目录

主页

关于 MatrixOne

- MatrixOne 简介
- MatrixOne 功能列表
- MatrixOne 技术架构
- MySQL 兼容性
- 最新动态

快速开始

- 单机部署 MatrixOne
  - 单机部署 MatrixOne 概述
  - 在 macOS 上部署
    - 使用源代码部署
    - 使用二进制包部署
    - 使用 Docker 部署
  - 在 Linux 上部署
    - 使用源代码部署
    - 使用二进制包部署
    - 使用 Docker 部署
- SQL 的基本操作

开发指南

- 数据库模式设计
  - 概述
  - 创建数据库
  - 创建表
  - 数据完整性
    - 数据完整性约束概述
    - NOT NULL 完整性约束

UNIQUE KEY 完整性约束
PRIMARY KEY 完整性约束
FOREIGN KEY 完整性约束
连接 MatrixOne
配置 MatrixOne SSL 连接
客户端工具连接 MatrixOne 服务
Java 连接 MatrixOne 服务
使用 JDBC 连接器连接 MatrixOne
使用 Java ORMs 连接 MatrixOne
Python 连接 MatrixOne 服务
导入数据
单条导入
批量导入
批量导入概述
导入 csv 文件
导入 jsonlines 文件
从 S3 读取数据并导入至 MatrixOne
导出数据
使用 `SELECT INTO...OUTFILE` 导出数据
使用 `modump` 导出数据
数据读取
单表读取
多表连接查询
子查询
视图
公共表表达式
事务
事务通用概念

MatrixOne 的事务
MatrixOne 的事务概述
MatrixOne 的显式事务
MatrixOne 的隐式事务
MatrixOne 的乐观事务
如何使用 MatrixOne 事务
MatrixOne 的事务隔离级别
MVCC
MatrixOne 中的事务应用场景
部署指南
MatrixOne 分布式集群部署
教程
构建一个 Java CRUD 示例
使用 SpringBoot 和 Spring JPA 构建一个 CRUD 示例
使用 SpringBoot 和 MyBatis 构建一个 CRUD 示例
使用 Python 和 SQLAlchemy 构建一个 CRUD 示例
构建一个 Python CRUD 示例
运维
备份与恢复
数据挂载
挂载目录到 Docker 容器
数据迁移
将数据从 MySQL 迁移至 MatrixOne
测试
性能测试
SSB 测试
TPCH 测试
TPCC 测试

测试工具
MO-Tester
MO-Tester 规范要求
性能调优
SQL 性能调优方法概述
MatrixOne 执行计划
MatrixOne 执行计划概述
使用 EXPLAIN 理解执行计划
JOIN 查询的执行计划
子查询的执行计划
聚合查询的执行计划
视图的执行计划
性能调优最佳实践
使用 Cluster by 语句调优
权限
TLS 安全连接
什么是权限管理
关于 MatrixOne 权限管理
最佳实践
操作指南
快速开始
创建租户，验证资源隔离
新租户创建用户、创建角色和授权
权限管理操作
应用场景
参考手册
SQL 结构与语法
关键字

SQL 目录
数据定义语言（DDL）
CREATE DATABASE
CREATE INDEX
CREATE TABLE
CREATE EXTERNAL TABLE
CREATE VIEW
ALTER VIEW
DROP DATABASE
DROP INDEX
DROP TABLE
DROP VIEW
TRUNCATE TABLE
数据修改语言（DML）
INSERT
INSERT INTO SELECT
DELETE
UPDATE
LOAD DATA
INTERVAL
Information Functions
LAST_QUERY_ID()
LAST_INSERT_ID()
OPERATORS
运算符概述
运算符的优先级
算数运算符
算数运算符概述
%,MOD

- *
- +
- -
- -
- /
- DIV

赋值运算符

- 赋值运算符概述
- =

二进制运算符

- 二进制运算符概述
- &
- >>
- <<
- ^
- |
- ~

强制转换函数和运算符

- 强制转换函数和运算符概述
- CAST
- CONVERT

比较函数和运算符

- 比较函数和运算符概述
- >
- >=
- <
- <>,!=
- <=
- =

BETWEEN ... AND ...
IN
IS
IS NOT
IS NOT NULL
IS NULL
LIKE
NOT BETWEEN ... AND ...
NOT LIKE
COALESCE
控制流函数
控制流函数概述
CASE WHEN
IF
逻辑运算符
逻辑运算符概述
AND,&&
NOT,!
OR
XOR
数据查询语言（DQL）
SELECT
SUBQUERY
SUBQUERY 概述
Derived Tables
子查询与比较操作符的使用
SUBQUERY with ANY or SOME
SUBQUERY with ALL
SUBQUERY with EXISTS

SUBQUERY with IN
JOIN
JOIN 概述
INNER JOIN
LEFT JOIN
RIGHT JOIN
FULL JOIN
OUTER JOIN
NATURAL JOIN
With CTE
联合查询
联合查询概述
UNION
INTERSECT
MINUS
数据控制语言（DCL）
CREATE ACCOUNT
ALTER ACCOUNT
CREATE ROLE
CREATE USER
DROP ACCOUNT
DROP USER
DROP ROLE
GRANT
REVOKE
其他
SHOW
SHOW ACCOUNTS
SHOW DATABASES

SHOW CREATE TABLE
SHOW CREATE VIEW
SHOW TABLES
SHOW INDEX
SHOW COLLATION
SHOW COLUMNS
SHOW GRANT
SHOW VARIABLES
SET
SET ROLE
USE
USE DATABASE
Prepared
Prepared 概述
PREPARE
EXECUTE
DEALLOCATE
Explain
EXPLAIN
EXPLAIN Output Format
Explain Analyze
聚合函数
ANY_VALUE
AVG
BIT_AND
BIT_OR
BIT_XOR
COUNT
GROUP_CONCAT

MAX
MEDIAN
MIN
SLEEP
STDDEV_POP
SUM
系统变量
自定义变量
数据类型
数据类型概览
数据类型转换
TIMESTAMP 和 DATETIME 的自动初始化和更新
JSON 数据类型
BLOB 和 TEXT 数据类型
精确数值类型-DECIMAL
系统函数
数学类
ABS()
ACOS()
ATAN()
CEIL()
COS()
COT()
EXP()
FLOOR()
LN()
LOG()
PI()
POWER()

ROUND()
SIN()
SINH()
TAN()
UUID()
日期时间类
CURDATE()
CURRENT_TIMESTAMP()
DATE()
DATE_ADD()
DATE_FORMAT()
DATE_SUB()
DATEDIFF()
DAY()
DAYOFYEAR()
EXTRACT()
FROM_UNIXTIME
MONTH()
TIMEDIFF()
TIMESTAMP()
TO_DATE()
UNIX_TIMESTAMP
UTC_TIMESTAMP()
WEEKDAY()
YEAR()
字符串类
BIN()
BIT_LENGTH()
CHAR_LENGTH()

CONCAT()
CONCAT_WS()
EMPTY()
ENDSWITH()
FIELD()
FIND_IN_SET()
FORMAT()
HEX()
LEFT()
LENGTH()
LPAD()
LTRIM()
OCT()
REVERSE()
RPAD()
RTRIM()
SPACE()
STARTSWITH()
SUBSTRING()
SUBSTRING_INDEX()
TRIM()
系统配置
通用参数配置
时区配置
系统表目录
权限分类列表
使用限制
MatrixOne 的 JDBC 功能支持列表
MatrixOne DDL 语句分区支持的说明

MatrixOne 文件目录结构
故障诊断
慢查询
常用统计数据查询
数据库统计信息
审计
错误码
常见问题解答
产品常见问题
部署常见问题
SQL 常见问题
版本发布纪要
版本发布历史记录
v0.7.0
v0.6.0
v0.5.1
v0.5.0
v0.4.0
v0.3.0
v0.2.0
v0.1.0
名词术语表
社区贡献指南
快速贡献
贡献指南
有哪些贡献类型
贡献准备
报告 Issue
贡献代码

审核修改 · · ·

MatrixOne 文档

欢迎来到 MatrixOne 官方文档网站！

MatrixOne 是一款致力于“One Size Fits Most”的超融合异构数据库。

MatrixOne 通过超融合数据引擎实现单一数据库系统支持事务型（TP，Transactional Processing）、分析型（AP，Analytical Processing）、流式（Streaming）等多种数据负载，通过异构云原生架构实现单一数据库系统支持公有云原生、私有云、边缘云部署和使用。从而简化开发运维，消简数据碎片，提升数据端到端存算性能和开发敏捷度。

如果你发现了任何文档问题，请直接在 GitHub 上创建 Issue 报告该问题，或者直接提出 PR 来修复！

推荐阅读 👉

概述	快速上手	参考指南
MatrixOne 简介	MatrixOne 安装部署	SQL 参考指南
MatrixOne 架构	MatrixOne 的 SQL 基本操作	v0.7.0 发布公告
MySQL 兼容性	使用 MatrixOne 完成 SSB 测试	

MatrixOne 简介

MatrixOne 是一款面向未来的超融合异构数据库，通过超融合数据引擎实现单一数据库系统支持 TP、AP、Streaming 等多种数据负载，通过异构云原生架构实现单一数据库系统支持公有云原生、私有云、边缘云部署和使用。从而简化开发运维，消简数据碎片，提升数据端到端存算性能和开发敏捷度。

核心特性

超融合引擎

- **超融合引擎**

  融合数据引擎，单数据库即可支持 TP、AP、时序、机器学习等混合工作负载。

- **内置流引擎**

  利用独有的增量物化视图能力，无需跨数据库即可实现实时数据流处理。

异构云原生

- **异构统一**

  支持跨机房协同/多地协同/云边协同，实现无感知扩缩容，提供高效统一的数据管理。

- **多地多活**

  MatrixOne 采用最优的一致性协议，实现业内最短网络延迟的多地多活。

极致的性能

- **高性能**

  特有的向量化执行引擎，支持极速的复杂查询。单表、星型和雪花查询都具备极速分析性能。

- **强一致**

  提供跨存储引擎的高性能全局分布式事务能力，在保证极速分析性能的同时支持更新、删除和实时点查询。

- **高可用**

  存算分离，支持存储节点与计算节点独立扩缩容，高效应对负载变化。

用户价值

- **简化数据开发和运维**

  随着业务发展，企业使用的数据引擎和中间件越来越多，而每一个数据引擎平均依赖5+个基础组件，存储 3+个数据副本，每一个数据引擎都要各自安装、监控、补丁和升级。这些都导致数据引擎的选型、开发及运维成本高昂且不可控。在 MatrixOne 的一体化架构下，用户使用单个数据库即可服务多种数据应用，引入的数据组件和技术栈减少80%，大大简化了数据库管理和维护的成本。

- **消减数据碎片和不一致**

  在既有复杂的系统架构内，存在多条数据管道多份数据存储冗余。数据依赖复杂，导致数据更新维护复杂，上下游数据不一致问题频发，人工校对难度增大。MatrixOne 的高内聚架构和独有的增量物化视图能力，使得下游可以支持上游数据的实时更新，摆脱冗余的 ETL 流程，实现端到端实时数据处理。

- **无需绑定基础设施**

  因为基础设施的碎片化，企业的私有化数据集群和公有云数据集群之间数据架构和建设方案割裂，数据迁移成本高。而数据上云一旦选型确定数据库厂商，后续的集群扩容、其他组件采购等都将被既有厂商绑定。MatrixOne 提供统一的云边基础架构和高效统一的数据管理，企业数据架构不再被基础设施绑定，实现单数据集群跨云无感知扩缩容，提升性价比。

- **极速的分析性能**

  目前，由于缓慢的复杂查询性能以及冗余的中间表，数据仓库在业务敏捷性上的表现不尽人意，大量宽表的创建也严重影响迭代速度。MatrixOne 通过特有的因子化计算和向量化执行引擎，支持极速的复杂查询，单表、星型和雪花查询都具备极速分析性能。

- **像 TP 一样可靠的 AP 体验**

  传统数据仓库数据更新代价非常高，很难做到数据更新即可见。在营销风控，无人驾驶，智能工厂等实时计算要求高的场景或者上游数据变化快的场景中，当前的大数据分析系统无法支持增量更新，往往需要做全量的更新，耗时耗力。MatrixOne 通过提供跨存储引擎的高性能全局分布式事务能力，支持条级别的实时增量更新，在保证极速分析性能的同时支持更新、删除和实时点查询。

- **不停服自动扩缩容**

  传统数仓无法兼顾性能和灵活度，性价比无法做到最优。MatrixOne 基于存算分离的技术架构，支持存储节点与计算节点独立扩缩容，高效应对负载变化。

相关信息

本节描述了 MatrixOne 的产品核心特性和用户价值主张，如果您想了解有关 MatrixOne 产品架构设计的更多信息，请参阅以下内容：

- MatrixOne 技术架构

MatrixOne 功能列表

本文档列出了 MatrixOne 最新版本所支持的功能。

数据定义语言（Data definition language, DDL）

数据定义语言 (DDL)	支持（Y）/不支持（N）
CREATE DATABASE	Y
DROP DATABASE	Y
RENAME DATABASE	N
CREATE TABLE	Y
ALTER TABLE	N
RENAME TABLE	N
DROP TABLE	Y
CREATE INDEX	Y
DROP INDEX	Y
MODIFY COLUMN	N
PRIMARY KEY	Y
CREATE VIEW	Y
ALTER VIEW	Y
DROP VIEW	Y
CREATE OR REPLACE VIEW	N
TRUNCATE	Y
SEQUENCE	N
AUTO_INCREMENT	Y
Temporary tables	Y

SQL 语句

SQL 语句	支持（Y）/不支持（N）
SELECT	Y
INSERT	Y
UPDATE	Y
DELETE	Y
REPLACE	N
INSERT ON DUPLICATE KEY	N
LOAD DATA INFILE	Y
SELECT INTO OUTFILE	Y
INNER/LEFT/RIGHT/OUTER JOIN	Y
UNION, UNION ALL	Y
EXCEPT, INTERSECT	Y
GROUP BY, ORDER BY	Y
Common Table Expressions(CTE)	Y
START TRANSACTION, COMMIT, ROLLBACK	Y
EXPLAIN	Y
EXPLAIN ANALYZE	Y
Stored Procedure	N
Trigger	N
Event Scheduler	N
PARTITION BY	Y
LOCK TABLE	N

数据类型

数据类型分类	数据类型	支持（Y）/不支持（N）
整数类型	TINYINT	Y
	SMALLINT	Y
	INT	Y
	BIGINT	Y
	TINYINT UNSIGNED	Y
	SMALLINT UNSIGNED	Y
	INT UNSIGNED	Y
	BIGINT UNSIGNED	Y
浮点类型	FLOAT	Y
	DOUBLE	Y
字符串类型	CHAR	Y
	VARCHAR	Y
	TINYTEXT/TEXT/MEDIUMTEXT/LONGTEXT	Y
二进制类型	TINYBLOB/BLOB/MEDIUMBLOB/LONGBLOB	Y
时间与日期	Date	Y
	Time	Y
	DateTime	Y
	Timestamp	Y
Boolean	BOOL	Y
定点类型	DECIMAL	Y
JSON 类型	JSON	Y

索引与约束

索引与约束	支持（Y）/不支持（N）
主键约束	Y
复合主键	Y
唯一约束	Y
Secondary KEY	Y，仅语法实现
外键约束	Y
无效数据强制约束	Y
ENUM 和 SET 约束	N
非空约束	Y

事务

事务	支持（Y）/不支持（N）
1PC	Y
悲观事务	N
乐观事务	Y
分布式事务	Y
隔离级别	Y

函数与操作符

函数与操作符	名称
聚合函数	AVG()
	COUNT()
	MAX()
	MIN()

函数与操作符	名称
	Median()
	SUM()
	ANY_VALUE()
	BIT_OR()
	BIT_AND()
	BIT_XOR()
	STD()
	VARIANCE()
	GROUP_CONCAT()
	SLEEP()
数学类	ABS()
	SIN()
	COS()
	TAN()
	COT()
	ACOS()
	ATAN()
	SINH()
	FLOOR()
	ROUND()
	CEIL()
	POWER()
	PI()
	LOG()
	LN()

函数与操作符	名称
	UUID()
	EXP()
日期时间类	DATE_FORMAT()
	YEAR()
	MONTH()
	DATE()
	WEEKDAY()
	TIMESTAMP()
	DAYOFYEAR()
	EXTRACT()
	DATE_ADD()
	DATE_SUB()
	TO_DATE()
	DAY()
	UNIX_TIMESTAMP()
	FROM_UNIXTIME()
	UTC_TIMESTAMP()
	NOW()
	CURRENT_TIMESTAMP()
	DATEDIFF()
	TIMEDIFF()
	CURDATE()
字符串类	BIN()
	BIT_LENGTH()
	HEX()

函数与操作符	名称
	CONCAT()
	CONCAT_WS()
	FIND_IN_SET()
	FORMAT()
	OCT()
	EMPTY()
	LENGTH()
	BIT_LENGTH()
	LENGTHUTF8()
	CHAR_LENGTH()
	LEFT()
	TRIM()
	LTRIM()
	RTRIM()
	LPAD()
	RPAD()
	STARTSWITH()
	ENDSWITH()
	SUBSTRING()
	SPACE()
	REVERSE()
	SUBSTRING_INDEX()
	FIELD()
操作符	%, MOD
	*

函数与操作符	名称
	+
	-
	/
	Div
	=
	&
	>>
	<<
	^
	|
	~
	CAST()
	CONVERT()
	>
	>=
	<
	<>, !=
	<=
	=
	LIKE
	BETWEEN ... AND ...
	IN()
	IS/IS NOT
	IS/IS NOT NULL
	NOT BETWEEN ... AND ...

函数与操作符	名称
	LIKE
	NOT LIKE
	COALESCE()
	CASE...WHEN
	IF
	AND
	OR
	XOR
	NOT

MatrixOne 架构设计

MatrixOne 概述

MatrixOne 是一款面向未来的超融合异构云原生数据库，通过简化的分布式数据库引擎，跨多个数据中心、云、边缘和其他异构基础设施，同时支持事务性（OLTP）、分析性（OLAP）和流式工作（Sreaming）负载，这种多种引擎的融合，称为 HSTAP。

MatrixOne HSTAP 数据库对 HTAP 数据库进行了重新定义，HSTAP 旨在满足单一数据库内事务处理（TP）和分析处理（AP）的所有需求。与传统的 HTAP 相比，HSTAP 强调其内置的用于连接 TP 和 AP 表数据流处理能力，为用户提供了数据库可以像大数据平台一样灵活的使用体验。也恰恰得益于大数据的繁荣，很多用户已经熟悉了这种体验。用户使用 MatrixOne 只需要少量的集成工作，即可以获得覆盖整个 TP 和 AP 场景的一站式体验，同时可以摆脱传统大数据平台的臃肿架构及各种限制。

MatrixOne 架构

MatrixOne 架构如下如所示：

MatrixOne Architecture

参照上面的图示，MatrixOne 的体系结构分为五层，以下内容是从上至下对每层的介绍：

集群管理层

这一层负责集群管理，在云原生环境中与 Kubernetes 交互动态获取资源；在本地部署时，根据配置获取资源。集群状态持续监控，根据资源信息分配每个节点的任务。提供系统维护服务以确保所有系统组件在偶尔出现节点和网络故障的情况下正常运行，并在必要时重新平衡节点上的负载。集群管理层的主要组件是：

- Prophet 调度：提供负载均衡和节点 Keep-alive。
- 资源管理：提供物理资源。

Serverless 层

Serverless 层是一系列无状态节点的总称，整体上包含三类：

- 后台任务：最主要的功能是 Offload Worker，负责卸载成本高的压缩任务，以及将数据刷新到 S3 存储。
- SQL 计算节点：负责执行 SQL 请求，这里分为写节点和读节点，写节点还提供读取最新数据的能力。
- 流任务处理节点：负责执行流处理请求。

日志层

作为 MatrixOne 的单一数据源 (即 Single source of truth)，数据一旦写入日志层，则将永久地存储在 MatrixOne 中。它建立在我们世界级的复制状态机模型的专业知识之上，以保证我们的数据具有最先进的高吞吐量、高可用性和强一致性。它本身遵循完全模块化和分解的设计，也帮助解耦存储和计算层的核心组件，与传统的 NewSQL 架构相比，我们的架构具有更高的弹性。

存储层

存储层将来自日志层的传入数据转换为有效的形式，以供将来对数据进行处理和存储。包括为快速访问已写入 S3 的数据进行的缓存维护等。在 MatrixOne 中，TAE（即 Transactional Analytic Engine）是存储层的主要公开接口，它可以同时支持行和列存储以及事务处理能力。此外，存储层还包括其他内部使用的存储功能，例如流媒体的中间存储。

存储供应层

作为与基础架构解耦的 DBMS，MatrixOne 可以将数据存储在 S3/HDFS、本地磁盘、本地服务器、混合云或其他各类型云，以及智能设备的共享存储中。存储供应层通过为上层提供一个统一的接口来访问这些多样化的存储资源，并且不向上层暴露存储的复杂性。

MatrixOne 系统组件

MatrixOne Component

在 MatrixOne 中，为实现分布式与多引擎的融合，构建了多种不同的系统组件用于完成架构相关的层级的功能：

File Service

File Service 是 MatrixOne 负责所有存储介质读写的组件。存储介质包括内存、磁盘、对象存储等，它提供了如下特性：

- File Service 提供了一个统一的接口，使不同介质的读写，可以使用相同的接口。
- 接口的设计，遵循了数据不可变的理念。文件写入之后，就不允许再更新。数据的更新，通过产生新的文件来实现。
- 这样的设计，简化了数据的缓存、迁移、校验等操作，有利于提高数据操作的并发能力。
- 基于统一的读写接口，File Service 提供了分级的缓存，提供了灵活的缓存策略，以平衡读写速度和容量。

Log Service

Log Service 是 MatrixOne 中专门用于处理事务日志的组件，它具有如下功能特性：

- 采用 Raft 协议来保证一致性，采用多副本方式确保可用性。
- 保存并处理 MatrixOne 中所有的事务日志，在事务提交前确保 Log Service 的日志读写正常，在实例重启时，检查并回放日志内容。
- 在完成事务的提交与落盘后，对 Log Service 内容做截断，从而控制 Log Service 的大小，截断后仍然保留在 Log Service 中的内容，称为 Logtail。
- 如果多个 Log Service 副本同时出现宕机，那么整个 MatrixOne 将会发生宕机。

Database Node

Database Node（DN)，是用来运行 MatrixOne 的分布式存储引擎 TAE 的载体，它提供了如下特性：

- 管理 MatrixOne 中的元数据信息以及 Log Service 中保存的事务日志内容。
- 接收 Computing Node（CN) 发来的分布式事务请求，对分布式事务的读写请求进行裁决，将事务裁决结果推给 CN，将事务内容推给 Log Service，确保事务的 ACID 特性。
- 在事务中根据检查点生成快照，确保事务的快照隔离性，在事务结束后将快照信息释放。

Computing Node

Computing Node（CN），是 Matrixone 接收用户请求并处理 SQL 的组件，具体包括以下模块：

- Frontend，处理客户端 SQL 协议，接受客户端发送的 SQL 报文，然后解析得到 MatrixOne 可执行的 SQL，调用其他模块执行 SQL 后将查询结果组织成报文返回给客户端。
- Plan，解析 Frontend 处理后的 SQL，并根据 MatrixOne 的计算引擎生成逻辑执行计划发送给 Pipeline。
- Pipeline，解析逻辑计划，将逻辑计划转成实际的执行计划，然后 Pipeline 运行执行计划。
- Disttae，负责具体的读写任务，既包含了从 DN 同步 Logtail 和从 S3 读取数据，也会把写入的数据发送给 DN。

Stream Engine

Stream Engine 是一个新组件，用于简化从 OLTP 到 OLAP 的 ETL 处理过程。Stream Engine 还处在 MatrixOne 开发过程中，暂尚未实现。

MatrixOne 特性

在 MatrixOne 版本 0.7 中，具有如下特性，让你在使用 MatrixOne 的过程中更加高效：

分布式架构

在 MatrixOne 中，采用了的分布式存算分离的分布式架构，存储层、数据层、计算层的分离，使得 MatrixOne 在遇到系统资源瓶颈时，能够灵活实现节点的扩容。同时，多节点的架构下，资源可以进行更加有效率地分配，一定程度上避免了热点与资源征用。

事务与隔离

在 MatrixOne 中，事务采用了乐观事务与快照隔离。

在分布式架构下，乐观事务可以通过较少的冲突获得更优的性能。同时在实现方式上，能够实现隔离级别更高的快照隔离。为了保证事务的 ACID 四个要素，MatrixOne 目前支持且仅支持快照隔离一种隔离级别。该隔离级别较常见的读已提交相比，隔离级别更加严格，既可以有效防止脏读，又能够更好地适配分布式乐观事务。

云原生

MatrixOne 是一款云原生的数据库，从存储层，适配本地磁盘、AWS S3、NFS 等多种存储方式，通过 File service 实现了对多种不同类型存储的无感知管理。MatrixOne 集群可以在多种基础设施环境下稳定运行，既可以适配企业私有云，又可以在不同的公有云厂商环境下提供服务。

相关信息

本节介绍了 MatrixOne 的整体架构概览。其他信息可参见：

- 安装单机版 MatrixOne
- MySQL 兼容性
- 最新发布信息

MySQL 兼容性

MatrixOne 的 SQL 语法兼容了 MySQL 8.0.23 版本。

语句类型	语法	兼容性
数据定义语言（Data Definition Language，DDL）	CREATE DATABASE	不支持中文命名的表名
		支持部分拉丁语
		`CHARSET`，`COLLATE`，`ENCRYPTION` 目前可以使用但无法生效
	CREATE TABLE	暂不支持表的分区
		不支持 `Create table .. as` 语句
		不支持列级约束
		暂不支持 `KEY(column)` 语法
		支持 `AUTO_INCREMENT`，但暂不支持自定义起始值
	ALTER	暂不支持
	DROP DATABASE	同 MySQL
	DROP TABLE	同 MySQL.
	CREAT VIEW	不支持 with check option 子句
数据操作语言（Data Manipulation Language，DML）	UPDATE	同 MySQL
	DELETE	同 MySQL
	INSERT	暂不支持 `LOW_PRIORITY`，`DELAYED`，`HIGH_PRIORITY`
		暂不支持使用 `INSERT INTO VALUES` 函数或表达式
		支持批量插入，最多支持 160,000 行
		暂不支持 `ON DUPLICATE KEY UPDATE`

语句类型	语法	兼容性
		暂不支持 `DELAYED`
		不分支持拉丁文命名
		当前的 SQL 模式同 MySQL 中的 `only_full_group_by` 模式
	SELECT	在 `GROUP BY` 中不支持表别名
		暂不支持 `SELECT...FOR UPDATE` 从句
		部分支持 `INTO OUTFILE`
	LOAD DATA	支持导入 csv 和 jsonline 文件
		包括符 enclosed 应该为""
		字段分隔符 `FILEDS TERMINATED BY` 应该为 ”,” 或 ”
		行分隔符 `LINES TERMINATED BY` 应该为 “\n”
		支持 `SET`，但仅支持 `SET columns_name=nullif(expr1,expr2)`
		不支持本地关键词
		不支持 `ESCAPED BY`
	JOIN	同 MySQL
	SUBQUERY	暂不支持 `Non-scalar` 子查询，但可以作为过滤条件使用
数据库管理语句	SHOW	部分支持
工具类语句	Explain	分析的结果与 MySQL 有所不同
		不支持 json 类型的输出
	Other statements	暂不支持
数据类型	Boolean	与 MySQL 的布尔值类型 int 不同，MatrixOne 的布尔值是一个新的类型，它的值只能是 true 或 false。
	Int/Bigint/Smallint/Tinyint	同 MySQL
	char/varchar	同 MySQL
	Float/double	与 MySQL 的精度有所不同

语句类型	语法	兼容性
	DECIMAL	最大精度为 38 位。
	Date	只支持 `YYYY-MM-DD` 与 `YYYYMMDD` 形式
	Datetime	只支持 `YYYY-MM-DD hh:mi:ssssss` 与 `YYYYMMDD hh:mi:ssssss` 形式
	Timestamp	同 MySQL
	Other types	暂不支持
运算符	"+","-","*","/"	同 MySQL
	DIV, %, MOD	同 MySQL
	LIKE	同 MySQL
	IN	只支持常数列表
	NOT, AND, &&,OR, "||"	同 MySQL
	CAST	支持不同转换规则。
Functions	MAX, MIN, COUNT, AVG, SUM	同 MySQL
	any_value	`Any_value` 是 MatrixOne 中的一个聚合函数。不能在 `GROUP` 或过滤条件中使用。
	正则表达式	即 `REGEXP_INSTR()`，`REGEXP_LIKE()`，`REGEXP_REPLACE()`，`REGEXP_SUBSTR()` 暂不支持第三个参数
	to_date	对于日期的入参仅支持常量
系统命令	SHOW GRANTS FOR USERS	目前只能看到直接授权的角色所具有的权限，继承角色的权限不能显示

最新发布

MatrixOne 的最新版本为 0.7.0，发布于 2023 年 2 月 23 日，详情请见：

- v0.7.0 发布公告

单机部署 MatrixOne

单机版 MatrixOne 适用场景即是使用单台服务器，体验 MatrixOne 最小的完整拓扑，并模拟开发环境下的部署步骤。

推荐安装环境：

作为一款开源数据库，MatrixOne 目前支持主流的 **Linux** 和 **MacOS** 系统。为了快速上手，本文档中优先推荐如下硬件规格：

操作系统	操作系统版本	CPU	内存
CentOS	7.3 及以上	x86 CPU；4 核	32 GB
macOS	Monterey 12.3 及以上	- x86 CPU；4 核 - ARM；4 核	32 GB

你也可以查阅硬件与操作系统要求，查看更多关于硬件规格推荐，选用合适的硬件环境。

在 macOS 上部署

你可以在以下三种方式中选择最适合你的一种，在 macOS 上安装并连接 MatrixOne：

- 使用源代码部署
- 使用二进制包部署
- 使用 Docker 部署

在 Linux 上部署

你可以在以下三种方式中选择最适合你的一种，在 Linux 上安装并连接 MatrixOne：

- 使用源代码部署
- 使用二进制包部署
- 使用 Docker 部署

参考文档

- 更多有关连接 MatrixOne 的方式，参见：
  - 客户端连接 MatrixOne 服务
  - JDBC 连接 MatrixOne 服务
  - Python 连接 MatrixOne 服务。
- 常见的安装和部署问题，参见安装和部署常见问题。
- 关于分布式部署 MatrixOne，参见 MatrixOne 分布式集群部署。

使用源代码部署

本篇文档将指导你使用源代码部署单机版 MatrixOne。

步骤 1：安装部署 Go 语言

1. 点击 Go Download and install 入到 **Go** 的官方文档，按照官方指导安装步骤完成 **Go** 语言的安装。

   Note: 建议 Go 语言版本为 1.19 版本。

2. 验证 **Go** 是否安装，请执行代码 `go version`，安装成功代码行示例如下：

   ```
   go version go1.19 darwin/arm64
   ```

步骤 2：安装 GCC

1. 验证 GCC 环境是否需要安装：

   ```
   gcc -v
   bash: gcc: command not found
   ```

   如代码所示，未显示 GCC 的版本，则表示 **GCC** 的环境需要安装。

2. 点击 GCC Download and install 入到 **GCC** 的官方文档，按照官方指导安装步骤完成 **GCC** 的安装。

   Note: 建议 GCC 版本为 8.5 版本及以上。

3. 验证 **GCC** 是否安装，请执行代码 `gcc -v`，安装成功代码行示例如下：

   ```
   Apple clang version 14.0.0 (clang-1400.0.29.202)
   Target: arm64-apple-darwin22.2.0
   Thread model: posix
   InstalledDir:
   /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoo
   ```

步骤 3：获取 MatrixOne 源代码

根据您的需要，选择您所获取的代码永远保持最新，还是获得稳定版本的代码。

=== “通过 MatrixOne (开发版本) 代码搭建”

main 分支是默认分支，主分支上的代码总是最新的，但不够稳定。

1. 获取 MatrixOne(开发版本) 代码方法如下：

```
git clone https://github.com/matrixorigin/matrixone.git
cd matrixone
```

2. 运行 `make build` 编译文件：

```
make build
```

Tips: 你也可以运行 `make debug` 与 `make clean` 或者其他任何 `Makefile` 支持的命令；`make debug` 可以用来调试构建进程，`make clean` 可以清除构建进程。如果在 `make build` 时产生 `Get "https://proxy.golang.org/........": dial tcp 142.251.43.17:443: i/o timeout` 报错，参见[安装和部署常见问题]进行解决。

=== “通过 MatrixOne (稳定版本) 代码搭建”

1. 如果您想获得 MatrixOne 发布的最新稳定版本代码，请先从 **main** 切换选择至 **0.7.0** 版本分支。

```
git clone https://github.com/matrixorigin/matrixone.git
cd matrixone
git checkout 0.7.0
```

2. 运行 `make config` 和 `make build` 编译文件：

```
make config
make build
```

Tips: 你也可以运行 `make debug` 与 `make clean` 或者其他任何 `Makefile` 支持的命令；`make debug` 可以用来调试构建进程，`make clean` 可以清除构建进程。如果在 `make build` 时产生

```
Get "https://proxy.golang.org/........": dial tcp 142.251.43.17:443: i/o timeout
```

报错，参见[安装和部署常见问题](#)进行解决。

> ⚠ **注意**
>
> ARM 芯片硬件配置下，MatrixOne 仅支持通过源代码方式进行安装部署；如果你使用的是 MacOS 系统 M1 及以上版本，请参见本篇文档进行安装部署 MatrixOne。

步骤 4：启动 MatrixOne 服务

=== "**在终端的前台启动 MatrixOne 服务**"

该启动方式会在终端的前台运行 `mo-service` 进行，实时打印系统日志。如果你想停止 MatrixOne 服务器，只需按 CTRL+C 或关闭当前终端。

```
# Start mo-service in the frontend
./mo-service -launch ./etc/quickstart/launch.toml
```

在前台启动模式下，产生很多日志，接下来你可以启动新的终端，连接 MatrixOne。

=== "**推荐使用：在终端的后台启动 MatrixOne 服务**"

该启动方法会在后台运行 `mo-service` 进程，系统日志将重定向到 `test.log` 文件中。如果你想停止 MatrixOne 服务器，你需要通过以下命令查找出它的 `PID` 进程号并消除进程。下面是整个过程的完整示例。

```
# Start mo-service in the backend
./mo-service --daemon --launch ./etc/quickstart/launch.toml &> test.log &

# Find mo-service PID
ps aux | grep mo-service

[root ~]# ps aux | grep mo-service
root 15277 2.8 16.6 8870276 5338016 ? Sl Nov25 156:59 ./mo-service -
launch ./etc/quickstart/launch.toml
root 836740 0.0 0.0 12136 1040 pts/0 S+ 10:39 0:00 grep --color=auto mo-
service

# Kill the mo-service process
kill -9 15277
```

Tips: 如上述示例所示，使用命令 `ps aux | grep mo-service` 首先查找出 MatrixOne 运行的进程号为 `15277`， `kill -9 15277` 即表示停止进程号为 `15277` 的 MatrixOne。

接下来你可以进行下一步 - 连接 MatrixOne。

> ⚠ **注意**
>
> 如果在某一分支上构建以后，需要切换分支再进行构建，运行后则出现 panic，需要清理数据文件目录。参见[安装和部署常见问题]进行解决。

步骤 5：连接 MatrixOne

安装并配置 MySQL 客户端

1. 点击 [MySQL Community Downloads]，进入到 MySQL 客户端下载安装页面，根据你的操作系统和硬件环境，下拉选择 **Select Operating System > macOS**，再下拉选择 **Select OS Version**，按需选择下载安装包进行安装。

   Note: 建议 MySQL 客户端版本为 8.0.30 版本及以上。

2. 配置 MySQL 客户端环境变量：

   1. 打开一个新的终端，输入如下命令：

```
cd ~
sudo vim .bash_profile
```

2. 回车执行上面的命令后，需要输入 root 用户密码，即你在安装 MySQL 客户端时，你在安装窗口设置的 root 密码；如果没有设置密码，则直接回车跳过即可。

3. 输入/跳过 root 密码后，即进入了*. bash_profile*，点击键盘上的 *i* 进入 insert 状态，即可在文件下方输入如下命令：

```
export PATH=$\{PATH}:/usr/local/mysql/bin
```

4. 输入完成后，点击键盘上的 esc 退出 insert 状态，并在最下方输入 `:wq` 保存退出。

5. 输入命令 `source .bash_profile`，回车执行，运行环境变量。

6. 测试 MySQL 是否可用：

   - 方式一：输入命令 `mysql -u root -p`，回车执行，需要 root 用户密码，显示 `mysql>` 即表示 MySQL 客户端已开启。

   - 方式二：执行命令 `mysql --version`，安装成功提示：
     `mysql Ver 8.0.31 for macos12 on arm64 (MySQL Community Server - GPL)`

7. MySQL 如可用，关闭当前终端，继续浏览下一章节**连接 MatrixOne 服务**。

Tips: 目前，MatrixOne 只兼容 Oracle MySQL 客户端，因此一些特性可能无法在 MariaDB、Percona 客户端下正常工作。

连接 MatrixOne

- 你可以使用 MySQL 命令行客户端来连接 MatrixOne。打开一个新的终端，直接输入以下指令：

```
mysql -h IP -P PORT -uUsername -p
```

输入完成上述命令后，终端会提示你提供用户名和密码。你可以使用我们的内置帐号：

· user: dump · password: 111

- 你也可以使用 MySQL 客户端下述命令行，输入密码，来连接 MatrixOne 服务：

```
mysql -h 127.0.0.1 -P 6001 -udump -p
Enter password:
```

目前，MatrixOne 只支持 TCP 监听。

使用二进制包部署

本篇文档将指导你使用二进制包部署单机版 MatrixOne。

步骤 1：安装下载工具

我们提供**下载二进制包**的方式安装 MatrixOne，如果你喜欢通过命令行进行操作，那么你可以提前准备安装好 `wget` 或 `curl`。

Tips: 建议你下载安装这两个下载工具其中之一，方便后续通过命令行下载二进制包。

=== "安装 `wget`"

`wget` 工具用来从指定的 URL 下载文件。`wget` 是专门的文件下载工具，它非常稳定，而且下载速度快。

进入到Homebrew页面按照步骤提示，先安装 **Homebrew**，再安装 `wget`。 验证 `wget` 是否安装成功可以使用如下命令行：

```
wget -V
```

安装成功结果(仅展示一部分代码)如下：

```
GNU Wget 1.21.3 在 darwin21.3.0 上编译。
...
Copyright © 2015 Free Software Foundation, Inc.
...
```

=== "安装 `curl`"

`curl` 是一个利用 URL 规则在命令行下工作的文件传输工具。`curl` 是综合传输工具，支持文件的上传和下载。

进入到Curl官网按照官方指导安装 `curl`。 验证 `curl` 是否安装成功可以使用如下命令行：

```
curl --version
```

安装成功结果(仅展示一部分代码)如下：

```
curl 7.84.0 (x86_64-apple-darwin22.0) libcurl/7.84.0 (SecureTransport)
LibreSSL/3.3.6 zlib/1.2.11 nghttp2/1.47.0
Release-Date: 2022-06-27
...
```

步骤 2：下载二进制包并解压

下载方式一和**下载方式二**需要先安装下载工具 `wget` 货 `curl`，如果你未安装，请先安装下载工具。

=== "**下载方式一：`wget` 工具下载安装二进制包**"

```
wget
https://github.com/matrixorigin/matrixone/releases/download/v0.7.0/mo-
v0.7.0-darwin-x86_64.zip
unzip mo-v0.7.0-darwin-x86_64.zip
```

=== "**下载方式二：`curl` 工具下载二进制包**"

```
curl -OL
https://github.com/matrixorigin/matrixone/releases/download/v0.7.0/mo-
v0.7.0-darwin-x86_64.zip
unzip mo-v0.7.0-darwin-x86_64.zip
```

=== "**下载方式三：页面下载**"

如果你想通过更直观的页面下载的方式下载，直接点击进入版本 0.7.0，下拉找到 **Assets** 栏，点击安装包 *mo-v0.7.0-darwin-x86_64.zip* 下载即可。

> ⚠ **注意**
>
> ARM 芯片硬件配置下，MatrixOne 仅支持通过源代码方式进行安装部署；如果你使用的是 MacOS 系统 M1 及以上版本，请使用源代码构建的方式安装部署 MatrixOne。若果在 X86 硬件配置下使用二进制方式安装部署 MatrixOne 会导致未知问题。

步骤 3：启动 MatrixOne 服务

=== "**在终端的前台启动 MatrixOne 服务**"

该启动方式会在终端的前台运行 `mo-service` 进行，实时打印系统日志。如果你想停止 MatrixOne 服务器，只需按 CTRL+C 或关闭当前终端。

```
# Start mo-service in the frontend
./mo-service -launch ./etc/quickstart/launch.toml
```

在前台启动模式下，产生很多日志，接下来你可以启动新的终端，连接 MatrixOne。

=== "**推荐使用：在终端的后台启动 MatrixOne 服务**"

该启动方法会在后台运行 `mo-service` 进程，系统日志将重定向到 `test.log` 文件中。如果你想停止 MatrixOne 服务器，你需要通过以下命令查找出它的 `PID` 进程号并消除进程。下面是整个过程的完整示例。

```
# Start mo-service in the backend
./mo-service --daemon --launch ./etc/quickstart/launch.toml &> test.log &

# Find mo-service PID
ps aux | grep mo-service

[root@VM-0-10-centos ~]# ps aux | grep mo-service
root 15277 2.8 16.6 8870276 5338016 ? Sl Nov25 156:59 ./mo-service -
launch ./etc/quickstart/launch.toml
root 836740 0.0 0.0 12136 1040 pts/0 S+ 10:39 0:00 grep --color=auto mo-
service

# Kill the mo-service process
kill -9 15277
```

Tips: 如上述示例所示，使用命令 `ps aux | grep mo-service` 首先查找出 MatrixOne 运行的进程号为 `15277`， `kill -9 15277` 即表示停止进程号为 `15277` 的 MatrixOne。

接下来你可以进行下一步 - 连接 MatrixOne。

> ⚠ **注意**
>
> 如果在某一分支上构建以后，需要切换分支再进行构建，运行后则出现 panic，需要清理数据文件目录。参见安装和部署常见问题进行解决。

步骤 4：连接 MatrixOne

安装并配置 MySQL 客户端

1. 点击 MySQL Community Downloads，进入到 MySQL 客户端下载安装页面，根据你的操作系统和硬件环境，下拉选择 **Select Operating System > macOS**，再下拉选择 **Select OS Version**，按需选择下载安装包进行安装。

   Note: 建议 MySQL 客户端版本为 8.0.30 版本及以上。

2. 配置 MySQL 客户端环境变量：

   1. 打开一个新的终端，输入如下命令：

```
cd ~
sudo vim .bash_profile
```

2. 回车执行上面的命令后，需要输入 root 用户密码，即你在安装 MySQL 客户端时，你在安装窗口设置的 root 密码；如果没有设置密码，则直接回车跳过即可。

3. 输入/跳过 root 密码后，即进入了*. bash_profile*，点击键盘上的 *i* 进入 insert 状态，即可在文件下方输入如下命令：

```
export PATH=$\{PATH}:/usr/local/mysql/bin
```

4. 输入完成后，点击键盘上的 esc 退出 insert 状态，并在最下方输入 `:wq` 保存退出。

5. 输入命令 `source .bash_profile`，回车执行，运行环境变量。

6. 测试 MySQL 是否可用：

   - 方式一：输入命令 `mysql -u root -p`，回车执行，需要 root 用户密码，显示 `mysql>` 即表示 MySQL 客户端已开启。

   - 方式二：执行命令 `mysql --version`，安装成功提示：
     `mysql Ver 8.0.31 for macos12 on arm64 (MySQL Community Server - GPL)`

7. MySQL 如可用，关闭当前终端，继续浏览下一章节**连接 MatrixOne 服务**。

Tips: 目前，MatrixOne 只兼容 Oracle MySQL 客户端，因此一些特性可能无法在 MariaDB、Percona 客户端下正常工作。

连接 MatrixOne

- 你可以使用 MySQL 命令行客户端来连接 MatrixOne。打开一个新的终端，直接输入以下指令：

```
mysql -h IP -P PORT -uUsername -p
```

输入完成上述命令后，终端会提示你提供用户名和密码。你可以使用我们的内置帐号：

· user: dump · password: 111

- 你也可以使用 MySQL 客户端下述命令行，输入密码，来连接 MatrixOne 服务：

```
mysql -h 127.0.0.1 -P 6001 -udump -p
Enter password:
```

目前，MatrixOne 只支持 TCP 监听。

使用 Docker 部署

本篇文档将指导你使用 Docker 部署单机版 MatrixOne。

步骤 1：下载安装 Docker

1. 点击 Get Docker，进入 Docker 的官方文档页面，根据你的操作系统，下载安装对应的 Docker。
2. 安装完成后，通过下述代码行确认 Docker 版本，验证 Docker 安装是否成功：

   ```
   docker --version
   ```

   安装成功，代码示例如下：

   ```
   Docker version 20.10.17, build 100c701
   ```

3. 直接打开你本地 Docker 客户端，启动 Docker。

步骤 2：获取 MatrixOne 镜像并启动

使用以下命令将从 Docker Hub 中拉取 MatrixOne 镜像，你可以选择稳定版本镜像，或开发版本镜像。

=== “稳定版本的镜像（0.7.0）”

```
```bash
docker pull matrixorigin/matrixone:0.7.0
docker run -d -p 6001:6001 --name matrixone matrixorigin/matrixone:0.7.0
```
```

=== “开发版本的镜像”

```
获取最新开发版本的镜像，参见[Docker Hub]
(https://hub.docker.com/r/matrixorigin/matrixone/tags)，找到最新Tag，拉取镜
像。拉取镜像代码示例如下：

```bash
docker pull matrixorigin/matrixone:nightly-commitnumber
docker run -d -p 6001:6001 --name matrixone
matrixorigin/matrixone:nightly-commitnumber
```

__Note__: 如上面代码所示，*nightly* 为标识的 Tag 版本每天都进行更新，请注意获取最新
的镜像。
```

如需挂载数据目录或配置自定义文件，参见[挂载目录到 Docker 容器]。

步骤 3：连接 MatrixOne

安装并配置 MySQL 客户端

1. 点击 [MySQL Community Downloads]，进入到 MySQL 客户端下载安装页面，根据你的操作系统和硬件环境，下拉选择 **Select Operating System > macOS**，再下拉选择 **Select OS Version**，按需选择下载安装包进行安装。

   Note: 建议 MySQL 客户端版本为 8.0.30 版本及以上。

2. 配置 MySQL 客户端环境变量：

   1. 打开一个新的终端，输入如下命令：

      ```
      cd ~
      sudo vim .bash_profile
      ```

   2. 回车执行上面的命令后，需要输入 root 用户密码，即你在安装 MySQL 客户端时，你在安装窗口设置的 root 密码；如果没有设置密码，则直接回车跳过即可。

   3. 输入/跳过 root 密码后，即进入了*. bash_profile*，点击键盘上的 *i* 进入 insert 状态，即可在文件下方输入如下命令：

      ```
      export PATH=${PATH}:/usr/local/mysql/bin
      ```

4. 输入完成后，点击键盘上的 esc 退出 insert 状态，并在最下方输入 `:wq` 保存退出。

5. 输入命令 `source .bash_profile`，回车执行，运行环境变量。

6. 测试 MySQL 是否可用：

   - 方式一：输入命令 `mysql -u root -p`，回车执行，需要 root 用户密码，显示 `mysql>` 即表示 MySQL 客户端已开启。

   - 方式二：执行命令 `mysql --version`，安装成功提示：
     `mysql Ver 8.0.31 for macos12 on arm64 (MySQL Community Server - GPL)`

7. MySQL 如可用，关闭当前终端，继续浏览下一章节**连接 MatrixOne 服务**。

Tips: 目前，MatrixOne 只兼容 Oracle MySQL 客户端，因此一些特性可能无法在 MariaDB、Percona 客户端下正常工作。

连接 MatrixOne

- 你可以使用 MySQL 命令行客户端来连接 MatrixOne。打开一个新的终端，直接输入以下指令：

```
```
mysql -h IP -P PORT -uUsername -p
```

输入完成上述命令后，终端会提示你提供用户名和密码。你可以使用我们的内置帐号：

· user: dump
· password: 111
```

- 你也可以使用 MySQL 客户端下述命令行，输入密码，来连接 MatrixOne 服务：

```
```
mysql -h 127.0.0.1 -P 6001 -udump -p
Enter password:
```
```

目前，MatrixOne 只支持 TCP 监听。

使用源代码部署

本篇文档将指导你使用源代码部署单机版 MatrixOne。

步骤 1：安装部署 Go 语言

1. 点击 Go Download and install 入到 **Go** 的官方文档，按照官方指导安装步骤完成 **Go** 语言的安装。

   Note: 建议 Go 语言版本为 1.19 版本。

2. 验证 **Go** 是否安装，请执行代码 `go version`，安装成功代码行示例如下：

   ```
   go version go1.19.4 linux/amd64
   ```

步骤 2：安装 GCC

1. 验证 GCC 环境是否需要安装：

   ```
   gcc -v
   bash: gcc: command not found
   ```

   如代码所示，未显示 GCC 的版本，则表示 **GCC** 的环境需要安装。

2. 点击 GCC Download and install 入到 **GCC** 的官方文档，按照官方指导安装步骤完成 **GCC** 的安装。

   Note: 建议 GCC 版本为 8.5 版本及以上。

3. 验证 **GCC** 是否安装，请执行代码 `gcc -v`，安装成功代码行示例如下（只展示部分代码）：

   ```
   Using built-in specs.
   COLLECT_GCC=gcc
   ...
   Thread model: posix
   gcc version 9.3.1 20200408 (Red Hat 9.3.1-2) (GCC)
   ```

步骤 3：获取 MatrixOne 源代码

根据您的需要，选择您所获取的代码永远保持最新，还是获得稳定版本的代码。

=== “通过 MatrixOne (开发版本) 代码搭建”

main 分支是默认分支，主分支上的代码总是最新的，但不够稳定。

1. 获取 MatrixOne(开发版本) 代码方法如下：

```
git clone https://github.com/matrixorigin/matrixone.git
cd matrixone
```

2. 运行 `make build` 编译文件：

```
make build
```

Tips: 你也可以运行 `make debug` 与 `make clean` 或者其他任何 `Makefile` 支持的命令；`make debug` 可以用来调试构建进程，`make clean` 可以清除构建进程。如果在 `make build` 时产生 `Get "https://proxy.golang.org/........": dial tcp 142.251.43.17:443: i/o timeout` 报错，参见[安装和部署常见问题]进行解决。

=== “通过 MatrixOne (稳定版本) 代码搭建”

1. 如果您想获得 MatrixOne 发布的最新稳定版本代码，请先从 **main** 切换选择至 **0.7.0** 版本分支。

```
git clone https://github.com/matrixorigin/matrixone.git
cd matrixone
git checkout 0.7.0
```

2. 运行 `make config` 和 `make build` 编译文件：

```
make config
make build
```

Tips: 你也可以运行 `make debug` 与 `make clean` 或者其他任何 `Makefile` 支持的命令；`make debug` 可以用来调试构建进程，`make clean` 可以清除构建进程。如果在 `make build` 时产生

```
Get "https://proxy.golang.org/........": dial tcp 142.251.43.17:443: i/o timeout
```

报错，参见[安装和部署常见问题](#)进行解决。

步骤 4：启动 MatrixOne 服务

=== "**在终端的前台启动 MatrixOne 服务**"

该启动方式会在终端的前台运行 `mo-service` 进行，实时打印系统日志。如果你想停止 MatrixOne 服务器，只需按 CTRL+C 或关闭当前终端。

```
# Start mo-service in the frontend
./mo-service -launch ./etc/quickstart/launch.toml
```

在前台启动模式下，产生很多日志，接下来你可以启动新的终端，连接 MatrixOne。

=== "**推荐使用：在终端的后台启动 MatrixOne 服务**"

该启动方法会在后台运行 `mo-service` 进程，系统日志将重定向到 `test.log` 文件中。如果你想停止 MatrixOne 服务器，你需要通过以下命令查找出它的 `PID` 进程号并消除进程。下面是整个过程的完整示例。

```
# Start mo-service in the backend
./mo-service --daemon --launch ./etc/quickstart/launch.toml &> test.log &

# Find mo-service PID
ps aux | grep mo-service

[root@VM-0-10-centos ~]# ps aux | grep mo-service
root 15277 2.8 16.6 8870276 5338016 ? Sl Nov25 156:59 ./mo-service -launch ./etc/quickstart/launch.toml
root 836740 0.0 0.0 12136 1040 pts/0 S+ 10:39 0:00 grep --color=auto mo-service

# Kill the mo-service process
kill -9 15277
```

Tips: 如上述示例所示，使用命令 `ps aux | grep mo-service` 首先查找出 MatrixOne 运行的进程号为 `15277`， `kill -9 15277` 即表示停止进程号为 `15277` 的 MatrixOne。

接下来你可以进行下一步 - 连接 MatrixOne。

> **⚠ 注意**
>
> 如果在某一分支上构建以后，需要切换分支再进行构建，运行后则出现 panic，需要清理数据文件目录。参见安装和部署常见问题进行解决。

步骤 5：连接 MatrixOne

安装并配置 MySQL 客户端

1. 点击 MySQL Community Downloads，进入到 MySQL 客户端下载安装页面，根据你的操作系统和硬件环境，下拉选择 **Select Operating System**，再下拉选择 **Select OS Version**，按需选择下载安装包进行安装。

   Note: 建议 MySQL 客户端版本为 8.0.30 版本及以上。

2. 配置 MySQL 客户端环境变量：

   1. 打开一个新的终端，输入如下命令：

      ```
      cd ~
      sudo vim /etc/profile
      ```

   2. 回车执行上面的命令后，需要输入 root 用户密码，即你在安装 MySQL 客户端时，你在安装窗口设置的 root 密码；如果没有设置密码，则直接回车跳过即可。

   3. 输入/跳过 root 密码后，即进入了*. bash_profile*，点击键盘上的 *i* 进入 insert 状态，即可在文件下方输入如下命令：

      ```
      export PATH=/software/mysql/bin:$PATH
      ```

   4. 输入完成后，点击键盘上的 esc 退出 insert 状态，并在最下方输入 `:wq` 保存退出。

5. 输入命令 `source /etc/profile`，回车执行，运行环境变量。

6. 测试 MySQL 是否可用：

   - 方式一：输入命令 `mysql -u root -p`，回车执行，需要 root 用户密码，显示 `mysql>` 即表示 MySQL 客户端已开启。

   - 方式二：执行命令 `mysql --version`，安装成功提示：`mysql Ver 8.0.31 for Linux on x86_64 (Source distribution)`

7. MySQL 如可用，关闭当前终端，继续浏览下一章节**连接 MatrixOne 服务**。

Tips: 目前，MatrixOne 只兼容 Oracle MySQL 客户端，因此一些特性可能无法在 MariaDB、Percona 客户端下正常工作。

连接 MatrixOne

- 你可以使用 MySQL 命令行客户端来连接 MatrixOne。打开一个新的终端，直接输入以下指令：

```
mysql -h IP -P PORT -uUsername -p
```

  输入完成上述命令后，终端会提示你提供用户名和密码。你可以使用我们的内置帐号：

  · user: dump · password: 111

- 你也可以使用 MySQL 客户端下述命令行，输入密码，来连接 MatrixOne 服务：

```
mysql -h 127.0.0.1 -P 6001 -udump -p
Enter password:
```

目前，MatrixOne 只支持 TCP 监听。

使用二进制包部署

本篇文档将指导你使用二进制包部署单机版 MatrixOne。

步骤 1：安装下载工具

我们提供**下载二进制包**的方式安装 MatrixOne，如果你喜欢通过命令行进行操作，那么你可以提前准备安装好 `wget` 或 `curl`。

Tips: 建议你下载安装这两个下载工具其中之一，方便后续通过命令行下载二进制包。

=== "安装 `wget`"

`wget` 工具用来从指定的 URL 下载文件。`wget` 是专门的文件下载工具，它非常稳定，而且下载速度快。

进入到Homebrew页面按照步骤提示，先安装 **Homebrew**，再安装 `wget`。 验证 `wget` 是否安装成功可以使用如下命令行：

```
wget -V
```

安装成功结果(仅展示一部分代码)如下：

```
GNU Wget 1.21.3 built on linux-gnu.
...
Copyright (C) 2015 Free Software Foundation, Inc.
...
```

=== "安装 `curl`"

`curl` 是一个利用 URL 规则在命令行下工作的文件传输工具。`curl` 是综合传输工具，支持文件的上传和下载。

进入到Curl官网按照官方指导安装 `curl`。 验证 `curl` 是否安装成功可以使用如下命令行：

```
curl --version
```

安装成功结果(仅展示一部分代码)如下：

```
curl 7.84.0 (x86_64-pc-linux-gnu) libcurl/7.84.0 OpenSSL/1.1.1k-fips
zlib/1.2.11
Release-Date: 2022-06-27
...
```

步骤 2：下载二进制包并解压

下载方式一和**下载方式二**需要先安装下载工具 `wget` 货 `curl`，如果你未安装，请先安装下载工具。

=== "**下载方式一：`wget` 工具下载安装二进制包**"

```
wget
https://github.com/matrixorigin/matrixone/releases/download/v0.7.0/mo-
v0.7.0-linux-amd64.zip
unzip mo-v0.7.0-linux-amd64.zip
```

=== "**下载方式二：`curl` 工具下载二进制包**"

```
curl -OL
https://github.com/matrixorigin/matrixone/releases/download/v0.7.0/mo-
v0.7.0-linux-amd64.zip
unzip mo-v0.7.0-linux-amd64.zip
```

=== "**下载方式三：页面下载**"

如果你想通过更直观的页面下载的方式下载，直接点击进入版本 0.7.0，下拉找到 **Assets** 栏，点击安装包 *mo-v0.7.0-linux-amd64.zip* 下载即可。

步骤 3：启动 MatrixOne 服务

=== "**在终端的前台启动 MatrixOne 服务**"

该启动方式会在终端的前台运行 `mo-service` 进行，实时打印系统日志。如果你想停止 MatrixOne 服务器，只需按 CTRL+C 或关闭当前终端。

```
# Start mo-service in the frontend
./mo-service -launch ./etc/quickstart/launch.toml
```

=== "在终端的后台启动 MatrixOne 服务"

该启动方法会在后台运行 `mo-service` 进程，系统日志将重定向到 `test.log` 文件中。如果你想停止 MatrixOne 服务器，你需要通过以下命令查找出它的 `PID` 进程号并消除进程。下面是整个过程的完整示例。

```
# Start mo-service in the backend
./mo-service --daemon --launch ./etc/quickstart/launch.toml &> test.log &

# Find mo-service PID
ps aux | grep mo-service

[root@VM-0-10-centos ~]# ps aux | grep mo-service
root 15277 2.8 16.6 8870276 5338016 ? Sl Nov25 156:59 ./mo-service -
launch ./etc/quickstart/launch.toml
root 836740 0.0 0.0 12136 1040 pts/0 S+ 10:39 0:00 grep --color=auto mo-
service

# Kill the mo-service process
kill -9 15277
```

Tips: 如上述示例所示，使用命令 `ps aux | grep mo-service` 首先查找出 MatrixOne 运行的进程号为 `15277`， `kill -9 15277` 即表示停止进程号为 `15277` 的 MatrixOne。

接下来你可以进行下一步 - 连接 MatrixOne。

> ⚠ **注意**
>
> 如果在某一分支上构建以后，需要切换分支再进行构建，运行后则出现 panic，需要清理数据文件目录。参见安装和部署常见问题进行解决。

步骤 4：连接 MatrixOne

安装并配置 MySQL 客户端

1. 点击 MySQL Community Downloads，进入到 MySQL 客户端下载安装页面，根据你的操作系统和硬件环境，下拉选择 **Select Operating System**，再下拉选择 **Select OS Version**，按需选择下载安装包进行安装。

   Note: 建议 MySQL 客户端版本为 8.0.30 版本及以上。

2. 配置 MySQL 客户端环境变量：

   1. 打开一个新的终端，输入如下命令：

      ```
      cd ~
      sudo vim /etc/profile
      ```

   2. 回车执行上面的命令后，需要输入 root 用户密码，即你在安装 MySQL 客户端时，你在安装窗口设置的 root 密码；如果没有设置密码，则直接回车跳过即可。

   3. 输入/跳过 root 密码后，即进入了*. bash_profile*，点击键盘上的 *i* 进入 insert 状态，即可在文件下方输入如下命令：

      ```
      export PATH=/software/mysql/bin:$PATH
      ```

   4. 输入完成后，点击键盘上的 esc 退出 insert 状态，并在最下方输入 `:wq` 保存退出。

   5. 输入命令 `source /etc/profile`，回车执行，运行环境变量。

   6. 测试 MySQL 是否可用：

- 方式一：输入命令 `mysql -u root -p`，回车执行，需要 root 用户密码，显示 `mysql>` 即表示 MySQL 客户端已开启。
- 方式二：执行命令 `mysql --version`，安装成功提示：`mysql Ver 8.0.31 for Linux on x86_64 (Source distribution)`

7. MySQL 如可用，关闭当前终端，继续浏览下一章节**连接 MatrixOne 服务**。

Tips: 目前，MatrixOne 只兼容 Oracle MySQL 客户端，因此一些特性可能无法在 MariaDB、Percona 客户端下正常工作。

连接 MatrixOne

- 你可以使用 MySQL 命令行客户端来连接 MatrixOne。打开一个新的终端，直接输入以下指令：

```
mysql -h IP -P PORT -uUsername -p
```

输入完成上述命令后，终端会提示你提供用户名和密码。你可以使用我们的内置帐号：

· user: dump · password: 111

- 你也可以使用 MySQL 客户端下述命令行，输入密码，来连接 MatrixOne 服务：

```
mysql -h 127.0.0.1 -P 6001 -udump -p
Enter password:
```

目前，MatrixOne 只支持 TCP 监听。

使用 Docker 部署

本篇文档将指导你使用 Docker 部署单机版 MatrixOne。

步骤 1：下载安装 Docker

1. 点击 Get Docker，进入 Docker 的官方文档页面，根据你的操作系统，下载安装对应的 Docker。

2. 安装完成后，通过下述代码行确认 Docker 版本，验证 Docker 安装是否成功：

   ```
   docker --version
   ```

   安装成功，代码示例如下：

   ```
   Docker version 20.10.17, build 100c701
   ```

3. 在你终端里执行如下命令，启动 Docker 并查看运行状态是否成功：

   ```
   systemctl start docker
   systemctl status docker
   ```

   表示 Docker 正在运行的代码示例如下，出现 `Active: active (running)` 即表示 Docker 已经在运行中。

   ```
   docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled;
   vendor preset: disabled)
   Active: active (running) since Sat 2022-11-26 17:48:32 CST; 6s ago
   Docs: https://docs.docker.com
   Main PID: 234496 (dockerd)
   Tasks: 8
   Memory: 23.6M
   ```

步骤 2：获取 MatrixOne 镜像并启动

使用以下命令将从 Docker Hub 中拉取 MatrixOne 镜像，你可以选择稳定版本镜像，或开发版本镜像。

=== “稳定版本的镜像（0.7.0）”

```
```bash
docker pull matrixorigin/matrixone:0.7.0
docker run -d -p 6001:6001 --name matrixone matrixorigin/matrixone:0.7.0
```
```

=== “开发版本的镜像”

```
获取最新开发版本的镜像，参见[Docker Hub]
(https://hub.docker.com/r/matrixorigin/matrixone/tags)，找到最新Tag，拉取镜
像。拉取镜像代码示例如下：

```bash
docker pull matrixorigin/matrixone:nightly-commitnumber
docker run -d -p 6001:6001 --name matrixone
matrixorigin/matrixone:nightly-commitnumber
```

__Note__: 如上面代码所示，*nightly* 为标识的 Tag 版本每天都进行更新，请注意获取最新
的镜像。
```

如需挂载数据目录或配置自定义文件，参见[挂载目录到 Docker 容器]。

步骤 3：连接 MatrixOne

安装并配置 MySQL 客户端

1. 点击 [MySQL Community Downloads]，进入到 MySQL 客户端下载安装页面，根据你的操作系统和硬件环境，下拉选择 **Select Operating System**，再下拉选择 **Select OS Version**，按需选择下载安装包进行安装。

   Note: 建议 MySQL 客户端版本为 8.0.30 版本及以上。

2. 配置 MySQL 客户端环境变量：

    1. 打开一个新的终端，输入如下命令：

    ```
    cd ~
    sudo vim /etc/profile
    ```

    2. 回车执行上面的命令后，需要输入 root 用户密码，即你在安装 MySQL 客户端时，你在安装窗口设置的 root 密码；如果没有设置密码，则直接回车跳过即可。

    3. 输入/跳过 root 密码后，即进入了*. bash_profile*，点击键盘上的 *i* 进入 insert 状态，即可在文件下方输入如下命令：

    ```
    export PATH=/software/mysql/bin:$PATH
    ```

    4. 输入完成后，点击键盘上的 esc 退出 insert 状态，并在最下方输入 `:wq` 保存退出。

    5. 输入命令 `source /etc/profile`，回车执行，运行环境变量。

    6. 测试 MySQL 是否可用：

        - 方式一：输入命令 `mysql -u root -p`，回车执行，需要 root 用户密码，显示 `mysql>` 即表示 MySQL 客户端已开启。

        - 方式二：执行命令 `mysql --version`，安装成功提示：`mysql Ver 8.0.31 for Linux on x86_64 (Source distribution)`

    7. MySQL 如可用，关闭当前终端，继续浏览下一章节**连接 MatrixOne 服务**。

    Tips: 目前，MatrixOne 只兼容 Oracle MySQL 客户端，因此一些特性可能无法在 MariaDB、Percona 客户端下正常工作。

连接 MatrixOne

- 你可以使用 MySQL 命令行客户端来连接 MatrixOne。打开一个新的终端，直接输入以下指令：

```
mysql -h IP -P PORT -uUsername -p
```

输入完成上述命令后，终端会提示你提供用户名和密码。你可以使用我们的内置帐号：

- user: dump
- password: 111

- 你也可以使用 MySQL 客户端下述命令行，输入密码，来连接 MatrixOne 服务：

```
mysql -h 127.0.0.1 -P 6001 -udump -p
Enter password:
```

目前，MatrixOne 只支持 TCP 监听。

MatrixOne 的 SQL 基本操作

MatrixOne 兼容 MySQL，你可以使用 MySQL 客户端或其他方式连接 MatrixOne。参加 MySQL 兼容性和客户端连接 MatrixOne 服务。

什么是 SQL 语言?

SQL (Structured Query Language: 结构化查询语言) 是用于管理关系数据库管理系统（RDBMS）。SQL 的范围包括数据插入、查询、更新和删除，数据库模式创建和修改，以及数据访问控制。

MatrixOne 的 SQL 都分为哪几类?

SQL 语言通常按照功能划分成以下的 6 个部分：

- DDL (Data Definition Language)：数据定义语言，用来定义数据库对象，包括库、表、视图等。例如，`CREATE`，`ALTER`，和 `DROP` 等。
- DML (Data Manipulation Language)：数据修改语言，用来修改和业务相关的记录。例如，`SELECT`，`DELETE`，或 `INSERT` 等。
- DQL (Data Query Language)：数据查询语言，用来查询经过条件筛选的记录。例如，`SELECT...FROM...[WHERE]` 这种多个子句组合而成的 SQL 语句。
- TCL（Transaction Control Language）：事务控制语言，用于事务管控。例如，`COMMIT`，`ROLLBACK`，或 `SET TRANSACTION` 等。
- DCL (Data Control Language)：数据控制语言，用来分配与回收资源，创建和阐述用户与角色，授权和回收权限。例如，`CREATET ACCOUNT`，`DROP ACCOUNT`，或 `GRANT` 等。
- 其他：其他类型的管理语言在 MatrixOne 中，与数据没有直接关联，对数据库参数、资源分配的获取与修改的总称。例如，`SHOW`，`SET variable`，或 `KILL` 等。

查看、创建和删除数据库

MatrixOne 中的数据库是表的集合。

- 查看数据库列表，使用 `SHOW DATABASES` 语句：

```
SHOW DATABASES;
```

- 新建一个名称为 *dbdemo* 的数据库，使用 `CREATE DATABASE db_name [options];` 语句，例如：

```
CREATE DATABASE dbdemo;
```

或：

```
CREATE DATABASE IF NOT EXISTS dbdemo;
```

添加 `IF NOT EXISTS` 条件可以防止错误。

- 使用命名为 *dbdemo* 的数据库，使用如下语句：

```
USE dbdemo;
```

- 使用 `SHOW TABLES` 语句查看数据库中所有的表，例如：

```
SHOW TABLES FROM dbdemo;
```

- 删除数据库，使用 `DROP DATABASE` 语句：

```
DROP DATABASE dbdemo;
```

创建、查看和删除表

- 创建表，使用 `CREATE TABLE` 语句：

```
CREATE TABLE table_name column_name data_type constraint;
```

例如，要创建一个名为 *person* 的表，包括编号、名字、生日等字段，可使用以下语句：

```
CREATE TABLE person (
id INT(11),
name VARCHAR(255),
birthday DATE
);
```

- 查看新建的表，使用 `SHOW CREATE` 语句：

```
SHOW CREATE table person;
```

结果如下所示：

```
+--------+-----------------------------------------------------------------
-------------------------------------------------+
| Table | Create Table |
+--------+-----------------------------------------------------------------
-------------------------------------------------+
| person | CREATE TABLE `person` (
`id` INT DEFAULT NULL,
`name` VARCHAR(255) DEFAULT NULL,
`birthday` DATE DEFAULT NULL
) |
+--------+-----------------------------------------------------------------
-------------------------------------------------+
1 row in set (0.01 sec)
```

- 删除表，使用 `DROP TABLE` 语句：

```
DROP TABLE person;
```

增价、删出、修改表记录

通用的 DML 即增删改表记录，基本的语句为 `INSERT`， `UPDATE` 和 `DELETE`。

- 向表中插入数据，使用 `INSERT` 语句：

```
INSERT INTO person VALUES(1,'tom','20170912');
```

- 向表内插入包含部分字段数据的表记录，使用 `INSERT` 语句，例如：

```
INSERT INTO person(id,name) VALUES('2','bob');
```

- 向表内修改表记录的部分字段数据，使用 `UPDATE` 语句，例如：

```
UPDATE person SET birthday='20180808' WHERE id=2;
```

- 向表内删除部分表记录，使用 `DELETE` 语句，例如：

```
DELETE FROM person WHERE id=2;
```

> **注意**
>
> `UPDATE` 和 `DELETE` 操作如果不带 `WHERE` 过滤条件则是对全表进行操作。

查询数据

DQL 用于从一个表或多个表检索所需的数据行。

- 查看表内数据，使用 `SELECT` 语句：

```
SELECT * FROM person;
```

结果如下所示：

```
+------+------+------------+
| id | name | birthday |
+------+------+------------+
| 1 | tom | 2017-09-12 |
+------+------+------------+
1 row in set (0.00 sec)
```

- 查询某一列，使用 `SELECT` 语句加上要查询的列名：

```
SELECT name FROM person;
+------+
| name |
+------+
| tom |
+------+
1 rows in set (0.00 sec)
```

使用 `WHERE` 子句，对所有记录进行是否符合条件的筛选后再返回。例如：

```
SELECT * FROM person where id\<5;
```

结果如下所示：

```
+------+------+------------+
| id | name | birthday |
+------+------+------------+
| 1 | tom | 2017-09-12 |
+------+------+------------+
1 row in set (0.00 sec)
```

创建、授权和删除用户

如果你安装部署单机版的 MatrixOne，连接登录 MatrixOne 时，你本身便具有最高权限，你可以参考下面的内容创建用户并授权，以便体验 MatrixOne 集群的权限管理功能。

- 使用 `CREATE USER` 语句创建一个用户 *mouser*，密码为 *111*：

```
> CREATE USER mouser IDENTIFIED BY '111';
Query OK, 0 rows affected (0.10 sec)
```

- 创建角色 *role_r1* 测试是否能够创建成功：

```
> CREATE ROLE role_r1;
Query OK, 0 rows affected (0.05 sec)
```

- 将角色 *role_r1* 赋予用户 *mouser*：

```
> GRANT role_r1 to mouser;
Query OK, 0 rows affected (0.04 sec)
```

- 授权角色 *role_r1* 可以在 *dbdemo* 数据库里建表的权限：

```
> GRANT create table on database * to role_r1;
Query OK, 0 rows affected (0.03 sec)
```

- 查询用户 mouser 的权限：

```
> SHOW GRANTS for mouser@localhost;
+-------------------------------------------------------+
| Grants for mouser@localhost |
+-------------------------------------------------------+
| GRANT create table ON database * `mouser`@`localhost` |
| GRANT connect ON account `mouser`@`localhost` |
+-------------------------------------------------------+
2 rows in set (0.02 sec)
```

你可以看到你已经成功把在 *dbdemo* 数据库里建表的权限授予给了 *mouser*。

- 删除用户 *mouser*：

```
DROP USER mouser;
```

权限管理是数据库中体系庞大但是非常有用的功能，如果你想了解更多，参见[权限管理]。

数据库模式设计概述

本篇文章简要概述了 MatrixOne 的数据库模式。本篇概述主要介绍 MatrixOne 数据库相关术语和后续的数据读写示例。

关键术语 - 数据库模式

数据库模式 (Schema)：本篇文章所提到的**数据库模式**等同于逻辑对象*数据库*，与 MySQL 一样，不做区分。

数据库 Database

MatrixOne 数据库或 MatrixOne Database，为表的集合。

你可以使用 `SHOW DATABASES;` 查看 MatrixOne 所包含的默认数据库。你也可以使用 `CREATE DATABASE database_name;` 创建一个新的数据库。

表 Table

MatrixOne 所指的表或 Table，从属于 MatrixOne 的某个数据库。

表包含数据行。每行数据中的每个值都属于一个特定的列。每列都只允许单一数据类型的数据值。

其他对象

MatrixOne 支持一些和表同级的对象：

- 视图：视图是一张虚拟表，该虚拟表的结构由创建视图时的 SELECT 语句定义，MatrixOne 暂不支持物化视图。
- 临时表：临时表是数据不持久化的表。

访问控制

MatrixOne 支持基于用户或角色的访问控制。你可以通过角色或直接指向用户，从而授予用户查看、修改或删除数据对象和数据模式的权限。

更多信息，参见 MatrixOne 中的权限控制。

对象大小限制

标识符长度限制

对象	限制
数据库名称	64 字符
表名称	64 字符
列名称	64 字符
视图名称	64 字符

单个表内限制

对象	限制
列数	默认为 1017，最大可调至 4096
分区数	8192
单行大小	默认为 6MB
单行内单列大小	6MB

数据类型限制

更多关于数据类型的参考文档，参见数据类型。

行数

MatrixOne 可通过增加集群的节点数来支持任意数量的行。

创建数据库

本篇文档中介绍如何使用 SQL 来创建数据库，及创建数据库时应遵守的规则。

> **注意**
>
> 此处仅对 `CREATE DATABASE` 语句进行简单描述。更多信息，参见 CREATE DATABASE。

开始前准备

在阅读本页面之前，你需要准备以下事项：

- 了解并已经完成构建 MatrixOne 集群。
- 了解什么是数据库模式。

什么是数据库

在 MatrixOne 中数据库对象可以包含表、视图等对象。

创建数据库

可使用 `CREATE DATABASE` 语句来创建数据库。

```
CREATE DATABASE IF NOT EXISTS `modatabase`;
```

此语句会创建一个名为 *modatabase* 的数据库（如果尚不存在）。

要查看集群中的数据库，可在命令行执行一条 `SHOW DATABASES` 语句：

```
SHOW DATABASES;
```

运行结果为：

```
+--------------------+
| Database |
+--------------------+
| mo_catalog |
| system |
| system_metrics |
| mysql |
| information_schema |
| modatabase |
+--------------------+
```

数据库创建时应遵守的规则

- 你可以使用 `CREATE DATABASE` 语句来创建数据库，并且在 SQL 会话中使用 `USE \{databasename};` 语句来使用你所创建的数据库。
- 租户或用户创建新的数据库、角色或子级用户时，只赋予租户或用户必要的权限，参见 MatrixOne 中的权限控制。
- 你已经准备完毕 *modatabase* 数据库，可以将表添加到该数据库中，参见下一章节创建表。

创建表

本篇文档中介绍如何使用 SQL 来创建表。上一篇文档中介绍了创建一个名为 *modatabase* 的数据库，本篇文档我们介绍在这个数据库中创建一个表。

> **注意**
>
> 此处仅对 `CREATE TABLE` 语句进行简单描述。更多信息，参见 CREATE TABLE。

开始前准备

在阅读本页面之前，你需要准备以下事项：

- 了解并已经完成构建 MatrixOne 集群。
- 了解什么是数据库模式。
- 创建了一个数据库。

什么是表

表是 MatrixOne 数据库集群中的一种逻辑对象，它从属于某个数据库，用于保存数据。

表以行和列的形式组织数据记录，一张表至少有一列。若在表中定义了 n 个列，那么每一行数据都将拥有与这 n 个列中数据格式完全一致的字段。

命名表

创建一个有实际意义的表名称，含有关键词或编号规范的表名称，遵循命名规范，方便查找和使用。

`CREATE TABLE` 语句通常采用以下形式：

```
CREATE TABLE \{table_name} (\{elements});
```

参数描述

- {table_name}：表名。
- {elements}：以逗号分隔的表元素列表，比如列定义，主键定义等。

定义列

列从属于表，每张表都至少有一列。列通过将每行中的值分成一个个单一数据类型的小单元来为表提供结构。

列定义通常使用以下形式：

```
\{column_name} \{data_type} \{column_qualification}
```

参数描述

- {column_name}：列名。
- {data_type}：列的数据类型。
- {column_qualification}：列的限定条件。

这里介绍创建一个命名为 *NATION* 的表来存储 *modatabase* 库中的用户信息。

可以为 *NATION* 表添加一些列。

```
CREATE TABLE NATION(
N_NATIONKEY INTEGER NOT NULL,
N_NAME CHAR(25) NOT NULL,
N_REGIONKEY INTEGER NOT NULL,
N_COMMENT VARCHAR(152),
);
```

示例解释

下表将解释上述示例中的字段：

字段名	数据类型	作用	解释
N_NATIONKEY	INTEGER	民族的唯一标识	所有标识都应该是 INTEGER 类型的
N_NAME	CHAR	民族名字	民族的名称都是 char 类型，且不超过 25 字符

字段名	数据类型	作用	解释
N_REGIONKEY	INTEGER	地区区号，唯一标识	所有标识都应该是 INTEGER 类型的
N_COMMENT	VARCHAR	comment信息	varchar 类型，且不超过 152 字符

MatrixOne 支持许多其他的列数据类型，包含整数、浮点数、时间等，参见数据类型。

创建一个复杂表

创建一张 *ORDERS* 表。

```
CREATE TABLE ORDERS(
O_ORDERKEY BIGINT NOT NULL,
O_CUSTKEY INTEGER NOT NULL,
O_ORDERSTATUS CHAR(1) NOT NULL,
O_TOTALPRICE DECIMAL(15,2) NOT NULL,
O_ORDERDATE DATE NOT NULL,
O_ORDERPRIORITY CHAR(15) NOT NULL,
O_CLERK CHAR(15) NOT NULL,
O_SHIPPRIORITY INTEGER NOT NULL,
O_COMMENT VARCHAR(79) NOT NULL,
PRIMARY KEY (O_ORDERKEY)
);
```

这张表比 *NATION* 表包含更多的数据类型：

字段名	数据类型	作用	解释
O_TOTALPRICE	DECIMAL	用于标记价格	精度为 15，比例为 2，即精度代表字段数值的总位数，而比例代表小数点后有多少位，例如: decimal(5,2)，即精度为 5，比例为 2 时，其取值范围为 -999.99 到 999.99。decimal(6,1) ，即精度为 6，比例为 1 时，其取值范围为 -99999.9 到 99999.9。
O_ORDERDATE	DATE	日期值	订单产生的日期

选择主键

主键是一个或一组列，这个由所有主键列组合起来的值是数据行的唯一标识。

主键在 `CREATE TABLE` 语句中定义。主键约束要求所有受约束的列仅包含非 `NULL` 值。

一个表可以没有主键，主键也可以是非整数类型。

添加列约束

除主键约束外，MatrixOne 还支持其他的列约束，如：非空约束 NOT NULL、默认值 DEFAULT 等。

填充默认值

如需在列上设置默认值，请使用 `DEFAULT` 约束。默认值将可以使你无需指定每一列的值，就可以插入数据。

你可以将 `DEFAULT` 与支持的 SQL 函数结合使用，将默认值的计算移出应用层，从而节省应用层的资源（当然，计算所消耗的资源并不会凭空消失，只是被转移到了 MatrixOne 集群中）。常见的，希望实现数据插入时，可默认默认填充某个值的时间。这里使用一个简单例子，可使用以下语句：

```
create table t1(a int default (1), b int);
insert into t1(b) values(1), (1);
> select * from t1;
+------+------+
| a | b |
+------+------+
| 1 | 1 |
| 1 | 1 |
+------+------+
2 rows in set (0.01 sec)
```

可以看到，a 的值默认是 1。

防止重复

如果你需要防止列中出现重复值，那你可以使用 `UNIQUE` 约束。

例如，你需要确保民族标记的值唯一，可以这样改写 *NATION* 表的创建 SQL：

```
CREATE TABLE NATION(
N_NATIONKEY INTEGER NOT NULL,
N_NAME CHAR(25) NOT NULL,
N_REGIONKEY INTEGER NOT NULL,
N_COMMENT VARCHAR(152),
UNIQUE KEY (N_NATIONKEY)
);
```

如果你在 *NATION* 表中尝试插入相同的 *N_NATIONKEY*，将返回错误。

防止空值

如果你需要防止列中出现空值，那就可以使用 `NOT NULL` 约束。

还是使用民族名称来举例子，除了民族标记的值唯一，还希望民族名称不可为空，于是此处可以这样写 *NATION* 表的创建 SQL：

```
CREATE TABLE NATION(
N_NATIONKEY INTEGER NOT NULL,
N_NAME CHAR(25) NOT NULL,
N_REGIONKEY INTEGER NOT NULL,
N_COMMENT VARCHAR(152),
PRIMARY KEY (N_NATIONKEY)
);
```

执行 `SHOW TABLES` 语句

需查看 *modatabase* 数据库下的所有表，可使用 `SHOW TABLES` 语句：

```
SHOW TABLES IN `modatabase`;
```

运行结果为：

```
+----------------------+
| tables_in_modatabase |
+----------------------+
| nation |
| orders |
+----------------------+
```

创建表时应遵守的规则

命名表时应遵守的规则

- 使用完全限定的表名称（例如：`CREATE TABLE \{database_name}.\{table_name}`）。这是因为你在不指定数据库名称时，MatrixOne 将使用你 SQL 会话中的当前数据库。若你未在 SQL 会话中使用 `USE \{databasename};` 来指定数据库，MatrixOne 将会返回错误。
- 请使用有意义的表名，例如，若你需要创建一个用户表，你可以使用名称：*user*，*t_user*，*users* 等，或遵循你公司或组织的命名规范。如果你的公司或组织没有相应的命名规范，可参考表命名规范。
- 多个单词以下划线分隔，不推荐超过 32 个字符。
- 不同业务模块的表单独建立 *DATABASE*，并增加相应注释。

定义列时应遵守的规则

- 查看支持的列的数据类型。
- 查看选择主键时应遵守的规则，决定是否使用主键列。
- 查看添加列约束，决定是否添加约束到列中。
- 请使用有意义的列名，推荐你遵循公司或组织的表命名规范。如果你的公司或组织没有相应的命名规范，可参考列命名规范。

选择主键时应遵守的规则

- 在表内定义一个主键或唯一索引。
- 尽量选择有意义的列作为主键。

- 出于为性能考虑，尽量避免存储超宽表，表字段数不建议超过 60 个，建议单行的总数据大小不要超过 64K，数据长度过大字段最好拆到另外的表。
- 不推荐使用复杂的数据类型。
- 需要 `JOIN` 的字段，数据类型保障绝对一致，避免隐式转换。
- 避免在单个单调数据列上定义主键。如果你使用单个单调数据列（例如：AUTO_INCREMENT 的列）来定义主键，有可能会对写性能产生负面影响。

数据完整性约束概述

在 MatrixOne 数据库中，为了确保数据的正确性、完整性、有效性，在建表语句中，会对某些列加入限制条件，确保数据库内存储的信息遵从一定的业务规则，这些限制条件被称为约束。例如，如果 DML 语句的执行结果违反了完整性约束（Integrity Constraint），将回滚语句并返回错误消息。

完整性约束类型

MatrixOne 存在多种约束，不同的约束对于数据库行为有着不同的限制。目前支持的约束都是表级别的约束：

- NOT NULL 完整性约束：

  非空约束是指，某一列的数据不能出现空值（NULL），违反了该限制条件的数据不能被插入或更新在对应列中。在 MatrixOne 中，一张表允许有零个、一个或多个非空约束。

- UNIQUE KEY 完整性约束

  唯一键约束是指，在一张表中存的某一列或多列组合中，被插入或更新的数据行此列（或列集）的值是唯一的。在 MatrixOne 中，一张表中允许存在零个、一个或多个唯一键约束，但与其他关系型数据库不同的是，MatrixOne 的唯一键约束也必须非空。

- PRIMARY KEY 完整性约束

  主键约束是指，在一张表中存的某一列或多列组合中，每一数据行都可以由某一个键值唯一地确定且非空的，，并且主键约束在一张表中最多只能有 1 个。

- FOREIGN KEY 完整性约束

  外键约束是指，在一张表中存的某一列或多列组合中，被另一张表中的某一列或多列所引用，被引用表通常称为父表，引用表称为子表。子表引用父表的对应列的数据，只能是父表的数据或空值，这种约束被称为外键约束。

  在 MatrixOne 当前版本中，只能支持单列外键约束。

NOT NULL 完整性约束

NOT NULL 约束可用于限制一个列中不能包含 NULL 值。

语法说明

```
> column_name data_type NOT NULL;
```

你无法向包含 `NOT NULL` 约束的列中插入 `NULL` 值，或更新旧值为 `NULL`。

示例

```
create table t1(a int not null,b int);
mysql> insert into t1 values(null,1);
ERROR 3819 (HY000): constraint violation: Column 'a' cannot be null
mysql> insert into t1 values(1,null);
Query OK, 1 row affected (0.01 sec)
mysql> update t1 set a=null where a=1;
ERROR 3819 (HY000): constraint violation: Column 'a' cannot be null
```

示例解释：在上述示例中，因为 a 列存在非空约束，因此第 1 条插入语句会执行失败，第 2 条语句满足 a 列的非空约束，b 列不存在非空约束，因此可以成功插入。而更新语句因为触发了 a 列的非空约束，因此更新失败。

限制

MatrixOne 暂不支持 `alter table`，所以也不支持删除 `NOT NULL` 约束。

UNIQUE KEY 完整性约束

UNIQUE KEY 约束可用于确保将要被插入或更新的数据行的列或列组的值是唯一的，表的任意两行的某列或某个列集的值不重复，并且唯一键约束也必须非空。

语法说明

```
> column_name data_type UNIQUE KEY;
```

示例

```
create table t1(a int unique key, b int, c int, unique key(b,c));
mysql> insert into t1 values(1,1,1);
Query OK, 1 row affected (0.01 sec)
mysql> insert into t1 values(2,1,1);
ERROR 20307 (HY000): Duplicate entry '3a15013a1501' for key
'__mo_index_idx_col'
mysql> insert into t1 values(1,1,2);
ERROR 20307 (HY000): Duplicate entry '1' for key '__mo_index_idx_col'
```

示例解释：在上述示例中，存在两个唯一键约束列 a 与列 (b,c)。在插入数据时，第 2 条插入语句违反了 (b,c) 的唯一约束，与第 1 条插入值重复，因此插入失败。第 3 条插入语句违反了列 a 的约束，也因此插入失败。

限制

MatrixOne 暂不支持 `alter table`，所以也不支持删除 `UNIQUE KEY` 约束。

PRIMARY KEY 完整性约束

PRIMARY KEY 约束可用于确保表内的每一数据行都可以由某一个键值唯一地确定。 并且每个数据库表上最多只能定义一个 `PRIMARY KEY` 约束。

语法说明

```
> column_name data_type PRIMARY KEY;
```

示例

```
mysql> create table t1(a int primary key, b int, c int, primary
key(b,c));
ERROR 20301 (HY000): invalid input: more than one primary key defined
mysql> create table t2(a int, b int, c int, primary key(b,c));
Query OK, 0 rows affected (0.01 sec)

mysql> create table t3(a int, b int, c int, primary key(a));
Query OK, 0 rows affected (0.02 sec)

mysql> insert into t2 values(1,1,1);
Query OK, 1 row affected (0.02 sec)

mysql> insert into t2 values(1,1,2);
Query OK, 1 row affected (0.01 sec)

mysql> insert into t3 values(1,1,1);
Query OK, 1 row affected (0.01 sec)

mysql> insert into t3 values(2,1,1);
Query OK, 1 row affected (0.01 sec)
```

示例解释：在上述示例中，t1 包含了两组主键，因此创建失败。t2 和 t3 只有一组主键，因此可以创建。四条插入语句都没有违反约束，均可成功执行。

限制

MatrixOne 暂不支持 `alter table`，所以也不支持删除 `PRIMARY KEY` 约束。

FOREIGN KEY 完整性约束

FOREIGN KEY 约束可用于在跨表交叉引用相关数据时，保持相关数据的一致性。

定义外键时，需要遵守下列规则：

- 主表必须已经存在于数据库中，或者是当前正在创建的表。如果是后一种情况，则主表与从表是同一个表，这样的表称为自参照表，这种结构称为自参照完整性。
- 必须为主表定义主键。
- 在主表的表名后面指定列名或列名的组合。这个列或列的组合必须是主表的主键或候选键。当前 MatrixOne 仅支持单列外键约束。
- 外键中列的数目必须和主表的主键中列的数目相同。
- 外键中列的数据类型必须和主表主键中对应列的数据类型相同。

语法说明

```
> column_name data_type FOREIGN KEY;
```

示例

```
create table t1(a int primary key,b varchar(5));
create table t2(a int ,b varchar(5),c int, foreign key(c) references
t1(a));
mysql> insert into t1 values(101,'abc'),(102,'def');
Query OK, 2 rows affected (0.01 sec)

mysql> insert into t2 values(1,'zs1',101),(2,'zs2',102);
Query OK, 2 rows affected (0.01 sec)

mysql> insert into t2 values(3,'xyz',null);
Query OK, 1 row affected (0.01 sec)

mysql> insert into t2 values(3,'xxa',103);
ERROR 20101 (HY000): internal error: Cannot add or update a child row: a
foreign key constraint fails
```

示例解释：在上述示例中，t2 的 c 列只能引用 t1 中 a 列的值或空值，因此插入 t2 的前 3 行操作都能够成功插入，但是第 4 行中的 103 并不是 t1 中 a 列的某个值，违反了外键约束，因此插入失败。

限制

MatrixOne 暂不支持 `alter table`，所以也不支持删除 `FOREIGN KEY` 约束。

如何配置 MatrixOne SSL 连接

概述

本文介绍如何配置 SSL 安全连接 MatrixOne 服务器。在传送信息时，采用 SSL 连接的方式，可以避免恶意用户拦截你的流量。

配置 MatrixOne SSL 连接

新建目录存储 SSL 密钥

创建包含 SSL 密钥的目录，执行以下步骤：

1. 通过 SSH 登录 MatrixOne 服务，先确认你已安装 `mysql_ssl_rsa_setup` 工具。一般如果 MySQL 安装完成的话，`mysql_ssl_rsa_setup` 也会被一起安装。

   检查你是否安装 `mysql_ssl_rsa_setup`：如果你已安装 `mysql_ssl_rsa_setup`，在命令行工具中执行以下命令，如果没有出现下列结果，则需要重新安装 MySQL, 可以参见 install MySQL，`mysql_ssl_rsa_setup` 也将一并安装。另外你也可以通过 `whereis mysql_ssl_rsa_setup` 命令查看 `mysql_ssl_rsa_setup` 可执行文件的路径。

   ```
   [pcusername@VM-0-12-centos matrixone]$ mysql_ssl_rsa_setup
   2022-10-19 10:57:30 [ERROR] Failed to access directory pointed by --
   datadir. Please make sure that directory exists and is accessible by
   mysql_ssl_rsa_setup. Supplied value : /var/lib/mysql
   [pcusername@VM-0-12-centos matrixone]$ whereis mysql_ssl_rsa_setup
   mysql_ssl_rsa_setup: /usr/bin/mysql_ssl_rsa_setup
   /usr/share/man/man1/mysql_ssl_rsa_setup.1.gz
   ```

2. 创建一个 MatrixOne 可以访问的 SSL 密钥存储目录。例如，执行命令 `mkdir /home/user/mo_keys` 创建目录 *mo_keys*。

创建 SSL 密钥

执行以下步骤创建 SSL 密钥：

1. 运行命令创建 CA (Certificate Authority) 密钥：

```
mysql_ssl_rsa_setup --datadir=/home/user/mo_keys
```

文件夹将创建产生多个*. pem* 文件。

```
/mo_keys
├── ca-key.pem
├── ca.pem
├── client-cert.pem
├── client-key.pem
├── private_key.pem
├── public_key.pem
├── server-cert.pem
└── server-key.pem
```

2. 在 MatrixOne 目录下的 *etc/launch-tae-CN-tae-DN/cn.toml* 文件内的 `[cn.frontend]` 部分插入以下代码段：

```
[cn.frontend]
enableTls = true
tlsCertFile = "/home/user/mo_keys/server-cert.pem"
tlsKeyFile = "/home/user/mo_keys/server-key.pem"
tlsCaFile = "/home/user/mo_keys/ca.pem"
```

如果 `[cn.frontend]` 部分在 MatrixOne 系统设置文件中不存在，你可以用上述设置创建一个。

测试 SSL 配置是否成功

执行以下步骤，测试测试 SSL 配置是否成功：

1. 单机部署 MatrixOne 服务，具体步骤，参见单机部署 MatrixOne。

2. 完成上述步骤 1 中单机部署 MatriOne 服务并连接成功后，运行 `status` 命令，输出结果示例如下：

```
mysql> status
--------------
mysql Ver 8.0.28 for Linux on x86_64 (MySQL Community Server - GPL)

Connection id: 1001
Current database:
Current user: dump@0.0.0.0
SSL: Cipher in use is TLS_AES_128_GCM_SHA256
Current pager: stdout
Using outfile: ''
Using delimiter: ;
Server version: 8.0.30-MatrixOne-v0.7.0 MatrixOne
Protocol version: 10
Connection: 127.0.0.1 via TCP/IP
Client characterset: utf8mb4
Server characterset: utf8mb4
TCP port: 6002
Binary data as: Hexadecimal
--------------
```

客户端工具连接 MatrixOne 服务

MatrixOne 现在支持通过以下几种数据库客户端工具的方式连接 MatrixOne 服务：

- MySQL Client
- Navicat
- DBeaver

前期准备

已完成安装并启动 MatrixOne。

通过 MySQL Client 连接 MatrixOne 服务

1. 下载安装 MySQL Client。
2. 下载完成后，你可以使用 MySQL 命令行客户端来连接 MatrixOne 服务。

   ```
   mysql -h IP -P PORT -uUsername -p
   ```

   连接符的格式与 MySQL 格式相同，你需要提供用户名和密码。

   此处以内置帐号作为示例：

   - user: dump
   - password: 111

   ```
   mysql -h 127.0.0.1 -P 6001 -udump -p
   Enter password:
   ```

3. 连接成功提示如下：

```
Welcome to the MySQL monitor. Commands end with ; or \g. Your MySQL
connection id is 1031
Server version: 8.0.30-MatrixOne-v0.7.0 MatrixOne
Copyright (c) 2000, 2022, Oracle and/or its affiliates.

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.
```

更多关于安装部署的问题，参见部署常见问题。

通过 Navicat 连接 MatrixOne 服务

1. 由于 MatrixOne 0.7 不能完全兼容 MySQL 8.0，我们将 MatrixOne 服务器版本重置为 0.7.0 以适配 Navicat 连接。

   在启动 MatrixOne 之前，进入 MatrixOne 文件夹中的 *etc/launch-tae-CN-tae-DN/cn.toml* 文件，将 `[cn.frontend]` 行命令插入到文件中，再启动 MatrixOne。`[cn.frontend]` 行命令如下：

   ```
   [cn.frontend]
   ServerVersionPrefix = " "
   ```

   插入完成后保存文件，打开一个新的终端窗口，输入以下命令，启动 MatrixOne：

   ```
   #Launch MatrixOne (Source code method)
   ./mo-service -launch ./etc/quickstart/launch.toml
   ```

   Note: 如果你使用 Docker 的方式安装启动的 MatrixOne，并且你需要修改这个配置文件，请参考挂载目录到 Docker 容器.

2. 下载安装 Navicat。

3. 安装 Navicat 完成后，打开 Navicat，点击左上角 **Connection > MySQL**，在弹窗中填入如下参数：

   - **Connction Name**: MatrixOne

- **Host**: 127.0.0.1
- **Port**: 6001
- **User Name**: dump
- **Password**: 111
- **Save password**：勾选

4. 点击 **Save** 保存设置。

   navicat_config

5. 双击左侧数据库目录中的 **MatrixOne**，图标点亮，连接成功。

6. 连接到 MatrixOne 后，在左侧数据库目录栏，你将看到 6 个默认系统数据库：

   右侧窗口可查看有关此连接的基本信息：

通过 DBeaver 连接 MatrixOne 服务

1. 下载安装 DBeaver。

2. 安装 DBeaver 完成后，打开 DBeaver，点击左上角**连接**图标，在弹窗中选择 **MySQL**，点击 **Next**。

   dbeaver-mysql

   在 **Connect to a database** 窗口的 **Main** 区中填写如下参数：

   - **Host**: 127.0.0.1
   - **Port**: 6001
   - **Database**: MatrixOne
   - **User Name**: dump
   - **Password**: 111
   - **Save password locally**: 勾选

dbeaver-connection

3. 双击左侧目录中的 **MatrixOne**，连接 MatrixOne 服务。你可以在左侧目录树中看到默认的四个系统数据库：

   dbeaver-databases

4. 默认情况下，DBeaver 中不展示视图。如需显示完整的系统数据库，你可以右键单击 **MatrixOne**，选择 **Connection view** 并打开 **Show system objects**：

   设置完成后，你将看到 6 个系统数据库。

   dbeaver-databases-with-view

使用 JDBC 连接 MatrixOne

在 Java 中，我们可以通过 Java 代码使用 JDBC 连接器（Java Database Connectivity）连接到 MatrixOne。JDBC 是用于数据库连接的标准 API 之一，使用它我们可以轻松地运行 SQL 语句并且从数据库中获取数据。

开始前准备

使用 MatrixOne 进行 Java 数据库连接前，需要完成以下下载安装任务：

1. 已完成安装并启动 MatrixOne。
2. 下载安装 JDK 8+ version。
3. 下载安装 MySQL 客户端。
4. 下载安装 JAVA IDE，本篇文档以 IntelliJ IDEA 为例，你也可以下载其他 IDE 工具。

步骤

1. 使用 MySQL 客户端连接 MatrixOne。在 MatrixOne 新建一个名为 *test* 数据库和一个新的表 *t1*：

```
create database test;
use test;
create table t1
(
code int primary key,
title char(35)
);
```

2. 在 IDEA 中新建 Java 名称为 **testJDBC** 的项目并选择在 **Build System** 中选择 **Maven** 作为构建系统，点击 **Create**。

   JDBC create project

3. 点击 **File > Project Structure**，进入到 **Project Setting**，点选 **Library**，并点击 **+** 按钮，添加 **From Maven**。

JDBC project structure

JDBC add library

4. 输入框中输入 **mysql-connector-java** 搜索整个库，选择 **mysql:mysql-connector-java:8.0.30**，应用到本项目中。

   JDBC add driver

5. 修改 **src/main/java/org/example/Main.java** 中的默认 Java 源代码。如下面的代码示例种所示，这段代码使用连接地址和凭据创建连接。连接到 MatrixOne 后，你可以使用 Java 语言对 MatrixOne 数据库和表进行操作。

   有关如何使用 JDBC 在 MatrixOne 中开发 CRUD（创建、读取、更新、删除）应用程序的完整示例，参考 Java CRUD 示例。

```
package org.example;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;



public class Main {


private static String jdbcURL = "jdbc:mysql://127.0.0.1:6001/test";
private static String jdbcUsername = "dump";
private static String jdbcPassword = "111";

public static void main(String[] args) {


try {
Connection connection = DriverManager.getConnection(jdbcURL,
jdbcUsername, jdbcPassword);
// Do something with the Connection

} catch (SQLException ex) {
// handle any errors
System.out.println("SQLException: " + ex.getMessage());
System.out.println("SQLState: " + ex.getSQLState());
System.out.println("VendorError: " + ex.getErrorCode());
}
}
}
```

参考文档

有关 MatrixOne 对 JDBC 特性支持的完整列表，参见 [MatrixOne 的 JDBC 功能支持列表](#)。

使用 Java ORMs 连接 MatrixOne

除了使用 JDBC 连接 MatrixOne 之外，我们还可以使用对象关系映射 (ORM) 框架连接到 MySQL 数据库。在本篇文档中，介绍了如何使用 Spring Data JPA 和 MyBatis 连接到 MatrixOne。

MyBatis

MyBatis 是 SQL 映射框架，它的优点是简单易用。你可以参考 SpringBoot 和 MyBatis CRUD 示例完整教程学习如何构建一个 CRUD 应用程序。在本篇文档中，将重点介绍如何使用 MatrixOne 配置 MyBatis。

下面的示例是 **Maven** 构建系统的典型设置。

1. 在 *Pom.xml* 中添加 *MyBatis-Spring-Boot-Starter*

在 Spring Boot 上构建 MyBatis 应用程序，你需要将 *MyBatis-Spring-Boot-Starter* 模块添加到 *pom.xml* 中，*MyBatis-Spring-Boot-Starter* 模块则是在选择 Maven 项目时进行创建的。

```
\<dependency>
\<groupId>org.mybatis.spring.boot\</groupId>
\<artifactId>mybatis-spring-boot-starter\</artifactId>
\<version>2.1.4\</version>
\</dependency>
```

2. 添加配置

在 `application.properties` 中需要修改的参数如下，其余参数可以保存默认值：

- `spring.datasource.driver-class-name`： MySQL 连接器的驱动程序名称。
- `spring.datasource.url`： JDBC 连接 URL 参数。
- `spring.datasource.username`： 数据库用户名。

- `spring.datasource.password`：数据库密码。
- `mybatis.mapper-locations`：Mapper XML 配置文件的位置。

MatrixOne 中推荐配置如下：

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://127.0.0.1:6001/test?
characterSetResults=UTF-
8&continueBatchOnError=false&useServerPrepStmts=true&alwaysSendSetIsolation=
spring.datasource.username=dump
spring.datasource.password=111
mybatis.mapper-locations=classpath:mapping/*xml
```

> **注意**
>
> 需要使用推荐配置的 JDBC 连接 URL，否则将导致连接失败。

Spring Data JPA

Spring Data JPA 是 Spring 基于 ORM 框架、JPA 规范的基础上封装的一套 JPA 应用框架，可使开发者用极简的代码即可实现对数据库的访问和操作，它有助于减少样板代码，并提供了一种通过几个预定义的存储库接口之一实现基本 CRUD 操作的机制，并且它也提供了包括增删改查等在内的常用功能，且易于扩展。

Spring Data JPA 是一个强大的应用框架，它有助于减少样板代码，并提供了一种通过几个预定义的存储库接口之一实现基本 CRUD 操作的机制。你可以参考 [SpringBoot 和 Hibernate CRUD 示例]完整教程学习如何构建 CRUD 应用程序。在本篇文档中，将重点介绍如何使用 MatrixOne 连接配置 Spring JPA。

下面的示例是 **Maven** 构建系统的典型设置。

1. 在 *Pom.xml* 中添加 *spring-boot-starter-data-jpa*

在 Spring Boot 上构建 Spring Data JPA 应用程序，你需要将 *spring-boot-starter-data-jpa* 模块添加到 *pom.xml* 中，*spring-boot-starter-data-jpa* 模块则是在选择 Maven 项目时进行

创建的。

```
\<dependency>
\<groupId>org.springframework.boot\</groupId>
\<artifactId>spring-boot-starter-data-jpa\</artifactId>
\</dependency>
```

2. 添加配置

在 `application.properties` 中需要修改的参数如下，其余参数可以保存默认值：

- `spring.datasource.driver-class-name`： MySQL 连接器的驱动程序名称。
- `spring.datasource.url`： JDBC 连接 URL 参数。
- `spring.datasource.username`： 数据库用户名。
- `spring.datasource.password`： 数据库密码。
- `spring.jpa.properties.hibernate.dialect`： *SQL dialect*（即 SQL 方言）使 Hibernate 为所选数据库生成更好的 SQL。MatrixOne 当前仅支持 `org.hibernate.dialect.MySQLDialect`。
- `spring.jpa.hibernate.ddl-auto`： `spring.jpa.hibernate.ddl-auto` 属性采用一个枚举，该枚举以更可控的方式控制模式生成。可能的选项和效果如下表所示。MatrixOne 当前仅支持 *none* 和 *validate*。

选项	效果
none	无数据库架构初始化
create	在应用程序启动时删除并创建模式。使用此选项，每次启动时你所有的数据都会消失。
create-drop	在启动时创建模式并在上下文关闭时销毁模式。可用于单元测试。
validate	仅检查模式是否与实体匹配。如果模式不匹配，则应用程序启动将失败。不更改数据库。

选项	效果
update	仅在必要时更新模式。例如，如果在实体中添加了一个新字段，那么它将简单地为新列更改表，而不会破坏数据。

MatrixOne 中推荐配置如下：

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://127.0.0.1:6001/test?
characterSetResults=UTF-
8&continueBatchOnError=false&useServerPrepStmts=true&alwaysSendSetIsolation=
spring.datasource.username=dump
spring.datasource.password=111
spring.jpa.properties.hibernate.dialect =
org.hibernate.dialect.MySQLDialect
spring.jpa.hibernate.ddl-auto = validate
```

Python 连接 MatrixOne 服务

MatrixOne 支持 Python 连接，在 0.7.0 版本中，MatrixOne 支持 `pymysql` 和 `sqlalchemy` 两种驱动程序。

本篇文档将指导你了解如何通过这两个 *python* 驱动程序连接 MatrixOne。

开始前准备

- 已完成安装并启动 MatrixOne。
- 已安装 Python 3.8(or plus) version。

```
#检查 Python 版本号，确认是否安装
python3 -V
```

- 已安装 MySQL 客户端。

使用 pymysql 工具连接 MatrixOne 服务

PyMySQL 是一个纯 Python MySQL 客户端库。

1. 下载安装 pymysql 和 cryptography 工具：

```
pip3 install pymysql
pip3 install cryptography

#If you are in China mainland and have a low downloading speed, you
can speed up the download by following commands.
pip3 install pymysql -i https://pypi.tuna.tsinghua.edu.cn/simple
pip3 install cryptography -i https://pypi.tuna.tsinghua.edu.cn/simple
```

2. 使用 MySQL 客户端连接 MatrixOne。新建一个名称为 *test* 数据库：

```
mysql> create database test;
```

3. 创建一个纯文本文件 *pymysql_connect_matrixone.py* 并将代码写入文件：

```
#!/usr/bin/python3

import pymysql

# Open database connection
db = pymysql.connect(
host='127.0.0.1',
port=6001,
user='dump',
password = "111",
db='test',
)
# prepare a cursor object using cursor() method
cursor = db.cursor()

# execute SQL query using execute() method.
cursor.execute("SELECT VERSION()")

# Fetch a single row using fetchone() method.
data = cursor.fetchone()
print ("Database version : %s " % data)

# disconnect from server
db.close()
```

4. 打开一个终端，在终端内执行下面的命令：

```
> python3 pymysql_connect_matrixone.py
Database version : 8.0.30-MatrixOne-v0.7.0
```

使用 sqlalchemy 连接 MatrixOne

SQLAlchemy 是 Python SQL 工具包和对象关系映射器 (ORM)，它为应用开发人员提供了 SQL 的全部功能。

1. 下载并安装 sqlalchemy 工具，下载代码示例如下：

```
pip3 install sqlalchemy

#If you are in China mainland and have a low downloading speed, you
can speed up the download by following commands.
pip3 install sqlalchemy -i https://pypi.tuna.tsinghua.edu.cn/simple
```

2. 使用 MySQL 客户端连接 MatrixOne。新建一个名称为 *test* 数据库，并且新建一个名称为 *student* 表，然后插入两条数据：

```
mysql> create database test;
mysql> use test;
mysql> create table student (name varchar(20), age int);
mysql> insert into student values ("tom", 11), ("alice", "10");
```

3. 创建一个纯文本文件 *sqlalchemy_connect_matrixone.py* 并将代码写入文件：

```
#!/usr/bin/python3
from sqlalchemy import create_engine, text

# Open database connection
my_conn =
create_engine("mysql+mysqldb://dump:111@127.0.0.1:6001/test")

# execute SQL query using execute() method.
query=text("SELECT * FROM student LIMIT 0,10")
my_data=my_conn.execute(query)

# print SQL result
for row in my_data:
print("name:", row["name"])
print("age:", row["age"])
```

4. 打开一个终端，在终端内执行下面的命令：

```
python3 sqlalchemy_connect_matrixone.py
name: tom
age: 11
name: alice
age: 10
```

插入数据

本文档介绍如何使用 SQL 语句在 MatrixOne 中插入数据。

开始前准备

已完成单机部署 MatrixOne。

INSERT INTO 语句

`INSERT INTO` 语句有以下写法：

1. 指定要插入的列名和值：

```
INSERT INTO tbl_name (a,b,c) VALUES (1,2,3);
```

2. 如果要为表的所有列添加值，则不需要在 SQL 查询中指定列名。必须确保值的顺序与表中列的顺序相同。`INSERT INTO` 语法如下：

```
INSERT INTO tbl_name VALUES (1,2,3);
```

3. 使用 `INSERT...VALUES...` 语句可以插入多行。语句中必须包含多个用逗号分隔的值列表，值列表用圆括号括起来，并用逗号分隔。示例如下：

```
INSERT INTO tbl_name (a,b,c) VALUES(1,2,3), (4,5,6), (7,8,9);
```

数据库示例

以下是从 Northwind 示例数据库中的 **Customers** 表中选择的表示例：

```
CREATE TABLE Customers (
CustomerID INT AUTO_INCREMENT NOT NULL,
CustomerName VARCHAR(40) NOT NULL,
ContactName VARCHAR(30) NULL,
Address VARCHAR(60) NULL,
City VARCHAR(15) NULL,
PostalCode VARCHAR(10) NULL,
Country VARCHAR(15) NULL,
PRIMARY KEY (CustomerID)
);
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland

INSERT INTO 示例

下面的 SQL 语句在 **Customers** 表中插入了一条新记录：

Example

```
INSERT INTO Customers (CustomerName, ContactName, Address, City,
PostalCode, Country)
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006',
'Norway');
```

Customers 表展示出来如下所示：

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland
92	Cardinal	Tom B. Erichsen	Skagen 21	Stavanger	4006	Norway

仅在指定列中插入数据

MatrixOne 也支持使用 SQL 语句仅在特定列中插入数据。

示例

使用下面的 SQL 语句将插入一条新记录，但只插入 *CustomerName*、*City* 和 *Country* 列中的数据，同时 *CustomerID* 将自动更新：

```
INSERT INTO Customers (CustomerName, City, Country)
VALUES ('Cardinal', 'Stavanger', 'Norway');
```

Customers 表展示出来如下所示：

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland
92	Cardinal	null	null	Stavanger	null	Norway

INSERT INTO...SELECT

使用 `INSERT INTO SELECT`，你可以从 `SELECT` 语句的结果中快速插入多行到表中，`SELECT` 语句可以从一个或多个表中进行选择。`INSERT INTO SELECT` 语句要求源表和目标表中的数据类型匹配。

INSERT INTO SELECT 语法解释

从一个表复制所有列到另一个表：

```
INSERT INTO *table2*
SELECT * FROM *table1
*WHERE *condition*;
```

只从一个表复制一些列到另一个表：

```
INSERT INTO *table2* (*column1*, *column2*, *column3*, ...)
SELECT *column1*, *column2*, *column3*, ...
FROM *table1*
WHERE *condition*;
```

Northwind 数据库示例

以下是从 Northwind 示例数据库中的表中选择的表示例：

```
CREATE TABLE Customers (
CustomerID INT AUTO_INCREMENT NOT NULL,
CustomerName VARCHAR(40) NOT NULL,
ContactName VARCHAR(30) NULL,
Address VARCHAR(60) NULL,
City VARCHAR(15) NULL,
PostalCode VARCHAR(10) NULL,
Country VARCHAR(15) NULL,
PRIMARY KEY (CustomerID)
);
CREATE TABLE Suppliers (
SupplierID INT AUTO_INCREMENT NOT NULL,
SupplierName VARCHAR(40) NOT NULL,
ContactName VARCHAR(30) NULL,
Address VARCHAR(60) NULL,
City VARCHAR(15) NULL,
PostalCode VARCHAR(10) NULL,
Country VARCHAR(15) NULL,
PRIMARY KEY (SupplierID)
);
```

Customers 表展示出来如下所示：

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

Suppliers 表展示出来如下所示：

SupplierID	SupplierName	ContactName	Address	City	PostalCode	Country
1	Exotic Liquid	Charlotte Cooper	49 Gilbert St.	Londona	EC1 4SD	UK
2	New Orleans Cajun	Shelley Burke	P.O. Box	New	70117	USA

SupplierID	SupplierName	ContactName	Address	City	PostalCode	Country
	Delights		78934	Orleans		
3	Grandma Kelly's Homestead	Regina Murphy	707 Oxford Rd.	Ann Arbor	48104	USA

示例

下面的 SQL 语句将 **Suppliers** 复制到 **Customers** 中，同时未填充数据的列将填充为 `NULL`：

```
INSERT INTO Customers (CustomerName, City, Country)
SELECT SupplierName, City, Country FROM Suppliers;
```

Customers 表展示出来如下所示：

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Exotic Liquid	null	null	Londona	null	UK
5	New Orleans Cajun Delights	null	null	New Orleans	null	USA
6	Grandma Kelly's Homestead	null	null	Ann Arbor	null	USA

限制

MatrixOne 暂不支持 `INSERT ... ON DUPLICATE KEY UPDATE` 语句。你需要使用 `UPDATE` 执行此操作。

批量导入概述

批量导入是将大量行插入至 MatrixOne 数据表中最快的方法。MatrixOne 支持从本地文件系统或 *S3 对象存储服务*批量导入 *csv* 文件、*jsonline* 文件。

MatrixOne 支持导入不同的数据格式

根据*数据文件类型不同*的情况，MatrixOne 支持导入*. csv* 和*. jl* 格式。

- 一种是支持导入*. csv* 格式的数据，具体导入方式可以参考导入*. csv* 格式数据。
- 一种是支持导入*. jl* 格式的数据，即 jsonlines 格式，具体导入方式可以参考导入 jsonlines 数据。

MatrixOne 支持从不同存储位置进行导入

根据*数据存储位置不同*的情况，MatrixOne 支持从*本地进行导入*和*从对象存储服务（Simple Storage Service, S3）导入*。

- 从本地导入数据的方式，参考从本地导入*. csv* 格式数据或从本地导入 jsonlines 数据。
- 从 S3 导入数据的方式，参考从 S3 读取数据并导入 MatrixOne。

MatrixOne 支持并行加载数据文件

数据文件较大时，为了提升加载速度，MatrixOne 也支持并行加载

导入 *csv* 格式数据

本篇文档将指导你在 MySQL 客户端启动 MatrixOne 时如何完成 *csv* 格式数据导入。

语法结构

- 场景一：数据文件与 MatrixOne 服务器在同一台机器上：

```
LOAD DATA
INFILE 'file_name'
INTO TABLE tbl_name
[{FIELDS | COLUMNS}
[TERMINATED BY 'string']
[[OPTIONALLY] ENCLOSED BY 'char']
]
[LINES
[STARTING BY 'string']
[TERMINATED BY 'string']
]
[IGNORE number {LINES | ROWS}]
[PARALLEL {'TRUE' | 'FALSE'}]
```

- 场景二：数据文件与 MatrixOne 服务器在不同的机器上：

```
LOAD DATA LOCAL
INFILE 'file_name'
INTO TABLE tbl_name
[{FIELDS | COLUMNS}
[TERMINATED BY 'string']
[[OPTIONALLY] ENCLOSED BY 'char']
]
[LINES
[STARTING BY 'string']
[TERMINATED BY 'string']
]
[IGNORE number {LINES | ROWS}]
[PARALLEL {'TRUE' | 'FALSE'}]
```

开始前准备

已完成单机部署 MatrixOne。

MySQL Client 中使用 `Load data` 命令导入数据

你可以使用 `Load Data` 从大数据文件中导入数据，本章将介绍如何导入 *csv* 格式文件。

Note: *csv*（逗号分隔值）文件是一种特殊的文件类型，可在 Excel 中创建或编辑，*csv* 文件不是采用多列的形式存储信息，而是使用逗号分隔的形式存储信息。

步骤

数据文件与 MatrixOne 服务器在同一台机器上

1. 在 MatrixOne 中执行 `Load Data` 之前，需要提前在 MatrixOne 中创建完成数据表。

2. 启动 MySQL 客户端，连接 MatrixOne：

   ```
   mysql -h 127.0.0.1 -P 6001 -udump -p111
   ```

   Note: 如果你的数据文件与 MatrixOne 服务器在不同的机器上，即数据文件在你所使用的客户端机器上时，那么你连接 MatrixOne 服务主机需要使用命令行：`mysql -h <mo-host-ip> -P 6001 -udump -p111 --local-infile`。

3. 在 MySQL 客户端对对应的文件路径执行 `LOAD DATA`：

   ```
   mysql> LOAD DATA INFILE '/tmp/xxx.csv'
   INTO TABLE table_name
   FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED
   BY "\r\n";
   ```

数据文件与 MatrixOne 服务器在不同的机器上

1. 在 MatrixOne 中执行 `LOAD DATA LOCAL` 之前，需要提前在 MatrixOne 中创建完成数据表。

2. 启动 MySQL 客户端，连接 MatrixOne：

```
mysql -h <mo-host-ip> -P 6001 -udump -p111 --local-infile
```

3. 在 MySQL 客户端对对应的文件路径执行 `LOAD DATA LOCAL`：

```
mysql> LOAD DATA LOCAL INFILE '/tmp/xxx.csv'
INTO TABLE table_name
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED
BY "\r\n";
```

示例：使用 *docker* 启动 MatrixOne 执行 `Load data`

如果你通过 **Docker** 安装 MatrixOne，那么文件默认存储在 **Docker** 镜像中。如果你需要将文件存储在本地目录，你需要先将本地目录挂载到容器。

在以下示例中，本地文件系统路径 `~/tmp/docker_loaddata_demo/` 挂载到 MatrixOne Docker 镜像，并映射到 Docker 容器内的 `/ssb-dbgen-path` 目录。本篇示例将指导你使用 Docker 加载数据。

1. 下载数据集，并且将数据集存储到本地 *~/tmp/docker_loaddata_demo/* 路径下：

```
cd ~/tmp/docker_loaddata_demo/
wget https://community-shared-data-1308875761.cos.ap-
beijing.myqcloud.com/lineorder_flat.tar.bz2
```

2. 解压数据集：

```
tar -jxvf lineorder_flat.tar.bz2
```

3. 使用 Docker 启动 MatrixOne，启动时将存放了数据文件的目录 *~/tmp/docker_loaddata_demo/* 挂载到容器的某个目录下，这里容器目录以 */ssb-dbgen-path* 为例：

```
sudo docker run --name matrixone --privileged -d -p 6001:6001 -v
~/tmp/docker_loaddata_demo/:/ssb-dbgen-path:rw
matrixorigin/matrixone:0.6.0
```

4. 连接 MatrixOne 服务：

```
mysql -h 127.0.0.1 -P 6001 -udump -p111
```

Note: 如果你的数据文件与 MatrixOne 服务器在不同的机器上，即数据文件在你所使用的客户端机器上时，那么你连接 MatrixOne 服务主机需要使用命令行：`mysql -h <mo-host-ip> -P 6001 -udump -p111 --local-infile`；并且导入的命令行需要使用 `LOAD DATA LOCAL INFILE` 语法。

5. 在 MatrixOne 中新建表 *lineorder_flat*，并且将数据集导入至 MatriOne：

```
mysql> create database if not exists ssb;
mysql> use ssb;
mysql> drop table if exists lineorder_flat;
mysql> CREATE TABLE lineorder_flat(
LO_ORDERKEY bigint key,
LO_LINENUMBER int,
LO_CUSTKEY int,
LO_PARTKEY int,
LO_SUPPKEY int,
LO_ORDERDATE date,
LO_ORDERPRIORITY char(15),
LO_SHIPPRIORITY tinyint,
LO_QUANTITY double,
LO_EXTENDEDPRICE double,
LO_ORDTOTALPRICE double,
LO_DISCOUNT double,
LO_REVENUE int unsigned,
LO_SUPPLYCOST int unsigned,
LO_TAX double,
LO_COMMITDATE date,
LO_SHIPMODE char(10),
C_NAME varchar(25),
C_ADDRESS varchar(25),
C_CITY char(10),
C_NATION char(15),
C_REGION char(12),
C_PHONE char(15),
C_MKTSEGMENT char(10),
S_NAME char(25),
S_ADDRESS varchar(25),
S_CITY char(10),
S_NATION char(15),
S_REGION char(12),
S_PHONE char(15),
P_NAME varchar(22),
P_MFGR char(6),
P_CATEGORY char(7),
P_BRAND char(9),
P_COLOR varchar(11),
P_TYPE varchar(25),
P_SIZE int,
P_CONTAINER char(10)
);
mysql> load data infile '/ssb-dbgen-path/lineorder_flat.tbl' into
```

```
table lineorder_flat FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY
'"' LINES TERMINATED BY '\n';
```

6. 导入成功后，可以使用 SQL 语句查看导入数据的行数：

```
select count(*) from lineorder_flat;
/*
expected results:
*/
+----------+
| count(*) |
+----------+
| 10272594 |
+----------+
```

限制

加载 `csv` 格式支持 JSON 类型，但是需要确保 JSON 内不含有字段终止符号，如果 JSON 内含有字段终止符号，那么 JSON 需要用双引号包裹起来。例如：

- 正确示例：`"{"a":1, "b":2}", 2`
- 错误示例：`{"a":1, "b":2}, 2`

导入 JSONLines 数据

本篇文档将指导你如何将 JSONLines 格式数据（即 *jl* 或 *jsonl* 文件）导入 MatrixOne。

有关 JSONLines 格式

JSON（JavaScript Object Notation）是一种轻量级的数据交换格式。你可以参见官方文档获取更多有关 JSON 的信息。

JSONLines 文本格式，也称为换行符分隔的 JSON，它是一种更为方便存储结构化的数据格式，可以一次处理一条记录。它每一行都是完整、合法的 JSON 值；它采用 `\n` 作为行分隔符。JSONLines 的每一行都是独立的，因此行的开头或结尾不需要逗号。JSONLines 的全部内容也不需要用 `[]` 或 `\{}` 括起来。

JSONLines 对于数据流来说更为友好。因为每一个新的行意味着一个单独的条目，因此 JSON 行格式的文件可以流式传输。它不需要自定义解析器。只需读取一行，解析为 JSON，再读取一行，解析为 JSON，一直到完成。

JSONLines 格式有以下三个要求：

- **UTF-8 编码**：JSON 允许仅使用 ASCII 转义序列对 Unicode 字符串进行编码，但是在文本编辑器中，这些转义难以阅读。JSON Lines 文件的作者可以选择转义字符来处理纯 ASCII 文件。

JSON 允许仅用 ASCII 转义序列编码 Unicode 字符串，但是当在文本编辑器中查看时，这些转义将很难阅读。JSON Lines 文件的作者可以选择转义字符来处理纯 ASCII 文件。

- **每行都是一个合法的 JSON 值**：最常见的值是对象或数组，任何 JSON 值都是合法的。
- **行分隔符为 `\n`**：由于在解析 JSON 值时会隐式忽略周围的空格在支持行分隔符 `\n` 的同时也支持 “\r\n”。

对于 MatrixOne 有效的 JSONLines 格式

JSONLines 格式只需要每一行都有一个有效的 JSON 值。但 MatrixOne 需要更结构化的 JSONLines 格式，在 MatrixOne 中只允许具有相同类型值和普通结构的 JSON 对象或 JSON 数组。如果您的 JSONLines 文件有嵌套结构，MatrixOne 暂时不支持加载它。

一个有效对象 JSONLines 示例：

```
\{"id":1,"father":"Mark","mother":"Charlotte"}
\{"id":2,"father":"John","mother":"Ann"}
\{"id":3,"father":"Bob","mother":"Monika"}
```

无效对象 JSONLines 示例（具有嵌套结构）：

```
\{"id":1,"father":"Mark","mother":"Charlotte","children":["Tom"]}
\{"id":2,"father":"John","mother":"Ann","children":
["Jessika","Antony","Jack"]}
\{"id":3,"father":"Bob","mother":"Monika","children":["Jerry","Karol"]}
```

一个有效数组 JSONLines 示例，它更像是 *csv* 格式。

```
["Name", "Session", "Score", "Completed"]
["Gilbert", "2013", 24, true]
["Alexa", "2013", 29, true]
["May", "2012B", 14, false]
["Deloise", "2012A", 19, true]
```

无效数组 JSONLines 示例（无效原因是因为数据类型和列号不匹配）：

```
["Gilbert", "2013", 24, true, 100]
["Alexa", "2013", "twenty nine", true]
["May", "2012B", 14, "no"]
["Deloise", "2012A", 19, true, 40]
```

语法结构

- 数据文件与 MatrixOne 服务器在同一台机器上：

```
LOAD DATA INFILE
\{'filepath'='FILEPATH', 'compression'='COMPRESSION_FORMAT',
'format'='FILE_FORMAT', 'jsondata'='object'/'array'} INTO TABLE
table_name [IGNORE x LINES/ROWS]
[PARALLEL \{'TRUE' | 'FALSE'}];
```

- 数据文件与 MatrixOne 服务器在不同的机器上：

```
LOAD DATA LOCAL INFILE
\{'filepath'='FILEPATH', 'compression'='COMPRESSION_FORMAT',
'format'='FILE_FORMAT', 'jsondata'='object'/'array'} INTO TABLE
table_name [IGNORE x LINES/ROWS]
[PARALLEL \{'TRUE' | 'FALSE'}];
```

参数说明

参数	值	必须/可选	描述
filepath	String	必须	文件路径
compression	auto/none/bz2/gzip/lz4	可选	压缩格式
format	csv/jsonline	可选	加载文件格式，默认 *.csv*
jsondata	object/array	可选	JSON 数据格式。如果 `format` 为 *jsonline*，则**必须**指定 *jsondata*
table_name	String	必须	需加载数据到表的表名称
x	Number	可选	加载时要忽略的行

JSONLines 格式数据的 DDL 指南

在将 JSONLines 数据加载到 MatrixOne 之前，你需要先创建一个表。

由于 JSON 数据类型与 MatrixOne 的数据类型不同，参见下表，可以查看 JSON 数据类型对应到 MatrixOne 中时的数据类型：

JSON 类型	MatrixOne 中的数据类型
String	VARCHAR (定长字符串)
String	TEXT (长文本数据)
String	DATETIME or TIMESTAMP (格式为 “YYYY-MM-DD HH:MM .XXXXXX”)
String	DATE (格式为 “YYYY-MM-DD”)
String	TIME (格式为 “HH-MM-SS.XXXXXX”)

JSON 类型	MatrixOne 中的数据类型
Number	INT (整数)
Number	FLOAT 或 DOUBLE (浮点数)
Boolean	BOOL(true/false)
Object	Json 类型
Array	Json 类型
Null	支持所有类型

例如，你可以先试用 SQL 语句为 JSONLines 格式文件先创建一个数据表，如下所示：

```
mysql> create table t1 (name varchar(100), session varchar(100), score
int, completed bool);
```

```
["Name", "Session", "Score", "Completed"]
["Gilbert", "2013", 24, true]
["Alexa", "2013", 29, true]
["May", "2012B", 14, false]
["Deloise", "2012A", 19, true]
```

示例

以下代码段是将 JSONLines 文件加载到 MatrixOne 的完整 SQL 示例。

```
#Load a BZIP2 compressed jsonline object file
load data infile \{'filepath'='data.bzip2',
'compression'='bz2','format'='jsonline','jsondata'='object'} into table
db.a

#Load a plain jsonline array file
load data infile \{'filepath'='data.jl',
'format'='jsonline','jsondata'='array'} into table db.a

#Load a gzip compressed jsonline array file and ignore the first line
load data infile \{'filepath'='data.jl.gz',
'compression'='gzip','format'='jsonline','jsondata'='array'} into table
db.a ignore 1 lines;
```

教程示例

在本教程中将指导你如何加载两个具有对象和数组 json 格式的 jsonline 文件。

Note: 本教程中，数据文件与 MatrixOne 服务器在同一台机器上。如果数据文件与 MatrixOne 服务器在不同的机器上，也可以使用 `Load Data` 进行数据导入。

1. 准备数据。你也可以下载使用我们准备好的 *jl* 文件。

   - 示例数据 1：*jsonline_object.jl*
   - 示例数据 2：*jsonline_array.jl*

2. 打开终端，进入到 *jl* 文件所在目录，输入下面的命令行，显示文件内的具体内容：

```
> cd /$filepath
> head jsonline_object.jl
\{"col1":true,"col2":1,"col3":"var","col4":"2020-09-07","col5":"2020-
09-07 00:00:00","col6":"2020-09-07
00:00:00","col7":"18","col8":121.11}
\{"col1":"true","col2":"1","col3":"var","col4":"2020-09-
07","col5":"2020-09-07 00:00:00","col6":"2020-09-07
00:00:00","col7":"18","col8":"121.11"}
\{"col6":"2020-09-07
00:00:00","col7":"18","col8":"121.11","col4":"2020-09-
07","col5":"2020-09-07
00:00:00","col1":"true","col2":"1","col3":"var"}
\{"col2":1,"col3":"var","col1":true,"col6":"2020-09-07
00:00:00","col7":"18","col4":"2020-09-07","col5":"2020-09-07
00:00:00","col8":121.11}
> head jsonline_array.jl
[true,1,"var","2020-09-07","2020-09-07 00:00:00","2020-09-07
00:00:00","18",121.11]
["true","1","var","2020-09-07","2020-09-07 00:00:00","2020-09-07
00:00:00","18","121.11"]
```

3. 启动 MySQL 客户端，连接到 MatrixOne。

```
mysql -h 127.0.0.1 -P 6001 -udump -p111
```

Note: 如果你的数据文件与 MatrixOne 服务器在不同的机器上，即数据文件在你所使用的客户端机器上时，那么你连接 MatrixOne 服务主机需要使用命令行：

`mysql -h \<mo-host-ip> -P \<mo-host-ip> -udump -p111 --local-infile`；并且导入的命令行需要使用 `LOAD DATA LOCAL INFILE` 语法。

4. 在 MatrixOne 建表：

```
create database db1;
use db1;
drop table if exists t1;
create table t1(col1 bool,col2 int,col3 varchar, col4 date,col5
datetime,col6 timestamp,col7 decimal,col8 float);
drop table if exists t2;
create table t2(col1 bool,col2 int,col3 varchar, col4 date,col5
datetime,col6 timestamp,col7 decimal,col8 float);
```

5. 在 MySQL 客户端对对应的文件路径执行 `LOAD DATA`，导入 *jsonline_object.jl* 和 *jsonline_array.jl* 文件：

```
load data infile \
{'filepath'='$filepath/jsonline_object.jl','format'='jsonline','jsondata
into table t1;
load data infile \
{'filepath'='$filepath/jsonline_array.jl','format'='jsonline','jsondata'
into table t2;
```

6. 导入成功后，使用如下 SQL 语句查看导入结果：

```
select * from t1;
col1 col2 col3 col4 col5 col6 col7 col8
true 1 var 2020-09-07 2020-09-07 00:00:00 2020-09-07 00:00:00 18
121.11
true 1 var 2020-09-07 2020-09-07 00:00:00 2020-09-07 00:00:00 18
121.11
true 1 var 2020-09-07 2020-09-07 00:00:00 2020-09-07 00:00:00 18
121.11
true 1 var 2020-09-07 2020-09-07 00:00:00 2020-09-07 00:00:00 18
121.11
```

> **注意**
>
> 如果您使用 Docker 启动 MatrixOne，当你需要导入 JSONline 文件时，请确保你已将数据目录挂载到容器。你也可以查看导入 *csv* 格式数据，了解如何使用 Docker 挂载数据。

从 S3 对象存储服务读取数据并导入 MatrixOne

概述

S3（Simple Storage Service）对象存储是指亚马逊的简单存储服务。你还可以使用与 S3 兼容的对象存储来存储几乎任何类型和大小的数据，包括数据湖、云原生应用程序和移动应用程序。如果你不熟悉 S3 对象服务，你可以在 AWS 中查找一些基本介绍。

AWS S3 十多年来一直非常成功，因此它成为了对象存储的标准。因此几乎所有主流公有云厂商都提供了兼容 S3 的对象存储服务。

MatrixOne 支持将文件从 S3 兼容的对象存储服务加载到数据库中。MatrixOne 支持 AWS 和国内主流云厂商（阿里云、腾讯云）。

在 MatrixOne 中，有两种方法可以从 S3 兼容的对象存储中导入数据：

- 使用带有 s3option 的 `Load data` 将文件加载到 MatrixOne 中。此方法会将数据加载到 MatrixOne 中，所有接下来的查询都将在 MatrixOne 中进行。
- 创建一个带有 s3option 映射到 S3 文件的 “外部表”，并直接查询这个外部表。该方法允许通过 S3 兼容的对象存储服务进行数据访问；每个查询的网络延迟都将被计算在内。

方式 1: `LOAD DATA`

语法结构

