云数据仓库 PostgreSQL

操作指南

产品文档

腾讯云

文档目录

操作指南
管理集群
创建集群

最近更新时间：2024-07-19 11:40:16

操作场景

本文为您介绍云数据仓库 PostgreSQL 控制台创建集群的操作。

操作步骤

1. 进入 云数据仓库 PostgreSQL 控制台，在控制台选择集群列表。
2. 单击**新建**进入购买页，根据需求输入参数信息。

计费模式：支持包年包月和按量计费。
地域：目前支持广州、上海、北京、新加坡地域。
可用区：不同可用区不互通。
网络：选择希望连接到云数据仓库 PostgreSQL 的 VPC 网络以及子网信息，可在 私有网络 进行查询或规划。如需要其他子网或者外网能够访问，请参见 申请外网地址 及 创建 IP 白名单。
集群名称 ：用来对不同的集群进行区分。
集群版本：集群的支持版本。
节点类型：目前支持 nc.large、nc.4xlarge、nc2.large、nc2.4xlarge、ns.large、ns2.large 六种规格，详细信息请参见 计费概述。
节点数量 ：支持2 - 50个节点，一般要求大于等于2个计算节点，最多50个节点。
数据库端口：5436，目前无法修改。
用户名：使用该用户名来登录云数据仓库 PostgreSQL 集群，集群管理者账号，创建后不能改名。
密码：连接云数据仓库 PostgreSQL 集群需要使用的登录密码，可在控制台修改。


可用区 *
北京六区
北京七区
网络信息
网络 *
新建私有网络
子网 *
新建子网
共 253 个子网 IP，剩 236 个可用
网络/子网选项仅展示20条，更多选项可通过名称关键字搜索获取
账户信息
密码 *
请输入密码
以字母、数字开头，包含小写字母、大写字母、数字、特殊字符(!@#%^*)中的三种，8-16位字符
确认密码 *
请输入密码
默认用户名为dbadmin，集群创建后，您可以通过「账户管理」修改密码
集群配置
CN 节点配置 *
标准型
高性能型
大数据型
计算规格 *
4核16G
存储规格 *
增强型SSD云硬盘
200
GB
单节点最小支持 200GB，最大支持 32000GB，步长为100
节点数量 *
2
节点数量大于等于2台，小于等于1000台，库存节点1576，库存不足时请更换规格或 联系我们 协调资源
DN 节点配置 *
标准型
高性能型
大数据型
计算规格 *
4核16G
存储规格 *
增强型SSD云硬盘
200
GB
单节点最小支持 200GB，最大支持 320000GB，步长为100
节点数量 *
2
节点数量大于等于2台，小于等于1000台，库存节点1999，库存不足时请更换规格或 联系我们 协调资源
数据复制方式 *
强同步（可退化）
异步
强同步
仅存活 1 个节点时，实例将自动退化为异步复制
字符集 *
UTF8
LATIN1
EUC_CN
SQL_ASCII
若字符集设置不当会导致数据库导入发生错误
其他配置
标签信息
标签键
标签值
添加
键值粘贴板

3. 配置完成后，单击**立即购买**或**开通**创建集群。
4. 返回集群列表，待集群处于**运行中**状态时，即可正常使用集群。

集群信息

最近更新时间：2024-07-19 11:40:46

本文为您介绍通过控制台修改云数据仓库 PostgreSQL 的基本信息、计费模式，以及查看集群的性能监控、实时查询、历史查询、事件监控。

基本信息

从集群列表页进入到集群**基础配置**页，可对集群名称、网络以及管理用户密码进行设置。

注意：

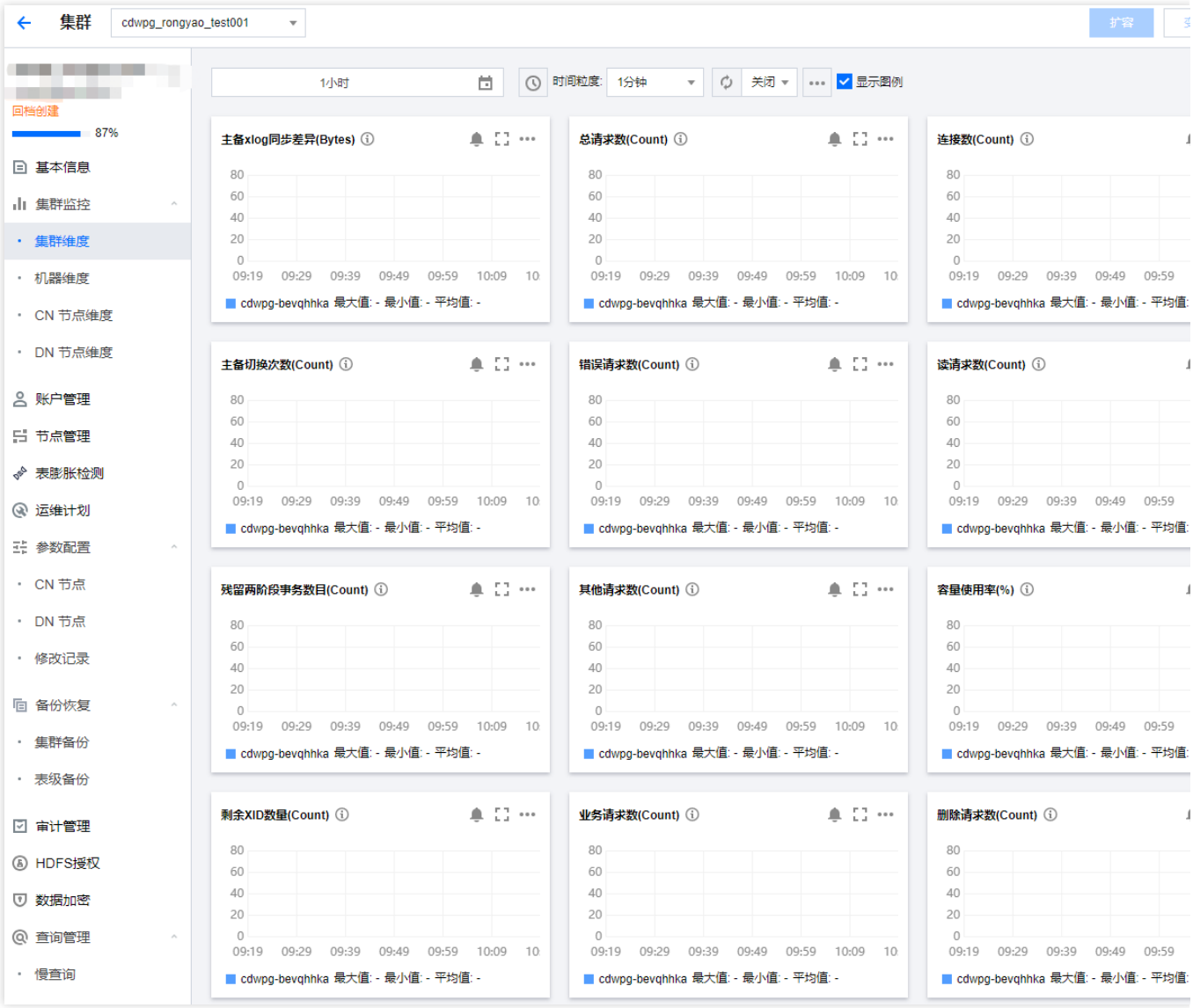
网络地址修改后会导致集群访问连接发生变化，需要同步修改调用地址。

计费信息

可在**基础配置**页，对集群的计费模式进行修改，将按量计费集群转换为包年包月集群，并设置是否需要**自动续费**。

性能监控

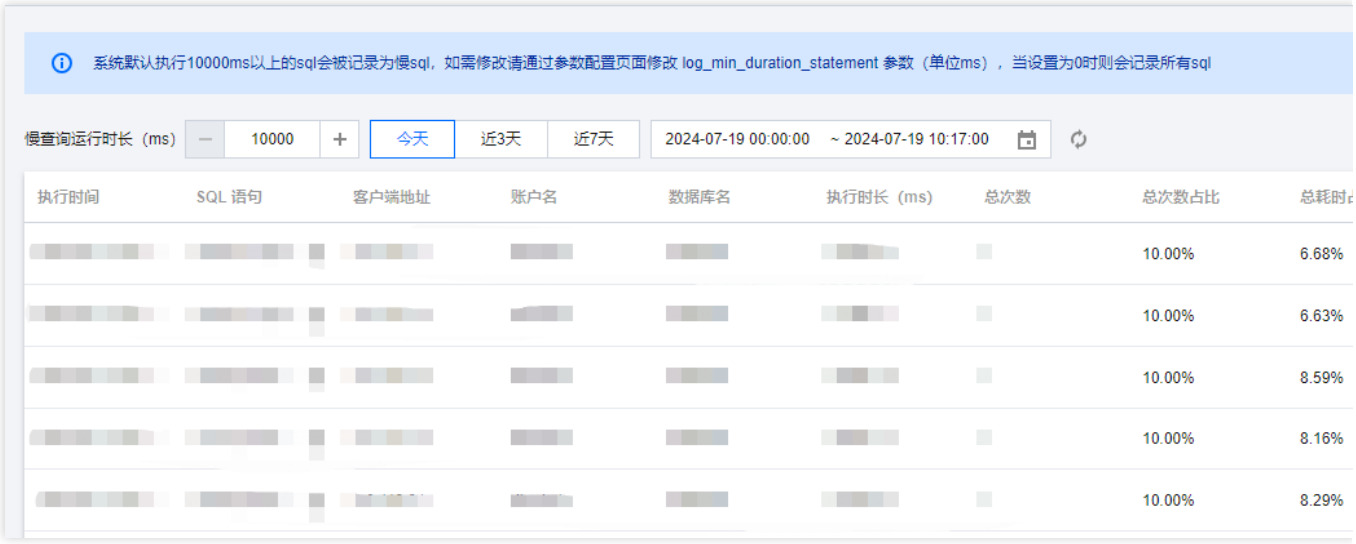
在**性能监控**页面，可以从节点维度、机器维度和集群维度查看各项指标。



通过单击**终止查询**可以终止列表中的查询。

慢查询

在**慢查询**页面，可以查看已经完成的查询。


系统默认执行10000ms以上的sql会被记录为慢sql，如需修改请通过参数配置页面修改 log_min_duration_statement 参数（单位ms），当设置为0时则会记录所有sql
慢查询运行时长（ms）
10000
今天
近3天
近7天
2024-07-19 00:00:00 ~ 2024-07-19 10:17:00
执行时间
SQL 语句
客户端地址
账户名
数据库名
执行时长（ms）
总次数
总次数占比
总耗时占
10.00%
6.68%
10.00%
6.63%
10.00%
8.59%
10.00%
8.16%
10.00%
8.29%

扩容集群

最近更新时间：2024-07-19 11:41:10

操作场景

当集群资源计算资源或者存储资源达到瓶颈后，可通过扩容的方式来增加系统资源。

操作步骤

1. 在集群列表的**操作**列，单击**管理**进入集群详情页。
2. 单击右上角**扩容**进入扩容页面，根据需求选择扩容后信息。

注意：

输入节点数量，该数量为扩容后集群节点数量，节点数量必须大于或等于当前集群节点数量 + 2。修改节点数量时，总容量处的值会根据节点数量动态变化。

3. 确定好扩容数量后，单击**确定变更**开始扩容流程。

4. 返回集群列表，扩容中集群状态为**扩容中**，扩容时间与集群已有数据量有关，扩容成功后，集群状态恢复为**运行中**。

管理 IP 白黑名单

最近更新时间：2024-07-19 11:41:43

操作场景

除使用指定子网中的 CVM 进行访问云数据仓库 PostgreSQL 集群方式外，如需通过其他子网中的 CVM 或者外网机器访问云数据仓库 PostgreSQL 集群时，需要将这些 IP 地址加入到白名单中。
添加 IP 白名单时，指定了某个网段，但想要禁止该网段中的某个或某几个 IP 地址访问云数据仓库 PostgreSQL 集群，可以通过额外添加 IP 黑名单完成。
注意：
黑名单的优先级高于白名单。

前提条件

使用 IP 白名单之前，需确定已经申请了集群外网地址，否则即使加入白名单中，也无法使用指定子网 CVM 以外的机器访问云数据仓库 PostgreSQL 集群。

操作步骤

管理 IP 白名单

1. 在集群列表单击**管理**进入集群详情页。
2. 选择**配置** > **访问白名单**页面。
3. 单击**新建白名单**，根据需要输入 IP、用户名、数据库名和 CIDR 地址。

4. 单击**确认**后，将会把 CIDR 指定的地址段添加到白名单中，添加成功后，在该地址或者网络段的主机都能访问到该云数据仓库 PostgreSQL 集群。

说明：

当不需要某个 IP 白名单时，在白名单列表中，选中对应地址单击**删除**即可。

管理 IP 黑名单

1. 选择**配置** > **访问黑名单**页面。
2. 单击**新建黑名单**，根据需要输入 IP、用户名、数据库名和 CIDR 地址。
3. 单击**确认**后，将会把 CIDR 指定的地址段添加到黑名单中，添加成功后，在该地址或者网络段的主机无法访问到该云数据仓库 PostgreSQL 集群。

说明：

当要解除对某个 IP 或地址段的禁止访问时，在黑名单列表中，选中对应地址单击**删除**即可。

申请外网地址

最近更新时间：2024-07-19 11:42:27

操作场景

除使用指定子网中的 CVM 进行访问云数据仓库 PostgreSQL 集群外，如需通过其他子网中的 CVM 或者外网机器访问云数据仓库 PostgreSQL 集群时，需要申请外网地址。

注意：

申请外网地址，只是表明云数据仓库 PostgreSQL 集群能够提供外网地址访问的能力，但具体要访问云数据仓库 PostgreSQL 集群的外网 IP 需要通过 IP 白名单 进行添加。

操作步骤

1. 在集群列表单击**管理**进入集群详情页。
2. 选择**基础配置**页面，单击**申请外网地址**，将会生成一个能够访问此云数据仓库 PostgreSQL 集群的外网地址。

说明：

当不需要使用外网地址时，单击**释放外网地址**删除外网地址。

管理资源队列

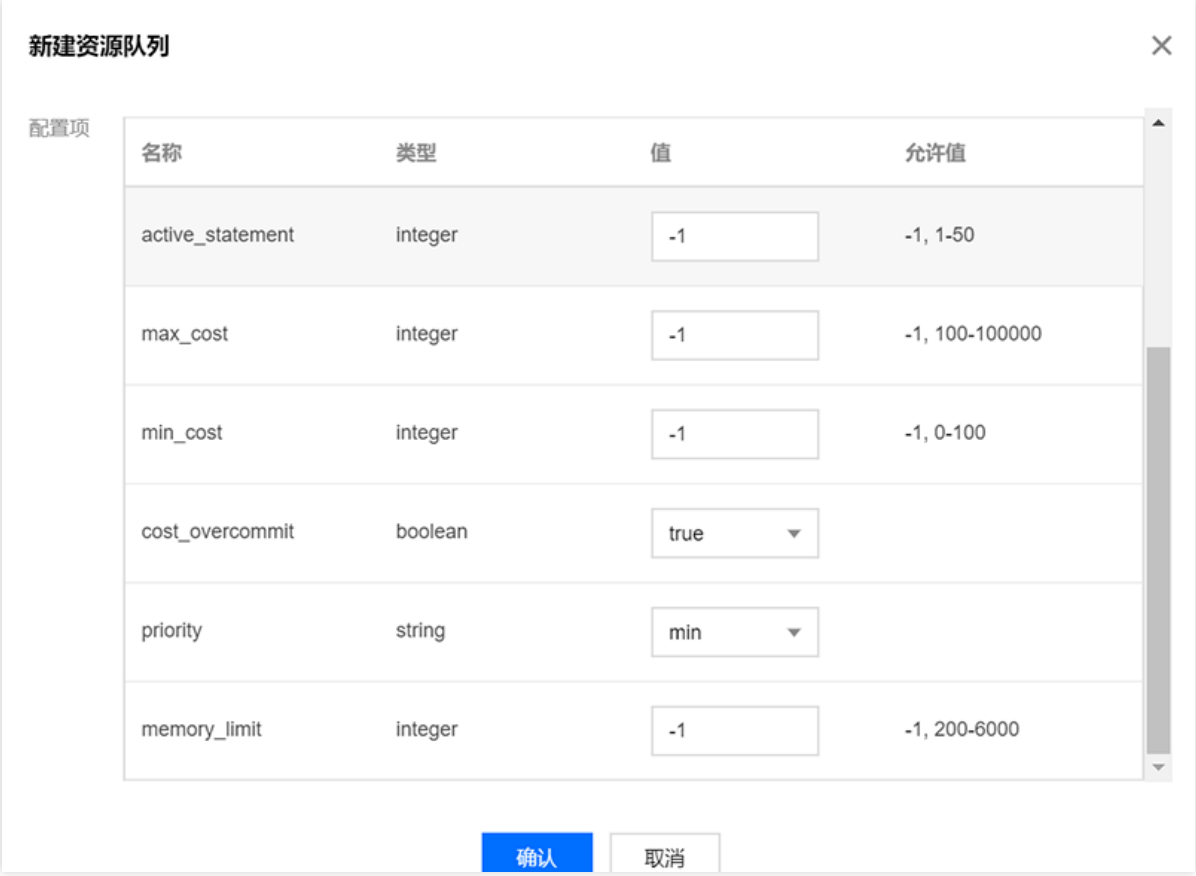
最近更新时间：2024-07-19 11:42:45

操作场景

在使用云数据仓库 PostgreSQL 的过程中，单个复杂查询可能会消耗过多资源，会影响其他用户的查询或者计算，当需要对单个用户或者查询语句进行系统资源消耗限制时，可以使用资源队列进行限制。

操作步骤

1. 在集群列表单击**管理**，进入集群详情页。
2. 选择**配置** > **资源队列**，单击**新建资源队列**，配置各项参数值 。
3. 单击**确认**，资源队列创建后，查询时可指定资源队列对资源进行限制。



销毁集群

最近更新时间：2024-07-19 11:43:58

操作场景

当您不再需要按量计费集群时，可以通过 云数据仓库 PostgreSQL 控制台 删除集群。

注意：

集群删除后将直接删除所有数据，请谨慎操作。

操作步骤

1. 在集群列表单击**管理**，进入集群详情页。
2. 在右上角单击**销毁**，进入销毁页。

3. 单击**确认**，按量计费集群将进入**删除中**状态。

注意：

按量计费和包年包月集群，销毁页面不同。

按量计费直接确认是否销毁，一旦确认，则启动销毁流程，一经销毁，数据将无法找回。

包年包月集群的销毁会进入订单退费页面，退费确认后，将开始集群的隔离操作，集群隔离后保留7天，7天后销毁数据，在此期间可以通过续费操作进行恢复，但该操作只能进行一次。

访问数据仓库
连接数据库

最近更新时间：2024-07-23 12:03:18

默认情况下仅支持同一 VPC 子网下的 CVM 服务器对云数据仓库 PostgreSQL 集群进行访问；如需支持公网直接访问数据仓库集群，请申请 外网地址。
您在创建好数据仓库集群，开始使用数据库服务前，需要使用数据库客户端连接到数据库。请参考以下指导，使用 psql 客户端工具连接数据库。
1. 获取集群访问地址：通过集群 JDBC URL 中的 IP 和端口连接数据库。
2. 连接到集群数据库：安装客户端并连接集群数据库。

前提条件

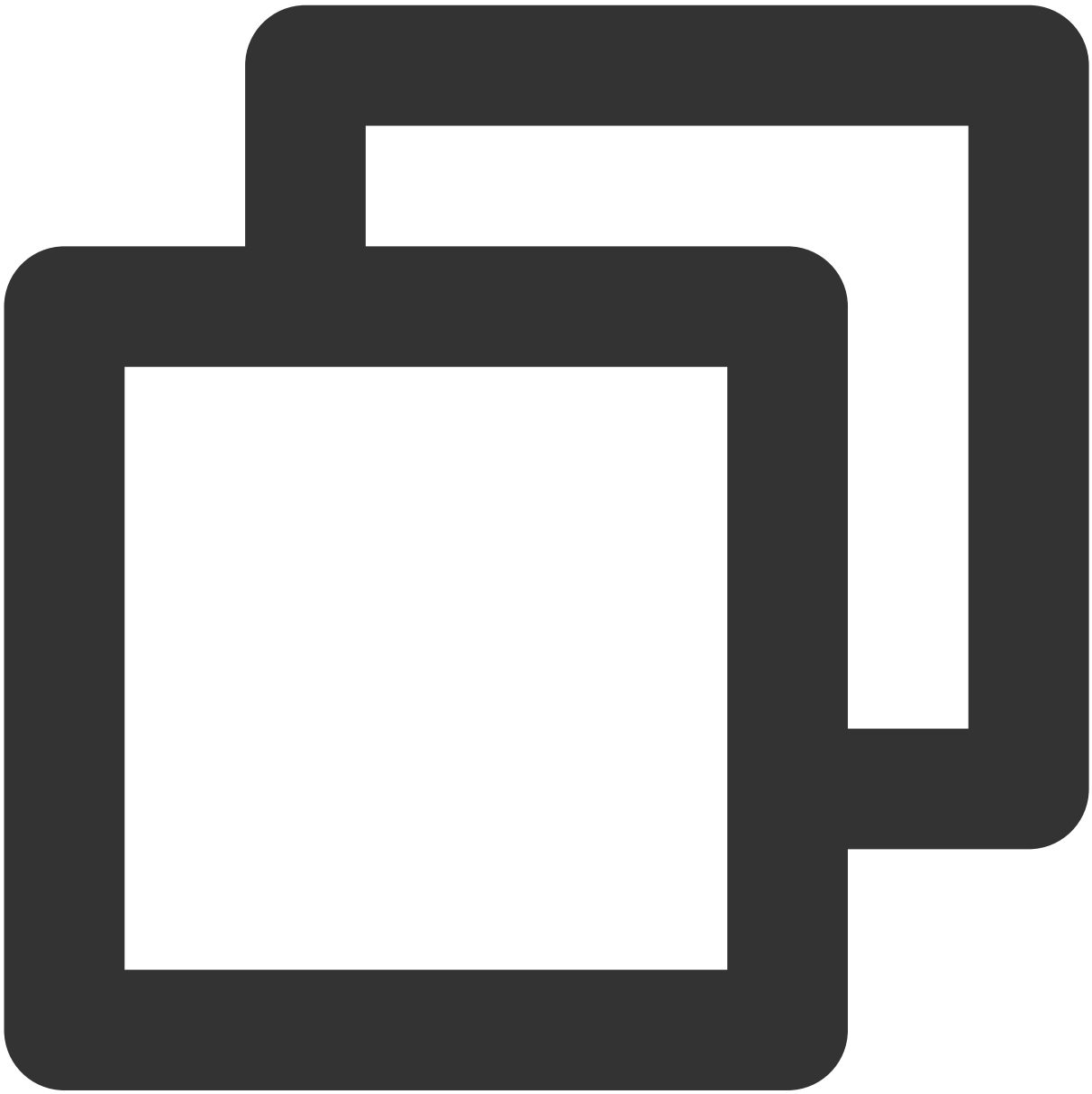
1. 已获取云数据仓库 PostgreSQL 集群的数据库管理员密码。数据库管理员密码为开始创建集群时设置的管理员账号密码。
2. 已获取创建好的云数据仓库 PostgreSQL 集群的访问 IP、端口以及 VPC 和子网。

获取集群访问地址和本地网络

选择对应的集群，详细信息如图所示，获取 VPC 网络：vpc-aejsd98p，子网：subnet-83knqldq。链接 Snova 的 IP 为10.0.6.10，端口为5432，登录帐户为 lambuser。

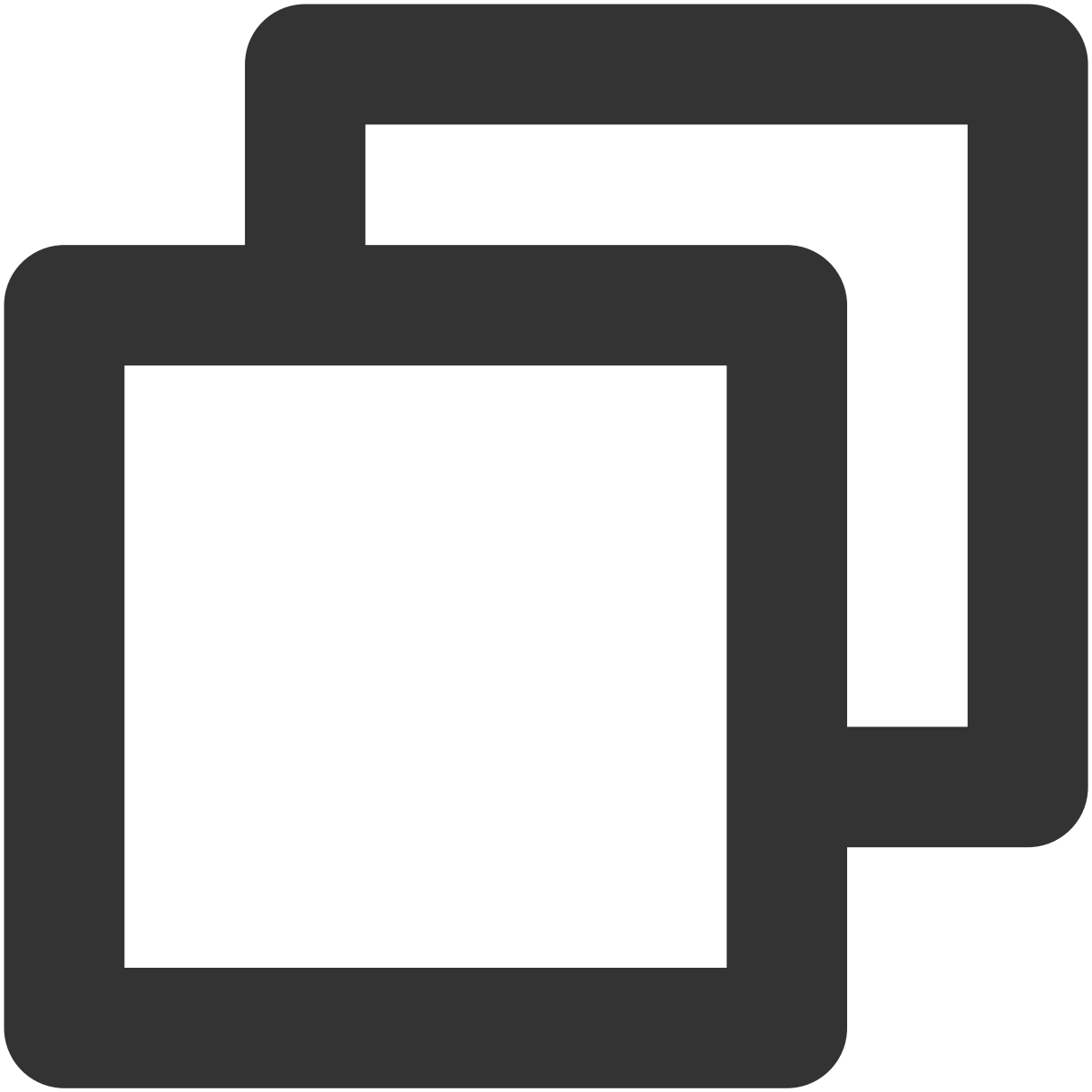
命令行连接到集群数据库

在获取的 VPC 网络：vpc-aejsd98p，子网：subnet-83knqldq 中选择一台 CVM 服务器（如果不存在，则购买一台即可）。登录该服务器，执行以下命令，即可安装 PostgreSQL 客户端。



```
yum install -y postgresql.x86_64
```

