华为认证 GaussDB 系列教程

HCIP-GaussDB-OLTP

高级数据库工程师

实验指导手册

版本:1.5

华为技术有限公司

商标声明

 和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： http://e.huawei.com

华为认证体系介绍

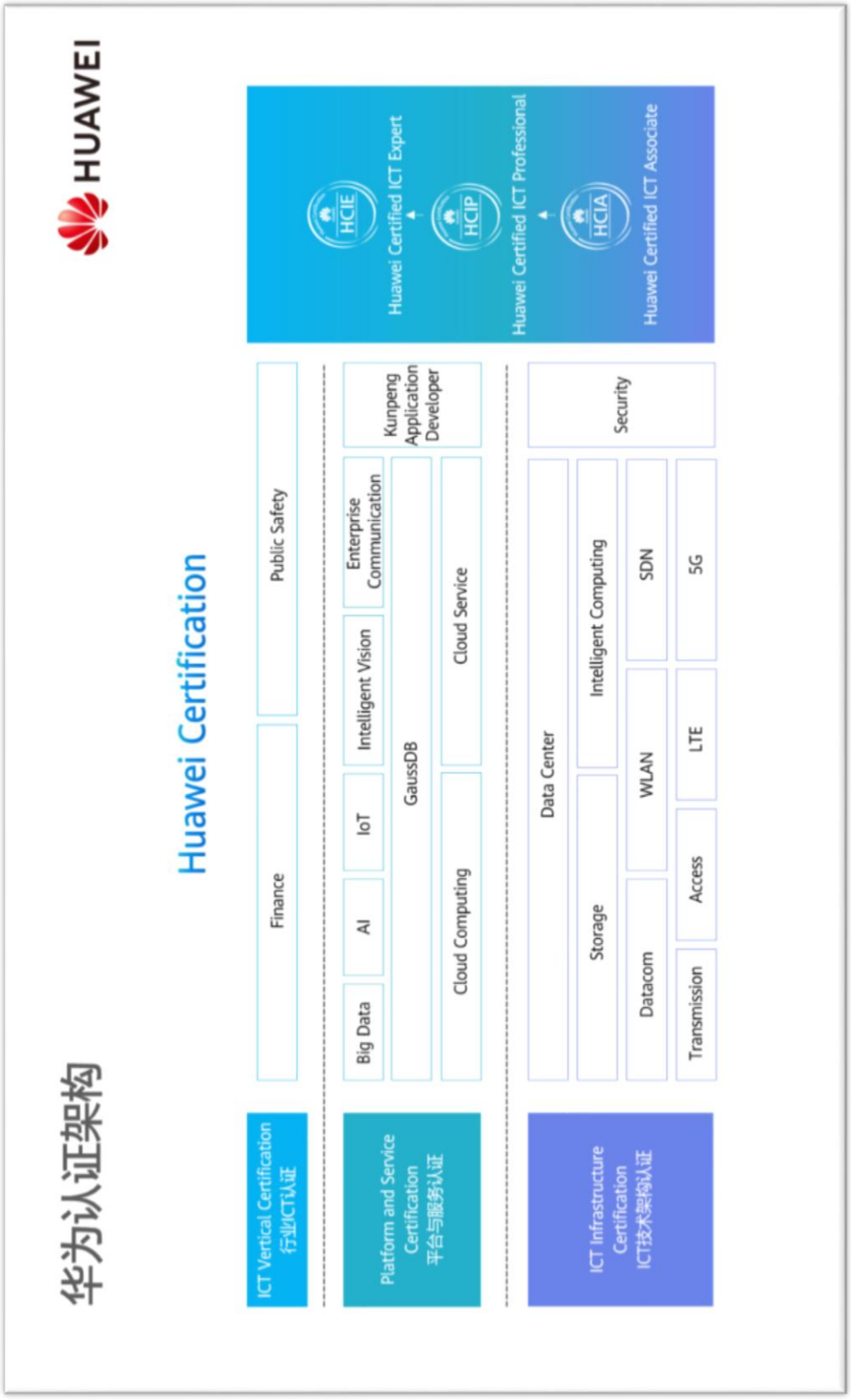
华为认证是华为公司基于“平台+AI+生态”战略，围绕“云-管-端”协同的新ICT技术架构，华为公司打造了业界唯一覆盖ICT全技术领域的认证体系，包含ICT技术架构认证、平台与服务认证和行业ICT认证三类认证。

根据ICT从业者的学习和进阶需求，华为认证分为工程师级别、高级工程师级别和专家级别三个认证等级。

华为认证HCIP-GaussDB-OLTP V1.5定位于培养和认证具备基于GaussDB数据库应用开发及云DBA能力的高级工程师。

通过HCIP-GaussDB-OLTP V1.5认证，您将掌握GaussDB(for MySQL)架构、高级特性和高级SQL语法，具备云上数据库管理和运维、云数据库性能调优、云上数据库应用开发等能力，能够胜任云数据库应用开发及云DBA岗位

华为认证协助您打开行业之窗，开启改变之门，屹立在WLAN网络世界的潮头浪尖！


华为认证架构
HUAWEI
Huawei Certification
ICT Vertical Certification
行业ICT认证
Finance
Public Safety
Platform and Service Certification
平台与服务认证
Big Data
AI
IoT
Intelligent Vision
Enterprise Communication
GaussDB
Kunpeng Application Developer
Cloud Computing
Cloud Service
ICT Infrastructure Certification
ICT技术架构认证
Data Center
Storage
Intelligent Computing
Security
Datacom
WLAN
SDN
Transmission
Access
LTE
5G
HCIE
Huawei Certified ICT Expert
HCIP
Huawei Certified ICT Professional
HCIA
Huawei Certified ICT Associate

前 言

简介

本书为 HCIP-GaussDB-OLTP 认证培训教程，适用于准备参加 HCIP-GaussDB-OLTP 考试的学员或者希望了解 GaussDB(for MySQL)高级特性、高阶语法及函数、数据库性能调优、云 DBA 操作以及云 DBA 安全管理等相关 GaussDB 数据库技术的读者。

内容描述

本实验指导书共包含 8 个实验，从云上数据库环境搭建，到高阶语法操作，云上数据库日常操作，到云 DBA 智能运维，数据库性能调优以及数据库安全管理，最后一个综合大实验来巩固知识点。

- 实验一为 GaussDB(for MySQL)数据库环境搭建实验，通过华为云上购买 GaussDB(for MySQL)，帮助读者熟悉掌握华为云的购买流程和数据库连接的基本操作。
- 实验二为数据库高阶语法实验，通过 DAS 服务，编写 SQL 语句，掌握高阶函数及语法和存储过程等相关知识。
- 实验三为云上数据库基本操作，通过华为云的 Web 界面，完成数据库相关参数修改，用户授权、备份恢复、监控查看等操作，帮助读者掌握云上数据库的基本操作内容。
- 实验四为云 DBA 智能运维实验，通过 DAS 的云 DBA 服务，帮助读者掌握云数据库的运维相关操作。
- 实验五为数据库性能调优实验，通过从存储引擎到数据库表设计以及执行计划的分析，帮助读者掌握数据库性能调优的基本操作。
- 实验六为数据库安全管理实验，通过安全组设定，用户权限及密码设定等操作，帮助读者掌握数据库基本的安全管理相关知识。
- 实验七为综合大实验，通过几个综合模块来将知识点串起来，帮助读者巩固相关知识点。
- 实验八为资源释放实验，通过华为云操作释放购买的资源，避免资源浪费。

读者知识背景

本课程为华为认证基础课程，为了更好地掌握本书内容，阅读本书的读者应首先具备以下基本条件：

- 具有基本的数据库知识背景，同时熟悉华为云界面，了解基本 Linux 知识。

实验环境说明

组网说明

本实验环境面向准备 HCIP-GaussDB-OLTP 考试的高级数据库工程师。每套实验环境包括 GaussDB(for MySQL)服务一个，数据库管理服务一个，弹性公网 IP 一个，RDS 服务一个，DRS 服务一个。每套实验环境适用于 4~12 名学员同时上机操作。

设备介绍

为了满足 HCIP-GaussDB-OLTP 实验需要，建议每套实验环境采用以下配置：

设备名称、型号与版本的对应关系如下：

表1-1 对应关系

设备名称	设备型号	软件版本
GaussDB数据库	GaussDB(for MySQL)	MySQL 8.0内核
数据库管理服务	DAS服务	-
弹性公网服务	EIP	-
数据复制服务	DRS服务	-
RDS数据库	RDS for MySQL	8.0

准备实验环境

检查设备

实验开始之前请每组学员检查自己的实验设备是否齐全，实验清单如下。

表1-2 实验清单

设备名称	数量	备注
GaussDB(for MySQL)	1	所有实验组共用
DAS 服务	1	所有实验组共用
EIP	1	所有实验组共用

DRS 服务	1	每人一个
RDS for MySQL	1	每人一个
笔记本或台式机	每组 1 台	台式机要有无线网卡

每组检查自己的设备列表如下：

- 笔记本或台式机 1 台

目录

1 数据库搭建实验

1.1 实验介绍

1.1.1 关于本实验

本实验通过在华为云上购买 GaussDB(for MySQL)，了解 GaussDB(for MySQL)配置流程，掌握 GaussDB(for MySQL)的连接和 SQL 编写操作。

1.1.2 实验目的

- 掌握 GaussDB(for MySQL)购买流程；
- 了解 GaussDB(for MySQL)配置参数；
- 掌握 GaussDB(for MySQL)基本操作；
- 掌握 DAS 登录流程。

1.2 实验任务配置

1.2.1 购买 GaussDB(for MySQL)

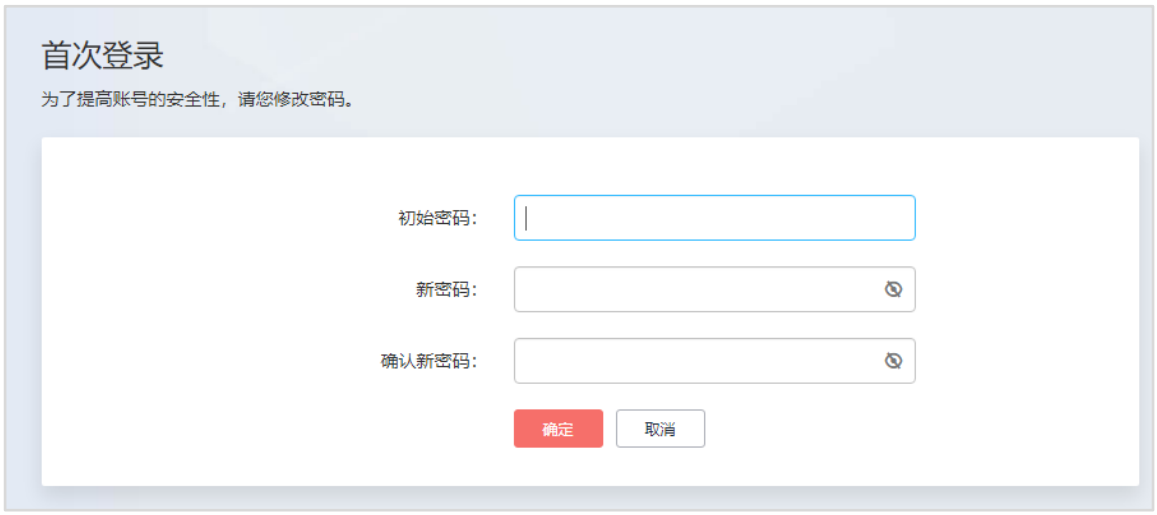
步骤 1 登录华为云

通过链接 https://auth.huaweicloud.com/authui/login.html#/login 登录华为云。

点击“IAM 用户登录”，进入 IAM 登录界面。

输入对应的账号名，IAM 用户和密码。

点击“登录”，完成登录。



初次登录需要修改初始密码，按照要求填写新密码。

点击“确定”，然后下次使用新密码重新登录华为云。

步骤 2　进入 GaussDB(for MySQL)控制台

点击“服务列表”>“数据库”>“云数据库 GaussDB”，进入 GaussDB(for MySQL)控制台。

控制台如下：

在控制台上可以看到对应的区域下 GaussDB(for MySQL)的所有数据库实例。

步骤 3　购买 GaussDB(for MySQL)

在 GaussDB(for MySQL)控制台界面上，点击右上角的“购买数据库实例”，进入购买数据库实例配置界面。

配置 GaussDB(for MySQL)如下：

图上信息为：

计费模式：选择“按需计费”；

区域：选择“华东-上海一”；

实例名称：默认即可，如有特殊需求，可以按照规则重命名；

数据库引擎：默认为“GaussDB(for MySQL)”；

兼容的数据库版本：默认为“MySQL8.0”；

可用区类型：默认为“单可用区”。

性能规格配置如下：

图上信息为：

性能规格：选择“鲲鹏通用计算增强型”；

CPU|内存：选择“16 核|64GB”；

只读节点数量：默认为“1”。

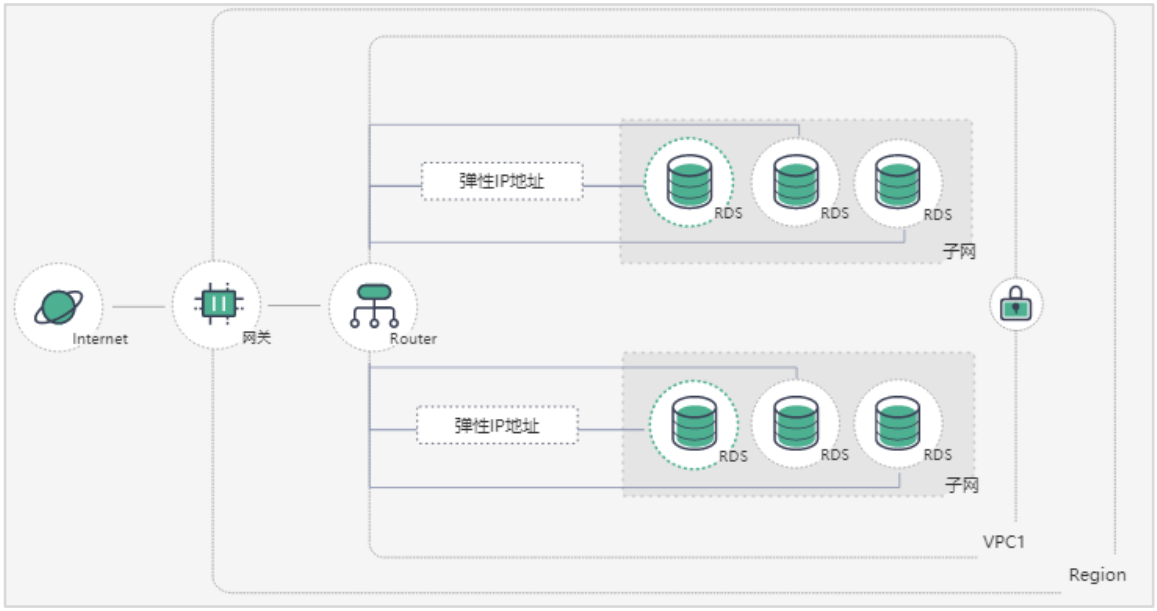
网络设置如下：

图上信息为：

虚拟私有云：默认即可，没有则直接创建默认虚拟私有云即可；

内网安全组：使用默认“Sys-default”，没有则直接创建默认即可。

虚拟私有云、子网、安全组与实例的关系如下：



用户密码及模板配置如下：

图上信息为：

管理员密码：满足密码规则即可；

确认密码：再次输入密码；

参数模板：默认即可；

表名大小写敏感：默认为“否”。

点击“立即购买”，查看并确认配置的信息是否正确。

确认无误后，点击“提交”，完成购买。

点击“返回云数据库 GaussDB 列表”，查看 GaussDB(for MySQL)创建的进程。

等待一会之后，云数据库 GaussDB(for MySQL)创建成功。

此时数据库的运行状态为“正常”。

1.2.2 通过 DAS 登录 GaussDB(for MySQL)

步骤 1　登录 DAS

在数据库实例控制台上，选中数据库实例，点击“操作”>“登录”。

进入 DAS 登录界面，输入用户名和密码。

点击“测试连接”，查看是否可以连接数据库。

出现“连接成功”字样，说明数据库连接成功。

勾选“记住密码”、“定时采集”和“SQL 执行记录”。

点击“登录”，登录 GaussDB(for MySQL)数据库实例。

步骤 2　操作 DAS

点击“新建数据库”。

在弹出窗口中配置数据库名称为“conn-test”。

点击数据库名称，进入数据库界面信息。

至此数据库搭建实验完成。

1.2.3 数据库绑定公网 IP

步骤 1 购买弹性公网 IP

在华为云上点击“产品”>“基础服务”>“网络”>“弹性公网 IP EIP”。进入弹性公网 IP 界面。

点击“立即购买”，配置弹性公网 IP。

配置如下：

计费模式：按需计费；

区域：华东-上海一（与 GaussDB(for MySQL)保持在同一个区域）；

线路：全动态 BGP；

公网带宽：按流量计费；

带宽大小：5Mbit/s；

其余默认。

点击“立即购买”，确认 EIP 规格。

确认无误后，点击“提交”，完成 EIP 购买。

此时弹性公网 IP 地址为：**124.70.87.29**。

步骤 2　绑定公网 IP

点击 GaussDB(for MySQL)数据库名称，进入基本信息界面，点击“绑定”。

选择购买的公网 IP 地址。

点击“确定”，完成绑定。

2 SQL 语法进阶

2.1 实验介绍

2.1.1 关于本实验

本实验主要完成常见的函数、正则表达式、控制流函数、JSON 函数、窗口函数、存储过程及触发器的练习。

2.1.2 实验目的

- 掌握常见函数、正则表达式的使用方法；
- 掌握控制流函数、窗口函数的使用方法；
- 熟悉 JSON 函数的操作；
- 能够自行编写存储过程及触发器。

2.2 实验任务配置

2.2.1 环境准备

创建数据库 advanced_prac。说明：若实际使用中使用同一个实例，建议数据库名额外加上个人姓名简写，便于区分，例如 advanced_prac_hql。

2.2.2 一般常用函数

TRIM([{BOTH | LEADING | TRAILING} [remstr] FROM] str), TRIM([remstr FROM] str): 将字符串 str 中前后包含的 remstr 部分删除。

输入：

```
SELECT TRIM(TRAILING 'xyz' FROM 'barxxyz');
```

返回结果：

	TRIM(TRAILING...
1	barx

LTRIM(str)/RTRIM(str): 删除字符串 str 开头的空格/删除字符串 str 结尾的空格。

输入：

```
SELECT LTRIM('    barbar');
```

返回结果：

	LTRIM(' barb...
1	barbar

RPAD(str,len,padstr): 在字符串 str 右侧用 padstr 填充，使其长度为 len。若字符串 str 长度超过 len，则将其截断。

输入：

```
SELECT RPAD('hi',15,'GaussDB');
```

返回结果：

	RPAD('hi', 15...
1	hiGaussDBGaussD

输入：

```
SELECT RPAD('GaussDB',5,'?');
```

返回结果：

	RPAD('GaussDB...
1	Gauss

2.2.3 正则表达式操作符和函数

expr REGEXP pat, expr RLIKE pat: 若 expr 处表达式与正则表达式匹配，则返回 1，否则返回 0。

输入：

```
SELECT 'a' REGEXP 'A', 'a' REGEXP BINARY 'A';
```

返回结果：

	'a' REGEXP 'A'	'a' REGEXP BI...

1	1	0

REGEXP_INSTR(expr, pat[, pos[, occurrence[, return_option[, match_type]]]]): 返回与正则表达式模式匹配的子字符串的起始索引。

输入：

```
SELECT REGEXP_INSTR('dog cat dog', 'dog', 2);
```

返回结果：

	REGEXP_INSTR(...
1	9

输入：

```
SELECT REGEXP_INSTR('aa aaa aaaa', 'a{4}');
```

返回结果：

	REGEXP_INSTR(...
1	8

REGEXP_LIKE(expr, pat[, match_type]): 用于模式匹配。比较给定的字符串，如果字符串相同则返回 1，否则返回 0。

输入：

```
SELECT REGEXP_LIKE('Michael!', '.*');
```

返回结果：

	REGEXP_LIKE('...
1	1

输入：

```
SELECT REGEXP_LIKE('abc', 'ABC', 'c');
```

返回结果：

	REGEXP_LIKE('...
1	0

REGEXP_REPLACE(expr, pat, repl[, pos[, occurrence[, match_type]]]): 通过匹配字符来替换给定的字符串。

输入：

```
SELECT REGEXP_REPLACE('abc def ghi', '[a-z]+', 'X', 1, 3);
```

返回结果：

	REGEXP_REPLAC...
1	abc def X

REGEXP_SUBSTR(expr, pat[, pos[, occurrence[, match_type]]]): 从给定的字符串中返回子字符串。

输入：

```
SELECT REGEXP_SUBSTR('abc def ghi', '[a-z]+', 1, 3);
```

返回结果：

	REGEXP_SUBSTR...
1	ghi

2.2.4 控制流函数

CASE

CASE 语句是实现选择结构程序设计的一种语句。

创建表（本节以表 bonuses_depa1 和 new_bonuses_depa1 为例）。输入：

```
CREATE TABLE bonuses_depa1
(
staff_id INT NOT NULL,
staff_name CHAR(50),
job VARCHAR(30),
bonus NUMERIC
);
CREATE TABLE new_bonuses_depa1
(
staff_id INT NOT NULL,
staff_name CHAR(50),
job VARCHAR(30),
bonus NUMERIC
);
```

注意事项：

- 表名必须指定，并且表名不能和用户下的其他表重名。
- 列名必须指定，同时指定列的数据类型、是否可以为 NULL、size 等条件。
- 如果表名重复，修改表名或者删除重名的表。

向表 bonuses_depa1、new_bonuses_depa1 插入数据。输入：

```
INSERT INTO bonuses_depa1(staff_id, staff_name, job, bonus) VALUES(23,'wangxia','developer',5000);
INSERT INTO bonuses_depa1(staff_id, staff_name, job, bonus) VALUES(24,'limingying','tester',7000);
INSERT INTO bonuses_depa1(staff_id, staff_name, job, bonus) VALUES(25,'liulili','quality control', 8000);
INSERT INTO bonuses_depa1(staff_id, staff_name, job, bonus) VALUES(29,'liuxue','tester',8000);
INSERT INTO bonuses_depa1(staff_id, staff_name, job, bonus) VALUES(21,'caoming','document
developer',11000);
INSERT INTO new_bonuses_depa1(staff_id, staff_name, job, bonus) VALUES(23,'wangxia','developer',
7000);
INSERT INTO new_bonuses_depa1(staff_id, staff_name, job, bonus) VALUES(27,'wangxuefen','document
developer',7000);
INSERT INTO new_bonuses_depa1(staff_id, staff_name, job, bonus) VALUES(28,'denghui','quality
control',8000);
INSERT INTO new_bonuses_depa1(staff_id, staff_name, job, bonus) VALUES(25,'liulili','quality
control',10000);
```

```
INSERT INTO new_bonuses_depa1(staff_id, staff_name, job, bonus) VALUES(21,'caoming','document developer',12000);
```

注意事项：

本步骤中的 insert 语句是值插入，即构造一行记录并插入到表中。

INSERT 语句所指定的字段名必须是表中已存在的字段名。

如果 INSERT 语句所指定的字段名包含表中的所有字段，则可能省略字段名。

查询表 bonuses_depa1 中的数据。输入：

```
SELECT * FROM bonuses_depa1;
```

返回结果：

	staff_id	staff_name	job	bonus
1	23	wangxia	developer	5000
2	24	limingying	tester	7000
3	25	liulili	quality control	8000
4	29	liuxue	tester	8000
5	21	caoming	document developer	11000

查询表 new_bonuses_depa1 中的数据。输入：

```
SELECT * FROM new_bonuses_depa1;
```

返回结果：

	staff_id	staff_name	job	bonus
1	23	wangxia	developer	7000
2	27	wangxuefen	document developer	7000
3	28	denghui	quality control	8000
4	25	liulili	quality control	10000
5	21	caoming	document developer	12000

比较两表 new_bonuses_depa1 和 bonuses_depa1 的津贴数据，对比员工津贴变化情况。输入：

```
SELECT bd.STAFF_NAME,
CASE
WHEN nbd.BONUS > bd.BONUS THEN 'increased'
WHEN nbd.BONUS = bd.BONUS THEN 'equal'
ELSE 'decreased' END AS DIFF
FROM new_bonuses_depa1 nbd, bonuses_depa1 bd
WHERE nbd.STAFF_ID = bd.STAFF_ID
```

返回结果：

	staff_name	DIFF
1	wangxia	increased
2	liulili	increased
3	caoming	increased

删除表。输入：

```
DROP TABLE bonuses_depa1;
DROP TABLE new_bonuses_depa1;
```

IFNULL

本步骤主要实现：返回员工编号和工资列表，工资如果为 NULL 则替换为“unknown”。

删除同名表 staffs_tab。输入：

```
DROP TABLE IF EXISTS staffs_tab;
```

创建表 staffs_tab。输入：

```
CREATE TABLE staffs_tab
(
staff_ID NUMERIC(6) not null,
NAME VARCHAR(20),
EMAIL VARCHAR(25),
PHONE_NUMBER VARCHAR(20),
HIRE_DATE DATE,
employment_ID VARCHAR(10),
SALARY NUMERIC(8,2),
MANAGER_ID NUMERIC(6),
section_ID NUMERIC(4)
);
```

向表 staffs_tab 中插入记录 1。输入：

```
INSERT INTO staffs_tab (staff_ID, NAME, EMAIL, PHONE_NUMBER, HIRE_DATE, employment_ID,
SALARY,MANAGER_ID, section_ID)
values (198, '王莹', 'wangying@126.com', '18095605632', date_format('19990621', '%Y%m%d'),
'SH_CLERK',NULL, 124, 50);
```

向表 staffs_tab 中插入记录 2。输入：

```
INSERT INTO staffs_tab (staff_ID, NAME, EMAIL, PHONE_NUMBER, HIRE_DATE, employment_ID,
SALARY,MANAGER_ID, section_ID)
values (199, '何开平', 'hekaipng02@126.com', '18095605532', date_format('20000113',
'%Y%m%d'),'SH_CLERK', 2600.00, 124, 50);
```

向表 staffs_tab 中插入记录 3。输入：

```
INSERT INTO staffs_tab (staff_ID, NAME, EMAIL, PHONE_NUMBER, HIRE_DATE, employment_ID,
SALARY,MANAGER_ID, section_ID)
values (200, '李瑞', 'lirui03@126.com', '18095565632', date_format('19870917', '%Y%m%d'),
'AD_ASST',4400.00, 101, 10);
```

返回员工编号和工资列表，工资如果为 NULL 则替换为“unknown”。输入：

SELECT staff_ID, IFNULL(SALARY, 'unknown') SALARY FROM staffs_tab WHERE staff_ID IS NOT NULL ORDER BY staff_ID;

返回结果：

	staff_ID	SALARY
1	198	unknown
2	199	2600.00
3	200	4400.00

删除表。输入：

DROP TABLE staffs_tab;

2.2.5 JSON 函数

步骤 1 JSON 创建函数

JSON_ARRAY()：创建 JSON 数组

输入：

SELECT JSON_ARRAY(1, "abc", NULL, TRUE, CURDATE());

返回结果：

	JSON_ARRAY(1,...
1	[1, "abc", null, true, "2020-10-28"]

JSON_OBJECT()：创建 JSON 对象

输入：

SELECT JSON_OBJECT('id', 87, 'name', 'GaussDB');

返回结果：

	JSON_OBJECT('...
1	{"id": 87, "name": "GaussDB"}

步骤 2 JSON 搜索函数

JSON_CONTAINS(target, candidate[, path])：判断是否包含特定的 JSON 文档，返回值为 0 或 1。

输入：

```
SET @j = '{"a": 1, "b": 2, "c": {"d": 4}}';
SET @j2 = '{"d": 4}';
SELECT JSON_CONTAINS(@j, @j2, '$.a');
```

返回结果：

	JSON_CONTAINS...
1	0

输入：

```
SELECT JSON_CONTAINS(@j, @j2, '$.c');
```

返回结果：

	JSON_CONTAINS...
1	1

JSON_CONTAINS_PATH(json_doc, one_or_all, path[, path] ...)：判断给定 path/paths 中是否包含 JSON 文档，返回值为 0 或 1。

输入：

```
SET @j = '{"a": 1, "b": 2, "c": {"d": 4}}';
SELECT JSON_CONTAINS_PATH(@j, 'one', '$.c.d');
```

返回结果：

	JSON_CONTAINS...
1	1

输入：

```
SELECT JSON_CONTAINS_PATH(@j, 'one', '$.a.d');
```

返回结果：

	JSON_CONTAINS...
1	0

JSON_EXTRACT(json_doc, path[, path] ...)：从 JSON 文档中返回特定路径的内容。

输入：

```
SELECT JSON_EXTRACT('[10, 20, [30, 40]]', '$[2][*]');
```

返回结果：

	JSON_EXTRACT(...
1	[30, 40]

JSON_KEYS(json_doc[, path])：返回 JSON 对象的 KEY 值，返回一个数组。

输入：

```
SELECT JSON_KEYS('{"a": 1, "b": {"c": 30}}');
```

返回结果：

	JSON_KEYS('{"...
1	["a", "b"]

步骤 3 JSON 修改函数

JSON_ARRAY_APPEND(json_doc, path, val[, path, val] ...)：将值追加到 JSON 文档中指定数组的末尾并返回结果。

输入：

```
SET @j = '{"a": 1, "b": [2, 3], "c": 4}';
SELECT JSON_ARRAY_APPEND(@j, '$.b', 'x');
```

返回结果：

	JSON_ARRAY_AP...
1	{"a": 1, "b": [2, 3, "x"], "c": 4}

输入：

```
SELECT JSON_ARRAY_APPEND(@j, '$.c', 'y');
```

返回结果：

	JSON_ARRAY_AP...
1	{"a": 1, "b": [2, 3], "c": [4, "y"]}

输入：

```
SET @j = '{"a": 1}';
SELECT JSON_ARRAY_APPEND(@j, '$', 'z');
```

返回结果：

	JSON_ARRAY_AP...
1	[{"a": 1}, "z"]

JSON_ARRAY_INSERT(json_doc, path, val[, path, val] ...)：更新 JSON 文档，向文档中插入数组并返回修改后的文档。

输入：

```
SET @j = '["a", {"b": [1, 2]}, [3, 4]]';
SELECT JSON_ARRAY_INSERT(@j, '$[1].b[0]', 'x');
```

返回结果：

	JSON_ARRAY_IN...
1	["a", {"b": ["x", 1, 2]}, [3, 4]]

输入：

```
SELECT JSON_ARRAY_INSERT(@j, '$[0]', 'x', '$[2][1]', 'y');
```

返回结果：

	JSON_ARRAY_IN...
1	["x", "a", {"b": [1, 2]}, [3, 4]]

JSON_INSERT(json_doc, path, val[, path, val] ...)：向 JSON 文档中插入数据并返回结果。如果路径下已包含值，则不进行插入操作，即不会覆盖已经存在的值。

输入：

```
SET @j = '{ "a": 1, "b": [2, 3]}';
SELECT JSON_INSERT(@j, '$.a', 10, '$.c', '[true, false]');
```

返回结果：

	JSON_INSERT(@...
1	{"a": 1, "b": [2, 3], "c": "[true, false]"}

输入：

```
SELECT JSON_INSERT(@j, '$.a', 10, '$.c', CAST('[true, false]' AS JSON));
```

返回结果：

	JSON_INSERT(@...
1	{"a": 1, "b": [2, 3], "c": [true, false]}

JSON_SET(json_doc, path, val[, path, val] ...)：向 JSON 文档中插入或更新数据并返回结果。

说明：

JSON_SET()：替换已存在的值，添加不存在的值。

JSON_INSERT()：若已存在值则忽略，若不存在则插入新值。

JSON_REPLACE()：只替换已存在的值。

输入：

```
SET @j = '{ "a": 1, "b": [2, 3]}';
SELECT JSON_SET(@j, '$.a', 10, '$.c', '[true, false]');
```

返回结果：

	JSON_SET(@j, ...
1	{"a": 10, "b": [2, 3], "c": "[true, false]"}

输入：

```
SELECT JSON_INSERT(@j, '$.a', 10, '$.c', '[true, false]');
```

返回结果：

	JSON_INSERT(@...
1	{"a": 1, "b": [2, 3], "c": "[true, false]"}

输入：

```
SELECT JSON_REPLACE(@j, '$.a', 10, '$.c', '[true, false]');
```

返回结果：

	JSON_REPLACE(...
1	{"a": 10, "b": [2, 3]}

步骤 4　JSON 属性函数

JSON_DEPTH(json_doc)：返回 JSON 文档最大深度。

输入：

```
SELECT JSON_DEPTH('[10, {"a": 20}]');
```

返回结果：

	JSON_DEPTH('[...
1	3

JSON_LENGTH(json_doc[, path])：返回 JSON 文档的长度，若指定了路径，则返回指定路径元素长度。

输入：

```
SELECT JSON_LENGTH('[1, 2, {"a": 3}]');
```

返回结果：

	JSON_LENGTH('...
1	3

输入：

```
SELECT JSON_LENGTH('{"a": 1, "b": {"c": 30}}', '$.b');
```

返回结果：

	JSON_LENGTH('...
1	1

2.2.6 窗口函数

删除重名表 staffs。输入：

```
DROP TABLE if exists staffs;
```

创建表（以表 staffs 为例）。输入：

```
CREATE TABLE staffs
(
staff_id INT not null,
first_name VARCHAR(20),
last_name VARCHAR(25),
email VARCHAR(25),
phone_number VARCHAR(20),
hire_date DATE,
employment_id VARCHAR(10),
salary NUMERIC(8,2),
commission_pct NUMERIC(2,2),
manager_id NUMERIC(6),
section_id NUMERIC(4)
);
```

向表 staffs 中插入记录。输入：

```
insert into staffs values (198, 'Donald', 'OConnell', 'DOCONNEL', '650.507.9833', date_format('19990622', '%Y%m%d'), 'SH_CLERK', 2200.00, null, 124, 50);
insert into staffs values (198, 'Donald', 'OConnell', 'DOCONNEL', '650.507.9833', date_format('19990622', '%Y%m%d'), 'SH_CLERK', 2400.00, null, 124, 50);
insert into staffs values (198, 'Donald', 'OConnell', 'DOCONNEL', '650.507.9833', date_format('19990622', '%Y%m%d'), 'SH_CLERK', 2600.00, null, 124, 50);
insert into staffs values (199, 'Douglas', 'Grant', 'DGRANT', '650.507.9844', date_format('20000113', '%Y%m%d'), 'SH_CLERK', 4000.00, null, 124, 50);
insert into staffs values (199, 'Douglas', 'Grant', 'DGRANT', '650.507.9844', date_format('20000113', '%Y%m%d'), 'SH_CLERK', 4200.00, null, 124, 50);
insert into staffs values (200, 'Jennie', 'Grant', 'SMITH', '650.507.9855', date_format('20010327', '%Y%m%d'), 'HZ_CLERK', 5000.00, null, 124, 51);
insert into staffs values (200, 'Jennie', 'Grant', 'SMITH', '650.507.9855', date_format('20010327', '%Y%m%d'), 'HZ_CLERK', 5000.00, null, 124, 51);
insert into staffs values (200, 'Jennie', 'Grant', 'SMITH', '650.507.9855', date_format('20010327', '%Y%m%d'), 'HZ_CLERK', 5100.00, null, 124, 51);
insert into staffs values (200, 'Jennie', 'Grant', 'SMITH', '650.507.9855', date_format('20010327', '%Y%m%d'), 'HZ_CLERK', 5200.00, null, 124, 51);
```

查询表 staffs 的数据。输入：

```
SELECT * FROM staffs;
```

返回结果：

staff_id	first_name	last_name	email	phone_number	hire_date	employment_id	salary	commission_pct	manager_id	section_id	staff_id
1	198	Donald	OConnell	DOCONNEL	650.507.9833	1999-06-22	SH_CLERK	2200.00		124	1
2	198	Donald	OConn	DOCONNEL	650.507.98	1999-06-22	SH_CLE	2400.00		124	2

			ell		33		RK				
3	198	Donald	OConnell	DOCONNEL	650.507.9833	1999-06-22	SH_CLERK	2600.00		124	3
4	199	Douglas	Grant	DGRANT	650.507.9844	2000-01-13	SH_CLERK	4000.00		124	4
5	199	Douglas	Grant	DGRANT	650.507.9844	2000-01-13	SH_CLERK	4200.00		124	5
6	200	Jennie	Grant	SMITH	650.507.9855	2001-03-27	HZ_CLERK	5000.00		124	6
7	200	Jennie	Grant	SMITH	650.507.9855	2001-03-27	HZ_CLERK	5000.00		124	7

调用窗口函数 RANK。输入：

```
select staff_ID,salary,rank() over(partition by staff_ID order by salary) as rank_result from staffs;
```

返回结果：

	staff_id	salary	rank_result
1	198	2200.00	1
2	198	2400.00	2
3	198	2600.00	3
4	199	4000.00	1
5	199	4200.00	2
6	200	5000.00	1
7	200	5000.00	1
8	200	5100.00	3
9	200	5200.00	4

调用窗口函数 ROW_NUMBER。输入：

```
select staff_ID,salary,ROW_NUMBER() over(partition by staff_ID order by salary) as row_number_result
from staffs;
```

返回结果：

	staff_id	salary	row_number_result
1	198	2200.00	1
2	198	2400.00	2
3	198	2600.00	3
4	199	4000.00	1
5	199	4200.00	2
6	200	5000.00	1
7	200	5000.00	2
8	200	5100.00	3
9	200	5200.00	4

调用窗口函数 DENSE_RANK。输入：

```
select staff_ID,salary,DENSE_RANK() over(partition by staff_ID order by salary) as row_number_result from staffs;
```

返回结果：

	staff_id	salary	row_number_result
1	198	2200.00	1
2	198	2400.00	2
3	198	2600.00	3
4	199	4000.00	1
5	199	4200.00	2
6	200	5000.00	1
7	200	5000.00	1
8	200	5100.00	2
9	200	5200.00	3

调用窗口函数 LAG。输入：

```
select staff_ID,salary,lag(salary,1,0) over(partition by staff_ID order by staff_ID) as 'lag' from staffs;
```

返回结果：

	staff_id	salary	lag
1	198	2200.00	0.00
2	198	2400.00	2200.00
3	198	2600.00	2400.00
4	199	4000.00	0.00
5	199	4200.00	4000.00
6	200	5000.00	0.00
7	200	5000.00	5000.00
8	200	5100.00	5000.00
9	200	5200.00	5100.00

调用窗口函数 LEAD。输入：

```
select staff_ID,salary,lead(salary,1,0) over(partition by staff_ID order by staff_ID) as 'lag' from staffs;
```

返回结果：

	staff_id	salary	lag
1	198	2200.00	2400.00
2	198	2400.00	2600.00
3	198	2600.00	0.00
4	199	4000.00	4200.00
5	199	4200.00	0.00
6	200	5000.00	5000.00
7	200	5000.00	5100.00
8	200	5100.00	5200.00
9	200	5200.00	0.00

使用聚集函数。输入：

```
select staff_ID,salary,AVG(salary) over(partition by staff_ID) as avg_result,MAX(salary) over(partition by staff_ID) as max_result from staffs;
```

返回结果：

	staff_id	salary	avg_result	max_result
1	198	2200.00	2400.000000	2600.00
2	198	2400.00	2400.000000	2600.00
3	198	2600.00	2400.000000	2600.00
4	199	4000.00	4100.000000	4200.00
5	199	4200.00	4100.000000	4200.00
6	200	5000.00	5075.000000	5200.00
7	200	5000.00	5075.000000	5200.00
8	200	5100.00	5075.000000	5200.00
9	200	5200.00	5075.000000	5200.00