```
LOAD DATA
| URL s3options \{"endpoint"='\<string>', "access_key_id"='\<string>',
"secret_access_key"='\<string>', "bucket"='\<string>', "role_arn"='xxxx',
"external_id"='yyy', "filepath"='\<string>', "region"='\<string>',
"compression"='\<string>'}
INTO TABLE tbl_name
[\{FIELDS | COLUMNS}
[TERMINATED BY 'string']
[[OPTIONALLY] ENCLOSED BY 'char']
]
[IGNORE number \{LINES | ROWS}]
[PARALLEL \{'TRUE' | 'FALSE'}]
```

参数说明

参数	描述
endpoint	可以连接到对象存储服务的 URL。例如：s3.us-west-2.amazonaws.com
access_key_id	Access key ID
secret_access_key	Secret access key
bucket	S3 需要访问的桶
role_arn	
external_id	
filepath	相对文件路径。 /files/*.csv 支持正则表达式。
region	对象存储服务区域
compression	S3 文件的压缩格式。如果为空或 “none”，，则表示未压缩的文件。支持的字段或压缩格式为“auto”、“none”、“gzip”、“bz2”和“lz4”。

其他参数与通用 `LOAD DATA` 参数相同，更多信息，参见 LOAD DATA。

语法示例：

```
# LOAD a csv file from AWS S3 us-east-1 region, test-load-mo bucket,
without compression
LOAD DATA URL s3option\{"endpoint"='s3.us-east-1.amazonaws.com',
"access_key_id"='XXXXXX', "secret_access_key"='XXXXXX', "bucket"='test-
load-mo', "filepath"='test.csv', "region"='us-east-1',
"compression"='none'} INTO TABLE t1 FIELDS TERMINATED BY ',' ENCLOSED BY
'\"' LINES TERMINATED BY '\n';

# LOAD all csv files from Alibaba Cloud OSS Shanghai region, test-load-
data bucket, without compression
LOAD DATA URL s3option\{"endpoint"='oss-cn-shanghai.aliyuncs.com',
"access_key_id"='XXXXXX', "secret_access_key"='XXXXXX', "bucket"='test-
load-data', "filepath"='/test/*.csv', "region"='oss-cn-shanghai',
"compression"='none'} INTO TABLE t1 FIELDS TERMINATED BY ',' ENCLOSED BY
'\"' LINES TERMINATED BY '\n';

# LOAD a csv file from Tencent Cloud COS Shanghai region, test-1252279971
bucket, without bz2 compression
LOAD DATA URL s3option\{"endpoint"='cos.ap-shanghai.myqcloud.com',
"access_key_id"='XXXXXX', "secret_access_key"='XXXXXX', "bucket"='test-
1252279971', "filepath"='test.csv.bz2', "region"='ap-shanghai',
"compression"='bz2'} INTO TABLE t1 FIELDS TERMINATED BY ',' ENCLOSED BY
'\"' LINES TERMINATED BY '\n';
```

教程：从 AWS S3 加载文件

本教程中将指导你完成从 AWS S3 加载**. csv** 文件的过程。

如果你已经拥有一个 AWS 账户并且已经在你的 S3 服务中准备好数据文件，那么请继续阅读本教程章节。

如果你还没有准备好数据文件，请先注册并上传你的数据文件；你也可以查看 AWS S3 [官方教程](#)。如果你的数据文件想要上传到阿里云 OSS 或者腾讯云 COS 上，那么操作流程与 AWS S3 类似。

> **注意**
>
> 由于帐户隐私，此代码示例不会显示帐户信息，例如 `access_key_id` 和 `secret_access_key`。 你可以阅读本文档以了解主要步骤；具体数据和账户信息将不会显示。

1. 下载数据文件。进入 **AWS S3 > buckets**，创建一个具有公共访问权限的存储桶 **test-loading** 并上传文件 *char_varchar_1.csv*。

   public block

2. 获取或创建你的 AWS Access key。输入 **Your Account Name > Security Credentials**，获取你现有的访问密钥或创建一个新的访问密钥。

   Access Key

   你可以从下载的凭据或此网页中获取 `Access key` 和 `Secret access key`。

   Retrieve Access Key

3. 启动 MySQL 客户端，在 MatrixOne 中创建表，SQL 示例如下：

```
create database db;
use db;
drop table if exists t1;
create table t1(col1 char(225), col2 varchar(225), col3 text, col4
varchar(225));
```

4. 将文件导入 MatrixOne：

```
LOAD DATA URL s3option\{"endpoint"='s3.us-east-1.amazonaws.com',
"access_key_id"='XXXXXX', "secret_access_key"='XXXXXX',
"bucket"='test-loading', "filepath"='char_varchar_1.csv',
"region"='us-east-1', "compression"='none'} INTO TABLE t1;
```

5. 导入完成后，你可以运行 SQL 语句检查文件导入是否成功：

```
mysql> select * from t1;
+-----------+-----------+-----------+-----------+
| col1 | col2 | col3 | col4 |
+-----------+-----------+-----------+-----------+
| a | b | c | d |
| a | b | c | d |
| 'a' | 'b' | 'c' | 'd' |
| 'a' | 'b' | 'c' | 'd' |
| aa,aa | bb,bb | cc,cc | dd,dd |
| aa, | bb, | cc, | dd, |
| aa,,,aa | bb,,,bb | cc,,,cc | dd,,,dd |
| aa',',,aa | bb',',,bb | cc',',,cc | dd',',,dd |
| aa"aa | bb"bb | cc"cc | dd"dd |
| aa"aa | bb"bb | cc"cc | dd"dd |
| aa"aa | bb"bb | cc"cc | dd"dd |
| aa""aa | bb""bb | cc""cc | dd""dd |
| aa""aa | bb""bb | cc""cc | dd""dd |
| aa",aa | bb",bb | cc",cc | dd",dd |
| aa"",aa | bb"",bb | cc"",cc | dd"",dd |
| | | | |
| | | | |
| NULL | NULL | NULL | NULL |
| | | | |
| " | " | " | " |
| "" | "" | "" | "" |
+-----------+-----------+-----------+-----------+
21 rows in set (0.03 sec)
```

方式 2：指定 S3 文件到外部表

语法结构

```
create external table t(...) URL s3option\{"endpoint"='\<string>',
"access_key_id"='\<string>', "secret_access_key"='\<string>', "bucket"='\
<string>', "filepath"='\<string>', "region"='\<string>', "compression"='\
<string>'}
[\{FIELDS | COLUMNS}
[TERMINATED BY 'string']
[[OPTIONALLY] ENCLOSED BY 'char']
]
[IGNORE number \{LINES | ROWS}];
```

> **注意**
>
> MatrixOne 当前仅支持对外部表进行 `select`，暂不支持 `Delete`，`insert`，`update`。

参数说明

参数	描述
endpoint	可以连接到对象存储服务的 URL。例如：s3.us-west-2.amazonaws.com
access_key_id	Access key ID
secret_access_key	Secret access key
bucket	S3 需要访问的桶
filepath	相对文件路径。 /files/*.csv 支持正则表达式。
region	对象存储服务区域
compression	S3 文件的压缩格式。如果为空或 “none”，，则表示未压缩的文件。支持的字段或压缩格式为“auto”、“none”、“gzip”、“bz2”和“lz4”。

其他参数与通用 `LOAD DATA` 参数相同，更多信息，参见 LOAD DATA。

有关外部表的更多信息，参见创建外部表。

语法示例：

```
## Create a external table for a .csv file from AWS S3
create external table t1(col1 char(225)) url s3option\{"endpoint"='s3.us-east-1.amazonaws.com', "access_key_id"='XXXXXX', "secret_access_key"='XXXXXX', "bucket"='test-loading', "filepath"='test.csv', "region"='us-east-1', "compression"='none'} fields terminated by ',' enclosed by '\"' lines terminated by '\n';

## Create a external table for a .csv file compressed with BZIP2 from Tencent Cloud
create external table t1(col1 char(225)) url s3option\{"endpoint"='cos.ap-shanghai.myqcloud.com', "access_key_id"='XXXXXX', "secret_access_key"='XXXXXX', "bucket"='test-1252279971', "filepath"='test.csv.bz2', "region"='ap-shanghai', "compression"='bz2'} fields terminated by ',' enclosed by '\"' lines terminated by '\n' ignore 1 lines;
```

教程：使用 S3 文件创建外部表

本教程将指导你完成使用来自 AWS S3 的**. csv** 文件创建外部表的整个过程。

> **注意**
>
> 由于帐户隐私，此代码示例不会显示帐户信息，例如 `access_key_id` 和 `secret_access_key` 。 你可以阅读本文档以了解主要步骤；具体数据和账户信息将不会显示。

1. 下载数据文件。进入 **AWS S3 > buckets**，创建一个具有公共访问权限的存储桶 **test-loading** 并上传文件 *char_varchar_1.csv*。

   public block

2. 获取或创建你的 AWS Access key。输入 **Your Account Name > Security Credentials**，获取你现有的访问密钥或创建一个新的访问密钥。

Access Key

你可以从下载的凭据或此网页中获取 `Access key` 和 `Secret access key`。

Retrieve Access Key

3. 启动 MySQL 客户端，指定 S3 文件到外部表：

```
create database db;
use db;
drop table if exists t1;
create external table t1(col1 char(225), col2 varchar(225), col3 text,
col4 varchar(225)) url s3option\{"endpoint"='s3.us-east-
1.amazonaws.com', "access_key_id"='XXXXXX',
"secret_access_key"='XXXXXX', "bucket"='test-loading',
"filepath"='char_varchar_1.csv', "region"='us-east-1',
"compression"='none'} fields terminated by ',' enclosed by '\"' lines
terminated by '\n';
```

4. 导入成功后，你可以运行如下 SQL 语句查看导入数据的结果。你将可以看到查询速度明显慢于从本地表查询。

```
select * from t1;
+-----------+-----------+-----------+-----------+
| col1 | col2 | col3 | col4 |
+-----------+-----------+-----------+-----------+
| a | b | c | d |
| a | b | c | d |
| 'a' | 'b' | 'c' | 'd' |
| 'a' | 'b' | 'c' | 'd' |
| aa,aa | bb,bb | cc,cc | dd,dd |
| aa, | bb, | cc, | dd, |
| aa,,,aa | bb,,,bb | cc,,,cc | dd,,,dd |
| aa',',,aa | bb',',,bb | cc',',,cc | dd',',,dd |
| aa"aa | bb"bb | cc"cc | dd"dd |
| aa"aa | bb"bb | cc"cc | dd"dd |
| aa"aa | bb"bb | cc"cc | dd"dd |
| aa""aa | bb""bb | cc""cc | dd""dd |
| aa""aa | bb""bb | cc""cc | dd""dd |
| aa",aa | bb",bb | cc",cc | dd",dd |
| aa"",aa | bb"",bb | cc"",cc | dd"",dd |
| | | | |
| | | | |
| NULL | NULL | NULL | NULL |
| | | | |
| " | " | " | " |
| "" | "" | "" | "" |
+-----------+-----------+-----------+-----------+
21 rows in set (1.32 sec)
```

5. （选做）如果需要将外部表数据导入到 MatrixOne 中的数据表，使用如下 SQL 语句：

在 MatrixOne 中新建一个表 *t2*：

```
create table t2(col1 char(225), col2 varchar(225), col3 text, col4
varchar(225));
```

将外部表 *t1* 导入到 *t2*：

```
insert into t2 select * from t1;
```

使用 `SELECT INTO...OUTFILE` 导出数据

MatrixOne 支持以下两种方式导出数据：

- `SELECT INTO...OUTFILE`
- `modump`

本篇文档主要介绍如何使用 `SELECT INTO...OUTFILE` 导出数据。

使用 `SELECT...INTO OUTFILE` 语法可以将表数据导出到主机上的文本文件中。

语法结构

`SELECT...INTO OUTFILE` 语法是 `SELECT` 语法和 `INTO OUTFILE filename` 的结合。默认输出格式与 `LOAD DATA` 命令相同。因此，以下语句是将名称为 **test** 的表导出到目录路径为 **/root/test** 的*. csv* 文件中。

```
mysql> SELECT * FROM TEST
-> INTO OUTFILE '/root/test.csv';
```

你可以采用多种形式和选项更改输出格式，用于表示如何引用、分隔列和记录。

使用以下代码以*. csv* 格式导出 *TEST* 表，下面的代码行是用回车换行进行展示的：

```
mysql> SELECT * FROM TEST INTO OUTFILE '/root/test.csv'
-> FIELDS TERMINATED BY ',' ENCLOSED BY '"'
-> LINES TERMINATED BY '\r\n';
```

`SELECT ... INTO OUTFILE` **特性如下**：

- 导出的文件是由 MatrixOne 服务直接创建的，因此命令行中的 `filename` 应该指向你需要文件存入的服务器主机的位置。MatrixOne 暂不支持将文件导出到客户端文件系统。
- `SELECT ... INTO OUTFILE` 是用于将检索出来的数据按格式导出到文件中，即需要导出的文件是由 MatrixOne 服务直接创建，导出的文件只能位于 MatrixOne 所在的服务器主

机上，所以你必须得有登录 MatrixOne 所在的服务器主机的用户名密码，并且你有权限可以从 MatrixOne 检索文件。

- 你必须有执行 `SELECT` 的权限。
- 检查文件需要导出到的目录里不要有重名的文件，否则会被新导出的文件覆盖。

示例

开始前准备

已完成单机部署 MatrixOne。

> **注意**
>
> 如果你是通过 `docker` 安装的 MatrixOne，那么导出目录默认位于 docker 镜像中。如果你要需要挂载本地目录，参见下面的代码示例：本地文件系统路径 *${local_data_path}/mo-data* 挂载到 MatrixOne Docker 镜像中，并映射到 */mo-data* 路径下。更多信息，参见 Docker Mount Volume tutorial。

```
sudo docker run --name \<name> --privileged -d -p 6001:6001 -v $\
{local_data_path}/mo-data:/mo-data:rw matrixorigin/matrixone:0.7.0
```

步骤

1. 在 MatrixOne 中新建一个数据表：

```
create database aaa;
use aaa;
CREATE TABLE `user` (`id` int(11) ,`user_name` varchar(255) ,`sex`
varchar(255));
insert into user(id,user_name,sex) values('1', 'weder', 'man'), ('2',
'tom', 'man'), ('3', 'wederTom', 'man');
select * from user;
+------+-----------+------+
| id | user_name | sex |
+------+-----------+------+
| 1 | weder | man |
| 2 | tom | man |
| 3 | wederTom | man |
+------+-----------+------+
```

2. 对于使用源代码或二进制文件的方式安装构建 MatrixOne，将表导出到本地目录，例如 *~/tmp/export_demo/export_datatable.txt*，命令示例如下：

```
select * from user into outfile
'~/tmp/export_demo/export_datatable.txt'
```

使用 Docker 安装启动 MatrixOne，导出到你挂载的容器目录路径，如下例所示。其中目录 *mo-data* 指的是本地路径 *~/tmp/docker_export_demo/mo-data*。

```
select * from user into outfile 'mo-data/export_datatable.txt';
```

3. 到你本地 *export_datatable.txt* 文件下查看导出情况：

```
id,user_name,sex
1,"weder","man"
2,"tom","man"
3,"wederTom","man"
```

使用 `mo-dump` 导出数据

MatrixOne 支持以下两种方式导入数据：

- `SELECT INTO...OUTFILE`
- `mo-dump`

本篇文档主要介绍如何使用 `mo-dump` 导出数据。

什么是 `mo-dump`

`mo-dump` 是 MatrixOne 的一个客户端实用工具，与 `mysqldump` 一样，它可以被用于通过导出 `.sql` 类型的文件来对 MatrixOne 数据库进行备份，该文件类型包含可执行以重新创建原始数据库的 SQL 语句。

使用 `mo-dump` 工具，你必须能够访问运行 MatrixOne 实例的服务器。你还必须拥有导出的数据库的用户权限。

语法结构

```
./mo-dump -u ${user} -p ${password} -h ${host} -P ${port} -db ${database} [-tbl ${table}...] -net-buffer-length ${net-buffer-length} > {dumpfilename.sql}
```

参数释义

- **-u [user]**：连接 MatrixOne 服务器的用户名。只有具有数据库和表读取权限的用户才能使用 `mo-dump` 实用程序，默认值 dump。
- **-p [password]**：MatrixOne 用户的有效密码。默认值：111。
- **-h [host]**：MatrixOne 服务器的主机 IP 地址。默认值：127.0.0.1
- **-P [port]**：MatrixOne 服务器的端口。默认值：6001
- **-db [数据库名称]**：必需参数。要备份的数据库的名称。

- **-net-buffer-length [数据包大小]**：数据包大小，即 SQL 语句字符的总大小。数据包是 SQL 导出数据的基本单位，如果不设置参数，则默认 1048576 Byte（1M），最大可设置 16777216 Byte（16M）。假如这里的参数设置为 16777216 Byte（16M），那么，当要导出大于 16M 的数据时，会把数据拆分成多个 16M 的数据包，除最后一个数据包之外，其它数据包大小都为 16M。

- **-tbl [表名]**：可选参数。如果参数为空，则导出整个数据库。如果要备份指定表，则可以在命令中指定多个 `-tbl` 和表名。

构建 mo-dump 二进制文件

`mo-dump` 命令程序嵌入在 MatrixOne 源代码中，你首先需要从 MatrixOne 源代码构建二进制文件。

Tips: 由于 `mo-dump` 是基于 Go 语言进行开发，所以你同时需要安装部署 Go 语言。

1. 执行下面的代码即可从 MatrixOne 源代码构建 `mo-dump` 二进制文件：

```
git clone https://github.com/matrixorigin/matrixone.git
cd matrixone
make build modump
```

2. 你可以在 MatrixOne 文件夹中找到 `mo-dump` 可执行文件：*mo-dump*。

> **注意**
>
> 构建好的 `mo-dump` 文件也可以在相同的硬件平台上工作。但是需要注意在 x86 平台中构建的 `mo-dump` 二进制文件在 Darwin ARM 平台中则无法正常工作。你可以在同一套操作系统和硬件平台内构建并使用 `mo-dump` 二进制文件。`mo-dump` 目前只支持 Linux 和 macOS。

如何使用 `mo-dump` 导出 MatrixOne 数据库

`mo-dump` 在命令行中非常易用。参见以下步骤，导出*. sql* 文件格式完整数据库：

在你本地计算机上打开终端窗口，输入以下命令，连接到 MatrixOne，并且导出数据库：

```
./mo-dump -u username -p password -h host_ip_address -P port -db database
> exporteddb.sql
```

例如，如果你在与 MatrixOne 实例相同的服务器中启动终端，并且你想要生成单个数据库的备份，请运行以下命令。该命令将在 *t.sql* 文件中生成 **t** 数据库的结构和数据的备份。*t.sql* 文件将与您的 `mo-dump` 可执行文件位于同一目录中。

```
./mo-dump -u dump -p 111 -h 127.0.0.1 -P 6001 -db t > t.sql
```

如果要在数据库中生成单个表的备份，可以运行以下命令。该命令将生成命名为 *t* 的数据库的 *t1* 表的备份，其中包含 *t.sql* 文件中的结构和数据。

```
./mo-dump -u dump -p 111 -db t -tbl t1 > t1.sql
```

限制

- `mo-dump` 仅支持导出单个数据库的备份，如果你有多个数据库需要备份，需要手动运行 `mo-dump` 多次。
- `mo-dump` 暂不支持只导出数据库的结构或数据。如果你想在没有数据库结构的情况下生成数据的备份，或者仅想导出数据库结构，那么，你需要手动拆分 `.sql` 文件。

单表查询

本篇文章介绍如何使用 SQL 来对数据库中的数据进行查询。

开始前准备

你需要确认在开始之前，已经完成了以下任务：

- 已完成单机部署 MatrixOne。

数据准备

新建一个命名为 *token_demo* 的数据库

```
CREATE DATABASE token_demo;
USE token_demo;
```

新建一个命名为 *token_count* 的表

```
CREATE TABLE token_count (
id int,
token varchar(100) DEFAULT '' NOT NULL,
count int DEFAULT 0 NOT NULL,
qty int,
phone char(1) DEFAULT '' NOT NULL,
times datetime DEFAULT '2000-01-01 00:00:00' NOT NULL
);
INSERT INTO token_count VALUES
(21,'e45703b64de71482360de8fec94c3ade',3,7800,'n','1999-12-23 17:22:21');
INSERT INTO token_count VALUES
(22,'e45703b64de71482360de8fec94c3ade',4,5000,'y','1999-12-23 17:22:21');
INSERT INTO token_count VALUES
(18,'346d1cb63c89285b2351f0ca4de40eda',3,13200,'b','1999-12-23
11:58:04');
INSERT INTO token_count VALUES
(17,'ca6ddeb689e1b48a04146b1b5b6f936a',4,15000,'b','1999-12-23
11:36:53');
INSERT INTO token_count VALUES
(16,'ca6ddeb689e1b48a04146b1b5b6f936a',3,13200,'b','1999-12-23
11:36:53');
INSERT INTO token_count VALUES
(26,'a71250b7ed780f6ef3185bfffe027983',5,1500,'b','1999-12-27 09:44:24');
INSERT INTO token_count VALUES
(24,'4d75906f3c37ecff478a1eb56637aa09',3,5400,'y','1999-12-23 17:29:12');
INSERT INTO token_count VALUES
(25,'4d75906f3c37ecff478a1eb56637aa09',4,6500,'y','1999-12-23 17:29:12');
INSERT INTO token_count VALUES
(27,'a71250b7ed780f6ef3185bfffe027983',3,6200,'b','1999-12-27 09:44:24');
INSERT INTO token_count VALUES
(28,'a71250b7ed780f6ef3185bfffe027983',3,5400,'y','1999-12-27 09:44:36');
INSERT INTO token_count VALUES
(29,'a71250b7ed780f6ef3185bfffe027983',4,17700,'b','1999-12-27
09:45:05');
```

简单的查询

在 MySQL Client 等客户端输入并执行如下 SQL 语句：

```
mysql> SELECT id, token FROM token_count;
```

输出结果如下：

```
+------+----------------------------------+
| id | token |
+------+----------------------------------+
| 21 | e45703b64de71482360de8fec94c3ade |
| 22 | e45703b64de71482360de8fec94c3ade |
| 18 | 346d1cb63c89285b2351f0ca4de40eda |
| 17 | ca6ddeb689e1b48a04146b1b5b6f936a |
| 16 | ca6ddeb689e1b48a04146b1b5b6f936a |
| 26 | a71250b7ed780f6ef3185bfffe027983 |
| 24 | 4d75906f3c37ecff478a1eb56637aa09 |
| 25 | 4d75906f3c37ecff478a1eb56637aa09 |
| 27 | a71250b7ed780f6ef3185bfffe027983 |
| 28 | a71250b7ed780f6ef3185bfffe027983 |
| 29 | a71250b7ed780f6ef3185bfffe027983 |
+------+----------------------------------+
```

对结果进行筛选

如果你需要从诸多查询得到的结果中筛选出你需要的结果，可以通过 `WHERE` 语句对查询的结果进行过滤，从而找到想要查询的部分。

在 SQL 中，可以使用 `WHERE` 子句添加筛选的条件：

```
mysql> SELECT * FROM token_count WHERE id = 25;
```

输出结果如下：

```
+------+----------------------------------+-------+------+-------+-------
--------------+
| id | token | count | qty | phone | times |
+------+----------------------------------+-------+------+-------+-------
--------------+
| 25 | 4d75906f3c37ecff478a1eb56637aa09 | 4 | 6500 | y | 1999-12-23
17:29:12 |
+------+----------------------------------+-------+------+-------+-------
--------------+
```

对结果进行排序

使用 `ORDER BY` 语句可以让查询结果按照期望的方式进行排序。

例如，可以通过下面的 SQL 语句对 token_count 表的数据按照 times 列进行降序 (DESC) 排序。

```
mysql> SELECT id, token, times FROM token_count ORDER BY times DESC;
```

输出结果如下：

```
+------+----------------------------------+---------------------+
| id | token | times |
+------+----------------------------------+---------------------+
| 29 | a71250b7ed780f6ef3185bfffe027983 | 1999-12-27 09:45:05 |
| 28 | a71250b7ed780f6ef3185bfffe027983 | 1999-12-27 09:44:36 |
| 26 | a71250b7ed780f6ef3185bfffe027983 | 1999-12-27 09:44:24 |
| 27 | a71250b7ed780f6ef3185bfffe027983 | 1999-12-27 09:44:24 |
| 24 | 4d75906f3c37ecff478a1eb56637aa09 | 1999-12-23 17:29:12 |
| 25 | 4d75906f3c37ecff478a1eb56637aa09 | 1999-12-23 17:29:12 |
| 21 | e45703b64de71482360de8fec94c3ade | 1999-12-23 17:22:21 |
| 22 | e45703b64de71482360de8fec94c3ade | 1999-12-23 17:22:21 |
| 18 | 346d1cb63c89285b2351f0ca4de40eda | 1999-12-23 11:58:04 |
| 17 | ca6ddeb689e1b48a04146b1b5b6f936a | 1999-12-23 11:36:53 |
| 16 | ca6ddeb689e1b48a04146b1b5b6f936a | 1999-12-23 11:36:53 |
+------+----------------------------------+---------------------+
```

限制查询结果数量

如果希望只返回部分结果，可以使用 `LIMIT` 语句限制查询结果返回的记录数。

```
mysql> SELECT id, token, times FROM token_count ORDER BY times DESC LIMIT
5;
```

运行结果如下：

```
+------+----------------------------------+---------------------+
| id | token | times |
+------+----------------------------------+---------------------+
| 29 | a71250b7ed780f6ef3185bfffe027983 | 1999-12-27 09:45:05 |
| 28 | a71250b7ed780f6ef3185bfffe027983 | 1999-12-27 09:44:36 |
| 26 | a71250b7ed780f6ef3185bfffe027983 | 1999-12-27 09:44:24 |
| 27 | a71250b7ed780f6ef3185bfffe027983 | 1999-12-27 09:44:24 |
| 24 | 4d75906f3c37ecff478a1eb56637aa09 | 1999-12-23 17:29:12 |
+------+----------------------------------+---------------------+
```

聚合查询

如果你想要关注数据整体的情况，而不是部分数据，你可以通过使用 `GROUP BY` 语句配合聚合函数，构建一个聚合查询来帮助你对数据的整体情况有一个更好的了解。

比如说，你可以将基本信息按照 id、count、times 列进行分组，然后分别统计：

```
mysql> SELECT id, count, times FROM token_count GROUP BY id, count, times
ORDER BY times DESC
LIMIT 5;
```

运行结果如下：

```
+------+-------+---------------------+
| id | count | times |
+------+-------+---------------------+
| 29 | 4 | 1999-12-27 09:45:05 |
| 28 | 3 | 1999-12-27 09:44:36 |
| 26 | 5 | 1999-12-27 09:44:24 |
| 27 | 3 | 1999-12-27 09:44:24 |
| 24 | 3 | 1999-12-23 17:29:12 |
+------+-------+---------------------+
```

多表连接查询

一些使用数据库的场景中，需要一个查询当中使用到多张表的数据，你可以通过 `JOIN` 语句将两张或多张表的数据组合在一起。

开始前准备

你需要确认在开始之前，已经完成了以下任务：

已完成单机部署 MatrixOne。

数据准备

1. 下载数据集：

```
https://community-shared-data-1308875761.cos.ap-beijing.myqcloud.com/tpch/tpch-1g.zip
```

2. 创建数据库和数据表：

```
create database d1;
use d1;
CREATE TABLE NATION ( N_NATIONKEY INTEGER NOT NULL,
N_NAME CHAR(25) NOT NULL,
N_REGIONKEY INTEGER NOT NULL,
N_COMMENT VARCHAR(152),
PRIMARY KEY (N_NATIONKEY));

CREATE TABLE REGION ( R_REGIONKEY INTEGER NOT NULL,
R_NAME CHAR(25) NOT NULL,
R_COMMENT VARCHAR(152),
PRIMARY KEY (R_REGIONKEY));

CREATE TABLE PART ( P_PARTKEY INTEGER NOT NULL,
P_NAME VARCHAR(55) NOT NULL,
P_MFGR CHAR(25) NOT NULL,
P_BRAND CHAR(10) NOT NULL,
P_TYPE VARCHAR(25) NOT NULL,
P_SIZE INTEGER NOT NULL,
P_CONTAINER CHAR(10) NOT NULL,
P_RETAILPRICE DECIMAL(15,2) NOT NULL,
P_COMMENT VARCHAR(23) NOT NULL,
PRIMARY KEY (P_PARTKEY));

CREATE TABLE SUPPLIER ( S_SUPPKEY INTEGER NOT NULL,
S_NAME CHAR(25) NOT NULL,
S_ADDRESS VARCHAR(40) NOT NULL,
S_NATIONKEY INTEGER NOT NULL,
S_PHONE CHAR(15) NOT NULL,
S_ACCTBAL DECIMAL(15,2) NOT NULL,
S_COMMENT VARCHAR(101) NOT NULL,
PRIMARY KEY (S_SUPPKEY));

CREATE TABLE PARTSUPP ( PS_PARTKEY INTEGER NOT NULL,
PS_SUPPKEY INTEGER NOT NULL,
PS_AVAILQTY INTEGER NOT NULL,
PS_SUPPLYCOST DECIMAL(15,2) NOT NULL,
PS_COMMENT VARCHAR(199) NOT NULL,
PRIMARY KEY (PS_PARTKEY, PS_SUPPKEY));

CREATE TABLE CUSTOMER ( C_CUSTKEY INTEGER NOT NULL,
C_NAME VARCHAR(25) NOT NULL,
C_ADDRESS VARCHAR(40) NOT NULL,
C_NATIONKEY INTEGER NOT NULL,
```

```
C_PHONE CHAR(15) NOT NULL,
C_ACCTBAL DECIMAL(15,2) NOT NULL,
C_MKTSEGMENT CHAR(10) NOT NULL,
C_COMMENT VARCHAR(117) NOT NULL,
PRIMARY KEY (C_CUSTKEY));

CREATE TABLE ORDERS ( O_ORDERKEY BIGINT NOT NULL,
O_CUSTKEY INTEGER NOT NULL,
O_ORDERSTATUS CHAR(1) NOT NULL,
O_TOTALPRICE DECIMAL(15,2) NOT NULL,
O_ORDERDATE DATE NOT NULL,
O_ORDERPRIORITY CHAR(15) NOT NULL,
O_CLERK CHAR(15) NOT NULL,
O_SHIPPRIORITY INTEGER NOT NULL,
O_COMMENT VARCHAR(79) NOT NULL,
PRIMARY KEY (O_ORDERKEY));

CREATE TABLE LINEITEM ( L_ORDERKEY BIGINT NOT NULL,
L_PARTKEY INTEGER NOT NULL,
L_SUPPKEY INTEGER NOT NULL,
L_LINENUMBER INTEGER NOT NULL,
L_QUANTITY DECIMAL(15,2) NOT NULL,
L_EXTENDEDPRICE DECIMAL(15,2) NOT NULL,
L_DISCOUNT DECIMAL(15,2) NOT NULL,
L_TAX DECIMAL(15,2) NOT NULL,
L_RETURNFLAG CHAR(1) NOT NULL,
L_LINESTATUS CHAR(1) NOT NULL,
L_SHIPDATE DATE NOT NULL,
L_COMMITDATE DATE NOT NULL,
L_RECEIPTDATE DATE NOT NULL,
L_SHIPINSTRUCT CHAR(25) NOT NULL,
L_SHIPMODE CHAR(10) NOT NULL,
L_COMMENT VARCHAR(44) NOT NULL,
PRIMARY KEY (L_ORDERKEY, L_LINENUMBER));
```

3. 把数据导入到数据表中：

```
load data infile '/YOUR_TPCH_DATA_PATH/nation.tbl' into table NATION
FIELDS TERMINATED BY '|' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED
BY '\n';

load data infile '/YOUR_TPCH_DATA_PATH/region.tbl' into table REGION
FIELDS TERMINATED BY '|' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED
BY '\n';

load data infile '/YOUR_TPCH_DATA_PATH/part.tbl' into table PART
FIELDS TERMINATED BY '|' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED
BY '\n';

load data infile '/YOUR_TPCH_DATA_PATH/supplier.tbl' into table
SUPPLIER FIELDS TERMINATED BY '|' OPTIONALLY ENCLOSED BY '"' LINES
TERMINATED BY '\n';

load data infile '/YOUR_TPCH_DATA_PATH/partsupp.tbl' into table
PARTSUPP FIELDS TERMINATED BY '|' OPTIONALLY ENCLOSED BY '"' LINES
TERMINATED BY '\n';

load data infile '/YOUR_TPCH_DATA_PATH/orders.tbl' into table ORDERS
FIELDS TERMINATED BY '|' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED
BY '\n';

load data infile '/YOUR_TPCH_DATA_PATH/customer.tbl' into table
CUSTOMER FIELDS TERMINATED BY '|' OPTIONALLY ENCLOSED BY '"' LINES
TERMINATED BY '\n';

load data infile '/YOUR_TPCH_DATA_PATH/lineitem.tbl' into table
LINEITEM FIELDS TERMINATED BY '|' OPTIONALLY ENCLOSED BY '"' LINES
TERMINATED BY '\n';
```

现在你可以使用这些数据进行查询。

Join 类型

内连接

内连接的连接结果只返回匹配连接条件的行。

语法	图示
SELECT <select_list> FROM TableA A INNER JOIN TableB B ON A.Key=B.Key	innerjoin

内连接有两种书写方式，在结果上是完全等价的：

```
mysql> SELECT
l_orderkey,
SUM(l_extendedprice * (1 - l_discount)) AS revenue,
o_orderdate,
o_shippriority
FROM
CUSTOMER,
ORDERS,
LINEITEM
WHERE
c_mktsegment = 'BUILDING'
AND c_custkey = o_custkey
AND l_orderkey = o_orderkey
AND o_orderdate < DATE '1995-03-15'
AND l_shipdate > DATE '1995-03-15'
GROUP BY l_orderkey , o_orderdate , o_shippriority
ORDER BY revenue DESC , o_orderdate
LIMIT 10;
+------------+---------------------+-------------+----------------+
| l_orderkey | revenue | o_orderdate | o_shippriority |
+------------+---------------------+-------------+----------------+
| 2456423 | 406181.011100000000 | 1995-03-05 | 0 |
| 3459808 | 405838.698900000000 | 1995-03-04 | 0 |
| 492164 | 390324.061000000000 | 1995-02-19 | 0 |
| 1188320 | 384537.935900000000 | 1995-03-09 | 0 |
| 2435712 | 378673.055800000000 | 1995-02-26 | 0 |
| 4878020 | 378376.795200000000 | 1995-03-12 | 0 |
| 5521732 | 375153.921500000000 | 1995-03-13 | 0 |
| 2628192 | 373133.309400000000 | 1995-02-22 | 0 |
| 993600 | 371407.459500000000 | 1995-03-05 | 0 |
| 2300070 | 367371.145200000000 | 1995-03-13 | 0 |
+------------+---------------------+-------------+----------------+
10 rows in set (0.20 sec)
```

写成 `Join` 的形式，语法如下：

```
mysql> SELECT
l_orderkey,
SUM(l_extendedprice * (1 - l_discount)) AS revenue,
o_orderdate,
o_shippriority
FROM
CUSTOMER
join ORDERS on c_custkey = o_custkey
join LINEITEM on l_orderkey = o_orderkey
WHERE
c_mktsegment = 'BUILDING'
AND o_orderdate < DATE '1995-03-15'
AND l_shipdate > DATE '1995-03-15'
GROUP BY l_orderkey , o_orderdate , o_shippriority
ORDER BY revenue DESC , o_orderdate
LIMIT 10;
+------------+--------------------+-------------+----------------+
| l_orderkey | revenue | o_orderdate | o_shippriority |
+------------+--------------------+-------------+----------------+
| 2456423 | 406181.011100000000 | 1995-03-05 | 0 |
| 3459808 | 405838.698900000000 | 1995-03-04 | 0 |
| 492164 | 390324.061000000000 | 1995-02-19 | 0 |
| 1188320 | 384537.935900000000 | 1995-03-09 | 0 |
| 2435712 | 378673.055800000000 | 1995-02-26 | 0 |
| 4878020 | 378376.795200000000 | 1995-03-12 | 0 |
| 5521732 | 375153.921500000000 | 1995-03-13 | 0 |
| 2628192 | 373133.309400000000 | 1995-02-22 | 0 |
| 993600 | 371407.459500000000 | 1995-03-05 | 0 |
| 2300070 | 367371.145200000000 | 1995-03-13 | 0 |
+------------+--------------------+-------------+----------------+
10 rows in set (0.20 sec)
```

外连接

外连接又分为**左连接**、**右连接**，两者之间是可以实现等价语义的：

- `LEFT JOIN`

左外连接会返回左表中的所有数据行，以及右表当中能够匹配连接条件的值，如果在右表当中没有找到能够匹配的行，则使用 NULL 填充。

语法	图示
SELECT <select_list> FROM TableA A LEFT JOIN TableB B ON A.Key=B.Key	leftjoin
SELECT <select_list> FROM TableA A LEFT JOIN TableB B ON A.Key=B.Key WHERE B.Key IS NULL	leftjoinwhere

- `RIGHT JOIN`

右外连接返回右表中的所有记录，以及左表当中能够匹配连接条件的值，没有匹配的值则使用 NULL 填充。

语法	图示
SELECT <select_list> FROM TableA A RIGHT JOIN TableB B ON A.Key=B.Key	leftjoinwhere
SELECT <select_list> FROM TableA A RIGHT JOIN TableB B ON A.Key=B.Key WHERE A.Key IS NULL	leftjoinwhere

语句示例如下：

```
SELECT
c_custkey, COUNT(o_orderkey) AS c_count
FROM
CUSTOMER
LEFT OUTER JOIN ORDERS ON (c_custkey = o_custkey
AND o_comment NOT LIKE '%special%requests%')
GROUP BY c_custkey limit 10;

+-----------+---------+
| c_custkey | c_count |
+-----------+---------+
| 147457 | 16 |
| 147458 | 7 |
| 147459 | 0 |
| 147460 | 16 |
| 147461 | 7 |
| 147462 | 0 |
| 147463 | 14 |
| 147464 | 11 |
| 147465 | 0 |
| 147466 | 17 |
+-----------+---------+
10 rows in set (0.93 sec)
```

或者：

```
SELECT
c_custkey, COUNT(o_orderkey) AS c_count
FROM
ORDERS
RIGHT OUTER JOIN CUSTOMER ON (c_custkey = o_custkey
AND o_comment NOT LIKE '%special%requests%')
GROUP BY c_custkey limit 10;

+-----------+---------+
| c_custkey | c_count |
+-----------+---------+
| 147457 | 16 |
| 147458 | 7 |
| 147459 | 0 |
| 147460 | 16 |
| 147461 | 7 |
| 147462 | 0 |
| 147463 | 14 |
| 147464 | 11 |
| 147465 | 0 |
| 147466 | 17 |
+-----------+---------+
10 rows in set (0.93 sec)
```

全连接

全连接是左右外连接的并集。连接表包含被连接的表的所有记录，如果缺少匹配的记录，即以 NULL 填充。

```
SELECT
c_custkey, COUNT(o_orderkey) AS c_count
FROM
CUSTOMER
FULL JOIN ORDERS ON (c_custkey = o_custkey
AND o_comment NOT LIKE '%special%requests%')
GROUP BY c_custkey limit 10;

+-----------+---------+
| c_custkey | c_count |
+-----------+---------+
| 1 | 6 |
| 2 | 7 |
| 4 | 20 |
| 5 | 4 |
| 7 | 16 |
| 8 | 13 |
| 10 | 20 |
| 11 | 13 |
| 13 | 18 |
| 14 | 9 |
+-----------+---------+
10 rows in set (0.77 sec)
```

全连接同样可以通过改写的方式获得相同的语义：

```
SELECT
c_custkey, COUNT(o_orderkey) AS c_count
FROM
CUSTOMER
LEFT OUTER JOIN ORDERS ON (c_custkey = o_custkey
AND o_comment NOT LIKE '%special%requests%')
GROUP BY c_custkey
UNION
SELECT
c_custkey, COUNT(o_orderkey) AS c_count
FROM
CUSTOMER
LEFT OUTER JOIN ORDERS ON (c_custkey = o_custkey
AND o_comment NOT LIKE '%special%requests%')
WHERE c_custkey IS NULL
GROUP BY c_custkey
limit 10;

+-----------+---------+
| c_custkey | c_count |
+-----------+---------+
| 147457 | 16 |
| 147458 | 7 |
| 147459 | 0 |
| 147460 | 16 |
| 147461 | 7 |
| 147462 | 0 |
| 147463 | 14 |
| 147464 | 11 |
| 147465 | 0 |
| 147466 | 17 |
+-----------+---------+
10 rows in set (1.09 sec)
```

隐式连接

在 SQL 语句当中，除了使用 `JOIN`，也可以通过 `FROM t1, t2` 子句来连接两张或多张表，通过 `WHERE t1.id = t2.id` 子句来指定连接的条件。

子查询

本篇文档向你介绍 MatrixOne 的子查询功能。

概述

子查询是嵌套在另一个查询中的 SQL 表达式，借助子查询，可以在一个查询当中使用另外一个查询的查询结果。

通常情况下，从 SQL 语句结构上，子查询语句一般有以下几种形式：

- 标量子查询（Scalar Subquery），如 `SELECT (SELECT s1 FROM t2) FROM t1`。
- 派生表（Derived Tables），如 `SELECT t1.s1 FROM (SELECT s1 FROM t2) t1`。
- 存在性测试（Existential Test），如 `WHERE NOT EXISTS(SELECT ... FROM t2)`，`WHERE t1.a IN (SELECT ... FROM t2)`。
- 集合比较（Quantified Comparison），如 `WHERE t1.a = ANY(SELECT ... FROM t2)`

  。
- 作为比较运算符操作数的子查询，如 `WHERE t1.a > (SELECT ... FROM t2)`。

关于子查询 SQL 语句，参见 SUBQUERY。

另外，从 SQL 语句执行情况上，子查询语句一般有以下两种形式：

- 关联子查询（Correlated Subquery）：数据库嵌套查询中内层查询和外层查询不相互独立，内层查询也依赖于外层查询。

  执行顺序为：

  - 先从外层查询中查询中一条记录。
  - 再将查询到的记录放到内层查询中符合条件的记录，再放到外层中查询。
  - 重复以上步骤

例如：

```
select * from tableA where tableA.cloumn < (select column from tableB
where tableA.id = tableB.id))
```

- 无关联子查询 (Self-contained Subquery)：数据库嵌套查询中内层查询是完全独立于外层查询的。

  执行顺序为：

  - 先执行内层查询。
  - 得到内层查询的结果后带入外层，再执行外层查询。

  例如：

```
select * from tableA where tableA.column = (select tableB.column from
tableB )
```

子查询的作用：

- 子查询允许结构化的查询，这样就可以把一个查询语句的每个部分隔开。
- 子查询提供了另一种方法来执行有些需要复杂的 `JOIN` 和 `UNION` 来实现的操作。

我们将举一个简单的例子帮助你理解**关联子查询**和**无关联子查询**。

示例

开始前准备

你需要确认在开始之前，已经完成了以下任务：

- 已完成单机部署 MatrixOne。

数据准备

1. 下载数据集：

```
https://community-shared-data-1308875761.cos.ap-
beijing.myqcloud.com/tpch/tpch-1g.zip
```

2. 创建数据库和数据表：

```
create database d1;
use d1;
CREATE TABLE NATION ( N_NATIONKEY INTEGER NOT NULL,
N_NAME CHAR(25) NOT NULL,
N_REGIONKEY INTEGER NOT NULL,
N_COMMENT VARCHAR(152),
PRIMARY KEY (N_NATIONKEY));

CREATE TABLE REGION ( R_REGIONKEY INTEGER NOT NULL,
R_NAME CHAR(25) NOT NULL,
R_COMMENT VARCHAR(152),
PRIMARY KEY (R_REGIONKEY));

CREATE TABLE PART ( P_PARTKEY INTEGER NOT NULL,
P_NAME VARCHAR(55) NOT NULL,
P_MFGR CHAR(25) NOT NULL,
P_BRAND CHAR(10) NOT NULL,
P_TYPE VARCHAR(25) NOT NULL,
P_SIZE INTEGER NOT NULL,
P_CONTAINER CHAR(10) NOT NULL,
P_RETAILPRICE DECIMAL(15,2) NOT NULL,
P_COMMENT VARCHAR(23) NOT NULL,
PRIMARY KEY (P_PARTKEY));

CREATE TABLE SUPPLIER ( S_SUPPKEY INTEGER NOT NULL,
S_NAME CHAR(25) NOT NULL,
S_ADDRESS VARCHAR(40) NOT NULL,
S_NATIONKEY INTEGER NOT NULL,
S_PHONE CHAR(15) NOT NULL,
S_ACCTBAL DECIMAL(15,2) NOT NULL,
S_COMMENT VARCHAR(101) NOT NULL,
PRIMARY KEY (S_SUPPKEY));

CREATE TABLE PARTSUPP ( PS_PARTKEY INTEGER NOT NULL,
PS_SUPPKEY INTEGER NOT NULL,
PS_AVAILQTY INTEGER NOT NULL,
PS_SUPPLYCOST DECIMAL(15,2) NOT NULL,
PS_COMMENT VARCHAR(199) NOT NULL,
PRIMARY KEY (PS_PARTKEY, PS_SUPPKEY));

CREATE TABLE CUSTOMER ( C_CUSTKEY INTEGER NOT NULL,
C_NAME VARCHAR(25) NOT NULL,
C_ADDRESS VARCHAR(40) NOT NULL,
C_NATIONKEY INTEGER NOT NULL,
```

```
C_PHONE CHAR(15) NOT NULL,
C_ACCTBAL DECIMAL(15,2) NOT NULL,
C_MKTSEGMENT CHAR(10) NOT NULL,
C_COMMENT VARCHAR(117) NOT NULL,
PRIMARY KEY (C_CUSTKEY));

CREATE TABLE ORDERS ( O_ORDERKEY BIGINT NOT NULL,
O_CUSTKEY INTEGER NOT NULL,
O_ORDERSTATUS CHAR(1) NOT NULL,
O_TOTALPRICE DECIMAL(15,2) NOT NULL,
O_ORDERDATE DATE NOT NULL,
O_ORDERPRIORITY CHAR(15) NOT NULL,
O_CLERK CHAR(15) NOT NULL,
O_SHIPPRIORITY INTEGER NOT NULL,
O_COMMENT VARCHAR(79) NOT NULL,
PRIMARY KEY (O_ORDERKEY));

CREATE TABLE LINEITEM ( L_ORDERKEY BIGINT NOT NULL,
L_PARTKEY INTEGER NOT NULL,
L_SUPPKEY INTEGER NOT NULL,
L_LINENUMBER INTEGER NOT NULL,
L_QUANTITY DECIMAL(15,2) NOT NULL,
L_EXTENDEDPRICE DECIMAL(15,2) NOT NULL,
L_DISCOUNT DECIMAL(15,2) NOT NULL,
L_TAX DECIMAL(15,2) NOT NULL,
L_RETURNFLAG CHAR(1) NOT NULL,
L_LINESTATUS CHAR(1) NOT NULL,
L_SHIPDATE DATE NOT NULL,
L_COMMITDATE DATE NOT NULL,
L_RECEIPTDATE DATE NOT NULL,
L_SHIPINSTRUCT CHAR(25) NOT NULL,
L_SHIPMODE CHAR(10) NOT NULL,
L_COMMENT VARCHAR(44) NOT NULL,
PRIMARY KEY (L_ORDERKEY, L_LINENUMBER));
```

3. 把数据导入到数据表中：

```
load data infile '/YOUR_TPCH_DATA_PATH/nation.tbl' into table NATION
FIELDS TERMINATED BY '|' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED
BY '\n';

load data infile '/YOUR_TPCH_DATA_PATH/region.tbl' into table REGION
FIELDS TERMINATED BY '|' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED
BY '\n';

load data infile '/YOUR_TPCH_DATA_PATH/part.tbl' into table PART
FIELDS TERMINATED BY '|' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED
BY '\n';

load data infile '/YOUR_TPCH_DATA_PATH/supplier.tbl' into table
SUPPLIER FIELDS TERMINATED BY '|' OPTIONALLY ENCLOSED BY '"' LINES
TERMINATED BY '\n';

load data infile '/YOUR_TPCH_DATA_PATH/partsupp.tbl' into table
PARTSUPP FIELDS TERMINATED BY '|' OPTIONALLY ENCLOSED BY '"' LINES
TERMINATED BY '\n';

load data infile '/YOUR_TPCH_DATA_PATH/orders.tbl' into table ORDERS
FIELDS TERMINATED BY '|' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED
BY '\n';

load data infile '/YOUR_TPCH_DATA_PATH/customer.tbl' into table
CUSTOMER FIELDS TERMINATED BY '|' OPTIONALLY ENCLOSED BY '"' LINES
TERMINATED BY '\n';

load data infile '/YOUR_TPCH_DATA_PATH/lineitem.tbl' into table
LINEITEM FIELDS TERMINATED BY '|' OPTIONALLY ENCLOSED BY '"' LINES
TERMINATED BY '\n';
```

现在你可以使用这些数据进行查询。

无关联子查询

对于将子查询作为比较运算符 (`>` / `>=` / `<` / `<=` / `=` / `!=`) 操作数的这类无关联子查询而言，内层子查询只需要进行一次查询，MatrixOne 在生成执行计划阶段会将内层子查询改写为常量。

```
mysql> select p.p_name from (select * from part where p_brand='Brand#21'
and p_retailprice between 1100 and 1200) p, partsupp ps where
p.p_partkey=ps.ps_partkey and p.p_name like '%pink%' limit 10;
```

在 MatrixOne 执行上述查询的时候会先执行一次内层子查询：

```
mysql> select * from part where p_brand='Brand#21' and p_retailprice
between 1100 and 1200
```

运行结果为：

```
+-----------------------------------+
| p_name |
+-----------------------------------+
| olive chartreuse smoke pink tan |
| olive chartreuse smoke pink tan |
| olive chartreuse smoke pink tan |
| olive chartreuse smoke pink tan |
| pink sienna dark bisque turquoise |
| pink sienna dark bisque turquoise |
| pink sienna dark bisque turquoise |
| pink sienna dark bisque turquoise |
| honeydew orchid cyan magenta pink |
| honeydew orchid cyan magenta pink |
+-----------------------------------+
10 rows in set (0.06 sec)
```

对于存在性测试和集合比较两种情况下的无关联列子查询，MatrixOne 会将其进行改写和等价替换以获得更好的执行性能。

关联子查询

对于关联子查询而言，由于内层的子查询引用外层查询的列，子查询需要对外层查询得到的每一行都执行一遍，也就是说假设外层查询得到一千万的结果，那么子查询也会被执行一千万次，这会导致查询需要消耗更多的时间和资源。

因此在处理过程中，MatrixOne 会尝试对关联子查询去关联，以从执行计划层面上提高查询效率。

```
mysql> select p_name from part where P_PARTKEY in (select PS_PARTKEY from
PARTSUPP where PS_SUPPLYCOST>=500) and p_name like '%pink%' limit 10;
```

MatrixOne 在处理该 SQL 语句是会将其改写为等价的 `JOIN` 查询：

```
select p_name from part join partsupp on P_PARTKEY=PS_PARTKEY where
PS_SUPPLYCOST>=500 and p_name like '%pink%' limit 10;
```

运行结果为：

```
+-----------------------------------+
| p_name |
+-----------------------------------+
| papaya red almond hot pink |
| turquoise hot smoke green pink |
| purple cornsilk red pink floral |
| pink cyan purple white burnished |
| sandy dark pink indian cream |
| powder cornsilk chiffon slate pink |
| rosy light black pink orange |
| pink white goldenrod ivory steel |
| cornsilk dim pink tan sienna |
| lavender navajo steel sandy pink |
+-----------------------------------+
10 rows in set (0.23 sec)
```

作为最佳实践，在实际开发当中，为提高计算效率，尽量选择等价计算方法进行查询，避免使用关联子查询的方式进行查询。

视图

本篇文档向你介绍 MatrixOne 的视图功能。

概述

视图作为一个虚拟表，进行存储查询，在调用时产生结果集。

视图的作用：

- 简化用户操作：视图机制使用户可以将注意力集中在所关心地数据上。如果这些数据不是直接来自基本表，则可以通过定义视图，使数据库看起来结构简单、清晰，并且可以简化用户的的数据查询操作。
- 以多种角度看待同一数据：视图机制能使不同的用户以不同的方式看待同一数据，当许多不同种类的用户共享同一个数据库时，这种灵活性是非常必要的。
- 对重构数据库提供了一定程度的逻辑独立性：数据的物理独立性是指用户的应用程序不依赖于数据库的物理结构。数据的逻辑独立性是指当数据库重构造时，如增加新的关系或对原有的关系增加新的字段，用户的应用程序不会受影响。层次数据库和网状数据库一般能较好地支持数据的物理独立性，而对于逻辑独立性则不能完全的支持。

开始前准备

你需要确认在开始之前，已经完成了以下任务：

- 已完成单机部署 MatrixOne。

数据准备

新建两张表，方便后续为使用视图做准备：

```
CREATE TABLE t00(a INTEGER);
INSERT INTO t00 VALUES (1),(2);
CREATE TABLE t01(a INTEGER);
INSERT INTO t01 VALUES (1);
```

可以查看一下表 *t00* 的结构：

```
mysql> select * from t00;
+------+
| a |
+------+
| 1 |
| 2 |
+------+
```

可以查看一下表 *t01* 的结构：

```
> select * from t01;
+------+
| a |
+------+
| 1 |
+------+
```

创建视图

你可以通过 `CREATE VIEW` 语句来将某个较为复杂的查询定义为视图，其语法如下：

```
CREATE VIEW view_name AS query;
```

创建的视图名称不能与已有的视图或表重名。

示例如下：

```
CREATE VIEW v0 AS SELECT t00.a, t01.a AS b FROM t00 LEFT JOIN t01
USING(a);
Query OK, 0 rows affected (0.02 sec)
```

查询视图

视图创建完成后，便可以使用 `SELECT` 语句像查询一般数据表一样查询视图。

```
mysql> SELECT * FROM v0;
+------+------+
| a | b |
+------+------+
| 1 | 1 |
| 2 | NULL |
+------+------+
```

获取视图相关信息

使用 `SHOW CREATE TABLE|VIEW view_name` 语句：

```
mysql> SHOW CREATE VIEW v0;
+------+-----------------------------------------------------------------
-----------+
| View | Create View |
+------+-----------------------------------------------------------------
-----------+
| v0 | CREATE VIEW v0 AS SELECT t00.a, t01.a AS b FROM t00 LEFT JOIN t01
USING(a) |
+------+-----------------------------------------------------------------
-----------+
1 row in set (0.00 sec)
```

删除视图

通过 `DROP VIEW view_name;` 语句可以删除已经创建的视图。

```
mysql> DROP VIEW v0;
```

公共表表达式 (CTE)

公用表表达式（CTE,Common table expression) 是一个命名的临时结果集，仅在单个 SQL 语句 (例如 `SELECT`， `INSERT`， `UPDATE` 或 `DELETE`) 的执行范围内存在。

与派生表类似，CTE 不作为对象存储，仅在查询执行期间持续；与派生表不同，CTE 可以是自引用 (递归 CTE，暂时不支持)，也可以在同一查询中多次引用。此外，与派生表相比，CTE 提供了更好的可读性和性能。

CTE 的结构包括名称，可选列列表和定义 CTE 的查询。

CTE 语法如下：

```
WITH <query_name> AS (
<query_definition>
)
SELECT ... FROM <query_name>;
```

开始前准备

你需要确认在开始之前，已经完成了以下任务：

已完成单机部署 MatrixOne。

数据准备

你可以新建一个简单的表，插入一些数据，帮助你理解后续所展示的 CTE 语句：

```
drop table if exists t1;
create table t1(a int, b int, c int);
insert into t1 values(null,null,null),(2,3,4);
```

CTE 语句使用实例

在下面的示例中， `qn` 作为一个临时的结果集被创建，此时相应的查询结果会被缓存在 MatrixOne 中，你在执行正式的 `qn` 查询时，比非 CTE 场景的性能有所提升。

```
mysql> WITH qn AS (SELECT a FROM t1), qn2 as (select b from t1)
SELECT * FROM qn;
```

查询结果如下：

```
+------+
| a |
+------+
| NULL |
| 2 |
+------+
2 rows in set (0.00 sec)
```

事务通用概念

为什么需要事务?

在许多大型、关键的应用程序中，计算机每秒钟都在执行大量的任务。更为经常的不是这些任务本身，而是将这些任务结合在一起完成一个业务要求，称为事务。如果能成功地执行一个任务，而在第二个或第三个相关的任务中出现错误，将会发生什么？这个错误很可能使系统数据处于不一致状态。这时事务变得非常重要，它能使系统数据摆脱这种不一致的状态。如何理解事务呢？例如在某家银行的银行系统中，如果没有事务对数据进行控制和管理，很可能出现 A 从企业账户中取出一笔钱，同时 B 和 C 也从同一企业账户中取钱。每一笔转账涉及到最少两个账户信息的变化（例如，A 的钱到账，企业账户出账；B 的钱到账，企业账户出账；C 的钱到账，企业账户出账），如果没有事务，那么账面金额的具体数值将无法确定。在引入事务这一业务要求之后，事务的基本特性（ACID）确保了银行账面的资金操作是原子性（不可再分割）的，其他人看到的金额是具备隔离性的，每一次操作都是具有一致性的，所有操作是持久性的，这样保证了银行系统数据出入账保持一致。

什么是事务?

数据库事务（即，Transaction，事务）是数据库管理系统执行过程中的一个逻辑单位，由一个有限的数据库操作序列构成。 事务就是作为一个逻辑单位提交或者回滚一系列的 SQL 语句。

事务的特征

通常事务需要具备 ACID 四个特征：

- **原子性（Atomicity）**： 事务的原子性是指事务是一个不可分割的单位，在一个事务中的操作要么都发生，要么都不发生。

  例如，我们现在有如下事务：

  ```
  start transaction;
  insert into t1 values(1,2,3),(4,5,6);
  update t2 set c1='b' where c1='B';
  commit;
  ```

如果对 t1 插入数据或修改 t2 数据中的任意一条发生错误，整个事务都会回滚，而只有两条语句同时成功时，才会提交成功，不会出现一个操作成功而另一个操作失败。

- **一致性（Consistency）**：事务的一致性是指在事务前后，数据必须是保持正确并遵守所有数据相关的约束。

  例如，我们在数据库中建立一个新表

```
create table t1(a int primary key,b varchar(5) not null);
```

  此处为了确保数据一致性，我们在插入数据时，你要保证 a 和 b 列的数据类型与范围，同时还要满足 a 列的主键约束与 b 列的非空约束：

```
insert into t1 values(1,'abcde'),(2,'bcdef');
```

- **隔离性（Isolation）**：事务的隔离性是在多个用户并发访问时，事务之间要遵守规定好的隔离级别，在确定的隔离级别范围内，一个事务不能被另一个事务所干扰。

  例如，如下事务示例，会话隔离级别是读已提交，在会话 1 能看到的数据如下：

```
select * from t1;
+------+------+
| a | b |
+------+------+
| 1 | a |
| 2 | b |
| 3 | c |
+------+------+
```

  此时在会话 2 中，做如下操作：

```
begin;
delete from t1 where a=3;
```

  在会话 1 中，你可以看到的数据仍然不会有变化：

```
select * from t1;
+------+------+
| a | b |
+------+------+
| 1 | a |
| 2 | b |
| 3 | c |
+------+------+
```

直到在会话 2 中提交当前事务：

```
commit;
```

在会话 1 中才会看到已提交事务的结果：

```
select * from t1;
+------+------+
| a | b |
+------+------+
| 1 | a |
| 2 | b |
+------+------+
```

- **持久性（Durability）**：事务的持久性是指，在数据库中一个事务被提交时，它对数据库中的数据的改变是永久性的，无论数据库软件是否重启。

事务的分类

在数据库中，事务分为以下几类：

- 按照是否有明确的起止分为**显示事务**和**隐式事务**。
- 按照对资源锁的使用阶段分为**乐观事务**和**悲观事务**。

这两大类事务的分类彼此不受对方限制，一个显式事务可以是乐观事务或悲观事务，同时一个悲观事务可能是显式事务也可能是隐式事务。

显式事务和隐式事务

- **显式事务**：一般来说，可以通过执行 BEGIN 语句显式启动事务。可以通过执行 COMMIT 或 ROLLBACK 显式结束事务。MatrixOne 的显示事务启动和执行方式略有不同，可以参见 MatrixOne 的显式事务。

- **隐式事务**：即事务可以隐式开始和结束，无需使用 BEGIN TRANSACTION、COMMIT 或者 ROLLBACK 语句就可以开始和结束。隐式事务的行为方式与显式事务相同。但是，确定隐式事务何时开始的规则不同于确定显式事务何时开始的规则。MatrixOne 的隐式事务启动和执行方式略有不同，可以参见 MatrixOne 的隐式事务。

乐观事务和悲观事务

无论是乐观事务，还是悲观事务，其事务的执行结果都是一样的，即一个事务中的操作，对 ACID 级别的要求，完全一样，无论是原子性、一致性、隔离性或者持久性，都是完全一致，不存在乐观事务就宽松一些，悲观事务就严格一些的情况。

乐观事务与悲观事务的区别，它只是两阶段提交基于待处理业务状态的不同执行策略，其选择基于执行者的判断，其效率高低基于被处理业务的实际状态（并发事务的写冲突频繁度）。即在于对于事务相关资源的状态做出不同的假设，从而将写锁放在不同的阶段中。

在乐观事务开始时，会假定事务相关的表处于一个不会发生写冲突的状态，把对数据的插入、修改或删除缓存在内存中，在这一阶段不会对数据加锁，而在数据提交时对相应的数据表或数据行上锁，在完成提交后解锁。

而在悲观事务中，一个事务在开始时，会假定事务相关的表一定会发生写冲突，预先对相关表或行加锁。然后才在内存中，对相关数据进行插入、修改或删除并提交。只有在提交或回滚完成后才对数据进行解锁。

乐观事务与悲观事务在使用过程中，有着如下的优缺点：

- 乐观事务对于写操作较少，读操作较多的系统更为友好，避免了死锁的出现。
- 乐观事务在较大的事务提交时，可能会因为出现冲突导致反复重试却最终失败。
- 悲观事务对于写操作较多的系统更加友好，从数据库层面避免了写写冲突。
- 悲观事务在并发较大的场景下，如果出现一个执行时间较长的事务，可能会导致系统阻塞并影响吞吐量。

MatrixOne 的乐观事务详情可以参见 MatrixOne 的乐观事务。

MatrixOne 的暂时还不支持悲观事务。

事务隔离

关于事务特征中有一条是隔离性，我们通常称之为事务隔离。

在数据库事务的 ACID 四个属性中，隔离性是一个限制最宽松的。为了获取更高的隔离等级，数据库系统的通常使用锁机制或者多版本并发控制机制。应用软件也需要额外的逻辑来使其正常工作。

很多数据库管理系统（DBMS）定义了不同的 “事务隔离等级” 来控制锁的程度。在很多数据库系统中，多数的事务都避免高等级的隔离等级（如可串行化）从而减少锁的开销。程序员需要小心的分析数据库访问部分的代码来保证隔离级别的降低不会造成难以发现的代码错误。相反的，更高的隔离级别会增加死锁发生的几率，同样需要在编程过程中去避免。

由于更高的隔离级别中不存在被一个更低的隔离级别禁止的操作，DBMS 被允许使用一个比请求的隔离级别更高的隔离级别。

ANSI/ISO SQL 定义的标准隔离级别共有四个：

隔离级别	脏写 (Dirty Write)	脏读 (Dirty Read)	不可重复读 (Fuzzy Read)	幻读 (Phantom)
READ UNCOMMITTED	Not Possible	Possible	Possible	Possible
READ COMMITTED	Not Possible	Not Possible	Possible	Possible
REPEATABLE READ	Not Possible	Not Possible	Not Possible	Possible
SERIALIZABLE	Not Possible	Not Possible	Not Possible	Not Possible

- **读未提交**：未提交读（READ UNCOMMITTED）是最低的隔离级别。允许 “脏读”（dirty reads），事务可以看到其他事务 “尚未提交” 的修改。
- **读已提交**：在提交读（READ COMMITTED）级别中，基于锁机制并发控制的 DBMS 需要对选定对象的写锁一直保持到事务结束，但是读锁在 SELECT 操作完成后马上释放。和前一种隔离级别一样，也不要求 “范围锁”。
- **可重复读**：在可重复读（REPEATABLE READS）隔离级别中，基于锁机制并发控制的 DBMS 需要对选定对象的读锁（read locks）和写锁（write locks）一直保持到事务结束，但不要求 “范围锁”，因此可能会发生 “幻读”。MatixOne 实现了快照隔离（即 Snapshot Isolation），为了与 MySQL 隔离级别保持一致，MatixOne 快照隔离又叫做可重复读（REPEATABLE READS）。

- **可串行化**：可串行化（SERIALIZABLE）是最高的隔离级别。在基于锁机制并发控制的 DBMS 上，可串行化要求在选定对象上的读锁和写锁直到事务结束后才能释放。在 SELECT 的查询中使用一个 “WHERE” 子句来描述一个范围时应该获得一个 “范围锁”（range-locks）。

通过比低一级的隔离级别要求更多的限制，高一级的级别提供更强的隔离性。标准允许事务运行在更强的事务隔离级别上。

> **注意**
>
> MatrixOne 的事务隔离与通用的隔离的定义个隔离级别的划分略有不同，可以参见 MatrixOne 的快照隔离。

MatrixOne 的事务概述

什么是 MatrixOne 的事务?

MatrixOne 事务遵循数据库事务的标准定义与基本特征 (ACID)。它旨在帮助用户在分布式数据库环境下，确保每一次数据库数据操作行为，都能够令结果保证数据的一致性和完整性，在并发请求下互相隔离不受干扰。

MatrixOne 的事务类型

在 MatrixOne 中，事务与通用事务一样，也分为以下两大类：

- 按照是否有明确的起止分为显式事务和隐式事务。
- 按照对资源锁的使用阶段分为乐观事务和悲观事务。

这两大类事务的分类彼此不受对方限制，一个显式事务可以是乐观事务或悲观事务，同时一个悲观事务可能是显式事务也可能是隐式事务。

Note: MatrixOne 暂不支持悲观事务。

显式事务

在 MatrixOne 中，一个事务以 `START TRANSACTION` 显式声明，即成为一个显式事务。

隐式事务

在 MatrixOne 中，如果一个事务并没有通过 `START TRANSACTION` 或 `BEGIN` 来显式声明，那么为隐式事务。

乐观事务

在乐观事务开始时，会假定事务相关的表处于一个不会发生写冲突的状态，把对数据的插入、修改或删除缓存在内存中，在这一阶段不会对数据加锁，而在数据提交时对相应的数据表或数据行上锁，在完成提交后解锁。

MatrixOne 的事务隔离级别

快照隔离

与 SQL 标准所定义的四个隔离级别不同，在 MatrixOne 中，支持的隔离级别是快照隔离（Snapshot Isolation），该级别的隔离在 SQL-92 标准的 **REPEATABLE READ** 和 **SERIALIZABLE** 之间。与其他隔离级别有所区别的是，快照隔离具备如下特性：

- 快照隔离对于指定事务内读取的数据不会反映其他同步的事务对数据所做的更改。指定事务使用本次事务开始时读取的数据行。
- 读取数据时不会对数据进行锁定，因此快照事务不会阻止其他事务写入数据。
- 写入数据的事务也不会阻止快照事务读取数据。

与其他隔离级别相比，快照隔离对于脏读 (读取未提交数据)、脏写（写了修改后未提交的记录）、幻读 (前后多次读取，数据总量不一致) 等场景也实现了有效回避：

Isolation Level	P0 Dirty Write	P1 Dirty Read	P4C Cursor Lost Update	P4 Lost Update	P2 Fuzzy Read	P3 Phantom	A5A Read Skew	A5B Write Skew
MatrixOne's Snapshot Isolation	Not Possible	Not Possible	Not Possible	Not Possible	Not Possible	Not Possible	Not Possible	Possible

MatrixOne 的显式事务

在 MatrixOne 的事务类别中，显式事务还遵循以下规则：

显式事务规则

- 显式事务是指以 `BEGIN...END` 或 `START TRANSACTIONS...COMMIT` 或 `ROLLBACK` 作为起始结束。
- 在显式事务中，DML（Data Manipulation Language，数据操纵语言）与 DDL（Data Definition Language，数据库定义语言）可以同时存在，但是 DDL 仅限于 `CREATE`，其他一律无法支持。
- 显式事务中，无法嵌套其他显式事务，例如 `START TRANSACTIONS` 之后再遇到 `START TRANSACTIONS`，两个 `START TRANSACTIONS` 之间的所有语句都会强制提交，无论 `AUTOCOMMIT` 的值是 1 或 0。
- 显式事务中，只能包含 DML 与 DDL，不能带有修改参数配置或管理命令，如 `set [parameter] = [value]`，`create user` 等等。
- 显式事务中，如果在未发生显式提交或回滚而开启一个新事务而发生写写冲突，之前未提交的事务将会回滚并报错。

显式事务示例

```
CREATE TABLE t1(a bigint, b varchar(10), c varchar(10));
START TRANSACTION;
INSERT INTO t1 values(1,2,3);
COMMIT;
```

隐式事务

在 MatrixOne 的事务类别中，隐式事务还遵循以下规则：

隐式事务规则

- 在 `AUTOCOMMIT` 发生变化的时候，隐式事务未提交的情况下，mo 会报错并提示用户需要提交后变化。
- `AUTOCOMMIT=0`，且在当前没有活跃事务的情况下，DDL（Data Definition Language，数据库定义语言）与参数配置文件均可执行，但 `CREATE` 以外的行为不能回滚。
- 在 `AUTOCOMMIT=1` 的情况下，每一条 DML（Data Manipulation Language，数据操纵语言）语句都是一个单独的事务，在执行后立即提交。
- 在 `AUTOCOMMIT=0` 的情况下，每一条 DML 语句都不会在执行后立即提交，需要手动进行 `COMMIT` 或 `ROLLBACK`，如果在尚未提交或回滚的状态下退出客户端，则默认回滚。
- 在 `AUTOCOMMIT=0` 的情况下，DML 与 DDL 可以同时存在，但是 DDL 仅限于 `CREATE`，其他一律无法支持。
- 在隐式事务存在未提交的内容时，开启一个显式事务，会强制提交之前未提交的内容。

隐式事务示例

例如，在上一个显式事务结束后，继续对 *t1* 插入数据 (4,5,6)，即成为一个隐式事务。而该隐式事务是否立即提交，取决于 `AUTOCOMMIT` 参数的值：

```
CREATE TABLE t1(a bigint, b varchar(10), c varchar(10));
START TRANSACTION;
INSERT INTO t1 values(1,2,3);
COMMIT;

-- 查看 AUTOCOMMIT 开关参数
mysql> SHOW VARIABLES LIKE 'AUTOCOMMIT';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| autocommit | 1 |
+---------------+-------+
1 row in set (0.00 sec)
-- 此处开始一个隐式事务，在 AUTOCOMMIT=1 的情况下，每一条 DML 在执行后立即提交
insert into t1 values(4,5,6);

-- 隐式事务自动提交，表结构如下所示
mysql> select * from t1;
+------+------+------+
| a | b | c |
+------+------+------+
| 1 | 2 | 3 |
| 4 | 5 | 6 |
+------+------+------+
2 rows in set (0.00 sec)
```

MatrixOne 的乐观事务

乐观事务原理

乐观事务事务开始时，不会做冲突检测或锁，会将当前相关数据缓存至对应内存区域，并对该数据进行增删改。

在完成修改后，进入提交阶段时，将分为两个步骤进行提交：

步骤一：将待写数据中的某一列当做主键列，并对该列上锁并创建时间戳。基于此时间戳之后对相关行进行的写入均判定为写冲突。

步骤二：写入数据，并且记录此时的时间戳，解开锁。

乐观事务模型

MatrixOne 支持乐观事务模型。你在使用乐观并发读取一行时不会锁定该行。当你想要更新一行时，应用程序必须确定其他用户是否在读取该行后对该行进行了。乐观并发事务通常用于数据争用较低的环境中。

在乐观并发模型中，如果你从数据库接收到一个值后，另一个用户在你试图修改该值之前修改了该值，则产生报错。

模型示例

下面为乐观并发的示例，将为你展示 MatrixOne 如何解决并发冲突。

1. 在下午 1:00，用户 1 从数据库中读取一行，其值如下：

```
CustID LastName FirstName
101 Smith Bob
```

Column name	Original value	Current value	Value in database
CustID	101	101	101
LastName	Smith	Smith	Smith
FirstName	Bob	Bob	Bob

2. 在下午 1:01，用户 2 从数据库中读取同一行。

3. 在下午 1:03，用户 2 将 FirstName 列的 “Bob” 改为 “Robert”，并更新到数据库里。

Column name	Original value	Current value	Value in database
CustID	101	101	101
LastName	Smith	Smith	Smith
FirstName	Bob	Robert	Bob

4. 如上表所示，更新时数据库中的值与用户 2 的原始值匹配，表示更新成功。

5. 在下午 1:05，用户 1 将 FirstName 列的 “Bob” 改为 “James”，并尝试进行更新。

Column name	Original value	Current value	Value in database
CustID	101	101	101
LastName	Smith	Smith	Smith
FirstName	Bob	James	Robert

6. 此时，用户 1 遇到了乐观并发冲突，因为数据库中的值 “Robert” 不再与用户 1 期望的原始值 “Bob” 匹配，并发冲突提示更新失败。下一步需要决定，是采用用户 1 的更改覆盖用户 2 的更改，还是取消用户 1 的更改。

如何使用 MatrixOne 事务

本章节向你介绍如何简单的开启、提交、回滚一个事务，以及如何自动提交事务。

开启事务

开启事务可以通过使用 `START TRANSACTION` 开始一个事务，也可以用方言命令 `BEGIN`。 代码示例：

```
START TRANSACTION;
insert into t1 values(123,'123');
```

或：

```
BEGIN;
insert into t1 values(123,'123');
```

提交事务

提交事务时，MatrixOne 接受 `COMMIT` 命令作为提交命令。代码示例如下：

```
START TRANSACTION;
insert into t1 values(123,'123');
commit;
```

回滚事务

回滚事务时，MatrixOne 接受 `ROLLBACK` 命令作为提交命令。代码示例如下：

```
START TRANSACTION;
insert into t1 values(123,'123');
rollback;
```

自动提交

在 MatrixOne 中，有一个参数 `AUTOCOMMIT`，决定了没有 `START TRANSACTION` 或 `BEGIN` 的情况下，没有单条 SQL 语句的是否被当做独立事务自动提交。语法如下：

```
-- 设置该参数的值
SET AUTOCOMMIT={on|off|0|1}
SHOW VARIABLES LIKE 'AUTOCOMMIT';
```

在该参数设置为 ON 或 1 的时候，意味着自动提交，所有不在 `START TRANSACTION` 或 `BEGIN` 中的单条 SQL 语句，都会在执行时自动提交。

```
-- 此时自动提交
insert into t1 values(1,2,3);
```

在该参数设置为 OFF 或 0 的时候，意味着非自动提交，所有不在 `START TRANSACTION` 或 `BEGIN` 中的 SQL 语句，需要用 `COMMIT` 或 `ROLLBACK` 来执行提交或回滚。

```
insert into t1 values(1,2,3);
-- 此处需要手动提交
COMMIT;
```

MatrixOne 快照隔离

在 MatrixOne 中，支持的隔离级别是快照隔离（Snapshot Isolation），为了与 MySQL 隔离级别保持一致，MatixOne 快照隔离又叫做可重复读（REPEATABLE READS）。该级别的隔离实现原理如下：

快照隔离原理

- 当一个事务开始时，数据库会为该事务生成一个事务 ID，这是一个独一无二的 ID。
- 在生成该事务 ID 的时间戳，生成一个对应数据的快照，此时事务的所有操作都是基于该快照来执行。
- 当事务提交完成对数据的修改后，释放事务 ID 与数据快照。

快照隔离示例

你可以参加下面的示例，来帮助理解快照隔离。

首先在 MatrixOne 中，我们建立一个数据库 *test* 与表 *t1*：

```
create database test;
use test;
CREATE TABLE t1
(
tid INT NOT NULL primary key,
tname VARCHAR(50) NOT NULL
);
INSERT INTO t1 VALUES(1,'version1');
INSERT INTO t1 VALUES(2,'version2');
```

在会话 1 中，开启一个事务：

```
use test;
begin;
UPDATE t1 SET tname='version3' WHERE tid=2;
SELECT * FROM t1;
```

在会话 1 中，我们可以看到的结果是如下，根据快照的数据所进行的修改结果：

```
+------+----------+
| tid | tname |
+------+----------+
| 2 | version3 |
| 1 | version1 |
+------+----------+
```

此时开启会话 2，去查询 t1 的内容：

```
use test;
SELECT * FROM t1;
```

看到的结果仍然是原始数据：

```
+------+----------+
| tid | tname |
+------+----------+
| 1 | version1 |
| 2 | version2 |
+------+----------+
```

在会话 1 中，我们将事务提交：

```
COMMIT;
```

此时，在会话 2 中查询 *t1* 的内容就变成了提交后的数据：

```
SELECT * FROM t1;
+------+----------+
| tid | tname |
+------+----------+
| 1 | version1 |
| 2 | version3 |
+------+----------+
```

MVCC

MVCC（Multiversion Concurrency Control，多版本并发控制）应用于 MatrixOne，以保证事务快照隔离，实现事务的隔离性。

基于数据元组（Tuple，即表中的每行）的指针字段来创建一个 Latch Free 的链表，称为版本链。这个版本链允许数据库定位一个 Tuple 的所需版本。因此这些版本数据的存放机制是数据库存储引擎设计的一个重要考量。

一个方案是采用 Append Only 机制，一个表的所有 Tuple 版本都存储在同一个存储空间。这种方法被用于 Postgre SQL，为了更新一个现有的 Tuple，数据库首先为新的版本从表中获取一个空的槽（Slot），然后，它将当前版本的内容复制到新版本。最后，它在新分配的 Slot 中应用对 Tuple 的修改。Append Only 机制的关键决定是如何为 Tuple 的版本链排序，由于不可能维持一个无锁（Lock free）的双向链表，因此版本链只指向一个方向，或者从 Old 到 New（O2N），或者从 New 到 Old（N2O）。

另外一个类似的方案称为时间旅行（Time Travel），它会把版本链的信息单独存放，而主表维护主版本数据。

第三种方案，是在主表中维护 Tuple 的主版本，在一个单独的数据库对比工具（Delta）存储中维护一系列 Delta 版本。这种存储在 MySQL 和 Oracle 中被称为回滚段。为了更新一个现有的 Tuple，数据库从 Delta 存储中获取一个连续的空间来创建一个新的 Delta 版本。这个 Delta 版本包含修改过的属性的原始值，而不是整个 Tuple。然后数据库直接对主表中的主版本进行原地更新（In Place Update）。

image-20221026152318567

MatrixOne 中的事务应用场景

在一个财务系统中，不同用户之间的转账是非常常见的场景，而转账在数据库中的实际操作，通常是两个步骤，首先是对一个用户的账面金额抵扣之后，然后是对另一个用户的账面金额进行增加。只有利用事务的原子性，才能确保总账面资金没有变化，同时两个用户之间的账户都完成了各自的抵扣与增加，例如 A 用户此时对 B 用户转账 50：

```
start transaction;
update accounts set balance=balance-50 where name='A';
update accounts set balance=balance+50 where name='B';
commit;
```

当两次 `update` 都成功并且最终提交，整个转账才算真正完成，任何一步的失败，必须让整个事务回滚，才能确保原子性。

此外，两个账户转账的过程中，在没有提交之前，无论是 A 或者 B，看到的都是尚未转账完成的账面余额，这是事务的隔离性。

在转账过程中，数据库会检查 A 的账面资金是否大于 50，A 和 B 在系统中是否确有其人，只有都满足这些约束，才能保证事务的一致性。

完成转账后，无论系统是否重启，数据已经完成了持久化，体现了事务的持久性。

MatrixOne 分布式集群部署

本篇文档将主要讲述如何从 0 开始部署一个基于**私有化 Kubernetes 集群的云原生存算分离的分布式数据库 MatrixOne**。

主要步骤

1. 部署 Kubernetes 集群
2. 部署对象存储 MinIO
3. 创建并连接 MatrixOne 集群

名词解释

由于该文档会涉及到众多 Kubernetes 相关的名词，为了让大家能够理解搭建流程，这里对涉及到的重要名词进行简单解释，如果需要详细了解 Kubernetes 相关的内容，可以直接参考 [Kubernetes 中文社区 | 中文文档]

- Pod

  Pod 是 Kubernetes 中最小的资源管理组件，Pod 也是最小化运行容器化应用的资源对象。一个 Pod 代表着集群中运行的一个进程。简单理解，我们可以把一组提供特定功能的应用成为一个 pod，它会包含一个或者多个容器对象，共同对外提供服务。

- Storage Class

  Storage Class，简称 **SC**，用于标记存储资源的特性和性能，管理员可以将存储资源定义为某种类别，正如存储设备对于自身的配置描述（Profile）。根据 SC 的描述可以直观的得知各种存储资源的特性，就可以根据应用对存储资源的需求去申请存储资源了。

- PersistentVolume

  PersistentVolume，简称 **PV**，PV 作为存储资源，主要包括存储能力、访问模式、存储类型、回收策略、后端存储类型等关键信息的设置。

- PersistentVolumeClaim

  PersistentVolumeClaim，简称 **PVC**，作为用户对存储资源的需求申请, 主要包括存储空间请求、访问模式、PV 选择条件和存储类别等信息的设置。

1. 部署 Kubernetes 集群

由于 MatrixOne 的分布式部署依赖于 Kubernetes 集群，因此我们需要一个 Kubernetes 集群。本篇文章将指导你通过使用 **Kuboard-Spray** 的方式搭建一个 Kubernetes 集群。

准备集群环境

对于集群环境，需要做如下准备：

- 3 台 VirtualBox 虚拟机
- 操作系统使用 Ubuntu 20.04 (默认不允许 root 账号远程登入，请预先修改 `sshd` 的配置文件，允许 root 远程登入)：其中两台作为部署 Kubernetes 以及 MatrixOne 相关依赖环境的机器，另外一台作为跳板机，来搭建 Kubernetes 集群。

各个机器情况分布具体如下所示：

host	IP	mem	cpu	disk	role
kuboardspray	192.168.56.9	2G	1C	50G	跳板机
master0	192.168.56.10	4G	2C	50G	master etcd
node0	192.168.56.11	4G	2C	50G	worker

跳板机部署 Kuboard Spray

Kuboard-Spray 是用来可视化部署 Kubernetes 集群的一个工具。它会使用 Docker 快速拉起一个能够可视化部署 Kubernetes 集群的 Web 应用。Kubernetes 集群环境部署完成后，可以将该 Docker 应用停掉。

跳板机环境准备

- 安装 Docker

由于会使用到 Docker，因此需要具备 Docker 的环境。使用以下命令在跳板机安装并启动 Docker：

```
sudo apt-get update && sudo apt-get install -y docker.io
```

环境准备完成后，即可部署 Kuboard-Spray。

部署 Kuboard-Spray

执行以下命令安装 Kuboard-Spray：

```
docker run -d \
 --privileged \
 --restart=unless-stopped \
 --name=kuboard-spray \
 -p 80:80/tcp \
 -v /var/run/docker.sock:/var/run/docker.sock \
 -v ~/kuboard-spray-data:/data \
eipwork/kuboard-spray:v1.2.2-amd64
```

如果由于网络问题导致镜像拉取失败，可以使用下面的备用地址：

```
docker run -d \
 --privileged \
 --restart=unless-stopped \
 --name=kuboard-spray \
 -p 80:80/tcp \
 -v /var/run/docker.sock:/var/run/docker.sock \
 -v ~/kuboard-spray-data:/data \
swr.cn-east-2.myhuaweicloud.com/kuboard/kuboard-spray:latest-amd64
```

执行完成后，即可在浏览器输入 `http://192.168.56.9`（跳板机 IP 地址）打开 Kuboard-Spray 的 Web 界面，输入用户名 `admin`，默认密码 `Kuboard123`，即可登录 Kuboard-Spray 界面，如下所示：

登录之后，即可开始部署 Kubernetes 集群。

可视化部署 Kubernetes 集群

登录 Kuboard-Spray 界面之后，即可开始可视化部署 Kubernetes 集群。

导入 Kubernetes 相关资源包

安装界面会通过在线下载的方式，下载 Kubernetes 集群所对应的资源包，以实现离线安装 Kubernetes 集群。

1. 点击**资源包管理**，选择对应版本的 Kubernetes 资源包下载：

下载 `spray-v2.19.0c_Kubernetes-v1.24.10_v2.9-amd64` 版本

2. 点击**导入**后，选择**加载资源包**，选择合适的下载源，等待资源包下载完成。

3. 此时会 `pull` 相关的镜像依赖：

4. 镜像资源包拉取成功后，返回 Kuboard-Spray 的 Web 界面，可以看到对应版本的资源包已经导入完成。

安装 Kubernetes 集群

本章节将指导你进行 Kubernetes 集群的安装。

1. 选择**集群管理**，选择**添加集群安装计划**：

2. 在弹出的对话框中，定义集群的名称，选择刚刚导入的资源包的版本，再点击**确定**。如下图所示：

集群规划

按照事先定义好的角色分类，Kubernetes 集群采用 `1 master + 1 worker +1 etcd` 的模式进行部署。

在上一步定义完成集群名称，并选择完成资源包版本，点击**确定**之后，接下里可以直接接入到集群规划阶段。

1. 选择对应节点的角色和名称：

   - master 节点：选择 ETCD 和控制节点；名称填写 master0
   - worker 节点：仅选择工作节点；名称填写 node0

2. 在每一个节点填写完角色和节点名称后，请在右侧填写对应节点的连接信息，如下图所示：

3. 填写完所有的角色之后，点击**保存**。接下里就可以准备安装 Kubernetes 集群了。

开始安装 Kubernetes 集群

在上一步填写完成所有角色，并**保存**后，点击**执行**，即可开始 Kubernetes 集群的安装。

1. 如下图所示，点击**确定**，开始安装 Kubernetes 集群：

2. 安装 Kubernetes 集群时，会在对应节点上执行 `ansible` 脚本，安装 Kubernetes 集群。整体事件会根据机器配置和网络不同，需要等待的时间不同，一般情况下需要 5 ~ 10 分钟。

   Note: 如果出现错误，你可以看日志的内容，确认是否是 Kuboard-Spray 的版本不匹配，如果版本不匹配，请更换合适的版本。

3. 安装完成后，到 Kubernetes 集群的 master 节点上执行 `kubectl get node`：

4. 命令结果如上图所示，即表示 Kubernetes 集群安装完成。

2. 部署 helm

Operator 的安装依赖于 helm，因此需要先安装 helm。

Note: 本章节均是在 master0 节点操作。

1. 下载 helm 安装包：

   ```
   wget https://get.helm.sh/helm-v3.10.2-linux-amd64.tar.gz
   ```

   如果由于网络问题造成下载缓慢，你可以到官网下载最新的二进制安装包，上传到服务器。

2. 解压并安装：

```
tar -zxf helm-v3.10.2-linux-amd64.tar.gz
mv linux-amd64/helm /usr/local/bin/helm
```

3. 验证版本，查看是否安装完成：

```
[root@k8s01 home]# helm version
version.BuildInfo\{Version:"v3.10.2",
GitCommit:"50f003e5ee8704ec937a756c646870227d7c8b58",
GitTreeState:"clean", GoVersion:"go1.18.8"}
```

出现上面所示版本信息即表示安装完成。

3. CSI 部署

CSI 为 Kubernetes 的存储插件，为 MinIO 和 MarixOne 提供存储服务。本章节将指导你使用 `local-path-provisioner` 插件。

Note: 本章节均是在 master0 节点操作。

1. 使用下面的命令行，安装 CSI：

```
wget https://github.com/rancher/local-path-
provisioner/archive/refs/tags/v0.0.23.zip
unzip v0.0.23.zip
cd local-path-provisioner-0.0.23/deploy/chart/local-path-provisioner
helm install --set nodePathMap[0].paths[0]="/opt/local-path-
provisioner",nodePathMap[0].node=DEFAULT_PATH_FOR_NON_LISTED_NODES --
create-namespace --namespace local-path-storage local-path-storage ./
```

2. 安装成功后，命令行显示如下所示：

```
root@master0:~# kubectl get pod -n local-path-storage
NAME READY STATUS RESTARTS AGE
local-path-storage-local-path-provisioner-57bf67f7c-lcb88 1/1 Running
0 89s
```

Note: 安装完成后，该 storageClass 会在 worker 节点的 “/opt/local-path-provisioner” 目录提供存储服务。你可以修改为其它路径。

3. 设置缺省 `storageClass`：

```
kubectl patch storageclass local-path -p '\{"metadata": \
{"annotations":\{"storageclass.kubernetes.io/is-default-
class":"true"}}}'
```

4. 设置缺省成功后，命令行显示如下：

```
root@master0:~# kubectl get storageclass
NAME PROVISIONER RECLAIMPOLICY VOLUMEBINDINGMODE ALLOWVOLUMEEXPANSION
AGE
local-path (default) cluster.local/local-path-storage-local-path-
provisioner Delete WaitForFirstConsumer true 115s
```

4. MinIO 部署

MinIO 的作用是为 MatrixOne 提供对象存储。本章节将指导你部署一个单节点的 MinIO。

Note: 本章节均是在 master0 节点操作。

安装启动

1. 安装并启动 MinIO 的命令行如下：

```
helm repo add minio https://charts.min.io/
helm install --create-namespace --namespace mostorage --set
resources.requests.memory=512Mi --set replicas=1 --set
persistence.size=10G --set mode=standalone --set
rootUser=rootuser,rootPassword=rootpass123 --set
consoleService.type=NodePort minio minio/minio
```

> **注意**
>
> - `--set resources.requests.memory=512Mi` 设置了 MinIO 的内存最低消耗
> - `--set persistence.size=1G` 设置了 MinIO 的存储大小为 1G
> - `--set rootUser=rootuser,rootPassword=rootpass123` 这里的 rootUser 和 rootPassword 设置的参数，在后续创建 Kubernetes 集群的 scrects 文件时，需要用到，因此使用一个能记住的信息。

:::

2. 安装并启动 MinIO 成功后，命令行显示如下所示：

   然后，执行下面的命令行，使 mo-log 连接至 9000 端口：

   ```
   nohup kubectl port-forward --address 0.0.0.0 pod-name -n mostorage
   9000:9000 &
   ```

3. 启动后，使用 http:// Kubernetes 集群任何一台机器的 ip:32001 即可登录 MinIO 的页面，创建对象存储的信息。如下图所示，账户密码即上述步骤中 `--set rootUser=rootuser,rootPassword=rootpass123` 设置的 rootUser 和 rootPassword：

4. 登录完成后，你需要创建对象存储相关的信息：

   a. 点击 **Bucket > Create Bucket**，在 **Bucket Name** 中填写 Bucket 的名称 **minio-mo**。填写完成后，点击右下方按钮 **Create Bucket**。

   b. 在当前 **minio-mo** 中，点击 **Choose or create a new path**，在 **New Folder Path** 中填写名称 **test**，填写完成后，点击 **Create**，即完成创建。

5. MatrixOne 集群部署

本章节将指导你部署 MatrixOne 集群。

Note: 本章节均是在 master0 节点操作。

安装 matrixone-operator

使用如下命令行安装 matrixone-operator：

```
wget https://github.com/matrixorigin/matrixone-operator/releases/download/0.7.0-alpha.1/matrixone-operator-0.7.0-alpha.1.tgz
tar -xvf matrixone-operator-0.7.0-alpha.1.tgz
cd /root/matrixone-operator/
helm install --create-namespace --namespace mo-hn matrixone-operator ./ --dependency-update
```

安装成功后，使用如下命令行进行再次确认：

```
root@master0:~# kubectl get pod -n mo-hn
NAME READY STATUS RESTARTS AGE
matrixone-operator-66b896bbdd-qdfrp 1/1 Running 0 2m28s
```

如上上代码行所示，对应 Pod 状态均正常。

创建 MatrixOne 集群

自定义 MatrixOne 集群的 `yaml` 文件，示例如下：

1. 编写如下 `mo.yaml` 的文件：

```
apiVersion: core.matrixorigin.io/v1alpha1
kind: MatrixOneCluster
metadata:
name: mo
namespace: mo-hn
spec:
dn:
config: |
[dn.Txn.Storage]
backend = "TAE"
log-backend = "logservice"
[dn.Ckp]
flush-interval = "60s"
min-count = 100
scan-interval = "5s"
incremental-interval = "60s"
global-interval = "100000s"
[log]
level = "error"
format = "json"
max-size = 512
replicas: 1
logService:
replicas: 3
sharedStorage:
s3:
type: minio
path: minio
endpoint: http://minio.mostorage:9000
secretRef:
name: minio
pvcRetentionPolicy: Retain
volume:
size: 1Gi
config: |
[log]
level = "error"
format = "json"
max-size = 512
tp:
serviceType: NodePort
config: |
[cn.Engine]
type = "distributed-tae"
```

```
[log]
level = "debug"
format = "json"
max-size = 512
replicas: 1
version: nightly-556de418
imageRepository: matrixorigin/matrixone
imagePullPolicy: Always
```

2. 定义 MatrixOne 访问 MinIO 的 secret 服务：

```
kubectl -n mo-hn create secret generic minio --from-literal=AWS_ACCESS_KEY_ID=rootuser --from-literal=AWS_SECRET_ACCESS_KEY=rootpass123
```

用户名和密码使用创建 MinIO 集群时设定的 rootUser 和 rootPassword。

3. 使用如下命令行部署 MatrixOne 集群：

```
kubectl apply -f mo.yaml
```

4. 需等待 10 来分钟，如发生 pod 重启，请继续等待。直到如下显示表示部署成功：

```
root@k8s-master0:~# kubectl get pods -n mo-hn
NAME READY STATUS RESTARTS AGE
matrixone-operator-66b896bbdd-qdfrp 1/1 Running 1 (99m ago) 10h
mo-dn-0 1/1 Running 0 46m
mo-log-0 1/1 Running 0 47m
mo-log-1 1/1 Running 0 47m
mo-log-2 1/1 Running 0 47m
mo-tp-cn-0 1/1 Running 1 (45m ago) 46m
```

6. 连接 MatrixOne 集群

由于提供对外访问的 CN 的 pod id 不是 node ip，因此，你需要将对应服务的端口映射到 MatrixOne 节点上。本章节将指导你使用 `kubectl port-forward` 连接 MatrixOne 集群。

- 只允许本地访问：

```
nohup kubectl port-forward svc/mo-tp-cn 6001:6001 &
```

- 指定某台机器或者所有机器访问：

```
nohup kubectl port-forward --address 0.0.0.0 svc/mo-tp-cn 6001:6001 &
```

指定**允许本地访问**或**指定某台机器或者所有机器访问**后，使用 MySQL 客户端连接 MatrixOne：

```
mysql -h $(kubectl get svc/mo-tp-cn -n mo-hn -o jsonpath='\
{.spec.clusterIP}') -P 6001 -udump -p111
mysql: [Warning] Using a password on the command line interface can be
insecure.
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 1004
Server version: 638358 MatrixOne

Copyright (c) 2000, 2019, 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>
```

显式 `mysql>` 后，分布式的 MatrixOne 集群搭建连接完成。

Java CRUD 示例

> **注意**
>
> 本篇文档所介绍的演示程序的源代码下载地址为：Java CRUD 示例。

配置环境

在开始之前，请确保已经下载并安装了以下软件。

- 完成单机部署 MatrixOne，通过 MySQL 客户端创建数据库。

```
mysql> create database test;
```

- 下载安装 IntelliJ IDEA(2022.2.1 or later version)。
- 根据你的系统环境选择 JDK 8+ version 版本进行下载安装。
- MySQL JDBC connector 8.0+ version：推荐下载平台独立版本，并解压下载文件。

image-20220927102516885

> **注意**
>
> 我们使用 IDEA 作为一个 IDE 示例来演示这个过程，你可以自由地选择 Eclipse 或其他 IDE 工具实践。

初始化一个新的 Java 项目

启动 IDEA，并创建一个新的 Java 项目，如下所示：

image-20220927104740221

进入菜单 **Project Setting > Libraries**，导入 *mysql-connector-java-8.0.30.jar* 文件。

image-20220927104904770

编写 Java 代码连接 MatrixOne

首先，创建一个名为 *JDBCUtils* 的 Java 类作为连接实用程序。这个类将作为连接 MatrixOne 和执行 SQL 查询的工具。

在 *src* 目录下，创建一个名为 `JDBCUtils.java` 的文件，并使用以下代码编辑该文件：

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class JDBCUtils \{
private static String jdbcURL = "jdbc:mysql://127.0.0.1:6001/test";
private static String jdbcUsername = "dump";
private static String jdbcPassword = "111";

public static Connection getConnection() \{
Connection connection = null;
try \{
connection = DriverManager.getConnection(jdbcURL, jdbcUsername,
jdbcPassword);
} catch (SQLException e) \{
// TODO Auto-generated catch block
e.printStackTrace();
}
return connection;
}

public static void printSQLException(SQLException ex) \{
for (Throwable e : ex) \{
if (e instanceof SQLException) \{
e.printStackTrace(System.err);
System.err.println("SQLState: " + ((SQLException) e).getSQLState());
System.err.println("Error Code: " + ((SQLException) e).getErrorCode());
System.err.println("Message: " + e.getMessage());
Throwable t = ex.getCause();
while (t != null) \{
System.out.println("Cause: " + t);
t = t.getCause();
}
}
}
}
}
```

其次，我们用 MatrixOne 编写创建、插入、更新和删除操作的示例代码。

我们需要在 *src* 目录下创建相应的 java 源代码文件：`Create.java`、`Insert.java`、`Update.java`、`Select.java`，并将下面的代码对应放在这些文件中。

创建（`Create.java`）

```
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;

public class Create \{
private static final String createTableSQL = "create table student (\r\n"
+ " id int primary key,\r\n" +
" name varchar(20),\r\n" + " email varchar(20),\r\n" + " country
varchar(20),\r\n" +
" age int\r\n" + " );";

public static void main(String[] argv) throws SQLException \{
Create createTable = new Create();
createTable.createTable();
}

public void createTable() throws SQLException \{

System.out.println(createTableSQL);
// Step 1: Establishing a Connection
try (Connection connection = JDBCUtils.getConnection();
// Step 2:Create a statement using connection object
Statement statement = connection.createStatement();) \{

// Step 3: Execute the query or update query
statement.execute(createTableSQL);
} catch (SQLException e) \{

// print SQL exception information
JDBCUtils.printSQLException(e);
}

// Step 4: try-with-resource statement will auto close the connection.
}
}
```

执行上述代码会在 `test` 数据库中创建一个表，然后你可以在 MySQL 客户端中使用如下代码验证是否创建了表。

```
mysql> show create table student;
+---------+-----------------------------------------------------------------
-------------------------------------------------------------------------
-------------------------------------------------------------+
| Table | Create Table |
+---------+-----------------------------------------------------------------
-------------------------------------------------------------------------
-------------------------------------------------------------+
| student | CREATE TABLE `student` (
`id` INT DEFAULT NULL,
`name` VARCHAR(20) DEFAULT NULL,
`email` VARCHAR(20) DEFAULT NULL,
`country` VARCHAR(20) DEFAULT NULL,
`age` INT DEFAULT NULL,
PRIMARY KEY (`id`)
) |
+---------+-----------------------------------------------------------------
-------------------------------------------------------------------------
-------------------------------------------------------------+
1 row in set (0.01 sec)
```

插入（`Insert.java`）

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class Insert \{
private static final String INSERT_STUDENT_SQL = "INSERT INTO student" +
" (id, name, email, country, age) VALUES " +
" (?, ?, ?, ?, ?);";

public static void main(String[] argv) throws SQLException \{
Insert insertTable = new Insert();
insertTable.insertRecord();
}

public void insertRecord() throws SQLException \{
System.out.println(INSERT_STUDENT_SQL);
// Step 1: Establishing a Connection
try (Connection connection = JDBCUtils.getConnection();
// Step 2:Create a statement using connection object
PreparedStatement preparedStatement =
connection.prepareStatement(INSERT_STUDENT_SQL)) \{
preparedStatement.setInt(1, 1);
preparedStatement.setString(2, "Tony");
preparedStatement.setString(3, "tony@gmail.com");
preparedStatement.setString(4, "US");
preparedStatement.setString(5, "20");

System.out.println(preparedStatement);
// Step 3: Execute the query or update query
preparedStatement.executeUpdate();
} catch (SQLException e) \{

// print SQL exception information
JDBCUtils.printSQLException(e);
}

// Step 4: try-with-resource statement will auto close the connection.
}

}
```

执行结果：

```
mysql> select * from student;
+------+------+----------------+---------+------+
| id | name | email | country | age |
+------+------+----------------+---------+------+
| 1 | Tony | tony@gmail.com | US | 20 |
+------+------+----------------+---------+------+
1 row in set (0.01 sec)
```

更新（`Update.java`）

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class Update \{
private static final String UPDATE_STUDENT_SQL = "update student set name
= ? where id = ?;";

public static void main(String[] argv) throws SQLException \{
Update updateTable = new Update();
updateTable.updateRecord();
}

public void updateRecord() throws SQLException \{
System.out.println(UPDATE_STUDENT_SQL);
// Step 1: Establishing a Connection
try (Connection connection = JDBCUtils.getConnection();
// Step 2:Create a statement using connection object
PreparedStatement preparedStatement =
connection.prepareStatement(UPDATE_STUDENT_SQL)) \{
preparedStatement.setString(1, "Ram");
preparedStatement.setInt(2, 1);

// Step 3: Execute the query or update query
preparedStatement.executeUpdate();
} catch (SQLException e) \{

// print SQL exception information
JDBCUtils.printSQLException(e);
}

// Step 4: try-with-resource statement will auto close the connection.
}
}
```

运行结果：

```
mysql> select * from student;
+------+------+----------------+---------+------+
| id | name | email | country | age |
+------+------+----------------+---------+------+
| 1 | Ram | tony@gmail.com | US | 20 |
+------+------+----------------+---------+------+
1 row in set (0.00 sec)
```

查询（`Select.java`）

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class Select \{
private static final String QUERY = "select id,name,email,country,age
from student where id =?";

public static void main(String[] args) \{

// using try-with-resources to avoid closing resources (boiler plate
code)

// Step 1: Establishing a Connection
try (Connection connection = JDBCUtils.getConnection();

// Step 2:Create a statement using connection object
PreparedStatement preparedStatement =
connection.prepareStatement(QUERY);) \{
preparedStatement.setInt(1, 1);
System.out.println(preparedStatement);
// Step 3: Execute the query or update query
ResultSet rs = preparedStatement.executeQuery();

// Step 4: Process the ResultSet object.
while (rs.next()) \{
int id = rs.getInt("id");
String name = rs.getString("name");
String email = rs.getString("email");
String country = rs.getString("country");
String password = rs.getString("age");
System.out.println(id + "," + name + "," + email + "," + country + "," +
password);
}
} catch (SQLException e) \{
JDBCUtils.printSQLException(e);
}
// Step 4: try-with-resource statement will auto close the connection.
}
}
```

运行结果：

image-20220927113440917

使用 SpringBoot 和 Spring Data JPA 构建一个 CRUD 示例

本篇文档将指导你如何使用 **SpringBoot**、**Spring Data JPA** 和 **Intellij IDEA** 构建一个简单的应用程序，并实现 CRUD（创建、读取、更新、删除）功能。

开始之前

本篇教程涉及到的软件介绍如下：

- Spring Data JPA：JPA（Java Persistence API，Java 持久层 API）是一种规范，是 JDK 5.0 注解或 XML 描述对象与关系表的映射关系，并将运行期的实体对象持久化到数据库中。Spring Data JPA 是一个 Java 对象映射关系的解决方案的 ORM（Object-Relational Mapping）框架，是一个将面向对象的域模型映射到关系数据库的开源框架。
- Intellij IDEA：IntelliJ IDEA 是一种商业化销售的 Java 集成开发环境（Integrated Development Environment，IDE）工具软件。它所拥有诸多插件，可以提高我们的工作效率。
- Maven：Maven 是 Java 中功能强大的项目管理工具，可以根据 *pom.xml* 文件中的配置自动下载和导入 *Jar* 文件。这个特性减少了不同版本 Jar 文件之间的冲突。
- Spring：Spring 是 Java 中最流行的框架之一，越来越多的企业使用 Spring 框架来构建他们的项目。Spring Boot 构建在传统的 Spring 框架之上。因此，它提供了 Spring 的所有特性，而且比 Spring 更易用。
- Postman: Postman 是一个用于 API 测试的应用程序。它是一个 HTTP 客户端，利用图形用户界面测试 HTTP 请求，以获得需要进行验证的不同类型的响应。

配置环境

1. 安装构建 MatrixOne

按照步骤介绍完成[安装单机版 MatrixOne]，在 MySQL 客户端新建一个命名为 `test` 数据库。

```
mysql> create database test;
```

2. 使用 IntelliJ IDEA 创建一个新的 Spring Boot 项目

选择 **Spring Initializer**，按需命名项目名称。

image-20221027094625081

选择如下依赖项：

- **Spring Web**
- **JDBC API**
- **Spring Data JPA**
- **MySQL Driver**

image-20221027101504418

点击 **Create** 创建项目。依赖项列在 *pom.xml* 文件中。通常你无需修改任何东西。

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.example</groupId>
<artifactId>jpademo</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>jpademo</name>
<description>jpademo</description>

<properties>
<java.version>1.8</java.version>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
<spring-boot.version>2.3.7.RELEASE</spring-boot.version>
</properties>

<dependencies>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-jdbc</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>

<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<scope>runtime</scope>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
<exclusions>
```

```
<exclusion>
<groupId>org.junit.vintage</groupId>
<artifactId>junit-vintage-engine</artifactId>
</exclusion>
</exclusions>
</dependency>
</dependencies>

<dependencyManagement>
<dependencies>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-dependencies</artifactId>
<version>${spring-boot.version}</version>
<type>pom</type>
<scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>

<build>
<plugins>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-compiler-plugin</artifactId>
<version>3.8.1</version>
<configuration>
<source>1.8</source>
<target>1.8</target>
<encoding>UTF-8</encoding>
</configuration>
</plugin>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
<version>2.3.7.RELEASE</version>
<configuration>
<mainClass>com.example.jpademo.JpademoApplication</mainClass>
</configuration>
<executions>
<execution>
<id>repackage</id>
<goals>
<goal>repackage</goal>
```

```
</goals>
</execution>
</executions>
</plugin>
</plugins>
</build>

</project>
```

3. 修改 *application.properties* 文件

进入到 *resources* 文件目录下，配置 *application.properties* 文件，完成 MatrixOne 连接。

```
# Application Name
spring.application.name=jpademo
# Database driver
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
# Data Source name
spring.datasource.name=defaultDataSource

# Database connection url, modify to MatrixOne address and port, with
parameters
spring.datasource.url=jdbc:mysql://127.0.0.1:6001/test?
characterSetResults=UTF-
8&continueBatchOnError=false&useServerPrepStmts=true&alwaysSendSetIsolation=
# Database username and password
spring.datasource.username=dump
spring.datasource.password=111
# Web application port
server.port=8080

# Hibernate configurations
spring.jpa.properties.hibernate.dialect =
org.hibernate.dialect.MySQLDialect
spring.jpa.properties.hibernate.id.new_generator_mappings = false
spring.jpa.properties.hibernate.format_sql = true
spring.jpa.hibernate.ddl-auto = validate
```

4. 在 MatrixOne 中新建表并插入数据

使用 MySQL 客户端连接到 MatrixOne 并执行以下 SQL 语句。你可以将这些 SQL 语句保存在 */resource/database/* 目录下的 *book.sql* 中。

```
mysql> USE test;
mysql> CREATE TABLE IF NOT EXISTS `book` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`author` varchar(255) DEFAULT NULL,
`category` varchar(255) DEFAULT NULL,
`name` varchar(255) DEFAULT NULL,
`pages` int(11) DEFAULT NULL,
`price` int(11) DEFAULT NULL,
`publication` varchar(255) DEFAULT NULL,
PRIMARY KEY (`id`)
);
mysql> INSERT INTO `book` (`id`, `author`, `category`, `name`, `pages`,
`price`, `publication`) VALUES
(1, 'Antoine de Saint-Exupery', 'Fantancy', 'The Little Prince', 100, 50,
'Amazon'),
(2, 'J. K. Rowling', 'Fantancy', 'Harry Potter and the Sorcerer''s
Stone', 1000, 200, 'Amazon'),
(3, 'Lewis Carroll', 'Fantancy', 'Alice''s Adventures in Wonderland',
1500, 240, 'Amazon');
```

编写代码

完成环境配置后，我们编写代码来实现一个简单的 CRUD 应用程序。

在完成编写编码后，你将拥有一个如下所示的项目结构。你可以预先创建这些包和 java 类。

我们将为这个演示应用程序编写创建、更新、插入、删除和选择操作。

image-20221027105233860

1. BookStoreController.java

```
package com.example.jpademo.controller;

import com.example.jpademo.entity.Book;
import com.example.jpademo.services.IBookStoreService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@Controller
@RequestMapping("bookservice")
public class BookStoreController {

@Autowired
private IBookStoreService service;

@GetMapping("books")
public ResponseEntity<List<Book>> getBooks(){

List<Book> books = service.getBooks();
return new ResponseEntity<List<Book>>(books, HttpStatus.OK);

}

@GetMapping("books/{id}")
public ResponseEntity<Book> getBook(@PathVariable("id") Integer id){
Book book = service.getBook(id);
return new ResponseEntity<Book>(book, HttpStatus.OK);
}

@PostMapping("books")
public ResponseEntity<Book> createBook(@RequestBody Book book){
Book b = service.createBook(book);
return new ResponseEntity<Book>(b, HttpStatus.OK);

}

@PutMapping("books/{id}")
public ResponseEntity<Book> updateBook(@PathVariable("id") int id,
```

```
@RequestBody Book book){

Book b = service.updateBook(id, book);
return new ResponseEntity<Book>(b, HttpStatus.OK);
}

@DeleteMapping("books/{id}")
public ResponseEntity<String> deleteBook(@PathVariable("id") int id){
boolean isDeleted = service.deleteBook(id);
if(isDeleted){
String responseContent = "Book has been deleted successfully";
return new ResponseEntity<String>(responseContent,HttpStatus.OK);
}
String error = "Error while deleting book from database";
return new ResponseEntity<String>
(error,HttpStatus.INTERNAL_SERVER_ERROR);
}

}
```

2. BooStoreDAO.java

```
package com.example.jpademo.dao;

import com.example.jpademo.entity.Book;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.Query;
import java.util.List;

@Transactional
@Repository
public class BookStoreDAO implements IBookStoreDAO {

@PersistenceContext
private EntityManager entityManager;

/**
* This method is responsible to get all books available in database and
return it as List<Book>
*/
@SuppressWarnings("unchecked")
@Override
public List<Book> getBooks() {

String hql = "FROM Book as atcl ORDER BY atcl.id";
return (List<Book>) entityManager.createQuery(hql).getResultList();
}

/**
* This method is responsible to get a particular Book detail by given
book id
*/
@Override
public Book getBook(int bookId) {

return entityManager.find(Book.class, bookId);
}

/**
* This method is responsible to create new book in database
```

```
*/
@Override
public Book createBook(Book book) {
entityManager.persist(book);
Book b = getLastInsertedBook();
return b;
}

/**
* This method is responsible to update book detail in database
*/
@Override
public Book updateBook(int bookId, Book book) {

//First We are taking Book detail from database by given book id and
// then updating detail with provided book object
Book bookFromDB = getBook(bookId);
bookFromDB.setName(book.getName());
bookFromDB.setAuthor(book.getAuthor());
bookFromDB.setCategory(book.getCategory());
bookFromDB.setPublication(book.getPublication());
bookFromDB.setPages(book.getPages());
bookFromDB.setPrice(book.getPrice());

entityManager.flush();

//again i am taking updated result of book and returning the book object
Book updatedBook = getBook(bookId);

return updatedBook;
}

/**
* This method is responsible for deleting a particular(which id will be
passed that record)
* record from the database
*/
@Override
public boolean deleteBook(int bookId) {
Book book = getBook(bookId);
entityManager.remove(book);

//we are checking here that whether entityManager contains earlier
deleted book or not
```

```
// if contains then book is not deleted from DB that's why returning
false;
boolean status = entityManager.contains(book);
if(status){
return false;
}
return true;
}

/**
* This method will get the latest inserted record from the database and
return the object of Book class
* @return book
*/
private Book getLastInsertedBook(){
String hql = "from Book order by id DESC";
Query query = entityManager.createQuery(hql);
query.setMaxResults(1);
Book book = (Book)query.getSingleResult();
return book;
}
}
```

3. IBookStoreDAO.java

```
package com.example.jpademo.dao;

import com.example.jpademo.entity.Book;

import java.util.List;

public interface IBookStoreDAO {

List<Book> getBooks();
Book getBook(int bookId);
Book createBook(Book book);
Book updateBook(int bookId,Book book);
boolean deleteBook(int bookId);
}
```

4. Book.java

```
package com.example.jpademo.entity;

import javax.persistence.*;
import java.io.Serializable;

@Entity
@Table(name="book")
public class Book implements Serializable {

private static final long serialVersionUID = 1L;
@Id
@GeneratedValue(strategy= GenerationType.AUTO)
@Column(name="id")
private int id;

@Column(name="name")
private String name;

@Column(name="author")
private String author;

@Column(name="publication")
private String publication;

@Column(name="category")
private String category;

@Column(name="pages")
private int pages;

@Column(name="price")
private int price;

public int getId() {
return id;
}

public void setId(int id) {
this.id = id;
}

public String getName() {
```

```
return name;
}

public void setName(String name) {
this.name = name;
}

public String getAuthor() {
return author;
}

public void setAuthor(String author) {
this.author = author;
}

public String getPublication() {
return publication;
}

public void setPublication(String publication) {
this.publication = publication;
}

public String getCategory() {
return category;
}

public void setCategory(String category) {
this.category = category;
}

public int getPages() {
return pages;
}

public void setPages(int pages) {
this.pages = pages;
}

public int getPrice() {
return price;
}

public void setPrice(int price) {
```

```
this.price = price;
}

}
```

5. BookStoreService.java

```
package com.example.jpademo.services;

import com.example.jpademo.dao.IBookStoreDAO;
import com.example.jpademo.entity.Book;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class BookStoreService implements IBookStoreService {

@Autowired
private IBookStoreDAO dao;

@Override
public List<Book> getBooks() {
return dao.getBooks();
}

@Override
public Book createBook(Book book) {
return dao.createBook(book);
}

@Override
public Book updateBook(int bookId, Book book) {
return dao.updateBook(bookId, book);
}

@Override
public Book getBook(int bookId) {
return dao.getBook(bookId);
}

@Override
public boolean deleteBook(int bookId) {
return dao.deleteBook(bookId);
}

}
```

6. IBookStoreService.java

```
package com.example.jpademo.services;

import com.example.jpademo.entity.Book;

import java.util.List;

public interface IBookStoreService {

List<Book> getBooks();
Book createBook(Book book);
Book updateBook(int bookId, Book book);
Book getBook(int bookId);
boolean deleteBook(int bookId);

}
```

7. JpademoApplication

```
package com.example.jpademo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class JpademoApplication {

public static void main(String[] args) {
SpringApplication.run(JpademoApplication.class, args);
}

}
```

测试

构建并启动这个项目。

image-20221027110133726

当出现下面的消息时，表示应用程序已经正常启动，你可以使用 Postman 调用 REST 端口。

```
2022-10-27 11:16:16.793 INFO 93488 --- [ main]
com.example.jpademo.JpademoApplication : Starting JpademoApplication on
nandeng-macbookpro.local with PID 93488
(/Users/nandeng/IdeaProjects/jpademo/target/classes started by nandeng in
/Users/nandeng/IdeaProjects/jpademo)
2022-10-27 11:16:16.796 INFO 93488 --- [ main]
com.example.jpademo.JpademoApplication : No active profile set, falling
back to default profiles: default
2022-10-27 11:16:18.022 INFO 93488 --- [ main]
.s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA
repositories in DEFAULT mode.
2022-10-27 11:16:18.093 INFO 93488 --- [ main]
.s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data
repository scanning in 50ms. Found 0 JPA repository interfaces.
2022-10-27 11:16:18.806 INFO 93488 --- [ main]
o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with
port(s): 8080 (http)
2022-10-27 11:16:18.814 INFO 93488 --- [ main]
o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-10-27 11:16:18.814 INFO 93488 --- [ main]
org.apache.catalina.core.StandardEngine : Starting Servlet engine:
[Apache Tomcat/9.0.41]
2022-10-27 11:16:18.886 INFO 93488 --- [ main] o.a.c.c.C.[Tomcat].
[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-10-27 11:16:18.886 INFO 93488 --- [ main]
w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext:
initialization completed in 2005 ms
2022-10-27 11:16:19.068 INFO 93488 --- [ main]
o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing
PersistenceUnitInfo [name: default]
2022-10-27 11:16:19.119 INFO 93488 --- [ main] org.hibernate.Version :
HHH000412: Hibernate ORM core version 5.4.25.Final
2022-10-27 11:16:19.202 INFO 93488 --- [ main]
o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons
Annotations {5.1.2.Final}
2022-10-27 11:16:19.282 INFO 93488 --- [ main]
com.zaxxer.hikari.HikariDataSource : defaultDataSource - Starting...
2022-10-27 11:16:20.025 INFO 93488 --- [ main]
com.zaxxer.hikari.HikariDataSource : defaultDataSource - Start completed.
2022-10-27 11:16:20.035 INFO 93488 --- [ main]
org.hibernate.dialect.Dialect : HHH000400: Using dialect:
org.hibernate.dialect.MySQLDialect
2022-10-27 11:16:21.929 INFO 93488 --- [ main]
o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform
```

```
implementation:
[org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
2022-10-27 11:16:21.937 INFO 93488 --- [ main]
j.LocalContainerEntityManagerFactoryBean : Initialized JPA
EntityManagerFactory for persistence unit 'default'
2022-10-27 11:16:22.073 WARN 93488 --- [ main]
JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is
enabled by default. Therefore, database queries may be performed during
view rendering. Explicitly configure spring.jpa.open-in-view to disable
this warning
2022-10-27 11:16:22.221 INFO 93488 --- [ main]
o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService
'applicationTaskExecutor'
2022-10-27 11:16:22.415 INFO 93488 --- [ main]
o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080
(http) with context path ''
2022-10-27 11:16:22.430 INFO 93488 --- [ main]
com.example.jpademo.JpademoApplication : Started JpademoApplication in
6.079 seconds (JVM running for 8.765)
2022-10-27 11:16:40.180 INFO 93488 --- [nio-8080-exec-1] o.a.c.c.C.
[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet
'dispatcherServlet'
2022-10-27 11:16:40.183 INFO 93488 --- [nio-8080-exec-1]
o.s.web.servlet.DispatcherServlet : Initializing Servlet
'dispatcherServlet'
2022-10-27 11:16:40.249 INFO 93488 --- [nio-8080-exec-1]
o.s.web.servlet.DispatcherServlet : Completed initialization in 66 ms
```

1. 获取 Book 列表，使用 GET 请求调用以下端口

```
http://localhost:8080/bookservice/books
```

image-20221027112426189

2. 创建一个新 Book，使用 POST 请求调用以下端口

```
http://localhost:8080/bookservice/books
```

将 Header 中的内容类型设置为 `application/json`，将 Request Body 设置为原始 JSON 有效负载。

```
{
"name": "The Lion, the Witch and the Wardrobe",
"author": "C. S. Lewis",
"publication": "Amazon",
"category": "Fantancy",
"pages": 123,
"price": 10
}
```

image-20221027115733788

3. 如需获取特定 Book，使用 GET 请求调用以下端口

```
http://localhost:8080/bookservice/books/<id>
```

image-20221027115844378

4. 在数据库中升级 Book，使用 PUT 请求调用以下端口

```
http://localhost:8080/bookservice/books/<id>
```

set content type as in header as `application/json`

set request body as raw with JSON payload

- 将 Header 中的内容类型设置为 `application/json`。
- 将 Request Body 设置为原始 JSON 有效负载

```
{
"name": "Black Beauty",
"author": "Anna Sewell",
"publication": "Amazon",
"category": "Fantancy",
"pages": 134,
"price": 12
}
```

image-20221027120144112

5. 如需从数据库中删除特定的 Book，使用 DELETE 请求调用以下端口

```
http://localhost:8080/bookservice/books/<id>
```

image-20221027120306830

```
mysql> select * from book;
+----+-------------------------+----------+----------------------------
-----+-------+-------+-------------+
| id | author | category | name | pages | price | publication |
+----+-------------------------+----------+----------------------------
-----+-------+-------+-------------+
| 1 | Antoine de Saint-Exupery | Fantancy | The Little Prince | 100 | 50
| Amazon |
| 2 | Anna Sewell | Fantancy | Black Beauty | 134 | 12 | Amazon |
| 3 | Lewis Carroll | Fantancy | Alice's Adventures in Wonderland | 1500
| 240 | Amazon |
+----+-------------------------+----------+----------------------------
-----+-------+-------+-------------+
3 rows in set (0.02 sec)
```

使用 SpringBoot 和 MyBatis 构建一个 CRUD 示例

本篇文档将指导你如何使用 **SpringBoot**、**Mybatis** 和 **Intellij IDEA** 构建一个简单的应用程序，并实现 CRUD（创建、读取、更新、删除）功能。

开始之前

本篇教程涉及到的软件介绍如下：

- MyBatis：MyBatis 是一款优秀的持久层框架，它支持自定义 SQL、存储过程以及高级映射。我们只需要关注项目中的 SQL 本身。
- Intellij IDEA：IntelliJ IDEA 是一种商业化销售的 Java 集成开发环境（Integrated Development Environment，IDE）工具软件。它所拥有诸多插件，可以提高我们的工作效率。
- Maven：Maven 是 Java 中功能强大的项目管理工具，可以根据 *pom.xml* 文件中的配置自动下载和导入 *Jar* 文件。这个特性减少了不同版本 Jar 文件之间的冲突。
- Spring：Spring 是 Java 中最流行的框架之一，越来越多的企业使用 Spring 框架来构建他们的项目。Spring Boot 构建在传统的 Spring 框架之上。因此，它提供了 Spring 的所有特性，而且比 Spring 更易用。

配置环境

1. 安装构建 MatrixOne

按照步骤介绍完成安装单机版 MatrixOne，在 MySQL 客户端新建一个命名为 `test` 数据库。

```
mysql> create database test;
```

2. 使用 IntelliJ IDEA 创建一个新的 Spring Boot 项目

选择 **Spring Initializer**，按需命名项目名称。

image-20221026152318567

选择如下依赖项：

- **Spring Web**
- **MyBatis Framework**
- **JDBC API**
- **MySQL Driver**

image-20221026152447954

点击 **Create** 创建项目。依赖项列在 *pom.xml* 文件中。通常你无需修改任何东西。

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.example</groupId>
<artifactId>mybatis-demo</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>mybatis-demo</name>
<description>mybatis-demo</description>

<properties>
<java.version>1.8</java.version>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
<spring-boot.version>2.3.7.RELEASE</spring-boot.version>
</properties>

<dependencies>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-jdbc</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
<groupId>org.mybatis.spring.boot</groupId>
<artifactId>mybatis-spring-boot-starter</artifactId>
<version>2.1.4</version>
</dependency>

<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<scope>runtime</scope>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
```

```
<exclusions>
<exclusion>
<groupId>org.junit.vintage</groupId>
<artifactId>junit-vintage-engine</artifactId>
</exclusion>
</exclusions>
</dependency>
</dependencies>

<dependencyManagement>
<dependencies>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-dependencies</artifactId>
<version>${spring-boot.version}</version>
<type>pom</type>
<scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>

<build>
<plugins>
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-compiler-plugin</artifactId>
<version>3.8.1</version>
<configuration>
<source>1.8</source>
<target>1.8</target>
<encoding>UTF-8</encoding>
</configuration>
</plugin>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
<version>2.3.7.RELEASE</version>
<configuration>
<mainClass>com.example.mybatisdemo.MybatisDemoApplication</mainClass>
</configuration>
<executions>
<execution>
<id>repackage</id>
<goals>
```

```
<goal>repackage</goal>
</goals>
</execution>
</executions>
</plugin>
</plugins>
</build>

</project>
```

3. 修改 *application.properties* 文件

进入到 *resources* 文件目录下，配置 *application.properties* 文件，完成 MatrixOne 连接。

```
# Application Name
spring.application.name=MyBatisDemo
# Database driver
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
# Data Source name
spring.datasource.name=defaultDataSource

# Database connection url, modify to MatrixOne address and port, with
paratemers
spring.datasource.url=jdbc:mysql://127.0.0.1:6001/test?
characterSetResults=UTF-
8&continueBatchOnError=false&useServerPrepStmts=true&alwaysSendSetIsolation=
# Database username and password
spring.datasource.username=dump
spring.datasource.password=111

# Mybatis mapper location
mybatis.mapper-locations=classpath:mapping/*xml
# Mybatis entity package
mybatis.type-aliases-package=com.example.mybatisdemo.entity
# Web application port
server.port=8080
```

编写代码

完成环境配置后，我们编写代码来实现一个简单的 CRUD 应用程序。

在完成编写编码后，你将拥有一个如下所示的项目结构。你可以预先创建这些包和 java 类。

我们将为这个演示应用程序编写创建、更新、插入、删除和选择操作。

image-20221026155656694

1. UserController.java

```
package com.example.mybatisdemo.controller;

import com.example.mybatisdemo.entity.User;
import com.example.mybatisdemo.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/test")
public class UserController {

String tableName = "user";
@Autowired
private UserService userService;

@RequestMapping(value = "/create", produces =
"application/json;charset=UTF-8", method = RequestMethod.GET)
@ResponseBody
public String createTable(){
return userService.createTable(tableName);
}

@RequestMapping(value = "/selectUserByid", produces =
"application/json;charset=UTF-8", method = RequestMethod.GET)
@ResponseBody
public String GetUser(User user){
return userService.Sel(user).toString();
}

@RequestMapping(value = "/add", produces = "application/json;charset=UTF-
8", method = RequestMethod.GET)
public String Add(User user){
return userService.Add(user);
}

@RequestMapping(value = "/update", produces =
"application/json;charset=UTF-8", method = RequestMethod.GET)
public String Update(User user){
return userService.Update(user);
```

```
}

@RequestMapping(value = "/delete", produces =
"application/json;charset=UTF-8", method = RequestMethod.GET)
public String Delete(User user){
return userService.Delete(user);
}
}
```

2. User.java

```
package com.example.mybatisdemo.entity;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

public class User {
private Integer id;
private String username;
private String password;
private String address;

public User(Integer id, String username, String password, String address)
{
this.id = id;
this.username = username;
this.password = password;
this.address = address;
}

public Integer getId() {
return id;
}

public String getUsername() {
return username;
}

public String getPassword() {
return password;
}

public String getAddress() {
return address;
}

public void setId(Integer id) {
this.id = id;
}

public void setUsername(String username) {
this.username = username;
}
```

```
public void setPassword(String password) {
this.password = password;
}

public void setAddress(String address) {
this.address = address;
}

@Override
public String toString() {
return "User{" +
"id=" + id +
", username='" + username + '\'' +
", password='" + password + '\'' +
", address='" + address + '\'' +
'}';
}
}
```

3. UserMapper.java

```
package com.example.mybatisdemo.mapper;

import com.example.mybatisdemo.entity.User;
import org.apache.ibatis.annotations.Param;
import org.springframework.stereotype.Repository;

@Repository
public interface UserMapper {

int createTable(@Param("tableName") String tableName);

User Sel(@Param("user")User user);

int Add(@Param("user")User user);

int Update(@Param("user")User user);

int Delete(@Param("user")User user);

}
```

4. UserService.java

```
package com.example.mybatisdemo.service;

import com.example.mybatisdemo.entity.User;
import com.example.mybatisdemo.mapper.UserMapper;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class UserService {
@Autowired
UserMapper userMapper;

public String createTable(String table){
int a = userMapper.createTable(table);
if (a == 1) {
return "Create table failed";
} else {
return "Create table successfully";
}
}

public User Sel(User user) {
return userMapper.Sel(user);
}

public String Add(User user) {
int a = userMapper.Add(user);
if (a == 1) {
return "Add user successfully";
} else {
return "Add user failed";
}
}

public String Update(User user) {
int a = userMapper.Update(user);
if (a == 1) {
return "Update user successfully";
} else {
return "Update user failed";
}
}
```

```
public String Delete(User user) {
int a = userMapper.Delete(user);
if (a == 1) {
return "Delete user successfully";
} else {
return "Delete user failed";
}
}

};
```

5. MyBatisDemoApplication.java

```
package com.example.mybatisdemo;

import org.mybatis.spring.annotation.MapperScan;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@MapperScan("com.example.mybatisdemo.mapper")
@SpringBootApplication
public class MyBatisDemoApplication {
public static void main(String[] args) {
SpringApplication.run(MyBatisDemoApplication.class, args);
}
}
```

6. UserMapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.example.mybatisdemo.mapper.UserMapper">

<resultMap id="BaseResultMap" type="com.example.mybatisdemo.entity.User">
<result column="id" jdbcType="INTEGER" property="id"/>
<result column="userName" jdbcType="VARCHAR" property="username"/>
<result column="passWord" jdbcType="VARCHAR" property="password"/>
<result column="realName" jdbcType="VARCHAR" property="address"/>
</resultMap>

<update id="createTable" parameterType="string">
CREATE TABLE ${tableName} (
`id` int(11) NOT NULL AUTO_INCREMENT,
`username` varchar(255) DEFAULT NULL,
`password` varchar(255) DEFAULT NULL,
`address` varchar(255) DEFAULT NULL,
PRIMARY KEY (`id`)
);
</update>

<select id="Sel" resultType="com.example.mybatisdemo.entity.User">
select * from user where 1=1
<if test="user.id != null">
AND id = #{user.id}
</if>
</select>

<insert id="Add" parameterType="com.example.mybatisdemo.entity.User">
INSERT INTO user
<trim prefix="(" suffix=")" suffixOverrides=",">
<if test="user.username != null">
username,
</if>
<if test="user.password != null">
password,
</if>
<if test="user.address != null">
address,
</if>
```

```
</trim>
<trim prefix="VALUES (" suffix=")" suffixOverrides=",">
<if test="user.username != null">
#{user.username,jdbcType=VARCHAR},
</if>
<if test="user.password != null">
#{user.password,jdbcType=VARCHAR},
</if>
<if test="user.address != null">
#{user.address,jdbcType=VARCHAR},
</if>
</trim>
</insert>

<update id="Update" parameterType="com.example.mybatisdemo.entity.User">
UPDATE user
<set>
<if test="user.username != null">
username = #{user.username},
</if>
<if test="user.password != null">
password = #{user.password},
</if>
<if test="user.address != null">
address = #{user.address},
</if>
</set>
WHERE
id=#{user.id}
</update>

<delete id="Delete" parameterType="com.example.mybatisdemo.entity.User">
DELETE FROM user WHERE id = #{user.id}
</delete>

</mapper>
```

测试

构建并启动这个项目。

image-20221026161226923

当出现下面的消息时，表示应用程序已经正常启动，你可以打开浏览器并发送 HTTP 请求。

```
2022-10-26 16:13:24.030 INFO 60253 --- [ main]
c.e.mybatisdemo.MyBatisDemoApplication : Starting MyBatisDemoApplication
on nandeng-macbookpro.local with PID 60253
(/Users/nandeng/IdeaProjects/MyBatisDemo/target/classes started by
nandeng in /Users/nandeng/IdeaProjects/MyBatisDemo)
2022-10-26 16:13:24.035 INFO 60253 --- [ main]
c.e.mybatisdemo.MyBatisDemoApplication : No active profile set, falling
back to default profiles: default
2022-10-26 16:13:25.415 INFO 60253 --- [ main]
o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with
port(s): 8080 (http)
2022-10-26 16:13:25.421 INFO 60253 --- [ main]
o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-10-26 16:13:25.421 INFO 60253 --- [ main]
org.apache.catalina.core.StandardEngine : Starting Servlet engine:
[Apache Tomcat/9.0.41]
2022-10-26 16:13:25.476 INFO 60253 --- [ main] o.a.c.c.C.[Tomcat].
[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-10-26 16:13:25.477 INFO 60253 --- [ main]
w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext:
initialization completed in 1390 ms
2022-10-26 16:13:26.020 INFO 60253 --- [ main]
o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService
'applicationTaskExecutor'
2022-10-26 16:13:26.248 INFO 60253 --- [ main]
o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080
(http) with context path ''
2022-10-26 16:13:26.272 INFO 60253 --- [ main]
c.e.mybatisdemo.MyBatisDemoApplication : Started MyBatisDemoApplication
in 2.669 seconds (JVM running for 3.544)
```

1. 测试新建表

打开你的的浏览器并输入网址：`http://localhost:8080/test/create`

image-20221026161929338

在 MySQL 客户端中，验证表是否已成功创建。

```
mysql> use test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+----------------+
| tables_in_test |
+----------------+
| user |
+----------------+
1 row in set (0.00 sec)

mysql> show create table user;
+-------+-----------------------------------------------------------------
-----------------------------------------------------------------------
-----------------------------------------------------+
| Table | Create Table |
+-------+-----------------------------------------------------------------
-----------------------------------------------------------------------
-----------------------------------------------------+
| user | CREATE TABLE `user` (
`id` INT NOT NULL AUTO_INCREMENT,
`username` VARCHAR(255) DEFAULT null,
`password` VARCHAR(255) DEFAULT null,
`address` VARCHAR(255) DEFAULT null,
PRIMARY KEY (`id`)
) |
+-------+-----------------------------------------------------------------
-----------------------------------------------------------------------
-----------------------------------------------------+
1 row in set (0.01 sec)
```

2. 测试增加用户

打开你的浏览器并输入网址：

`http://localhost:8080/test/add?username=tom&password=123456&address=shanghai`

image-20221026162317800

在 MySQL 客户端中，使用如下命令验证是否成功添加记录：

```
mysql> select * from user;
+------+----------+----------+----------+
| id | username | password | address |
+------+----------+----------+----------+
| 1 | tom | 123456 | shanghai |
+------+----------+----------+----------+
1 row in set (0.00 sec)
```

3. 测试查询用户

打开你的浏览器并输入网址：`http://localhost:8080/test/selectUserByid?id=1`

image-20221026162455058

你可以直接从浏览器中获取查询结果。

4. 测试更新用户

打开你的浏览器并输入网址：
`http://localhost:8080/test/update/username=tom&password=654321&address=beijing`

image-20221026162613066

在 MySQL 客户端中，使用如下命令验证是否成功更新记录：

```
mysql> select * from user;
+------+----------+----------+---------+
| id | username | password | address |
+------+----------+----------+---------+
| 1 | tom | 654321 | beijing |
+------+----------+----------+---------+
1 row in set (0.00 sec)
```