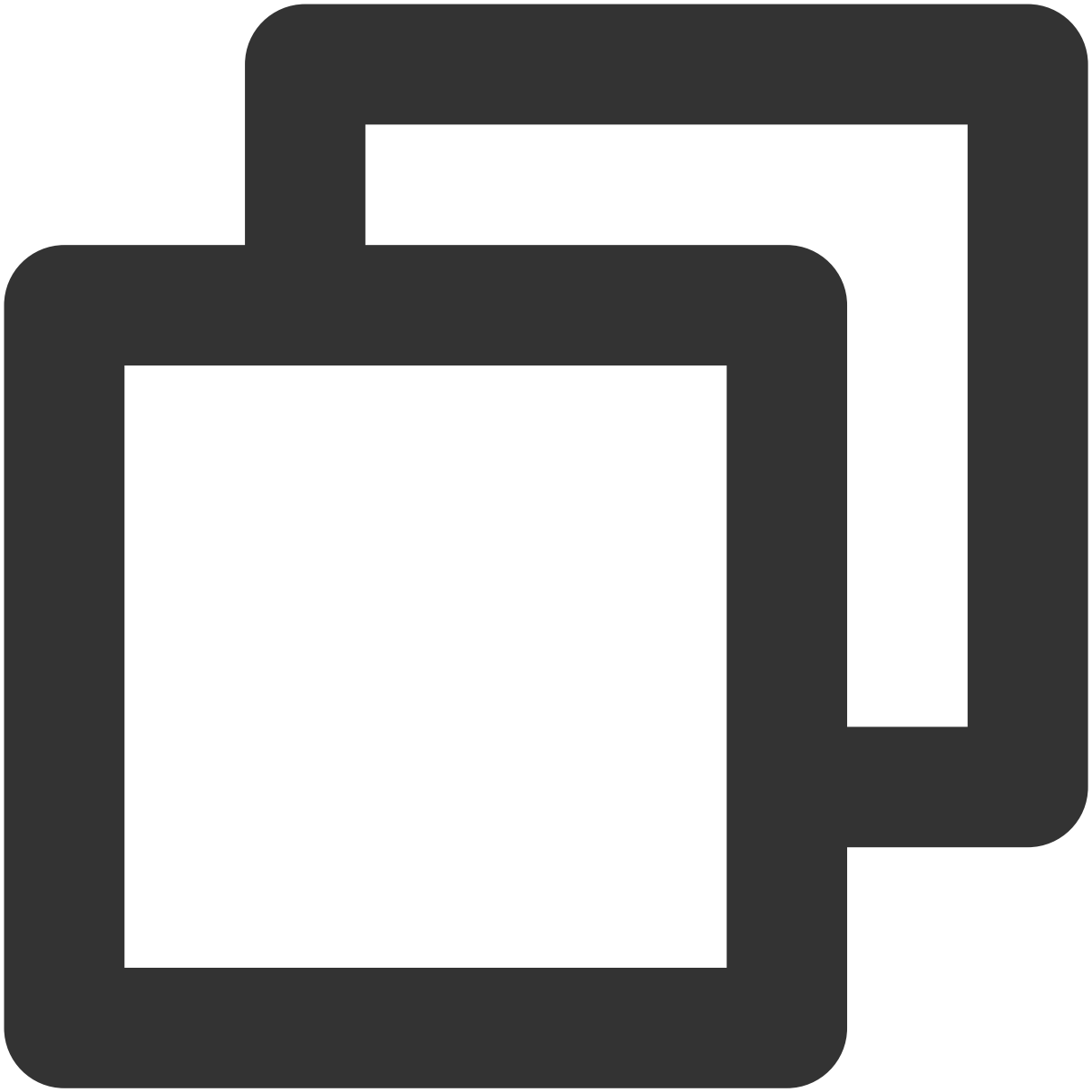
执行下面的 SQL 命令，然后输入创建集群时输入的密码，即可登录成功。



```
psql -h10.0.6.10  -p5432  -dpostgres  -Ulambuser
```

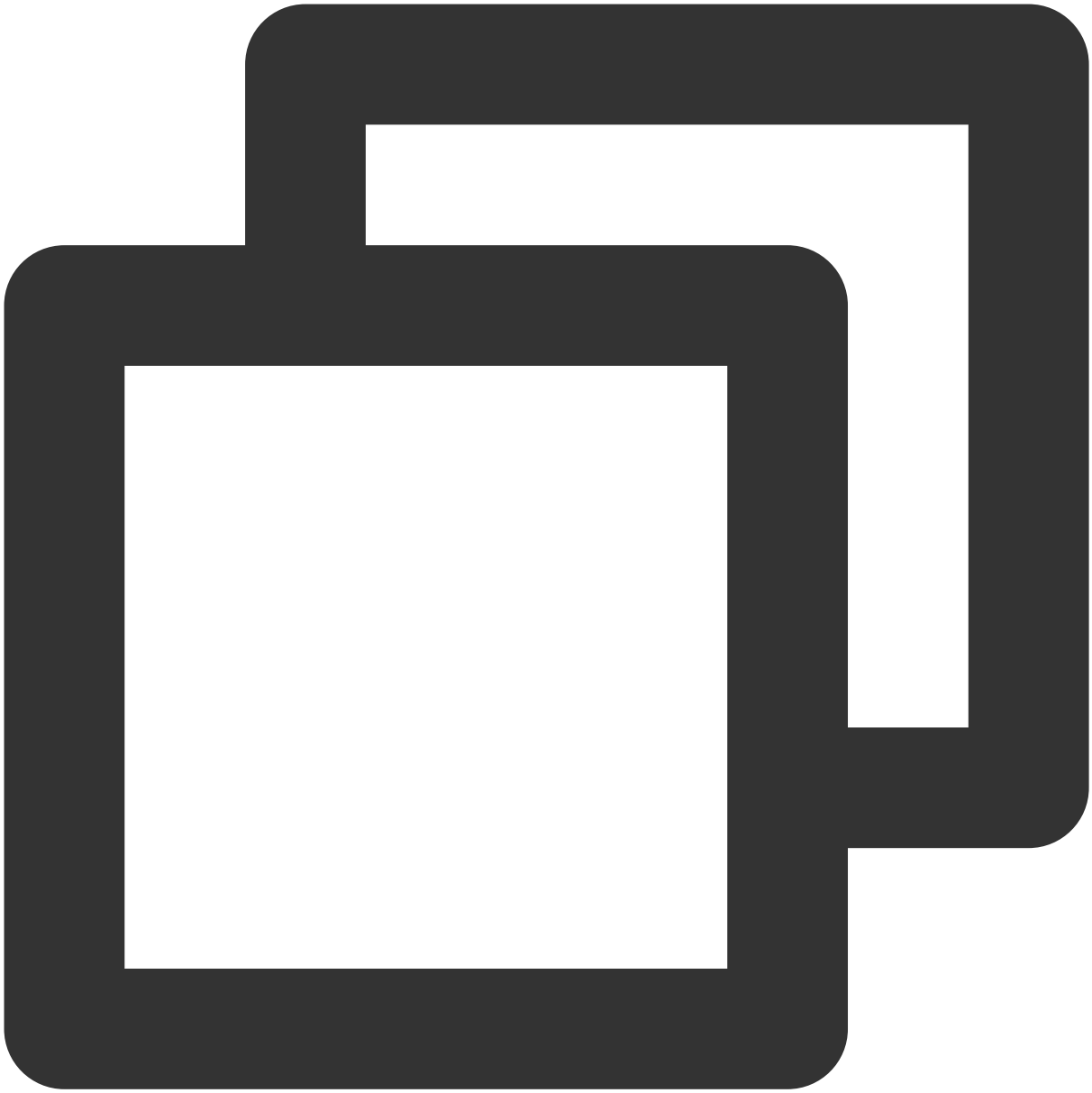
JDBC 连接数据库

需要获取 PostgresSQL 官方提供的 JDBC， 下载地址。
或者在 pom.xml 文件中添加如下配置：



```
<dependencies>
        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <version>42.2.2</version>
        </dependency>
</dependencies>
```

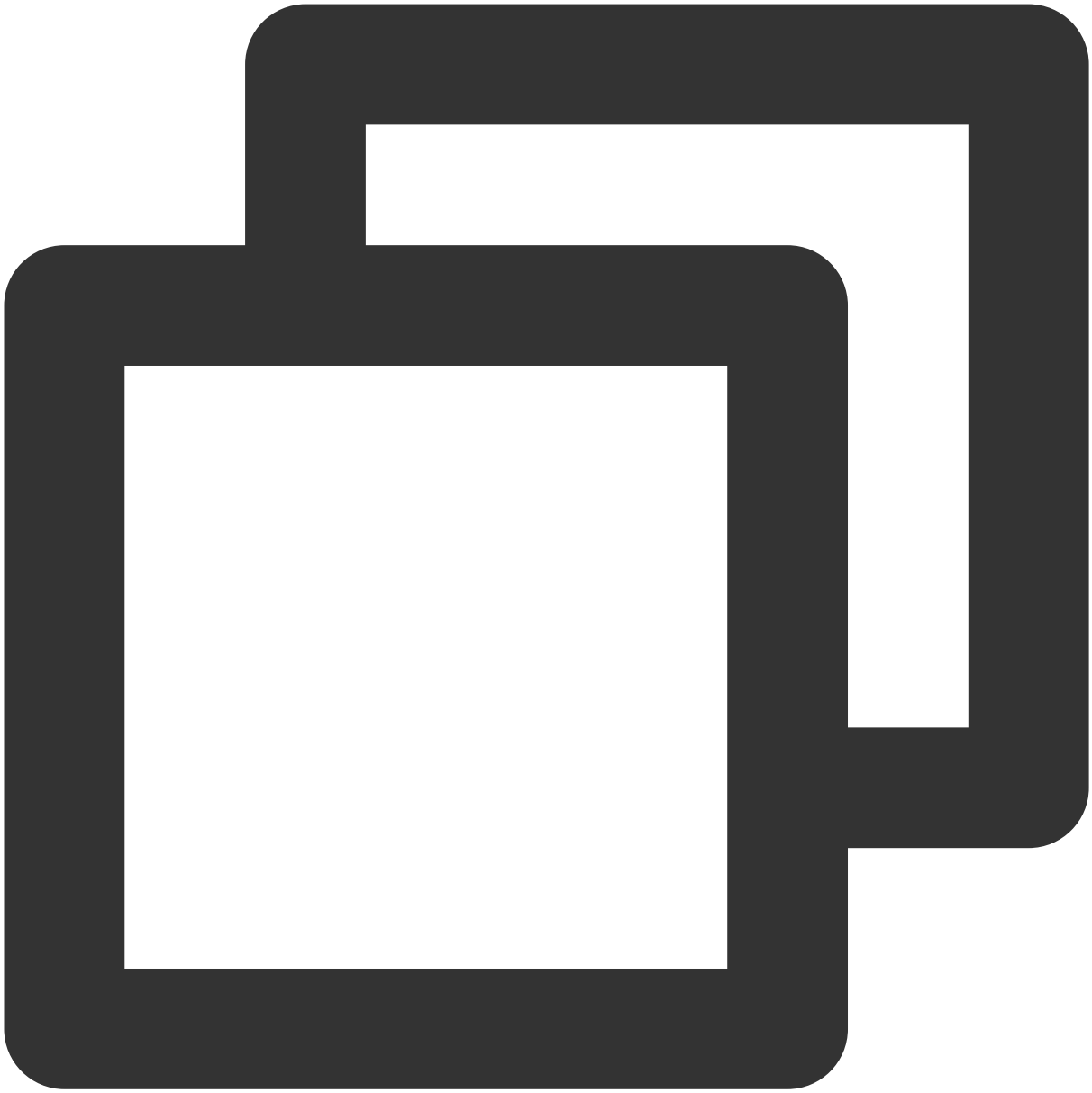
示例代码



```
package com.qcloud.snova_conn;

import java.io.InputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.Statement;
import java.sql.Timestamp;
import java.util.ArrayList;
```

```
import java.util.List;
import java.util.Properties;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import com.yammer.metrics.core.Meter;

public class SnovaConn {
    /*
     * args: vip vport user pwd
     */
    public static void main(String[] args) throws ClassNotFoundException, SQLExcept

        if (args.length < 4){
                System.out.println("args err");
                return;
        }

        String vip =  args[0];
        String vport =  args[1];
        String userName =  args[2];
        String userPwd =  args[3];

        System.out.printf("vip:%s, vport:%s, userName:%s, userPwd:%s\\n",vip, vport
        String jdbcUrl = "jdbc:postgresql://" + vip+":"+vport+"/maxluo";
        System.out.printf("jdbcUrl:%s \\n",jdbcUrl);

        Class.forName("org.postgresql.Driver");
        Connection snova = DriverManager.getConnection(jdbcUrl,userName,userPwd);
        Statement st = snova.createStatement();
        ResultSet rs = st.executeQuery("select * from test;");
        while (rs.next()) {
                System.out.print(rs.getString(1));
                System.out.print("\\n");
        }
        rs.close();
        st.close();
    }
}
```

pom.xml 配置



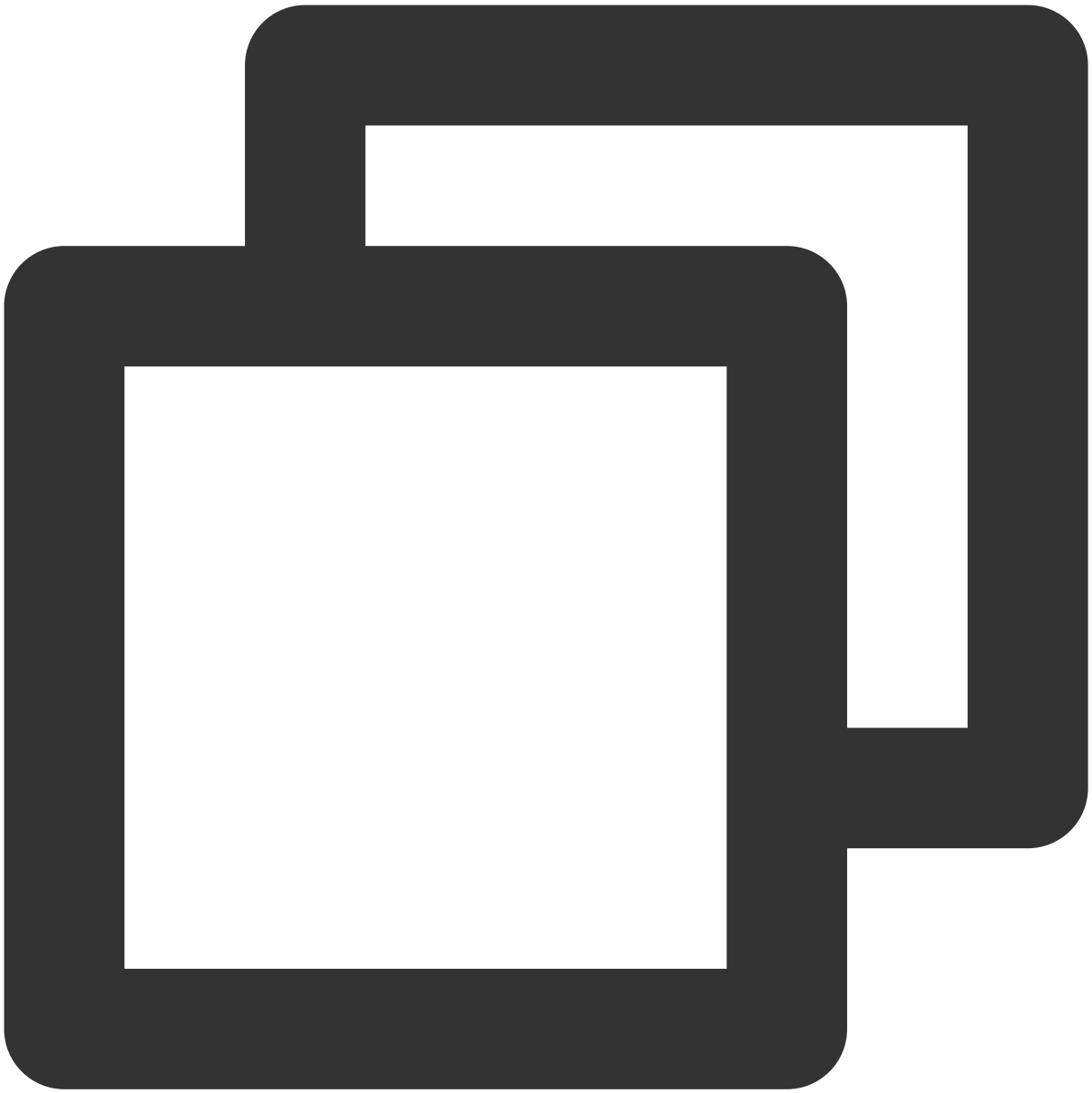
```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/200
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd
<modelVersion>4.0.0</modelVersion>

<groupId>com.qcloud</groupId>
<artifactId>snova-conn</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>

<name>snova-conn</name>
<url>http://maven.apache.org</url>
```

```
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
</properties>

<dependencies>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.40</version>
    </dependency>
    <dependency>
        <groupId>org.postgresql</groupId>
        <artifactId>postgresql</artifactId>
        <version>42.2.2</version>
    </dependency>
    <dependency>
        <groupId>com.microsoft.sqlserver</groupId>
        <artifactId>mssql-jdbc</artifactId>
        <version>6.4.0.jre8</version>
    </dependency>
    <dependency>
        <groupId>com.yammer.metrics</groupId>
        <artifactId>metrics-core</artifactId>
        <version>2.2.0</version>
    </dependency>
    <dependency>
        <groupId>ch.qos.logback</groupId>
        <artifactId>logback-classic</artifactId>
        <version>1.1.9</version>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <artifactId>maven-assembly-plugin</artifactId>
                <configuration>
                <descriptorRefs>
                <descriptorRef>jar-with-dependencies</descriptorRef>
                </descriptorRefs>
                </configuration>
            <executions>
                <execution>
```

```
                <id>make-assembly</id>
                <phase>package</phase>
                <goals>
                <goal>single</goal>
                </goals>
                </execution>
            </executions>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-jar-plugin</artifactId>
            <configuration>
                <excludes>
                    <exclude>*.properties</exclude>
                    <exclude>*.xml</exclude>
                    <exclude>*.json</exclude>
                    <exclude>*.sh</exclude>
                </excludes>
            </configuration>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-dependency-plugin</artifactId>
            <executions>
                <execution>
                    <id>copy-dependencies</id>
                    <phase>package</phase>
                    <goals>
                        <goal>copy-dependencies</goal>
                    </goals>
                    <configuration>
                        <type>jar</type>
                        <includeTypes>jar</includeTypes>
                        <outputDirectory>
                            ${project.build.directory}/lib
                        </outputDirectory>
                    </configuration>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
</project>
```

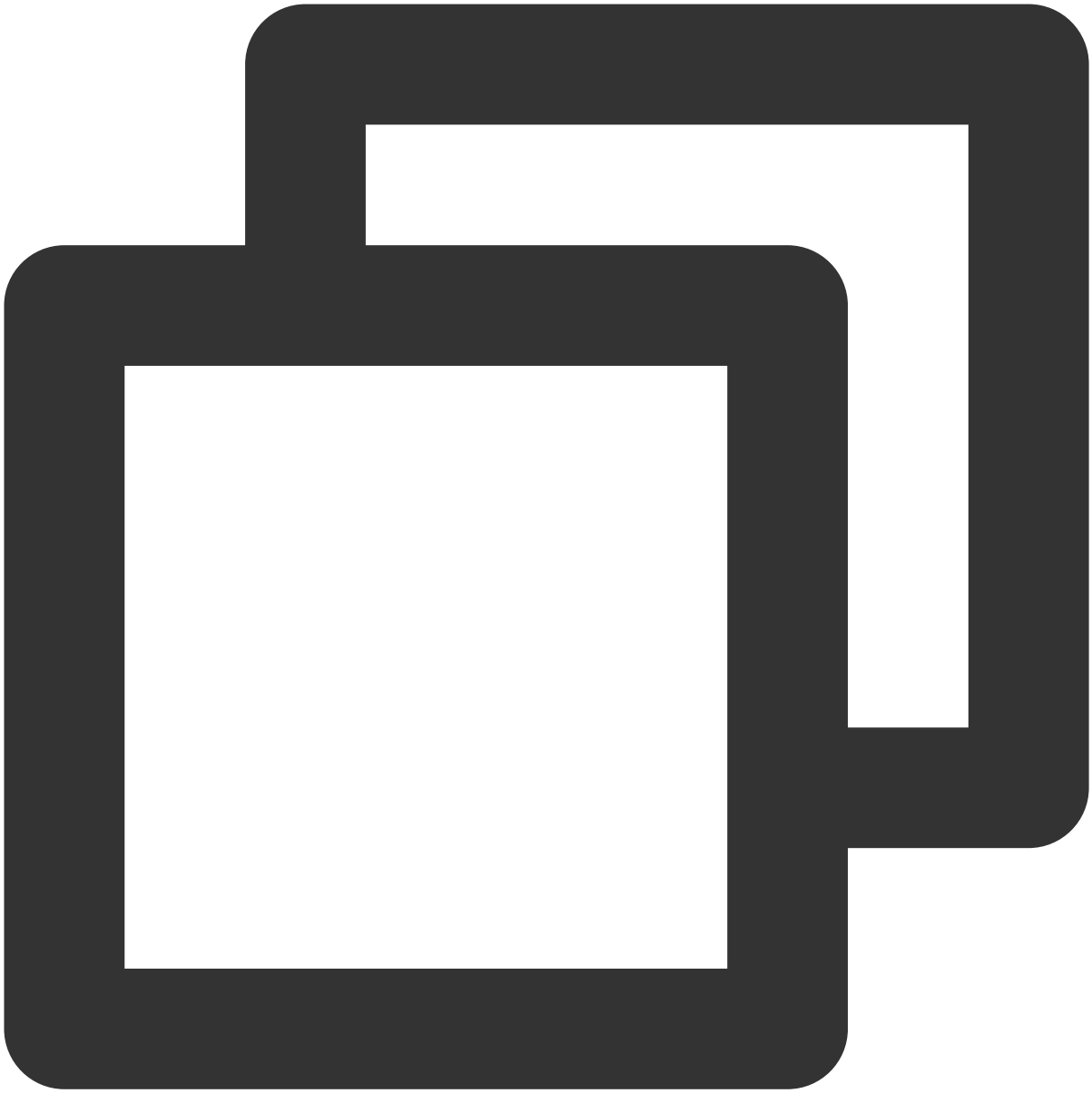
然后采用 Maven 打包生成 jar 文件，并把该 jar 包上传至 CVM 服务器（该云数据仓库 PostgreSQL 集群所在 VPC 子网中的任意一台 CVM 均可）。

执行以下命令安装 jdk。



```
yum install java
```

执行命令：



```
java -cp  snova-conn-0.0.1-SNAPSHOT-jar-with-dependencies.jar  com.qcloud.snova_con
```

注意：

VIP 和端口分别为云数据仓库 PostgreSQL 集群链接地址，用户名和密码分别是创建集群时填写的信息，具体获取方式见前文。

通过命令行方式建立数据库和数据表，并插入一定量的数据。

查询结果如下，可以读取到事先建立的数据库 maxluo 中表 test 中的数据：

```
postgres=> \c maxluo;
psql (9.2.23, server 8.3.23)
WARNING: psql version 9.2, server version 8.3.
         Some psql features might not work.
You are now connected to database "maxluo" as user "lambuser".
maxluo=> \d
No relations found.
maxluo=> create table test(a1 int);
NOTICE:  Table doesn't have 'DISTRIBUTED BY' clause -- Using column named 'a1' as the Greenplum Database data distribution key for
HINT:  The 'DISTRIBUTED BY' clause determines the distribution of data. Make sure column(s) chosen are the optimal data distributi
CREATE TABLE
maxluo=>
maxluo=> \d
        List of relations
 Schema | Name | Type  |  Owner
--------+------+-------+----------
 public | test | table | lambuser
(1 row)

maxluo=> insert into test values(1),(32),(323);
INSERT 0 3
maxluo=> select * from test;
 a1
-----
  32
 323
   1
(3 rows)

maxluo=>
```