2.2.7 存储过程

GaussDB(for MySQL)中，stored routine 指的是存储过程或者是函数。存储过程是一组为了完成特定功能的 SQL 语句集，经过编译后存储在数据库中。函数是一种与存储过程十分相似的过程式数据库对象。它与存储过程一样，都是由 SQL 语句和过程式语句组成的代码片段，并且可以被应用程序和其他 SQL 语句调用。

步骤 1 存储过程参数

IN 输入参数。输入：

```
# 创建存储过程
delimiter //
create procedure in_param(in p_in int)
begin
    select p_in;
    set p_in=2;
    select p_in;
end //
delimiter ;
# 设置变量
set @p_in=1;

# 调用存储过程
call in_param(@p_in);

# 查看变量值
select @p_in;
```

返回结果：

结果集 1：

	p_in
1	1

结果集 2：

	p_in
1	2

结果集 3：

	@p_in
1	1

OUT 输出参数。输入：

```
# 创建存储过程
delimiter //
create procedure out_param(out p_out int)
begin
        select p_out;
        set p_out=2;
        select p_out;
end //
delimiter ;

# 设置变量
set @p_out=1;

# 调用存储过程
call out_param(@p_out);

# 查看变量值
select @p_out;
```

返回结果：

结果集 1：

	p_out
1	

结果集 2：

	p_out
1	2

结果集 3：

	@p_out
1	2

INOUT 输入输出参数。输入：

```
# 创建存储过程
delimiter //
create procedure inout_param(inout p_inout int)
begin
        select p_inout;
        set p_inout=2;
        select p_inout;
    end
 //
delimiter ;

# 设置变量
set @p_inout=1;
# 调用存储过程
call inout_param(@p_inout);
# 查看变量值
select @p_inout;
```

返回结果：

结果集 1：

	p_inout
1	1

结果集 2：

	p_inout
1	2

结果集 3：

	@p_inout
1	2

步骤 2　变量

局部变量。输入：

```
# 定义存储过程
DELIMITER //
  CREATE PROCEDURE proc_v()
   begin   declare v1 varchar(10) default 'welcome';
       select v1;
set v1 = 'GaussDB';
       select v1;
   end
   //
   DELIMITER ;

# 调用存储过程
call proc_v();
```

返回结果：

结果集 1：

	v1
1	welcome

结果集 2：

	v1
1	GaussDB

用户变量。输入：

```
DELIMITER //
CREATE PROCEDURE proc1()
BEGIN
   SET @para_procedure='proc1';
   SELECT @para_procedure;
END
//
CALL proc1();
```

返回结果：

	@para_procedure
1	proc1

输入：

```
CREATE PROCEDURE proc2()
SELECT CONCAT('the parameter of procedure is ',@para_procedure);
CALL proc2();
```

返回结果：

	CONCAT('the paramete...
1	the parameter of procedure is proc1

步骤 3　IF 语句

输入：

```
# 创建表并插入记录
drop table if exists test_if;
create table test_if(col int);
insert into test_if values(1);

# 创建存储过程
DELIMITER //
    CREATE PROCEDURE if_proc(IN par int)
    begin
declare var int;
    set var=par+1;
    if var=0 then    insert into test_if values(17);
    end if;
    if par=0 then    update test_if set col=col+1;
    else
    update test_if set col=col+2;
    end if;
    end
    //
    DELIMITER ;
# 调用存储过程
 call if_proc(3);
# 查看表数据
 select * from test_if;
```

返回结果：

	col
1	3

输入：

```
# 调用存储过程
call if_proc(-1);
# 查看表数据
select * from test_if;
```

返回结果：

	col
1	5
2	19

步骤 4 CASE 语句

输入：

```
# 创建表
drop table if exists test_case;
create table test_case(col int);

# 创建存储过程
DELIMITER //
  CREATE PROCEDURE case_proc (in par int)
  begin
  declare var int;
  set var=par+1;
  case var
  when 0 then
  insert into test_case values(17);
  when 1 then
  insert into test_case values(18);
  else
  insert into test_case values(19);
  end case;
  end
  //
  DELIMITER ;

# 调用存储过程
call case_proc(5);

# 查看表数据
select * from test_case;
```

返回结果：

	col
1	19

步骤 5 WHILE 循环语句

输入：

```
# 创建表
```

```
drop table if exists test_while;
create table test_while(col int);

# 创建存储过程
DELIMITER //
    CREATE PROCEDURE while_proc()
    begin   declare var int;
    set var=0;
    while var<6 do
    insert into test_while values(var);
    set var=var+1;
    end while;
    end
    //
    DELIMITER ;

# 调用存储过程
 call while_proc();

# 查看表数据
 select * from test_while;
```

返回结果：

	col
1	0
2	1
3	2
4	3
5	4
6	5

步骤 6　REPEAT 循环语句

输入：

```
# 创建表
drop table if exists test_repeat;
create table test_repeat(col int);

# 创建存储过程
DELIMITER //
    CREATE PROCEDURE repeat_proc ()
    begin
```

```
    declare var int;
    set var=0;
    repeat
    insert into test_repeat values(var);
    set var=var+1;
    until var>=5
    end repeat;
    end
    //
    DELIMITER ;

# 调用存储过程
 call repeat_proc();

# 查看表数据
 select * from test_repeat;
```

返回结果：

	col
1	0
2	1
3	2
4	3
5	4

步骤 7　LOOP 循环语句

输入：

```
# 创建表
 drop table if exists test_loop;
 create table test_loop(col int);

# 创建存储过程
DELIMITER //
    CREATE PROCEDURE loop_proc ()
    begin
    declare var int;
    set var=0;
    LOOP_LABLE:loop
    insert into test_loop values(var);
    set var=var+1;
    if var >=5 then
```

```
    leave LOOP_LABLE;
    end if;
    end loop;
    end;
    //
    DELIMITER ;

# 调用存储过程
 call loop_proc();

# 查看表数据
 select * from test_loop;
```

返回结果：

	col
1	0
2	1
3	2
4	3
5	4

步骤 8　查看数据库中的存储过程

输入：

```
select routine_name from information_schema.routines where routine_schema='advanced_prac';
```

注意：请将 advanced_prac 替换为实际的库名。

返回结果：

	ROUTINE_NAME
1	case_proc
2	if_proc
3	inout_param
4	in_param
5	loop_proc
6	out_param
7	proc1

8	proc2
9	proc_v
10	repeat_proc
11	while_proc

步骤 9 删除存储过程

输入：

```
DROP PROCEDURE case_proc;
DROP PROCEDURE if_proc;
DROP PROCEDURE loop_proc;
DROP PROCEDURE repeat_proc;
DROP PROCEDURE while_proc;
DROP PROCEDURE proc1;
DROP PROCEDURE proc2;
DROP PROCEDURE proc_v;
DROP PROCEDURE inout_param;
DROP PROCEDURE in_param;
DROP PROCEDURE out_param;
```

2.2.8 触发器

此小节主要通过触发器来完成：增加一个商品信息的时候，往日志表中插入一条自定义记录。

步骤 1 创建触发器

输入：

```
# 创建商品表
drop table if exists goods;
create table goods(id int,name varchar(100), g_desc varchar(200));
# 创建日志表
drop table if exists goods_log;
create table goods_log(insert_time datetime, content varchar(1024));
# 创建触发器
DELIMITER //
CREATE TRIGGER goods_insert_log after INSERT on goods for each row
    BEGIN
        DECLARE logInfo VARCHAR(1024);
        set logInfo = CONCAT('id:',new.id,'添加成功');
        insert into goods_log(insert_time, content) VALUES(now(), logInfo);
    END
//
```

```
DELIMITER ;
# 往 goods 表中插入一条记录
INSERT INTO goods VALUES (1,'Apple','苹果');
# 查询 goods 表数据
select * from goods;
# 查询 goods_log 表数据
select * from goods_log;
```

返回结果：

结果集 1：

	id	name	g_desc
1	1	Apple	苹果

结果集 2：

	insert_time	content
1	2020-10-28 14:44:36	id:1 添加成功

由此可以看到，该触发器成功实现了：当 goods 表中插入新记录时，相应的插入时间等信息会记录到 goods_log 表中。

步骤 2　查看触发器

此外，可以通过 SHOW TRIGGERS 来查看数据库中已经创建的触发器。

输入：

```
SHOW TRIGGERS
```

返回结果（此处只截取了部分列的内容）：

	Trigger	Event	Table	Statement	Timing
1	goods_insert_log	INSERT	goods	BEGIN DECLARE logInfo VARCHAR(1024); set logInfo = CONCAT('id:',new.id,'添加成功'); insert into goods_log(insert_time, content) VALUES(now(), logInfo); END	AFTER

步骤 3　删除触发器

输入：

```
DROP TRIGGER goods_insert_log;
```

2.3 实验小节

本实验主要完成了 GaussDB(for MySQL)中常见的函数、正则表达式、控制流函数、JSON 函数、窗口函数、存储过程及触发器的相关操作。

3 数据库日常操作实验

3.1 实验介绍

3.1.1 关于本实验

本实验通过修改数据库参数、用户管理实验、用户授权实验、备份恢复实验、系统表及监控指标查看、通过 JDBC 连接数据库等实验来介绍数据库日常操作。

3.1.2 实验目的

- 掌握数据库参数修改方法；
- 掌握用户管理及授权实验；
- 掌握数据库备份恢复方法；
- 掌握系统表及监控指标查看；
- 掌握 JDBC 连接数据库。

3.2 实验任务配置

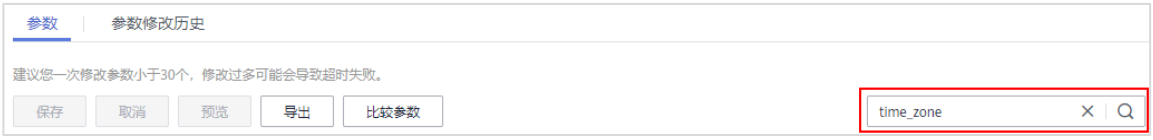
3.2.1 修改数据库参数

本实验以修改数据库时区为例，介绍如何修改数据库参数。

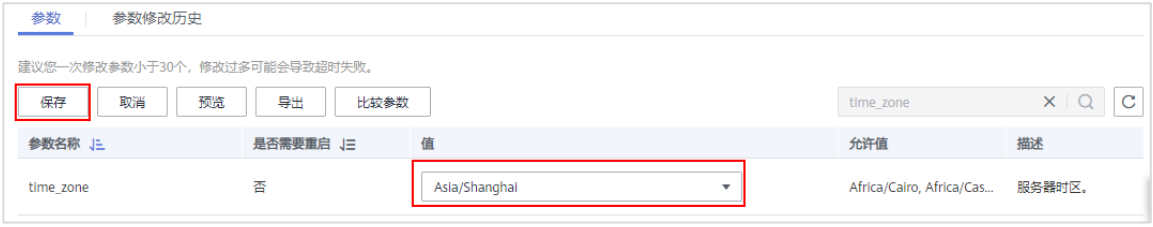
步骤 1 进入华为云云数据库 GaussDB(for MySQL)页面，在“实例管理”界面，查看右侧实例。

步骤 2 点击“更多”，在下拉菜单中选择“参数修改”。

步骤 3 在搜索栏中，输入“time_zone”进行搜索。



步骤 4 在“值”一栏中，将参数修改为“Asia/Shanghai”，点击保存，修改完毕。



步骤 5 点击“参数修改历史”标签，查看修改记录。

3.2.2 用户管理及授权实验

3.2.2.1 新建数据库

步骤 1 通过 DAS 登录云数据库 GaussDB(for MySQL)。

步骤 2 “新建数据库”按钮。

步骤 3 输入数据库名称，点击“确定”。

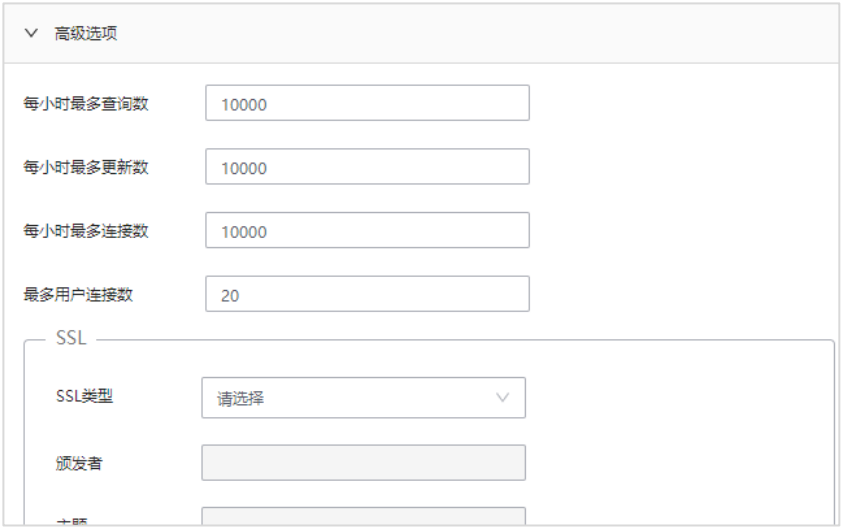
3.2.2.2 创建用户并授予权限

步骤 1 在菜单栏中选择“其他操作”选项，并在下拉菜单中选择“用户管理”。

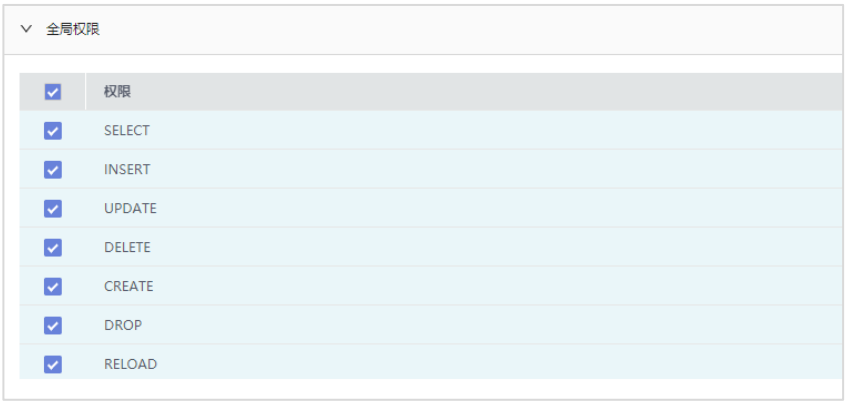
步骤 2 点击“新建用户”，完成“基本信息”、“高级选项”、“全局限制”、“对象权限”参数设置，具体参数根据实际情况而定。

基本信息：

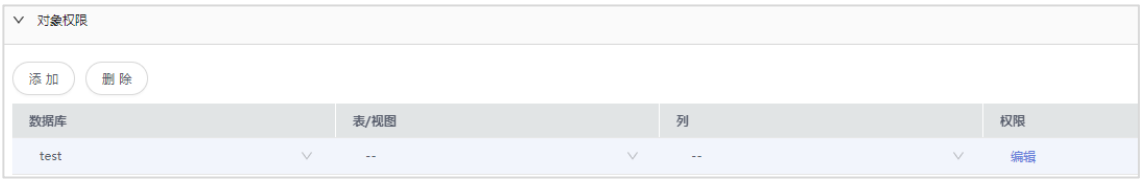
高级选项：



全局权限：



对象权限：点击“添加”，选择数据库对象，这里选择刚刚创建的数据库 test。

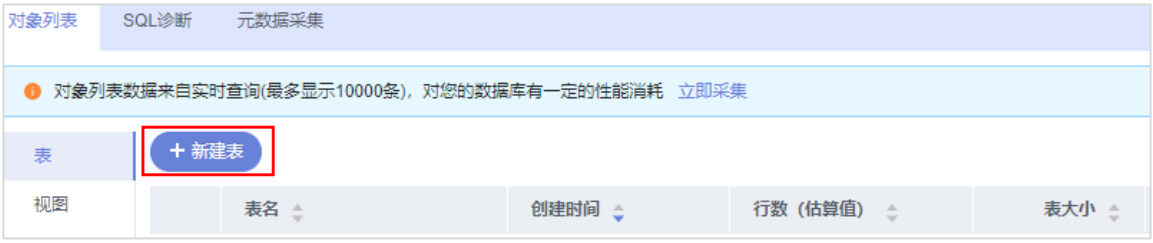


步骤 3 完成用户创建。

3.2.2.3 修改用户对象权限

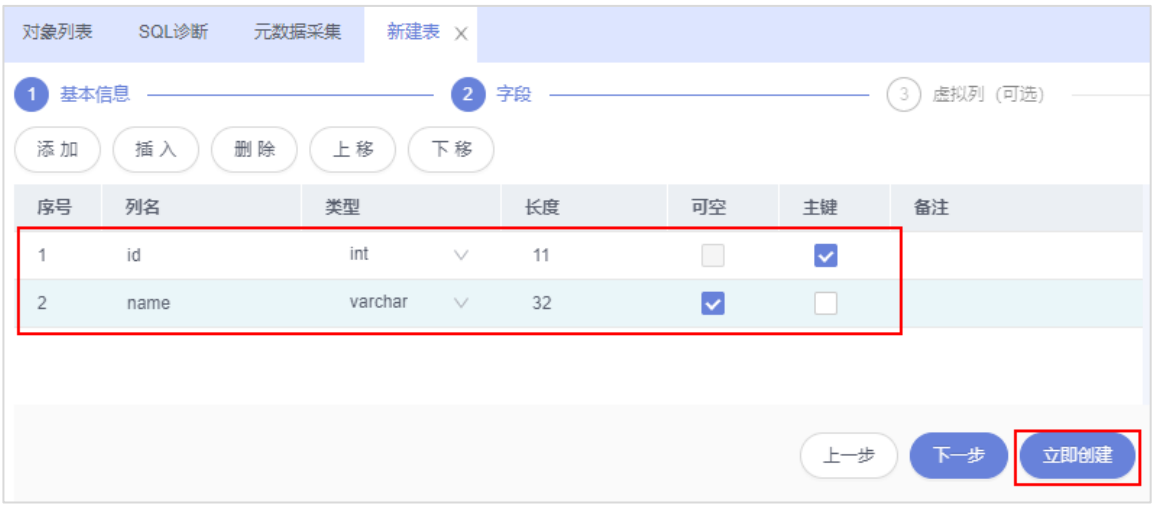
步骤 1 新建表，在数据库列表中，选择数据库“test”，点击“新建表”。

步骤 2 点击“新建表”。



步骤 3 输入表名“test1”，点击下一步。

步骤 4 在步骤“字段”中，新建列名 id 和 name，点击“立即创建”。

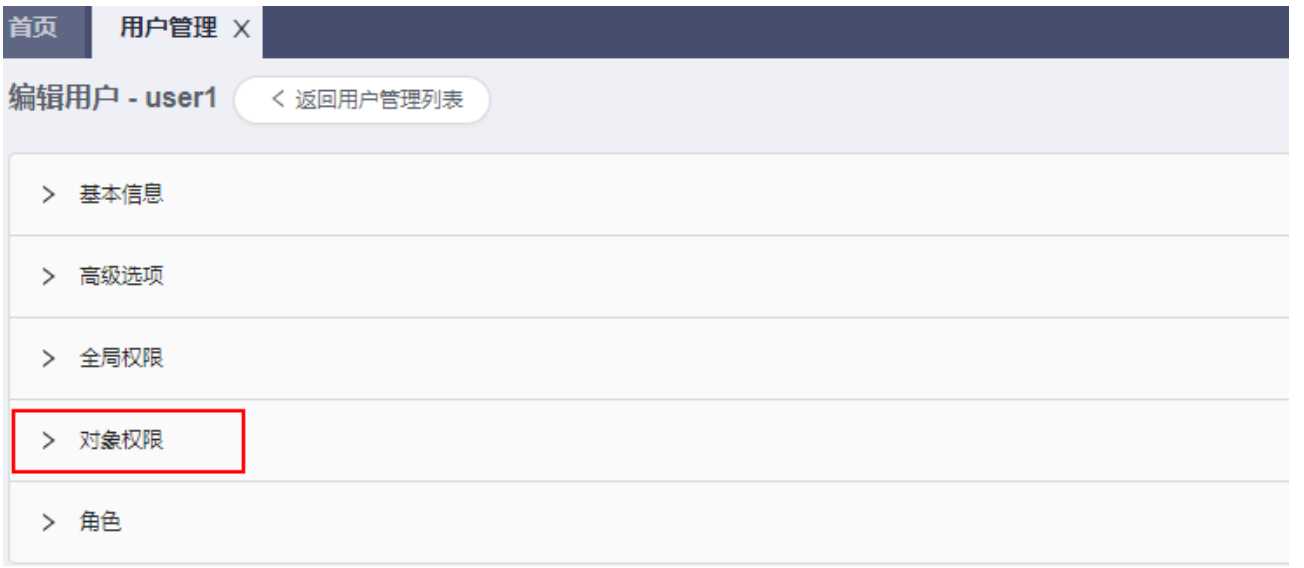


步骤 5 在“SQL 查询中”输入如下代码，添加测试数据。

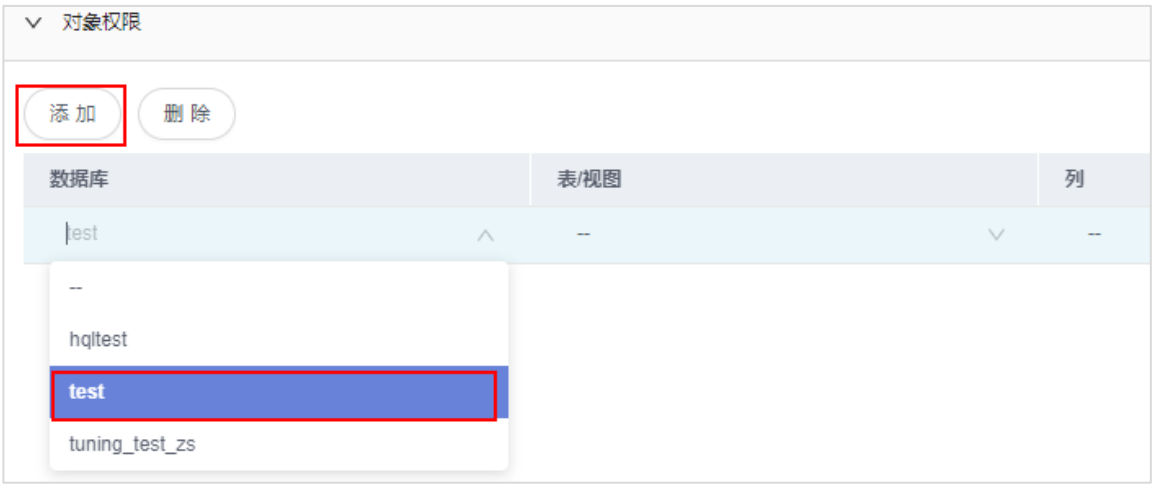
```
INSERT INTO test1 values(1,'TOM');
```

步骤 6 在“用户管理”页面中，查看新建用户“user1”，在“操作”一栏中，点击“编辑”。

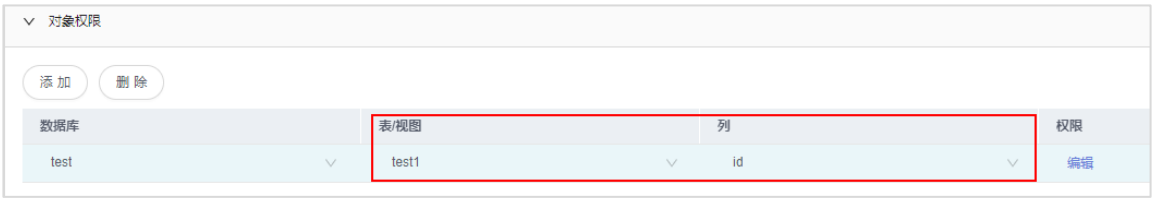
步骤 7 在“编辑用户”中选择“对象权限”。



步骤 8 点击“添加”，在“数据库”一栏中选择“test”数据库。

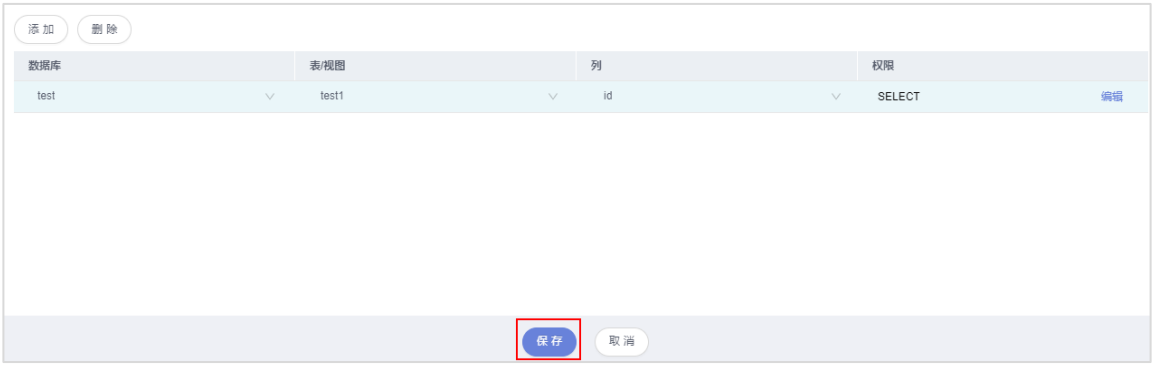


步骤 9 选择表 test 和列 id。



步骤 10 在“编辑”一栏中，修改权限，选择“SELECT”。

步骤 11 点击“保存”，完成修改。



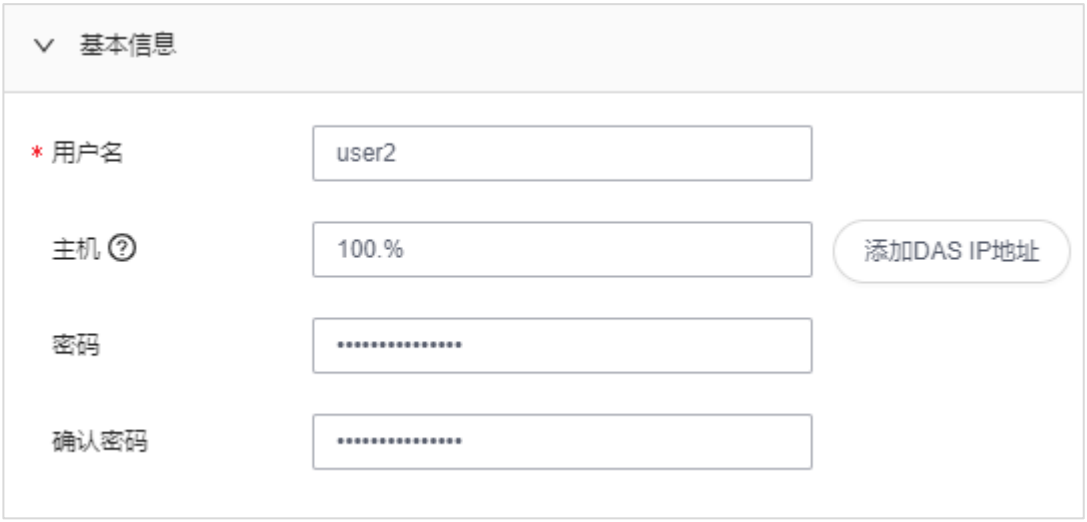
3.2.2.4 通过角色赋予权限

3.2.2.4.1 新建测试用户

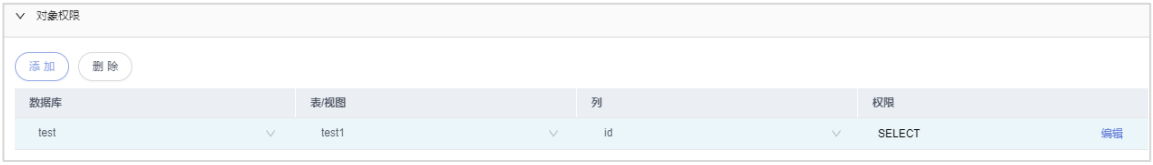
新建用户 user2、user3，将角色 user2 权限赋予 user3 并测试。

步骤 1 新建用户 user2，具体配置如下。

基本信息：



对象权限：



具体参数：

数据库：test

表/视图：test1

列：id

权限：SELECT

其它参数使用默认。

步骤 2 新建用户 user3，只配置“基本信息”即可，配置完点击“保存”。

3.2.2.4.2 测试用户 user2 权限

步骤 1 选择右上角华为云账号，点击“切换连接”。

步骤 2 输入用户名 user2 及密码，点击“登录”。

步骤 3 在首页页面，找到数据库 test，在“操作”一栏中点击“SQL 查询”。

步骤 4 输入如下查询语句进行测试。

```
SELECT * FROM test1;
```

结果如下：

步骤 5 输入另一条查询语句进行测试。

SELECT id FROM test1;

结果如下：

步骤 4 和步骤 5 验证了，user2 当前的对象权限是针对 test1 表中 id 这一列的 SELECT 查询权限。

3.2.2.4.3 测试用户 user3 权限

步骤 1 选择右上角华为云账号，点击“切换连接”。

步骤 2 输入用户名 user3 及密码，点击“登录”。

步骤 3 查看首页页面，发现并无 test 数据库。

由于 user3 没有设置对象权限因此无法对表 test 进行操作。

3.2.2.5 将角色 user2 权限赋予用户 user3

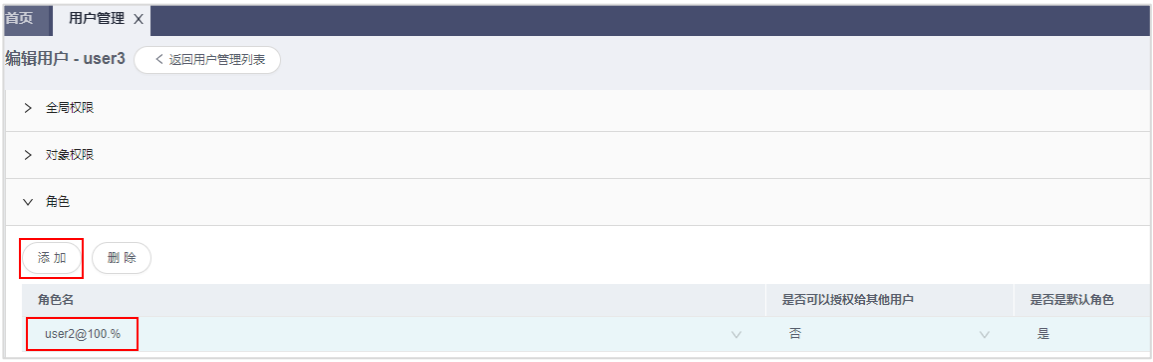
步骤 1 切换用户，用 root 账户登录数据库实例。

步骤 2 在“SQL 查询”中输入如下语句。

```
grant role_admin on *.* to current_user();
```

步骤 3 在“用户管理”页面，找到 user3 用户，点击“编辑”。

步骤 4 在“角色”一栏下，点击“添加”，选择角色“user2@100.%”，点击自动“保存”。



步骤 5 切换用户为 user3，输入如下 SQL 语句进行测试。

```
SELECT * FROM test1;
```

SELECT id FROM test1;

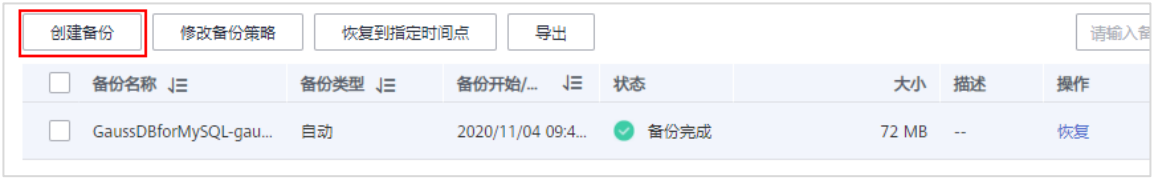
查询结果有反馈，表明 user3 被赋予了查询当前表中列名为 id 的权限。

3.3 数据库备份及恢复

步骤 1 在“云数据库 GaussDB”界面，点击实例名称，进入实例。

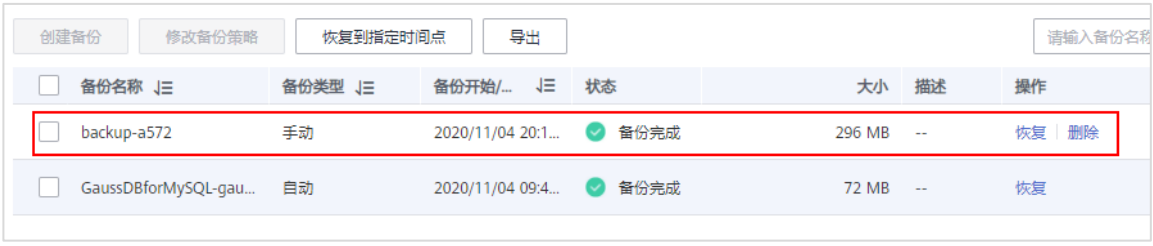
步骤 2 在左侧栏中点击“备份恢复”。

步骤 3 在“备份恢复-创建备份”页面，点击“创建备份”。

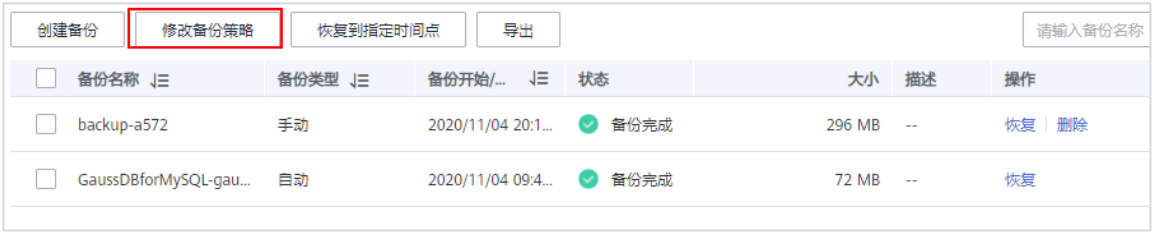


步骤 4 输入备份名称，点击“确定”。

步骤 5 查看备份列表。



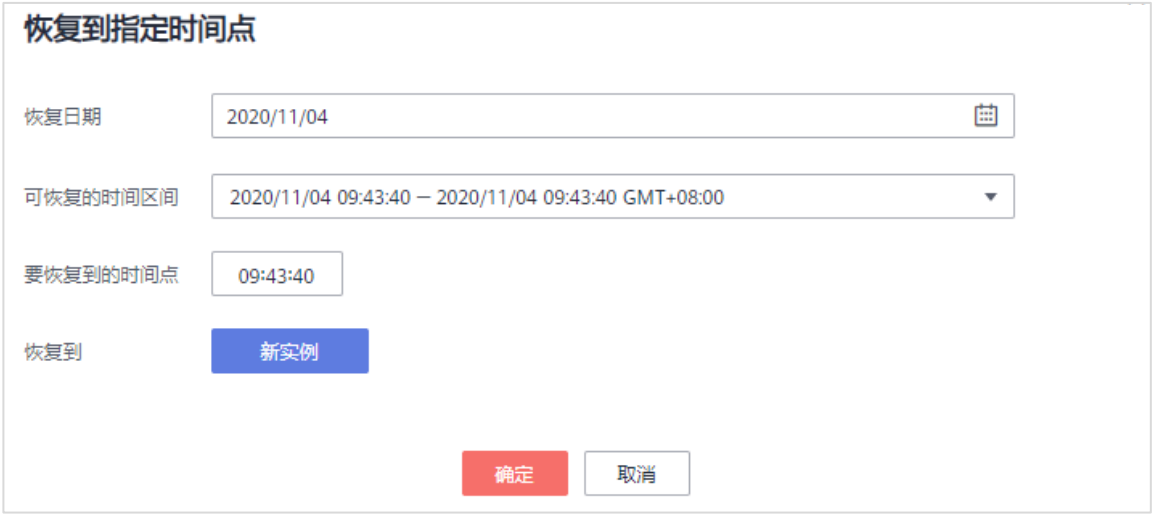
步骤 6 点击“修改备份策略”，进行备份策略修改。



步骤 7 根据实际业务需求，选择参数，点击“确定”，完成备份策略修改。

步骤 8 点击“恢复到制定时间点”，进行数据恢复。

步骤 9 设置时间点，点击“确定”，完成恢复。



注意：此处需要创建新实例。

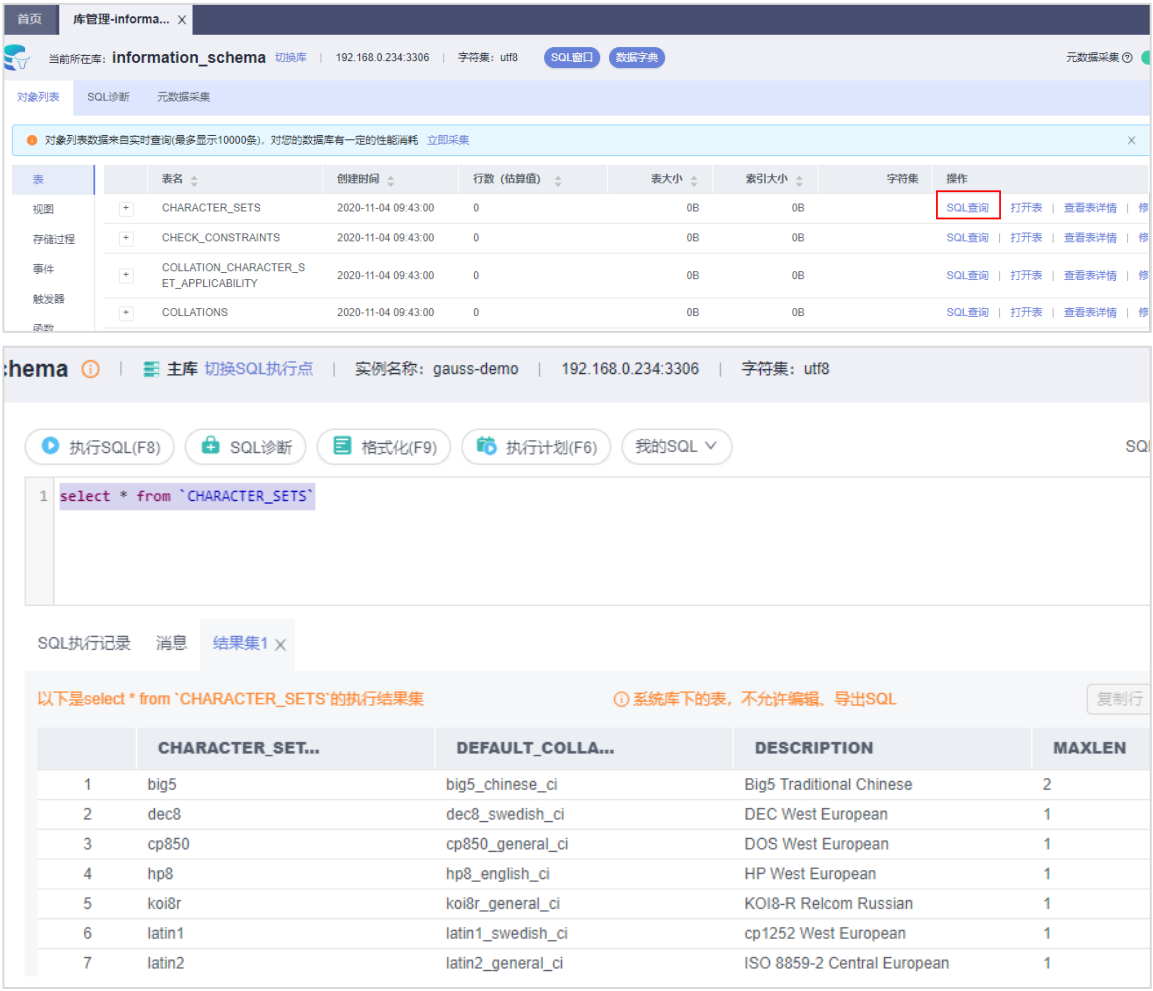
步骤 10 点击“导出”，导出备份信息。

3.4 查看系统表

3.4.1 查看 information_schema

步骤 1 在“首页”页面，点击“库管理”，进入“库管理-information_schema”页面。

步骤 2 在“库管理-information_schema”页面中，选择一个表名，点击“SQL 查询”，即可查询该表全部内容，等同于 SELECT *查询。



步骤 3 返回“对象列表”，选择一个表名，点击“打开表”，即可查看该表内容。

步骤 4 返回“对象列表”，点击“查看表详情”，即可查看该表结构。

3.4.2 查看其它系统表

步骤 1 点击“切换库”，选择“系统库”，点击“mysql”库，即可打开 mysql 库。

步骤 2 点击“切换库”，选择“系统库”，点击“performance_schema”库，即可打开 performance_schema 库。

步骤 3 点击“切换库”，选择“系统库”，点击“sys”库，即可打开 sys 库。

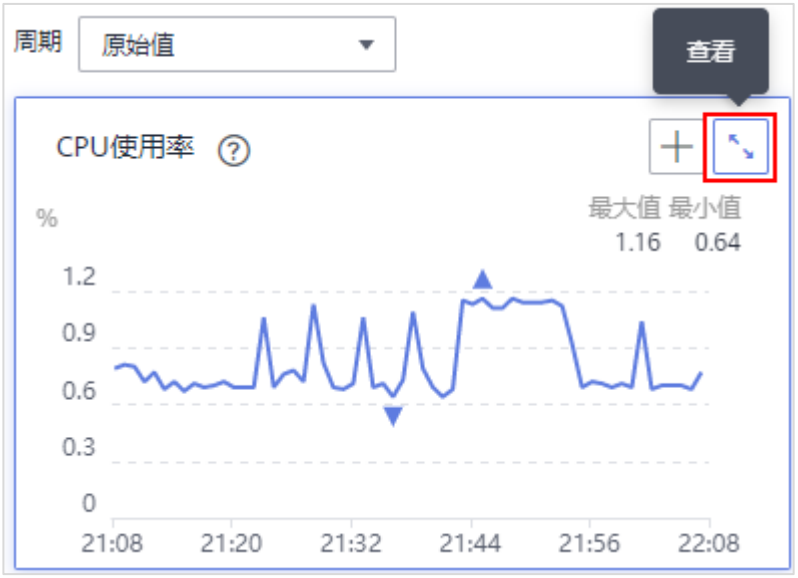
3.5 查看监控指标

步骤 1 在华为云主页搜索栏中，输入“云监控服务”，选择“云监控服务 ECS”，进入“云监控”服务信息页面。

步骤 2 在“云监控服务”中，选择“云服务监控”，点击“云数据库 GaussDB(for MySQL)”。

步骤 3 展开数据库实例，选择主节点，点击“查看监控指标”。

步骤 4 以“CPU 使用率”为例，点击“查看”图标，放大显示。



步骤 5 在“CPU 使用率”界面中，可通过选择时长，查看对应时间的监控数据。

步骤 6 点击“创建告警规则”图标，查看“创建告警规则”。

步骤 7 根据实际生成场景更改告警规则，修改完后，点击“立即创建”即可。

3.6 通过 JDBC 连接数据库

3.6.1 准备连接环境

步骤 1 Java 连接 GaussDB(for MySQL)可以使用 MySQL8.0 的驱动包，下载 Java 连接 MySQL 的驱动包，并将其导入对应的使用工具。