5. 测试删除用户

打开你的浏览器并输入网址：`http://localhost:8080/test/delete?id=1`

image-20221026162756460

在 MySQL 客户端中，使用如下命令验证是否成功删除记录：

```
mysql> select * from user;
Empty set (0.00 sec)
```

使用 Python 和 SQLAlchemy 构建一个 CRUD 示例

本篇文档将指导你如何使用 **Python** 和 **SQLAlchemy** 构建一个简单的应用程序，并实现 CRUD（创建、读取、更新、删除）功能。

SQLAlchemy 是 Python 语言中最流行的 ORM 工具之一。

开始前准备

相关软件的简单介绍：

- SQLAlchemy：SQLAlchemy 是一个 Python 库，可以促进 Python 程序和数据库之间的通信。大多数时候，这个库用作对象关系映射器 (ORM) 工具，将 Python 类转换为关系数据库上的表，并自动将函数调用转换为 SQL 语句。
- Faker：Faker 是一个生成假数据的 Python 库。虚假数据通常用于测试或用一些虚拟数据填充数据库。

环境配置

在你开始之前，确认你已经下载并安装了如下软件：

- 确认你已完成单机部署 MatrixOne。通过 MySQL 客户端连接 MatrixOne 并创建一个命名为 *test* 的数据库：

  ```
  mysql> create database test;
  ```

- 确认你已完成安装 Python 3.8(or plus) version。

  使用下面的代码检查 Python 版本确认安装成功：

  ```
  #To check with Python installation and its version
  python3 -V
  ```

- 确认你已完成安装 MySQL。
- 下载安装 `sqlalchemy`、`pymysql`、`cryptography` 和 `faker` 工具。

使用下面的代码下载安装 `sqlalchemy`、`pymysql`、`cryptography` 和 `faker` 工具：

```
pip3 install sqlalchemy
pip3 install pymysql
pip3 install cryptography
pip3 install faker

#If you are in China mainland and have a low downloading speed, you can speed up the download by following commands.
pip3 install sqlalchemy -i https://pypi.tuna.tsinghua.edu.cn/simple
pip3 install pymysql -i https://pypi.tuna.tsinghua.edu.cn/simple
pip3 install cryptography -i https://pypi.tuna.tsinghua.edu.cn/simple
pip3 install faker -i https://pypi.tuna.tsinghua.edu.cn/simple
```

你可以参考 [Python 连接 MatrixOne 服务](#)了解如何通过 `SQLAlchemy` 连接到 MatrixOne，本篇文档将指导你如何实现 CRUD（创建、读取、更新、删除)。

新建表

作为对象关系映射器（ORM）工具，SQLAlchemy 允许开发人员创建 Python 类来映射关系数据库中的表。

在下面的代码示例中，将创建一个 `Customer` 类，它定义的 `Customer` 的代码相当于一条 SQL 语句，它表示 MatrixOne 中的命名为 `Customer` 的表：

```
CREATE TABLE `User` (
`id` INT NOT NULL AUTO_INCREMENT,
`cname` VARCHAR(64) DEFAULT NULL,
`caddress` VARCHAR(512) DEFAULT NULL,
PRIMARY KEY (`id`)
)
```

新建一个 `sqlalchemy_create.py` 的文本文件，将以下代码拷贝粘贴到文件内：

```
from faker import Factory
from sqlalchemy import create_engine, Column, Integer, String
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

faker = Factory.create()

engine = create_engine('mysql+pymysql://dump:111@127.0.0.1:6001/test')

Session = sessionmaker(bind=engine)
session = Session()

Base = declarative_base()


class Customer(Base):
__tablename__ = "Customer"
id = Column(Integer, primary_key=True,autoincrement=True)
cname = Column(String(64))
caddress = Column(String(512))

def __init__(self,name,address):
self.cname = name
self.caddress = address

def __str__(self):
return "cname:"+self.cname +" caddress:"+self.caddress

def __repr__(self):
return "cname:"+self.cname +" caddress:"+self.caddress


# Generate 10 Customer records
Customers = [Customer(name= faker.name(),address = faker.address()) for i
in range(10)]

# Create the table
Base.metadata.create_all(engine)

# Insert all customer records to Customer table
session.add_all(Customers)

session.commit()
```

打开终端，使用以下代码运行此 *python* 文件：

```
> python3 sqlalchemy_create.py
```

你可以使用 MySQL 客户端验证表是否创建成功：

```
mysql> show tables;
+----------------+
| tables_in_test |
+----------------+
| Customer |
+----------------+
1 row in set (0.04 sec)
mysql> select * from `Customer`;
+------+------------------+---------------------------------------------
-------+
| id | cname | caddress |
+------+------------------+---------------------------------------------
-------+
| 1 | Wendy Luna | 002 Brian Plaza
Andrewhaven, SC 88456 |
| 2 | Meagan Rodriguez | USCGC Olson
FPO AP 21249 |
| 3 | Angela Ramos | 029 Todd Curve Apt. 352
Mooreville, FM 15950 |
| 4 | Lisa Bruce | 68103 Mackenzie Mountain
North Andrew, UT 29853 |
| 5 | Julie Moore | Unit 1117 Box 1029
DPO AP 87468 |
| 6 | David Massey | 207 Wayne Groves Apt. 733
Vanessashire, NE 34549 |
| 7 | David Mccann | 97274 Sanders Tunnel Apt. 480
Anthonyberg, DC 06558 |
| 8 | Morgan Price | 57463 Lisa Drive
Thompsonshire, NM 88077 |
| 9 | Samuel Griffin | 186 Patel Crossing
North Stefaniechester, WV 08221 |
| 10 | Tristan Pierce | 593 Blankenship Rapids
New Jameshaven, SD 89585 |
+------+------------------+---------------------------------------------
-------+
10 rows in set (0.03 sec)
```

读取数据

在下面的示例中，将通过两种方式从 `Customer` 表中读取数据。

第一种方式是全表扫描：

```
select * from `Customer`
```

第二种方式是点查询：

```
select * from `Customer` where `cname` = 'David Mccann';
```

新建一个 `sqlalchemy_read.py` 的文本文件，将以下代码拷贝粘贴到文件内：

```
from sqlalchemy import create_engine, Column, Integer, String
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

engine = create_engine('mysql+pymysql://dump:111@127.0.0.1:6001/test')

Session = sessionmaker(bind=engine)
session = Session()

Base = declarative_base()

class Customer(Base):
__tablename__ = "Customer"
id = Column(Integer, primary_key=True,autoincrement=True)
cname = Column(String(64))
caddress = Column(String(512))

def __init__(self,name,address):
self.cname = name
self.caddress = address

def __str__(self):
return "cname:"+self.cname +" caddress:"+self.caddress

def __repr__(self):
return "cname:"+self.cname +" caddress:"+self.caddress


# query all data
customers = session.query(Customer).all()

for customer in customers:
print(customer.__str__() +"\n-------------------------\n")

# query with a filter condition
Mccann = session.query(Customer).filter_by(cname='David Mccann').first()
print(Mccann)
print("\n------------------------\n")
```

打开终端，使用以下代码运行此 *python* 文件并查看结果：

```
> python3 sqlalchemy_read.py
cname:Wendy Luna caddress:002 Brian Plaza
Andrewhaven, SC 88456
--------------------------

cname:Meagan Rodriguez caddress:USCGC Olson
FPO AP 21249
--------------------------

cname:Angela Ramos caddress:029 Todd Curve Apt. 352
Mooreville, FM 15950
--------------------------

cname:Lisa Bruce caddress:68103 Mackenzie Mountain
North Andrew, UT 29853
--------------------------

cname:Julie Moore caddress:Unit 1117 Box 1029
DPO AP 87468
--------------------------

cname:David Massey caddress:207 Wayne Groves Apt. 733
Vanessashire, NE 34549
--------------------------

cname:David Mccann caddress:97274 Sanders Tunnel Apt. 480
Anthonyberg, DC 06558
--------------------------

cname:Morgan Price caddress:57463 Lisa Drive
Thompsonshire, NM 88077
--------------------------

cname:Samuel Griffin caddress:186 Patel Crossing
North Stefaniechester, WV 08221
--------------------------

cname:Tristan Pierce caddress:593 Blankenship Rapids
New Jameshaven, SD 89585
--------------------------

cname:David Mccann caddress:97274 Sanders Tunnel Apt. 480
Anthonyberg, DC 06558
```

```
-----------------------
```

更新数据

在下面的示例中，将指导你更新 *Customer* 表的第一个 *cname* 列为另一个值。

新建一个 `sqlalchemy_update.py` 的文本文件，将以下代码拷贝粘贴到文件内：

```
from sqlalchemy import create_engine, Column, Integer, String
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

engine = create_engine('mysql+pymysql://dump:111@127.0.0.1:6001/test')

Session = sessionmaker(bind=engine)
session = Session()

Base = declarative_base()

class Customer(Base):
__tablename__ = "Customer"
id = Column(Integer, primary_key=True,autoincrement=True)
cname = Column(String(64))
caddress = Column(String(512))

def __init__(self,name,address):
self.cname = name
self.caddress = address

def __str__(self):
return "cname:"+self.cname +" caddress:"+self.caddress

def __repr__(self):
return "cname:"+self.cname +" caddress:"+self.caddress


customer = session.query(Customer).first()
print(customer)
print("\n---------------------\n")

# Rename customer
customer.cname = "Coby White"


session.commit()

# See the updated result
customer = session.query(Customer).first()
print(customer)
```

打开终端，使用以下代码运行此 *python* 文件并查看结果：

```
> python3 sqlalchemy_update.py
cname:Wendy Luna caddress:002 Brian Plaza
Andrewhaven, SC 88456

---------------------

cname:Coby White caddress:002 Brian Plaza
Andrewhaven, SC 88456
```

你可以使用 MySQL 客户端验证表是否更新成功：

```
mysql> select * from `Customer`;
+------+------------------+---------------------------------------------
-------+
| id | cname | caddress |
+------+------------------+---------------------------------------------
-------+
| 1 | Coby White | 002 Brian Plaza
Andrewhaven, SC 88456 |
| 2 | Meagan Rodriguez | USCGC Olson
FPO AP 21249 |
| 3 | Angela Ramos | 029 Todd Curve Apt. 352
Mooreville, FM 15950 |
| 4 | Lisa Bruce | 68103 Mackenzie Mountain
North Andrew, UT 29853 |
| 5 | Julie Moore | Unit 1117 Box 1029
DPO AP 87468 |
| 6 | David Massey | 207 Wayne Groves Apt. 733
Vanessashire, NE 34549 |
| 7 | David Mccann | 97274 Sanders Tunnel Apt. 480
Anthonyberg, DC 06558 |
| 8 | Morgan Price | 57463 Lisa Drive
Thompsonshire, NM 88077 |
| 9 | Samuel Griffin | 186 Patel Crossing
North Stefaniechester, WV 08221 |
| 10 | Tristan Pierce | 593 Blankenship Rapids
New Jameshaven, SD 89585 |
+------+------------------+---------------------------------------------
-------+
10 rows in set (0.02 sec)
```

删除数据

在下面的示例中，将指导你删除 *Customer* 表的第一条数据。

新建一个 `sqlalchemy_detele.py` 的文本文件，将以下代码拷贝粘贴到文件内：

```
from sqlalchemy import create_engine, Column, Integer, String
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

engine = create_engine('mysql+pymysql://dump:111@127.0.0.1:6001/test')

Session = sessionmaker(bind=engine)
session = Session()

Base = declarative_base()

class Customer(Base):
__tablename__ = "Customer"
id = Column(Integer, primary_key=True,autoincrement=True)
cname = Column(String(64))
caddress = Column(String(512))

def __init__(self,name,address):
self.cname = name
self.caddress = address

def __str__(self):
return "cname:"+self.cname +" caddress:"+self.caddress

def __repr__(self):
return "cname:"+self.cname +" caddress:"+self.caddress


# delete the first record
customer = session.query(Customer).first()

session.delete(customer)
session.commit()

# query all data
customers = session.query(Customer).all()

for customer in customers:
print(customer.__str__() +"\n-------------------------\n")
```

打开终端，使用以下代码运行此 *python* 文件并查看结果：

```
> python3 sqlalchemy_delete.py
cname:Meagan Rodriguez caddress:USCGC Olson
FPO AP 21249
--------------------------

cname:Angela Ramos caddress:029 Todd Curve Apt. 352
Mooreville, FM 15950
--------------------------

cname:Lisa Bruce caddress:68103 Mackenzie Mountain
North Andrew, UT 29853
--------------------------

cname:Julie Moore caddress:Unit 1117 Box 1029
DPO AP 87468
--------------------------

cname:David Massey caddress:207 Wayne Groves Apt. 733
Vanessashire, NE 34549
--------------------------

cname:David Mccann caddress:97274 Sanders Tunnel Apt. 480
Anthonyberg, DC 06558
--------------------------

cname:Morgan Price caddress:57463 Lisa Drive
Thompsonshire, NM 88077
--------------------------

cname:Samuel Griffin caddress:186 Patel Crossing
North Stefaniechester, WV 08221
--------------------------

cname:Tristan Pierce caddress:593 Blankenship Rapids
New Jameshaven, SD 89585
--------------------------
```

你可以使用 MySQL 客户端验证表种的记录是否删除成功：

```
mysql> select * from `Customer`;
+------+------------------+---------------------------------------------
-------+
| id | cname | caddress |
+------+------------------+---------------------------------------------
-------+
| 2 | Meagan Rodriguez | USCGC Olson
FPO AP 21249 |
| 3 | Angela Ramos | 029 Todd Curve Apt. 352
Mooreville, FM 15950 |
| 4 | Lisa Bruce | 68103 Mackenzie Mountain
North Andrew, UT 29853 |
| 5 | Julie Moore | Unit 1117 Box 1029
DPO AP 87468 |
| 6 | David Massey | 207 Wayne Groves Apt. 733
Vanessashire, NE 34549 |
| 7 | David Mccann | 97274 Sanders Tunnel Apt. 480
Anthonyberg, DC 06558 |
| 8 | Morgan Price | 57463 Lisa Drive
Thompsonshire, NM 88077 |
| 9 | Samuel Griffin | 186 Patel Crossing
North Stefaniechester, WV 08221 |
| 10 | Tristan Pierce | 593 Blankenship Rapids
New Jameshaven, SD 89585 |
+------+------------------+---------------------------------------------
-------+
9 rows in set (0.04 sec)
```

构建一个 Python CRUD 示例

本篇文档将指导你如何使用 **Python** 构建一个简单的应用程序，并实现 CRUD（创建、读取、更新、删除）功能。

开始前准备

环境配置

在你开始之前，确认你已经下载并安装了如下软件：

- 确认你已完成单机部署 MatrixOne。
- 确认你已完成安装 Python 3.8(or plus)。

  使用下面的代码检查 Python 版本确认安装成功：

  ```
  #To check with Python installation and its version
  python3 -V
  ```

- 确认你已完成安装 MySQL。
- 下载安装 `pymysql` 和 `cryptography` 工具。

  使用下面的代码下载安装 `pymysql` 和 `cryptography` 工具：

  ```
  pip3 install pymysql
  pip3 install cryptography

  #If you are in China mainland and have a low downloading speed, you
  can speed up the download by following commands.
  pip3 install pymysql -i https://pypi.tuna.tsinghua.edu.cn/simple
  pip3 install cryptography -i https://pypi.tuna.tsinghua.edu.cn/simple
  ```

你可以参考 Python 连接 MatrixOne 服务了解如何通过 `pymysql` 连接到 MatrixOne，本篇文档将指导你如何实现 CRUD（创建、读取、更新、删除)。

新建表

新建一个 `create.py` 的文本文件，将以下代码拷贝粘贴到文件内：

```
#!/usr/bin/python3

import pymysql.cursors

SQL_CONNECTION = pymysql.connect(
host='127.0.0.1',
port=6001,
user='dump',
password = "111",
db='test',
cursorclass=pymysql.cursors.DictCursor,
autocommit=True
)

SQL = "CREATE TABLE cars (id INT NOT NULL AUTO_INCREMENT, car_model
VARCHAR(45) NULL,car_brand VARCHAR(45) NULL,PRIMARY KEY (`id`))"

with SQL_CONNECTION.cursor() as cursor:
try:
sql_exec = cursor.execute(SQL)
print("Table created")
except (pymysql.Error, pymysql.Warning) as e:
print(f'error! {e}')

finally:
SQL_CONNECTION.close()
```

打开终端，使用以下代码运行此 *python* 文件。这将在 MatrixOne 中的数 据库 *test* 内创建一个名为 *cars* 表。

```
> python3 create.py
Table created
```

你可以使用 MySQL 客户端验证表是否创建成功：

```
mysql> show tables;
+----------------+
| tables_in_test |
+----------------+
| cars |
+----------------+
1 row in set (0.03 sec)
mysql> show create table cars;
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------+
| Table | Create Table |
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------+
| cars | CREATE TABLE `cars` (
`id` INT NOT NULL AUTO_INCREMENT,
`car_model` VARCHAR(45) DEFAULT NULL,
`car_brand` VARCHAR(45) DEFAULT NULL,
PRIMARY KEY (`id`)
) |
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------+
1 row in set (0.03 sec)
```

插入数据

新建一个 `insert.py` 的文本文件，将以下代码拷贝粘贴到文件内：

```
#!/usr/bin/python3

import pymysql.cursors

SQL_CONNECTION = pymysql.connect(
host='127.0.0.1',
port=6001,
user='dump',
password = "111",
db='test',
cursorclass=pymysql.cursors.DictCursor,
autocommit=True
)

SQL = "INSERT INTO cars(car_model, car_brand) VALUES ('accord', 'honda')"

with SQL_CONNECTION.cursor() as cursor:
try:
sql_exec = cursor.execute(SQL)
if sql_exec:
print(sql_exec)
print("Record Added")
else:
print(sql_exec)
print("Not Added")
except (pymysql.Error, pymysql.Warning) as e:
print(f'error! {e}')

finally:
SQL_CONNECTION.close()
```

执行下面的代码会在 *cars* 表中插入一条记录：

```
> python3 insert.py
1
Record Added
```

你可以在 MySQL 客户端中验证这条记录是否插入成功：

```
mysql> select * from cars;
+------+-----------+-----------+
| id | car_model | car_brand |
+------+-----------+-----------+
| 1 | accord | honda |
+------+-----------+-----------+
1 row in set (0.03 sec)
```

查询数据

新建一个 `read.py` 的文本文件，将以下代码拷贝粘贴到文件内：

```
#!/usr/bin/python3

import pymysql.cursors

SQL_CONNECTION = pymysql.connect(
host='127.0.0.1',
port=6001,
user='dump',
password = "111",
db='test',
cursorclass=pymysql.cursors.DictCursor,
autocommit=True
)

SQL = "SELECT * FROM cars"

with SQL_CONNECTION.cursor() as cursor:
try:
sql_exec = cursor.execute(SQL)
if sql_exec:
print(sql_exec)
print(cursor.fetchall())
else:
print(sql_exec)
print("No Record")
except (pymysql.Error, pymysql.Warning) as e:
print(f'error! {e}')

finally:
SQL_CONNECTION.close()
```

执行下面的代码查询并返回 *cars* 表中的所有记录：

```
> python3 read.py
1
[{'id': 1, 'car_model': 'accord', 'car_brand': 'honda'}]
```

更新数据

新建一个 `update.py` 的文本文件，将以下代码拷贝粘贴到文件内：

```
#!/usr/bin/python3

import pymysql.cursors

SQL_CONNECTION = pymysql.connect(
host='127.0.0.1',
port=6001,
user='dump',
password = "111",
db='test',
cursorclass=pymysql.cursors.DictCursor,
autocommit=True
)

SQL = "UPDATE cars SET car_model = 'explorer', car_brand = 'ford' WHERE
id = '1'"

with SQL_CONNECTION.cursor() as cursor:
try:
sql_exec = cursor.execute(SQL)
if sql_exec:
print(sql_exec)
print("Record Updated")
else:
print(sql_exec)
print("Not Updated")
except (pymysql.Error, pymysql.Warning) as e:
print(f'error! {e}')

finally:
SQL_CONNECTION.close()
```

执行下面代码更新 id 为 “1” 的记录：

```
> python3 update.py
1
Record Updated
```

你可以在 MySQL 客户端中验证这条记录是否更新成功：

```
mysql> select * from cars;
+------+-----------+-----------+
| id | car_model | car_brand |
+------+-----------+-----------+
| 1 | explorer | ford |
+------+-----------+-----------+
1 row in set (0.02 sec)
```

删除数据

新建一个 `delete.py` 的文本文件，将以下代码拷贝粘贴到文件内：

```
#!/usr/bin/python3

import pymysql.cursors

SQL_CONNECTION = pymysql.connect(
host='127.0.0.1',
port=6001,
user='dump',
password = "111",
db='test',
cursorclass=pymysql.cursors.DictCursor,
autocommit=True
)

SQL = "DELETE FROM cars WHERE id = '1'"

with SQL_CONNECTION.cursor() as cursor:
try:
sql_exec = cursor.execute(SQL)
if sql_exec:
print(sql_exec)
print("Record Deleted")
else:
print(sql_exec)
print("Not Deleted")
except (pymysql.Error, pymysql.Warning) as e:
print(f'error! {e}')

finally:
SQL_CONNECTION.close()
```

执行下面代码删除 id 为 “1” 的记录：

```
> python3 delete.py
1
Record Deleted
```

你可以在 MySQL 客户端中验证这条记录是否删除成功：

```
mysql> select * from cars;
Empty set (0.03 sec)
```

备份与恢复

对于企业而言，每天都会产生大量数据，那么对于数据库的备份就非常重要。在系统崩溃或者硬件故障，又或者用户误操作的情况下，你可以恢复数据并重启系统，不会造成数据丢失。

另外，数据备份也作为升级 MatrixOne 安装之前的保障，同时数据备份也可以可用于将 MatrixOne 安装转移到另一个系统。

MatrixOne 目前仅支持通过 `modump` 实用程序进行逻辑备份。`modump` 是一个命令行实用程序，用于生成 MatrixOne 数据库的逻辑备份。它生成可用于重新创建数据库对象和数据的 SQL 语句。你可以在 modump 章节中查找它的语法说明和使用指南。

我们将通过一个简单的示例来讲述如何使用 `modump` 实用程序完成数据备份和还原的过程。

步骤

1. 构建 modump 二进制文件

参见构建 modump 二进制文件章节，完成 `modump` 二进制文件构建。

如果你已经完成了 `modump` 的构建，那么你可以继续阅读下一章节 **2. 生成单个数据库的备份**。

2. 生成单个数据库的备份

示例如下，使用以下 SQL 创建的数据库 *t* 及其表 *t1*：

```
DROP DATABASE IF EXISTS `t`;
CREATE DATABASE `t`;
USE `t`;
create table t1
(
c1 int primary key auto_increment,
c2 tinyint not null default 4,
c3 smallint,
c4 bigint,
c5 tinyint unsigned,
c6 smallint unsigned,
c7 int unsigned,
c8 bigint unsigned,
c9 float,
c10 double,
c11 date,
c12 datetime,
c13 timestamp on update current_timestamp,
c14 char,
c15 varchar,
c16 json,
c17 decimal,
c18 text,
c19 blob,
c20 uuid
);
insert into t1 values (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, '2019-01-01', '2019-
01-01 00:00:00', '2019-01-01 00:00:00', 'a', 'a', '\{"a":1}','1212.1212',
'a', 'aza', '00000000-0000-0000-0000-000000000000');
```

如果要生成单个数据库的备份，可以运行以下命令。该命令将生成命名为 *t* 的数据库的备份，其中包含 *t.sql* 文件中的结构和数据。

```
./modump -u dump -p 111 -h 127.0.0.1 -P 6001 -db t > t.sql
```

如果要在数据库中生成单个表的备份，可以运行以下命令。该命令将生成命名为 *t* 的数据库的 *t1* 表的备份，其中包含 *t.sql* 文件中的结构和数据。

```
./modump -u dump -p 111 -db t -tbl t1 > t1.sql
```

> **注意**
>
> 如果你有多个数据库，你需要多次运行 `modump` 来一条一条生成 SQL。

3. 恢复备份到 MatrixOne 服务器

将导出的*. sql* 文件恢复至 MatrixOne 数据库相对简单。要恢复你的数据库，你必须先创建一个空数据库，并使用 *MySQL 客户端*进行恢复。

将 MatrixOne 与 MySQL 客户端连接至同一服务器上，并确保导出的*. sql* 文件也在同一服务器上。

```
mysql> create database t if not exists;
mysql> source /YOUR_SQL_FILE_PATH/t.sql
```

成功执行以上命令后，执行以下命令，检查是否在命名为 *t* 数据库上创建了所有对象。

```
mysql> use t;
mysql> show tables;
mysql> select count(*) from t1;
```

挂载目录到 Docker 容器

本篇文档将指导你在使用 Docker 启动 MatrixOne 的情况下，如何挂载*数据目录*或*自定义配置文件*到 Docker 容器。

挂载数据目录

为了保证数据目录安全不丢失，参考以下详细步骤，挂载本地数据目录到 Docker 容器：

1. 检查 Docker 内是否已经启动 MatrixOne：

```
docker ps -a
```

2. 如果 Docker 内有正在运行的 MatrixOne，需要先停止：

```
docker stop <containerID>
docker rm <containerID>
```

   如果没有正在运行的 MatrxiOne，请忽略这一步。

3. 把本地**空目录**挂载到 Docker 容器目录 */mo-data* 下，命令示例如下：

```
sudo docker run --name <name> --privileged -d -p 6001:6001 -v
${local_data_path}/mo-data:/mo-data:rw matrixorigin/matrixone:0.7.0
```

参数	描述
${local_data_path}/mo-data:/mo-data	挂载本地数据目录 *${local_data_path}/mo-data* 到容器 */mo-data* 文件夹 **Tips**: 需要挂载的本地数据目录必须为**空目录**。

挂载自定义配置文件

如果你需要修改启动配置文件，建议你先将 Docker 内的启动配置文件拷贝到你本地目录，然后将存放配置文件的本地目录挂载到 Docker 容器目录下，参考以下详细步骤，挂载配置文件到 Docker 容器：

1. 检查 Docker 内是否已经启动 MatrixOne：

```
docker ps -a
```

2. 如果 Docker 内还没有正在运行的 MatrixOne，请先启动：

```
docker run -d -p 6001:6001 --name matrixone
matrixorigin/matrixone:0.7.0
```

3. 查看 Docker 已经启动 MatrixOne 的 containerID，并将配置文件目录拷贝到本地目录内：

```
docker ps -a
docker cp <containerID>:/etc .
```

4. 拷贝完成后，关停当前的 MatrixOne：

```
docker stop <containerID>
docker rm <containerID>
```

5. （选做）修改本地配置文件并保存。

6. 挂载配置文件到 Docker 容器目录，同时启动 MatrixOne，挂载命令示例如下：

```
sudo docker run --name <name> --privileged -d -p 6001:6001 -v
${local_config_path}/etc:/etc:rw --entrypoint "/mo-service"
matrixorigin/matrixone:0.7.0 -launch /etc/quickstart/launch.toml
```

参数	描述
${local_config_path}/etc:/etc	挂载本地配置文件目录 *${local_config_path}/etc* 到容器 */etc* 文件夹
—entrypoint “/mo-service”	指定容器启动 MatrixOne 服务
-launch /etc/quickstart/launch.toml	启动 */etc/* 中的 MatrixOne 启动配置文件

更多关于 *Docker run* 的指令释义，运行命令 `docker run --help` 进行查看。

将数据从 MySQL 迁移至 MatrixOne

本篇文档将指导你如何将数据从 MySQL 迁移至 MatrixOne。

1. MySQL 数据转储

你需要拥有对 MySQL 实例的完全访问权限。

首先，使用 `mysqldump` 将 MySQL 表结构和数据通过以下命令转储到一个文件中。如果你不熟悉如何使用 `mysqldump`，可参见 [mysqldump 教程]。

```
mysqldump -h IP_ADDRESS -uUSERNAME -pPASSWORD -d DB_NAME1 DB_NAME2 ...
OUTPUT_FILE_NAME.SQL
```

示例如下，使用命令将一个命名为 *test* 数据库的所有表结构和数据转储到一个名为 *a.sql* 的文件中。

```
mysqldump -h 127.0.0.1 -uroot -proot -d test > a.sql
```

2. 修改*.sql* 文件

从 MySQL 转储的 SQL 文件还不完全兼容 MatrixOne。你需要删除和修改几个元素，以使 SQL 文件适应 MatrixOne 的格式。

- 需要删除不支持的语法或功能：`CHARACTER SET/CHARSET`、`COLLATE`、`ROW_FORMAT`、`USING BTREE`、`LOCK TABLE` `SET SYSTEM_VARIABLE`、`ENGINE`。
- 需要修改不支持的数据类型：如果你使用 BINARY 类型，你可以将其修改为 BLOB 类型。

下面示例是一个典型的 `mysqldump` 表：

```
DROP TABLE IF EXISTS `tool`;
CREATE TABLE IF NOT EXISTS `tool` (
`id` bigint NOT NULL AUTO_INCREMENT,
`tool_id` bigint DEFAULT NULL COMMENT 'id',
`operation_type` varchar(50) CHARACTER SET utf8mb4 COLLATE
utf8mb4_0900_ai_ci DEFAULT NULL COMMENT 'type',
`remark` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci
DEFAULT NULL COMMENT 'remark',
`create_user` varchar(20) CHARACTER SET utf8mb4 COLLATE
utf8mb4_0900_ai_ci DEFAULT NULL COMMENT 'create user',
`create_time` datetime DEFAULT CURRENT_TIMESTAMP COMMENT 'create time',
PRIMARY KEY (`id`) USING BTREE,
KEY `tool_id_IDX` (`tool_id`) USING BTREE,
KEY `operation_type_IDX` (`operation_type`) USING BTREE
) ENGINE=InnoDB AUTO_INCREMENT=1913 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci ROW_FORMAT=DYNAMIC COMMENT='tool table';
```

在 MatrixOne 中成功创建这个表，它需要被修改为：

```
DROP TABLE IF EXISTS `tool`;
CREATE TABLE IF NOT EXISTS `tool` (
`id` bigint NOT NULL AUTO_INCREMENT,
`tool_id` bigint DEFAULT NULL COMMENT 'id',
`operation_type` varchar(50) DEFAULT NULL COMMENT 'type',
`remark` varchar(100) DEFAULT NULL COMMENT 'remark',
`create_user` varchar(20) DEFAULT NULL COMMENT 'create user',
`create_time` datetime DEFAULT CURRENT_TIMESTAMP COMMENT 'create time',
PRIMARY KEY (`id`),
KEY `tool_id_IDX` (`tool_id`),
KEY `operation_type_IDX` (`operation_type`)
) COMMENT='tool table';
```

3. 导入至 MatrixOne

转储的*. sql* 文件修改完成之后，就可以将整个表结构和数据导入到 MatrixOne 中。

1. 打开 MySQL 终端并连接到 MatrixOne。
2. 通过 `source` 命令将*. sql* 文件导入 MatrixOne。

```
mysql> source '/YOUR_PATH/a.sql'
```

如果*. sql* 文件较大，可以使用如下命令在后台运行导入任务：

```
nohup mysql -h 127.0.0.1 -P 6001 -udump -p111 -e 'source
/YOUR_PATH/a.sql' &
```

完成 SSB 测试

SSB 星型模式基准测试是 OLAP 数据库性能测试的常用场景，通过本篇教程，您可以了解到如何在 MatrixOne 中实现 SSB 测试。

准备工作

确保你已经完成了[单机部署 MatrixOne](#)。

1. 编译 dbgen

```
git clone https://github.com/vadimtk/ssb-dbgen.git
cd ssb-dbgen
make
```

> **注意**
>
> 如果你的硬件配置为 M1 芯片，编译 dbgen 仍需进行其他配置，参见[部署常见问题](#)。

2. 生成数据

选项一：生成单表数据集

当使用 `-s 1` 时 `dbgen` 命令会生产近 600 万行数据 (670MB)，当使用 `-s 10` 时会生产近 6000 万行数据，会耗费大量时间。

```
./dbgen -s 1 -T c
./dbgen -s 1 -T l
./dbgen -s 1 -T p
./dbgen -s 1 -T s
./dbgen -s 1 -T d
```

选项二：下载大宽表数据集

我们准备了 1GB 的大宽表数据集供你下载。

1. 在下面链接中直接获取大宽表数据集文件：

```
https://community-shared-data-1308875761.cos.ap-beijing.myqcloud.com/lineorder_flat.tar.bz2
```

2. 下载完成后将数据文件解压。

3. 在 MatrixOne 中建表

```
create database if not exists ssb;
use ssb;
drop table if exists lineorder;
drop table if exists part;
drop table if exists supplier;
drop table if exists customer;
drop table if exists date;
drop table if exists lineorder_flat;

create table lineorder (
lo_orderkey bigint,
lo_linenumber int,
lo_custkey int,
lo_partkey int,
lo_suppkey int,
lo_orderdate date,
lo_orderpriority char (15),
lo_shippriority tinyint,
lo_quantity double,
lo_extendedprice double,
lo_ordtotalprice double,
lo_discount double,
lo_revenue double,
lo_supplycost double,
lo_tax double,
lo_commitdate date,
lo_shipmode char (10)
) ;

create table part (
p_partkey int,
p_name varchar (22),
p_mfgr char (6),
p_category char (7),
p_brand char (9),
p_color varchar (11),
p_type varchar (25),
p_size int,
p_container char (10)
) ;

create table supplier (
```

```
s_suppkey int,
s_name char (25),
s_address varchar (25),
s_city char (10),
s_nation char (15),
s_region char (12),
s_phone char (15)
) ;

create table customer (
c_custkey int,
c_name varchar (25),
c_address varchar (25),
c_city char (10),
c_nation char (15),
c_region char (12),
c_phone char (15),
c_mktsegment char (10)
) ;

create table date (
d_datekey date,
d_date char (18),
d_dayofweek char (9),
d_month char (9),
d_year int,
d_yearmonthnum int,
d_yearmonth char (7),
d_daynuminweek varchar(12),
d_daynuminmonth int,
d_daynuminyear int,
d_monthnuminyear int,
d_weeknuminyear int,
d_sellingseason varchar (12),
d_lastdayinweekfl varchar (1),
d_lastdayinmonthfl varchar (1),
d_holidayfl varchar (1),
d_weekdayfl varchar (1)
) ;

CREATE TABLE lineorder_flat(
LO_ORDERKEY bigint key,
LO_LINENUMBER int,
LO_CUSTKEY int,
```

```
LO_PARTKEY int,
LO_SUPPKEY int,
LO_ORDERDATE date,
LO_ORDERPRIORITY char(15),
LO_SHIPPRIORITY tinyint,
LO_QUANTITY double,
LO_EXTENDEDPRICE double,
LO_ORDTOTALPRICE double,
LO_DISCOUNT double,
LO_REVENUE int unsigned,
LO_SUPPLYCOST int unsigned,
LO_TAX double,
LO_COMMITDATE date,
LO_SHIPMODE char(10),
C_NAME varchar(25),
C_ADDRESS varchar(25),
C_CITY char(10),
C_NATION char(15),
C_REGION char(12),
C_PHONE char(15),
C_MKTSEGMENT char(10),
S_NAME char(25),
S_ADDRESS varchar(25),
S_CITY char(10),
S_NATION char(15),
S_REGION char(12),
S_PHONE char(15),
P_NAME varchar(22),
P_MFGR char(6),
P_CATEGORY char(7),
P_BRAND char(9),
P_COLOR varchar(11),
P_TYPE varchar(25),
P_SIZE int,
P_CONTAINER char(10)
);
```

4. 导入数据

选项一：使用如下命令导入单表数据集

```
load data infile '/ssb-dbgen-path/supplier.tbl' into table supplier
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY
'\n';

load data infile '/ssb-dbgen-path/customer.tbl' into table customer
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY
'\n';

load data infile '/ssb-dbgen-path/date.tbl' into table date FIELDS
TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\n';

load data infile '/ssb-dbgen-path/part.tbl' into table part FIELDS
TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\n';

load data infile '/ssb-dbgen-path/lineorder.tbl' into table lineorder
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY
'\n';
```

接着你可以在 MatrixOne 中进行查询操作。

选项二：使用如下命令导入大宽表数据集

运行以下命令将数据导入 `lineorder_flat`：

```
load data infile '/ssb-dbgen-path/lineorder_flat.tbl' into table
lineorder_flat FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES
TERMINATED BY '\n';
```

5. 运行 SSB 测试命令

> **注意**
>
> `GROUP BY` 暂不支持使用别名。

单表查询

```
-- Q1.1
SELECT sum(LO_EXTENDEDPRICE * LO_DISCOUNT) AS revenue FROM lineorder_flat
WHERE year(LO_ORDERDATE)=1993 AND LO_DISCOUNT BETWEEN 1 AND 3 AND
LO_QUANTITY \< 25;

-- Q1.2
SELECT sum(LO_EXTENDEDPRICE * LO_DISCOUNT) AS revenue FROM lineorder_flat
WHERE year(LO_ORDERDATE)=1994 AND LO_DISCOUNT BETWEEN 4 AND 6 AND
LO_QUANTITY BETWEEN 26 AND 35;

-- Q1.3
SELECT sum(LO_EXTENDEDPRICE * LO_DISCOUNT) AS revenue FROM lineorder_flat
WHERE year(LO_ORDERDATE)=1994 AND LO_DISCOUNT BETWEEN 5 AND 7 AND
LO_QUANTITY BETWEEN 26 AND 35;

-- Q2.1
SELECT sum(LO_REVENUE),year(LO_ORDERDATE) AS year,P_BRAND FROM
lineorder_flat WHERE P_CATEGORY = 'MFGR#12' AND S_REGION = 'AMERICA'
GROUP BY year(LO_ORDERDATE), P_BRAND ORDER BY year,P_BRAND;

-- Q2.2
SELECT sum(LO_REVENUE), year(LO_ORDERDATE) AS year, P_BRAND FROM
lineorder_flat WHERE P_BRAND BETWEEN 'MFGR#2221' AND 'MFGR#2228' AND
S_REGION = 'ASIA' GROUP BY year(LO_ORDERDATE), P_BRAND ORDER BY year,
P_BRAND;

-- Q2.3
SELECT sum(LO_REVENUE), year(LO_ORDERDATE) AS year, P_BRAND FROM
lineorder_flat WHERE P_BRAND = 'MFGR#2239' AND S_REGION = 'EUROPE' GROUP
BY year(LO_ORDERDATE), P_BRAND ORDER BY year, P_BRAND;

-- Q3.1
SELECT C_NATION, S_NATION, year(LO_ORDERDATE) AS year, sum(LO_REVENUE) AS
revenue FROM lineorder_flat WHERE C_REGION = 'ASIA' AND S_REGION = 'ASIA'
AND year(LO_ORDERDATE) between 1992 AND 1997 GROUP BY C_NATION, S_NATION,
year(LO_ORDERDATE) ORDER BY year asc, revenue desc;

-- Q3.2
SELECT C_CITY, S_CITY, year(LO_ORDERDATE) AS year, sum(LO_REVENUE) AS
revenue FROM lineorder_flat WHERE C_NATION = 'CHINA' AND S_NATION =
'CHINA' AND year(LO_ORDERDATE) between 1992 AND 1997 GROUP BY C_CITY,
S_CITY, year(LO_ORDERDATE) ORDER BY year asc, revenue desc;
```

```sql
-- Q3.3
SELECT C_CITY, S_CITY, year(LO_ORDERDATE) AS year, sum(LO_REVENUE) AS
revenue FROM lineorder_flat WHERE (C_CITY = 'UNITED KI0' OR C_CITY =
'UNITED KI7') AND (S_CITY = 'UNITED KI0' OR S_CITY = 'UNITED KI7') AND
year(LO_ORDERDATE) between 1992 AND 1997 GROUP BY C_CITY, S_CITY,
year(LO_ORDERDATE) ORDER BY year asc, revenue desc;

-- Q3.4
SELECT C_CITY, S_CITY, year(LO_ORDERDATE) AS year, sum(LO_REVENUE) AS
revenue FROM lineorder_flat WHERE (C_CITY = 'UNITED KI0' OR C_CITY =
'UNITED KI7') AND (S_CITY = 'MOZAMBIQU1' OR S_CITY = 'KENYA 4') AND
year(LO_ORDERDATE)= 1997 GROUP BY C_CITY, S_CITY, year(LO_ORDERDATE)
ORDER BY year asc, revenue desc;

-- Q4.1
SELECT year(LO_ORDERDATE) AS year, C_NATION, sum(LO_REVENUE -
LO_SUPPLYCOST) AS profit FROM lineorder_flat WHERE C_REGION = 'AMERICA'
AND S_REGION = 'AMERICA' AND (P_MFGR = 'MFGR#1' OR P_MFGR = 'MFGR#2')
GROUP BY year(LO_ORDERDATE), C_NATION ORDER BY year, C_NATION;

-- Q4.2
SELECT year(LO_ORDERDATE) AS year, S_NATION, P_CATEGORY, sum(LO_REVENUE -
LO_SUPPLYCOST) AS profit FROM lineorder_flat WHERE C_REGION = 'AMERICA'
AND S_REGION = 'AMERICA' AND (year(LO_ORDERDATE) = 1997 OR
year(LO_ORDERDATE) = 1998) AND (P_MFGR = 'MFGR#1' OR P_MFGR = 'MFGR#2')
GROUP BY year(LO_ORDERDATE), S_NATION, P_CATEGORY ORDER BY year,
S_NATION, P_CATEGORY;

-- Q4.3
SELECT year(LO_ORDERDATE) AS year, S_CITY, P_BRAND, sum(LO_REVENUE -
LO_SUPPLYCOST) AS profit FROM lineorder_flat WHERE S_NATION = 'UNITED
STATES' AND (year(LO_ORDERDATE) = 1997 OR year(LO_ORDERDATE) = 1998) AND
P_CATEGORY = 'MFGR#14' GROUP BY year(LO_ORDERDATE), S_CITY, P_BRAND ORDER
BY year, S_CITY, P_BRAND;
```

多表查询

```
-- Q1.1
select sum(lo_revenue) as revenue
from lineorder join date on lo_orderdate = d_datekey
where year(d_datekey) = 1993 and lo_discount between 1 and 3 and
lo_quantity \< 25;

-- Q1.2
select sum(lo_revenue) as revenue
from lineorder
join date on lo_orderdate = d_datekey
where d_yearmonthnum = 199401
and lo_discount between 4 and 6
and lo_quantity between 26 and 35;

-- Q1.3
select sum(lo_revenue) as revenue
from lineorder
join date on lo_orderdate = d_datekey
where d_weeknuminyear = 6 and year(d_datekey) = 1994
and lo_discount between 5 and 7
and lo_quantity between 26 and 35;

-- Q2.1
select sum(lo_revenue) as lo_revenue, year(d_datekey) as year, p_brand
from lineorder
join date on lo_orderdate = d_datekey
join part on lo_partkey = p_partkey
join supplier on lo_suppkey = s_suppkey
where p_category = 'MFGR#12' and s_region = 'AMERICA'
group by year(d_datekey), p_brand
order by year, p_brand;

-- Q2.2
select sum(lo_revenue) as lo_revenue, year(d_datekey) as year, p_brand
from lineorder
join date on lo_orderdate = d_datekey
join part on lo_partkey = p_partkey
join supplier on lo_suppkey = s_suppkey
where p_brand between 'MFGR#2221' and 'MFGR#2228' and s_region = 'ASIA'
group by year(d_datekey), p_brand
order by year, p_brand;
```

```sql
-- Q2.3
select sum(lo_revenue) as lo_revenue, year(d_datekey) as year, p_brand
from lineorder
join date on lo_orderdate = d_datekey
join part on lo_partkey = p_partkey
join supplier on lo_suppkey = s_suppkey
where p_brand = 'MFGR#2239' and s_region = 'EUROPE'
group by year(d_datekey), p_brand
order by year, p_brand;

-- Q3.1
select c_nation, s_nation, year(d_datekey) as year, sum(lo_revenue) as
lo_revenue
from lineorder
join date on lo_orderdate = d_datekey
join customer on lo_custkey = c_custkey
join supplier on lo_suppkey = s_suppkey
where c_region = 'ASIA' and s_region = 'ASIA' and year(d_datekey) between
1992 and 1997
group by c_nation, s_nation, year(d_datekey)
order by year asc, lo_revenue desc;

-- Q3.2
select c_city, s_city, year(d_datekey) as year, sum(lo_revenue) as
lo_revenue
from lineorder
join date on lo_orderdate = d_datekey
join customer on lo_custkey = c_custkey
join supplier on lo_suppkey = s_suppkey
where c_nation = 'UNITED STATES' and s_nation = 'UNITED STATES'
and year(d_datekey) between 1992 and 1997
group by c_city, s_city, year(d_datekey)
order by year asc, lo_revenue desc;

-- Q3.3
select c_city, s_city, year(d_datekey) as year, sum(lo_revenue) as
lo_revenue
from lineorder
join date on lo_orderdate = d_datekey
join customer on lo_custkey = c_custkey
join supplier on lo_suppkey = s_suppkey
where (c_city='UNITED KI1' or c_city='UNITED KI5')
and (s_city='UNITED KI1' or s_city='UNITED KI5')
and year(d_datekey) between 1992 and 1997
```

```
group by c_city, s_city, year(d_datekey)
order by year asc, lo_revenue desc;

-- Q3.4
select c_city, s_city, year(d_datekey) as year, sum(lo_revenue) as
lo_revenue
from lineorder
join date on lo_orderdate = d_datekey
join customer on lo_custkey = c_custkey
join supplier on lo_suppkey = s_suppkey
where (c_city='UNITED KI1' or c_city='UNITED KI5') and (s_city='UNITED
KI1' or s_city='UNITED KI5') and d_yearmonth = '199712'
group by c_city, s_city, year(d_datekey)
order by year(d_datekey) asc, lo_revenue desc;

-- Q4.1
select year(d_datekey) as year, c_nation, sum(lo_revenue) -
sum(lo_supplycost) as profit
from lineorder
join date on lo_orderdate = d_datekey
join customer on lo_custkey = c_custkey
join supplier on lo_suppkey = s_suppkey
join part on lo_partkey = p_partkey
where c_region = 'AMERICA' and s_region = 'AMERICA' and (p_mfgr =
'MFGR#1' or p_mfgr = 'MFGR#2')
group by year(d_datekey), c_nation
order by year, c_nation;

-- Q4.2
select year(d_datekey) as year, s_nation, p_category, sum(lo_revenue) -
sum(lo_supplycost) as profit
from lineorder
join date on lo_orderdate = d_datekey
join customer on lo_custkey = c_custkey
join supplier on lo_suppkey = s_suppkey
join part on lo_partkey = p_partkey
where c_region = 'AMERICA'and s_region = 'AMERICA'
and (year(d_datekey) = 1997 or year(d_datekey) = 1998)
and (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')
group by year(d_datekey), s_nation, p_category
order by year, s_nation, p_category;

-- Q4.3
select year(d_datekey) as year, s_city, p_brand, sum(lo_revenue) -
```

```
sum(lo_supplycost) as profit
from lineorder
join dates on lo_orderdate = d_datekey
join customer on lo_custkey = c_custkey
join supplier on lo_suppkey = s_suppkey
join part on lo_partkey = p_partkey
where
(year(d_datekey) = 1997 or year(d_datekey) = 1998)
and c_region = 'AMERICA'
and s_nation = 'UNITED STATES'
and p_category = 'MFGR#14'
group by year, s_city, p_brand
order by year, s_city, p_brand;
```

6. 运行预期结果

单表查询运行预期结果

```
-- Q1.1
+--------------+
| revenue |
+--------------+
| 702223464659 |
+--------------+

-- Q1.2
+---------------+
| revenue |
+---------------+
| 1842875090496 |
+---------------+

-- Q1.3
+---------------+
| revenue |
+---------------+
| 2208738861324 |
+---------------+

-- Q2.1
+-----------------+------+-----------+
| sum(lo_revenue) | year | p_brand |
+-----------------+------+-----------+
| 283684289 | 1992 | MFGR#121 |
| 1344702529 | 1992 | MFGR#1210 |
| 757158682 | 1992 | MFGR#1211 |
| 1156022815 | 1992 | MFGR#1212 |
| 676164367 | 1992 | MFGR#1213 |
| 522779256 | 1992 | MFGR#1214 |
| 233507213 | 1992 | MFGR#1215 |
| 726755819 | 1992 | MFGR#1216 |
| 1865504710 | 1992 | MFGR#1217 |
| 685600451 | 1992 | MFGR#1218 |
| 814671470 | 1992 | MFGR#1219 |
| 655405800 | 1992 | MFGR#122 |
| 962120553 | 1992 | MFGR#1220 |
| 961393626 | 1992 | MFGR#1221 |
| 922110830 | 1992 | MFGR#1222 |
```

```
| 814842712 | 1992 | MFGR#1223 |
| 1402773165 | 1992 | MFGR#1224 |
| 977517439 | 1992 | MFGR#1225 |
| 1392114944 | 1992 | MFGR#1226 |
| 658922951 | 1992 | MFGR#1227 |
| 892495927 | 1992 | MFGR#1228 |
| 806609100 | 1992 | MFGR#1229 |
| 578875657 | 1992 | MFGR#123 |
| 691236902 | 1992 | MFGR#1230 |
| 482715249 | 1992 | MFGR#1231 |
| 618556590 | 1992 | MFGR#1232 |
| 981657009 | 1992 | MFGR#1233 |
| 1050794669 | 1992 | MFGR#1234 |
| 1335217991 | 1992 | MFGR#1235 |
| 686009527 | 1992 | MFGR#1236 |
| 605242584 | 1992 | MFGR#1237 |
| 430455641 | 1992 | MFGR#1238 |
| 359654993 | 1992 | MFGR#1239 |
| 818818457 | 1992 | MFGR#124 |
| 1388502581 | 1992 | MFGR#1240 |
| 711566198 | 1992 | MFGR#125 |
| 893045647 | 1992 | MFGR#126 |
| 1240534333 | 1992 | MFGR#127 |
| 871966127 | 1992 | MFGR#128 |
| 719176622 | 1992 | MFGR#129 |
| 245880758 | 1993 | MFGR#121 |
| 1480662753 | 1993 | MFGR#1210 |
| 982292725 | 1993 | MFGR#1211 |
| 1001136766 | 1993 | MFGR#1212 |
| 227144072 | 1993 | MFGR#1213 |
| 291611370 | 1993 | MFGR#1214 |
| 454576668 | 1993 | MFGR#1215 |
| 772027256 | 1993 | MFGR#1216 |
| 1155751851 | 1993 | MFGR#1217 |
| 897883050 | 1993 | MFGR#1218 |
| 1209508962 | 1993 | MFGR#1219 |
| 530577973 | 1993 | MFGR#122 |
| 901650471 | 1993 | MFGR#1220 |
| 739540494 | 1993 | MFGR#1221 |
| 698751459 | 1993 | MFGR#1222 |
| 1327979157 | 1993 | MFGR#1223 |
| 1476697469 | 1993 | MFGR#1224 |
| 792103134 | 1993 | MFGR#1225 |
| 1420277376 | 1993 | MFGR#1226 |
```

```
| 1446032835 | 1993 | MFGR#1227 |
| 884375309 | 1993 | MFGR#1228 |
| 522705868 | 1993 | MFGR#1229 |
| 601685233 | 1993 | MFGR#123 |
| 806727248 | 1993 | MFGR#1230 |
| 399477390 | 1993 | MFGR#1231 |
| 894047578 | 1993 | MFGR#1232 |
| 496067089 | 1993 | MFGR#1233 |
| 1050223756 | 1993 | MFGR#1234 |
| 891681399 | 1993 | MFGR#1235 |
| 1402903631 | 1993 | MFGR#1236 |
| 347380448 | 1993 | MFGR#1237 |
| 514853194 | 1993 | MFGR#1238 |
| 410543863 | 1993 | MFGR#1239 |
| 673483594 | 1993 | MFGR#124 |
| 713343630 | 1993 | MFGR#1240 |
| 512610707 | 1993 | MFGR#125 |
| 1228110634 | 1993 | MFGR#126 |
| 936958961 | 1993 | MFGR#127 |
| 579067515 | 1993 | MFGR#128 |
| 636174833 | 1993 | MFGR#129 |
| 370347074 | 1994 | MFGR#121 |
| 483900410 | 1994 | MFGR#1210 |
| 1240306281 | 1994 | MFGR#1211 |
| 1003345253 | 1994 | MFGR#1212 |
| 557310864 | 1994 | MFGR#1213 |
| 314972328 | 1994 | MFGR#1214 |
| 1130260810 | 1994 | MFGR#1215 |
| 401618319 | 1994 | MFGR#1216 |
| 652173601 | 1994 | MFGR#1217 |
| 923612074 | 1994 | MFGR#1218 |
| 469711377 | 1994 | MFGR#1219 |
| 580152115 | 1994 | MFGR#122 |
| 433134653 | 1994 | MFGR#1220 |
| 730569849 | 1994 | MFGR#1221 |
| 607609104 | 1994 | MFGR#1222 |
| 949877669 | 1994 | MFGR#1223 |
| 1644687916 | 1994 | MFGR#1224 |
| 492974194 | 1994 | MFGR#1225 |
| 921499688 | 1994 | MFGR#1226 |
| 719059761 | 1994 | MFGR#1227 |
| 1000497056 | 1994 | MFGR#1228 |
| 486968927 | 1994 | MFGR#1229 |
| 734124906 | 1994 | MFGR#123 |
```

```
| 645016873 | 1994 | MFGR#1230 |
| 526638240 | 1994 | MFGR#1231 |
| 1358112405 | 1994 | MFGR#1232 |
| 1167074905 | 1994 | MFGR#1233 |
| 1102915239 | 1994 | MFGR#1234 |
| 693058125 | 1994 | MFGR#1235 |
| 1673392892 | 1994 | MFGR#1236 |
| 849630029 | 1994 | MFGR#1237 |
| 721392705 | 1994 | MFGR#1238 |
| 1237195774 | 1994 | MFGR#1239 |
| 1107832795 | 1994 | MFGR#124 |
| 827906290 | 1994 | MFGR#1240 |
| 682827304 | 1994 | MFGR#125 |
| 1198768141 | 1994 | MFGR#126 |
| 1274148181 | 1994 | MFGR#127 |
| 738849138 | 1994 | MFGR#128 |
| 751136619 | 1994 | MFGR#129 |
| 318978803 | 1995 | MFGR#121 |
| 383199448 | 1995 | MFGR#1210 |
| 1300165814 | 1995 | MFGR#1211 |
| 1550400731 | 1995 | MFGR#1212 |
| 451958158 | 1995 | MFGR#1213 |
| 431434279 | 1995 | MFGR#1214 |
| 713735582 | 1995 | MFGR#1215 |
| 919323722 | 1995 | MFGR#1216 |
| 1542358864 | 1995 | MFGR#1217 |
| 500930092 | 1995 | MFGR#1218 |
| 1208162086 | 1995 | MFGR#1219 |
| 785707989 | 1995 | MFGR#122 |
| 993828211 | 1995 | MFGR#1220 |
| 667253893 | 1995 | MFGR#1221 |
| 1654114297 | 1995 | MFGR#1222 |
| 986528377 | 1995 | MFGR#1223 |
| 755014642 | 1995 | MFGR#1224 |
| 1090300100 | 1995 | MFGR#1225 |
| 1063626454 | 1995 | MFGR#1226 |
| 1382528859 | 1995 | MFGR#1227 |
| 919953351 | 1995 | MFGR#1228 |
| 457795295 | 1995 | MFGR#1229 |
| 953851827 | 1995 | MFGR#123 |
| 807209283 | 1995 | MFGR#1230 |
| 236304454 | 1995 | MFGR#1231 |
| 668449537 | 1995 | MFGR#1232 |
| 240657083 | 1995 | MFGR#1233 |
```

```
| 920389826 | 1995 | MFGR#1234 |
| 684096065 | 1995 | MFGR#1235 |
| 1005844219 | 1995 | MFGR#1236 |
| 626170996 | 1995 | MFGR#1237 |
| 984581826 | 1995 | MFGR#1238 |
| 602850634 | 1995 | MFGR#1239 |
| 1172025628 | 1995 | MFGR#124 |
| 489788581 | 1995 | MFGR#1240 |
| 643100327 | 1995 | MFGR#125 |
| 894596661 | 1995 | MFGR#126 |
| 706917239 | 1995 | MFGR#127 |
| 428671983 | 1995 | MFGR#128 |
| 971611472 | 1995 | MFGR#129 |
| 306497573 | 1996 | MFGR#121 |
| 890719726 | 1996 | MFGR#1210 |
| 1761977172 | 1996 | MFGR#1211 |
| 633845765 | 1996 | MFGR#1212 |
| 475801202 | 1996 | MFGR#1213 |
| 271930385 | 1996 | MFGR#1214 |
| 366399844 | 1996 | MFGR#1215 |
| 877472476 | 1996 | MFGR#1216 |
| 970366290 | 1996 | MFGR#1217 |
| 537175690 | 1996 | MFGR#1218 |
| 956970528 | 1996 | MFGR#1219 |
| 711962074 | 1996 | MFGR#122 |
| 1062161683 | 1996 | MFGR#1220 |
| 406293744 | 1996 | MFGR#1221 |
| 785404335 | 1996 | MFGR#1222 |
| 579267044 | 1996 | MFGR#1223 |
| 1220640256 | 1996 | MFGR#1224 |
| 490130196 | 1996 | MFGR#1225 |
| 1603805636 | 1996 | MFGR#1226 |
| 1530646510 | 1996 | MFGR#1227 |
| 1093328922 | 1996 | MFGR#1228 |
| 596520140 | 1996 | MFGR#1229 |
| 450815571 | 1996 | MFGR#123 |
| 315053350 | 1996 | MFGR#1230 |
| 198951017 | 1996 | MFGR#1231 |
| 579778438 | 1996 | MFGR#1232 |
| 480905486 | 1996 | MFGR#1233 |
| 1433336215 | 1996 | MFGR#1234 |
| 560925251 | 1996 | MFGR#1235 |
| 1038766181 | 1996 | MFGR#1236 |
| 783697960 | 1996 | MFGR#1237 |
```

```
| 972656445 | 1996 | MFGR#1238 |
| 614528801 | 1996 | MFGR#1239 |
| 1418931894 | 1996 | MFGR#124 |
| 995139591 | 1996 | MFGR#1240 |
| 824028471 | 1996 | MFGR#125 |
| 669475113 | 1996 | MFGR#126 |
| 831704651 | 1996 | MFGR#127 |
| 920514555 | 1996 | MFGR#128 |
| 436162421 | 1996 | MFGR#129 |
| 553684594 | 1997 | MFGR#121 |
| 1317368046 | 1997 | MFGR#1210 |
| 1617056983 | 1997 | MFGR#1211 |
| 1196031005 | 1997 | MFGR#1212 |
| 1056458336 | 1997 | MFGR#1213 |
| 352179650 | 1997 | MFGR#1214 |
| 511058114 | 1997 | MFGR#1215 |
| 658259312 | 1997 | MFGR#1216 |
| 1238450697 | 1997 | MFGR#1217 |
| 376245955 | 1997 | MFGR#1218 |
| 913437812 | 1997 | MFGR#1219 |
| 1114996000 | 1997 | MFGR#122 |
| 814059433 | 1997 | MFGR#1220 |
| 817328516 | 1997 | MFGR#1221 |
| 541428597 | 1997 | MFGR#1222 |
| 1260539052 | 1997 | MFGR#1223 |
| 1766426582 | 1997 | MFGR#1224 |
| 1221271245 | 1997 | MFGR#1225 |
| 1499152922 | 1997 | MFGR#1226 |
| 491586909 | 1997 | MFGR#1227 |
| 911517084 | 1997 | MFGR#1228 |
| 728186585 | 1997 | MFGR#1229 |
| 904363416 | 1997 | MFGR#123 |
| 605369014 | 1997 | MFGR#1230 |
| 290370455 | 1997 | MFGR#1231 |
| 602414397 | 1997 | MFGR#1232 |
| 765339065 | 1997 | MFGR#1233 |
| 1170973957 | 1997 | MFGR#1234 |
| 860319765 | 1997 | MFGR#1235 |
| 1031080311 | 1997 | MFGR#1236 |
| 736404810 | 1997 | MFGR#1237 |
| 1012330790 | 1997 | MFGR#1238 |
| 681055343 | 1997 | MFGR#1239 |
| 601626600 | 1997 | MFGR#124 |
| 920404157 | 1997 | MFGR#1240 |
```

```
| 1007678757 | 1997 | MFGR#125 |
| 622347203 | 1997 | MFGR#126 |
| 1215295592 | 1997 | MFGR#127 |
| 822274972 | 1997 | MFGR#128 |
| 643903475 | 1997 | MFGR#129 |
| 470008435 | 1998 | MFGR#121 |
| 568508492 | 1998 | MFGR#1210 |
| 323759101 | 1998 | MFGR#1211 |
| 572013331 | 1998 | MFGR#1212 |
| 448137748 | 1998 | MFGR#1213 |
| 137422458 | 1998 | MFGR#1214 |
| 346491756 | 1998 | MFGR#1215 |
| 454542243 | 1998 | MFGR#1216 |
| 759205210 | 1998 | MFGR#1217 |
| 674544462 | 1998 | MFGR#1218 |
| 735952270 | 1998 | MFGR#1219 |
| 490186568 | 1998 | MFGR#122 |
| 769456686 | 1998 | MFGR#1220 |
| 654540341 | 1998 | MFGR#1221 |
| 800329859 | 1998 | MFGR#1222 |
| 263849231 | 1998 | MFGR#1223 |
| 445461642 | 1998 | MFGR#1224 |
| 387808862 | 1998 | MFGR#1225 |
| 675424382 | 1998 | MFGR#1226 |
| 265906673 | 1998 | MFGR#1227 |
| 585938371 | 1998 | MFGR#1228 |
| 683188537 | 1998 | MFGR#1229 |
| 304403717 | 1998 | MFGR#123 |
| 533781674 | 1998 | MFGR#1230 |
| 304060011 | 1998 | MFGR#1231 |
| 635275907 | 1998 | MFGR#1232 |
| 658295080 | 1998 | MFGR#1233 |
| 524133341 | 1998 | MFGR#1234 |
| 363911877 | 1998 | MFGR#1235 |
| 300885635 | 1998 | MFGR#1236 |
| 532608453 | 1998 | MFGR#1237 |
| 484291410 | 1998 | MFGR#1238 |
| 445336624 | 1998 | MFGR#1239 |
| 719027801 | 1998 | MFGR#124 |
| 518860961 | 1998 | MFGR#1240 |
| 491235383 | 1998 | MFGR#125 |
| 520917638 | 1998 | MFGR#126 |
| 1158787745 | 1998 | MFGR#127 |
| 401190922 | 1998 | MFGR#128 |
```

```
| 406656337 | 1998 | MFGR#129 |
+----------------+------+-----------+

-- Q2.2
+----------------+------+-----------+
| sum(lo_revenue) | year | p_brand |
+-----------------+------+-----------+
| 1259802358 | 1992 | MFGR#2221 |
| 1728549344 | 1992 | MFGR#2222 |
| 1375260024 | 1992 | MFGR#2223 |
| 1299982475 | 1992 | MFGR#2224 |
| 1541960331 | 1992 | MFGR#2225 |
| 1151853513 | 1992 | MFGR#2226 |
| 1271175264 | 1992 | MFGR#2227 |
| 1726441695 | 1992 | MFGR#2228 |
| 1251460032 | 1993 | MFGR#2221 |
| 1331062515 | 1993 | MFGR#2222 |
| 902809293 | 1993 | MFGR#2223 |
| 980512417 | 1993 | MFGR#2224 |
| 1253088003 | 1993 | MFGR#2225 |
| 959195148 | 1993 | MFGR#2226 |
| 555593932 | 1993 | MFGR#2227 |
| 2186479174 | 1993 | MFGR#2228 |
| 1094092222 | 1994 | MFGR#2221 |
| 1491699323 | 1994 | MFGR#2222 |
| 1501160826 | 1994 | MFGR#2223 |
| 1387107418 | 1994 | MFGR#2224 |
| 1641588884 | 1994 | MFGR#2225 |
| 1387296390 | 1994 | MFGR#2226 |
| 1038341470 | 1994 | MFGR#2227 |
| 1565763138 | 1994 | MFGR#2228 |
| 1412945650 | 1995 | MFGR#2221 |
| 1546178356 | 1995 | MFGR#2222 |
| 1218352073 | 1995 | MFGR#2223 |
| 1052197762 | 1995 | MFGR#2224 |
| 1822921900 | 1995 | MFGR#2225 |
| 728142181 | 1995 | MFGR#2226 |
| 966131607 | 1995 | MFGR#2227 |
| 1379320517 | 1995 | MFGR#2228 |
| 1042767284 | 1996 | MFGR#2221 |
| 994733835 | 1996 | MFGR#2222 |
| 1615788545 | 1996 | MFGR#2223 |
| 1113980216 | 1996 | MFGR#2224 |
| 1622570253 | 1996 | MFGR#2225 |
```

```
| 1540226758 | 1996 | MFGR#2226 |
| 1115687883 | 1996 | MFGR#2227 |
| 1716355343 | 1996 | MFGR#2228 |
| 867705636 | 1997 | MFGR#2221 |
| 1529877498 | 1997 | MFGR#2222 |
| 1594444450 | 1997 | MFGR#2223 |
| 587421043 | 1997 | MFGR#2224 |
| 1112274470 | 1997 | MFGR#2225 |
| 1327884722 | 1997 | MFGR#2226 |
| 884180880 | 1997 | MFGR#2227 |
| 1664207656 | 1997 | MFGR#2228 |
| 827743515 | 1998 | MFGR#2221 |
| 662242310 | 1998 | MFGR#2222 |
| 861445539 | 1998 | MFGR#2223 |
| 694538672 | 1998 | MFGR#2224 |
| 675179021 | 1998 | MFGR#2225 |
| 480728720 | 1998 | MFGR#2226 |
| 643763072 | 1998 | MFGR#2227 |
| 994499201 | 1998 | MFGR#2228 |
+-----------------+------+-----------+

-- Q2.3
+-----------------+------+-----------+
| sum(lo_revenue) | year | p_brand |
+-----------------+------+-----------+
| 1428843284 | 1992 | MFGR#2239 |
| 1865666054 | 1993 | MFGR#2239 |
| 2242753254 | 1994 | MFGR#2239 |
| 1446677305 | 1995 | MFGR#2239 |
| 921681503 | 1996 | MFGR#2239 |
| 1549990572 | 1997 | MFGR#2239 |
| 926327433 | 1998 | MFGR#2239 |
+-----------------+------+-----------+

-- Q3.1
+-----------+-----------+------+-------------+
| c_nation | s_nation | year | revenue |
+-----------+-----------+------+-------------+
| VIETNAM | CHINA | 1992 | 17194479086 |
| JAPAN | CHINA | 1992 | 15572594510 |
| JAPAN | JAPAN | 1992 | 13861682954 |
| INDONESIA | CHINA | 1992 | 13499663933 |
| VIETNAM | INDONESIA | 1992 | 13163103649 |
| JAPAN | INDONESIA | 1992 | 13035158590 |
```

```
| INDIA | CHINA | 1992 | 12987688902 |
| INDONESIA | JAPAN | 1992 | 12939737918 |
| VIETNAM | JAPAN | 1992 | 12174715858 |
| JAPAN | VIETNAM | 1992 | 11669093753 |
| INDIA | INDONESIA | 1992 | 11452602145 |
| INDONESIA | INDONESIA | 1992 | 10394407561 |
| INDIA | JAPAN | 1992 | 10313084900 |
| JAPAN | INDIA | 1992 | 10035511089 |
| CHINA | CHINA | 1992 | 9828744666 |
| VIETNAM | VIETNAM | 1992 | 9701522505 |
| INDONESIA | INDIA | 1992 | 9271105764 |
| INDIA | INDIA | 1992 | 8879645522 |
| CHINA | INDONESIA | 1992 | 8373693838 |
| CHINA | JAPAN | 1992 | 8051248951 |
| VIETNAM | INDIA | 1992 | 7804539029 |
| INDONESIA | VIETNAM | 1992 | 7615465790 |
| CHINA | INDIA | 1992 | 7344868842 |
| INDIA | VIETNAM | 1992 | 6830508508 |
| CHINA | VIETNAM | 1992 | 6529888238 |
| JAPAN | CHINA | 1993 | 18839180326 |
| VIETNAM | CHINA | 1993 | 14761890330 |
| JAPAN | INDONESIA | 1993 | 13648082171 |
| INDONESIA | CHINA | 1993 | 13518181805 |
| INDIA | CHINA | 1993 | 13249555999 |
| JAPAN | JAPAN | 1993 | 12667833152 |
| JAPAN | VIETNAM | 1993 | 11529854580 |
| CHINA | CHINA | 1993 | 11216468573 |
| INDONESIA | INDONESIA | 1993 | 10953284722 |
| VIETNAM | INDONESIA | 1993 | 10582912267 |
| INDIA | JAPAN | 1993 | 10482950584 |
| VIETNAM | JAPAN | 1993 | 10370811002 |
| INDIA | INDONESIA | 1993 | 10145286112 |
| INDONESIA | JAPAN | 1993 | 9850020303 |
| VIETNAM | VIETNAM | 1993 | 9591468153 |
| CHINA | INDONESIA | 1993 | 9015864524 |
| CHINA | JAPAN | 1993 | 8972996729 |
| INDONESIA | INDIA | 1993 | 8903638786 |
| JAPAN | INDIA | 1993 | 8848048514 |
| INDONESIA | VIETNAM | 1993 | 8024464882 |
| VIETNAM | INDIA | 1993 | 7806575746 |
| INDIA | VIETNAM | 1993 | 7537331106 |
| INDIA | INDIA | 1993 | 7211053846 |
| CHINA | VIETNAM | 1993 | 6700022269 |
| CHINA | INDIA | 1993 | 6327331541 |
```

```
| JAPAN | CHINA | 1994 | 15661051644 |
| VIETNAM | CHINA | 1994 | 13958591931 |
| JAPAN | JAPAN | 1994 | 13566252348 |
| CHINA | CHINA | 1994 | 12870010072 |
| VIETNAM | JAPAN | 1994 | 12728320716 |
| INDONESIA | CHINA | 1994 | 12295790872 |
| INDIA | CHINA | 1994 | 12166419121 |
| JAPAN | INDONESIA | 1994 | 11358955025 |
| INDIA | INDONESIA | 1994 | 11111248365 |
| JAPAN | INDIA | 1994 | 10078806371 |
| VIETNAM | INDONESIA | 1994 | 9923852578 |
| INDIA | JAPAN | 1994 | 9839136767 |
| CHINA | JAPAN | 1994 | 9836586308 |
| INDONESIA | JAPAN | 1994 | 9786694572 |
| INDIA | VIETNAM | 1994 | 9551081406 |
| JAPAN | VIETNAM | 1994 | 9035431932 |
| VIETNAM | INDIA | 1994 | 9032319402 |
| INDONESIA | INDONESIA | 1994 | 8876012426 |
| CHINA | INDONESIA | 1994 | 8375581981 |
| VIETNAM | VIETNAM | 1994 | 8095638136 |
| INDONESIA | INDIA | 1994 | 7943993512 |
| INDONESIA | VIETNAM | 1994 | 7927236697 |
| INDIA | INDIA | 1994 | 7534915457 |
| CHINA | VIETNAM | 1994 | 6062387221 |
| CHINA | INDIA | 1994 | 5816794324 |
| VIETNAM | CHINA | 1995 | 15128423080 |
| INDONESIA | CHINA | 1995 | 14794647970 |
| INDIA | CHINA | 1995 | 14724240804 |
| JAPAN | CHINA | 1995 | 14579848516 |
| CHINA | CHINA | 1995 | 14296657586 |
| INDIA | JAPAN | 1995 | 13511381754 |
| JAPAN | JAPAN | 1995 | 12015968288 |
| VIETNAM | INDONESIA | 1995 | 11290647784 |
| JAPAN | INDONESIA | 1995 | 10968840402 |
| INDIA | INDONESIA | 1995 | 10879296370 |
| CHINA | INDONESIA | 1995 | 10611767914 |
| VIETNAM | JAPAN | 1995 | 10493043807 |
| INDONESIA | INDONESIA | 1995 | 10350165199 |
| VIETNAM | INDIA | 1995 | 10147175135 |
| CHINA | JAPAN | 1995 | 9967113498 |
| JAPAN | VIETNAM | 1995 | 9871240910 |
| INDONESIA | JAPAN | 1995 | 9554798320 |
| JAPAN | INDIA | 1995 | 9224478715 |
| INDIA | INDIA | 1995 | 8880501531 |
```

```
| VIETNAM | VIETNAM | 1995 | 8530802028 |
| INDIA | VIETNAM | 1995 | 8470249830 |
| CHINA | INDIA | 1995 | 8460557790 |
| INDONESIA | VIETNAM | 1995 | 8393411088 |
| CHINA | VIETNAM | 1995 | 7838238263 |
| INDONESIA | INDIA | 1995 | 7001659338 |
| JAPAN | CHINA | 1996 | 14974943391 |
| INDIA | CHINA | 1996 | 14236197987 |
| VIETNAM | CHINA | 1996 | 13723231674 |
| JAPAN | INDONESIA | 1996 | 13304501801 |
| INDONESIA | CHINA | 1996 | 12444022202 |
| CHINA | CHINA | 1996 | 12120893189 |
| INDIA | JAPAN | 1996 | 11649117519 |
| INDONESIA | JAPAN | 1996 | 11345350775 |
| VIETNAM | JAPAN | 1996 | 11294284203 |
| INDONESIA | INDONESIA | 1996 | 11111201530 |
| JAPAN | INDIA | 1996 | 10871364136 |
| JAPAN | JAPAN | 1996 | 10836947449 |
| INDIA | INDONESIA | 1996 | 10568008435 |
| JAPAN | VIETNAM | 1996 | 10503890555 |
| VIETNAM | INDONESIA | 1996 | 10494783196 |
| INDONESIA | VIETNAM | 1996 | 9940440124 |
| INDONESIA | INDIA | 1996 | 9864980677 |
| VIETNAM | VIETNAM | 1996 | 9560258720 |
| INDIA | VIETNAM | 1996 | 9324764214 |
| INDIA | INDIA | 1996 | 9023346020 |
| VIETNAM | INDIA | 1996 | 8968179949 |
| CHINA | INDONESIA | 1996 | 8877441837 |
| CHINA | JAPAN | 1996 | 8749420872 |
| CHINA | VIETNAM | 1996 | 6973983457 |
| CHINA | INDIA | 1996 | 6515658476 |
| JAPAN | CHINA | 1997 | 15365039212 |
| INDONESIA | CHINA | 1997 | 14159930904 |
| VIETNAM | CHINA | 1997 | 13678288757 |
| INDIA | CHINA | 1997 | 13599028484 |
| JAPAN | JAPAN | 1997 | 12921870544 |
| CHINA | CHINA | 1997 | 12720975220 |
| VIETNAM | JAPAN | 1997 | 11929000810 |
| VIETNAM | INDONESIA | 1997 | 11325447090 |
| JAPAN | INDONESIA | 1997 | 10764312416 |
| INDONESIA | JAPAN | 1997 | 10555558162 |
| INDONESIA | INDONESIA | 1997 | 10416928126 |
| CHINA | INDONESIA | 1997 | 10317902565 |
| INDIA | JAPAN | 1997 | 10272590051 |
```

```
| JAPAN | VIETNAM | 1997 | 9940032294 |
| CHINA | JAPAN | 1997 | 9519485461 |
| JAPAN | INDIA | 1997 | 9465935835 |
| INDIA | INDONESIA | 1997 | 9405085270 |
| INDONESIA | INDIA | 1997 | 8930955270 |
| INDIA | INDIA | 1997 | 8295504178 |
| VIETNAM | VIETNAM | 1997 | 8293412532 |
| INDONESIA | VIETNAM | 1997 | 8116443059 |
| INDIA | VIETNAM | 1997 | 7960292262 |
| VIETNAM | INDIA | 1997 | 7529455873 |
| CHINA | VIETNAM | 1997 | 7038413355 |
| CHINA | INDIA | 1997 | 6530770558 |
+-----------+-----------+------+-------------+

-- Q3.2

+------------+------------+------+-----------+
| c_city | s_city | year | revenue |
+------------+------------+------+-----------+
| CHINA 3 | CHINA 0 | 1992 | 539864249 |
| CHINA 0 | CHINA 6 | 1992 | 471363128 |
| CHINA 8 | CHINA 1 | 1992 | 421384110 |
| CHINA 6 | CHINA 1 | 1992 | 382204882 |
| CHINA 6 | CHINA 7 | 1992 | 355755835 |
| CHINA 8 | CHINA 9 | 1992 | 349006417 |
| CHINA 7 | CHINA 7 | 1992 | 320232842 |
| CHINA 8 | CHINA 3 | 1992 | 296105733 |
| CHINA 5 | CHINA 3 | 1992 | 277283951 |
| CHINA 6 | CHINA 6 | 1992 | 265527771 |
| CHINA 4 | CHINA 1 | 1992 | 237402078 |
| CHINA 8 | CHINA 6 | 1992 | 234720401 |
| CHINA 4 | CHINA 6 | 1992 | 230169075 |
| CHINA 9 | CHINA 1 | 1992 | 223815249 |
| CHINA 1 | CHINA 1 | 1992 | 223467947 |
| CHINA 2 | CHINA 1 | 1992 | 219559691 |
| CHINA 9 | CHINA 6 | 1992 | 205915890 |
| CHINA 7 | CHINA 9 | 1992 | 201288909 |
| CHINA 1 | CHINA 6 | 1992 | 195622902 |
| CHINA 9 | CHINA 7 | 1992 | 190345063 |
| CHINA 8 | CHINA 4 | 1992 | 174478626 |
| CHINA 1 | CHINA 7 | 1992 | 173803257 |
| CHINA 9 | CHINA 9 | 1992 | 162458028 |
| CHINA 6 | CHINA 0 | 1992 | 154260702 |
| CHINA 8 | CHINA 0 | 1992 | 149794069 |
```

```
| CHINA 5 | CHINA 9 | 1992 | 149369922 |
| CHINA 8 | CHINA 8 | 1992 | 147607252 |
| CHINA 6 | CHINA 4 | 1992 | 147137516 |
| CHINA 7 | CHINA 8 | 1992 | 139974858 |
| CHINA 5 | CHINA 6 | 1992 | 138467127 |
| CHINA 3 | CHINA 6 | 1992 | 119521008 |
| CHINA 8 | CHINA 7 | 1992 | 109887269 |
| CHINA 6 | CHINA 3 | 1992 | 107201214 |
| CHINA 9 | CHINA 4 | 1992 | 101504450 |
| CHINA 1 | CHINA 3 | 1992 | 101388208 |
| CHINA 7 | CHINA 0 | 1992 | 98475237 |
| CHINA 5 | CHINA 8 | 1992 | 98370738 |
| CHINA 2 | CHINA 6 | 1992 | 93254616 |
| CHINA 2 | CHINA 4 | 1992 | 86394644 |
| CHINA 3 | CHINA 7 | 1992 | 81027008 |
| CHINA 5 | CHINA 4 | 1992 | 78587418 |
| CHINA 3 | CHINA 9 | 1992 | 78114762 |
| CHINA 2 | CHINA 0 | 1992 | 77786892 |
| CHINA 2 | CHINA 8 | 1992 | 75605732 |
| CHINA 4 | CHINA 3 | 1992 | 75101512 |
| CHINA 7 | CHINA 4 | 1992 | 74119240 |
| CHINA 2 | CHINA 9 | 1992 | 73413108 |
| CHINA 5 | CHINA 7 | 1992 | 73199718 |
| CHINA 4 | CHINA 4 | 1992 | 72839118 |
| CHINA 1 | CHINA 9 | 1992 | 68538220 |
| CHINA 0 | CHINA 8 | 1992 | 65856888 |
| CHINA 0 | CHINA 9 | 1992 | 65590624 |
| CHINA 3 | CHINA 8 | 1992 | 64556586 |
| CHINA 2 | CHINA 7 | 1992 | 63336330 |
| CHINA 4 | CHINA 9 | 1992 | 57645963 |
| CHINA 0 | CHINA 7 | 1992 | 55251918 |
| CHINA 0 | CHINA 1 | 1992 | 51774462 |
| CHINA 6 | CHINA 8 | 1992 | 45676858 |
| CHINA 3 | CHINA 3 | 1992 | 41147560 |
| CHINA 3 | CHINA 4 | 1992 | 36838082 |
| CHINA 5 | CHINA 0 | 1992 | 36554488 |
| CHINA 3 | CHINA 1 | 1992 | 32036313 |
| CHINA 4 | CHINA 8 | 1992 | 31517575 |
| CHINA 0 | CHINA 3 | 1992 | 25524054 |
| CHINA 1 | CHINA 4 | 1992 | 12681846 |
| CHINA 7 | CHINA 3 | 1992 | 11395152 |
| CHINA 6 | CHINA 9 | 1992 | 8642375 |
| CHINA 8 | CHINA 6 | 1993 | 638396852 |
| CHINA 7 | CHINA 6 | 1993 | 576731239 |
```

```
| CHINA 2 | CHINA 6 | 1993 | 528008729 |
| CHINA 8 | CHINA 9 | 1993 | 522412584 |
| CHINA 8 | CHINA 7 | 1993 | 475478848 |
| CHINA 8 | CHINA 1 | 1993 | 452064153 |
| CHINA 0 | CHINA 1 | 1993 | 425902649 |
| CHINA 9 | CHINA 1 | 1993 | 405252987 |
| CHINA 6 | CHINA 9 | 1993 | 385005953 |
| CHINA 8 | CHINA 8 | 1993 | 382884778 |
| CHINA 0 | CHINA 6 | 1993 | 344911487 |
| CHINA 6 | CHINA 7 | 1993 | 341436211 |
| CHINA 3 | CHINA 6 | 1993 | 291652051 |
| CHINA 7 | CHINA 1 | 1993 | 257769861 |
| CHINA 8 | CHINA 0 | 1993 | 231981252 |
| CHINA 4 | CHINA 6 | 1993 | 215180968 |
| CHINA 3 | CHINA 0 | 1993 | 213320777 |
| CHINA 9 | CHINA 6 | 1993 | 207281000 |
| CHINA 5 | CHINA 9 | 1993 | 206555882 |
| CHINA 6 | CHINA 1 | 1993 | 205665388 |
| CHINA 5 | CHINA 1 | 1993 | 193491875 |
| CHINA 2 | CHINA 9 | 1993 | 193324425 |
| CHINA 5 | CHINA 8 | 1993 | 190521023 |
| CHINA 7 | CHINA 0 | 1993 | 183487919 |
| CHINA 0 | CHINA 9 | 1993 | 170223958 |
| CHINA 6 | CHINA 8 | 1993 | 166821272 |
| CHINA 3 | CHINA 8 | 1993 | 163053528 |
| CHINA 2 | CHINA 0 | 1993 | 158276154 |
| CHINA 3 | CHINA 1 | 1993 | 153652018 |
| CHINA 5 | CHINA 6 | 1993 | 151359347 |
| CHINA 6 | CHINA 0 | 1993 | 140494698 |
| CHINA 8 | CHINA 4 | 1993 | 139857147 |
| CHINA 2 | CHINA 7 | 1993 | 136009418 |
| CHINA 5 | CHINA 7 | 1993 | 133892119 |
| CHINA 9 | CHINA 9 | 1993 | 118965507 |
| CHINA 1 | CHINA 1 | 1993 | 108898379 |
| CHINA 6 | CHINA 6 | 1993 | 100311475 |
| CHINA 0 | CHINA 4 | 1993 | 93483068 |
| CHINA 1 | CHINA 4 | 1993 | 87714152 |
| CHINA 4 | CHINA 1 | 1993 | 87690658 |
| CHINA 4 | CHINA 7 | 1993 | 83701574 |
| CHINA 1 | CHINA 0 | 1993 | 82670983 |
| CHINA 7 | CHINA 4 | 1993 | 77396461 |
| CHINA 5 | CHINA 4 | 1993 | 73556161 |
| CHINA 4 | CHINA 8 | 1993 | 72203335 |
| CHINA 0 | CHINA 7 | 1993 | 70395334 |
```

```
| CHINA 3 | CHINA 4 | 1993 | 64771003 |
| CHINA 7 | CHINA 8 | 1993 | 64514099 |
| CHINA 3 | CHINA 7 | 1993 | 62868516 |
| CHINA 8 | CHINA 3 | 1993 | 56504804 |
| CHINA 2 | CHINA 4 | 1993 | 56031779 |
| CHINA 1 | CHINA 7 | 1993 | 48951262 |
| CHINA 7 | CHINA 3 | 1993 | 45962220 |
| CHINA 4 | CHINA 9 | 1993 | 43158138 |
| CHINA 7 | CHINA 9 | 1993 | 42611979 |
| CHINA 2 | CHINA 8 | 1993 | 38092546 |
| CHINA 1 | CHINA 9 | 1993 | 29665374 |
| CHINA 1 | CHINA 3 | 1993 | 23991216 |
| CHINA 6 | CHINA 6 | 1994 | 596294890 |
| CHINA 8 | CHINA 6 | 1994 | 542104721 |
| CHINA 6 | CHINA 1 | 1994 | 504359553 |
| CHINA 3 | CHINA 7 | 1994 | 476727294 |
| CHINA 3 | CHINA 6 | 1994 | 476349724 |
| CHINA 8 | CHINA 9 | 1994 | 427241348 |
| CHINA 6 | CHINA 9 | 1994 | 358191581 |
| CHINA 9 | CHINA 6 | 1994 | 352344057 |
| CHINA 3 | CHINA 0 | 1994 | 351708546 |
| CHINA 8 | CHINA 0 | 1994 | 351131413 |
| CHINA 3 | CHINA 3 | 1994 | 339279574 |
| CHINA 0 | CHINA 1 | 1994 | 298307857 |
| CHINA 0 | CHINA 7 | 1994 | 289536010 |
| CHINA 0 | CHINA 6 | 1994 | 285639032 |
| CHINA 7 | CHINA 6 | 1994 | 263170455 |
| CHINA 2 | CHINA 8 | 1994 | 250332990 |
| CHINA 6 | CHINA 4 | 1994 | 235897763 |
| CHINA 5 | CHINA 1 | 1994 | 234681515 |
| CHINA 8 | CHINA 7 | 1994 | 234390101 |
| CHINA 1 | CHINA 6 | 1994 | 232792764 |
| CHINA 8 | CHINA 1 | 1994 | 223808842 |
| CHINA 4 | CHINA 6 | 1994 | 209522926 |
| CHINA 8 | CHINA 4 | 1994 | 208632636 |
| CHINA 7 | CHINA 3 | 1994 | 202424117 |
| CHINA 4 | CHINA 7 | 1994 | 185487544 |
| CHINA 2 | CHINA 7 | 1994 | 183551771 |
| CHINA 7 | CHINA 1 | 1994 | 178421732 |
| CHINA 4 | CHINA 1 | 1994 | 176262868 |
| CHINA 5 | CHINA 6 | 1994 | 173651872 |
| CHINA 0 | CHINA 4 | 1994 | 173584501 |
| CHINA 8 | CHINA 8 | 1994 | 172179808 |
| CHINA 9 | CHINA 1 | 1994 | 169617585 |
```

```
| CHINA 0 | CHINA 9 | 1994 | 167569085 |
| CHINA 5 | CHINA 8 | 1994 | 162066559 |
| CHINA 7 | CHINA 9 | 1994 | 161041255 |
| CHINA 5 | CHINA 4 | 1994 | 154820955 |
| CHINA 7 | CHINA 0 | 1994 | 152844960 |
| CHINA 2 | CHINA 6 | 1994 | 149839190 |
| CHINA 7 | CHINA 8 | 1994 | 149536114 |
| CHINA 1 | CHINA 4 | 1994 | 142403628 |
| CHINA 9 | CHINA 9 | 1994 | 131064832 |
| CHINA 2 | CHINA 1 | 1994 | 124489283 |
| CHINA 2 | CHINA 0 | 1994 | 114263273 |
| CHINA 5 | CHINA 7 | 1994 | 113311766 |
| CHINA 8 | CHINA 3 | 1994 | 112573609 |
| CHINA 3 | CHINA 4 | 1994 | 104903651 |
| CHINA 4 | CHINA 0 | 1994 | 101914439 |
| CHINA 3 | CHINA 1 | 1994 | 98253251 |
| CHINA 1 | CHINA 7 | 1994 | 94582288 |
| CHINA 4 | CHINA 4 | 1994 | 92818317 |
| CHINA 1 | CHINA 9 | 1994 | 85220541 |
| CHINA 6 | CHINA 3 | 1994 | 84604801 |
| CHINA 0 | CHINA 3 | 1994 | 77574978 |
| CHINA 1 | CHINA 3 | 1994 | 74435316 |
| CHINA 4 | CHINA 9 | 1994 | 72622300 |
| CHINA 3 | CHINA 8 | 1994 | 72559366 |
| CHINA 9 | CHINA 0 | 1994 | 69298222 |
| CHINA 3 | CHINA 9 | 1994 | 67472592 |
| CHINA 6 | CHINA 8 | 1994 | 66271372 |
| CHINA 7 | CHINA 4 | 1994 | 59634606 |
| CHINA 2 | CHINA 9 | 1994 | 56882136 |
| CHINA 1 | CHINA 1 | 1994 | 56592337 |
| CHINA 5 | CHINA 9 | 1994 | 52879724 |
| CHINA 9 | CHINA 4 | 1994 | 49324497 |
| CHINA 2 | CHINA 3 | 1994 | 45042384 |
| CHINA 7 | CHINA 7 | 1994 | 44458451 |
| CHINA 5 | CHINA 0 | 1994 | 39091925 |
| CHINA 9 | CHINA 3 | 1994 | 39082405 |
| CHINA 0 | CHINA 8 | 1994 | 28203459 |
| CHINA 6 | CHINA 7 | 1994 | 27243775 |
| CHINA 0 | CHINA 0 | 1994 | 15591040 |
| CHINA 2 | CHINA 6 | 1995 | 832176707 |
| CHINA 8 | CHINA 6 | 1995 | 793322102 |
| CHINA 3 | CHINA 7 | 1995 | 505446788 |
| CHINA 7 | CHINA 9 | 1995 | 483519933 |
| CHINA 4 | CHINA 6 | 1995 | 440320366 |
```

```
| CHINA 8 | CHINA 1 | 1995 | 394522570 |
| CHINA 7 | CHINA 1 | 1995 | 393861389 |
| CHINA 5 | CHINA 1 | 1995 | 343166828 |
| CHINA 1 | CHINA 7 | 1995 | 341736584 |
| CHINA 8 | CHINA 7 | 1995 | 323623203 |
| CHINA 6 | CHINA 6 | 1995 | 312876143 |
| CHINA 3 | CHINA 6 | 1995 | 306516324 |
| CHINA 7 | CHINA 6 | 1995 | 294840537 |
| CHINA 3 | CHINA 3 | 1995 | 290066240 |
| CHINA 8 | CHINA 3 | 1995 | 289182495 |
| CHINA 3 | CHINA 1 | 1995 | 288853766 |
| CHINA 0 | CHINA 1 | 1995 | 279082523 |
| CHINA 0 | CHINA 8 | 1995 | 265291443 |
| CHINA 1 | CHINA 6 | 1995 | 262283412 |
| CHINA 4 | CHINA 1 | 1995 | 246559891 |
| CHINA 2 | CHINA 8 | 1995 | 246465167 |
| CHINA 6 | CHINA 7 | 1995 | 246385862 |
| CHINA 9 | CHINA 6 | 1995 | 231314393 |
| CHINA 2 | CHINA 7 | 1995 | 224354491 |
| CHINA 4 | CHINA 7 | 1995 | 222368398 |
| CHINA 0 | CHINA 7 | 1995 | 221334917 |
| CHINA 6 | CHINA 3 | 1995 | 217756587 |
| CHINA 6 | CHINA 9 | 1995 | 215736018 |
| CHINA 4 | CHINA 9 | 1995 | 210496516 |
| CHINA 0 | CHINA 6 | 1995 | 197891458 |
| CHINA 8 | CHINA 9 | 1995 | 192018213 |
| CHINA 7 | CHINA 0 | 1995 | 188804482 |
| CHINA 5 | CHINA 6 | 1995 | 186378531 |
| CHINA 6 | CHINA 1 | 1995 | 165831073 |
| CHINA 1 | CHINA 3 | 1995 | 165118263 |
| CHINA 6 | CHINA 8 | 1995 | 157640218 |
| CHINA 1 | CHINA 1 | 1995 | 150838433 |
| CHINA 1 | CHINA 4 | 1995 | 147632879 |
| CHINA 6 | CHINA 0 | 1995 | 147314401 |
| CHINA 5 | CHINA 4 | 1995 | 142820978 |
| CHINA 5 | CHINA 9 | 1995 | 141416829 |
| CHINA 2 | CHINA 0 | 1995 | 135608473 |
| CHINA 5 | CHINA 7 | 1995 | 131596218 |
| CHINA 0 | CHINA 4 | 1995 | 129159370 |
| CHINA 3 | CHINA 9 | 1995 | 126837748 |
| CHINA 8 | CHINA 0 | 1995 | 126564932 |
| CHINA 0 | CHINA 3 | 1995 | 121337041 |
| CHINA 7 | CHINA 7 | 1995 | 118697587 |
| CHINA 5 | CHINA 8 | 1995 | 116538842 |
```

```
| CHINA 8 | CHINA 8 | 1995 | 110161904 |
| CHINA 9 | CHINA 0 | 1995 | 109582187 |
| CHINA 9 | CHINA 1 | 1995 | 103455098 |
| CHINA 2 | CHINA 1 | 1995 | 100264691 |
| CHINA 7 | CHINA 3 | 1995 | 99011859 |
| CHINA 3 | CHINA 0 | 1995 | 90383390 |
| CHINA 4 | CHINA 3 | 1995 | 89908903 |
| CHINA 7 | CHINA 8 | 1995 | 81425699 |
| CHINA 3 | CHINA 4 | 1995 | 77577579 |
| CHINA 4 | CHINA 8 | 1995 | 74805746 |
| CHINA 9 | CHINA 7 | 1995 | 74597020 |
| CHINA 9 | CHINA 9 | 1995 | 73514511 |
| CHINA 5 | CHINA 0 | 1995 | 73274726 |
| CHINA 8 | CHINA 4 | 1995 | 61708487 |
| CHINA 1 | CHINA 0 | 1995 | 58753734 |
| CHINA 3 | CHINA 8 | 1995 | 57133566 |
| CHINA 9 | CHINA 4 | 1995 | 53259334 |
| CHINA 1 | CHINA 9 | 1995 | 46177797 |
| CHINA 2 | CHINA 4 | 1995 | 45147325 |
| CHINA 0 | CHINA 0 | 1995 | 43963173 |
| CHINA 0 | CHINA 9 | 1995 | 40184107 |
| CHINA 1 | CHINA 8 | 1995 | 18859188 |
| CHINA 8 | CHINA 7 | 1996 | 621957444 |
| CHINA 3 | CHINA 9 | 1996 | 530082848 |
| CHINA 8 | CHINA 6 | 1996 | 525755549 |
| CHINA 8 | CHINA 1 | 1996 | 399229343 |
| CHINA 6 | CHINA 7 | 1996 | 365540749 |
| CHINA 8 | CHINA 8 | 1996 | 351864283 |
| CHINA 1 | CHINA 6 | 1996 | 329186504 |
| CHINA 9 | CHINA 6 | 1996 | 321113085 |
| CHINA 3 | CHINA 6 | 1996 | 318264871 |
| CHINA 2 | CHINA 6 | 1996 | 315233397 |
| CHINA 2 | CHINA 9 | 1996 | 285852841 |
| CHINA 9 | CHINA 9 | 1996 | 264510548 |
| CHINA 5 | CHINA 6 | 1996 | 261385523 |
| CHINA 8 | CHINA 9 | 1996 | 259497265 |
| CHINA 6 | CHINA 6 | 1996 | 258200131 |
| CHINA 4 | CHINA 9 | 1996 | 257345949 |
| CHINA 6 | CHINA 9 | 1996 | 247667288 |
| CHINA 2 | CHINA 7 | 1996 | 234569026 |
| CHINA 2 | CHINA 1 | 1996 | 218568966 |
| CHINA 4 | CHINA 1 | 1996 | 207383476 |
| CHINA 0 | CHINA 1 | 1996 | 204596428 |
| CHINA 3 | CHINA 0 | 1996 | 204375870 |
```

```
| CHINA 4 | CHINA 0 | 1996 | 202299286 |
| CHINA 4 | CHINA 4 | 1996 | 191983261 |
| CHINA 4 | CHINA 8 | 1996 | 183961012 |
| CHINA 4 | CHINA 6 | 1996 | 183872085 |
| CHINA 6 | CHINA 8 | 1996 | 182132356 |
| CHINA 7 | CHINA 9 | 1996 | 170941341 |
| CHINA 0 | CHINA 6 | 1996 | 168082672 |
| CHINA 1 | CHINA 7 | 1996 | 165942066 |
| CHINA 1 | CHINA 9 | 1996 | 165878775 |
| CHINA 9 | CHINA 8 | 1996 | 156009357 |
| CHINA 7 | CHINA 7 | 1996 | 155842944 |
| CHINA 2 | CHINA 0 | 1996 | 147709906 |
| CHINA 5 | CHINA 7 | 1996 | 147257366 |
| CHINA 1 | CHINA 8 | 1996 | 141840928 |
| CHINA 2 | CHINA 4 | 1996 | 136244052 |
| CHINA 9 | CHINA 0 | 1996 | 130997019 |
| CHINA 1 | CHINA 0 | 1996 | 124362038 |
| CHINA 0 | CHINA 9 | 1996 | 114011231 |
| CHINA 7 | CHINA 3 | 1996 | 112398764 |
| CHINA 4 | CHINA 7 | 1996 | 110567337 |
| CHINA 3 | CHINA 4 | 1996 | 109269982 |
| CHINA 5 | CHINA 1 | 1996 | 107482704 |
| CHINA 6 | CHINA 4 | 1996 | 105485170 |
| CHINA 1 | CHINA 4 | 1996 | 105320270 |
| CHINA 0 | CHINA 7 | 1996 | 102545071 |
| CHINA 2 | CHINA 3 | 1996 | 100407151 |
| CHINA 0 | CHINA 4 | 1996 | 95913303 |
| CHINA 7 | CHINA 0 | 1996 | 94706269 |
| CHINA 6 | CHINA 1 | 1996 | 86949951 |
| CHINA 8 | CHINA 3 | 1996 | 84157344 |
| CHINA 2 | CHINA 8 | 1996 | 83176903 |
| CHINA 5 | CHINA 9 | 1996 | 83104330 |
| CHINA 7 | CHINA 8 | 1996 | 81490639 |
| CHINA 9 | CHINA 3 | 1996 | 79655829 |
| CHINA 5 | CHINA 0 | 1996 | 77489995 |
| CHINA 8 | CHINA 0 | 1996 | 76989056 |
| CHINA 9 | CHINA 1 | 1996 | 72011031 |
| CHINA 7 | CHINA 4 | 1996 | 64764322 |
| CHINA 5 | CHINA 4 | 1996 | 62827767 |
| CHINA 5 | CHINA 8 | 1996 | 62673237 |
| CHINA 7 | CHINA 6 | 1996 | 61880459 |
| CHINA 3 | CHINA 7 | 1996 | 56642844 |
| CHINA 3 | CHINA 1 | 1996 | 50799366 |
| CHINA 3 | CHINA 3 | 1996 | 42601269 |
```

```
| CHINA 4 | CHINA 3 | 1996 | 38290290 |
| CHINA 3 | CHINA 8 | 1996 | 21263056 |
| CHINA 7 | CHINA 1 | 1996 | 14836937 |
| CHINA 5 | CHINA 3 | 1996 | 13611339 |
| CHINA 1 | CHINA 3 | 1996 | 8430793 |
| CHINA 1 | CHINA 1 | 1996 | 1601332 |
| CHINA 1 | CHINA 7 | 1997 | 664436721 |
| CHINA 8 | CHINA 9 | 1997 | 585552148 |
| CHINA 8 | CHINA 6 | 1997 | 543571889 |
| CHINA 8 | CHINA 7 | 1997 | 516131917 |
| CHINA 6 | CHINA 7 | 1997 | 467477883 |
| CHINA 3 | CHINA 9 | 1997 | 444914344 |
| CHINA 5 | CHINA 6 | 1997 | 353316321 |
| CHINA 6 | CHINA 4 | 1997 | 338136205 |
| CHINA 0 | CHINA 7 | 1997 | 329137493 |
| CHINA 5 | CHINA 1 | 1997 | 328142466 |
| CHINA 8 | CHINA 4 | 1997 | 308276385 |
| CHINA 6 | CHINA 9 | 1997 | 306814317 |
| CHINA 5 | CHINA 9 | 1997 | 301145803 |
| CHINA 7 | CHINA 1 | 1997 | 299575802 |
| CHINA 8 | CHINA 8 | 1997 | 282083295 |
| CHINA 4 | CHINA 9 | 1997 | 280242025 |
| CHINA 9 | CHINA 1 | 1997 | 253155313 |
| CHINA 4 | CHINA 6 | 1997 | 234247182 |
| CHINA 5 | CHINA 0 | 1997 | 217246162 |
| CHINA 9 | CHINA 4 | 1997 | 215424663 |
| CHINA 0 | CHINA 6 | 1997 | 211152240 |
| CHINA 3 | CHINA 6 | 1997 | 205982217 |
| CHINA 7 | CHINA 6 | 1997 | 196440117 |
| CHINA 1 | CHINA 6 | 1997 | 195757737 |
| CHINA 2 | CHINA 3 | 1997 | 189836909 |
| CHINA 7 | CHINA 8 | 1997 | 189291379 |
| CHINA 9 | CHINA 6 | 1997 | 189236146 |
| CHINA 3 | CHINA 1 | 1997 | 188537684 |
| CHINA 9 | CHINA 7 | 1997 | 182516267 |
| CHINA 0 | CHINA 0 | 1997 | 182459980 |
| CHINA 5 | CHINA 8 | 1997 | 177077882 |
| CHINA 2 | CHINA 6 | 1997 | 176030529 |
| CHINA 2 | CHINA 1 | 1997 | 168770050 |
| CHINA 8 | CHINA 0 | 1997 | 167294093 |
| CHINA 4 | CHINA 3 | 1997 | 161980658 |
| CHINA 3 | CHINA 4 | 1997 | 154433882 |
| CHINA 6 | CHINA 6 | 1997 | 153336736 |
| CHINA 6 | CHINA 3 | 1997 | 151596497 |
```

```
| CHINA 8 | CHINA 1 | 1997 | 145432603 |
| CHINA 1 | CHINA 4 | 1997 | 126773981 |
| CHINA 1 | CHINA 0 | 1997 | 120594770 |
| CHINA 7 | CHINA 3 | 1997 | 119618460 |
| CHINA 6 | CHINA 1 | 1997 | 119529805 |
| CHINA 2 | CHINA 9 | 1997 | 114591288 |
| CHINA 7 | CHINA 7 | 1997 | 111335941 |
| CHINA 5 | CHINA 3 | 1997 | 111044153 |
| CHINA 6 | CHINA 0 | 1997 | 104404276 |
| CHINA 1 | CHINA 1 | 1997 | 98869501 |
| CHINA 7 | CHINA 0 | 1997 | 97198605 |
| CHINA 7 | CHINA 9 | 1997 | 92872632 |
| CHINA 0 | CHINA 9 | 1997 | 91097832 |
| CHINA 9 | CHINA 9 | 1997 | 86479272 |
| CHINA 2 | CHINA 7 | 1997 | 79380820 |
| CHINA 9 | CHINA 0 | 1997 | 78499693 |
| CHINA 1 | CHINA 9 | 1997 | 73589328 |
| CHINA 2 | CHINA 8 | 1997 | 71633835 |
| CHINA 8 | CHINA 3 | 1997 | 70505885 |
| CHINA 3 | CHINA 0 | 1997 | 61039282 |
| CHINA 0 | CHINA 3 | 1997 | 58325113 |
| CHINA 5 | CHINA 7 | 1997 | 55476389 |
| CHINA 4 | CHINA 7 | 1997 | 46480159 |
| CHINA 0 | CHINA 1 | 1997 | 38223038 |
| CHINA 4 | CHINA 1 | 1997 | 21636342 |
| CHINA 9 | CHINA 3 | 1997 | 13092788 |
| CHINA 6 | CHINA 8 | 1997 | 2490092 |
+------------+------------+------+-----------+

-- Q3.3
+------------+------------+------+-----------+
| c_city | s_city | year | revenue |
+------------+------------+------+-----------+
| UNITED KI0 | UNITED KI7 | 1992 | 251282102 |
| UNITED KI0 | UNITED KI0 | 1992 | 170005406 |
| UNITED KI7 | UNITED KI7 | 1992 | 36835396 |
| UNITED KI0 | UNITED KI7 | 1993 | 560335810 |
| UNITED KI0 | UNITED KI0 | 1993 | 294257692 |
| UNITED KI7 | UNITED KI0 | 1993 | 159005896 |
| UNITED KI7 | UNITED KI7 | 1993 | 139029264 |
| UNITED KI0 | UNITED KI7 | 1994 | 739847089 |
| UNITED KI0 | UNITED KI0 | 1994 | 302339390 |
| UNITED KI7 | UNITED KI7 | 1994 | 275609814 |
| UNITED KI7 | UNITED KI0 | 1994 | 117654093 |
```

```
| UNITED KI0 | UNITED KI7 | 1995 | 540994655 |
| UNITED KI0 | UNITED KI0 | 1995 | 230825439 |
| UNITED KI7 | UNITED KI0 | 1995 | 197347696 |
| UNITED KI7 | UNITED KI7 | 1995 | 136620517 |
| UNITED KI0 | UNITED KI7 | 1996 | 448412094 |
| UNITED KI0 | UNITED KI0 | 1996 | 203511607 |
| UNITED KI7 | UNITED KI7 | 1996 | 94528075 |
| UNITED KI7 | UNITED KI0 | 1996 | 35448536 |
| UNITED KI7 | UNITED KI0 | 1997 | 289323850 |
| UNITED KI7 | UNITED KI7 | 1997 | 214791175 |
| UNITED KI0 | UNITED KI7 | 1997 | 196510174 |
| UNITED KI0 | UNITED KI0 | 1997 | 125066127 |
+------------+------------+------+-----------+

-- Q3.4
+------------+------------+------+-----------+
| c_city | s_city | year | revenue |
+------------+------------+------+-----------+
| UNITED KI7 | KENYA 4 | 1997 | 170083300 |
| UNITED KI0 | MOZAMBIQU1 | 1997 | 155234463 |
| UNITED KI0 | KENYA 4 | 1997 | 87283610 |
+------------+------------+------+-----------+

-- Q4.1
+------+---------------+-------------+
| year | c_nation | profit |
+------+---------------+-------------+
| 1992 | ARGENTINA | 13746243380 |
| 1992 | BRAZIL | 15762831144 |
| 1992 | CANADA | 17477043721 |
| 1992 | PERU | 14698567030 |
| 1992 | UNITED STATES | 14043501956 |
| 1993 | ARGENTINA | 13992888207 |
| 1993 | BRAZIL | 15146262693 |
| 1993 | CANADA | 12463985574 |
| 1993 | PERU | 11385007831 |
| 1993 | UNITED STATES | 10651361815 |
| 1994 | ARGENTINA | 13128610315 |
| 1994 | BRAZIL | 13764866493 |
| 1994 | CANADA | 13723188154 |
| 1994 | PERU | 12784683808 |
| 1994 | UNITED STATES | 12554422837 |
| 1995 | ARGENTINA | 14337205612 |
| 1995 | BRAZIL | 15068918320 |
```

```
| 1995 | CANADA | 14529005783 |
| 1995 | PERU | 13086675480 |
| 1995 | UNITED STATES | 11330297649 |
| 1996 | ARGENTINA | 13659108915 |
| 1996 | BRAZIL | 12660837584 |
| 1996 | CANADA | 14558903190 |
| 1996 | PERU | 14162285166 |
| 1996 | UNITED STATES | 11117076866 |
| 1997 | ARGENTINA | 12556399750 |
| 1997 | BRAZIL | 13961587144 |
| 1997 | CANADA | 15567856947 |
| 1997 | PERU | 13595325340 |
| 1997 | UNITED STATES | 10779073839 |
| 1998 | ARGENTINA | 7843424759 |
| 1998 | BRAZIL | 8853904827 |
| 1998 | CANADA | 8286104334 |
| 1998 | PERU | 5822590950 |
| 1998 | UNITED STATES | 8526236814 |
+------+---------------+-------------+

-- Q4.2
+------+---------------+------------+------------+
| year | s_nation | p_category | profit |
+------+---------------+------------+------------+
| 1997 | ARGENTINA | MFGR#11 | 1636950553 |
| 1997 | ARGENTINA | MFGR#12 | 1265547847 |
| 1997 | ARGENTINA | MFGR#13 | 1505131346 |
| 1997 | ARGENTINA | MFGR#14 | 1405447137 |
| 1997 | ARGENTINA | MFGR#15 | 1564085340 |
| 1997 | ARGENTINA | MFGR#21 | 1335009490 |
| 1997 | ARGENTINA | MFGR#22 | 1309054179 |
| 1997 | ARGENTINA | MFGR#23 | 1305213794 |
| 1997 | ARGENTINA | MFGR#24 | 1089725126 |
| 1997 | ARGENTINA | MFGR#25 | 1291995512 |
| 1997 | BRAZIL | MFGR#11 | 721240147 |
| 1997 | BRAZIL | MFGR#12 | 928318830 |
| 1997 | BRAZIL | MFGR#13 | 1164674879 |
| 1997 | BRAZIL | MFGR#14 | 1215622587 |
| 1997 | BRAZIL | MFGR#15 | 940971658 |
| 1997 | BRAZIL | MFGR#21 | 1158909618 |
| 1997 | BRAZIL | MFGR#22 | 1251221641 |
| 1997 | BRAZIL | MFGR#23 | 1552552455 |
| 1997 | BRAZIL | MFGR#24 | 929057361 |
| 1997 | BRAZIL | MFGR#25 | 574645288 |
```

```
| 1997 | CANADA | MFGR#11 | 1170341370 |
| 1997 | CANADA | MFGR#12 | 1220238121 |
| 1997 | CANADA | MFGR#13 | 1245774025 |
| 1997 | CANADA | MFGR#14 | 1032046642 |
| 1997 | CANADA | MFGR#15 | 738650612 |
| 1997 | CANADA | MFGR#21 | 1476055209 |
| 1997 | CANADA | MFGR#22 | 1239005798 |
| 1997 | CANADA | MFGR#23 | 869393804 |
| 1997 | CANADA | MFGR#24 | 1466964051 |
| 1997 | CANADA | MFGR#25 | 1358922727 |
| 1997 | PERU | MFGR#11 | 1031023174 |
| 1997 | PERU | MFGR#12 | 731821491 |
| 1997 | PERU | MFGR#13 | 1044642877 |
| 1997 | PERU | MFGR#14 | 654877417 |
| 1997 | PERU | MFGR#15 | 1201769474 |
| 1997 | PERU | MFGR#21 | 1275496672 |
| 1997 | PERU | MFGR#22 | 599324545 |
| 1997 | PERU | MFGR#23 | 1200754744 |
| 1997 | PERU | MFGR#24 | 942152801 |
| 1997 | PERU | MFGR#25 | 1064322995 |
| 1997 | UNITED STATES | MFGR#11 | 2365218925 |
| 1997 | UNITED STATES | MFGR#12 | 1132346574 |
| 1997 | UNITED STATES | MFGR#13 | 2460882362 |
| 1997 | UNITED STATES | MFGR#14 | 2190816877 |
| 1997 | UNITED STATES | MFGR#15 | 1687829921 |
| 1997 | UNITED STATES | MFGR#21 | 2125880770 |
| 1997 | UNITED STATES | MFGR#22 | 2013348097 |
| 1997 | UNITED STATES | MFGR#23 | 2570581084 |
| 1997 | UNITED STATES | MFGR#24 | 2724372315 |
| 1997 | UNITED STATES | MFGR#25 | 1480012758 |
| 1998 | ARGENTINA | MFGR#11 | 783662770 |
| 1998 | ARGENTINA | MFGR#12 | 472818450 |
| 1998 | ARGENTINA | MFGR#13 | 585091533 |
| 1998 | ARGENTINA | MFGR#14 | 507297527 |
| 1998 | ARGENTINA | MFGR#15 | 549185408 |
| 1998 | ARGENTINA | MFGR#21 | 972928972 |
| 1998 | ARGENTINA | MFGR#22 | 1508294213 |
| 1998 | ARGENTINA | MFGR#23 | 517896738 |
| 1998 | ARGENTINA | MFGR#24 | 240754731 |
| 1998 | ARGENTINA | MFGR#25 | 757030162 |
| 1998 | BRAZIL | MFGR#11 | 826283793 |
| 1998 | BRAZIL | MFGR#12 | 482293349 |
| 1998 | BRAZIL | MFGR#13 | 1037202334 |
| 1998 | BRAZIL | MFGR#14 | 743598666 |
```

```
| 1998 | BRAZIL | MFGR#15 | 584176304 |
| 1998 | BRAZIL | MFGR#21 | 557259779 |
| 1998 | BRAZIL | MFGR#22 | 535654445 |
| 1998 | BRAZIL | MFGR#23 | 403656721 |
| 1998 | BRAZIL | MFGR#24 | 1305217551 |
| 1998 | BRAZIL | MFGR#25 | 1109801463 |
| 1998 | CANADA | MFGR#11 | 936169617 |
| 1998 | CANADA | MFGR#12 | 1017751308 |
| 1998 | CANADA | MFGR#13 | 850046376 |
| 1998 | CANADA | MFGR#14 | 808138010 |
| 1998 | CANADA | MFGR#15 | 701990010 |
| 1998 | CANADA | MFGR#21 | 402611051 |
| 1998 | CANADA | MFGR#22 | 382705122 |
| 1998 | CANADA | MFGR#23 | 509674722 |
| 1998 | CANADA | MFGR#24 | 1003021250 |
| 1998 | CANADA | MFGR#25 | 574602788 |
| 1998 | PERU | MFGR#11 | 552608732 |
| 1998 | PERU | MFGR#12 | 500581456 |
| 1998 | PERU | MFGR#13 | 894607711 |
| 1998 | PERU | MFGR#14 | 386487826 |
| 1998 | PERU | MFGR#15 | 1044780577 |
| 1998 | PERU | MFGR#21 | 184346232 |
| 1998 | PERU | MFGR#22 | 674942976 |
| 1998 | PERU | MFGR#23 | 665523956 |
| 1998 | PERU | MFGR#24 | 631374203 |
| 1998 | PERU | MFGR#25 | 602609608 |
| 1998 | UNITED STATES | MFGR#11 | 1230069867 |
| 1998 | UNITED STATES | MFGR#12 | 1557720319 |
| 1998 | UNITED STATES | MFGR#13 | 999206739 |
| 1998 | UNITED STATES | MFGR#14 | 605040268 |
| 1998 | UNITED STATES | MFGR#15 | 850219215 |
| 1998 | UNITED STATES | MFGR#21 | 1032550760 |
| 1998 | UNITED STATES | MFGR#22 | 1370141401 |
| 1998 | UNITED STATES | MFGR#23 | 1226632297 |
| 1998 | UNITED STATES | MFGR#24 | 1528135100 |
| 1998 | UNITED STATES | MFGR#25 | 1127867278 |
+------+---------------+------------+------------+

-- Q4.3
+------+------------+-----------+-----------+
| year | s_city | p_brand | profit |
+------+------------+-----------+-----------+
| 1997 | UNITED ST0 | MFGR#1410 | 58481513 |
| 1997 | UNITED ST0 | MFGR#1412 | 33582225 |
```

```
| 1997 | UNITED ST0 | MFGR#1413 | 135625490 |
| 1997 | UNITED ST0 | MFGR#1414 | 18581969 |
| 1997 | UNITED ST0 | MFGR#142 | 164080005 |
| 1997 | UNITED ST0 | MFGR#1420 | 30831591 |
| 1997 | UNITED ST0 | MFGR#1424 | 4085253 |
| 1997 | UNITED ST0 | MFGR#1425 | 163183170 |
| 1997 | UNITED ST0 | MFGR#1427 | 87578288 |
| 1997 | UNITED ST0 | MFGR#1428 | 109488143 |
| 1997 | UNITED ST0 | MFGR#143 | 198055627 |
| 1997 | UNITED ST0 | MFGR#1430 | 52544552 |
| 1997 | UNITED ST0 | MFGR#1432 | 158742311 |
| 1997 | UNITED ST0 | MFGR#144 | 43479982 |
| 1997 | UNITED ST0 | MFGR#1440 | 40412893 |
| 1997 | UNITED ST0 | MFGR#145 | 175568435 |
| 1997 | UNITED ST1 | MFGR#141 | 11932912 |
| 1997 | UNITED ST1 | MFGR#1411 | 40637463 |
| 1997 | UNITED ST1 | MFGR#1415 | 27562355 |
| 1997 | UNITED ST1 | MFGR#1421 | 100271780 |
| 1997 | UNITED ST1 | MFGR#1422 | 103286764 |
| 1997 | UNITED ST1 | MFGR#1423 | 106114459 |
| 1997 | UNITED ST1 | MFGR#1427 | 157715681 |
| 1997 | UNITED ST1 | MFGR#1428 | 91550168 |
| 1997 | UNITED ST1 | MFGR#1430 | 56560173 |
| 1997 | UNITED ST1 | MFGR#1431 | 248448914 |
| 1997 | UNITED ST1 | MFGR#1435 | 994228 |
| 1997 | UNITED ST1 | MFGR#144 | 55729825 |
| 1997 | UNITED ST1 | MFGR#145 | 118034196 |
| 1997 | UNITED ST1 | MFGR#146 | 99170724 |
| 1997 | UNITED ST1 | MFGR#147 | 5123001 |
| 1997 | UNITED ST2 | MFGR#141 | 111908637 |
| 1997 | UNITED ST2 | MFGR#1414 | 96864725 |
| 1997 | UNITED ST2 | MFGR#1415 | 123601050 |
| 1997 | UNITED ST2 | MFGR#1421 | 21014618 |
| 1997 | UNITED ST2 | MFGR#1427 | 46524767 |
| 1997 | UNITED ST2 | MFGR#1429 | 18800062 |
| 1997 | UNITED ST2 | MFGR#1431 | 79199532 |
| 1997 | UNITED ST2 | MFGR#1432 | 53841788 |
| 1997 | UNITED ST2 | MFGR#1433 | 133842836 |
| 1997 | UNITED ST2 | MFGR#1434 | 96443006 |
| 1997 | UNITED ST2 | MFGR#1435 | 50858424 |
| 1997 | UNITED ST2 | MFGR#1438 | 64571457 |
| 1997 | UNITED ST2 | MFGR#144 | 61319000 |
| 1997 | UNITED ST2 | MFGR#146 | 69558050 |
| 1997 | UNITED ST2 | MFGR#147 | 41160961 |
```

```
| 1997 | UNITED ST2 | MFGR#149 | 31735872 |
| 1997 | UNITED ST3 | MFGR#1410 | 306449140 |
| 1997 | UNITED ST3 | MFGR#1411 | 114677189 |
| 1997 | UNITED ST3 | MFGR#1412 | 49229127 |
| 1997 | UNITED ST3 | MFGR#1413 | 174911640 |
| 1997 | UNITED ST3 | MFGR#1415 | 134932298 |
| 1997 | UNITED ST3 | MFGR#1416 | 97111854 |
| 1997 | UNITED ST3 | MFGR#1417 | 176279103 |
| 1997 | UNITED ST3 | MFGR#1418 | 70684147 |
| 1997 | UNITED ST3 | MFGR#1420 | 27591782 |
| 1997 | UNITED ST3 | MFGR#1422 | 39411253 |
| 1997 | UNITED ST3 | MFGR#1424 | 226736650 |
| 1997 | UNITED ST3 | MFGR#1426 | 63997112 |
| 1997 | UNITED ST3 | MFGR#1429 | 556053 |
| 1997 | UNITED ST3 | MFGR#143 | 73550925 |
| 1997 | UNITED ST3 | MFGR#1430 | 218807697 |
| 1997 | UNITED ST3 | MFGR#1431 | 39936281 |
| 1997 | UNITED ST3 | MFGR#1432 | 44356689 |
| 1997 | UNITED ST3 | MFGR#1435 | 49225455 |
| 1997 | UNITED ST3 | MFGR#1436 | 90326644 |
| 1997 | UNITED ST3 | MFGR#1439 | 84615817 |
| 1997 | UNITED ST3 | MFGR#144 | 59081596 |
| 1997 | UNITED ST3 | MFGR#1440 | 59601014 |
| 1997 | UNITED ST3 | MFGR#145 | 100692258 |
| 1997 | UNITED ST3 | MFGR#147 | 142417874 |
| 1997 | UNITED ST3 | MFGR#148 | 38233221 |
| 1997 | UNITED ST5 | MFGR#1416 | 62387773 |
| 1997 | UNITED ST5 | MFGR#1417 | 54974702 |
| 1997 | UNITED ST5 | MFGR#1418 | 87301086 |
| 1997 | UNITED ST5 | MFGR#1421 | 9869673 |
| 1997 | UNITED ST5 | MFGR#1422 | 58912225 |
| 1997 | UNITED ST5 | MFGR#1424 | 80038584 |
| 1997 | UNITED ST5 | MFGR#1428 | 44422717 |
| 1997 | UNITED ST5 | MFGR#1430 | 67186074 |
| 1997 | UNITED ST5 | MFGR#1433 | 105646942 |
| 1997 | UNITED ST5 | MFGR#1434 | 13923867 |
| 1997 | UNITED ST5 | MFGR#145 | 104286534 |
| 1997 | UNITED ST5 | MFGR#146 | 20965182 |
| 1997 | UNITED ST5 | MFGR#148 | 170596496 |
| 1997 | UNITED ST5 | MFGR#149 | 42639213 |
| 1997 | UNITED ST6 | MFGR#1411 | 48199726 |
| 1997 | UNITED ST6 | MFGR#1413 | 28825982 |
| 1997 | UNITED ST6 | MFGR#1414 | 107783723 |
| 1997 | UNITED ST6 | MFGR#1415 | 92119787 |
```

```
| 1997 | UNITED ST6 | MFGR#1416 | 35390328 |
| 1997 | UNITED ST6 | MFGR#1417 | 92594053 |
| 1997 | UNITED ST6 | MFGR#1418 | 67638716 |
| 1997 | UNITED ST6 | MFGR#1421 | 98608466 |
| 1997 | UNITED ST6 | MFGR#143 | 23938737 |
| 1997 | UNITED ST6 | MFGR#1432 | 104846191 |
| 1997 | UNITED ST6 | MFGR#1435 | 185809031 |
| 1997 | UNITED ST6 | MFGR#1436 | 82920407 |
| 1997 | UNITED ST6 | MFGR#1438 | 137524730 |
| 1997 | UNITED ST6 | MFGR#146 | 28124052 |
| 1997 | UNITED ST7 | MFGR#141 | 65266383 |
| 1997 | UNITED ST7 | MFGR#1411 | 78295166 |
| 1997 | UNITED ST7 | MFGR#1413 | 37554700 |
| 1997 | UNITED ST7 | MFGR#1414 | 20428356 |
| 1997 | UNITED ST7 | MFGR#1416 | 92381468 |
| 1997 | UNITED ST7 | MFGR#1418 | 105276410 |
| 1997 | UNITED ST7 | MFGR#1419 | 116086880 |
| 1997 | UNITED ST7 | MFGR#1420 | 62010492 |
| 1997 | UNITED ST7 | MFGR#1428 | 50904528 |
| 1997 | UNITED ST7 | MFGR#1430 | 103558679 |
| 1997 | UNITED ST7 | MFGR#1431 | 38342548 |
| 1997 | UNITED ST7 | MFGR#1436 | 59859992 |
| 1997 | UNITED ST7 | MFGR#1437 | 90701341 |
| 1997 | UNITED ST7 | MFGR#147 | 133840269 |
| 1997 | UNITED ST7 | MFGR#148 | 175852097 |
| 1997 | UNITED ST9 | MFGR#1411 | 62786695 |
| 1997 | UNITED ST9 | MFGR#1416 | 25354497 |
| 1997 | UNITED ST9 | MFGR#1417 | 47367797 |
| 1997 | UNITED ST9 | MFGR#1418 | 27220077 |
| 1997 | UNITED ST9 | MFGR#142 | 41015203 |
| 1997 | UNITED ST9 | MFGR#1423 | 41473506 |
| 1997 | UNITED ST9 | MFGR#1424 | 10735092 |
| 1997 | UNITED ST9 | MFGR#1425 | 27926087 |
| 1997 | UNITED ST9 | MFGR#1426 | 136645966 |
| 1997 | UNITED ST9 | MFGR#1430 | 41283531 |
| 1997 | UNITED ST9 | MFGR#1433 | 497505 |
| 1997 | UNITED ST9 | MFGR#1434 | 101147110 |
| 1997 | UNITED ST9 | MFGR#1436 | 30923170 |
| 1997 | UNITED ST9 | MFGR#145 | 18049495 |
| 1997 | UNITED ST9 | MFGR#146 | 43726737 |
| 1998 | UNITED ST0 | MFGR#1413 | 131487843 |
| 1998 | UNITED ST0 | MFGR#1426 | 52942692 |
| 1998 | UNITED ST0 | MFGR#146 | 13567224 |
| 1998 | UNITED ST1 | MFGR#1410 | 65992198 |
```

```
| 1998 | UNITED ST1 | MFGR#1416 | 115552383 |
| 1998 | UNITED ST1 | MFGR#1418 | 15646035 |
| 1998 | UNITED ST1 | MFGR#1419 | 129708776 |
| 1998 | UNITED ST1 | MFGR#1428 | 18176281 |
| 1998 | UNITED ST1 | MFGR#1431 | 17985830 |
| 1998 | UNITED ST1 | MFGR#1436 | 16714417 |
| 1998 | UNITED ST1 | MFGR#145 | 48297153 |
| 1998 | UNITED ST2 | MFGR#1418 | 9240384 |
| 1998 | UNITED ST2 | MFGR#1419 | 40909344 |
| 1998 | UNITED ST2 | MFGR#1420 | 78625306 |
| 1998 | UNITED ST2 | MFGR#1426 | 67161050 |
| 1998 | UNITED ST2 | MFGR#1430 | 19028508 |
| 1998 | UNITED ST2 | MFGR#1434 | 127804385 |
| 1998 | UNITED ST2 | MFGR#1435 | 75092689 |
| 1998 | UNITED ST2 | MFGR#1436 | 54579894 |
| 1998 | UNITED ST2 | MFGR#1440 | 29067722 |
| 1998 | UNITED ST2 | MFGR#148 | 78886426 |
| 1998 | UNITED ST3 | MFGR#141 | 4311846 |
| 1998 | UNITED ST3 | MFGR#1412 | 98979253 |
| 1998 | UNITED ST3 | MFGR#1415 | 102275672 |
| 1998 | UNITED ST3 | MFGR#1416 | 50781431 |
| 1998 | UNITED ST3 | MFGR#1419 | 37451476 |
| 1998 | UNITED ST3 | MFGR#1420 | 24660608 |
| 1998 | UNITED ST3 | MFGR#1422 | 98548762 |
| 1998 | UNITED ST3 | MFGR#1424 | 96601854 |
| 1998 | UNITED ST3 | MFGR#1425 | 74508450 |
| 1998 | UNITED ST3 | MFGR#1426 | 330583054 |
| 1998 | UNITED ST3 | MFGR#1427 | 41352585 |
| 1998 | UNITED ST3 | MFGR#1428 | 61979722 |
| 1998 | UNITED ST3 | MFGR#1429 | 869295 |
| 1998 | UNITED ST3 | MFGR#1432 | 66991135 |
| 1998 | UNITED ST3 | MFGR#146 | 35929398 |
| 1998 | UNITED ST3 | MFGR#147 | 8484972 |
| 1998 | UNITED ST3 | MFGR#149 | 11793257 |
| 1998 | UNITED ST5 | MFGR#1410 | 55951811 |
| 1998 | UNITED ST5 | MFGR#1413 | 13403140 |
| 1998 | UNITED ST5 | MFGR#142 | 24156762 |
| 1998 | UNITED ST5 | MFGR#1422 | 105826683 |
| 1998 | UNITED ST5 | MFGR#1430 | 67851607 |
| 1998 | UNITED ST5 | MFGR#1431 | 84833774 |
| 1998 | UNITED ST5 | MFGR#1434 | 45541810 |
| 1998 | UNITED ST5 | MFGR#1437 | 33353745 |
| 1998 | UNITED ST5 | MFGR#146 | 19891496 |
| 1998 | UNITED ST6 | MFGR#1413 | 135522572 |
```

```
| 1998 | UNITED ST6 | MFGR#1416 | 185707286 |
| 1998 | UNITED ST6 | MFGR#1417 | 80511133 |
| 1998 | UNITED ST6 | MFGR#1419 | 127132766 |
| 1998 | UNITED ST6 | MFGR#142 | 72629474 |
| 1998 | UNITED ST6 | MFGR#1435 | 158543190 |
| 1998 | UNITED ST7 | MFGR#1412 | 56750777 |
| 1998 | UNITED ST7 | MFGR#1424 | 89508621 |
| 1998 | UNITED ST7 | MFGR#1425 | 160377031 |
| 1998 | UNITED ST7 | MFGR#1434 | 20882477 |
| 1998 | UNITED ST7 | MFGR#146 | 100783548 |
| 1998 | UNITED ST7 | MFGR#147 | 61595522 |
| 1998 | UNITED ST9 | MFGR#1412 | 5049765 |
| 1998 | UNITED ST9 | MFGR#142 | 69919113 |
| 1998 | UNITED ST9 | MFGR#1425 | 11003199 |
| 1998 | UNITED ST9 | MFGR#1426 | 103616972 |
| 1998 | UNITED ST9 | MFGR#1435 | 18879758 |
| 1998 | UNITED ST9 | MFGR#1438 | 101903219 |
+------+------------+-----------+-----------+
```