管理用户权限

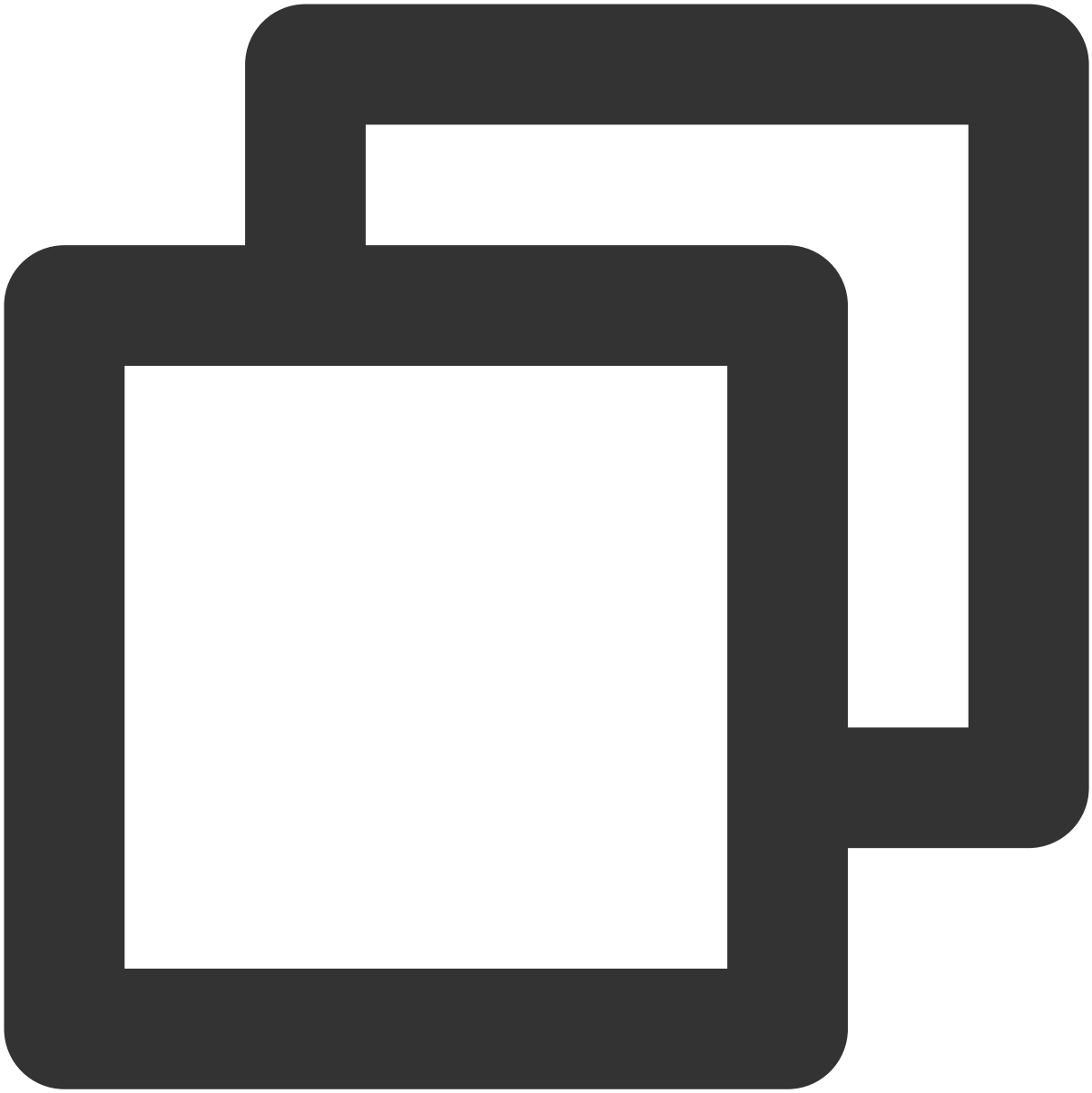
最近更新时间：2024-02-19 10:49:28

角色概述

在云数据仓库 PostgreSQL 中使用 roles 来对数据库的访问权限进行管理，角色（roles）的概念实际上包含了 users 与 groups 两部分,一个 role 可以是数据库普通用户，即 user，也可以是数据库的用户组，即 group。角色可以拥有数据库对象（例如表、视图等），并且可以将这些对象的访问权限分配给其他角色。
集群创建时，会提示用户设置初始用户名和密码，这个初始用户为“管理员用户”，该用户拥有创建用户、创建数据库、登录的权限。集群创建完成后，可以使用“管理员用户”连接数据库，一般地，管理员用户拥有最大化的权限，也意味着这个账号要被尽量少的人使用，因此您可以使用管理员用户创建其他用户，并为这些用户授予所需要的权限，具体授权参考 用户组 与 对象权限管理。您也可以创建数据库以及其他对象，参考 定义数据库。登录数据库可以参考 连接数据库。

创建用户

role 分为用户 user 和用户组 group，通常 user 级别的 role（后面统称 user）具有能够登录云数据仓库 PostgreSQL 数据库并初始化一个会话的权限，因此，在建立一个 user 时，必须为其赋予 LOGIN 的权限。例如：



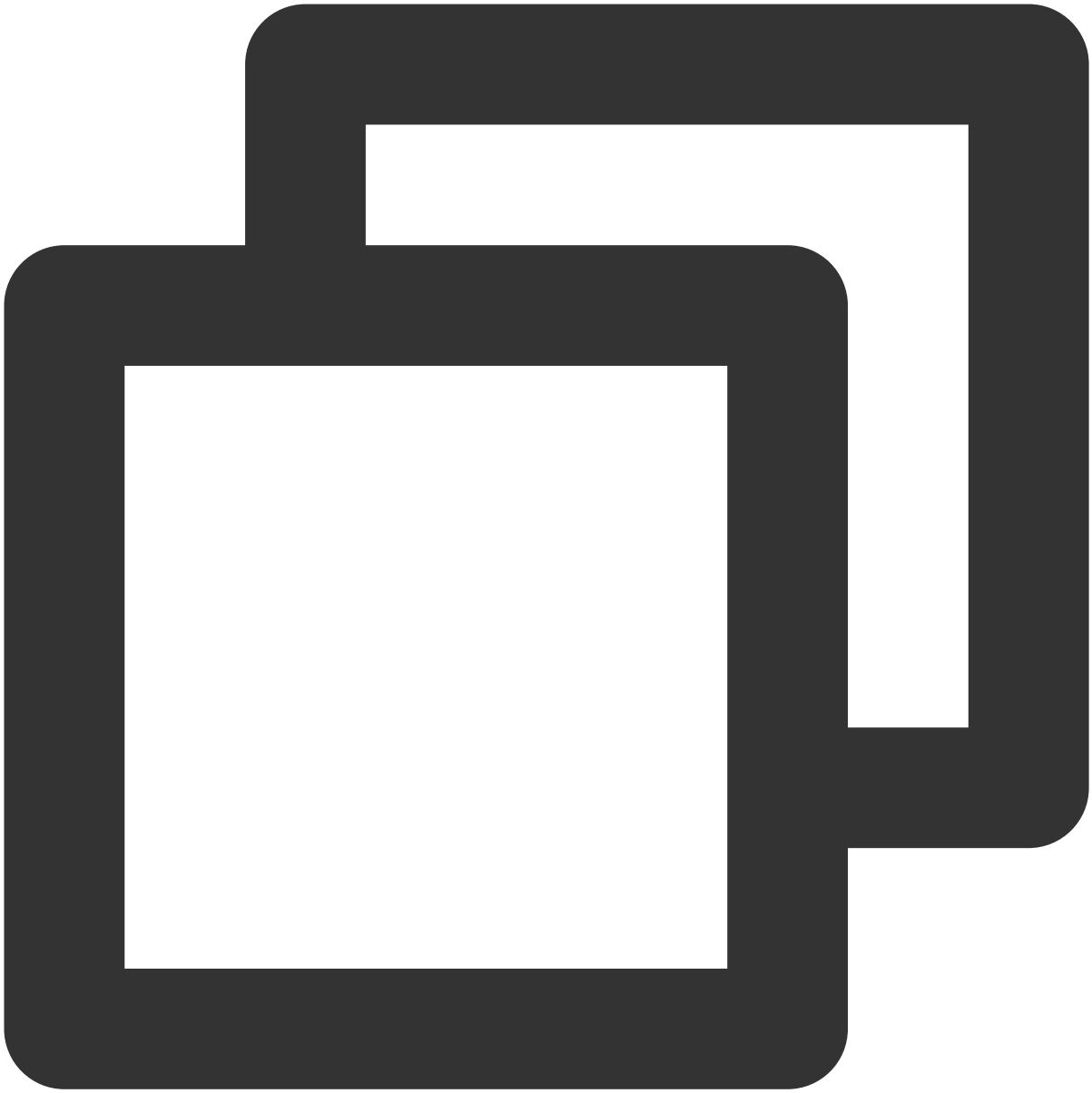
```
CREATE role jsmith with LOGIN;
```

通过上述操作，创建了一个具有 LOGIN 权限的 user，通过这个 user，可以对数据库进行连接。云数据仓库 PostgreSQL 对用户的访问管理除了 LOGIN 权限外，还有以下其他权限，可以在使用 CREATE ROLE 语句创建角色时对其所能拥有的权限进行授权。

权限取值	作用	默认值
SUPERUSER &lota; NOSUPERUSER	超级用户权限，只有超级用户才能创建其他的超级用户	NOSUPERUSER

CREATEDB Ι NOCREATEDB	创建数据库的权限	NOCREATEDB
CREATEROLE Ι NOCREATEROLE	创建和管理角色	NOCREATEROLE
INHERIT Ι NOINHERIT	决定了用户继承所属 group 的权限	INHERIT
LOGIN Ι NOLOGIN	连接登录数据库的权限，一般用户拥有该权限，用户组无该权限	NOLOGIN
CONNECTION LIMIT	限制能够并发连接数据库的连接数，-1表示无限制	-1
CREATEEXTTABLE Ι NOCREATEEXTTABLE	创建外表的权限	NOCREATEEXTTABLE
PASSWORD	创建用户时设置密码	无
VALID UNTIL 'timestamp'	密码的到期时间	无
RESOURCE QUEUE 'name'	用户连接后，建立的查询被安排到的资源队列名	pg_default

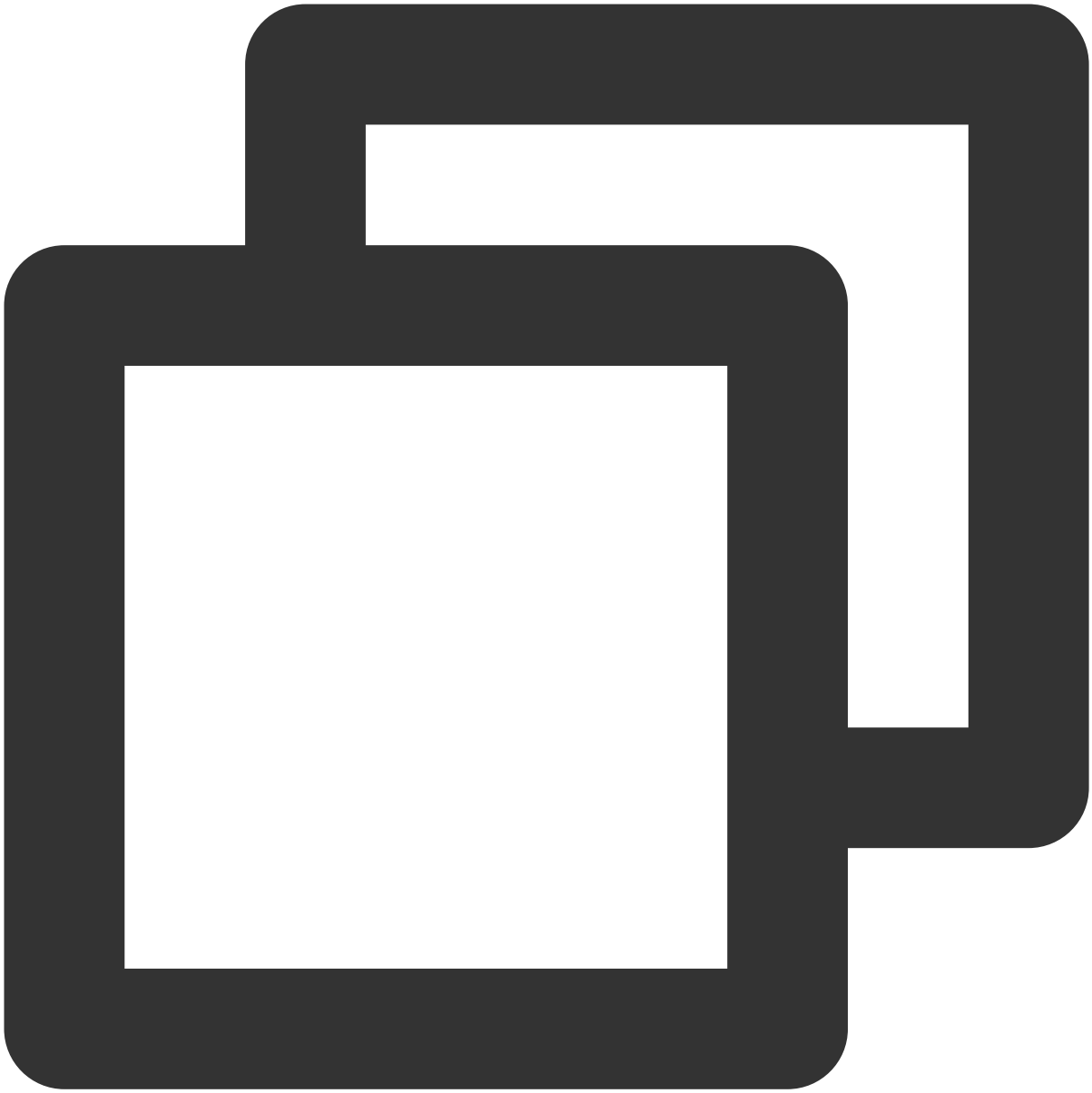
除了在创建用户时能够对其进行权限授予，还能在创建完成后通过 ALTER ROLE 语法对其进行权限的重新赋予。例如：



```
ALTER role jsmith with CREATEROLE;
```

用户组

group，即用户组，是一种特殊的 role，没有赋予 LOGIN 权限，group 通常被设定为经常搭配使用的权限组合，权限可以被作为一个整体授予给某个用户或者从某个用户处取消。
您可以通过使用以下语句来创建一个被授予权限组合的角色，即 group。



```
Create role, Create DB, Cannot login;
```

您还可以很方便地通过 GRANT TO 或者 REVOKE FROM 语句建立或者取消其他用户与该用户组的从属关系，拥有从属关系的用户会从这个用户组继承权限。

GRANT TO 语句示例：

```
gpadmincloud=# GRANT manager TO jsmith;
GRANT ROLE
gpadmincloud=# \du+
                                                   List of roles
 Role name   |                                           Attributes                                                   | Member of | Description
-------------+----------------------------------------------------------------------------------------------------------+-----------+-------------
 gpadmincloud | Superuser, Create role, Create DB, Ext gpfdist Table, Wri Ext gpfdist Table, Ext http Table, Ext hdfs Table, Wri Ext hdfs Table | {}        |
 gpmon       | Superuser, Create DB                                                                                     | {}        |
 jsmith      | Create role                                                                                              | {manager} |
 lambuser    | Create role, Create DB, Ext gpfdist Table                                                                | {}        |
 manager     | Create role, Create DB, Cannot login                                                                     | {}        |
```

用户 jsmith 归属于用户组 manager。

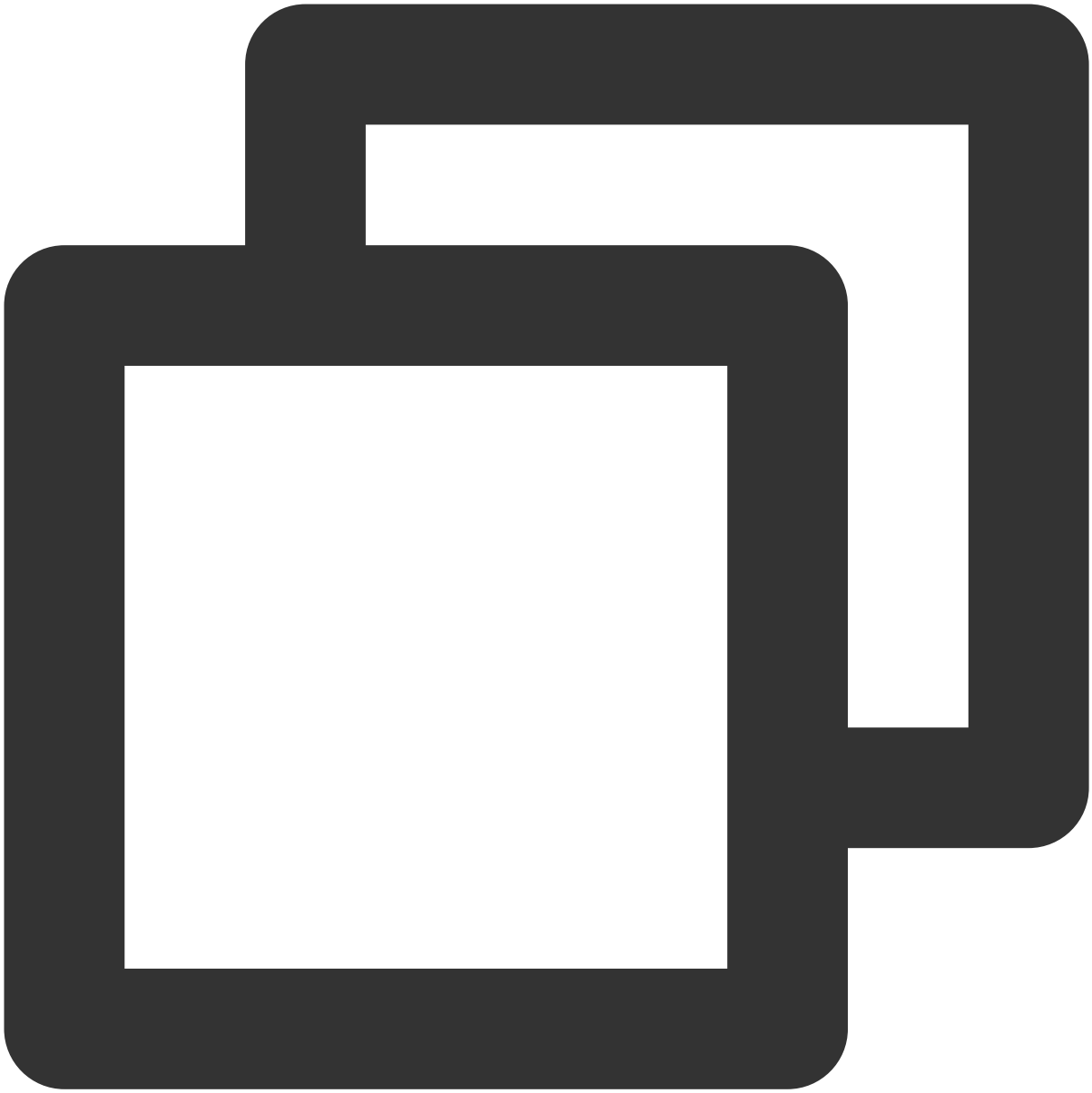
REVOKE FROM 语句示例：

```
gpadmincloud=# \du+
                                                   List of roles
 Role name   |                                           Attributes                                                   | Member of | Description
-------------+----------------------------------------------------------------------------------------------------------+-----------+-------------
 gpadmincloud | Superuser, Create role, Create DB, Ext gpfdist Table, Wri Ext gpfdist Table, Ext http Table, Ext hdfs Table, Wri Ext hdfs Table | {}        |
 gpmon       | Superuser, Create DB                                                                                     | {}        |
 jsmith      | Create role                                                                                              | {}        |
 lambuser    | Create role, Create DB, Ext gpfdist Table                                                                | {}        |
 manager     | Create role, Create DB, Cannot login                                                                     | {}        |

gpadmincloud=#
```

将用户 jsmith 归属于用户组 manager 的关系取消。

对象权限管理

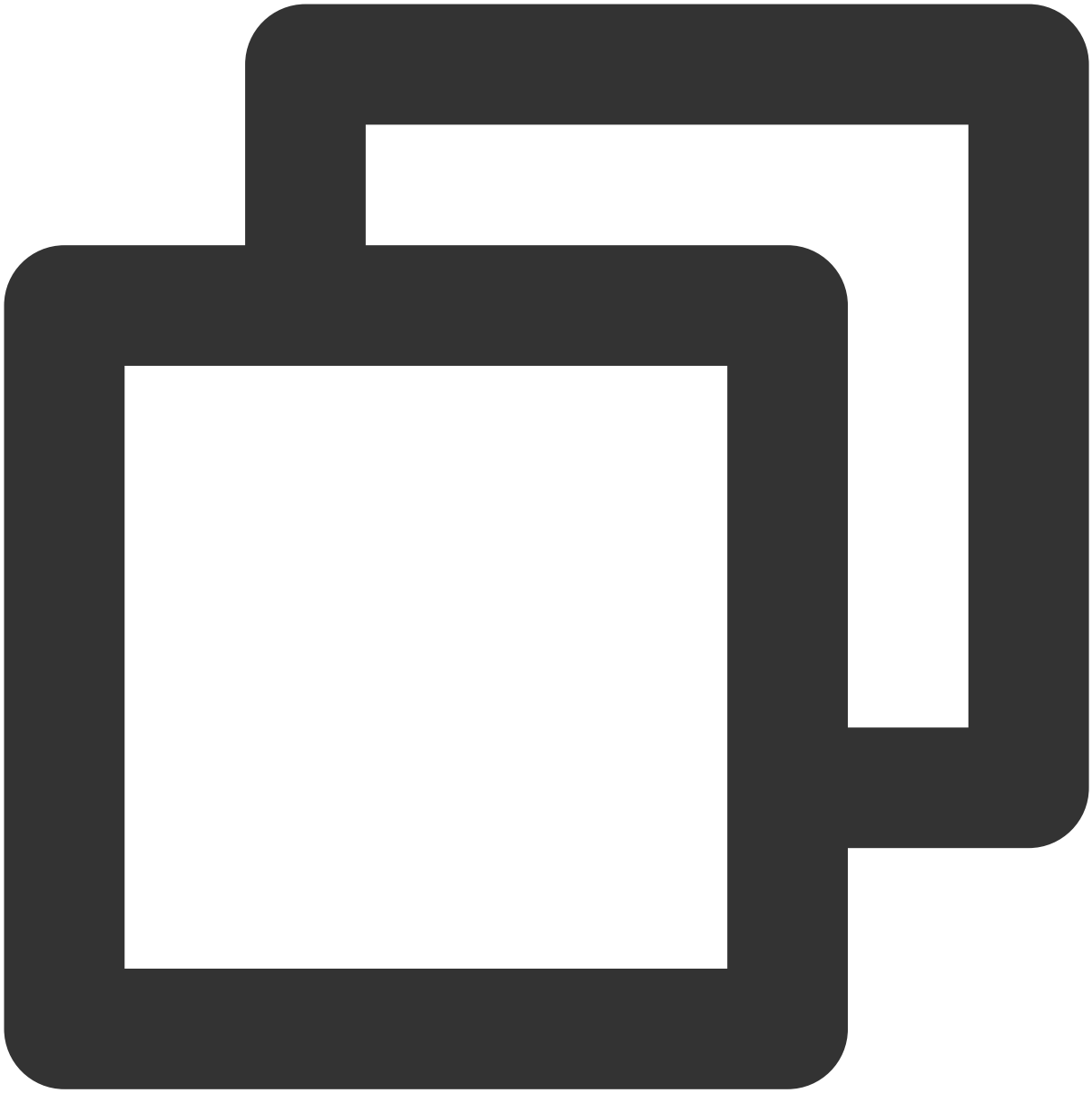
当一个对象（例如 database、table、schema、function 等）被创建时，必然会有一个归属者，这个归属者一般为执行创建对象语句的用户。初始时，只有归属者对这个对象拥有所有的操作权限。例如：



```
GRANT INSERT ON test TO  jsmith;
```

我们可以通过上述语句，将 test 的 INSERT 权限授予 jsmith 用户，同样也可以通过 REVOKE FROM 取消。

同样，也可以通过 REASSIGN OWNED 语句将归属于某个用户的所有对象转移给别的用户，例如：



```
SET ROLE jsmith;    //切换到jsmith用户
CREATE TABLE jsmithtest (age int, id int); //创建新的表
SET ROLE gpadmincloud;          //切回到超级用户
reassign owned by jsmith to lambuser;    //将归属于jsmith的对象都转移给lambuser。
```

由于归属于超级用户的对象不能转移给别的用户，因为有属于系统的对象归属于这个超级对象，因此我们需要使用非超级用户建立一个表。

```
gpadmincloud=# reassign owned by jsmith to lambuser;
REASSIGN OWNED
gpadmincloud=# \d
                List of relations
 Schema |    Name    | Type  |    Owner     | Storage
--------+------------+-------+--------------+----------
 public | data_dir   | table | gpadmincloud | external
 public | jsmithtest | table | lambuser     | heap
 public | test       | table | gpadmincloud | heap
(3 rows)
```