通过以下链接，下载驱动包

https://static.runoob.com/download/mysql-connector-java-8.0.16.jar

假设文件存放在 d:\Download 目录下。（后续 java 文件也放在同一目录下面）

步骤 2 创建测试数据库 demo

使用 DAS 新建数据库，库名为 demo，字符集为 utf-8。

步骤 3 创建测试表 websites

在“SQL 查询”页面输入如下语句：

```
CREATE TABLE `websites` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` char(20) NOT NULL DEFAULT '',
  `url` varchar(255) NOT NULL DEFAULT '',
  `alexa` int(11) NOT NULL DEFAULT '0' ,
  `country` char(10) NOT NULL DEFAULT '' ,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8;
```

步骤 4 插入数据

```
INSERT INTO `websites` VALUES
('1', 'openGauss', 'https://opengauss.org/zh/', '3', 'CN'),
('2', '华为云', 'https://www.huaweicloud.com/', '1', 'CN'),
('3', 'openEuler', 'https://openeuler.org/zh/', '4', ''),
('4', '华为 support 中心', 'https://support.huaweicloud.com/', '2', 'CN');
```

3.6.2 JDK 环境变量配置

步骤 1 根据系统情况下载 JDK 安装包，建议下载 JDK1.8。（链接：https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html）

步骤 2 双击安装包，根据安装向导完成 JDK 安装。

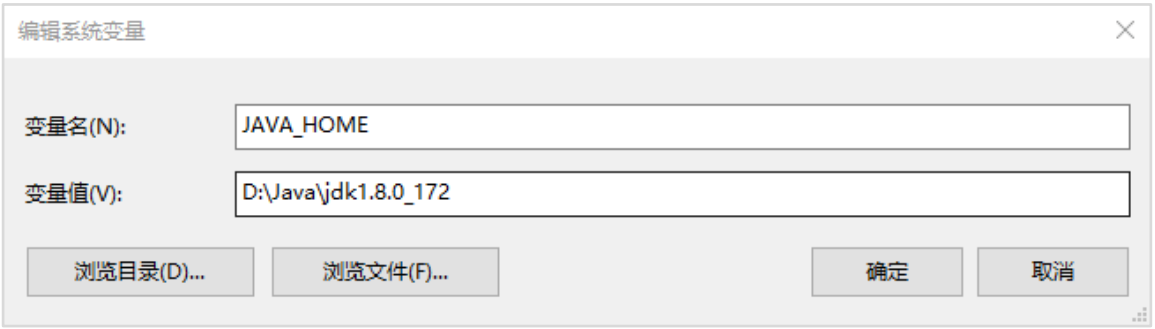
步骤 3 Windows 桌面右键单击“计算机”，选择“属性”。

步骤 4 在左侧导航中选择“高级系统设置”，弹出“系统属性”窗口。

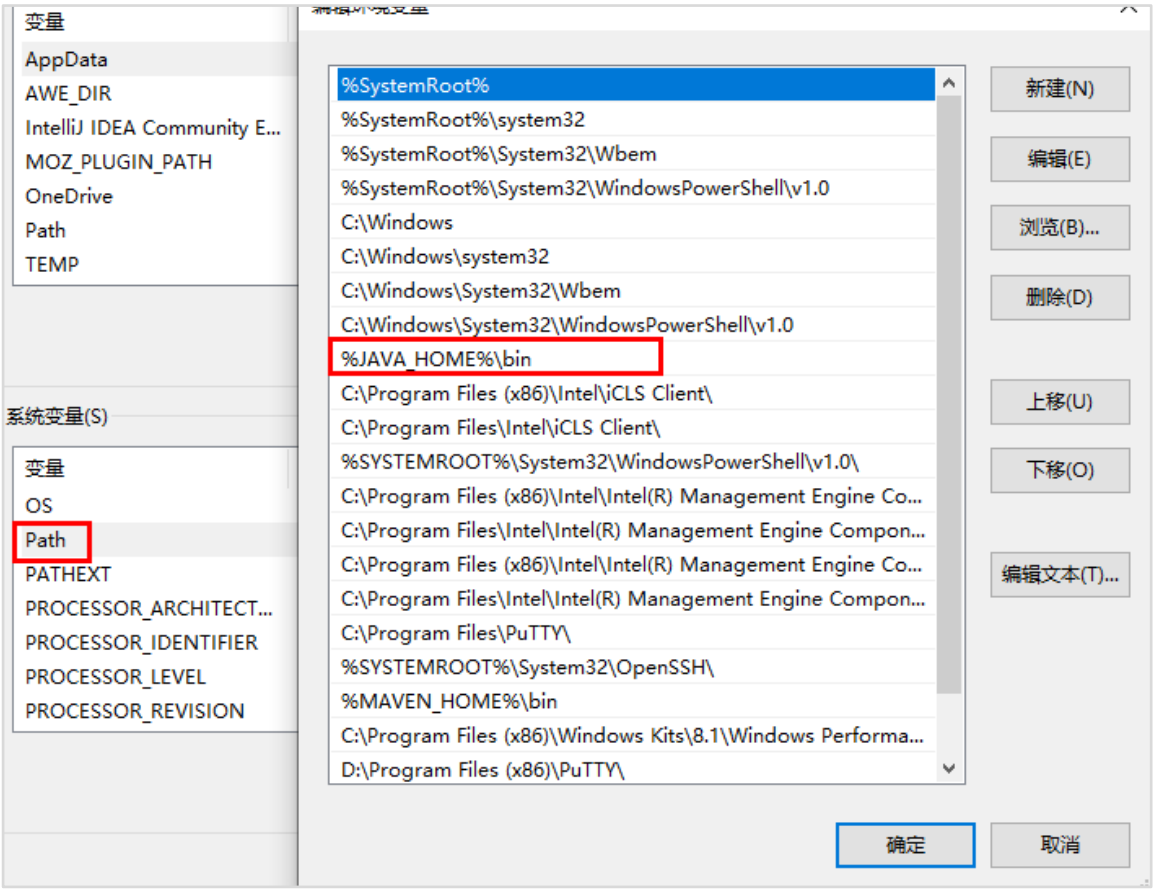
步骤 5 单击“环境变量”，弹出“环境变量”窗口。

步骤 6 配置环境变量。

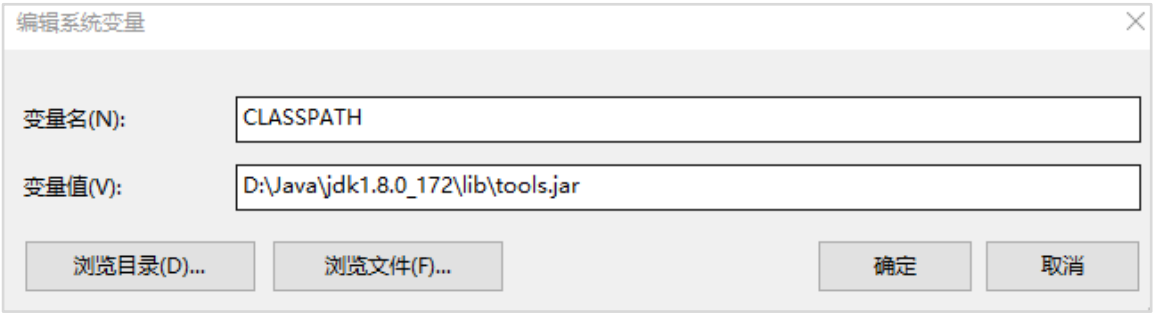
在“用户变量”下单击“新建”，新建用户变量，变量名为“JAVA_HOME”，变量值为 JDK 的安装路径，如“D:\Java\jdk1.8.0_172”。（此处路径根据实际情况设置）



在“系统变量”下选择“PATH”变量，单击“编辑”，将变量值末尾新增“;%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin”。（此处路径根据实际情况设置）



在“系统变量”下新建系统变量，变量名为“CLASSPATH”，变量值为“D:\Java\jdk1.8.0_172\lib\tools.jar”。（此处路径根据实际情况设置）



单击“确定”，完成环境变量的配置。

步骤 7 选择“开始”->“运行”，输入“cmd”，执行命令：java –version。

若命令执行成功，则说明环境变量配置成功。

3.6.3 连接 GaussDB(for MySQL)并执行 Java 代码

步骤 1 在 d:\Download 文件夹下新建 txt 文档，并以.java 格式保存。

步骤 2 使用 Java 程序连接数据库并进行查询

代码如下，需要将其中“公网 IP”、USER 和 PASS 标红处根据实际进行对应修改。

```
import java.sql.*;

public class GaussDBMySQLDemo {

    // MySQL 8.0 以上版本 - JDBC 驱动名及数据库 URL
    static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://公网
IP:3306/demo?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC";


    // 数据库的用户名与密码，需要根据自己的设置
    static final String USER = "root";
    static final String PASS = "123456";

    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        try{
            // 注册 JDBC 驱动
            Class.forName(JDBC_DRIVER);

            // 打开链接
            System.out.println("连接数据库...");
            conn = DriverManager.getConnection(DB_URL,USER,PASS);

            // 执行查询
            System.out.println(" 实例化 Statement 对象...");
            stmt = conn.createStatement();
            String sql;
            sql = "SELECT id, name, url FROM websites";
            ResultSet rs = stmt.executeQuery(sql);

            // 展开结果集数据库
            while(rs.next()){
                // 通过字段检索
                int id  = rs.getInt("id");
                String name = rs.getString("name");
                String url = rs.getString("url");
```

```
                // 输出数据
                System.out.print("ID: " + id);
                System.out.print(", 站点名称: " + name);
                System.out.print(", 站点 URL: " + url);
                System.out.print("\n");
            }
            // 完成后关闭
            rs.close();
            stmt.close();
            conn.close();
        }catch(SQLException se){
            // 处理 JDBC 错误
            se.printStackTrace();
        }catch(Exception e){
            // 处理 Class.forName 错误
            e.printStackTrace();
        }finally{
            // 关闭资源
            try{
                if(stmt!=null) stmt.close();
            }catch(SQLException se2){
            }// 什么都不做
            try{
                if(conn!=null) conn.close();
            }catch(SQLException se){
                se.printStackTrace();
            }
        }
        System.out.println("Goodbye!");
    }
}
```

步骤 3 在安装 Java 的本机，打开 cmd 对 Java 程序编译后执行。

在 cmd 中先对 Java 程序进行编译

```
javac -encoding utf-8 -cp d:\Download\mysql-connector-java-8.0.16.jar  GaussDBMySQLDemo.java
```

再执行以下命令

```
java -cp d:\Download\mysql-connector-java-8.0.16.jar; GaussDBMySQLDemo
```

步骤 4 执行结果

执行结果如下：

```
连接数据库...
 实例化 Statement 对象...
ID: 1, 站点名称: openGauss, 站点 URL: https://opengauss.org/zh/
ID: 2, 站点名称: 华为云, 站点 URL: https://www.huaweicloud.com/
```

```
ID: 3, 站点名称: openEuler, 站点 URL: https://openeuler.org/zh/
ID: 4, 站点名称: 华为 support 中心, 站点 URL: https://support.huaweicloud.com/
Goodbye!
```

4 云 DBA 智能运维实验

4.1 云 DBA 概览

4.1.1 实验介绍

4.1.1.1 关于本实验

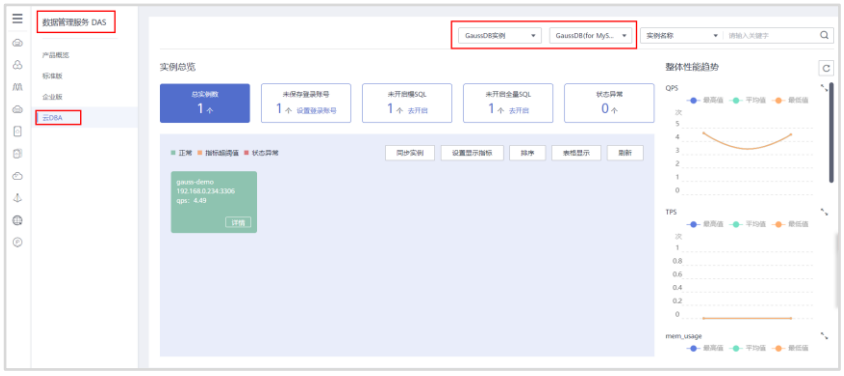
本实验通过在华为云端，使用 DAS 的云 DBA 功能，对 GaussDB(for MySQL)实例进行运维和管理操作。

4.1.1.2 实验目的

- 掌握云 DBA 功能的使用；
- 掌握查看实例的相关信息和设置指标等展示。

4.1.2 实验任务

步骤 1 登录华为云，在“数据管理服务 DAS”中，选择“云 DBA”功能，在右上角的实例类型选择“GaussDB 实例”和“GaussDB(for MySQL)”类型。



步骤 2 点击“设置登录账号”，将“用户名”和“密码”录入，点击“测试连接”，测试连接成功后，连接状态变为“连接成功”，点击“确定”，完成登录账号（注：由于云 DBA 功能中，需要进行一些实例的运维工作，因此建议使用管理员权限的用户登录）。

步骤 3 在“未开启慢 SQL”处，点击“去开启”，选择需要开启的实例，点击下方的“一键开启”，完成慢 SQL 的开启

“慢 SQL”可以在每个实例中，单独开启，并建议开启慢 SQL；开启慢 SQL 后，可以直接通过 DAS 的云 DBA 功能查看，若未开启该功能，云 DBA 中的“慢 SQL”功能将无数据展示。

步骤 4 在“未开启全量 SQL”处，点击“去开启”，选择需要开启的实例，点击下方的“一键开启”，完成全量 SQL 的开启

全量 SQL 可以在每个实例中，单独开启。用户可以根据需求，选择开启全量 SQL，开启全量 SQL 后，实例的性能会有 5%以内损耗；开启全量 SQL 后，可以直接通过 DAS 的云 DBA 功能查看，若未开启该功能，云 DBA 中的“全量 SQL”功能将无数据展示。

步骤 5　点击“设置显示指标”，对实例列表右侧的指标展示进行设置。

默认是展示前六个指标，从图中可看到是灰色的，是无法取消的，其余的指标如果有需求，需要进行勾选后，点击“确定”。

步骤 6　点击“排序”，对实例列表中，多个实例间采用何种方式进行排序进行设定。如图，默认是使用 QPS 的降序进行排序的，用户可以根据需求进行自定义设置。

步骤 7　点击“表格显示”/“图表显示”，可以将实例信息以表格/图表的样式，在实例列表处进行显示。

以图表显示：

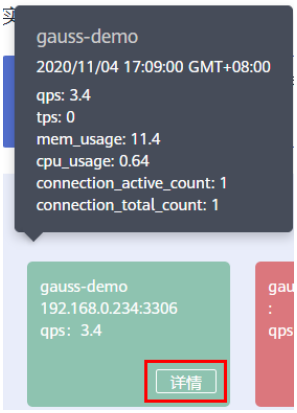
以表格显示：

步骤 8　点击“同步实例”，是手工触发同步实例的信息；点击“刷新”，是手工触发刷新实例信息。

同步实例：将刚购买的 GaussDB(for MySQL)的实例信息，同步给 DAS 平台。

步骤 9　实例列表中的颜色信息，可以清晰的展示出实例的状态，当实例处于正常时，显示为绿色；当实例的指标超出阈值时，显示为黄色；当实例处于异常状态时，显示为红色。

步骤 10 鼠标放置在具体某个实例上时，会显示出在“设置显示指标”中设置的指标信息，点击每个实例的“详情”，则进入每个具体实例中。



4.1.3 实验总结

本小节为云 DBA 功能概览，在初次登录云 DBA 模块时，需要对实例信息进行一些设置，希望能够通过本小节实验，熟悉云 DBA 中的实例的维护工作。

4.2 性能

4.2.1 实验介绍

4.2.1.1 关于本实验

本实验通过在华为云端，使用 DAS 的云 DBA 功能，对 GaussDB(for MySQL)实例性能进行监控与分析。

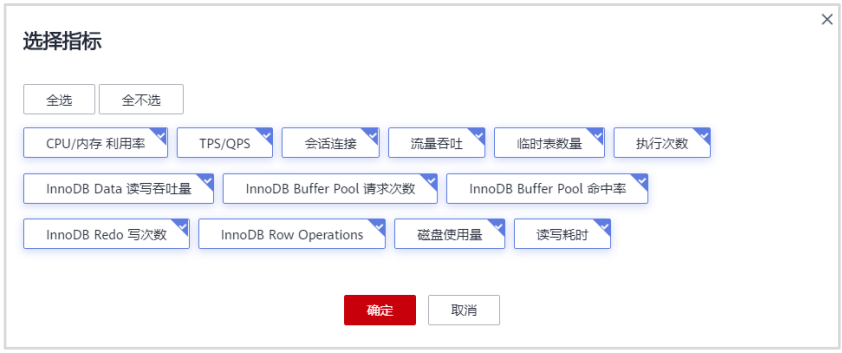
4.2.1.2 实验目的

- 熟悉云 DBA 功能的使用。
- 掌握通过云 DBA 性能模块，对 GaussDB(for MySQL)实例进行性能监控与分析。

4.2.2 实验任务

步骤 1 登录云 DBA，进入某具体实例的“详情”页。在“性能”中，选择“历史性能”，则能展示实例的历史性能指标。

步骤 2 通过“选择指标”，可以设置需要展示或者查看的历史性能指标。建议保留默认设置，可以更全方位的查看实例性能。



步骤 3 通过对右侧的时间设定，来展示历史性能指标，可快速使用平台提供的时间区间，也可使用右侧自定义的时间区间来进行查询。

步骤 4 当问题发生后，可以根据时间段来进行自定义事件范围，通过多维度的指标，来判断故障的原因。如下图，进行排查时间区间可以定义为 12 小时，在早晨 9 点 46 分左右时，CPU 使用率偏高，然后再将时间定义为 9 点 46 分前后，针对具体的性能指标进行排查。

步骤 5 在“性能”中选择“实时性能”，可以看到实例当前的实时性能信息，包括实例运行的时间、当前会话数和资源使用率等。

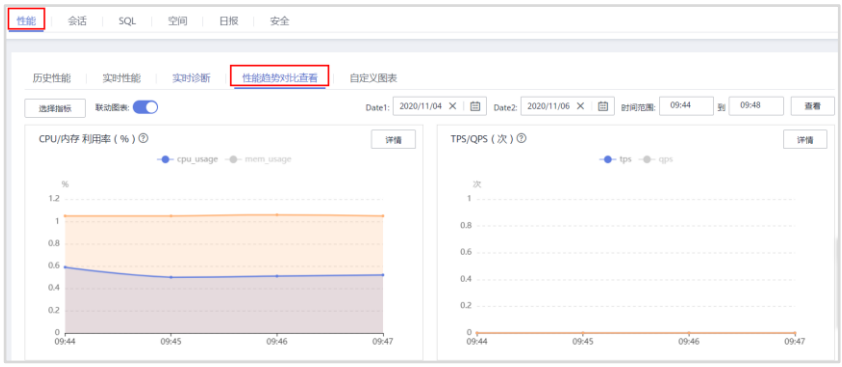
步骤 6 通过“时间间隔”的设置，设置性能指标刷新频率；通过“设置指标”，设置展示具体的性能指标。

步骤 7 下方展示了“设置指标”中设置的性能指标，在对实例性能进行观测时，可以通过这些性能指标对实例当前状态进行检测。

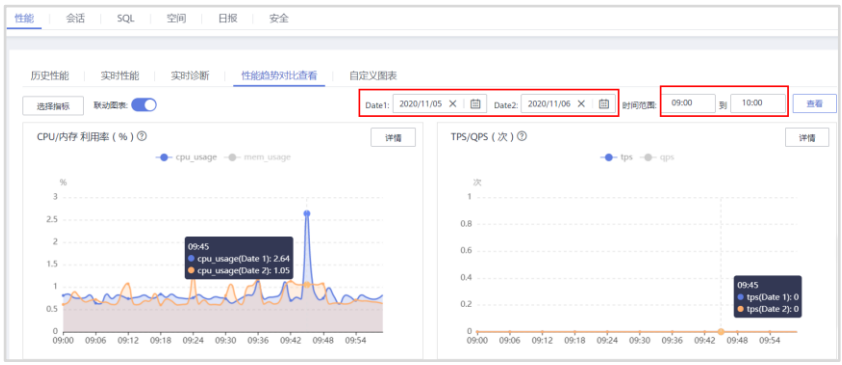
步骤 8 在“性能”中选择“实时诊断”，进入实例实时诊断界面。在该界面，能直观看到当前实例的资源使用率、实时性能（TPS/QPS 以及连接数）。

步骤 9 下方的慢 SQL，为初步诊断实例中是否存在慢 SQL，或者诊断是否由于慢 SQL 造成实例当前资源利用率较高。慢 SQL 趋势图中，可以根据时间自定义展示。

步骤 10 在“性能”中选择“性能趋势对比查看”，可以进行两个时间点的性能趋势对比。

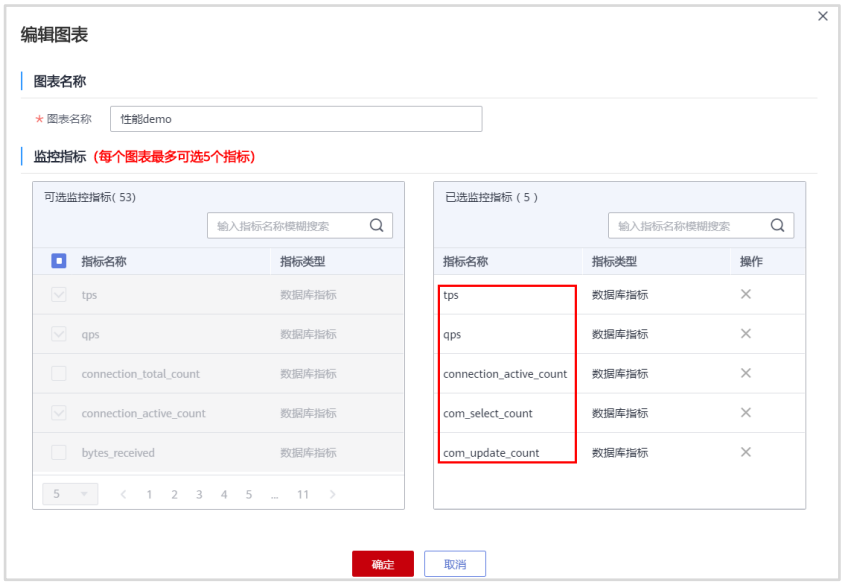


步骤 11 如图，对比 11 月 5 日与 11 月 6 日的 9 点与 10 点间的趋势对比，在 CPU 趋势中，蓝色曲线为时间点 1 的趋势，橙色曲线为时间点 2 的趋势。通过对比两个时间点的性能趋势，也是故障排查时的一种方式。

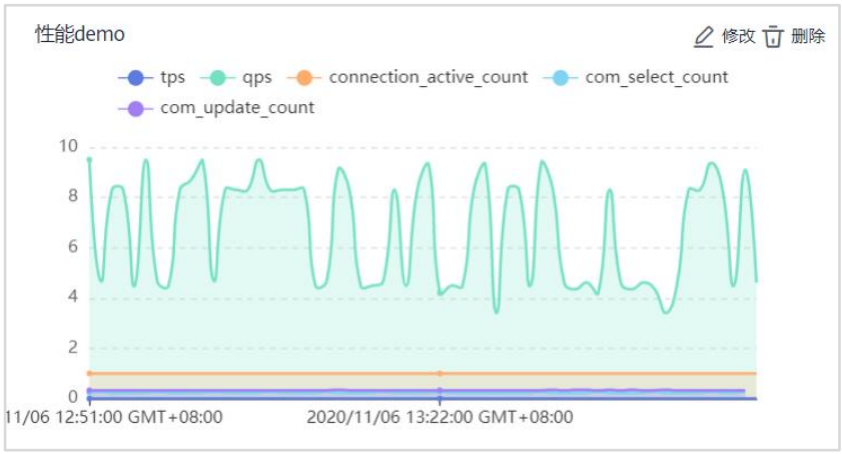


步骤 12 在“性能”中选择“自定义图表”，可以根据实际需求，将需要观测的指标放置与一张图表中展示。

步骤 13 点击“创建图表”，为自定义的图表命名为“性能 demo”，并将需要检测的指标：tps、qps、connection_active_count、com_select_count 和 com_update_count 选中（每个图表最多只能选择 5 个指标），点击“确定”。



步骤 14 完成后，在下方将展示出自定义图表的性能曲线，并可以通过对时间的选择，自行展示的时间区间。在每个自定义图表右上角，都有修改和删除按钮，可以对自定义图表进行修改和删除操作。



4.2.3 实验总结

通过上述的步骤，能够掌握对于云 DBA 中，性能模块的使用功能。性能问题是 DBA 日常运维中，经常处理的问题之一。云 DBA 的性能模块，能够方便、快速地帮助 DBA 分析、定位实例的性能问题。

4.3 会话

4.3.1 实验介绍

4.3.1.1 关于本实验

本实验通过在华为云端，使用云 DBA 的会话模块，对实例的会话进行管理。

4.3.1.2 实验目的

- 熟悉云 DBA 功能的使用。
- 掌握通过云 DBA 会话模块，对 GaussDB(for MySQL)实例进行会话管理。

4.3.2 实验任务

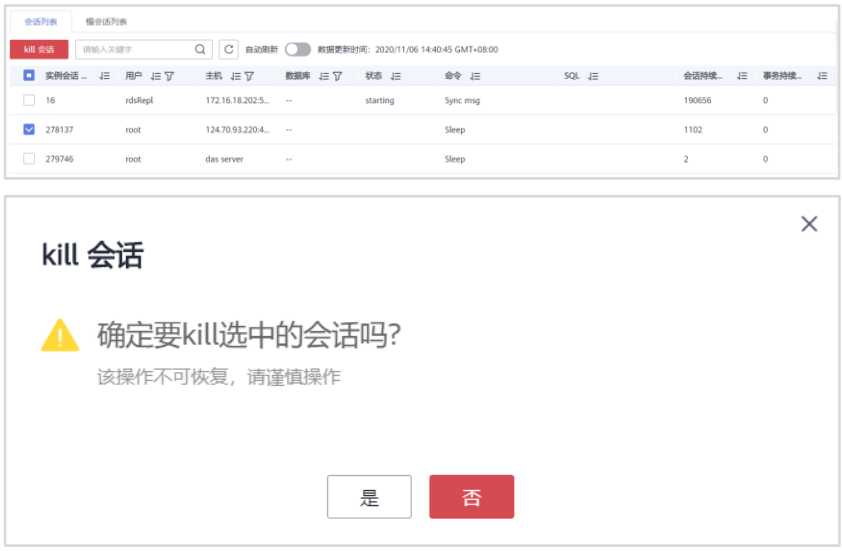
步骤 1　登录云 DBA，进入某具体实例的“详情”页。在“会话”中选择“实时会话”。

步骤 2　在“实时会话”中，展示了当前实例中的会话概要，并且通过用户、来源和数据库三个维度，将当前会话信息进行分组展示。

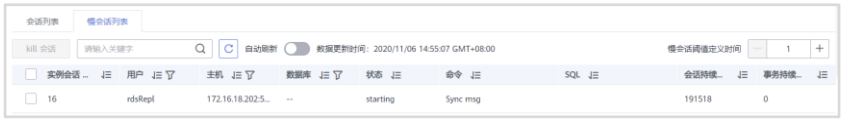
步骤 3　每个分组中，总数部分的数值都是可以点击的，点击数值，会展示会话列表的详细信息。如点击“按用户统计”中的 root 用户数值，弹出下方 root 用户的会话列表详细信息。

步骤 4　在“实时会话”的“会话列表”中，则为实例中的全部会话及信息。可以在检索栏中检索需要的会话信息。也可将“自动刷新”的开关打开，打开后，会话列表中的信息每 30 秒刷新一次。

步骤 5　选中需要进行 kill 的会话，点击“kill 会话”，弹出确认框，点击“是”，则会下发剔除会话功能。系统会话，无法通过“kill 会话”剔除，如下图中的 16 号线程。

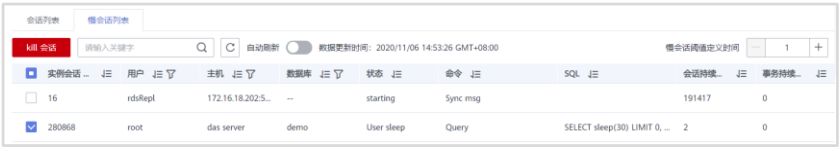


步骤 6　“慢会话列表”中，则可以定义慢会话阈值的时间，如下图中，定义 1 秒以上的会话为慢会话，相关会话信息则会在慢会话列表中展示。



步骤 7　例如，在 DAS 的 SQL 查询中，执行 select sleep(30);的操作。

步骤 8 在“慢会话列表”中，则可以看到对应的慢会话信息，可以选择该慢会话，点击“kill 会话”，将其剔除。



步骤 9 当实例会话数达到连接上限时，可以通过“紧急 kill 会话”通道，对实例中的会话进行剔除，在“会话”中选择“紧急 kill 会话”，进入该通道。

步骤 10 该列表中，为当前实例中的会话数，选中需要剔除的会话，点击“kill 会话”，则下发剔除会话，Kill 操作将会以日志的形式被记录下来，通过“历史急救日志”，可以看到具体的操作。该功能平时建议不要使用，如果需要对会话进行 kill 操作，可以在“实时会话”中进行操作。

4.3.3 实验总结

本小节，通过会话模块，对实例中的会话进行管理，当遇到锁表、大并发或者慢 SQL 造成数据库实例异常时，可以使用该模块快速定位到问题会话，并执行 kill 操作；当实例的会话连接数达到上限时，会话模块提供了紧急通道，可以通过该通道迅速释放会话，使得实例恢复正常。

4.4 SQL

4.4.1 实验介绍

4.4.1.1 关于本实验

本实验通过在华为云端，使用 DAS 的云 DBA 功能，对实例中的 SQL 进行管理，包括实例中的慢 SQL、实例中执行 SQL 的统计以及 SQL 的诊断。

4.4.1.2 实验目的

- 熟悉云 DBA 中的 SQL 功能。
- 掌握通过云 DBA 中 SQL 模块，对 GaussDB(for MySQL)实例进行 SQL 管理。

4.4.2 实验任务

步骤 1 在“SQL”中选择“慢 SQL”，进入慢 SQL 界面，默认实例是未开启慢 SQL 的，如果在实验 1.1 中未开启，则可以在具体的实例中，开启（关闭）慢 SQL。

步骤 2 开启了慢 SQL 后，DAS 会收集实例中的慢 SQL 日志，并将其展示出来。时间区间也可以通过右侧的时间区间进行定义。

步骤 3 例如在 DAS 中执行 select sleep(30); 后，会在慢 SQL 趋势以及慢日志统计和慢日志明细中看到具体的信息。通过诊断，可以智能给出该 SQL 的诊断建议。（如下图）

步骤 4 在“SQL”中选择“全量 SQL 洞察”，在 SQL 列表中，可以查询到实例中执行的 SQL 的全部信息。默认是关闭收集全量 SQL 的，如果在实验 1.1 中未开启，可以单独开启实例的该功能，该功能开启对实例有性能损耗，损耗低于 5%，管理员需要根据实际情况是否开启，此处将该功能开启。

步骤 5 开启了全量 SQL 后，会将实例中的 SQL 内容全部记录，超过 10000 条数据，只能使用导出进行查看，全量 SQL 可以通过以下的各种维度进行检索。

用户维度：可以针对用户进行检索展示。

数据库维度：可以针对不同的数据库进行检索展示。

关键字维度：可以针对 SQL 中的关键字进行检索展示。

操作类型维度：可以从 select、insert、update、delete、show、create、drop、alter、replace、use、start、commit 和 rollback 的操作类型中，进检索展示。

执行状态维度：可以从 SQL 执行成功或者失败的维度，对 SQL 进行检索展示。

线程 ID 维度：可以针对某个或者几个线程进行检索展示。

执行耗时维度：可以设定执行耗时区间，对 SQL 进行检索展示。

扫描行数、更新行数和返回行数维度：可以设定对应的行数区间，对 SQL 进行检索展示。

步骤 6 在“全量 SQL 洞察”中选择“SQL 模板”，可以获取时间区间内，执行耗时和执行次数的趋势。时间区间也可以自定义。

步骤 7　下方的 SQL 模板，会展示出时间区间范围内的 SQL，包括 SQL 执行的指标，可以按照耗时、扫描行数、执行次数等维度进行排序。管理员也可以通过该功能，获取在某个时间区间内 SQL 的样本，能够快速定位问题 SQL。

步骤 8　在“全量 SQL 洞察”中，选择“SQL 操作类型”。在时间区间内，将实例中执行的 SQL，按照执行类型的维度进行展示和统计。从下图中，可以看到，该实例主要执行的 SQL 类型为 select 语句。

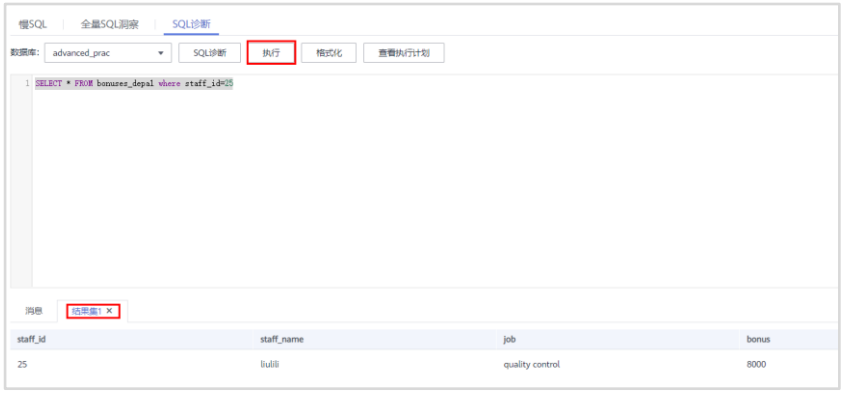
步骤 9　下方则通过图表的方式，将时间区间内的 SQL 执行类型进行统计，并将一些基础信息如总次数、平均耗时等进行展示。

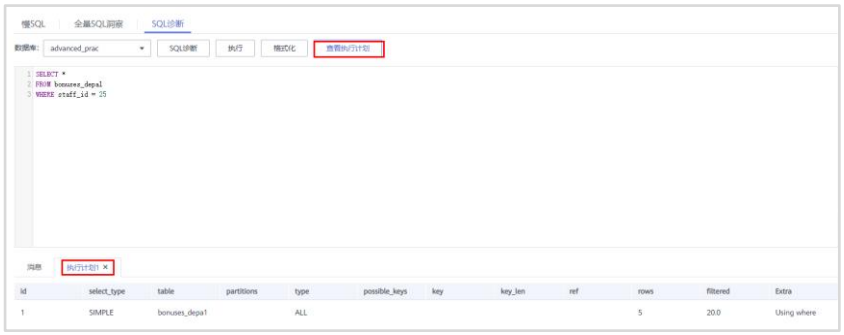
步骤 10　在“SQL”中选择“SQL 诊断”，可以对需要优化的 SQL 进行诊断。

步骤 11 选择数据库 advanced_prac（高阶语法小节创建的数据库），并将输入以下 SQL：

```
SELECT * FROM bonuses_depa1 where staff_id=25;
```

点击“SQL 诊断”，可以针对这条 SQL 语句进行智能诊断；点击“执行”，会进入到数据库中执行这条 SQL 语句；点击“格式化”，会优化 SQL 语句的格式；点击“查看执行计划”，可以查看这条 SQL 的执行计划（如下图）。





4.4.3 实验总结

SQL 作为数据库中重要一部分，管理员对于 SQL 的优化与维护，是日常工作之一。通过使用云 DBA 智能运维的 SQL 模块，能够较为清晰的掌握 SQL 的执行情况，能够方便看到 SQL 执行的信息以及对应的执行计划。