多表查询运行预期结果

```
-- Q1.1
+--------------+
| revenue |
+--------------+
| 218453880421 |
+--------------+

-- Q1.2
+---------+
| revenue |
+---------+
| NULL |
+---------+

-- Q1.3
+-------------+
| revenue |
+-------------+
| 17527575453 |
+-------------+

-- Q2.1
+------------+------+-----------+
| lo_revenue | year | p_brand |
+------------+------+-----------+
| 1135676414 | 1992 | MFGR#121 |
| 1221327580 | 1992 | MFGR#1210 |
| 1101539324 | 1992 | MFGR#1211 |
| 1298411712 | 1992 | MFGR#1212 |
| 1248062482 | 1992 | MFGR#1213 |
| 1340976936 | 1992 | MFGR#1214 |
| 1266304940 | 1992 | MFGR#1215 |
| 1349693562 | 1992 | MFGR#1216 |
| 1350186870 | 1992 | MFGR#1217 |
| 1200404140 | 1992 | MFGR#1218 |
| 1076087188 | 1992 | MFGR#1219 |
| 1310653344 | 1992 | MFGR#122 |
| 1080525764 | 1992 | MFGR#1220 |
| 1112241266 | 1992 | MFGR#1221 |
| 1181525554 | 1992 | MFGR#1222 |
| 1070897302 | 1992 | MFGR#1223 |
| 1407505222 | 1992 | MFGR#1224 |
```

```
| 1141665736 | 1992 | MFGR#1225 |
| 1228123186 | 1992 | MFGR#1226 |
| 1163518776 | 1992 | MFGR#1227 |
| 1289285184 | 1992 | MFGR#1228 |
| 1281716860 | 1992 | MFGR#1229 |
| 1579511670 | 1992 | MFGR#123 |
| 937070174 | 1992 | MFGR#1230 |
| 1184873312 | 1992 | MFGR#1231 |
| 1328550304 | 1992 | MFGR#1232 |
| 1227770200 | 1992 | MFGR#1233 |
| 1334798562 | 1992 | MFGR#1234 |
| 1280580140 | 1992 | MFGR#1235 |
| 1003785122 | 1992 | MFGR#1236 |
| 1182963006 | 1992 | MFGR#1237 |
| 954847540 | 1992 | MFGR#1238 |
| 1276518748 | 1992 | MFGR#1239 |
| 1144708392 | 1992 | MFGR#124 |
| 1480958496 | 1992 | MFGR#1240 |
| 957554190 | 1992 | MFGR#125 |
| 1184349232 | 1992 | MFGR#126 |
| 1412303264 | 1992 | MFGR#127 |
| 1084613292 | 1992 | MFGR#128 |
| 1163974704 | 1992 | MFGR#129 |
| 1646175404 | 1993 | MFGR#121 |
| 1296321412 | 1993 | MFGR#1210 |
| 1269487796 | 1993 | MFGR#1211 |
| 1571278566 | 1993 | MFGR#1212 |
| 1276510058 | 1993 | MFGR#1213 |
| 1233674474 | 1993 | MFGR#1214 |
| 1269375950 | 1993 | MFGR#1215 |
| 1276707800 | 1993 | MFGR#1216 |
| 1326745902 | 1993 | MFGR#1217 |
| 1367971710 | 1993 | MFGR#1218 |
| 1293900066 | 1993 | MFGR#1219 |
| 1245065968 | 1993 | MFGR#122 |
| 1061660254 | 1993 | MFGR#1220 |
| 1086692674 | 1993 | MFGR#1221 |
| 1513842406 | 1993 | MFGR#1222 |
| 1067088700 | 1993 | MFGR#1223 |
| 1831832170 | 1993 | MFGR#1224 |
| 946014762 | 1993 | MFGR#1225 |
| 1478072248 | 1993 | MFGR#1226 |
| 1184357774 | 1993 | MFGR#1227 |
| 1167014116 | 1993 | MFGR#1228 |
```

```
| 1234906982 | 1993 | MFGR#1229 |
| 1275727736 | 1993 | MFGR#123 |
| 1251068620 | 1993 | MFGR#1230 |
| 1160655270 | 1993 | MFGR#1231 |
| 1394746196 | 1993 | MFGR#1232 |
| 1031142832 | 1993 | MFGR#1233 |
| 1303871516 | 1993 | MFGR#1234 |
| 1151558960 | 1993 | MFGR#1235 |
| 1183757334 | 1993 | MFGR#1236 |
| 1219237152 | 1993 | MFGR#1237 |
| 889228020 | 1993 | MFGR#1238 |
| 1190512654 | 1993 | MFGR#1239 |
| 1321172474 | 1993 | MFGR#124 |
| 1577460118 | 1993 | MFGR#1240 |
| 1232449078 | 1993 | MFGR#125 |
| 1234253508 | 1993 | MFGR#126 |
| 1308876648 | 1993 | MFGR#127 |
| 1463314002 | 1993 | MFGR#128 |
| 1096096790 | 1993 | MFGR#129 |
| 1128811296 | 1994 | MFGR#121 |
| 1290809698 | 1994 | MFGR#1210 |
| 1263241270 | 1994 | MFGR#1211 |
| 1136664696 | 1994 | MFGR#1212 |
| 1357571714 | 1994 | MFGR#1213 |
| 1068004660 | 1994 | MFGR#1214 |
| 1308800484 | 1994 | MFGR#1215 |
| 1117292682 | 1994 | MFGR#1216 |
| 1375691282 | 1994 | MFGR#1217 |
| 1093348694 | 1994 | MFGR#1218 |
| 1134545884 | 1994 | MFGR#1219 |
| 1319768124 | 1994 | MFGR#122 |
| 1125164344 | 1994 | MFGR#1220 |
| 1197237994 | 1994 | MFGR#1221 |
| 1202032882 | 1994 | MFGR#1222 |
| 1110268808 | 1994 | MFGR#1223 |
| 1474844604 | 1994 | MFGR#1224 |
| 1141491910 | 1994 | MFGR#1225 |
| 1492604490 | 1994 | MFGR#1226 |
| 1303414962 | 1994 | MFGR#1227 |
| 1147387094 | 1994 | MFGR#1228 |
| 1295836746 | 1994 | MFGR#1229 |
| 1160899184 | 1994 | MFGR#123 |
| 986540824 | 1994 | MFGR#1230 |
| 1207092296 | 1994 | MFGR#1231 |
```

```
| 1439730662 | 1994 | MFGR#1232 |
| 1277964476 | 1994 | MFGR#1233 |
| 1486495354 | 1994 | MFGR#1234 |
| 1197361918 | 1994 | MFGR#1235 |
| 1231452194 | 1994 | MFGR#1236 |
| 1085139630 | 1994 | MFGR#1237 |
| 1147021562 | 1994 | MFGR#1238 |
| 1159711706 | 1994 | MFGR#1239 |
| 1369146644 | 1994 | MFGR#124 |
| 1747471474 | 1994 | MFGR#1240 |
| 1120976608 | 1994 | MFGR#125 |
| 1314073028 | 1994 | MFGR#126 |
| 1245142366 | 1994 | MFGR#127 |
| 1173691328 | 1994 | MFGR#128 |
| 1069083050 | 1994 | MFGR#129 |
| 1412939022 | 1995 | MFGR#121 |
| 1205785606 | 1995 | MFGR#1210 |
| 1290332184 | 1995 | MFGR#1211 |
| 1226578566 | 1995 | MFGR#1212 |
| 1199172958 | 1995 | MFGR#1213 |
| 1125141608 | 1995 | MFGR#1214 |
| 1345057510 | 1995 | MFGR#1215 |
| 1338001944 | 1995 | MFGR#1216 |
| 1450724898 | 1995 | MFGR#1217 |
| 1314053270 | 1995 | MFGR#1218 |
| 1039318006 | 1995 | MFGR#1219 |
| 1449455482 | 1995 | MFGR#122 |
| 1035912262 | 1995 | MFGR#1220 |
| 1271482702 | 1995 | MFGR#1221 |
| 1128736820 | 1995 | MFGR#1222 |
| 1201330298 | 1995 | MFGR#1223 |
| 1525400702 | 1995 | MFGR#1224 |
| 1343339172 | 1995 | MFGR#1225 |
| 1145137496 | 1995 | MFGR#1226 |
| 1060722600 | 1995 | MFGR#1227 |
| 1266714170 | 1995 | MFGR#1228 |
| 1095920488 | 1995 | MFGR#1229 |
| 1321422154 | 1995 | MFGR#123 |
| 1205471716 | 1995 | MFGR#1230 |
| 999704292 | 1995 | MFGR#1231 |
| 1430601506 | 1995 | MFGR#1232 |
| 1114299142 | 1995 | MFGR#1233 |
| 1420046118 | 1995 | MFGR#1234 |
| 1244850478 | 1995 | MFGR#1235 |
```

```
| 1269131002 | 1995 | MFGR#1236 |
| 1145694540 | 1995 | MFGR#1237 |
| 1098637824 | 1995 | MFGR#1238 |
| 1187703424 | 1995 | MFGR#1239 |
| 1170843630 | 1995 | MFGR#124 |
| 1414415776 | 1995 | MFGR#1240 |
| 1076493744 | 1995 | MFGR#125 |
| 1211598042 | 1995 | MFGR#126 |
| 1331956224 | 1995 | MFGR#127 |
| 1293921912 | 1995 | MFGR#128 |
| 1017498802 | 1995 | MFGR#129 |
| 1047758290 | 1996 | MFGR#121 |
| 1287290106 | 1996 | MFGR#1210 |
| 1190130678 | 1996 | MFGR#1211 |
| 1349252880 | 1996 | MFGR#1212 |
| 992594174 | 1996 | MFGR#1213 |
| 1166499010 | 1996 | MFGR#1214 |
| 1404369714 | 1996 | MFGR#1215 |
| 1203618668 | 1996 | MFGR#1216 |
| 1409796774 | 1996 | MFGR#1217 |
| 1057686172 | 1996 | MFGR#1218 |
| 1172492660 | 1996 | MFGR#1219 |
| 1424220984 | 1996 | MFGR#122 |
| 1036888430 | 1996 | MFGR#1220 |
| 998638828 | 1996 | MFGR#1221 |
| 1358938712 | 1996 | MFGR#1222 |
| 1257525508 | 1996 | MFGR#1223 |
| 1449689712 | 1996 | MFGR#1224 |
| 1321241174 | 1996 | MFGR#1225 |
| 1335349458 | 1996 | MFGR#1226 |
| 967676170 | 1996 | MFGR#1227 |
| 1219710782 | 1996 | MFGR#1228 |
| 1317919114 | 1996 | MFGR#1229 |
| 1132435704 | 1996 | MFGR#123 |
| 1057759996 | 1996 | MFGR#1230 |
| 1178962388 | 1996 | MFGR#1231 |
| 1405611792 | 1996 | MFGR#1232 |
| 1327359894 | 1996 | MFGR#1233 |
| 1142298900 | 1996 | MFGR#1234 |
| 957296148 | 1996 | MFGR#1235 |
| 1136498730 | 1996 | MFGR#1236 |
| 1185232334 | 1996 | MFGR#1237 |
| 933352296 | 1996 | MFGR#1238 |
| 1341387438 | 1996 | MFGR#1239 |
```

```
| 1121335438 | 1996 | MFGR#124 |
| 1642335900 | 1996 | MFGR#1240 |
| 953728666 | 1996 | MFGR#125 |
| 1116061768 | 1996 | MFGR#126 |
| 1271747782 | 1996 | MFGR#127 |
| 1102021236 | 1996 | MFGR#128 |
| 1121141260 | 1996 | MFGR#129 |
| 1174026414 | 1997 | MFGR#121 |
| 1232575784 | 1997 | MFGR#1210 |
| 1097177522 | 1997 | MFGR#1211 |
| 1179187784 | 1997 | MFGR#1212 |
| 848613340 | 1997 | MFGR#1213 |
| 1023943820 | 1997 | MFGR#1214 |
| 1263544492 | 1997 | MFGR#1215 |
| 1384270280 | 1997 | MFGR#1216 |
| 1555989914 | 1997 | MFGR#1217 |
| 1414107440 | 1997 | MFGR#1218 |
| 1122339054 | 1997 | MFGR#1219 |
| 1329832490 | 1997 | MFGR#122 |
| 1188932314 | 1997 | MFGR#1220 |
| 1177696342 | 1997 | MFGR#1221 |
| 1057977920 | 1997 | MFGR#1222 |
| 1074196422 | 1997 | MFGR#1223 |
| 1349526332 | 1997 | MFGR#1224 |
| 900804584 | 1997 | MFGR#1225 |
| 1402721444 | 1997 | MFGR#1226 |
| 1012023140 | 1997 | MFGR#1227 |
| 1171157474 | 1997 | MFGR#1228 |
| 1245488032 | 1997 | MFGR#1229 |
| 1293006336 | 1997 | MFGR#123 |
| 1143601882 | 1997 | MFGR#1230 |
| 1005203580 | 1997 | MFGR#1231 |
| 1355849312 | 1997 | MFGR#1232 |
| 1068911952 | 1997 | MFGR#1233 |
| 1429869430 | 1997 | MFGR#1234 |
| 1534302840 | 1997 | MFGR#1235 |
| 1237754358 | 1997 | MFGR#1236 |
| 1279276114 | 1997 | MFGR#1237 |
| 803906838 | 1997 | MFGR#1238 |
| 1221513428 | 1997 | MFGR#1239 |
| 1086496174 | 1997 | MFGR#124 |
| 1350265384 | 1997 | MFGR#1240 |
| 958198730 | 1997 | MFGR#125 |
| 1141393136 | 1997 | MFGR#126 |
```

```
| 1166149184 | 1997 | MFGR#127 |
| 1390266208 | 1997 | MFGR#128 |
| 1311277552 | 1997 | MFGR#129 |
| 689151850 | 1998 | MFGR#121 |
| 834304832 | 1998 | MFGR#1210 |
| 634136336 | 1998 | MFGR#1211 |
| 748683032 | 1998 | MFGR#1212 |
| 665481806 | 1998 | MFGR#1213 |
| 609746004 | 1998 | MFGR#1214 |
| 732202264 | 1998 | MFGR#1215 |
| 758267796 | 1998 | MFGR#1216 |
| 719016994 | 1998 | MFGR#1217 |
| 641246668 | 1998 | MFGR#1218 |
| 692365724 | 1998 | MFGR#1219 |
| 624880054 | 1998 | MFGR#122 |
| 696247922 | 1998 | MFGR#1220 |
| 679690796 | 1998 | MFGR#1221 |
| 710832322 | 1998 | MFGR#1222 |
| 689779644 | 1998 | MFGR#1223 |
| 793813382 | 1998 | MFGR#1224 |
| 580417756 | 1998 | MFGR#1225 |
| 838831414 | 1998 | MFGR#1226 |
| 716932680 | 1998 | MFGR#1227 |
| 503099910 | 1998 | MFGR#1228 |
| 766277720 | 1998 | MFGR#1229 |
| 592661122 | 1998 | MFGR#123 |
| 874362486 | 1998 | MFGR#1230 |
| 797888984 | 1998 | MFGR#1231 |
| 848124910 | 1998 | MFGR#1232 |
| 813934376 | 1998 | MFGR#1233 |
| 857734480 | 1998 | MFGR#1234 |
| 704555562 | 1998 | MFGR#1235 |
| 723654172 | 1998 | MFGR#1236 |
| 683237138 | 1998 | MFGR#1237 |
| 489478462 | 1998 | MFGR#1238 |
| 828303606 | 1998 | MFGR#1239 |
| 660164742 | 1998 | MFGR#124 |
| 830624906 | 1998 | MFGR#1240 |
| 720579248 | 1998 | MFGR#125 |
| 683315160 | 1998 | MFGR#126 |
| 755014122 | 1998 | MFGR#127 |
| 722832994 | 1998 | MFGR#128 |
| 637539146 | 1998 | MFGR#129 |
+------------+------+-----------+
```

```
-- Q2.2
+------------+------+-----------+
| lo_revenue | year | p_brand |
+------------+------+-----------+
| 1419049858 | 1992 | MFGR#2221 |
| 1567692788 | 1992 | MFGR#2222 |
| 1530104004 | 1992 | MFGR#2223 |
| 1302977924 | 1992 | MFGR#2224 |
| 1293057178 | 1992 | MFGR#2225 |
| 1419301096 | 1992 | MFGR#2226 |
| 1491112632 | 1992 | MFGR#2227 |
| 1513803750 | 1992 | MFGR#2228 |
| 1533042206 | 1993 | MFGR#2221 |
| 1382951194 | 1993 | MFGR#2222 |
| 1516441504 | 1993 | MFGR#2223 |
| 1339325414 | 1993 | MFGR#2224 |
| 1547708456 | 1993 | MFGR#2225 |
| 1474175036 | 1993 | MFGR#2226 |
| 1563935532 | 1993 | MFGR#2227 |
| 1361760432 | 1993 | MFGR#2228 |
| 1371555036 | 1994 | MFGR#2221 |
| 1333049614 | 1994 | MFGR#2222 |
| 1467987180 | 1994 | MFGR#2223 |
| 1415738080 | 1994 | MFGR#2224 |
| 1442503934 | 1994 | MFGR#2225 |
| 1644991838 | 1994 | MFGR#2226 |
| 1441674256 | 1994 | MFGR#2227 |
| 1652450700 | 1994 | MFGR#2228 |
| 1550874148 | 1995 | MFGR#2221 |
| 1522709584 | 1995 | MFGR#2222 |
| 1275665150 | 1995 | MFGR#2223 |
| 1179531414 | 1995 | MFGR#2224 |
| 1416580078 | 1995 | MFGR#2225 |
| 1494712766 | 1995 | MFGR#2226 |
| 1605005080 | 1995 | MFGR#2227 |
| 1791873572 | 1995 | MFGR#2228 |
| 1400020016 | 1996 | MFGR#2221 |
| 1554620170 | 1996 | MFGR#2222 |
| 1312190628 | 1996 | MFGR#2223 |
| 1313719834 | 1996 | MFGR#2224 |
| 1531641792 | 1996 | MFGR#2225 |
| 1616355468 | 1996 | MFGR#2226 |
| 1459126606 | 1996 | MFGR#2227 |
```

```
| 1639331748 | 1996 | MFGR#2228 |
| 1454684764 | 1997 | MFGR#2221 |
| 1329067558 | 1997 | MFGR#2222 |
| 1496576784 | 1997 | MFGR#2223 |
| 1260844162 | 1997 | MFGR#2224 |
| 1514782406 | 1997 | MFGR#2225 |
| 1495778514 | 1997 | MFGR#2226 |
| 1457715798 | 1997 | MFGR#2227 |
| 1550625970 | 1997 | MFGR#2228 |
| 670609008 | 1998 | MFGR#2221 |
| 818694274 | 1998 | MFGR#2222 |
| 918219154 | 1998 | MFGR#2223 |
| 826636144 | 1998 | MFGR#2224 |
| 820804190 | 1998 | MFGR#2225 |
| 907030088 | 1998 | MFGR#2226 |
| 781012810 | 1998 | MFGR#2227 |
| 795878206 | 1998 | MFGR#2228 |
+------------+------+-----------+

-- Q2.3
+------------+------+-----------+
| lo_revenue | year | p_brand |
+------------+------+-----------+
| 1452854972 | 1992 | MFGR#2239 |
| 1410477918 | 1993 | MFGR#2239 |
| 1328290268 | 1994 | MFGR#2239 |
| 1427678672 | 1995 | MFGR#2239 |
| 1456985730 | 1996 | MFGR#2239 |
| 1467793064 | 1997 | MFGR#2239 |
| 760511462 | 1998 | MFGR#2239 |
+------------+------+-----------+

-- Q3.1
+-----------+-----------+------+-------------+
| c_nation | s_nation | year | lo_revenue |
+-----------+-----------+------+-------------+
| INDONESIA | INDONESIA | 1992 | 13811397976 |
| CHINA | INDONESIA | 1992 | 13232157738 |
| CHINA | CHINA | 1992 | 12912862954 |
| VIETNAM | INDONESIA | 1992 | 12680363414 |
| VIETNAM | CHINA | 1992 | 12665688780 |
| INDONESIA | CHINA | 1992 | 12621419066 |
| INDIA | INDONESIA | 1992 | 12477614708 |
| JAPAN | INDONESIA | 1992 | 12445131276 |
```

```
| CHINA | INDIA | 1992 | 12379662702 |
| CHINA | JAPAN | 1992 | 12315357786 |
| JAPAN | CHINA | 1992 | 12134201310 |
| INDIA | CHINA | 1992 | 12132923622 |
| VIETNAM | JAPAN | 1992 | 11727572698 |
| JAPAN | INDIA | 1992 | 11605499970 |
| INDONESIA | INDIA | 1992 | 11540406436 |
| VIETNAM | INDIA | 1992 | 11397022802 |
| INDONESIA | JAPAN | 1992 | 11327531220 |
| JAPAN | JAPAN | 1992 | 11296069422 |
| INDIA | JAPAN | 1992 | 10843918562 |
| CHINA | VIETNAM | 1992 | 10824644052 |
| JAPAN | VIETNAM | 1992 | 10803385110 |
| INDIA | INDIA | 1992 | 10722487510 |
| INDONESIA | VIETNAM | 1992 | 10605276744 |
| INDIA | VIETNAM | 1992 | 10490661242 |
| VIETNAM | VIETNAM | 1992 | 10223463556 |
| INDONESIA | INDONESIA | 1993 | 13862726524 |
| INDONESIA | CHINA | 1993 | 13225782498 |
| CHINA | INDONESIA | 1993 | 13163026732 |
| VIETNAM | INDONESIA | 1993 | 13023278704 |
| CHINA | CHINA | 1993 | 12889027574 |
| CHINA | INDIA | 1993 | 12843388242 |
| VIETNAM | CHINA | 1993 | 12827159998 |
| INDIA | INDONESIA | 1993 | 12662117188 |
| JAPAN | CHINA | 1993 | 12584587990 |
| INDIA | CHINA | 1993 | 12418707584 |
| CHINA | JAPAN | 1993 | 12390933768 |
| VIETNAM | INDIA | 1993 | 12322348954 |
| INDONESIA | INDIA | 1993 | 12303328612 |
| INDONESIA | JAPAN | 1993 | 12295210498 |
| JAPAN | INDONESIA | 1993 | 12107892626 |
| INDIA | JAPAN | 1993 | 11990417970 |
| CHINA | VIETNAM | 1993 | 11770046456 |
| VIETNAM | JAPAN | 1993 | 11748533734 |
| INDONESIA | VIETNAM | 1993 | 11680575444 |
| JAPAN | INDIA | 1993 | 11646686314 |
| INDIA | INDIA | 1993 | 11143151598 |
| VIETNAM | VIETNAM | 1993 | 11108322366 |
| JAPAN | JAPAN | 1993 | 10860637166 |
| JAPAN | VIETNAM | 1993 | 10813139306 |
| INDIA | VIETNAM | 1993 | 10467742974 |
| VIETNAM | CHINA | 1994 | 13419766884 |
| CHINA | CHINA | 1994 | 13297885930 |
```

```
| INDONESIA | CHINA | 1994 | 12967201820 |
| CHINA | JAPAN | 1994 | 12698074042 |
| VIETNAM | INDONESIA | 1994 | 12694883862 |
| JAPAN | CHINA | 1994 | 12640018436 |
| INDONESIA | INDONESIA | 1994 | 12630662172 |
| CHINA | INDIA | 1994 | 12595165622 |
| CHINA | INDONESIA | 1994 | 12469575792 |
| VIETNAM | JAPAN | 1994 | 12463946094 |
| INDONESIA | INDIA | 1994 | 12396824490 |
| INDIA | INDONESIA | 1994 | 12336379718 |
| INDONESIA | JAPAN | 1994 | 12282391938 |
| JAPAN | INDONESIA | 1994 | 12026069236 |
| CHINA | VIETNAM | 1994 | 11770637466 |
| INDIA | CHINA | 1994 | 11630045428 |
| VIETNAM | INDIA | 1994 | 11578797382 |
| JAPAN | JAPAN | 1994 | 11507642964 |
| JAPAN | INDIA | 1994 | 11291637744 |
| INDONESIA | VIETNAM | 1994 | 11248692736 |
| INDIA | INDIA | 1994 | 11169873030 |
| VIETNAM | VIETNAM | 1994 | 10836996318 |
| INDIA | JAPAN | 1994 | 10788269948 |
| JAPAN | VIETNAM | 1994 | 10551643274 |
| INDIA | VIETNAM | 1994 | 10502079630 |
| CHINA | INDONESIA | 1995 | 14149078888 |
| INDONESIA | CHINA | 1995 | 13857241240 |
| CHINA | CHINA | 1995 | 13249333224 |
| JAPAN | CHINA | 1995 | 13039778770 |
| VIETNAM | CHINA | 1995 | 12665462536 |
| INDONESIA | INDONESIA | 1995 | 12537062642 |
| VIETNAM | JAPAN | 1995 | 12527914040 |
| CHINA | INDIA | 1995 | 12493312748 |
| VIETNAM | INDIA | 1995 | 12396883914 |
| INDONESIA | INDIA | 1995 | 12347610366 |
| VIETNAM | INDONESIA | 1995 | 12115640296 |
| CHINA | JAPAN | 1995 | 12043708260 |
| INDONESIA | JAPAN | 1995 | 12038187742 |
| INDIA | CHINA | 1995 | 12021065586 |
| INDIA | INDONESIA | 1995 | 11951037194 |
| JAPAN | JAPAN | 1995 | 11904558258 |
| JAPAN | INDONESIA | 1995 | 11894001470 |
| VIETNAM | VIETNAM | 1995 | 11509455214 |
| JAPAN | INDIA | 1995 | 11461486252 |
| INDONESIA | VIETNAM | 1995 | 11149948132 |
| INDIA | INDIA | 1995 | 11131991100 |
```

```
| JAPAN | VIETNAM | 1995 | 11002627550 |
| CHINA | VIETNAM | 1995 | 10979872126 |
| INDIA | JAPAN | 1995 | 10938406854 |
| INDIA | VIETNAM | 1995 | 10414126568 |
| INDONESIA | INDONESIA | 1996 | 13500112566 |
| CHINA | INDONESIA | 1996 | 13314250150 |
| INDONESIA | CHINA | 1996 | 13226878224 |
| CHINA | CHINA | 1996 | 13183395830 |
| VIETNAM | CHINA | 1996 | 12857307780 |
| VIETNAM | INDONESIA | 1996 | 12591253464 |
| JAPAN | INDONESIA | 1996 | 12454895712 |
| INDIA | CHINA | 1996 | 12397135638 |
| INDIA | INDONESIA | 1996 | 12378484116 |
| CHINA | INDIA | 1996 | 12307574730 |
| INDONESIA | INDIA | 1996 | 12277621726 |
| CHINA | JAPAN | 1996 | 12211132648 |
| JAPAN | CHINA | 1996 | 12177971128 |
| INDONESIA | JAPAN | 1996 | 12111276444 |
| VIETNAM | JAPAN | 1996 | 11839994300 |
| VIETNAM | VIETNAM | 1996 | 11721684604 |
| INDIA | JAPAN | 1996 | 11683329610 |
| VIETNAM | INDIA | 1996 | 11614973966 |
| JAPAN | INDIA | 1996 | 11289159232 |
| JAPAN | JAPAN | 1996 | 11132409590 |
| INDIA | INDIA | 1996 | 11064146206 |
| INDONESIA | VIETNAM | 1996 | 10877028774 |
| CHINA | VIETNAM | 1996 | 10869545636 |
| JAPAN | VIETNAM | 1996 | 10668555098 |
| INDIA | VIETNAM | 1996 | 10587783062 |
| CHINA | INDONESIA | 1997 | 13306469392 |
| INDONESIA | CHINA | 1997 | 13154792628 |
| CHINA | CHINA | 1997 | 12927589590 |
| JAPAN | INDONESIA | 1997 | 12858540252 |
| INDONESIA | INDONESIA | 1997 | 12796855642 |
| VIETNAM | INDONESIA | 1997 | 12727166240 |
| CHINA | JAPAN | 1997 | 12569467036 |
| VIETNAM | CHINA | 1997 | 12328437446 |
| INDIA | CHINA | 1997 | 12306564428 |
| CHINA | INDIA | 1997 | 12168567966 |
| INDONESIA | JAPAN | 1997 | 12002855912 |
| INDIA | INDONESIA | 1997 | 11966878600 |
| JAPAN | CHINA | 1997 | 11947699374 |
| CHINA | VIETNAM | 1997 | 11816508352 |
| JAPAN | INDIA | 1997 | 11593843984 |
```

```
| JAPAN | JAPAN | 1997 | 11580900078 |
| INDONESIA | INDIA | 1997 | 11578734210 |
| VIETNAM | INDIA | 1997 | 11460243216 |
| INDIA | INDIA | 1997 | 11386057814 |
| VIETNAM | JAPAN | 1997 | 11378690460 |
| INDONESIA | VIETNAM | 1997 | 11331356264 |
| VIETNAM | VIETNAM | 1997 | 11240502648 |
| INDIA | JAPAN | 1997 | 11175655826 |
| JAPAN | VIETNAM | 1997 | 10499749228 |
| INDIA | VIETNAM | 1997 | 10007249674 |
+-----------+-----------+------+-------------+

-- Q3.2
+------------+------------+------+------------+
| c_city | s_city | year | lo_revenue |
+------------+------------+------+------------+
| UNITED ST4 | UNITED ST1 | 1992 | 204054910 |
| UNITED ST1 | UNITED ST0 | 1992 | 193978982 |
| UNITED ST7 | UNITED ST0 | 1992 | 192156020 |
| UNITED ST9 | UNITED ST0 | 1992 | 189626588 |
| UNITED ST4 | UNITED ST0 | 1992 | 189288484 |
| UNITED ST2 | UNITED ST4 | 1992 | 182361000 |
| UNITED ST5 | UNITED ST0 | 1992 | 180864600 |
| UNITED ST6 | UNITED ST7 | 1992 | 175316534 |
| UNITED ST3 | UNITED ST9 | 1992 | 172284096 |
| UNITED ST6 | UNITED ST5 | 1992 | 171765932 |
| UNITED ST7 | UNITED ST3 | 1992 | 167531332 |
| UNITED ST2 | UNITED ST9 | 1992 | 167411236 |
| UNITED ST4 | UNITED ST6 | 1992 | 163772748 |
| UNITED ST2 | UNITED ST1 | 1992 | 163678330 |
| UNITED ST9 | UNITED ST1 | 1992 | 161590604 |
| UNITED ST6 | UNITED ST3 | 1992 | 157556436 |
| UNITED ST6 | UNITED ST0 | 1992 | 157393912 |
| UNITED ST0 | UNITED ST1 | 1992 | 154534792 |
| UNITED ST0 | UNITED ST0 | 1992 | 151244244 |
| UNITED ST1 | UNITED ST9 | 1992 | 150734118 |
| UNITED ST3 | UNITED ST1 | 1992 | 147274980 |
| UNITED ST2 | UNITED ST0 | 1992 | 144420436 |
| UNITED ST1 | UNITED ST7 | 1992 | 142945946 |
| UNITED ST6 | UNITED ST4 | 1992 | 142173888 |
| UNITED ST4 | UNITED ST4 | 1992 | 140222670 |
| UNITED ST6 | UNITED ST1 | 1992 | 138817376 |
| UNITED ST4 | UNITED ST3 | 1992 | 138003574 |
| UNITED ST5 | UNITED ST7 | 1992 | 136667302 |
```

```
| UNITED ST4 | UNITED ST9 | 1992 | 135675940 |
| UNITED ST7 | UNITED ST6 | 1992 | 131026410 |
| UNITED ST4 | UNITED ST5 | 1992 | 130115744 |
| UNITED ST7 | UNITED ST4 | 1992 | 129801776 |
| UNITED ST1 | UNITED ST1 | 1992 | 129338140 |
| UNITED ST3 | UNITED ST5 | 1992 | 128478096 |
| UNITED ST0 | UNITED ST9 | 1992 | 127959992 |
| UNITED ST3 | UNITED ST4 | 1992 | 126289544 |
| UNITED ST5 | UNITED ST6 | 1992 | 125256186 |
| UNITED ST4 | UNITED ST7 | 1992 | 125058752 |
| UNITED ST3 | UNITED ST0 | 1992 | 124883312 |
| UNITED ST9 | UNITED ST4 | 1992 | 122979026 |
| UNITED ST8 | UNITED ST6 | 1992 | 121080880 |
| UNITED ST7 | UNITED ST9 | 1992 | 120652084 |
| UNITED ST7 | UNITED ST7 | 1992 | 120242772 |
| UNITED ST5 | UNITED ST1 | 1992 | 119890574 |
| UNITED ST5 | UNITED ST4 | 1992 | 115251254 |
| UNITED ST7 | UNITED ST5 | 1992 | 115133604 |
| UNITED ST2 | UNITED ST5 | 1992 | 114042730 |
| UNITED ST9 | UNITED ST7 | 1992 | 113766718 |
| UNITED ST0 | UNITED ST3 | 1992 | 112718634 |
| UNITED ST1 | UNITED ST3 | 1992 | 111454948 |
| UNITED ST5 | UNITED ST3 | 1992 | 107927106 |
| UNITED ST0 | UNITED ST7 | 1992 | 101166818 |
| UNITED ST5 | UNITED ST9 | 1992 | 100382182 |
| UNITED ST7 | UNITED ST1 | 1992 | 100334416 |
| UNITED ST0 | UNITED ST8 | 1992 | 99465280 |
| UNITED ST0 | UNITED ST4 | 1992 | 99353614 |
| UNITED ST9 | UNITED ST3 | 1992 | 95362330 |
| UNITED ST8 | UNITED ST4 | 1992 | 93514038 |
| UNITED ST3 | UNITED ST3 | 1992 | 90174432 |
| UNITED ST8 | UNITED ST0 | 1992 | 88737678 |
| UNITED ST0 | UNITED ST6 | 1992 | 84943612 |
| UNITED ST6 | UNITED ST8 | 1992 | 84927380 |
| UNITED ST8 | UNITED ST7 | 1992 | 83795802 |
| UNITED ST3 | UNITED ST8 | 1992 | 82551528 |
| UNITED ST6 | UNITED ST9 | 1992 | 81183442 |
| UNITED ST0 | UNITED ST5 | 1992 | 80241772 |
| UNITED ST1 | UNITED ST4 | 1992 | 78652692 |
| UNITED ST3 | UNITED ST7 | 1992 | 78057158 |
| UNITED ST3 | UNITED ST6 | 1992 | 77597430 |
| UNITED ST9 | UNITED ST9 | 1992 | 72096686 |
| UNITED ST2 | UNITED ST8 | 1992 | 72092898 |
| UNITED ST2 | UNITED ST3 | 1992 | 71963926 |
```

```
| UNITED ST8 | UNITED ST1 | 1992 | 71361504 |
| UNITED ST1 | UNITED ST6 | 1992 | 70809980 |
| UNITED ST8 | UNITED ST5 | 1992 | 70375220 |
| UNITED ST1 | UNITED ST5 | 1992 | 67942502 |
| UNITED ST5 | UNITED ST8 | 1992 | 67756106 |
| UNITED ST2 | UNITED ST7 | 1992 | 67405558 |
| UNITED ST8 | UNITED ST3 | 1992 | 61898648 |
| UNITED ST8 | UNITED ST8 | 1992 | 58618216 |
| UNITED ST5 | UNITED ST5 | 1992 | 58559136 |
| UNITED ST1 | UNITED ST8 | 1992 | 57131158 |
| UNITED ST9 | UNITED ST5 | 1992 | 56150008 |
| UNITED ST2 | UNITED ST6 | 1992 | 55627478 |
| UNITED ST0 | UNITED ST2 | 1992 | 55437466 |
| UNITED ST2 | UNITED ST2 | 1992 | 51487308 |
| UNITED ST8 | UNITED ST9 | 1992 | 45368942 |
| UNITED ST4 | UNITED ST8 | 1992 | 43856884 |
| UNITED ST9 | UNITED ST8 | 1992 | 42772200 |
| UNITED ST5 | UNITED ST2 | 1992 | 40991634 |
| UNITED ST6 | UNITED ST6 | 1992 | 36274210 |
| UNITED ST9 | UNITED ST6 | 1992 | 31759136 |
| UNITED ST4 | UNITED ST2 | 1992 | 24123690 |
| UNITED ST7 | UNITED ST8 | 1992 | 23791404 |
| UNITED ST6 | UNITED ST2 | 1992 | 23641396 |
| UNITED ST9 | UNITED ST2 | 1992 | 23246354 |
| UNITED ST8 | UNITED ST2 | 1992 | 21943122 |
| UNITED ST1 | UNITED ST2 | 1992 | 15413456 |
| UNITED ST7 | UNITED ST2 | 1992 | 9886408 |
| UNITED ST3 | UNITED ST2 | 1992 | 2194416 |
| UNITED ST0 | UNITED ST9 | 1993 | 219668080 |
| UNITED ST7 | UNITED ST0 | 1993 | 219576048 |
| UNITED ST5 | UNITED ST0 | 1993 | 213645194 |
| UNITED ST0 | UNITED ST0 | 1993 | 213485096 |
| UNITED ST1 | UNITED ST0 | 1993 | 198611904 |
| UNITED ST4 | UNITED ST4 | 1993 | 196300930 |
| UNITED ST3 | UNITED ST4 | 1993 | 184987840 |
| UNITED ST0 | UNITED ST1 | 1993 | 182393186 |
| UNITED ST4 | UNITED ST1 | 1993 | 177042846 |
| UNITED ST8 | UNITED ST0 | 1993 | 176712742 |
| UNITED ST4 | UNITED ST7 | 1993 | 176344396 |
| UNITED ST4 | UNITED ST0 | 1993 | 173836916 |
| UNITED ST6 | UNITED ST3 | 1993 | 166834322 |
| UNITED ST6 | UNITED ST1 | 1993 | 166691878 |
| UNITED ST7 | UNITED ST9 | 1993 | 160621402 |
| UNITED ST3 | UNITED ST1 | 1993 | 156460556 |
```

```
| UNITED ST6 | UNITED ST7 | 1993 | 156394588 |
| UNITED ST5 | UNITED ST9 | 1993 | 152573078 |
| UNITED ST0 | UNITED ST3 | 1993 | 152342566 |
| UNITED ST5 | UNITED ST8 | 1993 | 148718558 |
| UNITED ST9 | UNITED ST1 | 1993 | 148118838 |
| UNITED ST4 | UNITED ST9 | 1993 | 146593918 |
| UNITED ST5 | UNITED ST1 | 1993 | 142909246 |
| UNITED ST6 | UNITED ST4 | 1993 | 139293826 |
| UNITED ST2 | UNITED ST1 | 1993 | 139263402 |
| UNITED ST6 | UNITED ST0 | 1993 | 136495078 |
| UNITED ST7 | UNITED ST7 | 1993 | 136219640 |
| UNITED ST2 | UNITED ST3 | 1993 | 133944876 |
| UNITED ST3 | UNITED ST0 | 1993 | 133253852 |
| UNITED ST9 | UNITED ST7 | 1993 | 133250966 |
| UNITED ST1 | UNITED ST8 | 1993 | 132292396 |
| UNITED ST2 | UNITED ST7 | 1993 | 128370028 |
| UNITED ST5 | UNITED ST4 | 1993 | 126831278 |
| UNITED ST9 | UNITED ST9 | 1993 | 126521526 |
| UNITED ST1 | UNITED ST4 | 1993 | 125768694 |
| UNITED ST7 | UNITED ST4 | 1993 | 123313226 |
| UNITED ST3 | UNITED ST6 | 1993 | 117169616 |
| UNITED ST2 | UNITED ST4 | 1993 | 113300782 |
| UNITED ST3 | UNITED ST5 | 1993 | 111814610 |
| UNITED ST6 | UNITED ST9 | 1993 | 109801884 |
| UNITED ST1 | UNITED ST7 | 1993 | 109702366 |
| UNITED ST3 | UNITED ST9 | 1993 | 109525192 |
| UNITED ST8 | UNITED ST6 | 1993 | 109266124 |
| UNITED ST8 | UNITED ST3 | 1993 | 108099748 |
| UNITED ST5 | UNITED ST7 | 1993 | 105491076 |
| UNITED ST0 | UNITED ST5 | 1993 | 105402104 |
| UNITED ST1 | UNITED ST9 | 1993 | 105029804 |
| UNITED ST8 | UNITED ST5 | 1993 | 104475674 |
| UNITED ST1 | UNITED ST3 | 1993 | 104195892 |
| UNITED ST8 | UNITED ST4 | 1993 | 102838712 |
| UNITED ST0 | UNITED ST6 | 1993 | 100864564 |
| UNITED ST5 | UNITED ST5 | 1993 | 100714378 |
| UNITED ST3 | UNITED ST7 | 1993 | 100270896 |
| UNITED ST0 | UNITED ST4 | 1993 | 98520134 |
| UNITED ST0 | UNITED ST7 | 1993 | 97592720 |
| UNITED ST2 | UNITED ST9 | 1993 | 96377014 |
| UNITED ST1 | UNITED ST1 | 1993 | 95077220 |
| UNITED ST9 | UNITED ST3 | 1993 | 93887294 |
| UNITED ST7 | UNITED ST5 | 1993 | 89527384 |
| UNITED ST1 | UNITED ST6 | 1993 | 89457080 |
```

```
| UNITED ST8 | UNITED ST1 | 1993 | 88830868 |
| UNITED ST7 | UNITED ST8 | 1993 | 87805256 |
| UNITED ST9 | UNITED ST6 | 1993 | 87734320 |
| UNITED ST2 | UNITED ST0 | 1993 | 85690970 |
| UNITED ST3 | UNITED ST8 | 1993 | 84503696 |
| UNITED ST0 | UNITED ST8 | 1993 | 84005364 |
| UNITED ST4 | UNITED ST8 | 1993 | 83315164 |
| UNITED ST1 | UNITED ST5 | 1993 | 81387026 |
| UNITED ST9 | UNITED ST5 | 1993 | 79370538 |
| UNITED ST7 | UNITED ST3 | 1993 | 79047722 |
| UNITED ST8 | UNITED ST8 | 1993 | 77580470 |
| UNITED ST8 | UNITED ST9 | 1993 | 77032722 |
| UNITED ST2 | UNITED ST5 | 1993 | 74813690 |
| UNITED ST9 | UNITED ST8 | 1993 | 74369392 |
| UNITED ST8 | UNITED ST7 | 1993 | 73804436 |
| UNITED ST6 | UNITED ST8 | 1993 | 72913482 |
| UNITED ST7 | UNITED ST1 | 1993 | 68782318 |
| UNITED ST6 | UNITED ST5 | 1993 | 68458164 |
| UNITED ST5 | UNITED ST3 | 1993 | 68063622 |
| UNITED ST2 | UNITED ST8 | 1993 | 66890892 |
| UNITED ST4 | UNITED ST3 | 1993 | 66258824 |
| UNITED ST6 | UNITED ST6 | 1993 | 66101326 |
| UNITED ST9 | UNITED ST0 | 1993 | 65306610 |
| UNITED ST4 | UNITED ST6 | 1993 | 61398510 |
| UNITED ST9 | UNITED ST4 | 1993 | 61289374 |
| UNITED ST4 | UNITED ST5 | 1993 | 58239188 |
| UNITED ST7 | UNITED ST6 | 1993 | 54201004 |
| UNITED ST4 | UNITED ST2 | 1993 | 54025356 |
| UNITED ST2 | UNITED ST6 | 1993 | 52964452 |
| UNITED ST5 | UNITED ST6 | 1993 | 50715358 |
| UNITED ST3 | UNITED ST3 | 1993 | 43554288 |
| UNITED ST3 | UNITED ST2 | 1993 | 43118146 |
| UNITED ST5 | UNITED ST2 | 1993 | 41220484 |
| UNITED ST7 | UNITED ST2 | 1993 | 40438608 |
| UNITED ST6 | UNITED ST2 | 1993 | 37628734 |
| UNITED ST9 | UNITED ST2 | 1993 | 35436780 |
| UNITED ST1 | UNITED ST2 | 1993 | 33689076 |
| UNITED ST0 | UNITED ST2 | 1993 | 30084290 |
| UNITED ST2 | UNITED ST2 | 1993 | 29043990 |
| UNITED ST8 | UNITED ST2 | 1993 | 19968732 |
| UNITED ST8 | UNITED ST0 | 1994 | 198441578 |
| UNITED ST3 | UNITED ST9 | 1994 | 194952370 |
| UNITED ST6 | UNITED ST1 | 1994 | 193874294 |
| UNITED ST6 | UNITED ST9 | 1994 | 189366618 |
```

```
| UNITED ST9 | UNITED ST1 | 1994 | 180881896 |
| UNITED ST0 | UNITED ST9 | 1994 | 179730404 |
| UNITED ST5 | UNITED ST7 | 1994 | 178179922 |
| UNITED ST9 | UNITED ST0 | 1994 | 175341146 |
| UNITED ST3 | UNITED ST1 | 1994 | 171047306 |
| UNITED ST4 | UNITED ST9 | 1994 | 167644786 |
| UNITED ST0 | UNITED ST0 | 1994 | 167053754 |
| UNITED ST7 | UNITED ST0 | 1994 | 164531072 |
| UNITED ST2 | UNITED ST1 | 1994 | 162600178 |
| UNITED ST5 | UNITED ST0 | 1994 | 157296114 |
| UNITED ST4 | UNITED ST7 | 1994 | 153908280 |
| UNITED ST4 | UNITED ST4 | 1994 | 153674762 |
| UNITED ST0 | UNITED ST1 | 1994 | 153226758 |
| UNITED ST1 | UNITED ST3 | 1994 | 151984918 |
| UNITED ST7 | UNITED ST1 | 1994 | 150641598 |
| UNITED ST4 | UNITED ST0 | 1994 | 147438680 |
| UNITED ST5 | UNITED ST1 | 1994 | 147016836 |
| UNITED ST4 | UNITED ST1 | 1994 | 144439114 |
| UNITED ST2 | UNITED ST9 | 1994 | 139342108 |
| UNITED ST6 | UNITED ST5 | 1994 | 132923068 |
| UNITED ST2 | UNITED ST3 | 1994 | 131241520 |
| UNITED ST3 | UNITED ST0 | 1994 | 131045454 |
| UNITED ST5 | UNITED ST3 | 1994 | 130669822 |
| UNITED ST7 | UNITED ST4 | 1994 | 129557430 |
| UNITED ST3 | UNITED ST4 | 1994 | 126824730 |
| UNITED ST8 | UNITED ST4 | 1994 | 124283362 |
| UNITED ST0 | UNITED ST4 | 1994 | 123039488 |
| UNITED ST0 | UNITED ST7 | 1994 | 122961640 |
| UNITED ST0 | UNITED ST6 | 1994 | 122577556 |
| UNITED ST2 | UNITED ST0 | 1994 | 120364306 |
| UNITED ST6 | UNITED ST4 | 1994 | 119659978 |
| UNITED ST4 | UNITED ST5 | 1994 | 118794056 |
| UNITED ST8 | UNITED ST9 | 1994 | 117333812 |
| UNITED ST4 | UNITED ST6 | 1994 | 117266964 |
| UNITED ST5 | UNITED ST5 | 1994 | 112470426 |
| UNITED ST6 | UNITED ST3 | 1994 | 112246476 |
| UNITED ST2 | UNITED ST4 | 1994 | 111358754 |
| UNITED ST8 | UNITED ST3 | 1994 | 110407682 |
| UNITED ST1 | UNITED ST1 | 1994 | 108766348 |
| UNITED ST1 | UNITED ST7 | 1994 | 107706212 |
| UNITED ST6 | UNITED ST0 | 1994 | 107457706 |
| UNITED ST5 | UNITED ST9 | 1994 | 106734662 |
| UNITED ST9 | UNITED ST9 | 1994 | 103961698 |
| UNITED ST5 | UNITED ST4 | 1994 | 103599186 |
```

```
| UNITED ST7 | UNITED ST9 | 1994 | 100288170 |
| UNITED ST7 | UNITED ST7 | 1994 | 92892884 |
| UNITED ST6 | UNITED ST6 | 1994 | 92399444 |
| UNITED ST7 | UNITED ST5 | 1994 | 91790728 |
| UNITED ST3 | UNITED ST3 | 1994 | 91254306 |
| UNITED ST8 | UNITED ST5 | 1994 | 89106112 |
| UNITED ST9 | UNITED ST4 | 1994 | 87821522 |
| UNITED ST1 | UNITED ST0 | 1994 | 86450402 |
| UNITED ST1 | UNITED ST9 | 1994 | 86000074 |
| UNITED ST7 | UNITED ST8 | 1994 | 85552934 |
| UNITED ST0 | UNITED ST5 | 1994 | 83616602 |
| UNITED ST2 | UNITED ST6 | 1994 | 83052210 |
| UNITED ST1 | UNITED ST4 | 1994 | 82763116 |
| UNITED ST3 | UNITED ST7 | 1994 | 81870262 |
| UNITED ST8 | UNITED ST1 | 1994 | 80304192 |
| UNITED ST9 | UNITED ST8 | 1994 | 78557616 |
| UNITED ST5 | UNITED ST6 | 1994 | 77316902 |
| UNITED ST2 | UNITED ST5 | 1994 | 75280634 |
| UNITED ST8 | UNITED ST7 | 1994 | 75201374 |
| UNITED ST9 | UNITED ST5 | 1994 | 74293452 |
| UNITED ST6 | UNITED ST7 | 1994 | 74115616 |
| UNITED ST8 | UNITED ST6 | 1994 | 73553138 |
| UNITED ST3 | UNITED ST6 | 1994 | 72580514 |
| UNITED ST9 | UNITED ST3 | 1994 | 71693000 |
| UNITED ST2 | UNITED ST8 | 1994 | 67535548 |
| UNITED ST0 | UNITED ST8 | 1994 | 63690866 |
| UNITED ST4 | UNITED ST3 | 1994 | 63198866 |
| UNITED ST9 | UNITED ST7 | 1994 | 63172346 |
| UNITED ST1 | UNITED ST6 | 1994 | 62574652 |
| UNITED ST1 | UNITED ST8 | 1994 | 60490306 |
| UNITED ST7 | UNITED ST3 | 1994 | 58849680 |
| UNITED ST9 | UNITED ST6 | 1994 | 58425854 |
| UNITED ST0 | UNITED ST3 | 1994 | 54655658 |
| UNITED ST6 | UNITED ST8 | 1994 | 53185992 |
| UNITED ST3 | UNITED ST5 | 1994 | 52395750 |
| UNITED ST6 | UNITED ST2 | 1994 | 51618000 |
| UNITED ST1 | UNITED ST5 | 1994 | 49878276 |
| UNITED ST7 | UNITED ST6 | 1994 | 49263874 |
| UNITED ST1 | UNITED ST2 | 1994 | 47113172 |
| UNITED ST4 | UNITED ST2 | 1994 | 46071784 |
| UNITED ST2 | UNITED ST7 | 1994 | 44365516 |
| UNITED ST0 | UNITED ST2 | 1994 | 44035908 |
| UNITED ST4 | UNITED ST8 | 1994 | 41370704 |
| UNITED ST7 | UNITED ST2 | 1994 | 39310162 |
```

```
| UNITED ST5 | UNITED ST8 | 1994 | 37863782 |
| UNITED ST2 | UNITED ST2 | 1994 | 36137314 |
| UNITED ST3 | UNITED ST8 | 1994 | 31872102 |
| UNITED ST8 | UNITED ST8 | 1994 | 20046824 |
| UNITED ST3 | UNITED ST2 | 1994 | 19990468 |
| UNITED ST9 | UNITED ST2 | 1994 | 19401978 |
| UNITED ST5 | UNITED ST2 | 1994 | 14325592 |
| UNITED ST8 | UNITED ST2 | 1994 | 7579252 |
| UNITED ST5 | UNITED ST1 | 1995 | 239587338 |
| UNITED ST4 | UNITED ST9 | 1995 | 198980136 |
| UNITED ST7 | UNITED ST0 | 1995 | 196062590 |
| UNITED ST6 | UNITED ST0 | 1995 | 183436942 |
| UNITED ST4 | UNITED ST1 | 1995 | 181757306 |
| UNITED ST0 | UNITED ST1 | 1995 | 181527198 |
| UNITED ST8 | UNITED ST9 | 1995 | 177710178 |
| UNITED ST7 | UNITED ST7 | 1995 | 173143248 |
| UNITED ST3 | UNITED ST0 | 1995 | 168925466 |
| UNITED ST9 | UNITED ST1 | 1995 | 165877934 |
| UNITED ST2 | UNITED ST4 | 1995 | 164864610 |
| UNITED ST1 | UNITED ST0 | 1995 | 163353246 |
| UNITED ST5 | UNITED ST4 | 1995 | 162033522 |
| UNITED ST7 | UNITED ST1 | 1995 | 159928724 |
| UNITED ST5 | UNITED ST3 | 1995 | 156198260 |
| UNITED ST5 | UNITED ST0 | 1995 | 155231492 |
| UNITED ST9 | UNITED ST9 | 1995 | 153031916 |
| UNITED ST7 | UNITED ST9 | 1995 | 150635418 |
| UNITED ST4 | UNITED ST4 | 1995 | 149174142 |
| UNITED ST9 | UNITED ST4 | 1995 | 145051372 |
| UNITED ST1 | UNITED ST9 | 1995 | 144941740 |
| UNITED ST4 | UNITED ST7 | 1995 | 138528814 |
| UNITED ST6 | UNITED ST3 | 1995 | 135026124 |
| UNITED ST2 | UNITED ST3 | 1995 | 130436258 |
| UNITED ST2 | UNITED ST9 | 1995 | 130110356 |
| UNITED ST7 | UNITED ST6 | 1995 | 130041342 |
| UNITED ST3 | UNITED ST1 | 1995 | 129525630 |
| UNITED ST1 | UNITED ST1 | 1995 | 128398664 |
| UNITED ST6 | UNITED ST9 | 1995 | 126914210 |
| UNITED ST0 | UNITED ST9 | 1995 | 126506998 |
| UNITED ST5 | UNITED ST9 | 1995 | 124729794 |
| UNITED ST4 | UNITED ST5 | 1995 | 124163010 |
| UNITED ST1 | UNITED ST7 | 1995 | 123031482 |
| UNITED ST2 | UNITED ST7 | 1995 | 120000416 |
| UNITED ST8 | UNITED ST6 | 1995 | 117980808 |
| UNITED ST1 | UNITED ST4 | 1995 | 115071198 |
```

```
| UNITED ST0 | UNITED ST3 | 1995 | 112721416 |
| UNITED ST8 | UNITED ST0 | 1995 | 110463328 |
| UNITED ST5 | UNITED ST7 | 1995 | 107481518 |
| UNITED ST2 | UNITED ST0 | 1995 | 105121676 |
| UNITED ST3 | UNITED ST7 | 1995 | 103159096 |
| UNITED ST9 | UNITED ST0 | 1995 | 103097242 |
| UNITED ST6 | UNITED ST6 | 1995 | 101909354 |
| UNITED ST5 | UNITED ST5 | 1995 | 100788014 |
| UNITED ST7 | UNITED ST4 | 1995 | 99799090 |
| UNITED ST3 | UNITED ST3 | 1995 | 96316178 |
| UNITED ST6 | UNITED ST4 | 1995 | 95394482 |
| UNITED ST9 | UNITED ST7 | 1995 | 92929178 |
| UNITED ST4 | UNITED ST0 | 1995 | 92285798 |
| UNITED ST1 | UNITED ST3 | 1995 | 91646112 |
| UNITED ST2 | UNITED ST1 | 1995 | 90874680 |
| UNITED ST6 | UNITED ST5 | 1995 | 90856304 |
| UNITED ST8 | UNITED ST5 | 1995 | 89989726 |
| UNITED ST7 | UNITED ST3 | 1995 | 87399468 |
| UNITED ST9 | UNITED ST6 | 1995 | 86964988 |
| UNITED ST2 | UNITED ST5 | 1995 | 86764834 |
| UNITED ST6 | UNITED ST8 | 1995 | 83947840 |
| UNITED ST0 | UNITED ST6 | 1995 | 81437884 |
| UNITED ST3 | UNITED ST5 | 1995 | 80115630 |
| UNITED ST7 | UNITED ST5 | 1995 | 78030586 |
| UNITED ST0 | UNITED ST0 | 1995 | 77969004 |
| UNITED ST6 | UNITED ST1 | 1995 | 76656704 |
| UNITED ST4 | UNITED ST6 | 1995 | 76219048 |
| UNITED ST3 | UNITED ST9 | 1995 | 74729246 |
| UNITED ST4 | UNITED ST3 | 1995 | 74712792 |
| UNITED ST2 | UNITED ST6 | 1995 | 74292576 |
| UNITED ST9 | UNITED ST5 | 1995 | 72019848 |
| UNITED ST1 | UNITED ST8 | 1995 | 69837586 |
| UNITED ST8 | UNITED ST1 | 1995 | 68435560 |
| UNITED ST0 | UNITED ST7 | 1995 | 66790626 |
| UNITED ST1 | UNITED ST5 | 1995 | 63714904 |
| UNITED ST8 | UNITED ST7 | 1995 | 61836404 |
| UNITED ST2 | UNITED ST8 | 1995 | 61008378 |
| UNITED ST3 | UNITED ST4 | 1995 | 60844692 |
| UNITED ST5 | UNITED ST6 | 1995 | 60409474 |
| UNITED ST8 | UNITED ST3 | 1995 | 58699876 |
| UNITED ST0 | UNITED ST4 | 1995 | 58340076 |
| UNITED ST1 | UNITED ST6 | 1995 | 54278806 |
| UNITED ST7 | UNITED ST8 | 1995 | 52888980 |
| UNITED ST6 | UNITED ST7 | 1995 | 47667954 |
```

```
| UNITED ST4 | UNITED ST8 | 1995 | 46106472 |
| UNITED ST4 | UNITED ST2 | 1995 | 45574006 |
| UNITED ST3 | UNITED ST8 | 1995 | 45010478 |
| UNITED ST9 | UNITED ST8 | 1995 | 42585054 |
| UNITED ST8 | UNITED ST4 | 1995 | 38574622 |
| UNITED ST8 | UNITED ST2 | 1995 | 36565980 |
| UNITED ST9 | UNITED ST3 | 1995 | 35078204 |
| UNITED ST3 | UNITED ST6 | 1995 | 33477060 |
| UNITED ST0 | UNITED ST8 | 1995 | 32786498 |
| UNITED ST5 | UNITED ST2 | 1995 | 29902046 |
| UNITED ST2 | UNITED ST2 | 1995 | 26910062 |
| UNITED ST5 | UNITED ST8 | 1995 | 26693864 |
| UNITED ST3 | UNITED ST2 | 1995 | 25773658 |
| UNITED ST9 | UNITED ST2 | 1995 | 25306724 |
| UNITED ST0 | UNITED ST5 | 1995 | 22907418 |
| UNITED ST6 | UNITED ST2 | 1995 | 22727102 |
| UNITED ST8 | UNITED ST8 | 1995 | 22571734 |
| UNITED ST1 | UNITED ST2 | 1995 | 15983352 |
| UNITED ST0 | UNITED ST2 | 1995 | 9552920 |
| UNITED ST7 | UNITED ST2 | 1995 | 7947130 |
| UNITED ST6 | UNITED ST0 | 1996 | 264573526 |
| UNITED ST4 | UNITED ST0 | 1996 | 213795126 |
| UNITED ST5 | UNITED ST0 | 1996 | 209003958 |
| UNITED ST0 | UNITED ST4 | 1996 | 206457498 |
| UNITED ST9 | UNITED ST1 | 1996 | 203967654 |
| UNITED ST1 | UNITED ST0 | 1996 | 189723108 |
| UNITED ST0 | UNITED ST1 | 1996 | 183897554 |
| UNITED ST6 | UNITED ST1 | 1996 | 179411740 |
| UNITED ST2 | UNITED ST1 | 1996 | 176512310 |
| UNITED ST1 | UNITED ST1 | 1996 | 174531696 |
| UNITED ST4 | UNITED ST7 | 1996 | 167355628 |
| UNITED ST6 | UNITED ST3 | 1996 | 164336458 |
| UNITED ST2 | UNITED ST7 | 1996 | 160936954 |
| UNITED ST8 | UNITED ST1 | 1996 | 157943512 |
| UNITED ST7 | UNITED ST4 | 1996 | 155882022 |
| UNITED ST1 | UNITED ST3 | 1996 | 155221810 |
| UNITED ST9 | UNITED ST9 | 1996 | 154603480 |
| UNITED ST0 | UNITED ST9 | 1996 | 151870418 |
| UNITED ST7 | UNITED ST0 | 1996 | 151204890 |
| UNITED ST3 | UNITED ST1 | 1996 | 149493398 |
| UNITED ST7 | UNITED ST7 | 1996 | 148081288 |
| UNITED ST4 | UNITED ST1 | 1996 | 145639734 |
| UNITED ST5 | UNITED ST9 | 1996 | 145228228 |
| UNITED ST1 | UNITED ST9 | 1996 | 139647538 |
```

```
| UNITED ST9 | UNITED ST4 | 1996 | 139233228 |
| UNITED ST6 | UNITED ST4 | 1996 | 138592010 |
| UNITED ST2 | UNITED ST0 | 1996 | 134190244 |
| UNITED ST5 | UNITED ST1 | 1996 | 130692778 |
| UNITED ST6 | UNITED ST9 | 1996 | 126512364 |
| UNITED ST4 | UNITED ST6 | 1996 | 124378656 |
| UNITED ST0 | UNITED ST0 | 1996 | 123057710 |
| UNITED ST8 | UNITED ST9 | 1996 | 120933382 |
| UNITED ST3 | UNITED ST0 | 1996 | 120453680 |
| UNITED ST8 | UNITED ST6 | 1996 | 119493310 |
| UNITED ST2 | UNITED ST3 | 1996 | 119297196 |
| UNITED ST0 | UNITED ST5 | 1996 | 115525790 |
| UNITED ST8 | UNITED ST7 | 1996 | 115047850 |
| UNITED ST2 | UNITED ST4 | 1996 | 114974114 |
| UNITED ST6 | UNITED ST7 | 1996 | 114181238 |
| UNITED ST3 | UNITED ST4 | 1996 | 109676518 |
| UNITED ST4 | UNITED ST9 | 1996 | 108269680 |
| UNITED ST1 | UNITED ST6 | 1996 | 108112732 |
| UNITED ST3 | UNITED ST7 | 1996 | 107974436 |
| UNITED ST2 | UNITED ST9 | 1996 | 106982830 |
| UNITED ST4 | UNITED ST8 | 1996 | 106071324 |
| UNITED ST9 | UNITED ST5 | 1996 | 105651844 |
| UNITED ST7 | UNITED ST3 | 1996 | 104713772 |
| UNITED ST6 | UNITED ST8 | 1996 | 104273568 |
| UNITED ST1 | UNITED ST5 | 1996 | 102379298 |
| UNITED ST8 | UNITED ST4 | 1996 | 102066108 |
| UNITED ST1 | UNITED ST4 | 1996 | 100271094 |
| UNITED ST3 | UNITED ST9 | 1996 | 99224608 |
| UNITED ST9 | UNITED ST0 | 1996 | 99181402 |
| UNITED ST3 | UNITED ST3 | 1996 | 98527592 |
| UNITED ST9 | UNITED ST7 | 1996 | 97597518 |
| UNITED ST7 | UNITED ST1 | 1996 | 97568350 |
| UNITED ST9 | UNITED ST6 | 1996 | 97370126 |
| UNITED ST2 | UNITED ST5 | 1996 | 94057952 |
| UNITED ST9 | UNITED ST3 | 1996 | 94042036 |
| UNITED ST2 | UNITED ST8 | 1996 | 93730226 |
| UNITED ST4 | UNITED ST3 | 1996 | 92921880 |
| UNITED ST6 | UNITED ST5 | 1996 | 92060208 |
| UNITED ST2 | UNITED ST6 | 1996 | 90833298 |
| UNITED ST8 | UNITED ST5 | 1996 | 86960946 |
| UNITED ST5 | UNITED ST5 | 1996 | 86041444 |
| UNITED ST6 | UNITED ST6 | 1996 | 85846064 |
| UNITED ST4 | UNITED ST5 | 1996 | 85616824 |
| UNITED ST3 | UNITED ST6 | 1996 | 83763256 |
```

```
| UNITED ST1 | UNITED ST7 | 1996 | 83443012 |
| UNITED ST5 | UNITED ST7 | 1996 | 81892660 |
| UNITED ST8 | UNITED ST0 | 1996 | 79690854 |
| UNITED ST8 | UNITED ST3 | 1996 | 79071880 |
| UNITED ST1 | UNITED ST8 | 1996 | 78861764 |
| UNITED ST5 | UNITED ST6 | 1996 | 76664088 |
| UNITED ST0 | UNITED ST6 | 1996 | 74464124 |
| UNITED ST7 | UNITED ST6 | 1996 | 73071256 |
| UNITED ST9 | UNITED ST8 | 1996 | 72224602 |
| UNITED ST3 | UNITED ST8 | 1996 | 67849464 |
| UNITED ST3 | UNITED ST5 | 1996 | 67434878 |
| UNITED ST5 | UNITED ST4 | 1996 | 66849718 |
| UNITED ST5 | UNITED ST3 | 1996 | 65839852 |
| UNITED ST4 | UNITED ST4 | 1996 | 65575990 |
| UNITED ST7 | UNITED ST5 | 1996 | 65568448 |
| UNITED ST5 | UNITED ST8 | 1996 | 64831364 |
| UNITED ST0 | UNITED ST7 | 1996 | 62782362 |
| UNITED ST0 | UNITED ST3 | 1996 | 59591330 |
| UNITED ST7 | UNITED ST9 | 1996 | 50056182 |
| UNITED ST7 | UNITED ST8 | 1996 | 48697702 |
| UNITED ST6 | UNITED ST2 | 1996 | 40895694 |
| UNITED ST8 | UNITED ST8 | 1996 | 32681206 |
| UNITED ST0 | UNITED ST8 | 1996 | 30336524 |
| UNITED ST4 | UNITED ST2 | 1996 | 24903734 |
| UNITED ST1 | UNITED ST2 | 1996 | 20165072 |
| UNITED ST5 | UNITED ST2 | 1996 | 17088466 |
| UNITED ST7 | UNITED ST2 | 1996 | 16780940 |
| UNITED ST9 | UNITED ST2 | 1996 | 16216070 |
| UNITED ST8 | UNITED ST2 | 1996 | 14056668 |
| UNITED ST0 | UNITED ST2 | 1996 | 13814398 |
| UNITED ST3 | UNITED ST2 | 1996 | 8623600 |
| UNITED ST5 | UNITED ST0 | 1997 | 242915532 |
| UNITED ST0 | UNITED ST9 | 1997 | 239712536 |
| UNITED ST5 | UNITED ST1 | 1997 | 213800322 |
| UNITED ST9 | UNITED ST9 | 1997 | 212445590 |
| UNITED ST5 | UNITED ST4 | 1997 | 206865854 |
| UNITED ST7 | UNITED ST1 | 1997 | 202653880 |
| UNITED ST5 | UNITED ST9 | 1997 | 194785280 |
| UNITED ST8 | UNITED ST0 | 1997 | 178869690 |
| UNITED ST1 | UNITED ST3 | 1997 | 170351276 |
| UNITED ST4 | UNITED ST1 | 1997 | 169222376 |
| UNITED ST4 | UNITED ST7 | 1997 | 169213992 |
| UNITED ST1 | UNITED ST4 | 1997 | 166185138 |
| UNITED ST0 | UNITED ST1 | 1997 | 160334278 |
```

```
| UNITED ST4 | UNITED ST9 | 1997 | 159395854 |
| UNITED ST1 | UNITED ST0 | 1997 | 155335732 |
| UNITED ST2 | UNITED ST0 | 1997 | 155182940 |
| UNITED ST1 | UNITED ST7 | 1997 | 154091444 |
| UNITED ST2 | UNITED ST7 | 1997 | 152967604 |
| UNITED ST1 | UNITED ST1 | 1997 | 152680888 |
| UNITED ST0 | UNITED ST7 | 1997 | 145154980 |
| UNITED ST4 | UNITED ST0 | 1997 | 139751608 |
| UNITED ST6 | UNITED ST3 | 1997 | 139451012 |
| UNITED ST2 | UNITED ST9 | 1997 | 139087968 |
| UNITED ST7 | UNITED ST0 | 1997 | 138708624 |
| UNITED ST9 | UNITED ST7 | 1997 | 138105260 |
| UNITED ST8 | UNITED ST3 | 1997 | 133836788 |
| UNITED ST0 | UNITED ST0 | 1997 | 132617032 |
| UNITED ST9 | UNITED ST0 | 1997 | 132133582 |
| UNITED ST2 | UNITED ST3 | 1997 | 130858906 |
| UNITED ST2 | UNITED ST1 | 1997 | 130792270 |
| UNITED ST4 | UNITED ST4 | 1997 | 125064692 |
| UNITED ST9 | UNITED ST1 | 1997 | 124836812 |
| UNITED ST3 | UNITED ST7 | 1997 | 122190600 |
| UNITED ST7 | UNITED ST4 | 1997 | 120246988 |
| UNITED ST4 | UNITED ST3 | 1997 | 119268306 |
| UNITED ST3 | UNITED ST4 | 1997 | 116712282 |
| UNITED ST6 | UNITED ST9 | 1997 | 116462526 |
| UNITED ST6 | UNITED ST4 | 1997 | 114430044 |
| UNITED ST2 | UNITED ST4 | 1997 | 114025222 |
| UNITED ST5 | UNITED ST3 | 1997 | 113579864 |
| UNITED ST9 | UNITED ST5 | 1997 | 112183840 |
| UNITED ST6 | UNITED ST0 | 1997 | 111649838 |
| UNITED ST6 | UNITED ST1 | 1997 | 110235418 |
| UNITED ST7 | UNITED ST9 | 1997 | 110079940 |
| UNITED ST5 | UNITED ST7 | 1997 | 109068630 |
| UNITED ST3 | UNITED ST1 | 1997 | 108301366 |
| UNITED ST3 | UNITED ST0 | 1997 | 108100344 |
| UNITED ST3 | UNITED ST9 | 1997 | 102740616 |
| UNITED ST1 | UNITED ST5 | 1997 | 102104220 |
| UNITED ST6 | UNITED ST7 | 1997 | 99591698 |
| UNITED ST5 | UNITED ST6 | 1997 | 98060032 |
| UNITED ST1 | UNITED ST9 | 1997 | 97888222 |
| UNITED ST3 | UNITED ST3 | 1997 | 96770466 |
| UNITED ST0 | UNITED ST5 | 1997 | 95976836 |
| UNITED ST2 | UNITED ST8 | 1997 | 92783818 |
| UNITED ST4 | UNITED ST6 | 1997 | 92473698 |
| UNITED ST9 | UNITED ST3 | 1997 | 92243448 |
```

```
| UNITED ST8 | UNITED ST9 | 1997 | 91705592 |
| UNITED ST7 | UNITED ST8 | 1997 | 90952532 |
| UNITED ST8 | UNITED ST1 | 1997 | 86568278 |
| UNITED ST7 | UNITED ST7 | 1997 | 85133206 |
| UNITED ST0 | UNITED ST4 | 1997 | 82387606 |
| UNITED ST8 | UNITED ST7 | 1997 | 81756858 |
| UNITED ST8 | UNITED ST8 | 1997 | 81498800 |
| UNITED ST2 | UNITED ST5 | 1997 | 81325772 |
| UNITED ST0 | UNITED ST3 | 1997 | 80157016 |
| UNITED ST6 | UNITED ST8 | 1997 | 75976890 |
| UNITED ST9 | UNITED ST6 | 1997 | 75193764 |
| UNITED ST6 | UNITED ST5 | 1997 | 75143576 |
| UNITED ST2 | UNITED ST2 | 1997 | 74068666 |
| UNITED ST7 | UNITED ST5 | 1997 | 73779472 |
| UNITED ST8 | UNITED ST4 | 1997 | 73201168 |
| UNITED ST3 | UNITED ST6 | 1997 | 72151688 |
| UNITED ST7 | UNITED ST3 | 1997 | 70337844 |
| UNITED ST2 | UNITED ST6 | 1997 | 68548934 |
| UNITED ST5 | UNITED ST8 | 1997 | 65821892 |
| UNITED ST3 | UNITED ST5 | 1997 | 65623926 |
| UNITED ST4 | UNITED ST8 | 1997 | 65199472 |
| UNITED ST5 | UNITED ST5 | 1997 | 65137776 |
| UNITED ST4 | UNITED ST5 | 1997 | 63991736 |
| UNITED ST9 | UNITED ST4 | 1997 | 63530956 |
| UNITED ST7 | UNITED ST2 | 1997 | 62819180 |
| UNITED ST9 | UNITED ST8 | 1997 | 62544770 |
| UNITED ST0 | UNITED ST8 | 1997 | 60482740 |
| UNITED ST3 | UNITED ST8 | 1997 | 58204440 |
| UNITED ST7 | UNITED ST6 | 1997 | 55079862 |
| UNITED ST8 | UNITED ST5 | 1997 | 53347486 |
| UNITED ST6 | UNITED ST6 | 1997 | 49966582 |
| UNITED ST0 | UNITED ST2 | 1997 | 47168458 |
| UNITED ST0 | UNITED ST6 | 1997 | 45848092 |
| UNITED ST1 | UNITED ST2 | 1997 | 41198260 |
| UNITED ST8 | UNITED ST6 | 1997 | 40146000 |
| UNITED ST1 | UNITED ST6 | 1997 | 36410652 |
| UNITED ST1 | UNITED ST8 | 1997 | 30750516 |
| UNITED ST6 | UNITED ST2 | 1997 | 29493360 |
| UNITED ST5 | UNITED ST2 | 1997 | 27726876 |
| UNITED ST8 | UNITED ST2 | 1997 | 24107412 |
| UNITED ST3 | UNITED ST2 | 1997 | 15783756 |
| UNITED ST4 | UNITED ST2 | 1997 | 5696314 |
| UNITED ST9 | UNITED ST2 | 1997 | 5323304 |
+------------+------------+------+------------+
```

```
-- Q3.3
+------------+------------+------+------------+
| c_city | s_city | year | lo_revenue |
+------------+------------+------+------------+
| UNITED KI1 | UNITED KI1 | 1992 | 93471990 |
| UNITED KI5 | UNITED KI1 | 1992 | 72554110 |
| UNITED KI5 | UNITED KI5 | 1992 | 50710534 |
| UNITED KI1 | UNITED KI5 | 1992 | 43835692 |
| UNITED KI5 | UNITED KI1 | 1993 | 122035214 |
| UNITED KI1 | UNITED KI1 | 1993 | 91339070 |
| UNITED KI5 | UNITED KI5 | 1993 | 68198784 |
| UNITED KI1 | UNITED KI5 | 1993 | 42888412 |
| UNITED KI5 | UNITED KI1 | 1994 | 72564326 |
| UNITED KI1 | UNITED KI1 | 1994 | 69736882 |
| UNITED KI5 | UNITED KI5 | 1994 | 69014568 |
| UNITED KI1 | UNITED KI5 | 1994 | 42443560 |
| UNITED KI5 | UNITED KI1 | 1995 | 165911792 |
| UNITED KI1 | UNITED KI1 | 1995 | 71762372 |
| UNITED KI5 | UNITED KI5 | 1995 | 41079610 |
| UNITED KI1 | UNITED KI5 | 1995 | 34353020 |
| UNITED KI5 | UNITED KI1 | 1996 | 131534098 |
| UNITED KI1 | UNITED KI1 | 1996 | 119846074 |
| UNITED KI5 | UNITED KI5 | 1996 | 92154684 |
| UNITED KI1 | UNITED KI5 | 1996 | 27400508 |
| UNITED KI1 | UNITED KI1 | 1997 | 140686266 |
| UNITED KI5 | UNITED KI1 | 1997 | 129956718 |
| UNITED KI5 | UNITED KI5 | 1997 | 54664054 |
| UNITED KI1 | UNITED KI5 | 1997 | 32821336 |
+------------+------------+------+------------+

-- Q3.4
+------------+------------+------+------------+
| c_city | s_city | year | lo_revenue |
+------------+------------+------+------------+
| UNITED KI5 | UNITED KI1 | 1997 | 18235692 |
| UNITED KI5 | UNITED KI5 | 1997 | 12407602 |
| UNITED KI1 | UNITED KI5 | 1997 | 3740140 |
+------------+------------+------+------------+

-- Q4.1
+------+---------------+-------------+
| year | c_nation | profit |
+------+---------------+-------------+
```

```
| 1992 | ARGENTINA | 19317928938 |
| 1992 | BRAZIL | 18453966110 |
| 1992 | CANADA | 19286353574 |
| 1992 | PERU | 18821353194 |
| 1992 | UNITED STATES | 19698855306 |
| 1993 | ARGENTINA | 19952665706 |
| 1993 | BRAZIL | 18937598458 |
| 1993 | CANADA | 19794604840 |
| 1993 | PERU | 18618891672 |
| 1993 | UNITED STATES | 20007970172 |
| 1994 | ARGENTINA | 19880610430 |
| 1994 | BRAZIL | 18697303354 |
| 1994 | CANADA | 19165295192 |
| 1994 | PERU | 18590530026 |
| 1994 | UNITED STATES | 19039760850 |
| 1995 | ARGENTINA | 20287682760 |
| 1995 | BRAZIL | 18312154700 |
| 1995 | CANADA | 19125224320 |
| 1995 | PERU | 19556174422 |
| 1995 | UNITED STATES | 18621130488 |
| 1996 | ARGENTINA | 20003855790 |
| 1996 | BRAZIL | 18336970302 |
| 1996 | CANADA | 20123208406 |
| 1996 | PERU | 18710271348 |
| 1996 | UNITED STATES | 19539424348 |
| 1997 | ARGENTINA | 19709120522 |
| 1997 | BRAZIL | 18243142094 |
| 1997 | CANADA | 20194743556 |
| 1997 | PERU | 18631051834 |
| 1997 | UNITED STATES | 21013447758 |
| 1998 | ARGENTINA | 11668480814 |
| 1998 | BRAZIL | 10712796190 |
| 1998 | CANADA | 10846422392 |
| 1998 | PERU | 11452371940 |
| 1998 | UNITED STATES | 12018924038 |
+------+---------------+-------------+

-- Q4.2
+------+---------------+------------+------------+
| year | s_nation | p_category | profit |
+------+---------------+------------+------------+
| 1997 | ARGENTINA | MFGR#11 | 1814143132 |
| 1997 | ARGENTINA | MFGR#12 | 1848231124 |
| 1997 | ARGENTINA | MFGR#13 | 1945723642 |
```

```
| 1997 | ARGENTINA | MFGR#14 | 1950820690 |
| 1997 | ARGENTINA | MFGR#15 | 1877734750 |
| 1997 | ARGENTINA | MFGR#21 | 2029565148 |
| 1997 | ARGENTINA | MFGR#22 | 1746033566 |
| 1997 | ARGENTINA | MFGR#23 | 2060714604 |
| 1997 | ARGENTINA | MFGR#24 | 1786921158 |
| 1997 | ARGENTINA | MFGR#25 | 2012622806 |
| 1997 | BRAZIL | MFGR#11 | 2146438656 |
| 1997 | BRAZIL | MFGR#12 | 1979717666 |
| 1997 | BRAZIL | MFGR#13 | 2256960758 |
| 1997 | BRAZIL | MFGR#14 | 2388513444 |
| 1997 | BRAZIL | MFGR#15 | 2188838248 |
| 1997 | BRAZIL | MFGR#21 | 1820053664 |
| 1997 | BRAZIL | MFGR#22 | 1986284096 |
| 1997 | BRAZIL | MFGR#23 | 2215345748 |
| 1997 | BRAZIL | MFGR#24 | 2116027298 |
| 1997 | BRAZIL | MFGR#25 | 1989467528 |
| 1997 | CANADA | MFGR#11 | 1709450040 |
| 1997 | CANADA | MFGR#12 | 1877436328 |
| 1997 | CANADA | MFGR#13 | 1918531780 |
| 1997 | CANADA | MFGR#14 | 2005624900 |
| 1997 | CANADA | MFGR#15 | 1696366026 |
| 1997 | CANADA | MFGR#21 | 1999610544 |
| 1997 | CANADA | MFGR#22 | 1556839526 |
| 1997 | CANADA | MFGR#23 | 1856719290 |
| 1997 | CANADA | MFGR#24 | 1699790256 |
| 1997 | CANADA | MFGR#25 | 1809175930 |
| 1997 | PERU | MFGR#11 | 2200485754 |
| 1997 | PERU | MFGR#12 | 1988730700 |
| 1997 | PERU | MFGR#13 | 1694972210 |
| 1997 | PERU | MFGR#14 | 1895539366 |
| 1997 | PERU | MFGR#15 | 1998791356 |
| 1997 | PERU | MFGR#21 | 1735846788 |
| 1997 | PERU | MFGR#22 | 1977494918 |
| 1997 | PERU | MFGR#23 | 2133290172 |
| 1997 | PERU | MFGR#24 | 1871331450 |
| 1997 | PERU | MFGR#25 | 1962908258 |
| 1997 | UNITED STATES | MFGR#11 | 2093412096 |
| 1997 | UNITED STATES | MFGR#12 | 1818427418 |
| 1997 | UNITED STATES | MFGR#13 | 2192557812 |
| 1997 | UNITED STATES | MFGR#14 | 1868564222 |
| 1997 | UNITED STATES | MFGR#15 | 1925521686 |
| 1997 | UNITED STATES | MFGR#21 | 2001352948 |
| 1997 | UNITED STATES | MFGR#22 | 2153895230 |
```

```
| 1997 | UNITED STATES | MFGR#23 | 1874576204 |
| 1997 | UNITED STATES | MFGR#24 | 2006772726 |
| 1997 | UNITED STATES | MFGR#25 | 2107332104 |
| 1998 | ARGENTINA | MFGR#11 | 1135224454 |
| 1998 | ARGENTINA | MFGR#12 | 1054050084 |
| 1998 | ARGENTINA | MFGR#13 | 1165583584 |
| 1998 | ARGENTINA | MFGR#14 | 1047452736 |
| 1998 | ARGENTINA | MFGR#15 | 1044156534 |
| 1998 | ARGENTINA | MFGR#21 | 1009425370 |
| 1998 | ARGENTINA | MFGR#22 | 1012123472 |
| 1998 | ARGENTINA | MFGR#23 | 1120959602 |
| 1998 | ARGENTINA | MFGR#24 | 1049158236 |
| 1998 | ARGENTINA | MFGR#25 | 1095680422 |
| 1998 | BRAZIL | MFGR#11 | 1277156976 |
| 1998 | BRAZIL | MFGR#12 | 1292625362 |
| 1998 | BRAZIL | MFGR#13 | 1310323544 |
| 1998 | BRAZIL | MFGR#14 | 1105352340 |
| 1998 | BRAZIL | MFGR#15 | 1327625418 |
| 1998 | BRAZIL | MFGR#21 | 1337644896 |
| 1998 | BRAZIL | MFGR#22 | 1183583836 |
| 1998 | BRAZIL | MFGR#23 | 1381297754 |
| 1998 | BRAZIL | MFGR#24 | 1124724440 |
| 1998 | BRAZIL | MFGR#25 | 1408364752 |
| 1998 | CANADA | MFGR#11 | 1018172250 |
| 1998 | CANADA | MFGR#12 | 976179544 |
| 1998 | CANADA | MFGR#13 | 973066594 |
| 1998 | CANADA | MFGR#14 | 1055674454 |
| 1998 | CANADA | MFGR#15 | 1071738598 |
| 1998 | CANADA | MFGR#21 | 911737302 |
| 1998 | CANADA | MFGR#22 | 1188554616 |
| 1998 | CANADA | MFGR#23 | 1148250140 |
| 1998 | CANADA | MFGR#24 | 1017060848 |
| 1998 | CANADA | MFGR#25 | 1095515984 |
| 1998 | PERU | MFGR#11 | 1135677094 |
| 1998 | PERU | MFGR#12 | 1081089514 |
| 1998 | PERU | MFGR#13 | 1182663766 |
| 1998 | PERU | MFGR#14 | 962670128 |
| 1998 | PERU | MFGR#15 | 1140492276 |
| 1998 | PERU | MFGR#21 | 1067466660 |
| 1998 | PERU | MFGR#22 | 1055581312 |
| 1998 | PERU | MFGR#23 | 1272786442 |
| 1998 | PERU | MFGR#24 | 1178150524 |
| 1998 | PERU | MFGR#25 | 1086502230 |
| 1998 | UNITED STATES | MFGR#11 | 1112552464 |
```

```
| 1998 | UNITED STATES | MFGR#12 | 1224771964 |
| 1998 | UNITED STATES | MFGR#13 | 1244827854 |
| 1998 | UNITED STATES | MFGR#14 | 1110013774 |
| 1998 | UNITED STATES | MFGR#15 | 1050239138 |
| 1998 | UNITED STATES | MFGR#21 | 1126813672 |
| 1998 | UNITED STATES | MFGR#22 | 1160957470 |
| 1998 | UNITED STATES | MFGR#23 | 1312160930 |
| 1998 | UNITED STATES | MFGR#24 | 1076890116 |
| 1998 | UNITED STATES | MFGR#25 | 1178223904 |
+------+---------------+------------+------------+

-- Q4.3
+------+------------+-----------+---------------+
| year | s_city | p_brand | profit |
+------+------------+-----------+---------------+
| 1997 | UNITED ST1 | MFGR#1414 | 10001830.0000 |
| 1997 | UNITED ST1 | MFGR#147 | 13643806.0000 |
| 1997 | UNITED ST9 | MFGR#144 | 4953914.0000 |
| 1998 | UNITED ST0 | MFGR#1424 | 14202690.0000 |
| 1998 | UNITED ST5 | MFGR#1430 | 7742358.0000 |
+------+------------+-----------+---------------+
5 rows in set (0.25 sec)
```

完成 TPCH 测试

TPC Benchmark™H（TPC-H）是决策支持基准。它由一套面向业务的即时查询（ad-hoc）和并发数据修改组成。选择查询和填充数据库的数据具有广泛的行业相关性。该基准测试解释说明了决策支持系统，该系统可检查大量数据，执行高度复杂的查询并为关键业务问题提供答案。TPC-H 是 OLAP 数据库广泛使用的基准测试。

通过阅读本教程，您将学习如何使用 MatrixOne 完成 TPC-H 测试。

开始前准备

确保你已经完成了[单机部署 MatrixOne](#)。

1. 编译 dbgen

默认情况下，TPCH dbgen 实用程序是用来生成测试数据集表格的工具，它根据比例因子 Scale Factor（SF) 的大小确定数据集的大小，并生成一组平面文件（Flat File)，这些文件适合加载到 tpch 模式中。

当使用 `-s 1` 时 `dbgen` 命令会产生 1GB 的完整数据集，当使用 `-s 10` 时会产生大约 10GB 的数据集，以此类推。

```
git clone https://github.com/electrum/tpch-dbgen.git
cd tpch-dbgen
make
```

2. 生成数据

运行 `dbgen`，获得适当的数据库大小因子 (在示例中为 1GB)。

```
./dbgen -s 1
```

生成完整数据集可能需要一段时间。完成后，您可以看到结果文件。

```
total 2150000
-rw-r--r-- 1 deister staff 24346144 13 may 12:05 customer.tbl
-rw-r--r-- 1 deister staff 759863287 13 may 12:05 lineitem.tbl
-rw-r--r-- 1 deister staff 2224 13 may 12:05 nation.tbl
-rw-r--r-- 1 deister staff 171952161 13 may 12:05 orders.tbl
-rw-r--r-- 1 deister staff 24135125 13 may 12:05 part.tbl
-rw-r--r-- 1 deister staff 118984616 13 may 12:05 partsupp.tbl
-rw-r--r-- 1 deister staff 389 13 may 12:05 region.tbl
-rw-r--r-- 1 deister staff 1409184 13 may 12:05 supplier.tbl
```

我们同时也准备了 1GB 的数据集供您下载。您可以在以下链接中直接获取数据文件:

```
https://community-shared-data-1308875761.cos.ap-
beijing.myqcloud.com/tpch/tpch-1g.zip
```

3. 在 MatrixOne 中建表

MatrixOne 暂不支持复合主键和分区，`PARTSUPP` 和 `LINEITEM` 表的创建代码有以下修改：

- 移除了 `PARTSUPP` 和 `LINEITEM` 表的复合主键。
- 移除了 `LINEITEM` 表的 `PARTITION BY KEY()`。

```
drop database if exists TPCH;
create database if not exists TPCH;
use tpch;
CREATE TABLE NATION(
N_NATIONKEY INTEGER NOT NULL,
N_NAME CHAR(25) NOT NULL,
N_REGIONKEY INTEGER NOT NULL,
N_COMMENT VARCHAR(152),
PRIMARY KEY (N_NATIONKEY)
);

CREATE TABLE REGION(
R_REGIONKEY INTEGER NOT NULL,
R_NAME CHAR(25) NOT NULL,
R_COMMENT VARCHAR(152),
PRIMARY KEY (R_REGIONKEY)
);

CREATE TABLE PART(
P_PARTKEY INTEGER NOT NULL,
P_NAME VARCHAR(55) NOT NULL,
P_MFGR CHAR(25) NOT NULL,
P_BRAND CHAR(10) NOT NULL,
P_TYPE VARCHAR(25) NOT NULL,
P_SIZE INTEGER NOT NULL,
P_CONTAINER CHAR(10) NOT NULL,
P_RETAILPRICE DECIMAL(15,2) NOT NULL,
P_COMMENT VARCHAR(23) NOT NULL,
PRIMARY KEY (P_PARTKEY)
);

CREATE TABLE SUPPLIER(
S_SUPPKEY INTEGER NOT NULL,
S_NAME CHAR(25) NOT NULL,
S_ADDRESS VARCHAR(40) NOT NULL,
S_NATIONKEY INTEGER NOT NULL,
S_PHONE CHAR(15) NOT NULL,
S_ACCTBAL DECIMAL(15,2) NOT NULL,
S_COMMENT VARCHAR(101) NOT NULL,
PRIMARY KEY (S_SUPPKEY)
);

CREATE TABLE PARTSUPP(
PS_PARTKEY INTEGER NOT NULL,
```

```
PS_SUPPKEY INTEGER NOT NULL,
PS_AVAILQTY INTEGER NOT NULL,
PS_SUPPLYCOST DECIMAL(15,2) NOT NULL,
PS_COMMENT VARCHAR(199) NOT NULL
);

CREATE TABLE CUSTOMER(
C_CUSTKEY INTEGER NOT NULL,
C_NAME VARCHAR(25) NOT NULL,
C_ADDRESS VARCHAR(40) NOT NULL,
C_NATIONKEY INTEGER NOT NULL,
C_PHONE CHAR(15) NOT NULL,
C_ACCTBAL DECIMAL(15,2) NOT NULL,
C_MKTSEGMENT CHAR(10) NOT NULL,
C_COMMENT VARCHAR(117) NOT NULL,
PRIMARY KEY (C_CUSTKEY)
);

CREATE TABLE ORDERS(
O_ORDERKEY BIGINT NOT NULL,
O_CUSTKEY INTEGER NOT NULL,
O_ORDERSTATUS CHAR(1) NOT NULL,
O_TOTALPRICE DECIMAL(15,2) NOT NULL,
O_ORDERDATE DATE NOT NULL,
O_ORDERPRIORITY CHAR(15) NOT NULL,
O_CLERK CHAR(15) NOT NULL,
O_SHIPPRIORITY INTEGER NOT NULL,
O_COMMENT VARCHAR(79) NOT NULL,
PRIMARY KEY (O_ORDERKEY)
);

CREATE TABLE LINEITEM(
L_ORDERKEY BIGINT NOT NULL,
L_PARTKEY INTEGER NOT NULL,
L_SUPPKEY INTEGER NOT NULL,
L_LINENUMBER INTEGER NOT NULL,
L_QUANTITY DECIMAL(15,2) NOT NULL,
L_EXTENDEDPRICE DECIMAL(15,2) NOT NULL,
L_DISCOUNT DECIMAL(15,2) NOT NULL,
L_TAX DECIMAL(15,2) NOT NULL,
L_RETURNFLAG VARCHAR(1) NOT NULL,
L_LINESTATUS VARCHAR(1) NOT NULL,
L_SHIPDATE DATE NOT NULL,
L_COMMITDATE DATE NOT NULL,
```

```
L_RECEIPTDATE DATE NOT NULL,
L_SHIPINSTRUCT CHAR(25) NOT NULL,
L_SHIPMODE CHAR(10) NOT NULL,
L_COMMENT VARCHAR(44) NOT NULL
);
```

4. 导入数据

在 MatrixOne 中使用以下命令将数据加载到相关的表中。

```
load data infile '/YOUR_TPCH_DATA_PATH/nation.tbl' into table NATION
FIELDS TERMINATED BY '|' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY
'\n';

load data infile '/YOUR_TPCH_DATA_PATH/region.tbl' into table REGION
FIELDS TERMINATED BY '|' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY
'\n';

load data infile '/YOUR_TPCH_DATA_PATH/part.tbl' into table PART FIELDS
TERMINATED BY '|' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\n';

load data infile '/YOUR_TPCH_DATA_PATH/supplier.tbl' into table SUPPLIER
FIELDS TERMINATED BY '|' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY
'\n';

load data infile '/YOUR_TPCH_DATA_PATH/partsupp.tbl' into table PARTSUPP
FIELDS TERMINATED BY '|' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY
'\n';

load data infile '/YOUR_TPCH_DATA_PATH/orders.tbl' into table ORDERS
FIELDS TERMINATED BY '|' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY
'\n';

load data infile '/YOUR_TPCH_DATA_PATH/customer.tbl' into table CUSTOMER
FIELDS TERMINATED BY '|' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY
'\n';

load data infile '/YOUR_TPCH_DATA_PATH/lineitem.tbl' into table LINEITEM
FIELDS TERMINATED BY '|' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY
'\n';
```

加载完成后，可以使用创建的表查询 MatrixOne 中的数据。

5. 运行 TPCH 测试命令

```
-- Q1
select
l_returnflag,
l_linestatus,
sum(l_quantity) as sum_qty,
sum(l_extendedprice) as sum_base_price,
sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
avg(l_quantity) as avg_qty,
avg(l_extendedprice) as avg_price,
avg(l_discount) as avg_disc,
count(*) as count_order
from
lineitem
where
l_shipdate <= date '1998-12-01' - interval '112' day
group by
l_returnflag,
l_linestatus
order by
l_returnflag,
l_linestatus
;

-- Q2
select
s_acctbal,
s_name,
n_name,
p_partkey,
p_mfgr,
s_address,
s_phone,
s_comment
from
part,
supplier,
partsupp,
nation,
region
where
p_partkey = ps_partkey
```

```
and s_suppkey = ps_suppkey
and p_size = 48
and p_type like '%TIN'
and s_nationkey = n_nationkey
and n_regionkey = r_regionkey
and r_name = 'MIDDLE EAST'
and ps_supplycost = (
select
min(ps_supplycost)
from
partsupp,
supplier,
nation,
region
where
p_partkey = ps_partkey
and s_suppkey = ps_suppkey
and s_nationkey = n_nationkey
and n_regionkey = r_regionkey
and r_name = 'MIDDLE EAST'
)
order by
s_acctbal desc,
n_name,
s_name,
p_partkey
limit 100
;


-- Q3
select
l_orderkey,
sum(l_extendedprice * (1 - l_discount)) as revenue,
o_orderdate,
o_shippriority
from
customer,
orders,
lineitem
where
c_mktsegment = 'HOUSEHOLD'
and c_custkey = o_custkey
and l_orderkey = o_orderkey
```

```
and o_orderdate < date '1995-03-29'
and l_shipdate > date '1995-03-29'
group by
l_orderkey,
o_orderdate,
o_shippriority
order by
revenue desc,
o_orderdate
limit 10
;

-- Q4
select
o_orderpriority,
count(*) as order_count
from
orders
where
o_orderdate >= date '1997-07-01'
and o_orderdate < date '1997-07-01' + interval '3' month
and exists (
select
*
from
lineitem
where
l_orderkey = o_orderkey
and l_commitdate < l_receiptdate
)
group by
o_orderpriority
order by
o_orderpriority
;


-- Q5
select
n_name,
sum(l_extendedprice * (1 - l_discount)) as revenue
from
customer,
orders,
```

```
lineitem,
supplier,
nation,
region
where
c_custkey = o_custkey
and l_orderkey = o_orderkey
and l_suppkey = s_suppkey
and c_nationkey = s_nationkey
and s_nationkey = n_nationkey
and n_regionkey = r_regionkey
and r_name = 'AMERICA'
and o_orderdate >= date '1994-01-01'
and o_orderdate < date '1994-01-01' + interval '1' year
group by
n_name
order by
revenue desc
;


-- Q6
select
sum(l_extendedprice * l_discount) as revenue
from
lineitem
where
l_shipdate >= date '1994-01-01'
and l_shipdate < date '1994-01-01' + interval '1' year
and l_discount between 0.03 - 0.01 and 0.03 + 0.01
and l_quantity < 24;

-- Q7
select
supp_nation,
cust_nation,
l_year,
sum(volume) as revenue
from
(
select
n1.n_name as supp_nation,
n2.n_name as cust_nation,
extract(year from l_shipdate) as l_year,
```

```
l_extendedprice * (1 - l_discount) as volume
from
supplier,
lineitem,
orders,
customer,
nation n1,
nation n2
where
s_suppkey = l_suppkey
and o_orderkey = l_orderkey
and c_custkey = o_custkey
and s_nationkey = n1.n_nationkey
and c_nationkey = n2.n_nationkey
and (
(n1.n_name = 'FRANCE' and n2.n_name = 'ARGENTINA')
or (n1.n_name = 'ARGENTINA' and n2.n_name = 'FRANCE')
)
and l_shipdate between date '1995-01-01' and date '1996-12-31'
) as shipping
group by
supp_nation,
cust_nation,
l_year
order by
supp_nation,
cust_nation,
l_year
;

-- Q8
select
o_year,
(sum(case
when nation = 'ARGENTINA' then volume
else 0
end) / sum(volume)) as mkt_share
from
(
select
extract(year from o_orderdate) as o_year,
l_extendedprice * (1 - l_discount) as volume,
n2.n_name as nation
from
```

```
part,
supplier,
lineitem,
orders,
customer,
nation n1,
nation n2,
region
where
p_partkey = l_partkey
and s_suppkey = l_suppkey
and l_orderkey = o_orderkey
and o_custkey = c_custkey
and c_nationkey = n1.n_nationkey
and n1.n_regionkey = r_regionkey
and r_name = 'AMERICA'
and s_nationkey = n2.n_nationkey
and o_orderdate between date '1995-01-01' and date '1996-12-31'
and p_type = 'ECONOMY BURNISHED TIN'
) as all_nations
group by
o_year
order by
o_year
;

-- Q9
select
nation,
o_year,
sum(amount) as sum_profit
from
(
select
n_name as nation,
extract(year from o_orderdate) as o_year,
l_extendedprice * (1 - l_discount) - ps_supplycost * l_quantity as amount
from
part,
supplier,
lineitem,
partsupp,
orders,
nation
```

```
where
s_suppkey = l_suppkey
and ps_suppkey = l_suppkey
and ps_partkey = l_partkey
and p_partkey = l_partkey
and o_orderkey = l_orderkey
and s_nationkey = n_nationkey
and p_name like '%pink%'
) as profit
group by
nation,
o_year
order by
nation,
o_year desc
;


-- Q10
select
c_custkey,
c_name,
sum(l_extendedprice * (1 - l_discount)) as revenue,
c_acctbal,
n_name,
c_address,
c_phone,
c_comment
from
customer,
orders,
lineitem,
nation
where
c_custkey = o_custkey
and l_orderkey = o_orderkey
and o_orderdate >= date '1993-03-01'
and o_orderdate < date '1993-03-01' + interval '3' month
and l_returnflag = 'R'
and c_nationkey = n_nationkey
group by
c_custkey,
c_name,
c_acctbal,
```

```
c_phone,
n_name,
c_address,
c_comment
order by
revenue desc
limit 20
;


-- Q11
select
ps_partkey,
sum(ps_supplycost * ps_availqty) as value
from
partsupp,
supplier,
nation
where
ps_suppkey = s_suppkey
and s_nationkey = n_nationkey
and n_name = 'JAPAN'
group by
ps_partkey having
sum(ps_supplycost * ps_availqty) > (
select
sum(ps_supplycost * ps_availqty) * 0.0001000000
from
partsupp,
supplier,nation
where
ps_suppkey = s_suppkey
and s_nationkey = n_nationkey
and n_name = 'JAPAN'
)
order by
value desc
;

-- Q12
select
l_shipmode,
sum(case
when o_orderpriority = '1-URGENT'
```

```
or o_orderpriority = '2-HIGH'
then 1
else 0
end) as high_line_count,
sum(case
when o_orderpriority <> '1-URGENT'
and o_orderpriority <> '2-HIGH'
then 1
else 0
end) as low_line_count
from
orders,
lineitem
where
o_orderkey = l_orderkey
and l_shipmode in ('FOB', 'TRUCK')
and l_commitdate < l_receiptdate
and l_shipdate < l_commitdate
and l_receiptdate >= date '1996-01-01'
and l_receiptdate < date '1996-01-01' + interval '1' year
group by
l_shipmode
order by
l_shipmode
;

-- Q13
select
c_count,
count(*) as custdist
from
(
select
c_custkey,
count(o_orderkey)
from
customer left outer join orders on
c_custkey = o_custkey
and o_comment not like '%pending%accounts%'
group by
c_custkey
) as c_orders (c_custkey, c_count)
group by
c_count
```

```
order by
custdist desc,
c_count desc
;

-- Q14
select
100.00 * sum(case
when p_type like 'PROMO%'
then l_extendedprice * (1 - l_discount)
else 0
end) / sum(l_extendedprice * (1 - l_discount)) as promo_revenue
from
lineitem,
part
where
l_partkey = p_partkey
and l_shipdate >= date '1996-04-01'
and l_shipdate < date '1996-04-01' + interval '1' month;

-- Q15
with q15_revenue0 as (
select
l_suppkey as supplier_no,
sum(l_extendedprice * (1 - l_discount)) as total_revenue
from
lineitem
where
l_shipdate >= date '1995-12-01'
and l_shipdate < date '1995-12-01' + interval '3' month
group by
l_suppkey
)
select
s_suppkey,
s_name,
s_address,
s_phone,
total_revenue
from
supplier,
q15_revenue0
where
s_suppkey = supplier_no
```

```
and total_revenue = (
select
max(total_revenue)
from
q15_revenue0
)
order by
s_suppkey
;

-- Q16
select
p_brand,
p_type,
p_size,
count(distinct ps_suppkey) as supplier_cnt
from
partsupp,
part
where
p_partkey = ps_partkey
and p_brand <> 'Brand#35'
and p_type not like 'ECONOMY BURNISHED%'
and p_size in (14, 7, 21, 24, 35, 33, 2, 20)
and ps_suppkey not in (
select
s_suppkey
from
supplier
where
s_comment like '%Customer%Complaints%'
)
group by
p_brand,
p_type,
p_size
order by
supplier_cnt desc,
p_brand,
p_type,
p_size
;

-- Q17
```

```
select
sum(l_extendedprice) / 7.0 as avg_yearly
from
lineitem,
part
where
p_partkey = l_partkey
and p_brand = 'Brand#54'
and p_container = 'LG BAG'
and l_quantity < (
select
0.2 * avg(l_quantity)
from
lineitem
where
l_partkey = p_partkey
);

-- Q18
select
c_name,
c_custkey,
o_orderkey,
o_orderdate,
o_totalprice,
sum(l_quantity)
from
customer,
orders,
lineitem
where
o_orderkey in (
select
l_orderkey
from
lineitem
group by
l_orderkey having
sum(l_quantity) > 314
)
and c_custkey = o_custkey
and o_orderkey = l_orderkey
group by
c_name,
```

```
c_custkey,
o_orderkey,
o_orderdate,
o_totalprice
order by
o_totalprice desc,
o_orderdate
limit 100
;

-- Q19
select
sum(l_extendedprice* (1 - l_discount)) as revenue
from
lineitem,
part
where
(
p_partkey = l_partkey
and p_brand = 'Brand#23'
and p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
and l_quantity >= 5 and l_quantity <= 5 + 10
and p_size between 1 and 5
and l_shipmode in ('AIR', 'AIR REG')
and l_shipinstruct = 'DELIVER IN PERSON'
)
or
(
p_partkey = l_partkey
and p_brand = 'Brand#15'
and p_container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK')
and l_quantity >= 14 and l_quantity <= 14 + 10
and p_size between 1 and 10
and l_shipmode in ('AIR', 'AIR REG')
and l_shipinstruct = 'DELIVER IN PERSON'
)
or
(
p_partkey = l_partkey
and p_brand = 'Brand#44'
and p_container in ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
and l_quantity >= 28 and l_quantity <= 28 + 10
and p_size between 1 and 15
and l_shipmode in ('AIR', 'AIR REG')
```

```
and l_shipinstruct = 'DELIVER IN PERSON'
);

-- Q20
select
s_name,
s_address
from
supplier,
nation
where
s_suppkey in (
select
ps_suppkey
from
partsupp
where
ps_partkey in (
select
p_partkey
from
part
where
p_name like 'lime%'
)
and ps_availqty > (
select
0.5 * sum(l_quantity)
from
lineitem
where
l_partkey = ps_partkey
and l_suppkey = ps_suppkey
and l_shipdate >= date '1993-01-01'
and l_shipdate < date '1993-01-01' + interval '1' year
)
)
and s_nationkey = n_nationkey
and n_name = 'VIETNAM'
order by s_name
;

-- Q21
select
```

```
s_name,
count(*) as numwait
from
supplier,
lineitem l1,
orders,
nation
where
s_suppkey = l1.l_suppkey
and o_orderkey = l1.l_orderkey
and o_orderstatus = 'F'
and l1.l_receiptdate > l1.l_commitdate
and exists (
select
*
from
lineitem l2
where
l2.l_orderkey = l1.l_orderkey
and l2.l_suppkey <> l1.l_suppkey
)
and not exists (
select
*
from
lineitem l3
where
l3.l_orderkey = l1.l_orderkey
and l3.l_suppkey <> l1.l_suppkey
and l3.l_receiptdate > l3.l_commitdate
)
and s_nationkey = n_nationkey
and n_name = 'BRAZIL'
group by
s_name
order by
numwait desc,
s_name
limit 100
;

-- Q22
select
cntrycode,
```

```
count(*) as numcust,
sum(c_acctbal) as totacctbal
from
(
select
substring(c_phone from 1 for 2) as cntrycode,
c_acctbal
from
customer
where
substring(c_phone from 1 for 2) in
('10', '11', '26', '22', '19', '20', '27')
and c_acctbal > (
select
avg(c_acctbal)
from
customer
where
c_acctbal > 0.00
and substring(c_phone from 1 for 2) in
('10', '11', '26', '22', '19', '20', '27')
)
and not exists (
select
*
from
orders
where
o_custkey = c_custkey
)
) as custsale
group by
cntrycode
order by
cntrycode
;
```

