从MySQL迁移到PostgreSQL经验总结

Java分布式架构实战 2025-01-16 1,397 阅读7分钟

‹‹‹ TRAE 2.0 SOLO 出道，一键贯通从灵感火花到上线部署的全程协作 ›››

背景

最近一两周在做 从MySQL迁移到PostgreSQL 的任务(新项目，历史包袱较小，所以迁移比较顺利), 感觉还是有一些知识，可以拿出来分享，希望对大家有所帮助。

为什么要转到PostgreSQL

因架构团队安全组安全需求，需要将Mysql迁移到PostgreSQL。实际迁移下来，发现PostgreSQL挺优秀的，比MySQL严谨很多，很不错。

迁移经验

1. **引入PostgreSQL驱动，调整链接字符串**
2. **pagehelper方言调整**
3. **涉及order, group，name, status, type 等关键字，要用 引号 括起来**
4. **JSON字段及JsonTypeHandler**

项目中用到了比较多的JSON字段。在mysql中，也有JSON字段类型，但是有时候我们用了varchar或text，在mybatis typehandler中是当成字符来处理的。但是在postgresql中，相对严谨，如果字段类型是json,那么在java中会被封装为PGObject，所以我们原来的JsonTypeHandler就要被改造。

typescript 体验AI代码助手 代码解读 复制代码

```typescript
/**
 * JSON类型处理器
 *
 * @author james.h.fu
 * @create 2024/10/9 20:45
 */
@Slf4j
public class JsonTypeHandler<T> extends BaseTypeHandler<T> {
    private static final ObjectMapper mapper = new ObjectMapper();

    static {
        mapper.configure(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES, Boolean.FALSE);
        mapper.setSerializationInclusion(JsonInclude.Include.NON_NULL);
    }

    private final Class<T> clazz;
    private TypeReference<? extends T> typeReference;

    public JsonTypeHandler(Class<T> clazz) {
        if (clazz == null) throw new IllegalArgumentException("Type argument cannot be null");
        this.clazz = clazz;
    }

    @Override
    public void setNonNullParameter(PreparedStatement ps, int i, T parameter, JdbcType jdbcType) th
        setObject(ps, i, parameter);
    }

    @Override
    public T getNullableResult(ResultSet rs, String columnName) throws SQLException {
        return toObject(rs, columnName);
    }

    @Override
    public T getNullableResult(ResultSet rs, int columnIndex) throws SQLException {
```

Java分布式架构... LV.3
Java Senior Engineer @某垂...
25 文章 32k 阅读 55 粉丝
关注 私信

目录 收起

相关推荐
11个提高生产力的Go小技巧
2.0k阅读 · 27点赞
K8s太繁琐？快试试开源轻量级的 Kube...
861阅读 · 4点赞
再见 PowerDesigner！这款国人开源的...
604阅读 · 0点赞
Apache Dubbo 3.3 全新发布：Triple X ...
1.3k阅读 · 14点赞
史上最全docker-compose.yaml语法规则
2.5k阅读 · 39点赞