完成对象从 jsmith 到 lambuser 归属权的转移。

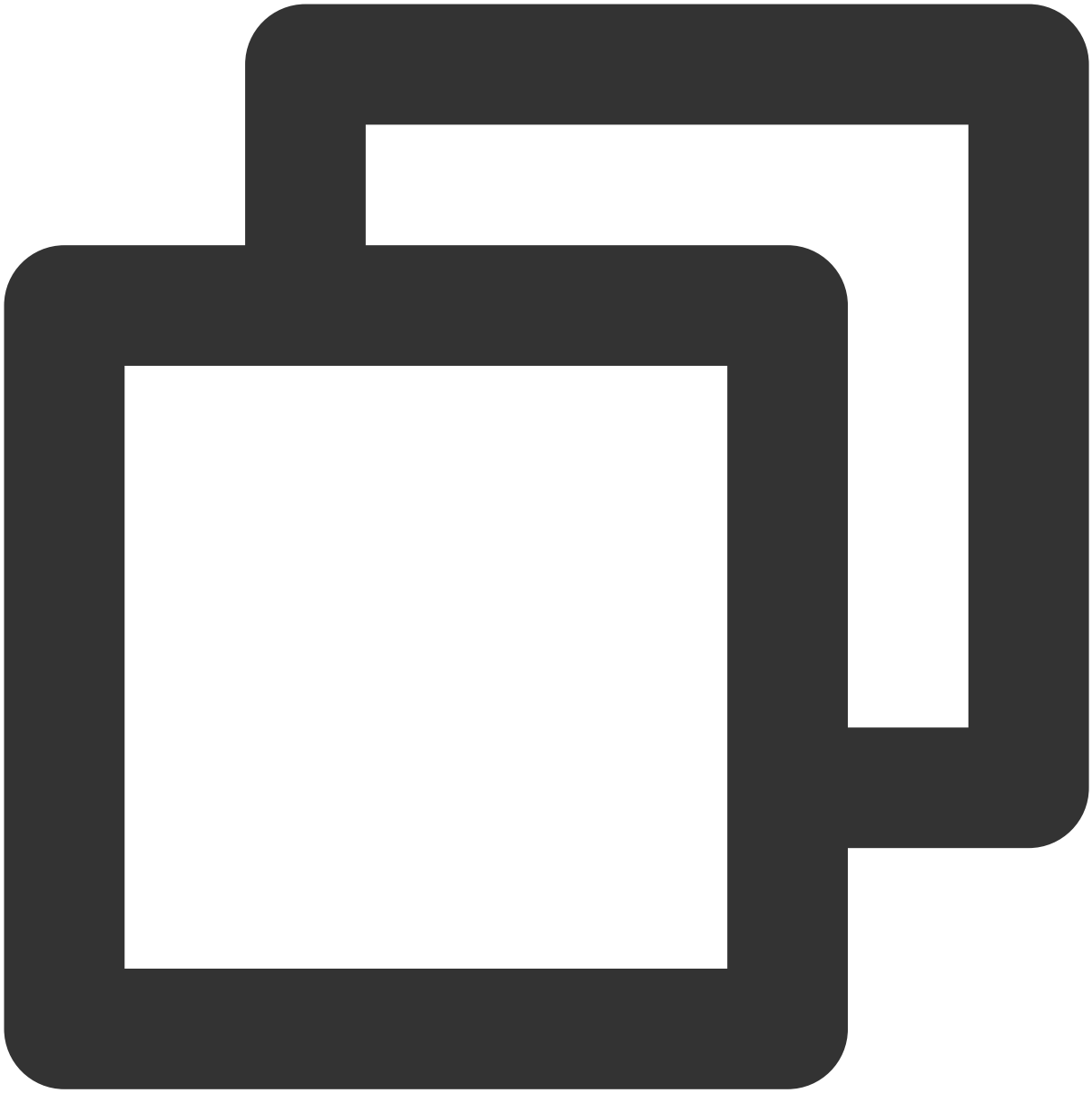
定义数据库

最近更新时间：2024-02-19 10:49:28

数据库创建与管理

在云数据仓库 PostgreSQL 中，您可以创建自己的数据库对象。

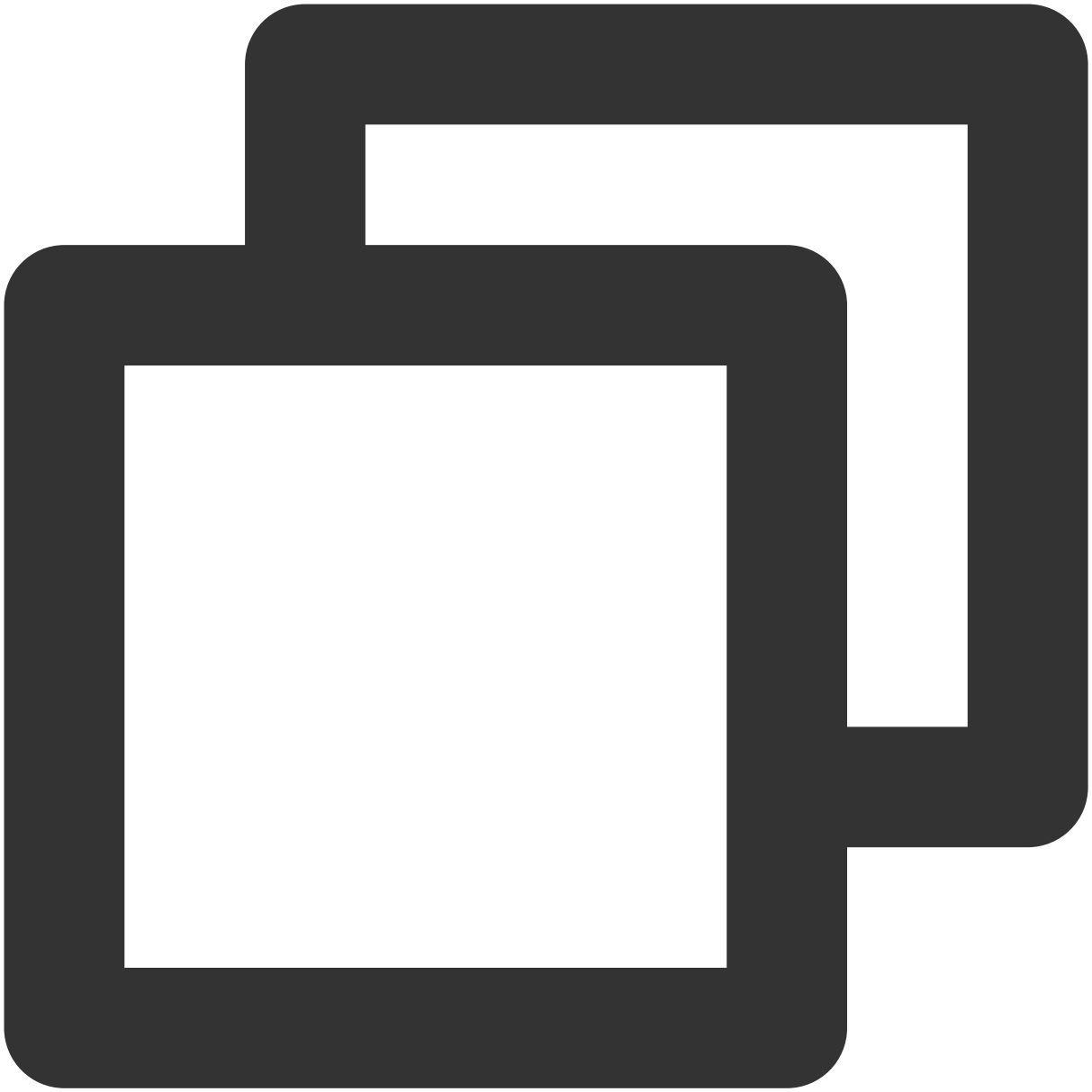
1. 若想要创建自己使用的数据库，首先需要参照 管理用户权限 中的方式创建用户并授权、登录，然后使用 CREATE DATABASE 语句进行数据库的创建，但是在创建数据库之前，必须保证登录的用户拥有 CRAETEROLE 的权限，权限问题参见 管理用户权限。创建数据库示例如下：



```
CREATE DATABASE test;
```

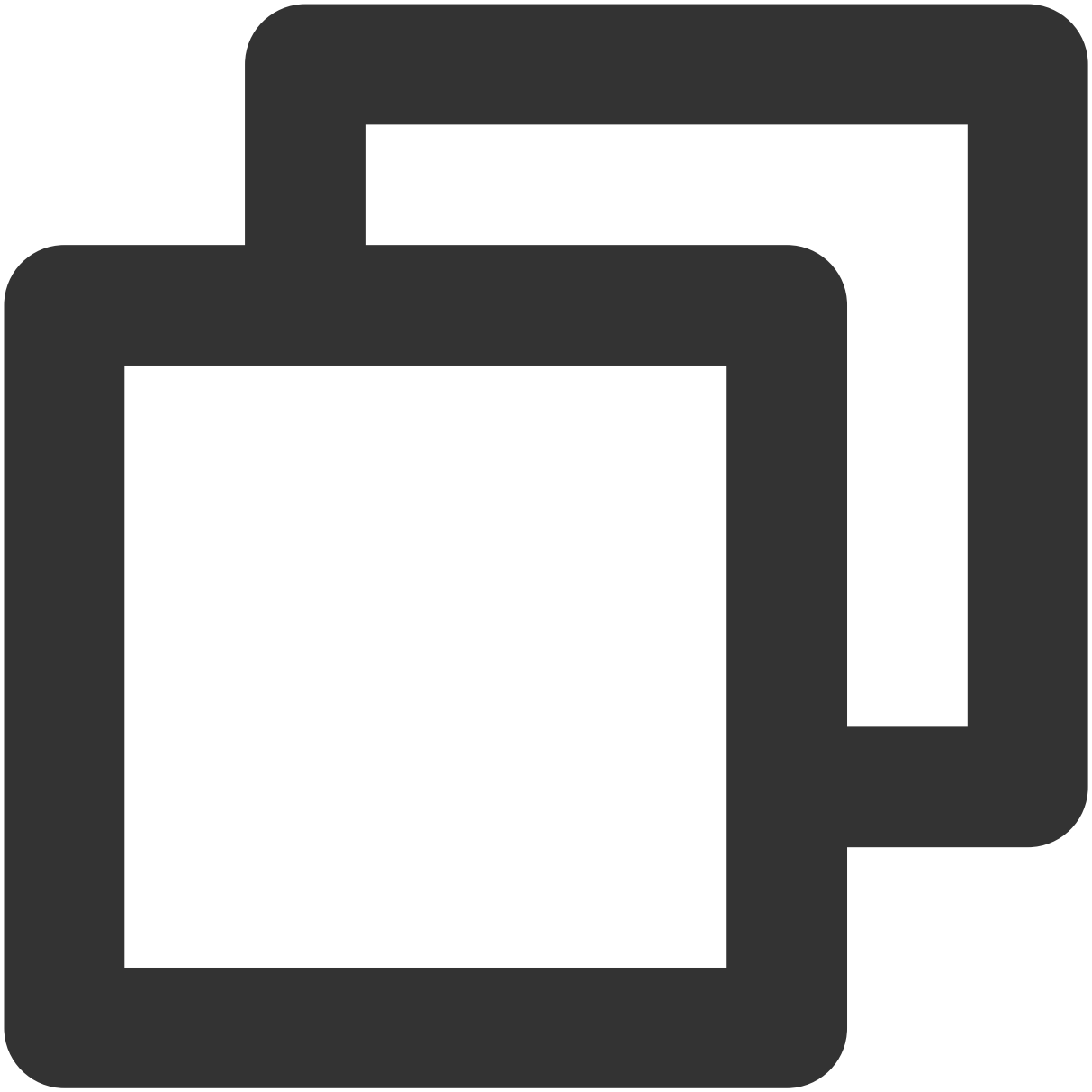
通过 `\\l` 可以列举出所有的数据库。

2. 在创建数据库的过程中，往往会选择一个数据库模板对新数据库进行创建，默认的数据库模板为空。若模板数据库中有任何对象，则新创建的数据库中也会有相应的对象，也可以指定模板创建。例如，先在 test 中使用下列语句创建一个表。



```
create table ttable (age int, id int);
```

然后以 test 为模板创建数据库 test2。



```
CREATE DATABASE test2 TEMPLATE test;
```

切换到 test2 可以看到，test2 中也有 ttable，因此 template1 中尽量不要创建任何对象，否则以 template1 为模板创建出来的数据库会创建相应的对象，可以通过 `\\d` 查看选中数据库中的所有表。

```
test=# create database test2 template test;
CREATE DATABASE
test=# \c test2
You are now connected to database "test2" as user "gpadmincloud".
test2=# \d
               List of relations
 Schema |  Name  | Type  |    Owner     | Storage
--------+--------+-------+--------------+---------
 public | ttable | table | gpadmincloud | heap
(1 row)
```

3. 可以通过 `\\l` 来列出所有的数据库。

```
test2=# \l
                          List of databases
   Name    |    Owner     | Encoding |       Access privileges
-----------+--------------+----------+-------------------------------
 gpperfmon | gpadmincloud | UTF8     | gpadmincloud=CTc/gpadmincloud
                                     : =c/gpadmincloud
 postgres  | gpadmincloud | UTF8     |
 template0 | gpadmincloud | UTF8     | =c/gpadmincloud
                                     : gpadmincloud=CTc/gpadmincloud
 template1 | gpadmincloud | UTF8     | =c/gpadmincloud
                                     : gpadmincloud=CTc/gpadmincloud
 test      | gpadmincloud | UTF8     |
 test1     | gpadmincloud | UTF8     |
 test2     | gpadmincloud | UTF8     |
(7 rows)
```

4. 可以通过 DROP DATABASE 删除数据库，进行删除操作时，必须确保登录的用户为超级用户或者是具有删除数据库权限的普通用户，并且，只有当该数据库的连接数为0时才能被删除。例如：

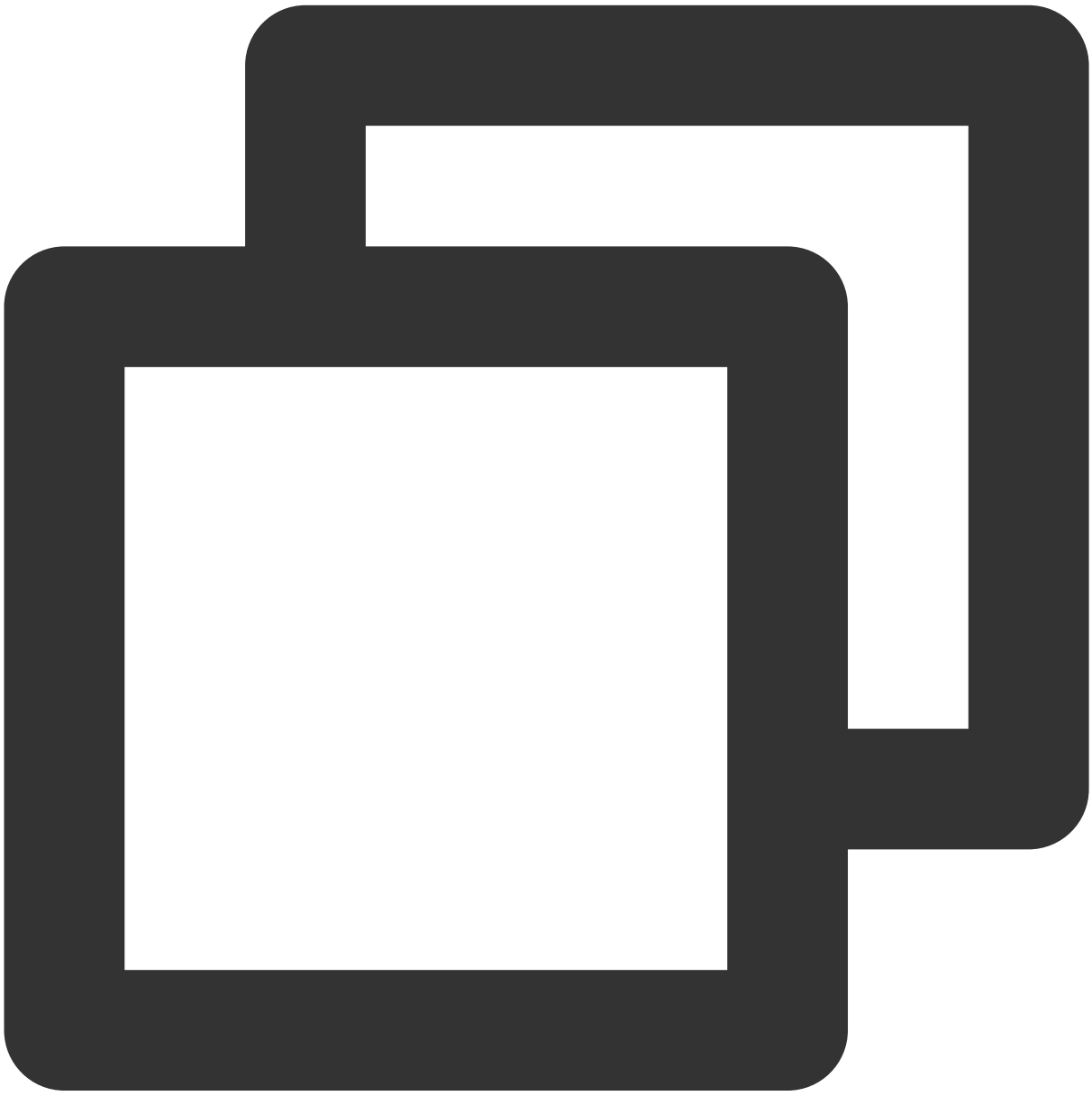
```
test2=# DROP DATABASE test2;
ERROR:  cannot drop the currently open database
test2=# \c test
You are now connected to database "test" as user "gpadmincloud".
test=# DROP DATABASE test2;
DROP DATABASE
```

可以看到当用户选择 test2 时，这时 test2 的连接数必然大于等于1，切换到 test 后，才能正常删除 test2。

模式创建与管理

在云数据仓库 PostgreSQL 中，模式 Schema 作为一个逻辑概念，是为了对数据库空间进行更加详细的划分而存在的，每个数据库在建立时拥有一个名为 public 的模式。在一个数据库中，不能创建同名的表，但是如果选择在不同的 schema 中创建同名的表，则是被允许的，数据库系统是以 database.schema.table 的形式来标识一个表的，此外，不同的数据库下可以创建同名的 schema。

1. 创建模式。

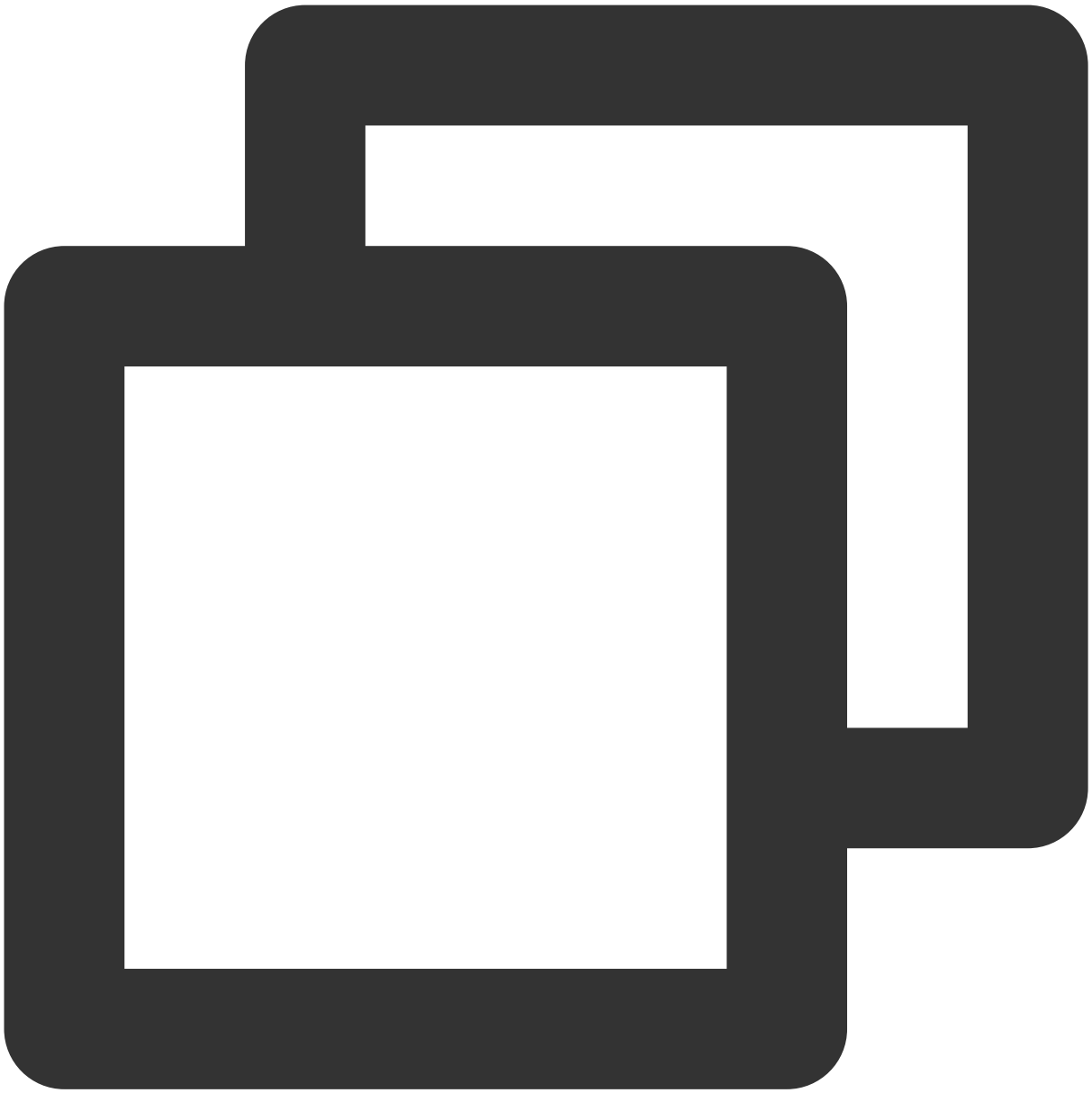


```
CREATE SCHEMA testschema;
```

2. 指定模式创建对象。

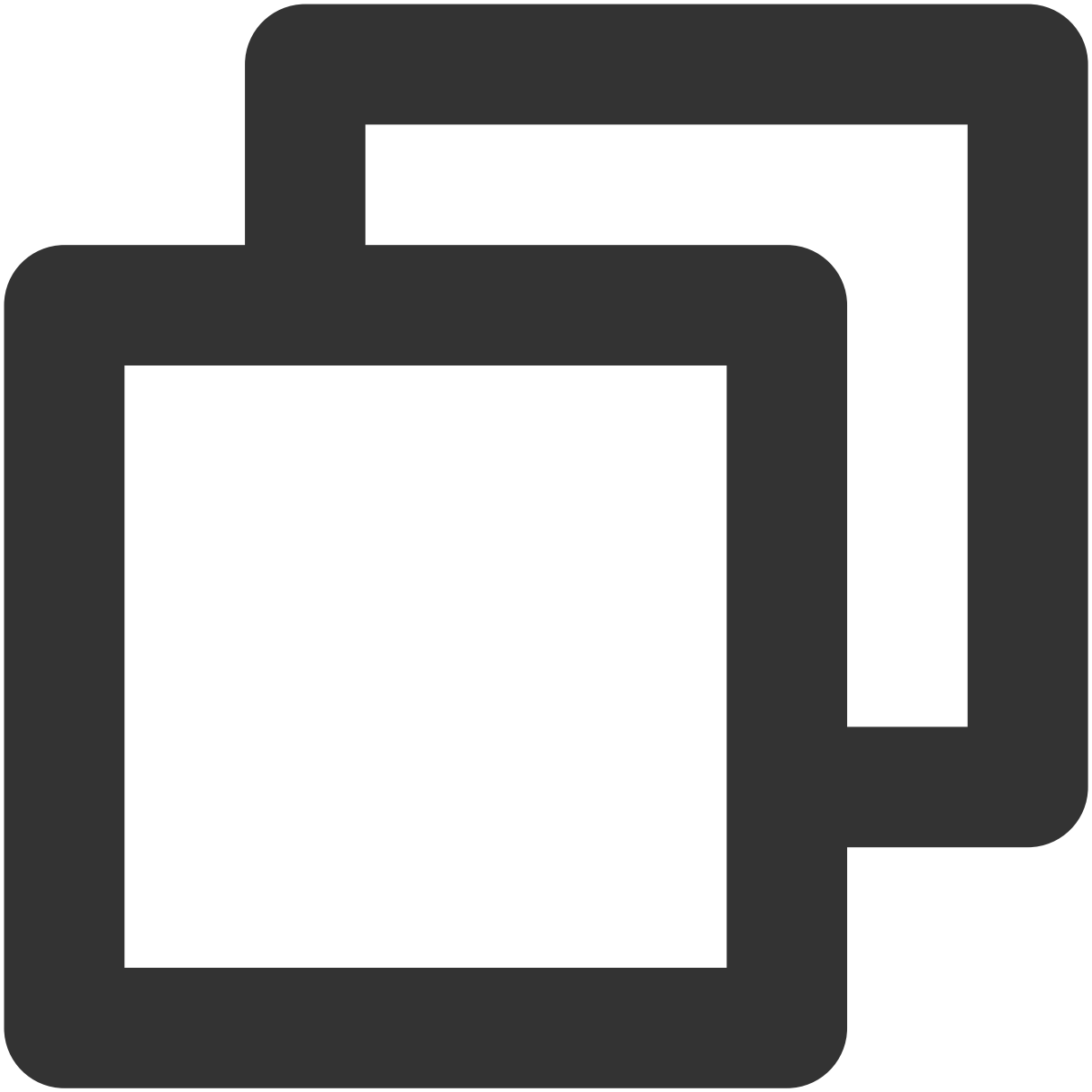
创建表、函数等对象时，可以加上模式前缀来表示在不同模式下创建对象，不加时则默认表示在 public 下创建，例

如：



```
CREATE TABLE testschema.test;
```

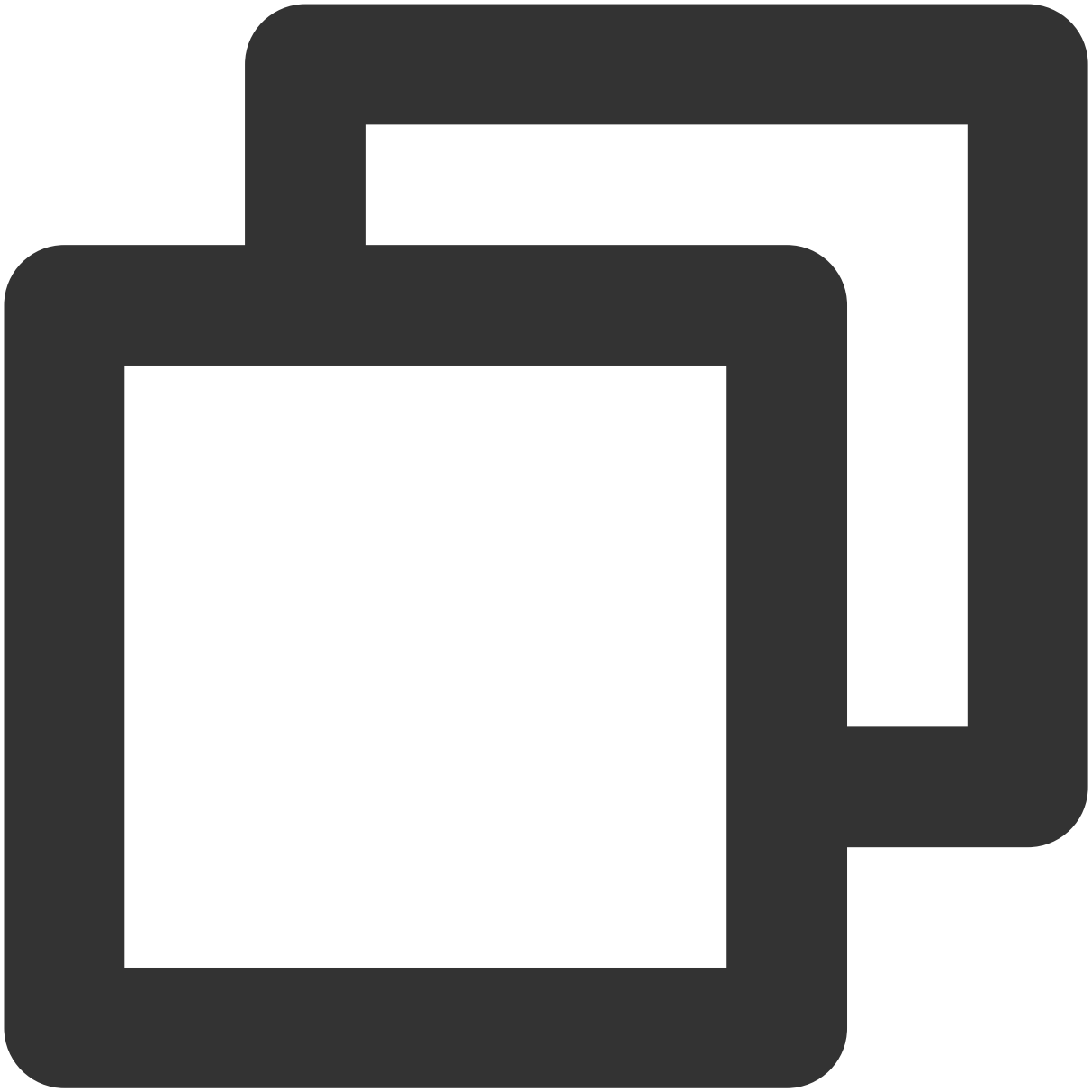
3. 设置模式的优先级。



```
ALTER DATABASE test set search_path to testshcema,public;
```

设置了数据库 test 的 testschema 为 public 模式，表示优先级最高的模式为 testschema，不加任何模式前缀时，首先匹配 testschema。