4.5 空间

4.5.1 实验介绍

4.5.1.1 关于本实验

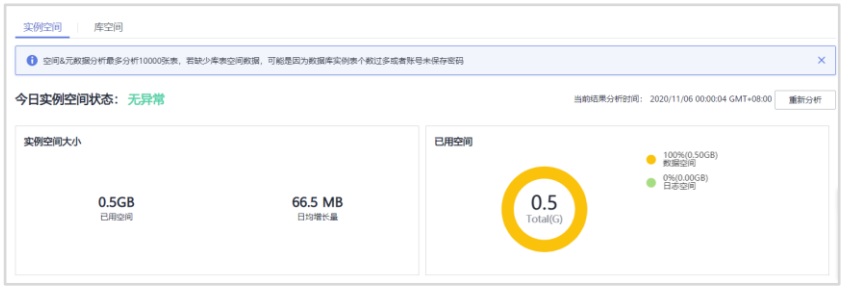
本实验通过在华为云端，使用云 DBA 对实例进行空间管理，包括实例空间和库空间。

4.5.1.2 实验目的

- 熟悉云 DBA 中的空间功能。
- 掌握通过云 DBA 中空间模块，对 GaussDB(for MySQL)实例进行空间管理。

4.5.2 实验任务

步骤 1　在“空间”中，选择“实例空间”，可以进入实例的空间查看。在该界面，可以看到当前实例使用的空间大小、日均增长量和已使用的空间中，各个空间的占比。



步骤 2　设定时间区间，查看在区间内的实例空间变化趋势。从实例使用空间、数据空间和日志空间三个维度进行展示。

步骤 3　在“空间”中，选择“库空间”，可以看到每个库空间的使用大小，其中包括库空间的数据空间、索引空间等信息以及空间的碎片率。点击“库表空间&元数据分析”，可以看到每个数据库的具体信息，从这里我们可以看到，有哪些表并未创建主键、有哪些表的自增 ID 即将耗尽等管理员需要关注的信息，也可以获取到数据库中，具体的表锁使用的空间大小等信息。同时，也能获取到表的表定义和表数据的增长趋势，这样对于是否需要对表进行分表，以及表的碎片化程度等相关信息进行判断。

4.5.3 实验总结

通过对实例和数据库的空间查看，管理员能够判断出是否需要对实例进行扩容操作；通过对表的数据量增长，管理员可以判断是否需要对数据量大的表，进行分库分表或者改造为分区表的优化操作；通过对表碎片化程度的观测，管理员可以判断是否需要对表进行降低碎片化操作。

4.6 日报

4.6.1 实验介绍

4.6.1.1 关于本实验

本实验通过在华为云端，使用云 DBA 对当前实例进行日报分析，通过生成的日报，对实例的健康状态进行评估。

4.6.1.2 实验目的

- 理解日报功能的应用场景；
- 掌握通过云 DBA 中日报功能，对 GaussDB(for MySQL)实例进行健康分析。

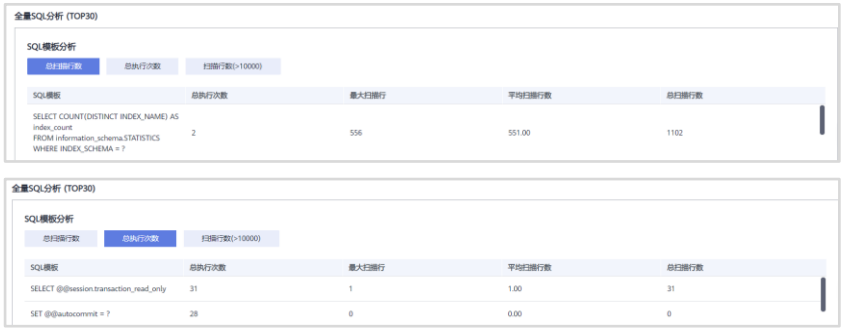
4.6.2 实验任务

步骤 1 选择“日报”，进入日报功能。系统每天会在早晨自动生成日报，用户也可以手动发起诊断，诊断的结果和晨报都可以在历史诊断汇总查看，同时也可以对诊断报告进行下载和订阅。订阅时，需要输入订阅的邮箱。

步骤 2 当诊断报告正常时，会显示为绿色；当诊断报告异常时，会显示为红色，并且在风险点出注明诊断报告中的风险点。

步骤 3　在诊断报告的 SQL 部分，有慢 SQL 分析，会将报告中的慢 SQL，通过执行次数、平均执行时间、最大执行时间和扫描返回比四个维度进行分组排列，可以按照不同的维度进行查看。

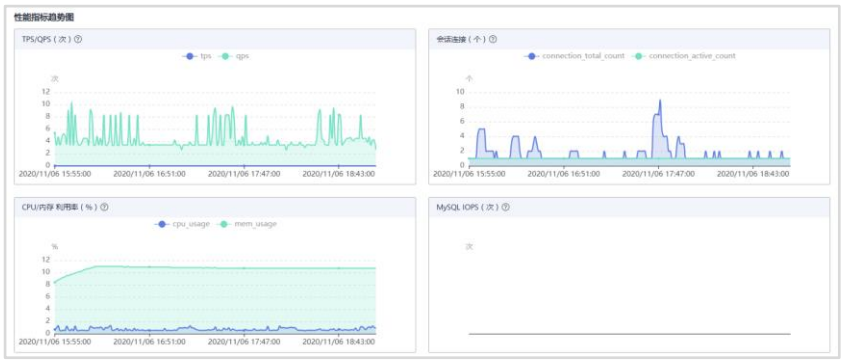
步骤 4　全量 SQL 分析通过总扫描行数、总执行次数和扫描行数大于一万行这三个维度进行分组排序。不同的维度运用于不同的判断点，有的问题 SQL 可能执行的次数不多，但是每次扫描的行数很多；有的问题 SQL，可能单次扫描的行数不算很多，但是执行的次数很多，这样也会造成系统性能问题。



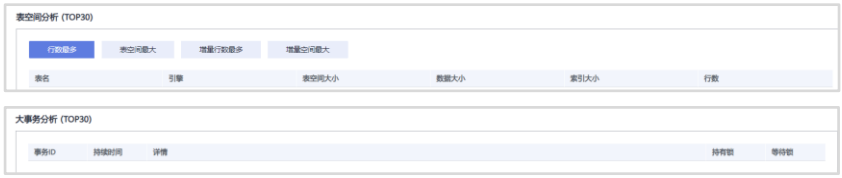
步骤 5　针对 DML 的类型，将 DML 的 SQL 进行分类，可以清晰看到不同类型的 DML 在诊断区间内的执行次数，以及 DML 操作的对象等信息。

步骤 6　诊断报告中，对于实例的资源性能整体做了一个评估。如示例：可以看到实例中的总体性能指标都处于低水位的状态，因此实例整体负载不高。

步骤 7　针对主要性能指标，通过性能指标趋势图进行展示，通过该趋势图，可以较快定位问题时间点。



步骤 8　表空间分析和大事物分析，作为诊断报告的最后两部分，可以方便管理员对于表空间的趋势、数据库中表记录数等进行快速分析；大事物能够定位到诊断区间内，数据库是否存在大事物，并且大事物发生的信息等。



4.6.3 实验总结

日报功能，对于管理员，即可以当做每日的晨检报告，又可以作为问题回溯或者问题发生时的诊断报告。通过诊断报告，能够从性能、SQL、空间等多个维度，综合地判断实例的健康情况，也能迅速地定位问题，能够提高管理员定位问题、处理问题的效率。

4.7 安全

4.7.1 实验介绍

4.7.1.1 关于本实验

本实验通过在华为云端，使用云 DBA 的安全模块，检测是否有异常的客户端或用户对数据库实例进行攻击。

4.7.1.2 实验目的

- 理解安全模块的应用场景。

4.7.2 实验任务

步骤 1 点击“安全”，进入该模块，攻击检测开关默认是关闭的，如果需要对客户端或用户进行安全检测，需要将其开启；使用 SQL 攻击检测服务，会随着全量 SQL 监测开启而开启。

步骤 2 开启“攻击检测开关”后，可以从时间、操作类型、客户端 IP/用户这几个维度进行检索。

4.7.3 实验总结

作为数据库，数据安全是重中之重。华为云在安全层面，做了很多措施与限制，但是为避免有异常客户端或用户对数据库进行非法操作，该界面可以提供攻击检测，管理员可以通过该界面来获取对数据库实例进行攻击行为的客户端 IP 或者用户。

4.8 DAS 企业版

4.8.1 实验介绍

4.8.1.1 关于本实验

本实验通过在华为云端，使用 DAS 企业版功能，了解和掌握 DAS 企业版相关功能。

4.8.1.2 实验目的

- 掌握 DAS 企业版应用场景。
- 理解通过 DAS 企业版，提高运维效率。

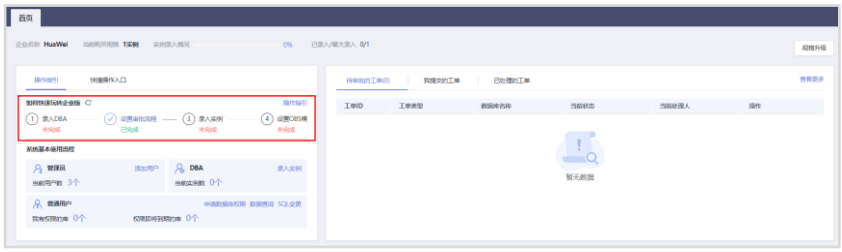
4.8.2 实验任务

步骤 1 DAS 企业版环境准备。在“数据管理服务 DAS”中选择“企业版”，并通过右上角的“开通企业”，配置 DAS 企业版服务。

步骤 2 根据实例的个数以及适合企业的计费方式，选择对应的规格，实验中我们选择 1 实例和按需计费，点击“立刻购买”。

步骤 3 点击“进入企业”，首次登录，需要设置企业名称。输入企业名称，确认后，点击“进入企业。”

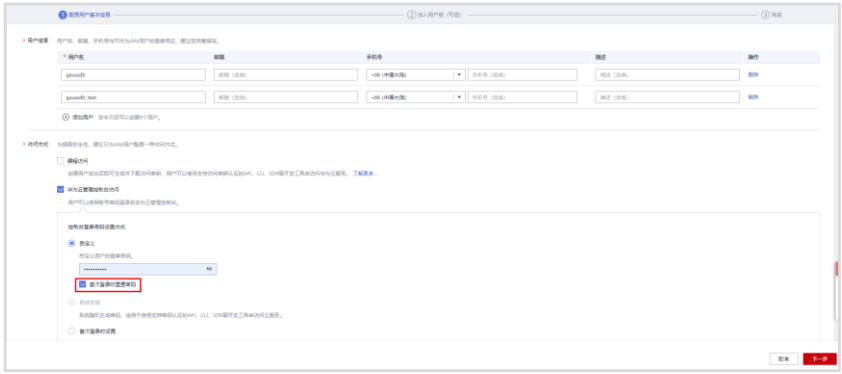
步骤 4 通过快速指引，完成 DAS 企业版设置。



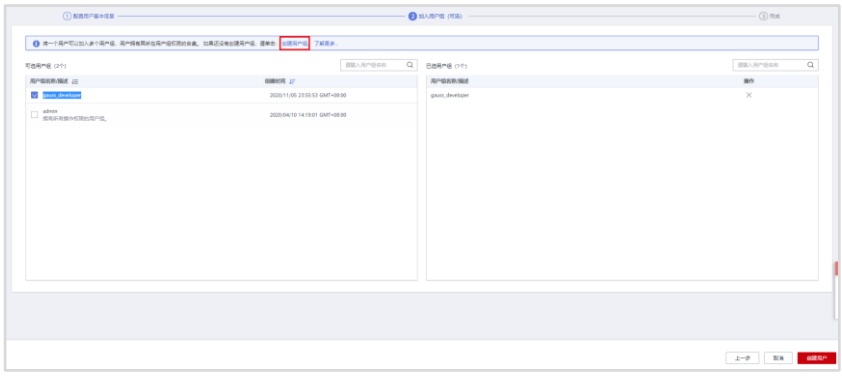
步骤 5 通过“录入 DBA”，对登录 DAS 企业版的用户进行设置。可以进行录入用户和 IAM 账号、EPS 账号的同步，以及对用户进行启用、停用和移出操作。

步骤 6 此处示例中，在原有华为云账号下创建了两个 IAM 子账号：gaussdb 和 gaussdb_test。如果已创建完成 IAM 子账号，可以跳过步骤 6~步骤 9。点击用户列表下“统一身份认证”。

步骤 7 点击右上角创建用户，进入“创建用户”界面，设置用户的名称为 gaussdb 和 gaussdb_test，设置访问方式，如果使用华为云，就选择“华为云管理控制台访问”，并设定密码，建议将“首次登录时重置密码”勾选上。完成后，点击“下一步”。

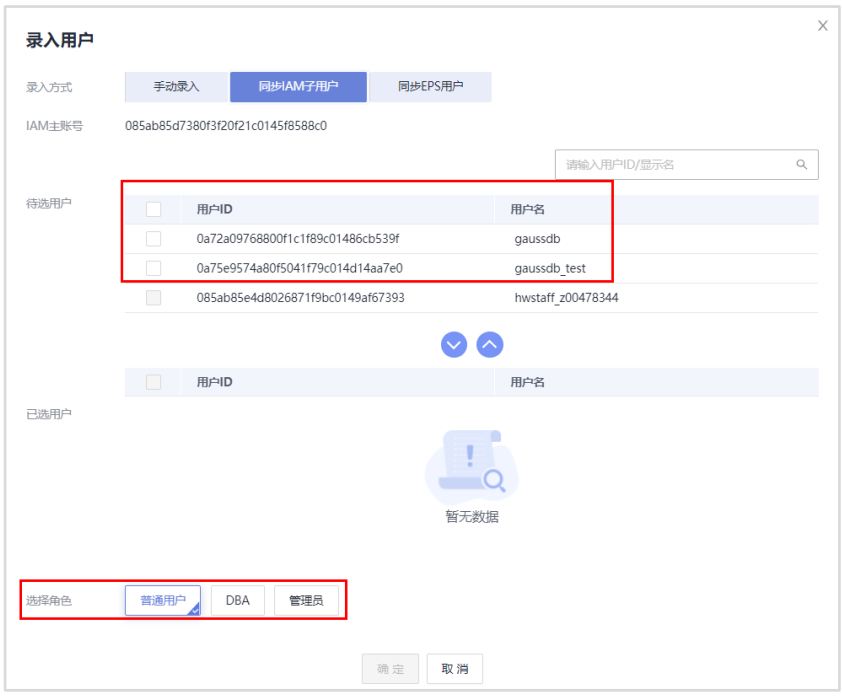


步骤 8 勾选对应的属组，将用户添加到对应的属组中，如图我们将 gaussdb 和 gaussdb_test 添加到 gauss_developer（该用户组是自定义的）中，若需要自定义其他用户组，可以点击红框中的“创建用户组”自行定义。选择完成后，点击“创建用户”。



步骤 9　当出现“2 个用户全部创建成功”时，表明用户创建完成，反馈 DAS 企业版的录入用户界面。

步骤 10　在录入用户时，需要对录入的用户进行角色选择，可以使用手动录入，也可以同华为云中的账号进行同步后录入。用户角色分为普通用户、管理员和 DBA 三种角色。将 gaussdb 设置为 DBA 角色，将 gaussdb_test 设置为普通用户。

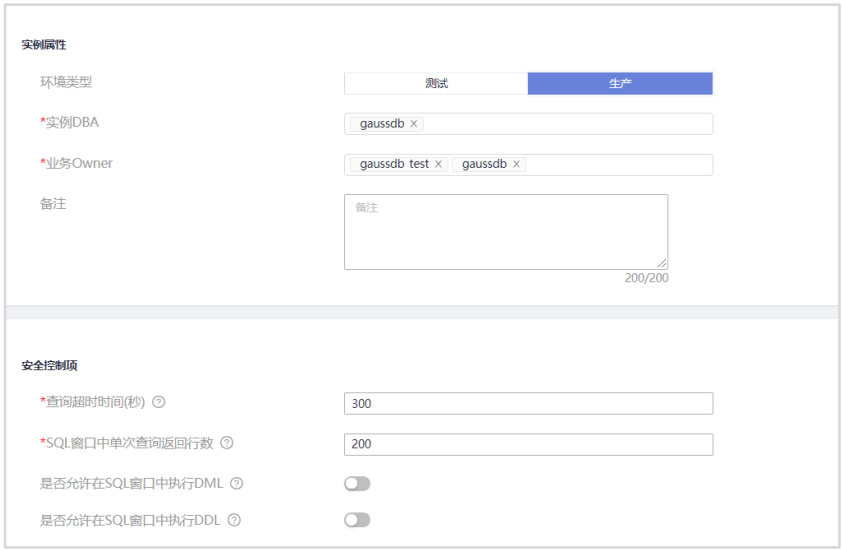


步骤 11　完成后如下：

步骤 12　返回首页操作指引，选择“设置审批流程”，系统自带两步审批，管理员可以根据企业自身的审批流程，自行添加，系统默认的两个审批流程无法删除。

步骤 13　返回首页操作指引，选择“录入实例”，选择 GaussDB(for MySQL)实例进行录入。

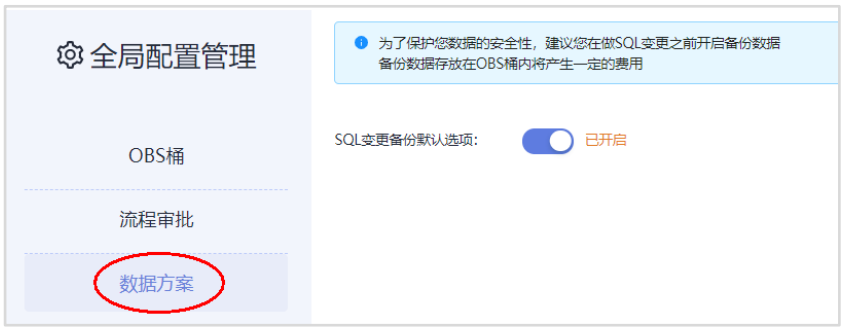
步骤 14　同时，设置该实例的 DBA 账号和业务 owner 账号，DBA 账号只能是角色为 DBA 的用户，业务 owner 可以在角色为普通用户、DBA 和管理员之间进行选择。



步骤 15　返回首页操作指引，选择“设置 OBS 桶”，选择对应区域的 OBS 桶，OBS 桶中主要存放 SQL 变更的备份数据。

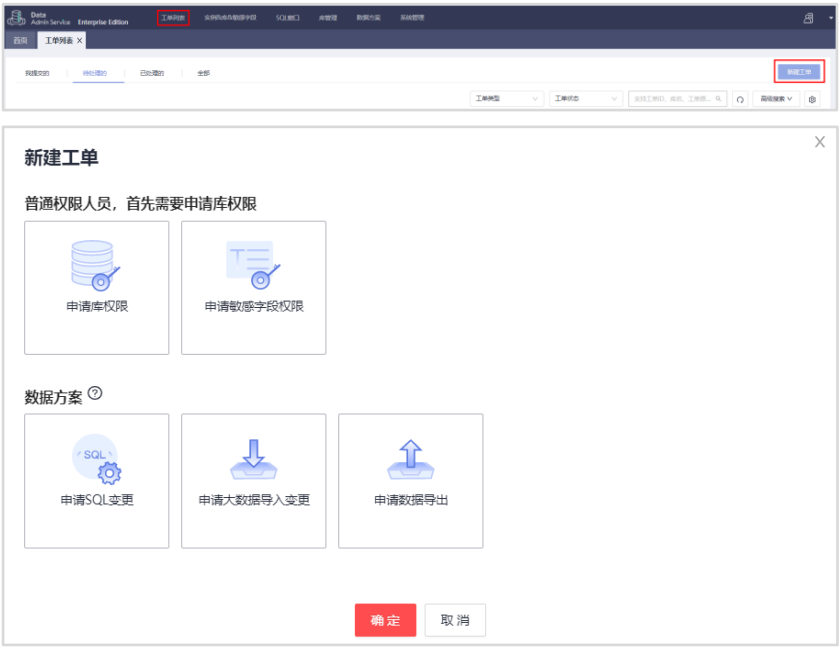
步骤 16　在“流程审批”中，可以针对审批环节的相关内容进行设置，包括实例拥有的 DBA 和 Owner 数以及审批流程的设置等。

步骤 17　在“数据方案”中，选择是否开启 SQL 变更前的备份，建议对重要的系统，开启该功能。



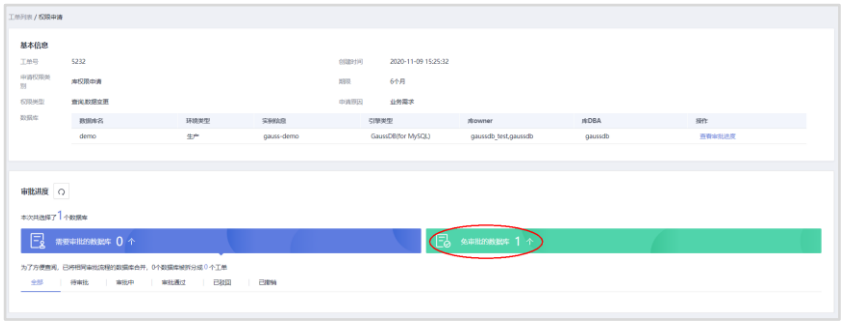
步骤 18　作为管理员，完成“操作指引”中的配置，可以进行 DAS 企业版的使用。

步骤 19 在“工单列表”中，普通用户需要先申请数据库的相关权限，此处为了方便演示，使用管理员用户，生成中需要切换成普通用户登录 DAS 企业版。在工单列表中，点击新建工单，选择“申请权限”。



步骤 20 在数据库中选择需要操作的数据库；对于普通用户，一般对数据库具有查询和数据变更的操作，因此申请查询和数据变更的权限；权限的时间根据实际业务要求进行选择。完成进行提交。

步骤 21 因为实验中，使用的是管理员用户进行申请，因此属于免审批申请，如果是普通用户申请，需要按照流程审批的规则进行审批。申请完成后，用户具有该数据库的相关操作，可通过工单申请对应权限类的变更操作。

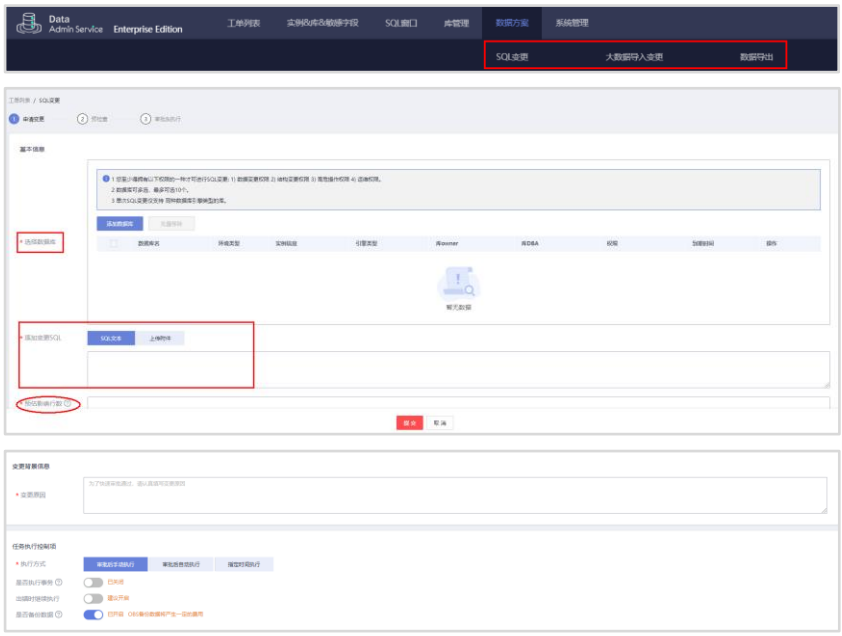


步骤 22 对于具有操作权限的数据库，用户可以通过“实例&库&敏感字段”中的库列表，对所具有权限的数据库进行查看。（实例列表同库列表）

步骤 23 对于敏感列，用户可以通过“实例&库&敏感字段”中的“敏感字段列表”，进行申请和删除操作。

步骤 24 “SQL 窗口”与“库管理”的使用方式，同 DAS 标准版，此处不再进行实验。

步骤 25 “数据方案”中的“SQL 变更”、“大数据导入变更”和“数据导出”，均同工单申请中的对应操作相同，不一一进行实验。如 SQL 变更时，需要将执行的数据库选中，将 SQL 文本输入文本框或进行脚本的上传，并填写“预估影响行数”、“变更原因”和“执行方式”（审批后手动、审批后自动和指定时间执行）信息。



步骤 26 在首页中的“操作指引”里设置的对应信息，都可通过“系统管理”中的“实例管理”、“用户管理”和“全局配置管理”进行再次设置和修改，使用方式同“操作指引”。

步骤 27 通过“系统管理”中的访问设置，可以设置登录 DAS 企业版的白名单，白名单可以是单独的 IP 也可以是网段，多个 IP 之间需要用英文逗号进行分隔，完成后，点击“点击开启”，立即生效。

步骤 28 通过“系统管理”中的“操作审计”，可以将 DAS 企业版中的操作动作进行审计，管理员可以针对该模块，对所有的操作进行审计查询。

4.8.3 实验总结

DAS 企业版，是 DAS 服务针对企业进行设计的，能够帮助企业用户完成如：工单审批、变更审批、用户权限申请等相关操作，以及支持 DAS 标准版的相关操作。用户通过 DAS 企业版，能方便地进行相关操作，并且提高工作效率。本节实验，通过对 DAS 企业版的相关操作，使用户能够熟悉 DAS 企业版的内容，能够熟练使用 DAS 企业版，完成数据库相关运维管理工作。

5 数据库性能调优实验

5.1 实验介绍

5.1.1 关于本实验

本实验通过分析执行计划和 Hint 语法操作来体现数据库性能调优内容。

5.1.2 实验目的

- 掌握执行计划的生成原则；
- 理解数据库性能调优的思路。

5.2 实验任务配置

5.2.1 Explain 语法实操

步骤 1 创建调优实验测试数据库

通过 DAS 登录 GaussDB(for MySQL)数据库。

点击“新建数据库”，创建调优数据库。

在弹出窗口输入数据库名 Tuning_test_xxx（为方便区分，xxx 为实验操作者名字首字母缩写，如张三，数据库名称为 Tuning_test_zs）。

点击“确定”，数据库创建成功。

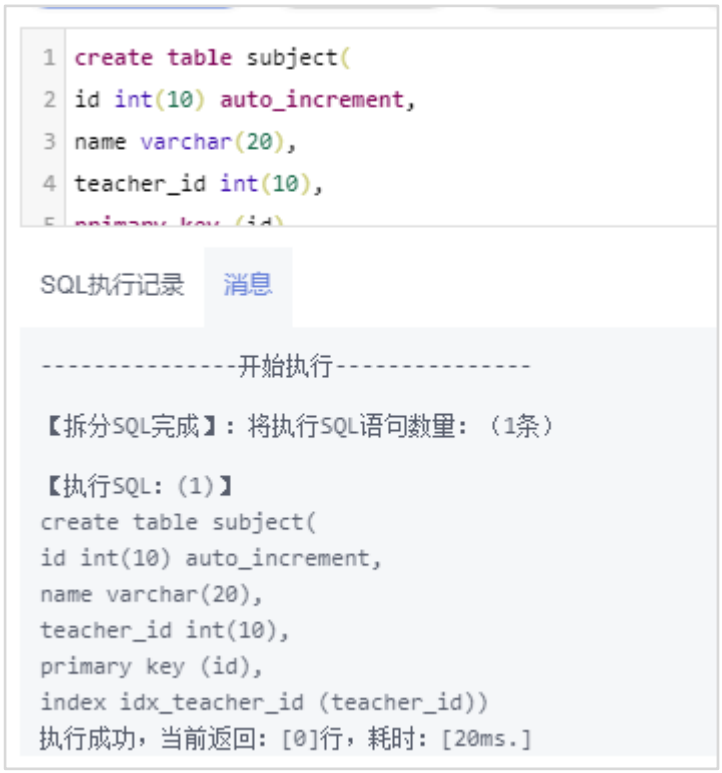
步骤 2　创建数据表并导入数据

点击上一步骤创建的数据库名称，进入调优测试数据库。

点击右上角的“SQL 窗口”，进入 SQL 编辑界面。

创建学科表 subject。

```
create table subject(
id int auto_increment,
name varchar(20),
teacher_id int,
primary key (id),
index idx_teacher_id (teacher_id)
);
```



创建教师表 teacher。

```
create table teacher(
id int auto_increment,
```

```
name varchar(20),
teacher_no varchar(20),
primary key (id),
unique index unx_teacher_no (teacher_no(20))
);
```

创建学生表 student。

```
create table student(
id int auto_increment,
name varchar(20),
student_no varchar(20),
primary key (id),
unique index unx_student_no (student_no(20))
);
```

```
1  create table student(
2 id int(10) auto_increment,
3 name varchar(20),
4 student_no varchar(20),
5 primary key (id)
```

SQL执行记录 消息

```
---------------开始执行---------------

【拆分SQL完成】：将执行SQL语句数量：（1条）

【执行SQL：(1)】
create table student(
id int(10) auto_increment,
name varchar(20),
student_no varchar(20),
primary key (id),
unique index unx_student_no (student_no(20)))
执行成功，当前返回：[0]行，耗时：[24ms.]
```

创建学生成绩表 student_score。

```
create table student_score(
id int auto_increment,
student_id int,
subject_id int,
score int,
primary key (id),
index idx_student_id (student_id),
index idx_subject_id (subject_id)
);
```

```
CREATE TABLE student_score (
    id int(10) AUTO_INCREMENT,
    student_id int(10),
    subject_id int(10),
```

SQL执行记录 消息

```
CREATE TABLE student_score (
        id int(10) AUTO_INCREMENT,
        student_id int(10),
        subject_id int(10),
        score int(10),
        PRIMARY KEY (id),
        INDEX idx_student_id(student_id),
        INDEX idx_subject_id(subject_id)
)
执行成功，当前返回：[0]行，耗时：[26ms.]
```

为教师表增加名字普通索引。

alter table teacher add index idx_name(name);

```
alter table teacher add index idx_name(name);
```

SQL执行记录 消息

```
---------------开始执行---------------

【拆分SQL完成】：将执行SQL语句数量：（1条）

【执行SQL：(1)】
alter table teacher add index idx_name(name)
执行成功，耗时：[15ms.]
```

导入数据。

insert into student(name,student_no) values ('zhangsan','20200001'),('lisi','20200002'),('yan','20200003'),('dede','20200004');

insert into teacher(name,teacher_no) values('wangsi','T2010001'),('sunsi','T2010002'),('jiangsi','T2010003'),('zhousi','T2010004');

insert into subject(name,teacher_id) values('math',1),('Chinese',2),('English',3),('history',4);

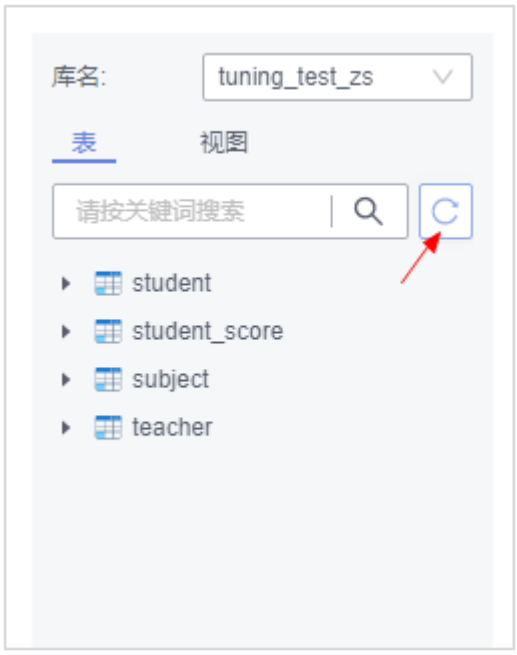
insert into student_score(student_id,subject_id,score) values(1,1,90),(1,2,60),(1,3,80),(1,4,100),(2,4,60),(2,3,50),(2,2,80),(2,1,90),(3,1,90),(3,4,100),(4,1,40),(4,2,80),(4,3,80),(4,5,100);

```
insert into subject(name,teacher_id) values('math',1),('chinese',2),('english',3),('history',4);
insert into student_score(student_id,subject_id,score) values(1,1,90),(1,2,60),(1,3,80),(1,4,100),(2,4,60),(2,3,50),(2,2,80),(2,1,90
```

SQL执行记录 消息

```
----------开始执行----------
【拆分SQL完成】：将执行SQL语句数量：（4条）
【执行SQL：(1)】
insert into student(name,student_no) values ('zhangsan','20200001'),('lisi','20200002'),('yan','20200003'),('dede','20200004')
执行成功，耗时：[14ms.]
【执行SQL：(2)】
insert into teacher(name,teacher_no) values('wangsi','T2010001'),('sunsi','T2010002'),('jiangsi','T2010003'),('zhousi','T2010004')
执行成功，耗时：[9ms.]
```

点击“刷新”，效果如下：



步骤 3 Id 字段

explain 执行效果：

```
EXPLAIN SELECT *
FROM subject
WHERE id = 1;
```

效果如下：

以下是explain select * from subject where id = 1的执行结果集 ⓘ元数据信息，不能编辑、翻页和导出SQL 复制行 复制列 列设置

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref
1	1	SIMPLE	subject		const	PRIMARY	PRIMARY	4	const

id 相同时，执行顺序从上到下

```
EXPLAIN SELECT subject.*
FROM subject, student_score, teacher
WHERE subject.id = student_id
```

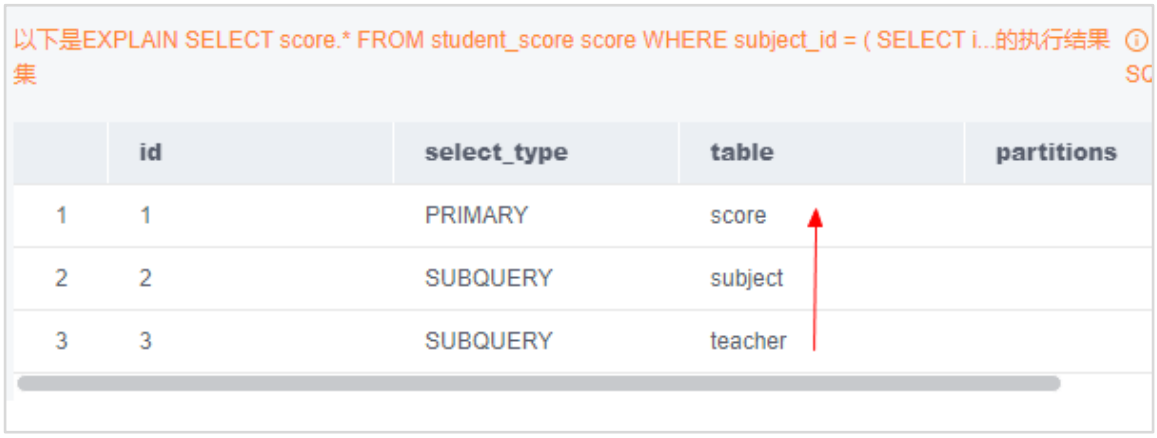
```
    AND subject.teacher_id = teacher.id;
```

读取顺序：subject -> teacher -> student_score。

Id 不同。

```
EXPLAIN SELECT score.*
FROM student_score score
WHERE subject_id = (
    SELECT id
    FROM subject
    WHERE teacher_id = (
        SELECT id
        FROM teacher
        WHERE id = 2
    )
);
```

如果是子查询，id 的序号会递增，id 的值越大优先级越高，越先被执行。



读取顺序：teacher -> subject -> student_score

id 相同与不同都存在时。

```
EXPLAIN SELECT subject.*
FROM subject
    LEFT JOIN teacher ON subject.teacher_id = teacher.id
UNION
SELECT subject.*
FROM subject
    RIGHT JOIN teacher ON subject.teacher_id = teacher.id;
```

id 如果相同，可以认为是一组，从上往下顺序执行。

在所有组中，id 值越大，优先级越高，越先执行。

	id	select_type	table
1	1	PRIMARY	subject
2	1	PRIMARY	teacher
3	2	UNION	teacher
4	2	UNION	subject

读取顺序：2.teacher -> 2.subject -> 1.subject -> 1.teacher。

步骤 4　Select_type 字段

表5-1 Select_type 字段说明

Select_type字段	意义
SIMPLE	简单SELECT，不使用UNION或子查询等
PRIMARY	子查询中最外层查询，查询中若包含子查询，最外层的SELECT被标记为PRIMARY
UNION	UNION中的第二个或后面的SELECT语句
DEPENDENT UNION	UNION中的第二个或后面的SELECT语句
UNION RESULT	UNION的结果
SUBQUERY	子查询中的第一个SELECT，结果不依赖于外部查询
DEPENDENT SUBQUERY	子查询中的第一个SELECT，依赖于外部查询
DERIVED	派生表的SELECT, FROM子句的子查询
UNCACHEABLE SUBQUERY	一个子查询的结果不能被缓存，必须重新评估外链接的第一行

SIMPLE 类型。

简单 SELECT，不使用 UNION 或子查询等。

```
EXPLAIN SELECT subject.*
FROM subject
LIMIT 2;
```

以下是EXPLAIN SELECT subject.* FROM subject limit 2的执行结果集

	id	select_type	table
1	1	SIMPLE	subject

PRIMARY 类型。

```
EXPLAIN SELECT score.*
FROM student_score score
WHERE subject_id = (
    SELECT id
    FROM subject
    WHERE teacher_id = (
            SELECT id
            FROM teacher
            WHERE id = 2
    )
);
```

查询中若包含任何复杂的子部分，最外层查询则被标记为主查询。

以下是EXPLAIN SELECT score.* FROM student_score score WHERE subject_id = (SELECT i...的执行结果集

	id	select_type	table	partitions
1	1	PRIMARY	score	
2	2	SUBQUERY	subject	
3	3	SUBQUERY	teacher	

SUBQUERY 类型。

```
EXPLAIN SELECT score.*
FROM student_score score
WHERE subject_id = (
    SELECT id
    FROM subject
    WHERE teacher_id = (
            SELECT id
            FROM teacher
            WHERE id = 2
    )
);
```

在 select 或 where 中包含子查询。

以下是EXPLAIN SELECT score.* FROM student_score score WHERE subject_id = (SELECT i...的执行结果集 元数据信息 SQL

	id	select_type	table	partitions
1	1	PRIMARY	score	
2	2	SUBQUERY	subject	
3	3	SUBQUERY	teacher	

UNION 类型。

```
EXPLAIN SELECT subject.*
FROM subject
    LEFT JOIN teacher ON subject.teacher_id = teacher.id
UNION
SELECT subject.*
FROM subject
    RIGHT JOIN teacher ON subject.teacher_id = teacher.id;
```

若第二个 select 出现在 uion 之后，则被标记为 UNION。

以下是EXPLAIN SELECT subject.* FROM subject LEFT JOIN teacher ON subject.teacher_id...的执行结果集 元数据信息

	id	select_type	table	partitions
1	1	PRIMARY	subject	
2	1	PRIMARY	teacher	
3	2	UNION	teacher	
4	2	UNION	subject	

UNION RESULT 类型。

```
EXPLAIN SELECT subject.*
FROM subject
    LEFT JOIN teacher ON subject.teacher_id = teacher.id
UNION
SELECT subject.*
FROM subject
    RIGHT JOIN teacher ON subject.teacher_id = teacher.id;
```

