I'm not a robot

Yandex开源的数据分析的数据库，名字叫做ClickHouse，适合流式或批次入库的时序数据。 Yandex开源的数据分析的数据库，名字叫做ClickHouse，适合流式或批次入库的时序数据。ClickHouse不应该被用作通用数据库，而是作为超高性能的海量数据快速查询的分布式实时处理平台，在数据汇总查询方面(如GROUP BY)，ClickHouse的查询速度非常快。 2、数据分析能力 · 大多数是读请求 · 数据总是以相当大的批(> 1000 rows)进行写入 · 不修改已添加的数据 · 每次查询都从数据库中读取大量的行，但是同时又仅需要少量的列 · 宽表，即每个表包含着大量的列 · 较少的查询(通常每台服务器每秒数百个查询或更少) · 对于简单查询，允许延迟大约50毫秒 · 列中的数据相对较小： 数字和短字符串(例如，每个URL 60个字节) · 处理单个查询时需要高吞吐量（每个服务器每秒高达数十亿行） · 事务不是必须的 · 对数据一致性要求低 · 每一个查询除了一个大表外都很小 · 查询结果明显小于源数据，换句话说，数据被过滤或聚合后能够被盛放在单台服务器的内存中 (1)、行式数据 (2)、列式数据 (3)、对比分析 分析类查询，通常只需要读取表的一小部分列。在列式数据库中可以只读取需要的数据。数据总是打包成批量读取的，所以压缩是非常容易的。同时数据按列分别存储这也更容易压缩。这进一步降低了I/O的体积。由于I/O的降低，这将帮助更多的数据被系统缓存。 二、整合SpringBoot框架 该案例基于：Druid连接池和mybatis进行整合。 Druid 1.1.10 版本 SQL Parser对clickhouse的开始提供支持。 ru.yandex.clickhouse clickhouse-jdbc 0.1.53 spring: datasource: type: com.alibaba.druid.pool.DruidDataSource click: driverClassName: ru.yandex.clickhouse.ClickHouseDriver url: jdbc:clickhouse://127.0.0.1:8123/default initialSize: 10 maxActive: 100 minIdle: 10 maxWait: 6000 @Configuration public class DruidConfig { @Resource private JdbcParamConfig jdbcParamConfig ; @Bean public DataSource dataSource() { DruidDataSource datasource = new DruidDataSource(); datasource.setUrl(jdbcParamConfig.getUrl()); datasource.setDriverClassName(jdbcParamConfig.getDriverClassName()); datasource.setInitialSize(jdbcParamConfig.getInitialSize()); datasource.setMinIdle(jdbcParamConfig.getMinIdle()); datasource.setMaxActive(jdbcParamConfig.getMaxActive()); datasource.setMaxWait(jdbcParamConfig.getMaxWait()); return datasource; } } @Component @ConfigurationProperties(prefix = "spring.datasource.click") public class JdbcParamConfig { private String driverClassName ; private String url ; private Integer initialSize ; private Integer maxActive ; private Integer minIdle ; private Integer maxWait ; // 省略 GET 和 SET } 这样整合代码就完成了。 三、操作案例演示 public interface UserInfoMapper { // 写入数据 void saveData (UserInfo userInfo) ; // ID 查询 UserInfo selectById (@Param("id") Integer id) ; // 查询全部 List selectList () ; } 这里就演示简单的三个接口。 2、Mapper.xml文件 id,user_name,pass_word,phone,email,create_day INSERT INTO cs_user_info (id,user_name,pass_word,phone,email,create_day) VALUES (#{id,jdbcType=INTEGER},#{userName,jdbcType=VARCHAR},#{passWord,jdbcType=VARCHAR}, #{phone,jdbcType=VARCHAR},#{email,jdbcType=VARCHAR},#{createDay,jdbcType=VARCHAR}) select from cs_user_info where id = #{id,jdbcType=INTEGER} select from cs_user_info 这里 create_day 是以字符串的方式在转换，这里需要注意下。 3、控制层接口 @RestController @RequestMapping("/user") public class UserInfoController { @Resource private UserInfoService userInfoService ; @RequestMapping("/saveData") public String saveData (){ UserInfo userInfo = new UserInfo () ; userInfo.setId(4); userInfo.setUserName("winter"); userInfo.setPassWord("567"); userInfo.setPhone("13977776789"); userInfo.setEmail("winter"); userInfo.setCreateDay("2020-02-20"); userInfoService.saveData(userInfo); return "sus"; } @RequestMapping("/selectById") public UserInfo selectById () { return userInfoService.selectById(1) ; } @RequestMapping("/selectList") public List selectList () { return userInfoService.selectList() ; } } Please wait... 1、clickhouse应场景 1.绝大多数请求都是用于读访问的 2.数据需要以大批次（大于1000行）进行更新，而不是单行更新；或者根本没有更新操作 3.数据只是添加到数据库，没有必要修改 4.读取数据时，会从数据库中提取出大量的行，但只用到一小部分列 5.表很“宽”，即表中包含大量的列 6.查询频率相对较低（通常每台服务器每秒查询数百次或更少） 7.对于简单查询，允许大约50毫秒的延迟 8.列的值是比较小的数值和短字符串（例如，每个URL只有60个字节） 9.在处理单个查询时需要高吞吐量（每台服务器每秒高达数十亿行） 10.不需要事务 11.数据一致性要求较低 12.每次查询中只会查询一个大表。除了一个大表，其余都是小表 13.查询结果显著小于数据源。即数据有过滤或聚合。