精选内容
ThreadLocal在什么情况下会导OOM?
程序员飞鱼 · 277阅读 · 27点赞
Bun技术评估 - 25 Utils(实用工具)
JohnYan · 39阅读 · 0点赞
深度理解 volatile 与 synchronized：并...
回家路上绕了弯 · 56阅读 · 1点赞
AutoMQ × CloudCanal：打造秒级响应...
AutoMQ · 21阅读 · 0点赞
基于领域事件驱动的微服务架构设计与...
就是帅我不改 · 52阅读 · 0点赞

```
36          return toObject(rs, columnIndex);
37      }
38
39      @Override
40      public T getNullableResult(CallableStatement cs, int columnIndex) throws SQLException {
41          return toObject(cs, columnIndex);
42      }
43
44      protected TypeReference<? extends T> getTypeReference() {
45          return new TypeReference<T>() {};
46      }
47
48      private String toJson(T object) {
49          try {
50              return mapper.writeValueAsString(object);
51          } catch (Exception ex) {
52              log.error("JsonTypeHandler error on toJson content:{}", JsonUtil.toJson(object), ex);
53              throw new RuntimeException("JsonTypeHandler error on toJson", ex);
54          }
55      }
56
57      private T toObject(String content) {
58          if (!StringUtils.hasText(content)) {
59              return null;
60          }
61          try {
62              if (clazz.getName().equals("java.util.List")) {
63                  if (Objects.isNull(typeReference)) {
64                      typeReference = getTypeReference();
65                  }
66                  return (T) mapper.readValue(content, typeReference);
67              }
68              return mapper.readValue(content, clazz);
69          } catch (Exception ex) {
70              log.error("JsonTypeHandler error on toObject content:{},class:{}", content, clazz.getNa
71              throw new RuntimeException("JsonTypeHandler error on toObject", ex);
72          }
73      }
74
75  //    protected boolean isPostgre() {
76  //        SqlSessionFactory sqlSessionFactory = SpringUtil.getBean(SqlSessionFactory.class);
77  //        Configuration conf = sqlSessionFactory.getConfiguration();
78  //        DataSource dataSource = conf.getEnvironment().getDataSource();
79  //        try (Connection connection = dataSource.getConnection()) {
80  //            String url = connection.getMetaData().getURL();
81  //            return url.contains("postgresql");
82  //        } catch (SQLException e) {
83  //            throw new RuntimeException("Failed to determine database type", e);
84  //        }
85  //    }
86
87      @SneakyThrows
88      private void setObject(PreparedStatement ps, int i, T parameter) {
89          PGobject jsonObject = new PGobject();
90          jsonObject.setType("json");
91          jsonObject.setValue(JsonUtil.toJson(parameter));
92          ps.setObject(i, jsonObject);
93      }
94
95      @SneakyThrows
96      private T toObject(ResultSet rs, String columnName) {
97          Object object = rs.getObject(columnName);
98          return toObject(object);
99      }
100
101     @SneakyThrows
102     private T toObject(ResultSet rs, int columnIndex) {
103         Object object = rs.getObject(columnIndex);
104         return toObject(object);
105     }
106
107     @SneakyThrows
108     private T toObject(CallableStatement rs, int columnIndex) {
109         Object object = rs.getObject(columnIndex);
110         return toObject(object);
111     }
112
113     public T toObject(Object object) {
114         if (object instanceof String json) {
115             return this.toObject(json);
116         }
117         if (object instanceof PGobject pgObject) {
118             String json = pgObject.getValue();
119             return this.toObject(json);
120         }
121
```

```
122         return null;
123     }
124 }
125 
126 <resultMap>
127     <result column="router_info" jdbcType="OTHER" property="routerInfo" typeHandler="***.cms.cmslib
128 </resultMap>
129 
130 <set>
131 <if test="routerInfo != null">
132   router_info = #{routerInfo,typeHandler=***.cms.cmslib.mybatis.JsonTypeHandler}
133  </if>
134 </set>
135 where id = #{id}
```

如果JSON中存储是的List<T>, Map<String, Object>， Set<T>等类型时, 会存在 泛型类型中类型擦除的问题 。
因此，如果存在这种情况，我们需要扩展子类，在子类中提供详细的类型信息 TypeReference 。

scala 体验AI代码助手 代码解读 复制代码

```scala
/**
 * @author james.h.fu
 * @create 2024/12/9 20:45
 */
public class ComponentUpdateListJsonTypeHandler extends JsonTypeHandler<List<ComponentUpdate>> {
    public ComponentUpdateListJsonTypeHandler(Class<List<ComponentUpdate>> clazz) {
        super(clazz);
    }

    @Override
    protected TypeReference getTypeReference() {
        return new TypeReference<List<ComponentUpdate>>() {
        };
    }
}
```

4. pgsql不支持mysql insert ignore语法，pgsql提供了类似的语法:

vbnet 体验AI代码助手 代码解读 复制代码

```vbnet
INSERT INTO orders (product_id, user_id)

VALUES (101, 202)

ON CONFLICT (product_id, user_id) DO NOTHING;
```

但是与mysql insert ignore并不完全等价，关于这个点如何改造，需要结合场景或者业务逻辑来斟酌定夺.

5. pgsql也不支持INSERT ... ON DUPLICATE KEY UPDATE， 如果代码中有使用这个语法，pgsql提供了类似的语法:

sql 体验AI代码助手 代码解读 复制代码

```sql
INSERT INTO users (email, name, age)
VALUES ('test@example.com', 'John', 30)
ON CONFLICT (email)
DO UPDATE SET
name = EXCLUDED.name,
age = EXCLUDED.age;
```

> EXCLUDED 是一个特殊的表别名，用于引用因冲突而被排除（Excluded）的、尝试插入的那条数据.