从 UNION 表获取结果的 select。

以下是EXPLAIN SELECT subject.* FROM subject LEFT JOIN teacher ON subject.teacher_id...的执行结果集 元数据信息

	id	select_type	table	partitions
1	1	PRIMARY	subject	
2	1	PRIMARY	teacher	
3	2	UNION	teacher	
4	2	UNION	subject	
5		UNION RESULT	<union1,2>	

步骤 5 Type 字段

表5-2 Type 字段说明

Type字段	意义
ALL	将遍历全表以找到匹配的行
Index	index与ALL区别为index类型只遍历索引树
Range	只检索给定范围的行，使用一个索引来选择行
Index_range	表示使用了索引合并的优化方法
Ref_or_null	类似ref，可以搜索值为NULL的行
Ref	表示上述表的连接匹配条件，即哪些列或常量被用于查找索引列上的值
eq_ref	类似ref，区别就在使用的索引是唯一索引
const、system	对查询某部分进行优化，并转换为一个常量时，使用这些类型访问
NULL	在优化过程中分解语句，执行时甚至不用访问表或索引

NULL 类型。

```
EXPLAIN SELECT MIN(id)
FROM subject;
```

以下是explain select min(id) from subject的执行结果集 元数据信息，不能编辑、翻页和导出SQL

	select_type	table	partitions	type
	SIMPLE			

在优化阶段分解查询语句，在执行阶段用不着再访问表或索引。

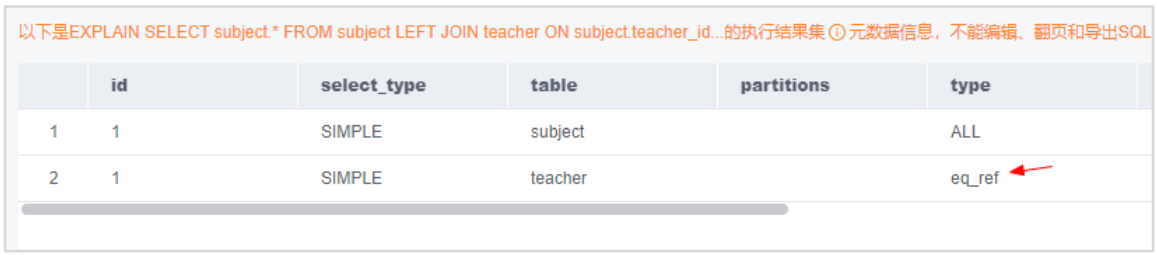
const 类型。

```
EXPLAIN SELECT *
FROM teacher
WHERE teacher_no = 'T2010001';
```

通过索引一次就找到了，const 用于比较 primary key 或 unique 索引，因为只匹配一行数据，所以很快。如将主键置于 where 列表中，就能将该查询转换为一个常量。

eq_ref 类型。

```
EXPLAIN SELECT subject.*
FROM subject
    LEFT JOIN teacher ON subject.teacher_id = teacher.id;
```

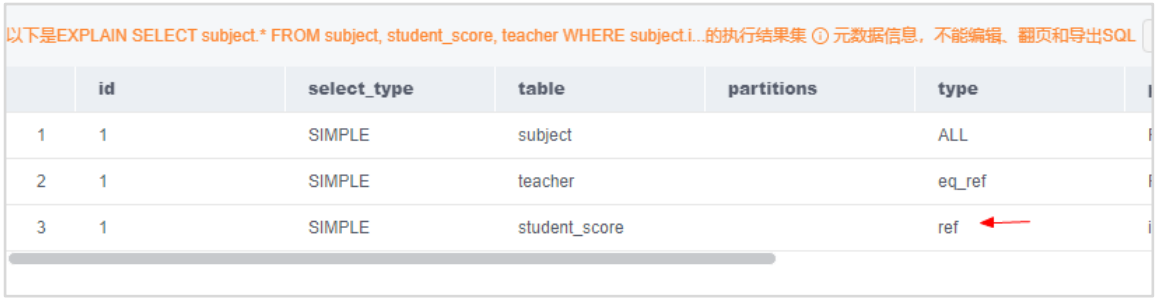


唯一性索引扫描，对于每个索引键，表中只有一条记录与之匹配，常见于主键或唯一索引扫描。

ref 类型。

```
EXPLAIN SELECT subject.*
FROM subject, student_score, teacher
WHERE subject.id = student_id
    AND subject.teacher_id = teacher.id;
```

注：如果结果只有两行，请重新执行一遍，方便元数据信息采集。



非唯一性索引扫描，返回匹配某个单独值的所有行。

本质上也是一种索引访问，返回所有匹配某个单独值的行。

然而可能会找到多个符合条件的行，应该属于查找和扫描的混合体。

ref_or_null 类型。

```
EXPLAIN SELECT *
```

```
FROM teacher
WHERE name = 'wangsi'
   OR name IS NULL;
```

以下是EXPLAIN SELECT * FROM teacher WHERE name = 'wangsi' OR name IS NULL的执行结果集 元数据信息，不能编辑、翻页和导出SQL

	id	select_type	table	partitions	type
1	1	SIMPLE	teacher		ref_or_null

类似 ref，但是可以搜索值为 NULL 的行。

index_merge 类型。

```
EXPLAIN SELECT *
FROM teacher
WHERE id = 1
   OR teacher_no = 'T2010001'
```

以下是EXPLAIN SELECT * FROM teacher WHERE id = 1 OR teacher_no = 'T2010001'的执行结果集 元数据信息，不能编辑、翻页和导出SQL

	id	select_type	table	partitions	type
1	1	SIMPLE	teacher		index_merge

使用了索引合并的优化方法。

range 类型。

```
EXPLAIN SELECT *
FROM subject
WHERE id BETWEEN 1 AND 3;
```

以下是EXPLAIN SELECT * FROM subject WHERE id BETWEEN 1 AND 3的执行结果集 元数据信息，不能编辑、翻页和导出SQL

	id	select_type	table	partitions	type	possible_keys	key
1	1	SIMPLE	subject		range	PRIMARY	PRIMARY

只检索给定范围的行，使用一个索引来选择行，key 列显示使用了哪个索引。

一般在 where 语句中出现 between、<>、in 等的查询时会使用 range。

index 类型。

```
EXPLAIN SELECT id
FROM subject;
```

以下是EXPLAIN SELECT id FROM subject的执行结果集 元数据信息，不能编辑、翻页和导出SQL

	id	select_type	table	partitions	type
1	1	SIMPLE	subject		index

当获取的数据直接通过读取索引树就可以获取时，使用 index 类型。

ALL 类型。

```
EXPLAIN SELECT *
FROM subject;
```

	id	select_type	table	partitions	type
1	1	SIMPLE	subject		ALL

需要遍历全表以找到匹配行。

步骤 6 Extra 字段

指一些不适合在其它列中显示但十分重要的额外信息。

表5-3 Extra 字段说明

Extra字段	意义
Using where	不用读取表中所有信息，仅通过索引就可以获取所需数据
Using temporary	需要使用临时表来存储结果集，常见于排序和分组查询
Using filesort	当Query中包含 order by 操作，而且无法利用索引完成的排序操作
Using join buffer	强调在获取连接条件时没有使用索引，并且需要连接缓冲区来存储中间结果
Impossible where	强调了where语句会导致没有符合条件的行
Distinct	一旦找到了与行相联合匹配的行，就不再搜索
Select tables optimized away	仅通过使用索引，优化器可能仅从聚合函数结果中返回一行
No tables used	Query语句中使用from dual 或不含任何from子句

Using where 内容。

```
EXPLAIN SELECT *
FROM student
WHERE name = 'Bob';
```

	rows	filtered	Extra
	4	25.0	Using where

使用了 where 条件。

Using temporary 内容。

```
EXPLAIN SELECT name, COUNT(name)
```

```
FROM subject
    LEFT JOIN student_score ON subject.id = student_score.subject_id
GROUP BY name;
```

ref	rows	filtered	Extra
	4	100.0	Using temporary
tuning_test_zs.subj...	2	100.0	Using index

Using filesort 内容。

```
EXPLAIN SELECT name
FROM student
ORDER BY name;
```

	rows	filtered	Extra
	4	100.0	Using filesort

Using join buffer 内容。

```
EXPLAIN SELECT student.*, teacher.*, subject.*
FROM student, teacher, subject;
```

...	filtered	Extra
4	100.0	
4	100.0	Using join buffer (Bl...
4	100.0	Using join buffer (Bl...

使用的是 Using join buffer(Block Nested Loop)连接缓存。

Impossible where 内容。

```
EXPLAIN SELECT *
FROM teacher
WHERE name = 'wangsi'
    AND name = 'lisi';
```

rows	filtered	Extra
		Impossible WHERE

where 子句的值总是 false，不能用来获取任何元组。

Distinct 内容。

```
EXPLAIN SELECT DISTINCT teacher.name
FROM teacher
    LEFT JOIN subject ON teacher.id = subject.teacher_id;
```

rows	filtered	Extra
4	100.0	Using index; Using ...
1	100.0	Using index; Distinct

表示一旦找到了一行，就不再搜索了。

Select tables optimized away 内容。

```
EXPLAIN SELECT MIN(id)
FROM subject;
```

	rows	filtered	Extra
			Select tables optimi...

SELECT 操作已经优化到不能再优化了。

No tables used 内容。

```
EXPLAIN SELECT 1 + 1;
```

	rows	filtered	Extra
			No tables used

无需使用 table 来实现 select 语句。

5.2.2 SQL 语句优化

步骤 1　加载测试数据 employees 员工数据

注：测试数据纯属虚构。

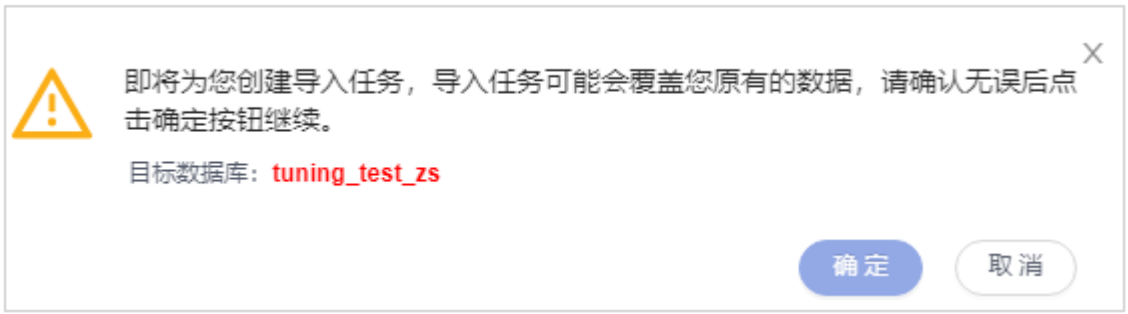
点击“导入·导出”，选择“导入”，选择“新建任务”。

点击“创建 OBS 桶”，然后点击“确定”。

选择需要上传的 employees.sql 文件，并选择数据库为 tuning_test_zs，其他默认。

employees.rar

点击“创建导入任务”，在弹出窗口中点击“确定”。

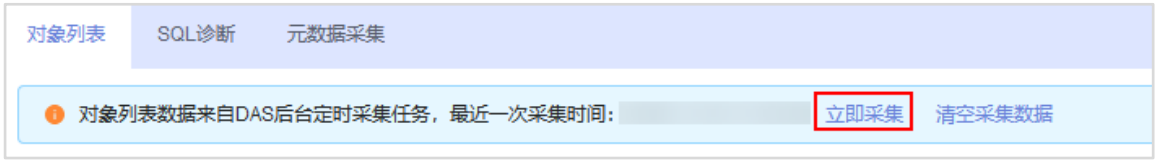


等待 SQL 数据导入。

等待一会后（大致为 6 分钟左右），数据导入成功。

点击“首页”，选择数据库 tuning_test_zs，此时进入“库管理-tuning_test_zs”界面，查看导入效果。

导入完成后，点击“立即采集”，采集元数据。



结果如下：

步骤 2 表结构信息

表之间的关系图如下：

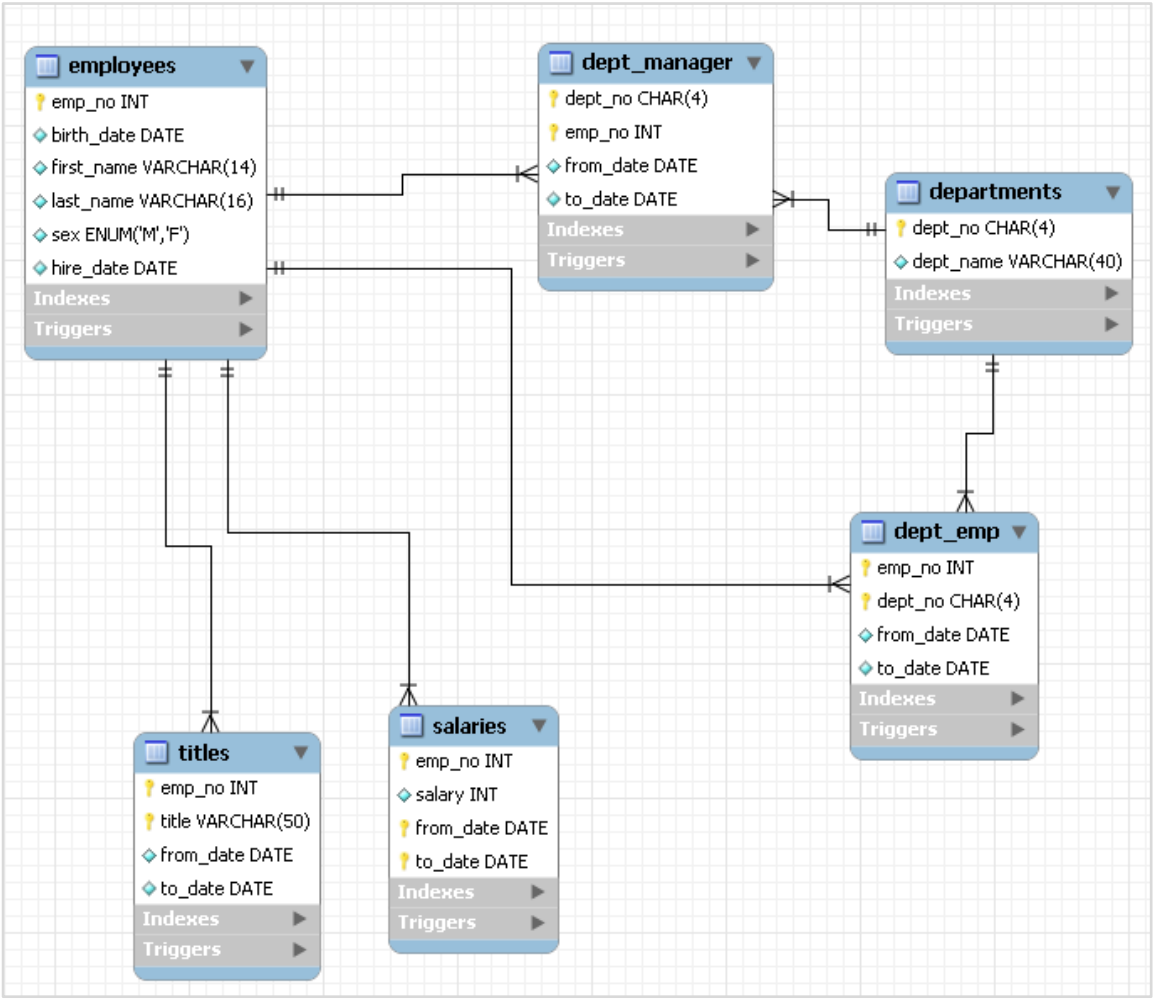


表5-4 员工表 employees

字段	类型	长度	意义	备注
emp_no	int	11	员工id	主键
birth_date	date	-	生日	非空
first_name	varchar	14	姓氏	非空
last_name	varchar	16	名字	非空
gender	enum	M,F	性别	非空
hire_date	date	-	入职时间	非空

表5-5 工资表 salaries

字段	类型	长度	意义	备注
emp_no	int	11	员工id	主键

salary	int	11	工资	主键、索引
from_date	date	-	开始日期	非空
to_date	date	-	结束日期	非空

表5-6 员工职位表 titles

字段	类型	长度	意义	备注
emp_no	int	11	员工id	主键
title	varchar	50	职位名称	主键
from_date	date	-	开始日期	主键
to_date	date	-	结束日期	空

表5-7 管理者部门表 dept_manager

字段	类型	长度	意义	备注
emp_no	int	11	员工id	主键
dept_no	char	4	部门编号	主键
from_date	date	-	开始日期	非空
to_date	date	-	结束日期	非空

表5-8 员工部门表 dept_emp

字段	类型	长度	意义	备注
emp_no	int	11	员工id	主键
dept_no	char	4	部门编号	主键、索引
from_date	date	-	开始日期	非空
to_date	date	-	结束日期	非空

表5-9 部门表 departments

字段	类型	长度	意义	备注
dept_no	char	4	部门编号	主键
dept_name	varchar	40	部门名称	非空、唯一索引

步骤 3 SQL 语句编写

公司允许各个部门举办生日晚会，并按照该部门的寿星人数分配活动资金。现需要数据库人员按照部门查询当下 2020 年 10 月份各个部门的寿星个数、部门名称、所占总寿星比例，并按照人数倒序排序。

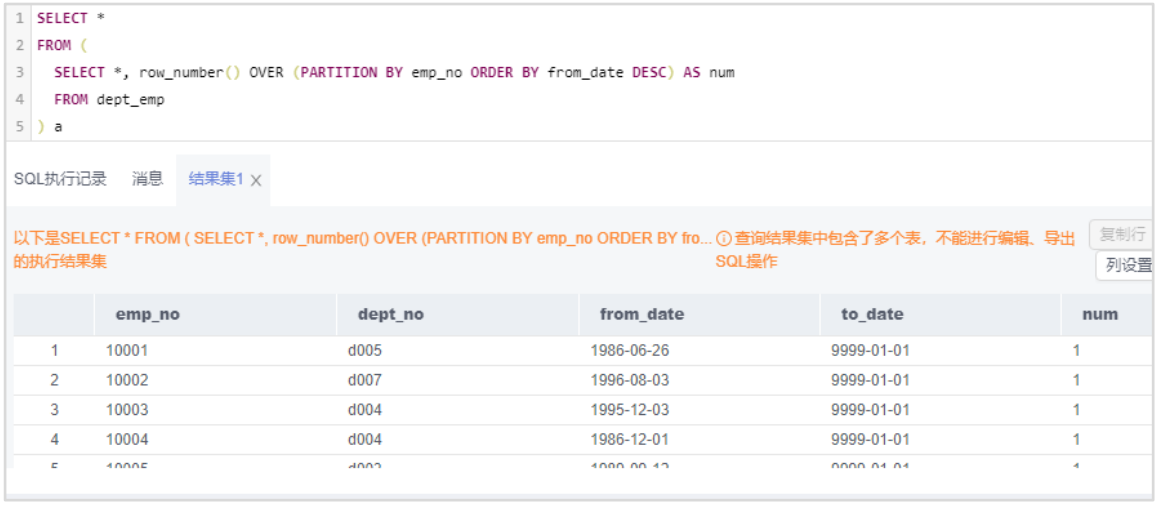
首先根据条件，通过员工表 employees 查询生日在 10 月的员工信息。

```
SELECT emp_no
FROM employees
WHERE date_format(birth_date, '%m') = '10';
```

	emp_no
1	10012
2	10025
3	10039

通过员工部门表 dept_emp 查询当前 2020 年 10 月还在公司的员工信息。

```
SELECT *
FROM (
  SELECT *, row_number() OVER (PARTITION BY emp_no ORDER BY to_date DESC) AS num
  FROM dept_emp
) a
WHERE num = 1
  AND date_format(from_date,'%Y-%m') <= '2020-10'
  AND date_format(to_date,'%Y-%m') >= '2020-10';
```



	emp_no	dept_no	from_date	to_date	num
1	10001	d005	1986-06-26	9999-01-01	1
2	10002	d007	1996-08-03	9999-01-01	1
3	10003	d004	1995-12-03	9999-01-01	1
4	10004	d004	1986-12-01	9999-01-01	1

查询对应员工的部门名称 dept_name，需要和部门员工表 dept_emp 以及部门表 departments 做 join。

```
SELECT a.emp_no, d.dept_name
FROM (
    SELECT emp_no
    FROM employees
    WHERE date_format(birth_date, '%m') = '10'
) a
    JOIN (
        SELECT emp_no, dept_no
        FROM (
            SELECT *, row_number() OVER (PARTITION BY emp_no ORDER BY to_date DESC) AS num
            FROM dept_emp
        ) b
        WHERE num = 1
            AND date_format(from_date, '%Y-%m') <= '2020-10'
            AND date_format(to_date, '%Y-%m') >= '2020-10'
    ) c
    ON a.emp_no = c.emp_no
    JOIN departments d ON c.dept_no = d.dept_no;
```

以下是SELECT a.emp_no, d.dept_name FROM (SELECT emp_no FROM employees WHERE date_f...的执行结果集

ⓘ 查询结果集中包含了多个表，不能进行编辑、导出SQL操作

	emp_no	dept_name
1	10060	Customer Service
2	10183	Customer Service
3	10817	Customer Service
4	10841	Customer Service
5	10971	Customer Service

根据部门名称，查询各个部门的寿星个数。

```
SELECT dept_name, COUNT(dept_name) AS num
FROM (
    SELECT emp_no
    FROM employees
    WHERE date_format(birth_date, '%m') = '10'
) a
    JOIN (
        SELECT emp_no, dept_no
        FROM (
            SELECT *, row_number() OVER (PARTITION BY emp_no ORDER BY to_date DESC) AS num
            FROM dept_emp
        ) b
        WHERE num = 1
            AND date_format(from_date, '%Y-%m') <= '2020-10'
            AND date_format(to_date, '%Y-%m') >= '2020-10'
    ) c
    ON a.emp_no = c.emp_no
    JOIN departments d ON c.dept_no = d.dept_no
GROUP BY dept_name;
```

以下是SELECT dept_name, COUNT(dept_name) AS num FROM (SELECT emp_no FROM employees...的执行结果集 ⓘ 查询结果集中包含了多个表，SQL操作

	dept_name	num
1	Customer Service	1543
2	Development	5179
3	Finance	1055
4	Human Resources	1006
5	Marketing	1252

题干要求各个部门寿星所占比例，因此需要用 sum 函数求出所有的寿星个数。

```
SELECT SUM(1) AS total
FROM (
    SELECT emp_no
    FROM employees
    WHERE date_format(birth_date, '%m') = '10'
) a
    JOIN (
        SELECT emp_no, dept_no
        FROM (
            SELECT *, row_number() OVER (PARTITION BY emp_no ORDER BY to_date DESC) AS num
            FROM dept_emp
        ) b
        WHERE num = 1
            AND date_format(from_date, '%Y-%m') <= '2020-10'
            AND date_format(to_date, '%Y-%m') >= '2020-10'
    ) c
    ON a.emp_no = c.emp_no
    JOIN departments d ON c.dept_no = d.dept_no;
```

SQL执行记录 消息 结果集1 ×

以下是SELECT SUM(1) FROM (SELECT emp_no FROM employees WHERE da
集

	SUM(1)
1	20375

将 sum 结果结合起来，使用逗号或 join。

```
SELECT f.dept_name, f.num, g.total
FROM (
    SELECT dept_name, COUNT(dept_name) AS num
    FROM (
        SELECT emp_no
        FROM employees
```

```
        WHERE date_format(birth_date, '%m') = '10'
    ) a
        JOIN (
            SELECT emp_no, dept_no
            FROM (
                SELECT *, row_number() OVER (PARTITION BY emp_no ORDER BY to_date DESC) AS
num
                FROM dept_emp
            ) b
            WHERE num = 1
                AND date_format(from_date, '%Y-%m') <= '2020-10'
                AND date_format(to_date, '%Y-%m') >= '2020-10'
        ) c
        ON a.emp_no = c.emp_no
        JOIN departments d ON c.dept_no = d.dept_no
    GROUP BY dept_name
) f, (
        SELECT SUM(1) AS total
        FROM (
            SELECT emp_no
            FROM employees
            WHERE date_format(birth_date, '%m') = '10'
        ) a
            JOIN (
                SELECT emp_no, dept_no
                FROM (
                    SELECT *, row_number() OVER (PARTITION BY emp_no ORDER BY to_date DESC)
AS num
                    FROM dept_emp
                ) b
                WHERE num = 1
                    AND date_format(from_date, '%Y-%m') <= '2020-10'
                    AND date_format(to_date, '%Y-%m') >= '2020-10'
            ) c
            ON a.emp_no = c.emp_no
            JOIN departments d ON c.dept_no = d.dept_no
    ) g;
```

以下是SELECT f.dept_name, f.num, g.total FROM (SELECT dept_name, COUNT(dept_name) ...的执行结果集

ⓘ 查询结果集中包含了多个表，不能进行编辑、导出SQL操作

	dept_name	num	total
1	Customer Service	1543	20375
2	Development	5179	20375
3	Finance	1055	20375
4	Human Resources	1006	20375
5	Marketing	1252	20375

最后做排序和相关输出格式化操作。

```
SELECT f.dept_name, f.num, g.total
```

```
    , round(f.num / g.total * 100, 2) AS percent
FROM (
    SELECT dept_name, COUNT(dept_name) AS num
    FROM (
        SELECT emp_no
        FROM employees
        WHERE date_format(birth_date, '%m') = '10'
    ) a
        JOIN (
            SELECT emp_no, dept_no
            FROM (
                SELECT *, row_number() OVER (PARTITION BY emp_no ORDER BY to_date DESC) AS
num
                FROM dept_emp
            ) b
            WHERE num = 1
                AND date_format(from_date, '%Y-%m') <= '2020-10'
                AND date_format(to_date, '%Y-%m') >= '2020-10'
        ) c
        ON a.emp_no = c.emp_no
        JOIN departments d ON c.dept_no = d.dept_no
    GROUP BY dept_name
) f
    JOIN (
        SELECT SUM(1) AS total
        FROM (
            SELECT emp_no
            FROM employees
            WHERE date_format(birth_date, '%m') = '10'
        ) a
            JOIN (
                SELECT emp_no, dept_no
                FROM (
                    SELECT *, row_number() OVER (PARTITION BY emp_no ORDER BY to_date DESC)
AS num
                    FROM dept_emp
                ) b
                WHERE num = 1
                    AND date_format(from_date, '%Y-%m') <= '2020-10'
                    AND date_format(to_date, '%Y-%m') >= '2020-10'
            ) c
            ON a.emp_no = c.emp_no
            JOIN departments d ON c.dept_no = d.dept_no
    ) g
ORDER BY f.num DESC;
```

以下是SELECT f.dept_name, f.num, g.total , round(f.num / g.total * 100, 2) AS perce...的执行结果集 ⓘ 查询结果集中包含了多个表，不能进行编辑、导出SQL操作 复制行 复制列 列设置

	dept_name	num	total	percent
1	Development	5179	20375	25.42
2	Production	4531	20375	22.24
3	Sales	3269	20375	16.04
4	Customer Service	1543	20375	7.57
5	Research	1332	20375	6.54

点击“消息”，查看 SQL 耗时。

SQL执行记录 消息 结果集1

```
                ) a
                        JOIN (
                                SELECT emp_no, dept_no
                                FROM (
                                        SELECT *, row_number() OVER (PARTITION BY emp_no ORDER BY to_date DESC) AS num
                                        FROM dept_emp
                                ) b
                                WHERE num = 1
                                        AND date_format(from_date, '%Y-%m') <= '2020-10'
                                        AND date_format(to_date, '%Y-%m') >= '2020-10'
                        ) c
                        ON a.emp_no = c.emp_no
                        JOIN departments d ON c.dept_no = d.dept_no
        ) g
ORDER BY f.num DESC
执行成功，当前返回：[9]行，耗时：[2361ms.]
```

步骤 4 SQL 语句修改

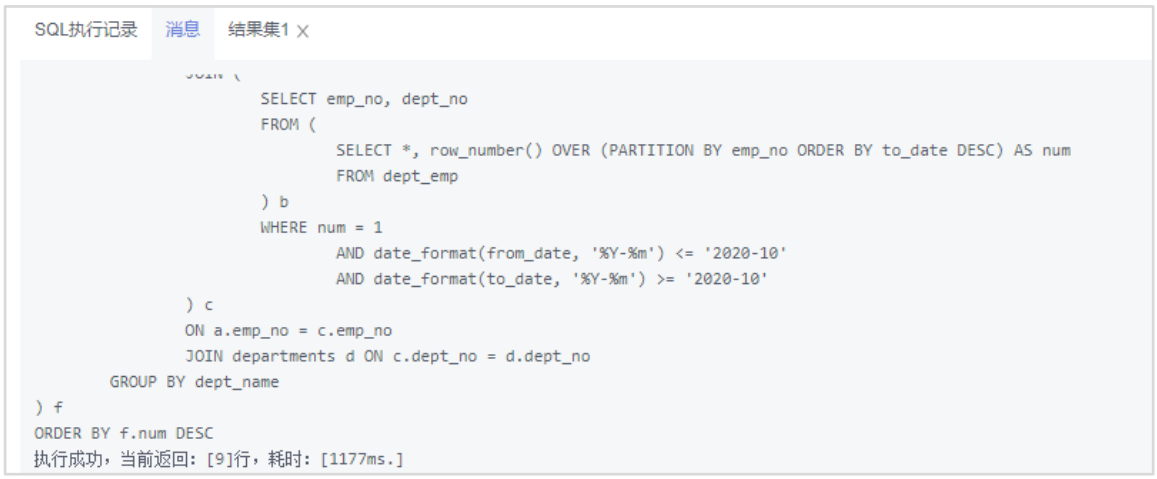
SQL 语句中两个比较大的 SQL 语句进行拼接，为了减少 SQL 语句的执行，可以使用 SUM OVER()函数做简单优化。

```
SELECT f.dept_name, f.num, SUM(f.num) OVER () total
    , round(f.num / SUM(f.num) OVER () * 100, 2) AS percent
FROM (
    SELECT dept_name, COUNT(dept_name) AS num
    FROM (
        SELECT emp_no
        FROM employees
        WHERE date_format(birth_date, '%m') = '10'
    ) a
        JOIN (
            SELECT emp_no, dept_no
            FROM (
                SELECT *, row_number() OVER (PARTITION BY emp_no ORDER BY to_date DESC) AS
num
                FROM dept_emp
            ) b
            WHERE num = 1
                AND date_format(from_date, '%Y-%m') <= '2020-10'
                AND date_format(to_date, '%Y-%m') >= '2020-10'
        ) c
        ON a.emp_no = c.emp_no
        JOIN departments d ON c.dept_no = d.dept_no
```

```
    GROUP BY dept_name
) f
ORDER BY f.num DESC;
```

	dept_name	num	total	percent
1	Development	5179	20375	25.42
2	Production	4531	20375	22.24
3	Sales	3269	20375	16.04
4	Customer Service	1543	20375	7.57
5	Research	1332	20375	6.54
6	Marketing	1252	20375	6.14

再次点击“消息”，查看查询耗时。

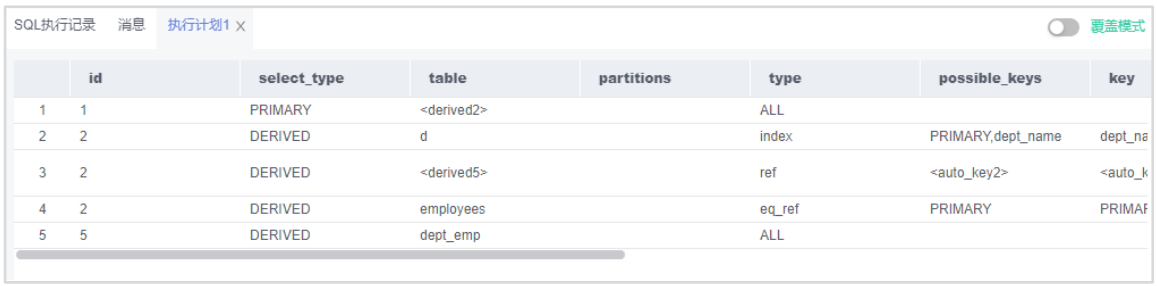


相比于上一次时间缩短了 50%。

步骤 5 EXPLAIN 查看执行计划

点击“执行计划”，查看执行计划。

效果如下：



	id	select_type	table	partitions	type	possible_keys	key
1	1	PRIMARY	<derived2>		ALL		
2	2	DERIVED	d		index	PRIMARY,dept_name	dept_na
3	2	DERIVED	<derived5>		ref	<auto_key2>	<auto_k
4	2	DERIVED	employees		eq_ref	PRIMARY	PRIMAF
5	5	DERIVED	dept_emp		ALL		

key	key_len	ref	rows	filtered	Extra
			4470	100.0	Using filesort
dept_name	122		9	100.0	Using index
<auto_key2>	12	tuning_test_zs.d.dept_no	4967	10.0	Using where
PRIMARY	4	b.emp_no	1	100.0	Using where
			331143	100.0	Using filesort

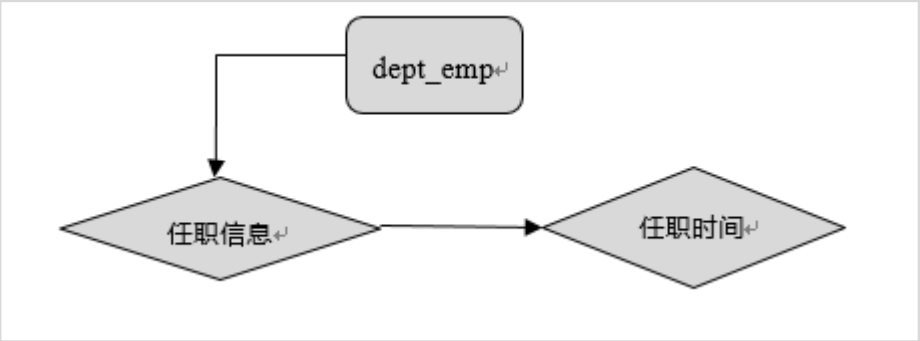
重点查看 type 类型为 ALL 的计划。

可以看到对于表 dept_emp 在查询员工是否在职的信息时，使用了全表扫描的方式。

在查询 employees 表时，也没有使用索引的方式，而是 Using where。

步骤 6　修改查询是否在职的 SQL 语句

是否在职原先的 SQL 查询逻辑是，首先找寻员工当前所在的部门最新的任职信息，然后比较任职期间是否在 2020 年 10 月。



业务逻辑分析：

任职时间满足条件的员工必定还在公司，并未离职，同时也必定在最新的部门中。

因此直接通过任职时间来获取，会减少 SQL 逻辑计算。

```
SELECT *
FROM dept_emp
WHERE date_format(from_date, '%Y-%m') <= '2020-10'
    AND date_format(to_date, '%Y-%m') >= '2020-10';
```

以下是SELECT * FROM dept_emp WHERE date_format(from_date, '%Y-%m') <= '2020-10' AND...的执行结果集　点击单元格可编辑数据，新增或编辑后需要提交编辑以保存　复制行　复制列　列设置

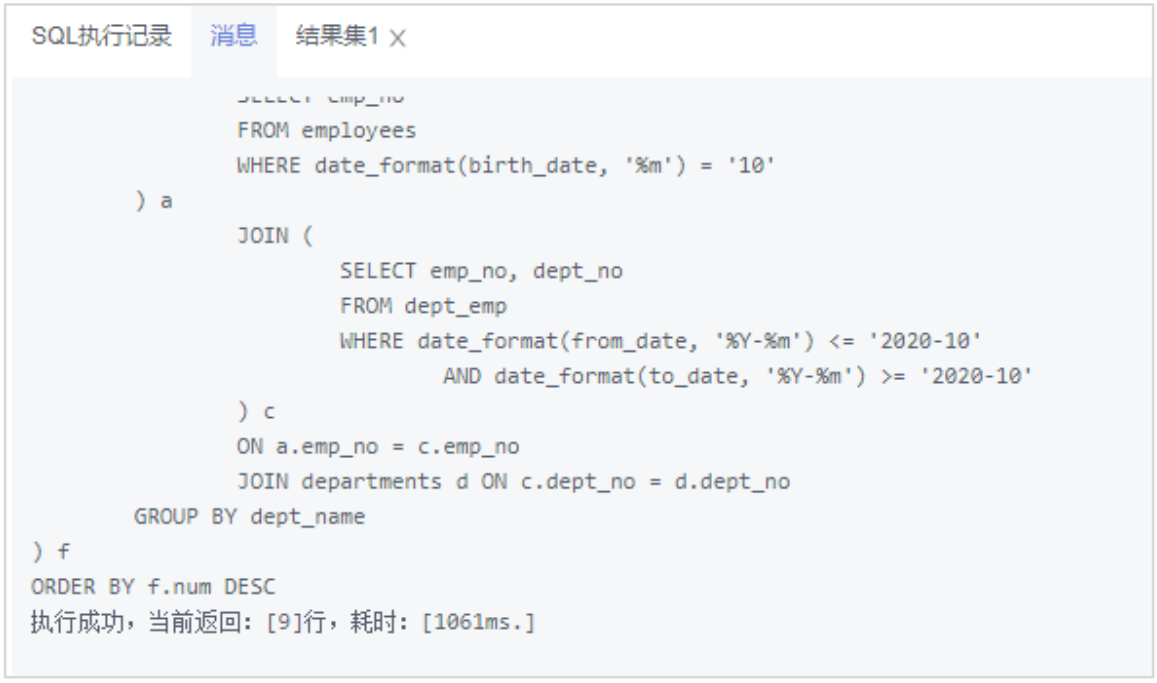
	emp_no	dept_no	from_date	to_date
1	10001	d005	1986-06-26	9999-01-01
2	10002	d007	1996-08-03	9999-01-01
3	10003	d004	1995-12-03	9999-01-01

整体 SQL 为：

```
SELECT f.dept_name, f.num, SUM(f.num) OVER () AS total
    , round(f.num / SUM(f.num) OVER () * 100, 2) AS percent
FROM (
    SELECT dept_name, COUNT(dept_name) AS num
    FROM (
        SELECT emp_no
```

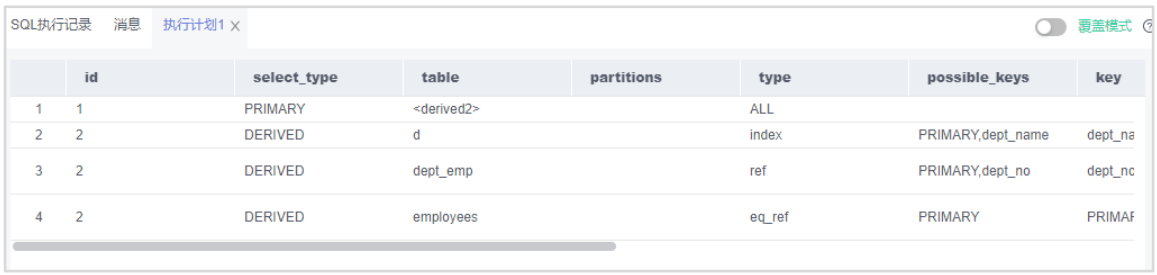
```
        FROM employees
        WHERE date_format(birth_date, '%m') = '10'
    ) a
        JOIN (
            SELECT emp_no,dept_no
            FROM dept_emp
            WHERE date_format(from_date, '%Y-%m') <= '2020-10'
                AND date_format(to_date, '%Y-%m') >= '2020-10'
        ) c
        ON a.emp_no = c.emp_no
        JOIN departments d ON c.dept_no = d.dept_no
    GROUP BY dept_name
) f
ORDER BY f.num DESC;
```

