или application.properties [4].

Пример конфигурационного файла application.yaml приведён в листинге 1.

Листинг 1 – Пример файла application.yaml

```
spring:
  clickhouse-data:
    entity-packages: "com.example.table"
    client:
      # Общие настройки подключения
      endpoint: http://localhost:8123
      username: "default"
      password: ""
```

Данный листинг демонстрирует простоту настройки подключения к БД. После указания необходимой конфигурации можно использовать ClickHouse Client, внедрив его через аннотацию @Autowired. Таблицы, помеченные аннотацией @Table, автоматически регистрируются в клиенте, и не требуют предварительной ручной регистрации.

Пример класса сущности и использования клиента представлен в листинге 2.

Листинг 2 – Пример использования аннотации @Table и ClickHouse Client

```
@Getter
@Setter
@EqualsAndHashCode
@ToString
@Table(name = "person")
@AllArgsConstructor
```

Продолжение листинга 2.

```
@NoArgsConstructor
public class PersonEntity {
   private Long id;
   private String firstName;
   private String lastName;
}

@Component
@Slf4j
@RequiredArgsConstructor
public class PersonService {

   @Autowired
   private Client client;

   public void doSomething() {
      List<PersonEntity> persons = client.queryAll(
            "select * from person",
            PersonEntity.class,
            client.getTableSchema("person")
      );
   }
}
```

Благодаря такому подходу больше не требуется вручную создавать или конфигурировать объект клиента, так же как не нужно вручную регистрировать классы сущностей в клиенте. Результаты выполнения запросов автоматически отображаются на экземпляры сущностей.

Использование репозиториев

Разработанная библиотека также поддерживает работу с репозиториями, аналогичными Spring Data JPA. Для использования этого подхода необходимо пометить основной класс приложения аннотацией @EnableClickHouseRepositories, а затем унаследовать интерфейс сущности от интерфейса-маркера ClickHouseRepositoryMarker или готового стандартного интерфейса ClickHouseRepository. Библиотека автоматически создаст прокси-объекты этих репозиториев, преобразуя именованные методы в SQL-запросы. Пример использования репозиториев представлен в листинге 3.

Листинг 3 – Пример использования функционала репозиториев

```
public interface PersonRepository extends ClickHouseRepository<PersonEntity, Long> {
   List<PersonEntity> findByFirstName(String firstName);
}

public interface ClickHouseRepository<T, ID> extends ClickHouseRepositoryMarker {
   List<T> findAll();
   long count();
   <S extends T> List<S> saveAll(Iterable<S> entities);
}

@Component
@Slf4j
@RequiredArgsConstructor
public class PersonService {
```

```
    private final PersonRepository personRepository;
    public void execute() throws ExecutionException, InterruptedException {
        List<PersonEntity> findByNameResult = personRepository.findByFirstName("FirstName");
        List<PersonEntity> findAllResult = personRepository.findAll();
        List<PersonEntity> saveAllResult = personRepository.saveAll(insertFakeData());
    }
}
```

Представленные выше примеры демонстрируют, как легко и удобно можно интегрировать ClickHouse с инфраструктурой Spring, сохранив удобство и декларативность, присущие подходу Spring Data JPA.

Перспективы развития и направления дальнейших исследований

Завершение разработки библиотеки позволит реализовать следующие задачи, ориентированные как на повышение удобства, так и на расширение функциональных возможностей:

Конвертация типов: создание расширяемого механизма для преобразования ClickHouse-специфичных типов (например, Nullable(DateTime64), Array(String)) в соответствующие типы Java. В перспективе — поддержка пользовательских конвертеров.

Batch-запросы: реализация эффективной вставки и обработки больших объёмов данных в пакетном режиме с учётом специфики ClickHouse (оптимизация по чанкам, контроль ограничения по размеру блока).

DSL для построения запросов: внедрение внутреннего языка построения запросов, позволяющего задавать условия фильтрации, сортировки, агрегации и объединений в виде цепочек вызовов (fluent API).

Интеграция с инфраструктурой Spring: внедрение метрик (Micrometer), логирования (Slf4j, Logback), поддержки Spring AOP для внедрения кросс-секционных аспектов, таких как трейсинг или аудит.

Инструменты тестирования: разработка mock-инфраструктуры для ClickHouse, позволяющей писать unit- и integration-тесты без подключения к живой БД.

Заключение

Разработанный модуль представляет собой важный шаг к унификации взаимодействия Spring Framework с ClickHouse, позволяющий преодолеть разрыв между низкоуровневым API ClickHouse Client и высокоуровневой декларативной моделью доступа к данным, характерной для Spring Data. За счёт автоконфигурации, генерации прокси-репозиториев и декларативного синтаксиса, библиотека упрощает разработку, снижает количество шаблонного кода и делает переход к ClickHouse менее затратным. Таким образом, представленное решение не только повышает производительность труда разработчиков, но и способствует стандартизации подходов при построении микросервисных архитектур на базе ClickHouse и Spring.

Список литературы:

1. ClickHouse. Официальная документация ClickHouse Client [Электронный ресурс]. – URL: https://clickhouse.com/docs/integrations/language-clients/java/client (дата обращения: 12.03.2025).
2. Spring Data JPA. Документация проекта Spring [Электронный ресурс]. – URL: https://spring.io/projects/spring-data-jpa (дата обращения: 15.03.2025).
3. Афанасьев Г.И., Белоногов И.Б., Булатова И.Г. Сравнение «строковой» СУБД PostgreSQL и «столбцовой» СУБД ClickHouse на наборе статистических временных данных в рамках OLAP // Аллея Науки. – 2018. – №1(17). – С. 10–15.

4. Spring Boot Auto-Configuration Reference [Электронный ресурс]. – URL: https://docs.spring.io/spring-boot/docs/current/reference/html/auto-configuration.html (дата обращения: 12.04.2025).
5. Еровлева Р.В., Еровлев П.А. Написание динамических запросов с помощью Spring Data JPA // Постулат. – 2021. – №12. – ISSN 2414-4487.

References:

1. ClickHouse. Official documentation for ClickHouse Client [online]. Available at: https://clickhouse.com/docs/integrations/language-clients/java/client (accessed: March 12, 2025).
2. Spring Data JPA. Official documentation of the Spring project [online]. Available at: https://spring.io/projects/spring-data-jpa (accessed: March 15, 2025).
3. Afanasyev G.I., Belonogov I.B., Bulatova I.G. Comparison of row-oriented DBMS PostgreSQL and column-oriented DBMS ClickHouse on a set of statistical time data within OLAP // Alley of Science. – 2018. – No. 1(17). – P. 10–15.
4. Spring Boot Auto-Configuration Reference [online]. Available at: https://docs.spring.io/spring-boot/docs/current/reference/html/auto-configuration.html (accessed: April 12, 2025).
5. Erovleva R.V., Erovlev P.A. Writing dynamic queries with Spring Data JPA // Postulat. – 2021. – No. 12. – ISSN 2414-4487.