> CONFLICT也可以直接面向唯一性约束. 假如users有一个唯一性约束: unique_email_constraint，上述SQL可以改成:

sql 体验AI代码助手 代码解读 复制代码

```sql
INSERT INTO users (email, name, age)
VALUES ('test@example.com', 'John', 30)
ON CONFLICT ON CONSTRAINT unique_email_constraint
DO UPDATE SET
name = EXCLUDED.name,
age = EXCLUDED.age;
```

6. 分页：mysql的分页使用的是： limit B(offset),A(count)， 但是pgsql不支持这种语法, pgsql支持的是如下两种:

(1)、limit A offset B; (2)、OFFSET B ROWS FETCH NEXT A ROWS ONLY;

7. pgsql查询区分大小写，而mysql是不区分的
8. 其它情况 (1)、代码中存在取1个数据的场景，原来mysql写法是 `limit 0,1` , 要调整为 `limit 1` ; (2)、在mysql中BIT(1)或tinyint（值0，1）可以转换为Boolean。但是在pgsql中不支持。需要明确使用boolean类型或INT类型, 或者使用typerhandler处理。

```sql
ALTER TABLE layout
ALTER COLUMN init_instance TYPE INT2
USING CASE
    WHEN init_instance = B'1' THEN 1
    WHEN init_instance = B'0' THEN 0
    ELSE NULL
END;

update component c
set init_instance = cp.init_instance
from  component_publish cp
where c.init_instance is null and c.id = cp.component_id ;
```

(3)、迁移数据后，统一将自增列修改

```vbnet
DO $$
DECLARE
    rec RECORD;
BEGIN
    FOR rec IN
        SELECT
            tc.sequencename
        FROM pg_sequences tc
    LOOP
        EXECUTE format('ALTER SEQUENCE %I RESTART WITH 100000', rec.sequencename);
        RAISE NOTICE 'Reset sequence % to 100000', rec.sequencename;
    END LOOP;
END $$;
```

总结

在日常开发中，我们一定要再严谨一些，规范编码。这样能让写我的代码质量更好，可移植性更高。

附录

在PostgreSQL 中，有哪些数据类型?

PostgreSQL 支持多种数据类型，下面列出一些常用的数据类型：

- **数值类型**
  - `smallint`： 2字节整数
  - `integer`： 4字节整数
  - `bigint`： 8字节整数
  - `decimal` 或 `numeric`： 任意精度的数值
  - `real`： 4字节浮点数
  - `double precision`： 8字节浮点数
  - `smallserial`： 2字节序列整数
  - `serial`： 4字节序列整数
  - `bigserial`： 8字节序列整数
- **字符与字符串类型**
  - `character varying(n)` 或 `varchar(n)`： 变长字符串，最大长度为n
  - `character(n)` 或 `char(n)`： 定长字符串，长度为n
  - `text`： 变长字符串，没有长度限制
- **日期/时间类型**

  - `date`：存储日期（年月日）
  - `time [ (p) ] [ without time zone ]`：存储时间（时分秒），可指定精度p，默认不带时区
  - `time [ (p) ] with time zone`：存储时间（时分秒），可指定精度p，带时区
  - `timestamp [ (p) ] [ without time zone ]`：存储日期和时间，默认不带时区
  - `timestamp [ (p) ] with time zone`：存储日期和时间，带时区
  - `interval`：存储时间间隔

- **布尔类型**
  - `boolean`：存储真或假值

- **二进制数据类型**
  - `bytea`：存储二进制字符串

- **几何类型**
  - `point`：二维坐标点
  - `line`：无限长直线
  - `lseg`：线段
  - `box`：矩形框
  - `path`：闭合路径或多边形
  - `polygon`：多边形
  - `circle`：圆

- **网络地址类型**
  - `cidr`：存储IPv4或IPv6网络地址
  - `inet`：存储IPv4或IPv6主机地址和可选的CIDR掩码
  - `macaddr`：存储MAC地址

- **枚举类型**
  - `enum`：用户定义的一组排序标签

- **位串类型**
  - `bit( [n] )`：固定长度位串
  - `bit varying( [n] )`：变长位串

- **JSON类型**
  - `json`：存储JSON数据
  - `jsonb`：存储JSON数据，以二进制形式存储，并支持查询操作

- **UUID类型**
  - `uuid`：存储通用唯一标识符

- **XML类型**
  - `xml`：存储XML数据

这些数据类型可以满足大多数应用的需求。在创建表时，根据实际需要选择合适的数据类型是非常重要的。

在MyBatis中，jdbcType有哪些?