执行之后的 SQL 耗时为：



此时整体耗时降低了 10%。

再次点击“执行计划”，结果为：



key	key_len	ref	rows	filtered	Extra
			372535	100.0	Using filesort
dept_name	122		9	100.0	Using index
dept_no	12	tuning_test_zs.d.dept_no	41392	100.0	Using where
PRIMARY	4	tuning_test_zs.dept_emp.emp_no	1	100.0	Using where

执行计划从原来的 5 层变成了 4 层，dept_emp 的 type 从 ALL 变更为 ref。dept_emp 和 employees 在查询时间范围数据时都是使用的 use where。

查看 SQL 语法，发现对于时间的比较是通过函数处理时间字段，然后再与字符串比较，因此哪怕创建了索引也无法使用索引。

```
date_format(birth_date, '%m') = '10'
date_format(from_date, '%Y-%m') <= '2020-10'
date_format(to_date, '%Y-%m') >= '2020-10'
```

同时因对生日的比对是月份比对，如果需要优化，需要单独抽离月份字段，因此月份比对的处理保留，需要处理的是对 from_date 和 to_date 的优化。

步骤 7　修改时间比对处理方式

首先创建日期索引。

```
ALTER TABLE dept_emp ADD INDEX index_date (from_date,to_date);
```

SQL执行记录　消息

```
----------开始执行----------
【拆分SQL完成】：将执行SQL语句数量：（1条）
【执行SQL：(1)】
ALTER TABLE dept_emp ADD INDEX index_date (from_date,to_date)
执行成功，耗时：[402ms.]
```

根据数据库对于时间处理时使用索引的原则，修改时间比对处理如下：

```
from_date <= '2020-10-01'
to_date >= '2020-10-01'
```

整体 SQL 语句如下：

```
SELECT f.dept_name, f.num, SUM(f.num) OVER () AS total
    , round(f.num / SUM(f.num) OVER () * 100, 2) AS percent
FROM (
    SELECT dept_name, COUNT(dept_name) AS num
    FROM (
            SELECT emp_no
            FROM employees
            WHERE date_format(birth_date, '%m') = '10'
    ) a
            JOIN (
                SELECT emp_no, dept_no
```

```
            FROM dept_emp
            WHERE from_date <= '2020-10-01'
                AND to_date >= '2020-10-01'
        ) c
        ON a.emp_no = c.emp_no
        JOIN departments d ON c.dept_no = d.dept_no
    GROUP BY dept_name
) f
ORDER BY f.num DESC;
```

查看执行计划如下：

id	select_type	table	...	type	possible_keys	key	key_len	ref
1	PRIMARY	<derived2>		ALL				
2	DERIVED	d		index	PRIMARY,dept_name	dept_name	122	
2	DERIVED	dept_emp		ref	PRIMARY,dept_no,index_date	dept_no	12	.d.dept_no
2	DERIVED	employees		eq_ref	PRIMARY	PRIMARY	4	.dept_emp.emp_no

虽然创建了索引，但依旧没有生效，因此使用 Hint 语法强制使用索引。

```
SELECT f.dept_name, f.num, SUM(f.num) OVER () AS total
    , round(f.num / SUM(f.num) OVER () * 100, 2) AS percent
FROM (
    SELECT dept_name, COUNT(dept_name) AS num
    FROM (
        SELECT emp_no
        FROM employees
        WHERE date_format(birth_date, '%m') = '10'
    ) a
        JOIN (
            SELECT emp_no, dept_no
            FROM dept_emp FORCE INDEX (index_date)
            WHERE from_date <= '2020-10-01'
                AND to_date >= '2020-10-01'
        ) c
        ON a.emp_no = c.emp_no
        JOIN departments d ON c.dept_no = d.dept_no
    GROUP BY dept_name
) f
ORDER BY f.num DESC;
```

此时查询耗时为：

SQL执行记录 消息 结果集1 ×

```
                FROM employees
                WHERE date_format(birth_date, '%m') = '10'
        ) a
                JOIN (
                        SELECT emp_no, dept_no
                        FROM dept_emp FORCE INDEX (index_date)
                        WHERE from_date <= '2020-10-01'
                                AND to_date >= '2020-10-01'
                ) c
                ON a.emp_no = c.emp_no
                JOIN departments d ON c.dept_no = d.dept_no
        GROUP BY dept_name
) f
ORDER BY f.num DESC
执行成功，当前返回: [9]行，耗时: [811ms.]
```

整体耗时相比于上一次效率提升了 20%。

执行计划为：

id	se...	table	...	type	possible_keys	key	key_len	ref
1	PRIMARY	<derived2>		ALL				
2	DERIVED	dept_emp		range	index_date	index_date	3	
2	DERIVED	d		eq_ref	PRIMARY,dept_name	PRIMARY	12	tuning_test_hql.dept_emp.dept_no
2	DERIVED	employees		eq_ref	PRIMARY	PRIMARY	4	tuning_test_hql.dept_emp.emp_no

再次分析执行计划，可以发现首先处理的是 dept_emp 表，然后是 d 表也就是 departments 表，最后是 employees 表。

步骤 8 修改表 JOIN 顺序

数据库表在 join 过程中一般来说小表在左（优先）大表在右，为了让小表优先在缓存中。因此查看各个子查询中的结果数，比对数据量大小。

查看 dept_emp 表数据。

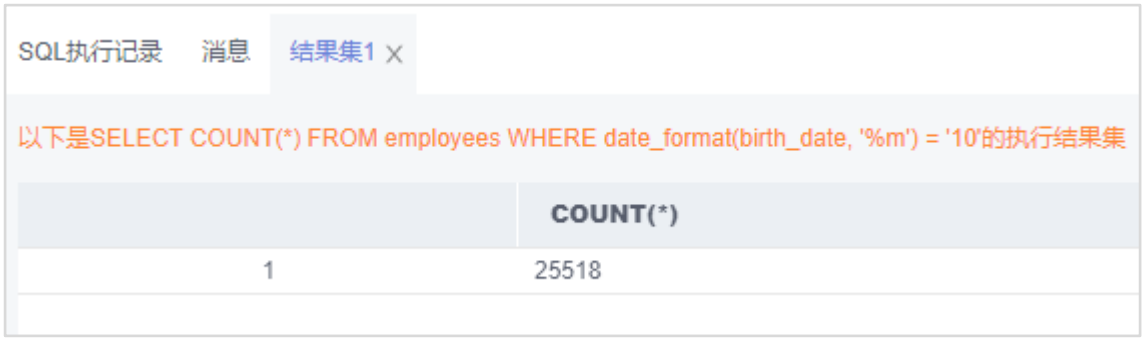
```
SELECT COUNT(*)
FROM dept_emp FORCE INDEX (index_date)
WHERE from_date <= '2020-10-01'
   AND to_date >= '2020-10-01';
```

结果为：

查看 employees 表数据。

```
SELECT COUNT(*)
FROM employees
WHERE date_format(birth_date, '%m') = '10';
```

结果为：



查看 departments 表数据。

```
SELECT COUNT(*)
FROM departments;
```

结果为：

因此可以发现数据量大小为 departments<employees<dept_emp。

修改 SQL 语法如下：

```
SELECT f.dept_name, f.num, SUM(f.num) OVER () AS total
    , round(f.num / SUM(f.num) OVER () * 100, 2) AS percent
FROM (
    SELECT dept_name, COUNT(dept_name) AS num
    FROM departments a
        STRAIGHT_JOIN (
            SELECT emp_no, dept_no
```

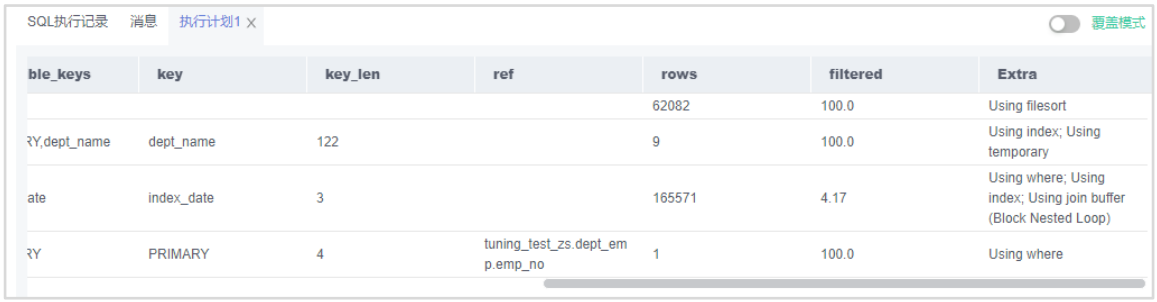
```
            FROM dept_emp FORCE INDEX (index_date)
            WHERE from_date <= '2020-10-01'
                AND to_date >= '2020-10-01'
        ) b
        ON a.dept_no = b.dept_no
        JOIN (
            SELECT emp_no
            FROM employees
            WHERE date_format(birth_date, '%m') = '10'
        ) c
        ON b.emp_no = c.emp_no
    GROUP BY dept_name
) f
ORDER BY f.num DESC;
```

此时查询耗时：

时间为 655ms，相比上次提升了 20%。

此时执行计划为：

	id	select_type	table	partitions	type	possible_keys
1	1	PRIMARY	<derived2>		ALL	
2	2	DERIVED	a		index	PRIMARY,dept_name
3	2	DERIVED	dept_emp		range	index_date
4	2	DERIVED	employees		eq_ref	PRIMARY



SQL执行记录 消息 执行计划1 ×

覆盖模式

ble_keys	key	key_len	ref	rows	filtered	Extra
				62082	100.0	Using filesort
RY,dept_name	dept_name	122		9	100.0	Using index; Using temporary
ate	index_date	3		165571	4.17	Using where; Using index; Using join buffer (Block Nested Loop)
RY	PRIMARY	4	tuning_test_zs.dept_emp.emp_no	1	100.0	Using where

步骤 9 最外层优化

最外层为一个 GROUP BY 和一个 order by 语句，可以考虑结合一起来执行。

```
SELECT dept_name, COUNT(dept_name) AS num
    , SUM(COUNT(dept_name)) OVER () AS total
    , round(COUNT(dept_name) / SUM(COUNT(dept_name)) OVER () * 100, 2) AS percent
FROM departments a
    STRAIGHT_JOIN (
        SELECT emp_no, dept_no
        FROM dept_emp FORCE INDEX (index_date)
        WHERE from_date <= '2020-10-01'
            AND to_date >= '2020-10-01'
    ) b
    ON a.dept_no = b.dept_no
    JOIN (
        SELECT emp_no
        FROM employees
        WHERE date_format(birth_date, '%m') = '10'
    ) c
    ON b.emp_no = c.emp_no
GROUP BY dept_name
ORDER BY COUNT(dept_name) DESC;
```

此时执行时间为：

SQL执行记录 消息 结果集1 ×

```
        SELECT emp_no, dept_no
        FROM dept_emp FORCE INDEX (index_date)
        WHERE from_date <= '2020-10-01'
                AND to_date >= '2020-10-01'
    ) b
    ON a.dept_no = b.dept_no
    JOIN (
        SELECT emp_no
        FROM employees
        WHERE date_format(birth_date, '%m') = '10'
    ) c
    ON b.emp_no = c.emp_no
GROUP BY dept_name
ORDER BY COUNT(dept_name) DESC
执行成功，当前返回：[9]行，耗时：[647ms.]
```

性能并未提升太多，因为 count 函数计算了 5 次（SQL 语法中 count 函数执行了 5 次），原先的 SQL 语法 count 函数只计算了一次。

执行结果如下：

以下是SELECT dept_name, COUNT(dept_name) AS num , SUM(COUNT(dept_name)) OVER () AS ...的执行结果集ⓘ 虚表不能进行编辑、导出SQL操作 复制行 复制

	dept_name	num	total	percent
1	Development	5179	20375	25.42
2	Production	4531	20375	22.24
3	Sales	3269	20375	16.04
4	Customer Service	1543	20375	7.57
5	Research	1332	20375	6.54
6	Marketing	1252	20375	6.14
7	Quality Management	1208	20375	5.93
8	Finance	1055	20375	5.18

查看执行计划：

SQL执行记录 消息 执行计划1 × 覆盖模式

	id	select_type	table	partitions	type	possible_keys	key
1	1	SIMPLE	a		index	PRIMARY,dept_name	dept_na
2	1	SIMPLE	dept_emp		range	index_date	index_d
3	1	SIMPLE	employees		eq_ref	PRIMARY	PRIMAI

ble_keys	key	key_len	ref	rows	filtered	Extra
RY,dept_name	dept_name	122		9	100.0	Using index; Using temporary; Using filesort
ate	index_date	3		165571	4.17	Using where; Using index; Using join buffer (Block Nested Loop)
RY	PRIMARY	4	tuning_test_zs.dept_emp.emp_no	1	100.0	Using where

最终只剩 3 层。

至此本次的 SQL 优化操作结束，从原先的 2361ms 到最后的 647ms，整体 SQL 性能提升了 70%。

5.3 思考题

SQL 语法的调优后一定会比之前更好吗?

参考答案：SQL 调优过程中按照一定的理论基础去实践是没有问题的，但最后的效果不一定会比之前的 SQL 语句更好，SQL 的耗时除了本身语法之外，还与其他硬件资源有关，需要综合考虑。

6 数据库安全管理实验

6.1 实验介绍

6.1.1 关于本实验

本实验通过设定安全组、SSL 连接及 DAS 用户权限设置实验介绍基于 DAS 的数据库安全管理。

6.1.2 实验目的

- 掌握数据库安全组设定；
- 掌握 SSL 连接；
- 掌握 DAS 用户权限设置。

6.2 实验准备

6.2.1 购买部署 Linux 系统

在华为云上购买一个 ECS（linux 系统，本实验以 Ubuntu 18.04.4 系统为例）。

步骤 1 在华为云网页搜索 ECS，找到“弹性云服务器 ECS”，点击“立即购买”。

步骤 2 根据如下配置，进行购买。

计费模式：按需计费

区域：选择和数据库同一个区域

CPU 架构：x86 计算

规格：通用计算增强型 c6.large.4 2vCPUs|8GB

镜像：公共镜像 Ubuntu Ubuntu 18.04 server 64bit（40GB）

主机安全：不勾选

系统盘：通用型 SSD 40GB

步骤 3 点击“下一步：网络配置”，根据如下设置完成配置。

网络：选择和数据库相同的 VPC。

安全组：选择和数据库相同的安全组。

线路：全球动态 BGP

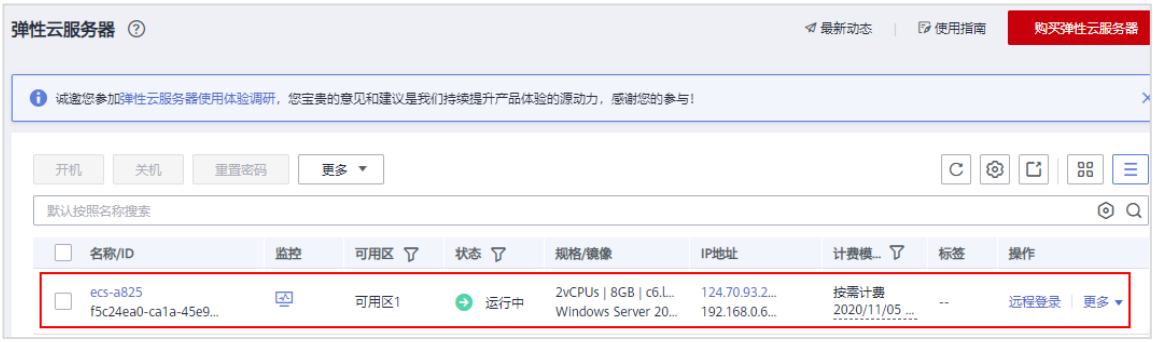
公网带宽：按流量计算

带宽大小：5Mbit/s

步骤 4 点击“下一步：高级配置”，根据参数设置完成配置。

步骤 5 配置完成后，点击“立即购买”，完成采购。

步骤 6 可在“弹性云服务器”界面，查看已购买的 ECS。



6.2.2 在 Linux 系统上安装 mysql8.0 server

步骤 1 使用 putty 或 MobaXterm 等软件连接 ECS，并登录。

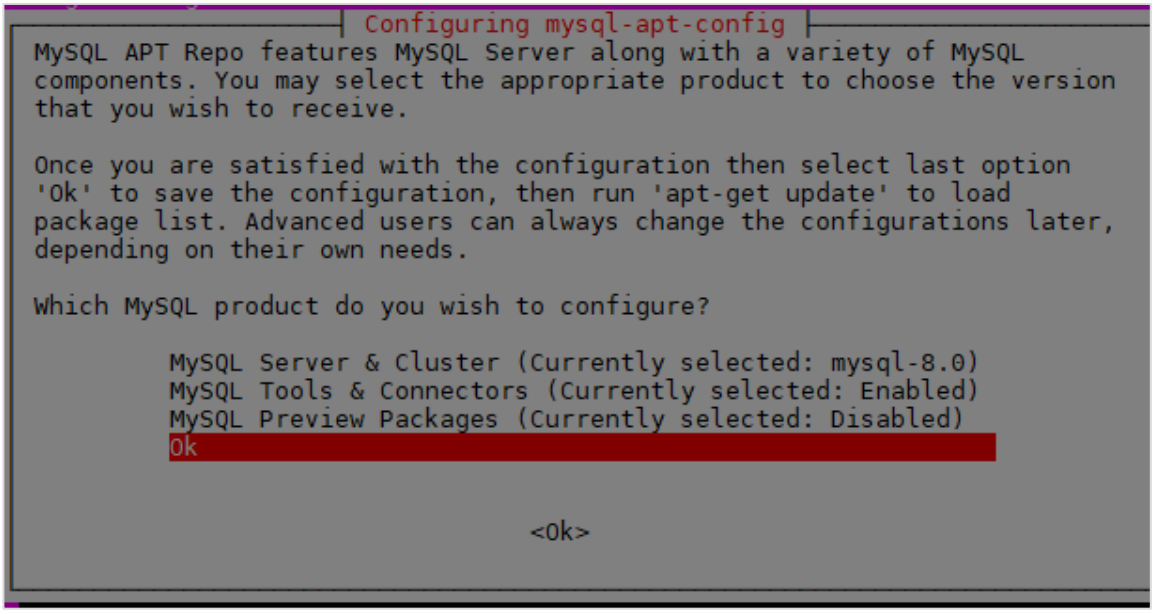
步骤 2 使用命令下载软件包：

```
wget -c https://dev.mysql.com/get/mysql-apt-config_0.8.10-1_all.deb
```

步骤 3 使用命令安装下载的安装包。

```
sudo dpkg -i mysql-apt-config_0.8.10-1_all.deb
```

安装过程中提示选择安装版本，默认安装的就是 8.0 版本，所以直接选择“OK”确认即可。



步骤 4 从所有已配置的存储库（包括新添加的 MySQL 8 存储库）中下载最新的软件包信息。

```
sudo apt update
```

步骤 5 在执行步骤 4 过程中，会出现签名无效等错误

```
W: GPG error: http://repo.mysql.com/apt/ubuntu bionic InRelease: The following signatures were invalid: EXPKEYSIG 8C718D3B5072E1F5 MySQL Release Engineering <mysql-build@oss.oracle.com>
E: The repository 'http://repo.mysql.com/apt/ubuntu bionic InRelease' is not signed.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
```

查看签名列表：

```
apt-key list
```

```
root@ecs-216c:~# apt-key list
/etc/apt/trusted.gpg
--------------------
pub   dsa1024 2003-02-03 [SCA] [expired: 2019-02-17]
      A4A9 4068 76FC BD3C 4567  70C8 8C71 8D3B 5072 E1F5
uid           [ expired] MySQL Release Engineering <mysql-build@oss.oracle.com>

/etc/apt/trusted.gpg.d/ubuntu-keyring-2012-archive.gpg
-----------------------------------------------------
pub   rsa4096 2012-05-11 [SC]
      790B C727 7767 219C 42C8  6F93 3B4F E6AC C0B2 1F32
uid           [ unknown] Ubuntu Archive Automatic Signing Key (2012) <ftpmaster@
ubuntu.com>

/etc/apt/trusted.gpg.d/ubuntu-keyring-2012-cdimage.gpg
-----------------------------------------------------
pub   rsa4096 2012-05-11 [SC]
      8439 38DF 228D 22F7 B374  2BC0 D94A A3F0 EFE2 1092
uid           [ unknown] Ubuntu CD Image Automatic Signing Key (2012) <cdimage@u
buntu.com>

/etc/apt/trusted.gpg.d/ubuntu-keyring-2018-archive.gpg
```

删除过期的签名：

sudo apt-key del dsa1024

重新添加新的签名：

sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 8C718D3B5072E1F5

```
root@ecs-216c:~# sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 8
C718D3B5072E1F5
Executing: /tmp/apt-key-gpghome.QfnqBHsUsw/gpg.1.sh --keyserver keyserver.ubuntu
.com --recv-keys 8C718D3B5072E1F5
gpg: key 8C718D3B5072E1F5: 1 duplicate signature removed
gpg: key 8C718D3B5072E1F5: 56 signatures not checked due to missing keys
gpg: key 8C718D3B5072E1F5: "MySQL Release Engineering <mysql-build@oss.oracle.co
m>" 29 new signatures
gpg: Total number processed: 1
gpg:         new signatures: 29
```

之后继续执行更新命令：

sudo apt update

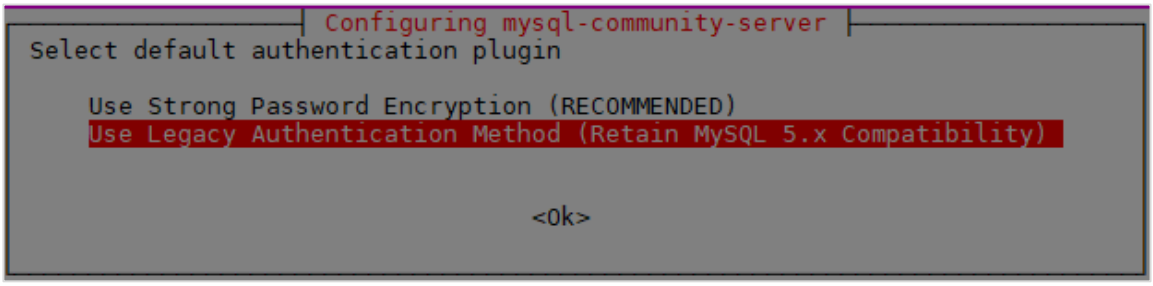
步骤 6 完成下载后，安装 Mysql8.0。

sudo apt install mysql-server

安装过程中会提示设置 root 密码，按照提示输入即可。

注意：这里可能需要 15 分钟左右下载时间，视具体网络状况而定。

步骤 7 选择“Use Legacy Authentication Method (Retain MySQL 5.X Compatibility)”。



步骤 8 验证安装完成。

mysql -uroot -p

输入 root 密码，如下图，则表明登录成功。

```
root@ecs-88f8:~# mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.22 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

6.3 数据库安全组设定

步骤 1 在云数据库“GaussDB(for MySQL)”界面，点击数据库实例名称，进入实例管理界面。

步骤 2 在“网络信息”中，找到“内网安全组”，点击安全组名称，进入安全组管理界面。

步骤 3 点击编辑图标，可修改安全组名称及描述。

步骤 4 点击“入方向规则”，进入入方向规则设置界面。

步骤 5 点击“添加规则”，弹出“添加入方向规则”。

步骤 6 在协议端口中选择“TCP”及端口“3306”，其它使用默认参数，点击确认完成配置。

注意：

源地址可以是单个 IP 地址、IP 地址段或安全组：

单个 IP 地址：例如 192.168.10.10/32

IP 地址段：例如 192.168.52.0/24

所有 IP 地址：0.0.0.0/0

安全组：例如 sg-abc

步骤 7 出方向规则设置和入方向一样，一般情况下只需要设置入方向规则即可。

6.4 SSL 连接实验

6.4.1 绑定弹性公网 IP

步骤 1 在云数据库“GaussDB(for MySQL)”界面，点击数据库实例名称，进入实例管理界面。

步骤 2 在“网络信息”模块，单击“读写公网地址”后面的“绑定”

步骤 3 在“绑定弹性公网 IP”页面，查看弹性公网 IP，如果没有可用资源，则点击“查看弹性公网 IP”进行购买。

步骤 4 点击“购买弹性公网 IP”进行购买。

步骤 5 根据以下参数设置完成购买。

计费模式：按需计费

区域：和数据库同一个区域

线路：全动态 BGP

公网带宽：按流量计算

带宽大小：5Mbit/s

使用默认参数，点击“立即购买”。

确认购买参数，点击“提交”。

步骤 6 返回“绑定弹性公网 IP”页面，点击刷新。

步骤 7 点击“确定”，完成绑定。

步骤 8 确认绑定状态。

6.4.2 通过 SSL 连接 GaussDB(for MySQL)

步骤 1 在云数据库“GaussDB(for MySQL)”界面，点击数据库实例名称，进入实例管理界面。

步骤 2 在“实例管理”页面，单击实例名称进入“基本信息”页面，单击“数据库信息”模块“SSL”处的下载图标，下载根证书或捆绑包。

步骤 3 下载完成后，将解压后的“ca.pem”上传到弹性云服务器 Linux 操作系统，放到任意文件夹下。

步骤 4 输入以下指令，连接 GaussDB(for MySQL)数据库实例。

```
mysql –h<hostname> -P3306 –uroot –p --ssl-ca=ca.pem
```

```
root@ecs-88f8:~# mysql -h124.70.94.32 -P3306 -uroot -p --ssl-ca=ca.pem
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 278137
Server version: 8.0.18 Source distribution

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

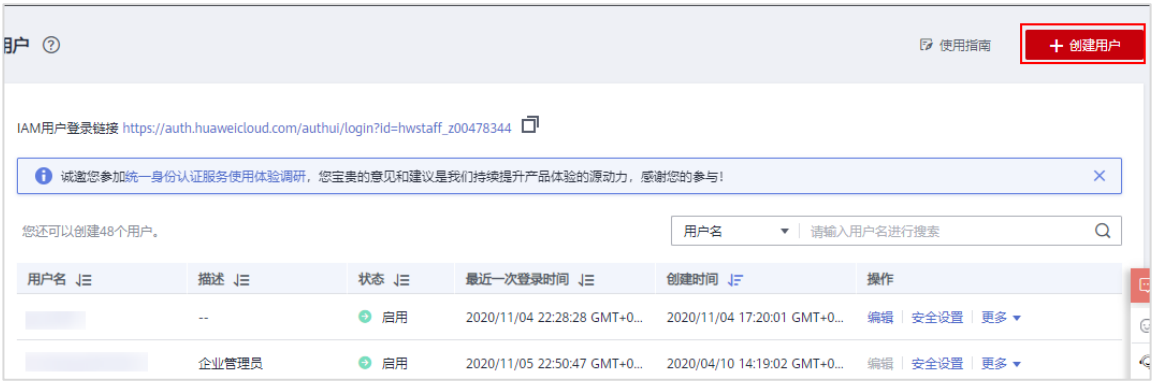
6.5 用户权限配置

6.5.1 创建用户组并授权

步骤 1 登录华为云，在右上角单击控制台。

步骤 2 在控制台页面，鼠标移动至右上方的账号名，在下拉列表中选择“统一身份认证”。

步骤 3 在统一身份认证服务，左侧导航窗格中，单击“用户组”>“创建用户”。

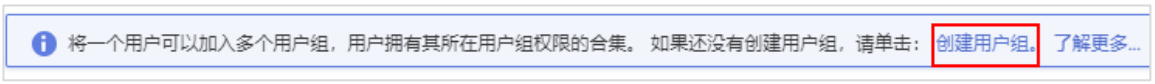


步骤 4 在“用户信息”中输入用户名。

步骤 5 在“访问方式”中，选择“华为云管理控制台访问”，在“自定义”中输入密码，取消“首次登录时重置密码”的选项。

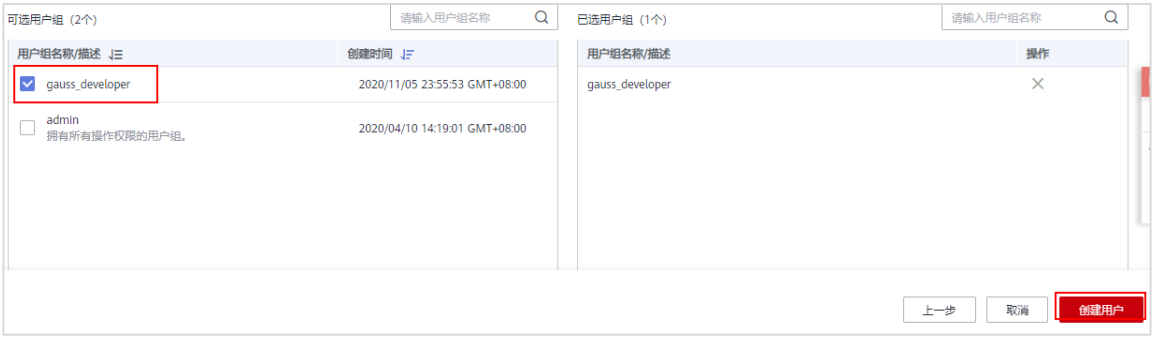
步骤 6 其它配置使用默认参数，点击创建用户。

步骤 7 创建用户组。



步骤 8 在“创建用户组”页面输入用户组名称，点击确定。

步骤 9 勾选新建的用户组，点击“创建用户”。



步骤 10 完成用户创建。

6.5.2 给用户组授权

步骤 1 在用户组列表中，单击新建用户组”gauss_developer”，右侧的“权限配置”。

步骤 2 在用户组权限页签中，单击列表左上方的“配置权限”。

步骤 3 选择“区域级项目”，选择数据库所在区域，此处根据实际数据库所在区域选择。

步骤 4 选择“数据管理服务（DAS）”，勾选“DAS FullAccess”、“DAS Administrator”。

步骤 5 选择“云数据库 GaussDB(GaussDB)”，勾选“GaussDB ReadOnlyAccess”、“GaussDB FullAccess”。

步骤 6 点击确定，完成配置。

6.5.3 新用户测试

步骤 1 退出当前登录，回到登录界面。

步骤 2 点击“IAM 用户登录”。

步骤 3 在“IAM 用户登录”界面，输入 IAM 用户所属账号名称、IAM 用户及密码，点击登录。

步骤 4 在服务列表中，点击“云数据库 GaussDB”。

步骤 5 在云数据库 GaussDB 页面，点击登录。

步骤 6 在实例登录中，输入数据库账号密码，进行登录。

步骤 7 登录数据库，证明该新用户具备权限。

步骤 8 可尝试使用别的服务，会因为权限不足，而提示联系主账户重新授权。

7 场景化综合实验

7.1 实验介绍

7.1.1 关于本实验

本实验指导书共包含 3 个实验：

- 子实验一 7.2 以薪酬系统为例，将数据导入线上 GaussDB(for MySQL)数据库中，并针对线上的数据库实例完成对应的用户创建、权限管理以及对应的参数修改等相关操作，主要目的是为了让读者熟悉 GaussDB(for MySQL)数据库的基本运维和数据管理服务(DAS)的使用。
- 子实验二 7.3 以薪酬系统为例，使用高阶 SQL 和 PLSQL 完成部分功能的展示，并结合数据管理服务(DAS)的云 DBA 功能，完成对 GaussDB(for MySQL)数据库的优化工作。主要目的是为了让读者掌握高阶 SQL 语法、GaussDB(for MySQL)数据库优化的思路和如何结合数据管理服务(DAS)对数据库进行运维管理。
- 子实验三 7.4 以电商系统为例，使用 RDS for MySQL 部署，并通过数据复制服务(DRS)同步迁移至 GaussDB(for MySQL)数据库中；对 GaussDB(for MySQL)进行备份并模拟数据库被删除后的恢复操作；对删除的动作通过云审计服务(CTS)进行查询和追踪。

本实验中的薪酬系统数据模型和电商数据模型，主要是为了实现实验操作而构造的，若与现实场景中模型相似，纯属巧合。

7.2 数据库运维操作

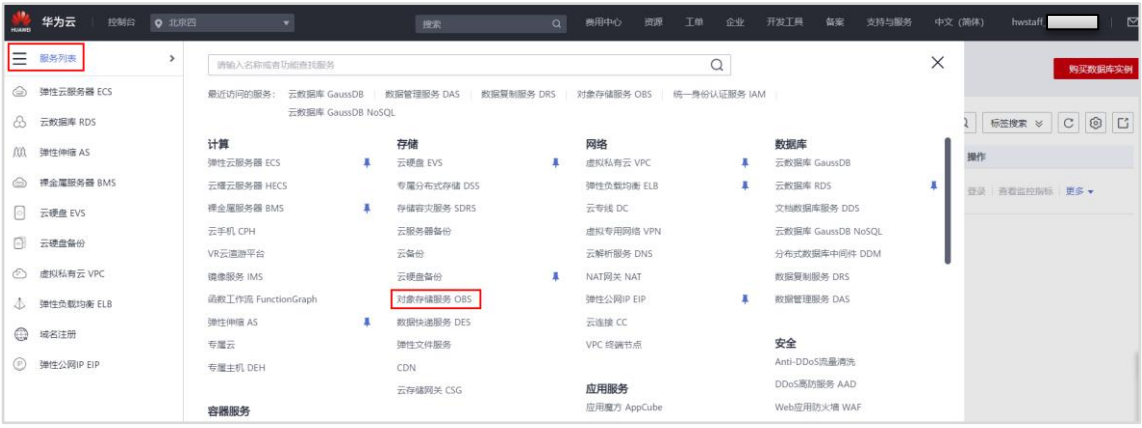
7.2.1 数据导入

实验数据如下：

实验数据.rar

步骤 1 将需要导入的数据上传至对象存储服务(OBS)中。

在华为云“控制台”界面，选择左侧的“服务列表”，在“存储”中选择“对象存储服务”。



步骤 2 点击右上角的“创建桶”，进入创建界面。

步骤 3 对 OBS 桶进行命名，并选择相应的选项，完成后点击“立即创建”。

由于本实验对 OBS 的要求不高，可以使用低频存储，可参考以下方式，如果是生产环境可按照业务需求进行选择。注意：区域需跟 GaussDB(for MySQL)在同一区域中。

步骤 4 在创建完成的 obs 列表中，点击刚才创建的 obs 桶，进入 obs 中。

步骤 5 选择左侧的“对象”模块，点击里面的“上传对象”。

步骤 6 将本地文件拖拽至虚框中或点击“添加文件”按钮，添加完成后，点击“上传”。

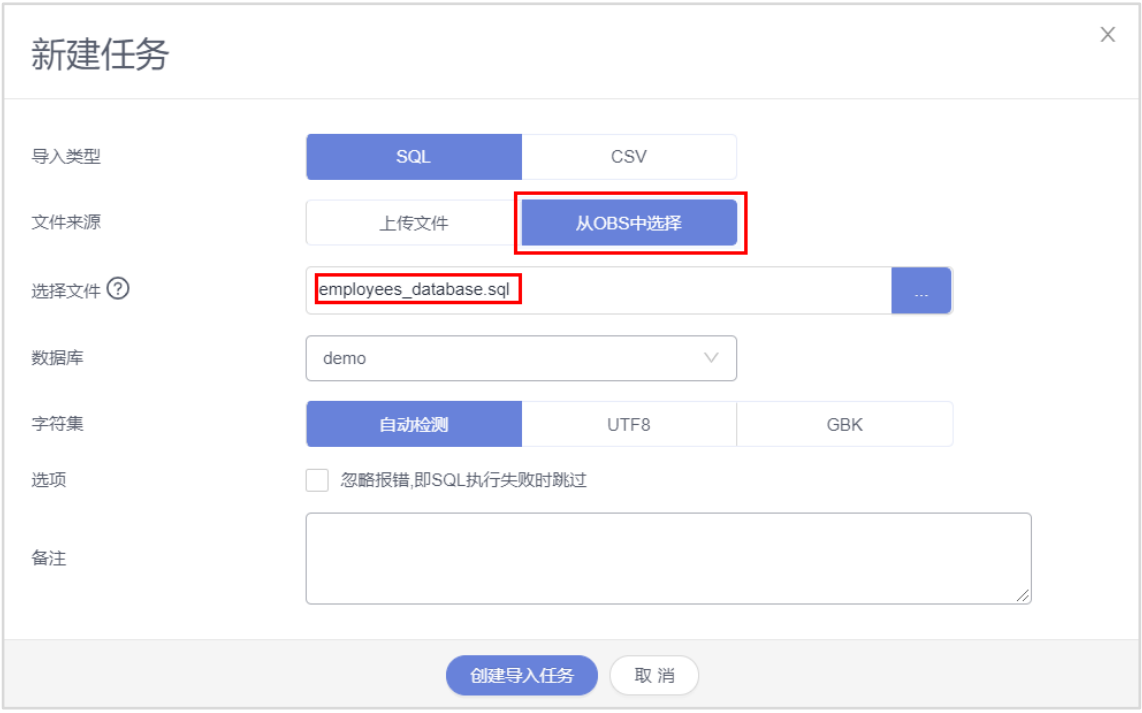
上传进度可以通过底框“任务管理”中的“正在运行”查看。

步骤 7 返回 DAS 标准版中，在“导入导出”中选择“导入”。

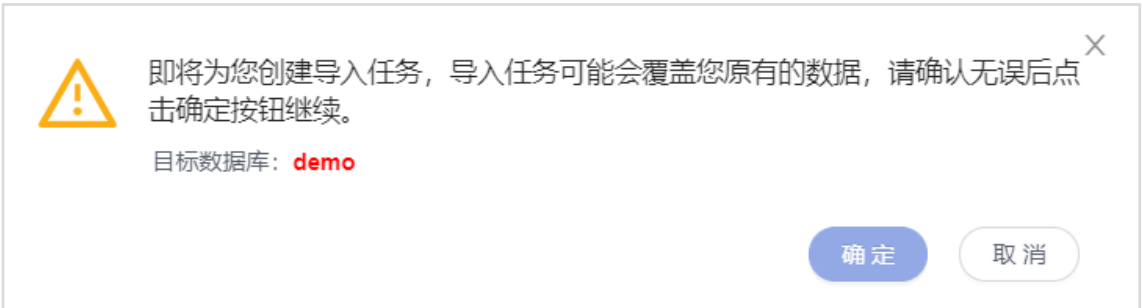
步骤 8 在导入任务中，点击“新建任务”。

步骤 9 在“文件来源”选择“从 OBS 中选择”，并选取上传至 OBS 中的文件，其他选项可保持默认，完成后，点击“创建导入任务”。

注意：目前导入任务只支持 SQL 和 CSV 文件；首次使用需要创建 OBS 桶，后续可直接使用“上传文件”（跳过前面创建 OBS 的过程），上传到创建完成的 OBS 桶中；字符集可以使用自动检测或者指定，根据实际情况进行选择；因为本 SQL 文本中，包含创建数据库的 SQL，所以无需提前手工创建数据库，如果 SQL 文本中没有 create database 语句，需选择对应的数据库进行导入；推荐不要跳过执行失败的 SQL，不方便进行定位。