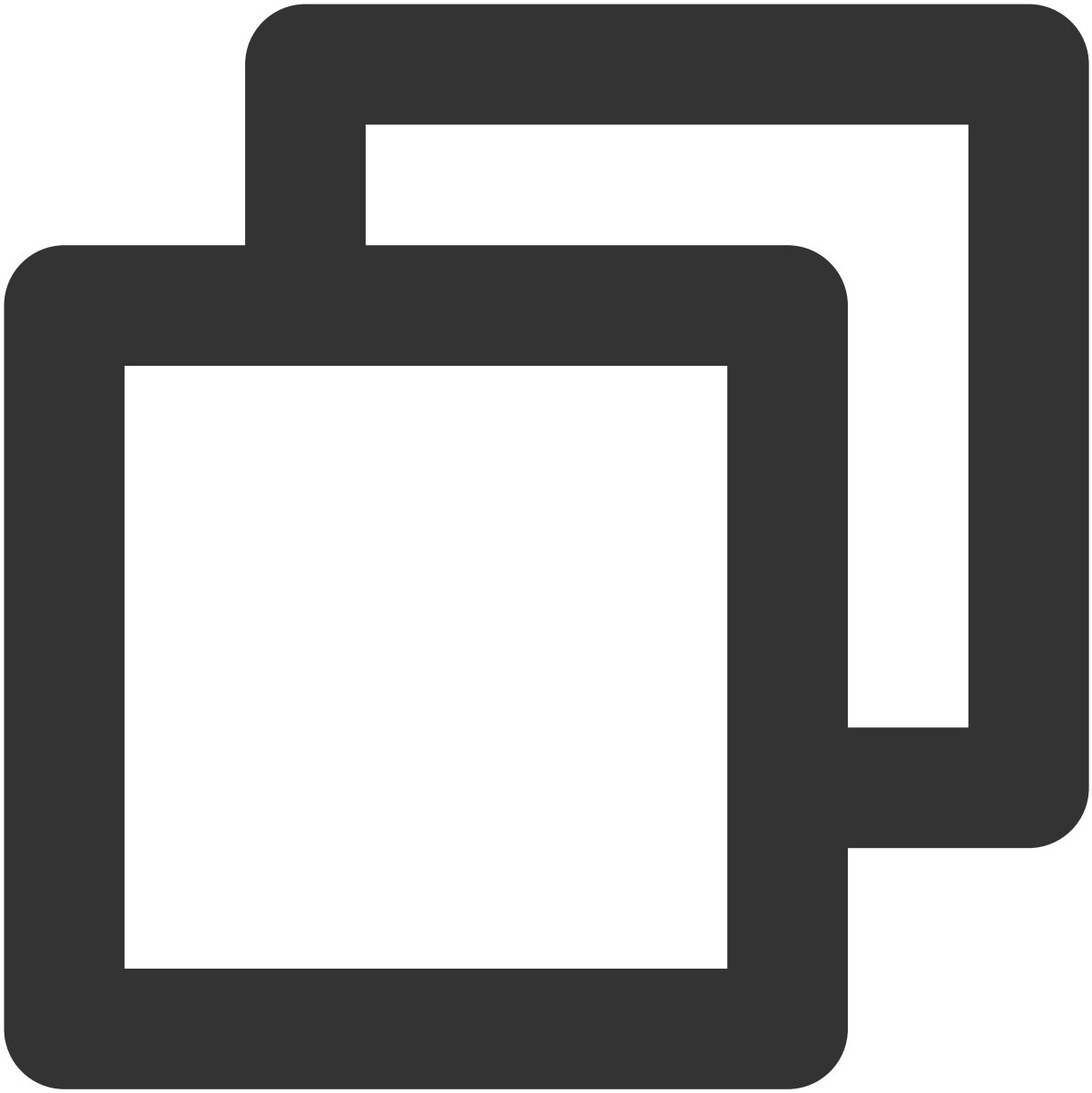
4. 模式的切换。



```
SET search_path TO public;
```

如果当前处于 testschema 模式，则可以通过该语句将模式切换到 public 下面。

5. 删除模式。

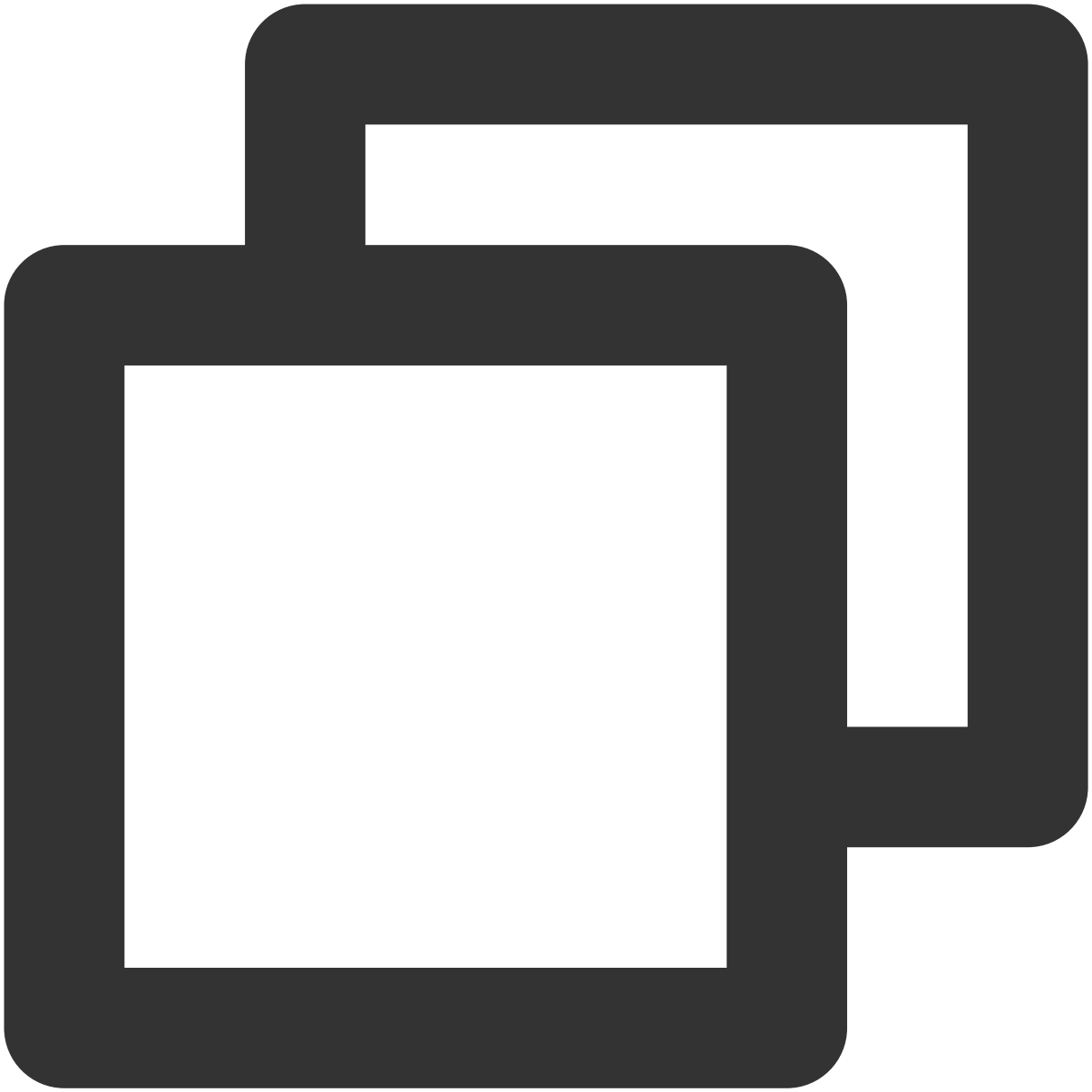


```
DROP SCHEMA testschema;
```

表创建与管理

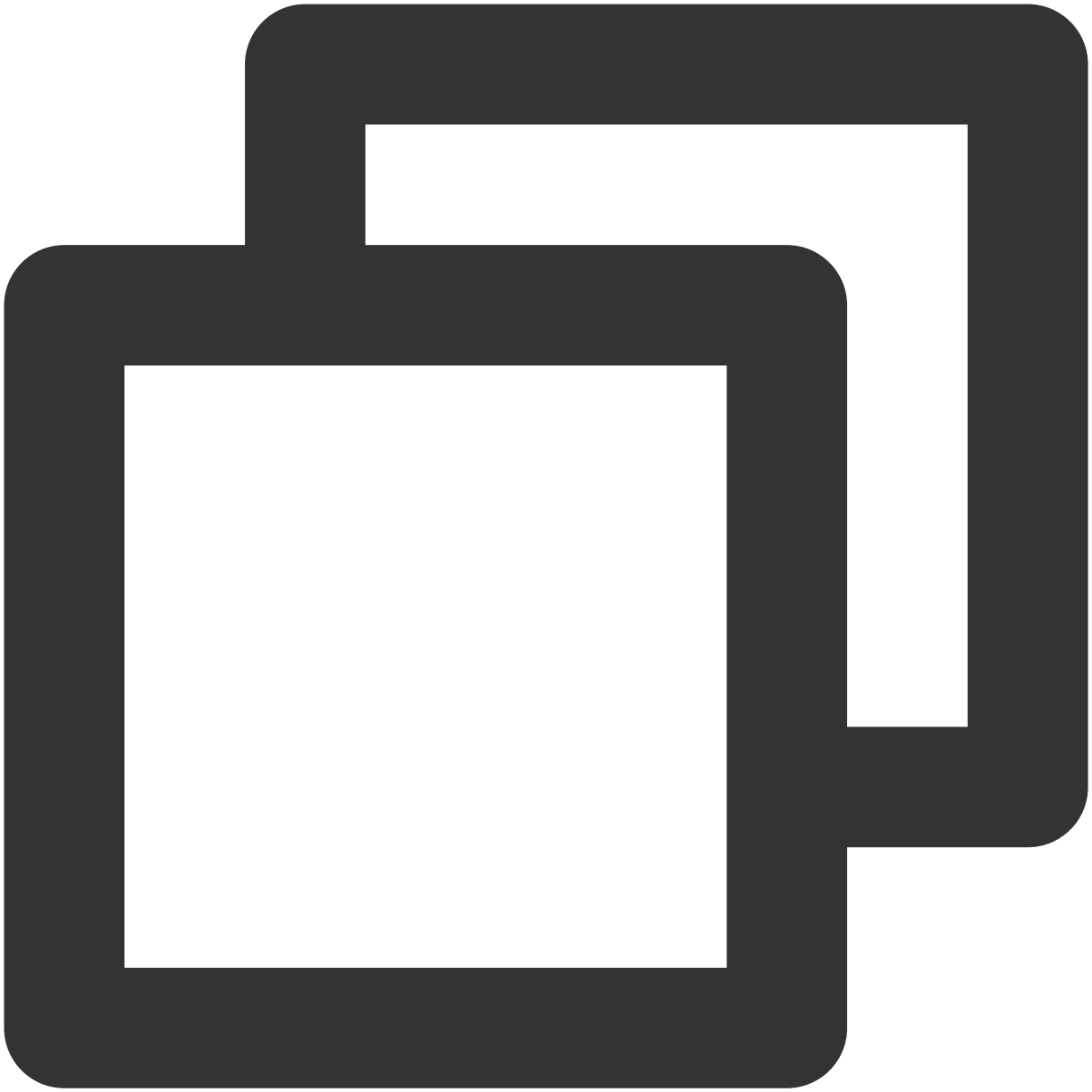
1. 设置表与列约束。

CHECK 约束，此约束可以指定某列数据必须满足某个表达式，例如：



```
CREATE TABLE products (product_no int, name text, price int CHECK(price > 0));
```

NOT NULL 约束，此约束可以指定某列数据不能为空，例如：

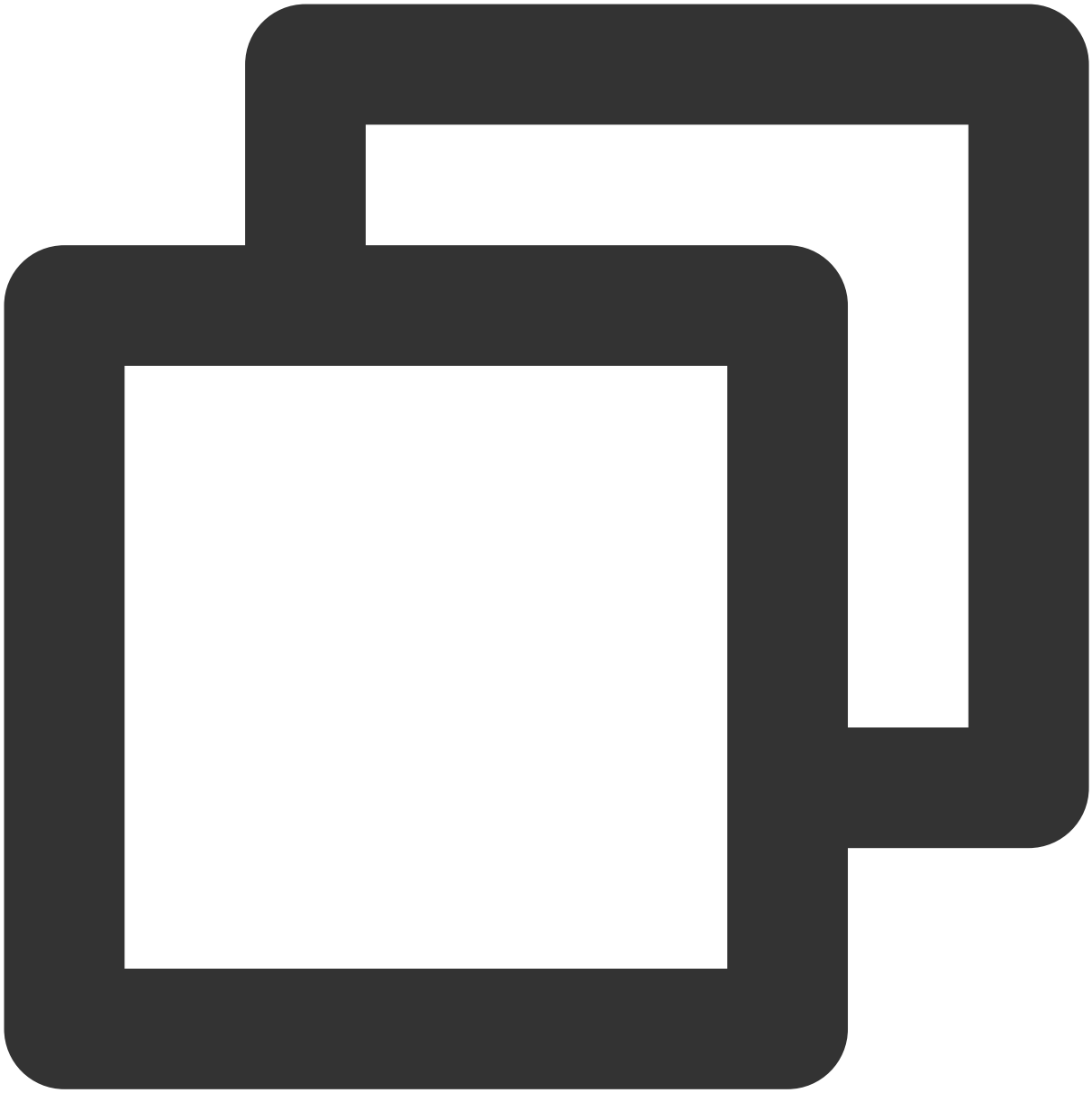


```
CREATE TABLE products (product_no int NOT NULL, name text NOT NULL, price int CHECK
```

2. 数据分布策略。

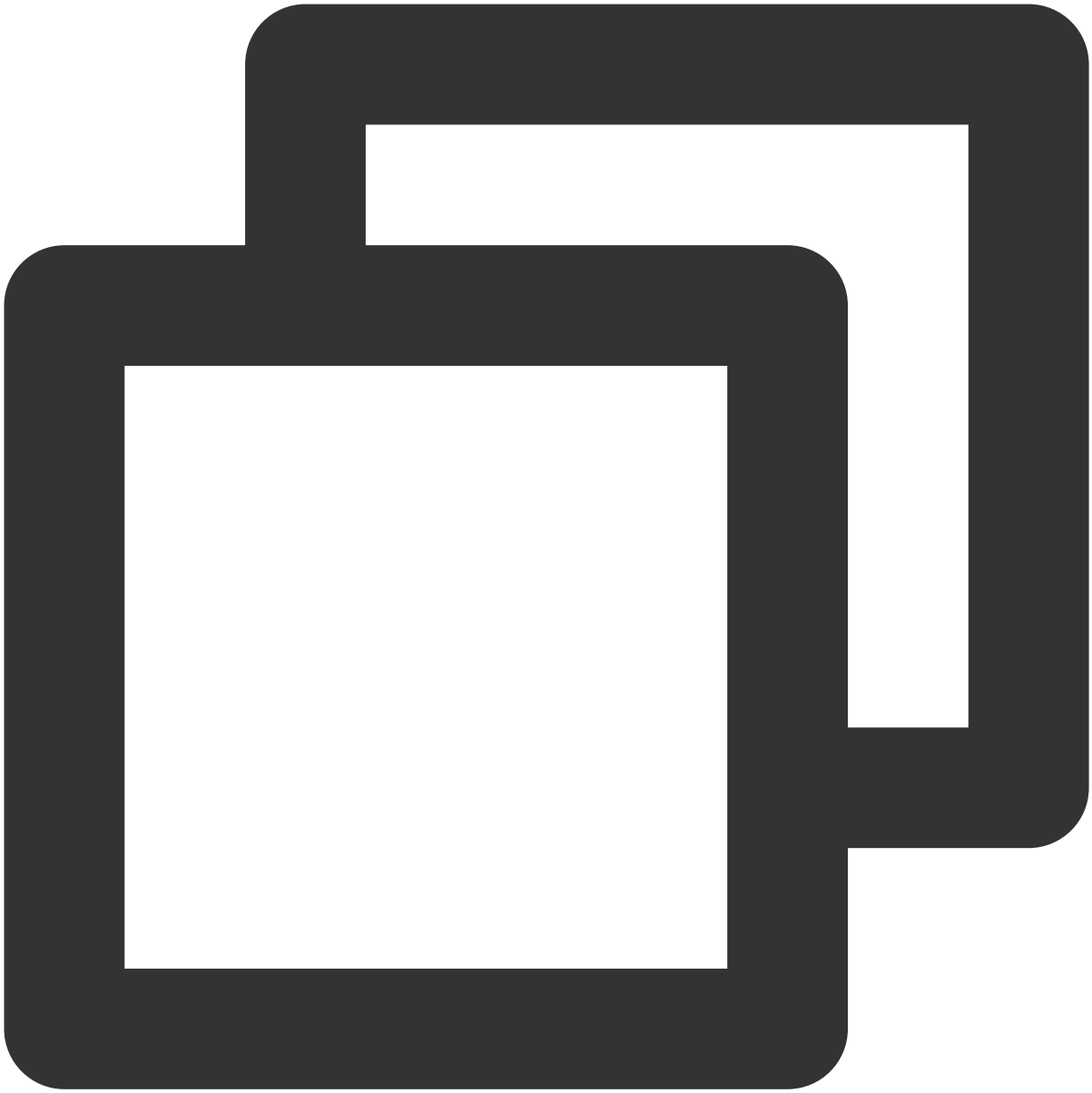
对于分布式数据库仓库来说，每个节点中存储的数据量相同时能够获得最好的处理性能。如果数据分布不平衡，那么数据量多的节点将会使用更多的时间去完成查询过程，从而导致整个查询过程性能的降低。

哈希分布，使用 DISTRIBUTED BY 语法可以在创建表时指定哈希分布的方式，哈希分布的方式会将指定作为哈希分布的所有键进行组合，通过 hash 算法决定数据分布结果。语句如下：



```
CREATE TABLE test (id int, age int) DISTRIBUTED BY (id);
```

随机分布，使用 DISTRIBUTED RANDOMLY 语法可以在创建表时指定为随机分布的方式，顾名思义，通过随机的方式来决定数据分布结果。语句如下：



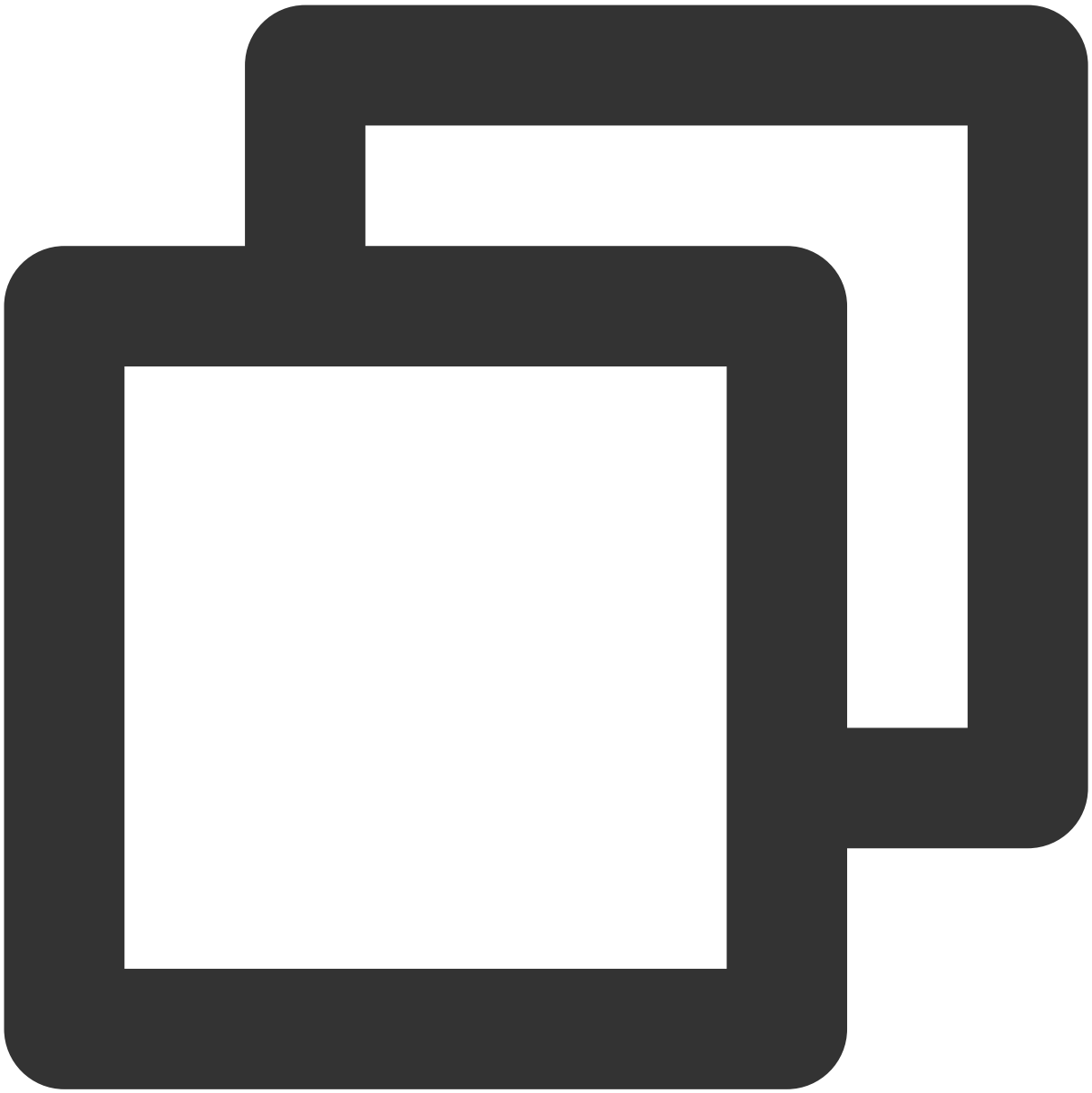
```
CREATE TABLE test (id int, age int) DISTRIBUTED RANDOMLY;
```

其中，拥有 PRIMARY KEY 或者 UNIQUE 的列必须指定其中一种数据分布策略，对于没有 PRIMARY KEY 和 UNIQUE 属性的列，将默认以第一列作为数据分布的参考，默认数据分布策略是哈希分布策略。

视图创建与管理

视图是一个逻辑上的概念，与表不同的是，视图在硬盘上没有实际的数据结构与之对应。

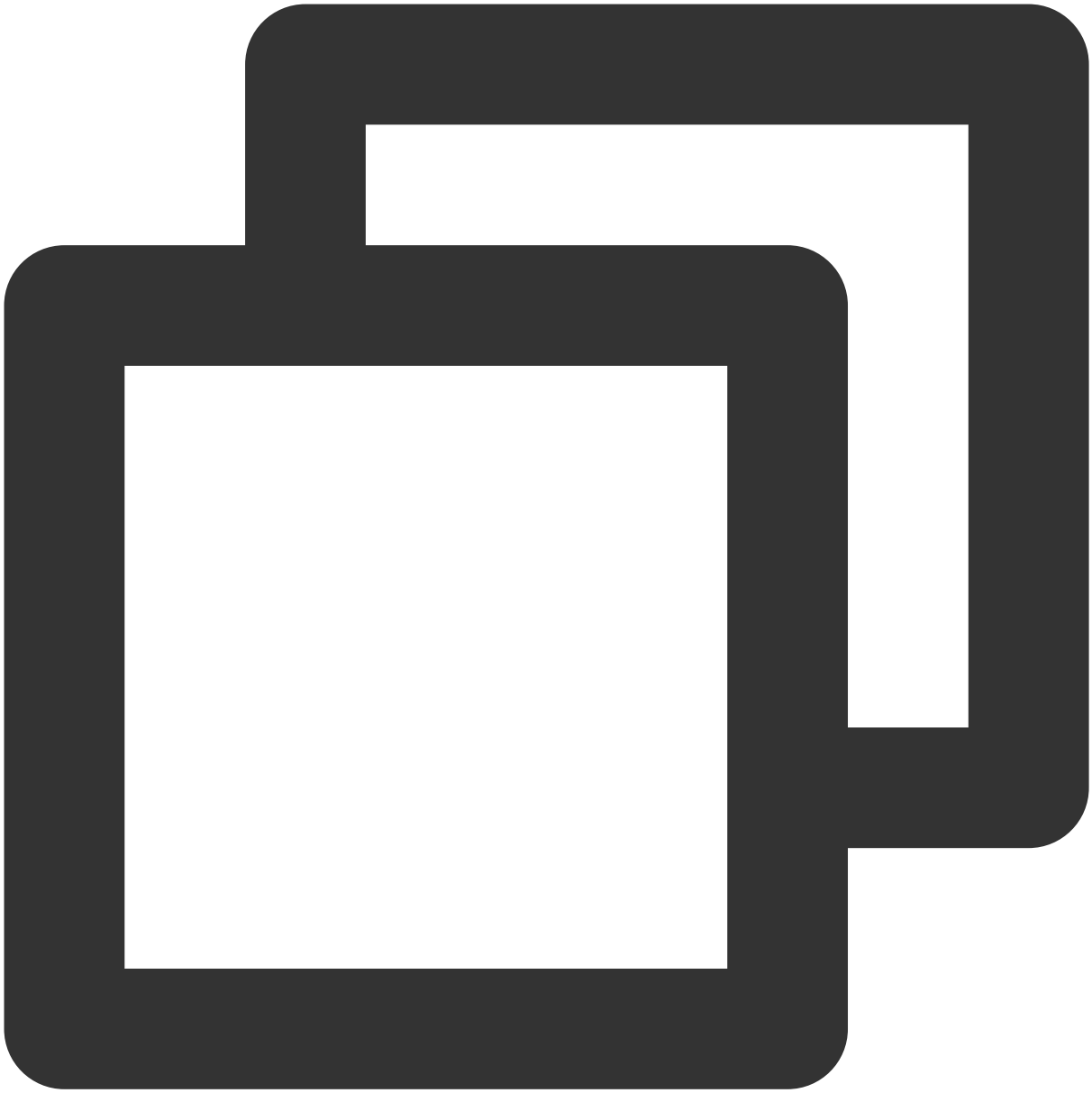
1. 创建视图。



```
CREATE VIEW testview AS SELECT * FROM ttable where age=28;
```