6. 运行预期结果

以下为 22 个 TPCH 查询的预期结果。

说明：由于 Q16 的结果段落过长，无法在下文展示，请参阅此连结的完整结果：Q16 运行预期结果

```
Q1
+--------------+--------------+-------------+-----------------+----------
---------+--------------------+---------+-----------+----------+--------
-----+
| l_returnflag | l_linestatus | sum_qty | sum_base_price | sum_disc_price
| sum_charge | avg_qty | avg_price | avg_disc | count_order |
+--------------+--------------+-------------+-----------------+----------
---------+--------------------+---------+-----------+----------+--------
-----+
| A | F | 37734107.00 | 56586554400.73 | 53758257134.8700 |
55909065222.827692 | 25.52 | 38273.12 | 0.04 | 1478493 |
| N | F | 991417.00 | 1487504710.38 | 1413082168.0541 | 1469649223.194375
| 25.51 | 38284.46 | 0.05 | 38854 |
| N | O | 76633518.00 | 114935210409.19 | 109189591897.4720 |
113561024263.013782 | 25.50 | 38248.01 | 0.05 | 3004998 |
| R | F | 37719753.00 | 56568041380.90 | 53741292684.6040 |
55889619119.831932 | 25.50 | 38250.85 | 0.05 | 1478870 |
+--------------+--------------+-------------+-----------------+----------
---------+--------------------+---------+-----------+----------+--------
-----+
4 rows in set

Q2
+-----------+--------------------+--------------+-----------+-----------
----+-----------------------------------------+-----------------+-------
-----------------------------------------------------------------------
--------------------+
| s_acctbal | s_name | n_name | p_partkey | p_mfgr | s_address | s_phone
| s_comment |
+-----------+--------------------+--------------+-----------+-----------
----+-----------------------------------------+-----------------+-------
-----------------------------------------------------------------------
--------------------+
| 9973.93 | Supplier#000004194 | JORDAN | 14193 | Manufacturer#1 |
A8AoX9AK,qhf,CpEF | 23-944-413-2681 | t fluffily. regular requests about
the regular, unusual somas play of the furiously busy |
| 9956.34 | Supplier#000005108 | IRAN | 140079 | Manufacturer#5 |
d3PLCdpPP3uE4GzbbAh4bWmU 7ecOifL9e1mNnzG | 20-842-882-7047 | ronic
accounts. carefully bold accounts sleep beyond |
| 9836.43 | Supplier#000000489 | IRAN | 190488 | Manufacturer#2 |
y9NMoYGxDUPfrB1GwjYhLtCeV7pOt | 20-375-500-2226 | quickly carefully
pending accounts. fina |
| 9825.95 | Supplier#000007554 | IRAQ | 40041 | Manufacturer#5 | Huq0k
qKET hByp3RcMcr | 21-787-637-9651 | ending, final requests thrash
```

```
pending, |
| 9806.52 | Supplier#000005276 | IRAQ | 132762 | Manufacturer#2 |
inh0KOhRfLM7WRhdRNvJJDQx | 21-834-496-7360 | the slyly unusual
theodolites; carefully even accounts use slyly. sl |
| 9796.31 | Supplier#000005847 | IRAQ | 188292 | Manufacturer#1 |
obol3bfh0zWi | 21-530-950-2847 | equests. blithely regular deposits
should have to impress. final platelets integrate fluffily |
| 9775.37 | Supplier#000007245 | IRAQ | 169696 | Manufacturer#5 |
5VOUnvxlJeOJ | 21-663-724-2985 | ic deposits about the slyly bold
requests |
| 9755.43 | Supplier#000002439 | EGYPT | 172438 | Manufacturer#5 |
rXxojWV0VefSK7A0dhF | 14-410-168-5354 | p quickly packages. carefully
pending pinto beans after the |
| 9714.60 | Supplier#000007317 | EGYPT | 29810 | Manufacturer#4 |
nJGsPl2ruoq4Ydtv0fwWG385eOFV6 VUokbCZ | 14-666-351-2592 | ons integrate
furiously? slyly ironic requests sl |
| 9557.33 | Supplier#000007367 | EGYPT | 197366 | Manufacturer#3 |
bTP8DLvaRAB0n | 14-234-934-5255 | ep slyly regular accounts-- regular,
regular excuses dazzle furiously about the reg |
| 9538.15 | Supplier#000000979 | IRAN | 55968 | Manufacturer#1 |
cdvHjrKZR7iDlmSWU2a | 20-151-688-1408 | ckages cajole quietly carefully
regular in |
| 9513.31 | Supplier#000004163 | SAUDI ARABIA | 109142 | Manufacturer#4 |
2VnQypC7pJPJRu6HzYRg7nAvhzckcYAQFbI | 30-544-852-3772 | he regular
requests. blithely final |
| 9450.15 | Supplier#000002067 | EGYPT | 9566 | Manufacturer#3 |
9dO68x0XLMCUDuFk1J6k9hpvLoKx 4qasok4lIb | 14-252-246-5791 | rding to the
regular dolphins. quickly ir |
| 9359.59 | Supplier#000005087 | JORDAN | 55086 | Manufacturer#4 |
q0c6r9wYVQx31IeGBZKfe | 23-335-392-5204 | osits. quickly regular packages
haggle among the quickly silent ins |
| 9343.35 | Supplier#000006337 | IRAN | 173819 | Manufacturer#1 |
ilQgNOdCAysclt3SgODb6LeJ7d4RzYD | 20-950-345-8173 | quickly ironic
packages affix aft |
| 9318.47 | Supplier#000003834 | SAUDI ARABIA | 11332 | Manufacturer#5 |
cxGQnW3nShks59xA5bPoaC bIvcBWUt | 30-823-353-6520 | regular instructions.
express packages run slyly pending |
| 9318.47 | Supplier#000003834 | SAUDI ARABIA | 108813 | Manufacturer#2 |
cxGQnW3nShks59xA5bPoaC bIvcBWUt | 30-823-353-6520 | regular instructions.
express packages run slyly pending |
| 9315.33 | Supplier#000003009 | IRAN | 40504 | Manufacturer#2 |
oTbwfQ,Yfdcf3ysgc60NKtTpSIc | 20-306-556-2450 | the carefully special
requests. express instructions wake |
| 9296.31 | Supplier#000008213 | JORDAN | 163180 | Manufacturer#2 |
```

```
YhdN9ESxYvhJp9ngr12Bbeo4t3zLPD, | 23-197-507-9431 | g to the blithely
regular accounts! deposit |
| 9284.57 | Supplier#000009781 | EGYPT | 4780 | Manufacturer#4 |
49NAd1iDiY4, | 14-410-636-4312 | its. ironic pinto beans are blithely.
express depths use caref |
| 9261.13 | Supplier#000000664 | EGYPT | 125639 | Manufacturer#5 |
ln6wISAnC8Bpj q4V | 14-244-772-4913 | ly special foxes cajole slyly
ironic reque |
| 9260.78 | Supplier#000001949 | IRAN | 86932 | Manufacturer#2 |
W79M2lpYiSY76Ujo6fSRUQiu | 20-531-767-2819 | thinly ironic excuses haggle
slyly. f |
| 9227.16 | Supplier#000009461 | EGYPT | 126948 | Manufacturer#2 | Eweba
0sfaF,l4sAxXGTgEjzsJsNWWIGjYhFkRWV | 14-983-137-4957 | lly bold packages.
carefully express deposits integrate about the unusual accounts. regular,
|
| 9185.89 | Supplier#000007888 | EGYPT | 27887 | Manufacturer#1 |
nq06Y48amPfS8YBuduy1RYu | 14-462-704-3828 | ole slyly-- blithely careful
foxes wake against the furiously ironic accounts. pending ideas |
| 9185.89 | Supplier#000007888 | EGYPT | 190330 | Manufacturer#4 |
nq06Y48amPfS8YBuduy1RYu | 14-462-704-3828 | ole slyly-- blithely careful
foxes wake against the furiously ironic accounts. pending ideas |
| 9132.92 | Supplier#000007878 | IRAN | 92859 | Manufacturer#3 |
aq6T3tUVq1, | 20-861-274-6282 | waters cajole ironic, ironic requests.
furi |
| 9058.94 | Supplier#000002789 | IRAN | 142788 | Manufacturer#4 |
7EkIldjP7M6psSWcJ11tf65GT7ZC7UaiCh | 20-842-716-4307 | s. platelets use
carefully. busily regular accounts cajole fluffily above the slowly final
pinto be |
| 9026.80 | Supplier#000005436 | SAUDI ARABIA | 92926 | Manufacturer#3 |
3HiusYZGYmHItExgX5VfNCdJwkW8W7R | 30-453-280-6340 | . blithely unusual
requests |
| 9007.16 | Supplier#000001747 | EGYPT | 121746 | Manufacturer#3 |
UyDlFjVxanP,ifej7L5jtNaubC | 14-141-360-9722 | maintain bravely across
the busily express pinto beans. sometimes |
| 8846.35 | Supplier#000005446 | EGYPT | 57930 | Manufacturer#2 |
Nusoq0ckv9SwnJZV8Rw2dUqE,V0ylm Bon | 14-468-853-6477 | deposits. ironic,
bold ideas wake. fluffily ironic deposits must have to sleep furiously
pending |
| 8837.21 | Supplier#000007210 | JORDAN | 144695 | Manufacturer#3 |
G7MYkWkkJDVu,rr23aXjQCwNqZ2Vk6 | 23-560-295-1805 | en, express foxes use
across the blithely bold |
| 8800.91 | Supplier#000008182 | EGYPT | 143153 | Manufacturer#5 |
KGMyipBiAF00tSB99DwH | 14-658-338-3635 | thely even excuses integrate
blithel |
```

```
| 8788.46 | Supplier#000003437 | IRAN | 118414 | Manufacturer#4 |
JfgZDO9fsP4ljfzsi,s7431Ld3A7zXtHfrF74 | 20-547-871-1712 | ickly unusual
dependencies. carefully regular dolphins ha |
| 8750.12 | Supplier#000001064 | IRAQ | 31063 | Manufacturer#5 |
QgmUIaEs5KpuW ,oREZV2b4wr3HEC1z4F | 21-440-809-7599 | sly even deposits?
furiously regular pack |
| 8594.80 | Supplier#000007553 | IRAN | 5052 | Manufacturer#4 | wAU2Lui
w9 | 20-663-409-7956 | old, stealthy accounts are blithely. fluffily
final |
| 8594.80 | Supplier#000007553 | IRAN | 195033 | Manufacturer#1 | wAU2Lui
w9 | 20-663-409-7956 | old, stealthy accounts are blithely. fluffily
final |
| 8588.63 | Supplier#000008094 | SAUDI ARABIA | 148093 | Manufacturer#1 |
SNS6FCscBNZBFecA | 30-465-665-6735 | ake quickly blithely ironic
theodolites. quickly ironic ideas boost. furiously iro |
| 8522.70 | Supplier#000004208 | IRAQ | 166659 | Manufacturer#5 |
x3jZYF7ZYN 8a4LY1c1kEsh | 21-468-998-1571 | furiously regular accounts! |
| 8514.86 | Supplier#000006347 | JORDAN | 181310 | Manufacturer#5 |
wwR5j4kdIAwLe33uBwo | 23-340-942-3641 | uests breach blithely ironic
deposi |
| 8473.01 | Supplier#000003912 | IRAQ | 33911 | Manufacturer#3 |
Op7,1zt3MAxs34Qo4O W | 21-474-809-6508 | es. regular, brave instructions
wa |
| 8405.28 | Supplier#000007886 | IRAQ | 192847 | Manufacturer#4 |
sFTj5nzc5EIVmzjXwenFTXD U | 21-735-778-5786 | ven dependencies boost
blithely ironic de |
| 8375.58 | Supplier#000001259 | IRAQ | 38755 | Manufacturer#2 |
32cJBGFFpGEkEjx1sF8JZAy0A72uXL5qU | 21-427-422-4993 | ironic accounts
haggle slyly alongside of the carefully ironic deposit |
| 8351.75 | Supplier#000007495 | IRAQ | 114983 | Manufacturer#4 |
3jQQGvfs,5Aryhn0Z | 21-953-463-7239 | requests. carefully final accounts
after the qui |
| 8230.12 | Supplier#000001058 | SAUDI ARABIA | 68551 | Manufacturer#2 |
fJ8egP,xkLygXGv8bmtc9T1FJ | 30-496-504-3341 | requests haggle? regular,
regular pinto beans integrate fluffily. dependenc |
| 8195.44 | Supplier#000009805 | IRAQ | 4804 | Manufacturer#4 |
dTTmLRYJNat,JbhlQlxwWp HjMR | 21-838-243-3925 | lets. quickly even
theodolites dazzle slyly even a |
| 8175.17 | Supplier#000003172 | IRAN | 55656 | Manufacturer#5 |
8ngbGS7BQoTDmJyMa5WV9XbaM31u5FAayd2vT3 | 20-834-374-7746 | ss deposits
use furiously after the quickly final sentiments. fluffily ruthless ideas
believe regu |
| 8159.13 | Supplier#000007486 | EGYPT | 17485 | Manufacturer#1 |
AjfdzbrrJE1 | 14-970-643-1521 | ld accounts. enticingly furious requests
```

```
cajole. final packages s |
| 8111.40 | Supplier#000007567 | IRAN | 197566 | Manufacturer#1 |
7W4k2qEVoBkRehprGliXRSYVOQEh | 20-377-181-7435 | gular foxes. silent
attainments boost furiousl |
| 8046.55 | Supplier#000001625 | IRAQ | 14121 | Manufacturer#2 |
yKlKMbENR6bfmIu7aCFmbs | 21-769-404-7617 | deposits. ideas boost
blithely. slyly even Tiresias according to the platelets are q |
| 8040.16 | Supplier#000001925 | SAUDI ARABIA | 4424 | Manufacturer#4 |
Cu5Ub AAdXT | 30-969-417-1108 | pending packages across the regular req |
| 8031.68 | Supplier#000002370 | SAUDI ARABIA | 147341 | Manufacturer#5 |
xGQB9xSPqRtCuMZaJavOrFuTY7km | 30-373-388-2352 | dependencies. carefully
express deposits use slyly among the slyly unusual pearls. dogge |
| 8031.42 | Supplier#000008216 | IRAN | 83199 | Manufacturer#2 |
jsqlyr1ViAo | 20-224-305-7298 | to the carefully even excuses haggle
blithely against the pending pinto be |
| 8007.83 | Supplier#000006266 | JORDAN | 81249 | Manufacturer#1 |
XWBf5Jd2V5SOurbn11Tt1 | 23-363-445-7184 | as cajole carefully against the
quickly special ac |
| 7995.78 | Supplier#000006957 | IRAN | 161924 | Manufacturer#1 |
8lvRhU5xtXv | 20-312-173-2216 | ly ironic accounts. stealthily regular
foxes about the blithely ironic requests play blithely abo |
| 7913.40 | Supplier#000003148 | JORDAN | 58137 | Manufacturer#1 |
CpCJWI4PHeiwYuq0 | 23-767-770-9172 | ove the quickly final packages boost
fluffily among the furiously final platelets. carefully s |
| 7910.16 | Supplier#000002102 | IRAQ | 99592 | Manufacturer#2 |
1kuyUn5q6czLOGB60fAVgpv68M2suwchpmp2nK | 21-367-198-9930 | accounts after
the blithely |
| 7893.58 | Supplier#000000918 | SAUDI ARABIA | 13414 | Manufacturer#1 |
e0sB7xAU3,cWF7pzXrpIbATUNydCUZup | 30-303-831-1662 | ependencies wake
carefull |
| 7885.17 | Supplier#000004001 | JORDAN | 38994 | Manufacturer#2 |
3M39sZY1XeQXPDRO | 23-109-632-6806 | efully express packages integrate
across the regular pearls. blithely unusual packages mainta |
| 7880.20 | Supplier#000005352 | JORDAN | 351 | Manufacturer#3 |
PP9gHTn946hXqUF5E7idIPLkhnN | 23-557-756-7951 | egular frays. final
instructions sleep a |
| 7844.31 | Supplier#000006987 | IRAQ | 44482 | Manufacturer#5 |
UH1zBxTNjTminnmHRe h YUT1eR | 21-963-444-7397 | nag quickly carefully
regular requests. ironic theo |
| 7812.27 | Supplier#000006967 | SAUDI ARABIA | 151936 | Manufacturer#4 |
S4i1HfrSM4m3di3R9Cxxp59M1 | 30-193-457-6365 | ely. dependencies cajole
quickly. final warhorses across the furiously ironic foxes integr |
| 7767.63 | Supplier#000004306 | IRAN | 31802 | Manufacturer#2 |
SkZkJZflW5mDg9wL fJ | 20-911-180-1895 | uickly regular ideas. blithely
```

```
express accounts along the carefully sile |
| 7741.42 | Supplier#000000899 | IRAQ | 53383 | Manufacturer#5 |
oLlkiVghtro IwzcwFuzwMCG94rRpux | 21-980-994-3905 | equests wake quickly
special, express accounts. courts promi |
| 7741.42 | Supplier#000000899 | IRAQ | 105878 | Manufacturer#3 |
oLlkiVghtro IwzcwFuzwMCG94rRpux | 21-980-994-3905 | equests wake quickly
special, express accounts. courts promi |
| 7741.10 | Supplier#000001059 | IRAN | 103528 | Manufacturer#4 |
4tBenOMokWbWVRB8i8HwENeO cQjM9 | 20-620-710-8984 | to the carefully
special courts. |
| 7599.20 | Supplier#000006596 | SAUDI ARABIA | 184077 | Manufacturer#2 |
k8qeFxfXKIGYdQ82RXAfCwddSrc | 30-804-947-3851 | ously unusual deposits
boost carefully after the enticing |
| 7598.31 | Supplier#000008857 | IRAQ | 63844 | Manufacturer#4 |
dP2th8vneyOLIUFwNBwqixkFD6 | 21-691-170-4769 | s. quickly ironic frays
detect carefully |
| 7591.79 | Supplier#000009723 | JORDAN | 104702 | Manufacturer#2 |
Q1CkkpDdlLOpCJiV,zIf,Mv86otWhxj7slGc | 23-710-907-3873 | e fluffily even
instructions. packages impress enticingly. |
| 7575.12 | Supplier#000007557 | IRAQ | 77556 | Manufacturer#1 |
udLvpjNvIx9qeRNdjL1ZAO0OZNOBo6h | 21-629-935-9941 | ally special accounts
nod; f |
| 7496.91 | Supplier#000005828 | IRAN | 103317 | Manufacturer#1 |
Xt0EqDCNU6X00sNsIO7nd0ws3H | 20-435-850-8703 | furiously about the
fluffily careful idea |
| 7472.88 | Supplier#000004204 | EGYPT | 14203 | Manufacturer#1 |
0rGZJ6VZXdH | 14-520-667-4690 | y pending pinto beans. even, final
requests sleep care |
| 7472.88 | Supplier#000004204 | EGYPT | 161687 | Manufacturer#3 |
0rGZJ6VZXdH | 14-520-667-4690 | y pending pinto beans. even, final
requests sleep care |
| 7467.63 | Supplier#000003270 | IRAN | 45757 | Manufacturer#2 |
7j4n5FnNEHVJxFhiyz | 20-450-599-9591 | regular, even instructions boost
deposits |
| 7465.41 | Supplier#000008686 | EGYPT | 188685 | Manufacturer#4 |
4Onf4yxuNwHCRIC0y | 14-454-946-4151 | ly final ideas. bravely unusual
deposits doze carefully. expr |
| 7460.80 | Supplier#000008701 | IRAQ | 83684 | Manufacturer#3 |
PLR2QehcW08 | 21-747-984-4244 | ideas use carefully pending, final
deposits. ironic, pe |
| 7447.86 | Supplier#000005877 | JORDAN | 120852 | Manufacturer#2 |
EyqOHClZZMJkq grnOX9 4alZx6P7B2fq | 23-419-288-6451 | lar pinto beans
breach carefully final pinto |
| 7445.03 | Supplier#000009802 | IRAQ | 164769 | Manufacturer#5 | y6wLN
```

```
KiZuTf5HT9Hbm0BELn1GUTD6yl | 21-116-708-2013 | nic requests. pinto beans
across the carefully regular grouches snooze among the final pin |
| 7401.46 | Supplier#000008677 | IRAN | 123652 | Manufacturer#5 |
WNa780JZzivxuGBEsDszqoT1Pj | 20-899-256-5288 | onic instructions along
the furiously ironic accounts haggle fluffily silently un |
| 7393.50 | Supplier#000007056 | IRAQ | 54550 | Manufacturer#1 |
M5cAJQvW9D5zwC7o2qkoe | 21-175-383-4727 | slyly even requests. forges
haggle boldly express requests. furio |
| 7376.11 | Supplier#000003982 | IRAQ | 118959 | Manufacturer#1 |
jx9EloF33Ez | 21-890-236-4160 | s the furiously special warhorses affix
after the car |
| 7264.42 | Supplier#000001565 | IRAQ | 14061 | Manufacturer#4 |
bOwKHdBteMkZoZcxdigk4Tnu07w1gDztmV7hvCw | 21-791-273-8592 | to beans.
express accounts nag around the |
| 7256.46 | Supplier#000009116 | IRAQ | 4115 | Manufacturer#3 |
ULjaQwNbcB XUG9dvbZDHvJVwLo08utswt | 21-241-469-8343 | ending deposits.
slyly ironic dependencies breach. blithely speci |
| 7256.46 | Supplier#000009116 | IRAQ | 99115 | Manufacturer#1 |
ULjaQwNbcB XUG9dvbZDHvJVwLo08utswt | 21-241-469-8343 | ending deposits.
slyly ironic dependencies breach. blithely speci |
| 7256.46 | Supplier#000009116 | IRAQ | 131576 | Manufacturer#4 |
ULjaQwNbcB XUG9dvbZDHvJVwLo08utswt | 21-241-469-8343 | ending deposits.
slyly ironic dependencies breach. blithely speci |
| 7254.81 | Supplier#000005664 | EGYPT | 35663 | Manufacturer#2 |
b8VWuTXRt66wF9bfrgTmNGuxf1PU0x3O9e | 14-214-171-8987 | ts across the
quickly pending pin |
| 7186.63 | Supplier#000006958 | IRAN | 71943 | Manufacturer#4 |
0n9BD,gRzUc3B,PsFcxDBGp4BFf4P | 20-185-413-5590 | against the
instructions. requests are. speci |
| 7166.36 | Supplier#000003541 | EGYPT | 116007 | Manufacturer#1 |
DbwyOxoaMEdhEtIB3y045QrKCi2fQpGRu, | 14-508-763-1850 | ages. carefully
unusual requests across the pending instructions aff |
| 7128.81 | Supplier#000000677 | JORDAN | 50676 | Manufacturer#4 |
8mhrffG7D2WJBSQbOGstQ | 23-290-639-3315 | nder blithely. slyly unusual
theod |
| 7051.73 | Supplier#000003349 | IRAQ | 125812 | Manufacturer#3 |
wtTK9df9kY7mQ5QUM0Xe5bHLMRLgwE | 21-614-525-7451 | ar theodolites cajole
fluffily across the pending requests. slyly final requests a |
| 7023.47 | Supplier#000009543 | SAUDI ARABIA | 47038 | Manufacturer#1 |
VYKinyOBNXRr Hdqn8kOxfTw | 30-785-782-6088 | sts. furiously pending
packages sleep slyly even requests. final excuses print deposits. final
pac |
| 6985.93 | Supplier#000006409 | IRAQ | 131382 | Manufacturer#1 |
eO8JDNM19HrlQMR | 21-627-356-3992 | sts. slyly final deposits around the
```

```
regular accounts are along the furiously final pac |
| 6964.75 | Supplier#000009931 | EGYPT | 57425 | Manufacturer#1 |
peQYiRFk G0xZKfJ | 14-989-166-5782 | deposits according to the sometimes
silent requests wake along the packages-- blithely f |
| 6964.04 | Supplier#000007399 | IRAQ | 77398 | Manufacturer#2 |
zdxjENOGR4QiCFP | 21-859-733-1999 | e blithely after the even requests.
carefully ironic packages use slyly a |
| 6913.81 | Supplier#000002625 | IRAQ | 22624 | Manufacturer#3 |
a4V0rWemgbsT ZMj w7DB8rUbZ4F4lqqW5VKljQF | 21-136-564-3910 | . asymptotes
among the express requests cajole furiously after the ca |
| 6880.18 | Supplier#000006704 | IRAN | 26703 | Manufacturer#4 |
97rxJlAImbO1 sUlChUWoOJ0ZzvQ2NI3KI6VDOwk | 20-588-916-1286 | old accounts
wake quickly. ca |
| 6878.62 | Supplier#000001697 | IRAQ | 146668 | Manufacturer#5 | 37nm
ODTeHy0xWTWegplgdWQqelh | 21-377-544-4864 | ironic theodolites. furiously
regular d |
| 6790.39 | Supplier#000008703 | IRAN | 123678 | Manufacturer#4 |
wMslK1A8SEUTIIdApQ | 20-782-266-2552 | eep blithely regular, pending w |
| 6763.46 | Supplier#000007882 | EGYPT | 137881 | Manufacturer#5 |
JDv8BZiYG0UlZ | 14-111-252-9120 | the silent accounts wake foxes. furious
|
| 6751.81 | Supplier#000003156 | EGYPT | 165607 | Manufacturer#2 |
alRWaW4FTFERMM4vf2rHKIKE | 14-843-946-7775 | are furiously. final
theodolites affix slyly bold deposits. even packages haggle idly slyly
specia |
| 6702.07 | Supplier#000006276 | EGYPT | 31269 | Manufacturer#2 |
,dE1anEjKQGZfgquYfkx2fkGcXH | 14-896-626-7847 | ze about the carefully
regular pint |
+-----------+--------------------+-------------+-----------+------------
----+-------------------------------------------+-----------------+-------
-----------------------------------------------------------------------
--------------------+
100 rows in set

Q3
+------------+-------------+-------------+----------------+
| l_orderkey | revenue | o_orderdate | o_shippriority |
+------------+-------------+-------------+----------------+
| 2152675 | 431309.8065 | 1995-03-28 | 0 |
| 4994400 | 423834.7976 | 1995-03-09 | 0 |
| 2160291 | 401149.7805 | 1995-03-18 | 0 |
| 2845094 | 401094.1393 | 1995-03-06 | 0 |
| 1902471 | 400497.3847 | 1995-03-01 | 0 |
| 5624358 | 395710.6806 | 1995-03-20 | 0 |
```

```
| 2346242 | 392580.0394 | 1995-03-17 | 0 |
| 2529826 | 387365.1560 | 1995-02-17 | 0 |
| 5168933 | 385433.6198 | 1995-03-20 | 0 |
| 2839239 | 380503.7310 | 1995-03-22 | 0 |
+------------+-------------+-------------+----------------+
10 rows in set

Q4
+-----------------+-------------+
| o_orderpriority | order_count |
+-----------------+-------------+
| 1-URGENT | 10623 |
| 2-HIGH | 10465 |
| 3-MEDIUM | 10309 |
| 4-NOT SPECIFIED | 10618 |
| 5-LOW | 10541 |
+-----------------+-------------+
5 rows in set

Q5
+---------------+---------------+
| n_name | revenue |
+---------------+---------------+
| PERU | 56206762.5035 |
| CANADA | 56052846.0161 |
| ARGENTINA | 54595012.8076 |
| BRAZIL | 53601776.5201 |
| UNITED STATES | 50890580.8962 |
+---------------+---------------+
5 rows in set

Q6
+---------------+
| revenue |
+---------------+
| 61660051.7967 |
+---------------+

Q7
+-------------+-------------+--------+---------------+
| supp_nation | cust_nation | l_year | revenue |
+-------------+-------------+--------+---------------+
| ARGENTINA | FRANCE | 1995 | 57928886.8015 |
| ARGENTINA | FRANCE | 1996 | 55535134.8474 |
```

```
| FRANCE | ARGENTINA | 1995 | 52916227.7375 |
| FRANCE | ARGENTINA | 1996 | 51077995.8841 |
+-------------+------------+--------+---------------+
4 rows in set

Q8
+--------+----------------------+
| o_year | mkt_share |
+--------+----------------------+
| 1995 | 0.035094304475112484 |
| 1996 | 0.03724375099464825 |
+--------+----------------------+
2 rows in set

Q9
+----------------+--------+---------------+
| nation | o_year | sum_profit |
+----------------+--------+---------------+
| ALGERIA | 1998 | 29931671.4862 |
| ALGERIA | 1997 | 49521023.1139 |
| ALGERIA | 1996 | 51283603.7356 |
| ALGERIA | 1995 | 50206939.3447 |
| ALGERIA | 1994 | 48738988.5891 |
| ALGERIA | 1993 | 48084070.1204 |
| ALGERIA | 1992 | 49725592.1793 |
| ARGENTINA | 1998 | 26407044.9262 |
| ARGENTINA | 1997 | 46224601.0785 |
| ARGENTINA | 1996 | 44579611.0571 |
| ARGENTINA | 1995 | 45081953.2540 |
| ARGENTINA | 1994 | 48291282.8512 |
| ARGENTINA | 1993 | 48063838.9130 |
| ARGENTINA | 1992 | 45277890.2991 |
| BRAZIL | 1998 | 28577022.6384 |
| BRAZIL | 1997 | 46808660.3688 |
| BRAZIL | 1996 | 47119265.0765 |
| BRAZIL | 1995 | 47706399.9100 |
| BRAZIL | 1994 | 48377469.9386 |
| BRAZIL | 1993 | 46933565.7471 |
| BRAZIL | 1992 | 47272215.5408 |
| CANADA | 1998 | 30500303.6521 |
| CANADA | 1997 | 50046257.5687 |
| CANADA | 1996 | 52638586.9029 |
| CANADA | 1995 | 50433911.3289 |
| CANADA | 1994 | 51605251.7124 |
```

```
| CANADA | 1993 | 50117218.8464 |
| CANADA | 1992 | 50347111.2789 |
| CHINA | 1998 | 26956001.9487 |
| CHINA | 1997 | 48311246.7866 |
| CHINA | 1996 | 51133929.1033 |
| CHINA | 1995 | 48024289.1049 |
| CHINA | 1994 | 50027433.6557 |
| CHINA | 1993 | 48240226.3801 |
| CHINA | 1992 | 47769117.6007 |
| EGYPT | 1998 | 26972573.1604 |
| EGYPT | 1997 | 46708654.7666 |
| EGYPT | 1996 | 46095050.4457 |
| EGYPT | 1995 | 44901908.2949 |
| EGYPT | 1994 | 48522762.8892 |
| EGYPT | 1993 | 49055807.7642 |
| EGYPT | 1992 | 46909796.1083 |
| ETHIOPIA | 1998 | 26364411.6457 |
| ETHIOPIA | 1997 | 44889623.0645 |
| ETHIOPIA | 1996 | 47554295.2892 |
| ETHIOPIA | 1995 | 44747639.5440 |
| ETHIOPIA | 1994 | 46497570.0631 |
| ETHIOPIA | 1993 | 43853718.5460 |
| ETHIOPIA | 1992 | 44005773.0397 |
| FRANCE | 1998 | 27033406.6353 |
| FRANCE | 1997 | 45763555.5515 |
| FRANCE | 1996 | 47178544.9301 |
| FRANCE | 1995 | 48821282.1929 |
| FRANCE | 1994 | 46444640.9397 |
| FRANCE | 1993 | 46602311.0590 |
| FRANCE | 1992 | 47769356.5113 |
| GERMANY | 1998 | 26165681.8305 |
| GERMANY | 1997 | 46600844.4431 |
| GERMANY | 1996 | 44873520.1979 |
| GERMANY | 1995 | 47761215.6058 |
| GERMANY | 1994 | 42283120.0209 |
| GERMANY | 1993 | 46954873.9820 |
| GERMANY | 1992 | 46263626.6361 |
| INDIA | 1998 | 27651103.0250 |
| INDIA | 1997 | 46000888.8340 |
| INDIA | 1996 | 43993476.7354 |
| INDIA | 1995 | 44015709.1914 |
| INDIA | 1994 | 44281439.6282 |
| INDIA | 1993 | 45367255.7857 |
| INDIA | 1992 | 45350810.5330 |
```

```
| INDONESIA | 1998 | 27120545.3120 |
| INDONESIA | 1997 | 45745362.3667 |
| INDONESIA | 1996 | 45347554.8232 |
| INDONESIA | 1995 | 45685709.4978 |
| INDONESIA | 1994 | 44738603.1901 |
| INDONESIA | 1993 | 45172063.2033 |
| INDONESIA | 1992 | 44623924.3942 |
| IRAN | 1998 | 27876287.0949 |
| IRAN | 1997 | 47184621.5647 |
| IRAN | 1996 | 47397859.7878 |
| IRAN | 1995 | 49579120.6991 |
| IRAN | 1994 | 48032316.8744 |
| IRAN | 1993 | 48295593.2066 |
| IRAN | 1992 | 50531453.3934 |
| IRAQ | 1998 | 29997323.2927 |
| IRAQ | 1997 | 52851471.1377 |
| IRAQ | 1996 | 53671825.6297 |
| IRAQ | 1995 | 53251012.1025 |
| IRAQ | 1994 | 50934553.4361 |
| IRAQ | 1993 | 51961214.1186 |
| IRAQ | 1992 | 50840364.3833 |
| JAPAN | 1998 | 26054615.4955 |
| JAPAN | 1997 | 43557394.2595 |
| JAPAN | 1996 | 46531743.0980 |
| JAPAN | 1995 | 41688293.4741 |
| JAPAN | 1994 | 45526719.0728 |
| JAPAN | 1993 | 45619475.4478 |
| JAPAN | 1992 | 44545639.3069 |
| JORDAN | 1998 | 24793092.4101 |
| JORDAN | 1997 | 42050730.7748 |
| JORDAN | 1996 | 42562783.8663 |
| JORDAN | 1995 | 42253019.5330 |
| JORDAN | 1994 | 45027034.7721 |
| JORDAN | 1993 | 44797510.9808 |
| JORDAN | 1992 | 41313405.2890 |
| KENYA | 1998 | 24550926.4693 |
| KENYA | 1997 | 42767120.5848 |
| KENYA | 1996 | 45000095.1105 |
| KENYA | 1995 | 43250458.0109 |
| KENYA | 1994 | 42891596.7158 |
| KENYA | 1993 | 43599201.5126 |
| KENYA | 1992 | 45286145.8141 |
| MOROCCO | 1998 | 23482053.5970 |
| MOROCCO | 1997 | 41503033.0020 |
```

```
| MOROCCO | 1996 | 45645555.9409 |
| MOROCCO | 1995 | 44462858.7689 |
| MOROCCO | 1994 | 44768368.8310 |
| MOROCCO | 1993 | 44611871.2477 |
| MOROCCO | 1992 | 43057959.1352 |
| MOZAMBIQUE | 1998 | 28824737.9244 |
| MOZAMBIQUE | 1997 | 48682746.5995 |
| MOZAMBIQUE | 1996 | 50816940.9909 |
| MOZAMBIQUE | 1995 | 50010039.0178 |
| MOZAMBIQUE | 1994 | 48794892.1253 |
| MOZAMBIQUE | 1993 | 48451128.3332 |
| MOZAMBIQUE | 1992 | 50113858.5449 |
| PERU | 1998 | 30575758.1899 |
| PERU | 1997 | 49323405.6808 |
| PERU | 1996 | 50063490.6085 |
| PERU | 1995 | 51272843.6555 |
| PERU | 1994 | 50690589.2334 |
| PERU | 1993 | 49086129.3668 |
| PERU | 1992 | 50067216.3450 |
| ROMANIA | 1998 | 27367992.9903 |
| ROMANIA | 1997 | 45668932.7094 |
| ROMANIA | 1996 | 46594220.7498 |
| ROMANIA | 1995 | 44576835.1623 |
| ROMANIA | 1994 | 45640971.0684 |
| ROMANIA | 1993 | 46374545.0712 |
| ROMANIA | 1992 | 47130533.3076 |
| RUSSIA | 1998 | 27486839.8755 |
| RUSSIA | 1997 | 44050712.6907 |
| RUSSIA | 1996 | 45604597.4983 |
| RUSSIA | 1995 | 48972490.6009 |
| RUSSIA | 1994 | 45652045.5872 |
| RUSSIA | 1993 | 47139548.1597 |
| RUSSIA | 1992 | 47159990.1221 |
| SAUDI ARABIA | 1998 | 29766229.7961 |
| SAUDI ARABIA | 1997 | 51473031.6922 |
| SAUDI ARABIA | 1996 | 52859666.6646 |
| SAUDI ARABIA | 1995 | 50946175.0229 |
| SAUDI ARABIA | 1994 | 53085288.9954 |
| SAUDI ARABIA | 1993 | 50907571.2046 |
| SAUDI ARABIA | 1992 | 50334063.0381 |
| UNITED KINGDOM | 1998 | 27904712.8220 |
| UNITED KINGDOM | 1997 | 48170994.4362 |
| UNITED KINGDOM | 1996 | 46498116.9611 |
| UNITED KINGDOM | 1995 | 43210619.0456 |
```

```
| UNITED KINGDOM | 1994 | 47339709.9122 |
| UNITED KINGDOM | 1993 | 44308436.3275 |
| UNITED KINGDOM | 1992 | 45870809.6693 |
| UNITED STATES | 1998 | 25856187.3719 |
| UNITED STATES | 1997 | 44934753.2208 |
| UNITED STATES | 1996 | 44826974.2915 |
| UNITED STATES | 1995 | 44160425.4086 |
| UNITED STATES | 1994 | 43193241.6843 |
| UNITED STATES | 1993 | 45126307.2619 |
| UNITED STATES | 1992 | 44205926.3317 |
| VIETNAM | 1998 | 28289193.6726 |
| VIETNAM | 1997 | 48284585.4019 |
| VIETNAM | 1996 | 48360225.9084 |
| VIETNAM | 1995 | 48742082.6165 |
| VIETNAM | 1994 | 49035537.3894 |
| VIETNAM | 1993 | 47222674.6352 |
| VIETNAM | 1992 | 48628336.9011 |
+----------------+--------+---------------+
175 rows in set

Q10
+-----------+--------------------+-------------+-----------+-------------
---+----------------------------------------+-----------------+--------
--------------------------------------------------------------------------
-------------------------------+
| c_custkey | c_name | revenue | c_acctbal | n_name | c_address | c_phone
| c_comment |
+-----------+--------------------+-------------+-----------+-------------
---+----------------------------------------+-----------------+--------
--------------------------------------------------------------------------
-------------------------------+
| 95962 | Customer#000095962 | 704336.0774 | -9.33 | MOZAMBIQUE |
83wOMt9iAb9OJ0HbkQ1PaX3odXVBNEIMXaE | 26-127-693-7436 | nusual
theodolites maintain furiously fluffily iro |
| 87064 | Customer#000087064 | 684037.4349 | 5244.68 | BRAZIL |
0xej6ldT8zi7MwLdDJ1II3YWwprkvwB1 I0kwsf | 12-930-206-2571 | de of the
ironic, silent warthogs. bold, r |
| 56416 | Customer#000056416 | 661218.0492 | 4303.82 | INDIA |
CEuBN,xZVmP | 18-212-984-8331 | al waters cajole along the slyly unusual
dugouts. carefully regular deposits use slyly? packages h |
| 46450 | Customer#000046450 | 646205.6835 | 2400.59 | UNITED STATES |
rzWQxB9iFpd8i4KUCAPdv | 34-765-320-4326 | ss, final deposits cajole sly |
| 128713 | Customer#000128713 | 643240.1183 | 7200.30 | ARGENTINA |
mm0kxtHFCchaZX4eYSCCyQHno7vq,SRmv4 | 11-174-994-6880 | ording to the
```

```
express accounts cajole carefully across the bravely special packages.
carefully regular account |
| 102187 | Customer#000102187 | 637493.0787 | -896.03 | ETHIOPIA |
EAi6vcGnWHUMb6rJwn,PtUgSH74tR Aixa | 15-877-462-6534 | gular packages.
carefully regular deposits cajole carefully of the regular requests.
carefully special accou |
| 42541 | Customer#000042541 | 634546.9756 | 8082.14 | IRAN | IccOGHgp8g
| 20-442-159-1337 | cross the final asymptotes. final packages wake
furiously ironic dec |
| 51595 | Customer#000051595 | 611926.8265 | 7236.80 | UNITED STATES |
wQFWZk 7JCpeg50O0KCzSmUFnNNwX1aEQ7V3Q | 34-844-269-9070 | sts. always
express accounts use carefully along the quickly speci |
| 66391 | Customer#000066391 | 608385.5852 | 9404.57 | UNITED STATES |
V0XvU1Nh9NU4zsyOkm,RBa | 34-149-224-8119 | ages cajole carefully
carefully bold deposits: fluffily unusual deposits promise slyly
carefully ironic co |
| 48358 | Customer#000048358 | 603621.4823 | -611.15 | ETHIOPIA |
ycg3uMG7iDdwQvJ1irr | 15-687-936-5181 | the slyly unusual foxes--
carefully regular |
| 99175 | Customer#000099175 | 602125.3304 | 2218.76 | INDONESIA |
9wbW52xx9T84E0dZ Rvz1ozQ1 | 19-125-912-6494 | ide of the slyly ironic
foxes boost silently ironic, even instructions. blithe |
| 122509 | Customer#000122509 | 601580.1203 | 2613.83 | KENYA |
ZN1sc0eJrkD8t6X5Q1d3 | 24-421-308-3881 | brave deposits haggle across the
even deposits. instr |
| 148055 | Customer#000148055 | 601003.6812 | 455.31 | PERU |
Y,RCZ3Bislx64nTsPaRL,5gjx7xgC6y, yKYnCw | 27-473-476-4382 | uickly final
accounts wake carefully sl |
| 117451 | Customer#000117451 | 599792.7063 | 1090.48 | UNITED STATES |
bSwr7mNPiaf1f lNK9 uTJxWCL2sn1Lak5NIB | 34-354-586-6011 | ding to the
furiously express accounts boost carefully af |
| 104110 | Customer#000104110 | 588194.3118 | 2762.52 | JORDAN | mm7
ZuDX5Z5nAQbKObB 80XBCy,1nyW | 23-639-800-5768 | urts sleep furiously
alongside of the packages! slyly ironic packages sleep |
| 13666 | Customer#000013666 | 579926.1679 | 7453.98 | EGYPT |
DLRUWGcprmWqdROJvmZwpE | 14-316-135-4381 | ross the silent requests.
special theodolit |
| 96202 | Customer#000096202 | 571017.3398 | 4703.04 | CANADA |
4Vcxcx3w4zMjVYNQaqrweweQY6TJO AP9rdvQaLl | 13-194-779-9597 | en packages
use. fluffily regular dependencies boost. never pending requ |
| 70279 | Customer#000070279 | 561369.3650 | 9109.34 | CHINA |
ltie8o3ihwffMrqMrkvN957KZVWmH5 | 28-842-825-1717 | theodolites sleep:
blithely final requests are fur |
| 16972 | Customer#000016972 | 560435.8065 | 6408.66 | ROMANIA |
```

```
X6T8vRKy6kSO0f2wJJt | 29-483-958-3347 | sts. pending deposits are across
the regular, express instructions. carefully daring foxes cajol |
| 113443 | Customer#000113443 | 557272.6706 | -72.67 | UNITED KINGDOM |
SUHbS85cYxgVkKbfh9sUpEa6ezVSlQuCKe3CV | 33-819-742-6112 | ic foxes cajole
thinly furiously stealthy instructions. pinto beans are. quickly regular
accounts integrate car |
+-----------+--------------------+-------------+-----------+-------------
---+----------------------------------------+----------------+--------
--------------------------------------------------------------------------
--------------------------------+
20 rows in set

Q11
+------------+-------------+
| ps_partkey | value |
+------------+-------------+
| 131630 | 17882680.37 |
| 104150 | 17613017.18 |
| 128284 | 16502418.74 |
| 8978 | 16470438.59 |
| 147193 | 16462742.12 |
| 78788 | 16010246.37 |
| 76331 | 15776882.05 |
| 137287 | 15770471.15 |
| 51302 | 15730620.22 |
| 141553 | 15333540.19 |
| 137196 | 15035435.60 |
| 186531 | 14818272.68 |
| 103818 | 14690943.63 |
| 80080 | 14626441.35 |
| 1312 | 14330729.50 |
| 6531 | 14267308.08 |
| 96162 | 14154396.04 |
| 69605 | 14018927.25 |
| 30118 | 13854726.38 |
| 17006 | 13731495.60 |
| 95347 | 13716648.60 |
| 18722 | 13707978.71 |
| 122875 | 13640341.00 |
| 105499 | 13532912.80 |
| 165560 | 13509536.95 |
| 1531 | 13337454.55 |
| 34732 | 13304041.48 |
| 173221 | 13038078.41 |
```

```
| 180975 | 13038039.17 |
| 24703 | 12957050.80 |
| 72036 | 12939426.90 |
| 124814 | 12849842.04 |
| 174453 | 12814999.00 |
| 14209 | 12814858.56 |
| 185186 | 12657201.05 |
| 187868 | 12647101.80 |
| 125085 | 12639931.63 |
| 80331 | 12625007.00 |
| 118685 | 12515185.68 |
| 163988 | 12484272.80 |
| 124685 | 12432747.32 |
| 92838 | 12410071.57 |
| 140928 | 12396673.84 |
| 1218 | 12362877.75 |
| 39201 | 12328085.10 |
| 33237 | 12180622.98 |
| 183791 | 12150040.50 |
| 3243 | 12136315.74 |
| 62740 | 12131313.60 |
| 154171 | 12105470.89 |
| 49034 | 11982382.52 |
| 88673 | 11925499.04 |
| 52527 | 11923653.16 |
| 83974 | 11871084.73 |
| 88254 | 11870393.22 |
| 411 | 11806670.95 |
| 14320 | 11800136.02 |
| 164979 | 11794760.03 |
| 166149 | 11778499.72 |
| 74105 | 11750224.34 |
| 169104 | 11708532.18 |
| 15542 | 11687293.42 |
| 161538 | 11661769.80 |
| 63337 | 11592505.40 |
| 117197 | 11508165.60 |
| 102989 | 11497056.75 |
| 10836 | 11465875.43 |
| 199561 | 11431793.36 |
| 134683 | 11384564.54 |
| 136318 | 11351893.30 |
| 166270 | 11336004.81 |
| 32200 | 11324838.00 |
```

```
| 57033 | 11281026.52 |
| 18098 | 11245398.24 |
| 135174 | 11189782.12 |
| 181616 | 11183947.65 |
| 85064 | 11175761.43 |
| 120719 | 11164342.08 |
| 99670 | 11140257.47 |
| 46096 | 11034143.76 |
| 195124 | 11030197.30 |
| 78838 | 11012446.40 |
| 151656 | 11010376.90 |
| 156956 | 10996384.80 |
| 34028 | 10942671.24 |
| 15778 | 10937778.75 |
| 199707 | 10924333.33 |
| 118776 | 10920609.31 |
| 27640 | 10919693.42 |
| 15237 | 10918145.54 |
| 148243 | 10916765.29 |
| 111498 | 10867707.51 |
| 132024 | 10834280.47 |
| 35124 | 10806898.50 |
| 196818 | 10787371.25 |
| 197669 | 10779504.60 |
| 110042 | 10778828.37 |
| 197422 | 10770092.44 |
| 75160 | 10746976.60 |
| 191567 | 10642430.39 |
| 34225 | 10574664.41 |
| 102588 | 10567012.05 |
| 44148 | 10505249.34 |
| 126607 | 10484944.29 |
| 172625 | 10444857.62 |
| 157054 | 10406203.24 |
| 19322 | 10378704.98 |
| 136541 | 10371536.77 |
| 167526 | 10320346.58 |
| 136011 | 10302146.84 |
| 107431 | 10273992.76 |
| 16485 | 10257703.67 |
| 52580 | 10250264.05 |
| 839 | 10238243.36 |
| 31704 | 10196678.94 |
| 122558 | 10137326.18 |
```

```
| 180386 | 10123318.07 |
| 97705 | 10089163.37 |
| 96327 | 10087851.88 |
| 136143 | 10082137.97 |
| 15174 | 10057277.55 |
| 193324 | 10039922.93 |
| 33593 | 10019952.10 |
| 126288 | 10014855.05 |
| 64123 | 9985650.90 |
| 183712 | 9973256.18 |
| 138831 | 9963069.10 |
| 123694 | 9959096.38 |
| 51734 | 9952439.73 |
| 11861 | 9949647.12 |
| 119127 | 9942105.69 |
| 173308 | 9932264.52 |
| 40986 | 9921554.40 |
| 176970 | 9919708.65 |
| 54316 | 9913595.16 |
| 62644 | 9903936.27 |
| 185354 | 9895956.52 |
| 81468 | 9885132.60 |
| 104687 | 9883888.05 |
| 198959 | 9875351.28 |
| 179767 | 9872309.86 |
| 102835 | 9870743.52 |
| 163221 | 9856173.04 |
| 32633 | 9852565.04 |
| 19605 | 9850164.48 |
| 47378 | 9826135.11 |
| 44026 | 9822433.44 |
| 126629 | 9816227.30 |
| 199665 | 9812400.23 |
| 30989 | 9812295.52 |
| 102177 | 9810372.32 |
| 25765 | 9806344.88 |
| 110721 | 9804895.23 |
| 159532 | 9803738.34 |
| 101640 | 9801375.65 |
| 151569 | 9792489.20 |
| 180629 | 9782164.34 |
| 165528 | 9769074.10 |
| 23772 | 9766084.22 |
| 149727 | 9765190.96 |
```

```
| 189605 | 9761887.80 |
| 74703 | 9758757.28 |
| 83382 | 9758144.21 |
| 93775 | 9726901.71 |
| 56192 | 9725508.16 |
| 50060 | 9712714.65 |
| 15409 | 9706898.91 |
| 139104 | 9701070.72 |
| 177435 | 9686566.09 |
| 31351 | 9675197.98 |
| 20495 | 9672566.31 |
| 24537 | 9654516.03 |
| 160528 | 9650804.70 |
| 34706 | 9647241.90 |
| 149039 | 9643498.32 |
| 147139 | 9642356.34 |
| 118629 | 9624960.80 |
| 35359 | 9621549.92 |
| 33854 | 9616857.73 |
| 33707 | 9609988.84 |
| 149055 | 9599364.32 |
| 127429 | 9580670.49 |
| 67575 | 9579613.26 |
| 80727 | 9576545.81 |
| 181650 | 9574445.40 |
| 50176 | 9573389.08 |
| 171093 | 9571625.20 |
| 151342 | 9569230.21 |
| 123052 | 9561903.68 |
| 132633 | 9545052.14 |
| 130419 | 9524936.49 |
| 89241 | 9512992.32 |
| 138255 | 9503515.93 |
| 31680 | 9502841.07 |
| 151986 | 9500862.59 |
| 146390 | 9490242.96 |
| 62275 | 9475584.10 |
| 33518 | 9475074.40 |
| 5286 | 9473739.88 |
| 39020 | 9467701.22 |
| 113281 | 9466510.94 |
| 138789 | 9464407.24 |
| 165040 | 9462153.75 |
| 150766 | 9461855.88 |
```

```
| 54341 | 9459425.45 |
| 33464 | 9459377.37 |
| 15251 | 9455980.84 |
| 145308 | 9454189.29 |
| 192621 | 9449324.14 |
| 175218 | 9448987.35 |
| 58992 | 9446144.40 |
| 24548 | 9442739.03 |
| 177563 | 9440891.04 |
| 184482 | 9431486.10 |
| 78961 | 9430401.05 |
| 174167 | 9428622.96 |
| 88265 | 9423143.28 |
| 6057 | 9405359.37 |
| 85387 | 9402175.55 |
| 47053 | 9399707.66 |
| 128973 | 9399265.92 |
| 65668 | 9395584.45 |
| 50222 | 9394502.96 |
| 116534 | 9388011.08 |
| 140959 | 9386284.56 |
| 46897 | 9385056.21 |
| 141872 | 9383820.48 |
| 177181 | 9383551.92 |
| 168265 | 9376664.16 |
| 48974 | 9374769.12 |
| 46218 | 9364135.50 |
| 104039 | 9363227.03 |
| 61538 | 9360159.08 |
| 94688 | 9359604.98 |
| 122393 | 9357937.19 |
| 7323 | 9356712.30 |
| 197892 | 9356573.44 |
| 194056 | 9352381.73 |
| 61285 | 9348480.54 |
| 180336 | 9347874.15 |
| 121930 | 9347784.74 |
| 80652 | 9347143.50 |
| 18549 | 9346038.72 |
| 23992 | 9339908.16 |
| 136583 | 9337299.56 |
| 156151 | 9337138.10 |
| 160572 | 9336553.40 |
| 113391 | 9335558.10 |
```

```
| 48068 | 9334317.92 |
| 20409 | 9331093.65 |
| 39712 | 9324685.28 |
| 59364 | 9322249.86 |
| 1344 | 9308304.39 |
| 60549 | 9308293.20 |
| 83854 | 9307387.25 |
| 92092 | 9307165.64 |
| 193306 | 9306177.31 |
| 118265 | 9300250.20 |
| 107568 | 9296254.34 |
| 109127 | 9293552.10 |
| 184688 | 9291647.92 |
| 8718 | 9287337.37 |
| 80433 | 9286295.52 |
| 26670 | 9284963.44 |
| 139548 | 9283605.21 |
| 14736 | 9280119.20 |
| 97886 | 9273852.42 |
| 181442 | 9273130.50 |
| 172360 | 9272824.92 |
| 192714 | 9268366.36 |
| 106726 | 9264879.90 |
| 72157 | 9263498.40 |
| 70445 | 9257553.92 |
| 75148 | 9257420.83 |
| 26170 | 9256074.12 |
| 116531 | 9249721.71 |
| 133665 | 9245464.80 |
| 129041 | 9244629.48 |
| 136486 | 9240748.92 |
| 198924 | 9239976.06 |
| 115254 | 9233580.37 |
| 168135 | 9232693.98 |
| 22480 | 9232190.78 |
| 192018 | 9230386.58 |
| 111889 | 9228204.96 |
| 151661 | 9227926.90 |
| 96482 | 9226960.85 |
| 49198 | 9226436.40 |
| 41219 | 9222883.52 |
| 113502 | 9222208.59 |
| 84009 | 9218703.22 |
| 192788 | 9213468.00 |
```

```
| 160251 | 9206353.32 |
| 188162 | 9200537.88 |
| 167589 | 9195835.03 |
| 132673 | 9194021.22 |
| 191105 | 9192417.12 |
| 128748 | 9189941.55 |
| 130423 | 9184710.96 |
| 22639 | 9182963.16 |
| 199034 | 9180909.86 |
| 187644 | 9180350.20 |
| 970 | 9175757.70 |
| 59070 | 9170000.64 |
| 66568 | 9166070.04 |
| 52715 | 9161221.80 |
| 130276 | 9161201.57 |
| 24189 | 9160740.15 |
| 132402 | 9144498.48 |
| 37799 | 9142271.24 |
| 173337 | 9140566.68 |
| 176552 | 9135054.51 |
| 195714 | 9133679.77 |
| 119363 | 9123261.90 |
| 161160 | 9122259.60 |
| 196968 | 9111592.20 |
| 61943 | 9111527.33 |
| 79766 | 9109534.89 |
| 178082 | 9105694.92 |
| 38800 | 9105468.72 |
| 83608 | 9099493.68 |
| 146346 | 9098628.00 |
| 116690 | 9098099.93 |
| 64690 | 9095441.10 |
| 82061 | 9095381.18 |
| 89015 | 9092660.48 |
| 188457 | 9091400.40 |
| 125177 | 9090455.55 |
| 114776 | 9088177.68 |
| 4486 | 9087487.20 |
| 176940 | 9086842.84 |
| 93157 | 9084361.81 |
| 148624 | 9083370.78 |
| 4441 | 9079520.58 |
| 63590 | 9079125.44 |
| 174189 | 9078023.39 |
```

```
| 63054 | 9075441.98 |
| 14950 | 9073156.19 |
| 175646 | 9072322.47 |
| 63712 | 9067710.48 |
| 157197 | 9067452.77 |
| 147196 | 9064699.80 |
| 50551 | 9062434.72 |
| 43035 | 9061782.03 |
| 187679 | 9056529.40 |
| 96673 | 9056525.94 |
| 130148 | 9054217.06 |
| 159007 | 9053155.29 |
| 41544 | 9052820.94 |
| 109476 | 9048012.09 |
| 60092 | 9045562.44 |
| 197490 | 9044579.88 |
| 47311 | 9037223.52 |
| 87230 | 9033227.61 |
| 3860 | 9030622.02 |
| 5466 | 9029841.66 |
| 171537 | 9024699.30 |
| 39707 | 9022833.12 |
| 167048 | 9022709.18 |
| 109006 | 9022258.40 |
| 17910 | 9019688.45 |
| 132826 | 9017286.74 |
| 157502 | 9016444.08 |
| 142309 | 9016270.60 |
| 78891 | 9005693.25 |
| 88301 | 9002414.82 |
| 11496 | 9000803.97 |
| 163633 | 8996162.06 |
| 151809 | 8993104.95 |
| 131555 | 8988340.68 |
| 72812 | 8985370.68 |
| 77047 | 8981489.79 |
| 1553 | 8977226.10 |
| 162531 | 8973689.92 |
| 154026 | 8973320.24 |
| 125499 | 8969667.84 |
| 34547 | 8966116.43 |
| 41301 | 8965350.42 |
| 12853 | 8959403.59 |
| 27736 | 8957933.23 |
```

```
| 162817 | 8956868.20 |
| 155389 | 8955349.85 |
| 130360 | 8952928.25 |
| 120878 | 8952393.10 |
| 150671 | 8952112.72 |
| 190365 | 8951671.57 |
| 72364 | 8950587.82 |
| 71615 | 8949277.07 |
| 95277 | 8947796.58 |
| 78180 | 8946814.80 |
| 97062 | 8945057.46 |
| 170013 | 8944660.40 |
| 113426 | 8943016.29 |
| 173751 | 8942914.28 |
| 1478 | 8941906.24 |
| 26061 | 8941022.48 |
| 152527 | 8939654.10 |
| 148360 | 8939589.40 |
| 44057 | 8939101.36 |
| 13595 | 8936720.10 |
| 33337 | 8935366.48 |
| 169698 | 8931507.20 |
| 26155 | 8927283.11 |
| 17185 | 8927218.40 |
| 51996 | 8926661.08 |
| 101869 | 8919281.70 |
| 14561 | 8910653.92 |
| 190047 | 8909427.90 |
| 104143 | 8909328.40 |
| 133330 | 8907195.90 |
| 169144 | 8904989.34 |
| 87067 | 8900079.44 |
| 176075 | 8898845.64 |
| 25076 | 8895274.12 |
| 80838 | 8895205.30 |
| 40387 | 8890891.55 |
| 88004 | 8888748.80 |
| 105527 | 8888672.72 |
| 40741 | 8886674.24 |
| 76690 | 8880622.61 |
| 86485 | 8880488.57 |
| 75736 | 8877666.06 |
| 48704 | 8876626.52 |
| 56450 | 8872277.59 |
```

```
| 61683 | 8870173.93 |
| 24067 | 8867814.12 |
| 108012 | 8863632.38 |
| 180971 | 8862007.20 |
| 132986 | 8861335.20 |
| 35839 | 8859344.64 |
| 191553 | 8857411.14 |
| 163492 | 8855825.91 |
| 112101 | 8851904.10 |
| 27050 | 8847924.19 |
| 57481 | 8845309.59 |
| 163252 | 8842276.65 |
| 87958 | 8840221.67 |
| 60162 | 8838927.08 |
| 131928 | 8838900.48 |
| 123514 | 8833601.14 |
| 42891 | 8830401.37 |
| 71547 | 8829540.72 |
| 13975 | 8826582.48 |
| 31577 | 8825371.40 |
| 86165 | 8816308.38 |
| 164646 | 8815470.18 |
| 150176 | 8814992.11 |
| 152464 | 8814533.82 |
| 183434 | 8813941.24 |
| 58839 | 8808010.20 |
| 59952 | 8801497.32 |
| 151038 | 8800215.80 |
| 139523 | 8800032.57 |
| 8828 | 8798704.66 |
| 14080 | 8797032.12 |
| 194080 | 8792825.27 |
| 87199 | 8788933.64 |
| 91747 | 8785811.64 |
| 194429 | 8776185.03 |
| 118998 | 8776071.00 |
| 179467 | 8771474.74 |
| 68715 | 8771302.80 |
| 180572 | 8771095.68 |
| 19821 | 8770770.82 |
| 41702 | 8770565.71 |
| 27916 | 8769001.47 |
| 121302 | 8763598.50 |
| 107013 | 8762893.37 |
```

```
| 37287 | 8761196.43 |
| 117050 | 8758230.00 |
| 58547 | 8757757.40 |
| 197088 | 8749026.12 |
| 55839 | 8747234.02 |
| 71829 | 8744546.91 |
| 30961 | 8743416.92 |
| 134548 | 8741635.28 |
| 179833 | 8738680.00 |
| 79721 | 8737857.70 |
| 144577 | 8736427.08 |
| 29051 | 8729063.28 |
| 131481 | 8728799.64 |
| 73271 | 8727985.25 |
| 89553 | 8725727.19 |
| 31306 | 8724451.12 |
| 82181 | 8724017.16 |
| 95549 | 8723460.30 |
| 31507 | 8722094.40 |
| 21302 | 8722054.95 |
| 137953 | 8721611.83 |
| 195768 | 8721020.99 |
| 180105 | 8718021.20 |
| 98241 | 8717935.36 |
| 59431 | 8715482.28 |
| 143694 | 8713267.63 |
| 109020 | 8713043.36 |
| 46732 | 8711642.04 |
| 144172 | 8711013.10 |
| 139056 | 8710786.50 |
| 107543 | 8706135.75 |
| 89127 | 8705410.56 |
| 146544 | 8704812.86 |
| 195524 | 8699333.14 |
| 133563 | 8698060.14 |
| 112707 | 8694322.84 |
| 98951 | 8690376.70 |
| 132635 | 8689305.24 |
| 69056 | 8688980.25 |
| 134143 | 8688695.26 |
| 148150 | 8687553.16 |
| 89122 | 8686767.31 |
| 15085 | 8685772.26 |
| 196686 | 8682783.57 |
```

```
| 3076 | 8672940.78 |
| 137428 | 8672547.80 |
| 27263 | 8671719.36 |
| 101561 | 8667962.72 |
| 12597 | 8662223.52 |
| 143329 | 8661688.72 |
| 130813 | 8659409.04 |
| 183679 | 8658698.30 |
| 47449 | 8658493.58 |
| 164677 | 8658220.00 |
| 51437 | 8654713.02 |
| 116162 | 8649713.36 |
| 71889 | 8645159.67 |
| 6486 | 8639891.76 |
| 192102 | 8638102.72 |
| 101660 | 8634451.80 |
| 124703 | 8633146.86 |
| 150469 | 8631948.60 |
| 197467 | 8630739.78 |
| 97621 | 8630453.32 |
| 150354 | 8630288.15 |
| 179544 | 8630121.63 |
| 38972 | 8626072.00 |
| 110732 | 8625761.16 |
| 170791 | 8625203.06 |
| 149414 | 8617070.17 |
| 59527 | 8616079.20 |
| 157580 | 8615676.04 |
| 16268 | 8615087.46 |
| 76464 | 8610219.38 |
| 44474 | 8607934.92 |
| 125527 | 8607708.08 |
| 118076 | 8602251.65 |
| 180362 | 8601367.05 |
| 5808 | 8599851.04 |
| 28703 | 8599486.36 |
| 113373 | 8597996.36 |
| 118918 | 8597063.80 |
| 44868 | 8596304.52 |
| 43419 | 8596265.35 |
| 89763 | 8595248.64 |
| 119232 | 8594224.56 |
| 108649 | 8590683.68 |
| 10396 | 8588398.05 |
```

```
| 79536 | 8587117.83 |
| 149800 | 8587058.86 |
| 165839 | 8582991.20 |
| 115397 | 8581524.77 |
| 104394 | 8581384.42 |
| 142569 | 8581127.40 |
| 63676 | 8580930.08 |
| 29029 | 8580613.53 |
| 156604 | 8580477.00 |
| 7310 | 8579949.50 |
| 105381 | 8576164.24 |
| 84306 | 8573960.40 |
| 61217 | 8570393.04 |
| 164438 | 8569616.36 |
| 28073 | 8565639.60 |
| 125743 | 8563258.90 |
| 190032 | 8561620.55 |
| 147122 | 8561245.68 |
| 5384 | 8558830.08 |
| 70172 | 8558319.64 |
| 161966 | 8556193.38 |
| 69530 | 8554377.60 |
| 111243 | 8553627.55 |
| 72590 | 8551077.51 |
| 134423 | 8550604.77 |
| 44509 | 8547134.31 |
| 160707 | 8546000.68 |
| 54123 | 8545976.26 |
| 36547 | 8540333.04 |
| 48715 | 8537983.35 |
| 103078 | 8537142.60 |
| 137613 | 8536278.96 |
| 44995 | 8532416.72 |
| 191159 | 8532173.37 |
| 119345 | 8532070.56 |
| 109941 | 8531904.79 |
| 5449 | 8528034.35 |
| 134116 | 8526854.95 |
| 199268 | 8523599.58 |
| 168520 | 8523360.67 |
| 154189 | 8521620.13 |
| 108771 | 8513853.87 |
| 198651 | 8511238.80 |
| 93681 | 8510935.14 |
```

```
| 170680 | 8509087.68 |
| 106409 | 8506859.19 |
| 27110 | 8499811.75 |
| 43224 | 8499539.52 |
| 153225 | 8499434.28 |
| 16681 | 8498021.66 |
| 117983 | 8496934.32 |
| 192158 | 8492372.03 |
| 33900 | 8491139.64 |
| 37006 | 8489126.28 |
| 176554 | 8488633.92 |
| 69234 | 8484937.26 |
| 176652 | 8484496.02 |
| 41660 | 8480585.65 |
| 129104 | 8480411.17 |
| 66960 | 8478978.86 |
| 36296 | 8472438.75 |
| 98665 | 8471241.57 |
| 134173 | 8467888.57 |
| 60496 | 8467019.22 |
| 197520 | 8466553.20 |
| 116746 | 8465792.60 |
| 187394 | 8458248.24 |
| 140377 | 8455546.68 |
| 97326 | 8450501.67 |
| 26770 | 8449625.64 |
| 104884 | 8446152.26 |
| 143109 | 8443547.19 |
| 127361 | 8441094.08 |
| 104754 | 8436883.50 |
| 183676 | 8436165.76 |
| 906 | 8434608.12 |
| 55768 | 8433763.69 |
| 118654 | 8433465.57 |
| 39310 | 8433214.55 |
| 173261 | 8432992.53 |
| 93976 | 8432605.20 |
| 63318 | 8432149.26 |
| 128243 | 8424182.94 |
| 156063 | 8422743.54 |
| 195087 | 8421279.30 |
| 67668 | 8417594.98 |
| 49882 | 8417237.80 |
| 105631 | 8412628.07 |
```

```
| 40987 | 8406033.41 |
| 185735 | 8404112.83 |
| 173986 | 8403050.34 |
| 87372 | 8402838.40 |
| 24509 | 8398807.24 |
| 180522 | 8394989.75 |
| 76215 | 8394433.35 |
| 193872 | 8390435.23 |
| 141234 | 8390180.92 |
| 91138 | 8386645.20 |
| 28097 | 8385577.38 |
| 4053 | 8384952.75 |
| 17050 | 8380304.40 |
| 64050 | 8377921.56 |
| 80836 | 8375803.16 |
| 86084 | 8373551.95 |
| 168499 | 8373348.72 |
| 178642 | 8372218.52 |
| 8498 | 8370557.16 |
| 156312 | 8366249.30 |
| 136803 | 8361949.92 |
| 92109 | 8359503.23 |
| 138625 | 8358135.21 |
| 137540 | 8358031.08 |
| 176531 | 8355437.00 |
| 53783 | 8352395.63 |
| 106977 | 8352334.98 |
| 21385 | 8351786.37 |
| 114885 | 8351582.40 |
| 113643 | 8350530.65 |
| 89061 | 8349422.08 |
| 77752 | 8348730.24 |
| 28623 | 8348321.44 |
| 74478 | 8348064.27 |
| 41383 | 8347223.45 |
| 147632 | 8346967.80 |
| 40948 | 8346743.30 |
| 154324 | 8346521.91 |
| 89724 | 8346034.80 |
| 119083 | 8338084.92 |
| 124143 | 8335841.76 |
| 80512 | 8335705.69 |
| 105047 | 8332249.86 |
| 38243 | 8329017.19 |
```

```
| 42583 | 8328613.91 |
| 44240 | 8327684.64 |
| 57611 | 8321693.94 |
| 9730 | 8319725.70 |
| 91655 | 8318837.40 |
| 13140 | 8316216.96 |
| 112257 | 8315169.85 |
| 27182 | 8314740.99 |
| 166654 | 8314332.64 |
| 40572 | 8312654.55 |
| 26680 | 8311626.68 |
| 138947 | 8311347.29 |
| 184982 | 8310393.08 |
| 35540 | 8308058.43 |
| 181446 | 8304851.76 |
| 65160 | 8299581.90 |
| 9533 | 8299139.42 |
| 67836 | 8294228.46 |
| 159414 | 8293114.90 |
| 115025 | 8291746.65 |
| 30780 | 8291580.00 |
| 164680 | 8290263.02 |
| 4599 | 8288816.03 |
| 73366 | 8286818.96 |
| 135625 | 8284930.92 |
| 46497 | 8284638.88 |
| 63781 | 8284447.60 |
| 84332 | 8283372.14 |
| 196269 | 8276407.36 |
| 166651 | 8275663.35 |
| 142 | 8273960.31 |
| 56904 | 8272891.44 |
| 46821 | 8272603.71 |
| 76051 | 8272300.75 |
| 19666 | 8270192.64 |
| 92723 | 8267074.20 |
| 125843 | 8266816.38 |
| 158722 | 8266634.88 |
| 28941 | 8266245.12 |
| 39968 | 8265605.53 |
| 41429 | 8265317.84 |
| 61601 | 8264074.31 |
| 179159 | 8260137.47 |
| 15969 | 8259835.96 |
```

```
| 121125 | 8253912.49 |
| 66486 | 8253743.66 |
| 181031 | 8253570.14 |
| 43712 | 8250825.78 |
| 13842 | 8245765.00 |
| 76203 | 8245412.16 |
| 68992 | 8243081.46 |
| 119704 | 8241363.06 |
| 86109 | 8240377.92 |
| 29534 | 8239914.00 |
| 68596 | 8239825.29 |
| 168291 | 8237626.32 |
| 183308 | 8235947.21 |
| 78657 | 8233481.64 |
| 193545 | 8233037.49 |
| 23658 | 8232306.18 |
| 179945 | 8231365.25 |
| 53391 | 8231252.10 |
| 71380 | 8231125.68 |
| 53666 | 8226715.00 |
| 118592 | 8226181.00 |
| 67203 | 8225355.99 |
| 1178 | 8224625.05 |
| 147876 | 8224189.62 |
| 80042 | 8220826.70 |
| 48950 | 8218611.22 |
| 43331 | 8218448.04 |
| 177706 | 8215723.50 |
| 145442 | 8215706.16 |
| 197042 | 8215536.00 |
| 169952 | 8214698.43 |
| 57907 | 8211740.04 |
| 145741 | 8210316.57 |
| 91144 | 8209855.02 |
| 160266 | 8209468.80 |
| 31602 | 8209366.90 |
| 98672 | 8208412.85 |
| 199012 | 8207897.50 |
| 151148 | 8207645.16 |
| 116545 | 8207573.24 |
| 122176 | 8207508.04 |
| 11021 | 8206766.10 |
| 47752 | 8203436.82 |
| 124 | 8203209.30 |
```

```
| 148126 | 8202846.66 |
| 15753 | 8202695.55 |
| 50833 | 8200880.16 |
| 11523 | 8196478.02 |
| 71478 | 8195930.68 |
| 129262 | 8190520.80 |
| 43023 | 8186451.85 |
| 119193 | 8184853.14 |
| 85067 | 8182638.86 |
| 164534 | 8181563.04 |
| 82556 | 8180455.14 |
| 31813 | 8179417.14 |
| 81345 | 8173128.69 |
| 38413 | 8172464.04 |
| 106014 | 8171418.35 |
| 191180 | 8170663.97 |
| 43274 | 8169669.72 |
| 5837 | 8166123.50 |
| 63332 | 8161839.60 |
| 47668 | 8161790.04 |
| 112468 | 8160728.40 |
| 132541 | 8160680.00 |
| 59457 | 8160393.33 |
| 71751 | 8159865.19 |
| 118395 | 8156795.00 |
| 132390 | 8154867.54 |
| 44792 | 8153384.22 |
| 128838 | 8153018.30 |
| 87197 | 8152281.72 |
| 187978 | 8150832.56 |
| 147419 | 8150063.60 |
| 149166 | 8149406.78 |
| 196012 | 8147307.42 |
| 190519 | 8145402.96 |
| 151511 | 8144276.58 |
| 88891 | 8140166.24 |
| 168056 | 8139101.96 |
| 189186 | 8136933.25 |
| 117326 | 8136047.82 |
| 60575 | 8133316.80 |
| 75452 | 8130427.37 |
| 194126 | 8129751.80 |
| 130199 | 8129270.88 |
| 41680 | 8128823.40 |
```

```
| 107624 | 8125799.20 |
| 135069 | 8123999.10 |
| 119032 | 8123770.24 |
| 27635 | 8123076.65 |
| 14317 | 8121553.23 |
| 148018 | 8119898.16 |
| 51152 | 8118370.26 |
| 112643 | 8117331.37 |
| 119526 | 8116075.80 |
| 192084 | 8114896.38 |
| 151385 | 8114711.28 |
| 160836 | 8112053.68 |
| 91468 | 8111785.50 |
| 58877 | 8108256.25 |
| 41885 | 8107026.81 |
| 155542 | 8106757.18 |
| 149968 | 8104953.78 |
| 168380 | 8103576.00 |
| 134641 | 8101092.32 |
| 92470 | 8100877.70 |
| 113610 | 8098591.93 |
| 198538 | 8097343.20 |
| 122506 | 8096090.76 |
| 29082 | 8093543.55 |
| 161345 | 8093157.93 |
| 105743 | 8093045.53 |
| 103572 | 8091573.66 |
| 59514 | 8089470.48 |
| 8801 | 8088454.15 |
| 129062 | 8088206.58 |
| 155464 | 8086115.79 |
| 86363 | 8082561.00 |
| 180836 | 8082087.30 |
| 92558 | 8081407.80 |
| 85120 | 8073164.00 |
| 149026 | 8072285.40 |
| 51138 | 8072074.48 |
| 36306 | 8071648.86 |
| 102380 | 8070503.00 |
| 147597 | 8069397.60 |
| 41382 | 8059995.35 |
| 121856 | 8059809.11 |
| 86644 | 8058667.76 |
| 108481 | 8058214.81 |
```

```
| 41685 | 8057355.39 |
| 175712 | 8054878.30 |
| 72815 | 8052294.24 |
| 58794 | 8047848.00 |
| 118769 | 8047465.14 |
| 157192 | 8046501.96 |
| 195708 | 8045001.94 |
| 163683 | 8044727.02 |
| 189018 | 8043927.54 |
| 62904 | 8043011.65 |
| 80095 | 8042575.59 |
| 90500 | 8042502.65 |
| 73281 | 8040167.52 |
| 150710 | 8035910.80 |
| 139282 | 8034489.36 |
| 172904 | 8033791.68 |
| 38881 | 8032557.38 |
| 53055 | 8030796.15 |
| 105816 | 8025318.24 |
| 88304 | 8024637.06 |
| 115565 | 8023928.25 |
| 55376 | 8021432.16 |
| 56334 | 8019313.12 |
| 58875 | 8016065.00 |
| 4688 | 8012303.00 |
| 49117 | 8009207.80 |
| 57173 | 8008116.27 |
| 48176 | 8006765.85 |
| 112191 | 8003883.39 |
| 33265 | 8002391.76 |
| 181788 | 8002030.50 |
| 172799 | 8001050.55 |
| 2084 | 7999172.30 |
| 174747 | 7997167.48 |
| 171184 | 7996930.11 |
| 113271 | 7992683.04 |
| 68662 | 7991426.30 |
| 179375 | 7991170.88 |
| 188383 | 7990226.27 |
| 50208 | 7989363.27 |
| 23653 | 7988890.87 |
| 159419 | 7988841.36 |
| 74581 | 7987356.50 |
| 133590 | 7986046.81 |
```

```
| 195820 | 7985473.14 |
| 87903 | 7983482.88 |
| 69032 | 7981908.18 |
| 113975 | 7980561.00 |
| 178678 | 7975116.93 |
| 52316 | 7973618.16 |
| 135546 | 7972669.80 |
| 89425 | 7970077.44 |
| 115937 | 7966015.20 |
| 151483 | 7964850.88 |
| 73974 | 7964186.23 |
| 39976 | 7964104.24 |
| 130168 | 7961690.88 |
| 58973 | 7957416.76 |
| 16354 | 7956051.07 |
| 23988 | 7955837.92 |
| 138467 | 7955481.05 |
| 26096 | 7955212.32 |
| 192216 | 7953429.18 |
| 112833 | 7952279.26 |
| 60599 | 7951261.80 |
| 129116 | 7948811.85 |
| 79529 | 7947581.91 |
| 71616 | 7944476.54 |
| 136821 | 7942188.24 |
| 116204 | 7941096.90 |
| 165298 | 7939933.31 |
| 44009 | 7939859.65 |
| 194487 | 7938247.20 |
| 11299 | 7938135.81 |
| 76488 | 7935926.86 |
| 58998 | 7934414.04 |
| 25175 | 7931035.11 |
| 136144 | 7929283.23 |
| 132829 | 7926841.62 |
| 84176 | 7925781.05 |
| 68592 | 7922872.98 |
| 139280 | 7922119.48 |
| 160669 | 7921588.43 |
| 42938 | 7917524.56 |
| 183183 | 7915624.86 |
| 95449 | 7914292.08 |
| 115390 | 7912655.54 |
| 173723 | 7911329.40 |
```

```
| 48992 | 7911153.12 |
| 173464 | 7910458.65 |
| 26098 | 7910217.75 |
| 141115 | 7909496.38 |
| 195218 | 7906315.56 |
| 116608 | 7906302.60 |
| 163793 | 7905477.33 |
| 10419 | 7904598.30 |
| 106312 | 7901466.72 |
| 48674 | 7901010.24 |
| 35198 | 7899974.88 |
| 88954 | 7899573.52 |
| 41505 | 7897709.99 |
| 115586 | 7897301.88 |
| 167431 | 7895826.00 |
| 158787 | 7894948.50 |
| 161712 | 7893410.70 |
| 46930 | 7892707.77 |
| 58633 | 7892088.15 |
| 10599 | 7892067.69 |
| 99523 | 7891485.16 |
| 70126 | 7890247.41 |
| 32476 | 7890149.34 |
| 152617 | 7890136.50 |
| 162639 | 7889822.70 |
| 82056 | 7889345.05 |
| 186450 | 7887873.56 |
| 39082 | 7886019.89 |
| 183217 | 7885948.48 |
| 192551 | 7884432.48 |
| 164801 | 7882870.10 |
| 112804 | 7882772.00 |
| 5956 | 7878805.04 |
| 73054 | 7878479.63 |
| 62593 | 7878401.44 |
| 137687 | 7873755.91 |
| 80526 | 7871839.50 |
| 195354 | 7869617.75 |
| 4122 | 7867967.09 |
| 4057 | 7865176.80 |
| 63195 | 7864322.16 |
| 143370 | 7863444.54 |
| 41473 | 7862926.89 |
| 155060 | 7860900.96 |
```

```
| 76875 | 7858529.64 |
| 135778 | 7857660.51 |
| 30534 | 7855226.08 |
| 99405 | 7853410.95 |
| 161551 | 7852244.40 |
| 185034 | 7850752.00 |
| 17264 | 7850704.88 |
| 23652 | 7848909.16 |
| 123681 | 7848265.36 |
| 186170 | 7845527.50 |
| 81496 | 7840427.40 |
| 25407 | 7840234.72 |
| 96662 | 7839907.41 |
| 156407 | 7839647.75 |
| 165843 | 7839562.80 |
| 153361 | 7838813.07 |
| 149362 | 7838282.52 |
| 46057 | 7835709.81 |
| 114341 | 7835492.25 |
| 154823 | 7834898.61 |
| 139538 | 7834690.64 |
| 42853 | 7833252.60 |
| 177659 | 7831803.58 |
| 29158 | 7829880.80 |
| 85583 | 7825996.64 |
| 165714 | 7825006.46 |
| 58662 | 7821977.76 |
| 185839 | 7821640.74 |
| 93559 | 7821137.52 |
| 58481 | 7818648.16 |
| 162217 | 7817923.47 |
| 130014 | 7815929.34 |
| 125640 | 7815262.90 |
| 83723 | 7815021.48 |
| 54314 | 7813732.94 |
| 146652 | 7809817.39 |
| 189256 | 7808972.00 |
| 87994 | 7808660.48 |
| 157067 | 7806217.25 |
| 56859 | 7805947.60 |
| 118132 | 7804423.69 |
| 189457 | 7802777.91 |
| 1509 | 7802315.42 |
| 129101 | 7801994.70 |
```

```
| 162285 | 7801859.52 |
| 182358 | 7801430.46 |
| 6288 | 7800363.30 |
| 68972 | 7799224.95 |
| 51684 | 7795455.46 |
| 148645 | 7794585.92 |
| 94359 | 7794358.92 |
| 40451 | 7791437.70 |
| 44019 | 7790053.76 |
| 81470 | 7788716.85 |
| 12731 | 7786998.38 |
| 114393 | 7784963.34 |
| 69323 | 7783583.08 |
| 169794 | 7780968.30 |
| 25378 | 7778569.60 |
| 104509 | 7777137.62 |
| 81874 | 7775216.80 |
| 70859 | 7771185.07 |
| 135768 | 7769704.84 |
| 181960 | 7768847.90 |
| 28481 | 7768516.61 |
| 191604 | 7765367.68 |
| 754 | 7762507.02 |
| 127702 | 7761776.05 |
| 36488 | 7761744.00 |
| 183906 | 7759864.80 |
| 90365 | 7759602.50 |
| 60725 | 7759495.78 |
| 69436 | 7759033.52 |
| 12963 | 7756623.52 |
| 64571 | 7755731.04 |
| 160111 | 7753787.70 |
| 107970 | 7753735.88 |
| 132036 | 7753401.36 |
| 79965 | 7748656.15 |
| 149862 | 7747239.10 |
| 73218 | 7745499.42 |
| 161036 | 7742807.45 |
| 152467 | 7742471.40 |
| 163358 | 7742034.00 |
| 197951 | 7741768.84 |
| 15820 | 7740003.00 |
| 31444 | 7739519.60 |
| 151208 | 7738273.85 |
```

```
| 20410 | 7737192.99 |
| 45462 | 7736792.55 |
| 128966 | 7736467.65 |
| 118945 | 7735275.00 |
| 106458 | 7734069.72 |
| 162706 | 7730189.88 |
| 70528 | 7730088.25 |
| 107998 | 7728273.45 |
| 163110 | 7728042.40 |
| 74591 | 7727297.76 |
| 121454 | 7726200.56 |
| 181252 | 7724464.38 |
| 29154 | 7724129.66 |
| 63854 | 7720353.88 |
| 34157 | 7719803.30 |
| 30684 | 7718307.84 |
| 3985 | 7715042.96 |
| 29387 | 7714858.80 |
| 184703 | 7712545.12 |
| 124679 | 7712528.72 |
| 15606 | 7710658.46 |
| 123814 | 7709872.95 |
| 83760 | 7709633.92 |
| 22084 | 7707219.79 |
| 123210 | 7706030.42 |
| 75066 | 7704727.51 |
| 16337 | 7704517.80 |
| 47109 | 7704111.51 |
| 8232 | 7702887.50 |
| 11222 | 7702535.62 |
| 84961 | 7701923.72 |
| 157118 | 7700132.88 |
| 118362 | 7699210.20 |
| 193755 | 7698545.20 |
| 1520 | 7697759.37 |
| 114599 | 7697377.50 |
| 168842 | 7696152.00 |
| 172245 | 7694286.06 |
| 4584 | 7693352.79 |
| 113651 | 7689659.67 |
| 183207 | 7687955.66 |
| 175802 | 7686604.70 |
| 59066 | 7685120.43 |
| 130726 | 7684159.25 |
```

```
| 89672 | 7684049.50 |
| 7224 | 7683446.40 |
| 97533 | 7680694.62 |
| 59941 | 7680100.80 |
| 29298 | 7676823.42 |
| 163962 | 7675924.96 |
| 41086 | 7674518.14 |
| 185483 | 7673376.60 |
| 165010 | 7672469.70 |
| 3708 | 7671744.18 |
| 192994 | 7671712.00 |
| 79968 | 7668060.48 |
| 118494 | 7666659.00 |
| 59236 | 7666625.98 |
| 149509 | 7665930.67 |
| 3793 | 7664981.28 |
| 28979 | 7664632.93 |
| 178389 | 7662544.96 |
| 65315 | 7661085.88 |
| 59710 | 7657442.00 |
| 170276 | 7656813.89 |
| 182707 | 7656387.06 |
| 129170 | 7655820.48 |
| 59765 | 7655009.92 |
| 23337 | 7654271.94 |
| 90396 | 7653568.35 |
| 68842 | 7652742.72 |
| 16315 | 7652630.70 |
| 956 | 7652174.81 |
| 10639 | 7651375.80 |
| 112886 | 7649534.08 |
| 9561 | 7648502.73 |
| 65484 | 7647789.30 |
| 68677 | 7646879.14 |
| 196529 | 7645482.24 |
| 6556 | 7642116.06 |
| 9113 | 7640163.68 |
| 128139 | 7638760.00 |
| 143264 | 7635499.56 |
| 21569 | 7634785.86 |
| 193402 | 7633576.06 |
| 35545 | 7632210.69 |
| 65068 | 7632188.76 |
| 25515 | 7630952.93 |
```

```
| 180189 | 7630887.10 |
| 131680 | 7629593.64 |
| 80162 | 7629440.93 |
| 139054 | 7629417.37 |
| 8028 | 7629134.04 |
| 76804 | 7626731.00 |
| 74179 | 7624974.03 |
| 122507 | 7623903.87 |
| 141889 | 7623552.30 |
| 184279 | 7623048.17 |
| 8076 | 7620897.81 |
| 192681 | 7619802.09 |
| 21398 | 7617942.52 |
| 14825 | 7617843.60 |
| 17969 | 7617524.64 |
| 170764 | 7616119.96 |
| 115303 | 7615914.17 |
| 67708 | 7615306.08 |
| 33317 | 7613417.24 |
| 190782 | 7613203.42 |
| 113818 | 7612852.48 |
| 178091 | 7611457.30 |
| 87603 | 7611343.68 |
| 108317 | 7610509.71 |
| 106552 | 7609868.84 |
| 28679 | 7609292.20 |
| 192350 | 7609140.81 |
| 154801 | 7607944.38 |
| 5768 | 7607785.68 |
| 127689 | 7606313.94 |
| 62847 | 7605651.45 |
| 111212 | 7605052.00 |
| 156065 | 7603327.60 |
| 115140 | 7601161.68 |
| 19597 | 7601153.46 |
| 55233 | 7600940.23 |
| 89353 | 7600929.84 |
| 75701 | 7600492.60 |
| 64974 | 7599754.80 |
| 116156 | 7597452.48 |
| 59491 | 7596352.84 |
| 6138 | 7594861.54 |
| 62317 | 7594854.10 |
| 106575 | 7594520.08 |
```

```
| 161092 | 7594454.40 |
| 9872 | 7593734.34 |
| 77711 | 7593431.60 |
| 61206 | 7593153.00 |
| 123776 | 7592736.80 |
| 185141 | 7592617.12 |
| 5542 | 7592513.04 |
| 185296 | 7591439.31 |
| 72597 | 7591142.40 |
+------------+-------------+
1225 rows in set

Q12
+------------+-----------------+----------------+
| l_shipmode | high_line_count | low_line_count |
+------------+-----------------+----------------+
| FOB | 6273 | 9429 |
| TRUCK | 6336 | 9300 |
+------------+-----------------+----------------+

Q13
+---------+----------+
| c_count | custdist |
+---------+----------+
| 0 | 50005 |
| 10 | 6574 |
| 9 | 6554 |
| 11 | 6072 |
| 8 | 5934 |
| 12 | 5598 |
| 13 | 5032 |
| 19 | 4685 |
| 7 | 4663 |
| 20 | 4607 |
| 17 | 4550 |
| 18 | 4515 |
| 14 | 4480 |
| 15 | 4476 |
| 16 | 4341 |
| 21 | 4176 |
| 22 | 3710 |
| 6 | 3303 |
| 23 | 3172 |
| 24 | 2670 |
```

```
| 25 | 2111 |
| 5 | 1954 |
| 26 | 1605 |
| 27 | 1195 |
| 4 | 1030 |
| 28 | 898 |
| 29 | 620 |
| 3 | 408 |
| 30 | 353 |
| 31 | 225 |
| 32 | 135 |
| 2 | 128 |
| 33 | 82 |
| 34 | 54 |
| 35 | 33 |
| 1 | 18 |
| 36 | 17 |
| 37 | 7 |
| 41 | 3 |
| 40 | 3 |
| 38 | 3 |
| 39 | 1 |
+---------+----------+
42 rows in set


Q14
+-------------------+
| promo_revenue |
+-------------------+
| 16.65118731292792 |
+-------------------+

Q15
+-----------+--------------------+-----------------------------------+----
-------------+---------------+
| s_suppkey | s_name | s_address | s_phone | total_revenue |
+-----------+--------------------+-----------------------------------+----
-------------+---------------+
| 7895 | Supplier#000007895 | NYl,i8UhxTykLxGJ2voIRn20Ugk1KTzz | 14-559-
808-3306 | 1678635.2636 |
+-----------+--------------------+-----------------------------------+----
-------------+---------------+
```

```
Q16
+----------+--------------------------+--------+--------------+
| p_brand | p_type | p_size | supplier_cnt |
+----------+--------------------------+--------+--------------+
| Brand#55 | LARGE BURNISHED TIN | 21 | 36 |
| Brand#25 | PROMO BRUSHED STEEL | 24 | 28 |
| Brand#54 | STANDARD BRUSHED COPPER | 14 | 27 |
| Brand#12 | MEDIUM PLATED BRASS | 21 | 24 |
| Brand#14 | ECONOMY PLATED TIN | 33 | 24 |
| Brand#24 | ECONOMY PLATED TIN | 33 | 24 |
| Brand#25 | MEDIUM PLATED STEEL | 35 | 24 |
| Brand#32 | MEDIUM POLISHED COPPER | 20 | 24 |
| Brand#32 | SMALL ANODIZED BRASS | 7 | 24 |
| Brand#33 | ECONOMY PLATED STEEL | 7 | 24 |
| Brand#33 | MEDIUM PLATED COPPER | 20 | 24 |
| Brand#33 | PROMO POLISHED STEEL | 14 | 24 |
...
| Brand#31 | PROMO ANODIZED COPPER | 20 | 3 |
| Brand#41 | LARGE BURNISHED STEEL | 20 | 3 |
| Brand#43 | SMALL BRUSHED COPPER | 7 | 3 |
| Brand#52 | MEDIUM POLISHED BRASS | 21 | 3 |
| Brand#52 | SMALL POLISHED TIN | 2 | 3 |
+----------+--------------------------+--------+--------------+
18341 rows in set

Q17
+-------------------+
| avg_yearly |
+-------------------+
| 348406.0542857143 |
+-------------------+

Q18
+--------------------+-----------+------------+-------------+------------
--+-----------------+
| c_name | c_custkey | o_orderkey | o_orderdate | o_totalprice |
sum(l_quantity) |
+--------------------+-----------+------------+-------------+------------
--+-----------------+
| Customer#000128120 | 128120 | 4722021 | 1994-04-07 | 544089.09 | 323.00
|
| Customer#000144617 | 144617 | 3043270 | 1997-02-12 | 530604.44 | 317.00
|
```

```
| Customer#000066790 | 66790 | 2199712 | 1996-09-30 | 515531.82 | 327.00
|
| Customer#000015619 | 15619 | 3767271 | 1996-08-07 | 480083.96 | 318.00
|
| Customer#000147197 | 147197 | 1263015 | 1997-02-02 | 467149.67 | 320.00
|
| Customer#000117919 | 117919 | 2869152 | 1996-06-20 | 456815.92 | 317.00
|
| Customer#000126865 | 126865 | 4702759 | 1994-11-07 | 447606.65 | 320.00
|
| Customer#000036619 | 36619 | 4806726 | 1995-01-17 | 446704.09 | 328.00
|
| Customer#000119989 | 119989 | 1544643 | 1997-09-20 | 434568.25 | 320.00
|
+--------------------+-----------+------------+-------------+------------
--+-----------------+
9 rows in set

Q19
+--------------+
| revenue |
+--------------+
| 3083843.0578 |
+--------------+

Q20
+--------------------+-----------------------------------------+
| s_name | s_address |
+--------------------+-----------------------------------------+
| Supplier#000000035 | QymmGXxjVVQ5OuABCXVVsu,4eF gU0Qc6 |
| Supplier#000000068 | Ue6N50wH2CwE4PPgTGLmat,ibGYYlDoOb3xQwtgb |
| Supplier#000000080 | cJ2MHSEJ13rIL2Wj3D5i6hRo30,ZiNUXhqn |
| Supplier#000000100 | rIlN li8zvW22l2slbcx ECP4fL |
| Supplier#000000274 | usxbl9KSW41DTE6FAglxHU |
| Supplier#000000406 | zMhU58CDF4aHTeodxg9IgRZgq |
| Supplier#000000444 | mHr2VcUpRkvyQ9rjKMaPkeWbVZmEIhxhb8F |
| Supplier#000000453 | bpt98PxU5HSQt61bVB695JPjBmJKUv hNzQeHvC |
| Supplier#000000458 | IFNkUK1H53HwUHabiONkMFAUDb |
| Supplier#000000622 | gCQimU1jYHoQiglDmW1FkQM9wzi YC1P15pMy1 |
| Supplier#000000713 | DBMIf1HiYY8OyRFcbtHpKIz |
| Supplier#000000767 | bHEuqKKdmCMEKOV |
| Supplier#000000776 | nklfFoSkCwf,ooSuF |
| Supplier#000000823 | gC0DrEG5U,v893fp3nj mmXa6rYhJ0tjpJ |
| Supplier#000000828 | 0B2aPqJ6KTEr2fqxuC7z |
```

```
| Supplier#000000941 | gqG2XEnVlzUhjjfQGYGlwk,jcaNsplI8Rleg |
| Supplier#000000973 | 5 nhBZ 03rG6EcOEDkZXvt |
| Supplier#000000984 | 6H6qqye iYbYzCmwWhj |
| Supplier#000001149 | Nuno37wiZOjNGHF |
| Supplier#000001201 | Seh4D7pi9UdK,XQkF46A0O2N |
| Supplier#000001309 | 72RNUzKzbniUnnsSs24ZzGDvmcv2Pd |
| Supplier#000001344 | 6iF,zVDNTykohVKcb7FKvn82s74ez |
| Supplier#000001351 | zXdoBMmmRx1wOD7GKoHHBtemXGuYKLDb,U2KP |
| Supplier#000001391 | hkWoAM561QlLjBNk,SdFdIgFx |
| Supplier#000001481 | ARqVvJHMxBNKl2LrfPsR Wq9ZUXh,14 |
| Supplier#000001584 | gJbTkijteJxSMLmdzBSzeMAH |
| Supplier#000001651 | 6rJNoWL9YL |
| Supplier#000001710 | J,sdOOJwUhwPv2mrEiNEA0UZlmu5IRmgz |
| Supplier#000001755 | QstBVfnY,93NsbWXCqO |
| Supplier#000001869 | nogoCdaFQii,ri9rs3P8f5rPt1wVOMw9I7TmypxK |
| Supplier#000001895 | lywAGDbk37fYPDS |
| Supplier#000001910 | vih,zrhclXX1O9x |
| Supplier#000001930 | 2jCSw3KOLHol7y5omVO13 |
| Supplier#000001979 | UNW7nA,IC 5igvVsgUHA7OaLL,jOzUcT |
| Supplier#000002046 | BiTDgHknmvQGT6FpZXfRX,xlnR |
| Supplier#000002071 | zLH3QAtZuuOq8AoVNM |
| Supplier#000002270 | HIscbvhw8N94djn,3UbPaY4R |
| Supplier#000002350 | TWsO2iJGOl7v3vSwiscXp6X |
| Supplier#000002409 | oy39SaSQ,FIP pzLqblhxj |
| Supplier#000002520 | 5y55UzYQKByZP3 |
| Supplier#000002618 | 3UtbE1kKm29kKyx09hSEBMhRLM |
| Supplier#000002643 | eDN6YjGtp2dcj0IF,BKEEYjElO,sUjjcNI |
| Supplier#000002649 | agDQi9iCt1cUaS |
| Supplier#000002655 | i6v8dkQBuK0NSCeqQCE8 |
| Supplier#000002812 | Q9sO3wZkBU5QBe0VITRWShv |
| Supplier#000002888 | 3AtRoxBFh6HIBa9kdBX,6,Ml2SZGUA |
| Supplier#000002910 | nlH1gjApxHkQe5SU4iVZwi2xWk88wwhTWRkSvOBB |
| Supplier#000002914 | fUC4IkGB8pt1S |
| Supplier#000003000 | JtDvRf4iWHJkj54PYxl |
| Supplier#000003011 | vfL mV0MTdyozfRIPZkJbM1Z7Lcm2NCPIj6qSgBz |
| Supplier#000003038 | F5Tz7P juuCbABDuW8JGomRFxqVHBWyQrsLwg4i |
| Supplier#000003150 | XwSjsmzEnANK,wAQUp4Xf5xJDqR |
| Supplier#000003305 | GLZJimfuzKoQcqcv4 |
| Supplier#000003394 | R6D7n3WrQjWNGSQTb7eN ,X0oCMkhyuTHBOSPw |
| Supplier#000003452 | 7tMycIKhE,pe4OL3Du |
| Supplier#000003666 | ENS fE9iSrSzw,iTwA,zGorkflw |
| Supplier#000003698 | lnSEu64ca4B53BfznJPg |
| Supplier#000003773 | UWjSotAjkAD |
| Supplier#000003837 | SYXpXaKop3 |
```

```
| Supplier#000003846 | wl076KfcEpYLRegb1LfIf93b3n5HBabFK2R,mEM |
| Supplier#000003862 | 0XXFhF1IDBh |
| Supplier#000003868 | 5aP4VBn0t666NbGYB |
| Supplier#000003880 | DZo80mSznrhCpb8 |
| Supplier#000003955 | piECPB8qbn7s3XP |
| Supplier#000004007 | cvlSgCCKGOwpaB iFIPx4vU2qA5b6K hz9Z91 |
| Supplier#000004066 | TNBnJFDScUmsjBy6pSWTS sfMg9jpfKx |
| Supplier#000004127 | EduKm3NcCc75Cd |
| Supplier#000004174 | Bk97olQYwXmjYdQjwyt N |
| Supplier#000004328 | euddbWZRcVMD3W |
| Supplier#000004341 | ea8KZYvO7amq8A |
| Supplier#000004360 | w 7kM5J,fqjiqBu4SU0UPEDqspaUEm |
| Supplier#000004375 | Cmr952zcJJuW0xAYc0W0MA7N6vMcCjy |
| Supplier#000004391 | pcsiJBhSEHuFHNAxR3K c |
| Supplier#000004398 | khZZ0CmLip49Zncec |
| Supplier#000004402 | acagGfDWzwmS,,WVBsszubFs3LOA8rDRS0I |
| Supplier#000004714 | IKRla2xArMmR4p3Mbn8JV8g0 |
| Supplier#000004717 | H,Suh5pN230Ol,ggx0QEh3rrvzyQsq050Lat |
| Supplier#000004740 | yM0TXkhfjpObafbQhuWU |
| Supplier#000004763 | W 7kS9LLh4ZgLpk2 |
| Supplier#000004837 | tYHMZS4XlJjzvj34mH2PCoj |
| Supplier#000004882 | e,V Bo1KZEt |
| Supplier#000004913 | em,yC41xEl Fst9LwEik |
| Supplier#000005005 | K6 GI4WzmbsGEOh |
| Supplier#000005238 | jmtI76 8RNG8Z2BZu |
| Supplier#000005289 | 62XeOur9SnXgbdjGwb9E1aJIEBr5PA9 |
| Supplier#000005317 | lPOPHufNjwZaUJGVNHCC2DE FYQcKZBzHltL5 |
| Supplier#000005401 | eEOlCEAaIfVexStlrgTuzwQx7vjPF6ZT dm |
| Supplier#000005449 | fhc8lUuZdqWUujcVaWogowEq1WVL9Y8m1efwCl3G |
| Supplier#000005472 | LlyLSmvY9GFvMN4QhHzMokW0k5d |
| Supplier#000005572 | o0VYozeSbEyqck |
| Supplier#000005579 | ACVEMP4IwRf |
| Supplier#000005661 | pq5wuxmkIW0DyWU |
| Supplier#000005676 | HInJHZisl5svSU1oKsr |
| Supplier#000005815 | S6cu6cspYxHlTz2 |
| Supplier#000005835 | rYoXzV3EZ77Z |
| Supplier#000006103 | l32l8iaPdbHgRXoq,kdjFAj3hZk2d |
| Supplier#000006173 | hBdratcVfL4LpWxsEpCRP g0AksN0CDhBZ |
| Supplier#000006226 | CKuDyeGAxPHeRHwC4a |
| Supplier#000006254 | g7OY1vWNUb1vxIRgEl |
| Supplier#000006348 | f2KDn2rLnadX8I DZR |
| Supplier#000006359 | QyUuVHYBp8sTd7Y9WveNfsz |
| Supplier#000006430 | F2RrkeaNcs6po8x2PyYvcPa1rtKd,fT2AMxP |
| Supplier#000006516 | 89XwFOC,hLRxGq5rL0txv0EM9F |
```

```
| Supplier#000006700 | BWjerJH5kbEPu 8h9 |
| Supplier#000006785 | lyo6PpwulTeN9ZfIkvWag5NucL,XMC 89Kn7U |
| Supplier#000006998 | r2i3HfkSQh9dvho, NpoabdMsPBG |
| Supplier#000007019 | 2GQsALzRiTt2BQum6bocdeGawkOrsjNIZ |
| Supplier#000007114 | s9s4YLeLWo7fLRO3rdQKFfUnZhrZUPjOC |
| Supplier#000007170 | 9vABqu hZaciXSCQrbTj |
| Supplier#000007171 | DXerxFIhNRpqF9dWNRw hDOlLX gEJFxh0 |
| Supplier#000007213 | 2Nrby3JJHDJyWwVNiqPtm2U JGWlZpU |
| Supplier#000007219 | p5Ui3IGPcmotYu |
| Supplier#000007229 | iwNoWdaURFzLAsQHxK,BeOPpI5TOTo |
| Supplier#000007263 | malQPdYc8xiup2MiFuKHa |
| Supplier#000007270 | TksERECGdYZRPUjkUdDRZv5pW26cOTaA1 |
| Supplier#000007276 | Vi9,aBg2ychZf |
| Supplier#000007334 | NPXYWdJ8L9EDr20tw9CZQsEMqXlgXzI2JC Y |
| Supplier#000007400 | 7r9zZj8J,,hN2GRfWtDxzuGa |
| Supplier#000007442 | DzycM1,T6kh2EutfPeFpv0Ro |
| Supplier#000007456 | ITYEeccPVJi0HvnAwVs2Z |
| Supplier#000007559 | Wmzx1vskciC |
| Supplier#000007677 | OoTYQdxQyd7NukSaSRv |
| Supplier#000007712 | DyTQD 3ajuOtHQTpI4LsWSF kSd2SE6U4COgYHQ |
| Supplier#000007715 | gZHd7Yzbtv7yb7DYCCAQPJH8FRHTqi6T4w |
| Supplier#000007816 | 1ejcJ545bwLWLuY6Qq4qyEExZIsp0SG |
| Supplier#000007845 | agwGVTzLyRKOsZxLVi,mPWZ08Qxb |
| Supplier#000007875 | E0CkoBYngcIoH |
| Supplier#000007908 | ghhHapj7GK |
| Supplier#000007972 | WW0GuiWP2N3kUo4f |
| Supplier#000008162 | XASpbn08mRV0kgHRmUSKx |
| Supplier#000008235 | TjVWq6bTdGJB |
| Supplier#000008249 | PwUjvlMk y72zaMRtZQ8trbCmu4j |
| Supplier#000008309 | 6P,FQbW6sJouqunvttVO6vEeY |
| Supplier#000008339 | uWw8 P6u,S |
| Supplier#000008343 | BbHngAVqj0J8 |
| Supplier#000008349 | 8Hkx1IDd0mZCTX |
| Supplier#000008377 | ,Yk0mflw2LqQCTxMYR sU2juj5DorUAG4w6i |
| Supplier#000008468 | 5R4jsweitleustYlE3w,u5otW |
| Supplier#000008523 | C4ocdfNu5I2nnnVG2xSd3016J6KNLIg |
| Supplier#000008580 | t5ri71bM6Sox3riP4JUZsMMNC |
| Supplier#000008638 | yxj50B 8aMql |
| Supplier#000008642 | qnN9N9du9Dg2arf6kjD xW0DjMT9cM |
| Supplier#000008651 | pfw32RGA7BPXrUiavYqE |
| Supplier#000008679 | JWFVoSsCwn9p8o |
| Supplier#000008704 | a6DjHp0B6mifKBtqUk,C |
| Supplier#000008737 | MsdGxF9Xoq9 8s |
| Supplier#000008820 | uAsBvPBNsEsO |
```

```
| Supplier#000008829 | lNcY7xNLDonCw TuRYL |
| Supplier#000008947 | 1Ij3T0egGHnVbLich98HzY,UeCdVbxzYa ZpKDVc |
| Supplier#000008964 | U2YJW,Y1xCbUWbjuovtzsLfsl |
| Supplier#000008974 | 4JCXOJ3MyPfa51mIf,MQu |
| Supplier#000008997 | KY MmMEcyQ6FEDCooFj xa uCwF2GbaeA8 |
| Supplier#000009065 | ZELuiqWrWbJV9zAuco1OnXKTJClhR |
| Supplier#000009114 | nkn6bcPvlP5w,lUpO0nZTBSj |
| Supplier#000009125 | IQbCXbN1mmght |
| Supplier#000009131 | gDBXgWtg4rTxu0WUJhhV |
| Supplier#000009149 | yKX,bKryD6YtvF,cVLIKC0Z6rN |
| Supplier#000009182 | z56kNgeqaWQ1kHFBp |
| Supplier#000009220 | N4y,vP kdArpcmdypBh,fJVVB |
| Supplier#000009226 | yzT10vNTFJ |
| Supplier#000009288 | 251AA4ziZ3d7TTWXLGnXjb4BnXv |
| Supplier#000009360 | 1NVjjX8zMjyBX2UapDTP0Sz |
| Supplier#000009381 | rhCTm7QehIznqd8 Np7VT,H5J5zSGr |
| Supplier#000009403 | 70841REghyWBrHyyg762Jh4sjCG7CKaIc |
| Supplier#000009504 | Rqt07,ANI92kj1oU |
| Supplier#000009598 | PnTAz7rNRLVDFO3zoo2QRTlh4o |
| Supplier#000009609 | LV2rJUGfr0k3dPNRqufG1IoYHzV |
| Supplier#000009619 | K0RwcJ9S75Xil jqKukFoDNkD |
| Supplier#000009626 | Nm1FnIh4asUR3EnXv2Pvy3gXqI9es |
| Supplier#000009738 | 15RRSVTuOzwdMP LmfCtIguMGXK |
| Supplier#000009770 | Ag, SZfowit580QPDdbP8kmFHdpZ9ASI |
| Supplier#000009865 | extcOh9ZrdDCMsHhhsFTkTUAh,HM2UQ2qa8sRo |
| Supplier#000009866 | Auh6aZnOnQG1pPYKZ5o9ATramJBA |
| Supplier#000009890 | izJXemCM Ikpgxk |
| Supplier#000009937 | edZ9HQJ0KJAU6EWknTiDghKfRLHq6vtFqdey,0l |
| Supplier#000009954 | VzElx9ihlXFJLIQw2Hn4bC2 |
| Supplier#000009958 | ggiiSA4CSyvhwQUYjdJhWlKEY9PAfs |
+--------------------+----------------------------------------+
177 rows in set


Q21
+--------------------+---------+
| s_name | numwait |
+--------------------+---------+
| Supplier#000009302 | 21 |
| Supplier#000000342 | 20 |
| Supplier#000000632 | 19 |
| Supplier#000002196 | 19 |
| Supplier#000003325 | 18 |
| Supplier#000003915 | 18 |
```

```
| Supplier#000005045 | 18 |
| Supplier#000006442 | 18 |
| Supplier#000003093 | 17 |
| Supplier#000004498 | 17 |
| Supplier#000000906 | 16 |
| Supplier#000001183 | 16 |
| Supplier#000001477 | 16 |
| Supplier#000006043 | 16 |
| Supplier#000000689 | 15 |
| Supplier#000001955 | 15 |
| Supplier#000002066 | 15 |
| Supplier#000002146 | 15 |
| Supplier#000003253 | 15 |
| Supplier#000003527 | 15 |
| Supplier#000003947 | 15 |
| Supplier#000004915 | 15 |
| Supplier#000005248 | 15 |
| Supplier#000006718 | 15 |
| Supplier#000007773 | 15 |
| Supplier#000008121 | 15 |
| Supplier#000008169 | 15 |
| Supplier#000008645 | 15 |
| Supplier#000008684 | 15 |
| Supplier#000009079 | 15 |
| Supplier#000009956 | 15 |
| Supplier#000000737 | 14 |
| Supplier#000000775 | 14 |
| Supplier#000001474 | 14 |
| Supplier#000001502 | 14 |
| Supplier#000003196 | 14 |
| Supplier#000004415 | 14 |
| Supplier#000004940 | 14 |
| Supplier#000005253 | 14 |
| Supplier#000005703 | 14 |
| Supplier#000006308 | 14 |
| Supplier#000006789 | 14 |
| Supplier#000007161 | 14 |
| Supplier#000007952 | 14 |
| Supplier#000008062 | 14 |
| Supplier#000008414 | 14 |
| Supplier#000008442 | 14 |
| Supplier#000008508 | 14 |
| Supplier#000000300 | 13 |
| Supplier#000000727 | 13 |
```

```
| Supplier#000000921 | 13 |
| Supplier#000000992 | 13 |
| Supplier#000001282 | 13 |
| Supplier#000001582 | 13 |
| Supplier#000001662 | 13 |
| Supplier#000001683 | 13 |
| Supplier#000002933 | 13 |
| Supplier#000003177 | 13 |
| Supplier#000003428 | 13 |
| Supplier#000003640 | 13 |
| Supplier#000004842 | 13 |
| Supplier#000004951 | 13 |
| Supplier#000005795 | 13 |
| Supplier#000005981 | 13 |
| Supplier#000006118 | 13 |
| Supplier#000006433 | 13 |
| Supplier#000006484 | 13 |
| Supplier#000007268 | 13 |
| Supplier#000008599 | 13 |
| Supplier#000008675 | 13 |
| Supplier#000009474 | 13 |
| Supplier#000009521 | 13 |
| Supplier#000009853 | 13 |
| Supplier#000000021 | 12 |
| Supplier#000000211 | 12 |
| Supplier#000000743 | 12 |
| Supplier#000000951 | 12 |
| Supplier#000001654 | 12 |
| Supplier#000001868 | 12 |
| Supplier#000002089 | 12 |
| Supplier#000002879 | 12 |
| Supplier#000003060 | 12 |
| Supplier#000003215 | 12 |
| Supplier#000003365 | 12 |
| Supplier#000003873 | 12 |
| Supplier#000003985 | 12 |
| Supplier#000004452 | 12 |
| Supplier#000004639 | 12 |
| Supplier#000005122 | 12 |
| Supplier#000005633 | 12 |
| Supplier#000005671 | 12 |
| Supplier#000005782 | 12 |
| Supplier#000006088 | 12 |
| Supplier#000006477 | 12 |
```

```
| Supplier#000006508 | 12 |
| Supplier#000006750 | 12 |
| Supplier#000006802 | 12 |
| Supplier#000008236 | 12 |
| Supplier#000009294 | 12 |
| Supplier#000009329 | 12 |
+--------------------+---------+
100 rows in set

Q22
+-----------+---------+------------+
| cntrycode | numcust | totacctbal |
+-----------+---------+------------+
| 10 | 882 | 6606081.31 |
| 11 | 899 | 6702253.34 |
| 19 | 963 | 7230776.82 |
| 20 | 916 | 6824676.02 |
| 22 | 894 | 6636740.03 |
| 26 | 861 | 6404695.86 |
| 27 | 877 | 6565078.99 |
+-----------+---------+------------+
7 rows in set
```

完成 TPC-C 测试

通过阅读本教程，你将学习如何使用 MatrixOne 完成 TPC-C 测试。

TPC-C 概述

TPC-C 是一种衡量在线事务处理（OLTP）系统性能和可伸缩性的基准测试规范。TPC-C 模拟了一个比较有代表意义的 OLTP 应用环境：在线订单处理系统。TPC-C 基准测试中使用的数据库由 Warehouse、Customer、Order、Item 等九个表组成（参见下图）。除 Item 表外，每条记录都以每个 Warehouse 为基础进行填充，并且仓库的数量按比例配置。

TPCC diagram

TPC-C 需要处理的交易事务有五种：NewOrder、Payment、OrderStatus、Delivery 和 StockLevel。TPC-C 定义了每种事务的请求率，其中几乎 90% 请求率的事务为写密集型的 NewOrder 和 Payment。TPC-C 事务主要访问单个（本地）仓库，但大约 10% 的事务与另一个（远程）仓库交互。

开始前准备

安装并启动 MatrixOne

确保你已经完成了[单机部署 MatrixOne](#)。

克隆 mo-tpcc 仓库到本地

```
git clone https://github.com/matrixorigin/mo-tpcc.git
```

步骤

步骤简介

本节将介绍如何生成 TPCC 数据、创建 TPCC 表，并将数据加载到 MatrixOne 以及运行 TPCC 测试。

现在你可以按照以下描述逐步执行命令。

1. 配置 *props.mo* 文件

克隆 *mo-tpch* 仓库到本地后，在本地打开 *mo-tpch* 文件夹，按照下面的配置项，修改文件夹中的 *props.mo* 文件。数据仓库的数量可以通过该文件中的 `warehouse=XX` 行进行配置。

```
db=mo
driver=com.mysql.cj.jdbc.Driver
conn=jdbc:mysql://127.0.0.1:6001/tpcc?
characterSetResults=utf8&continueBatchOnError=false&useServerPrepStmts=true&
user=dump
password=111

//the number of warehouse
warehouses=10
loadWorkers=4

//the num of terminals that will simultaneously run
//must be less than warehouses*10
terminals=1
//To run specified transactions per terminal- runMins must equal zero
runTxnsPerTerminal=0
//To run for specified minutes- runTxnsPerTerminal must equal zero
runMins=1
//Number of total transactions per minute
limitTxnsPerMin=0
```

修改完成后，保存 *props.mo* 文件。

2. 创建 TPCC 数据库和表

打开一个新的终端，执行下面的代码：

```
cd mo-tpcc
./runSQL.sh props.mo tableCreates
```

Note: 如果在运行 `./runSQL.sh props.mo tableCreates` 时产生 `java:command not found` 报错，那么你需要在你计算机上安装或重新安装 Java 和 JDK。

上面的代码表示，进入到 *mo-tpch* 文件夹目录，执行代码创建完成 TPCC 数据库和表。

执行完成后，输出结果示例如下：

```
# ------------------------------------------------------------
# Loading SQL file ./sql/tableCreates.sql
# ------------------------------------------------------------
drop database if exists tpcc;
create database if not exists tpcc;
use tpcc;
create table bmsql_config (
cfg_name varchar(30) primary key,
cfg_value varchar(50)
);
create table bmsql_warehouse (
w_id integer not null,
w_ytd decimal(12,2),
w_tax decimal(4,4),
w_name varchar(10),
w_street_1 varchar(20),
w_street_2 varchar(20),
w_city varchar(20),
w_state char(2),
w_zip char(9),
primary key (w_id)
) PARTITION BY KEY(w_id);
create table bmsql_district (
d_w_id integer not null,
d_id integer not null,
d_ytd decimal(12,2),
d_tax decimal(4,4),
d_next_o_id integer,
d_name varchar(10),
d_street_1 varchar(20),
d_street_2 varchar(20),
d_city varchar(20),
d_state char(2),
d_zip char(9),
primary key (d_w_id, d_id)
) PARTITION BY KEY(d_w_id);
create table bmsql_customer (
c_w_id integer not null,
c_d_id integer not null,
c_id integer not null,
c_discount decimal(4,4),
c_credit char(2),
c_last varchar(16),
c_first varchar(16),
```

```
c_credit_lim decimal(12,2),
c_balance decimal(12,2),
c_ytd_payment decimal(12,2),
c_payment_cnt integer,
c_delivery_cnt integer,
c_street_1 varchar(20),
c_street_2 varchar(20),
c_city varchar(20),
c_state char(2),
c_zip char(9),
c_phone char(16),
c_since timestamp,
c_middle char(2),
c_data varchar(500),
primary key (c_w_id, c_d_id, c_id)
) PARTITION BY KEY(c_w_id);
create table bmsql_history (
hist_id integer auto_increment,
h_c_id integer,
h_c_d_id integer,
h_c_w_id integer,
h_d_id integer,
h_w_id integer,
h_date timestamp,
h_amount decimal(6,2),
h_data varchar(24),
primary key (hist_id)
);
create table bmsql_new_order (
no_w_id integer not null,
no_d_id integer not null,
no_o_id integer not null,
primary key (no_w_id, no_d_id, no_o_id)
) PARTITION BY KEY(no_w_id);
create table bmsql_oorder (
o_w_id integer not null,
o_d_id integer not null,
o_id integer not null,
o_c_id integer,
o_carrier_id integer,
o_ol_cnt integer,
o_all_local integer,
o_entry_d timestamp,
primary key (o_w_id, o_d_id, o_id)
```

```
) PARTITION BY KEY(o_w_id);
create table bmsql_order_line (
ol_w_id integer not null,
ol_d_id integer not null,
ol_o_id integer not null,
ol_number integer not null,
ol_i_id integer not null,
ol_delivery_d timestamp,
ol_amount decimal(6,2),
ol_supply_w_id integer,
ol_quantity integer,
ol_dist_info char(24),
primary key (ol_w_id, ol_d_id, ol_o_id, ol_number)
) PARTITION BY KEY(ol_w_id);
create table bmsql_item (
i_id integer not null,
i_name varchar(24),
i_price decimal(5,2),
i_data varchar(50),
i_im_id integer,
primary key (i_id)
) PARTITION BY KEY(i_id);
create table bmsql_stock (
s_w_id integer not null,
s_i_id integer not null,
s_quantity integer,
s_ytd integer,
s_order_cnt integer,
s_remote_cnt integer,
s_data varchar(50),
s_dist_01 char(24),
s_dist_02 char(24),
s_dist_03 char(24),
s_dist_04 char(24),
s_dist_05 char(24),
s_dist_06 char(24),
s_dist_07 char(24),
s_dist_08 char(24),
s_dist_09 char(24),
s_dist_10 char(24),
primary key (s_w_id, s_i_id)
) PARTITION BY KEY(s_w_id);
```

3. 生成 TPCC 数据集

执行下面的代码，生成 TPCC 数据集：

```
./runLoader.sh props.mo filelocation /yourpath/
```

执行完成后，输出结果示例如下：

```
Starting BenchmarkSQL LoadData

props.mo
driver=com.mysql.cj.jdbc.Driver
conn=jdbc:mysql://127.0.0.1:6001/tpcc?
characterSetResults=utf8&continueBatchOnError=false&useServerPrepStmts=true&
user=dump
password=***********
warehouses=10
loadWorkers=4
fileLocation (not defined)
csvNullValue (not defined - using default '')

Worker 000: Loading ITEM
Worker 001: Loading Warehouse 1
Worker 002: Loading Warehouse 2
Worker 003: Loading Warehouse 3
Worker 000: Loading ITEM done
Worker 000: Loading Warehouse 4
Worker 003: Loading Warehouse 3 done
Worker 003: Loading Warehouse 5
Worker 001: Loading Warehouse 1 done
Worker 001: Loading Warehouse 6
Worker 002: Loading Warehouse 2 done
Worker 002: Loading Warehouse 7
Worker 000: Loading Warehouse 4 done
Worker 000: Loading Warehouse 8
Worker 003: Loading Warehouse 5 done
Worker 003: Loading Warehouse 9
Worker 000: Loading Warehouse 8 done
Worker 000: Loading Warehouse 10
Worker 002: Loading Warehouse 7 done
Worker 001: Loading Warehouse 6 done
Worker 000: Loading Warehouse 10 done
Worker 003: Loading Warehouse 9 done
```

你会在你所指定路径中找到 10 个 *csv* 文件，每个 *csv* 文件都会映射到第 2 步中创建的表中。

```
config.csv
cust-hist.csv
customer.csv
district.csv
item.csv
new-order.csv
order-line.csv
order.csv
stock.csv
warehouse.csv
```

4. 将 TPCC 数据加载到 MatrixOne

使用 MySQL 客户端连接到 MatrixOne 并执行以下语句将 *csv* 文件加载到 MatrixOne 中。

```
mysql> load data infile '/yourpath/config.csv' INTO TABLE bmsql_config
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY
"\r\n";
load data infile '/yourpath/cust-hist.csv' INTO TABLE bmsql_history
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY
"\r\n";
load data infile '/yourpath/data/customer.csv' INTO TABLE bmsql_customer
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY
"\r\n";
load data infile '/yourpath/data/district.csv' INTO TABLE bmsql_district
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY
"\r\n";
load data infile '/yourpath/data/warehouse.csv' INTO TABLE
bmsql_warehouse FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES
TERMINATED BY "\r\n";
load data infile '/yourpath/item.csv' INTO TABLE bmsql_item FIELDS
TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY "\r\n";
load data infile '/yourpath/new-order.csv' INTO TABLE bmsql_new_order
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY
"\r\n";
load data infile '/yourpath/order-line.csv' INTO TABLE bmsql_order_line
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY
"\r\n";
load data infile '/yourpath/stock.csv' INTO TABLE bmsql_stock FIELDS
TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY "\r\n";
load data infile '/yourpath/order.csv' INTO TABLE bmsql_oorder FIELDS
TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY "\r\n";
```

5. 运行 TPCC 测试

执行下面的代码，运行 TPCC 测试：

```
./runBenchmark.sh props.mo
```

执行完成后，输出结果示例如下：

```
.:./lib/*
2022-12-22 21:15:35 INFO jTPCC:78 - Term-00,
2022-12-22 21:15:35 INFO jTPCC:79 - Term-00, +-------------------------
----------------------------------+
2022-12-22 21:15:35 INFO jTPCC:80 - Term-00, BenchmarkSQL v5.0
2022-12-22 21:15:35 INFO jTPCC:81 - Term-00, +-------------------------
----------------------------------+
2022-12-22 21:15:35 INFO jTPCC:82 - Term-00, (c) 2003, Raul Barbosa
2022-12-22 21:15:35 INFO jTPCC:83 - Term-00, (c) 2004-2016, Denis Lussier
2022-12-22 21:15:35 INFO jTPCC:84 - Term-00, (c) 2016, Jan Wieck
2022-12-22 21:15:35 INFO jTPCC:85 - Term-00, +-------------------------
----------------------------------+
2022-12-22 21:15:35 INFO jTPCC:86 - Term-00,
2022-12-22 21:15:35 INFO jTPCC:63 - Term-00, db=mo
2022-12-22 21:15:35 INFO jTPCC:63 - Term-00,
driver=com.mysql.cj.jdbc.Driver
2022-12-22 21:15:35 INFO jTPCC:63 - Term-00,
conn=jdbc:mysql://127.0.0.1:6001/tpcc?
characterSetResults=utf8&continueBatchOnError=false&useServerPrepStmts=true&
2022-12-22 21:15:35 INFO jTPCC:63 - Term-00, user=dump
2022-12-22 21:15:35 INFO jTPCC:93 - Term-00,
2022-12-22 21:15:35 INFO jTPCC:63 - Term-00, warehouses=10
2022-12-22 21:15:35 INFO jTPCC:63 - Term-00, terminals=1
2022-12-22 21:15:35 INFO jTPCC:100 - Term-00, runMins=1
2022-12-22 21:15:35 INFO jTPCC:63 - Term-00, limitTxnsPerMin=0
2022-12-22 21:15:35 INFO jTPCC:63 - Term-00, terminalWarehouseFixed=false
2022-12-22 21:15:35 INFO jTPCC:108 - Term-00,
2022-12-22 21:15:35 INFO jTPCC:63 - Term-00, newOrderWeight=45
2022-12-22 21:15:35 INFO jTPCC:63 - Term-00, paymentWeight=43
2022-12-22 21:15:35 INFO jTPCC:63 - Term-00, orderStatusWeight=4
2022-12-22 21:15:35 INFO jTPCC:63 - Term-00, deliveryWeight=4
2022-12-22 21:15:35 INFO jTPCC:63 - Term-00, stockLevelWeight=4
2022-12-22 21:15:35 INFO jTPCC:115 - Term-00,
2022-12-22 21:15:35 INFO jTPCC:63 - Term-00,
resultDirectory=my_result_%tY-%tm-%td_%tH%tM%tS
2022-12-22 21:15:35 INFO jTPCC:63 - Term-00, osCollectorScript=null
2022-12-22 21:15:35 INFO jTPCC:119 - Term-00,
2022-12-22 21:15:35 INFO jTPCC:710 - Term-00, Loading database driver:
'com.mysql.cj.jdbc.Driver'...
2022-12-22 21:15:35 INFO jTPCC:219 - Term-00, copied props.mo to
my_result_2022-12-22_211535/run.properties
2022-12-22 21:15:35 INFO jTPCC:239 - Term-00, created my_result_2022-12-
22_211535/data/runInfo.csv for runID 1
2022-12-22 21:15:35 INFO jTPCC:255 - Term-00, writing per transaction
```

```
results to my_result_2022-12-22_211535/data/result.csv
2022-12-22 21:15:35 INFO jTPCC:268 - Term-00,
2022-12-22 21:15:36 INFO jTPCC:324 - Term-00, C value for C_LAST during
load: 28
2022-12-22 21:15:36 INFO jTPCC:325 - Term-00, C value for C_LAST this
run: 132
2022-12-22 21:15:36 INFO jTPCC:326 - Term-00,
2022-12-22 21:15:36 INFO jTPCC:710 - Term-00, Session started! Memory
Usage: 17MB / 245MB
2022-12-22 21:15:36 INFO jTPCC:710 - Term-00, Creating 1 terminal(s) with
-1 transaction(s) per terminal...
2022-12-22 21:15:36 INFO jTPCC:710 - Term-00, Terminal Warehouse is NOT
fixed
2022-12-22 21:15:36 INFO jTPCC:710 - Term-00, Transaction Weights: 45%
New-Order, 43% Payment, 4% Order-Status, 4% Delivery, 4% Stock-Level
2022-12-22 21:15:36 INFO jTPCC:710 - Term-00, Number of Terminals 1
2022-12-22 21:15:36 INFO jTPCC:710 - Term-00, Creating database
connection for Term-01...
2022-12-22 21:15:36 INFO jTPCC:710 - Term-00, Term-01 7
2022-12-22 21:15:36 INFO jTPCC:710 - Term-00, Transaction Weight
2022-12-22 21:15:36 INFO jTPCC:710 - Term-00, % New-Order 45
2022-12-22 21:15:36 INFO jTPCC:710 - Term-00, % Payment 43
2022-12-22 21:15:36 INFO jTPCC:710 - Term-00, % Order-Status 4
2022-12-22 21:15:36 INFO jTPCC:710 - Term-00, % Delivery 4
2022-12-22 21:15:36 INFO jTPCC:710 - Term-00, % Stock-Level 4
2022-12-22 21:15:36 INFO jTPCC:710 - Term-00, Transaction Number Terminal
Type Execution Time (ms) Comment
2022-12-22 21:15:36 INFO jTPCC:710 - Term-00, Created 1 terminal(s)
successfully!
2022-12-22 21:15:36 INFO jTPCC:710 - Term-00, Starting all terminals...
2022-12-22 21:15:36 INFO jTPCC:710 - Term-00, All terminals started
executing 2022-12-22 21:15:36
Term-00, Running Average tpmTOTAL: 60000.00 Current tpmTOTAL: 12 Memory
Usage: 19MB / 245MB 2022-12-22 21:15:36 INFO jTPCCTerminal:350 - Term-01,
Executing for a limited time...
2022-12-22 21:16:42 INFO jTPCC:710 - Term-00, The time limit has been
reached.: 21MB / 245MB
2022-12-22 21:16:42 INFO jTPCC:710 - Term-00, Signalling all terminals to
stop...
2022-12-22 21:16:42 INFO jTPCCTerminal:350 - Term-01,
2022-12-22 21:16:42 INFO jTPCCTerminal:350 - Term-01, Terminal received
stop signal!
2022-12-22 21:16:42 INFO jTPCCTerminal:350 - Term-01, Finishing current
transaction before exit...
```

```
2022-12-22 21:16:42 INFO jTPCC:710 - Term-00, Waiting for all active
transactions to end...
2022-12-22 21:16:42 INFO jTPCCTerminal:350 - Term-01, OTAL: 24 Memory
Usage: 22MB / 245MB
2022-12-22 21:16:42 INFO jTPCCTerminal:350 - Term-01, Closing statement
and connection...
2022-12-22 21:16:42 INFO jTPCCTerminal:350 - Term-01,
2022-12-22 21:16:42 INFO jTPCCTerminal:350 - Term-01, Terminal 'Term-01'
finished after 0 transaction(s).
2022-12-22 21:16:42 INFO jTPCC:710 - Term-00, All terminals finished
executing 2022-12-22 21:16:42

2022-12-22 21:16:42 INFO jTPCC:694 - Term-00,
2022-12-22 21:16:42 INFO jTPCC:695 - Term-00,
2022-12-22 21:16:42 INFO jTPCC:696 - Term-00, Measured tpmC (NewOrders) =
2.74
2022-12-22 21:16:42 INFO jTPCC:697 - Term-00, Measured tpmTOTAL = 3.66
2022-12-22 21:16:42 INFO jTPCC:698 - Term-00, Measured tpmE (ErrorCount)
= 0.0
2022-12-22 21:16:42 INFO jTPCC:699 - Term-00, Session Start = 2022-12-22
21:15:36
2022-12-22 21:16:42 INFO jTPCC:700 - Term-00, Session End = 2022-12-22
21:16:42
2022-12-22 21:16:42 INFO jTPCC:701 - Term-00, Transaction Count = 3
2022-12-22 21:16:42 INFO jTPCC:702 - Term-00, Transaction Error = 0
2022-12-22 21:16:42 INFO jTPCC:703 - Term-00, Transaction NewOrders = 3
2022-12-22 21:16:42 INFO jTPCC:710 - Term-00, Session finished!
```

由上面的结果可以得到 tpmC（每分钟事务数）的值。
从 0.5.0 版本开始，MatrixOne 引入了一个自动测试框架 MO-Tester。

MO-Tester 测试框架，也可以称作为测试器，是通过 SQL 测试 MatrixOne 或其他数据库功能的。

MO-Tester 简介

MO-Tester 是基于 Java 语言进行开发，用于 MatrixOne 的测试套件。MO-Tester 构建了一整套完整的工具链来进行 SQL 自动测试。它包含测试用例和运行结果。MO-Tester 启动后，MO-Tester 将使用 MatrixOne 运行所有 SQL 测试用例，并将所有输出 SQL 测试结果与预期结果进行比较。所有案例的结果无论成功或者失败，都将记录在报告中。

MO-Tester 相关用例、结果和报告的放在 MatrixOne 仓库内，链接如下：

- *Cases*: https://github.com/matrixorigin/matrixone/tree/main/test/distributed/cases
- *Result*: 生成在 **/cases** 的具体测试用例下，例如 /cases/auto_increment 目录的具体测试用例同级目录下生成对应的*. result* 文件。
- *Report*: 运行结束后，本地目录自动生成 `mo-tester/report`。

测试用例和测试结果一一对应。如需添加新的测试用例和测试结果请进入右侧所示 MatrixOne 仓库路径中进行添加：
https://github.com/matrixorigin/matrixone/tree/main/test

使用 MO-Tester

1. 准备测试环境

- 请先确认已安装 jdk8。
- 启动 MatrixOne 或其他数据库用例。参见更多信息 >>安装单机版 MatrixOne.
- 克隆 *mo-tester* 仓库。

```
git clone https://github.com/matrixorigin/mo-tester.git
```

- 克隆 *matrixone* 仓库。

```
git clone https://github.com/matrixorigin/matrixone.git
```

2. 配置 MO-Tester

MO-tester 基于 Java 语言进行开发，因此 Mo-tester 所依赖的 Java 数据库连接（JDBC，Java Database Connectivity）驱动程序需要配置 *mo.yml* 文件里的参数信息：进入到 *mo-tester* 本地仓库，打开 *mo.yml* 文件，配置服务器地址、默认的数据库名称、用户名和密码等。

以下是本地独立版本 MatrixOne 的默认示例。

```
#jdbc
jdbc:
driver: "com.mysql.cj.jdbc.Driver"
server:
- addr: "127.0.0.1:6001"
database:
default: "test"
paremeter:
characterSetResults: "utf8"
continueBatchOnError: "false"
useServerPrepStmts: "true"
alwaysSendSetIsolation: "false"
useLocalSessionState: "true"
zeroDateTimeBehavior: "CONVERT_TO_NULL"
failoverReadOnly: "false"
serverTimezone: "Asia/Shanghai"
socketTimeout: 30000
#users
user:
name: "dump"
passwrod: "111"
```

3. 运行 MO-Tester

运行以下所示命令行，SQL 所有测试用例将自动运行，并将报告和错误消息生成至 *report/report.txt* 和 *report/error.txt* 文件中。

```
> ./run.sh -p \{path_name}/matrixone/test/cases
```

如果你想调整测试范围，你可以修改 `run.yml` 文件中的 `path` 参数。或者，在执行 `./run.sh` 命令时，你也可以指定一些参数，参数解释如下：

参数	参数释义
-p	设置由 MO-tester 执行的测试用例的路径。默认值可以参见 *run.yml* 文件中 `path` 的配置参数
-m	设置 MO-tester 测试的方法，即直接运行或者生成新的测试结果。默认值可以参见 *run.yaml* 文件中 `method` 的配置参数
-t	设置 MO-tester 执行 SQL 命令的格式类型。默认值可以参见 *run.yml* 文件中 `type` 的配置参数。

参数	参数释义
-r	设置测试用例应该达到的成功率。默认值可以参见 *run.yml* 文件中 `rate` 的配置参数。
-i	设置包含列表，只有路径中名称包含其中一个列表的脚本文件将被执行，如果有多个，如果没有指定，用'，'分隔，指的是包含的所有情况 set the including list, and only script files in the path whose name contains one of the lists will be executed, if more than one, separated by `,`, if not specified, refers to all cases included
-e	设置排除列表，如果路径下的脚本文件的名称包含一个排除列表，则不会被执行，如果有多个，用"分隔，如果没有指定，表示不排除任何情况 set the excluding list, and script files in the path whose name contains one of the lists will not be executed, if more than one, separated by `,`, if not specified, refers to none of the cases excluded

-g	表示带有[— @bvt
	#{issueNO.}]标志的 SQL 命令将不会被执行，该标志以 [— @bvt
	#{issueNO.}]开始，以 [— @bvt
	]结束。例如，
	— @bvt
	#3236
	
	select date_add("1997-12-31 23:59:59",INTERVAL "-10000:1" HOUR_MINUTE);
	
	select date_add("1997-12-31 23:59:59",INTERVAL "-100 1" YEAR_MONTH);
	
	— @bvt
	
	这两个 SQL 命令与问题 #3236 相关联，它们将不会在 MO-tester 测试中执行，直到问题 #3236 修复后标签移除才可以在测试中执行。

-n	表示在比较结果时将忽略结果集的元数据
-c	只需要检查 *case* 文件与 *result* 文件是否匹配

示例：

```
./run.sh -p \{path_name}/matrixone/test/cases -m run -t script -r 100 -i
select,subquery -e substring -g
```

如果你想测试新的 SQL 用例并自动生成 SQL 结果，运行命令中可以将 `-m run` 更改为 `-m genrs`，或者将 *run.yml* 文件里的 `method` 参数修改为 `genrs`，且*. result* 文件将生成在与这个新的 SQL 用例同级目录内，相关示例参见

示例 4

> **注意**
>
> 每次运行 `./run.sh` 都会覆盖 *mo-tester* 仓库内 *report/* 路径下 *error.txt*、*report.txt* 和 *success.txt* 报告文件。

4. 查看测试报告

测试完成后，*mo-tester* 仓库内将生成 *error.txt*、*report.txt* 和 *success.txt* 报告文件。

- *report.txt* 示例如下：

```
[SUMMARY] COST : 98s, TOTAL :12702, SUCCESS : 11851, FAILED :13, IGNORED
:838, ABNORAML :0, SUCCESS RATE : 99%
[\
{path_name}/matrixone/test/cases/auto_increment/auto_increment_columns.sql]
COST : 2.159s, TOTAL :185, SUCCESS :163, FAILED :0, IGNORED :22, ABNORAML
:0, SUCCESS RATE : 100%
[\
{path_name}/matrixone/test/cases/benchmark/tpch/01_DDL/01_create_table.sql]
COST : 0.226s, TOTAL :11, SUCCESS :11, FAILED :0, IGNORED :0, ABNORAML
:0, SUCCESS RATE : 100%
[\
{path_name}/matrixone/test/cases/benchmark/tpch/02_LOAD/02_insert_customer.s
COST : 0.357s, TOTAL :16, SUCCESS :16, FAILED :0, IGNORED :0, ABNORAML
:0, SUCCESS RATE : 100%
```

报告关键词	解释
TOTAL	执行的测试用例（SQL 语句）总数
SUCCESS	执行成功的测试用例（SQL 语句）总数
FAILED	执行失败的测试用例（SQL 语句）总数
IGNORED	忽略执行的测试用例（SQL 语句）总数，特指具有 `--bvt:issue` 标签的测试用例（SQL 语句）
ABNORAML	执行异常的测试用例（SQL 语句）总数，比如执行过程中被系统异常导致 MO 无法判断

报告关键词	解释
	实际结果，或者 *.result* 文件解析失败等
SUCCESS RATE	成功率，即 SUCCESS/(TOTAL - IGNORED)

- *error.txt* 示例如下：

```
[ERROR]
[SCRIPT FILE]: cases/transaction/atomicity.sql
[ROW NUMBER]: 14
[SQL STATEMENT]: select * from test_11 ;
[EXPECT RESULT]:
c d
1 1
2 2
[ACTUAL RESULT]:
c d
1 1
```

5. 测试示例

示例 1

示例描述：运行 *matrixone* 仓库内的 *test/cases* 路径下的所有测试用例。

步骤：

1. 拉取最新的 *matrixone* 远端仓库。

```
cd matrixone
git pull https://github.com/matrixorigin/matrixone.git
```

2. 先切换进入到 *mo-tester* 仓库，运行如下命令，即可运行 *matrixone* 仓库内所有的测试用例：

```
cd mo-tester
./run.sh -p \{path_name}/matrixone/test/cases
```

3. 在 *MO-Tester* 仓库内 *report/* 路径下的 *error.txt*、*report.txt* 和 *success.txt* 报告文件中查看运行结果。

示例 2

示例描述：运行 *matrixone* 仓库内 */cases/transaction/* 路径下的测试用例。

步骤：

1. 拉取最新的 *matrixone* 远端仓库。

```
cd matrixone
git pull https://github.com/matrixorigin/matrixone.git
```

2. 先切换进入到 *mo-tester* 仓库，运行如下命令，即可运行 *matrixone* 仓库内 *cases/transaction/* 路径的所有测试用例：

```
cd mo-tester
./run.sh -p \{path_name}/matrixone/test/cases/transaction/
```

3. 在 *MO-Tester* 仓库内 *report/* 路径下的 *error.txt*、*report.txt* 和 *success.txt* 报告文件中查看运行结果。例如，预期 *report.txt* 报告如下所示：

```
[SUMMARY] COST : 5s, TOTAL :1362, SUCCESS : 1354, FAILED :0, IGNORED
:8, ABNORAML :0, SUCCESS RATE : 100%
[\{path_name}/matrixone/test/cases/transaction/atomicity.sql] COST :
0.575s, TOTAL :66, SUCCESS :66, FAILED :0, IGNORED :0, ABNORAML :0,
SUCCESS RATE : 100%
[\{path_name}/matrixone/test/cases/transaction/autocommit.test] COST :
0.175s, TOTAL :50, SUCCESS :50, FAILED :0, IGNORED :0, ABNORAML :0,
SUCCESS RATE : 100%
[\{path_name}/matrixone/test/cases/transaction/autocommit_1.sql] COST
: 1.141s, TOTAL :296, SUCCESS :288, FAILED :0, IGNORED :8, ABNORAML
:0, SUCCESS RATE : 100%
[\
{path_name}/matrixone/test/cases/transaction/autocommit_atomicity.sql]
COST : 0.52s, TOTAL :75, SUCCESS :75, FAILED :0, IGNORED :0, ABNORAML
:0, SUCCESS RATE : 100%
[\
{path_name}/matrixone/test/cases/transaction/autocommit_isolation.sql]
COST : 1.607s, TOTAL :215, SUCCESS :215, FAILED :0, IGNORED :0,
ABNORAML :0, SUCCESS RATE : 100%
[\
{path_name}/matrixone/test/cases/transaction/autocommit_isolation_1.sql]
COST : 1.438s, TOTAL :241, SUCCESS :241, FAILED :0, IGNORED :0,
ABNORAML :0, SUCCESS RATE : 100%
[\{path_name}/matrixone/test/cases/transaction/isolation.sql] COST :
1.632s, TOTAL :202, SUCCESS :202, FAILED :0, IGNORED :0, ABNORAML :0,
SUCCESS RATE : 100%
[\{path_name}/matrixone/test/cases/transaction/isolation_1.sql] COST :
1.512s, TOTAL :217, SUCCESS :217, FAILED :0, IGNORED :0, ABNORAML :0,
SUCCESS RATE : 100%
```

示例 3

示例描述：运行 *matrixone* 仓库内 *cases/transaction/atomicity.sql* 单个测试用例。

步骤：

1. 拉取最新的 *matrixone* 远端仓库。

```
cd matrixone
git pull https://github.com/matrixorigin/matrixone.git
```

2. 先切换进入到 *mo-tester* 仓库，运行如下命令，即可运行 *mo-tester* 仓库内 *cases/transaction/atomicity.sql* 测试用例：

```
cd mo-tester
./run.sh -p \
{path_name}/matrixone/test/cases/transaction/atomicity.sql
```

3. 在 *MO-Tester* 仓库内 *report/* 路径下的 *error.txt*、*report.txt* 和 *success.txt* 报告文件中查看运行结果。例如，预期 *report.txt* 报告如下所示：

```
[SUMMARY] COST : 0s, TOTAL :66, SUCCESS : 66, FAILED :0, IGNORED :0,
ABNORAML :0, SUCCESS RATE : 100%
[\{path_name}/matrixone/test/cases/transaction/atomicity.sql] COST :
0.56s, TOTAL :66, SUCCESS :66, FAILED :0, IGNORED :0, ABNORAML :0,
SUCCESS RATE : 100%
```

示例 4

示例描述：

- 新建一个命名为 *local_test* 的文件夹，放在 *{path_name}/matrixone/test/cases* 目录下
- 本地新增一个命名为测试文件 *new_test.sql*，放在 *{path_name}/matrixone/testcases/local_test/* 路径下
- 仅想要运行 *new_test.sql** 测试用例

步骤

1. 拉取最新的 *matrixone* 远端仓库。

```
cd matrixone
git pull https://github.com/matrixorigin/matrixone.git
```

2. 生成测试结果：

   - 方式 1：先切换进入到 *mo-tester* 仓库，运行如下命令，生成测试结果。

```
cd mo-tester
./run.sh -p \
{path_name}/matrixone/test/cases/local_test/new_test.sql -m genrs
-g
```

- 方式 2：打开 *mo-tester* 仓库中 *run.yml* 文件，先将 *method* 参数由默认的 `run` 修改为 `genrs`，然后运行如下命令，生成测试结果。

```
cd mo-tester
./run.sh -p \
{path_name}/matrixone/test/cases/local_test/new_test.sql
```

3. 在 *matrixone* 仓库内 *test/cases/local_test/* 路径下查看 *new_test.result* 结果。

4. 在 *mo-tester* 仓库内 *report/* 路径下的 *error.txt*、*report.txt* 和 *success.txt* 报告文件中查看运行结果。例如，*report.txt* 中结果如下所示：

```
[SUMMARY] COST : 0s, TOTAL :66, SUCCESS : 66, FAILED :0, IGNORED :0,
ABNORAML :0, SUCCESS RATE : 100%
[\{path_name}/matrixone/test/cases/transaction/atomicity.sql] COST :
0.56s, TOTAL :66, SUCCESS :66, FAILED :0, IGNORED :0, ABNORAML :0,
SUCCESS RATE : 100%
```

参考文档

更多关于 MO-Tester 测试工具的注解以及测试用例编写规范，参见 [MO-Tester 规范要求]。

MO-Tester 规范要求

编写测试用例规范 - case

规范	详情
文件命名	1. 以 *.sql* 或 *.test* 作为后缀。 2. 文件名称有实际含义。例如，需要编写测试索引用例，可以将测试索引用例命名为 *index.sql* 或者*index_xxxx.sql*。
测试用例	1. 测试用例，即 case 内的具体示例内容，所有的空行在测试时都自动忽略，你可以通过添加空行来是整个文件内容更易读。 2. 每条 SQL 语句只写一行，如果必须要多行书写，那么每行必须顶格书写，且 SQL 结尾不能有空格，否则将造成 *case* 文件 和 *result* 文件中 SQL 不能完全匹配。 3. 添加注释，注明当前所写测试用例的目的。 4. 为需要增加 Tag 标签测试用例添加 Tag。

测试用例注解说明 - Tag

注解	起始	结束	说明
— @bvt	— @bvt #{issueNO.}]	— @bvt	带有此注解标记的 SQL 将不会被执行
— @sessio	— @session =X{	— @session}	带有此主键标记的所有 SQL 将在 id=X 的新会话中执行
— @separator	/	/	指定 SQL 语句在解析其 result 以及生成 result 文件的时候，使用的列分隔符，有两个取值
— @separator	/	/	表示结果中列分隔符为制表符 \t
— @separator	/	/	表示结果中列分隔符为 4 个空格
— @sortkey	— @sortkey:1,2,3	/	表示该SQL语句的结果是有序的，排序键为第1，2，3列（从0开始）；正常情况下，被测试系统返回的结果顺序是

注解	起始	结束	说明
			不固定的，所以工具在比对的时候，会把实际结果和预期结果都进行排序后比对，但是对于某些 SQL，其预期的结果就应该是有序的，比如存在 `order by` 的 SQL 语句，那么需要把类似这种 SQL 添加上这样的 Tag，工具在比对的时候，不会对 *sortkey* 中的这些列进行重新排序

编写测试结果规范 - result

规范	详情
普通测试用例生成测试结果	1. 如果新增了测试 case 文件，首先确保所有 SQL 都调试通过，使用 MO-Tester 工具自动生成测试结果。 2. 如果是在已有的 case 内新增一些 SQL，首先确保所有 SQL 都调试通过，使用 MO-Tester 工具自动生成测试结果。
含有 Tag 的测试用例生成测试结果	1. 如果新增测试 case 文件，且 case 文件内含有带 `--bvt:issue` 标签 SQL，使用 MO-Tester 工具自动生成测试结果，带 `--bvt:issue` 标签 SQL 在所生成的 result 文件中结果为 `unknown result because it is related to issue#XXX"`。 2. 如果是已有测试 case 文件，已通过 MO-Tester 工具自动生成测试结果，再次使用 MO-Tester 工具自动生成测试结果时，带 `--bvt:issue` 标签 SQL 在所生成的 result 文件中结果将保留原有值，不再更新。
手动编写测试结果	1. 如果是手动编写 result，不可有空行，否则结果将产生解析错误。 2. 如果预期某条 SQL 的执行结果中，存在制表符或者超过连续的 4 个空格，则在测试 case 文件中对应的 SQL 语句，必须增加 Tag 标签 `-- @separator:`，否则将解析失败。例如，若 case 文件中对应的 SQL 语句含连续 4 个空格，则需要指定 Tag 标签 `-- @separator:table`；若含含有制表符，则指定 Tag 标签 `-- @separator:space`。

在测试用例脚本中设置标记

有时，为了达到特定的目的，如暂停或创建新连接，您可以向脚本文件添加特殊的标记。Motester 提供以下标签供使用：

标签	说明
— @skip #{IssueNo.}	设置后，整个脚本文件将被跳过，并且不再执行 issue{IssueNo.}
— @bvt #{IssueNo.} — @bvt	不执行 issue{IssueNo.} 这两个标记之间的 sql 语句。
— @sleep:{time}	mo-tester 将等待时长{time}
— @session =2&user=root&password=111 — @session	mo tester 将创建一个新的连接来执行这两个标记之间的 sql 语句。 id 默认值为 1，最大值为 10。 在 *mo.yml* 中配置了用户和密码的默认值。
— @sortkey:	设置此标记，表示对结果进行排序。例如: ——@sortkey:0,1:表示排序键是第一列和第二列。

SQL 性能调优方法概述

SQL 性能调优是一种优化数据库查询和操作的过程，旨在提高数据库的性能和响应时间。常见的几种性能调优方式如下：

- 索引优化：索引可以加速查询，提高数据库的性能。通过使用正确的索引类型、选择正确的索引列、避免使用过多的索引以及定期重新构建索引等方法，可以最大化地利用索引来提高性能。
- 优化查询语句：通过优化查询语句的结构、避免使用不必要的子查询、使用更有效的 `JOIN` 语句、避免使用 `OR` 操作符等方法，可以减少查询所需的时间和资源。
- 优化表结构：优化表结构，如选择正确的数据类型、避免 NULL 值、使用合适的约束和默认值、归一化和反归一化等方法，可以减少表的存储空间和减少查询的时间。
- 控制数据量：通过限制返回的数据量、分页、缓存、使用存储过程等方法，可以减少查询所需的时间和资源。
- 优化服务器配置：通过增加服务器的内存、调整数据库参数、定期清理日志和缓存等方法，可以提高数据库的性能和响应时间。
- 监控和调试：使用数据库性能监控工具、调试 SQL 查询语句、查看数据库的日志和错误信息等方法，可以帮助发现并解决性能问题。

需要注意的是，SQL 性能调优是一种复杂的过程，需要综合考虑数据库的结构、数据量、查询模式等多个因素，同时需要不断测试和验证优化结果，才能最终提高数据库的性能和响应时间。

MatrixOne 在执行 SQL 语句时，会自动规划并选择最优的执行方案，并非按照 SQL 语句进行查询，当前 MatrixOne 支持通过 **EXPLAIN 解读执行计划**选择最优执行方案进行性能调优，也支持通过优化表的物理排列进行性能调优。为帮助你更好的 MatrixOne 对于 SQL 语句查询进行性能调优，你可以查看以下文档：

- 了解 MatrixOne 执行计划：介绍 MatrixOne 执行计划概念。
- 使用 EXPLAIN 解读执行计划：介绍如何使用 EXPLAIN 语句来理解 MatrixOne 是如何执行某个查询的。
- 性能调优最佳实践：介绍 MatrixOne 使用 `Cluster by` 达到性能调优的最佳实践，学习提高查询性能的方法。

MatrixOne 执行计划概述

什么是执行计划

执行计划（execution plan，也叫查询计划或者解释计划）是数据库执行 SQL 语句的具体步骤，例如通过索引还是全表扫描访问表中的数据，连接查询的实现方式和连接的顺序等；执行计划根据你的表、列、索引和 `WHERE` 子句中的条件的详细信息，可以告诉你这个查询将会被如何执行或者已经被如何执行过，可以在不读取所有行的情况下执行对巨大表的查询；可以在不比较行的每个组合的情况下执行涉及多个表的连接。。如果 SQL 语句性能不够理想，首先应该查看它的执行计划。和大多数成熟的数据库产品一样，MatrixOne 数据库也提供了这一分析查询语句性能的功能。

MatrixOne 查询优化器对输入的 SQL 查询语句通过**执行计划**而选择出效率最高的一种执行方案。你也可以通过执行计划看到 SQL 代码中那些效率比较低的地方。

使用 `EXPLAIN` 查询执行计划

使用 `EXPLAIN` 可查看 MatrixOne 执行某条 SQL 语句时的执行计划。

`EXPLAIN` 可以和 `SELECT`、`DELETE`、`INSERT`、`REPLACE`、`UPDATE` 语句结合使用。当 `EXPLAIN` 与可解释的语句一起使用时，MatrixOne 会解释它将如何处理该语句，包括有关表如何连接以及连接顺序的信息。

> **注意**
>
> 使用 MySQL 客户端连接到 MatrixOne 时，为避免输出结果在终端中换行，可先执行 `pager less -S` 命令。执行命令后，新的 `EXPLAIN` 的输出结果不再换行，可按右箭头 → 键水平滚动阅读输出结果。

EXPLAIN 示例

下面的例子帮助你了解 `EXPLAIN`。

数据准备：

```
CREATE TABLE t (id INT NOT NULL PRIMARY KEY auto_increment, a INT NOT
NULL, pad1 VARCHAR(255), INDEX(a));
INSERT INTO t VALUES (1, 1, 'aaa'),(2,2, 'bbb');
EXPLAIN SELECT * FROM t WHERE a = 1;
```

返回结果：

```
+-----------------------------------------------+
| QUERY PLAN |
+-----------------------------------------------+
| Project |
| -> Table Scan on aab.t |
| Filter Cond: (CAST(t.a AS BIGINT) = 1) |
+-----------------------------------------------+
```

`EXPLAIN` 实际不会执行查询。`EXPLAIN ANALYZE` 可用于实际执行查询并显示执行计划。如果 MatrixOne 所选的执行计划非最优，可用 `EXPLAIN` 或 `EXPLAIN ANALYZE` 来进行诊断。

EXPLAIN 输出分析

- QUERY PLAN，即本次执行的主题，查询计划
  - Filter Cond：过滤条件
  - Table Scan：对某个全表进行扫描
- Project 为这次查询过程中的执行顺序的父节点，Project 的结构是树状的，子节点计算完成后 “流入” 父节点。父节点、子节点和同级节点可能并行执行查询的一部分。

范围查询

在 `WHERE/HAVING/ON` 条件中，MatrixOne 优化器会分析主键或索引键的查询返回。如数字、日期类型的比较符，如大于、小于、等于以及大于等于、小于等于，字符类型的 `LIKE` 符号等。

使用 EXPLAIN 解读执行计划

SQL 是一种声明性语言，因此无法通过 SQL 语句直接判断一条查询的执行是否有效率，但是可以使用 `EXPLAIN` 语句查看当前的执行计划。

示例

我们这里准备一个简单的示例，帮助你理解使用 EXPLAIN 解读执行计划。

```
> drop table if exists a;
> create table a(a int);
> insert into a values(1),(2),(3),(4),(5),(6),(7),(8);
> select count(*) from a where a>=2 and a<=8;
+----------+
| count(*) |
+----------+
| 7 |
+----------+
1 row in set (0.00 sec)

> explain select count(*) from a where a>=2 and a<=8;
+-------------------------------------------------------------------------
-----------+
| QUERY PLAN |
+-------------------------------------------------------------------------
-----------+
| Project |
| -> Aggregate |
| Aggregate Functions: starcount(1) |
| -> Table Scan on aab.a |
| Filter Cond: (CAST(a.a AS BIGINT) >= 2), (CAST(a.a AS BIGINT) <= 8) |
+-------------------------------------------------------------------------
-----------+
5 rows in set (0.00 sec)
```

以上是该查询的执行计划结果。从 `Filter Cond` 算子开始向上看，查询的执行过程如下：

1. 先执行过滤条件 `Filter Cond`：即过滤出数据类型为 `BIGINT` 且大于等于 2，小于等于 8 的整数，按照计算推理，应该为 `(2),(3),(4),(5),(6),(7),(8)`。

2. 扫描数据库 aab 中的表 a。
3. 聚合计算满足条件整数的个数，为 7 个。

最终，得到查询结果为 7，即 `count(*)` = 7。

评估当前的性能

EXPLAIN 语句只返回查询的执行计划，并不执行该查询。若要获取实际的执行时间，可执行该查询，或使用 EXPLAIN ANALYZE 语句。

什么是 EXPLAIN ANALYZE

EXPLAIN ANALYZE 是一个用于查询的分析工具，它将向你显示 SQL 在查询上花费的时间以及原因。它将计划查询、检测它并执行它，同时计算行数并测量在执行计划的各个点花费的时间。执行完成后，EXPLAIN ANALYZE 将打印计划和测量结果，而不是查询结果。

除了正常 EXPLAIN 将打印的查询计划和估计成本之外，EXPLAIN ANALYZE 还打印执行计划中各个迭代器的实际成本。

如何使用它?