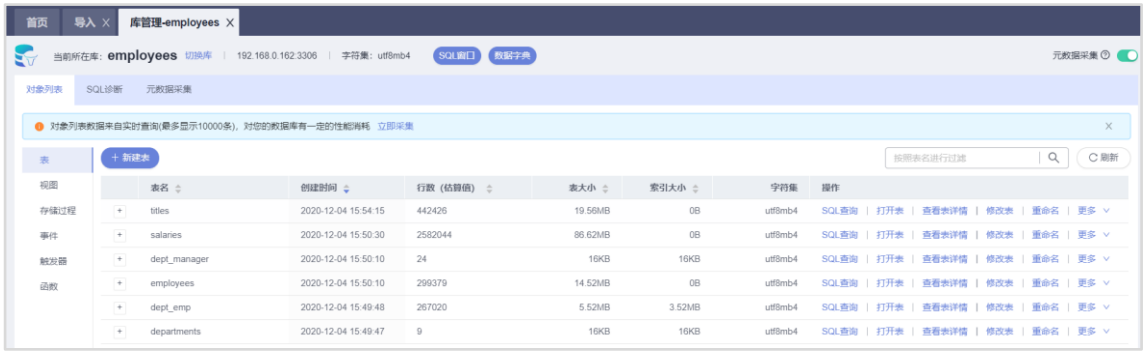
注：导入时可能会有数据被覆盖，请确认。



步骤 10 等待几分钟后，当任务状态变为“已完成”时，导入成功；一旦中间出现异常时，因为创建导入任务时，未勾选“跳过错误”，所以任务终止，可以点击“详情”对错误进行查看。

步骤 11 返回“首页”后刷新，看到 employees 数据库。

步骤 12 点击“库管理”，看到导入的表和具体的数据值。



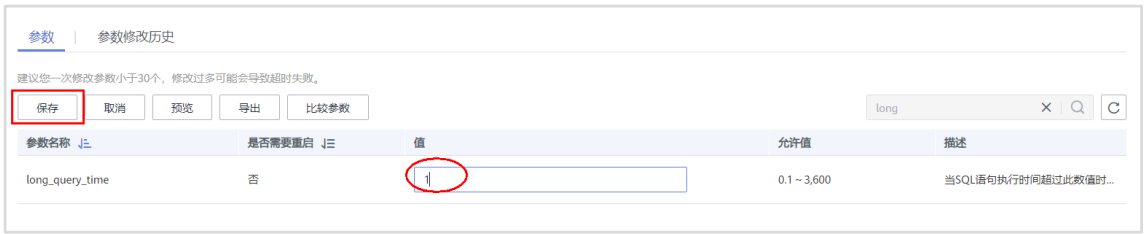
7.2.2 数据库实例参数修改

步骤 1 从华为云控制台界面左侧的服务列表中，选择云数据库 GaussDB。

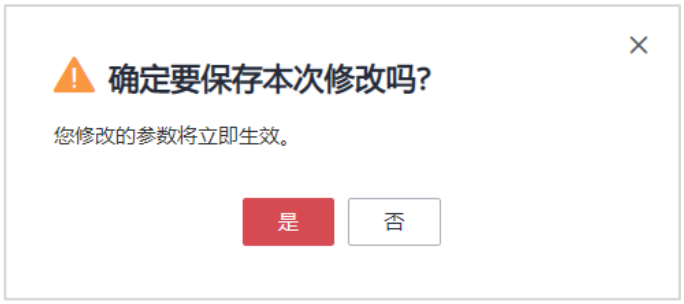
步骤 2 在云数据库 GaussDB(for MySQL)的实例列表中，选择数据库实例，在“更多”中点击“参数修改”。

步骤 3 在右侧的查询框中，输入“long”进行查询。

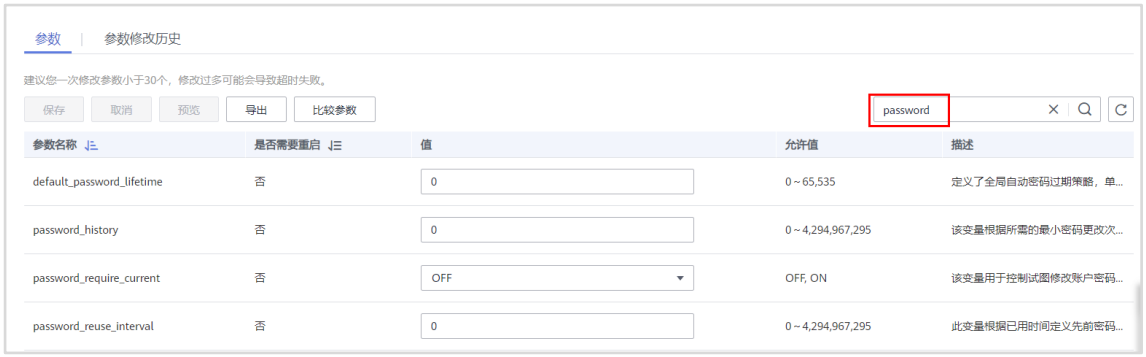
步骤 4 将查找到的参数“long_query_time”的值由默认的 10 改成 1，并点击“保存”。



步骤 5 再次确认后，点击“是”。

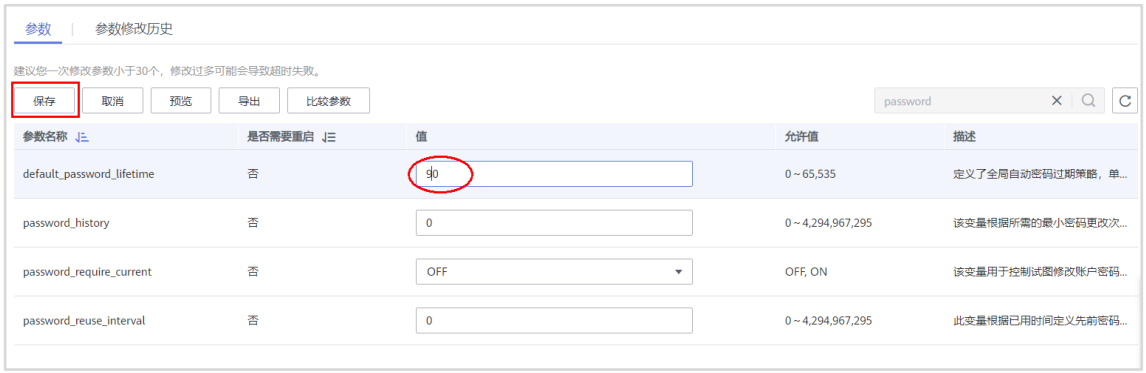


步骤 6 在右侧的查询框中，输入“password”进行查询。



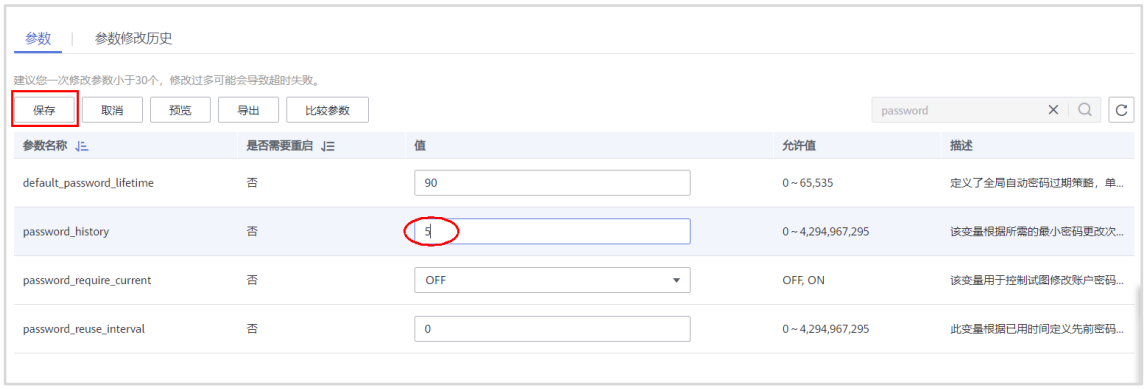
步骤 7 选择“default_password_lifetime”参数，将其参数值修改为 90，并点击“保存”。

default_password_lifetime 参数为密码的有效时间，是全局参数，用户可以根据不同业务系统的安全要求，进行设置。同时也可以使用 alter 语句，对单个用户进行密码周期设置，这部分内容在下一小节进行学习。



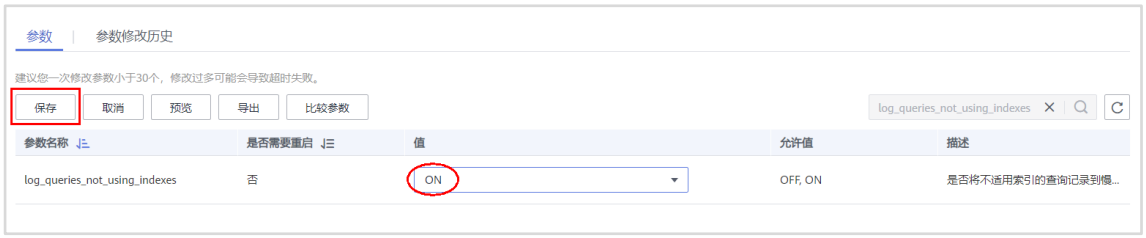
步骤 8 选择“password_history”参数，将其参数值修改为 5，并点击“保存”。

password_history 参数为密码重用前需要修改的次数，为提高安全性，很多系统要求密码不能重复使用，因此将该参数设置为 5 表明在修改 5 次密码后，才能重用之前设置过的密码。



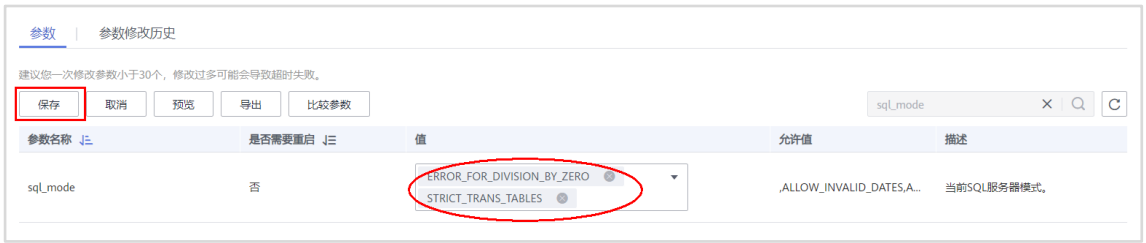
步骤 9 在右侧的查询框中，输入“log_queries_not_using_indexes”进行查询。将其值修改为 on，并点击“保存”。

该参数值为将 SQL 语句中未使用索引的 SQL 记录到慢日志中。即使该 SQL 执行的时间未达到 long_query_time 参数设置值，只要未使用索引都会记录。



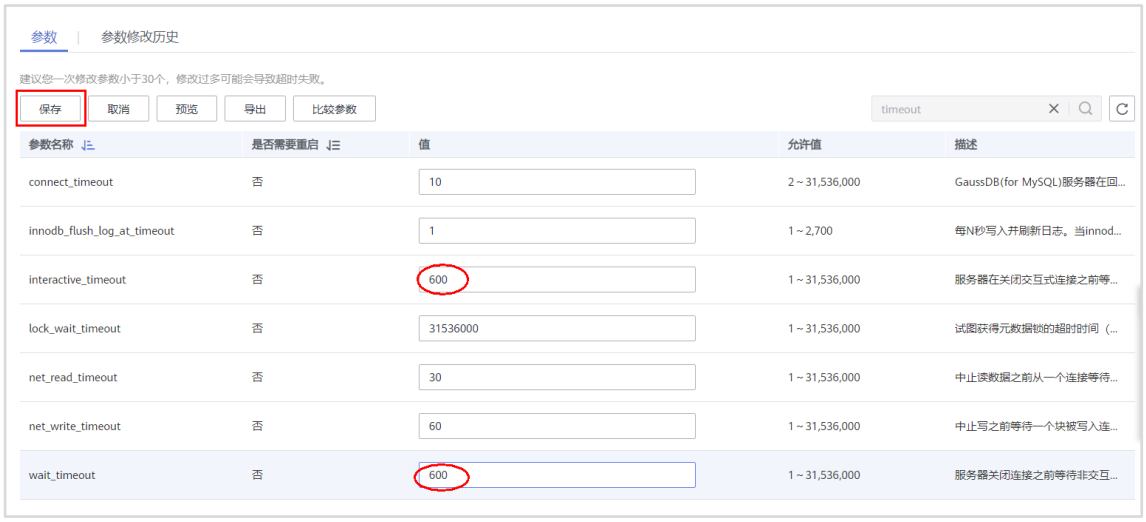
步骤 10 在右侧的查询框中，输入“sql_mode”进行查询。修改 sql_mode 参数，在复选框中，勾选上 STRICT_TRANS_TABLES、ERROR_FOR_DIVISION_BY_ZERO 两个个值，并点击“保存”。

用户可以根据系统定义，自行修改 sql_mode 值。STRICT_TRANS_TABLES 严格按照 SQL 语法执行，一旦有非法 SQL 语法，返回报错；ERROR_FOR_DIVISION_BY_ZERO，当 insert 或 update 时，被除数为 0 时，直接报错而非抛出告警。



步骤 11 在右侧的查询框中，输入“timeout”进行查询。将参数“interactive_timeout”和“wait_timeout”修改为 600 秒，并点击“保存”。

interactive_timeout 和 wait_timeout 参数分别为交互式和非交互式连接，空闲状态下被关闭的时间，默认是 8 小时，建议改为 10 分钟。也就是当一个线程处于 sleep 状态下，10 分钟后被后台进程释放。



步骤 12 点击“比较参数”，并选择默认的参数模板，可以比较出修改后的参数值。

步骤 13 下图中，为实验中修改后的参数值与原模板中的参数值。

参数名称	当前实例的参数	Default-GaussDB-for-MySQL 8.0
default_password_lifetime	90	0
interactive_timeout	600	28800
log_queries_not_using_indexes	ON	OFF
long_query_time	1	10
password_history	5	0
sql_mode	ERROR_FOR_DIVISION_BY_ZERO,STRICT_TRANS_TABL...	--
wait_timeout	600	28800

7.2.3 用户的管理

步骤 1 登陆 DAS 标准版，在数据库列表中，选择导入的 employees 数据库，点击右侧的“SQL 查询”。

步骤 2 在 SQL 命令框中，使用 create 语句创建用户，并点击“执行 SQL”。

注：此处是为了方便实验才设置为弱口令密码，生产系统请设置复杂密码。

```
CREATE USER 'hrapp'@'%' IDENTIFIED BY 'Huawei@qer321';
```

步骤 3 返回执行成功的消息，则用户创建完成。

步骤 4 在 SQL 命令框中，使用 grant 语句为创建的 hrapp 用户授权。

```
GRANT SELECT,INSERT,UPDATE,DELETE,CREATE on employees.* to 'hrapp'@'%';
```

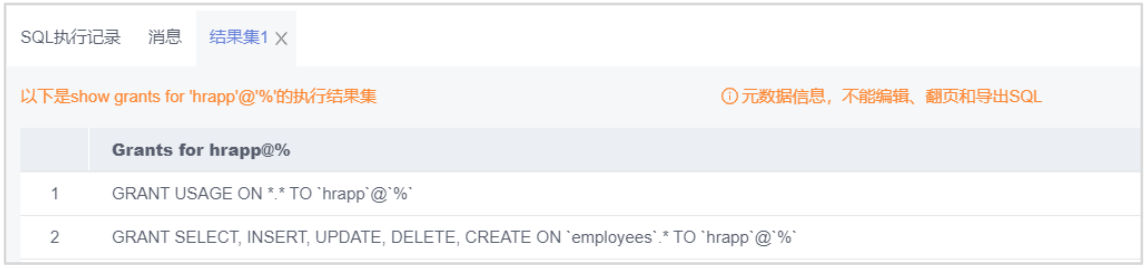
注：以上权限为实验环节需要使用，生产环境可以依照不同的需求对用户进行权限分配。

步骤 5 返回执行成功的消息，则给用户授权完成。

步骤 6 在 SQL 命令框中，执行 SHOW GRANT FOR 命令，查看用户的具体权限。

```
SHOW GRANTS FOR 'hrapp'@'%';
```

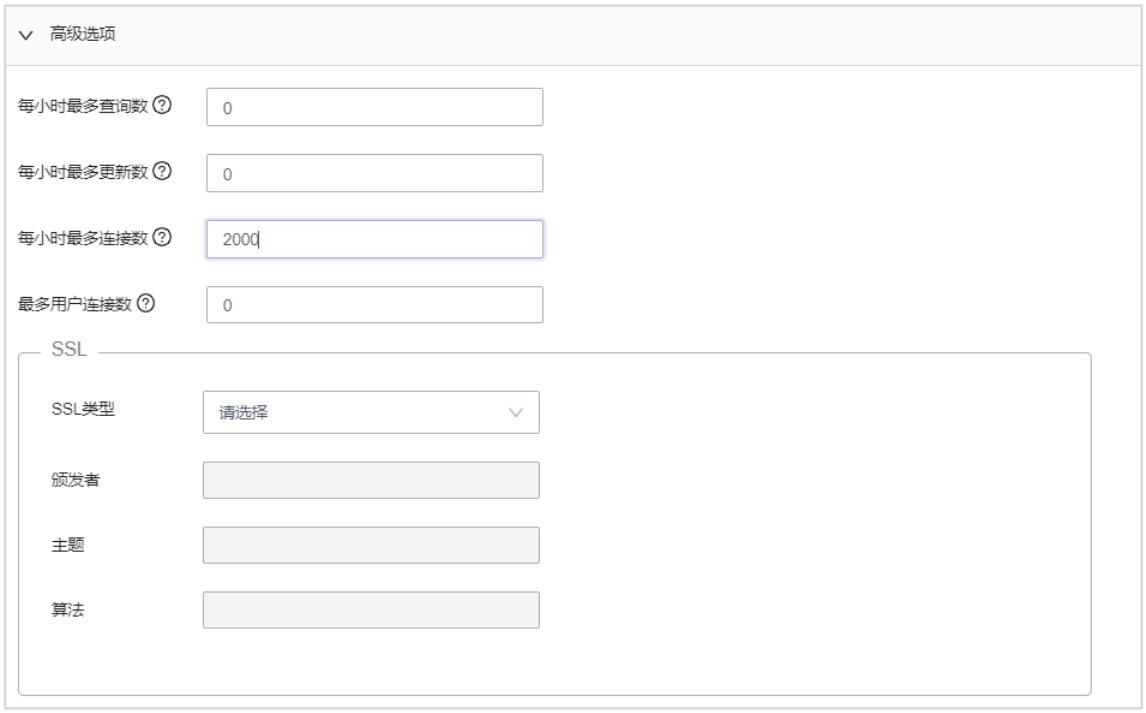
步骤 7 执行结果如下



步骤 8 除了使用 SQL 语句管理用户外，可以使用 DAS 上的用户管理进行可视化用户管理。登录 DAS 标准版。在“其他操作”中，选择“用户管理”。

步骤 9 可以在界面中看到上面创建的用户 hrapp，可以通过“编辑”，对创建的用户进行修改。

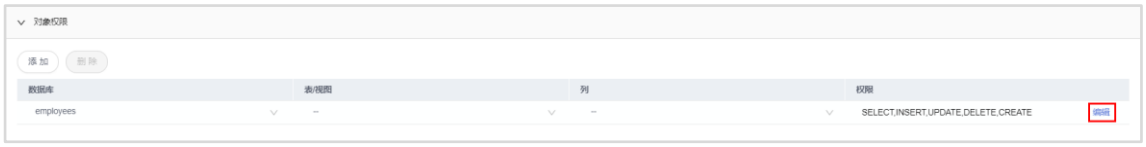
步骤 10 通过“高级选项”，可以对用户进行一些资源限制，这里可以尝试将“每小时最多连接数”设置为 2000。



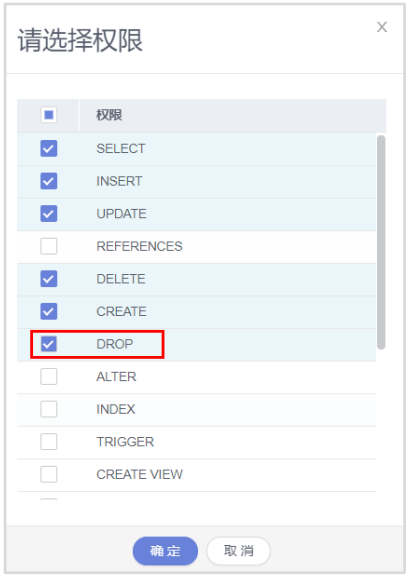
这个设置也可以通过以下语句设置：

```
ALTER USER 'hrapp'@'%' WITH MAX_CONNECTIONS_PER_HOUR 2000;
```

步骤 11 在“对象权限”中选择后侧的“编辑”，此处是针对整个 employee 数据库的权限设置，也属于对象权限，而全局权限是针对整个实例。



步骤 12 将“drop”权限勾选上，然后点击确定。



步骤 13 点击“保存”按钮后，会弹出需要修改的 SQL 预览，确认无误后点击“确定”。

步骤 14 返回“用户管理”界面，点击“对象权限”下的“1”，可以查看用户拥有的对象权限。

步骤 15 可以查看授权完成后的 hrapp 用户具有的权限。

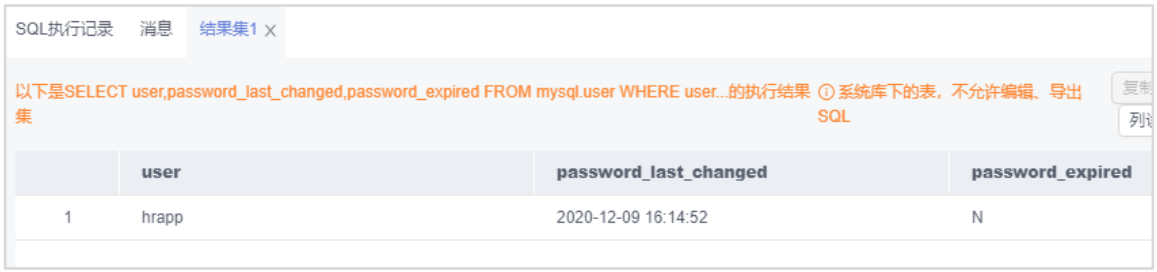
步骤 16 使用 alter user 命令，对 hrapp 用户的密码过期时间进行修改，上一小节将密码过期时间调整为 90 天，这里将 hrapp 用户的调整为 30 天。

```
ALTER USER 'hrapp'@'%' PASSWORD EXPIRE INTERVAL 30 DAY;
```

步骤 17 通过对 mysql.user 的表进行查询，可以得到用户密码修改的时间和是否过期。

```
SELECT user,password_last_changed,password_expired FROM mysql.user WHERE user='hrapp';
```

步骤 18 执行结果如下，用户的密码会在最后一次修改密码的时间后的 30 天过期。



步骤 19 可以使用 alter user 命令，将该用户密码直接至为过期。

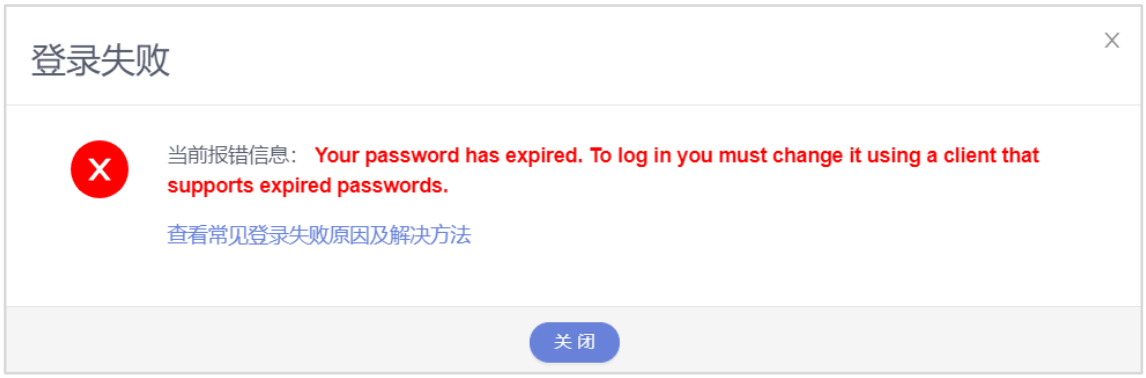
```
ALTER USER 'hrapp'@'%' PASSWORD EXPIRE;
```

步骤 20 退出 DAS，使用密码过期的用户登陆 DAS 会有以下报错。

点击右上角，切换连接。

输入用户 hrapp，密码 Huawei@qer321。

点击登录，效果如下：



步骤 21 可以使用 alter user 命令，禁止用户密码过期。注：当用户密码已过期后，无法通过该命令将用户的密码变为未过期，对于已过期的用户，只能对用户进行密码修改。

```
ALTER USER 'hrapp'@'%' PASSWORD EXPIRE NEVER;
```

7.2.4 实验小结

本小节实验，通过华为云平台的导入功能，将薪酬系统数据库的对象和数据导入到GaussDB(for MySQL)中；GaussDB(for MySQL)中的参数大多是平台进行优化过的，如buffer pool 的大小、刷新线程的个数等，本小节实验中的参数为可以进一步进行优化的参数，可以参考实验中的修改值；一般在生产环境中，每个系统都对应一个或几个数据库用户，因此用户管理对于生产系统也比较重要。

7.3 SQL 进阶与优化

7.3.1 针对全表扫描的 SQL 优化

步骤 1 HR 想查看 employees 系统中，1990 年至 2000 年间入职人员薪资低于 80000 的人员名单，针对这些人员进行调薪。

因为薪资 salaries 表中，存储着每位员工各个时期的薪资情况，因此需要通过最大的日期，将每个员工最新的薪资情况进行查询，并同 employees 表进行关联，将入职时间在 1990 到 2000 年间的员工查询出来。

```
SELECT
    e.first_name,
    e.last_name,
    s.salary
FROM
    employees e,
    salaries s
WHERE
    e.hire_date BETWEEN '1999-01-01'
    AND '2001-01-01'
    AND s.salary < 80000
```

```
    and s.to_date='9999-01-01'
    AND s.emp_no = e.emp_no
GROUP BY
    e.first_name,
    e.last_name
ORDER BY
    e.first_name;
```

步骤 2 在 GaussDB(for MySQL)的数据库列表中，点击对应数据库 gauss-demo。

步骤 3 在左侧选择“日志管理”，右侧点击“慢日志”，然后根据时间进行过滤。

步骤 4 找到执行的慢日志

执行语句	语句类型	发生时间	执行时间(s)	等待锁时间(s)	结果行数	扫描行数	所属数据库	账号	IP地址
SELECT e....	SELECT	2020/12/09 ...	0.130123 s	0.000181	50	306043		root	100.*.*.127

虽然该 SQL 执行时间为 0.13 秒，但是在慢日志中还是将其捕获，因为上一小节中我们将 log_queries_not_using_indexes 参数值设置为 on，也就是在执行 SQL 时未使用索引的 SQL 会被记录到慢日志中。执行结果为返回 50 条数据，扫描 306043 行数据。

步骤 5 使用 DAS 或 explain 命令获取 SQL 的执行计划

使用 DAS 标准版查看执行计划

或者使用 explain 命令查看执行计划

```
explain SELECT
    e.first_name,
```

```
    e.last_name,
    MAX( s.salary )
FROM
    employees e,
    salaries s
WHERE
    e.hire_date BETWEEN '1999-01-01'
    AND '2001-01-01'
    AND s.salary < 80000
    AND s.emp_no = e.emp_no
GROUP BY
    e.first_name,
    e.last_name
ORDER BY
    e.first_name;
```

得到的结果：

	id	select_type	table	partitions	type
1	1	SIMPLE	e		ALL
2	1	SIMPLE	s		ref

	key_len	ref	rows	filtered	Extra
			299246	11.11	Using where; Using temporary; Using filesort
	4	demo.e.emp_no	9	3.33	Using where

步骤 6 分析执行计划。

从执行计划中，可以看到表 s（也就是 salaries 表）是通过主键扫描了 9 行数据，而 e 表（也就是 employees 表）是全表扫描，扫描了 299246 行数据。该语句主要消耗点在于对 299246 行数据进行的扫描。

步骤 7 通过对 hire_date 列上创建索引，降低扫描的行数。

```
create index idx_hire_date on employees(hire_date);
```

步骤 8 再次查看执行计划

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows
1	1	SIMPLE	e		range	PRIMARY,idx_hire_date	idx_hire_date	3		1527
2	1	SIMPLE	s		ref	PRIMARY	PRIMARY	4	demo.e.emp_no	9

rows	filtered	Extra
1527	100.0	Using index condition; Using temporary; Using filesort
9	3.33	Using where

如上，可以看到在对 employees 表进行扫描时，只有 1527 行数据被扫描。

步骤 9 对比执行时间。

在 DAS 再次执行 SQL 语句，对比执行 SQL 的时间，未创建索引前用时 0.130 秒，创建索引后，用时 0.025 秒。

```
---------------开始执行---------------

【拆分SQL完成】：将执行SQL语句数量：（1条）

【执行SQL：(1)】
SELECT e.first_name, e.last_name, s.salary
FROM employees e, salaries s
WHERE e.hire_date BETWEEN '1999-01-01' AND '2001-01-01'
        AND s.salary < 80000
        AND s.to_date = '9999-01-01'
        AND s.emp_no = e.emp_no
GROUP BY e.first_name, e.last_name
ORDER BY e.first_name
执行成功，当前返回：[50]行，耗时：[25ms.]
```

7.3.2 窗口函数的使用与 SQL 优化

步骤 1 HR 想查看某个员工每年工资的变化情况。

此处可以使用窗口函数，将本年度的薪资同上一年度的薪资进行相减，就能得出这一年里，员工的薪资情况；在结合 employees 表中，通过 first_name 和 last_name 将需要查询的员工查找出来。

```
SELECT
   e.first_name,
    e.last_name,
   s.from_date,
    s.to_date,
    s.salary,
    LEAD ( s.salary ) OVER ( ORDER BY s.from_date ) - s.salary AS 'diff'
FROM
    salaries s,
   employees e
where e.emp_no=s.emp_no
```

and e.first_name='Eckart' and e.last_name='Axelband';

查询结果如下：

	first_name	last_name	from_date	to_date	salary	diff
1	Eckart	Axelband	1987-07-23	1988-07-22	42981	1444
2	Eckart	Axelband	1988-07-22	1989-07-22	44425	833
3	Eckart	Axelband	1989-07-22	1990-07-22	45258	1314
4	Eckart	Axelband	1990-07-22	1991-07-22	46572	2899
5	Eckart	Axelband	1991-07-22	1992-07-21	49471	4205

步骤 2 从慢日志中获取到该条 SQL 扫描行数过多。

执行语句	语句类型	发生时间	执行时间(s)	等待锁时间(s)	结果行数	扫描行数	所属数据库	账号	IP地址
SELECT action...	SELECT	2020/12/06 22:0...	0.000710 s	0.000296	0	43		root	115.*.*.169
SELECT action...	SELECT	2020/12/06 22:0...	0.000654 s	0.000279	0	43		root	115.*.*.169
SELECT e.first...	SELECT	2020/12/06 22:0...	0.078269 s	0.000238	16	300056		root	100.*.*.128

步骤 3 获取执行计划。

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows
1	1	SIMPLE	e		ALL	PRIMARY				299246
2	1	SIMPLE	s		ref	PRIMARY	PRIMARY	4	demo.e.emp_no	9

rows	filtered	Extra
299246	1.0	Using where; Using temporary; Using filesort
9	100.0	

步骤 4 分析执行计划

在 salaries 表中，根据主键索引过滤了 9 条数据，而 employees 表扫描了 299246 行数据，才找到对应的值，主要时间消耗在这么多条数据的扫描上，优化的方式为减少数据的行数的扫描。根据条件 emp_no 为主键，并且同 salaris 中的 emp_no 进行关联查询，不需要优化，但是另一个条件即 first_name 和 last_name 造成的扫描大量数据，尝试在这两个过滤条件上创建索引。

步骤 5 在 first_name 和 last_name 上创建索引。

```
create index id_name on employees(first_name,last_name);
```

步骤 6 再次生成执行计划

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows
1	1	SIMPLE	e		ref	PRIMARY,id_name	id_name	124	const,const	1
2	1	SIMPLE	s		ref	PRIMARY	PRIMARY	4	demo.e.emp_no	9

rows	filtered	Extra
1	100.0	Using index; Using temporary; Using filesort
9	100.0	

从执行计划中可以看到，直接通过索引命中记录，只扫描了一行数据。对于这种选择性较好的列（同名同姓较少），创建对应的索引能减少很多数据行的扫描。

步骤 7 执行时间上进行对比。

在 DAS 中再次执行该 SQL，速度从原来的 78 毫秒提高到了 5 毫秒。

```
【拆分SQL完成】：将执行SQL语句数量：（1条）

【执行SQL：(1)】
SELECT
  e.first_name,
        e.last_name,
  s.from_date,
        s.to_date,
        s.salary,
        LEAD ( s.salary ) OVER ( ORDER BY s.from_date ) - s.salary AS 'diff'
FROM
        salaries s,
  employees     e
where e.emp_no=s.emp_no
and e.first_name='Eckart' and e.last_name='Axelband'
执行成功，当前返回：[16]行，耗时：[5ms.]
```

7.3.3 通过分区改造进行查询优化

步骤 1 公司管理者想查看公司某时间段中，各部门的薪资状况。

该 SQL 需要将每个部门的所有人员的薪资进行求和，因此使用 sum 函数进行累加，并通过部门表和员工部门表进行关联，将每个部门的员工进行累加；使用 salaries 表中的薪资时间，进行时间段的查找。

```
SELECT
    b.dept_name,
    sum( a.salary ) AS de_sal
FROM
    salaries a,
    departments b,
    dept_emp e
WHERE
    a.from_date BETWEEN '1990-01-01'
    AND '1991-01-01'
    AND e.dept_no = b.dept_no
    AND a.emp_no = e.emp_no
GROUP BY
```

```
    b.dept_name
ORDER BY
    de_sal;
```

步骤 2 从慢日志中，可以看到该条 SQL 执行了 3.1 秒，扫描了 3473716 行数据。

步骤 3 获取执行计划

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows
1	1	SIMPLE	b		index	PRIMARY,dept_name	dept_name	162		9
2	1	SIMPLE	e		ref	PRIMARY,dept_no	dept_no	16	demo.b.dept_no	41392
3	1	SIMPLE	a		ref	PRIMARY	PRIMARY	4	demo.e.emp_no	9

rows	filtered	Extra
9	100.0	Using index; Using temporary; Using filesort
41392	100.0	Using index
9	11.11	Using where

步骤 4 分析执行计划

虽然从执行计划中看，都命中了索引，但是对于根据条件，扫描的结果还是不少。因为根据条件筛选出这么多记录，在同 salaries 表进行关联后，效率会慢。查看 salaries 表，数据量较大，并且其中有时间排序，再观察 SQL，其中按照 from_date 作为条件进行过滤，因此可以考虑将 salaries 表中，按照 from_date 字段进行分区表改造，这样按照年份，可以将每年的数据落在一个分区。

步骤 5 对表进行分区表改造

```
ALTER TABLE salaries
partition by range (to_days(from_date))
(
    partition p01 values less than (to_days('1985-01-01')),
    partition p02 values less than (to_days('1986-01-01')),
    partition p03 values less than (to_days('1987-01-01')),
    partition p04 values less than (to_days('1988-01-01')),
    partition p05 values less than (to_days('1989-01-01')),
    partition p06 values less than (to_days('1990-01-01')),
    partition p07 values less than (to_days('1991-01-01')),
    partition p08 values less than (to_days('1992-01-01')),
    partition p09 values less than (to_days('1993-01-01')),
```

```
    partition p10 values less than (to_days('1994-01-01')),
    partition p11 values less than (to_days('1995-01-01')),
    partition p12 values less than (to_days('1996-01-01')),
    partition p13 values less than (to_days('1997-01-01')),
    partition p14 values less than (to_days('1998-01-01')),
    partition p15 values less than (to_days('1999-01-01')),
    partition p16 values less than (to_days('2000-01-01')),
    partition p17 values less than (to_days('2001-01-01')),
    partition p18 values less than (to_days('2001-02-01')),
    partition p19 values less than (to_days('2001-03-01')),
    partition p20 values less than (to_days('2001-04-01')),
    partition p21 values less than (to_days('2001-05-01')),
    partition p22 values less than (to_days('2001-06-01')),
    partition p23 values less than (to_days('2001-07-01')),
    partition p24 values less than (to_days('2001-08-01')),
    partition p25 values less than (to_days('2001-09-01')),
    partition p26 values less than (to_days('2001-10-01')),
    partition p27 values less than (to_days('2001-11-01')),
    partition p28 values less than (to_days('2001-12-01')),
    partition p29 values less than (to_days('2002-01-01')),
    partition p30 values less than (to_days('2002-02-01')),
    partition p31 values less than (to_days('2002-03-01')),
    partition p32 values less than (to_days('2002-04-01')),
    partition p33 values less than (to_days('2002-05-01')),
    partition p34 values less than (to_days('2002-06-01')),
    partition p35 values less than (to_days('2002-07-01')),
    partition p36 values less than (to_days('2002-08-01')),
    partition p37 values less than (to_days('2002-09-01')),
    partition p38 values less than (to_days('2002-10-01')),
    partition p39 values less than (to_days('2002-11-01')),
    partition p40 values less than (to_days('2002-12-01')),
    partition pmax values less than (to_days('3000-01-01'))
);
```

将 salaries 表的 from_date 字段进行分区，按照年为单位进行 range 分区。将最大的分区设置为 pmax。

步骤 6 再次执行原查询 SQL 语句，然后查看慢 SQL 日志，发现无该 SQL 记录，说明该 SQL 未成为慢 SQL。

步骤 7 查看分区改造后，该 SQL 的执行计划。

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows
1	1	SIMPLE	a	p01,p07,p08	ALL	PRIMARY				247510
2	1	SIMPLE	e		ref	PRIMARY,dept_no	PRIMARY	4	employees.a.emp_no	1
3	1	SIMPLE	b		eq_ref	PRIMARY,dept_name	PRIMARY	16	employees.e.dept_no	1

filtered	Extra
11.11	Using where; Using temporary; Using filesort
100.0	Using index
100.0	

通过查看执行计划，跟原先的执行计划发生了变化，虽然是执行了全表扫描，实际上是对全分区进行了扫描，只扫描了 p01，p07 和 p08 三个分区。

步骤 8 在 DAS 中再次执行 SQL，可以对比 SQL 的执行时间。从改造前的 1.6 秒减少到 0.8 秒。

SQL执行记录 消息 结果集1 ×

```
【拆分SQL完成】：将执行SQL语句数量：（1条）

【执行SQL：(1)】
SELECT
        b.dept_name,
        sum( a.salary ) AS de_sal
FROM
        salaries a,
        departments b,
        dept_emp e
WHERE
        a.from_date BETWEEN '1990-01-01'
        AND '1991-01-01'
        AND e.dept_no = b.dept_no
        AND a.emp_no = e.emp_no
GROUP BY
        b.dept_name
ORDER BY
        de_sal
执行成功，当前返回：[9]行，耗时：[832ms.]
```

步骤 9 从云 DBA 中的全量 SQL 洞察中，可以看到该条 SQL 执行的信息。

SQL语句	用户名	数据库	线程ID	客户端IP	操作类型	状态	执行...	执行...	更新行数	返回...	扫描...	锁等...
SELECT b.dept_name, SUM(a.salary) AS de_sal FROM salaries a, departments b, dept_emp e WHERE a.from_date BETWEEN '1990-01-01' AND '1991-01-01' AND e.dept_no = b.dept_no AND a.emp_no = e.emp_no GROUP BY b.dept_name ORDER BY de_sal	root	employees	70813	*	select	成功	2020/12/08 ...	839.00	0	9	500747	0.00