以 ttable 中满足“age=28”这个条件的所有行建立视图 testview。

2. 删除视图。



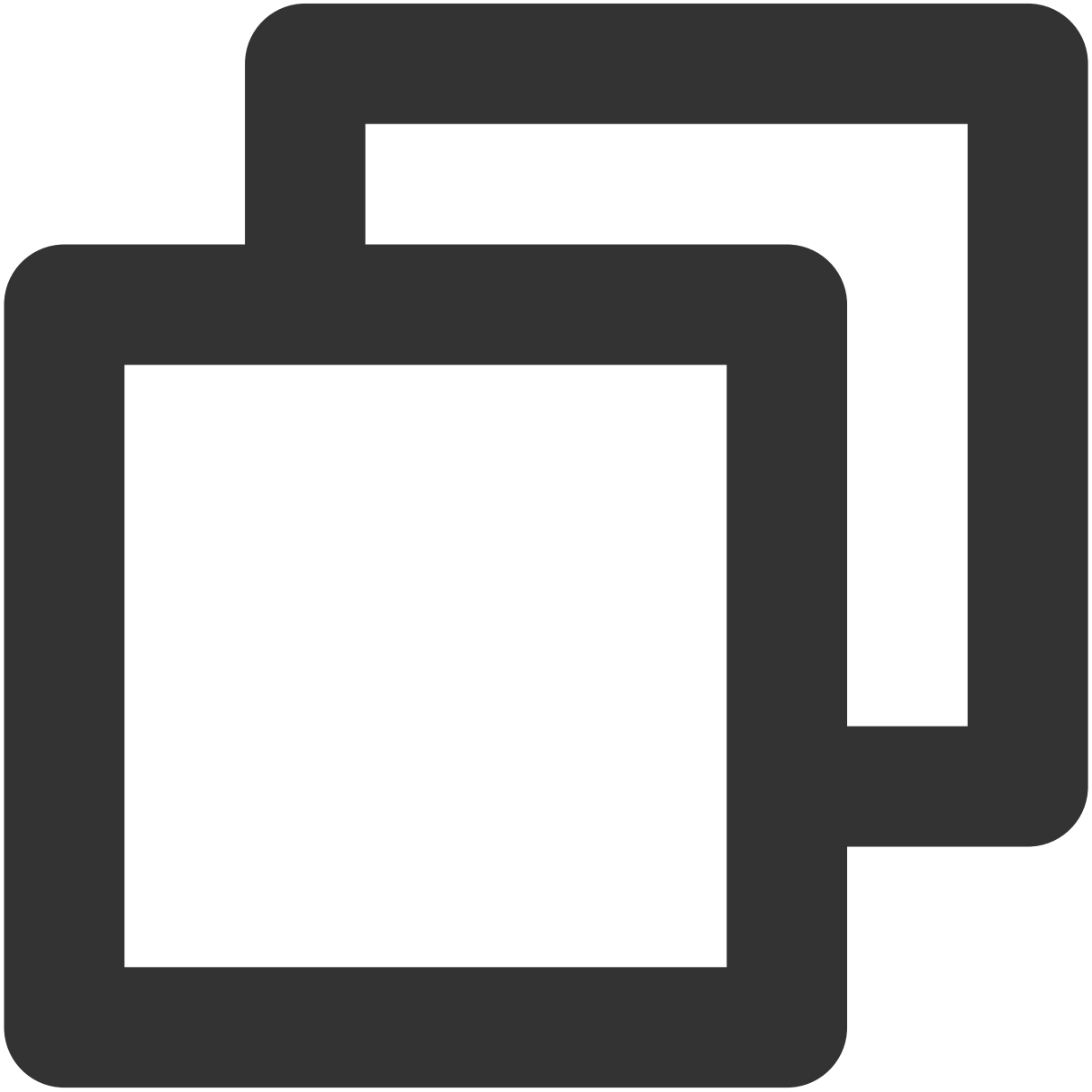
```
DROP VIEW testview;
```

管理数据

最近更新时间：2024-02-19 10:49:28

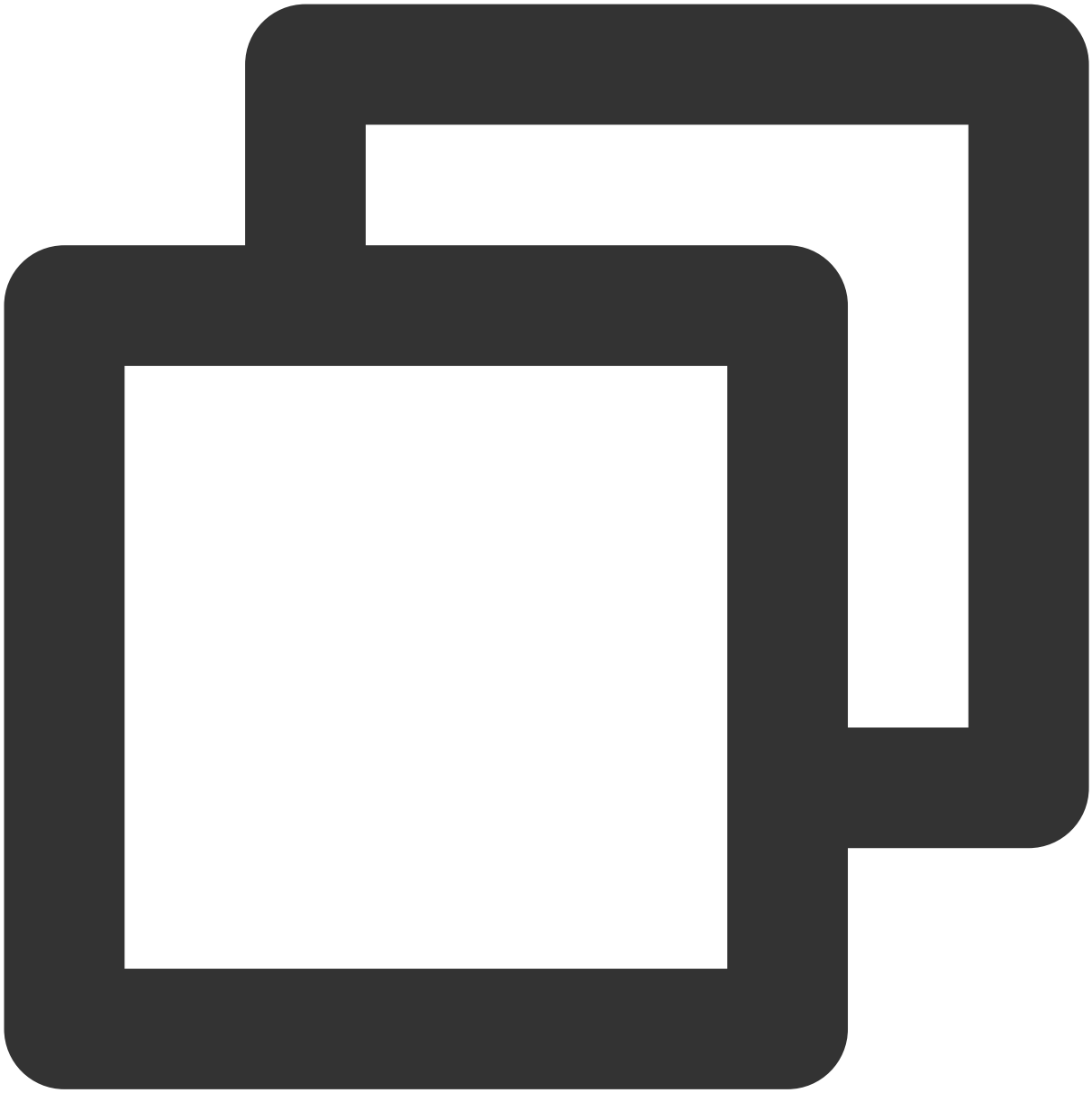
插入数据

1. 插入与列名对应的数据。



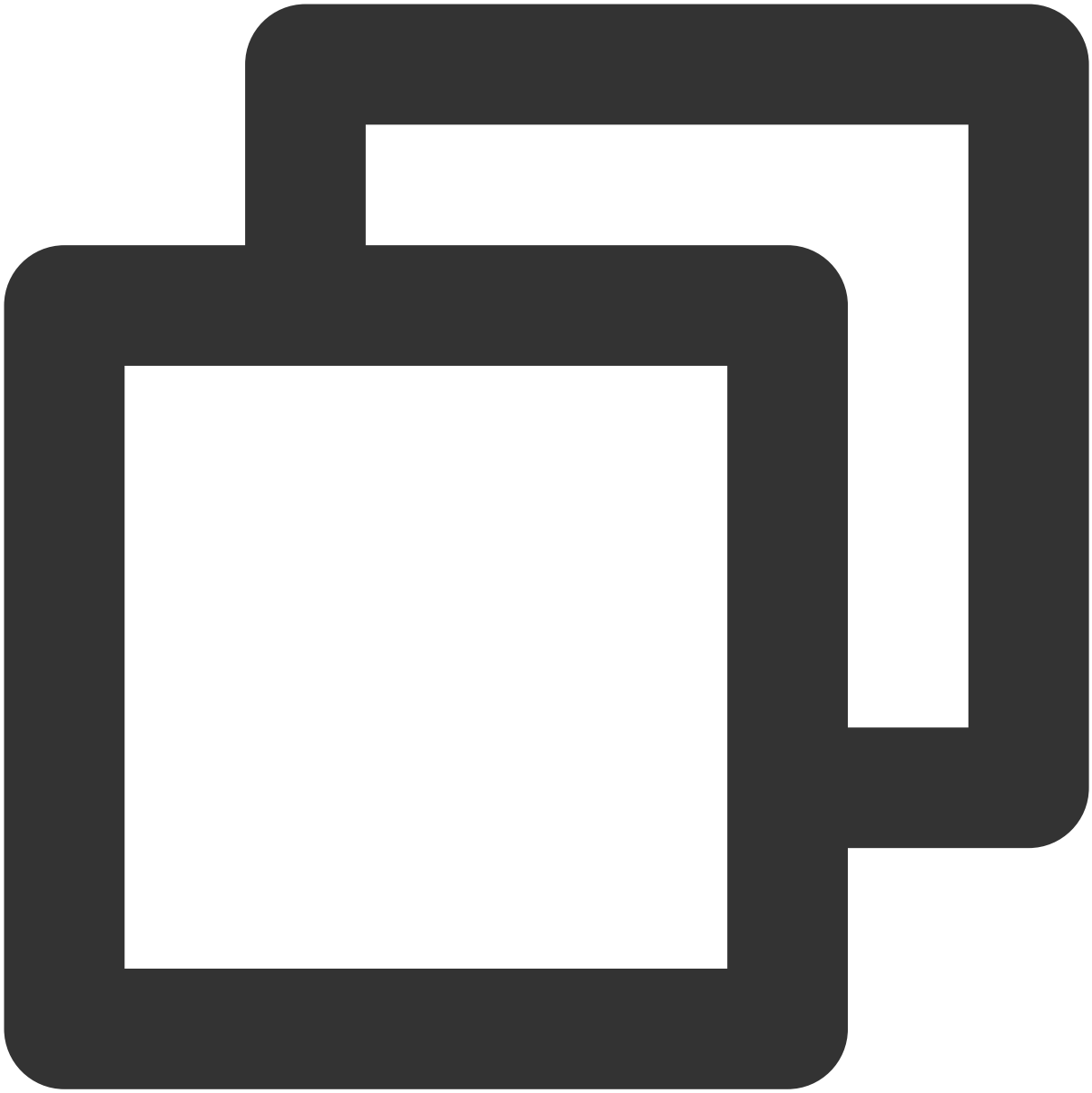
```
INSERT INTO products (name, price, product_no) VALUES ('cheese', 99, 1);
```

2. 按照表中定义列名顺序插入数据。



```
INSERT INTO products VALUES(2, 'chesse', 99);
```

3. 一次性插入多条数据。

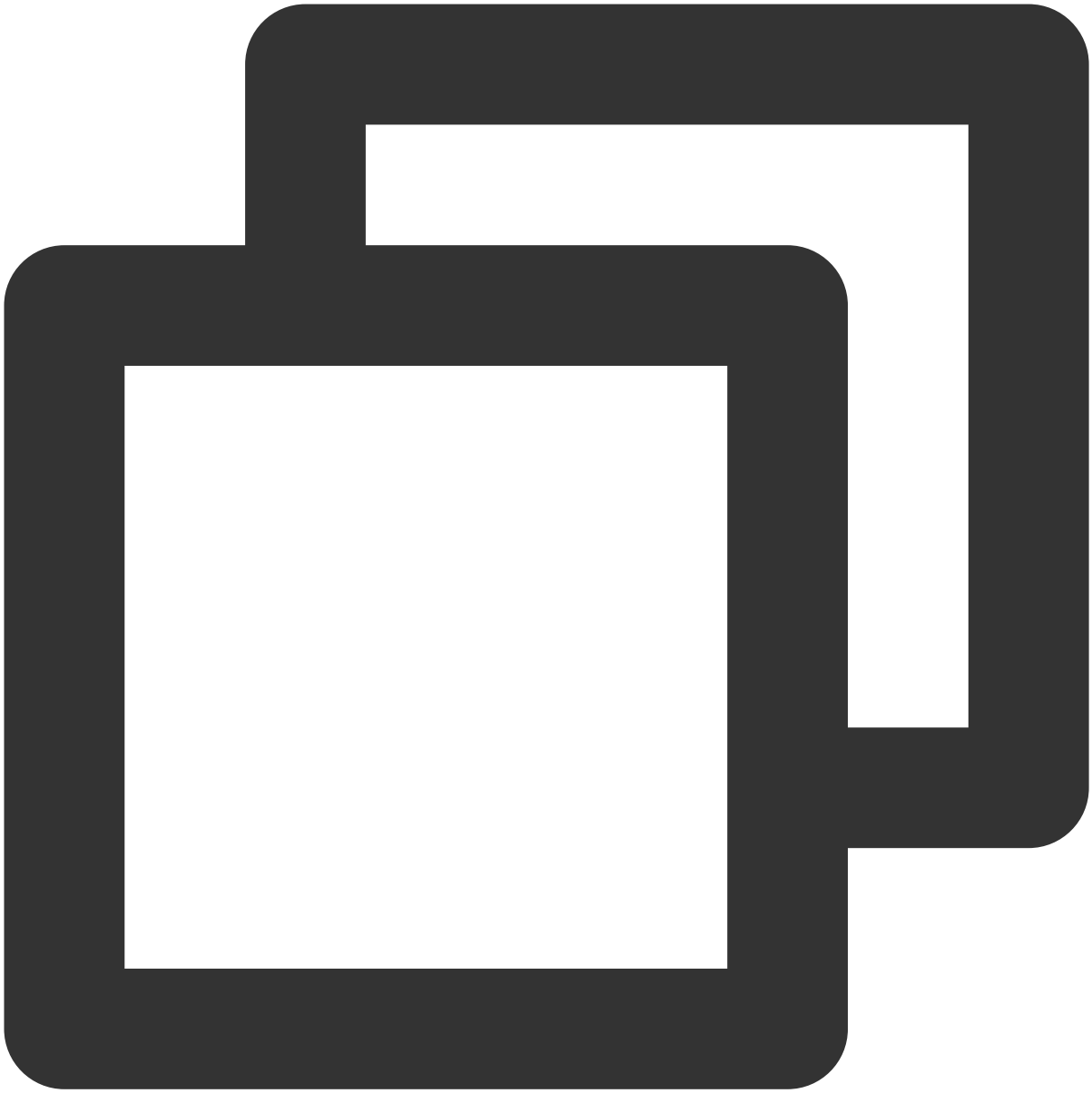


```
INSERT INTO products VALUES (3, 'a', 1), (4, 'b', 2), (5, 'c', 3);
```

4. 通过外表导入，详细可以参考 使用外表 查看外表导入的方法。

5. 通过插件从 TencentDB 中导入，详细可以参考 导入外部数据。

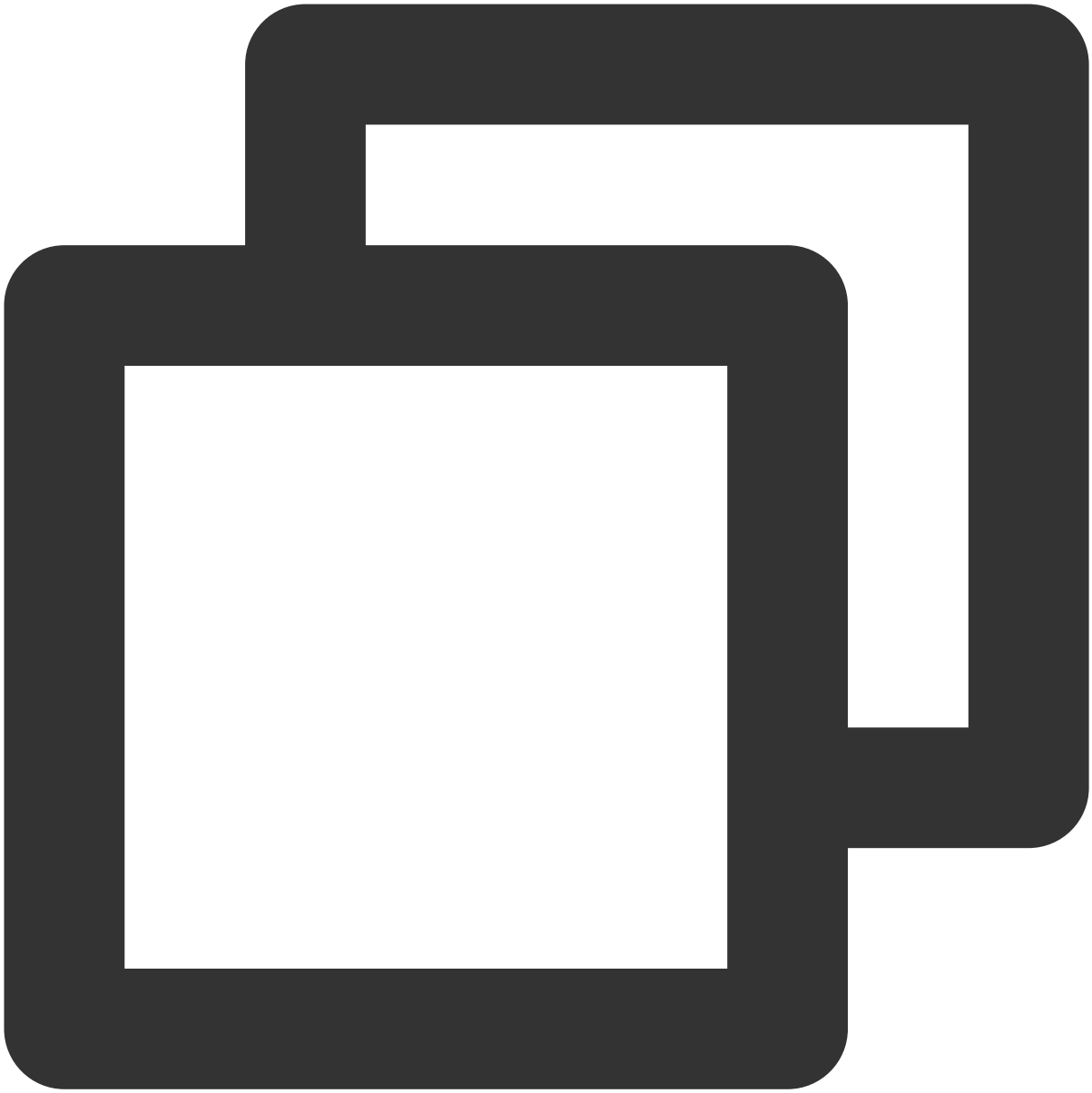
6. 通过 COPY 命令插入。首先您需要登录数据库系统，选定数据库，创建对应的表，然后使用 copy 命令从指定的文件 filename 中以指定的分隔符','将数据插入到 tablename 中。命令如下：



```
COPY tablename  FROM 'filename'  WITH DELIMITER ',';
```

更新数据

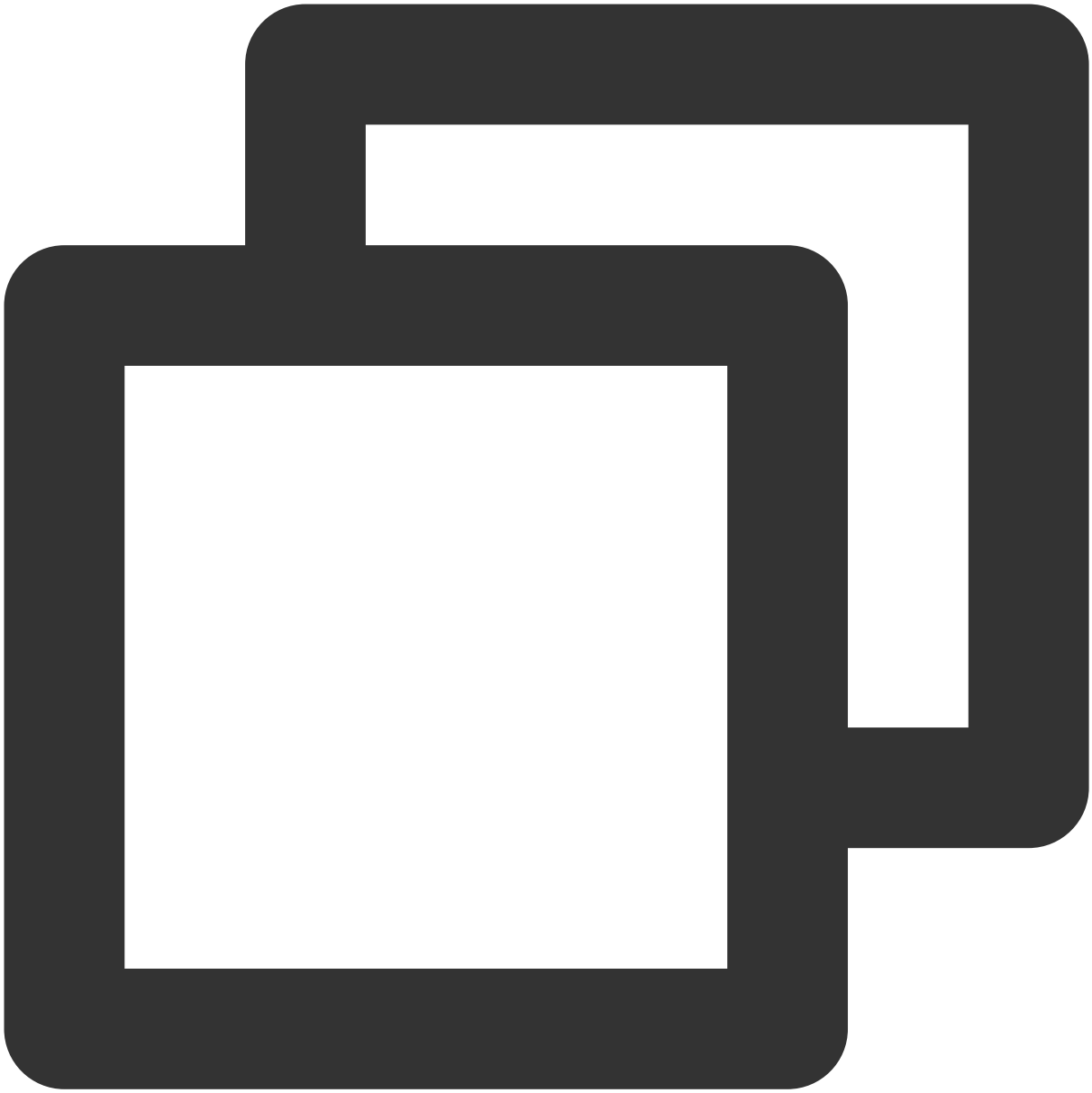
将满足 where 条件的行对应的列的数据更新为指定的值，示例如下：



```
UPDATE products SET price = 10 where product_no = 3;
```