这里还是继续使用上述示例，你可以执行下面的代码：

```
> explain analyze select count(*) from a where a>=2 and a<=8;
+-------------------------------------------------------------------------
-------------------------------------------------------+
| QUERY PLAN |
+-------------------------------------------------------------------------
-------------------------------------------------------+
| Project |
| Analyze: timeConsumed=0us inputRows=1 outputRows=1 inputSize=8bytes
outputSize=8bytes memorySize=8bytes |
| -> Aggregate |
| Analyze: timeConsumed=3317us inputRows=2 outputRows=2 inputSize=8bytes
outputSize=16bytes memorySize=16bytes |
| Aggregate Functions: starcount(1) |
| -> Table Scan on aab.a |
| Analyze: timeConsumed=6643us inputRows=31 outputRows=24
inputSize=96bytes outputSize=64bytes memorySize=64bytes |
| Filter Cond: (CAST(a.a AS BIGINT) >= 2), (CAST(a.a AS BIGINT) <= 8) |
+-------------------------------------------------------------------------
-------------------------------------------------------+
8 rows in set (0.00 sec)
```

从打印的执行结果来看，当分别执行聚合计算和扫描表时，都会得出以下几个测量值，这些测量值可以作为参考项：

- 总耗时 timeConsumed
- 读取的行数
- 读取的容量大小
- 内存大小

通过在这些信息，你可以分析查询并理解它们为何是这样的表现，可以从以下几个方面进行探索：

- 执行这些查询，需要花费多久？你可以查看总耗时。
- 为什么执行当前的查询计划，而不是其他的执行计划？你可以查看行计数器。当估计行数与实际行数之间的巨大差异（即，几个数量级或更多）时，说明优化器根据估计选择计划，但查看实际执行可以方便你得知到底哪个执行计划更好。

所以使用 EXPLAIN ANALYZE 就是分析查询执行。

从上面的输出结果来看，执行以上示例查询耗时 0.00 秒，说明执行性能较为理想。也由于我们这次示例中执行的查询简单，满足较高的执行性能。

更多关于 EXPLAIN ANALYZE 的信息，请参见 EXPLAIN ANALYZE。

用 EXPLAIN 查看 JOIN 查询的执行计划

SQL 查询中可能会使用 JOIN 进行表连接，可以通过 EXPLAIN 语句来查看 JOIN 查询的执行计划。

示例

我们这里准备一个简单的示例，帮助你理解使用 EXPLAIN 解读 JOIN 查询的执行计划。

```
> drop table if exists t1;
> create table t1 (id int,ti tinyint unsigned,si smallint,bi bigint
unsigned,fl float,dl double,de decimal,ch char(20),vch varchar(20),dd
date,dt datetime);
> insert into t1
values(1,1,4,3,1113.32,111332,1113.32,'hello','subquery','2022-04-
28','2022-04-28 22:40:11');
> insert into t1 values(2,2,5,2,2252.05,225205,2252.05,'bye','sub
query','2022-04-28','2022-04-28 22:40:11');
> insert into t1
values(3,6,6,3,3663.21,366321,3663.21,'hi','subquery','2022-04-28','2022-
04-28 22:40:11');
> insert into t1 values(4,7,1,5,4715.22,471522,4715.22,'good morning','my
subquery','2022-04-28','2022-04-28 22:40:11');
> insert into t1 values(5,1,2,6,51.26,5126,51.26,'byebye',' is
subquery?','2022-04-28','2022-04-28 22:40:11');
> insert into t1 values(6,3,2,1,632.1,6321,632.11,'good night','maybe
subquery','2022-04-28','2022-04-28 22:40:11');
> insert into t1
values(7,4,4,3,7443.11,744311,7443.11,'yes','subquery','2022-04-
28','2022-04-28 22:40:11');
> insert into t1 values(8,7,5,8,8758.00,875800,8758.11,'nice to
meet','just subquery','2022-04-28','2022-04-28 22:40:11');
> insert into t1 values(9,8,4,9,9849.312,9849312,9849.312,'see
you','subquery','2022-04-28','2022-04-28 22:40:11');

> drop table if exists t2;
> create table t2 (id int,ti tinyint unsigned,si smallint,bi bigint
unsigned,fl float,dl double,de decimal,ch char(20),vch varchar(20),dd
date,dt datetime);
> insert into t2
values(1,1,4,3,1113.32,111332,1113.32,'hello','subquery','2022-04-
28','2022-04-28 22:40:11');
> insert into t2 values(2,2,5,2,2252.05,225205,2252.05,'bye','sub
query','2022-04-28','2022-04-28 22:40:11');
> insert into t2
values(3,6,6,3,3663.21,366321,3663.21,'hi','subquery','2022-04-28','2022-
04-28 22:40:11');
> insert into t2 values(4,7,1,5,4715.22,471522,4715.22,'good morning','my
subquery','2022-04-28','2022-04-28 22:40:11');
> insert into t2 values(5,1,2,6,51.26,5126,51.26,'byebye',' is
subquery?','2022-04-28','2022-04-28 22:40:11');
> insert into t2 values(6,3,2,1,632.1,6321,632.11,'good night','maybe
subquery','2022-04-28','2022-04-28 22:40:11');
```

```
> insert into t2
values(7,4,4,3,7443.11,744311,7443.11,'yes','subquery','2022-04-
28','2022-04-28 22:40:11');
> insert into t2 values(8,7,5,8,8758.00,875800,8758.11,'nice to
meet','just subquery','2022-04-28','2022-04-28 22:40:11');
> insert into t2 values(9,8,4,9,9849.312,9849312,9849.312,'see
you','subquery','2022-04-28','2022-04-28 22:40:11');
```

Hash Join

在 `Hash Join` 操作中，MatrixOne 首先读取表 *t1* 与 *t2* 中相对较小的一个表对其中每个需要被连接的值使用哈希函数，得到一个哈希表；然后对另一个表的每一行进行扫描并计算哈希值，与上一步生成的哈希表进行比对，如果有符合连接标准的值，则根据连接谓词生成一个新的连接表。

MatrixOne 中的 Hash Join 算子是多线程的，并且可以并发执行。

下面是一个 `Hash Join` 示例：

```
> SELECT /*+ HASH_JOIN(t1, t2) */ * FROM t1, t2 WHERE t1.id = t2.id;
+------+------+------+------+----------+---------+------+--------------+-
--------------+------------+--------------------+------+------+------+-
-----+----------+---------+------+--------------+----------------+-------
-----+---------------------+
| id | ti | si | bi | fl | dl | de | ch | vch | dd | dt | id | ti | si |
bi | fl | dl | de | ch | vch | dd | dt |
+------+------+------+------+----------+---------+------+--------------+-
--------------+------------+--------------------+------+------+------+-
-----+----------+---------+------+--------------+----------------+-------
-----+---------------------+
| 1 | 1 | 4 | 3 | 1113.32 | 111332 | 1113 | hello | subquery | 2022-04-28
| 2022-04-28 22:40:11 | 1 | 1 | 4 | 3 | 1113.32 | 111332 | 1113 | hello |
subquery | 2022-04-28 | 2022-04-28 22:40:11 |
| 2 | 2 | 5 | 2 | 2252.05 | 225205 | 2252 | bye | sub query | 2022-04-28
| 2022-04-28 22:40:11 | 2 | 2 | 5 | 2 | 2252.05 | 225205 | 2252 | bye |
sub query | 2022-04-28 | 2022-04-28 22:40:11 |
| 3 | 6 | 6 | 3 | 3663.21 | 366321 | 3663 | hi | subquery | 2022-04-28 |
2022-04-28 22:40:11 | 3 | 6 | 6 | 3 | 3663.21 | 366321 | 3663 | hi |
subquery | 2022-04-28 | 2022-04-28 22:40:11 |
| 4 | 7 | 1 | 5 | 4715.22 | 471522 | 4715 | good morning | my subquery |
2022-04-28 | 2022-04-28 22:40:11 | 4 | 7 | 1 | 5 | 4715.22 | 471522 |
4715 | good morning | my subquery | 2022-04-28 | 2022-04-28 22:40:11 |
| 5 | 1 | 2 | 6 | 51.26 | 5126 | 51 | byebye | is subquery? | 2022-04-28
| 2022-04-28 22:40:11 | 5 | 1 | 2 | 6 | 51.26 | 5126 | 51 | byebye | is
subquery? | 2022-04-28 | 2022-04-28 22:40:11 |
| 6 | 3 | 2 | 1 | 632.1 | 6321 | 632 | good night | maybe subquery |
2022-04-28 | 2022-04-28 22:40:11 | 6 | 3 | 2 | 1 | 632.1 | 6321 | 632 |
good night | maybe subquery | 2022-04-28 | 2022-04-28 22:40:11 |
| 7 | 4 | 4 | 3 | 7443.11 | 744311 | 7443 | yes | subquery | 2022-04-28 |
2022-04-28 22:40:11 | 7 | 4 | 4 | 3 | 7443.11 | 744311 | 7443 | yes |
subquery | 2022-04-28 | 2022-04-28 22:40:11 |
| 8 | 7 | 5 | 8 | 8758 | 875800 | 8758 | nice to meet | just subquery |
2022-04-28 | 2022-04-28 22:40:11 | 8 | 7 | 5 | 8 | 8758 | 875800 | 8758 |
nice to meet | just subquery | 2022-04-28 | 2022-04-28 22:40:11 |
| 9 | 8 | 4 | 9 | 9849.312 | 9849312 | 9849 | see you | subquery | 2022-
04-28 | 2022-04-28 22:40:11 | 9 | 8 | 4 | 9 | 9849.312 | 9849312 | 9849 |
see you | subquery | 2022-04-28 | 2022-04-28 22:40:11 |
+------+------+------+------+----------+---------+------+--------------+-
--------------+------------+--------------------+------+------+------+-
-----+----------+---------+------+--------------+----------------+-------
-----+---------------------+
9 rows in set (0.00 sec)
```

```
mysql> EXPLAIN SELECT /*+ HASH_JOIN(t1, t2) */ * FROM t1, t2 WHERE t1.id
= t2.id;
+-----------------------------------+
| QUERY PLAN |
+-----------------------------------+
| Project |
| -> Join |
| Join Type: INNER |
| Join Cond: (t1.id = t2.id) |
| -> Table Scan on db1.t1 |
| -> Table Scan on db1.t2 |
+-----------------------------------+
6 rows in set (0.01 sec)
```

MatrixOne 会按照以下顺序执行 `Hash Join` 算子：

1. 分别并行扫描表 *t2* 和 *t1*。
2. 执行 JOIN 的过滤查询：`(t1.id = t2.id)`。
3. 执行 INNER JOIN。

用 EXPLAIN 查看子查询的执行计划

MatrixOne 会执行多种子查询相关的优化，以提升子查询的执行性能。本文档介绍一些常见子查询的优化方式，以及如何解读 EXPLAIN 语句返回的执行计划信息。

从 SQL 语句执行情况上，子查询语句一般有以下两种形式：

- **无关联子查询 (Self-contained Subquery)**：数据库嵌套查询中内层查询是完全独立于外层查询的。

  例如：
  `select * from t1 where t1.id in (select t2.id from t2 where t2.id>=3);` 执行顺序为：

  - 先执行内层查询：`(select t2.id from t2 where t2.id>=3)`。
  - 得到内层查询的结果后带入外层，再执行外层查询。

- **关联子查询（Correlated Subquery）**：数据库嵌套查询中内层查询和外层查询不相互独立，内层查询也依赖于外层查询。

  例如：
  ```
  SELECT * FROM t1 WHERE id in (SELECT id FROM t2 WHERE t1.ti = t2.ti and t2.id>=4);
  ```
  一般情况下，执行顺序为：

  - 先从外层查询中查询中一条记录：`SELECT * FROM t1 WHERE id`。
  - 再将查询到的记录放到内层查询中符合条件的记录，再放到外层中查询。
  - 重复以上步骤

  但是 MatrixOne 在处理该 SQL 语句是会将其改写为等价的 `JOIN` 查询：
  `select t1.* from t1 join t2 on t1.id=t2.id where t2.id>=4;`

示例

我们这里准备一个简单的示例，帮助你理解使用 EXPLAIN 解读子查询的执行计划。

```
> drop table if exists t1;
> create table t1 (id int,ti tinyint unsigned,si smallint,bi bigint
unsigned,fl float,dl double,de decimal,ch char(20),vch varchar(20),dd
date,dt datetime);
> insert into t1
values(1,1,4,3,1113.32,111332,1113.32,'hello','subquery','2022-04-
28','2022-04-28 22:40:11');
> insert into t1 values(2,2,5,2,2252.05,225205,2252.05,'bye','sub
query','2022-04-28','2022-04-28 22:40:11');
> insert into t1
values(3,6,6,3,3663.21,366321,3663.21,'hi','subquery','2022-04-28','2022-
04-28 22:40:11');
> insert into t1 values(4,7,1,5,4715.22,471522,4715.22,'good morning','my
subquery','2022-04-28','2022-04-28 22:40:11');
> insert into t1 values(5,1,2,6,51.26,5126,51.26,'byebye',' is
subquery?','2022-04-28','2022-04-28 22:40:11');
> insert into t1 values(6,3,2,1,632.1,6321,632.11,'good night','maybe
subquery','2022-04-28','2022-04-28 22:40:11');
> insert into t1
values(7,4,4,3,7443.11,744311,7443.11,'yes','subquery','2022-04-
28','2022-04-28 22:40:11');
> insert into t1 values(8,7,5,8,8758.00,875800,8758.11,'nice to
meet','just subquery','2022-04-28','2022-04-28 22:40:11');
> insert into t1 values(9,8,4,9,9849.312,9849312,9849.312,'see
you','subquery','2022-04-28','2022-04-28 22:40:11');

> drop table if exists t2;
> create table t2 (id int,ti tinyint unsigned,si smallint,bi bigint
unsigned,fl float,dl double,de decimal,ch char(20),vch varchar(20),dd
date,dt datetime);
> insert into t2
values(1,1,4,3,1113.32,111332,1113.32,'hello','subquery','2022-04-
28','2022-04-28 22:40:11');
> insert into t2 values(2,2,5,2,2252.05,225205,2252.05,'bye','sub
query','2022-04-28','2022-04-28 22:40:11');
> insert into t2
values(3,6,6,3,3663.21,366321,3663.21,'hi','subquery','2022-04-28','2022-
04-28 22:40:11');
> insert into t2 values(4,7,1,5,4715.22,471522,4715.22,'good morning','my
subquery','2022-04-28','2022-04-28 22:40:11');
> insert into t2 values(5,1,2,6,51.26,5126,51.26,'byebye',' is
subquery?','2022-04-28','2022-04-28 22:40:11');
> insert into t2 values(6,3,2,1,632.1,6321,632.11,'good night','maybe
subquery','2022-04-28','2022-04-28 22:40:11');
```

```
> insert into t2
values(7,4,4,3,7443.11,744311,7443.11,'yes','subquery','2022-04-
28','2022-04-28 22:40:11');
> insert into t2 values(8,7,5,8,8758.00,875800,8758.11,'nice to
meet','just subquery','2022-04-28','2022-04-28 22:40:11');
> insert into t2 values(9,8,4,9,9849.312,9849312,9849.312,'see
you','subquery','2022-04-28','2022-04-28 22:40:11');
```

无关联子查询

```
> select * from t1 where t1.id in (select t2.id from t2 where t2.id>=3);
+------+------+------+------+----------+---------+------+--------------+-
---------------+------------+---------------------+
| id | ti | si | bi | fl | dl | de | ch | vch | dd | dt |
+------+------+------+------+----------+---------+------+--------------+-
---------------+------------+---------------------+
| 3 | 6 | 6 | 3 | 3663.21 | 366321 | 3663 | hi | subquery | 2022-04-28 |
2022-04-28 22:40:11 |
| 4 | 7 | 1 | 5 | 4715.22 | 471522 | 4715 | good morning | my subquery |
2022-04-28 | 2022-04-28 22:40:11 |
| 5 | 1 | 2 | 6 | 51.26 | 5126 | 51 | byebye | is subquery? | 2022-04-28
| 2022-04-28 22:40:11 |
| 6 | 3 | 2 | 1 | 632.1 | 6321 | 632 | good night | maybe subquery |
2022-04-28 | 2022-04-28 22:40:11 |
| 7 | 4 | 4 | 3 | 7443.11 | 744311 | 7443 | yes | subquery | 2022-04-28 |
2022-04-28 22:40:11 |
| 8 | 7 | 5 | 8 | 8758 | 875800 | 8758 | nice to meet | just subquery |
2022-04-28 | 2022-04-28 22:40:11 |
| 9 | 8 | 4 | 9 | 9849.312 | 9849312 | 9849 | see you | subquery | 2022-
04-28 | 2022-04-28 22:40:11 |
+------+------+------+------+----------+---------+------+--------------+-
---------------+------------+---------------------+
7 rows in set (0.02 sec)

> explain select * from t1 where t1.id in (select t2.id from t2 where
t2.id>=3);
+-------------------------------------------------------------------+
| QUERY PLAN |
+-------------------------------------------------------------------+
| Project |
| -> Join |
| Join Type: SEMI |
| Join Cond: (t1.id = t2.id) |
| -> Table Scan on db1.t1 |
| -> Project |
| -> Table Scan on db1.t2 |
| Filter Cond: (CAST(t2.id AS BIGINT) >= 3) |
+-------------------------------------------------------------------+
8 rows in set (0.00 sec)
```

可以看到这个执行计划的执行顺序是：

1. 先执行内层查询：`(select t2.id from t2 where t2.id>=3)`。
2. 得到内层查询的结果后带入外层，再执行外层查询。

关联子查询

```
> SELECT * FROM t1 WHERE id in (SELECT id FROM t2 WHERE t1.ti = t2.ti and
t2.id>=4);
+------+------+------+------+----------+---------+------+--------------+-
---------------+------------+---------------------+
| id | ti | si | bi | fl | dl | de | ch | vch | dd | dt |
+------+------+------+------+----------+---------+------+--------------+-
---------------+------------+---------------------+
| 4 | 7 | 1 | 5 | 4715.22 | 471522 | 4715 | good morning | my subquery |
2022-04-28 | 2022-04-28 22:40:11 |
| 5 | 1 | 2 | 6 | 51.26 | 5126 | 51 | byebye | is subquery? | 2022-04-28
| 2022-04-28 22:40:11 |
| 6 | 3 | 2 | 1 | 632.1 | 6321 | 632 | good night | maybe subquery |
2022-04-28 | 2022-04-28 22:40:11 |
| 7 | 4 | 4 | 3 | 7443.11 | 744311 | 7443 | yes | subquery | 2022-04-28 |
2022-04-28 22:40:11 |
| 8 | 7 | 5 | 8 | 8758 | 875800 | 8758 | nice to meet | just subquery |
2022-04-28 | 2022-04-28 22:40:11 |
| 9 | 8 | 4 | 9 | 9849.312 | 9849312 | 9849 | see you | subquery | 2022-
04-28 | 2022-04-28 22:40:11 |
+------+------+------+------+----------+---------+------+--------------+-
---------------+------------+---------------------+
6 rows in set (0.01 sec)

mysql> explain SELECT * FROM t1 WHERE id in (SELECT id FROM t2 WHERE
t1.ti = t2.ti and t2.id>=4);
+-------------------------------------------------------------------+
| QUERY PLAN |
+-------------------------------------------------------------------+
| Project |
| -> Join |
| Join Type: SEMI |
| Join Cond: (t1.ti = t2.ti), (t1.id = t2.id) |
| -> Table Scan on db1.t1 |
| -> Project |
| -> Table Scan on db1.t2 |
| Filter Cond: (CAST(t2.id AS BIGINT) >= 4) |
+-------------------------------------------------------------------+
8 rows in set (0.01 sec)
```

MatrixOne 在处理该 SQL 语句是会将其改写为等价的 `JOIN` 查询：`select t1.* from t1 join t2 on t1.id=t2.id where t2.id>=4;`，可以看到这个执行

计划的执行顺序是：

1. 先执行过滤查询 `where t2.id>=4;` 。
2. 再扫描表 `Table Scan on db1.t2` ，将结果“流入”父节点后，
3. 扫描表 `Table Scan on db1.t1` 。
4. 最后执行 `JOIN` 查询。

用 EXPLAIN 查看聚合查询执行计划

SQL 查询中可能会使用聚合计算，可以通过 EXPLAIN 语句来查看聚合查询的执行计划。

示例

我们这里准备一个简单的示例，帮助你理解使用 EXPLAIN 解读聚合查询的执行计划。

```
> drop table if exists t1;
> create table t1 (id int,ti tinyint unsigned,si smallint,bi bigint
unsigned,fl float,dl double,de decimal,ch char(20),vch varchar(20),dd
date,dt datetime);
> insert into t1
values(1,1,4,3,1113.32,111332,1113.32,'hello','subquery','2022-04-
28','2022-04-28 22:40:11');
> insert into t1 values(2,2,5,2,2252.05,225205,2252.05,'bye','sub
query','2022-04-28','2022-04-28 22:40:11');
> insert into t1
values(3,6,6,3,3663.21,366321,3663.21,'hi','subquery','2022-04-28','2022-
04-28 22:40:11');
> insert into t1 values(4,7,1,5,4715.22,471522,4715.22,'good morning','my
subquery','2022-04-28','2022-04-28 22:40:11');
> insert into t1 values(5,1,2,6,51.26,5126,51.26,'byebye',' is
subquery?','2022-04-28','2022-04-28 22:40:11');
> insert into t1 values(6,3,2,1,632.1,6321,632.11,'good night','maybe
subquery','2022-04-28','2022-04-28 22:40:11');
> insert into t1
values(7,4,4,3,7443.11,744311,7443.11,'yes','subquery','2022-04-
28','2022-04-28 22:40:11');
> insert into t1 values(8,7,5,8,8758.00,875800,8758.11,'nice to
meet','just subquery','2022-04-28','2022-04-28 22:40:11');
> insert into t1 values(9,8,4,9,9849.312,9849312,9849.312,'see
you','subquery','2022-04-28','2022-04-28 22:40:11');
```

Hash Aggregation

Hash Aggregation 算法在执行聚合时使用 Hash 表存储中间结果。此算法采用多线程并发优化，执行速度快，但与 Stream Aggregation 算法相比会消耗较多内存。

下面是一个使用 Hash Aggregation 的例子：

```
> SELECT /*+ HASH_AGG() */ count(*) FROM t1;
+----------+
| count(*) |
+----------+
| 9 |
+----------+
1 row in set (0.01 sec)

mysql> EXPLAIN SELECT /*+ HASH_AGG() */ count(*) FROM t1;
+-----------------------------------------+
| QUERY PLAN |
+-----------------------------------------+
| Project |
| -> Aggregate |
| Aggregate Functions: starcount(1) |
| -> Table Scan on db1.t1 |
+-----------------------------------------+
4 rows in set (0.01 sec)
```

用 EXPLAIN 查看带视图的 SQL 执行计划

`EXPLAIN` 语句返回的结果会显示视图引用的表，而不是视图本身的名称。这是因为视图是一张虚拟表，本身并不存储任何数据。视图的定义会和查询语句的其余部分在 SQL 优化过程中进行合并。

示例

我们这里准备一个简单的示例，帮助你理解使用 EXPLAIN 解读 VIEW 的执行计划。

```
> drop table if exists t1;
> create table t1 (id int,ti tinyint unsigned,si smallint,bi bigint
unsigned,fl float,dl double,de decimal,ch char(20),vch varchar(20),dd
date,dt datetime);
> insert into t1
values(1,1,4,3,1113.32,111332,1113.32,'hello','subquery','2022-04-
28','2022-04-28 22:40:11');
> insert into t1 values(2,2,5,2,2252.05,225205,2252.05,'bye','sub
query','2022-04-28','2022-04-28 22:40:11');
> insert into t1
values(3,6,6,3,3663.21,366321,3663.21,'hi','subquery','2022-04-28','2022-
04-28 22:40:11');
> insert into t1 values(4,7,1,5,4715.22,471522,4715.22,'good morning','my
subquery','2022-04-28','2022-04-28 22:40:11');
> insert into t1 values(5,1,2,6,51.26,5126,51.26,'byebye',' is
subquery?','2022-04-28','2022-04-28 22:40:11');
> insert into t1 values(6,3,2,1,632.1,6321,632.11,'good night','maybe
subquery','2022-04-28','2022-04-28 22:40:11');
> insert into t1
values(7,4,4,3,7443.11,744311,7443.11,'yes','subquery','2022-04-
28','2022-04-28 22:40:11');
> insert into t1 values(8,7,5,8,8758.00,875800,8758.11,'nice to
meet','just subquery','2022-04-28','2022-04-28 22:40:11');
> insert into t1 values(9,8,4,9,9849.312,9849312,9849.312,'see
you','subquery','2022-04-28','2022-04-28 22:40:11');

> drop table if exists t2;
> create table t2 (id int,ti tinyint unsigned,si smallint,bi bigint
unsigned,fl float,dl double,de decimal,ch char(20),vch varchar(20),dd
date,dt datetime);
> insert into t2
values(1,1,4,3,1113.32,111332,1113.32,'hello','subquery','2022-04-
28','2022-04-28 22:40:11');
> insert into t2 values(2,2,5,2,2252.05,225205,2252.05,'bye','sub
query','2022-04-28','2022-04-28 22:40:11');
> insert into t2
values(3,6,6,3,3663.21,366321,3663.21,'hi','subquery','2022-04-28','2022-
04-28 22:40:11');
> insert into t2 values(4,7,1,5,4715.22,471522,4715.22,'good morning','my
subquery','2022-04-28','2022-04-28 22:40:11');
> insert into t2 values(5,1,2,6,51.26,5126,51.26,'byebye',' is
subquery?','2022-04-28','2022-04-28 22:40:11');
> insert into t2 values(6,3,2,1,632.1,6321,632.11,'good night','maybe
subquery','2022-04-28','2022-04-28 22:40:11');
```

```
> insert into t2
values(7,4,4,3,7443.11,744311,7443.11,'yes','subquery','2022-04-
28','2022-04-28 22:40:11');
> insert into t2 values(8,7,5,8,8758.00,875800,8758.11,'nice to
meet','just subquery','2022-04-28','2022-04-28 22:40:11');
> insert into t2 values(9,8,4,9,9849.312,9849312,9849.312,'see
you','subquery','2022-04-28','2022-04-28 22:40:11');

> create view v1 as select * from (select * from t1) sub where id > 4;
> select * from v1;
+------+------+------+------+----------+---------+------+--------------+-
---------------+------------+---------------------+
| id | ti | si | bi | fl | dl | de | ch | vch | dd | dt |
+------+------+------+------+----------+---------+------+--------------+-
---------------+------------+---------------------+
| 5 | 1 | 2 | 6 | 51.26 | 5126 | 51 | byebye | is subquery? | 2022-04-28
| 2022-04-28 22:40:11 |
| 6 | 3 | 2 | 1 | 632.1 | 6321 | 632 | good night | maybe subquery |
2022-04-28 | 2022-04-28 22:40:11 |
| 7 | 4 | 4 | 3 | 7443.11 | 744311 | 7443 | yes | subquery | 2022-04-28 |
2022-04-28 22:40:11 |
| 8 | 7 | 5 | 8 | 8758 | 875800 | 8758 | nice to meet | just subquery |
2022-04-28 | 2022-04-28 22:40:11 |
| 9 | 8 | 4 | 9 | 9849.312 | 9849312 | 9849 | see you | subquery | 2022-
04-28 | 2022-04-28 22:40:11 |
+------+------+------+------+----------+---------+------+--------------+-
---------------+------------+---------------------+
5 rows in set (0.01 sec)
```

如上述示例所示，新建了一个命名为 *v1* 的 VIEW，并查询 *v1* 的结果。那么我们看一下这个视图的执行过程：

```
> explain select * from v1;
+-------------------------------------------------------------+
| QUERY PLAN |
+-------------------------------------------------------------+
| Project |
| -> Project |
| -> Project |
| -> Table Scan on db1.t1 |
| Filter Cond: (CAST(t1.id AS BIGINT) > 4) |
+-------------------------------------------------------------+
5 rows in set (0.00 sec)
```

可以看到 Project 为这次查询过程中的执行顺序的父节点，首先是从缩进最多的子节点开始计算，完成后 “流入” 它的上层父节点，最终 “流入” Project 父节点。

先执行：

- Filter Cond：过滤条件

再执行：

- Table Scan：对某个全表进行扫描

下面的查询的执行方式与上述执行方式类似：

```
> explain select * from (select * from t1) sub where id > 4;
+-------------------------------------------------------+
| QUERY PLAN |
+-------------------------------------------------------+
| Project |
| -> Project |
| -> Table Scan on db1.t1 |
| Filter Cond: (CAST(t1.id AS BIGINT) > 4) |
+-------------------------------------------------------+
4 rows in set (0.03 sec)
```

使用 Cluster by 语句调优

`Cluster by` 是一种常用的性能调优技术，可以帮助优化查询的执行效率。本文将解释如何使用 `Cluster by` 进行性能调优。

什么是 `Cluster by`?

`Cluster by` 是一种用于优化表的物理排列方式的命令。在建表时使用 `Cluster by` 命令，对于无主键的表，可以按照指定的列对表进行物理排序，并将数据行重新排列成与该列的值的顺序相同的顺序。这种物理排序有助于提高查询性能。

以下是使用 `Cluster by` 的一些注意事项：

- `Cluster by` 不能和主键同时存在，否则会语法报错。
- `Cluster by` 只能在建表时指定，不支持动态创建。

下面是 `Cluster by` 的使用语法：

- 单列语法为：`create table() cluster by col;`
- 多列语法为：`create table() cluster by (col1, col2);`

如何使用 `Cluster by` 进行性能调优?

使用 `Cluster by` 进行性能调优的步骤如下：

1. 确定需要排序的列。

   首先需要确定需要排序的列。一般来说，可以选择那些经常用于过滤的列。对于具有时间序列特征的数据，时间列是常用的排序列。例如，如果有一个订单表，可以将订单按照时间进行排序。

2. 执行 `Cluster by` 命令。

   一旦确定了排序列，就可以在建表时执行 `Cluster by` 命令在建表时进行排序。

下面我们来看一个示例：

```
create table t1(a int, b int, c varchar(10)) cluster by(a,b,c);
desc t1;
+-------+-------------+------+------+---------+-------+---------+
| Field | Type | Null | Key | Default | Extra | Comment |
+-------+-------------+------+------+---------+-------+---------+
| a | INT(32) | YES | | NULL | | |
| b | INT(32) | YES | | NULL | | |
| c | VARCHAR(10) | YES | | NULL | | |
+-------+-------------+------+------+---------+-------+---------+
3 rows in set (0.02 sec)
```

在这个例子中，我们首先创建了一个名为 t1 的表。然后，使用 CLUSTER 命令按照 a，b，c 列对表进行物理排序。这样，所有的数据行将按照 a，b，c 列的值的顺序排列。

如何使 cluster by 发挥最好的效果?

- 一是尽可能将最常查询的列放在靠前的位置。

  对于 `Cluster by` 多多列的情况，第一列将取得最好的查询加速效果，因为第一列的数据分布是完全有序的。只有在第一列相同的情况下，剩下的数据会按照第二列进行排序。所以第二列的查询加速效果会弱于第一列。之后的列会依次递减。所以 `Cluster by` 通常不推荐指定太多列，一般 3-4 列即可。

- 二是将低基数的列放在靠前的位置。

  首先**基数**是指某一列上不同值的数量。例如性别，由于只有两个值，就是很典型的低基数列。例如身份证，一般情况下都不会重复，就是高基数列。如果将高基数列放在 `Cluster by` 第一列，整张表的数据分布已经完全在第一列上排好序了，导致后续的列不起作用。这种情况下建议使用 `Cluster by` 单列，或者将高基数列单独建索引，不要放在 `Cluster by` 中。

使用 `Cluster by` 需要注意的事项

- 对大表进行 `Cluster by` 可能需要很长时间

当对大表进行 `Cluster by` 时，可能需要很长时间才能完成。这是因为该操作需要重新组织表中的数据，重新排序并存储。因此，在执行 `Cluster by` 命令时，需要考虑表的大小和硬件配置。

- `Cluster by` 可能会影响插入和更新操作的性能

由于使用 `Cluster by` 会对表中的数据进行物理排序，因此插入和更新操作的性能可能会受到影响。当表中的数据按照某些列排序时，插入和更新操作可能需要移动许多行。因此，在使用 `Cluster by` 时，需要考虑这种影响。

- `Cluster by` 需要定期执行以保持性能

由于数据的增长和变化，表中的数据物理排序可能会失去效果。因此，需要定期执行 `Cluster by` 命令，以确保数据的物理排序仍然有效。

需要注意的是，`Cluster by` 命令的使用需要谨慎，建议在测试环境下先进行验证，以确保不会对表的数据产生负面影响。

TLS 安全连接

概述

传输层安全性 (Transport Layer Security, TLS) 是一种广泛采用的安全性协议，目的是为互联网通信提供安全及数据完整性保障。

MatrixOne 默认采用非加密连接，也支持启用基于 TLS 协议的加密连接，支持的协议版本有 TLS 1.0, TLS 1.1, TLS 1.2。

- 不开启 TLS 加密连接（默认）：直接使用用户名密码连接 MatrixOne 即可。
- 使用加密连接：需要在 MatrixOne 服务端开启加密连接支持，并在客户端指定使用加密连接。你可以参加下文指导，开启 TLS 安全连接。

本篇文档将指导你如何开启 TLS 安全连接。

TLS 安全连接配置主要步骤概述：

1. 首先在 MatrixOne 中开启 TLS。
2. 然后配置 MySQL 客户端安全连接参数。

完成这两个主要步骤的配置后，即可建立 TLS 安全连接，具体操作参见下文：

步骤一：开启 MatrixOne 的 TLS 支持

1. 生成证书及密钥：MatrixOne 尚不支持加载有密码保护的私钥，因此必须提供一个没有密码的私钥文件。证书和密钥可以使用 OpenSSL 签发和生成，推荐使用 MySQL 自带的工具 `mysql_ssl_rsa_setup` 快捷生成：

```
#检查你本地 MySQL 客户端的安装目录
ps -ef|grep mysql
#进入到你本地 MySQL 客户端的安装目录
cd /usr/local/mysql/bin
#生成证书和密钥
./mysql_ssl_rsa_setup --datadir=\<yourpath>
#检查你生成的 pem 文件
ls \<yourpath>
├── ca-key.pem
├── ca.pem
├── client-cert.pem
├── client-key.pem
├── private_key.pem
├── public_key.pem
├── server-cert.pem
└── server-key.pem
```

Note: 上述代码中的 `\<yourpath>` 是你需要存放生成的证书及密钥文件的本地目录路径。

2. 进入到你本地的 MatrixOne 文件目录路径 *matrixone/etc/launch-tae-CN-tae-DN/* 中的 *cn.toml* 配置文件：

   你也可以使用 vim 命令直接在终端中打开 cn.toml 文件

   ```
   vim $matrixone/etc/launch-tae-CN-tae-DN/cn.toml
   ```

   将下面的代码段复制粘贴到配置文件中：

```
[cn.frontend]
#default is false. With true. Server will support tls
enableTls = true

#default is ''. Path of file that contains X509 certificate in PEM
format for client
tlsCertFile = "\<yourpath>/server-cert.pem"

#default is ''. Path of file that contains X509 key in PEM format for
client
tlsKeyFile = "\<yourpath>/server-key.pem"

#default is ''. Path of file that contains list of trusted SSL CAs for
client
tlsCaFile = "\<yourpath>/ca.pem"
```

Note: 上述代码中的 `\<yourpath>` 是你需要存放生成的证书及密钥文件的本地目录路径

上述代码中，配置参数解释如下：

参数	描述
enableTls	布尔类型，是否在 MatrixOne 服务端打开 TLS 的支持。
tlsCertFile	指定 SSL 证书文件路径
tlsKeyFile	指定证书文件对应的私钥
tlsCaFile	可选，指定受信任的 CA 证书文件路径

Note: 如果你是使用 Docker 安装部署的 MatrixOne，修改配置文件之前，你需要先挂载配置文件再进行修改，操作具体参见挂载目录到 Docker 容器。

3. 验证 MatrixOne 的 SSL 是否启用。

① 使用 MySQL 客户端连接 MatrixOne：

```
mysql -h 127.0.0.1 -P 6001 -udump -p111

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

② 使用 `Status` 命令查看 SSL 是否启用。

成功启用，代码示例如下，可以看到 SSL 状态为 `Cipher in use is TLS_AES_128_GCM_SHA256`：

```
mysql> status
mysql Ver 8.0.28 for macos11 on arm64 (MySQL Community Server - GPL)

Connection id: 1001
Current database:
Current user: dump@0.0.0.0
SSL: Cipher in use is TLS_AES_128_GCM_SHA256
Current pager: stdout
Using outfile: ''
Using delimiter: ;
Server version: 8.0.30-MatrixOne-v0.7.0 MatrixOne
Protocol version: 10
Connection: 127.0.0.1 via TCP/IP
Server characterset: utf8mb4
DB characterset: utf8mb4
Client characterset: utf8mb4
Conn. characterset: utf8mb4
TCP port: 6001
Binary data as: Hexadecimal
--------------
```

未启用成功，则返回结果如下，可以看到 SSL 状态为 `Not in use`，你需要重新检查一下上述步骤中你所配置证书及密钥文件的本地目录路径（即 <yourpath>）是否正确：

```
mysql> status;
/usr/local/mysql/bin/mysql Ver 8.0.30 for macos12 on arm64 (MySQL
Community Server - GPL)

Connection id: 1009
Current database: test
Current user: root@0.0.0.0
SSL: Not in use
Current pager: stdout
Using outfile: ''
Using delimiter: ;
Server version: 8.0.30-MatrixOne-v0.7.0 MatrixOne
Protocol version: 10
Connection: 127.0.0.1 via TCP/IP
Server characterset: utf8mb4
Db characterset: utf8mb4
Client characterset: utf8mb4
Conn. characterset: utf8mb4
TCP port: 6001
Binary data as: Hexadecimal
--------------
```

完成上述步骤后，即开启了 MatrixOne 的 TLS。

步骤二：配置 MySQL 客户端参数

MySQL 客户端连接 MatrixOne Server 时，需要通过 `--ssl-mode` 参数指定加密连接行为，如：

```
mysql -h 127.0.0.1 -P 6001 -udump -p111 --ssl-mode=PREFFERED
```

ssl mode 取值类型如下：

ssl-mode 取值	含义
DISABLED	不使用 SSL/TLS 建立加密连接，与 skip-ssl 同义。
PREFFERED	默认行为，优先尝试使用 SSL/TLS 建立加密连接，如果无法建则尝试建立非 SSL/TLS 连接。
REQUIRED	只会尝试使用 SSL/TLS 建立加密连接，如果无法建立连接，则会连接失败。

ssl-mode 取值	含义
VERIFY_CA	与 REQUIRED 行为一样，并且还会验证 Server 端的 CA 证书是否有效。
VERIFY_IDENTITY	与 VERIFY_CA 行为一样，并且还验证 Server 端 CA 证书中的 host 是否与实际连接的 hostname 是否一致。

注意

客户端在指定了 `--ssl-mode=VERIFY_CA` 时，需要使用 `--ssl-ca` 来指定 CA 证书。客户端在指定了 `--ssl-mode=VERIFY_IDENTITY` 时，需要指定 CA 证书，且需要使用 `--ssl-key` 指定客户端的私钥和使用 `--ssl-cert` 指定客户端的证书。

关于 MatrixOne 权限管理

MatrixOne 权限管理概述

MatrixOne 权限管理帮助你管理租户、用户帐号生命周期，分配给用户相应的角色，控制 MatrixOne 中资源的访问权限。当数据库或集群单位中存在多个用户时，权限管理确保用户只访问已被授权的资源，赋予用户最少权限原则可降低企业信息安全风险。 MatrixOne 也可以通过权限管理实现多租户方案。在 MatrixOne 中，每个租户在集群中所拥有的数据或资源被安全的隔离，跨集群单位的用户不可访问其他集群单位的资源，在该集群中被赋权访问资源的用户才有权访问本集群单位内的资源。

MatrixOne 权限管理特性

MatrixOne 的权限管理是结合了基于角色的访问控制 (RBAC，Role-based access control) 和自主访问控制 (DAC，Discretionary access control) 两种安全模型设计和实现的，这两种安全模型是中立的访问控制机制，主要围绕角色和权限授权策略。它既保证了数据访问的安全性，又给数据库运维人员提供了灵活且便捷的管理方法。

- **基于角色的访问控制（RBAC）**：将权限分配给角色，再将角色分配给用户。

- **自主访问控制（DAC）**：每个对象都有一个所有者，所有者可以设置和授予对该对象的访问权限。

关键概念

对象

在 MatrixOne 中，为了方便管理多种操作权限，于是便把权限封装在一个实体内，这个实体就是**对象**。

例如，`Select`，`Insert`，`Update` 等操作权限，便封装在了 Table 对象内。更多关于对象权限的信息请参考 [MatrixOne 权限分类]。

对象与对象之间的关系

如下图中所示，从上之下，高层级对象可以创建（或删除）低层级对象。

上图中的层级关系均为 1：n 的关系，即，一个集群中可以创建多个租户（Account），一个租户下可以创建多个用户和角色，一个数据库中可以创建多个表和视图。

在 MatrixOne 中，尽管每个对象中的操作权限是相互独立的（例如 Database 对象中的 `SHOW TABLES` 权限和 Table 对象中的 `SELECT` 权限并没有直接关系），但对象之间的创建仍具有一定关联，例如 Database 对象中的 `CREAT TABLE` 权限可以创建 Table 对象，这便形成了对象之间的层级关系，

那么，由于高层级对象可以创建低层级对象，那么较高层级的对象就是**对象的创建者 (Owner)**。

对象的创建者（Owner)

当一个对象被创建后，创建者便是这个对象的 Owner，它具有管理这个对象的最高权限（即 **Ownership 权限**，它是对象内所封装的所有权限），那么 Owner 的操作权限集合了该对象的所有操作权限。

例如 Table 对象有 `Select`，`Insert`，`Update`，`Delete`，`Truncate`，`Ownership` 权限，如果一个角色拥有了某个 Table 的 Ownership 权限，那么该角色等同于拥有了 `Select`，`Insert`，`Update`，`Delete`，`Truncate` 权限。

由于权限、角色和用户之间的传递性，你可以把对象的创建者（以下称为对象 Owner）理解为一个角色。

如何理解对象的创建者是一个角色呢?

一个用户可以同时拥有多个角色，比如 User A 拥有 Role 1 和 Role 2，还有 Role 3 这三个角色，每个角色拥有的权限不同，如下图所示，帮助你快速理解这一行为：

假如 User A 当前正在使用的角色为 Role 1，User A 需要创建一个新的用户 New user B，可是当前 Role 1 这个角色没有创建新用户的权限，Role 2 拥有创建新用户的权限，那么 User A 需要切换到 Role 2 这个角色，然后再创建新的用户。那么，New user B 的 Owner 是 Role 2，其他角色 Role 1 和 Role 3 并不能拥有 New user B 的所有权。

对象的 Owner 要点

- 对象的 Owner 是一个角色，对象最初的 Owner 是创建它的角色。
- 一个对象的 Owner 在任意时刻有且只有一个。
- 一个角色可以创建多个对象，因此一个角色可以是多个对象的 Owner。
- 角色本身也是一个对象，因此角色也有 Owner。
- 当对象的 Owner 被删除时，该对象的 Owner 会自动变更为被删除角色的 Owner。
- Owner 可以转移给另一个角色。

Note: *ACCOUNTADMIN* (租户管理员角色，租户被创建后即自动生成) 虽然不是租户内所用对象的 Owner，但它拥有所有对象的 Ownership 权限。

集群

集群是 MatrixOne 权限管理中是最高层级的对象，当部署完 MatrixOne 后便创建了集群这个对象。

Tip: 对集群对象的操作权限的集合被称为系统权限。

租户

MatrixOne 集群内可以创建和管理多个数据和用户权限体系完全隔离的租户，并对这些资源隔离的租户进行管理，这种多租户功能既节省了部署和运维多套数据业务系统的成本，又能利用租户间的硬件资源共享最大限度的节约机器成本。

在 MatrixOne 中将租户称为 Account。

系统租户

为了兼容传统非多租户数据库的使用习惯，MatrixOne 在集群创建完成后会自动新建一个系统默认租户，也就是**系统租户**，即 Sys Account，如果你现在只有一套数据业务系统需要 MatrixOne 管理，便不需要创建更多的租户，直接登录并访问系统租户 (Sys Account) 即可。

角色

角色也是一个对象，它是 MatrixOne 中用来管理和分配权限的对象。

在租户中，用户如果没有被赋予角色，那么用户就不能做任何操作。首先，需要先有一个高权限的账号先做一些初期的资源分配，比如说，由**系统租户**或者**租户**创建一些角色和用户，

将对象权限授予给角色，再将角色赋予给用户，这个时候，用户就可以对对象进行操作了。

设立**角色**，是为了节省相同权限授予的操作成本。p1，p2，p3 这三个权限都需要被授予给用户 u1，u2，u3，你只需要先将 p1，p2，p3 授予角色 r1，再将角色 r1 一次性授予用户 u1，u2，u3，相比把每个权限都分别授予每个用户来说，操作上更为简单，并且随着用户和权限数目的增加，这一优势会越发明显。同时，角色的出现进一步抽象了权限集合及其关系，对于后期的权限维护也十分方便。

MatrixOne 在集群和租户 (Account) 创建后，会自动创建一些默认角色和用户（详见下面的**初始化访问**章节），这些角色具有最高管理权限，用于在最开始管理集群和租户 (Account)，我们不建议您将这些角色授予日常执行 SQL 的用户，权限过高会引入更多的安全问题，因此，MatrixOne 支持创建自定义角色，您可以根据用户的业务需要自定义角色，再将适合的权限赋予这些角色。

角色要点

在 MatrixOne 中，角色的行为细节如下：

- 一个角色可以被授予多个权限。
- 一个角色可以授予给多个用户。
- 一个角色可以将其权限传递给另一个角色。
  - 将某一角色的全部权限给另一个角色使用，例如将 role1 的所有权限传递给 role2 使用，那么 role2 继承了 role1 的权限。
- 角色和用户仅在各自的租户 (Account) 内生效，包括系统租户 (Sys Account)。

> **注意**
>
> 1. 角色的权限继承是动态的，如果被继承角色的权限发生了变化，那么继承角色所继承的权限范围也会动态变化。
> 2. 角色的继承关系不能成环。例如，role1 继承了 role2，role 2 继承了 role3，role3 继承了 role1。
> 3. 角色间的权限传递使得权限管理更加便捷，但同时也存在风险，为此，MatrixOne 只允许具有 *Manage Grants* 权限的角色才能做这样的操作，该权限被默认赋予给系统默认角色 *MOADMIN* 或 *ACCOUNTADMIN* 中，并且不建议在新建自定义角色是将该权限授予给自定义角色。

角色切换

一个用户被授予多个角色，用于执行不同类型的数据业务。

主要角色：用户在某一时刻只能使用其中一个角色，我们称当前所使用的这个角色为**主要角色**。 **次要角色**：除了主要角色之外该用户所拥有的其他角色集合称为**次要角色**。

在默认情况下，如果用户想去执行另一个角色权限的 SQL 时，需要先切换角色（即 `set role \<role>`）。此外，为了兼容经典数据库的权限行为，MatrixOne 还支持开启*使用次要角色*的功能：使用 `set secondary role all`，执行这条 SQL 后，该用户便可同时拥有他所有角色的权限了，执行 `set secondary role none` 即可关闭此功能。

应用场景

资源隔离场景介绍

A 公司购买了 MatrixOne 集群，并且完成了部署。由于 A 公司规模比较大，业务线多且复杂，数据量也非常庞大，想要针对某个业务线开发一款应用程序，假设命名为 *BusinessApp*，但是需要跟其他业务线的数据进行隔离，那么 MatrxiOne 怎么隔离出这些数据资源、权限资源呢?

完成部署 MatrixOne 集群，研发部门的 *Tom* 获取到集群管理员的账号，公司指派他来完成资源隔离这一任务。*Tom* 需要这么做：

1. *Tom* 使用集群管理员的账号登录 MatrixOne。

2. *Tom* 需要先创建两个租户，租户账号一个是 *BusinessAccount*，一个是 *ElseAccount*。

- *BusinessAccount* 内的数据资源主要用于开发应用程序 *BusinessApp*。
- *ElseAccount* 内的数据资源可以用于其他业务目的。

关于资源隔离的具体实操，可以参见[快速开始：验证资源隔离](#)。

用户创建和授权场景介绍

还是沿用上面的场景示例，*Tom* 把 *BusinessAccount* 这个租户账号给了公司的数据管理员 *Robert*，让 *Robert* 去分配新的用户账号和权限给其他研发同事。

研发同事 *Joe* 是这个 A 公司项目 *BusinessApp* 的应用开发者，*Joe* 有一个开发任务，*Joe* 需要使用数据库内所有的数据。那么 *Robert* 就要帮 *Joe* 开通账号，给 *Joe* 授权：

1. *Robert* 先给 *Joe* 创建了一个用户账号（即，用户），名字叫做 *Joe_G*，*Joe* 就使用 *Joe_G* 这个账号登录到 MatrixOne。
2. *Robert* 又给 *Joe* 创建了一个角色，名字叫做 *Appdeveloper*，并且把 *Appdeveloper* 角色赋予给 *Joe* 的用户账号 *Joe_G* 上。
3. *Robert* 又给角色 *Appdeveloper* 授予了 *ALL ON DATABASE* 的权限。
4. *Joe* 就可以使用 *Joe_G* 这个账号登录到 MatrixOne，并且全权操作数据库进行开发了。

关于用户创建和授权的具体实操，可以参见[快速开始：创建新租户，并由新租户创建用户、创建角色和授权](#)。

初始化访问

初始化集群或账户后，系统会自动生成一些默认用户和默认角色：

用户名	解释	所拥有的角色	所拥有的权限	描述
root	集群管理员	MOADMIN	创建、编辑、删除租户	集群创建后自动生成并授予
root	系统租户管理员	MOADMIN	管理系统租户下的所有资源，包含用户、角色、数据库/表/视图，授权管理	集群创建后自动生成并授予
<自定义>	租户管理员	ACCOUNTADMIN	管理普通租户下的所有资源，包含用户、角色、数据库/表/视图，授权管理	租户被创建后自动生成并授予

用户名	解释	所拥有的角色	所拥有的权限	描述
所有用户	普通用户	PUBLIC	连接 MatrixOne	所有用户被创建后，自动被授予 PUBLIC 角色

马上开始

- 快速开始：创建租户，验证资源隔离
- 快速开始：创建租户，并由新租户创建用户、创建角色和授权
- 快速了解：典型的应用场景

最佳实践

以下是 MatrixOne 中的典型角色以及建议的最低权限，供你进行参考。

负责数据库资源（用户、角色、权限）管理的工程师

- **数据库管理员**
  - 主要工作职能：管理租户内的所有配置信息、用户权限、备份恢复、性能调优、故障排查
  - 参考授予角色：默认创建租户时生成的管理员角色 accountadmin。
  - 参考授予权限：用户管理（CREATE USER，ALTER USER，DROP USER）、权限管理（MANAGE GRANTS）

负责数据管理（ALL ON ACCOUNT）的工程师

- **数据运维工程师**
  - 主要工作职能：管理租户内的所有数据与元数据信息，以及数据的权限授权
  - 参考授予权限：租户级别的数据管理（ALL ON ACCOUNT）
- **应用开发者**
  - 主要工作职能：对开发环境租户下特定数据库进行操作，并拥有系统租户的只读权限
  - 参考授予权限：数据库级别的数据管理（ALL ON DATABASE）、系统数据库只读（SELECT ON DATABASE）
- **应用系统管理工程师**
  - 主要工作职能：对生产环境租户下特定数据库进行操作
  - 参考授予权限：数据库级别的数据管理（ALL ON DATABASE）
- **系统监控工程师**
  - 主要工作职能：监控租户下所有的系统统计信息与错误信息
  - 参考授予权限：所有系统数据库的只读权限（SELECT ON DATABASE）

快速开始：创建租户，验证资源隔离

初始化接入 MatrixOne 集群，系统会自动生成一个默认账号，即集群管理员。集群管理员被自动默认赋予管理租户账号的权限，但不能管理租户下的资源。

本篇文档将指导你使用集群管理员的账号创建两个新的租户，并赋予租户管理员的权限，并检查是否实现了租户之间的资源隔离。

前提条件

- 已完成 MatrixOne 集群的部署与连接。
- 已获取集群管理员用户名和密码（用户名一般为 root，密码请联系 MatrixOne 的产品经理或者销售代表获取）。

操作步骤

1. 使用集群管理员的用户名（默认 root）和密码登录 MatrixOne：

```
mysql -h 127.0.0.1 -P 6001 -u root -p
```

2. 创建新的租户：

   - 租户 *a1* 的登录用户名和密码分别为：admin1，test123
   - 租户 *a2* 的登录用户名和密码分别为：admin2，test456

```
create account a1 ADMIN_NAME 'admin1' IDENTIFIED BY 'test123';
create account a2 ADMIN_NAME 'admin2' IDENTIFIED BY 'test456';
```

3. 使用 admin1 登录租户 *a1*，并创建数据表 *db1.t1*：

```
mysql -h 127.0.0.1 -P 6001 -u a1:admin1 -p
create database db1;
create table db1.t1(c1 int,c2 varchar);
insert into db1.t1 values (1,'shanghai'),(2,'beijing');
```

   验证租户 *a1* 是否成功创建表：

```
mysql> select * from db1.t1;
+------+----------+
| c1 | c2 |
+------+----------+
| 1 | shanghai |
| 2 | beijing |
+------+----------+
2 rows in set (0.01 sec)
```

4. 使用 admin2 登录租户 *a2*：

```
mysql -h 127.0.0.1 -P 6001 -u a2:admin2 -p
```

查看租户 *a1* 中的 *db1.t1* 数据：

```
mysql> select * from db1.t1;
ERROR 1064 (HY000): SQL parser error: table "t1" does not exist
```

上述命令运行报错，证明在租户 *a2* 中，并不能看到租户 *a1* 中的数据库 *db1*：

5. 在租户 *a2* 中也可以创建库 *db1* 和表 *db1.t1*：

```
mysql> create database db1;
Query OK, 0 rows affected (0.03 sec)

mysql> create table db1.t1(c1 int,c2 varchar);
Query OK, 0 rows affected (0.05 sec)

mysql> insert into db1.t1 values (3,'guangzhou');
Query OK, 1 row affected (0.05 sec)
```

在租户 *a2* 的 *db1.t1* 这张表内插入与租户 *a1* 中表 *db1.t1* 不同的数据并查看：

```
mysql> insert into db1.t1 values (3,'guangzhou');
Query OK, 1 row affected (0.05 sec)

mysql> select * from db1.t1;
+------+-----------+
| c1 | c2 |
+------+-----------+
| 3 | guangzhou |
+------+-----------+
1 row in set (0.01 sec)
```

可以看到，即使与租户 *a1* 中的数据库与表重名，但是这两个数据库与表互不干扰，完全隔离。

快速开始：创建新租户，并由新租户创建用户、创建角色和授权

初始化接入 MatrixOne 集群，系统会自动生成一个默认账号，即集群管理员。集群管理员默认用户名为 *root*，*root* 既是集群管理员，同时也是系统租户管理员，*root* 可以创建和管理其他普通租户（非系统租户管理员）。

本篇文档将指导你创建一个新的租户，并切换至新租户登录，用新租户账号创建用户、创建角色、创建权限，并赋予用户权限。

前提条件

- 已完成 MatrixOne 集群的部署与连接。
- 已获取集群管理员用户名和密码（用户名一般为 root，密码请联系 MatrixOne 的产品经理或者销售代表获取）。

操作步骤

步骤一：创建新租户

1. 使用集群管理员的用户名（默认 root）和密码登录 MatrixOne：

```
mysql -h 127.0.0.1 -P 6001 -u root -p
```

2. 创建一个新的租户 *a1*，用户名和密码分别为：admin，test123：

```
create account a1 ADMIN_NAME 'admin' IDENTIFIED BY 'test123';
```

查看集群中的所有租户信息（仅 root 可查看）：

```
mysql> select *from mo_account;
+------------+--------------+--------+---------------------+----------
------+----------------+
| account_id | account_name | status | created_time | comments |
suspended_time |
+------------+--------------+--------+---------------------+----------
------+----------------+
| 1 | a1 | open | 2022-12-19 14:47:19 | | NULL |
| 0 | sys | open | 2022-12-07 11:00:58 | system account | NULL |
+------------+--------------+--------+---------------------+----------
------+----------------+
```

步骤二：登录新租户账号，创建用户、创建角色和授权

1. 你可以重新打开一个新的会话，使用 admin 登录租户 *a1*：

```
mysql -h 127.0.0.1 -P 6001 -u a1:admin -p
```

2. 现在你可以作为租户 *a1* 查看租户下的默认用户和角色：

```
mysql> select * from mo_catalog.mo_role;
+---------+-------------+---------+-------+---------------------+----
------+
| role_id | role_name | creator | owner | created_time | comments |
+---------+-------------+---------+-------+---------------------+----
------+
| 2 | accountadmin | 0 | 0 | 2022-12-19 14:47:20 | |
| 1 | public | 0 | 0 | 2022-12-19 14:47:20 | |
+---------+-------------+---------+-------+---------------------+----
------+
2 rows in set (0.01 sec)

mysql> select * from mo_catalog.mo_user;
+---------+-----------+-----------+----------------------+--------+--
------------------+-------------+------------+---------+-------+----
----------+
| user_id | user_host | user_name | authentication_string | status |
created_time | expired_time | login_type | creator | owner |
default_role |
+---------+-----------+-----------+----------------------+--------+--
------------------+-------------+------------+---------+-------+----
----------+
| 2 | localhost | admin | test123 | unlock | 2022-12-19 14:47:20 |
NULL | PASSWORD | 0 | 0 | 2 |
+---------+-----------+-----------+----------------------+--------+--
------------------+-------------+------------+---------+-------+----
----------+
1 row in set (0.00 sec)
```

租户 a1 被创建成功后便默认拥有了租户管理员权限，所以可以查看租户 a1 下的系统表信息。在 *mo_user* 表中可以观察到当前有一个用户名为 *admin* 的用户账号，即创建租户时指定的；此外，还有 *accountadmin* 和 *public* 两个默认角色：

- *accountadmin* 拥有租户的最高权限，且默认授予用户名为 *admin* 的账号；
- 系统会为每一个新的普通用户默认授权 *public* 角色，*public* 角色初始化的权限是 `connect`，即连接 MatrixOne。

此外，你还可以在系统表中查看到这些默认角色的权限集合：

```
mysql> select * from mo_role_privs;
+---------+--------------+----------+--------+--------------+---------
-----------+----------------+------------------+-------------------
-+-------------------+
| role_id | role_name | obj_type | obj_id | privilege_id |
privilege_name | privilege_level | operation_user_id | granted_time |
with_grant_option |
+---------+--------------+----------+--------+--------------+---------
-----------+----------------+------------------+-------------------
-+-------------------+
| 2 | accountadmin | account | 0 | 3 | create user | * | 0 | 2022-12-
19 14:47:20 | true |
| 2 | accountadmin | account | 0 | 4 | drop user | * | 0 | 2022-12-19
14:47:20 | true |
| 2 | accountadmin | account | 0 | 5 | alter user | * | 0 | 2022-12-19
14:47:20 | true |
| 2 | accountadmin | account | 0 | 6 | create role | * | 0 | 2022-12-
19 14:47:20 | true |
| 2 | accountadmin | account | 0 | 7 | drop role | * | 0 | 2022-12-19
14:47:20 | true |
| 2 | accountadmin | account | 0 | 9 | create database | * | 0 | 2022-
12-19 14:47:20 | true |
| 2 | accountadmin | account | 0 | 10 | drop database | * | 0 | 2022-
12-19 14:47:20 | true |
| 2 | accountadmin | account | 0 | 11 | show databases | * | 0 | 2022-
12-19 14:47:20 | true |
| 2 | accountadmin | account | 0 | 12 | connect | * | 0 | 2022-12-19
14:47:20 | true |
| 2 | accountadmin | account | 0 | 13 | manage grants | * | 0 | 2022-
12-19 14:47:20 | true |
| 2 | accountadmin | account | 0 | 14 | account all | * | 0 | 2022-12-
19 14:47:20 | true |
| 2 | accountadmin | database | 0 | 18 | show tables | * | 0 | 2022-
12-19 14:47:20 | true |
| 2 | accountadmin | database | 0 | 20 | create table | * | 0 | 2022-
12-19 14:47:20 | true |
| 2 | accountadmin | database | 0 | 23 | drop table | * | 0 | 2022-12-
19 14:47:20 | true |
| 2 | accountadmin | database | 0 | 26 | alter table | * | 0 | 2022-
12-19 14:47:20 | true |
| 2 | accountadmin | database | 0 | 21 | create view | * | 0 | 2022-
12-19 14:47:20 | true |
| 2 | accountadmin | database | 0 | 24 | drop view | * | 0 | 2022-12-
19 14:47:20 | true |
```

```
| 2 | accountadmin | database | 0 | 27 | alter view | * | 0 | 2022-12-
19 14:47:20 | true |
| 2 | accountadmin | database | 0 | 28 | database all | * | 0 | 2022-
12-19 14:47:20 | true |
| 2 | accountadmin | database | 0 | 29 | database ownership | * | 0 |
2022-12-19 14:47:20 | true |
| 2 | accountadmin | table | 0 | 30 | select | *.* | 0 | 2022-12-19
14:47:20 | true |
| 2 | accountadmin | table | 0 | 31 | insert | *.* | 0 | 2022-12-19
14:47:20 | true |
| 2 | accountadmin | table | 0 | 32 | update | *.* | 0 | 2022-12-19
14:47:20 | true |
| 2 | accountadmin | table | 0 | 33 | truncate | *.* | 0 | 2022-12-19
14:47:20 | true |
| 2 | accountadmin | table | 0 | 34 | delete | *.* | 0 | 2022-12-19
14:47:20 | true |
| 2 | accountadmin | table | 0 | 35 | reference | *.* | 0 | 2022-12-19
14:47:20 | true |
| 2 | accountadmin | table | 0 | 36 | index | *.* | 0 | 2022-12-19
14:47:20 | true |
| 2 | accountadmin | table | 0 | 37 | table all | *.* | 0 | 2022-12-19
14:47:20 | true |
| 2 | accountadmin | table | 0 | 38 | table ownership | *.* | 0 |
2022-12-19 14:47:20 | true |
| 2 | accountadmin | table | 0 | 41 | values | t | 0 | 2022-12-19
14:47:20 | true |
| 1 | public | account | 0 | 12 | connect | * | 0 | 2022-12-19
14:47:20 | true |
+---------+--------------+----------+--------+--------------+---------
-----------+-----------------+-------------------+-------------------
-+-------------------+
```

3. 在租户 a1 中，创建新的用户和角色：

    - 用户 u1 的用户名和密码分别为：u1，user123
    - 用户 *u2* 的用户名和密码分别为：u2，user456
    - 角色 r1 的命名为：r1
    - 角色 r2 的命名为：r2

```
create user u1 identified by 'user123';
create user u2 identified by 'user456';
create role r1;
create role r2;
```

4. 创建数据库 *db1*，并在 *db1* 中创建表 *t1*：

```
create database db1;
create table db1.t1(c1 int,c2 varchar);
```

5. 将 *db1.t1* 的 `select` 权限授予给 *r1*，`insert` 权限授予给 *r2*：

```
grant select on table db1.t1 to r1;
grant insert on table db1.t1 to r2;
```

6. 将角色 *r1* 授予给用户 *u1*；将角色 *r2* 授予给用户 *u2*：

```
grant r1 to u1;
grant r2 to u2;
```

此时，新建的用户、角色、对象权限关系如下图所示：

步骤三：验证授权生效

分别使用用户 *u1* 和 *u2* 登录租户 *a1*，验证权限是否生效。

由于 *u2* 被授予了 *r2* 角色，且 *r2* 被授予了 *db1.t1* 的 `insert` 权限，所以 *u2* 具备 *db1.t1* 的 `insert` 权限，即可以向 *db1.t1* 插入数据，

使用 *u1* 登录 *a1* 进行验证：

```
mysql -h 127.0.0.1 -P 6001 -u a1:u2:r2 -p

mysql> insert into db1.t1 values (1,'shanghai'),(2,'beijing');
Query OK, 2 rows affected (0.04 sec)

mysql> select * from db1.t1;
ERROR 20101 (HY000): internal error: do not have privilege to execute the
statement
```

u2 可以成功向表 *db1.t1* 插入数据，但无法查看 *db1.t1* 表里的数据。

同样的，你可以使用 *u1* 登录 *a1* 进行权限验证：

```
mysql -h 127.0.0.1 -P 6001 -u a1:u1:r1 -p

mysql> select * from db1.t1;
+------+----------+
| c1 | c2 |
+------+----------+
| 1 | shanghai |
| 2 | beijing |
+------+----------+
2 rows in set (0.01 sec)

mysql> insert into db1.t1 values (3,'guangzhou');
ERROR 20101 (HY000): internal error: do not have privilege to execute the
statement
```

如上述代码所示，*u1* 可以成功的查询表 *db1.t1* 的数据，但不能向其插入数据。

> **注意**
>
> 上述操作步骤中，更多有关查看的系统表信息，参见 MatrixOne 系统数据库和表

权限管理操作

管理租户

- 前提条件：拥有集群管理员（默认账户为 root）才可以进行租户管理。

有关 root 账号对应的角色和权限如下表所示：

用户名	解释	所拥有的角色	所拥有的权限	描述
root	集群管理员	MOADMIN	创建、编辑、删除租户	集群创建后自动生成并授予
root	系统租户管理员	MOADMIN	管理系统租户下的所有资源，包含用户、角色、数据库/表/视图，授权管理	集群创建后自动生成并授予

创建租户

SQL 语法

```
create account \<account_name> admin_name='\<user_name>' identified by '\
<password>';
```

参数解释

参数	参数解释
<account_name>	新建租户的名称
<user_name>	新建租户的管理员用户名，其会被自动授予租户的最高权限角色，即 `ACCOUNTADMIN`
<password>	新建的租户管理员密码

更多信息，参见 CREATE ACCOUNT。

查看租户

SQL 语法

```
select * from mo_catalog.mo_account;
```

删除租户

SQL 语法

```
drop account if exists \<account_name>;
```

参数解释

参数	参数解释
<account_name>	需要删除的租户名称

> **注意**
>
> 删除租户后则无法恢复，包括租户账号下的所有数据，请谨慎使用。

更多信息，参见 DROP ACCOUNT。

管理用户

创建用户

- 前提条件：拥有 `CREATE USER` 权限。

  · 默认拥有这个权限的角色为 MOADMIN 或 ACCOUNTADMIN：集群管理员（默认账户为 root）和由集群管理员创建的租户管理员默认拥有权限。

- 操作说明：在当前租户中创建一个用户的用户名和密码。

SQL 语法

```
create user \<user_name> identified by '\<password>';
```

参数解释

参数	参数解释
<user_name>	新建用户的名称
<password>	新建的用户密码

更多信息，参见 CREATE USER。

查看用户

- 前提条件：拥有查看用户的权限。

  · 默认拥有这个权限的角色为 MOADMIN 或 ACCOUNTADMIN：集群管理员（默认账户为 root）和由集群管理员创建的租户管理员默认拥有权限。

- 操作说明：查看当前租户下所有的用户。

SQL 语法

```
select * from mo_catalog.mo_user;
```

删除用户

- 前提条件：拥有 `DROP USER` 权限。

  · 默认拥有这个权限的角色为 MOADMIN 或 ACCOUNTADMIN：集群管理员（默认账户为 root）和由集群管理员创建的租户管理员默认拥有权限。

- 操作说明：删除当前租户下的指定的用户。

SQL 语法

```
drop user if exist \<user_name>;
```

参数解释

参数	参数解释
<user_name>	新建用户的名称

> **注意**
>
> 删除用户时，需要先停止用户当前存在的会话，否则删除失败。

更多信息，参见 DROP USER。

管理角色

创建角色

- 前提条件：拥有 `CREATE ROLE` 权限。

  · 默认拥有这个权限的角色为 MOADMIN 或 ACCOUNTADMIN：集群管理员（默认账户为 root）和由集群管理员创建的租户管理员默认拥有权限。

- 操作说明：在当前租户下创建一个自定义角色。

SQL 语法

```
create role \<role_name>;
```

参数解释

参数	参数解释
<role_name>	新建角色的名称

更多信息，参见 CREATE ROLE。

查看角色

- 前提条件：拥有查看角色权限。

  · 默认拥有这个权限的角色为 MOADMIN 或 ACCOUNTADMIN：集群管理员（默认账户为 root）和由集群管理员创建的租户管理员默认拥有权限。

- 操作说明：查看当前租户下所有的角色。

SQL 语法

```
select * from mo_catalog.mo_role;
```

切换角色

- 前提条件：拥有 `SET ROLE` 权限。默认所有用户都拥有这个权限。
- 操作说明：在租户中切换用户的角色，获取其他角色的权限，以便执行相应的 SQL。

SQL 语法

```
set role \<role_name>;
```

参数解释

参数	参数解释
<role_name>	角色的名称

更多信息，参见 SET ROLE。

删除角色

- 前提条件：拥有 `DROP ROLE` 权限。

  · 默认拥有这个权限的角色为 MOADMIN 或 ACCOUNTADMIN：集群管理员（默认账户为 root）和由集群管理员创建的租户管理员默认拥有权限。

- 操作说明：删除当前租户下的特定角色。

SQL 语法

```
drop role if exists \<role_name>;
```

参数解释

参数	参数解释
<role_name>	需要删除的角色的名称

> **注意**
>
> 删除某个指定角色时，会同时回收已经被授权的用户的角色。

更多信息，参见 DROP ROLE。

管理权限

向角色授予某个对象权限

- 前提条件：拥有 `MANAGE GRANTS` 权限。

  · 默认拥有这个权限的角色为 MOADMIN 或 ACCOUNTADMIN：集群管理员（默认账户为 root）和由集群管理员创建的租户管理员默认拥有权限。

- 操作说明：向某个角色授予某个对象的某个权限。

SQL 语法

```
grant \<privilege> on \<object_type> \<object_name> to \<role_name>
```

参数解释

参数	参数解释
<privilege>	权限
<object_type>	对象类型
<object_name>	对象名称
<role_name>	被赋予权限的角色

更多信息，参见 DRANT PRIVILEGES。

向角色授予某类对象权限

- 前提条件：拥有 `MANAGE GRANTS` 权限。

  · 默认拥有这个权限的角色为 MOADMIN 或 ACCOUNTADMIN：集群管理员（默认账户为 root）和由集群管理员创建的租户管理员默认拥有权限。

- 操作说明：向角色授予所有数据库/数据表的某个权限。

SQL 语法

```
grant \<privilege> on database * to \<role_name>;
grant \<privilege> on table *.* to \<role_name>;
```

参数解释

参数	参数解释
<privilege>	权限名称
<role_name>	被赋予权限的角色名称

> **注意**
>
> 该操作虽然在授权多个相同类别对象时比较简便，但也很容易发生权限泄漏，请谨慎使用。

更多信息，参见 DRANT PRIVILEGES。

向用户授予角色

- 前提条件：拥有 `MANAGE GRANTS` 权限。

  · 默认拥有这个权限的角色为 MOADMIN 或 ACCOUNTADMIN：集群管理员（默认账户为 root）和由集群管理员创建的租户管理员默认拥有权限。

- 操作说明：向某个用户授予某个角色。

SQL 语法

```
grant \<role_name> to \<user_name>;
```

参数解释

参数	参数解释
<role_name>	被赋予权限的角色
<user_name>	被赋予权限的用户

更多信息，参见 DRANT ROLE。

让一个角色继承另一个角色的权限

- 前提条件：拥有 `MANAGE GRANTS` 权限。

  · 默认拥有这个权限的角色为 MOADMIN 或 ACCOUNTADMIN：集群管理员（默认账户为 root）和由集群管理员创建的租户管理员默认拥有权限。

- 操作说明：让 role_b 继承 role_a 的所有权限。

SQL 语法

```
grant \<role_a> to \<role_b>;
```

> **注意**
>
> 该权限继承为动态继承，若 role_a 的权限发生改变，则 role_b 所继承的权限也会动态更改。MatrixOne 不允许角色环继承，即 role1 继承 role2，role2 继承 role3，但是 role3 继承不能继承 role1。

更多信息，参见 DRANT ROLE。

查看某一用户所拥有的权限

- 前提条件：拥有 `SHOW GRANTS` 权限。

· 默认拥有这个权限的角色为 MOADMIN 或 ACCOUNTADMIN：集群管理员（默认账户为 root）和由集群管理员创建的租户管理员默认拥有权限。

- 操作说明：查看所指定用户当前所拥有的全部权限。

SQL 语法

```
show grants for \<user_name>@\<localhost>
```

参数解释

参数	参数解释
<user_name>	被赋予权限的用户

更多信息，参见 SHOW GRANTS。

回收授权用户的某个角色

- 前提条件：拥有 `REVOKE` 权限。

  · 默认拥有这个权限的角色为 MOADMIN 或 ACCOUNTADMIN：集群管理员（默认账户为 root）和由集群管理员创建的租户管理员默认拥有权限。

- 操作说明：将某一用户的某一角色移除。

SQL 语法

```
revoke \<role_name> from \<user_name>
```

参数解释

参数	参数解释
<role_name>	被赋予权限的角色
<user_name>	被赋予权限的用户

更多信息，参见 REVOKE。

回收角色中的某个对象权限

- 前提条件：拥有 `REVOKE` 权限。

  · 默认拥有这个权限的角色为 MOADMIN 或 ACCOUNTADMIN：集群管理员（默认账户为 root）和由集群管理员创建的租户管理员默认拥有权限。

- 操作说明：回收角色中的某个对象权限。

SQL 语法

```
revoke \<privilege> on \<object_type> \<object_name> to \<role_name>;
```

参数解释

参数	参数解释
<privilege>	权限名称
<object_type>	对象类型
<object_name>	对象名称
<role_name>	被赋予权限的角色

更多信息，参见 REVOKE。

应用场景

场景概述

- 如果你所在的企业部署了 MatrixOne 集群，那么部署完成后，集群初始化时即自动存在一个集群管理员账户，你可以联系 MatrxiOne 的项目经理或销售代表获取账号信息和初始密码。使用集群管理员的账号，你可以新建租户，管理租户生命周期，并将租户账号密码分配给你所在企业对应的负责人。管理租户的操作说明，详见快速开始：创建租户，验证资源隔离或权限管理操作指南。

- 如果你所在的企业仅需使用 MatrixOne 集群租户资源，完成部署后，MatrixOne 集群管理员会帮你开通租户管理员的账号，你可以联系 MatrxiOne 的项目经理或销售代表获取账号信息和初始密码。使用租户管理员的账号，你可以新建用户，管理用户生命周期、租户内的资源（用户、角色和权限），并将用户账号密码分配给你所在企业对应的负责人。管理用户的操作说明，详见快速开始：创建用户、创建角色和授权或权限管理操作指南。

场景一：新建数据管理员并赋权

场景介绍

在实际应用场景中，需要设立一个数据管理员的岗位，他负责管理整个数据库中资源分配的情况，比如说，公司其他成员需要被分配一个用户账号和密码，被分配角色，并被授予最低的使用权限。

前提条件

- 你首先需要拥有**租户管理员**的账号

- 已经连接上 MatrixOne 集群

解决方案

创建一个数据管理员的角色，授予他租户内的全局管理的权限，那么你需要做到如下几点：

- 创建一个新的用户账号，用户名为：dbauser；密码为：123456。
- 给这个用户账号分配一个数据管理员的角色，角色命名为：dba。

- 这个角色需要有以下权限：
  - 拥有租户对象的全部权限：拥有这个权限，那么数据管理员就可以创建新用户、新角色、分配权限给其他用户。
  - 拥有数据库对象的全部权限：拥有这个权限，那么数据管理员就可以新建、编辑、删除数据库。
  - 拥有表对象的全部权限：拥有这个权限，那么数据管理员就可以新建、编辑、删除数据表。

操作步骤

步骤一：租户管理员开通并授权数据库管理员账号

1. 使用你所拥有的租户管理员账号登录租户：

   Note: 此处的租户管理员账号 *account1* 为示例，你可以在创建租户管理员时进行自定义。

   ```
   mysql -h 127.0.0.1 -P 6001 -u account1:admin:admin -p
   ```

2. 创建一个用户账号，命名为 *dbauser*，密码为 *123456*：

   ```
   create user dbauser identified by "123456";
   ```

3. 创建一个数据管理员的角色，命名为 *dba*：

   ```
   create role dba;
   ```

4. 授权权限给角色如下权限：
   - 租户对象的全部权限
   - 数据库对象的全部权限
   - 表对象的全部权限

   ```
   grant all on account * to dba with grant option;
   grant all on database * to dba with grant option;
   grant all on table *.* to dba with grant option;
   ```

5. 授权角色 *dba* 给用户 *dbauser*：

```
grant dba to dbauser;
```

6. 查看权限授予情况：

```
show grants for dbauser@localhost;
```

步骤二：数据管理员登录账号并进行测试

1. 使用数据管理员账号 *dbauser* 登录 MatrixOne：

```
mysql -h 127.0.0.1 -P 6001 -u account1:dbauser:dba -p
```

2. 查看 *dbauser* 所拥有的权限：

```
show grants for dbauser@localhost;
```

3. 查看 *dbauser* 的角色：

```
SET SECONDARY ROLE ALL;
use mo_catalog;
select mu.user_name,mr.role_name from mo_role mr,mo_user
mu,mo_user_grant mug where mu.user_id=mug.user_id and
mr.role_id=mug.role_id and mu.user_name='dbauser';
```

4. 实际操作一个数据库进行验证：

```
drop database if exists test;
create database test;
use test;
create table t1(a int);
insert into t1 values(1),(2),(3);
select * from t1;
```

5. 上面代码表示验证成功。

场景二：新系统上线

场景介绍

应用系统上线时，会根据应用系统的使用需求，创建新的数据库与对应数据库用户，并且授予这个用户拥有目标数据库的所有权限。

前提条件

- 你首先需要具有租户管理员的账号和权限（或者你本身作为一个用户，已经拥有创建新用户并可以授权给新用户数据库对象全部权限）
- 连接上 MatrixOne 集群

解决方案

- 需求 1：应用系统需要一套新的数据库专门应用于应用的开发。
  - 解决方案：创建新的数据库，命名为 *appdb*。
- 需求 2：该应用系统需要专门的角色。
  - 解决方案：创建新的数据库角色，命名为 *approle*，授权给这个角色全部的数据库权限。
- 需求 3：该应用系统需要专门的负责人管理这个数据库。
  - 解决方案：创建新的数据库用户，命名为 *appuser*，把角色授权给这个用户。

操作步骤

步骤一：租户管理员开通并授权数据库用户账号

1. 使用你所拥有的租户管理员账号登录租户：

   Note: 此处的租户管理员账号 *account1* 为示例，你可以在创建租户管理员时进行自定义。

   ```
   mysql -h 127.0.0.1 -P 6001 -u account1:admin:admin -p
   ```

2. 创建应用所需要的数据库，给数据库命名为 *appdb*：

```
create database appdb;
```

3. 创建一个命名为 *approle* 的角色，并授权给这个角色对于数据库 *appdb* 的全部操作权限：

```
create role approle;
grant all on database appdb to approle;
grant all on table appdb.* to approle;
```

4. 创建数据库用户 *appuser*，密码为 *123456*，并将角色 *approle* 分配给 *appuser*：

```
create user appuser identified by "123456" default role approle;
```

步骤二：数据库用户登录账号并进行测试

1. 使用数据库用户账号 *appuser* 登录 MatrixOne：

```
mysql -h127.0.0.1 -utest:appuser -P6001 -p123456
```

2. 验证数据用户账号 *appuser* 的权限：

```
set secondary role all;
use appdb;
create table t1(a int);
insert into t1 values(1),(2),(3);
select * from t1;
drop table t1;
```

3. 上面代码表示验证成功。

关键字

本章介绍 MatrixOne 的关键字，在 MatrixOne 中对保留关键字和非保留关键字进行了分类，你在使用 SQL 语句时，可以查阅保留关键字和非保留关键字。

关键字是 SQL 语句中具有特殊含义的单词，例如 `SELECT`，`UPDATE`，`DELETE` 等等。

- **保留关键字**：关键字中需要经过特殊处理才能作为标识符的字，被称为保留关键字。

  将保留关键字作为标识符使用时，必须使用反引号包裹，否则将产生报错：

```
\\未将保留关键字 select 使用反引号包裹，产生报错
mysql> CREATE TABLE select (a INT);
ERROR 1064 (HY000): SQL parser error: You have an error in your SQL
syntax; check the manual that corresponds to your MatrixOne server
version for the right syntax to use. syntax error at line 1 column 19
near " select (a INT)";

\\正确将保留关键字 select 使用反引号包裹
mysql> CREATE TABLE `select` (a INT);
Query OK, 0 rows affected (0.02 sec)
```

- **非保留关键字**：关键字中可以直接作为标识符，被称为非保留关键字。

  将非保留关键字作为标识符使用时，可以直接使用，无需使用反引号包裹。

```
\\BEGIN 未非保留关键字，可以无需使用反引号包裹
mysql> CREATE TABLE `select` (BEGIN int);
Query OK, 0 rows affected (0.01 sec)
```

> **注意**
>
> 与 MySQL 不同，在 MatrixOne 中，如果使用了限定符*.*，保留关键字如果不使用反引号包裹也会产生报错，建议在创建表和数据库时，避免使用保留关键字：

```
mysql> CREATE TABLE test.select (BEGIN int);
ERROR 1064 (HY000): SQL parser error: You have an error in your SQL
syntax; check the manual that corresponds to your MatrixOne server
version for the right syntax to use. syntax error at line 1 column 24
near "select (BEGIN int)";
```

保留关键字

A

- ADD
- ADMIN_NAME
- ALL
- AND
- AS
- ASC
- ASCII
- AUTO_INCREMENT

B

- BETWEEN
- BINARY
- BY

C

- CASE
- CHAR

- CHARACTER
- CHECK
- COLLATE
- COLLATION
- CONVERT
- COALESCE
- COLUMN_NUMBER
- CONSTRAINT
- CREATE
- CROSS
- CURRENT
- CURRENT_DATE
- CURRENT_ROLE
- CURRENT_USER
- CURRENT_TIME
- CURRENT_TIMESTAMP
- CIPHER

D

- DATABASE
- DATABASES
- DECLARE
- DEFAULT
- DELAYED
- DELETE
- DESC
- DESCRIBE
- DISTINCT
- DISTINCTROW
- DIV

- DROP

E

- ELSE
- ENCLOSED
- END
- ESCAPE
- ESCAPED
- EXCEPT
- EXISTS
- EXPLAIN

F

- FALSE
- FAILED_LOGIN_ATTEMPTS
- FIRST
- FOLLOWING
- FOR
- FORCE
- FOREIGN
- FROM
- FULLTEXT

G

- GROUP
- GROUPS

H

- HAVING
- HOUR

- HIGH_PRIORITY

I

- IDENTIFIED
- IF
- IGNORE
- IMPORT
- IN
- INFILE
- INDEX
- INNER
- INSERT
- INTERVAL
- INTO
- INT1
- INT2
- INT3
- INT4
- INT8
- IS
- ISSUER

J

- JOIN

K

- KEY

L

- LAST

- LEADING
- LEFT
- LIKE
- LIMIT
- LINES
- LOAD
- LOCALTIME
- LOCALTIMESTAMP
- LOCK
- LOCKS
- LOW_PRIORITY

M

- MATCH
- MAXVALUE
- MICROSECOND
- MINUTE
- MOD
- MODUMP

N

- NATURAL
- NODE
- NOT
- NONE
- NULL
- NULLS

O

- ON

- OPTIONAL
- OPTIONALLY
- OR
- ORDER
- OUTER
- OUTFILE
- OVER

P

- PASSWORD_LOCK_TIME
- PARTITION
- PRECEDING
- PRIMARY

Q

- QUICK

R

- RANDOM
- REGEXP
- RENAME
- REPLACE
- RETURNS
- REUSE
- RIGHT
- REQUIRE
- REPEAT
- ROW
- ROWS
- ROW_COUNT

- REFERENCES
- RECURSIVE
- REVERSE

S

- SAN
- SECONDARY
- SSL
- SUBJECT
- SCHEMA
- SCHEMAS
- SELECT
- SECOND
- SEPARATOR
- SET
- SHOW
- SQL_SMALL_RESULT
- SQL_BIG_RESULT
- STRAIGHT_JOIN
- STARTING
- SUSPEND

T

- TABLE
- TABLE_NUMBER
- TABLE_SIZE
- TABLE_VALUES
- TERMINATED
- THEN
- TO

- TRAILING
- TRUE
- TRUNCATE

U

- UNBOUNDED
- UNION
- UNIQUE
- UPDATE
- USE
- USING
- UTC_DATE
- UTC_TIME
- UTC_TIMESTAMP

V

- VALUES

W

- WHEN
- WHERE
- WEEK
- WITH

非保留关键字

A

- ACCOUNT
- ACCOUNTS
- AGAINST

- AVG_ROW_LENGTH
- AUTO_RANDOM
- ATTRIBUTE
- ACTION
- ALGORITHM
- ANY

B

- BEGIN
- BIGINT
- BIT
- BLOB
- BOOL

C

- CHAIN
- CHECKSUM
- CLUSTER
- COMPRESSION
- COMMENT_KEYWORD
- COMMIT
- COMMITTED
- CHARSET
- COLUMNS
- CONNECTION
- CONSISTENT
- COMPRESSED
- COMPACT
- COLUMN_FORMAT
- CASCADE

D

- DATA
- DATE
- DATETIME
- DECIMAL
- DYNAMIC
- DISK
- DO
- DOUBLE
- DIRECTORY
- DUPLICATE
- DELAY_KEY_WRITE

E

- ENUM
- ENCRYPTION
- ENFORCED
- ENGINE
- ENGINES
- ERRORS
- EXPANSION
- EXPIRE
- EXTENDED
- EXTENSION
- EXTERNAL

F

- FORMAT
- FLOAT_TYPE
- FULL

- FIXED
- FIELDS
- FORCE_QUOTE

G

- GEOMETRY
- GEOMETRYCOLLECTION
- GLOBAL
- GRANT

H

- HASH
- HEADER
- HISTORY

I

- INT
- INTEGER
- INDEXES
- ISOLATION

J

- JSON

K

- KEY_BLOCK_SIZE
- KEYS

L

- LANGUAGE

- LESS
- LEVEL
- LINESTRING
- LINEAR
- LIST
- LONGBLOB
- LONGTEXT
- LOCAL
- LOW_CARDINALITY

M

- MAX_CONNECTIONS_PER_HOUR
- MAX_FILE_SIZE
- MAX_QUERIES_PER_HOUR
- MAX_ROWS
- MAX_UPDATES_PER_HOUR
- MAX_USER_CONNECTIONS
- MEDIUMBLOB
- MEDIUMINT
- MEDIUMTEXT
- MEMORY
- MIN_ROWS
- MODE
- MONTH
- MULTILINESTRING
- MULTIPOINT
- MULTIPOLYGON

N

- NAMES

- NCHAR
- NUMERIC
- NEVER
- NO

O

- OFFSET
- ONLY
- OPTIMIZE
- OPEN
- OPTION

P

- PACK_KEYS
- PASSWORD
- PARTIAL
- PARTITIONS
- POINT
- POLYGON
- PROCEDURE
- PROFILES
- PROXY

Q

- QUARTER
- QUERY

R

- ROLE
- RANGE

- READ
- REAL
- REORGANIZE
- REDUNDANT
- REPAIR
- REPEATABLE
- RELEASE
- REVOKE
- REPLICATION
- ROW_FORMAT
- ROLLBACK
- RESTRICT

S

- SESSION
- SERIALIZABLE
- SHARE
- SIGNED
- SMALLINT
- SNAPSHOT
- SOME
- SPATIAL
- START
- STATUS
- STORAGE
- STREAM
- STATS_AUTO_RECALC
- STATS_PERSISTENT
- STATS_SAMPLE_PAGES
- SUBPARTITIONS

- SUBPARTITION
- SIMPLE
- S3OPTION

T

- TABLES
- TEXT
- THAN
- TINYBLOB
- TIME
- TIMESTAMP
- TINYINT
- TINYTEXT
- TRANSACTION
- TRIGGER
- TRIGGERS
- TYPE

U

- UNCOMMITTED
- UNKNOWN
- UNSIGNED
- UNUSED
- UNLOCK
- URL
- USER

V

- VARBINARY
- VARCHAR

- VARIABLES
- VIEW

W

- WRITE
- WARNINGS
- WORK

X

- X509

Y

- YEAR

Z

- ZEROFILL

CREATE DATABASE

语法说明

`CREATE DATABASE` 语句同于创建一个数据库。

语法结构

```
> CREATE DATABASE [IF NOT EXISTS] <database_name> [create_option] ...

> create_option: [DEFAULT] {
CHARACTER SET [=] charset_name
| COLLATE [=] collation_name
| ENCRYPTION [=] {'Y' | 'N'}
}
```

示例

```
CREATE DATABASE IF NOT EXISTS test01;
```

预期结果

你可以使用 `SHOW DATABASES` 检查数据库是否已创建。

```
mysql> show databases;
+--------------------+
| Database |
+--------------------+
| mo_task |
| information_schema |
| mysql |
| system_metrics |
| system |
| test01 |
| mo_catalog |
+--------------------+
10 rows in set (0.01 sec)
```

可以看到，除了已存在的 6 个系统数据库以外，新的数据库 *test01* 已经创建。

限制

- 目前只支持 `UTF-8` 字符集。
- `CHARACTER SET`，`COLLATE`，`ENCRYPTION` 目前可以使用但无法生效。

CREATE INDEX

语法说明

在表中创建索引，以便更加快速高效地查询数据。

你无法看到索引，索引只能被用来加速搜索/查询。

更新一个包含索引的表需要比更新一个没有索引的表花费更多的时间，这是由于索引本身也需要更新。因此，理想的做法是仅仅在常常被搜索的列（以及表）上面创建索引。

语法结构

```
> CREATE [UNIQUE] INDEX index_name
ON tbl_name (key_part,...)
COMMENT 'string'
```

语法释义

CREATE UNIQUE INDEX 语法

在表上创建一个唯一的索引。不允许使用重复的值：唯一的索引意味着两个行不能拥有相同的索引值。

示例

```
drop table if exists t1;
create table t1(id int PRIMARY KEY,name VARCHAR(255),age int);
insert into t1 values(1,"Abby", 24);
insert into t1 values(2,"Bob", 25);
insert into t1 values(3,"Carol", 23);
insert into t1 values(4,"Dora", 29);
create unique index idx on t1(name);
mysql> select * from t1;
+------+-------+------+
| id | name | age |
+------+-------+------+
| 1 | Abby | 24 |
| 2 | Bob | 25 |
| 3 | Carol | 23 |
| 4 | Dora | 29 |
+------+-------+------+
4 rows in set (0.00 sec)

mysql> show create table t1;
+-------+-----------------------------------------------------------------
-------------------------------------------------------------------------
---------+
| Table | Create Table |
+-------+-----------------------------------------------------------------
-------------------------------------------------------------------------
---------+
| t1 | CREATE TABLE `t1` (
`id` INT NOT NULL,
`name` VARCHAR(255) DEFAULT NULL,
`age` INT DEFAULT NULL,
PRIMARY KEY (`id`),
UNIQUE KEY `idx` (`name`)
) |
+-------+-----------------------------------------------------------------
-------------------------------------------------------------------------
---------+
1 row in set (0.01 sec)


create table t2 (
col1 bigint primary key,
col2 varchar(25),
```

```
col3 float,
col4 varchar(50)
);
create unique index idx on t2(col2) comment 'create varchar index';
insert into t2 values(1,"Abby", 24,'zbcvdf');
insert into t2 values(2,"Bob", 25,'zbcvdf');
insert into t2 values(3,"Carol", 23,'zbcvdf');
insert into t2 values(4,"Dora", 29,'zbcvdf');
mysql> select * from t2;
+------+-------+------+--------+
| col1 | col2 | col3 | col4 |
+------+-------+------+--------+
| 1 | Abby | 24 | zbcvdf |
| 2 | Bob | 25 | zbcvdf |
| 3 | Carol | 23 | zbcvdf |
| 4 | Dora | 29 | zbcvdf |
+------+-------+------+--------+
4 rows in set (0.00 sec)
mysql> show create table t2;
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
---------+
| Table | Create Table |
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
---------+
| t2 | CREATE TABLE `t2` (
`col1` BIGINT NOT NULL,
`col2` VARCHAR(25) DEFAULT NULL,
`col3` FLOAT DEFAULT NULL,
`col4` VARCHAR(50) DEFAULT NULL,
PRIMARY KEY (`col1`),
UNIQUE KEY `idx` (`col2`) COMMENT `create varchar index`
) |
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
---------+
1 row in set (0.01 sec)
```

CREATE TABLE

语法说明

`CREATE TABLE` 语句用于在当前所选数据库创建一张新表。

语法结构

```
> CREATE [TEMPORARY] TABLE [IF NOT EXISTS] [db.]table_name [comment =
"comment of table"];
(
name1 type1 [comment 'comment of column'] [AUTO_INCREMENT] [[PRIMARY]
KEY] [[FOREIGN] KEY],
name2 type2 [comment 'comment of column'],
...
)
[cluster by (column_name1, column_name2, ...);]
[partition_options]
```

语法释义

TEMPORARY

在创建表时，可以使用 `TEMPORARY` 关键字创建一个临时表。`TEMPORARY` 表只在当前会话中可见，在会话关闭时自动删除。这表示两个不同的会话可以使用相同的临时表名，而不会彼此冲突或与同名的现有非临时表冲突。(在删除临时表之前，会隐藏现有表。)

删除数据库会自动删除数据库中创建的所有 `TEMPORARY` 表。

创建会话可以对表执行任何操作，例如 `DROP table`、`INSERT`、`UPDATE` 或 `SELECT`。

COMMENT

可以使用 `comment` 选项指定列或整张表的注释：

- `CREATE TABLE [IF NOT EXISTS] [db.]table_name [comment = "comment of table"];` 中的 `comment` 为整张表的注释，最长 2049 个字符。

- `(name1 type1 [comment 'comment of column'],...)` 中的 `comment` 为指定列的注释：最长 1024 个字符。

使用 `SHOW CREATE TABLE` 和 `SHOW FULL COLUMNS` 语句显示注释内容。注释内容也显示在 `INFORMATION_SCHEMA.COLUMN_COMMENT` 列中。

AUTO_INCREMENT

`AUTO_INCREMENT`：表的初始值，初始值从 1 开始，且数据列的值必须唯一。

- 设置 `AUTO_INCREMENT` 的列，需为整数或者浮点数据类型。
- 自增列需要设置为 `NOT NULL`，否则会直接存储 `NULL`。当你将 NULL（推荐）或 0 值插入索引的 `AUTO_INCREMENT` 列时，该列将设置为下一个序列值。通常这是 *value+1*，其中 *value* 是表中当前列的最大值。
- 每个表只能有一个 `AUTO_INCREMENT` 列，它必须可以被索引，且不能设置默认值。`AUTO_INCREMENT` 列需要含有正数值，如果插入一个负数被判断为插入一个非常大的正数，这样做是为了避免数字出现精度问题，并确保不会意外出现包含 0 的 `AUTO_INCREMENT` 列。

PRIMARY KEY

`PRIMARY KEY` 即主键约束，用于唯一标示表中的每条数据。 主键必须包含 `UNIQUE` 值，不能包含 `NULL` 值。 当前版本一个表只能有一个主键，并且这个主键只能有一个列组成。

- **在建表时创建主键**

以下 SQL 语句在创建 `Persons` 表时，在其中的 `ID` 列创建主键：

```
> CREATE TABLE Persons (
ID int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Age int,
PRIMARY KEY (ID)
);
```

注意区分

上述示例中只有一个主键 `PK_Person`，并且其中仅包含了一列（`ID`）

例如使用如下建表语句时会有错误：

```
> CREATE TABLE Students (
ID int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Age int,
PRIMARY KEY (ID,LastName)
);
ERROR 1105 (HY000): tae catalog: schema validation: compound idx not
supported yet
```

FOREIGN KEY

FOREIGN KEY 约束，即外键约束，是表的一个特殊字段，经常与主键约束一起使用。外键约束是用于防止破坏表之间链接的行为。对于两个具有关联关系的表而言，相关联字段中主键所在的表就是主表（父表），外键所在的表就是从表（子表）。外键用来建立主表与从表的关联关系，为两个表的数据建立连接，约束两个表中数据的一致性和完整性。

FOREIGN KEY 约束也能防止非法数据插入外键列，因为它必须是它指向的那个表中的值之一。

定义外键时，需要遵守下列规则：

- 主表必须已经存在于数据库中，或者是当前正在创建的表。如果是后一种情况，则主表与从表是同一个表，这样的表称为自参照表，这种结构称为自参照完整性。
- 必须为主表定义主键。
- 在主表的表名后面指定列名或列名的组合。这个列或列的组合必须是主表的主键或候选键。当前 MatrixOne 仅支持单列外键约束。
- 外键中列的数目必须和主表的主键中列的数目相同。
- 外键中列的数据类型必须和主表主键中对应列的数据类型相同。

下面通过一个例子进行说明通过 FOREIGN KEY 和 PRIMARY KEY 关联父表与子表：

首先创建一个父表，字段 a 为主键：

```
create table t1(a int primary key,b varchar(5));
insert into t1 values(101,'abc'),(102,'def');
mysql> select * from t1;
+------+------+
| a | b |
+------+------+
| 101 | abc |
| 102 | def |
+------+------+
2 rows in set (0.00 sec)
```

然后创建一个子表，字段 c 为外键，关联父表字段 a：

```
create table t2(a int ,b varchar(5),c int, foreign key(c) references
t1(a));
insert into t2 values(1,'zs1',101),(2,'zs2',102);
insert into t2 values(3,'xyz',null);
mysql> select * from t2;
+------+------+------+
| a | b | c |
+------+------+------+
| 1 | zs1 | 101 |
| 2 | zs2 | 102 |
| 3 | xyz | NULL |
+------+------+------+
3 rows in set (0.00 sec)
```

有关数据完整性约束的更多信息，参见[数据完整性约束概述](#)。

Cluster by

`Cluster by` 是一种用于优化表的物理排列方式的命令。在建表时使用 `Cluster by` 命令，对于无主键的表，可以按照指定的列对表进行物理排序，并将数据行重新排列成与该列的值的顺序相同的顺序。使用 `Cluster by` 提高查询性能。

- 单列语法为：`create table() cluster by col;`
- 多列语法为：`create table() cluster by (col1, col2);`

Note: `Cluster by` 不能和主键同时存在，否则会语法报错；`Cluster by` 只能在建表时指定，不支持动态创建。

更多关于使用 `Cluster by` 进行性能调优，参见[使用 Cluster by 语句调优](#).

Table PARTITION 和 PARTITIONS

```
partition_options:
PARTITION BY
\{ [LINEAR] HASH(expr)
| [LINEAR] KEY [ALGORITHM=\{1 | 2}] (column_list)
| RANGE\{(expr) | COLUMNS(column_list)}
| LIST\{(expr) | COLUMNS(column_list)} }
[PARTITIONS num]
[SUBPARTITION BY
\{ [LINEAR] HASH(expr)
| [LINEAR] KEY [ALGORITHM=\{1 | 2}] (column_list) }
]
[(partition_definition [, partition_definition] ...)]

partition_definition:
PARTITION partition_name
[VALUES
\{LESS THAN \{(expr | value_list) | MAXVALUE}
|
IN (value_list)}]
```

分区可以被修改、合并、添加到表中，也可以从表中删除。

· 和单个磁盘或文件系统分区相比，可以存储更多的数据。

· 优化查询。

- where 子句中包含分区条件时，可以只扫描必要的分区。
- 涉及聚合函数的查询时，可以容易的在每个分区上并行处理，最终只需汇总得到结果。

· 对于已经过期或者不需要保存的数据，可以通过删除与这些数据有关的分区来快速删除数据。

· 跨多个磁盘来分散数据查询，以获得更大的查询吞吐量。

- **PARTITION BY**

分区语法以 `PARTITION BY` 开头。该子句包含用于确定分区的函数，这个函数返回一个从 1 到 num 的整数值，其中 num 是分区的数目。

- **HASH(expr)**

在实际工作中经常遇到像会员表的没有明显可以分区的特征字段的大表。为了把这类的数据进行分区打散，MatrixOne 提供了 `HASH` 分区。基于给定的分区个数，将数据分配到不同的分区，`HASH` 分区只能针对整数进行 `HASH`，对于非整形的字段则通过表达式将其转换成整数。

· HASH 分区，基于给定的分区个数，把数据分配到不同的分区。

· Expr 是使用一个或多个表列的表达式。

示例如下：

```
CREATE TABLE t1 (col1 INT, col2 CHAR(5))
PARTITION BY HASH(col1);

CREATE TABLE t1 (col1 INT, col2 CHAR(5), col3 DATETIME)
PARTITION BY HASH ( YEAR(col3) );
```

- **KEY(column_list)**

· KEY 分区，按照某个字段取余。分区对象必须为列，不能是基于列的表达式，且允许多列，KEY 分区列可以不指定，默认为逐渐或者唯一键，不指定的情况下，则必须显性指定列。

类似于 `HASH`。`column_list` 参数只是一个包含 1 个或多个表列的列表（最大值：16）。下面的示例为一个按 `KEY` 分区的简单表，有 4 个分区：

```
CREATE TABLE tk (col1 INT, col2 CHAR(5), col3 DATE)
PARTITION BY KEY(col3)
PARTITIONS 4;
```

对于按 `KEY` 分区的表，可以使用 `LINEAR KEY` 来进行线性分区。这与使用 `HASH` 分区的表具有相同的效果。下面的示例为使用 `LINEAR KEY` 线性分区在 5 个分区之间分配数据：

```
CREATE TABLE tk (col1 INT, col2 CHAR(5), col3 DATE)
PARTITION BY LINEAR KEY(col3)
PARTITIONS 5;
```

- **RANGE(expr)**

`RANGE` 分区：基于一个给定连续区间范围，把数据（或者可以说是多行）分配到不同的分区。最常见的是基于时间字段。基于分区的列最好是整型，如果日期型的可以使用函数转换为整型。

在这种情况下，expr 使用一组 `VALUES LESS THAN` 运算符显示一系列值。使用范围分区时，你必须使用 `VALUES LESS THAN` 定义至少一个分区，且不能将 `VALUES IN` 与范围分区一起使用。

`VALUES LESS THAN MAXVALUE` 用于指定小于指定最大值的 “剩余” 值。

子句排列方式为：每个连续的 `VALUES LESS THAN` 中指定的上限大于前一个的上限，引用 `MAXVALUE` 的那个在列表中排在最后。

- **RANGE COLUMNS(column_list)**

`RANGE COLUMNS(column_list)` 为 `RANGE` 的另一种形式，常用作为使用多个列上的范围条件（即，诸如 `WHERE a = 1 AND b \< 10 或 WHERE a = 1 AND b = 10 AND c \< 10`）之类的条件对查询进行分区修剪。它使你能够通过使用 `COLUMNS` 子句中的列列表和每个 `PARTITION ... VALUES LESS THAN (value_list)` 分区定义子句中的一组列值来指定多个列中的值范围。（在最简单的情况下，该集合由单个列组成。）`column_list` 和 `value_list` 中可以引用的最大列数为 16。

`column_list` 使用：

1. `column_list` 可以只包含列名。
2. 列表中的每一列必须是整数类型、字符串类型、时间或日期列类型。
3. 不允许使用 `BLOB`、`TEXT`、`SET`、`ENUM`、`BIT` 或空间数据类型的列；也不允许使用浮点数类型的列。也不可以在 `COLUMNS` 子句中使用函数或算术表达式。

分区定义说明：

1. 分区定义中，用于每个 `VALUES LESS THAN` 子句的值列表必须包含与 `COLUMNS` 子句中列出的列相同数量的值。

2. `NULL` 不能出现在 `VALUES LESS THAN` 中的任何值。可以对除第一列以外的给定列多次使用 `MAXVALUE`，如下例所示：

```
CREATE TABLE rc (
a INT NOT NULL,
b INT NOT NULL
)
PARTITION BY RANGE COLUMNS(a,b) (
PARTITION p0 VALUES LESS THAN (10,5),
PARTITION p1 VALUES LESS THAN (20,10),
PARTITION p2 VALUES LESS THAN (50,MAXVALUE),
PARTITION p3 VALUES LESS THAN (65,MAXVALUE),
PARTITION p4 VALUES LESS THAN (MAXVALUE,MAXVALUE)
);
```

- **LIST(expr)**

`LIST` 分区和 `RANGE` 分区类似，区别在于 `LIST` 是枚举值列表的集合，`RANGE` 是连续的区间值的集合。二者在语法方面非常的相似。

分区使用：

1. `LIST` 分区列是非 null 列，否则插入 null 值如果枚举列表里面不存在 null 值会插入失败，这点和其它的分区不一样，`RANGE` 分区会将其作为最小分区值存储，`HASH` 或 `KEY` 分为会将其转换成 0 存储，因为 `LIST` 分区只支持整型，非整型字段需要通过函数转换成整形。
2. 使用 `LIST` 分区时，你必须使用 `VALUES IN` 定义至少一个分区，且不能将 `VALUES LESS THAN` 与 `PARTITION BY LIST` 一起使用。

示例如下：

```
CREATE TABLE client_firms (
id INT,
name VARCHAR(35)
)
PARTITION BY LIST (id) (
PARTITION r0 VALUES IN (1, 5, 9, 13, 17, 21),
PARTITION r1 VALUES IN (2, 6, 10, 14, 18, 22),
PARTITION r2 VALUES IN (3, 7, 11, 15, 19, 23),
PARTITION r3 VALUES IN (4, 8, 12, 16, 20, 24)
);
```

- **LIST COLUMNS(column_list)**

`LIST COLUMNS(column_list)` 是 `LIST` 的另一种书写形式，用于多列上的比较条件（即，具有诸如 `WHERE a = 5 AND b = 5 或 WHERE a = 1 AND b = 10 AND c = 5` 之类的条件）对查询进行分区修剪。通过使用 `COLUMNS` 子句中的列列表和每个 `PARTITION ... VALUES IN` (value_list) 分区定义子句中的一组列值来指定多个列中的值。

`LIST COLUMNS(column_list)` 中使用的列列表和 `VALUES IN(value_list)` 中使用的值列表的数据类型规则与 `RANGE COLUMNS(column_list)` 中使用的列列表的规则 `VALUES LESS THAN(value_list)` 中使用的值列表规则相同，但在 `VALUES IN` 子句中，不允许使用 `MAXVALUE`，可以使用 `NULL`。

与 `PARTITION BY LIST COLUMNS` 一起使用的 `VALUES IN` 值列表与与 `PARTITION BY LIST` 一起使用时的值列表有一个重要区别。当与 `PARTITION BY LIST COLUMNS` 一起使用时，`VALUES IN` 子句中的每个元素都必须是一组列值；每个集合中的值的数量必须与 `COLUMNS` 子句中使用的列数相同，并且这些值的数据类型必须与列的数据类型匹配（并且以相同的顺序出现）。在最简单的情况下，该集合由一列组成。在 `column_list` 和组成 `value_list` 的元素中可以使用的最大列数是 16。

示例如下：

```
CREATE TABLE lc (
a INT NULL,
b INT NULL
)
PARTITION BY LIST COLUMNS(a,b) (
PARTITION p0 VALUES IN( (0,0), (NULL,NULL) ),
PARTITION p1 VALUES IN( (0,1), (0,2), (0,3), (1,1), (1,2) ),
PARTITION p2 VALUES IN( (1,0), (2,0), (2,1), (3,0), (3,1) ),
PARTITION p3 VALUES IN( (1,3), (2,2), (2,3), (3,2), (3,3) )
);
```

- **PARTITIONS num**

可以选择使用 `PARTITIONS num` 子句指定分区数，其中 `num` 是分区数。如果使用此子句的同时，也使用了其他 `PARTITION` 子句，那么 `num` 必须等于使用 `PARTITION` 子句声明的分区的总数。

示例

- 示例 1：创建普通表

```
CREATE TABLE test(a int, b varchar(10));
INSERT INTO test values(123, 'abc');

mysql> SELECT * FROM test;
+------+---------+
| a | b |
+------+---------+
| 123 | abc |
+------+---------+
```

- 示例 2：创建表示增加注释

```
create table t2 (a int, b int) comment = "事实表";

mysql> show create table t2;
+-------+-----------------------------------------------------------------
-----------------------+
| Table | Create Table |
+-------+-----------------------------------------------------------------
-----------------------+
| t2 | CREATE TABLE `t2` (
`a` INT DEFAULT NULL,
`b` INT DEFAULT NULL
) COMMENT='事实表', |
+-------+-----------------------------------------------------------------
-----------------------+
```

- 示例 3：建表时为列增加注释

```
create table t3 (a int comment '列的注释', b int) comment = "table";

mysql> SHOW CREATE TABLE t3;
+-------+-----------------------------------------------------------------
----------------------------------------+
| Table | Create Table |
+-------+-----------------------------------------------------------------
----------------------------------------+
| t3 | CREATE TABLE `t3` (
`a` INT DEFAULT NULL COMMENT '列的注释',
`b` INT DEFAULT NULL
) COMMENT='table', |
+-------+-----------------------------------------------------------------
----------------------------------------+
```

- 示例 4：创建普通分区表

```
CREATE TABLE tp1 (col1 INT, col2 CHAR(5), col3 DATE) PARTITION BY
KEY(col3) PARTITIONS 4;

mysql> SHOW CREATE TABLE tp1;
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------+
| Table | Create Table |
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------+
| tp1 | CREATE TABLE `tp1` (
`col1` INT DEFAULT NULL,
`col2` CHAR(5) DEFAULT NULL,
`col3` DATE DEFAULT NULL
) partition by key algorithm = 2 (col3) partitions 4 |
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------+
1 row in set (0.00 sec)

-- 不指定分区数
CREATE TABLE tp2 (col1 INT, col2 CHAR(5), col3 DATE) PARTITION BY
KEY(col3);

mysql> SHOW CREATE TABLE tp2;
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
----+
| Table | Create Table |
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
----+
| tp2 | CREATE TABLE `tp2` (
`col1` INT DEFAULT NULL,
`col2` CHAR(5) DEFAULT NULL,
`col3` DATE DEFAULT NULL
) partition by key algorithm = 2 (col3) |
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
----+
1 row in set (0.00 sec)

-- 指定分区算法
```

```
CREATE TABLE tp3
(
col1 INT,
col2 CHAR(5),
col3 DATE
) PARTITION BY KEY ALGORITHM = 1 (col3);


mysql> show create table tp3;
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
----+
| Table | Create Table |
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
----+
| tp3 | CREATE TABLE `tp3` (
`col1` INT DEFAULT NULL,
`col2` CHAR(5) DEFAULT NULL,
`col3` DATE DEFAULT NULL
) partition by key algorithm = 1 (col3) |
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
----+
1 row in set (0.00 sec)

-- 指定分区算法及分区数
CREATE TABLE tp4 (col1 INT, col2 CHAR(5), col3 DATE) PARTITION BY LINEAR
KEY ALGORITHM = 1 (col3) PARTITIONS 5;

mysql> SHOW CREATE TABLE tp4;
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
------------------------+
| Table | Create Table |
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
------------------------+
| tp4 | CREATE TABLE `tp4` (
`col1` INT DEFAULT NULL,
`col2` CHAR(5) DEFAULT NULL,
`col3` DATE DEFAULT NULL
) partition by linear key algorithm = 1 (col3) partitions 5 |
+-------+-------------------------------------------------------------------
```

```
-----------------------------------------------------------------------------
------------------------+
1 row in set (0.01 sec)

-- 多列分区
CREATE TABLE tp5
(
col1 INT,
col2 CHAR(5),
col3 DATE
) PARTITION BY KEY(col1, col2) PARTITIONS 4;

mysql> SHOW CREATE TABLE tp5;
+-------+-----------------------------------------------------------------
-----------------------------------------------------------------------
------------------------+
| Table | Create Table |
+-------+-----------------------------------------------------------------
-----------------------------------------------------------------------
------------------------+
| tp5 | CREATE TABLE `tp5` (
`col1` INT DEFAULT NULL,
`col2` CHAR(5) DEFAULT NULL,
`col3` DATE DEFAULT NULL
) partition by key algorithm = 2 (col1, col2) partitions 4 |
+-------+-----------------------------------------------------------------
-----------------------------------------------------------------------
------------------------+
1 row in set (0.01 sec)

-- 创建主键列分区
CREATE TABLE tp6
(
col1 INT NOT NULL PRIMARY KEY,
col2 DATE NOT NULL,
col3 INT NOT NULL,
col4 INT NOT NULL
) PARTITION BY KEY(col1) PARTITIONS 4;

mysql> SHOW CREATE TABLE tp6;
+-------+-----------------------------------------------------------------
-----------------------------------------------------------------------
-------------------------------------------+
| Table | Create Table |
```

```
+-------+-------------------------------------------------------------------------------------------------------------------------------------------+
| tp6 | CREATE TABLE `tp6` (
`col1` INT NOT NULL,
`col2` DATE NOT NULL,
`col3` INT NOT NULL,
`col4` INT NOT NULL,
PRIMARY KEY (`col1`)
) partition by key algorithm = 2 (col1) partitions 4 |
+-------+-------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.01 sec)

-- 创建HASH分区
CREATE TABLE tp7
(
col1 INT,
col2 CHAR(5)
) PARTITION BY HASH(col1);

mysql> SHOW CREATE TABLE tp7;
+-------+-------------------------------------------------------------------------------------------------------------------------------------------+
| Table | Create Table |
+-------+-------------------------------------------------------------------------------------------------------------------------------------------+
| tp7 | CREATE TABLE `tp7` (
`col1` INT DEFAULT NULL,
`col2` CHAR(5) DEFAULT NULL
) partition by hash (col1) |
+-------+-------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.01 sec)

-- 创建 HASH 分区时指定分区数
CREATE TABLE tp8
(
col1 INT,
col2 CHAR(5)
) PARTITION BY HASH(col1) PARTITIONS 4;

mysql> SHOW CREATE TABLE tp8;
```

```
+-------+-------------------------------------------------------------------
-------------------------------------------------+
| Table | Create Table |
+-------+-------------------------------------------------------------------
-------------------------------------------------+
| tp8 | CREATE TABLE `tp8` (
`col1` INT DEFAULT NULL,
`col2` CHAR(5) DEFAULT NULL
) partition by hash (col1) partitions 4 |
+-------+-------------------------------------------------------------------
-------------------------------------------------+
1 row in set (0.00 sec)

-- 创建分区时，指定分区粒度
CREATE TABLE tp9
(
col1 INT,
col2 CHAR(5),
col3 DATETIME
) PARTITION BY HASH (YEAR(col3));

mysql> SHOW CREATE TABLE tp9;
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
-+
| Table | Create Table |
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
-+
| tp9 | CREATE TABLE `tp9` (
`col1` INT DEFAULT NULL,
`col2` CHAR(5) DEFAULT NULL,
`col3` DATETIME DEFAULT NULL
) partition by hash (year(col3)) |
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
-+
1 row in set (0.00 sec)

-- 创建分区时，指定分区粒度和分区数量
CREATE TABLE tp10
(
col1 INT,
col2 CHAR(5),
```

```
col3 DATE
) PARTITION BY LINEAR HASH( YEAR(col3)) PARTITIONS 6;

mysql> SHOW CREATE TABLE tp10;
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
------------------+
| Table | Create Table |
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
------------------+
| tp10 | CREATE TABLE `tp10` (
`col1` INT DEFAULT NULL,
`col2` CHAR(5) DEFAULT NULL,
`col3` DATE DEFAULT NULL
) partition by linear hash (year(col3)) partitions 6 |
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
------------------+
1 row in set (0.00 sec)

-- 创建分区时，使用主键列作为 HASH 分区
CREATE TABLE tp12 (col1 INT NOT NULL PRIMARY KEY, col2 DATE NOT NULL,
col3 INT NOT NULL, col4 INT NOT NULL) PARTITION BY HASH(col1) PARTITIONS
4;

mysql> SHOW CREATE TABLE tp12;
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
--------------------------------+
| Table | Create Table |
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
--------------------------------+
| tp12 | CREATE TABLE `tp12` (
`col1` INT NOT NULL,
`col2` DATE NOT NULL,
`col3` INT NOT NULL,
`col4` INT NOT NULL,
PRIMARY KEY (`col1`)
) partition by hash (col1) partitions 4 |
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
--------------------------------+
```

```
1 row in set (0.01 sec)

-- 创建 RANGE 分区，并划分分区范围
CREATE TABLE tp13 (id INT NOT NULL PRIMARY KEY, fname VARCHAR(30), lname
VARCHAR(30), hired DATE NOT NULL DEFAULT '1970-01-01', separated DATE NOT
NULL DEFAULT '9999-12-31', job_code INT NOT NULL, store_id INT NOT NULL)
PARTITION BY RANGE (id) (PARTITION p0 VALUES LESS THAN (6), PARTITION p1
VALUES LESS THAN (11), PARTITION p2 VALUES LESS THAN (16), PARTITION p3
VALUES LESS THAN (21));

mysql> SHOW CREATE TABLE tp13;
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------+
| Table | Create Table |
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------+
| tp13 | CREATE TABLE `tp13` (
`id` INT NOT NULL,
`fname` VARCHAR(30) DEFAULT NULL,
`lname` VARCHAR(30) DEFAULT NULL,
`hired` DATE DEFAULT '1970-01-01',
`separated` DATE DEFAULT '9999-12-31',
`job_code` INT NOT NULL,
`store_id` INT NOT NULL,
PRIMARY KEY (`id`)
) partition by range(id) (partition p0 values less than (6), partition p1
values less than (11), partition p2 values less than (16), partition p3
values less than (21)) |
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------+
1 row in set (0.01 sec)
```

```
CREATE TABLE tp14 (id INT NOT NULL, fname VARCHAR(30), lname VARCHAR(30),
hired DATE NOT NULL DEFAULT '1970-01-01', separated DATE NOT NULL DEFAULT
'9999-12-31', job_code INT, store_id INT) PARTITION BY RANGE (
YEAR(separated) ) ( PARTITION p0 VALUES LESS THAN (1991), PARTITION p1
VALUES LESS THAN (1996), PARTITION p2 VALUES LESS THAN (2001), PARTITION
p3 VALUES LESS THAN MAXVALUE);

mysql> SHOW CREATE TABLE tp14;
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-------+
| Table | Create Table |
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-------+
| tp14 | CREATE TABLE `tp14` (
`id` INT NOT NULL,
`fname` VARCHAR(30) DEFAULT NULL,
`lname` VARCHAR(30) DEFAULT NULL,
`hired` DATE DEFAULT '1970-01-01',
`separated` DATE DEFAULT '9999-12-31',
`job_code` INT DEFAULT NULL,
`store_id` INT DEFAULT NULL
) partition by range(year(separated)) (partition p0 values less than
(1991), partition p1 values less than (1996), partition p2 values less
than (2001), partition p3 values less than (MAXVALUE)) |
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-------+
1 row in set (0.00 sec)

-- 使用多列作为 RANGE 分区，并指定分区范围
```

```
CREATE TABLE tp15 (a INT NOT NULL, b INT NOT NULL) PARTITION BY RANGE
COLUMNS(a,b) PARTITIONS 4 (PARTITION p0 VALUES LESS THAN (10,5),
PARTITION p1 VALUES LESS THAN (20,10), PARTITION p2 VALUES LESS THAN
(50,20), PARTITION p3 VALUES LESS THAN (65,30));

mysql> SHOW CREATE TABLE tp15;
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------+
| Table | Create Table |
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------+
| tp15 | CREATE TABLE `tp15` (
`a` INT NOT NULL,
`b` INT NOT NULL
) partition by range columns (a, b) partitions 4 (partition p0 values
less than (10, 5), partition p1 values less than (20, 10), partition p2
values less than (50, 20), partition p3 values less than (65, 30)) |
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------+
1 row in set (0.00 sec)

-- 创建 LIST 分区
CREATE TABLE tp16 (id INT PRIMARY KEY, name VARCHAR(35), age INT
unsigned) PARTITION BY LIST (id) (PARTITION r0 VALUES IN (1, 5, 9, 13,
17, 21), PARTITION r1 VALUES IN (2, 6, 10, 14, 18, 22), PARTITION r2
VALUES IN (3, 7, 11, 15, 19, 23), PARTITION r3 VALUES IN (4, 8, 12, 16,
20, 24));

mysql> SHOW CREATE TABLE tp16;
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------+
| Table | Create Table |
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
```

```
-----------------------------------------------------------------------------
-------------------------------------------------------------+
| tp16 | CREATE TABLE `tp16` (
`id` INT DEFAULT NULL,
`name` VARCHAR(35) DEFAULT NULL,
`age` INT UNSIGNED DEFAULT NULL,
PRIMARY KEY (`id`)
) partition by list(id) (partition r0 values in (1, 5, 9, 13, 17, 21),
partition r1 values in (2, 6, 10, 14, 18, 22), partition r2 values in (3,
7, 11, 15, 19, 23), partition r3 values in (4, 8, 12, 16, 20, 24)) |
+-------+----------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-------------------------------------------------------------+
1 row in set (0.01 sec)

CREATE TABLE tp17 (id INT, name VARCHAR(35), age INT unsigned) PARTITION
BY LIST (id) (PARTITION r0 VALUES IN (1, 5, 9, 13, 17, 21), PARTITION r1
VALUES IN (2, 6, 10, 14, 18, 22), PARTITION r2 VALUES IN (3, 7, 11, 15,
19, 23), PARTITION r3 VALUES IN (4, 8, 12, 16, 20, 24));

mysql> SHOW CREATE TABLE tp17;
+-------+----------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
---------------------------------------+
| Table | Create Table |
+-------+----------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
---------------------------------------+
| tp17 | CREATE TABLE `tp17` (
`id` INT DEFAULT NULL,
`name` VARCHAR(35) DEFAULT NULL,
`age` INT UNSIGNED DEFAULT NULL
) partition by list(id) (partition r0 values in (1, 5, 9, 13, 17, 21),
partition r1 values in (2, 6, 10, 14, 18, 22), partition r2 values in (3,
7, 11, 15, 19, 23), partition r3 values in (4, 8, 12, 16, 20, 24)) |
+-------+----------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
```

```
-----------------------------------------------------------------------------
---------------------------------------+
1 row in set (0.01 sec)

-- 使用多列作为 LIST 分区
CREATE TABLE tp18 (a INT NULL,b INT NULL) PARTITION BY LIST COLUMNS(a,b)
(PARTITION p0 VALUES IN( (0,0), (NULL,NULL) ), PARTITION p1 VALUES IN(
(0,1), (0,2), (0,3), (1,1), (1,2) ), PARTITION p2 VALUES IN( (1,0),
(2,0), (2,1), (3,0), (3,1) ), PARTITION p3 VALUES IN( (1,3), (2,2),
(2,3), (3,2), (3,3) ));

mysql> SHOW CREATE TABLE tp18;
+-------+-----------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-------------------------------------------------------------+
| Table | Create Table |
+-------+-----------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-------------------------------------------------------------+
| tp18 | CREATE TABLE `tp18` (
`a` INT DEFAULT NULL,
`b` INT DEFAULT NULL
) partition by list columns (a, b) (partition p0 values in ((0, 0),
(null, null)), partition p1 values in ((0, 1), (0, 2), (0, 3), (1, 1),
(1, 2)), partition p2 values in ((1, 0), (2, 0), (2, 1), (3, 0), (3, 1)),
partition p3 values in ((1, 3), (2, 2), (2, 3), (3, 2), (3, 3))) |
+-------+-----------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-------------------------------------------------------------+
1 row in set (0.00 sec)
```

- 示例 5：主键自增

```
drop table if exists t1;
create table t1(a bigint primary key auto_increment, b varchar(10));
insert into t1(b) values ('bbb');
insert into t1 values (3, 'ccc');
insert into t1(b) values ('bbb1111');

mysql> select * from t1 order by a;
+------+---------+
| a | b |
+------+---------+
| 1 | bbb |
| 3 | ccc |
| 4 | bbb1111 |
+------+---------+
3 rows in set (0.01 sec)

insert into t1 values (2, 'aaaa1111');

mysql> select * from t1 order by a;
+------+----------+
| a | b |
+------+----------+
| 1 | bbb |
| 2 | aaaa1111 |
| 3 | ccc |
| 4 | bbb1111 |
+------+----------+
4 rows in set (0.00 sec)

insert into t1(b) values ('aaaa1111');

mysql> select * from t1 order by a;
+------+----------+
| a | b |
+------+----------+
| 1 | bbb |
| 2 | aaaa1111 |
| 3 | ccc |
| 4 | bbb1111 |
| 5 | aaaa1111 |
+------+----------+
5 rows in set (0.01 sec)

insert into t1 values (100, 'xxxx');
```

```
insert into t1(b) values ('xxxx');

mysql> select * from t1 order by a;
+------+----------+
| a | b |
+------+----------+
| 1 | bbb |
| 2 | aaaa1111 |
| 3 | ccc |
| 4 | bbb1111 |
| 5 | aaaa1111 |
| 100 | xxxx |
| 101 | xxxx |
+------+----------+
7 rows in set (0.00 sec)
```

限制

目前不支持带有 `ALTER TABLE` 的 `DROP PRIMARY KEY` 语句。

CREATE EXTERNAL TABLE

语法说明

外部表是指不在数据库里的表，是操作系统上的一个按照一定格式分割的文本文件，或是其他类型的表，对 MatrixOne 来说类似于视图，可以在数据库中像视图一样进行查询等操作，但是外部表在数据库中只有表结构，而数据存放在操作系统中。

本篇文档将讲述如何在 MatrixOne 数据库外建表。

语法结构

通用语法

```
> CREATE EXTERNAL TABLE [IF NOT EXISTS] [db.]table_name;
(
name1 type1,
name2 type2,
...
)
```

语法示例

```
## 创建指向本地文件的外表（指定压缩格式）
create external table t(...) localfile{"filepath"='<string>',
"compression"='<string>'} FIELDS TERMINATED BY ',' ENCLOSED BY '\"' LINES
TERMINATED BY '\n';

## 创建指向本地文件的外表（不指定压缩格式，则为auto格式，自动检查文件的格式）
create external table t(...) localfile{"filepath"='<string>'} FIELDS
TERMINATED BY ',' ENCLOSED BY '\"' LINES TERMINATED BY '\n';


## 创建指向S3文件的外表（指定压缩格式）
create external table t(...) URL s3option{"endpoint"='<string>',
"access_key_id"='<string>', "secret_access_key"='<string>',
"bucket"='<string>', "filepath"='<string>', "region"='<string>',
"compression"='<string>'} FIELDS TERMINATED BY ',' ENCLOSED BY '\"' LINES
TERMINATED BY '\n';

## 创建指向S3文件的外表（不指定压缩格式，则为auto格式，自动检查文件的格式）
create external table t(...) URL s3option{"endpoint"='<string>',
"access_key_id"='<string>', "secret_access_key"='<string>',
"bucket"='<string>', "filepath"='<string>', "region"='<string>'} FIELDS
TERMINATED BY ',' ENCLOSED BY '\"' LINES TERMINATED BY '\n';
```

语法说明

参数说明

参数	描述
endpoint	终端节点是作为 AWS Web 服务的入口点的 URL。例如：s3.us-west-2.amazonaws.com
access_key_id	S3 的 Access key ID
secret_access_key	S3 的 Secret access key
bucket	需要访问的桶
filepath	访问文件的相对路径
region	s3 所在的区域
compression	S3 文件的压缩格式，为空表示非压缩文件，支持的字段或压缩格式为"auto", "none", "gzip", "bzip2", "flate", "zlib", "lz4"

参数	描述
auto	压缩格式，表示通过文件后缀名自动检查文件的压缩格式
none	压缩格式，表示为非压缩格式，其余表示文件的压缩格式

示例

```
create external table ex_table_cpk(clo1 tinyint,clo2 smallint,clo3
int,clo4 bigint,clo5 tinyint unsigned,clo6 smallint unsigned,clo7 int
unsigned,clo8 bigint unsigned,col9 float,col10 double,col11
varchar(255),col12 Date,col13 DateTime,col14 timestamp,col15 bool,col16
decimal(5,2),col17 text,col18 varchar(255),col19 varchar(255),col20
varchar(255))infile{"filepath"='$resources/external_table_file/cpk_table_1.c
;
```

更多关于使用外表指定 S3 文件，参见[从 S3 对象存储服务读取数据并导入 MatrixOne]。

限制

当前 MatrixOne 仅支持对外部表进行 `select` 操作，暂时还不支持使用 `delete`、`insert`、`update` 对外部表插入数据。

CREATE VIEW

语法说明

视图是基于 SQL 语句的结果集的可视化的表。

视图包含行和列，就像一个真实的表。视图中的字段就是来自一个或多个数据库中的真实的表中的字段。

您可以向视图添加 SQL 函数，`WHERE` 或者 `JOIN` 语句，也同样可以呈现数据，类似于这些数据来自于某个单一的表一样。

`CREATE VIEW` 语句用于创建一个视图。

语法结构

```
> CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

> **注意**
>
> 视图总是显示最新的数据。每当你查询视图时，数据库引擎通过使用视图的 SQL 语句重建数据。

示例

- 示例 1：

```
CREATE TABLE t00(a INTEGER);
INSERT INTO t00 VALUES (1),(2);
CREATE TABLE t01(a INTEGER);
INSERT INTO t01 VALUES (1);
CREATE VIEW v0 AS SELECT t00.a, t01.a AS b FROM t00 LEFT JOIN t01
USING(a);

mysql> SELECT t00.a, t01.a AS b FROM t00 LEFT JOIN t01 USING(a);
+------+------+
| a | b |
+------+------+
| 1 | 1 |
| 2 | NULL |
+------+------+
2 rows in set (0.01 sec)

mysql> SELECT * FROM v0 WHERE b >= 0;
+------+------+
| a | b |
+------+------+
| 1 | 1 |
+------+------+
1 row in set (0.01 sec)

mysql> SHOW CREATE VIEW v0;
+------+-----------------------------------------------------------------
-----------+
| View | Create View |
+------+-----------------------------------------------------------------
-----------+
| v0 | CREATE VIEW v0 AS SELECT t00.a, t01.a AS b FROM t00 LEFT JOIN t01
USING(a) |
+------+-----------------------------------------------------------------
-----------+
1 row in set (0.00 sec)
```

- 示例 2：

```
drop table if exists t1;
create table t1 (id int,ti tinyint unsigned,si smallint,bi bigint
unsigned,fl float,dl double,de decimal,ch char(20),vch varchar(20),dd
date,dt datetime);
insert into t1
values(1,1,4,3,1113.32,111332,1113.32,'hello','subquery','2022-04-
28','2022-04-28 22:40:11');
insert into t1 values(2,2,5,2,2252.05,225205,2252.05,'bye','sub
query','2022-04-28','2022-04-28 22:40:11');
insert into t1
values(3,6,6,3,3663.21,366321,3663.21,'hi','subquery','2022-04-28','2022-
04-28 22:40:11');
insert into t1 values(4,7,1,5,4715.22,471522,4715.22,'good morning','my
subquery','2022-04-28','2022-04-28 22:40:11');
insert into t1 values(5,1,2,6,51.26,5126,51.26,'byebye',' is
subquery?','2022-04-28','2022-04-28 22:40:11');
insert into t1 values(6,3,2,1,632.1,6321,632.11,'good night','maybe
subquery','2022-04-28','2022-04-28 22:40:11');
insert into t1
values(7,4,4,3,7443.11,744311,7443.11,'yes','subquery','2022-04-
28','2022-04-28 22:40:11');
insert into t1 values(8,7,5,8,8758.00,875800,8758.11,'nice to meet','just
subquery','2022-04-28','2022-04-28 22:40:11');
insert into t1 values(9,8,4,9,9849.312,9849312,9849.312,'see
you','subquery','2022-04-28','2022-04-28 22:40:11');
drop table if exists t2;
create table t2 (id int,ti tinyint unsigned,si smallint,bi bigint
unsigned,fl float,dl double,de decimal,ch char(20),vch varchar(20),dd
date,dt datetime);
insert into t2
values(1,1,4,3,1113.32,111332,1113.32,'hello','subquery','2022-04-
28','2022-04-28 22:40:11');
insert into t2 values(2,2,5,2,2252.05,225205,2252.05,'bye','sub
query','2022-04-28','2022-04-28 22:40:11');
insert into t2
values(3,6,6,3,3663.21,366321,3663.21,'hi','subquery','2022-04-28','2022-
04-28 22:40:11');
insert into t2 values(4,7,1,5,4715.22,471522,4715.22,'good morning','my
subquery','2022-04-28','2022-04-28 22:40:11');
insert into t2 values(5,1,2,6,51.26,5126,51.26,'byebye',' is
subquery?','2022-04-28','2022-04-28 22:40:11');
insert into t2 values(6,3,2,1,632.1,6321,632.11,'good night','maybe
subquery','2022-04-28','2022-04-28 22:40:11');
insert into t2
```

```
values(7,4,4,3,7443.11,744311,7443.11,'yes','subquery','2022-04-
28','2022-04-28 22:40:11');
insert into t2 values(8,7,5,8,8758.00,875800,8758.11,'nice to meet','just
subquery','2022-04-28','2022-04-28 22:40:11');
insert into t2 values(9,8,4,9,9849.312,9849312,9849.312,'see
you','subquery','2022-04-28','2022-04-28 22:40:11');

mysql> select * from (select * from t1) sub where id > 4;
+------+------+------+------+----------+---------+------+--------------+-
---------------+------------+---------------------+
| id | ti | si | bi | fl | dl | de | ch | vch | dd | dt |
+------+------+------+------+----------+---------+------+--------------+-
---------------+------------+---------------------+
| 5 | 1 | 2 | 6 | 51.26 | 5126 | 51 | byebye | is subquery? | 2022-04-28
| 2022-04-28 22:40:11 |
| 6 | 3 | 2 | 1 | 632.1 | 6321 | 632 | good night | maybe subquery |
2022-04-28 | 2022-04-28 22:40:11 |
| 7 | 4 | 4 | 3 | 7443.11 | 744311 | 7443 | yes | subquery | 2022-04-28 |
2022-04-28 22:40:11 |
| 8 | 7 | 5 | 8 | 8758 | 875800 | 8758 | nice to meet | just subquery |
2022-04-28 | 2022-04-28 22:40:11 |
| 9 | 8 | 4 | 9 | 9849.312 | 9849312 | 9849 | see you | subquery | 2022-
04-28 | 2022-04-28 22:40:11 |
+------+------+------+------+----------+---------+------+--------------+-
---------------+------------+---------------------+
5 rows in set (0.01 sec)

create view v1 as select * from (select * from t1) sub where id > 4;
create view v2 as select ti as t,fl as f from (select * from t1) sub
where dl \<> 4;
create view v3 as select * from (select ti as t,fl as f from t1 where dl
\<> 4) sub;
create view v4 as select id,min(ti) from (select * from t1) sub group by
id;
create view v5 as select * from (select id,min(ti) from (select * from
t1) t1 group by id) sub;

mysql> select * from v1;
+------+------+------+------+----------+---------+------+--------------+-
---------------+------------+---------------------+
| id | ti | si | bi | fl | dl | de | ch | vch | dd | dt |
+------+------+------+------+----------+---------+------+--------------+-
---------------+------------+---------------------+
| 5 | 1 | 2 | 6 | 51.26 | 5126 | 51 | byebye | is subquery? | 2022-04-28
```

```
| 2022-04-28 22:40:11 |
| 6 | 3 | 2 | 1 | 632.1 | 6321 | 632 | good night | maybe subquery |
2022-04-28 | 2022-04-28 22:40:11 |
| 7 | 4 | 4 | 3 | 7443.11 | 744311 | 7443 | yes | subquery | 2022-04-28 |
2022-04-28 22:40:11 |
| 8 | 7 | 5 | 8 | 8758 | 875800 | 8758 | nice to meet | just subquery |
2022-04-28 | 2022-04-28 22:40:11 |
| 9 | 8 | 4 | 9 | 9849.312 | 9849312 | 9849 | see you | subquery | 2022-
04-28 | 2022-04-28 22:40:11 |
+------+------+------+------+----------+---------+------+--------------+-
--------------+------------+--------------------+
5 rows in set (0.00 sec)

mysql> select * from v2;
+------+----------+
| t | f |
+------+----------+
| 1 | 1113.32 |
| 2 | 2252.05 |
| 6 | 3663.21 |
| 7 | 4715.22 |
| 1 | 51.26 |
| 3 | 632.1 |
| 4 | 7443.11 |
| 7 | 8758 |
| 8 | 9849.312 |
+------+----------+
9 rows in set (0.00 sec)

mysql> select * from v3;
+------+----------+
| t | f |
+------+----------+
| 1 | 1113.32 |
| 2 | 2252.05 |
| 6 | 3663.21 |
| 7 | 4715.22 |
| 1 | 51.26 |
| 3 | 632.1 |
| 4 | 7443.11 |
| 7 | 8758 |
| 8 | 9849.312 |
+------+----------+
9 rows in set (0.00 sec)
```

```
mysql> select * from v4;
+------+---------+
| id | min(ti) |
+------+---------+
| 1 | 1 |
| 2 | 2 |
| 3 | 6 |
| 4 | 7 |
| 5 | 1 |
| 6 | 3 |
| 7 | 4 |
| 8 | 7 |
| 9 | 8 |
+------+---------+
9 rows in set (0.00 sec)

mysql> select * from v5;
+------+---------+
| id | min(ti) |
+------+---------+
| 1 | 1 |
| 2 | 2 |
| 3 | 6 |
| 4 | 7 |
| 5 | 1 |
| 6 | 3 |
| 7 | 4 |
| 8 | 7 |
| 9 | 8 |
+------+---------+
9 rows in set (0.01 sec)
```

ALTER VIEW

语法说明

`ALTER VIEW` 用于更改已存在的视图。

如果语法参数列表中命名的任何视图都不存在，语句报错，无法更改哪些不存在的视图。

语法结构

```
> ALTER VIEW view_name [(column_list)]
AS select_statement
[WITH [CASCADED | LOCAL] CHECK OPTION]
```

示例

```
drop table if exists t1;
create table t1 (a int);
insert into t1 values(1),(2),(3),(4);
create view v5 as select * from t1;

mysql> select * from v5;
+------+
| a |
+------+
| 1 |
| 2 |
| 3 |
| 4 |
+------+
4 rows in set (0.01 sec)

alter view v5 as select * from t1 where a=1;

mysql> select * from v5;
+------+
| a |
+------+
| 1 |
+------+
1 row in set (0.01 sec)

alter view v5 as select * from t1 where a > 2;

mysql> select * from v5;
+------+
| a |
+------+
| 3 |
| 4 |
+------+
2 rows in set (0.00 sec)
```

DROP DATABASE

语法说明

该语句用于删除一个数据库。

语法结构

```
> DROP DATABASE [IF EXISTS] <database_name>
```

示例

```
CREATE DATABASE test01;

mysql> DROP DATABASE test01;
Query OK, 0 rows affected (0.01 sec)
```

DROP INDEX

语法说明

该语句用于从当前所选的表中删除索引，如果索引不存在则会报错，除非使用 `IF EXISTS` 修饰符。

语法结构

```
> DROP INDEX index_name ON tbl_name
```

示例

```
create table t5(a int, b int, unique key(a));
mysql> show create table t5;
+-------+-------------------------------------------------------------------
------------------------+
| Table | Create Table |
+-------+-------------------------------------------------------------------
------------------------+
| t5 | CREATE TABLE `t5` (
`a` INT DEFAULT NULL,
`b` INT DEFAULT NULL,
UNIQUE KEY `a` (`a`)
) |
+-------+-------------------------------------------------------------------
------------------------+
1 row in set (0.01 sec)

create index b on t5(b);
mysql> show create table t5;
+-------+-------------------------------------------------------------------
--------------------------------------+
| Table | Create Table |
+-------+-------------------------------------------------------------------
--------------------------------------+
| t5 | CREATE TABLE `t5` (
`a` INT DEFAULT NULL,
`b` INT DEFAULT NULL,
UNIQUE KEY `a` (`a`),
KEY `b` (`b`)
) |
+-------+-------------------------------------------------------------------
--------------------------------------+
1 row in set (0.02 sec)

drop index b on t5;
mysql> show create table t5;
+-------+-------------------------------------------------------------------
------------------------+
| Table | Create Table |
+-------+-------------------------------------------------------------------
------------------------+
| t5 | CREATE TABLE `t5` (
`a` INT DEFAULT NULL,
```

```
`b` INT DEFAULT NULL,
UNIQUE KEY `a` (`a`)
) |
+-------+-------------------------------------------------------------------
-----------------------+
1 row in set (0.02 sec)
```

DROP TABLE

语法说明

该语句用于从当前所选的数据库中删除表，如果表不存在则会报错，除非使用 `IF EXISTS` 修饰符。

语法结构

```
> DROP TABLE [IF EXISTS] [db.]name
```

示例

```
CREATE TABLE table01(a int);

mysql> DROP TABLE table01;
Query OK, 0 rows affected (0.01 sec)
```

DROP VIEW

语法说明

`DROP VIEW` 语句表示删除视图。

如果语法参数列表中命名的任何视图都不存在，语句报错，并提示无法删除哪些不存在的视图，并且不做任何更改。

`IF EXISTS` 子句表示防止对不存在的视图发生错误。给出该子句时，将为每个不存在的视图生成一个 `NOTE`。

语法结构

```
> DROP VIEW [IF EXISTS]
view_name [, view_name] ...
```

示例

```
CREATE TABLE t1(c1 INT PRIMARY KEY, c2 INT);
CREATE VIEW v1 AS SELECT * FROM t1;

mysql> DROP VIEW v1;
Query OK, 0 rows affected (0.02 sec)
```

TRUNCATE TABLE

语法说明

`TRUNCATE TABLE` 语句用于用于删除表中的所有行，而不记录单个行删除操作。`TRUNCATE TABLE` 与没有 `WHERE` 子句的 `DELETE` 语句类似；但是，`TRUNCATE TABLE` 速度更快，使用的系统资源和事务日志资源更少。

`TRUNCATE TABLE` 有以下特点：

- `TRUNCATE TABLE` 删除之后，不可恢复。
- 如果表具有 `AUTO_INCREMENT` 列，则 `TRUNCATE TABLE` 语句将自动递增值重置为零。
- 如果表具有任何外键约束（`FOREIGN KEY`），则 `TRUNCATE TABLE` 语句会逐个删除行。
- 如果表没有任何外键约束（`FOREIGN KEY`），则 `TRUNCATE TABLE` 语句将删除该表并重新创建一个具有相同结构的新表

`DROP TABLE`、`TRUNCATE TABLE` 和 `DELETE TABLE` 的区别：

- `DROP TABLE`：当你不再需要该表时，用 `DROP TABLE`。
- `TRUNCATE TABLE`：当你仍要保留该表，但要删除所有记录时，用 `TRUNCATE TABLE`。
- `DELETE TABLE`：当你要删除部分记录时，用 `DELETE TABLE`。