`jdbcType` 是 MyBatis 和其他 JDBC 相关框架中用于指定 Java 类型和 SQL 类型之间映射的属性。以下是常见的 `jdbcType` 值及其对应的 SQL 数据类型：

- **NULL**：表示 SQL NULL 类型
- **VARCHAR**：表示 SQL VARCHAR 或 VARCHAR2 类型
- **CHAR**：表示 SQL CHAR 类型
- **NUMERIC**：表示 SQL NUMERIC 类型
- **DECIMAL**：表示 SQL DECIMAL 类型
- **BIT**：表示 SQL BIT 类型
- **TINYINT**：表示 SQL TINYINT 类型
- **SMALLINT**：表示 SQL SMALLINT 类型
- **INTEGER**：表示 SQL INTEGER 类型
- **BIGINT**：表示 SQL BIGINT 类型

- **REAL**：表示 SQL REAL 类型
- **FLOAT**：表示 SQL FLOAT 类型
- **DOUBLE**：表示 SQL DOUBLE 类型
- **DATE**：表示 SQL DATE 类型（只包含日期部分）
- **TIME**：表示 SQL TIME 类型（只包含时间部分）
- **TIMESTAMP**：表示 SQL TIMESTAMP 类型（包含日期和时间部分）
- **BLOB**：表示 SQL BLOB 类型（二进制大对象）
- **CLOB**：表示 SQL CLOB 类型（字符大对象）
- **ARRAY**：表示 SQL ARRAY 类型
- **DISTINCT**：表示 SQL DISTINCT 类型
- **STRUCT**：表示 SQL STRUCT 类型
- **REF**：表示 SQL REF 类型
- **DATALINK**：表示 SQL DATALINK 类型
- **BOOLEAN**：表示 SQL BOOLEAN 类型
- **ROWID**：表示 SQL ROWID 类型
- **LONGNVARCHAR**：表示 SQL LONGNVARCHAR 类型
- **NVARCHAR**：表示 SQL NVARCHAR 类型
- **NCHAR**：表示 SQL NCHAR 类型
- **NCLOB**：表示 SQL NCLOB 类型
- **SQLXML**：表示 SQL XML 类型
- **JAVA_OBJECT**：表示 SQL JAVA_OBJECT 类型
- **OTHER**：表示 SQL OTHER 类型
- **LONGVARBINARY**：表示 SQL LONGVARBINARY 类型
- **VARBINARY**：表示 SQL VARBINARY 类型
- **LONGVARCHAR**：表示 SQL LONGVARCHAR 类型

在使用 MyBatis 或其他 JDBC 框架时，选择合适的 `jdbcType` 可以确保数据正确地在 Java 和数据库之间进行转换。

标签： 后端

为你推荐

如何通过变更让 PostgreSQL 翻车
Bytebase | 1年前 | 604 | 1 | 评论 | 数据库

PostgreSQL的函数和存储过程
MemFireDB_Robot | 3年前 | 2.7k | 6 | 评论 | 数据库

Appflowy学习系列-熟悉Postgres数据库
武当山王也 | 12月前 | 283 | 点赞 | 评论 | 后端

mysql和postgresql差异？ gorm如何同时兼容?
sineycoder | 2年前 | 6.6k | 20 | 6 | Go | 后端

PostgreSQL技术问答00 - Why Postgres
JohnYan | 1年前 | 1.6k | 5 | 1 | 后端 | 数据库 | Postgr...

PostgreSQL 如何解决数据迁移过程中的数据类型不匹配问题?
墨松CC | 1年前 | 137 | 点赞 | 评论 | Postgr... | 数据库 | 服务器

PostgreSQL 数组类型使用详解
HelloWorld开发者社区 | 2年前 | 3.8k | 1 | 4 | 数据库 | PostgreS... | 后端

在 PostgreSQL 中，如何有效地处理数据的序列化和反序列化?
墨松CC | 1年前 | 736 | 点赞 | 评论 | 后端 | Postgr... | 数据库

postgreSql学习（一）
倔强的潴 | 2年前 | 320 | 1 | 评论 | 掘金·日新计划

现代全栈框架使用 Prisma 从 SQLite 数据库迁移到 PostgreSQL
编程杂货铺 | 1年前 | 686 | 3 | 评论 | 后端 | 前端 | Postgr...

如何在Hibernate 5框架的项目中使用Postgres JSON函数
Squids数据库云服务... | 2年前 | 889 | 点赞 | 评论 | Postgr... | Hibern... | JSON

PostgreSQL技术问答40 - Feature Matrix 功能特性矩阵
JohnYan | 11月前 | 140 | 点赞 | 评论 | 后端 | 数据库 | Postgr...

PostgreSQL应用技巧和示例(持续更新中)
JohnYan | 2年前 | 865 | 3 | 评论 | 后端 | MySQL

[草案]非主流实践：从MySQL到PostgreSQL
默默且听风 | 1年前 | 418 | 点赞 | 评论 | 数据库

全方位对比 Postgres 和 MySQL (2023 版)
Bytebase | 2年前 | 5.8k | 16 | 4 | 数据库 | MySQL | Postgr...