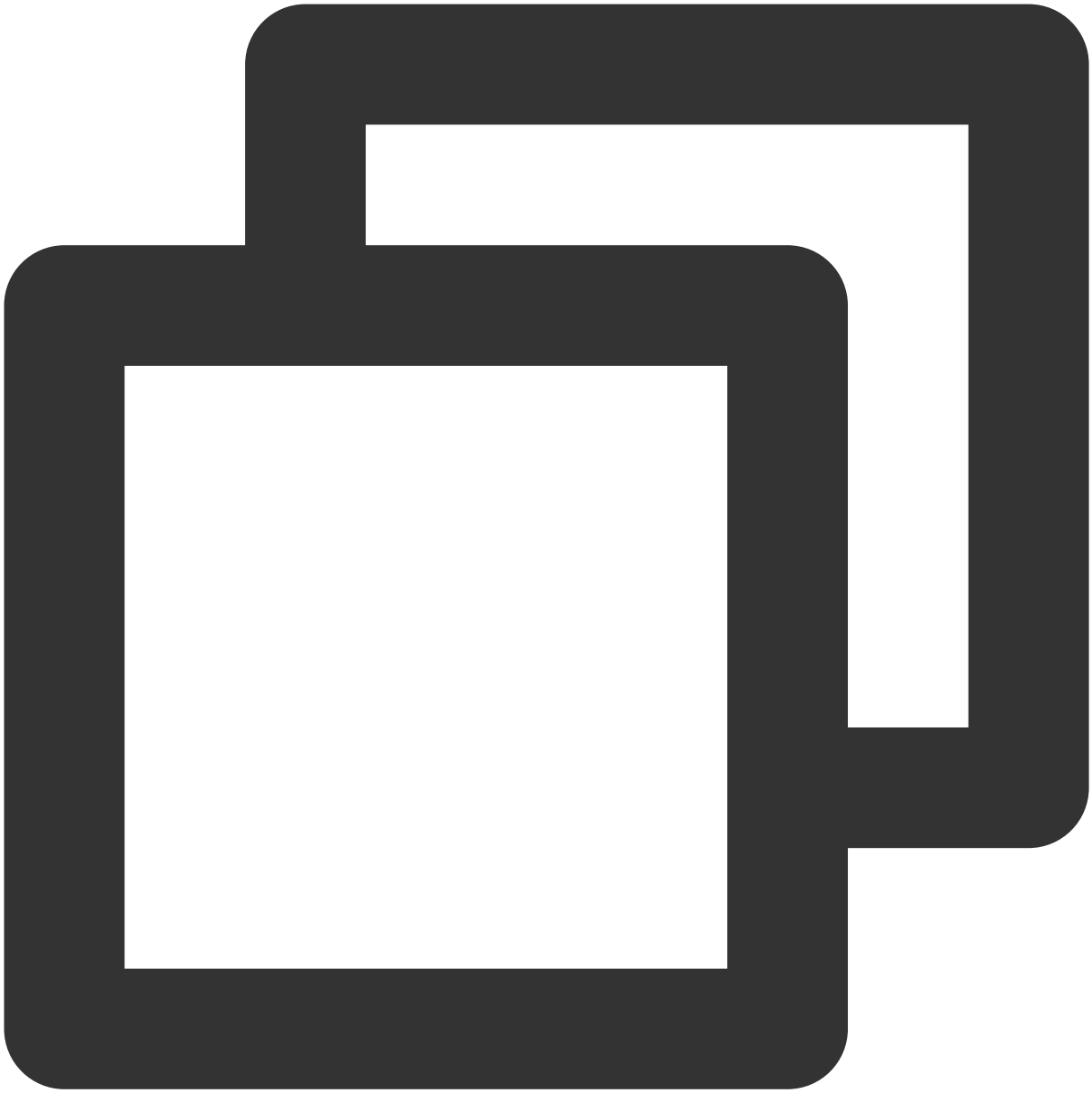
删除数据

将满足 where 条件的行删除，示例如下：



```
DELETE FROM products where price = 3;
```

删除表中所有数据，示例如下：

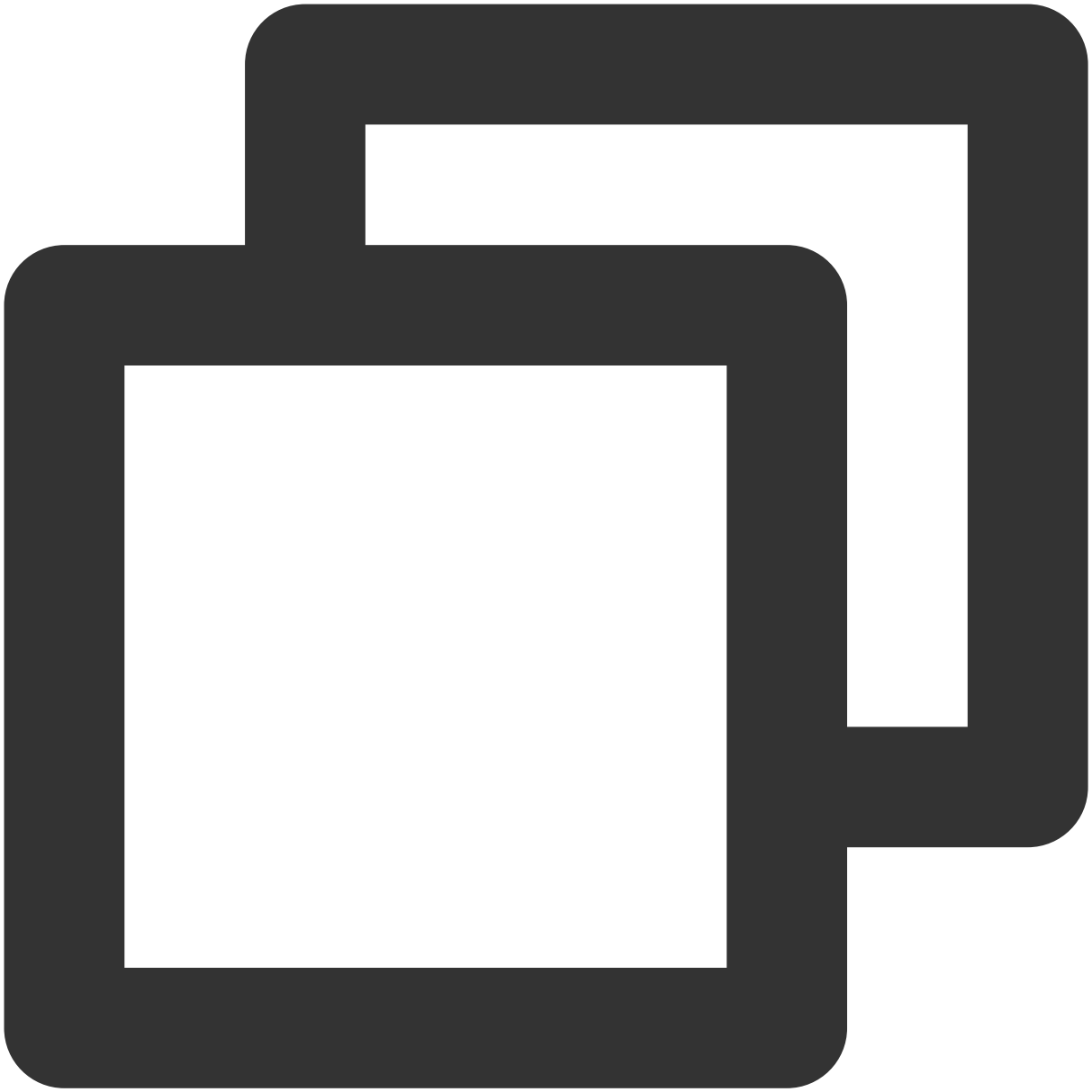


```
DELETE FROM products;
```

查询数据

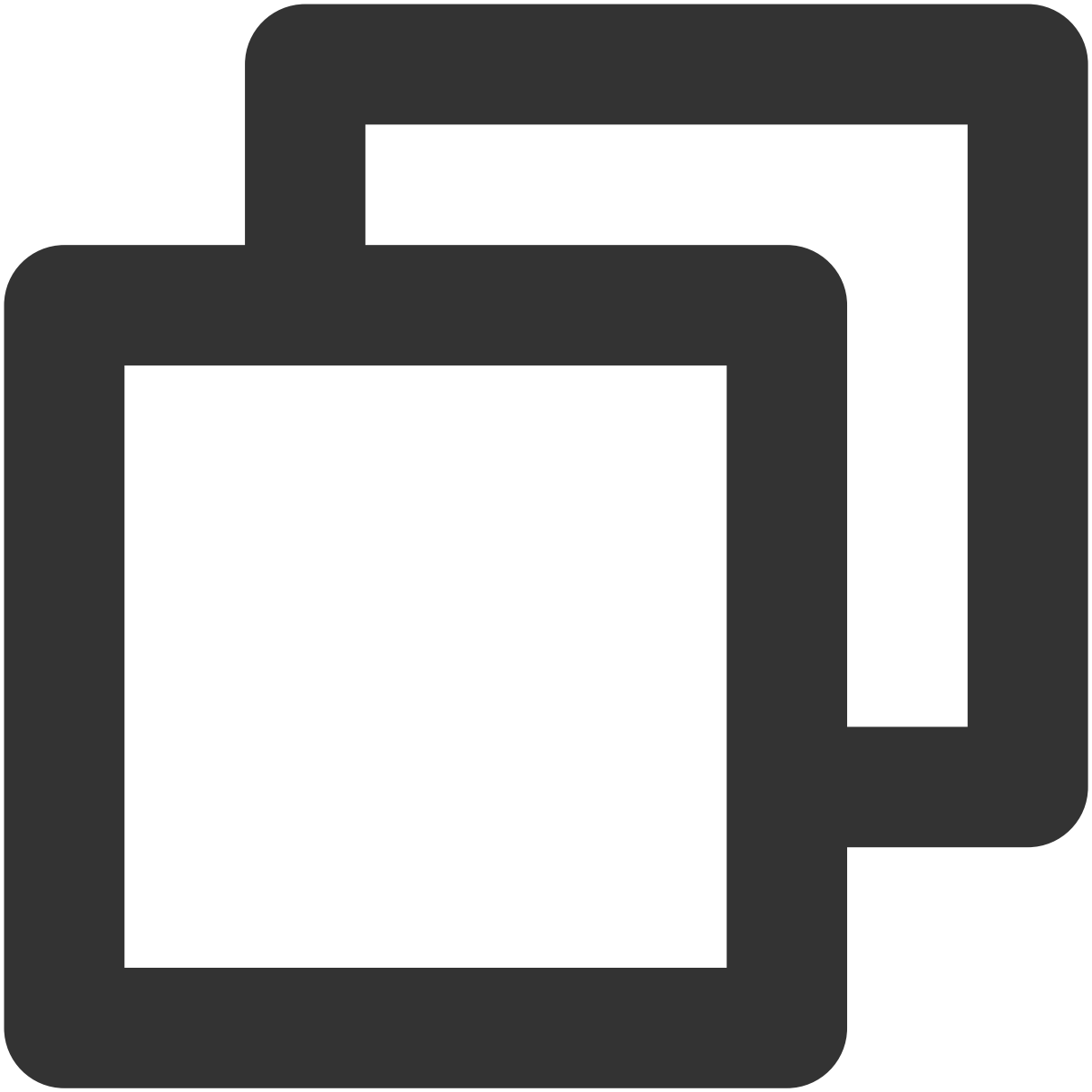
按照 访问数据仓库 方法访问数据库后，即可进行数据查询操作。示例如下：

1. 首先进入指定 DB，如下进入 test 数据库。



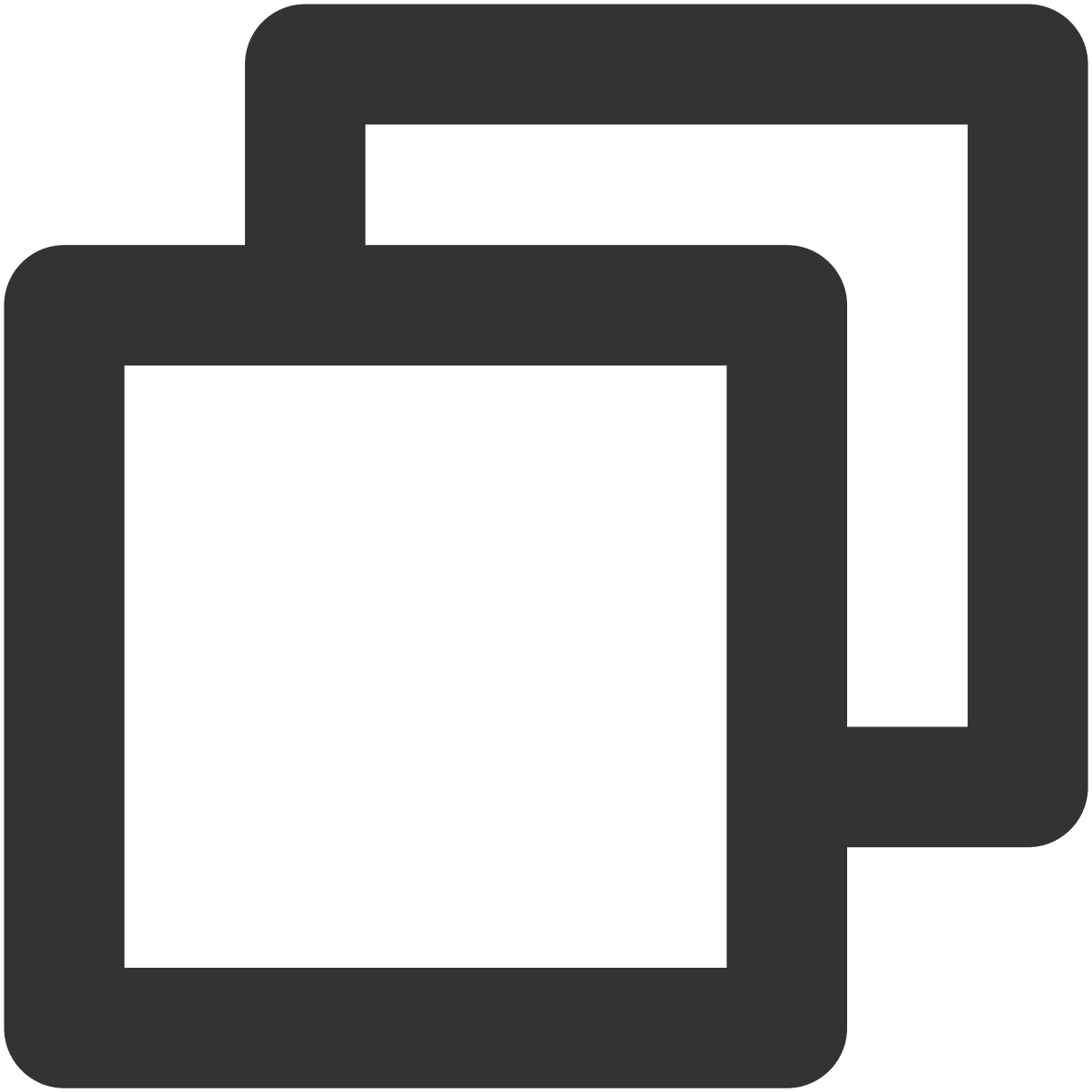
```
\\c test;
```

2. 创建表 test。



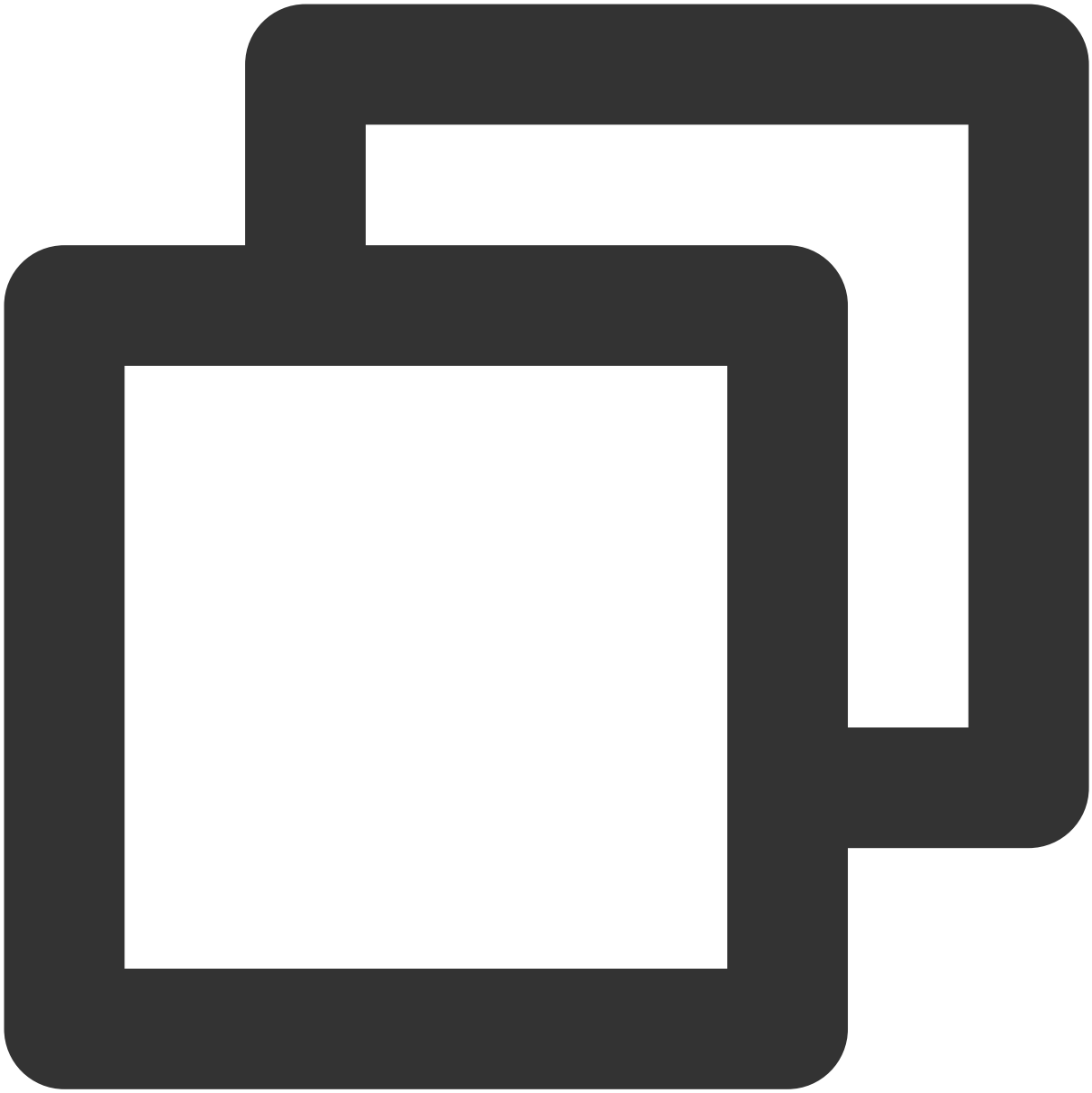
```
create table test(a1 int);
```

3. 插入数据。



```
insert into test values(3),(4);
```

4. 查询数据。



```
select * from test;
```

监控告警
告警配置

最近更新时间：2024-07-23 12:02:32

背景说明

云数据仓库 PostgreSQL 提供了性能监控面板，用户可以通过性能监控面板来观察集群各节点的运行指标的历史以及当前状态。云数据仓库 PostgreSQL 提供了告警通知的功能，以便让用户能够及时感知超过阈值的一些敏感指标，例如各节点的磁盘使用情况。

性能监控

进入 云数据仓库 PostgreSQL 控制台，在集群列表中单击集群名，进入集群详情页。在**性能监控**中，可以查看集群的各项指标。当有多个节点时，可在**节点维度**中选择想要查看的集群节点。
目前云数据仓库 PostgreSQL 提供了连接数、CPU 利用率、内存利用、网络接收吞吐量、网络输出吞吐量、写入 IOPS、读取 IOPS、磁盘空间使用率、读取吞吐量、写入吞吐量、读取延时、写入延时等指标。

告警接入

云数据仓库 PostgreSQL 的告警分为三个类型，分别为集群监控、主节点监控以及计算节点监控。这三种监控告警类型分为三个维度对用户进行告警通知。

新建告警策略

进入 云监控控制台，在**告警管理 > 告警配置 > 告警策略**中，单击**新增**，新增告警策略。新建告警策略时，**策略类型**可选择为 cdwpg 数据仓库-集群监控、cdwpg 数据仓库-主节点监控和 cdwpg 数据仓库-计算节点监控。本章以计算节点监控为例。
1. 选择**策略类型**为 cdwpg 数据仓库-计算节点监控。

2. 设置**告警对象**，可通过下拉框自行选择不同分组的计算节点。

3. 配置**触发条件**，可通过**选择模板**或**手动配置**两种方式。

若选择**选择模板**，可单击**新增配置触发条件**为每个指标配置告警触发的阈值，一旦该指标达到阈值满足条件后，系统将会向您发送告警信息。同时，也可单击**修改模板**修改已有模板。

若选择**手动配置**，可自行增加需要关注的其他指标，通过设置阈值来为不同的指标设置不同的告警阈值以及告警通知的周期策略等。

4. 配置**通知模板**，可单击**选择模板**选择已有模板或单击**新建模板**新建通知模板。

新建通知模板时，**接收对象**中配置对告警信息感兴趣或者需要处理告警信息的开发运营相关人员，可选择通过邮件、短信以及微信的方式进行告警。


新建通知模板
通知模板名称 *
仅支持中英文、数字、下划线、-，最多20个字符
接收对象 *
用户
新增
接收渠道 *
邮件
短信
微信
更多配置请到通知模板页
确定
取消

访问管理
访问管理概述

最近更新时间：2024-07-23 12:01:27

基本概念

访问管理（Cloud Access Management，CAM）是腾讯云提供的 Web 服务，主要用于帮助客户安全管理腾讯云账户下的资源的访问权限。用户可以通过访问管理创建、管理和销毁用户（组），并使用身份管理和策略管理控制其他用户使用腾讯云资源的权限。

访问授权

授予访问权限，指的是用户可以决定什么人、在何种条件下、对哪些资源、执行具体操作的控制能力组合。因此描述一个访问权限行为，通常包括四个元素：**身份、资源、操作、条件（可选）**。

访问授权元素说明

腾讯云身份

用户申请腾讯云账号时，系统会创建一个用于登录腾讯云服务的主账号身份。腾讯云主账号可通过用户管理功能对具有不同职责的分类用户进行管理。用户类型包括**协作者、消息接收人、子用户和角色**等，具体定义可参见 词汇表。

云数据仓库 PostgreSQL 集群资源

云数据仓库 PostgreSQL 的资源是指云数据仓库 PostgreSQL 集群，访问控制也是针对云数据仓库 PostgreSQL 集群进行控制，通常在控制台我们能看到云数据仓库 PostgreSQL 的标识，示例如下：

其中 snova-28fg7yl3 就是集群唯一标识，也可以理解为云数据仓库 PostgreSQL 资源的标识。

云数据仓库 PostgreSQL 集群的操作

集群的操作是指用户在云数据仓库 PostgreSQL 的控制台进行的操作行为，基本上每一个操作行为都可以映射到一个云 API，例如删除集群，或者查看集群详细信息，都拥有一个 action 标识，可以在访问控制时针对 action(读,写) 进行访问控制。

最小权限原则

授权时，请务必明确权限范围，需要明确授予**指定用户**在**何种条件**下，执行**何种操作**，访问**何种资源**的权限。

策略设置

最近更新时间：2024-02-19 10:49:28

概述

访问策略可用于授予访问云数据仓库 PostgreSQL 集群的权限。访问策略使用基于 JSON 的访问策略语言。您可以通过访问策略语言授权指定委托人（principal）对指定的 云数据仓库 PostgreSQL 集群资源执行指定的操作。
访问策略语言描述了策略的基本元素和用法，有关策略语言的说明可参考 CAM 策略管理。

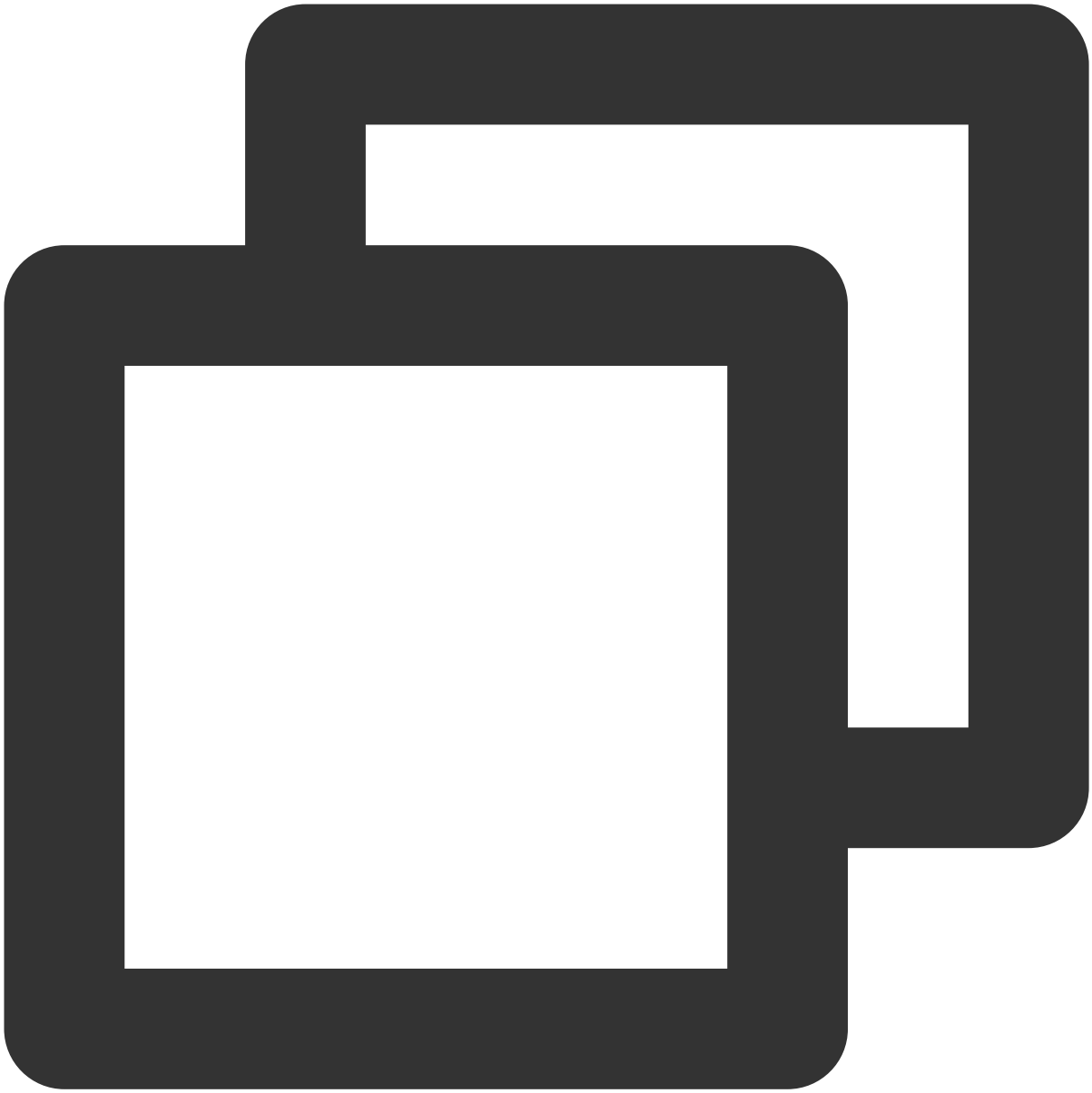
访问策略中的元素

访问策略语言包含以下基本意义的元素：
语句（statement）：描述一条或多条权限的详细信息。该元素包括效力、操作、资源、条件等多个其他元素的权限或权限集合。一条策略有且仅有一个语句元素。
效力（effect）：描述声明产生的结果是“允许”还是“显式拒绝”，包括 allow 和 deny 两种情况。该元素是必填项。
操作（action）：描述允许或拒绝的操作。操作可以是 API（以 name 前缀描述）或者功能集（一组特定的 API，以 permid 前缀描述）。该元素是必填项。
资源（resource）：描述授权的具体数据。资源是用六段式描述。每款产品的资源定义详情会有所区别。该元素是必填项。
条件（condition）：描述策略生效的约束条件。条件包括操作符、操作键和操作值组成。条件值可包括时间、IP 地址等信息。有些服务允许您在条件中指定其他值。该元素是选填项。