语法结构

```
> TRUNCATE [TABLE] table_name;
```

语法释义

TABLE

TABLE 关键字是可选的。使用它来区分 `TRUNCATE TABLE` 语句和 `TRUNCATE` 函数。

示例

```
create table index_table_05 (col1 bigint not null auto_increment,col2
varchar(25),col3 int,col4 varchar(50),primary key (col1),unique key
col2(col2),key num_id(col4));
insert into index_table_05(col2,col3,col4) values ('apple',1,'10'),
('store',2,'11'),('bread',3,'12');
mysql> select * from index_table_05;
+------+-------+------+------+
| col1 | col2 | col3 | col4 |
+------+-------+------+------+
| 1 | apple | 1 | 10 |
| 2 | store | 2 | 11 |
| 3 | bread | 3 | 12 |
+------+-------+------+------+
3 rows in set (0.00 sec)

mysql> truncate table index_table_05;
Query OK, 0 rows affected (0.12 sec)

mysql> select * from index_table_05;
Empty set (0.03 sec)
```

INSERT

语法描述

`INSERT` 用于在表中插入新行。

语法结构

```
> INSERT INTO [db.]table [(c1, c2, c3)] VALUES (v11, v12, v13), (v21,
v22, v23), ...
```

示例

```
drop table if exists t1;
create table t1(a int default (1+12), b int);
insert into t1(b) values(1), (1);

mysql> select * from t1;
+------+------+
| a | b |
+------+------+
| 13 | 1 |
| 13 | 1 |
+------+------+
2 rows in set (0.01 sec)

drop table if exists t1;
create table t1 (a date);
insert into t1 values(DATE("2017-06-15 09:34:21")),(DATE("2019-06-25
10:12:21")),(DATE("2019-06-25 18:20:49"));

mysql> select * from t1;
+------------+
| a |
+------------+
| 2017-06-15 |
| 2019-06-25 |
| 2019-06-25 |
+------------+
3 rows in set (0.00 sec)

drop table if exists t;
CREATE TABLE t (i1 INT, d1 DOUBLE, e2 DECIMAL(5,2));
INSERT INTO t VALUES ( 6, 6.0, 10.0/3), ( null, 9.0, 10.0/3), ( 1, null,
10.0/3), ( 2, 2.0, null );

mysql> select * from t;
+------+------+------+
| i1 | d1 | e2 |
+------+------+------+
| 6 | 6 | 3.33 |
| NULL | 9 | 3.33 |
| 1 | NULL | 3.33 |
| 2 | 2 | NULL |
```

```
+------+------+------+
4 rows in set (0.01 sec)
```

INSERT INTO SELECT

语法说明

`INSERT INTO SELECT` 语句从一个表复制数据，然后把数据插入到一个已存在的表中。且目标表中任何已存在的行都不会受影响。

语法结构

```
INSERT INTO table2 (column1, column2, column3, ...)
SELECT column1, column2, column3, ...
FROM table1
WHERE condition;
```

示例

```
create table t1(id int, name varchar(10));
insert into t1 values(1, 'a');
insert into t1 values(2, 'b');
insert into t1 values(3, 'c');
create table t2(id int, appname varchar(10), country varchar(10));
insert into t2 values(1, 'appone', 'CN');
insert into t2 values(2, 'apptwo', 'CN');
INSERT INTO t1 (name) SELECT appname FROM t2;

mysql> select * from t1;
+------+--------+
| id | name |
+------+--------+
| 1 | a |
| 2 | b |
| 3 | c |
| NULL | appone |
| NULL | apptwo |
+------+--------+
```

DELETE

语法说明

`DELETE` 用于删除单表或多表中的记录。

语法结构

单表语法结构

```
DELETE FROM tbl_name [[AS] tbl_alias]
[WHERE where_condition]
[ORDER BY ...]
[LIMIT row_count]
```

`DELETE` 语句从 `tbl_name` 中删除行，并返回已删除的行数。

参数释义

- `WHERE` 从句用于指定用于标识要删除哪些行的条件。若无 `WHERE` 从句，则删除所有行。
- `ORDER BY` 从句，指按照指定的顺序删除行。
- `LIMIT` 从句用于限制可删除的行数。

示例

- **单表示例**

```
CREATE TABLE t1 (a bigint(3), b bigint(5) primary key);
insert INTO t1 VALUES (1,1),(1,2);
delete from t1 where a=1 limit 1;

mysql> select * from t1;
+------+------+
| a | b |
+------+------+
| 1 | 2 |
+------+------+
```

- **多表示例**：

同时也支持多表 JOIN 语句。

```
drop table if exists t1;
drop table if exists t2;
create table t1 (a int);
insert into t1 values(1), (2), (4);
create table t2 (b int);
insert into t2 values(1), (2), (5);
delete t1 from t1 join t2 where t1.a = 2;

mysql> select * from t1;
+------+
| a |
+------+
| 1 |
| 4 |
+------+
2 rows in set (0.00 sec)
```

```
drop database if exists db1;
drop database if exists db2;
create database db1;
create database db2;
use db2;
drop table if exists t1;
create table t1 (a int);
insert into t1 values (1),(2),(4);
use db1;
drop table if exists t2;
create table t2 (b int);
insert into t2 values(1),(2),(3);
delete from db1.t2, db2.t1 using db1.t2 join db2.t1 on db1.t2.b =
db2.t1.a where 2 > 1;

mysql> select * from db1.t2;
+------+
| b |
+------+
| 3 |
+------+
mysql> select * from db2.t1;
+------+
| a |
+------+
| 4 |
+------+
1 row in set (0.00 sec)
```

UPDATE

语法描述

`UPDATE` 用于修改表中的现有记录。

语法结构

单表语法结构

```
UPDATE table_reference
SET assignment_list
[WHERE where_condition]
[ORDER BY ...]
[LIMIT row_count]
```

参数释义

- `UPDATE` 将新值更新到指定表中现有行的列中。
- `SET` 从句指出要修改哪些列以及它们应该被赋予的值。每个值可以作为表达式给出，或者通过 `DEFAULT` 明确将列设置为默认值。
- `WHERE` 从句，用于指定用于标识要更新哪些行的条件。若无 `WHERE` 从句，则更新所有行。
- `ORDER BY` 从句，指按照指定的顺序更新行。
- `LIMIT` 从句用于限制可更新的行数。

示例

- **单表示例**

```
CREATE TABLE t1 (a bigint(3), b bigint(5) primary key);
insert INTO t1 VALUES (1,1),(1,2);
update t1 set a=2 where a=1 limit 1;

mysql> select * from t1;
+------+------+
| a | b |
+------+------+
| 2 | 1 |
| 1 | 2 |
+------+------+
```

- 多表示例

```
drop table if exists t1;
create table t1 (a int);
insert into t1 values(1), (2), (4);
drop table if exists t2;
create table t2 (b int);
insert into t2 values(1), (2), (3);
update t1, t2 set a = 1, b =2;

mysql> select * from t1;
+------+
| a |
+------+
| 1 |
| 1 |
| 1 |
+------+

update t1, t2 set a = null, b =null;

mysql> select * from t2;
+------+
| b |
+------+
| NULL |
| NULL |
| NULL |
+------+
mysql> select * from t1;
+------+
| a |
+------+
| NULL |
| NULL |
| NULL |
+------+
```

支持多表 JOIN 语句。

```
drop table if exists t1;
drop table if exists t2;
create table t1 (a int, b int, c int);
insert into t1 values(1, 2, 3), (4, 5, 6), (7, 8, 9);
create table t2 (a int, b int, c int);
insert into t2 values(1, 2, 3), (4, 5, 6), (7, 8, 9);
update t1 join t2 on t1.a = t2.a set t1.b = 222, t1.c = 333, t2.b = 222,
t2.c = 333;

mysql> select * from t1;
+------+------+------+
| a | b | c |
+------+------+------+
| 1 | 222 | 333 |
| 4 | 222 | 333 |
| 7 | 222 | 333 |
+------+------+------+

mysql> with t11 as (select * from (select * from t1) as t22) update t11
join t2 on t11.a = t2.a set t2.b = 666;

mysql> select * from t2;
+------+------+------+
| a | b | c |
+------+------+------+
| 1 | 666 | 333 |
| 4 | 666 | 333 |
| 7 | 666 | 333 |
+------+------+------+
3 rows in set (0.00 sec)
```

LOAD DATA

概述

`LOAD DATA` 语句可以极快地将文本文件中的行读入表中。你可以从服务器主机或 [S3 兼容对象存储](#)读取该文件。`LOAD DATA` 是 `SELECT ... INTO OUTFILE` 相反的操作。

- 将文件读回表中，使用 `LOAD DATA`。
- 将表中的数据写入文件，使用 `SELECT ... INTO OUTFILE`。
- `FIELDS` 和 `LINES` 子句的语法对于 `LOAD DATA` 和 `SELECT ... INTO OUTFILE` 这两个语句的使用方式一致，使用 Fields 和 Lines 参数来指定如何处理数据格式。

语法结构

```
> LOAD DATA [LOCAL]
INFILE 'file_name'
INTO TABLE tbl_name
[{FIELDS | COLUMNS}
[TERMINATED BY 'string']
[[OPTIONALLY] ENCLOSED BY 'char']
]
[LINES
[STARTING BY 'string']
[TERMINATED BY 'string']
]
[IGNORE number {LINES | ROWS}]
[SET column_name_1=nullif(column_name_1, expr1),
column_name_2=nullif(column_name_2, expr2)...]
[PARALLEL {'TRUE' | 'FALSE'}]
```

参数解释

上述语法结构中的参数解释如下：

INFILE

- `LOAD DATA INFILE 'file_name'`：

  命令行使用场景：需要加载的数据文件与 MatrixOne 主机服务器在同一台机器上。`file_name` 可以是文件的存放位置的相对路径名称，也可以是绝对路径名称。

- `LOAD DATA LOCAL INFILE 'file_name'`：

  命令行使用场景：需要加载的数据文件与 MatrixOne 主机服务器不在同一台机器上，即，数据文件在客户机上。`file_name` 可以是文件的存放位置的相对路径名称，也可以是绝对路径名称。

IGNORE LINES

`IGNORE number LINES` 子句可用于忽略文件开头的行。例如，你可以使用 `IGNORE 1 LINES` 跳过包含列名的初始标题行：

```
LOAD DATA INFILE '/tmp/test.txt' INTO TABLE table1 IGNORE 1 LINES;
```

FIELDS 和 LINES 参数说明

使用 `FIELDS` 和 `LINES` 参数来指定如何处理数据格式。

对于 `LOAD DATA` 和 `SELECT ... INTO OUTFILE` 语句，`FIELDS` 和 `LINES` 子句的语法是相同的。这两个子句都是可选的，但如果两者都指定，则 `FIELDS` 必须在 `LINES` 之前。

如果指定 `FIELDS` 子句，那么 `FIELDS` 的每个子句（`TERMINATED BY`、`[OPTIONALLY] ENCLOSED BY`）也是可选的，除非你必须至少指定其中一个。

`LOAD DATA` 也支持使用十六进制 `ASCII` 字符表达式或二进制 `ASCII` 字符表达式作为 `FIELDS ENCLOSED BY` 和 `FIELDS TERMINATED BY` 的参数。

如果不指定处理数据的参数，则使用默认值如下：

```
FIELDS TERMINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '\n'
```

意义如下：

- `FIELDS TERMINATED BY ','`：以*,* 作为分隔符
- `ENCLOSED BY '"'`：使用双引号把各个字段括起来
- `LINES TERMINATED BY '\n'`：以 '\n' 为行间分隔符

FIELDS TERMINATED BY

`FIELDS TERMINATED BY` 表示字段与字段之间的分隔符，使用 `FIELDS TERMINATED BY` 就可以指定每个数据的分隔符号。

`FIELDS TERMINATED BY` 指定的值可以超过一个字符。

- **正确示例**：

例如，读取使用*逗号*分隔的文件，语法是：

```
LOAD DATA INFILE 'data.txt' INTO TABLE table1
FIELDS TERMINATED BY ',';
```

- **错误示例**：

如果你使用如下所示的语句读取文件，将会产生报错，因为它表示的是 `LOAD DATA` 查找字段之间的制表符：

```
LOAD DATA INFILE 'data.txt' INTO TABLE table1
FIELDS TERMINATED BY '\t';
```

这样可能会导致结果被解释为每个输入行都是一个字段，你可能会遇到 `ERROR 20101 (HY000): internal error: the table column is larger than input data column` 错误。

FIELDS ENCLOSED BY

`FIELDS TERMINATED BY` 指定的值包含输入值的字符。`ENCLOSED BY` 指定的值必须是单个字符；如果输入值不一定包含在引号中，需要在 `ENCLOSED BY` 选项之前使用 `OPTIONALLY`。

如下面的例子所示，即表示一部分输入值用可以用引号括起来，另一些可以不用引号括起来：

```
LOAD DATA INFILE 'data.txt' INTO TABLE table1
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"';
```

如果 `ENCLOSED BY` 前不加 `OPTIONALLY`，比如说，`ENCLOSED BY '"'` 就表示使用双引号把各个字段都括起来。

LINES TERMINATED BY

`LINES TERMINATED BY` 用于指定一行的分隔符。`LINES TERMINATED BY` 值可以超过一个字符。

例如，如果 *csv* 文件中的行以回车符/换行符对结束，你在加载它时，可以使用 `LINES TERMINATED BY '\r\n'`：

```
LOAD DATA INFILE 'data.txt' INTO TABLE table1
FIELDS TERMINATED BY ',' ENCLOSED BY '"'
LINES TERMINATED BY '\r\n';
```

LINE STARTING BY

如果所有输入行都有一个你想忽略的公共前缀，你可以使用 `LINES STARTING BY` 'prefix_string' 来忽略前缀和前缀之前的任何内容。

如果一行不包含前缀，则跳过整行。如下语句所示：

```
LOAD DATA INFILE '/tmp/test.txt' INTO TABLE table1
FIELDS TERMINATED BY ',' LINES STARTING BY 'xxx';
```

如果数据文件是如下样式：

```
xxx"abc",1
something xxx"def",2
"ghi",3
```

则输出的结果行是 (“abc”，1) 和 (“def”，2)。文件中的第三行由于没有前缀，则被忽略。

SET

MatrixOne 当前仅支持 `SET column_name=nullif(column_name,expr)`。即，当 `column_name = expr`，返回 `NULL`；否则，则返回 `column_name`。例如，`SET a=nullif(a, 1)`，当 a=1 时，返回 `NULL`；否则，返回 a 列原始的值。

使用这种方法，可以在加载文件时，设置参数 `SET column_name=nullif(column_name,"null")`，用于返回列中的 `NULL` 值。

示例

1. 本地文件 `test.txt` 详情如下：

```
id,user_name,sex
1,"weder","man"
2,"tom","man"
null,wederTom,"man"
```

2. 在 MatrixOne 中新建一个表 `user`：

```
create database aaa;
use aaa;
CREATE TABLE `user` (`id` int(11) ,`user_name` varchar(255) ,`sex`
varchar(255));
```

3. 使用下面的命令行将 `test.txt` 导入至表 `user`：

```
LOAD DATA INFILE '/tmp/test.txt' INTO TABLE user SET
id=nullif(id,"null");
```

4. 导入后的表内容如下：

```
select * from user;
+------+-----------+------+
| id | user_name | sex |
+------+-----------+------+
| 1 | weder | man |
| 2 | tom | man |
| null | wederTom | man |
+------+-----------+------+
```

PARALLEL

对于一个格式良好的大文件，例如 *JSONLines* 文件，或者一行数据中没有换行符的 *CSV* 文件，都可以使用 `PARALLEL` 就可以对该文件进行并行加载，以加快加载速度。

例如，对于 2 个 G 的大文件，使用两个线程去进行加载，第 2 个线程先拆分定位到 1G 的位置，然后一直往后读取并进行加载。这样就可以做到两个线程同时读取大文件，每个线程读取 1G 的数据。

开启/关闭并行加载命令行示例：

```
-- 打开并行加载
load data infile 'file_name' into table tbl_name FIELDS TERMINATED BY '|'
ENCLOSED BY '\"' LINES TERMINATED BY '\n' IGNORE 1 LINES PARALLEL 'TRUE';

-- 关闭并行加载
load data infile 'file_name' into table tbl_name FIELDS TERMINATED BY '|'
ENCLOSED BY '\"' LINES TERMINATED BY '\n' IGNORE 1 LINES PARALLEL
'FALSE';

-- 默认关闭并行加载
load data infile 'file_name' into table tbl_name FIELDS TERMINATED BY '|'
ENCLOSED BY '\"' LINES TERMINATED BY '\n' IGNORE 1 LINES;
```

Note: `LOAD` 语句中如果不加 `PARALLEL` 字段，对于 *CSV* 文件，是默认关闭并行加载；对于 *JSONLines* 文件，默认开启并行加载。如果 *CSV* 文件中有行结束符，比如 '\n'，否则有可能会导致文件加载时数据出错，如果文件过大，建议从换行符为起止点手动拆分文件后再开启并行加载。

支持的文件格式

在 MatrixOne 当前版本中，`LOAD DATA` 支持 *CSV* 格式和 *JSONLines* 格式文件。

有关导入这两种格式的文档，参见导入*. csv* 格式数据和导入 JSONLines 数据。

示例

你可以在 SSB 测试中了解 `LOAD DATA` 语句的用法，参见完成 SSB 测试。

语法示例如下：

```
> LOAD DATA INFILE '/ssb-dbgen-path/lineorder_flat.tbl ' INTO TABLE
lineorder_flat;
```

上面这行语句表示：将 */ssb-dbgen-path/* 这个目录路径下的 *lineorder_flat.tbl* 数据集加载到 MatrixOne 的数据表 *lineorder_flat* 中。

你也可以参考以下语法示例，来快速了解 `LOAD DATA`：

示例 1：LOAD CSV

简单导入示例

本地命名为 *char_varchar.csv* 文件内数据如下：

```
a|b|c|d
"a"|"b"|"c"|"d"
'a'|'b'|'c'|'d'
"'a'"|"'b'"|"'c'"|"'d'"
"aa|aa"|"bb|bb"|"cc|cc"|"dd|dd"
"aa|"|"bb|"|"cc|"|"dd|"
"aa|||aa"|"bb|||bb"|"cc|||cc"|"dd|||dd"
"aa'|'||aa"|"bb'|'||bb"|"cc'|'||cc"|"dd'|'||dd"
aa"aa|bb"bb|cc"cc|dd"dd
"aa"aa"|"bb"bb"|"cc"cc"|"dd"dd"
"aa""aa"|"bb""bb"|"cc""cc"|"dd""dd"
"aa"""aa"|"bb"""bb"|"cc"""cc"|"dd"""dd"
"aa""""aa"|"bb""""bb"|"cc""""cc"|"dd""""dd"
"aa""|aa"|"bb""|bb"|"cc""|cc"|"dd""|dd"
"aa"""""|aa"|"bb"""""|bb"|"cc"""""|cc"|"dd"""""|dd"
|||
||||
""|""|""|
"""""|"""""|"""""|"""""
""""""""|""""""""|""""""""|""""""""
```

在 MatrixOne 中建表：

```
mysql> drop table if exists t1;
Query OK, 0 rows affected (0.01 sec)

mysql> create table t1(
-> col1 char(225),
-> col2 varchar(225),
-> col3 text,
-> col4 varchar(225)
-> );
Query OK, 0 rows affected (0.02 sec)
```

将数据文件导入到 MatrixOne 中的表 t1：

```
load data infile '<your-local-file-path>/char_varchar.csv' into table t1
fields terminated by'|';
```

查询结果如下：

```
mysql> select * from t1;
+-----------+-----------+-----------+-----------+
| col1 | col2 | col3 | col4 |
+-----------+-----------+-----------+-----------+
| a | b | c | d |
| a | b | c | d |
| 'a' | 'b' | 'c' | 'd' |
| 'a' | 'b' | 'c' | 'd' |
| aa|aa | bb|bb | cc|cc | dd|dd |
| aa| | bb| | cc| | dd| |
| aa|||aa | bb|||bb | cc|||cc | dd|||dd |
| aa'|'||aa | bb'|'||bb | cc'|'||cc | dd'|'||dd |
| aa"aa | bb"bb | cc"cc | dd"dd |
| aa"aa | bb"bb | cc"cc | dd"dd |
| aa"aa | bb"bb | cc"cc | dd"dd |
| aa""aa | bb""bb | cc""cc | dd""dd |
| aa""aa | bb""bb | cc""cc | dd""dd |
| aa"|aa | bb"|bb | cc"|cc | dd"|dd |
| aa""|aa | bb""|bb | cc""|cc | dd""|dd |
| | | | |
| | | | |
| | | | |
| " | " | " | " |
| "" | "" | "" | "" |
+-----------+-----------+-----------+-----------+
20 rows in set (0.00 sec)
```

增加条件导入示例

沿用上面的简单示例，你可以修改一下 LOAD DATA 语句，在末尾增加条件 `LINES STARTING BY 'aa' ignore 10 lines;`：

```
delete from t1;
load data infile '<your-local-file-path>/char_varchar.csv' into table t1
fields terminated by'|' LINES STARTING BY 'aa' ignore 10 lines;
```

查询结果如下：

```
mysql> select * from t1;
+---------+---------+---------+---------+
| col1 | col2 | col3 | col4 |
+---------+---------+---------+---------+
| aa"aa | bb"bb | cc"cc | dd"dd |
| aa""aa | bb""bb | cc""cc | dd""dd |
| aa""aa | bb""bb | cc""cc | dd""dd |
| aa"|aa | bb"|bb | cc"|cc | dd"|dd |
| aa""|aa | bb""|bb | cc""|cc | dd""|dd |
| | | | |
| | | | |
| | | | |
| " | " | " | " |
| "" | "" | "" | "" |
+---------+---------+---------+---------+
10 rows in set (0.00 sec)
```

可以看到，查询结果忽略了前 10 行，并且忽略了公共前缀 aa。

有关如何导入 *CSV* 格式文件的详细步骤，参见[导入*. csv* 格式数据](#)。

示例 2：LOAD JSONLines

简单导入示例

本地命名为 *jsonline_array.jl* 文件内数据如下：

```
[true,1,"var","2020-09-07","2020-09-07 00:00:00","2020-09-07
00:00:00","18",121.11,["1",2,null,false,true,{"q":1}],"1qaz",null,null]
["true","1","var","2020-09-07","2020-09-07 00:00:00","2020-09-07
00:00:00","18","121.11",{"c":1,"b":["a","b",{"q":4}]},"1aza",null,null]
```

在 MatrixOne 中建表：

```
mysql> drop table if exists t1;
Query OK, 0 rows affected (0.01 sec)

mysql> create table t1(col1 bool,col2 int,col3 varchar(100), col4
date,col5 datetime,col6 timestamp,col7 decimal,col8 float,col9 json,col10
text,col11 json,col12 bool);
Query OK, 0 rows affected (0.03 sec)
```

将数据文件导入到 MatrixOne 中的表 t1：

```
load data infile {'filepath'='<your-local-file-
path>/jsonline_array.jl','format'='jsonline','jsondata'='array'} into
table t1;
```

查询结果如下：

```
mysql> select * from t1;
+------+------+------+------------+---------------------+----------------
-----+------+--------+---------------------------------------+-------+---
----+-------+
| col1 | col2 | col3 | col4 | col5 | col6 | col7 | col8 | col9 | col10 |
col11 | col12 |
+------+------+------+------------+---------------------+----------------
-----+------+--------+---------------------------------------+-------+---
----+-------+
| true | 1 | var | 2020-09-07 | 2020-09-07 00:00:00 | 2020-09-07 00:00:00
| 18 | 121.11 | ["1", 2, null, false, true, {"q": 1}] | 1qaz | NULL |
NULL |
| true | 1 | var | 2020-09-07 | 2020-09-07 00:00:00 | 2020-09-07 00:00:00
| 18 | 121.11 | {"b": ["a", "b", {"q": 4}], "c": 1} | 1aza | NULL | NULL
|
+------+------+------+------------+---------------------+----------------
-----+------+--------+---------------------------------------+-------+---
----+-------+
2 rows in set (0.00 sec)
```

增加条件导入示例

沿用上面的简单示例，你可以修改一下 LOAD DATA 语句，增加 `ignore 1 lines` 在语句的末尾，体验一下区别：

```
delete from t1;
load data infile {'filepath'='<your-local-file-
path>/jsonline_array.jl','format'='jsonline','jsondata'='array'} into
table t1 ignore 1 lines;
```

查询结果如下：

```
mysql> select * from t1;
+------+------+------+------------+---------------------+----------------
-----+------+--------+-------------------------------------+-------+-----
--+-------+
| col1 | col2 | col3 | col4 | col5 | col6 | col7 | col8 | col9 | col10 |
col11 | col12 |
+------+------+------+------------+---------------------+----------------
-----+------+--------+-------------------------------------+-------+-----
--+-------+
| true | 1 | var | 2020-09-07 | 2020-09-07 00:00:00 | 2020-09-07 00:00:00
| 18 | 121.11 | {"b": ["a", "b", {"q": 4}], "c": 1} | 1aza | NULL | NULL
|
+------+------+------+------------+---------------------+----------------
-----+------+--------+-------------------------------------+-------+-----
--+-------+
1 row in set (0.00 sec)
```

可以看到，查询结果忽略掉了第一行。

有关如何导入 *JSONLines* 格式文件的详细步骤，参见[导入 JSONLines 数据](#)。

限制

1. `REPLACE` 和 `IGNORE` 修饰符解决唯一索引的冲突：`REPLACE` 表示若表中已经存在则用新的数据替换掉旧的数据，而 `IGNORE` 则表示保留旧的数据，忽略掉新数据。这两个修饰符在 MatrixOne 中尚不支持。
2. MatrixOne 当前部分支持 `SET`，仅支持 `SET columns_name=nullif(col_name,expr2)`。
3. 开启并行加载操作时必须要保证文件中每行数据中不包含指定的行结束符，比如 '\n'，否则有可能会导致文件加载时数据出错。
4. 文件的并行加载要求文件必须是非压缩格式，暂不支持并行加载压缩格式的文件。
5. `LOAD DATA LOCAL` 暂不支持并行加载。

6. 如果你需要用 `LOAD DATA LOCAL` 进行本地加载，则需要使用命令行连接 MatrixOne 服务主机：`mysql -h <mo-host -ip> -P 6001 -udump -p111 --local-infile`。

7. MatrixOne 当前暂不支持 `ESCAPED BY`，写入或读取特殊字符与 MySQL 存在一定的差异。

INTERVAL

语法说明

- `INTERVAL` 用于日期和时间计算。
- `INTERVAL` 可以用于函数运算 `DATE_ADD()` 和 `DATE_SUB()`。
- `INTERVAL` 可以再表达式中使用 `+` 或 `-` 运算符来进行运算。

```
date + INTERVAL expr unit
date - INTERVAL expr unit
```

- 无论 `+` 运算符的左边或者右边，只要它其中一边的表达式是一个 `date` 或 `datetime` 值，则可以使用 `INTERVAL expr`。
- 对于 `-` 运算符，仅仅可以在 `-` 的右边使用 `INTERVAL expr`。

语法结构

```
> INTERVAL (expr,unit)
```

参数释义

参数	说明
expr	任何数值类型与字符串列的列名
unit	说明符，例如 HOUR、DAY 或 WEEK

> **注意**
>
> `INTERVAL` 关键字和 `unit` 不区分大小写。

- **Interval 表达式和 unit 参数**

unit 值	描述
MICROSECOND	MICROSECONDS
SECOND	SECONDS
MINUTE	MINUTES
HOUR	HOURS
DAY	DAYS
WEEK	WEEKS
MONTH	MONTHS
QUARTER	QUARTERS
YEAR	YEARS
SECOND_MICROSECOND	'SECONDS.MICROSECONDS'
MINUTE_MICROSECOND	'MINUTES .MICROSECONDS'
MINUTE_SECOND	'MINUTES '
HOUR_MICROSECOND	'HOURS:MINUTES .MICROSECONDS'
HOUR_SECOND	'HOURS:MINUTES '
HOUR_MINUTE	'HOURS '
DAY_MICROSECOND	'DAYS HOURS:MINUTES .MICROSECONDS'
DAY_SECOND	'DAYS HOURS:MINUTES

unit 值	描述
	'
DAY_MINUTE	'DAYS HOURS '
DAY_HOUR	'DAYS HOURS'
YEAR_MONTH	'YEARS-MONTHS'

你可以在 `expr` 中使用任何标点分隔符。上表所示为建议的分隔符。

示例

示例 1

- 与 `DATE_ADD()` 和 `DATE_SUB()` 一起使用：

```
mysql> SELECT DATE_SUB('2018-05-01',INTERVAL 1 YEAR);
+----------------------------------------+
| date_sub(2018-05-01, interval(1, year)) |
+----------------------------------------+
| 2017-05-01 |
+----------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT DATE_ADD('2020-12-31 23:59:59', INTERVAL 1 SECOND);
+-----------------------------------------------------+
| date_add(2020-12-31 23:59:59, interval(1, second)) |
+-----------------------------------------------------+
| 2021-01-01 00:00:00 |
+-----------------------------------------------------+
1 row in set (0.01 sec)

mysql> SELECT DATE_ADD('2018-12-31 23:59:59', INTERVAL 1 DAY);
+-------------------------------------------------+
| date_add(2018-12-31 23:59:59, interval(1, day)) |
+-------------------------------------------------+
| 2019-01-01 23:59:59 |
+-------------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT DATE_ADD('2100-12-31 23:59:59', INTERVAL '1:1'
MINUTE_SECOND);
+-----------------------------------------------------------------+
| date_add(2100-12-31 23:59:59, interval(1:1, minute_second)) |
+-----------------------------------------------------------------+
| 2101-01-01 00:01:00 |
+-----------------------------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT DATE_SUB('2025-01-01 00:00:00', INTERVAL '1 1:1:1'
DAY_SECOND);
+-----------------------------------------------------------------+
| date_sub(2025-01-01 00:00:00, interval(1 1:1:1, day_second)) |
+-----------------------------------------------------------------+
| 2024-12-30 22:58:59 |
+-----------------------------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT DATE_ADD('1900-01-01 00:00:00', INTERVAL '-1 10' DAY_HOUR);
+-----------------------------------------------------------+
```

```
| date_add(1900-01-01 00:00:00, interval(-1 10, day_hour)) |
+----------------------------------------------------------+
| 1899-12-30 14:00:00.000000 |
+----------------------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT DATE_SUB('1998-01-02', INTERVAL 31 DAY);
+----------------------------------------+
| date_sub(1998-01-02, interval(31, day)) |
+----------------------------------------+
| 1997-12-02 |
+----------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT DATE_ADD('1992-12-31 23:59:59.000002', INTERVAL '1.999999'
SECOND_MICROSECOND);
+--------------------------------------------------------------------------
------+
| date_add(1992-12-31 23:59:59.000002, interval(1.999999,
second_microsecond)) |
+--------------------------------------------------------------------------
------+
| 1993-01-01 00:00:01.000001 |
+--------------------------------------------------------------------------
------+
1 row in set (0.00 sec)
```

示例 2

- 与 `+` 或 `-` 一起使用：

```
mysql> SELECT '2018-12-31 23:59:59' + INTERVAL 1 SECOND;
+-------------------------------------------+
| 2018-12-31 23:59:59 + interval(1, second) |
+-------------------------------------------+
| 2019-01-01 00:00:00 |
+-------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT INTERVAL 1 DAY + '2018-12-31';
+-------------------------------+
| interval(1, day) + 2018-12-31 |
+-------------------------------+
| 2019-01-01 |
+-------------------------------+
1 row in set (0.00 sec)

mysql> SELECT '2025-01-01' - INTERVAL 1 SECOND;
+-----------------------------------+
| 2025-01-01 - interval(1, second) |
+-----------------------------------+
| 2024-12-31 23:59:59 |
+-----------------------------------+
1 row in set (0.00 sec)
```

示例 3

如果你在一个 `date` 值上加上或减去一个包含时间部分的值，执行结果会自动转换为一个 `datetime` 值：

```
mysql> SELECT DATE_ADD('2023-01-01', INTERVAL 1 DAY);
+-----------------------------------------+
| date_add(2023-01-01, interval(1, day)) |
+-----------------------------------------+
| 2023-01-02 |
+-----------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT DATE_ADD('2023-01-01', INTERVAL 1 HOUR);
+-----------------------------------------+
| date_add(2023-01-01, interval(1, hour)) |
+-----------------------------------------+
| 2023-01-01 01:00:00 |
+-----------------------------------------+
1 row in set (0.01 sec)
```

示例 4

如果添加了 `MONTH`、`YEAR_MONTH` 或 `YEAR`，并且执行结果的日期的某一天比当月的最大天数大，则该天将被调整为当月的最大天数：

```
mysql> SELECT DATE_ADD('2019-01-30', INTERVAL 1 MONTH);
+-----------------------------------------+
| date_add(2019-01-30, interval(1, month)) |
+-----------------------------------------+
| 2019-02-28 |
+-----------------------------------------+
1 row in set (0.00 sec)
```

示例 5

`date` 不能用错误的日期，如执行 *2016-07-00* 或格式严重错误的日期则结果为 `NULL`。

```
mysql> SELECT DATE_ADD('2016-07-00', INTERVAL 1 DAY);
+----------------------------------------+
| date_add(2016-07-00, interval(1, day)) |
+----------------------------------------+
| NULL |
+----------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT '2005-03-32' + INTERVAL 1 MONTH;
+----------------------------------+
| 2005-03-32 + interval(1, month) |
+----------------------------------+
| NULL |
+----------------------------------+
1 row in set (0.00 sec)
```

LAST_QUERY_ID()

语法说明

返回当前会话中指定查询的 ID。如果未指定查询，则返回最近执行的查询。

语法结构

```
> LAST_QUERY_ID([ <num> ])
```

参数释义

- **num**：根据当前会话中查询的位置指定要返回的查询。默认为 -1。

使用释义

num 为正数，即从会话中执行的第一个查询开始。例如：

- `LAST_QUERY_ID(1)`：返回第一个查询。
- `LAST_QUERY_ID(2)`：返回第二个查询。
- `LAST_QUERY_ID(6)`：返回第六个查询。

num 为负数，即从会话中最近执行的查询开始。例如：

- `LAST_QUERY_ID(-1)`：返回最近执行的查询（相当于 LAST_QUERY_ID()）。
- `LAST_QUERY_ID(-2)`：返回最近执行的第二个查询。

示例

```
mysql> SELECT LAST_QUERY_ID(-1);
+--------------------------------------+
| last_query_id(-1) |
+--------------------------------------+
| af974680-b1b5-11ed-8eb9-5ad2460dea4f |
+--------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT LAST_QUERY_ID();
+--------------------------------------+
| last_query_id() |
+--------------------------------------+
| 550e4d44-b1b5-11ed-8eb9-5ad2460dea4f |
+--------------------------------------+
1 row in set (0.00 sec)
```

LAST_INSERT_ID()

语法说明

若表中含自增字段 `AUTO_INCREMENT`，则向表中插入一条记录后，可以调用 `LAST_INSERT_ID()` 来获得最近插入的那行记录的自增字段值。

如果没有插入参数，LAST_INSERT_ID() 返回一个 BIGINT UNSIGNED（64 位）值，该值表示作为最近执行的 INSERT 语句的结果成功插入到 `AUTO_INCREMENT` 列的第一个自动生成的值。返回值取决于之前 `AUTO_INCREMENT` 列的值，如果你之前没有插入一个列，那么返回值从 1 开始，如果你之前插入了一个列，那么返回值为 `AUTO_INCREMENT` 列的值增加 1。

如果没有成功插入参数，`LAST_INSERT_ID()` 的值保持不变。

在 MySQL 中，如果使用单个 `INSERT` 语句插入多行，则 `LAST_INSERT_ID()` 仅返回为第一个插入行生成的值。例如：

```
mysql> CREATE TABLE t (id INT AUTO_INCREMENT NOT NULL PRIMARY KEY, name
VARCHAR(10) NOT NULL);
mysql> INSERT INTO t VALUES (NULL, 'Bob');
mysql> SELECT * FROM t;
+----+------+
| id | name |
+----+------+
| 1 | Bob |
+----+------+
mysql> SELECT LAST_INSERT_ID();
+------------------+
| LAST_INSERT_ID() |
+------------------+
| 1 |
+------------------+
mysql> INSERT INTO t VALUES (NULL, 'Mary'), (NULL, 'Jane'), (NULL,
'Lisa');
mysql> SELECT * FROM t;
+----+------+
| id | name |
+----+------+
| 1 | Bob |
| 2 | Mary |
| 3 | Jane |
| 4 | Lisa |
+----+------+
mysql> SELECT LAST_INSERT_ID();
+------------------+
| LAST_INSERT_ID() |
+------------------+
| 2 |
+------------------+
```

但是在 MatrixOne 中，我们有不同的行为；如果使用单个 `INSERT` 语句插入多行，则 `LAST_INSERT_ID()` 返回为最后插入的行生成的值。与上面的示例一样，当您执行 `INSERT INTO t VALUES (NULL, 'Mary'), (NULL, 'Jane'), (NULL, 'Lisa');` 时，`LAST_INSERT_ID()` 将返回 4。

语法结构

```
LAST_INSERT_ID(), LAST_INSERT_ID(expr)
```

示例

```
create table t1(a int auto_increment primary key);
insert into t1 values();
mysql> select last_insert_id();
+------------------+
| last_insert_id() |
+------------------+
| 1 |
+------------------+
1 row in set (0.02 sec)

insert into t1 values(11);
insert into t1 values();
mysql> select last_insert_id();
+------------------+
| last_insert_id() |
+------------------+
| 12 |
+------------------+
1 row in set (0.02 sec)

insert into t1 values(null);
mysql> select last_insert_id();
+------------------+
| last_insert_id() |
+------------------+
| 13 |
+------------------+
1 row in set (0.02 sec)

create table t2(a int auto_increment primary key);
insert into t2 values();
mysql> select last_insert_id();
+------------------+
| last_insert_id() |
+------------------+
| 1 |
+------------------+
1 row in set (0.02 sec)

insert into t2 values(100);
insert into t2 values();
mysql> select last_insert_id();
```

```
+------------------+
| last_insert_id() |
+------------------+
| 101 |
+------------------+
1 row in set (0.02 sec)
```

运算符概述

算数运算符

名称	描述
%,MOD	取余
*	乘法
+	加法
-	减法
-	负号
/	除法
DIV	用于整数相除

赋值运算符

名称	描述
=	等于运算符，用于赋值

二进制运算符

名称	描述
&	位运算符与，按位与
>>	位移运算符右移
<<	位移运算符右移
^	按位异或
[	](bit-functions-and-operators/bitwise-or.md)
~	一元运算符，二进制取反

强制转换函数和运算符

名称	描述
CAST()	将值转换为特定类型，用于小数转数值和字符型
CONVERT()	将值转换为特定类型，用于日期和时间值、小数之间进行转换

比较函数和运算符

名称	描述
>	大于
>=	大于等于
<	小于
<>,!=	不等于
<=	小于等于
=	等于
BETWEEN ... AND ...	在两值之间
IN()	在集合中
IS	测试值是否是布尔值，若是布尔值，则返回“true”
IS NOT	测试值是否是布尔值，IS 的否定用法
IS NOT NULL	不为空
IS NULL	为空
LIKE	模糊匹配
NOT BETWEEN ... AND ...	不在两值之间
NOT LIKE	模糊匹配，Like 的否定用法
COALESCE	返回第一个非空值

控制流函数

名称	描述
CASE	Case 运算符
IF()	If/else 语句

逻辑运算符

名称	描述
AND,&&	逻辑与
NOT,!	逻辑非
OR	逻辑或
XOR	逻辑异或

运算符的优先级

运算符从高到低的优先级如下所示。在同一行中的运算符优先级相等。

优先级从高到低排序	运算符
1	INTERVAL
2	BINARY, COLLATE
3	!
4	- (unary minus), ~ (unary bit inversion)
5	^
6	*, /, DIV, %, MOD
7	-, +
8	<<, >>
9	&
10	|
11	= (comparison), <=>, >=, >, <=, <, <>, !=, IS, LIKE, , IN, MEMBER OF
12	BETWEEN, CASE, WHEN, THEN, ELSE
13	NOT
14	AND, &&
15	XOR
16	OR, ||
17	= (assignment)

运算符 `=` 的优先级取决于它是用作比较操作符还是赋值操作符。当用作比较操作符时，它与 `>=`，`>`，`<=`，`<`，`<>`，`!=`，`IS`，`LIKE` 和 `IN()` 具有相同的优先级。

对于在表达式中具有相同优先级的运算符，求值将从左到右进行计算，但赋值将从右到左进行计算。

算数运算符概述

名称	描述
%,MOD	取余
*	乘法
+	加法
-	减法
-	负号
/	除法
DIV	整数相除

— 对于 `-`、`+` 和 `*`，如果两个运算数值都是整数，计算结果将以 BIGINT (64 位) 精度计算。

— 如果两个运算数值都是整数且其中任何一个都是无符号的，则结果为无符号整数。

— 如果 `+`、`-`、`/`、`*`、`%` 中的任何一个运算数值是一个实值或字符串值，则结果的精度为运算数的最大精度。

在使用 `/` 进行除法时，当使用两个精确值运算数时，结果的精度范围是第一个运算数值的精度范围加上 `div_precision_increment` 系统变量的值。例如，表达式 5.05 / 0.014 的结果有一个小数 13 位的精度 (360.7142857142857)。

这些规则适用于每个操作，例如嵌套计算取每个组件的精度。因此，(14620 / 9432456)/(24250 / 9432456) 首先解析为 (0.0014)/(0.0026)，最终结果有 16 位小数 (0.6028865979381443)。

限制

- 算数运算符只适用于数字之间的计算。
- 确保各个组件和子组件的计算的精度标准，参见强制转换函数和运算符概述。

%,MOD

运算符说明

`%,MOD` 运算符用于取余运算。返回结果为 N 除以 M 的余数。

语法结构

```
> SELECT N % M, N MOD M;
```

示例

```
mysql> select 12 mod 5;
+--------+
| 12 % 5 |
+--------+
| 2 |
+--------+
1 row in set (0.00 sec)
```

```
create table t2(c1 int, c2 int);
insert into t2 values (-3, 2);
insert into t2 values (1, 2);

mysql> select c1 mod 2 from t2;
+--------+
| c1 % 2 |
+--------+
| -1 |
| 1 |
+--------+
2 rows in set (0.01 sec)

mysql> select c1 mod c2 from t2;
+---------+
| c1 % c2 |
+---------+
| -1 |
| 1 |
+---------+
2 rows in set (0.01 sec)
```

*****

运算符说明

`*` 运算符用于乘法运算。

语法结构

```
> SELECT value1*value2;
```

```
> SELECT column1*column2... FROM table_name;
```

示例

```
mysql> select 1123.2333*1233.3331;
+-----------------------+
| 1123.2333 * 1233.3331 |
+-----------------------+
| 1385320.8079122303 |
+-----------------------+
1 row in set (0.01 sec)
```

```
create table t2(c1 int, c2 int);
insert into t2 values (-3, 2);
insert into t2 values (1, 2);

mysql> select c1*2 from t2;
+--------+
| c1 * 2 |
+--------+
| -6 |
| 2 |
+--------+
2 rows in set (0.00 sec)

mysql> select c1*c2 from t2;
+---------+
| c1 * c2 |
+---------+
| -6 |
| 2 |
+---------+
2 rows in set (0.01 sec)
```

➕

运算符说明

`+` 运算符用于加法运算。

语法结构

```
> SELECT value1+value2;
```

```
> SELECT column1+column2... FROM table_name;
```

示例

```
mysql> select 1123.2333+1233.3331;
+-----------------------+
| 1123.2333 + 1233.3331 |
+-----------------------+
| 2356.5664 |
+-----------------------+
1 row in set (0.01 sec)
```

```
create table t2(c1 int, c2 int);
insert into t2 values (-3, 2);
insert into t2 values (1, 2);

mysql> select c1+5 from t2;
+--------+
| c1 + 5 |
+--------+
| 2 |
| 6 |
+--------+
2 rows in set (0.00 sec)

mysql> select c1+c2 from t2;
+---------+
| c1 + c2 |
+---------+
| -1 |
| 3 |
+---------+
2 rows in set (0.00 sec)
```

-

运算符说明

`-` 运算符用于减法运算。

语法结构

```
> SELECT value1-value2;
```

```
> SELECT column1-column2... FROM table_name;
```

示例

```
mysql> select 1123.2333-1233.3331;
+-----------------------+
| 1123.2333 - 1233.3331 |
+-----------------------+
| -110.09979999999996 |
+-----------------------+
1 row in set (0.00 sec)
```

```
create table t2(c1 int, c2 int);
insert into t2 values (-3, 2);
insert into t2 values (1, 2);

mysql> select c1-5 from t2;
+--------+
| c1 - 5 |
+--------+
| -8 |
| -4 |
+--------+
2 rows in set (0.00 sec)

mysql> select c1-c2 from t2;
+---------+
| c1 - c2 |
+---------+
| -5 |
| -1 |
+---------+
2 rows in set (0.01 sec)
```

-

运算符说明

`-` 为负号运算符。负号运算符将表达式的符号从正数反转为负数，反之亦然。

语法结构

```
> SELECT -column1, -column2, ...
FROM table_name;
```

示例

```
mysql> select -2;
+------+
| -2 |
+------+
| -2 |
+------+
```

```
create table t2(c1 int, c2 int);
insert into t2 values (-3, 2);
insert into t2 values (1, 2);

mysql> select -c1 from t2;
+------+
| -c1 |
+------+
| 3 |
| -1 |
+------+
2 rows in set (0.00 sec)
```

/

运算符说明

`/` 运算符用于除法运算。

语法结构

```
> SELECT value1/value2;
```

```
> SELECT column1/column2... FROM table_name;
```

除法运算不可以除以 0。

示例

```
mysql> select 1123.2333/1233.3331;
+----------------------+
| 1123.2333 / 1233.3331 |
+----------------------+
| 0.9107298750029493 |
+----------------------+
1 row in set (0.00 sec)
```

```
create table t2(c1 int, c2 int);
insert into t2 values (-3, 2);
insert into t2 values (1, 2);

mysql> select c1/2 from t2;
+--------+
| c1 / 2 |
+--------+
| -1.5 |
| 0.5 |
+--------+
2 rows in set (0.00 sec)

mysql> select c1/c2 from t2;
+---------+
| c1 / c2 |
+---------+
| -1.5 |
| 0.5 |
+---------+
2 rows in set (0.01 sec)
```

DIV

运算符说明

`DIV` 运算符用于整数除法。

如果两个运算数值都是非整数类型，则会将运算数值转换为 `DECIMAL`，并在将结果转换为 `BIGINT` 之前使用 `DECIMAL` 算法进行除法。如果结果超出 `BIGINT` 范围，则会发生错误。

语法结构

```
> SELECT value1 DIV value2;
```

```
> SELECT column1 DIV column2... FROM table_name;
```

示例

```
mysql> SELECT 5 DIV 2, -5 DIV 2, 5 DIV -2, -5 DIV -2;
+---------+----------+----------+-----------+
| 5 div 2 | -5 div 2 | 5 div -2 | -5 div -2 |
+---------+----------+----------+-----------+
| 2 | -2 | -2 | 2 |
+---------+----------+----------+-----------+
1 row in set (0.00 sec)
```

```
create table t2(c1 int, c2 int);
insert into t2 values (-3, 2);
insert into t2 values (1, 2);

mysql> select c1 DIV 3 from t2;
+----------+
| c1 div 3 |
+----------+
| -1 |
| 0 |
+----------+
2 rows in set (0.00 sec)

mysql> select c1 DIV c2 from t2;
+-----------+
| c1 div c2 |
+-----------+
| -1 |
| 0 |
+-----------+
2 rows in set (0.00 sec)
```

赋值运算符概述

名称	描述
:=	等于运算符，用于赋值

=

运算符说明

`=` 运算符在以下情况下用于赋值：

- 在 `SET` 语句中，`=` 被视为赋值运算符，即运算符 `=` 左侧的用户变量取其右侧的值。`=` 右边的值可以是文本值、另一个存储值的变量，或者任何产生标量值的合法表达式，也可包括查询的结果 (前提是这个值是标量值)。可以在同一个 `SET` 语句中执行多个赋值。
- 在 `UPDATE` 语句的 `SET` 子句中，`=` 也充当赋值运算符。你可以在一个 `UPDATE` 语句的同一个 `SET` 子句中进行多个赋值。
- 在其他情况下中，`=` 作为比较运算符。

语法结构

```
> SELECT column1, column2, ...
FROM table_name
WHERE columnN = value1;
```

示例

```
create table t1 (a bigint(3), b bigint(5) primary key);
insert into t1 VALUES (1,1),(1,2);
update t1 set a=2 where a=1 limit 1;

mysql> select * from t1;
+------+------+
| a | b |
+------+------+
| 2 | 1 |
| 1 | 2 |
+------+------+
2 rows in set (0.00 sec)
```

二进制运算符

名称	描述
&	位运算符与，按位与
>>	位移运算符右移
<<	位移运算符右移
^	按位异或
[	](bitwise-or.md)
~	一元运算符，二进制取反

位函数和运算符通常与整数数据类型一起使用，不能直接应用于其他数据类型，例如，float，double 等。即，需要与 BIGINT(64 位整数) 参数一起使用并返回 BIGINT 值，因此它们的最大范围为 64 位。非 BIGINT 参数在执行操作之前被转换为 BIGINT，可能整数位数会被截断，例如，10.6496 和 -10.6496 在转换到 INT 类型期间可能会被截断或者被舍入。

位函数和操作符允许二进制字符串类型实参 (BINARY、VARBINARY 和 BLOB 类型) 并返回其同类型的值，这使它们能够接受实参并产生大于 64 位的返回值。非二进制字符串参数则被转换为 BIGINT 类型。

&

运算符说明

`Bitwise AND` 运算符，对每对比特位执行与（AND）操作。只有 a 和 b 都是 1 时，a AND b 才是 1。`Bitwise AND` 返回一个无符号的 64 位整数。

结果类型取决于参数是否被为二进制字符串或数字：

- 当参数为二进制字符串类型，并且其中至少一个不是十六进制 `literal`、位 `literal` 或 `NULL literal` 时，则进行二进制字符串求值计算；否则会进行数值求值计算，并根据需要将参数转换为无符号 64 位整数。
- 二进制字符串求值产生一个与参数长度相同的二进制字符串。如果参数的长度不相等，则会发生 `ER_INVALID_BITWISE_OPERANDS_SIZE` 错误。数值计算产生一个无符号的 64 位整数。

语法结构

```
> SELECT value1 & value2;
```

示例

```
mysql> SELECT 29 & 15;
+---------+
| 29 & 15 |
+---------+
| 13 |
+---------+
1 row in set (0.06 sec)

CREATE TABLE bitwise (a_int_value INT NOT NULL,b_int_value INT NOT NULL);
INSERT bitwise VALUES (170, 75);

mysql> SELECT a_int_value & b_int_value FROM bitwise;
+--------------------------+
| a_int_value & b_int_value |
+--------------------------+
| 10 |
+--------------------------+
1 row in set (0.02 sec)
```

>>

运算符说明

该操作符会将第一个操作数向右移动指定的位数。向右被移出的位被丢弃，左侧用 0 补充。

结果类型取决于参数是否被为二进制字符串或数字：

- 当参数为二进制字符串类型，并且其中至少一个不是十六进制 `literal`、位 `literal` 或 `NULL literal` 时，则进行二进制字符串求值计算；否则会进行数值求值计算，并根据需要将参数转换为无符号 64 位整数。
- 二进制字符串求值产生一个与参数长度相同的二进制字符串。数值计算产生一个无符号的 64 位整数。

无论参数类型如何，移出值末尾的位都会在没有警告的情况下丢失。特别是，如果移位计数大于或等于位参数中的位数，则结果中的所有位均为 0。

语法结构

```
> SELECT value1 >> value2;
```

示例

```
mysql> select 1024 >> 2;
+-----------+
| 1024 >> 2 |
+-----------+
| 256 |
+-----------+
1 row in set (0.01 sec)

mysql> select -5 >> 2;
+---------+
| -5 >> 2 |
+---------+
| -2 |
+---------+
1 row in set (0.01 sec)

mysql> select null >> 2;
+-----------+
| null >> 2 |
+-----------+
| NULL |
+-----------+
1 row in set (0.00 sec)

create table t1(a int, b int unsigned);
insert into t1 values (-1, 1), (-5, 5);

mysql> select a >> 2, b >> 2 from t1;
+--------+--------+
| a >> 2 | b >> 2 |
+--------+--------+
| -1 | 0 |
| -2 | 1 |
+--------+--------+
2 rows in set (0.01 sec)
```

<<

运算符说明

该操作符会将第一个操作数向左移动指定的位数。向左被移出的位被丢弃，右侧用 0 补充。

结果类型取决于参数是否被为二进制字符串或数字：

- 当参数为二进制字符串类型，并且其中至少一个不是十六进制 `literal`、位 `literal` 或 `NULL literal` 时，则进行二进制字符串求值计算；否则会进行数值求值计算，并根据需要将参数转换为无符号 64 位整数。
- 二进制字符串求值产生一个与参数长度相同的二进制字符串。数值计算产生一个无符号的 64 位整数。

无论参数类型如何，移出值末尾的位都会在没有警告的情况下丢失。特别是，如果移位计数大于或等于位参数中的位数，则结果中的所有位均为 0。

语法结构

```
> SELECT value1 << value2;
```

示例

```
mysql> SELECT 1 << 2;
+--------+
| 1 << 2 |
+--------+
| 4 |
+--------+
1 row in set (0.01 sec)

mysql> select -1 << 2;
+---------+
| -1 << 2 |
+---------+
| -4 |
+---------+
1 row in set (0.01 sec)

mysql> select null << 2;
+-----------+
| null << 2 |
+-----------+
| NULL |
+-----------+
1 row in set (0.01 sec)

create table t1(a int, b int unsigned);
insert into t1 values (-1, 1), (-5, 5);

mysql> select a << 2, b << 2 from t1;
+--------+--------+
| a << 2 | b << 2 |
+--------+--------+
| -4 | 4 |
| -20 | 20 |
+--------+--------+
2 rows in set (0.01 sec)
```

运算符说明

按位异或。对每一对比特位执行异或（XOR）操作。当 a 和 b 不相同时，a XOR b 的结果为 1。

结果类型取决于参数是否被为二进制字符串或数字：

- 当参数为二进制字符串类型，并且其中至少一个不是十六进制 `literal`、位 `literal` 或 `NULL literal` 时，则进行二进制字符串求值计算；否则会进行数值求值计算，并根据需要将参数转换为无符号 64 位整数。
- 二进制字符串求值产生一个与参数长度相同的二进制字符串。如果参数的长度不相等，则会发生 `ER_INVALID_BITWISE_OPERANDS_SIZE` 错误。数值计算产生一个无符号的 64 位整数。

语法结构

```
> SELECT value1 ^ value2;
```

示例

```
mysql> SELECT 1 ^ 1;
+-------+
| 1 ^ 1 |
+-------+
| 0 |
+-------+
1 row in set (0.00 sec)

mysql> SELECT 1 ^ 0;
+-------+
| 1 ^ 0 |
+-------+
| 1 |
+-------+
1 row in set (0.01 sec)

mysql> SELECT 11 ^ 3;
+--------+
| 11 ^ 3 |
+--------+
| 8 |
+--------+
1 row in set (0.01 sec)

mysql> select null ^ 2;
+----------+
| null ^ 2 |
+----------+
| NULL |
+----------+
1 row in set (0.01 sec)

create table t1(a int, b int unsigned);
insert into t1 values (-1, 1), (-5, 5);

mysql> select a ^ 2, b ^ 2 from t1;
+-------+-------+
| a ^ 2 | b ^ 2 |
+-------+-------+
| -3 | 3 |
| -7 | 7 |
```

```
+-------+-------+
2 rows in set (0.01 sec)
```

|

运算符说明

位运算符或，按位或。对每一对比特位执行或（OR）操作。如果 a 或 b 为 1，则 a OR b 结果为 1。

结果类型取决于参数是否被为二进制字符串或数字：

- 当参数为二进制字符串类型，并且其中至少一个不是十六进制 `literal`、位 `literal` 或 `NULL literal` 时，则进行二进制字符串求值计算；否则会进行数值求值计算，并根据需要将参数转换为无符号 64 位整数。
- 二进制字符串求值产生一个与参数长度相同的二进制字符串。如果参数的长度不相等，则会发生 `ER_INVALID_BITWISE_OPERANDS_SIZE` 错误。数值计算产生一个无符号的 64 位整数。

语法结构

```
> SELECT value1 | value2;
```

示例

```
mysql> SELECT 29 | 15;
+---------+
| 29 | 15 |
+---------+
| 31 |
+---------+
1 row in set (0.01 sec)

mysql> select null | 2;
+----------+
| null | 2 |
+----------+
| NULL |
+----------+
1 row in set (0.01 sec)

mysql> select null | 2;
+----------+
| null | 2 |
+----------+
| NULL |
+----------+
1 row in set (0.01 sec)

create table t1(a int, b int unsigned);
insert into t1 values (-1, 1), (-5, 5);

mysql> select a | 2, b | 2 from t1;
+-------+-------+
| a | 2 | b | 2 |
+-------+-------+
| -1 | 3 |
| -5 | 7 |
+-------+-------+
```

~

运算符说明

一元运算符，二进制取反。对每一个比特位执行非（NOT）操作。NOT a 结果为 a 的反转（即反码）。

结果类型取决于参数是否被为二进制字符串或数字：

- 当参数为二进制字符串类型，并且其中至少一个不是十六进制 `literal`、位 `literal` 或 `NULL literal` 时，则进行二进制字符串求值计算；否则会进行数值求值计算，并根据需要将参数转换为无符号 64 位整数。
- 二进制字符串求值产生一个与参数长度相同的二进制字符串。数值计算产生一个无符号的 64 位整数。

语法结构

```
> SELECT value1 ~ value2;
```

示例

```
mysql> select ~-5;
+--------+
| ~ (-5) |
+--------+
| 4 |
+--------+
1 row in set (0.00 sec)

mysql> select ~null;
+-------+
| ~null |
+-------+
| NULL |
+-------+
1 row in set (0.00 sec)

mysql> select ~a, ~b from t1;
+------+----------------------+
| ~a | ~b |
+------+----------------------+
| 0 | 18446744073709551614 |
| 4 | 18446744073709551610 |
+------+----------------------+
2 rows in set (0.00 sec)
```

强制转换函数和运算符概述

名称	描述
CAST()	将值转换为特定类型，用于小数转数值和字符型
CONVERT()	将值转换为特定类型，用于日期和时间值、小数之间进行转换

CAST

函数说明

`CAST()` 函数可以将任何类型的一个值转化为另一个特定类型。

语法结构

```
> CAST(value AS datatype)
```

相关参数

参数	说明
value	必要参数，待转化的值
datatype	必要参数，目标数据类型

目前，`cast` 可以进行如下转换：

- 数值类型之间转换，主要包括 SIGNED，UNSIGNED，FLOAT，DOUBLE 类型
- 数值类型向字符 CHAR 类型转换
- 格式为数值的字符类型向数值类型转换（负数需要转换为 SIGNED）

详细的数据类型转换规则，请参见[数据类型转换](#)。

示例

```
drop table if exists t1;
CREATE TABLE t1 (a int,b float,c char(1),d varchar(15));
INSERT INTO t1 VALUES (1,1.5,'1','-2');

mysql> SELECT CAST(a AS FLOAT) a_cast,CAST(b AS UNSIGNED) b_cast,CAST(c
AS SIGNED) c_cast, CAST(d AS SIGNED) d_cast from t1;
+--------+--------+--------+--------+
| a_cast | b_cast | c_cast | d_cast |
+--------+--------+--------+--------+
| 1.0000 | 1 | 1 | -2 |
+--------+--------+--------+--------+

mysql> SELECT CAST(a AS CHAR) a_cast, CAST(b AS CHAR) b_cast,CAST(c AS
DOUBLE) c_cast, CAST(d AS FLOAT) d_cast from t1;
+--------+--------+--------+---------+
| a_cast | b_cast | c_cast | d_cast |
+--------+--------+--------+---------+
| 1 | 1.5 | 1.0000 | -2.0000 |
+--------+--------+--------+---------+
```

限制

- 非数值的字符类型无法转化为数值类型
- 日期格式的数值类型、字符类型无法转化为 Date 类型
- Date，Datetime 类型无法转化为字符类型
- Date 与 Datetime 暂不能互相转化

CONVERT

函数说明

`CONVERT()` 函数将一个值转换为指定的数据类型或字符集。

语法结构

```
> CONVERT(value, type)
```

或：

```
> CONVERT(value USING charset)
```

相关参数

参数	说明
value	必要参数，待转化的值
datatype	必要参数，目标数据类型
charset	必要参数，目标字符集

目前，`convert` 可以进行如下转换：

- 数值类型之间转换，主要包括 SIGNED，UNSIGNED，FLOAT，DOUBLE 类型
- 数值类型向字符 CHAR 类型转换
- 格式为数值的字符类型向数值类型转换（负数需要转换为 SIGNED）

示例

```
mysql> select convert(150,char);
+-------------------+
| cast(150 as char) |
+-------------------+
| 150 |
+-------------------+
1 row in set (0.01 sec)
```

```
CREATE TABLE t1(a tinyint);
INSERT INTO t1 VALUES (127);

mysql> SELECT 1 FROM
-> (SELECT CONVERT(t2.a USING UTF8) FROM t1, t1 t2 LIMIT 1) AS s LIMIT 1;
+------+
| 1 |
+------+
| 1 |
+------+
1 row in set (0.00 sec)
```

限制

- 非数值的字符类型无法转化为数值类型
- 日期格式的数值类型、字符类型无法转化为 Date 类型
- Date，Datetime 类型无法转化为字符类型
- Date 与 Datetime 暂不能互相转化

比较函数和运算符概述

名称	描述
>	大于
>=	大于等于
<	小于
<>,!=	不等于
<=	小于等于
=	等于
BETWEEN ... AND ...	在两值之间
IN()	在集合中
IS	测试值是否是布尔值，若是布尔值，则返回“true”
IS NOT	测试值是否是布尔值，IS 的否定用法
IS NOT NULL	不为空
IS NULL	为空
LIKE	模糊匹配
NOT BETWEEN ... AND ...	不在两值之间
NOT LIKE	N 模糊匹配，Like 的否定用法
COALESCE	返回第一个非空值

比较运算的结果为 `TRUE`、`FALSE` 或 `NULL`。这些运算对数字和字符串均有效。字符串可以自动转换为数字，数字根据需要自动转换为字符串。

以下比较运算符不仅可以用于比较标量运算数，也可以用于比较行运算数：

```
= > < >= <= <> !=
```

>

运算符说明

`>` 运算符用于比较运算。当 `>` 左边运算数值大于 `>` 右侧运算数值时，`>` 运算符返回结果为 `true`。

语法结构

```
> SELECT x > y;
```

示例

```
mysql> SELECT 2 > 2;
+-------+
| 2 > 2 |
+-------+
| false |
+-------+
1 row in set (0.00 sec)
```

```
create table t1 (spID smallint,userID bigint,score int);
insert into t1 values (1,1,1);
insert into t1 values (2,2,2);
insert into t1 values (2,1,4);
insert into t1 values (3,3,3);
insert into t1 values (1,1,5);
insert into t1 values (4,6,10);
insert into t1 values (5,11,99);

mysql> select spID,userID,score from t1 where spID>(userID-1);
+------+--------+-------+
| spid | userid | score |
+------+--------+-------+
| 1 | 1 | 1 |
| 2 | 2 | 2 |
| 2 | 1 | 4 |
| 3 | 3 | 3 |
| 1 | 1 | 5 |
+------+--------+-------+
5 rows in set (0.01 sec)
```

>=

运算符说明

`>=` 运算符用于比较运算。当 `>=` 左边运算数值大于或等于 `>=` 右侧运算数值时，`>=` 运算符返回结果为 `true`。

语法结构

```
> SELECT x >= y;
```

示例

```
mysql> SELECT 2 >= 2;
+--------+
| 2 >= 2 |
+--------+
| true |
+--------+
1 row in set (0.01 sec)
```

```
create table t1 (spID smallint,userID bigint,score int);
insert into t1 values (1,1,1);
insert into t1 values (2,2,2);
insert into t1 values (2,1,4);
insert into t1 values (3,3,3);
insert into t1 values (1,1,5);
insert into t1 values (4,6,10);
insert into t1 values (5,11,99);

mysql> select userID,spID,score from t1 where spID>=userID*score;
+--------+------+-------+
| userid | spid | score |
+--------+------+-------+
| 1 | 1 | 1 |
+--------+------+-------+
1 row in set (0.01 sec)
```

<

运算符说明

`<` 运算符用于比较运算。当 `<` 左边运算数值小于 `<` 右侧运算数值时，`<` 运算符返回结果为 `true`。

语法结构

```
> SELECT x < y;
```

示例

```
mysql> SELECT 2 < 2;
+-------+
| 2 < 2 |
+-------+
| false |
+-------+
1 row in set (0.00 sec)
```

```
create table t1 (spID smallint,userID bigint,score int);
insert into t1 values (1,1,1);
insert into t1 values (2,2,2);
insert into t1 values (2,1,4);
insert into t1 values (3,3,3);
insert into t1 values (1,1,5);
insert into t1 values (4,6,10);
insert into t1 values (5,11,99);

mysql> select spID,userID,score from t1 where (userID-1)<spID;
+------+--------+-------+
| spid | userid | score |
+------+--------+-------+
| 1 | 1 | 1 |
| 2 | 2 | 2 |
| 2 | 1 | 4 |
| 3 | 3 | 3 |
| 1 | 1 | 5 |
+------+--------+-------+
5 rows in set (0.00 sec)
mysql> select spID,userID,score from t1 where spID<(userID-1);
+------+--------+-------+
| spid | userid | score |
+------+--------+-------+
| 4 | 6 | 10 |
| 5 | 11 | 99 |
+------+--------+-------+
2 rows in set (0.00 sec)
```

<>,!=

运算符说明

The `<>,!=` operator returns `true` only if the left-hand operand is not equal to the right-hand operand.

语法结构

```
> SELECT x <> y;
```

示例

```
mysql> SELECT 2 <> 2;
+--------+
| 2 != 2 |
+--------+
| false |
+--------+
1 row in set (0.01 sec)
```

```
create table t1 (spID smallint,userID bigint,score int);
insert into t1 values (1,1,1);
insert into t1 values (2,2,2);
insert into t1 values (2,1,4);
insert into t1 values (3,3,3);
insert into t1 values (1,1,5);
insert into t1 values (4,6,10);
insert into t1 values (5,11,99);

mysql> select userID,spID,score from t1 where userID=spID and
userID<>score;
+--------+------+-------+
| userid | spid | score |
+--------+------+-------+
| 1 | 1 | 5 |
+--------+------+-------+
1 row in set (0.00 sec)
```

<=

运算符说明

`<=` 运算符用于比较运算。当 `<=` 左边运算数值小于或等于 `<=` 右侧运算数值时，`<=` 运算符返回结果为 `true`。

语法结构

```
> SELECT x <= y;
```

示例

```
mysql> SELECT 2 <= 2;
+--------+
| 2 <= 2 |
+--------+
| true |
+--------+
1 row in set (0.00 sec)
```

```
create table t1 (spID smallint,userID bigint,score int);
insert into t1 values (1,1,1);
insert into t1 values (2,2,2);
insert into t1 values (2,1,4);
insert into t1 values (3,3,3);
insert into t1 values (1,1,5);
insert into t1 values (4,6,10);
insert into t1 values (5,11,99);

mysql> select userID,score,spID from t1 where userID<=score/spID;
+--------+-------+------+
| userid | score | spid |
+--------+-------+------+
| 1 | 1 | 1 |
| 1 | 4 | 2 |
| 1 | 5 | 1 |
| 11 | 99 | 5 |
+--------+-------+------+
4 rows in set (0.00 sec)
```

=

运算符说明

`=` 运算符用于比较运算。当 `=` 左边运算数值等于 `=` 右侧运算数值时，`=` 运算符的结果返回 `true`。

语法结构

```
> SELECT x = y;
```

示例

```
mysql> SELECT 2 = 2;
+-------+
| 2 = 2 |
+-------+
| true |
+-------+
1 row in set (0.01 sec)
```

```
create table t1 (spID smallint,userID bigint,score int);
insert into t1 values (1,1,1);
insert into t1 values (2,2,2);
insert into t1 values (2,1,4);
insert into t1 values (3,3,3);
insert into t1 values (1,1,5);
insert into t1 values (4,6,10);
insert into t1 values (5,11,99);

mysql> select userID,spID,score from t1 where userID=spID and
userID=score;
+--------+------+-------+
| userid | spid | score |
+--------+------+-------+
| 1 | 1 | 1 |
| 2 | 2 | 2 |
| 3 | 3 | 3 |
+--------+------+-------+
3 rows in set (0.00 sec)
```

BETWEEN ... AND ...

语法说明

`BETWEEN ... AND ...` 操作符选取介于两个值之间的数据范围内的值。这些值可以是数值、文本或者日期。如果取值介于两值之间，那么返回 `true`，否则返回结果为 `false`。

语法结构

```
> expr BETWEEN min AND max
```

示例

```
mysql> SELECT 2 BETWEEN 1 AND 3, 2 BETWEEN 3 and 1;
+------------------+------------------+
| 2 between 1 and 3 | 2 between 3 and 1 |
+------------------+------------------+
| true | false |
+------------------+------------------+
1 row in set (0.01 sec)
```

```
create table t (id bigint unsigned, b int);
insert into t values(8894754949779693574,1);
insert into t values(8894754949779693579,2);
insert into t values(17790886498483827171,3);

mysql> select count(*) from t where id>=8894754949779693574 and id
=17790886498483827171 order by 1 asc;
+----------+
| count(*) |
+----------+
| 0 |
+----------+

mysql> select count(*) from t where id between 8894754949779693574 and
17790886498483827171;
+----------+
| count(*) |
+----------+
| 0 |
+----------+
1 row in set (0.01 sec)
```

IN

语法说明

`IN` 运算符可以在 `WHERE` 语句中指定特定的多个值，本质上是多个 `OR` 条件的简写。

语法结构

```
> SELECT column1, column2, ...
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

示例

```
create table t2(a int,b varchar(5),c float, d date, e datetime);
insert into t2 values(1,'a',1.001,'2022-02-08','2022-02-08 12:00:00');
insert into t2 values(2,'b',2.001,'2022-02-09','2022-02-09 12:00:00');
insert into t2 values(1,'c',3.001,'2022-02-10','2022-02-10 12:00:00');
insert into t2 values(4,'d',4.001,'2022-02-11','2022-02-11 12:00:00');

mysql> select * from t2 where a in (2,4);
a b c d e
2 b 2.0010 2022-02-09 2022-02-09 12:00:00
4 d 4.0010 2022-02-11 2022-02-11 12:00:00

mysql> select * from t2 where a not in (2,4);
a b c d e
1 a 1.0010 2022-02-08 2022-02-08 12:00:00
1 c 3.0010 2022-02-10 2022-02-10 12:00:00

mysql> select * from t2 where b not in ('e',"f");
a b c d e
1 a 1.0010 2022-02-08 2022-02-08 12:00:00
2 b 2.0010 2022-02-09 2022-02-09 12:00:00
1 c 3.0010 2022-02-10 2022-02-10 12:00:00
4 d 4.0010 2022-02-11 2022-02-11 12:00:00

mysql> select * from t2 where e not in ('2022-02-09 12:00:00') and a in
(4,5);
a b c d e
4 d 4.0010 2022-02-11 2022-02-11 12:00:00
```

限制

- 目前，`IN` 左侧只支持常数列表。
- `IN` 左侧只支持单列数据，不支持多列组成的元组。
- `IN` 目前未对 `NULL` 值进行很好地处理，右侧不能使用 `NULL` 值。

IS

语法说明

`IS` 运算符用于测试数值是否为布尔值，若是布尔值，则返回结果为 `true`。其中 `boolean_value` 可以为 `TRUE`、`FALSE` 或 `UNKNOWN`。

语法结构

```
> IS boolean_value
```

示例

```
mysql> SELECT 1 IS TRUE, 0 IS FALSE, NULL IS UNKNOWN;
+----------+-----------+---------+
| 1 = true | 0 = false | null = |
+----------+-----------+---------+
| true | true | NULL |
+----------+-----------+---------+
1 row in set (0.01 sec)
```

```
create table t1 (a boolean,b bool);
insert into t1 values (0,1),(true,false),(true,1),(0,false),(NULL,NULL);

mysql> select * from t1;
+-------+-------+
| a | b |
+-------+-------+
| false | true |
| true | false |
| true | true |
| false | false |
| NULL | NULL |
+-------+-------+
mysql> select * from t1 where a<=b and a is not NULL;
+-------+-------+
| a | b |
+-------+-------+
| false | true |
| true | true |
| false | false |
+-------+-------+
3 rows in set (0.01 sec)
```

IS NOT

语法说明

`IS NOT` 运算符用于测试数值是否为布尔值，若不是布尔值，则返回结果为 `true`。其中 `boolean_value` 可以为 `TRUE`、`FALSE` 或 `UNKNOWN`。

语法结构

```
> IS NOT boolean_value
```

示例

```
mysql> SELECT 1 IS NOT TRUE, 0 IS NOT FALSE, NULL IS NOT UNKNOWN;
+-----------+------------+----------+
| 1 != true | 0 != false | null != |
+-----------+------------+----------+
| false | false | NULL |
+-----------+------------+----------+
1 row in set (0.01 sec)
```

IS NOT NULL

语法说明

`IS NOT NULL` 运算符用于判断列的值是否为空。如果值不为空，即不为 `NULL`，则返回 `true`，否则返回 `false`。它可以用于 `SELECT`、`INSERT`、`UPDATE` 或 `DELETE` 语句。

语法结构

```
> expression IS NOT NULL
```

示例

```
mysql> SELECT 1 IS NOT NULL, 0 IS NOT NULL, NULL IS NOT NULL;
+--------------+--------------+-----------------+
| 1 is not null | 0 is not null | null is not null |
+--------------+--------------+-----------------+
| true | true | false |
+--------------+--------------+-----------------+
1 row in set (0.01 sec)
```

```
create table t1 (a boolean,b bool);
insert into t1 values (0,1),(true,false),(true,1),(0,false),(NULL,NULL);

mysql> select * from t1;
+-------+-------+
| a | b |
+-------+-------+
| false | true |
| true | false |
| true | true |
| false | false |
| NULL | NULL |
+-------+-------+
mysql> select * from t1 where b is NOT NULL;
+-------+-------+
| a | b |
+-------+-------+
| false | true |
| true | false |
| true | true |
| false | false |
+-------+-------+
4 rows in set (0.01 sec)
```

IS NULL

语法说明

`IS NULL` 运算符用于判断列的值是否为空。如果值为空，即为 `NULL`，则返回 `true`，否则返回 `false`。它可以用于 `SELECT`、`INSERT`、`UPDATE` 或 `DELETE` 语句。

语法结构

```
> expression IS NULL
```

示例

```
mysql> SELECT 1 IS NULL, 0 IS NULL, NULL IS NULL;
+-----------+-----------+--------------+
| 1 is null | 0 is null | null is null |
+-----------+-----------+--------------+
| false | false | true |
+-----------+-----------+--------------+
1 row in set (0.01 sec)
```

```
create table t1 (a boolean,b bool);
insert into t1 values (0,1),(true,false),(true,1),(0,false),(NULL,NULL);

mysql> select * from t1;
+-------+-------+
| a | b |
+-------+-------+
| false | true |
| true | false |
| true | true |
| false | false |
| NULL | NULL |
+-------+-------+
mysql> select * from t1 where a IS NULL;
+------+------+
| a | b |
+------+------+
| NULL | NULL |
+------+------+
1 row in set (0.01 sec)
```

LIKE

语法说明

LIKE 操作符用于在 WHERE 子句中搜索列中的指定模式。

有两个通配符经常与 LIKE 操作符一起使用：

- 百分号 (%) 代表了 0、1 或多个字符。
- 下划线 (_) 代表单个字符。

语法结构

```
> SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;
```

示例

```
mysql> SELECT * FROM Customers
WHERE CustomerName LIKE 'a%'; //The following SQL statement selects all
customers with a CustomerName starting with "a"

mysql> SELECT * FROM Customers
WHERE CustomerName LIKE '%a'; //The following SQL statement selects all
customers with a CustomerName ending with "a"

mysql> SELECT * FROM Customers
WHERE CustomerName LIKE '%or%'; //The following SQL statement selects all
customers with a CustomerName that have "or" in any position

mysql> SELECT * FROM Customers
WHERE CustomerName LIKE '_r%'; //The following SQL statement selects all
customers with a CustomerName that have "r" in the second position

mysql> SELECT * FROM Customers
WHERE CustomerName LIKE 'a__%'; //The following SQL statement selects all
customers with a CustomerName that starts with "a" and are at least 3
characters in length

mysql> SELECT * FROM Customers
WHERE ContactName LIKE 'a%o'; //The following SQL statement selects all
customers with a ContactName that starts with "a" and ends with "o"

mysql> SELECT * FROM Customers
WHERE CustomerName NOT LIKE 'a%'; //The following SQL statement selects
all customers with a CustomerName that does NOT start with "a"
```

NOT BETWEEN ... AND ...

语法说明

`NOT BETWEEN ... AND ...` 操作符选取介于两个值之间的数据范围外的值。这些值可以是数值、文本或者日期。如果取值介于两值之间，那么返回 `false`，否则返回结果为 `true`。

语法结构

```
> expr NOT BETWEEN min AND max
```

示例

```
mysql> SELECT 2 NOT BETWEEN 1 AND 3, 2 NOT BETWEEN 3 and 1;
+----------------------+----------------------+
| 2 not between 1 and 3 | 2 not between 3 and 1 |
+----------------------+----------------------+
| false | true |
+----------------------+----------------------+
1 row in set (0.00 sec)
```

```
create table t (id bigint unsigned, b int);
insert into t values(8894754949779693574,1);
insert into t values(8894754949779693579,2);
insert into t values(17790886498483827171,3);

mysql> select count(*) from t where id>=8894754949779693574 and id
=17790886498483827171 order by 1 asc;
+----------+
| count(*) |
+----------+
| 0 |
+----------+
mysql> select count(*) from t where id not between 8894754949779693574
and 17790886498483827171;
+----------+
| count(*) |
+----------+
| 3 |
+----------+
1 row in set (0.00 sec)
```

NOT LIKE

语法说明

`NOT LIKE` 操作符用于在 `WHERE` 从句中搜索列中的指定模式，是 `LIKE` 的否定用法。

有两个通配符经常与 `LIKE` 操作符一起使用：

- 百分号 (%) 代表了 0、1 或多个字符。
- 下划线 (_) 代表单个字符。

语法结构

```
> SELECT column1, column2, ...
FROM table_name
WHERE columnN NOT LIKE pattern;
```

示例

```
create table t1 (a char(10));
insert into t1 values('abcdef');
insert into t1 values('_bcdef');
insert into t1 values('a_cdef');
insert into t1 values('ab_def');
insert into t1 values('abc_ef');
insert into t1 values('abcd_f');
insert into t1 values('abcde_');

mysql> select * from t1 where a not like 'a%';
+--------+
| a |
+--------+
| _bcdef |
+--------+
mysql> select * from t1 where a not like "%d_\_";
+--------+
| a |
+--------+
| abc_ef |
+--------+
1 row in set (0.01 sec)
```

COALESCE()

函数说明

`COALESCE (expression_1, expression_2, ...,expression_n)` 依次参考各参数表达式，遇到非 `null` 值即停止并返回该值。如果所有的表达式都是空值，最终将返回一个空值。

使用 `COALESCE` 在于大部分包含空值的表达式最终将返回空值。

语法

```
> COALESCE(value1, value2, ...., value_n)
```

参数释义

Arguments	Description
value1, value2, value_n	Required. The values to test

示例

- 示例：计算

```
mysql> SELECT COALESCE(1)+COALESCE(1);
+--------------------------+
| coalesce(1) + coalesce(1) |
+--------------------------+
| 2 |
+--------------------------+
```

- 示例：比较运算

```
drop table if exists t2;
create table t2(a float, b datetime);
insert into t2 values (12.345, '2022-02-20 10:10:10.999999');
insert into t2 values (3.45646, NULL);
insert into t2 values(NULL, '2023-04-03 22:10:29.999999');
insert into t2 values (NULL, NULL);

mysql> select * from t2;
+---------+---------------------+
| a | b |
+---------+---------------------+
| 12.345 | 2022-02-20 10:10:11 |
| 3.45646 | NULL |
| NULL | 2023-04-03 22:10:30 |
| NULL | NULL |
+---------+---------------------+
mysql> select coalesce(a, 1.0) from t2;
+--------------------+
| coalesce(a, 1.0) |
+--------------------+
| 12.345000267028809 |
| 3.4564599990844727 |
| 1 |
| 1 |
+--------------------+
mysql> select coalesce(a, 1) from t2;
+--------------------+
| coalesce(a, 1) |
+--------------------+
| 12.345000267028809 |
| 3.4564599990844727 |
| 1 |
| 1 |
+--------------------+
mysql> select coalesce(b, 2022-01-01) from t2;
+---------------------------+
| coalesce(b, 2022 - 1 - 1) |
+---------------------------+
| 2022-02-20 10:10:11 |
| |
| 2023-04-03 22:10:30 |
| |
+---------------------------+
```

- 示例：含有 `ORDER BY` 子句

```
CREATE TABLE t1 ( a INTEGER, b varchar(255) );
INSERT INTO t1 VALUES (1,'z');
INSERT INTO t1 VALUES (2,'y');
INSERT INTO t1 VALUES (3,'x');

mysql> SELECT MIN(b) AS min_b FROM t1 GROUP BY a ORDER BY
COALESCE(MIN(b), 'a');
+-------+
| min_b |
+-------+
| x |
| y |
| z |
+-------+

mysql> SELECT MIN(b) AS min_b FROM t1 GROUP BY a ORDER BY COALESCE(min_b,
'a');
+-------+
| min_b |
+-------+
| x |
| y |
| z |
+-------+

mysql> SELECT MIN(b) AS min_b FROM t1 GROUP BY a ORDER BY
COALESCE(MIN(b), 'a') DESC;
+-------+
| min_b |
+-------+
| z |
| y |
| x |
+-------+
```

- 示例：含有 `Case When` 子句

```
mysql> select if(1, cast(1111111111111111111 as unsigned), 1) i,case when
1 then cast(1111111111111111111 as unsigned) else 1 end c,
coalesce(cast(1111111111111111111 as unsigned), 1) co;
+---------------------+---------------------+---------------------+
| i | c | co |
+---------------------+---------------------+---------------------+
| 1111111111111111111 | 1111111111111111111 | 1111111111111111111 |
+---------------------+---------------------+---------------------+
```

- 示例：`IN Subquery`

```
CREATE TABLE ot (col_int_nokey int(11), col_varchar_nokey varchar(1));
INSERT INTO ot VALUES (1,'x');
CREATE TABLE it (col_int_key int(11), col_varchar_key varchar(1));
INSERT INTO it VALUES (NULL,'x'), (NULL,'f');

mysql> SELECT col_int_nokey FROM ot WHERE col_varchar_nokey IN(SELECT
col_varchar_key FROM it WHERE coalesce(col_int_nokey, 1) );
+---------------+
| col_int_nokey |
+---------------+
| 1 |
+---------------+
```

- 示例：含有 `WHERE` 子句

```
CREATE TABLE ot1(a INT);
CREATE TABLE ot2(a INT);
CREATE TABLE ot3(a INT);
CREATE TABLE it1(a INT);
CREATE TABLE it2(a INT);
CREATE TABLE it3(a INT);
INSERT INTO ot1 VALUES(0),(1),(2),(3),(4),(5),(6),(7);
INSERT INTO ot2 VALUES(0),(2),(4),(6);
INSERT INTO ot3 VALUES(0),(3),(6);
INSERT INTO it1 VALUES(0),(1),(2),(3),(4),(5),(6),(7);
INSERT INTO it2 VALUES(0),(2),(4),(6);
INSERT INTO it3 VALUES(0),(3),(6);

mysql> SELECT * FROM ot1 LEFT JOIN ot2 ON ot1.a=ot2.a WHERE
COALESCE(ot2.a,0) IN (SELECT a FROM it3);
+------+------+
| a | a |
+------+------+
| 0 | 0 |
| 1 | NULL |
| 3 | NULL |
| 5 | NULL |
| 6 | 6 |
| 7 | NULL |
+------+------+
```

- 示例：`HAVING`

```
drop table if exists t1;
create table t1(a datetime);
INSERT INTO t1 VALUES (NULL), ('2001-01-01 00:00:00.12'), ('2002-01-01
00:00:00.567');

mysql> select a from t1 group by a having COALESCE(a)<"2002-01-01";
+---------------------+
| a |
+---------------------+
| 2001-01-01 00:00:00 |
+---------------------+
```

- 示例：`ON CONDITION`

```
drop table if exists t1;
drop table if exists t2;
create table t1(a INT, b varchar(255));
create table t2(a INT, b varchar(255));
insert into t1 values(1, "你好"), (3, "再见");
insert into t2 values(2, "日期时间"), (4, "明天");
> SELECT t1.a, t2.a FROM t1 JOIN t2 ON (length(COALESCE(t1.b)) =
length(COALESCE(t2.b)));
+------+------+
| a | a |
+------+------+
| 1 | 4 |
| 3 | 4 |
+------+------+
```

控制流函数概述

名称	描述
CASE	Case 运算符
IF()	If/else 语句

CASE WHEN

语法说明

`CASE WHEN` 语句用于计算条件列表并返回多个可能结果表达式之一，`CASE WHEN` 可以比较等于、范围的条件。遇到第一个满足条件的即返回，不再往下比较，如果没有满足的条件则返回 `else` 里的结果，如果没有 `else` 则返回 `NULL`。

CASE 有两种格式，两种格式都支持可选的 `ELSE` 参数。：

- 一个简单的 `CASE` 函数将一个表达式与一组简单表达式进行比较以确定结果。
- `CASE` 搜索函数计算一组布尔表达式来确定结果。

语法结构

- **语法结构 1**：

```
CASE value WHEN compare_value THEN result [WHEN compare_value THEN result
...] [ELSE result] END
```

这里的 `CASE` 语法返回的是第一个 `value=compare_value` 为 `true` 的分支的结果。

- **语法结构 2**：

```
CASE WHEN condition THEN result [WHEN condition THEN result ...] [ELSE
result] END
```

这里的 `CASE` 语法返回的是第一个 `condition` 为 `true` 的分支的结果。

如果没有一个 `value=compare_value` 或者 `condition` 为 `true`，那么就会返回 `ELSE` 对应的结果，如果没有 `ELSE` 分支，那么返回 `NULL`。

> **“<font size=4>note</font>”**
>
> <font size=3> `CASE` 语句不能有 `ELSE NULL` 从句, 并且 `CASE` 语句必须以 `END CASE` 结尾。 </font>

示例

```
mysql> SELECT CASE WHEN 1>0 THEN 'true' ELSE 'false' END;
+----------------------------------------+
| case when 1 > 0 then true else false end |
+----------------------------------------+
| true |
+----------------------------------------+
1 row in set (0.00 sec)
```

```
CREATE TABLE t1 (a INT, b INT);
Query OK, 0 rows affected (0.01 sec)

INSERT INTO t1 VALUES (1,1),(2,1),(3,2),(4,2),(5,3),(6,3);
Query OK, 6 rows affected (0.01 sec)

mysql> SELECT CASE WHEN AVG(a)>=0 THEN 'Positive' ELSE 'Negative' END
FROM t1 GROUP BY b;
+-------------------------------------------------------+
| case when avg(a) >= 0 then Positive else Negative end |
+-------------------------------------------------------+
| Positive |
| Positive |
| Positive |
+-------------------------------------------------------+
3 rows in set (0.00 sec)
```

IF

语法说明

`IF()` 既可以作为表达式用，也可在存储过程中作为流程控制语句使用。

语法结构

```
> IF(expr1,expr2,expr3)
```

- 如果 expr1 为 `TRUE` (expr1 <> 0 and expr1 IS NOT NULL)，则结果返回 expr2；否则，返回 expr3。

— 如果 expr2 或 expr3 中只有一个显式为 `NULL`，则 `If()` 函数的结果类型为非 NULL 表达式的类型。

- `IF()` 的默认返回类型 (存储到临时表时可能会有影响) 计算如下：
  - 如果 expr2 或 expr3 生成的是字符串，则结果为字符串。
  - 如果 expr2 和 expr3 都是字符串，如果字符串中有一个是区分大小写的，则结果也是区分大小写的。
  - 如果 expr2 或 expr3 生成的是浮点值，则结果为浮点值。

示例

```
mysql> SELECT IF(1>2,2,3);
+-----------------+
| if(1 > 2, 2, 3) |
+-----------------+
| 3 |
+-----------------+
1 row in set (0.01 sec)
mysql> SELECT IF(1<2,'yes','no');
+--------------------+
| if(1 < 2, yes, no) |
+--------------------+
| yes |
+--------------------+
1 row in set (0.00 sec)
```

```
CREATE TABLE t1 (st varchar(255) NOT NULL, u int(11) NOT NULL);
INSERT INTO t1 VALUES ('a',1),('A',1),('aa',1),('AA',1),('a',1),
('aaa',0),('BBB',0);

mysql> select if(u=1,st,st) s from t1 order by s;
+------+
| s |
+------+
| A |
| AA |
| BBB |
| a |
| a |
| aa |
| aaa |
+------+
7 rows in set (0.00 sec)

mysql> select if(u=1,st,st) s from t1 where st like "%a%" order by s;
+------+
| s |
+------+
| a |
| a |
| aa |
| aaa |
+------+
4 rows in set (0.00 sec)
```

限制

函数 `IF` 暂不支持参数 `BIGINT` 和 `VARCHAR`。

逻辑运算符概述

名称	描述
AND,&&	逻辑与
NOT,!	逻辑非
OR	逻辑或
XOR	逻辑异或

AND,&&

运算符说明

`AND,&&` 逻辑运算符用作于*逻辑与*运算。如果所有操作数都非零且不为 `NULL`，则返回结果为 `true`；如果一个或多个操作数为 0，则返回结果为 `false`；如果一个或多个操作数非零且为 `NULL`，则返回 `NULL`。

语法结构

```
> SELECT column_1 AND column_2 FROM table_name;
```

示例

```
mysql> select 1 and 1;
+---------+
| 1 and 1 |
+---------+
| true |
+---------+
mysql> select 1 and 0;
+---------+
| 1 and 0 |
+---------+
| false |
+---------+
mysql> select 1 and null;
+------------+
| 1 and null |
+------------+
| NULL |
+------------+
mysql> select null and 0;
+------------+
| null and 0 |
+------------+
| false |
+------------+
1 row in set (0.01 sec)
```

```
create table t1 (a boolean,b bool);
insert into t1 values (0,1),(true,false),(true,1),(0,false),(NULL,NULL);
mysql> select a and b from t1;
+---------+
| a and b |
+---------+
| false |
| false |
| true |
| false |
| NULL |
+---------+
5 rows in set (0.00 sec)
```

NOT,!

运算符说明

`NOT,!` 逻辑运算符用作于*逻辑非*运算。如果操作数为零，则返回结果为 `true`；如果操作数非零，则返回结果为 `false`；若果操作数为 `NOT NUL` 则返回 `NULL`。

语法结构

```
> SELECT not column_name FROM table_name;
```

示例

```
mysql> select not 0;
+-------+
| not 0 |
+-------+
| true |
+-------+
1 row in set (0.02 sec)
mysql> select not null;
+----------+
| not null |
+----------+
| NULL |
+----------+
1 row in set (0.00 sec)
mysql> select not 1;
+-------+
| not 1 |
+-------+
| false |
+-------+
1 row in set (0.01 sec)
```

```
create table t1 (a boolean,b bool);
insert into t1 values (0,1),(true,false),(true,1),(0,false),(NULL,NULL);

mysql> select not a and not b from t1;
+-----------------+
| not a and not b |
+-----------------+
| false |
| false |
| false |
| true |
| NULL |
+-----------------+
5 rows in set (0.00 sec)
```

限制

MatrixOne 暂时还不支持 `!` 运算符。

OR

运算符说明

`OR,||` 逻辑运算符用作*逻辑或*运算。当两个操作数都非 `null` 时，如果操作数同时也非零，则返回结果为 `true`，否则为 `false`；对于 `NULL` 操作数，如果另一个操作数非零，则返回结果为 `true`，否则为 `NULL`；如果两个操作数都为 `NULL`，则返回结果为 `NULL`。

语法结构

```
> SELECT column_1 OR column_2 FROM table_name;
```

示例

```
mysql> select 1 or 1;
+--------+
| 1 or 1 |
+--------+
| true |
+--------+
1 row in set (0.01 sec)

mysql> select 1 or 0;
+--------+
| 1 or 0 |
+--------+
| true |
+--------+
1 row in set (0.00 sec)

mysql> select 0 or 0;
+--------+
| 0 or 0 |
+--------+
| false |
+--------+
1 row in set (0.01 sec)

mysql> select 0 or null;
+-----------+
| 0 or null |
+-----------+
| NULL |
+-----------+
1 row in set (0.00 sec)

mysql> select 1 or null;
+-----------+
| 1 or null |
+-----------+
| true |
+-----------+
1 row in set (0.00 sec)
```

```
create table t1 (a boolean,b bool);
insert into t1 values (0,1),(true,false),(true,1),(0,false),(NULL,NULL);

mysql> select a or b from t1;
+--------+
| a or b |
+--------+
| true |
| true |
| true |
| false |
| NULL |
+--------+
5 rows in set (0.00 sec)
```

XOR

运算符说明

`XOR` 逻辑运算符用作于*逻辑异或*运算。如果任意一个操作数为 `NULL` 则返回结果为 `NULL`；对于非 `NULL` 操作数，如果有奇数个操作数是非零，则返回结果为 `true`，否则返回结果为 `false`。

`a XOR b` 在数学运算上等于 `(a AND (NOT b)) OR ((NOT a) and b)`。

语法结构

```
> SELECT column_1 XOR column_2 FROM table_name;
```

示例

```
mysql> select 1 xor 1;
+---------+
| 1 xor 1 |
+---------+
| false |
+---------+
1 row in set (0.01 sec)

mysql> select 1 xor 0;
+---------+
| 1 xor 0 |
+---------+
| true |
+---------+
1 row in set (0.00 sec)

mysql> select 1 xor null;
+------------+
| 1 xor null |
+------------+
| NULL |
+------------+
1 row in set (0.01 sec)

mysql> select 1 xor 1 xor 1;
+---------------+
| 1 xor 1 xor 1 |
+---------------+
| true |
+---------------+
1 row in set (0.00 sec)
```

```
create table t1 (a boolean,b bool);
insert into t1 values (0,1),(true,false),(true,1),(0,false),(NULL,NULL);

mysql> select a xor b from t1;
+---------+
| a xor b |
+---------+
| true |
| true |
| false |
| false |
| NULL |
+---------+
5 rows in set (0.00 sec)
```

SELECT

语法描述

`SELECT` 语句用于从表中检索数据。

语法结构

```
SELECT
[ALL | DISTINCT ]
select_expr [, select_expr] [[AS] alias] ...
[INTO variable [, ...]]
[FROM table_references
[WHERE where_condition]
[GROUP BY {col_name | expr | position}
[ASC | DESC]]
[HAVING where_condition]
[ORDER BY {col_name | expr | position}
[ASC | DESC]] [ NULLS { FIRST | LAST } ]
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
```

语法解释

`SELECT` 语句中最常用的子句或条件释义如下：

`select_expr`

每个 `select_expr` 表达式表示你需要查询的列，并且必须至少有一个 `select_expr`。

`select_expr` 列表包含指示要查询所选列表的哪些列。`select_expr` 指定列，也可以使用 * 指定全部查询列：

```
SELECT * FROM t1
```

- `tbl_name.*` 可用作以从表中选择所有列：

```
SELECT t1.*, t2.* FROM t1
```

- `select_expr` 可以使用 `AS` 为表指定别名。

`table_references`

- 你可以将默认数据库中的表称为 `tbl_name` 或 `db_name.tbl_name`，主要用于明确指定数据库。您可以将列称为 `col_name`、`tbl_name.col_name` 或 `db_name.tbl_name.col_name`。你不需要为列指定 `tbl_name` 或 `db_name.tbl_name`，如果需要明确指定，可以添加 `tbl_name` 或 `db_name.tbl_name`。
- 可以使用 `tbl_name AS alias_name` 或 `tbl_name alias_name` 为表起别名。

`WHERE`

`WHERE` 子句（如果给定）指示要选择行必须满足的一个或多个条件。`where_condition` 表达式，对于要选择的每一行计算结果为真。如果没有 `WHERE` 子句，该语句将选择所有行。

`GROUP BY`

可以使用列名、列别名或列位置在 `ORDER BY` 和 `GROUP BY` 子句中引用选择的列。

`HAVING`

`HAVING` 子句与 `WHERE` 子句一样，指定选择条件。

`ORDER BY`

`ORDER BY` 默认为升序；可以使用 ASC 关键字明确指定。要以相反的顺序排序，请将（降序）关键字添加到你作为排序依据 DESC 的子句中的列的名称。

`LIMIT`

`LIMIT` 子句可用于限制 `SELECT` 语句返回的行数。

示例

```
create table t1 (spID int,userID int,score smallint);
insert into t1 values (1,1,1);
insert into t1 values (2,2,2);
insert into t1 values (2,1,4);
insert into t1 values (3,3,3);
insert into t1 values (1,1,5);
insert into t1 values (4,6,10);
insert into t1 values (5,11,99);
insert into t1 values (null,0,99);

mysql> SELECT * FROM t1 WHERE spID>2 AND userID <2 || userID >=2 OR
userID < 2 LIMIT 3;
+------+--------+-------+
| spid | userid | score |
+------+--------+-------+
| NULL | 0 | 99 |
| 1 | 1 | 1 |
| 2 | 2 | 2 |
+------+--------+-------+

mysql> SELECT userID,MAX(score) max_score FROM t1 WHERE userID <2 ||
userID > 3 GROUP BY userID ORDER BY max_score;
+--------+-----------+
| userid | max_score |
+--------+-----------+
| 1 | 5 |
| 6 | 10 |
| 0 | 99 |
| 11 | 99 |
+--------+-----------+

mysql> select userID,count(score) from t1 group by userID having
count(score)>1 order by userID;
+--------+--------------+
| userid | count(score) |
+--------+--------------+
| 1 | 3 |
+--------+--------------+

mysql> select userID,count(score) from t1 where userID>2 group by userID
having count(score)>1 order by userID;
Empty set (0.01 sec)s
```

```
mysql> select * from t1 order by spID asc nulls last;
+------+--------+-------+
| spid | userid | score |
+------+--------+-------+
| 1 | 1 | 1 |
| 1 | 1 | 5 |
| 2 | 2 | 2 |
| 2 | 1 | 4 |
| 3 | 3 | 3 |
| 4 | 6 | 10 |
| 5 | 11 | 99 |
| NULL | 0 | 99 |
+------+--------+-------+
```

限制

- 在 `GROUP BY` 中暂不支持表别名。
- 暂不支持 `SELECT...FOR UPDATE`。
- 部分支持 `INTO OUTFILE`。

子查询

子查询，也称为嵌套查询或子选择，是 `SELECT` 子查询语句嵌入在另一个 `SQL` 查询的查询方式。

参见下面的子查询示例：

```
SELECT * FROM t1 WHERE column1 = (SELECT column1 FROM t2);
```

在示例中，`SELECT * FROM t1 WHERE column1` 是外部查询（或外部语句），`(SELECT column1 FROM t2)` 是子查询。子查询语句必须写在括号内，然后嵌套在外部查询语句中，也可以嵌套在其他子查询语句中，形成多层嵌套。

子查询的主要优点：

- 子查询可以划分语句，提供结构化查询。
- 子查询可替代复杂的 `JOIN` 和 `UNIONS` 语句。
- 子查询比复杂的 `JOIN` 和 `UNIONS` 可读性强。

一个子查询有以下几类：

- SELECT 子查询
- FROM 子查询
- WHERE 子查询

更多信息，参见：

- 派生表
- 子查询与比较操作符的使用
- 子查询与 ANY 或 SOME 操作符的使用
- 子查询与 ALL 操作符的使用
- 子查询与 EXISTS 操作符的使用
- 子查询与 IN 操作符的使用

限制

MatrixOne 暂不支持选择多列进行子查询。

Derived Tables

语法描述

当 `SELECT` 语句的 `FROM` 从句中使用独立子查询时，我们也经常将其称为派生表，因为实际上外部查询将子查询的结果当作了一个数据源。

语法结构

每个 FROM 子查询的表都必须要有一个名字，因此 [AS] 操作符是必须的。子查询的 SELECT 列表中每个列也必须要有一个唯一的名字。

```
> SELECT ... FROM (subquery) [AS] name ...
```

示例

```
> CREATE TABLE tb1 (c1 INT, c2 CHAR(5), c3 FLOAT);
> INSERT INTO tb1 VALUES (1, '1', 1.0);
> INSERT INTO tb1 VALUES (2, '2', 2.0);
> INSERT INTO tb1 VALUES (3, '3', 3.0);
> select * from tb1;
+------+------+--------+
| c1 | c2 | c3 |
+------+------+--------+
| 1 | 1 | 1.0000 |
| 2 | 2 | 2.0000 |
| 3 | 3 | 3.0000 |
+------+------+--------+
3 rows in set (0.03 sec)

> SELECT sc1, sc2, sc3 FROM (SELECT c1 AS sc1, c2 AS sc2, c3*3 AS sc3
FROM tb1) AS sb WHERE sc1 > 1;
+------+------+--------+
| sc1 | sc2 | sc3 |
+------+------+--------+
| 2 | 2 | 6.0000 |
| 3 | 3 | 9.0000 |
+------+------+--------+
2 rows in set (0.02 sec)
``````sql
CREATE TABLE tb1 (c1 INT, c2 CHAR(5), c3 FLOAT);
INSERT INTO tb1 VALUES (1, '1', 1.0);
INSERT INTO tb1 VALUES (2, '2', 2.0);
INSERT INTO tb1 VALUES (3, '3', 3.0);

mysql> select * from tb1;
+------+------+--------+
| c1 | c2 | c3 |
+------+------+--------+
| 1 | 1 | 1.0000 |
| 2 | 2 | 2.0000 |
| 3 | 3 | 3.0000 |
+------+------+--------+
3 rows in set (0.03 sec)

mysql> SELECT sc1, sc2, sc3 FROM (SELECT c1 AS sc1, c2 AS sc2, c3*3 AS
sc3 FROM tb1) AS sb WHERE sc1 > 1;
+------+------+--------+
```
```

```
| sc1 | sc2 | sc3 |
+------+------+--------+
| 2 | 2 | 6.0000 |
| 3 | 3 | 9.0000 |
+------+------+--------+
2 rows in set (0.02 sec)
```

- **Subquery with Join**:

```
> create table t1 (libname1 varchar(21) not null primary key, city
varchar(20));
> create table t2 (isbn2 varchar(21) not null primary key, author
varchar(20), title varchar(60));
> create table t3 (isbn3 varchar(21) not null, libname3 varchar(21) not
null, quantity int);
> insert into t2 values ('001','Daffy','Aducklife');
> insert into t2 values ('002','Bugs','Arabbitlife');
> insert into t2 values ('003','Cowboy','Lifeontherange');
> insert into t2 values ('000','Anonymous','Wannabuythisbook?');
> insert into t2 values ('004','BestSeller','OneHeckuvabook');
> insert into t2 values ('005','EveryoneBuys','Thisverybook');
> insert into t2 values ('006','SanFran','Itisasanfranlifestyle');
> insert into t2 values ('007','BerkAuthor','Cool.Berkley.the.book');
> insert into t3 values('000','NewYorkPublicLibra',1);
> insert into t3 values('001','NewYorkPublicLibra',2);
> insert into t3 values('002','NewYorkPublicLibra',3);
> insert into t3 values('003','NewYorkPublicLibra',4);
> insert into t3 values('004','NewYorkPublicLibra',5);
> insert into t3 values('005','NewYorkPublicLibra',6);
> insert into t3 values('006','SanFransiscoPublic',5);
> insert into t3 values('007','BerkeleyPublic1',3);
> insert into t3 values('007','BerkeleyPublic2',3);
> insert into t3 values('001','NYC Lib',8);
> insert into t1 values ('NewYorkPublicLibra','NewYork');
> insert into t1 values ('SanFransiscoPublic','SanFran');
> insert into t1 values ('BerkeleyPublic1','Berkeley');
> insert into t1 values ('BerkeleyPublic2','Berkeley');
> insert into t1 values ('NYCLib','NewYork');
> select * from (select city,libname1,count(libname1) as a from t3 join
t1 on libname1=libname3 join t2 on isbn3=isbn2 group by city,libname1)
sub ;
+----------+--------------------+------+
| city | libname1 | a |
+----------+--------------------+------+
| NewYork | NewYorkPublicLibra | 6 |
| SanFran | SanFransiscoPublic | 1 |
| Berkeley | BerkeleyPublic1 | 1 |
| Berkeley | BerkeleyPublic2 | 1 |
+----------+--------------------+------+
4 rows in set (0.00 sec)
``````sql
create table t1 (libname1 varchar(21) not null primary key, city
varchar(20));
```
```

```
create table t2 (isbn2 varchar(21) not null primary key, author
varchar(20), title varchar(60));
create table t3 (isbn3 varchar(21) not null, libname3 varchar(21) not
null, quantity int);
insert into t2 values ('001','Daffy','Aducklife');
insert into t2 values ('002','Bugs','Arabbitlife');
insert into t2 values ('003','Cowboy','Lifeontherange');
insert into t2 values ('000','Anonymous','Wannabuythisbook?');
insert into t2 values ('004','BestSeller','OneHeckuvabook');
insert into t2 values ('005','EveryoneBuys','Thisverybook');
insert into t2 values ('006','SanFran','Itisasanfranlifestyle');
insert into t2 values ('007','BerkAuthor','Cool.Berkley.the.book');
insert into t3 values('000','NewYorkPublicLibra',1);
insert into t3 values('001','NewYorkPublicLibra',2);
insert into t3 values('002','NewYorkPublicLibra',3);
insert into t3 values('003','NewYorkPublicLibra',4);
insert into t3 values('004','NewYorkPublicLibra',5);
insert into t3 values('005','NewYorkPublicLibra',6);
insert into t3 values('006','SanFransiscoPublic',5);
insert into t3 values('007','BerkeleyPublic1',3);
insert into t3 values('007','BerkeleyPublic2',3);
insert into t3 values('001','NYC Lib',8);
insert into t1 values ('NewYorkPublicLibra','NewYork');
insert into t1 values ('SanFransiscoPublic','SanFran');
insert into t1 values ('BerkeleyPublic1','Berkeley');
insert into t1 values ('BerkeleyPublic2','Berkeley');
insert into t1 values ('NYCLib','NewYork');

mysql> select * from (select city,libname1,count(libname1) as a from t3
join t1 on libname1=libname3 join t2 on isbn3=isbn2 group by
city,libname1) sub ;
+----------+--------------------+------+
| city | libname1 | a |
+----------+--------------------+------+
| NewYork | NewYorkPublicLibra | 6 |
| SanFran | SanFransiscoPublic | 1 |
| Berkeley | BerkeleyPublic1 | 1 |
| Berkeley | BerkeleyPublic2 | 1 |
+----------+--------------------+------+
4 rows in set (0.00 sec)
```

限制

MatrixOne 暂不支持选择多列进行子查询。

子查询与比较操作符的使用

语法描述

子查询与比较操作符最常见的用法如下：

```
non_subquery_operand comparison_operator (subquery)
```

其中，`comparison_operator` 指以下操作符：

```
= > < >= <= <> != <=>
```

语法结构

```
> SELECT column_name(s) FROM table_name WHERE 'a' = (SELECT column1 FROM
t1)
```

示例

```
create table t1 (a int);
create table t2 (a int, b int);
create table t3 (a int);
create table t4 (a int not null, b int not null);
insert into t1 values (2);
insert into t2 values (1,7),(2,7);
insert into t4 values (4,8),(3,8),(5,9);
insert into t3 values (6),(7),(3);

mysql> select * from t3 where a = (select b from t2);
ERROR 1105 (HY000): scalar subquery returns more than 1 row
mysql> select * from t3 where a = (select distinct b from t2);
+------+
| a |
+------+
| 7 |
+------+
1 rows in set (0.01 sec)

mysql> select a,b from t4 where a > ( select a ,b from t2 where a>1);
ERROR 1105 (HY000): Internal error: Unknow type TUPLE
```

限制

MatrixOne 暂不支持选择多列进行子查询。

Subqueries with ANY or SOME

语法描述

由于列子查询返回的结果集是多行一列，因此不能直接使用（=，>，<，>=，<=，<>）这些比较操作符。在列子查询中可以使用 `ANY`、`SOME` 操作符与比较操作符联合使用：

- `ANY`：与比较操作符联合使用，表示与子查询返回的任何值比较为 `TRUE`，则返回结果为 `true`。
- `SOME`：`ANY` 的别名，与 `ANY` 意义相同，但较少使用。

语法结构

```
> SELECT column_name(s) FROM table_name WHERE column_name ANY (subquery);
```

示例

```
create table t1 (a int);
create table t2 (a int, b int);
create table t3 (a int);
create table t4 (a int not null, b int not null);
create table t5 (a int);
create table t6 (a int, b int);
insert into t1 values (2);
insert into t2 values (1,7),(2,7);
insert into t4 values (4,8),(3,8),(5,9);
insert into t5 values (null);
insert into t3 values (6),(7),(3);
insert into t6 values (10,7),(null,7);

mysql> select * from t3 where a <> any (select b from t2);
+------+
| a |
+------+
| 6 |
| 3 |
+------+
2 rows in set (0.00 sec)

mysql> select * from t3 where a <> some (select b from t2);
+------+
| a |
+------+
| 6 |
| 3 |
+------+
2 rows in set (0.00 sec)

mysql> select * from t3 where a = some (select b from t2);
+------+
| a |
+------+
| 7 |
+------+
1 row in set (0.00 sec)

mysql> select * from t3 where a = any (select b from t2);
+------+
| a |
```

```
+------+
| 7 |
+------+
1 row in set (0.00 sec)

mysql> select a,b from t6 where a > any ( select a ,b from t4 where a>3);
ERROR 1105 (HY000): subquery should return 1 column
```

限制

MatrixOne 暂不支持选择多列进行子查询。

Subqueries with ALL

语法描述

关键词 `ALL` 必须跟在比较操作符后面，指如果子查询返回的列中值的 `ALL` 的比较是 `TRUE`，则返回 `TRUE`。

```
operand comparison_operator ALL (subquery)
```

示例如下：

```
SELECT s1 FROM t1 WHERE s1 > ALL (SELECT s1 FROM t2);
```

如上述示例中，假设表 t1 中有一行包含 (10)，表 t2 包含 (-5,0，+5)，则表达式为 `TRUE`，因为 10 大于 t2 中的所有三个值。如果表 t2 包含 (12,6,NULL，-100)，则表达式为 `FALSE`，因为在表 t2 中有一个大于 10 的值 12。如果表 t2 包含 (0,NULL,1)，则表达式为 `NULL`。

- 如果表 t2 为空，则表达式为 `TRUE`。例如，当下表 t2 为空时，表达式是 `TRUE`：

```
SELECT * FROM t1 WHERE 1 > ALL (SELECT s1 FROM t2);
```

- 下面示例中，当表 t2 为空时，这个表达式是 `NULL`：

```
SELECT * FROM t1 WHERE 1 > (SELECT s1 FROM t2);
```

或：

```
SELECT * FROM t1 WHERE 1 > ALL (SELECT MAX(s1) FROM t2);
```

说明：在书写子查询语法时，要注意考虑到含有 `NULL` 值的表和空表的情况。

语法结构

```
> SELECT column_name(s) FROM table_name {WHERE | HAVING} [not] expression
comparison_operator ALL (subquery)
```

示例

```
create table t1 (a int);
create table t2 (a int, b int);
create table t3 (a int);
create table t4 (a int not null, b int not null);
create table t5 (a int);
create table t6 (a int, b int);
insert into t1 values (2);
insert into t2 values (1,7),(2,7);
insert into t4 values (4,8),(3,8),(5,9);
insert into t5 values (null);
insert into t3 values (6),(7),(3);
insert into t6 values (10,7),(null,7);

mysql> select * from t3 where a <> all (select b from t2);
+------+
| a |
+------+
| 6 |
| 3 |
+------+
2 rows in set (0.00 sec)

mysql> select * from t4 where 5 > all (select a from t5);
+------+------+
| a | b |
+------+------+
| 4 | 8 |
| 3 | 8 |
| 5 | 9 |
+------+------+
3 rows in set (0.01 sec)

mysql> select * from t3 where 10 > all (select b from t2);
+------+
| a |
+------+
| 6 |
| 7 |
| 3 |
+------+
3 rows in set (0.00 sec)
```

```
mysql> select a,b from t6 where a > all ( select a ,b from t4 where a>3);
ERROR 1105 (HY000): subquery should return 1 column
```

限制

MatrixOne 暂不支持选择多列进行子查询。

Subqueries with EXISTS or NOT EXISTS

语法描述

`EXISTS` 用于检查子查询是否至少会返回一行数据。即将主查询的数据，放到子查询中做条件验证，根据验证结果（TRUE 或 FALSE）来决定主查询的数据结果是否得以保留。

如果子查询返回任何行，`EXISTS` 子查询条件为 `TRUE`，`NOT EXISTS` 子查询条件为 `FALSE`。

语法结构

```
> SELECT column_name(s)
FROM table_name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE condition);
```

示例

```
create table t1 (a int);
create table t2 (a int, b int);
create table t3 (a int);
create table t4 (a int not null, b int not null);
insert into t1 values (2);
insert into t2 values (1,7),(2,7);
insert into t4 values (4,8),(3,8),(5,9);
insert into t3 values (6),(7),(3);

mysql> select * from t3 where exists (select * from t2 where t2.b=t3.a);
+------+
| a |
+------+
| 7 |
+------+
1 row in set (0.00 sec)
mysql> select * from t3 where not exists (select * from t2 where
t2.b=t3.a);
+------+
| a |
+------+
| 6 |
| 3 |
+------+
2 rows in set (0.00 sec)
```

限制

MatrixOne 暂不支持选择多列进行子查询。

Subqueries with IN

语法描述

子查询可以与 `IN` 操作符一起使用，作为 “表达式 IN (子查询)”，查询某个范围内的数据。子查询应该返回带有一行或多行的单个列，以形成 `IN` 操作使用的值列表。

对多记录、单列子查询使用 `IN` 子句。子查询返回 `IN` 或 `NOT IN` 引入的结果后，外部查询使用它们返回最终结果。

- 如果子查询结果中有匹配的行，则结果为 `TRUE`。
- 如果子查询结果为 `NULL`，则返回结果为 `false`。
- 如果子查询结果中没有匹配的行，结果也是 `FALSE`。
- 如果子查询结果中所有的值都为 `NULL`，则返回结果为 `false`。

语法结构

```
> SELECT ... FROM table_name WHERE column_name IN (subquery)
```

示例

```
create table t1(val varchar(10));
insert into t1 values ('aaa'), ('bbb'),('eee'),('mmm'),('ppp');

mysql> select count(*) from t1 as w1 where w1.val in (select w2.val from
t1 as w2 where w2.val like 'm%') and w1.val in (select w3.val from t1 as
w3 where w3.val like 'e%');
+----------+
| count(*) |
+----------+
| 0 |
+----------+
1 row in set (0.01 sec)
```

```
create table t1 (id int not null, text varchar(20) not null default '',
primary key (id));
insert into t1 (id, text) values (1, 'text1'), (2, 'text2'), (3,
'text3'), (4, 'text4'), (5, 'text5'), (6, 'text6'), (7, 'text7'), (8,
'text8'), (9, 'text9'), (10, 'text10'), (11, 'text11'), (12, 'text12');

mysql> select * from t1 where id not in (select id from t1 where id < 8);
+------+--------+
| id | text |
+------+--------+
| 8 | text8 |
| 9 | text9 |
| 10 | text10 |
| 11 | text11 |
| 12 | text12 |
+------+--------+
5 rows in set (0.00 sec)
```

```
CREATE TABLE t1 (a int);
CREATE TABLE t2 (a int, b int);
CREATE TABLE t3 (b int NOT NULL);
INSERT INTO t1 VALUES (1), (2), (3), (4);
INSERT INTO t2 VALUES (1,10), (3,30);

mysql> select * from t1 where t1.a in (SELECT t1.a FROM t1 LEFT JOIN t2
ON t2.a=t1.a);
+------+
| a |
+------+
| 1 |
| 2 |
| 3 |
| 4 |
+------+
4 rows in set (0.01 sec)

mysql> SELECT * FROM t2 LEFT JOIN t3 ON t2.b=t3.b WHERE t3.b IS NOT NULL
OR t2.a > 10;
Empty set (0.01 sec)
mysql> SELECT * FROM t1 WHERE t1.a NOT IN (SELECT a FROM t2 LEFT JOIN t3
ON t2.b=t3.b WHERE t3.b IS NOT NULL OR t2.a > 10);
+------+
| a |
+------+
| 1 |
| 2 |
| 3 |
| 4 |
+------+
4 rows in set (0.00 sec)
```

```
create table t1 (a int);
create table t2 (a int, b int);
create table t3 (a int);
create table t4 (a int not null, b int not null);
create table t5 (a int);
create table t6 (a int, b int);
insert into t1 values (2);
insert into t2 values (1,7),(2,7);
insert into t4 values (4,8),(3,8),(5,9);
insert into t5 values (null);
insert into t3 values (6),(7),(3);
insert into t6 values (10,7),(null,7);

mysql> select a,b from t6 where (a,b) in ( select a,b from t4 where a>3);
Empty set (0.02 sec)
```

限制

MatrixOne 暂不支持选择多列进行子查询。

JOIN

语法说明

`JOIN` 用于把来自两个或多个表的行结合起来。

下图展示了 `LEFT JOIN`、`RIGHT JOIN`、`INNER JOIN`、和 `OUTER JOIN`。

- `LEFT JOIN`

leftjoin

SELECT <select_list> FROM TableA A LEFT JOIN TableB B ON A.Key=B.Key

---

SELECT <select_list> FROM TableA A LEFT JOIN TableB B ON A.Key=B.Key WHERE B.Key IS NULL

leftjoinwhere

- `RIGHT JOIN`

leftjoinwhere

SELECT <select_list> FROM TableA A RIGHT JOIN TableB B ON A.Key=B.Key

---

SELECT <select_list> FROM TableA A RIGHT JOIN TableB B ON A.Key=B.Key WHERE A.Key IS NULL

leftjoinwhere

- `INNER JOIN`

innerjoin

SELECT <select_list> FROM TableA A INNER JOIN TableB B ON A.Key=B.Key

---

- `FULL JOIN`

SELECT <select_list> FROM TableA A FULL OUTER JOIN TableB B ON A.Key=B.Key	**leftjoin**
SELECT <select_list> FROM TableA A FULL OUTER JOIN TableB B ON A.Key=B.Key WHERE A.Key IS NULL OR B.Key IS NULL	fulljoinwhere

更多信息，参考：

- LEFT JOIN
- RIGHT JOIN
- INNER JOIN
- FULL JOIN
- OUTER JOIN
- NATURAL JOIN

INNER JOIN

语法说明

`INNER JOIN` 选取两个表中具有匹配值的数据记录。（等同于 `JOIN`）

语法结构

```
> SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

示例

```
drop table if exists t1,t2,t3;
create table t1 (libname1 varchar(21) not null primary key, city
varchar(20));
create table t2 (isbn2 varchar(21) not null primary key, author
varchar(20), title varchar(60));
create table t3 (isbn3 varchar(21) not null, libname3 varchar(21) not
null, quantity int);
insert into t2 values ('001','Daffy','Aducklife');
insert into t2 values ('002','Bugs','Arabbitlife');
insert into t2 values ('003','Cowboy','Lifeontherange');
insert into t2 values ('000','Anonymous','Wannabuythisbook?');
insert into t2 values ('004','BestSeller','OneHeckuvabook');
insert into t2 values ('005','EveryoneBuys','Thisverybook');
insert into t2 values ('006','SanFran','Itisasanfranlifestyle');
insert into t2 values ('007','BerkAuthor','Cool.Berkley.the.book');
insert into t3 values ('000','NewYorkPublicLibra',1);
insert into t3 values ('001','NewYorkPublicLibra',2);
insert into t3 values ('002','NewYorkPublicLibra',3);
insert into t3 values ('003','NewYorkPublicLibra',4);
insert into t3 values ('004','NewYorkPublicLibra',5);
insert into t3 values ('005','NewYorkPublicLibra',6);
insert into t3 values ('006','SanFransiscoPublic',5);
insert into t3 values ('007','BerkeleyPublic1',3);
insert into t3 values ('007','BerkeleyPublic2',3);
insert into t3 values ('001','NYC Lib',8);
insert into t1 values ('NewYorkPublicLibra','NewYork');
insert into t1 values ('SanFransiscoPublic','SanFran');
insert into t1 values ('BerkeleyPublic1','Berkeley');
insert into t1 values ('BerkeleyPublic2','Berkeley');
insert into t1 values ('NYCLib','NewYork');

mysql> select city,libname1,count(libname1) as a from t3 join t1 on
libname1=libname3 join t2 on isbn3=isbn2 group by city,libname1;
+----------+--------------------+------+
| city | libname1 | a |
+----------+--------------------+------+
| NewYork | NewYorkPublicLibra | 6 |
| SanFran | SanFransiscoPublic | 1 |
| Berkeley | BerkeleyPublic1 | 1 |
| Berkeley | BerkeleyPublic2 | 1 |
+----------+--------------------+------+
```

LEFT JOIN

语法说明

`LEFT JOIN` 关键字从左表（table1）返回所有的行，即使右表（table2）中没有匹配。如果右表中没有匹配，则结果为 `NULL`。

说明：在一些数据库中：`LEFT JOIN` 等同于 `LEFT OUTER JOIN`。

语法结构

```
> SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name=table2.column_name;
```

示例

```
drop table if exists t1,t2,t3;
create table t1 (libname1 varchar(21) not null primary key, city
varchar(20));
create table t2 (isbn2 varchar(21) not null primary key, author
varchar(20), title varchar(60));
create table t3 (isbn3 varchar(21) not null, libname3 varchar(21) not
null, quantity int);
insert into t2 values ('001','Daffy','Aducklife');
insert into t2 values ('002','Bugs','Arabbitlife');
insert into t2 values ('003','Cowboy','Lifeontherange');
insert into t2 values ('000','Anonymous','Wannabuythisbook?');
insert into t2 values ('004','BestSeller','OneHeckuvabook');
insert into t2 values ('005','EveryoneBuys','Thisverybook');
insert into t2 values ('006','SanFran','Itisasanfranlifestyle');
insert into t2 values ('007','BerkAuthor','Cool.Berkley.the.book');
insert into t3 values ('000','NewYorkPublicLibra',1);
insert into t3 values ('001','NewYorkPublicLibra',2);
insert into t3 values ('002','NewYorkPublicLibra',3);
insert into t3 values ('003','NewYorkPublicLibra',4);
insert into t3 values ('004','NewYorkPublicLibra',5);
insert into t3 values ('005','NewYorkPublicLibra',6);
insert into t3 values ('006','SanFransiscoPublic',5);
insert into t3 values ('007','BerkeleyPublic1',3);
insert into t3 values ('007','BerkeleyPublic2',3);
insert into t3 values ('001','NYC Lib',8);
insert into t1 values ('NewYorkPublicLibra','NewYork');
insert into t1 values ('SanFransiscoPublic','SanFran');
insert into t1 values ('BerkeleyPublic1','Berkeley');
insert into t1 values ('BerkeleyPublic2','Berkeley');
insert into t1 values ('NYCLib','NewYork');

mysql> select city,libname1,count(libname1) as a from t3 left join t1 on
libname1=libname3 join t2 on isbn3=isbn2 group by city,libname1;
+----------+--------------------+------+
| city | libname1 | a |
+----------+--------------------+------+
| NewYork | NewYorkPublicLibra | 6 |
| SanFran | SanFransiscoPublic | 1 |
| Berkeley | BerkeleyPublic1 | 1 |
| Berkeley | BerkeleyPublic2 | 1 |
```

```
| NULL | NULL | 0 |
+----------+--------------------+------+
```

RIGHT JOIN

语法说明

`RIGHT JOIN` 关键字从右表（table2）返回所有的行，即使左表（table1）中没有匹配。如果左表中没有匹配，则结果为 `NULL`。

说明：在一些数据库中：`RIGHT JOIN` 等同于 `RIGHT OUTER JOIN`。

语法结构

```
> SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name=table2.column_name;
```

示例

```
drop table if exists t1,t2,t3;
create table t1 (libname1 varchar(21) not null primary key, city
varchar(20));
create table t2 (isbn2 varchar(21) not null primary key, author
varchar(20), title varchar(60));
create table t3 (isbn3 varchar(21) not null, libname3 varchar(21) not
null, quantity int);
insert into t2 values ('001','Daffy','Aducklife');
insert into t2 values ('002','Bugs','Arabbitlife');
insert into t2 values ('003','Cowboy','Lifeontherange');
insert into t2 values ('000','Anonymous','Wannabuythisbook?');
insert into t2 values ('004','BestSeller','OneHeckuvabook');
insert into t2 values ('005','EveryoneBuys','Thisverybook');
insert into t2 values ('006','SanFran','Itisasanfranlifestyle');
insert into t2 values ('007','BerkAuthor','Cool.Berkley.the.book');
insert into t3 values ('000','NewYorkPublicLibra',1);
insert into t3 values ('001','NewYorkPublicLibra',2);
insert into t3 values ('002','NewYorkPublicLibra',3);
insert into t3 values ('003','NewYorkPublicLibra',4);
insert into t3 values ('004','NewYorkPublicLibra',5);
insert into t3 values ('005','NewYorkPublicLibra',6);
insert into t3 values ('006','SanFransiscoPublic',5);
insert into t3 values ('007','BerkeleyPublic1',3);
insert into t3 values ('007','BerkeleyPublic2',3);
insert into t3 values ('001','NYC Lib',8);
insert into t1 values ('NewYorkPublicLibra','NewYork');
insert into t1 values ('SanFransiscoPublic','SanFran');
insert into t1 values ('BerkeleyPublic1','Berkeley');
insert into t1 values ('BerkeleyPublic2','Berkeley');
insert into t1 values ('NYCLib','NewYork');

mysql> select city,libname1,count(libname1) as a from t3 right join t1 on
libname1=libname3 join t2 on isbn3=isbn2 group by city,libname1;
+----------+--------------------+------+
| city | libname1 | a |
+----------+--------------------+------+
| NewYork | NewYorkPublicLibra | 6 |
| SanFran | SanFransiscoPublic | 1 |
| Berkeley | BerkeleyPublic1 | 1 |
| Berkeley | BerkeleyPublic2 | 1 |
+----------+--------------------+------+
```

FULL JOIN

语法说明

`FULL JOIN` 关键字只要左表（table1）和右表（table2）其中一个表中存在匹配，则返回行。

`FULL JOIN` 关键字结合了 `LEFT JOIN` 和 `RIGHT JOIN` 的结果。

说明：在一些数据库中：`FULL JOIN` 等同于 `FULL OUTER JOIN`。

语法结构

```
> SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name=table2.column_name;
```

示例

```
drop table if exists t1,t2,t3;
create table t1 (libname1 varchar(21) not null primary key, city
varchar(20));
create table t2 (isbn2 varchar(21) not null primary key, author
varchar(20), title varchar(60));
create table t3 (isbn3 varchar(21) not null, libname3 varchar(21) not
null, quantity int);
insert into t2 values ('001','Daffy','Aducklife');
insert into t2 values ('002','Bugs','Arabbitlife');
insert into t2 values ('003','Cowboy','Lifeontherange');
insert into t2 values ('000','Anonymous','Wannabuythisbook?');
insert into t2 values ('004','BestSeller','OneHeckuvabook');
insert into t2 values ('005','EveryoneBuys','Thisverybook');
insert into t2 values ('006','SanFran','Itisasanfranlifestyle');
insert into t2 values ('007','BerkAuthor','Cool.Berkley.the.book');
insert into t3 values ('000','NewYorkPublicLibra',1);
insert into t3 values ('001','NewYorkPublicLibra',2);
insert into t3 values ('002','NewYorkPublicLibra',3);
insert into t3 values ('003','NewYorkPublicLibra',4);
insert into t3 values ('004','NewYorkPublicLibra',5);
insert into t3 values ('005','NewYorkPublicLibra',6);
insert into t3 values ('006','SanFransiscoPublic',5);
insert into t3 values ('007','BerkeleyPublic1',3);
insert into t3 values ('007','BerkeleyPublic2',3);
insert into t3 values ('001','NYC Lib',8);
insert into t1 values ('NewYorkPublicLibra','NewYork');
insert into t1 values ('SanFransiscoPublic','SanFran');
insert into t1 values ('BerkeleyPublic1','Berkeley');
insert into t1 values ('BerkeleyPublic2','Berkeley');
insert into t1 values ('NYCLib','NewYork');

mysql> select city,libname1,count(libname1) as a from t3 full join t1 on
libname1=libname3 join t2 on isbn3=isbn2 group by city,libname1;
+----------+--------------------+------+
| city | libname1 | a |
+----------+--------------------+------+
| NewYork | NewYorkPublicLibra | 6 |
| SanFran | SanFransiscoPublic | 1 |
| Berkeley | BerkeleyPublic1 | 1 |
| Berkeley | BerkeleyPublic2 | 1 |
+----------+--------------------+------+
```

OUTER JOIN

语法说明

在 `OUTER JOIN` 中，可以返回一个或两个表中的不匹配行。`OUT JOIN` 请参考：

- `LEFT JOIN` 关键字从左表（table1）返回所有的行。参见 LEFT JOIN.
- `RIGHT JOIN` 关键字从右表（table2）返回所有的行。参见 RIGHT JOIN.
- `FULL OUTER JOIN` 关键字只要左表（table1）和右表（table2）其中一个表中存在匹配，则返回行。参见 FULL JOIN.