从上图中可以看到，执行时间变为 800ms，扫描行数减少为 50W 行，而且是顺序读，相对速度提高不少，当然该条 SQL 还有继续优化的空间。

步骤 10 业务部门想将该 SQL 长期进行查询统计，因此需要将该条 SQL 改造成存储过程，方便进行调用使用。

```
DELIMITER $$
CREATE procedure proc_dept_sal(in begin_time varchar(20),in end_time varchar(20))
begin

    SELECT
        b.dept_name,
        sum( a.salary ) AS de_sal
    FROM
        salaries a,
        departments b,
        dept_emp e
    WHERE
        a.from_date BETWEEN begin_time
        AND end_time
        AND e.dept_no = b.dept_no
        AND a.emp_no = e.emp_no
    GROUP BY
        b.dept_name
    ORDER BY
        de_sal;
end;
$$
DELIMITER ;
```

步骤 11 使用 call 调用该存储过程。

```
call proc_dept_sal('1990-01-01','1993-01-01');
```

	dept_name	de_sal
1	Human Resources	1192963146
2	Quality Management	1391234971
3	Finance	1533966048
4	Research	1535772412
5	Customer Service	1680026450
6	Marketing	1777082152
7	Sales	5260026547
8	Production	5359599704
9	Development	6193777226

7.3.4 实验小结

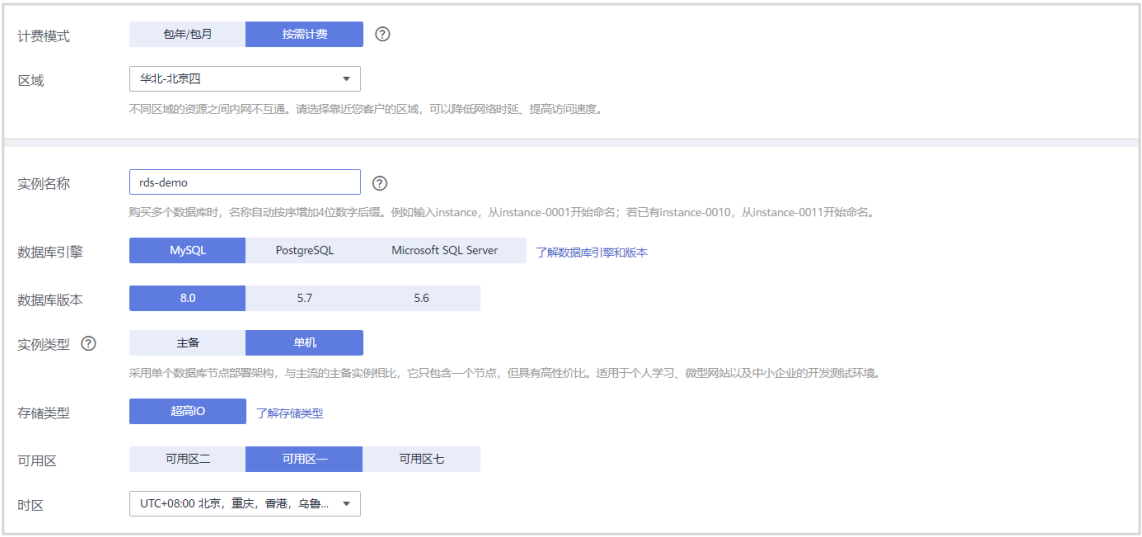
本小结实验，通过对 SQL 进行进阶查询，并且通过 SQL 优化，提高 SQL 在数据库中执行的效率。

7.4 DRS 与备份恢复

7.4.1 DRS 的使用

步骤 1 搭建测试 RDS for MySQL 环境。在华为云控制台界面，点击右侧的“服务列表”，在数据库中选择“云数据库 RDS”。

步骤 2 点击“购买数据库实例”，按照按需计费进行购买。

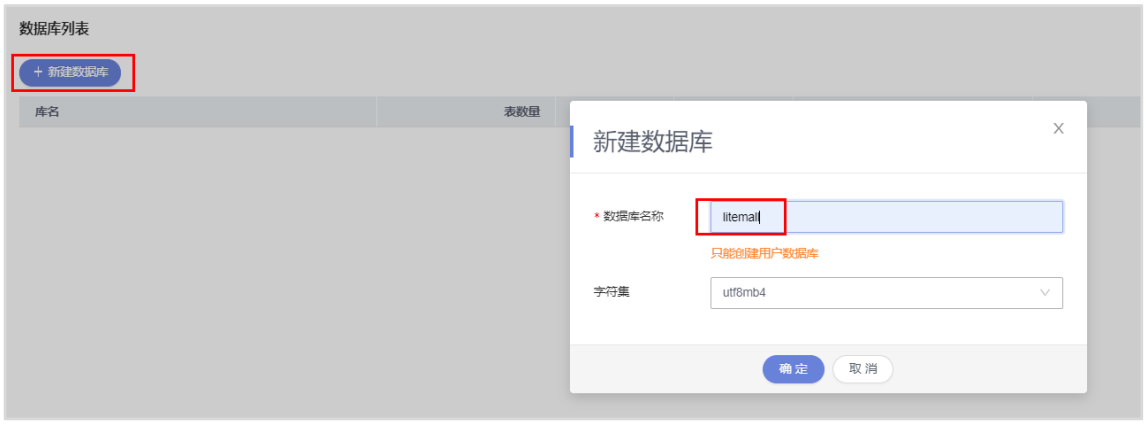


步骤 3 按照如下的规格进行购买。

步骤 4 选用和 GaussDB(for MySQL)一致的 VPC 和安全组，并对 root 用户设定密码和使用默认的数据库参数模板。

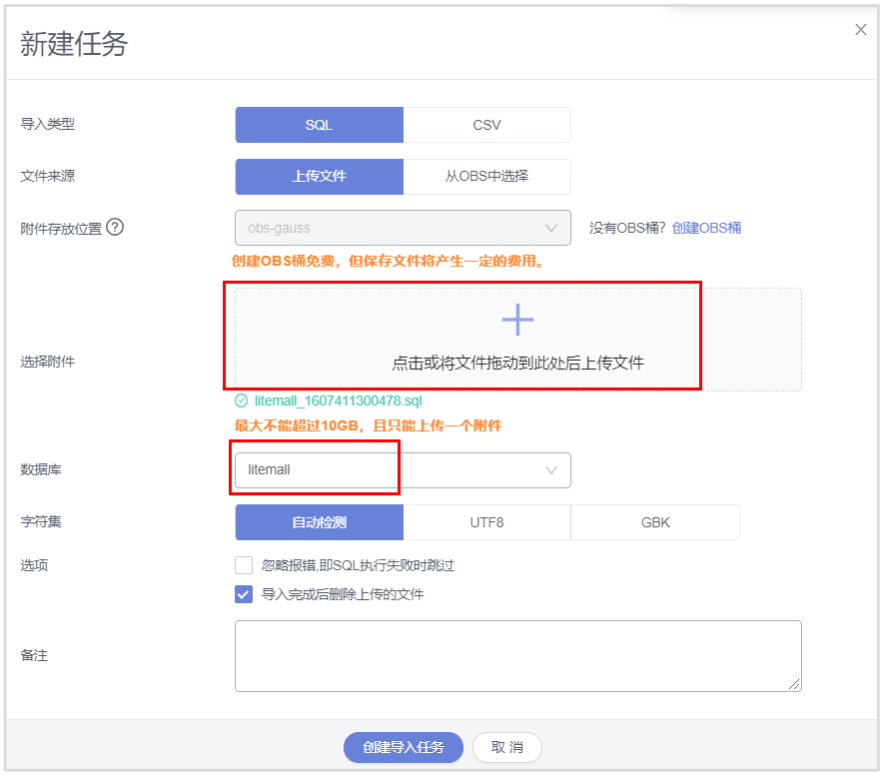
步骤 5 使用 DAS 登录 RDS for MySQL 中。

步骤 6 创建 litemall 数据库，点击“新建数据库”，数据库名称设置为 litemall。



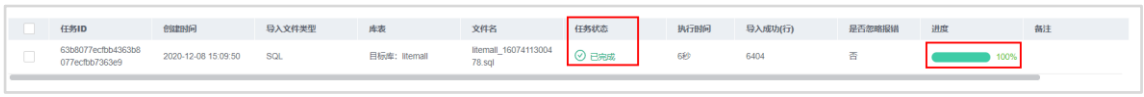
步骤 7 在“导入 导出”中选择导入。

步骤 8 将 litemall.sql 添加到“选择附件”处，数据库选择 litemall，创建导入任务。



步骤 9 确认无误后，点击“确定”。

步骤 10 当导入任务状态变为“已完成”时，并且进度达到 100%，表示导入完成。



步骤 11 进入“SQL 查询”，查看 litemall_goods 表的记录数，完成 RDS 环境的初始化。

select count(*) from litemall_goods;

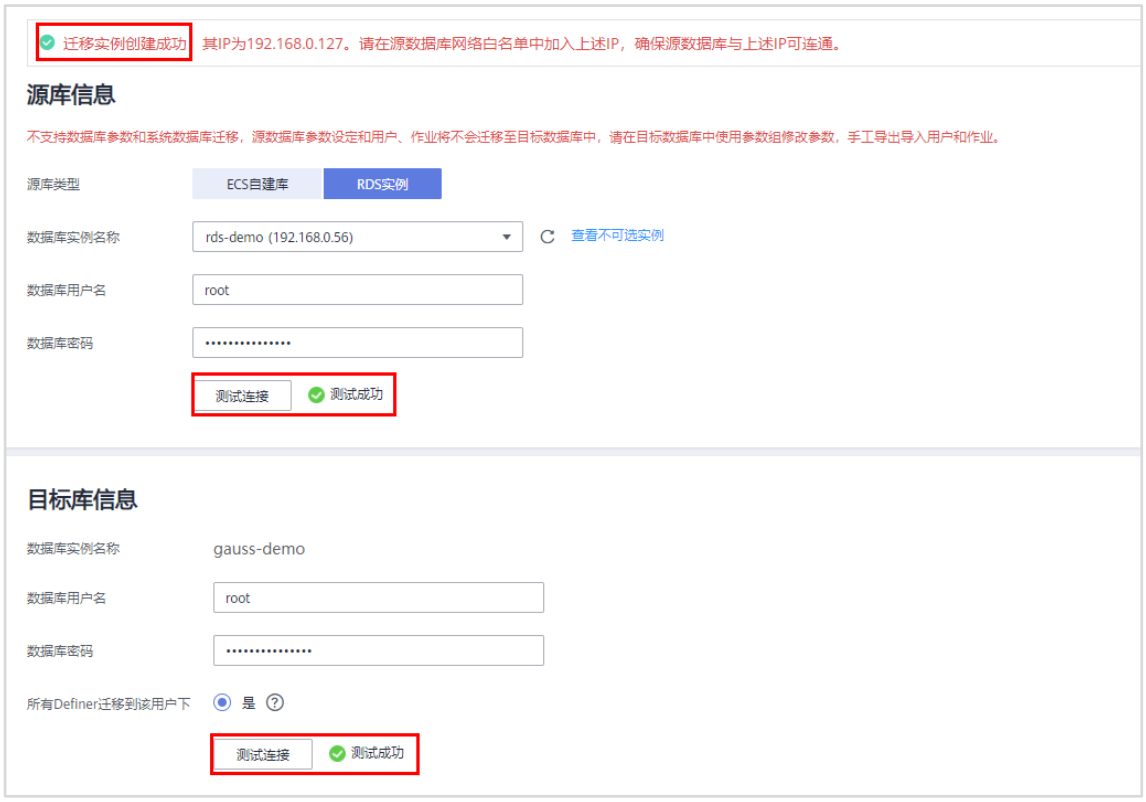
步骤 12 配置 DRS 环境。在华为云控制台界面，点击右侧的“服务列表”，在数据库中选择“数据复制服务 DRS”。

步骤 13 左侧选择“实时迁移管理”后，点击“创建迁移任务”。

步骤 14 场景选择中，源端与目标端都选择“本云云数据库”，进入下一步。

步骤 15 配置迁移实例信息，选择“入云方向”；源数据库选择“MySQL”；目标数据库选择“GaussDB(for MySQL)主备版”；网络类型选择：VPC；目标数据库实例选择创建的GaussDB(for MySQL)的实例；迁移方式选择为“全量+增量”。设置完成后，点击“下一步”。

步骤 16 迁移实例创建成功后（实例创建需要一些时间，大概十几分钟），输入源数据库和目标库的用户名、密码，并进行“测试连接”，测试成功后，进入下一步。



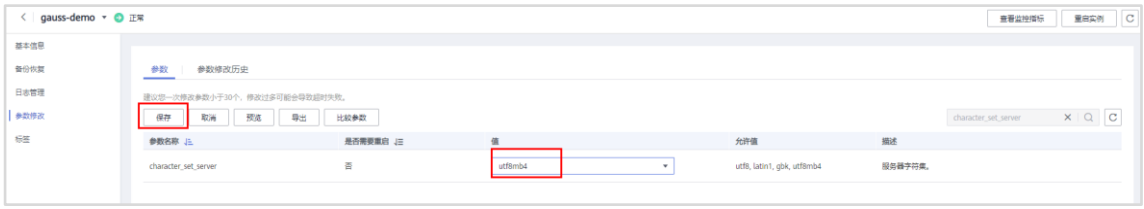
步骤 17 设置迁移对象，本实验中使用“全部迁移”，实际生产中，可以针对不同的需求进行选择。确认完成后，进入下一步。

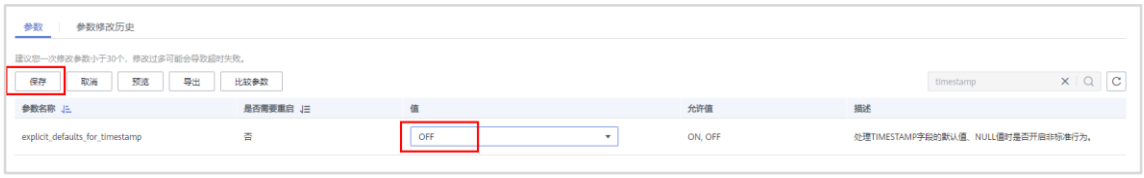
步骤 18 预检查，针对源数据库和目标数据库进行检查，将不一致的内容进行展示出来，展示出告警、失败和成功的内容。

步骤 19 处理告警的内容。点击“告警”的详情信息，在告警中，出现部分参数不一致的情况，可能会对数据库迁移产生影响，DRS 给出建议是将源库与目标库设置一致。

步骤 20 处理失败的内容。点击“失败”的详情信息，在详情中，出现参数不一致的情况，如果参数不一致会造成迁移失败，因此必须将将源库与目标库设置一致。点击“参考链接”，可以对目标端数据库进行修改。

步骤 21 跳转到目标库 GaussDB(for MySQL)的数据参数设置，将 character_set_server 取值修改为 utf8mb4，并点击“保存”。将 explicit_defaults_for_timestamp 修改为 off，并点击“保存”。





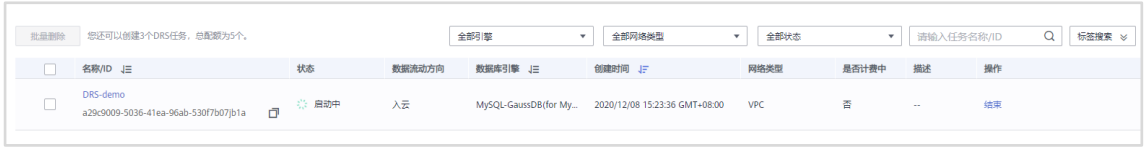
步骤 22 处理完成后，点击“重新校验”，再次检查。

步骤 23 校验成功后，进入下一步。

步骤 24 进入任务确认，针对任务中的信息进去确认，可以针对任务启动时间，进行设定；一般生产环境建议使用“稍后启动”，并将时间选择为业务低峰时；本实验因为并非生产环境，因此使用“立即启动”。任务确认完成后，点击“启动任务”。

步骤 25 点击“启动任务”后，会进行再次确认。阅读“迁移前须知”，确认无误后，勾选“我已阅读启动前须知”，点击“启动任务”。

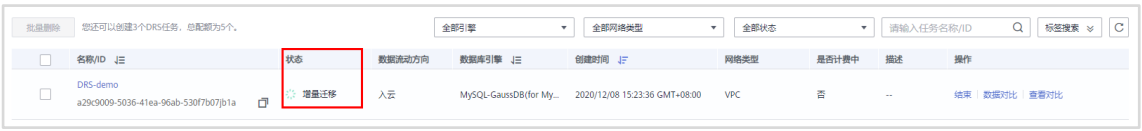
步骤 26 任务进入启动中，任务启动需要一些时间，大约需要十二分钟。



步骤 27 登录 GaussDB(for MySQL)的 DAS 中，可以看到数据库列表中，已经存在 litemall 数据库。进入 litemall 的“SQL 查询”。

步骤 28 对 litemall_goods 中的表记录数进行查询，查询结果同样为 239 条。

步骤 29 当 DRS 任务状态变为“增量迁移”时，进入下一步任务。



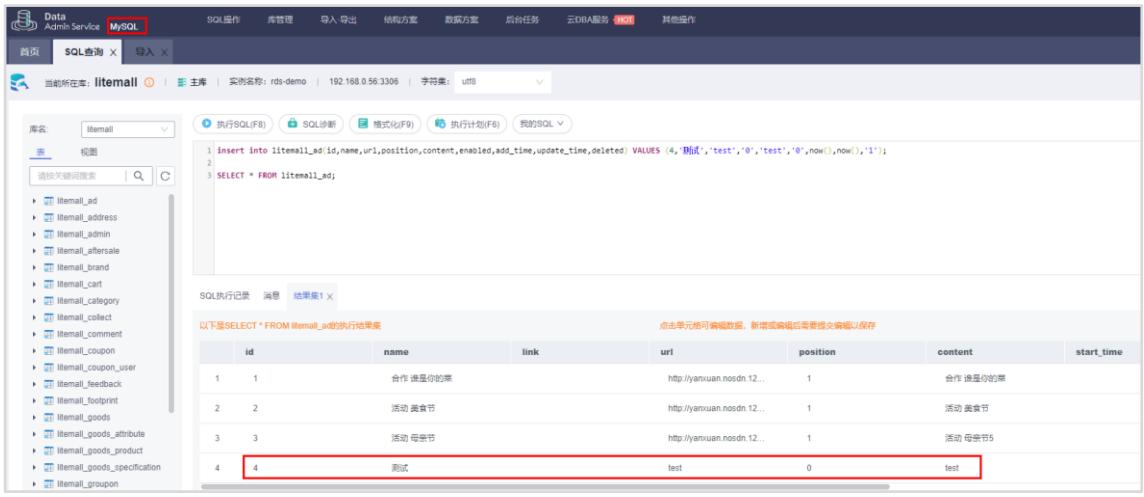
步骤 30 登录 RDS 数据库的 DAS 中，执行插入数据操作

```
insert into litemall_ad(id,name,url,position,content,enabled,add_time,update_time,deleted) VALUES (4,'测试','test','0','test','0',now(),now(),'1');
```

插入完成后进行查询

```
SELECT * FROM litemall_ad;
```

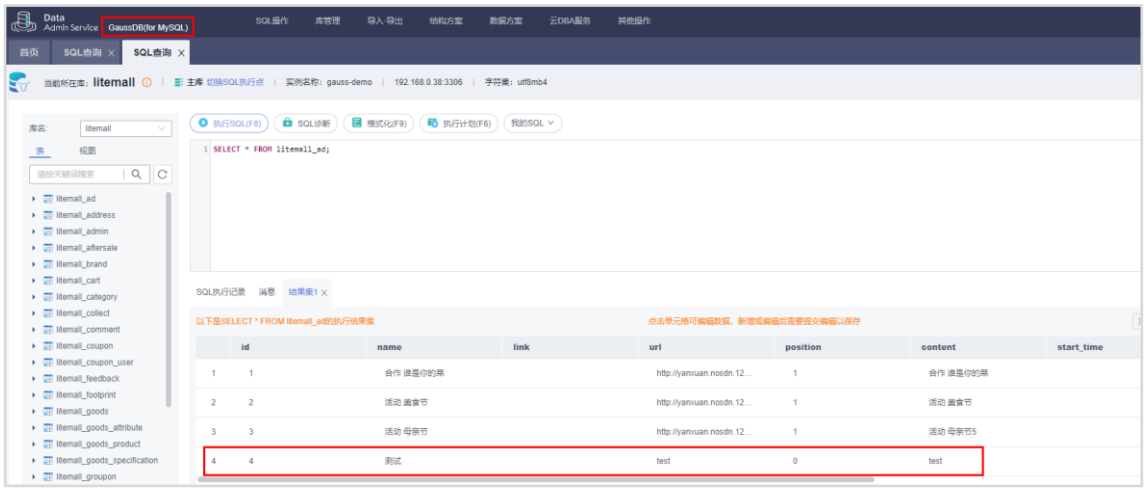
获得结果：



步骤 31 登录 GaussDB(for MySQL)数据库的 DAS 中，执行查询操作。

SELECT * FROM litemall_ad;

获得结果：



说明在进行“增量同步”任务时，实时将源库的数据同步到目标库中。

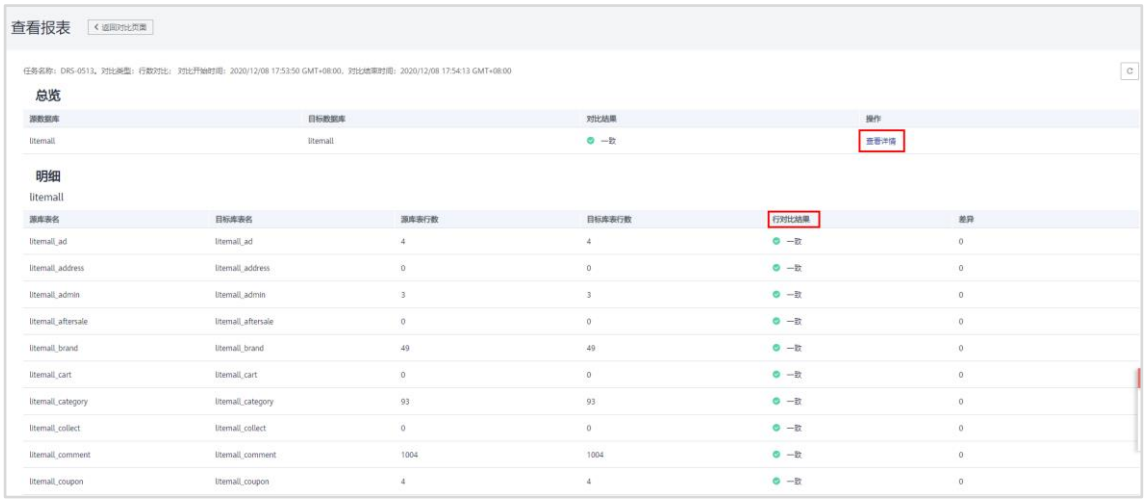
步骤 32 对当前的 DRS 任务创建数据对比任务。

步骤 33 点击“查看对比”，对创建的“数据对比”任务结果进行查看。

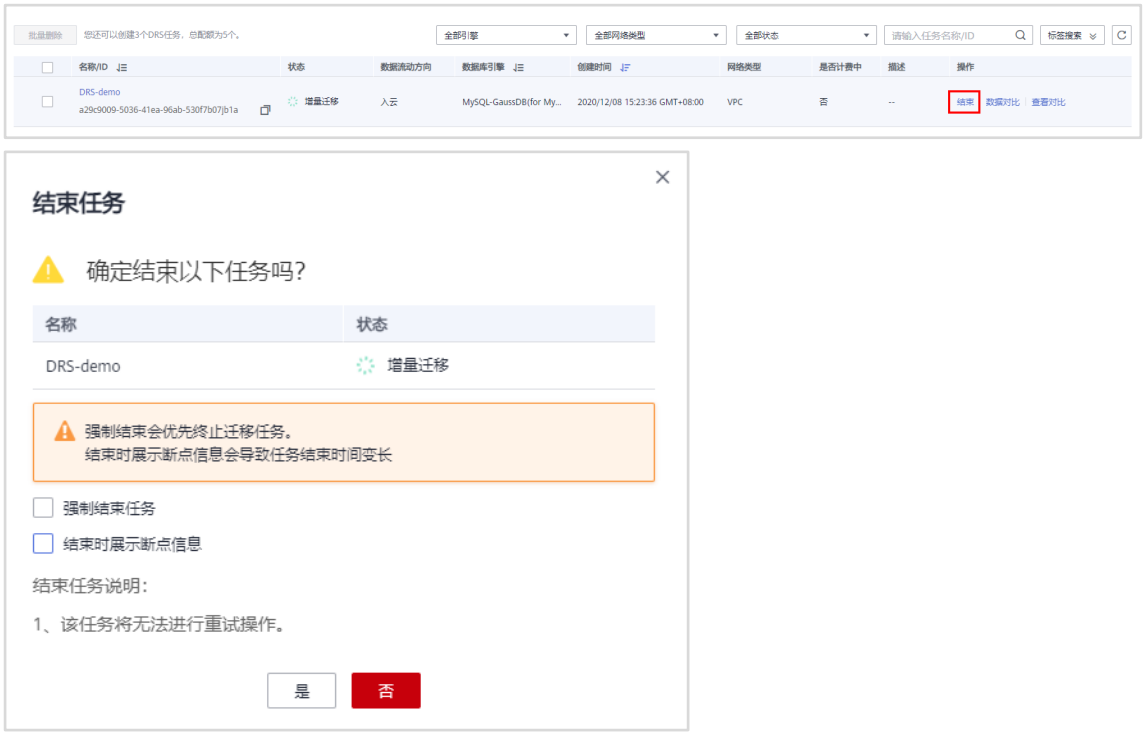
步骤 34 点击“迁移进度”，可以掌握当前迁移完成情况，从下图来看，当前进度已完成100%。

步骤 35 点击“迁移对比”中的“查看对比报表”，可以查看迁移完成后，源端与目标端数据的一致性情况。

步骤 36 点击“查看详情”，可以看到表级别的对比明细，从图中可以看到“行对比结果”都为一致的状态。



步骤 37 因为本实验数据量较小，在对比完成后，可以结束 DRS 的同步任务，建议不要进行强制结束。



7.4.2 完成数据库实例的备份与恢复

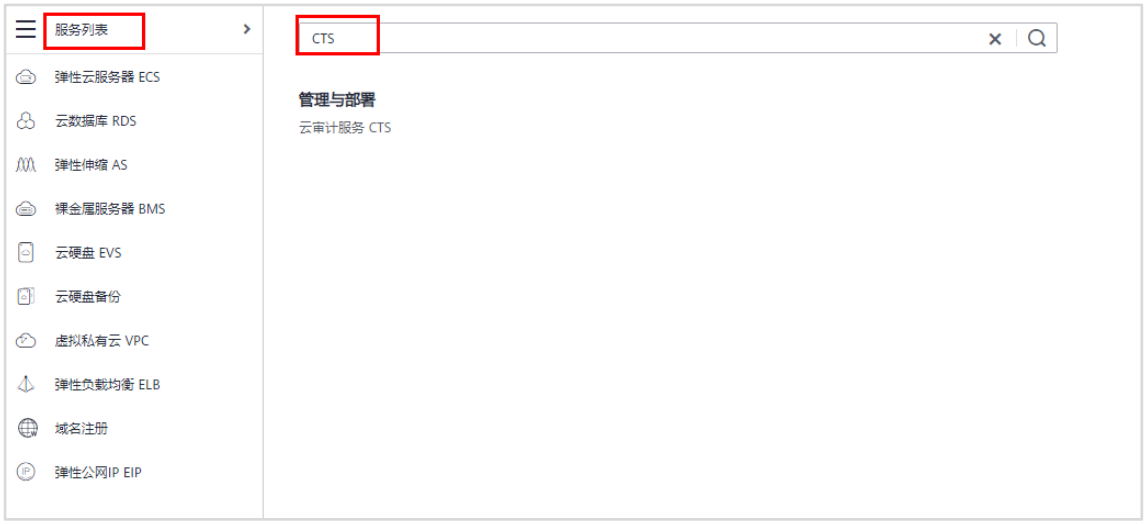
步骤 1 返回 GaussDB(for MySQL)实例列表界面，在“服务列表”中选择数据库的“云数据库 GaussDB”。

步骤 2 对迁移完成的数据库进行数据库备份。

步骤 3 查看备份情况，备份状态为已完成状态。

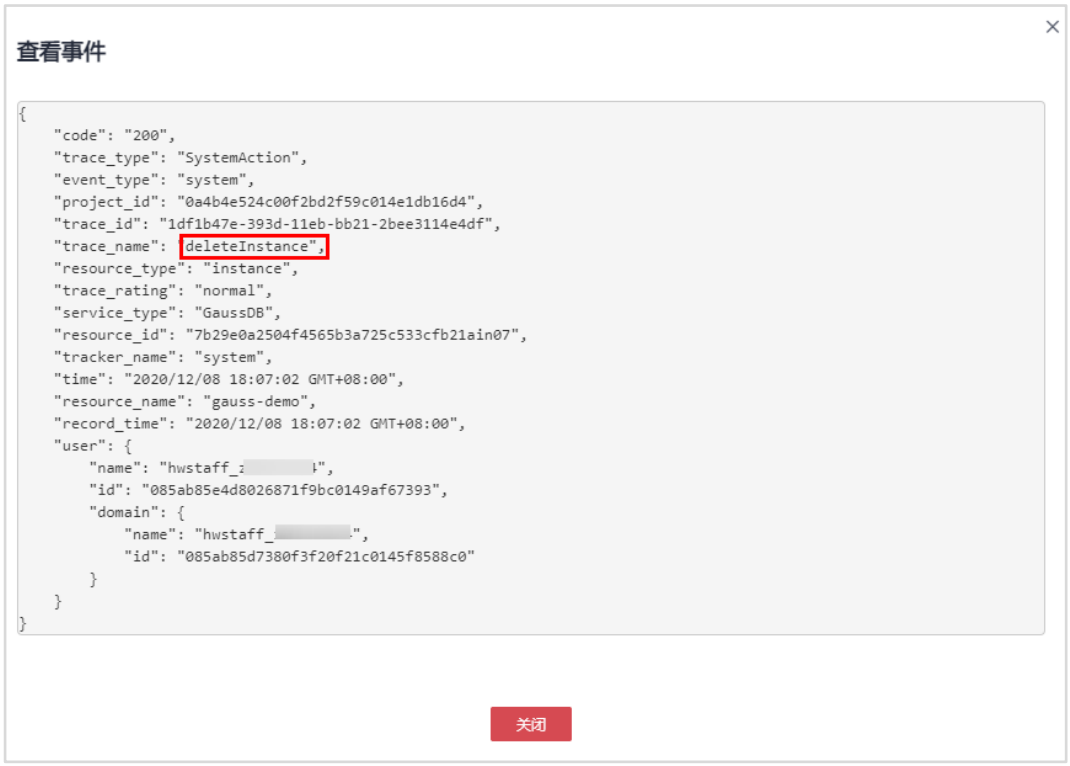
步骤 4 模拟从删库到跑路的场景，将数据库实例删除。（注意：此为实验环境，生产环境千万不要进行尝试）

步骤 5 当数据库实例删除完成后，进行问题回溯，通过云审计服务(CTS)进行问题回溯。在控制台界面右侧的“服务列表”中，检索 CTS 服务，并选择进入云审计服务。



步骤 6 在事件列表中，可以看到某用户对数据库实例进行了删除操作。

步骤 7 在“事件详情”中，可以获取更详细的事件信息。



步骤 8 返回 GaussDB(for MySQL)服务列表，在左侧的“备份管理”中，可以看到在迁移完成后，对数据库实例进行的备份还存在。

步骤 9 使用该备份对被删除的数据库实例进行恢复，点击“恢复”按钮。

步骤 10 使用备份恢复到新的实例中，对数据库实例进行“计费模式”和“实例名称”的设定。

步骤 11 对购买实例的规格和只读节点的个数进行选择。

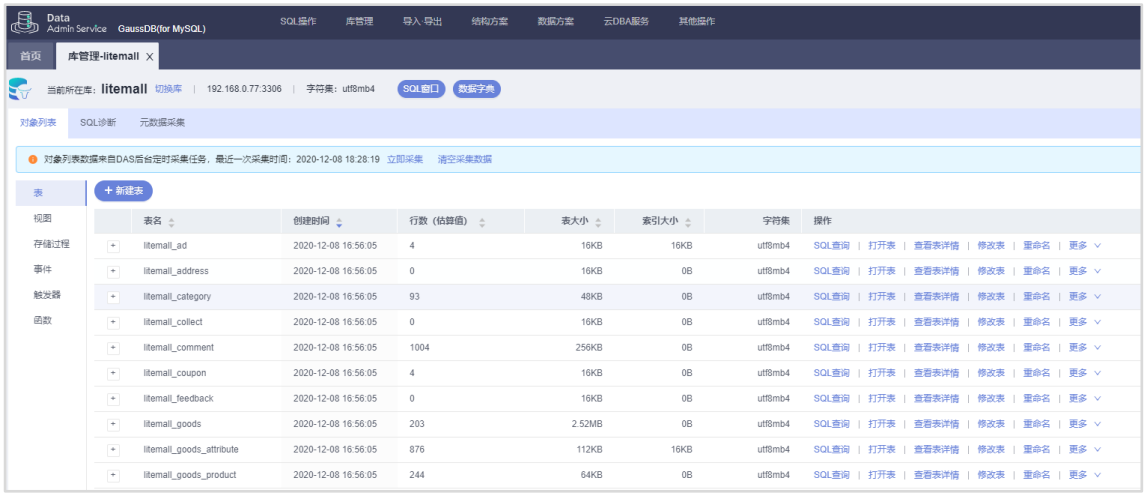
步骤 12 设置对应的 VPC、安全组和 root 密码，建议 VPC 和安全组按照被删除的实例进行设置，防止因为变化造成网络不通等问题。

步骤 13 确认规格无误后，进行提交。

步骤 14 几分钟后，当数据库实例状态为正常时，重新登录数据库。

步骤 15 实例中的数据库已恢复。

步骤 16 点击查看 litemall 数据库的“库管理”，可以数据库已恢复到迁移完成之后的时间点。



7.4.3 实验小结

本实验通过模拟数据复制服务(DRS)将 RDS for MySQL 中的 litemall 数据库迁移至 GaussDB(for MySQL)中。通过备份恢复实验，模拟数据库被删除后如何进行恢复和以及使用云审计服务(CTS)对删实例的行文进行追踪。

7.5 清除实验环境

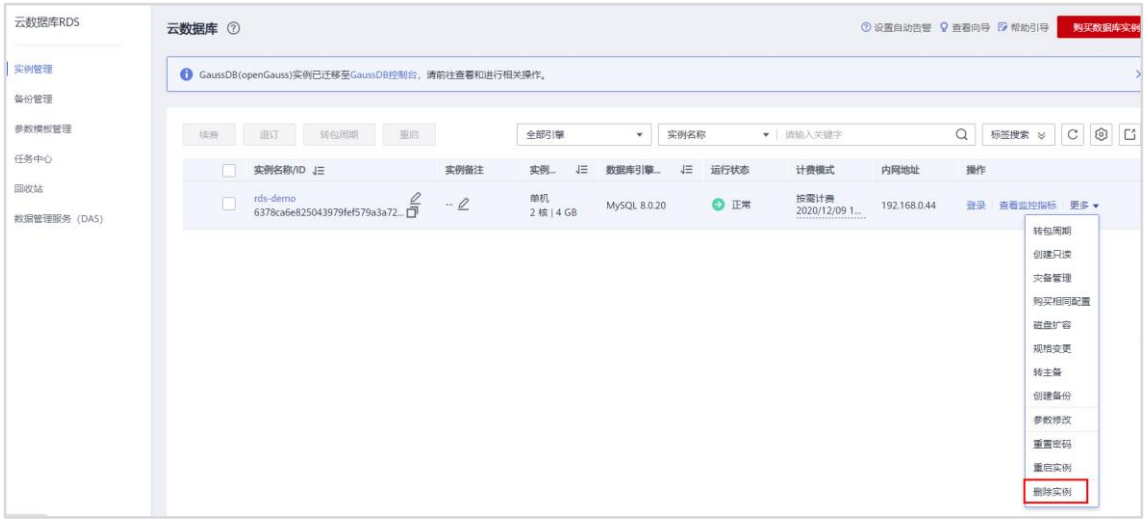
步骤 1 清除 GaussDB(for MySQL)资源。在华为云控制台界面中，在左侧的“服务列表”中选择“云数据库 GaussDB”。

步骤 2 在实例列表中，选择实验中购买的 GaussDB(for MySQL)数据库实例，点击“更多”中的“删除实例”，并进行确认。

步骤 3 在云数据库 GaussDB 的右侧，点击“备份管理”，将备份为手动的备份进行删除，并进行确认。

步骤 4 清除 RDS for MySQL 资源。在华为云控制台界面中，在左侧的“服务列表”中选择“云数据库 RDS”。

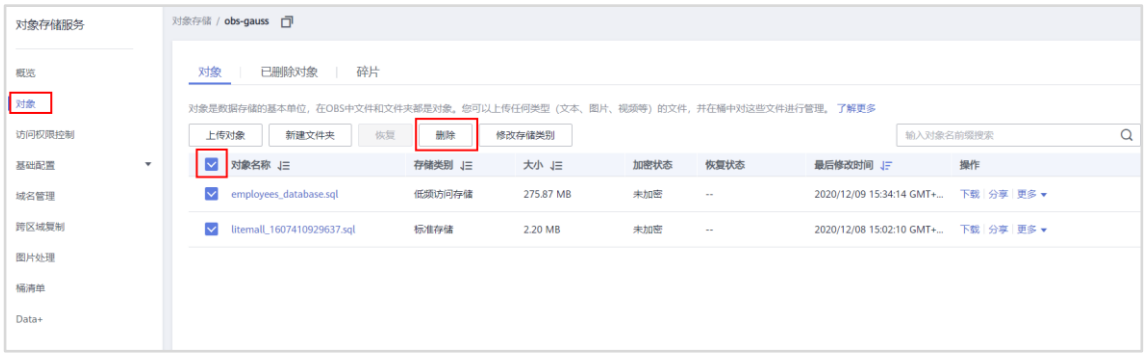
步骤 5 在实例列表中，选择实验中购买的 RDS for MySQL 数据库实例，点击“更多”中的“删除实例”，并进行确认。



步骤 6 在华为云控制台界面中，在左侧的“服务列表”中选择“对象存储服务 OBS”。

步骤 7 点击本实验开始创建的 obs 桶“obs-gauss”。

步骤 8 点击右侧的“对象”，将桶中对象进行全选后，点击删除按钮，并进行确认。



7.6 实验总结

本实验分别从数据库的开发与优化、数据库的数据加载、数据库迁移以及管理员的运维管理等方向，模拟生成环境中可能遇到的相关问题，希望通过本实验的练习，能够熟练掌握GaussDB(for MySQL)产品以及对应的云服务工具（DAS、DRS、CTS 等）。

8 资源释放实验

8.1 实验介绍

8.1.1 关于本实验

本实验通过在华为云上删除 GaussDB(for MySQL)实例，帮助学员和读者掌握在华为云上清除 GaussDB(for MySQL)资源。

8.1.2 实验目的

- 掌握 GaussDB(for MySQL)资源释放操作；
- 熟悉华为云操作。

8.2 实验任务配置

8.2.1 删除 GaussDB(for MySQL)

步骤 1 登录 GaussDB(for MySQL)控制台

点击“服务列表”>“数据库”>“云数据库 GaussDB”，进入 GaussDB(for MySQL)控制台。

控制台如下：

步骤 2　删除 GaussDB(for MySQL)数据库实例

选中需要删除的数据库实例，点击“更多”>“删除实例”。

在弹出窗口中，点击“是”，完成删除。

右上角会出现删除详情。

至此 GaussDB(for MySQL)资源释放实验完成。

8.3 实验总结

通过华为云操作，掌握了 GaussDB(for MySQL)的资源释放操作。