元素用法

指定效力

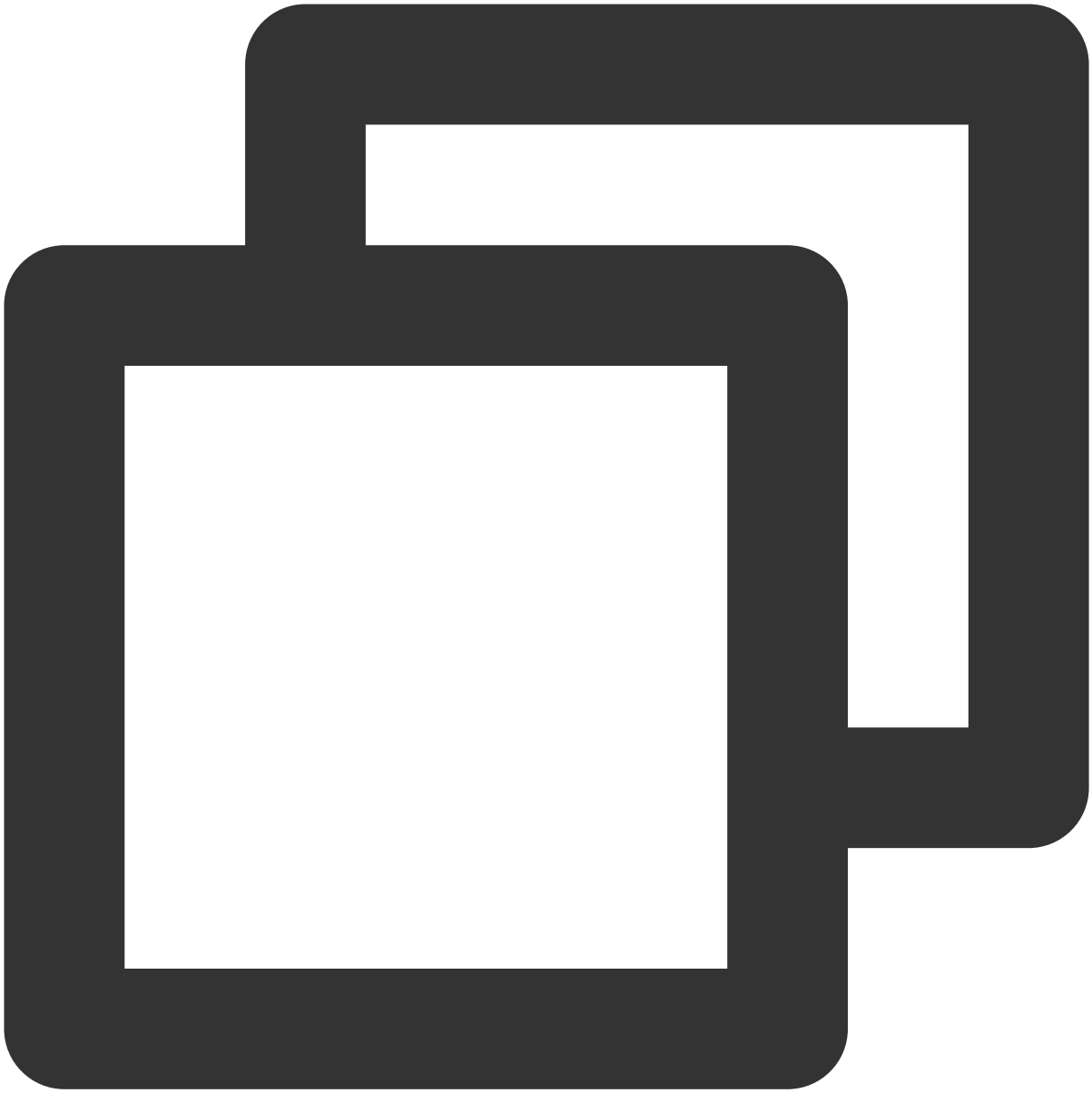
如果没有显式授予（允许）对资源的访问权限，则隐式拒绝访问。同时，也可以显式拒绝（deny）对资源的访问，这样可确保用户无法访问该资源，即使有其他策略授予了访问权限的情况下也无法访问。下面是指定允许效力的示例：



```
"effect" : "allow"
```

指定操作

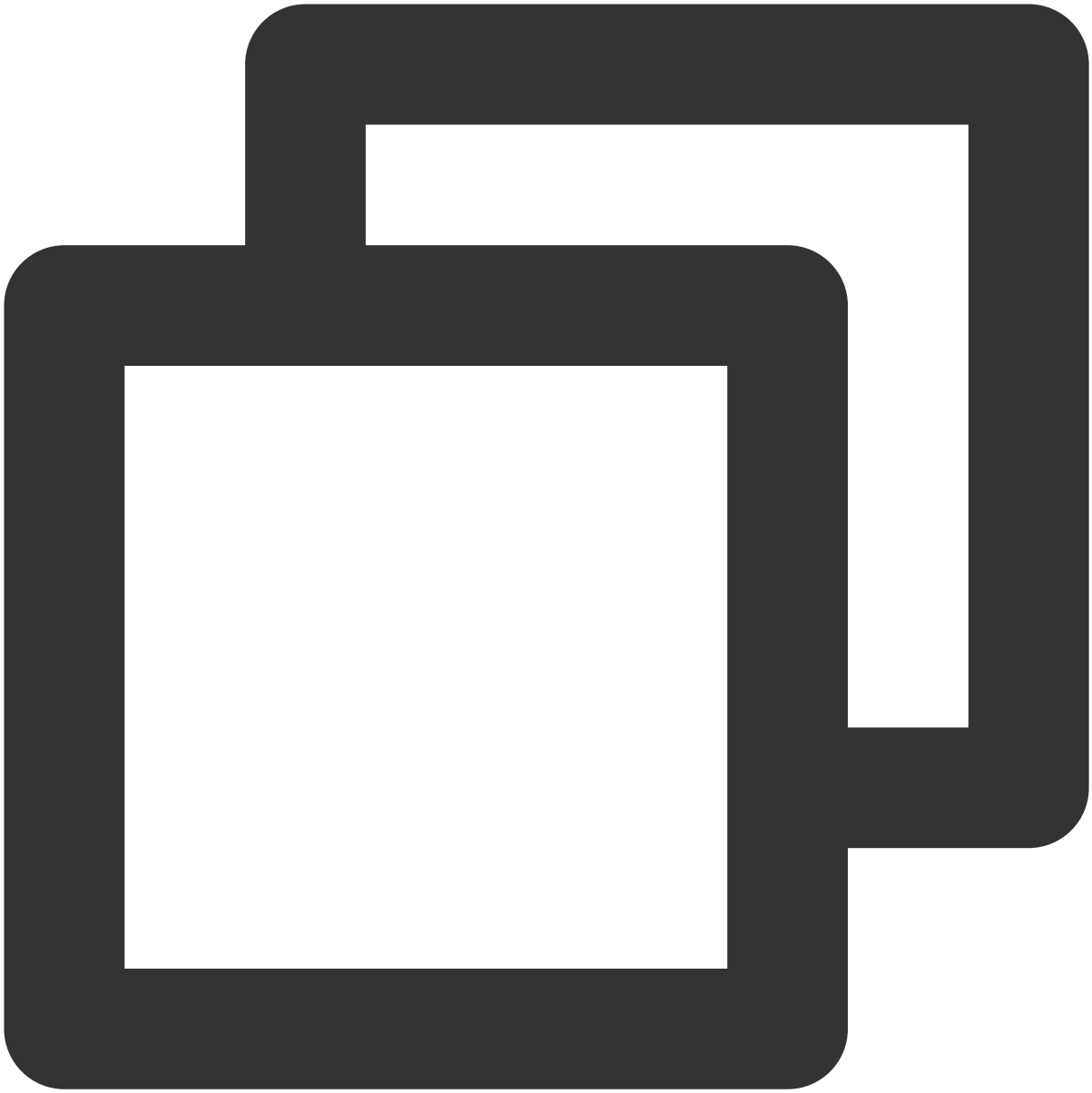
云数据仓库 PostgreSQL 定义了可在策略中指定一类控制台的操作，指定的操作按照操作性质分为读取部分接口（cdwpg:Describe*）和全部接口（cdwpg:*）。
指定允许操作的示例如下：



```
"action": [
  "name/cdwpg:Describe*"
]
```

指定资源

资源（resource）元素描述一个或多个操作对象，如云数据仓库 PostgreSQL 集群资源等。所有资源均可采用下述的六段式描述方式。



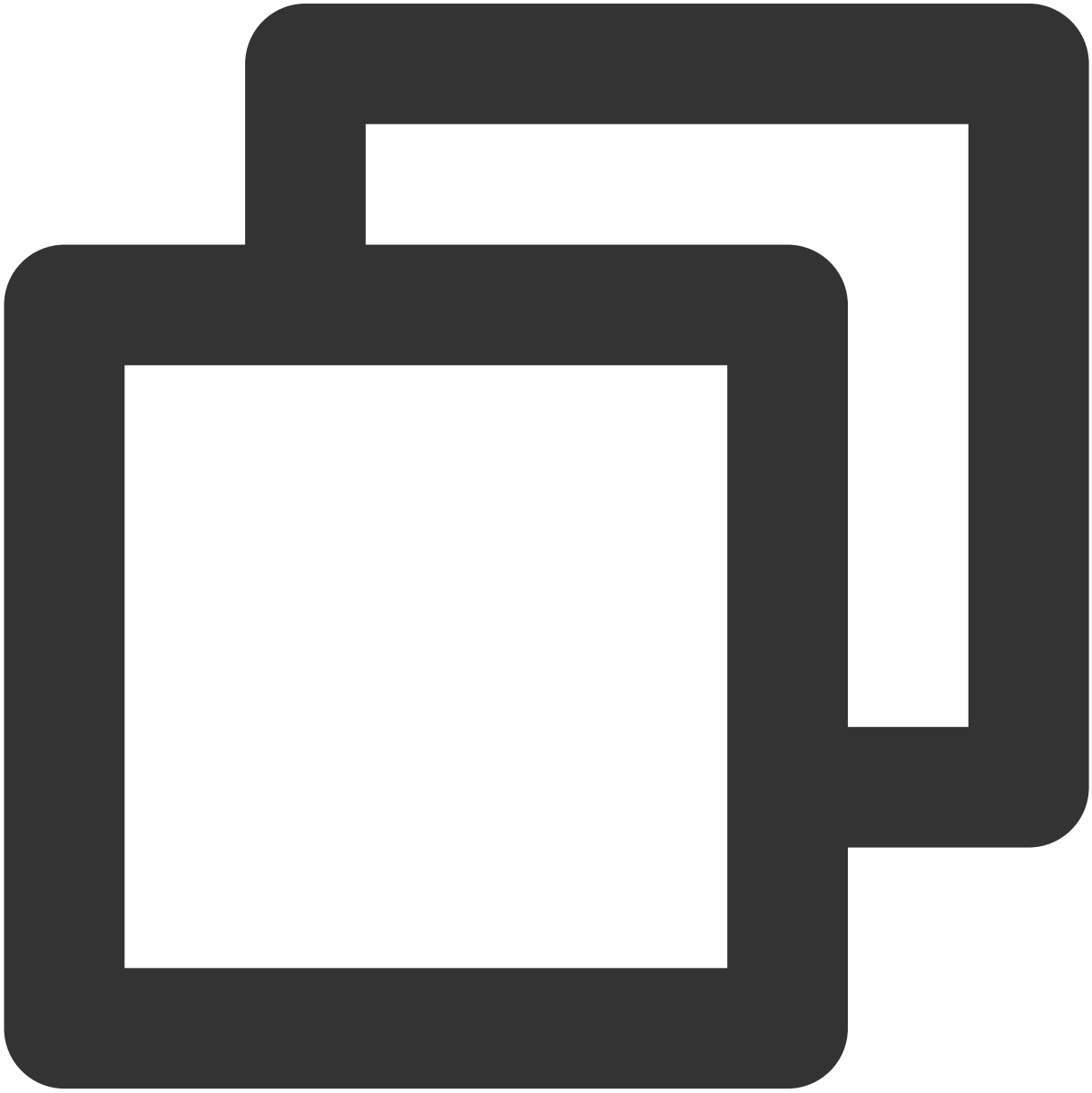
```
qcs:project_id:service_type:region:account:resource
```

参数说明如下：

参数	描述	是否必选
qcs	是 qcloud service 的简称，表示是腾讯云的云服务	是
project_id	描述项目信息，仅为了兼容 CAM 早期逻辑，一般不填	否

service_type	这里为 cdwpg	是
region	描述地域信息	是
account	描述资源拥有者的主账号信息，即主账号 UIN，表示为 `uin/${OwnerUin}`，如 uin/100000000001	是
resource	描述具体资源详情，前缀为 cdwpg-instance	是

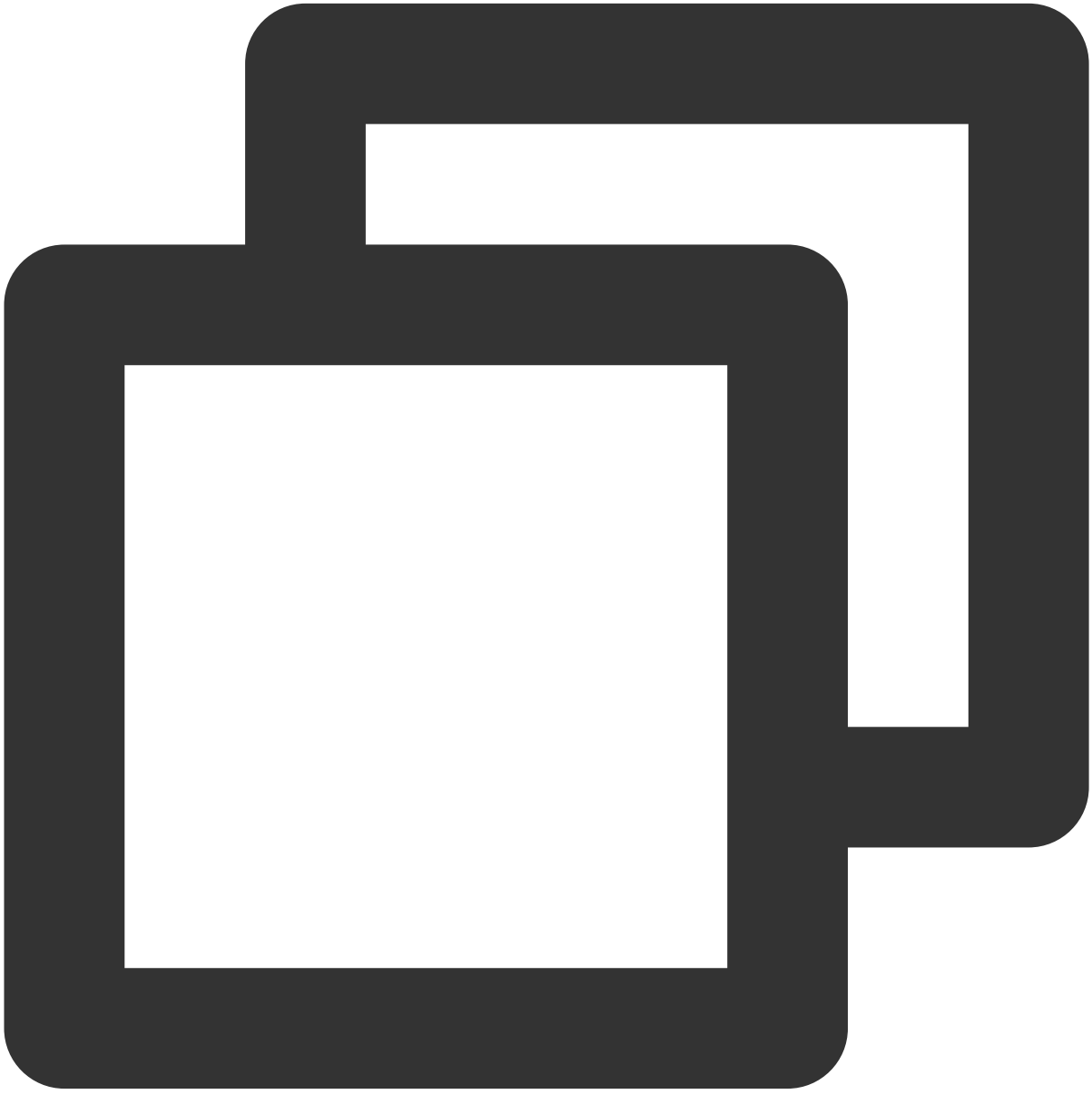
下面是一个云数据仓库 PostgreSQL 集群的六段式信息：



```
"resource":["qcs::cdwpg:ap-guangzhou:uin/100000000001:cdwpg-instance/snova-73jingds
```

指定条件

访问策略语言可使您在授予权限时指定条件。在 云数据仓库 PostgreSQL 场景下主要是用于设置标签鉴权，标签条件只对绑定了该标签的集群生效，标签策略示例如下：



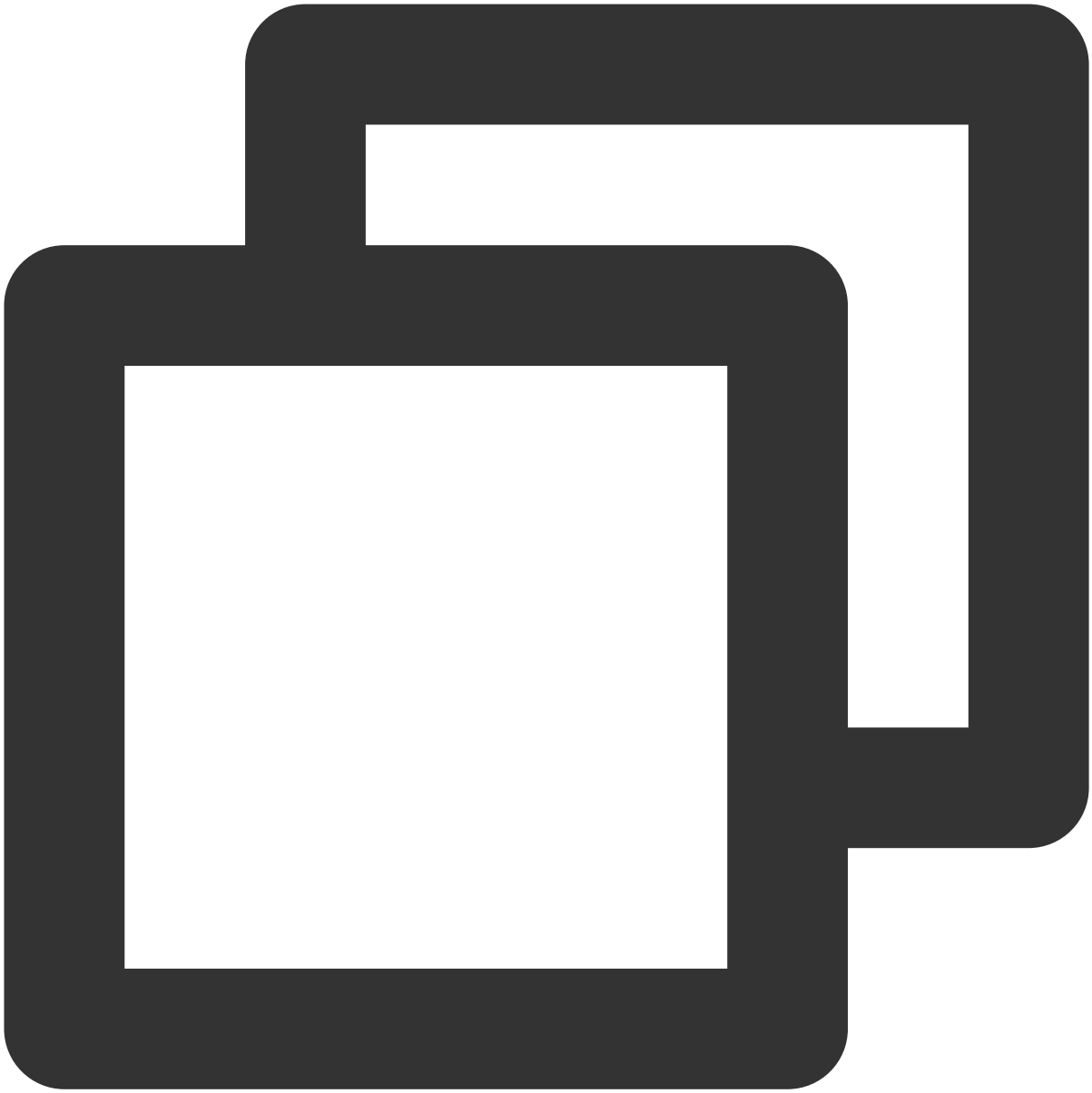
```
"condition": {
                "for_any_value:string_equal": {
                    "qcs:tag": [
```

```
                "jing&jingfdd"
            ]
        }
    }
```

这个语句含义是策略包含标签 key 为 jing，value 为 jingfdd 的资源。

实际案例

如下案例中，策略的含义是允许访问 UIN 为1250000000下面的集群 ID 为 snova-jidnshgdsh 的资源，以及绑定了标签键为 testkey，标签值为 testvalue 的资源。



```
{
   "version": "2.0",
    "statement": [
        {
            "action": [
                "name/cdwpg:Describe*",
            ],
            "condition": {
               "for_any_value:string_equal": {
                    "qcs:tag": [
                        "testkey&testvalue"
```

```
                    ]
                }
            },
            "effect": "allow",
            "resource": [
                "qcs::cdwpg:ap-guangzhou:uin/1250000000:cdwpg-instance/snova-jidnsh
            ]
        }
    ]
}
```

策略授予

最近更新时间：2024-07-23 12:02:00

云数据仓库 PostgreSQL 预设策略管理

为了方便用户授权子账号资源，云数据仓库 PostgreSQL 提前设置了两条预设策略，进入 访问管理 控制台，在页面右上角搜索框中搜索“CDWPG”，可看到如下两条策略：

策略	说明
QcloudCDWPGFullAccess	授予云数据仓库 PostgreSQL 管理的完整权限
QcloudCDWPGReadOnlyAccess	授予云数据仓库 PostgreSQL 管理的只读权限

设置让用户拥有创建和管理云数据仓库 PostgreSQL 实例的权限，可对此用户使用名称为 QcloudCDWPGFullAccess 的策略。
设置让用户只拥有查询云数据仓库 PostgreSQL 集群及相关资源（私有网络、安全组、Monitor）的权限，但不允许该用户拥有创建、删除和修改等操作的权限，可对此用户使用名称为 QcloudCDWPGReadOnlyAccess 的策略。

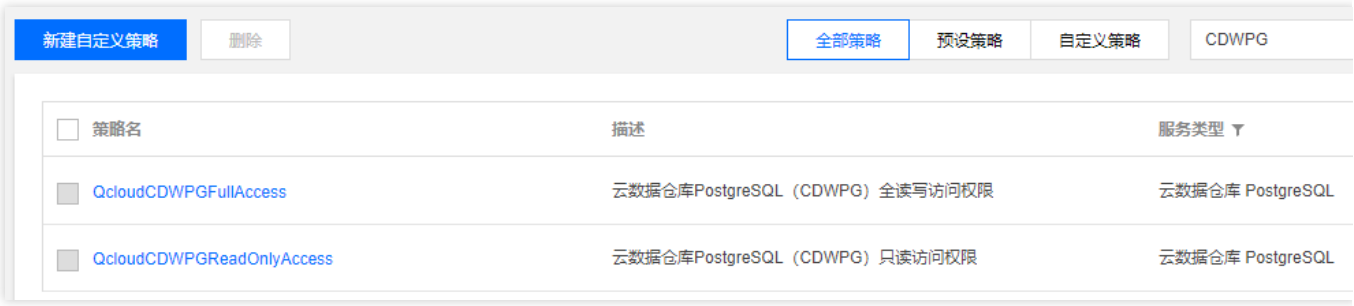
云数据仓库 PostgreSQL 自定义策略

如果预设策略无法满足需求，可单击**新建自定义策略**创建自定义策略。

创建自定义策略的方法可参考 策略设置。

策略授权

已设置好的策略可以通过关联用户组或者子用户来授予权限。



自定义策略可授权资源类型

资源级权限指的是能够指定用户对哪些资源具有执行操作的能力。云数据仓库 PostgreSQL 部分支持资源级权限，即表示针对支持资源级权限的云数据仓库 PostgreSQL 操作，您可以控制何时允许用户执行操作或是允许用户使用特定资源。访问管理 CAM 中可授权的资源类型如下：

资源类型	授权策略中的资源描述方法
云数据仓库 PostgreSQL	`qcs::cdwpg:$region:$account:cdwpg-instance/*` `qcs::cdwpg:$region:$account:cdwpg-instance/$clusterId`

下表将介绍当前支持资源级权限的云数据仓库 PostgreSQL API 操作，设置策略时，action 填入 API 操作名称就可以对单独 API 进行控制，设置 action 也可以使用*作为通配符。

支持资源级授权的 API 列表

API 操作	资源路径
ModifyClusterSize	修改集群节点数

DescribeClusters	获取集群详细信息
DescribeRealtimeQuery	获取集群实时查询详细信息
DescribeHistoryQuery	获取集群历史查询详细信息
AbortQuery	终止集群查询
DescribeRealtimeQueries	获取集群实时查询列表
DescribeGpStatus	获取集群数据库状态
RebootCluster	重启集群
DescribeClusterStatus	获取集群状态
ModifyClusterSubnet	修改集群子网
DescribeHistoryQueries	获取集群历史查询列表
DeleteCluster	删除集群
ModifyClusterUserPassword	重新设置集群密码
ModifyClusterBasic	修改集群名称
DescribeClustersStatistics	获取集群列表数量
DescribeVpcLinks	获取集群私有访问链接
CreateVpcLink	创建私有网络访问连接
DeleteVpcLink	删除私有网络访问连接
ExpandClusterSize	扩容集群大小
DescribeHbaConfigList	获取集群访问地址白名单
SetHbaConfigList	修改集群访问地址白名单
DescribeClusterResourceQueueList	查询数据库集群资源队列列表
DescribeClustersLimit	查询数据库集群资源限制配置
HandlerResourceQueue	操作数据库资源队列
AdminClusterOutnetAddres	管理外网访问地址
DescribeClustersNodesInfo	获取集群节点信息

不支持资源级授权的 API 列表

针对不支持资源级权限的云数据仓库 API 操作，您仍可以向用户授予使用该操作的权限，但策略语句的资源（resource）元素必须指定为*。

API 操作	API 描述
DescribeNodeConfigInfo	获取机型节点规格信息
DescribeEvents	获取所有集群事件信息
CreateCluster	创建集群
DescribeDbStatus	获取数据库状态
DescribeZones	获取可选购买区域
DescribeSegNodeMaxCount	查询计算节点最大个数
DescribeClusterExtend	获取所有集群运维信息
DescribeResidual	获取区域资源状态
DescribeSpecResidual	查询特定规格是否售罄
DescribeZonesResource	获取可用区资源信息
DescribeValidRegionAndZones	获取有效地域和可用区