返回结果不超过单个服务器内存大小 分析类查询，通常只需要读取表的一小部分列。在列式数据库中可以只读取需要的数据。数据总是打包成批量读取的，所以压缩是非常容易的。同时数据按列分别存储这也更容易压缩。这进一步降低了I/O的体积。由于I/O的降低，这将帮助更多的数据被系统缓存。 2、整合Springboot： 核心依赖（mybatis plus做持久层，druid做数据源）： ru.yandex.clickhouse clickhouse-jdbc 0.3.1-patch com.alibaba druid 1.2.6 com.baomidou mybatis-plus-boot-starter 3.4.3.1 配置yml文件： spring: datasource: type: com.alibaba.druid.pool.DruidDataSource click: driverClassName: ru.yandex.clickhouse.ClickHouseDriver url: jdbc:clickhouse://127.0.0.1:8123/dbname username: username password: 123456 initialSize: 10 maxActive: 100 minIdle: 10 maxWait: 6000 mybatis-plus: mapper-locations: classpath*:mapper/*.xml configuration: log-impl: org.apache.ibatis.logging.stdout.StdOutImpl map-underscore-to-camel-case: true cache-enabled: true lazy-loading-enabled: true multiple-result-sets-enabled: true use-generated-keys: true default-statement-timeout: 60 default-fetch-size: 100 type-aliases-package: com.example.tonghp.entity ClickHouse与Druid连接池配置类： 参数配置： package com.example.tonghp.config; import lombok.Data; import org.springframework.boot.context.properties.ConfigurationProperties; import org.springframework.stereotype.Component; /** * @author: tonghp * @create: 2021/07/26 16:23 */ @Data @Component @ConfigurationProperties(prefix = "spring.datasource.click") public class JdbcParamConfig { private String driverClassName ; private String url ; private Integer initialSize ; private Integer maxActive ; private Integer minIdle ; private Integer maxWait ; private String username; private String password; // 省略 GET 和 SET } Druid连接池配置 package com.example.tonghp.config; import com.alibaba.druid.pool.DruidDataSource; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import javax.annotation.Resource; import javax.sql.DataSource; import javax.swing.*; /** * @author: tonghp * @create: 2021/07/26 16:22 */ @Configuration public class DruidConfig { @Resource private JdbcParamConfig jdbcParamConfig ; @Bean public DataSource dataSource() { DruidDataSource datasource = new DruidDataSource(); datasource.setUrl(jdbcParamConfig.getUrl()); datasource.setDriverClassName(jdbcParamConfig.getDriverClassName()); datasource.setInitialSize(jdbcParamConfig.getInitialSize()); datasource.setMinIdle(jdbcParamConfig.getMinIdle()); datasource.setMaxActive(jdbcParamConfig.getMaxActive()); datasource.setMaxWait(jdbcParamConfig.getMaxWait()); datasource.setUsername(jdbcParamConfig.getUsername()); datasource.setPassword(jdbcParamConfig.getPassword()); return datasource; } } 接下来配置实体类，mapper，service，controlle以及mapper.xml。与mybatisplus操作mysql package com.example.tonghp.entity; import lombok.Data; import java.io.Serializable; /** * @author: tonghp * @create: 2021/07/26 16:31 */ @Data public class UserInfo implements Serializable { private static final long serialVersionUID = 1L; private int id; private String userName; private String passWord; private String phone; private String email; private String createDay; } package com.example.tonghp.mapper; import com.baomidou.mybatisplus.core.mapper.BaseMapper; import com.example.tonghp.entity.UserInfo; import org.apache.ibatis.annotations.Mapper; import org.apache.ibatis.annotations.Param; import org.springframework.stereotype.Repository; import java.util.List; /** * @author: tonghp * @create: 2021/07/26 16:32 */ @Repository public interface UserInfoMapper extends BaseMapper { // 写入数据 void saveData (UserInfo userInfo) ; // ID 查询 UserInfo selectById (@Param("id") Integer id) ; // 查询全部 List selectList () ; } UserInfoMapper.xml