示例

```
create table t1 (a1 int, a2 char(3));
insert into t1 values(10,'aaa'), (10,null), (10,'bbb'), (20,'zzz');
create table t2(a1 char(3), a2 int, a3 real);
insert into t2 values('AAA', 10, 0.5);
insert into t2 values('BBB', 20, 1.0);

mysql> select t1.a1, t1.a2, t2.a1, t2.a2 from t1 left outer join t2 on
t1.a1=10;
+------+------+------+------+
| a1 | a2 | a1 | a2 |
+------+------+------+------+
| 10 | aaa | AAA | 10 |
| 10 | aaa | BBB | 20 |
| 10 | NULL | AAA | 10 |
| 10 | NULL | BBB | 20 |
| 10 | bbb | AAA | 10 |
| 10 | bbb | BBB | 20 |
| 20 | zzz | NULL | NULL |
+------+------+------+------+
```

NATURAL JOIN

语法说明

`NATURAL JOIN` 相当于 `INNER JOIN`，作用是将两个表中具有相同名称的列进行匹配。

语法结构

```
> SELECT table_column1, table_column2...
FROM table_name1
NATURAL JOIN table_name2;
```

示例

```
create table t1(id int,desc1 varchar(50),desc2 varchar(50));
create table t2(id int,desc3 varchar(50),desc4 varchar(50));
INSERT INTO t1(id,desc1,desc2) VALUES(100,'desc11','desc12'),
(101,'desc21','desc22'),(102,'desc31','desc32');
INSERT INTO t2(id,desc3,desc4) VALUES(101,'desc41','desc42'),
(103,'desc51','desc52'),(105,'desc61','desc62');

mysql> SELECT t1.id,t2.id,desc1,desc2,desc3,desc4 FROM t1 NATURAL JOIN
t2;
+------+------+--------+--------+--------+--------+
| id | id | desc1 | desc2 | desc3 | desc4 |
+------+------+--------+--------+--------+--------+
| 101 | 101 | desc21 | desc22 | desc41 | desc42 |
+------+------+--------+--------+--------+--------+
1 row in set (0.00 sec)
```

WITH (Common Table Expressions)

语法说明

通用表表达式（CTE，common table expression），它是在单个语句的执行范围内定义的临时结果集，只在查询期间有效。它可以自引用，也可在同一查询中多次引用。

定义 `CTE` 后，可以像 `SELECT`，`INSERT`，`UPDATE`，`DELETE` 或 `CREATE VIEW` 语句一样引用它。

使用 `WITH` 从句指定通用表表达式，`WITH` 从句可以使用一个或多个逗号分隔。每个从句提供一个子查询，该子查询生成一个结果集，并将名称与子查询关联起来。

语法结构

```
WITH cte_name ( column_name [,...n] )
AS
(
CTE_query_definition -- Anchor member is defined.
)
```

参数释义

在包含 `WITH` 从句的语句中，可以引用每个 `CTE` 名称来查询相应的 `CTE` 结果集。

示例

```
CREATE TABLE t1
(a INTEGER,
b INTEGER,
c INTEGER
);
INSERT INTO t1 VALUES
(1, 1, 10), (1, 2, 20), (1, 3, 30), (2, 1, 40), (2, 2, 50), (2, 3, 60);
CREATE TABLE t2
(a INTEGER,
d INTEGER,
e INTEGER
);
INSERT INTO t2 VALUES
(1, 6, 60), (2, 6, 60), (3, 6, 60);
mysql> WITH
cte AS
(SELECT SUM(c) AS c, SUM(b) AS b, a
FROM t1
GROUP BY a)
SELECT t2.a, (SELECT MIN(c) FROM cte AS cte2 WHERE t2.d = cte2.b)
FROM t2 LEFT JOIN cte AS cte1 ON t2.a = cte1.a
LEFT JOIN t2 AS tx ON tx.e = cte1.c;
+------+-------------------------------------------------------+
| a | (select min(c) from cte as cte2 where t2.d = cte2.b) |
+------+-------------------------------------------------------+
| 1 | 60 |
| 1 | 60 |
| 1 | 60 |
| 2 | 60 |
| 3 | 60 |
+------+-------------------------------------------------------+
5 rows in set (0.01 sec)
```

限制

MatrixOne 暂不支持递归 CTE。

组合查询 (UNION, INTERSECT, MINUS)

两个查询的结果可以使用 `UNION`， `INTERSECT` 和 `MINUS` 语法进行组合查询。

示例语法如下：

```
query1 UNION [ALL] query2
query1 INTERSECT [ALL] query2
query1 MINUS [ALL] query2
```

Tips: *query1* 和 *query2* 是可以使用到目前为止讨论的任何功能的查询。

`UNION` 有效地将 *query2* 的结果合并到 *query1* 的结果中（但不能保证这是返回行的顺序）。此外，它以与 `DISTINCT` 语法相同，即从结果中消除重复行；使用了 `UNION ALL`，即从结果中不消除重复行。

`INTERSECT` 返回 *query1* 和 *query2* 相交的结果中的所有行。不使用 `INTERSECT ALL`，则消除结果中的重复的行；使用 `INTERSECT ALL`，不消除结果中的重复的行。

`MINUS` 返回 *query1* 结果，但不在 *query2* 中的所有行。即 *query1* 和 *query2* 的结果的差集。同样，不使用 `MINUS ALL`，则消除结果中的重复的行；使用 `MINUS ALL`，不消除结果中的重复的行。

要计算两个查询的并集、交集或差集，这两个查询必须是 “并集兼容的”，这意味着它们返回相同数量的列并且对应的列具有兼容的数据类型。

`UNION`， `INTERSECT` 和 `MINUS` 操作可以组合，例如：

```
query1 UNION query2 MINUS query3
```

它也等价于：

```
(query1 UNION query2) MINUS query3
```

如上述代码行所示，你可以使用括号来控制计算顺序。如果没有括号， `UNION` 和 `MINUS` 从左到右关联。但 `INTERSECT` 比这两个运算符优先级更高，因此参见下面的代码行：

```
query1 UNION query2 INTERSECT query3
```

表示：

```
query1 UNION (query2 INTERSECT query3)
```

你还可以用括号将单个查询括起来。如果查询需要使用以下示例中的子句（例如 `LIMIT` 子句），如果没有括号，将会导致语法错误，该子句在计算过程中将被理解为应用于组合操作的输出而不是其输入之一。如下述例子所示：

```
SELECT a FROM b UNION SELECT x FROM y LIMIT 10
```

它可被接受，但是它表示的计算顺序如下：

```
(SELECT a FROM b UNION SELECT x FROM y) LIMIT 10
```

而不是下面的计算顺序：

```
SELECT a FROM b UNION (SELECT x FROM y LIMIT 10)
```

参考

关于 `UNION`， `INTERSECT` 和 `MINUS` 单个语法的文档，可以参见如下：

- UNION
- INTERSECT
- MINUS

UNION

语法说明

`UNION` 运算符允许您将两个或多个查询结果集合并到一个结果集中。

语法结构

```
SELECT column_list
UNION [DISTINCT | ALL]
SELECT column_list
UNION [DISTINCT | ALL]
SELECT column_list ...
```

语法说明

`UNION` 和 `UNION ALL`

使用 `UNION` 运算符组合两个或多个查询的结果集，需要满足以下条件：

- 所有 `SELECT` 语句中出现的列的数量和顺序必须相同。
- 列的数据类型必须相同或可转换。

使用 `UNION ALL`，则重复行（如果可用）将保留在结果中。因为 `UNION ALL` 不需要处理重复项。

`UNION` 与 `ORDER BY`，`LIMIT`

使用 `ORDER BY` 或 `LIMIT` 子句来对全部 UNION 结果进行分类或限制，则应对单个地 `SELECT` 语句加圆括号，并把 `ORDER BY` 或 `LIMIT` 放到最后一个的后面。

例如：

```
(SELECT a FROM t1 WHERE a=10 AND B=1 ORDER BY a LIMIT 10)
UNION
(SELECT a FROM t2 WHERE a=11 AND B=2 ORDER BY a LIMIT 10);
```

或：

```
(SELECT a FROM t1 WHERE a=10 AND B=1)
UNION
(SELECT a FROM t2 WHERE a=11 AND B=2)
ORDER BY a LIMIT 10;
```

示例

```
CREATE TABLE t1 (id INT PRIMARY KEY);
CREATE TABLE t2 (id INT PRIMARY KEY);
INSERT INTO t1 VALUES (1),(2),(3);
INSERT INTO t2 VALUES (2),(3),(4);

mysql> SELECT id FROM t1 UNION SELECT id FROM t2;
+------+
| id |
+------+
| 4 |
| 1 |
| 2 |
| 3 |
+------+

mysql> SELECT id FROM t1 UNION ALL SELECT id FROM t2;
+------+
| id |
+------+
| 1 |
| 2 |
| 3 |
| 2 |
| 3 |
| 4 |
+------+
```

```
drop table t1;
CREATE TABLE t1 (a INT, b INT);
INSERT INTO t1 VALUES ROW(4,-2),ROW(5,9),ROW(10,1),ROW(11,2),ROW(13,5);
drop table t2;
CREATE TABLE t2 (a INT, b INT);
INSERT INTO t2 VALUES ROW(1,2),ROW(3,4),ROW(11,2),ROW(10,3),ROW(15,8);

mysql> (SELECT a FROM t1 WHERE a=10 AND B=1 ORDER BY a LIMIT 10) UNION
(SELECT a FROM t2 WHERE a=11 AND B=2 ORDER BY a LIMIT 10);
+------+
| a |
+------+
| 10 |
| 11 |
+------+
```

INTERSECT

语法说明

`INTERSECT` 运算符是一个集合运算符仅返回两个查询或多个查询的不同行。

语法结构

```
SELECT column_list FROM table_1
INTERSECT
SELECT column_list FROM table_2;
```

示例

```
drop table if exists t1;
drop table if exists t2;
create table t1 (a smallint, b bigint, c int);
insert into t1 values (1,2,3);
insert into t1 values (1,2,3);
insert into t1 values (3,4,5);
insert into t1 values (4,5,6);
insert into t1 values (4,5,6);
insert into t1 values (1,1,2);
create table t2 (a smallint, b bigint, c int);
insert into t2 values (1,2,3);
insert into t2 values (3,4,5);
insert into t2 values (1,2,1);

mysql> select * from t1 intersect select * from t2;
+------+------+------+
| a | b | c |
+------+------+------+
| 1 | 2 | 3 |
| 3 | 4 | 5 |
+------+------+------+
2 rows in set (0.01 sec)

mysql> select a, b from t1 intersect select b, c from t2;
+------+------+
| a | b |
+------+------+
| 4 | 5 |
+------+------+
1 row in set (0.01 sec)
```

MINUS

语法说明

`MINUS` 比较两个查询的结果，并返回第一个查询中不是由第二个查询输出的不同行。

语法结构

```
SELECT column_list_1 FROM table_1
MINUS
SELECT columns_list_2 FROM table_2;
```

示例

- 示例 1

```
CREATE TABLE t1 (id INT PRIMARY KEY);
CREATE TABLE t2 (id INT PRIMARY KEY);
INSERT INTO t1 VALUES (1),(2),(3);
INSERT INTO t2 VALUES (2),(3),(4);

mysql> SELECT id FROM t1 MINUS SELECT id FROM t2;
+------+
| id |
+------+
| 1 |
+------+
```

- 示例 2

```
drop table if exists t1;
drop table if exists t2;
create table t1 (a smallint, b bigint, c int);
insert into t1 values (1,2,3);
insert into t1 values (1,2,3);
insert into t1 values (3,4,5);
insert into t1 values (4,5,6);
insert into t1 values (4,5,6);
insert into t1 values (1,1,2);
create table t2 (a smallint, b bigint, c int);
insert into t2 values (1,2,3);
insert into t2 values (3,4,5);
insert into t2 values (1,2,1);

mysql> select * from t1 minus select * from t2;
+------+------+------+
| a | b | c |
+------+------+------+
| 1 | 1 | 2 |
| 4 | 5 | 6 |
+------+------+------+

mysql> select a, b from t1 minus select b, c from t2;
+------+------+
| a | b |
+------+------+
| 3 | 4 |
| 1 | 1 |
| 1 | 2 |
+------+------+
```

CREATE ACCOUNT

语法说明

为其中一个集群成员创建一个新的租户。

语法结构

```
> CREATE ACCOUNT [IF NOT EXISTS]
account auth_option
[COMMENT 'comment_string']

auth_option: {
ADMIN_NAME [=] 'admin_name'
IDENTIFIED BY 'auth_string'
}
```

语法说明

auth_option

指定租户默认的帐号名和授权方式，`auth_string` 表示显式返回指定密码。

示例

```
> create account tenant_test admin_name = 'root' identified by '111' open
comment 'tenant_test';
Query OK, 0 rows affected (0.08 sec)
```

ALTER ACCOUNT

语法说明

修改租户信息。

> **注意**
>
> 仅被授权 moadmin 角色的集群管理员（即 sysaccount 用户）可以执行**暂停（SUSPEND）和恢复（OPEN）**租户的操作。

语法结构

```
> ALTER ACCOUNT [IF EXISTS]
account auth_option [COMMENT 'comment_string']

auth_option: \{
ADMIN_NAME [=] 'admin_name'
IDENTIFIED BY 'auth_string'
}

status_option: \{
OPEN
| SUSPEND
}
```

参数释义

auth_option

修改租户的帐号名和授权方式，`auth_string` 表示显式返回指定密码。

status_option

设置租户的状态。作为 VARCHAR 类型存储在系统数据库 mo_catalog 下的 mo_account 表中。

- SUSPEND：暂停某个租户的服务，即暂停后该租户不能再访问 MatrixOne；正在访问租户的用户仍然可以继续访问，关闭会话后，将不能再访问 MatrixOne。
- OPEN：恢复某个暂停状态的租户，恢复后该租户将正常访问 MatrixOne。

comment

租户注释作为 VARCHAR 类型存储在系统数据库 mo_catalog 下的 mo_account 表中。

COMMENT 可以是任意引用的文本，新的 COMMENT 替换任何现有的用户注释。如下所示：

```
mysql> desc mo_catalog.mo_account;
+---------------+-------------+------+------+---------+-------+--------
-+
| Field | Type | Null | Key | Default | Extra | Comment |
+---------------+-------------+------+------+---------+-------+--------
-+
| account_id | INT | YES | | NULL | | |
| account_name | VARCHAR(300) | YES | | NULL | | |
| status | VARCHAR(300) | YES | | NULL | | |
| created_time | TIMESTAMP | YES | | NULL | | |
| comments | VARCHAR(256) | YES | | NULL | | |
| suspended_time | TIMESTAMP | YES | | null | | |
+---------------+-------------+------+------+---------+-------+--------
-+
6 rows in set (0.06 sec)
```

示例

- 示例 1：修改租户信息

```
-- 创建一个名为 "root1" 密码为 "111" 租户
mysql> create account acc1 admin_name "root1" identified by "111";
Query OK, 0 rows affected (0.42 sec)
-- 将租户的初始密码 "111" 修改为 "1234"
mysql> alter account acc1 admin_name "root1" identified by "1234";
Query OK, 0 rows affected (0.01 sec)
-- 修改租户 "root1" 的备注
mysql> alter account acc1 comment "new accout";
Query OK, 0 rows affected (0.02 sec)
-- 查看验证是否给租户 "root1" 增加了 "new accout" 的备注
mysql> show accounts;
+--------------+------------+---------------------+--------+-------------
---+----------+-------------+-----------+-------+----------------+
| account_name | admin_name | created | status | suspended_time |
db_count | table_count | row_count | size | comment |
+--------------+------------+---------------------+--------+-------------
---+----------+-------------+-----------+-------+----------------+
| acc1 | root1 | 2023-02-15 06:26:51 | open | NULL | 5 | 34 | 787 | 0.036
| new accout |
| sys | root | 2023-02-14 06:58:15 | open | NULL | 8 | 57 | 3767 | 0.599
| system account |
+--------------+------------+---------------------+--------+-------------
---+----------+-------------+-----------+-------+----------------+
3 rows in set (0.19 sec)
```

- 示例 2：修改租户状态

```
-- 创建一个名为 "root1" 密码为 "111" 租户
mysql> create account accx admin_name "root1" identified by "111";
Query OK, 0 rows affected (0.27 sec)
-- 修改租户状态为 "suspend", 即暂停用户访问 MatrixOne
mysql> alter account accx suspend;
Query OK, 0 rows affected (0.01 sec)
-- 查看一下是否修改状态成功
mysql> show accounts;
+--------------+------------+---------------------+---------+------------
---------+----------+-------------+-----------+-------+----------------+
| account_name | admin_name | created | status | suspended_time |
db_count | table_count | row_count | size | comment |
+--------------+------------+---------------------+---------+------------
---------+----------+-------------+-----------+-------+----------------+
| accx | root1 | 2023-02-15 06:26:51 | suspend | 2023-02-15 06:27:15 | 5
| 34 | 787 | 0.036 | new accout |
| sys | root | 2023-02-14 06:58:15 | open | NULL | 8 | 57 | 3767 | 0.599
| system account |
+--------------+------------+---------------------+---------+------------
---------+----------+-------------+-----------+-------+----------------+
2 rows in set (0.15 sec)
```

CREATE ROLE

语法说明

在系统中创建一个新角色。

创建角色后，可以将权限授予该角色，然后再将该角色授予其他角色或单个用户。

语法结构

```
> CREATE ROLE [IF NOT EXISTS] role [, role ] ...
```

示例

```
> create role rolex;
Query OK, 0 rows affected (0.02 sec)
```

CREATE USER

语法说明

在系统中创建一个新的用户。

使用 `CREATE USER`，你需要拥有 `CREATE USER` 权限。

- 默认拥有 `CREATE USER` 权限的角色为 MOADMIN 或 ACCOUNTADMIN：集群管理员（默认账户为 root）和由集群管理员创建的租户管理员默认拥有权限。

语法结构

```
> CREATE USER [IF NOT EXISTS]
user auth_option [, user auth_option] ...
[DEFAULT ROLE role]
[COMMENT 'comment_string' | ATTRIBUTE 'json_object']
auth_option: {
IDENTIFIED BY 'auth_string'
}
```

语法说明

首次创建的用户没有权限，默认角色为 `NONE`。要分配权限或角色，请使用 GRANT 语句。

`CREAT USER` 的基本 SQL 语句如下：

```
create user user_name identified by 'password';
```

IDENTIFIED BY auth_string

`CREATE USER` 允许这些 `auth_option`：

- 'auth_string'：在 MatrixOne 中，'auth_string' 为密码，即将密码存储在 *mo_user* 系统表的帐户行中。

DEFAULT ROLE

`DEFAULT ROLE` 子句定义当用户连接到 MatrixOne 并进行身份验证时，或者当用户在会话期间执行 `SET ROLE` 语句时，角色会变为激活/使用状态。

```
create user user_name identified by 'password' default role
role_rolename;
```

`DEFAULT ROLE` 子句允许列出一个或多个以逗号分隔的角色名称。这些角色必须在执行 `CREATE USER` 前就已经被创建好；否则该语句会引发错误，并且创建用户失败。

示例

```
> create user userx identified by '111';
Query OK, 0 rows affected (0.04 sec)
```

限制

MatrxiOne 暂不支持 `CREAT USER COMMENT` 和 `CREAT USER ATTRIBUTE`。

DROP ACCOUNT

语法说明

将指定的租户从某个集群成员中移除。

语法结构

```
> DROP ACCOUNT [IF EXISTS] account
```

示例

```
> drop account if exists tenant_test;
Query OK, 0 rows affected (0.12 sec)
```

> **注意**
>
> 如果租户正在会话中，当租户被移除，会话随即断开，无法再连接 MatrixOne。

DROP USER

语法说明

将指定的用户从系统中移除。

语法结构

```
> DROP USER [IF EXISTS] user [, user] ...
```

示例

```
> drop user if exists userx;
Query OK, 0 rows affected (0.02 sec)
```

> **注意**
>
> 如果用户正在会话中，当用户被移除，会话随即断开，无法再连接 MatrixOne。

DROP ROLE

语法说明

将指定的角色从系统中移除。

语法结构

```
> DROP ROLE [IF EXISTS] role [, role ] ...
```

示例

```
> drop role if exists rolex;
Query OK, 0 rows affected (0.02 sec)
```

> **注意**
>
> 如果使用这个角色的用户正在会话中，当角色被移除，会话随即断开，无法再使用这个角色进行操作。

GRANT

语法说明

`GRANT` 语句将权限和角色分配给 MatrixOne 用户和角色。

GRANT 概述

系统权限是初始系统租户管理员（对应的是 *root* 用户）的权限。系统租户管理员可以创建和删除其他*租户（Accounts）*，管理*租户（Accounts）*。系统租户管理员不能管理其他*租户（Accounts）*名下的资源。

要使用 `GRANT` 授予其他用户或角色权限，你首先必须具有 `WITH GRANT OPTION` 权限，并且你必须具有你正在授予的权限。了解你当前角色的授权情况或其他角色的授权情况，请使用 `SHOW GRANTS` 语句，更多信息，参见 SHOW GRANTS。

`REVOKE` 语句与 `GRANT` 相关，允许租户删除用户权限。有关 `REVOKE` 的更多信息，请参阅 REVOKE。

一般情况下，一个集群默认有一个 *root*，*root* 首先使用 `CREATE ACCOUNT` 创建一个新账户，并定义它的非特权权限，例如它的密码，然后租户使用 `CREATE USER` 创建用户并使用 `GRANT` 对其赋权。`ALTER ACCOUNT` 可用于修改现有租户的非特权特征。`ALTER USER` 用于修改现有用户的权限特征。如需了解 MatrixOne 支持的权限以及不同层级的权限，请参阅 MatrixOne 权限分类。

`GRANT` 在成功执行后，得到结果 `Query OK, 0 rows affected`。要查看操作产生的权限，请使用 SHOW GRANTS

语法结构

```
> GRANT
priv_type [(column_list)]
[, priv_type [(column_list)]] ...
ON [object_type] priv_level
TO user_or_role [, user_or_role] ...

GRANT role [, role] ...
TO user_or_role [, user_or_role] ...
[WITH ADMIN OPTION]

object_type: {
TABLE
| FUNCTION
| PROCEDURE
}

priv_level: {
*
| *.*
| db_name.*
| db_name.tbl_name
| tbl_name
| db_name.routine_name
}
```

参数释义

`GRANT` 语句允许*租户（Accounts）*授予权限和角色，这些权限和角色可以授予用户和角色。语法使用说明如下：

- `GRANT` 不能在同一语句中同时授予权限和角色。
- `ON` 子句区分语句是否授予特权或角色：
  - 使用 `ON`，该语句授予权限。
  - 如果没有 `ON`，则该语句授予角色。
  - 必须使用单独的 `GRANT` 语句将权限和角色分配给一个用户，每个 `GRANT` 语句的语法都与要授予的内容相适应。

数据库权限

数据库权限适用于给定数据库中的所有对象。要分配数据库级权限，请使用 `ON db_name *` 语法，示例如下：

```
grant all on database * to role1;
```

表权限

表权限适用于给定表中的所有列。要分配表级权限，请使用 `ON db_name.tbl_name` 语法，示例如下：

```
grant all on table *.* to role1;
```

授权角色

不携带 `ON` 子句的 `GRANT` 语法将赋权给角色，而不是赋权给个人。角色是权限的命名集合。示例如下：

```
grant role3 to role_user;
```

要授权给角色或者要授权给用户，必须确保用户和角色都存在。

授予角色需要这些权限：

- 你有权向用户或角色授予或撤销任何角色。

示例

```
> drop user if exists user_prepare_01;
> drop role if exists role_prepare_1;
> create user user_prepare_01 identified by '123456';
> create role role_prepare_1;
> create database if not exists p_db;
> grant create table ,drop table on database *.* to role_prepare_1;
Query OK, 0 rows affected (0.01 sec)

> grant connect on account * to role_prepare_1;
Query OK, 0 rows affected (0.01 sec)

> grant insert,select on table *.* to role_prepare_1;
Query OK, 0 rows affected (0.01 sec)

> grant role_prepare_1 to user_prepare_01;
Query OK, 0 rows affected (0.01 sec)
```

REVOKE

语法说明

将某个用户或者角色上被赋予的权限收回。

语法结构

```
> REVOKE [IF EXISTS]
priv_type [(column_list)]
[, priv_type [(column_list)]] ...
ON object_type priv_level

> REVOKE [IF EXISTS] role [, role ] ...
FROM user_or_role [, user_or_role ] ...
```

示例

```
> CREATE USER mouser IDENTIFIED BY '111';
Query OK, 0 rows affected (0.10 sec)

> CREATE ROLE role_r1;
Query OK, 0 rows affected (0.05 sec)

> GRANT role_r1 to mouser;
Query OK, 0 rows affected (0.04 sec)

> GRANT create table on database * to role_r1;
Query OK, 0 rows affected (0.03 sec)

> SHOW GRANTS for mouser@localhost;
+-------------------------------------------------------+
| Grants for mouser@localhost |
+-------------------------------------------------------+
| GRANT create table ON database * `mouser`@`localhost` |
| GRANT connect ON account `mouser`@`localhost` |
+-------------------------------------------------------+
2 rows in set (0.02 sec)

> REVOKE role_r1 from mouser;
Query OK, 0 rows affected (0.04 sec)

> SHOW GRANT for mouser@localhost;
+-----------------------------------------------+
| Grants for mouser@localhost |
+-----------------------------------------------+
| GRANT connect ON account `mouser`@`localhost` |
+-----------------------------------------------+
1 row in set (0.02 sec)
```

SHOW ACCOUNTS

函数说明

列出为你的账户下创建的租户用户的元信息和统计信息。

函数语法

```
> SHOW ACCOUNTS;
```

租户用户信息详情

列名	信息	类型	数据源头
ACCOUNT_NAME	租户名称	varchar	mo_account
ADMIN_NAME	创建时默认超级管理员名称	varchar	每个租户下的 mo_user 表中
CREATED	创建时间	timestamp	mo_account
STATUS	当前状态，OPEN 或 SUSPENDED	varchar	mo_account
SUSPENDED_TIME	停用时间	timestamp	mo_account
DB_COUNT	数据库数量	bigint unsigned	mo_tables
TABLE_COUNT	表数量	bigint unsigned	mo_tables
ROW_COUNT	总行数	bigint unsigned	sum(mo_table_rows())
SIZE	使用空间总量（MB）	decimal(29,3)	sum(mo_table_size(mt.reldatabase,mt.relname)
COMMENT	创建时的 COMMENT 信息	varchar	mo_account

示例

```
mysql> show accounts;
+--------------+------------+---------------------+--------+-------------
---+----------+-------------+-----------+-------+----------------+
| account_name | admin_name | created | status | suspended_time |
db_count | table_count | row_count | size | comment |
+--------------+------------+---------------------+--------+-------------
---+----------+-------------+-----------+-------+----------------+
| sys | root | 2023-02-14 06:58:15 | open | NULL | 8 | 57 | 2681 | 0.351
| system account |
+--------------+------------+---------------------+--------+-------------
---+----------+-------------+-----------+-------+----------------+
1 row in set (0.14 sec)
```

SHOW DATABASES

函数说明

`SHOW DATABASES` 列出 MatrixOne 上的数据库。`SHOW SCHEMAS` 是 `SHOW DATABASES` 的同义词。

如果存在 `LIKE` 子句，表示需要匹配哪些数据库名。`WHERE` 子句可以使用通用的条件来选择行。

MatrixOne 将数据库展示在数据目录中。

数据库信息也可以从 `INFORMATION_SCHEMA` SCHEMATA 表中获得。

函数语法

```
> SHOW {DATABASES | SCHEMAS}
[LIKE 'pattern' | WHERE expr]
```

示例

```
create database demo_1;

mysql> show databases;
+--------------------+
| Database |
+--------------------+
| mo_task |
| information_schema |
| mysql |
| system_metrics |
| system |
| demo_1 |
| mo_catalog |
+--------------------+
7 rows in set (0.00 sec)
```

SHOW CREATE TABLE

语法说明

以列表的形式展现当前数据库创建的某个表的表结构。

语法结构

```
> SHOW CREATE TABLE tbl_name
```

示例

```
drop table if exists t1;
create table t1(
col1 int comment 'First column',
col2 float comment '"%$^&*()_+@!',
col3 varchar comment 'ZD5lTndyuEzw49gxR',
col4 bool comment ''
);
mysql> show create table t1;
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
+
| Table | Create Table |
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
+
| t1 | CREATE TABLE `t1` (
`col1` INT DEFAULT NULL COMMENT 'First column',
`col2` FLOAT DEFAULT NULL COMMENT '"%$^&*()_+@!',
`col3` VARCHAR(65535) DEFAULT NULL COMMENT 'ZD5lTndyuEzw49gxR',
`col4` BOOL DEFAULT NULL
) |
+-------+-------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
+
1 row in set (0.00 sec)
```

SHOW CREATE VIEW

语法说明

这个语句显示了创建命名视图的 `CREATE VIEW` 语句。

语法结构

```
> SHOW CREATE VIEW view_name
```

示例

```
create table test_table(col1 int, col2 float, col3 bool, col4 Date, col5
varchar(255), col6 text);
create view test_view as select * from test_table;
mysql> show create view test_view;
+-----------+-----------------------------------------------------+
| View | Create View |
+-----------+-----------------------------------------------------+
| test_view | create view test_view as select * from test_table |
+-----------+-----------------------------------------------------+
1 row in set (0.01 sec)
```

SHOW TABLES

语法说明

以列表的形式展现当前数据库创建的所有表。

语法结构

```
> SHOW TABLES [LIKE 'pattern' | WHERE expr | FROM 'pattern' | IN
'pattern']
```

示例

```
> SHOW TABLES;
+---------------+
| name |
+---------------+
| clusters |
| contributors |
| databases |
| functions |
| numbers |
| numbers_local |
| numbers_mt |
| one |
| processes |
| settings |
| tables |
| tracing |
+---------------+
```

SHOW INDEX

语法说明

`SHOW INDEX` returns table index information.

`SHOW INDEX` returns the following fields:

Fields	Description
Table	The name of the table.
Non_unique	0 if the index cannot contain duplicates, 1 if it can.
Key_name	The name of the index. If the index is the primary key, the name is always PRIMARY.
Seq_in_index	The column sequence number in the index, starting with 1.
Column_name	The column name. See also the description for the Expression column.
Collation	How the column is sorted in the index. This can have values A (ascending), D (descending), or NULL (not sorted).
Cardinality	An estimate of the number of unique values in the index. To update this number, run ANALYZE TABLE or (for MyISAM tables) myisamchk -a. Cardinality is counted based on statistics stored as integers, so the value is not necessarily exact even for small tables. The higher the cardinality, the greater the chance that MySQL uses the index when doing joins.
Sub_part	The index prefix. That is, the number of indexed characters if the column is only partly indexed, NULL if the entire column is indexed. **Note:** Prefix limits are measured in bytes. However, prefix lengths for index specifications in CREATE TABLE, ALTER TABLE, and CREATE INDEX statements are interpreted as number of characters for nonbinary string types (CHAR, VARCHAR, TEXT) and number of bytes for binary string types (BINARY, VARBINARY, BLOB). Take this into account when specifying a prefix length for a nonbinary string column that uses a multibyte character set.
Packed	Indicates how the key is packed. NULL if it is not.
Null	Contains YES if the column may contain NULL values and ″ if not.
Index_type	The index method used (BTREE, FULLTEXT, HASH, RTREE).
Comment	Information about the index not described in its own column, such as disabled if the

Fields	Description
	index is disabled.
Visible	Whether the index is visible to the optimizer.
Expression	For a nonfunctional key part, Column_name indicates the column indexed by the key part and Expression is NULL. For a functional key part, Column_name column is NULL and Expression indicates the expression for the key part.

语法结构

```
> SHOW {INDEX | INDEXES}
{FROM | IN} tbl_name
[{FROM | IN} db_name]
```

Explanations

An alternative to tbl_name FROM db_name syntax is db_name.tbl_name.

示例

```
CREATE TABLE show_01(sname varchar(30),id int);
mysql> show INDEX FROM show_01;
+---------+------------+------------+--------------+-------------+-------
----+-------------+----------+--------+------+------------+--------------
----+---------+------------+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation
| Cardinality | Sub_part | Packed | Null | Index_type | Comment | Visible
| Expression |
+---------+------------+------------+--------------+-------------+-------
----+-------------+----------+--------+------+------------+--------------
----+---------+------------+
| show_01 | 0 | id | 1 | id | A | 0 | NULL | NULL | YES | | | YES | NULL
|
| show_01 | 0 | sname | 1 | sname | A | 0 | NULL | NULL | YES | | | YES |
NULL |
| show_01 | 0 | __mo_rowid | 1 | __mo_rowid | A | 0 | NULL | NULL | NO |
| Physical address | NO | NULL |
+---------+------------+------------+--------------+-------------+-------
----+-------------+----------+--------+------+------------+--------------
----+---------+------------+
3 rows in set (0.02 sec)
```

SHOW COLLATION

语法说明

显示 MatrixOne 支持字符集的排序规则。默认情况下，`SHOW COLLATION` 的输出包括所有可用的排序规则。`LIKE` 子句（如果存在）指示要匹配的排序规则名称。`WHERE` 子句可以使用更一般的条件来选择行。

语法结构

```
> SHOW COLLATION
[LIKE 'pattern' | WHERE expr]
```

示例

```
mysql> show collation;
+-------------+---------+------+----------+---------+
| Collation | Charset | Id | Compiled | Sortlen |
+-------------+---------+------+----------+---------+
| utf8mb4_bin | utf8mb4 | 46 | Yes | 1 |
+-------------+---------+------+----------+---------+
1 row in set (0.00 sec)

mysql> show collation like '%';
+-------------+---------+------+----------+---------+
| Collation | Charset | Id | Compiled | Sortlen |
+-------------+---------+------+----------+---------+
| utf8mb4_bin | utf8mb4 | 46 | Yes | 1 |
+-------------+---------+------+----------+---------+
1 row in set (0.00 sec)

mysql> show collation where 'Charset'='utf8mb4';
+-------------+---------+------+----------+---------+
| Collation | Charset | Id | Compiled | Sortlen |
+-------------+---------+------+----------+---------+
| utf8mb4_bin | utf8mb4 | 46 | Yes | 1 |
+-------------+---------+------+----------+---------+
1 row in set (0.00 sec)
```

SHOW COLUMNS

语法说明

`SHOW COLUMNS` 用于显示指定表中有关列的信息。

语法结构

```
> SHOW [FULL] {COLUMNS}
{FROM | IN} tbl_name
[{FROM | IN} db_name]
[LIKE 'pattern' | WHERE expr]
```

示例

```
drop table if exists t1;
create table t1(
col1 int comment 'First column',
col2 float comment '"%$^&*()_+@!',
col3 varchar comment 'ZD5lTndyuEzw49gxR',
col4 bool comment ''
);
mysql> show columns from t1;
+-------+----------------+------+------+---------+-------+---------------
----+
| Field | Type | Null | Key | Default | Extra | Comment |
+-------+----------------+------+------+---------+-------+---------------
----+
| col1 | INT | YES | | NULL | | First column |
| col2 | FLOAT | YES | | NULL | | "%$^&*()_+@! |
| col3 | VARCHAR(65535) | YES | | NULL | | ZD5lTndyuEzw49gxR |
| col4 | BOOL | YES | | NULL | | |
+-------+----------------+------+------+---------+-------+---------------
----+
4 rows in set (0.02 sec)
```

SHOW GRANTS

语法说明

使用 `SHOW GRANTS` 语句显示用户的所有授权信息。`SHOW GRANTS` 语句显示使用 `GRANT` 命令分配给用户的权限。

使用 `SHOW GRANTS` 语句需要拥有查询 *mo_catalog* 库下所有表的 `SELECT` 权限，但显示当前用户的权限和角色除外。

要为 `SHOW GRANTS` 命名帐户或角色，即使用与 `GRANT` 语句相同的格式，例如：

```
show grants for 'root'@'localhost';
```

语法结构

```
> SHOW GRANTS FOR {username[@hostname] | rolename};
```

示例

```
> create role role1;
> grant all on table *.* to role1;
> grant create table, drop table on database *.* to role1;
> create user user1 identified by 'pass1';
> grant role1 to user1;
> show grants for 'user1'@'localhost';
+-----------------------------------------------------------+
| Grants for user1@localhost |
+-----------------------------------------------------------+
| GRANT connect ON account `user1`@`localhost` |
| GRANT table all ON table *.* `user1`@`localhost` |
| GRANT create table ON database *.* `user1`@`localhost` |
| GRANT drop table ON database *.* `user1`@`localhost` |
+-----------------------------------------------------------+
4 rows in set (0.00 sec)
```

限制

当前 MatrixOne 还不支持查看角色权限，即暂不支持 `SHOW GRANTS FOR {rolename}` 。

SHOW VARIABLES

语法说明

以列表的形式展现当前数据库系统变量的值。

语法结构

```
> SHOW VARIABLES
[LIKE 'pattern']
```

语法释义

- `LIKE` 子句仅显示名称与模式匹配的那些变量的行。要获取名称与模式匹配的变量列表，请在子句中使用 % 通配符。

示例

```
mysql> SHOW VARIABLES;
+-------------------------+-----------------------------------------------
-----------------------------------------------------------------------------
-+
| Variable_name | Value |
+-------------------------+-----------------------------------------------
-----------------------------------------------------------------------------
-+
| auto_increment_increment | 1 |
| auto_increment_offset | 1 |
| autocommit | 1 |
| character_set_client | utf8mb4 |
| character_set_connection | utf8mb4 |
| character_set_database | utf8mb4 |
| character_set_results | utf8mb4 |
| character_set_server | utf8mb4 |
| collation_connection | default |
| collation_server | utf8mb4_bin |
| completion_type | NO_CHAIN |
| host | 0.0.0.0 |
| init_connect | |
| interactive_timeout | 28800 |
| license | APACHE |
| lower_case_table_names | 0 |
| max_allowed_packet | 16777216 |
| net_write_timeout | 60 |
| performance_schema | 0 |
| port | 6001 |
| profiling | 0 |
| query_result_maxsize | 100 |
| query_result_timeout | 24 |
| save_query_result | 0 |
| sql_mode |
ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FO
|
| sql_safe_updates | 0 |
| sql_select_limit | 18446744073709551615 |
| system_time_zone | |
| testbotchvar_nodyn | 0 |
| testbothvar_dyn | 0 |
| testglobalvar_dyn | 0 |
| testglobalvar_nodyn | 0 |
```

```
| testsessionvar_dyn | 0 |
| testsessionvar_nodyn | 0 |
| time_zone | SYSTEM |
| transaction_isolation | REPEATABLE-READ |
| transaction_read_only | 0 |
| tx_isolation | REPEATABLE-READ |
| tx_read_only | 0 |
| version_comment | MatrixOne |
| wait_timeout | 28800 |
+-------------------------+-----------------------------------------------
-----------------------------------------------------------------------------
-+
41 rows in set (0.01 sec)

mysql> show variables like 'auto%';
+-------------------------+-------+
| Variable_name | Value |
+-------------------------+-------+
| auto_increment_increment | 1 |
| auto_increment_offset | 1 |
| autocommit | 1 |
+-------------------------+-------+
3 rows in set (0.00 sec)

mysql> show variables like 'auto_increment_increment';
+-------------------------+-------+
| Variable_name | Value |
+-------------------------+-------+
| auto_increment_increment | 1 |
+-------------------------+-------+
1 row in set (0.00 sec)
```

SET ROLE

语法说明

设置会话的活动/当前主要角色。为当前活动的主角色设置上下文，以确定当前用户是否拥有执行 `CREATE <object>` 语句或执行任何其他 SQL 操作所需的权限。

除了创建对象之外，任何 SQL 操作的授权都可以由次级角色执行。

语法结构

```
> SET SECONDARY ROLE {
NONE
| ALL
}
SET ROLE role
```

语法说明

角色是权限的集合，一个用户可以对应多个角色。

例如，user1 拥有主要角色 role1，次要角色 role2 和 role3, role1 被授予 pri1 和 pri2 权限；role2 被赋予权限 pri3；role3 被赋予权限 pri4，授权示例表如下：

用户名	角色名	权限名
user1	role1	pri1,pri2
	role2	pri3
	role3	pri4

为了更容易理解，你可以参考如下示例：

用户	角色	权限名
Tom	应用开发者（Application Developer）	读数据（Read Data），写数据（Write Data）
	运维专家（O&M expert）	读数据（Read data）

用户	角色	权限名
	数据库管理员（Database Administrator）	管理员权限（Administrator Privileges）

此时 Tom 的主要角色是应用开发者，Tom 需要调用*管理员权限*，那么 Tom 可以使用以下两种方法：

—使用 `SET role role` 语句将其角色切换为 “数据库管理员”。

—如果需要使用主、从角色的所有权限，可以使用 `SET secondary ROLE all`。

这两种语句解释如下:

SET SECONDARY ROLE ALL

将该用户所有的 ROLE 取并集。

SET SECONDARY ROLE NONE

将除 PRIMARY ROLE 之外的所有角色从当前会话中去除。

SET ROLE role

将当前角色切换为新角色。

示例

```
> drop role if exists
use_role_1,use_role_2,use_role_3,use_role_4,use_role_5;
> drop user if exists use_user_1,use_user_2;
> drop database if exists use_db_1;
> create role use_role_1,use_role_2,use_role_3,use_role_4,use_role_5;
> create database use_db_1;
> create user use_user_1 identified by '123456' default role use_role_1;
#把所有表的 select, insert 和 update 权限授权给 use_role_1
> grant select ,insert ,update on table *.* to use_role_1;
#把数据库的所有权限授权给 use_role_2
> grant all on database * to use_role_2;
#把角色 use_role_2 分配给用户 use_user_1
> grant use_role_2 to use_user_1;
#创建表 use_table_1
> create table use_db_1.use_table_1(a int,b varchar(20),c double );
#设置用户 use_user_1 主要角色和次要角色全部可用
> set secondary role all;
#查看用户 use_user_1 现在拥有的权限
> show grants for 'use_user_1'@'localhost';
+-----------------------------------------------------------+
| Grants for use_user_1@localhost |
+-----------------------------------------------------------+
| GRANT select ON table *.* `use_user_1`@`localhost` |
| GRANT insert ON table *.* `use_user_1`@`localhost` |
| GRANT update ON table *.* `use_user_1`@`localhost` |
| GRANT connect ON account `use_user_1`@`localhost` |
| GRANT database all ON database * `use_user_1`@`localhost` |
+-----------------------------------------------------------+
5 rows in set (0.01 sec)
#可以看到, 用户 use_user_1 拥有默认的连接 MatrixOne 的权限 connect; 也拥有对所有表的 select, insert 和 update 权限, 同时也拥有对数据库的全部权限
```

USE

语法说明

`USE` 语句用于选择当前数据库，在此数据库进行后续操作。

语法结构

```
> USE db_name
```

示例

```
> USE db1;
> SELECT COUNT(*) FROM mytable;
```

Prepared 语句概述

MatrixOne 提供对服务器端预处理语句的支持。利用客户端或服务器二进制协议的高效性，对参数值使用带有占位符的语句进行预处理，执行过程中的优点如下：

- 每次执行语句时解析语句的效率提高。通常，数据库应用程序处理大量几乎相同的语句，只更改子句中的文字或变量值，例如用于查询和删除的 `WHERE`、用于更新的 `SET` 和用于插入的 `VALUES`。
- 防止 SQL 注入。参数值可以包含未转义的 SQL 引号和分隔符，一次编译，多次运行，省去了解析优化等过程。

`PREPARE`、`EXECUTE`、和 `DEALLOCATE PREPARE` 语句

PREPARE 语句的 SQL 基本语法主要为以下三种 SQL 语句：

- PREPARE：执行预编译语句。
- EXECUTE：执行已预编译的句。
- DEALLOCATE PREPARE：释放一条预编译的语句。

PREPARE

语法说明

`PREPARE` 语句准备一条 SQL 语句并给它分配一个名称。

准备好的语句用 `EXECUTE` 执行，用 `DEALLOCATE PREPARE` 释放。

SQL 语句的命名不区分大小写。

语法结构

```
PREPARE stmt_name FROM preparable_stmt
```

参数释义

参数	说明
stmt_name	预编译的 SQL 语句的名称
preparable_stmt	包含 SQL 语句文本的字符串文字或用户变量。文本必须代表单个语句，而不是多个语句。在声明中，? 字符可用作参数标记，以指示稍后在执行查询时将数据值绑定到查询的位置。? 字符不包含在引号内，即使你打算将它们绑定到字符串值。参数标记只能用于应出现数据值的地方，不能用于 SQL 关键字、标识符等。

如果已存在具有给定名称的预编译语句，则在准备新的语句之前需要将其隐式释放。

预编译语句创建时需要了解以下几点：

- 在一个会话中创建的预编译语句不可用于其他会话。
- 在一个会话中创建的准备好的语句不适用于其他会话。
- 当会话结束时，无论是正常还是异常，其预编译的语句都不再存在。如果启用了自动重新连接，也不会通知客户端连接丢失。因此，客户端可以禁用自动重新连接。

在预编译的语句中使用的参数需要再首次准备语句时确定其类型，并且在使用 `EXECUTE` 运行预编译语句时保留参数类型。以下内容列出了确定参数类型的规则：

- 二元算术运算符的两个操作数的参数数据类型需一致。
- 如果二元算术运算符的两个操作数都是形式参数，则参数类型由运算符的上下文决定。
- 如果一元算术运算符的操作数是形式参数，则参数类型由运算符的上下文决定。
- 如果算术运算符没有上下文来确定操作数参数类型，则相关参数的的派生类型都是 `DOUBLE PRECISION`。例如，当参数是 `SELECT` 列表中的顶级节点，或者当它是比较运算符的一部分时，那么它相关参数的的派生类型都是 `DOUBLE PRECISION`。
- 字符串运算符的操作数的形式参数具有与其他操作数的聚合类型相同的派生类型。如果运算符的所有操作数都是形式参数，则派生类型为 `VARCHAR`，其排序规则由 `collation_connection` 的值决定。
- 参数为时间操作符的操作数的形式参数，如果操作符返回 `DATETIME`，则参数类型为 `DATETIME`；如果操作符返回 `TIME`，则参数类型为 `TIME`；如果操作符返回 `DATE`，则参数类型为 `DATE`。
- 二元比较运算符的操作数的两个参数的派生类型一致。
- 参数为三元比较运算符的操作数的形式参数，例如 `BETWEEN` 运算中的参数，它们的派生类型与其他操作数的聚合类型相同。
- 如果比较运算符的所有操作数都是形式参数，那么每个操作数的派生类型都是 `VARCHAR`，他们的排序规则由 `collation_connection` 的值确定。
- `CASE`、`COALESCE`、`IF`、`IFNULL` 或 `NULLIF` 中的任意一个输出操作数的形式参数，其派生类型与操作符的其他输出操作数的聚合类型相同。
- 如果 `CASE`、`COALESCE`、`IF`、`IFNULL` 或 `NULLIF` 的所有输出操作数都是形式参数，或者它们都是 `NULL`，则参数的类型由操作符的上下文决定。

— 如果参数是 `CASE`、`COALESCE()`、`IF` 或 `IFNULL` 中的任意一个操作数，并且操作符上下文不能确定其参数类型，则每个参数的派生类型都是 `VARCHAR`，其排序规则由 `collation_connection` 的值决定。

— `CAST()` 操作数的形式参数与 `CAST()` 指定的类型相同。

— 如果一个形式参数是 `SELECT` 列表的直接成员，而不是 `INSERT` 语句的一部分，则形式参数的派生类型为 `VARCHAR`，其排序规则由 `collation_connection` 的值决定。

— 如果形式参数是 `INSERT` 语句的一部分 `SELECT` 列表的直接成员，则形式参数的派生类型为插入形式参数的对应列的类型。

- 如果一个形式参数被用作 `UPDATE` 语句的 `SET` 子句或 `INSERT` 语句的 `ON DUPLICATE KEY UPDATE` 子句中的赋值源，形式参数的派生类型是由 `SET` 或 `ON DUPLICATE KEY UPDATE` 子句更新的对应列的类型。
- 如果形式参数是函数的实际参数，则其派生类型取决于函数的返回类型。

对于实际类型和派生类型的某些组合，会触发自动重新准备预编译语句。以下情况，无需重新准备预编译语句：

- `NULL` 为实际参数。
- 形式参数是 `CAST()` 的操作数。(`CAST()` 会尝试将参数转换到派生类型，如果转换失败则会引发异常。)
- 参数类型为字符串。(在本例中，隐式执行 `CAST(? AS derived_type)`)。
- 参数的派生类型和实际类型均为 `INTEGER`，且具有相同的符号。
- 参数的派生类型为 `DECIMAL`，实际类型为 `DECIMAL` 或 `INTEGER`。
- 参数的派生类型为 `DOUBLE`，实际类型为任意数字类型。
- 参数的派生类型和实际类型都是字符串类型。
- 参数派生类型是时态类型，实际类型是时态类型。异常情况：参数的派生类型是 `TIME`，而实际类型不是 `TIME`；参数的派生类型是 `DATE`，而实际类型不是 `DATE`。
- 派生类型是时间类型，实际类型是数字类型。

除上述情况以外，需要重新准备预编译语句并使用实际参数类型，不能使用派生参数类型。

这些规则也适用于在预编译语句中引用的用户变量。

在预编译语句中为给定参数或用户变量时，如果在第一次执行时使用不同的数据类型，则会导致重新准备该预编译语句，不但运行效率低，而且还可能导致参数（或变量）的实际类型发生变化，实际执行结果与语句的预期结果不一致。建议在预编译语句中对给定参数使用相同的数据类型。

示例

```
> create table t13 (a int primary key);
> insert into t13 values (1);
> select * from t13 where 3 in (select (1+1) union select 1);
Empty set (0.01 sec)

> select * from t13 where 3 in (select (1+2) union select 1);
+------+
| a |
+------+
| 1 |
+------+
1 row in set (0.01 sec)

> prepare st_18492 from 'select * from t13 where 3 in (select (1+1) union
select 1)';
Query OK, 0 rows affected (0.00 sec)

> execute st_18492;
Empty set (0.01 sec)

> prepare st_18493 from 'select * from t13 where 3 in (select (2+1) union
select 1)';
Query OK, 0 rows affected (0.00 sec)

> execute st_18493;
+------+
| a |
+------+
| 1 |
+------+
1 row in set (0.00 sec)

> deallocate prepare st_18492;
Query OK, 0 rows affected (0.00 sec)

> deallocate prepare st_18493;
Query OK, 0 rows affected (0.00 sec)
```

EXECUTE

语法说明

`EXECUTE` 语句的作用是：使用 `PREPARE` 准备好一条语句后，可以使用 `EXECUTE` 语句引用预编译的语句名称并执行。如果预编译的语句包含任何参数标记，则必须提供一个 `USING` 子句，该子句列出包含要绑定到参数的值的用户变量。参数值只能由用户变量提供，并且 `USING` 子句必须命名与语句中参数标记的数量一样多的变量。

你可以多次执行给定的预编译语句，将不同的变量传递给它，或者在每次执行之前将变量设置为不同的值。

语法结构

```
EXECUTE stmt_name
[USING @var_name [, @var_name] ...]
```

参数释义

参数	说明
stmt_name	预编译的 SQL 语句的名称

示例

```
> CREATE TABLE numbers(pk INTEGER PRIMARY KEY, ui BIGINT UNSIGNED, si
BIGINT);
> INSERT INTO numbers VALUES (0, 0, -9223372036854775808), (1,
18446744073709551615, 9223372036854775807);
> SET @si_min = -9223372036854775808;
> SET @si_max = 9223372036854775807;
> PREPARE s2 FROM 'SELECT * FROM numbers WHERE si=?';
Query OK, 0 rows affected (0.00 sec)

> EXECUTE s2 USING @si_min;
+------+------+----------------------+
| pk | ui | si |
+------+------+----------------------+
| 0 | 0 | -9223372036854775808 |
+------+------+----------------------+
1 row in set (0.01 sec)

> EXECUTE s2 USING @si_max;
+------+----------------------+---------------------+
| pk | ui | si |
+------+----------------------+---------------------+
| 1 | 18446744073709551615 | 9223372036854775807 |
+------+----------------------+---------------------+
1 row in set (0.01 sec)

> DEALLOCATE PREPARE s2;
Query OK, 0 rows affected (0.00 sec)
```

DEALLOCATE PREPARE

语法说明

`DEALLOCATE PREPARE` 语句的作用是释放使用 `PREPARE` 生成的预编译语句。在释放预编译语句后，再次执行预编译的语句会导致错误。若创建了过多预编译的语句并且没有使用 `DEALLOCATE PREPARE` 语句进行释放，那么系统变量会强制执行预编译语句上限 `max_prepared_stmt_count` 提示。

语法结构

```
{DEALLOCATE | DROP} PREPARE stmt_name
```

参数释义

参数	说明
stmt_name	预编译的 SQL 语句的名称

示例

```
> CREATE TABLE numbers(pk INTEGER PRIMARY KEY, ui BIGINT UNSIGNED, si
BIGINT);
> INSERT INTO numbers VALUES (0, 0, -9223372036854775808), (1,
18446744073709551615, 9223372036854775807);
> SET @si_min = -9223372036854775808;
> SET @si_max = 9223372036854775807;
> PREPARE s2 FROM 'SELECT * FROM numbers WHERE si=?';
Query OK, 0 rows affected (0.00 sec)

> EXECUTE s2 USING @si_min;
+------+------+----------------------+
| pk | ui | si |
+------+------+----------------------+
| 0 | 0 | -9223372036854775808 |
+------+------+----------------------+
1 row in set (0.01 sec)

> EXECUTE s2 USING @si_max;
+------+----------------------+---------------------+
| pk | ui | si |
+------+----------------------+---------------------+
| 1 | 18446744073709551615 | 9223372036854775807 |
+------+----------------------+---------------------+
1 row in set (0.01 sec)

> DEALLOCATE PREPARE s2;
Query OK, 0 rows affected (0.00 sec)
```

EXPLAIN

EXPLAIN — 展示一个语句的执行计划。

语法结构

```
EXPLAIN [ ( option [, ...] ) ] statement

where option can be one of:
ANALYZE [ boolean ]
VERBOSE [ boolean ]
FORMAT { TEXT | JSON }
```

语法描述

此命令主要作用是显示出 MatrixOne 计划程序为提供的语句生成的执行计划。执行计划显示了如何通过普通顺序扫描、索引扫描等方式扫描语句引用的表，如果引用了多个表，将使用什么连接算法将每个输入表中所需的行聚集在一起。

显示的最关键部分是估计语句执行成本，即计划程序将估计运行语句所需时间 (以任意一种成本单位衡量，但通常是听过磁盘页获取)。实际上这里显示了两个数字：返回第一行之前的启动成本，以及返回所有行的总成本。对于大多数查询来说，总成本是最重要的，但在 `EXISTS` 中的子查询中，计划程序会选择最小的启动成本，而不是最小的总成本 (因为执行者在获得一行之后就会停止)。此外，如果您使用 `LIMIT` 从句限制返回的行数，计划程序将在端点成本之间进行适当的插值，以便估计哪个计划真正是最便宜的。

`ANALYZE` 子句语法选项为语句实际执行，而不仅仅是计划执行，然后将实际运行时统计信息添加到显示中，包括每个计划节点中花费的总运行时间 (以毫秒为单位) 和实际返回的行总数。这有助于了解规划者的期望是否接近实际。

参数释义

- ANALYZE:

执行该命令并显示实际运行时和其他统计数据。该参数默认为 `FALSE` 。

- VERBOSE:

`VERBOSE` 用作显示有关计划的其他信息。具体来说，包括计划树中每个节点的输出列列表、模式限定表和函数名称，始终使用范围表别名标记表达式中的变量，并且始终打印显示统计信息的每个触发器的名称。该参数默认为 `FALSE`。

- FORMAT:

`FORMAT` 用作指定输出格式，可以是 *TEXT*、*JSON*。非文本输出包含与文本输出格式相同的信息，且容易被程序解析。该参数默认为 `TEXT`。

- BOOLEAN:

`BOOLEAN` 指定所选选项是打开还是关闭。你可以写 `TRUE` 来启用该选项，或者写 `FALSE` 来禁用它。

`boolean` 值也可以省略，省略 `boolean` 值的情况下默认为 `TRUE`。

- STETEMENT

MatrixOne 支持任何 `SELECT`， `UPDATE`， `DELETE` 语句执行计划。在 MatrixOne 0.5.1 版本中仅支持 `INSERT` 语句类型中的 `INSERT INTO..SELECT` 语句，暂不支持 `INSERT INTO...VALUES` 语句。

示例

Node_TABLE_SCAN

```
mysql> explain verbose SELECT N_NAME, N_REGIONKEY a FROM NATION WHERE
N_NATIONKEY > 0 OR N_NATIONKEY < 10;
+-----------------------------------------------------------------------
------------+
| QUERY PLAN |
+-----------------------------------------------------------------------
------------+
| Project(cost=0.00..0.00 card=25.00 ndv=0.00 rowsize=0) |
| Output: #[0,0], #[0,1] |
| -> Table Scan on db1.nation(cost=0.00..0.00 card=25.00 ndv=0.00
rowsize=0) |
| Output: #[0,1], #[0,2] |
| Table: 'nation' (0:'n_nationkey', 1:'n_name', 2:'n_regionkey') |
| Filter Cond: ((CAST(#[0,0] AS INT64) > 0) or (CAST(#[0,0] AS INT64) <
10)) |
+-----------------------------------------------------------------------
------------+
```

Node_VALUE_SCAN

```
mysql> explain verbose select abs(-1);
+-----------------------------------------------------------------------
-----+
| QUERY PLAN |
+-----------------------------------------------------------------------
-----+
| Project(cost=0.00..0.00 card=1.00 ndv=0.00 rowsize=0) |
| Output: 1 |
| -> Values Scan "*VALUES*" (cost=0.00..0.00 card=1.00 ndv=0.00
rowsize=0) |
| Output: 0 |
+-----------------------------------------------------------------------
-----+
```

Node_SORT

```
mysql> explain verbose SELECT N_NAME, N_REGIONKEY a FROM NATION WHERE
N_NATIONKEY > 0 AND N_NATIONKEY < 10 ORDER BY N_NAME, N_REGIONKEY DESC;
+-----------------------------------------------------------------------
--------------------+
| QUERY PLAN |
+-----------------------------------------------------------------------
--------------------+
| Project(cost=0.00..0.00 card=25.00 ndv=0.00 rowsize=0) |
| Output: #[0,0], #[0,1] |
| -> Sort(cost=0.00..0.00 card=25.00 ndv=0.00 rowsize=0) |
| Output: #[0,0], #[0,1] |
| Sort Key: #[0,0] INTERNAL, #[0,1] DESC |
| -> Project(cost=0.00..0.00 card=25.00 ndv=0.00 rowsize=0) |
| Output: #[0,0], #[0,1] |
| -> Table Scan on db1.nation(cost=0.00..0.00 card=25.00 ndv=0.00
rowsize=0) |
| Output: #[0,1], #[0,2] |
| Table: 'nation' (0:'n_nationkey', 1:'n_name', 2:'n_regionkey') |
| Filter Cond: (CAST(#[0,0] AS INT64) > 0), (CAST(#[0,0] AS INT64) < 10)
|
+-----------------------------------------------------------------------
--------------------+
```

带有限制和偏移量：

```
mysql> explain SELECT N_NAME, N_REGIONKEY FROM NATION WHERE
abs(N_REGIONKEY) > 0 AND N_NAME LIKE '%AA' ORDER BY N_NAME DESC,
N_REGIONKEY limit 10;
+-----------------------------------------------------------------------
-------------------+
| QUERY PLAN |
+-----------------------------------------------------------------------
-------------------+
| Project(cost=0.00..0.00 card=25.00 ndv=0.00 rowsize=0) |
| -> Sort(cost=0.00..0.00 card=25.00 ndv=0.00 rowsize=0) |
| Sort Key: #[0,0] DESC, #[0,1] INTERNAL |
| Limit: 10 |
| -> Project(cost=0.00..0.00 card=25.00 ndv=0.00 rowsize=0) |
| -> Table Scan on db1.nation(cost=0.00..0.00 card=25.00 ndv=0.00
rowsize=0) |
| Filter Cond: (abs(CAST(#[0,1] AS INT64)) > 0), (#[0,0] like '%AA') |
+-----------------------------------------------------------------------
-------------------+
```

```
mysql> explain SELECT N_NAME, N_REGIONKEY FROM NATION WHERE
abs(N_REGIONKEY) > 0 AND N_NAME LIKE '%AA' ORDER BY N_NAME DESC,
N_REGIONKEY LIMIT 10 offset 20;
+-----------------------------------------------------------------------
-------------------+
| QUERY PLAN |
+-----------------------------------------------------------------------
-------------------+
| Project(cost=0.00..0.00 card=25.00 ndv=0.00 rowsize=0) |
| -> Sort(cost=0.00..0.00 card=25.00 ndv=0.00 rowsize=0) |
| Sort Key: #[0,0] DESC, #[0,1] INTERNAL |
| Limit: 10, Offset: 20 |
| -> Project(cost=0.00..0.00 card=25.00 ndv=0.00 rowsize=0) |
| -> Table Scan on db1.nation(cost=0.00..0.00 card=25.00 ndv=0.00
rowsize=0) |
| Filter Cond: (abs(CAST(#[0,1] AS INT64)) > 0), (#[0,0] like '%AA') |
+-----------------------------------------------------------------------
-------------------+
```

Node_AGG

```
mysql> explain verbose SELECT count(*) FROM NATION group by N_NAME;
+-----------------------------------------------------------------------
-------------+
| QUERY PLAN |
+-----------------------------------------------------------------------
-------------+
| Project(cost=0.00..0.00 card=25.00 ndv=0.00 rowsize=0) |
| Output: #[0,0] |
| -> Aggregate(cost=0.00..0.00 card=25.00 ndv=0.00 rowsize=0) |
| Output: #[-2,0] |
| Group Key:#[0,1] |
| Aggregate Functions: starcount(#[0,0]) |
| -> Table Scan on db1.nation(cost=0.00..0.00 card=25.00 ndv=0.00
rowsize=0) |
| Output: #[0,0], #[0,1] |
| Table: 'nation' (0:'n_nationkey', 1:'n_name') |
+-----------------------------------------------------------------------
-------------+
```

Node_JOIN

```
mysql> explain verbose SELECT NATION.N_NAME, REGION.R_NAME FROM NATION
join REGION on NATION.N_REGIONKEY = REGION.R_REGIONKEY WHERE
NATION.N_REGIONKEY > 10 AND LENGTH(NATION.N_NAME) >
LENGTH(REGION.R_NAME);
+-----------------------------------------------------------------------
--------------------+
| QUERY PLAN |
+-----------------------------------------------------------------------
--------------------+
| Project(cost=0.00..0.00 card=125.00 ndv=0.00 rowsize=0) |
| Output: #[0,1], #[0,0] |
| -> Filter(cost=0.00..0.00 card=125.00 ndv=0.00 rowsize=0) |
| Output: #[0,0], #[0,1] |
| Filter Cond: (length(CAST(#[0,1] AS CHAR)) > length(CAST(#[0,0] AS
CHAR))) |
| -> Join(cost=0.00..0.00 card=125.00 ndv=0.00 rowsize=0) |
| Output: #[0,1], #[1,0] |
| Join Type: INNER |
| Join Cond: (#[1,1] = #[0,0]) |
| -> Table Scan on tpch.region(cost=0.00..0.00 card=5.00 ndv=0.00
rowsize=0) |
| Output: #[0,0], #[0,1] |
| Table: 'region' (0:'r_regionkey', 1:'r_name') |
| -> Table Scan on tpch.nation(cost=0.00..0.00 card=25.00 ndv=0.00
rowsize=0) |
| Output: #[0,0], #[0,1] |
| Table: 'nation' (0:'n_name', 1:'n_regionkey') |
| Filter Cond: (CAST(#[0,1] AS INT64) > 10) |
+-----------------------------------------------------------------------
--------------------+
```

Node_INSERT

```
mysql> explain verbose INSERT NATION select * from nation;
+-----------------------------------------------------------------------
--------------------+
| QUERY PLAN |
+-----------------------------------------------------------------------
--------------------+
| Insert on db1.nation (cost=0.0..0.0 rows=0 ndv=0 rowsize=0) |
| Output: #[0,0], #[0,1], #[0,2], #[0,3] |
| -> Project(cost=0.00..0.00 card=25.00 ndv=0.00 rowsize=0) |
| Output: #[0,0], #[0,1], #[0,2], #[0,3] |
| -> Table Scan on db1.nation(cost=0.00..0.00 card=25.00 ndv=0.00
rowsize=0) |
| Output: #[0,0], #[0,1], #[0,2], #[0,3] |
| Table: 'nation' (0:'n_nationkey', 1:'n_name', 2:'n_regionkey',
3:'n_comment') |
+-----------------------------------------------------------------------
--------------------+
7 rows in set (0.00 sec)
```

Node_Update

```
mysql> explain verbose UPDATE NATION SET N_NAME ='U1', N_REGIONKEY=2
WHERE N_NATIONKEY > 10 LIMIT 20;
+-----------------------------------------------------------------------
-------------+
| QUERY PLAN |
+-----------------------------------------------------------------------
-------------+
| Update on db1.nation (cost=0.0..0.0 rows=0 ndv=0 rowsize=0) |
| -> Project(cost=0.00..0.00 card=25.00 ndv=0.00 rowsize=0) |
| Output: #[0,0], 'U1', CAST(2 AS INT32) |
| Limit: 20 |
| -> Table Scan on db1.nation(cost=0.00..0.00 card=25.00 ndv=0.00
rowsize=0) |
| Output: #[0,1] |
| Table: 'nation' (0:'n_nationkey', 1:'PADDR') |
| Filter Cond: (CAST(#[0,0] AS INT64) > 10) |
+-----------------------------------------------------------------------
-------------+
```

Node_Delete

```
mysql> explain verbose DELETE FROM NATION WHERE N_NATIONKEY > 10;
+-----------------------------------------------------------------------
-------------+
| QUERY PLAN |
+-----------------------------------------------------------------------
-------------+
| Delete on db1.nation (cost=0.0..0.0 rows=0 ndv=0 rowsize=0) |
| -> Project(cost=0.00..0.00 card=25.00 ndv=0.00 rowsize=0) |
| Output: #[0,0] |
| -> Table Scan on db1.nation(cost=0.00..0.00 card=25.00 ndv=0.00
rowsize=0) |
| Output: #[0,1] |
| Table: 'nation' (0:'n_nationkey', 1:'PADDR') |
| Filter Cond: (CAST(#[0,0] AS INT64) > 10) |
+-----------------------------------------------------------------------
-------------+
```

带有限制：

```
mysql> explain verbose DELETE FROM NATION WHERE N_NATIONKEY > 10 LIMIT
20;
+-----------------------------------------------------------------------
-------------+
| QUERY PLAN |
+-----------------------------------------------------------------------
-------------+
| Delete on db1.nation (cost=0.0..0.0 rows=0 ndv=0 rowsize=0) |
| -> Project(cost=0.00..0.00 card=25.00 ndv=0.00 rowsize=0) |
| Output: #[0,0] |
| Limit: 20 |
| -> Table Scan on db1.nation(cost=0.00..0.00 card=25.00 ndv=0.00
rowsize=0) |
| Output: #[0,1] |
| Table: 'nation' (0:'n_nationkey', 1:'PADDR') |
| Filter Cond: (CAST(#[0,0] AS INT64) > 10) |
+-----------------------------------------------------------------------
-------------+
```

EXPLAIN 输出格式

输出结构

语法结构执行结果是为 `statement` 选择的计划的文本描述，可以选择使用执行统计信息进行注释。

以下以 SQL 为例，演示输出结构：

```
explain select city,libname1,count(libname1) as a from t3 join t1 on
libname1=libname3 join t2 on isbn3=isbn2 group by city,libname1;
```

```
+-----------------------------------------------------------------------
--------------------+
| QUERY PLAN |
+-----------------------------------------------------------------------
--------------------+
| Project(cost=0.00..0.00 card=400.00 ndv=0.00 rowsize=0 |
| -> Aggregate(cost=0.00..0.00 card=400.00 ndv=0.00 rowsize=0 |
| Group Key:#[0,1], #[0,0] |
| Aggregate Functions: count(#[0,0]) |
| -> Join(cost=0.00..0.00 card=400.00 ndv=0.00 rowsize=0 |
| Join Type: INNER |
| Join Cond: (#[1,2] = #[0,0]) |
| -> Table Scan on abc.t2(cost=0.00..0.00 card=8.00 ndv=0.00 rowsize=0 |
| -> Join(cost=0.00..0.00 card=50.00 ndv=0.00 rowsize=0 |
| Join Type: INNER |
| Join Cond: (#[0,0] = #[1,1]) |
| -> Table Scan on abc.t1(cost=0.00..0.00 card=5.00 ndv=0.00 rowsize=0 |
| -> Table Scan on abc.t3(cost=0.00..0.00 card=10.00 ndv=0.00 rowsize=0 |
+-----------------------------------------------------------------------
--------------------+
13 rows in set (0.00 sec)
```

EXPLAIN 输出一个名称为 `Execution Plan Tree` 树形结构，每个叶子节点都包含节点类型、受影响的对象以及其他属性的信息，如 `cost`， `rowsize` 等。我们现在只使用节点类型信息来简化展示上面的示例。`Execution Plan Tree` 树形结构可以可视化 SQL 查询的整个过程，显示它所经过的操作节点以及它们的成本估计。

```
Project
└── Aggregate
└── Join
└── Table Scan
└── Join
└──Table Scan
└──Table Scan
```

节点类型

MatrixOne 支持以下节点类型。

节点类型	Explain 中的命名
Node_TABLE_SCAN	Table Scan
Node_VALUE_SCAN	Values Scan
Node_PROJECT	Project
Node_AGG	Aggregate
Node_FILTER	Filter
Node_JOIN	Join
Node_SORT	Sort
Node_INSERT	Insert
Node_UPDATE	Update
Node_DELETE	Delete

Table Scan

特性	格式	描述
cost	cost=0.00..0.00	The first is estimated start-up cost. This is the time expended before the output phase can begin, e.g., time to do the sorting in a sort node. The second is estimated total cost. This is stated on the assumption that the plan node is run to completion, i.e., all available rows are retrieved. In practice a node's parent node

特性	格式	描述
		might stop short of reading all available rows (see the `LIMIT` example below).
card	card=14.00	Estimated column cardinality.
ndv	ndv=0.00	Estimated number of distinct values.
rowsize	rowsize=0.00	Estimated rowsize.
output	Output: #[0,0], #[0,1], #[0,2], #[0,3], #[0,4], #[0,5], #[0,6], #[0,7]	Node output information.
Table	Table : 'emp' (0:'empno', 1:'ename', 2:'job', 3:'mgr',)	Table definition information after column pruning.
Filter Cond	Filter Cond: (CAST(#[0,5] AS DECIMAL128) > CAST(20 AS DECIMAL128))	Filter condition.

Values Scan

特性	格式	描述
cost	(cost=0.00..0.00 card=14.00 ndv=0.00 rowsize=0)	Estimated cost
output	Output: 0	Node output information

Project

特性	格式	描述
cost	(cost=0.00..0.00 card=25.00 ndv=0.00 rowsize=0)	Estimated cost
output	Output: (CAST(#[0,0] AS INT64) + 2)	Node output information

Aggregate

特性	格式	描述
cost	(cost=0.00..0.00 card=14.00 ndv=0.00 rowsize=0)	Estimated cost

特性	格式	描述
output	Output: #[0,0], #[0,1], #[0,2], #[0,3], #[0,4], #[0,5], #[0,6], #[0,7]	Node output information
Group Key	Group Key:#[0,0]	Key for grouping
Aggregate Functions	Aggregate Functions: max(#[0,1])	Aggregate function name

Filter

特性	格式	描述
cost	(cost=0.00..0.00 card=14.00 ndv=0.00 rowsize=0)	Estimated cost
output	Output: #[0,0], #[0,1], #[0,2], #[0,3], #[0,4], #[0,5], #[0,6], #[0,7]	Node output information
Filter Cond	Filter Cond: (CAST(#[0,1] AS INT64) > 10)	Filter condition

Join

特性	格式	描述
cost	(cost=0.00..0.00 card=14.00 ndv=0.00 rowsize=0)	Estimated cost
output	Output: #[0,0]	Node output information
Join Type: INNER	Join Type: INNER	Join type
Join Cond	Join Cond: (#[0,0] = #[1,0])	Join condition

Sort

特性	格式	描述
cost	(cost=0.00..0.00 card=25.00 ndv=0.00 rowsize=0)	Estimated cost
output	Output: #[0,0], #[0,1], #[0,2], #[0,3], #[0,4], #[0,5], #[0,6], #[0,7]	Node output information
Sort Key	Sort Key: #[0,0] DESC, #[0,1] INTERNAL	Sort key

特性	格式	描述
Limit	Limit: 10	Number limit for output data
Offset	Offset: 20	Number offset for output data

通过 `EXPLAIN ANALYZE` 获取信息

`EXPLAIN ANALYZE` 是一个用于查询的分析工具，它将向你显示 SQL 在查询上花费的时间以及原因。它将计划查询、检测它并执行它，同时计算行数并测量在执行计划的各个点花费的时间。执行完成后，`EXPLAIN ANALYZE` 将打印计划和测量结果，而不是查询结果。

`EXPLAIN ANALYZE`，它运行 SQL 语句产生 `EXPLAIN` 输出，此外，还产生其他信息，例如时间和基于迭代器的附加信息，以及关于优化器的预期与实际执行的匹配情况。

对于每个迭代器，提供以下信息：

- 预计执行成本

  成本模型没有考虑一些迭代器，因此不包括在估算中。

- 估计的返回的行数
- 返回第一行的时间
- 执行此迭代器（仅包括子迭代器，但不包括父迭代器）所花费的时间，以毫秒为单位。
- 迭代器返回的行数
- 循环数

查询执行信息使用 `TREE` 输出格式显示，其中节点代表迭代器。`EXPLAIN ANALYZE` 始终使用 `TREE` 输出格式，也可以选择使用 `FORMAT=TREE;` 显式指定。其他格式 `TREE` 暂不支持。

`EXPLAIN ANALYZE` 可以与 `SELECT` 语句一起使用，也可以与多表 `UPDATE` 和 `DELETE` 语句一起使用。

你可以使用 `KILL QUERY` 或 `CTRL-C` 终止此语句。

`EXPLAIN ANALYZE` 不能与 `FOR CONNECTION` 一起使用。

示例

建表

```
CREATE TABLE t1 (
c1 INTEGER DEFAULT NULL,
c2 INTEGER DEFAULT NULL
);

CREATE TABLE t2 (
c1 INTEGER DEFAULT NULL,
c2 INTEGER DEFAULT NULL
);

CREATE TABLE t3 (
pk INTEGER NOT NULL PRIMARY KEY,
i INTEGER DEFAULT NULL
);
```

表输出结果：

```
> EXPLAIN ANALYZE SELECT * FROM t1 JOIN t2 ON (t1.c1 = t2.c2)\G
*************************** 1. row ***************************
QUERY PLAN: Project
*************************** 2. row ***************************
QUERY PLAN: Analyze: timeConsumed=0us inputRows=0 outputRows=0
inputSize=0bytes outputSize=0bytes memorySize=0bytes
*************************** 3. row ***************************
QUERY PLAN: -> Join
*************************** 4. row ***************************
QUERY PLAN: Analyze: timeConsumed=5053us inputRows=0 outputRows=0
inputSize=0bytes outputSize=0bytes memorySize=0bytes
*************************** 5. row ***************************
QUERY PLAN: Join Type: INNER
*************************** 6. row ***************************
QUERY PLAN: Join Cond: (t1.c1 = t2.c2)
*************************** 7. row ***************************
QUERY PLAN: -> Table Scan on aaa.t1
*************************** 8. row ***************************
QUERY PLAN: Analyze: timeConsumed=2176us inputRows=0 outputRows=0
inputSize=0bytes outputSize=0bytes memorySize=0bytes
*************************** 9. row ***************************
QUERY PLAN: -> Table Scan on aaa.t2
**************************** 10. row ****************************
QUERY PLAN: Analyze: timeConsumed=0us inputRows=0 outputRows=0
inputSize=0bytes outputSize=0bytes memorySize=0bytes
10 rows in set (0.00 sec)

> EXPLAIN ANALYZE SELECT * FROM t3 WHERE i > 8\G
*************************** 1. row ***************************
QUERY PLAN: Project
*************************** 2. row ***************************
QUERY PLAN: Analyze: timeConsumed=0us inputRows=0 outputRows=0
inputSize=0bytes outputSize=0bytes memorySize=0bytes
*************************** 3. row ***************************
QUERY PLAN: -> Table Scan on aaa.t3
*************************** 4. row ***************************
QUERY PLAN: Analyze: timeConsumed=154us inputRows=0 outputRows=0
inputSize=0bytes outputSize=0bytes memorySize=0bytes
*************************** 5. row ***************************
QUERY PLAN: Filter Cond: (CAST(t3.i AS BIGINT) > 8)
5 rows in set (0.00 sec)

> EXPLAIN ANALYZE SELECT * FROM t3 WHERE pk > 17\G
*************************** 1. row ***************************
```

```
QUERY PLAN: Project
*************************** 2. row ***************************
QUERY PLAN: Analyze: timeConsumed=0us inputRows=0 outputRows=0
inputSize=0bytes outputSize=0bytes memorySize=0bytes
*************************** 3. row ***************************
QUERY PLAN: -> Table Scan on aaa.t3
*************************** 4. row ***************************
QUERY PLAN: Analyze: timeConsumed=309us inputRows=0 outputRows=0
inputSize=0bytes outputSize=0bytes memorySize=0bytes
*************************** 5. row ***************************
QUERY PLAN: Filter Cond: (CAST(t3.pk AS BIGINT) > 17)
5 rows in set (0.00 sec)
```

该语句输出中显示的实际时间值以毫秒为单位。

ANY_VALUE

函数说明

`ANY_VALUE` 在范围内任选一个值返回。

函数语法

```
> ANY_VALUE(arg)
```

参数释义

参数	说明
arg	可为任意类型。当 arg 为 NULL 时，该行不参与计算。

返回值

返回类型和输入类型相同。

说明：`ANY_VALUE` 的执行结果具有不确定性，相同的输入可能得到不同的执行结果。

示例

```
> create table t1(
-> a int,
-> b int,
-> c int
-> );
> create table t2(
-> a int,
-> b int,
-> c int
-> );
> insert into t1 values(1,10,34),(2,20,14);
> insert into t2 values(1,-10,-45);
> select ANY_VALUE(t1.b) from t1 left join t2 on t1.c=t1.b and t1.a=t1.c
group by t1.a;
+-----------------+
| any_value(t1.b) |
+-----------------+
| 10 |
| 20 |
+-----------------+
2 rows in set (0.01 sec)
> select 3+(5*ANY_VALUE(t1.b)) from t1 left join t2 on t1.c=t1.b and
t1.a=t1.c group by t1.a;
+---------------------------+
| 3 + (5 * any_value(t1.b)) |
+---------------------------+
| 53 |
| 103 |
+---------------------------+
2 rows in set (0.00 sec)
```

AVG

函数说明

`AVG()` 是聚合函数的一种，计算了参数列的算术平均值。

函数语法

```
> AVG(expr)
```

参数释义

参数	说明
expr	任何数值类型的列的列名

返回值

以 `Double` 类型返回该列的算术平均值。 若输入参数为空，则返回 `NaN` 值。

示例

```
> drop table if exists tbl1,tbl2;
> create table tbl1 (col_1a tinyint, col_1b smallint, col_1c int, col_1d
bigint, col_1e char(10) not null);
> insert into tbl1 values (0,1,1,7,"a");
> insert into tbl1 values (0,1,2,8,"b");
> insert into tbl1 values (0,1,3,9,"c");
> insert into tbl1 values (0,1,4,10,"D");
> insert into tbl1 values (0,1,5,11,"a");
> insert into tbl1 values (0,1,6,12,"c");

> select avg(col_1c) from tbl1;
+-------------+
| avg(col_1c) |
+-------------+
| 3.5000 |
+-------------+

> select sum(col_1d) as s1,avg(col_1d) as a3 from tbl1 group by col_1e
order by s1 desc;
+------+---------+
| s1 | a3 |
+------+---------+
| 21 | 10.5000 |
| 18 | 9.0000 |
| 10 | 10.0000 |
| 8 | 8.0000 |
+------+---------+

> select avg(col_1d) as a1 from tbl1 where col_1d < 13 group by col_1e
order by a1;
+---------+
| a1 |
+---------+
| 8.0000 |
| 9.0000 |
| 10.0000 |
| 10.5000 |
+---------+
```

BIT_AND

函数说明

BIT_AND() 是一个聚合函数，计算了列中所有位的按位与。

函数语法

```
> BIT_AND(expr)
```

参数释义

参数	说明
expr	UINT 类型的列

示例

```
> drop table if exists t1;
> CREATE TABLE t1 (id CHAR(1), number INT);
> INSERT INTO t1 VALUES
('a',111),('a',110),('a',100),
('a',000),('b',001),('b',011);

> select id, BIT_AND(number) FROM t1 GROUP BY id;
+------+-----------------+
| id | bit_and(number) |
+------+-----------------+
| a | 0 |
| b | 1 |
+------+-----------------+
```

BIT_OR

函数说明

BIT_OR() 是一个聚合函数，计算了列中所有位的按位与。

函数语法

```
> BIT_OR(expr)
```

参数释义

参数	说明
expr	UINT 类型的列

示例

```
> drop table if exists t1;
> CREATE TABLE t1 (id CHAR(1), number INT);
> INSERT INTO t1 VALUES
('a',111),('a',110),('a',100),
('a',000),('b',001),('b',011);

> select id, BIT_OR(number) FROM t1 GROUP BY id;
+------+----------------+
| id | bit_or(number) |
+------+----------------+
| a | 111 |
| b | 11 |
+------+----------------+
```

BIT_XOR

函数说明

BIT_XOR() 是一个聚合函数，计算了列中所有位的按位异或。

函数语法

```
> BIT_XOR(expr)
```

参数释义

参数	说明
expr	UINT 类型的列

示例

```
> drop table if exists t1;
> CREATE TABLE t1 (id CHAR(1), number INT);
> INSERT INTO t1 VALUES
('a',111),('a',110),('a',100),
('a',000),('b',001),('b',011);

> select id, bit_xor(number) from t1 group by id;
+------+-----------------+
| id | bit_xor(number) |
+------+-----------------+
| a | 101 |
| b | 10 |
+------+-----------------+
```

COUNT

函数说明 n

`COUNT()` 是聚合函数的一种，计算了查询结果的记录数（`NULL` 值不参与统计）。

函数语法

```
> COUNT(expr)
```

参数释义

参数	说明
expr	任何查询结果，既可以是列名，也可以是一个函数或者数学运算的结果。也可以使用 `*`，直接统计行数。

返回值

返回查询结果中 `expr` 列的 `NOT NULL` 的值的个数，返回数据类型为 `BIGINT`。 如果没有匹配的行，将返回 0。

示例

```
> drop table if exists tbl1,tbl2;
> create table tbl1 (col_1a tinyint, col_1b smallint, col_1c int, col_1d
bigint, col_1e char(10) not null);
> insert into tbl1 values (0,1,1,7,"a");
> insert into tbl1 values (0,1,2,8,"b");
> insert into tbl1 values (0,1,3,9,"c");
> insert into tbl1 values (0,1,4,10,"D");
> insert into tbl1 values (0,1,5,11,"a");
> insert into tbl1 values (0,1,6,12,"c");

> select count(col_1b) from tbl1;
+---------------+
| count(col_1b) |
+---------------+
| 6 |
+---------------+

> select count(*) from tbl1 where col_1d<10;
+----------+
| count(*) |
+----------+
| 3 |
+----------+
```

GROUP_CONCAT

函数说明

`GROUP_CONCAT` 函数返回一个字符串，它将通过列或者表达式指定的内容连接起来。

如果结果集没有任何行，此函数将返回 `NULL`。

函数语法

```
> GROUP_CONCAT(expr)
```

完整的语法如下：

```
GROUP_CONCAT([DISTINCT] expr [,expr ...]
[ORDER BY {unsigned_integer | col_name | expr}
[ASC | DESC] [,col_name ...]]
[SEPARATOR str_val])
```

参数释义

参数	说明
expr	必须参数。它指定了要连接的一个或者多个列或表达式。
DISTINCT	可选参数。它用于消除重复值。
ORDER BY	可选参数。它用于对要连接的内容排序。默认情况下，它按升序排序值。 如果要按降序对值进行排序，则需要明确指定 `DESC` 选项。
SEPARATOR	可选参数。连接符。默认是 `,`。

返回值

返回值是一个非二进制或二进制字符串，这取决于参数 `expr` 是非二进制还是二进制字符串。

如果结果集没有任何行，此函数将返回 `NULL` 。

示例

```
create table t1(a int,b text,c text);
insert into t1 values(1,"a","bc"),(2,"ab","c"),(3,"aa","bb"),
(3,"aa","bb");

mysql> select group_concat(distinct a,b,c separator '|') from t1;
+-----------------------------------+
| group_concat(distinct a, b, c, |) |
+-----------------------------------+
| 1abc|2abc|3aabb |
+-----------------------------------+
1 row in set (0.01 sec)

mysql> select group_concat(distinct b,c separator '|') from t1 group by
a;
+--------------------------------+
| group_concat(distinct b, c, |) |
+--------------------------------+
| abc |
| abc |
| aabb |
+--------------------------------+
3 rows in set (0.01 sec)

mysql> select group_concat(distinct b,c separator '|') from t1;
+--------------------------------+
| group_concat(distinct b, c, |) |
+--------------------------------+
| abc|abc|aabb |
+--------------------------------+
1 row in set (0.01 sec)
```

MAX

函数说明

`MAX()` 是聚合函数的一种，计算了一组值的最大值。

函数语法

```
> MAX(expr)
```

参数释义

参数	说明
expr	任何数值类型与字符串列的列名

返回值

返回 `expr` 列中的最大值，同时也可以在 `MAX()` 中使用字符串，如此会返回最大的字符串值。

示例

```
> drop table if exists tbl1,tbl2;
> create table tbl1 (col_1a tinyint, col_1b smallint, col_1c int, col_1d
bigint, col_1e char(10) not null);
> insert into tbl1 values (0,1,1,7,"a");
> insert into tbl1 values (0,1,2,8,"b");
> insert into tbl1 values (0,1,3,9,"c");
> insert into tbl1 values (0,1,4,10,"D");
> insert into tbl1 values (0,1,5,11,"a");
> insert into tbl1 values (0,1,6,12,"c");

> select max(col_1d) from tbl1;
+-------------+
| max(col_1d) |
+-------------+
| 12 |
+-------------+

> select max(col_1c) as m1 from tbl1 where col_1d<12 group by col_1e;
+------+
| m1 |
+------+
| 5 |
| 2 |
| 3 |
| 4 |
+------+
```

MEDIAN()

函数说明

`MEDIAN()` 用于返回一组数值的中值，即将一组数值排序后返回居于中间的数值。如果参数集合中包含偶数个数值，该函数将返回位于中间的两个数的平均值。可以将其用作聚合或分析函数。

函数语法

```
> MEDIAN(expr)
```

参数释义

参数	说明
expr	必要参数。指定要求中值的数组名称，参数类型属于数值数据类型或可以隐式转换为数字数据类型。

返回类型

该函数返回与参数的数值数据类型相同的数据类型。

示例

```
mysql> select median(null);
+--------------+
| median(null) |
+--------------+
| NULL |
+--------------+
1 row in set (0.00 sec)

drop table if exists t1;
create table t1 (a int,b int);
insert into t1 values (1,null);

mysql> select median(b) from t1;
+------------+
| median(b) |
+------------+
| NULL |
+------------+
1 row in set (0.01 sec)

insert into t1 values (1,1);

mysql> select median(b) from t1;
+------------+
| median(b) |
+------------+
| 1 |
+------------+
1 row in set (0.01 sec)

insert into t1 values (1,2);

mysql> select median(b) from t1;
+------------+
| median(b) |
+------------+
| 1.5 |
+------------+
1 row in set (0.01 sec)

mysql> select median(b) from t1 group by a order by a;
+------------+
```

```
| median(b) |
+-----------+
| 1.5 |
+-----------+
1 row in set (0.00 sec)

insert into t1 values (2,1),(2,2),(2,3),(2,4);

mysql> select median(b) from t1 group by a order by a;
+-----------+
| median(b) |
+-----------+
| 1.5 |
| 2.5 |
+-----------+
2 rows in set (0.01 sec)
```

MIN

函数说明

`MIN()` 是聚合函数的一种，计算了一组值的最小值。

函数语法

```
> MIN(expr)
```

参数释义

参数	说明
expr	任何数值类型与字符串列的列名

返回值

返回 `expr` 列中的最小值，同时也可以在 `MAX()` 中使用字符串，如此会返回最小的字符串值。

示例

```
> drop table if exists tbl1,tbl2;
> create table tbl1 (col_1a tinyint, col_1b smallint, col_1c int, col_1d
bigint, col_1e char(10) not null);
> insert into tbl1 values (0,1,1,7,"a");
> insert into tbl1 values (0,1,2,8,"b");
> insert into tbl1 values (0,1,3,9,"c");
> insert into tbl1 values (0,1,4,10,"D");
> insert into tbl1 values (0,1,5,11,"a");
> insert into tbl1 values (0,1,6,12,"c");

> select min(col_1d) from tbl1;
+-------------+
| min(col_1d) |
+-------------+
| 7 |
+-------------+

> select min(col_1c) as m1 from tbl1 where col_1d<12 group by col_1e;
+------+
| m1 |
+------+
| 1 |
| 2 |
| 3 |
| 4 |
+------+
```

SLEEP()

函数说明

`SLEEP()` 函数将当前查询暂停（睡眠）指定的秒数。结果将返回 0。

函数语法

```
>
SLEEP(duration)
```

参数释义

参数	说明
duration	必需的。 以秒为单位的睡眠时长。它应该大于或等于 0，并且可以带有小数部分。

返回值

- 当 `SLEEP()` 正常返回时（没有中断），它返回 0。
- 当 `SLEEP()` 被中断的查询调用唯一的结果时，它返回 1 并且查询本身不返回错误。

  示例如下：

  1. 在会话 1 中执行下面的命令，查询当前的 connection_id，并执行 `SLEEP()` 函数：

```
mysql> select connection_id();
+-----------------+
| connection_id() |
+-----------------+
| 1476 |
+-----------------+
1 row in set (0.03 sec)
mysql> select sleep(200);
```

2. 此时，打开一个新的会话，中断会话 1，执行如下命令：

```
mysql> kill 1463;
Query OK, 0 rows affected (0.00 sec)
```

3. 查看会话 1 的结果：

```
mysql> select sleep(200);
+------------+
| sleep(200) |
+------------+
| 1 |
+------------+
1 row in set (26.50 sec)
```

- 部分查询被打断时，SLEEP() 返回错误，例如：

```
mysql> SELECT 1 FROM t1 WHERE SLEEP(1000);
ERROR 20101 (HY000): internal error: pipeline closed unexpectedly
```

示例

```
-- without interruption
mysql> SELECT SLEEP(1);
+----------+
| sleep(1) |
+----------+
| 0 |
+----------+
1 row in set (1.01 sec)

-- without interruption
mysql> SELECT SLEEP(1000);
+-------------+
| sleep(1000) |
+-------------+
| 0 |
+-------------+
1 row in set (18 min 20.87 sec)

create table t1 (a int,b int);
insert into t1 values (1,1),(1,null);
mysql> select sleep(a) from t1;
+----------+
| sleep(a) |
+----------+
| 0 |
| 0 |
+----------+
2 rows in set (2.01 sec)
```

STDDEV_POP

函数说明

STDDEV_POP(expr) 是一个聚合函数，计算总体标准差。

函数语法

```
> STDDEV_POP(expr)
```

参数释义

参数	说明
expr	任何数值类型的列的列名

示例

```
> CREATE TABLE t1(PlayerName VARCHAR(100) NOT NULL,RunScored INT NOT
NULL,WicketsTaken INT NOT NULL);
> INSERT INTO t1 VALUES('KL Rahul', 52, 0 ),('Hardik Pandya', 30, 1 ),
('Ravindra Jadeja', 18, 2 ),('Washington Sundar', 10, 1),('D Chahar', 11,
2 ), ('Mitchell Starc', 0, 3);
> SELECT STDDEV_POP(RunScored) as Pop_Standard_Deviation FROM t1;
> SELECT STDDEV_POP(WicketsTaken) as Pop_Std_Dev_Wickets FROM t1;


> SELECT STDDEV_POP(RunScored) as Pop_Standard_Deviation FROM t1;
+------------------------+
| Pop_Standard_Deviation |
+------------------------+
| 16.8762 |
+------------------------+
1 row in set (0.02 sec)

> SELECT STDDEV_POP(WicketsTaken) as Pop_Std_Dev_Wickets FROM t1;
+---------------------+
| Pop_Std_Dev_Wickets |
+---------------------+
| 0.9574 |
+---------------------+
1 row in set (0.01 sec)
```

SUM

函数说明

`SUM()` 聚合函数计算了一组值的和（NULL 值被忽略）。

函数语法

```
> SUM(expr)
```

参数释义

参数	说明
expr	任何数值类型与字符串列的列名

返回值

返回 `expr` 列的数值的和，若输入参数为 `Double` 类型，则返回值为 `Double`，否则为整数类型。
如果没有匹配的行，则返回 `NULL` 值。

示例

```
> drop table if exists tbl1,tbl2;
> create table tbl1 (col_1a tinyint, col_1b smallint, col_1c int, col_1d
bigint, col_1e char(10) not null);
> insert into tbl1 values (0,1,1,7,"a");
> insert into tbl1 values (0,1,2,8,"b");
> insert into tbl1 values (0,1,3,9,"c");
> insert into tbl1 values (0,1,4,10,"D");
> insert into tbl1 values (0,1,5,11,"a");
> insert into tbl1 values (0,1,6,12,"c");

> select sum(col_1c) from tbl1;
+-------------+
| sum(col_1c) |
+-------------+
| 21 |
+-------------+

> select sum(col_1d) as c1 from tbl1 where col_1d < 13 group by col_1e
order by c1;
+------+
| c1 |
+------+
| 8 |
| 10 |
| 18 |
| 21 |
+------+
```

自定义变量

在 MatrixOne 中，你可以通过 `SET` 命令进行变量的自定义，并且在 SQL 中使用。具体语法如下：

```
SET variable = expr, [variable = expr ..,]
```

示例

现在以定义两个变量 a 和 b 为例：

```
> SET @a=2, @b=3;
Query OK, 0 rows affected (0.00 sec)

> select @a;
+------+
| @a |
+------+
| 2 |
+------+
1 row in set (0.00 sec)

> select @b;
+------+
| @b |
+------+
| 3 |
+------+
1 row in set (0.00 sec)
```

在 SQL 中使用自定义变量：

```
> create table t1(a int,b varchar(1));
Query OK, 0 rows affected (0.02 sec)

> insert into t1 values(@a,@b);
Query OK, 1 row affected (0.02 sec)

> select * from t1;
+------+------+
| a | b |
+------+------+
| 2 | 3 |
+------+------+
1 row in set (0.01 sec)
```

注意

变量 a 和 b 在此处都是 int 数据类型，如果想要一个字符串的 2 或 3，建议使用 `SET @a='2', @b='3';`。

数据类型

MatrixOne 的数据类型与 MySQL 数据类型的定义一致，可参考：
https://dev.mysql.com/doc/refman/8.0/en/data-types.html

整数类型

数据类型	存储空间	最小值	最大值
TINYINT	1 byte	-128	127
SMALLINT	2 byte	-32768	32767
INT	4 byte	-2147483648	2147483647
BIGINT	8 byte	-9223372036854775808	9223372036854775807
TINYINT UNSIGNED	1 byte	0	255
SMALLINT UNSIGNED	2 byte	0	65535
INT UNSIGNED	4 byte	0	4294967295
BIGINT UNSIGNED	8 byte	0	18446744073709551615

示例

- TINYINT and TINYINT UNSIGNED

```
-- Create a table named "inttable" with 2 attributes of a "tinyint", a
"tinyint unsigned",
create table inttable ( a tinyint not null default 1, tinyint8 tinyint
unsigned primary key);
insert into inttable (tinyint8) values (0),(255), (0xFE), (253);

mysql> select * from inttable order by 2 asc;
+------+----------+
| a | tinyint8 |
+------+----------+
| 1 | 0 |
| 1 | 253 |
| 1 | 254 |
| 1 | 255 |
+------+----------+
4 rows in set (0.03 sec)
```

- SMALLINT 和 SMALLINT UNSIGNED

```
-- Create a table named "inttable" with 2 attributes of a "smallint", a
"smallint unsigned",
drop table inttable;
create table inttable ( a smallint not null default 1, smallint16
smallint unsigned);
insert into inttable (smallint16) values (0),(65535), (0xFFFE), (65534),
(65533);

mysql> select * from inttable;
+------+------------+
| a | smallint16 |
+------+------------+
| 1 | 0 |
| 1 | 65535 |
| 1 | 65534 |
| 1 | 65534 |
| 1 | 65533 |
+------+------------+
5 rows in set (0.01 sec)
```

- INT 和 INT UNSIGNED

```
-- Create a table named "inttable" with 2 attributes of a "int", a "int
unsigned",
drop table inttable;
create table inttable ( a int not null default 1, int32 int unsigned
primary key);
insert into inttable (int32) values (0),(4294967295), (0xFFFFFFFE),
(4294967293), (4294967291);

mysql> select * from inttable order by a desc, 2 asc;
+------+------------+
| a | int32 |
+------+------------+
| 1 | 0 |
| 1 | 4294967291 |
| 1 | 4294967293 |
| 1 | 4294967294 |
| 1 | 4294967295 |
+------+------------+
5 rows in set (0.01 sec)
```

- BIGINT 和 BIGINT UNSIGNED

```
-- Create a table named "inttable" with 2 attributes of a "bigint", a
"bigint unsigned",
drop table inttable;
create table inttable ( a bigint, big bigint primary key );
insert into inttable values (122345515, 0xFFFFFFFFFFFFE), (1234567,
0xFFFFFFFFFFFF0);

mysql> select * from inttable;
+-----------+------------------+
| a | big |
+-----------+------------------+
| 122345515 | 4503599627370494 |
| 1234567 | 4503599627370480 |
+-----------+------------------+
2 rows in set (0.01 sec)
```

浮点类型

数据类型	存储空间	精度	语法表示
FLOAT32	4 bytes	23 bits	FLOAT
FLOAT64	8 bytes	53 bits	DOUBLE

示例

```
-- Create a table named "floattable" with 1 attributes of a "float"
create table floattable ( a float not null default 1, big float(20,5)
primary key);
insert into floattable (big) values (-1),(12345678.901234567),
(92233720368547.75807);

mysql> select * from floattable order by a desc, big asc;
+------+----------------+
| a | big |
+------+----------------+
| 1 | -1 |
| 1 | 12345679 |
| 1 | 92233720000000 |
+------+----------------+
3 rows in set (0.01 sec)

mysql> select min(big),max(big),max(big)-1 from floattable;
+----------+----------------+----------------+
| min(big) | max(big) | max(big) - 1 |
+----------+----------------+----------------+
| -1 | 92233720000000 | 92233718038527 |
+----------+----------------+----------------+
1 row in set (0.01 sec)
```

字符串类型

数据类型	存储空间	长度	语法表示	描述
char	24 bytes	0 ~ 4294967295	CHAR	定长字符串

数据类型	存储空间	长度	语法表示	描述
varchar	24 bytes	0 ~ 4294967295	VARCHAR	变长字符串
text	1 GB	other types mapping	TEXT	长文本数据，不区分 TINY TEXT、MEDIUM TEXT 和 LONG TEXT
blob	1 GB	other types mapping	BLOB	二进制的长文本数据，不区分 TINY BLOB、MEDIUM BLOB 和 LONGBLOB

示例

- CHAR 和 VARCHAR

```
-- Create a table named "names" with 2 attributes of a "varchar" and a
"char"
create table names(name varchar(255),age char(255));
insert into names(name, age) values('Abby', '24');
insert into names(name, age) values("Bob", '25');
insert into names(name, age) values('Carol', "23");
insert into names(name, age) values("Dora", "29");

mysql> select name,age from names;
+-------+------+
| name | age |
+-------+------+
| Abby | 24 |
| Bob | 25 |
| Carol | 23 |
| Dora | 29 |
+-------+------+
4 rows in set (0.00 sec)
```

- TEXT

```
-- Create a table named "texttest" with 1 attribute of a "text"
create table texttest (a text);
insert into texttest values('abcdef');
insert into texttest values('_bcdef');
insert into texttest values('a_cdef');
insert into texttest values('ab_def');
insert into texttest values('abc_ef');
insert into texttest values('abcd_f');
insert into texttest values('abcde_');

mysql> select * from texttest where a like 'ab\_def' order by 1 asc;
+--------+
| a |
+--------+
| ab_def |
+--------+
1 row in set (0.01 sec)
```

- BLOB

```
-- Create a table named "blobtest" with 1 attribute of a "blob"
create table blobtest (a blob);
insert into blobtest values('abcdef');
insert into blobtest values('_bcdef');
insert into blobtest values('a_cdef');
insert into blobtest values('ab_def');
insert into blobtest values('abc_ef');
insert into blobtest values('abcd_f');
insert into blobtest values('abcde_');

mysql> select * from blobtest where a like 'ab\_def' order by 1 asc;
+----------------+
| a |
+----------------+
| 0x61625F646566 |
+----------------+
1 row in set (0.01 sec)
```

JSON 数据类型

JSON 数据类型	解释
对象	对象使用 `{}` 括起来，元素之间用 `,` 分隔。JSON 对象中的值/键可以为 String、Nubmber、Bool、时间。
数组	数组使用 `[]` 括起来，元素之间用逗号 `,` 分隔。JSON 数组中值可以为 String、Nubmber、Bool、时间。

示例

```
-- Create a table named "jsontest" with 1 attribute of a "json"
create table jsontest (a json,b int);
insert into jsontest values ('{"t1":"a"}',1),('{"t1":"b"}',2);

mysql> select * from jsontest;
+-------------+------+
| a | b |
+-------------+------+
| {"t1": "a"} | 1 |
| {"t1": "b"} | 2 |
+-------------+------+
2 rows in set (0.01 sec)
```

时间与日期

数据类型	存储空间	精度	最小值	最大值	语法表示
Time	8 byte	microsecond	-2562047787:59:59.999999	2562047787:59:59.999999	hh:mm .ssssss
Date	4 byte	day	0001-01-01	9999-12-31	YYYY-MM-DD/YYYYMMD
DateTime	8 byte	microsecond	0001-01-01 00:00:00.000000	9999-12-31 23:59:59.999999	YYYY-MM-DD hh:mi
TIMESTAMP	8 byte	microsecond	0001-01-01 00:00:00.000000	9999-12-31 23:59:59.999999	YYYYMMDD hh:mi .ssssss

示例

- TIME

```
-- Create a table named "timetest" with 1 attributes of a "time"
create table time_02(t1 time);
insert into time_02 values(200);
insert into time_02 values("");

mysql> select * from time_02;
+----------+
| t1 |
+----------+
| 00:02:00 |
| NULL |
+----------+
2 rows in set (0.00 sec)
```

- DATE

```
-- Create a table named "datetest" with 1 attributes of a "date"
create table datetest (a date not null, primary key(a));
insert into datetest values ('2022-01-01'), ('20220102'),('2022-01-03'),
('20220104');

mysql> select * from datetest order by a asc;
+------------+
| a |
+------------+
| 2022-01-01 |
| 2022-01-02 |
| 2022-01-03 |
| 2022-01-04 |
+------------+
```

- DATETIME

```
-- Create a table named "datetimetest" with 1 attributes of a "datetime"
create table datetimetest (a datetime(0) not null, primary key(a));
insert into datetimetest values ('20200101000000'), ('2022-01-02'),
('2022-01-02 00:00:01'), ('2022-01-02 00:00:01.512345');

mysql> select * from datetimetest order by a asc;
+---------------------+
| a |
+---------------------+
| 2020-01-01 00:00:00 |
| 2022-01-02 00:00:00 |
| 2022-01-02 00:00:01 |
| 2022-01-02 00:00:02 |
+---------------------+
4 rows in set (0.02 sec)
```

- TIMESTAMP

```
-- Create a table named "timestamptest" with 1 attribute of a "timestamp"
create table timestamptest (a timestamp(0) not null, primary key(a));
insert into timestamptest values ('20200101000000'), ('2022-01-02'),
('2022-01-02 00:00:01'), ('2022-01-02 00:00:01.512345');

mysql> select * from timestamptest;
+---------------------+
| a |
+---------------------+
| 2020-01-01 00:00:00 |
| 2022-01-02 00:00:00 |
| 2022-01-02 00:00:01 |
| 2022-01-02 00:00:02 |
+---------------------+
```

Bool

数据类型	存储空间
True	1 byte
False	1 byte

示例

```
-- Create a table named "booltest" with 2 attribute of a "boolean" and b "bool"
create table booltest (a boolean,b bool);
insert into booltest values (0,1),(true,false),(true,1),(0,false),(NULL,NULL);

mysql> select * from booltest;
+-------+-------+
| a | b |
+-------+-------+
| false | true |
| true | false |
| true | true |
| false | false |
| NULL | NULL |
+-------+-------+
5 rows in set (0.00 sec)
```

定点类型 Decimal(Beta)

数据类型	存储空间	精度	语法表示
Decimal	8 byte	19 位	Decimal(N,S) N 表示数字位数的总数，范围是 (1 ~ 18)，小数点和 -（负数）符号不包括在 N 中 S 表示是小数点（标度）后面的位数，范围是 (0 ~ N) 如果 S 是 0，则值没有小数点或分数部分。如果 S 被省略，默认是 0。如果 N 被省略，默认是 1。 例如 Decimal(10,8)，即表示数字总长度为 10，小数位为 8。
Decimal	16 byte	38 位	Decimal(N,S) N 表示数字位数的总数，范围是 (18 ~ 38)，小数点和 -（负数）符号不包括在 N 中 S 表示是小数点（标度）后面的位数，范围是 (0 ~ N) 如果 S 是 0，则值没有小数点或分数部分。如果 S 被省略，默认是 0。如果 N 被省略，默认是 18。 例如 Decimal(20,19)，即表示数字总长度为 20，小数位为 19。

示例

```
-- Create a table named "decimalTest" with 2 attribute of a "decimal" and
b "decimal"
create table decimalTest(a decimal(6,3), b decimal(24,18));
insert into decimalTest values(123.4567, 123456.1234567891411241355);

mysql> select * from decimalTest;
+---------+---------------------------+
| a | b |
+---------+---------------------------+
| 123.457 | 123456.123456789141124136 |
+---------+---------------------------+
1 row in set (0.00 sec)
```

数据类型转换

MatrixOne 支持不同数据类型之间的转换，下表列出了数据类型转换支持情况：

- **可转换**：使用 `cast` 函数，进行显式转换。
- **强制转换**：不使用 `cast` 函数，进行隐式转换，即强制转换。

源数据类型	目标数据类型	显式转换	强制转换
BOOLEAN	INTEGER	❌	❌
	DECIMAL	❌	❌
	VARCHAR	✓	✓
DATE	TIMESTAMP	✓	✓
	DATETIME	✓	✓
	VARCHAR	✓	✓
DATETIME	TIMESTAMP	✓	✓
	DATE	✓	✓
	VARCHAR	✓	✓
FLOAT(Floating-point number)	INTEGER	❌	❌
	DECIMAL	✓	✓
	VARCHAR	✓	✓
INTEGER	BOOLEAN	❌	❌
	FLOAT	✓	✓
	TIMESTAMP	✓	✓
	VARCHAR	✓	✓
	DECIMAL	✓	✓
TIMESTAMP	DATE	✓	✓
	DATETIME	✓	✓

源数据类型	目标数据类型	显式转换	强制转换
	VARCHAR	✓	✓
VARCHAR	BOOLEAN	✓	✓
	DATE	✓	✓
	FLOAT	✓	✓
	INTEGER	✓	✓
	DECIMAL	✓	✓
	TIMESTAMP	✓	✓
	DATETIME	✓	✓

TIMESTAMP 和 DATETIME 的自动初始化和更新

`TIMESTAMP` 和 `DATETIME` 列可以自动初始化并更新为当前日期和时间（即当前时间戳）。

对于表中的任何 `TIMESTAMP` 或 `DATETIME` 列，你可以将当前时间戳指定为默认值、自动更新值或两者均可：

- 对于未为列指定值的插入行，将自动初始化的列设置为当前时间戳。
- 当行中任何其他列的值从当前值进行更改时，自动更新列将自动更新为当前时间戳。如果所有其他列都设置为当前值，则自动更新的列保持不变。若要防止自动更新的列在其他列更改时更新，请显式将其设置为当前值。要更新自动更新的列，即使其他列没有更改，也要显式地将其设置为它应该具有的值 (例如，将其设置为 `CURRENT_TIMESTAMP`)。

要指定自动属性，请在列定义中使用 `DEFAULT CURRENT_TIMESTAMP` 和 `ON UPDATE CURRENT_TIMESTAMP` 子句。两个子句如果同时定义一个列，它们的顺序可互换，不影响逻辑计算。另外，`CURRENT_TIMESTAMP` 与 `CURRENT_TIMESTAMP()` 或者 `NOW()` 意义一致。

`DEFAULT CURRENT_TIMESTAMP` 和 `ON UPDATE CURRENT_TIMESTAMP` 的使用是特定于 `TIMESTAMP` 和 `DATETIME` 的。`DEFAULT` 子句还可用于指定常量（非自动）默认值（例如，`DEFAULT 0` 或 `DEFAULT '2000-01-01 00:00:00'`）。

`TIMESTAMP` 或 `DATETIME` 列定义可以为默认值和自动更新值指定当前时间戳，仅指定其中一个，或者两者都不指定。不同的列可以有不同的自动属性组合。以下规则描述了这些可能性：

- 同时使用 `DEFAULT CURRENT_TIMESTAMP` 和 `ON UPDATE CURRENT_TIMESTAMP` 子句时，

  列的默认值是当前时间戳，并自动更新为当前时间戳。

```
CREATE TABLE t1 (
ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
dt DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);
```

- 仅使用 `DEFAULT` 子句，不使用 `ON UPDATE CURRENT_TIMESTAMP` 子句，该列具有给定的默认值，但不会自动更新为当前时间戳。

默认值取决于 `DEFAULT` 子句是指定 `CURRENT_TIMESTAMP` 还是常量值。使用 `CURRENT_TIMESTAMP`，默认为当前时间戳。

```
CREATE TABLE t1 (
ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
dt DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

对于常量，默认值是给定值。在这种情况下，该列没有自动属性。

```
CREATE TABLE t1 (
ts TIMESTAMP DEFAULT 0,
dt DATETIME DEFAULT 0
);
```

- 使用 `ON UPDATE CURRENT_TIMESTAMP` 子句和常量 `DEFAULT` 子句，该列会自动更新为当前时间戳并使用给定的常量默认值。

```
CREATE TABLE t1 (
ts TIMESTAMP DEFAULT 0 ON UPDATE CURRENT_TIMESTAMP,
dt DATETIME DEFAULT 0 ON UPDATE CURRENT_TIMESTAMP
);
```

- 仅使用 `ON UPDATE CURRENT_TIMESTAMP` 子句，但不使用 `DEFAULT` 子句时，列会自动更新为当前时间戳，但其默认值没有当前时间戳。

  `TIMESTAMP` 的默认值为 0；若使用 `NULL` 属性定义，则默认值为 `NULL`。

```
CREATE TABLE t1 (
ts1 TIMESTAMP ON UPDATE CURRENT_TIMESTAMP, -- default 0
ts2 TIMESTAMP NULL ON UPDATE CURRENT_TIMESTAMP -- default NULL
);
```

`DATETIME` 的默认值为 `NULL`；若使用 `NOT NULL` 属性定义，则默认值为 0。

```
CREATE TABLE t1 (
dt1 DATETIME ON UPDATE CURRENT_TIMESTAMP, -- default NULL
dt2 DATETIME NOT NULL ON UPDATE CURRENT_TIMESTAMP -- default 0
);
```

JSON 数据类型

MatrixOne 支持由 RFC 7159 定义的原生 JSON 数据类型，可以有效访问 JSON（JavaScript 对象表示法）文档中的数据。与将 JSON 格式的字符串存储在字符串列中相比，JSON 数据类型具有以下优势：

- 自动验证存储在 JSON 列中的 JSON 文档。无效的文档会产生报错。
- 自动优化存储格式。存储在 JSON 列中的 JSON 文档转换为允许对文档元素进行快速读取访问的内部格式。当服务器必须读取以这种二进制格式存储的 JSON 值时，不需要从文本中解析该值。二进制格式的结构使服务器能够直接通过键或数组索引查找子对象或嵌套值。

存储 JSON 文档所需的存储空间与 `BLOB` 或 `TEXT` 大致相同。

JSON 的类型

JSON 类型有 JSON 数组和 JSON 对象。

- JSON 数组即包含由逗号分隔并包含在 `[]` 字符内的值列表，如：

```
["abc", 10, null, true, false]
```

- JSON 对象包含一组键值对，由逗号分隔并包含在 `{}` 字符内，如：

```
{"k1": "value", "k2": 10}
```

JSON 数组和对象可以包含标量值，即字符串或数字、JSON 空字面量或 JSON 布尔真或假字面量。JSON 对象中的键值必须是字符串。JSON 也允许时间（日期，日期时间）标量值。如：

```
["12:18:29.000000", "2015-07-29", "2015-07-29 12:18:29.000000"]
```

在 JSON 数组元素和 JSON 对象键值中可嵌套，如：

```
[99, {"id": "HK500", "cost": 75.99}, ["hot", "cold"]]
{"k1": "value", "k2": [10, 20]}
```

JSON 值的规范化

当一个字符串被解析为有效的 JSON 文档时，它也会被规范化。这表示意味着具有与稍后在文档中找到的键重复的键的成员，从左到右读取，将被丢弃。这意味着从左往右读取时，后面出现的重复的的键值将被忽略。

将值插入 JSON 列时执行规范化，如下所示：

```
CREATE TABLE t1 (c1 JSON);
INSERT INTO t1 VALUES
('{"x": 17, "x": "red"}'),
('{"x": 17, "x": "red", "x": [3, 5, 7]}');

mysql> SELECT c1 FROM t1;
+------------------+
| c1 |
+------------------+
| {"x": "red"} |
| {"x": [3, 5, 7]} |
+------------------+
2 rows in set (0.01 sec)
```

BLOB 和 TEXT 数据类型

BLOB

- `BLOB` 是可以存储可变数量的大数据二进制对象。
- `BLOB` 值为二进制字符串（字节字符串），对应二进制字符集和排序规则，比较和排序基于列值中字节的数值。

TEXT

- `TEXT` 为非二进制字符串（字符串），用于存储所有类型的文本数据。它们具有二进制以外的字符集，并且基于字符集的排序规则对值进行排序和比较。

关于 BLOB 和 TEXT

如果为 `BLOB` 或 `TEXT` 列分配的值超过该列的最大长度，则该值超出长度的部分将被截断并生成告警。如果截断的是非空格字符，会发生错误（而不是告警）并禁止插入值。对于 `TEXT`，截断插入到 `TEXT` 列的值中多余的尾部空格时，总是会生成告警。

精确数值类型-Decimal 类型

Decimal 数据类型用来存储精确的数值。精确数值类型在需要用到非常准确的数值精度的场景下使用，比如银行账户数字，或者严谨的科学计算等。 在 Decimal 的列名声明中，整数和小数部分一般都会进行声明。比如以下例子：

```
salary DECIMAL(5,2)
```

在这个案例中，5 是有效数字位数，2 是小数位。整数精度表示数值部分存储的有效位数，小数精度表示小数部分存储的有效位数。 标准 SQL 要求 DECIMAL(5,2) 需要存储 5 位数字，2 位小数，因此如果以该形式表示的工资列数字取值范围应该在 -999.99 到 999.99 之间。 在 MatrixOne 的语法中，DECIMAL(M) 与 DECIMAL(M,0) 是相同的。如果直接只声明 DECIMAL 的话，语法也将解析成 DECIMAL(M,0) 的形式，M 的默认值为 10。 如果小数精度为 0 的话，DECIMAL 相当于纯整数，不含任何小数。 MatrixOne 中 DECIMAL 类型的最大位数为 38 位。另外 DECIMAL 中指定的整数或小数位数在实际列被赋值的时候超出指定范围，实际数值将会被自动转换成相应精度。

Decimal 数据类型特点

该部分内容主要介绍 Decimal 数据类型的一些特点，尤其是在位数精度和存储形式上的。

DECIMAL 列的声明语法是 DECIMAL(M, D). M 是有效数字的位数，取值范围是 1 到 38，D 是小数位数，取值范围是 1 到 38，但是不能大于 M。 如果不指定 D，默认为 0。如果不指定 M，默认为 10。

DECIMAL 列的数值以二进制的形式进行存储，在 MatrixOne 内部，只有 decimal64 和 decimal128 这两种表示形式。在 0-18 位精度内，一个 Decimal 数值占用 8 个字节的存储空间，在 19-38 位精度内，一个 Decimal 数值占用 16 个字节的存储空间。

位数	字节数
0-18	8 个字节
19-38	16 个字节

对于详细的 Decimal 类型实现方法，可以参考 Decimal 的设计文档。

ABS()

函数说明

ABS(X) 返回 X 的绝对值，或者 NULL 如果 X 是 NULL。

函数语法

```
> ABS(number)
```

参数释义

参数	说明
number	必要参数，可取任意数值数据类型

返回值类型与输入类型保持一致。

示例

```
drop table if exists t1;
create table t1(a int,b float);
insert into t1 values(1,-3.1416);
insert into t1 values(-1,1.57);

mysql> select abs(a),abs(b) from t1;
+--------+--------------------+
| abs(a) | abs(b) |
+--------+--------------------+
| 1 | 3.1415998935699463 |
| 1 | 1.5700000524520874 |
+--------+--------------------+
2 rows in set (0.01 sec)
```

ACOS()

函数说明

ACOS() 函数返回给定数值的余弦（用弧度表示）。

函数语法

```
> ACOS(number)
```

参数释义

参数	说明
number	必要参数，可取任意数值数据类型

示例

```
drop table if exists t1;
create table t1(a float,b int);
insert into t1 values(0.5,1);
insert into t1 values(-0.5,-1);

mysql> select acos(a),acos(b) from t1;
+--------------------+--------------------+
| acos(a) | acos(b) |
+--------------------+--------------------+
| 1.0471975511965976 | 0 |
| 2.0943951023931957 | 3.141592653589793 |
+--------------------+--------------------+
2 rows in set (0.01 sec)
```

ATAN()

函数说明

ATAN() 函数返回给定数值的反正切（用弧度表示）。

函数语法

```
> ATAN(number)
```

参数释义

参数	说明
number	必要参数，可取任意数值数据类型

示例

```
drop table if exists t1;
create table t1(a int,b float);
insert into t1 values(1,3.14159);
insert into t1 values(0,1);

mysql> select atan(a),atan(tan(b)) from t1;
+--------------------+--------------------------+
| atan(a) | atan(tan(b)) |
+--------------------+--------------------------+
| 0.7853981633974483 | -0.00000253518159011 3463 |
| 0 | 1 |
+--------------------+--------------------------+
2 rows in set (0.00 sec)
```

CEIL()

函数说明

CEIL(X) 函数返回不小于 X 的最小整数。

函数语法

```
> CEIL(X)
```

参数释义

参数	说明
X	必要参数，可取任意数值数据类型

对 int 类的绝对数值类型，返回值也是相同的绝对数值类型。对浮点数勒说，返回值也是浮点数。

示例

```
drop table if exists t1;
create table t1(a int ,b float);
insert into t1 values(1,0.5);
insert into t1 values(2,0.499);
insert into t1 values(3,0.501);
insert into t1 values(4,20.5);
insert into t1 values(5,20.499);
insert into t1 values(6,13.500);
insert into t1 values(7,-0.500);
insert into t1 values(8,-0.499);
insert into t1 values(9,-0.501);
insert into t1 values(10,-20.499);
insert into t1 values(11,-20.500);
insert into t1 values(12,-13.500);

mysql> select a,ceil(b) from t1;
+------+---------+
| a | ceil(b) |
+------+---------+
| 1 | 1 |
| 2 | 1 |
| 3 | 1 |
| 4 | 21 |
| 5 | 21 |
| 6 | 14 |
| 7 | -0 |
| 8 | -0 |
| 9 | -0 |
| 10 | -20 |
| 11 | -20 |
| 12 | -13 |
+------+---------+
12 rows in set (0.01 sec)

mysql> select sum(ceil(b)) from t1;
+--------------+
| sum(ceil(b)) |
+--------------+
| 6 |
+--------------+
1 row in set (0.01 sec)
```

COS()

函数说明

COS() 函数返回输入参数（用弧度表示）的余弦值。

函数语法

```
> COS(number)
```

参数释义

参数	说明
number	必要参数，可取任意数值数据类型

示例

```
drop table if exists t1;
create table t1(a int,b float);
insert into t1 values(1,3.14159);
insert into t1 values(-1,1.57);

mysql> select cos(a),cos(b) from t1;
+--------------------+----------------------+
| cos(a) | cos(b) |
+--------------------+----------------------+
| 0.5403023058681398 | -0.9999999999967865 |
| 0.5403023058681398 | 0.000796274258662553 |
+--------------------+----------------------+
2 rows in set (0.01 sec)
```

COT()

函数说明

COT() 函数返回输入参数（用弧度表示）的余切值。

函数语法

```
> COT(number)
```

参数释义

参数	说明
number	必要参数，可取任意数值数据类型

示例

```
mysql> SELECT COT(12);
+---------------------+
| cot(12) |
+---------------------+
| -1.5726734063976895 |
+---------------------+
1 row in set (0.00 sec)
```

```
drop table if exists t1;
create table t1(a int,b float);
insert into t1 values(1,3.14159);
insert into t1 values(-1,12);

mysql> select cot(a), cot(b) from t1;
+--------------------+---------------------+
| cot(a) | cot(b) |
+--------------------+---------------------+
| 0.6420926159343306 | -394449.0619219334 |
| -0.6420926159343308 | -1.5726734063976895 |
+--------------------+---------------------+
2 rows in set (0.01 sec)
```

EXP()

函数说明

EXP(number) 函数返回以自然常数 e 为底的 number 的指数。

函数语法

```
> EXP(number)
```

参数释义

参数	说明
number	必要参数，可取任意数值数据类型

示例

```
drop table if exists t1;
create table t1(a int ,b float);
insert into t1 values(-4, 2.45);
insert into t1 values(6, -3.62);

mysql> select exp(a), exp(b) from t1;
+--------------------+----------------------+
| exp(a) | exp(b) |
+--------------------+----------------------+
| 0.01831563888873418 | 11.588347271798835 |
| 403.4287934927351 | 0.026782679557672436 |
+--------------------+----------------------+
2 rows in set (0.00 sec)
```

FLOOR()

函数说明

`FLOOR()` 函数返回不大于某个数字的相应数位的数。

函数语法

```
> FLOOR(number, decimals)
> FLOOR(number)
```

参数释义

参数	说明
number	必要参数，任何当前支持的数值数据
decimals	可选参数，代表小数点后的位数。默认值为 0，代表四舍五入为整数，当为负数时四舍五入到小数点前的数位。

示例

```
drop table if exists t1;
create table t1(a int ,b float);
insert into t1 values(1,0.5);
insert into t1 values(2,0.499);
insert into t1 values(3,0.501);
insert into t1 values(4,20.5);
insert into t1 values(5,20.499);
insert into t1 values(6,13.500);
insert into t1 values(7,-0.500);
insert into t1 values(8,-0.499);
insert into t1 values(9,-0.501);
insert into t1 values(10,-20.499);
insert into t1 values(11,-20.500);
insert into t1 values(12,-13.500);

mysql> select a,floor(b) from t1;
+------+----------+
| a | floor(b) |
+------+----------+
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 20 |
| 5 | 20 |
| 6 | 13 |
| 7 | -1 |
| 8 | -1 |
| 9 | -1 |
| 10 | -21 |
| 11 | -21 |
| 12 | -14 |
+------+----------+
12 rows in set (0.01 sec)

mysql> select sum(floor(b)) from t1;
+---------------+
| sum(floor(b)) |
+---------------+
| -6 |
+---------------+
1 row in set (0.00 sec)
```

```
mysql> select a,sum(floor(b)) from t1 group by a order by a;
+------+---------------+
| a | sum(floor(b)) |
+------+---------------+
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 20 |
| 5 | 20 |
| 6 | 13 |
| 7 | -1 |
| 8 | -1 |
| 9 | -1 |
| 10 | -21 |
| 11 | -21 |
| 12 | -14 |
+------+---------------+
12 rows in set (0.00 sec)
```

LN()

函数说明

LN(X) 返回 X 的自然对数。这个函数是 LOG(X) 的同义函数。

函数语法

```
> LN(X)
```

其他请参考 LOG(X)。

LOG()

函数说明

LOG(X) 函数返回 X 的自然对数。

函数语法

```
> LOG(X)
```

参数释义

参数	说明
X	必要参数，任何当前支持的数值数据

示例

```
drop table if exists t1;
create table t1(a float, b float);
insert into t1 values(2,8);

mysql> select log(a), log(b) from t1;
+--------------------+--------------------+
| log(a) | log(b) |
+--------------------+--------------------+
| 0.6931471805599453 | 2.0794415416798357 |
+--------------------+--------------------+
1 row in set (0.00 sec)
```

限制

LOG(X) 目前仅支持单参数输入。

PI()

函数说明

PI() 返回数学常量 π (pi)。

函数语法

```
> PI()
```

示例

```
drop table if exists t1;
create table t1(a int,b float);
insert into t1 values(0,0),(-15,-20),(-22,-12.5);
insert into t1 values(0,360),(30,390),(90,450),(180,270),(180,180);

mysql> select acos(a*pi()/180) as acosa,acos(b*pi()/180) acosb from t1;
+-------------------+-------------------+
| acosa | acosb |
+-------------------+-------------------+
| 1.5707963267948966 | 1.5707963267948966 |
| 1.8356824738191324 | 1.927370391646567 |
| 1.9648910192076245 | 1.7907312931992256 |
| 1.5707963267948966 | NULL |
| 1.0197267436954502 | NULL |
| NULL | NULL |
| NULL | NULL |
| NULL | NULL |
+-------------------+-------------------+
8 rows in set (0.01 sec)

mysql> select acos(a*pi()/180)*acos(b*pi()/180) as
acosab,acos(acos(a*pi()/180)) as c from t1;
+-------------------+------+
| acosab | c |
+-------------------+------+
| 2.4674011002723395 | NULL |
| 3.5380400485035204 | NULL |
| 3.518591835821214 | NULL |
| NULL | NULL |
| NULL | NULL |
| NULL | NULL |
| NULL | NULL |
| NULL | NULL |
+-------------------+------+
8 rows in set (0.01 sec)
```

POWER()

函数说明

POWER(X, Y) 返回 X 的 Y 次方指数值。

函数语法

```
> POWER(X, Y)
```

参数释义

参数	说明
X	必要参数，任何当前支持的数值数据
Y	必要参数，任何当前支持的数值数据

示例

```
drop table if exists t1;
create table t1(a int,b int);
insert into t1 values(5,-2),(10,3),(100,0),(4,3),(6,-3);

mysql> select power(a,b) from t1;
+----------------------+
| power(a, b) |
+----------------------+
| 0.04 |
| 1000 |
| 1 |
| 64 |
| 0.004629629629629629 |
+----------------------+
5 rows in set (0.01 sec)

mysql> select power(a,2) as a1, power(b,2) as b1 from t1 where power(a,2)
> power(b,2) order by a1 asc;
+-------+------+
| a1 | b1 |
+-------+------+
| 16 | 9 |
| 25 | 4 |
| 36 | 9 |
| 100 | 9 |
| 10000 | 0 |
+-------+------+
5 rows in set (0.01 sec)
```

ROUND()

函数说明

`ROUND()` 函数返回了某个数字在特定位数四舍五入后的数值。 该函数返回指定位数上最接近的数字。如果给定的数字与周围的数字距离相等（比如为 5），那么将采用 “banker’s rounding”（银行进位法）的方式进行舍入。

函数语法

```
> ROUND(number, decimals)
> ROUND(number)
```

参数释义

参数	说明
number	必要参数，想要进行舍入的数值，可取任意数值数据类型
decimals	可选参数，表示将要舍入的小数点后的位数。默认值为 0，代表舍入到整数。 **decimals>0** 函数将舍入到小数点后的位数 **decimals<0** 函数将舍入到小数点前的位数 **decimals=0** 函数将舍入到整数

示例

```
drop table if exists t1;
create table t1(a int ,b float);
insert into t1 values(1,0.5);
insert into t1 values(2,0.499);
insert into t1 values(3,0.501);
insert into t1 values(4,20.5);
insert into t1 values(5,20.499);
insert into t1 values(6,13.500);
insert into t1 values(7,-0.500);
insert into t1 values(8,-0.499);
insert into t1 values(9,-0.501);
insert into t1 values(10,-20.499);
insert into t1 values(11,-20.500);
insert into t1 values(12,-13.500);

mysql> select a,round(b) from t1;
+------+----------+
| a | round(b) |
+------+----------+
| 1 | 0 |
| 2 | 0 |
| 3 | 1 |
| 4 | 20 |
| 5 | 20 |
| 6 | 14 |
| 7 | -0 |
| 8 | -0 |
| 9 | -1 |
| 10 | -20 |
| 11 | -20 |
| 12 | -14 |
+------+----------+
12 rows in set (0.00 sec)

mysql> select a,round(b,-1) from t1;
+------+--------------+
| a | round(b, -1) |
+------+--------------+
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 20 |
```

```
| 5 | 20 |
| 6 | 10 |
| 7 | -0 |
| 8 | -0 |
| 9 | -0 |
| 10 | -20 |
| 11 | -20 |
| 12 | -10 |
+------+--------------+
12 rows in set (0.01 sec)

mysql> select round(a*b) from t1;
+--------------+
| round(a * b) |
+--------------+
| 0 |
| 1 |
| 2 |
| 82 |
| 102 |
| 81 |
| -4 |
| -4 |
| -5 |
| -205 |
| -226 |
| -162 |
+--------------+
12 rows in set (0.01 sec)
```

SIN()

函数说明

SIN() 函数返回输入参数（用弧度表示）的正弦值。

函数语法

```
> SIN(number)
```

参数释义

参数	说明
number	必要参数，可取任意数值数据类型

示例

```
drop table if exists t1;
create table t1(a int,b float);
insert into t1 values(1,3.14159);
insert into t1 values(-1,1.57);

mysql> select sin(a),sin(b) from t1;
+--------------------+-------------------------+
| sin(a) | sin(b) |
+--------------------+-------------------------+
| 0.8414709848078965 | 0.0000025351815901107472 |
| -0.8414709848078965 | 0.9999996829736023 |
+--------------------+-------------------------+
2 rows in set (0.01 sec)
```

SINH()

函数说明

SINH() 函数返回输入参数（用弧度表示）的双曲正弦值。

函数语法

```
> SINH(number)
```

参数释义

参数	说明
number	必要参数，可取任意数值数据类型

示例

```
drop table if exists t1;
create table t1(a int,b float);
insert into t1 values(1,3.14159), (-1,-3.14159);

mysql> select sinh(a), sinh(b) from t1;
+--------------------+--------------------+
| sinh(a) | sinh(b) |
+--------------------+--------------------+
| 1.1752011936438014 | 11.548709969588323 |
| -1.1752011936438014 | -11.548709969588323 |
+--------------------+--------------------+
2 rows in set (0.00 sec)
```

TAN()

函数说明

TAN() 函数返回输入参数（用弧度表示）的正切值。

函数语法

```
> TAN(number)
```

参数释义

参数	说明
number	必要参数，可取任意数值数据类型

示例

```
drop table if exists t1;
create table t1(a int,b float);
insert into t1 values(1,3.14159);
insert into t1 values(-1,-3.14159);

mysql> select tan(a),tan(b) from t1;
+--------------------+--------------------------+
| tan(a) | tan(b) |
+--------------------+--------------------------+
| 1.557407724654902 | -0.00000253518159011889 4 |
| -1.557407724654902 | 0.00000253518159011889 4 |
+--------------------+--------------------------+
2 rows in set (0.01 sec)
```

UUID()

函数说明

`UUID()` 返回根据 RFC 4122 生成国际通用唯一标识符 UUID(Universally Unique IDentifier)。

UUID 在空间和时间上是全球唯一的数字。即使是在两个未连接的独立运行的设备上执行 UUID 调用，预计会生成两个不同的值。

> **⚠ 注意**
>
> 尽管 `UUID()` 值唯一，但它们并非是不可猜测或不可预测的。如果需要不可预测性，则应以其他方式生成 UUID 值。

`UUID()` 返回一个符合 RFC 4122 标准的版本 1 UUID 的值，为 128 位数字，它表示是一个 utf8mb3 由五个十六进制数字组成的字符串，即 aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee，格式解释如下：

- 前三个数字是从时间戳的低、中和高部分生成的。高位部分还包括 UUID 版本号。
- 第四个数字保留时间唯一性，以防时间戳值失去单一性（例如，夏令时）。
- 第五个数字是空间唯一性的 IEEE 802 节点号。如果后者不可用（例如，因为主机设备没有以太网卡，或者不知道如何在主机操作系统上找到接口的硬件地址），则用随机数代替。在这种情况下，无法保证空间唯一性。然而，第五位数字重合的概率很低。

函数语法

```
> UUID()
```

示例

```
drop table if exists t1;
create table t1(a INT, b float);
insert into t1 values(12124, -4213.413), (12124, -42413.409);

mysql> SELECT length(uuid()) FROM t1;
+----------------+
| length(uuid()) |
+----------------+
| 36 |
| 36 |
+----------------+
2 rows in set (0.00 sec)

mysql> SELECT UUID();
+--------------------------------------+
| uuid() |
+--------------------------------------+
| b293b688-70a7-11ed-a25a-5ad2460dea50 |
+--------------------------------------+
1 row in set (0.00 sec)
```

限制

`UUID()` 暂时不支持可选参数，即暂不支持 `UUID([number])`。

CURDATE()

函数说明

`CURDATE()` 函数返回当前日期的 `YYYY-MM-DD` 格式的值，根据函数是否用在字符串或数字语境中。

> **注意**
>
> 与 MySQL 行为不同的是：`curdate()+int` 表示当前日期至 1970-01-01 再加上 `int`（天数）的总天数。比如，`curdate()+1` 表示当前日期减去 1970-01-01 再加 1 天。

函数语法

```
> CURDATE()
```

示例

```
mysql> SELECT CURDATE();
+------------+
| curdate() |
+------------+
| 2023-02-02 |
+------------+
1 row in set (0.00 sec)

mysql> SELECT CURDATE() + 0;
+---------------+
| curdate() + 0 |
+---------------+
| 19390 |
+---------------+
1 row in set (0.00 sec)

mysql> select cast(now() as date)=curdate() q;
+------+
| q |
+------+
| true |
+------+
1 row in set (0.01 sec)

create table t1 (a int);
insert into t1 values (1),(2),(3);

mysql> select cast(now() as date)=curdate() q from t1;
+------+
| q |
+------+
| true |
| true |
| true |
+------+
3 rows in set (0.01 sec)
```

CURRENT_TIMESTAMP()

函数说明

`CURRENT_TIMESTAMP` 和 `CURRENT_TIMESTAMP()` 是 `NOW()` 的同义词。

将当前日期和时间以 `YYYY-MM-DD hh:mm:ss` 或 `YYYYMMDDhhmmss` 的格式返回，返回格式取决于函数是字符串还是数字。取值为当前会话所在的时区。

函数语法

```
> CURRENT_TIMESTAMP([fsp])
```

参数释义

参数	说明
fsp	可选。参数 `fsp` 参数用于指定分秒精度，有效值为 0 到 6 之间的整数。

示例

```
mysql> SELECT CURRENT_TIMESTAMP();
+----------------------------+
| current_timestamp() |
+----------------------------+
| 2022-09-21 11:46:44.153777 |
+----------------------------+
1 row in set (0.00 sec)

mysql> SELECT NOW();
+----------------------------+
| now() |
+----------------------------+
| 2022-09-21 12:56:36.915961 |
+----------------------------+
1 row in set (0.01 sec)
```

```
create table t1 (a int primary key, b int, c int, d timestamp default
current_timestamp);
insert into t1 select 1,1,1,now();
insert into t1 select 2,0,0,null;

mysql> select a,b,c,year(d) from t1;
+------+------+------+---------+
| a | b | c | year(d) |
+------+------+------+---------+
| 1 | 1 | 1 | 2022 |
| 2 | 0 | 0 | NULL |
+------+------+------+---------+
2 rows in set (0.01 sec)
```

限制

运算符 `+` 或 `-` 现在不支持与 `CURRENT_TIMESTAMP` 一起使用。

DATE()

函数说明

将 date 或者 datetime 格式的输入中的日期部分截取出来。

函数语法

```
> DATE(expr)
```

参数释义

参数	说明
expr	必要参数。需要提取日期的 date 或者 datetime 格式的输入值

示例

```
drop table if exists t1;
create table t1(a date, b datetime);
insert into t1 values('2022-01-01','2022-01-01 01:01:01');
insert into t1 values('2022-01-01','2022-01-01 01:01:01');
insert into t1 values(20220101,'2022-01-01 01:01:01');
insert into t1 values('2022-01-02','2022-01-02 23:01:01');
insert into t1 values('2021-12-31','2021-12-30 23:59:59');
insert into t1 values('2022-06-30','2021-12-30 23:59:59');

mysql> select date(a),date(b) from t1;
+------------+------------+
| date(a) | date(b) |
+------------+------------+
| 2022-01-01 | 2022-01-01 |
| 2022-01-01 | 2022-01-01 |
| 2022-01-01 | 2022-01-01 |
| 2022-01-02 | 2022-01-02 |
| 2021-12-31 | 2021-12-30 |
| 2022-06-30 | 2021-12-30 |
+------------+------------+
5 rows in set (0.01 sec)

mysql> select date(a),date(date(a)) as dda from t1;
+------------+------------+
| date(a) | dda |
+------------+------------+
| 2022-01-01 | 2022-01-01 |
| 2022-01-01 | 2022-01-01 |
| 2022-01-01 | 2022-01-01 |
| 2022-01-02 | 2022-01-02 |
| 2021-12-31 | 2021-12-31 |
| 2022-06-30 | 2022-06-30 |
+------------+------------+
5 rows in set (0.00 sec)
```

限制

目前 date 格式只支持 `yyyy-mm-dd` 和 `yyyymmdd` 的数据格式。

DATE_ADD()

函数说明

`DATE_ADD()` 用于执行日期运算：`DATE_ADD()` 函数向日期添加指定的时间间隔。如果 `date` 为 `NULL`，函数返回 `NULL`。

函数语法

```
> DATE_ADD(date,INTERVAL expr unit)
```

参数释义

参数	说明
date	必要参数。date 参数是合法的日期表达式。
expr	必要参数。expr 参数是需要添加进 date 的时间间隔，如果 expr 为负数，那么可以以“-”开头。
unit	必要参数。unit 参数可以是下列值： MICROSECOND SECOND MINUTE HOUR DAY WEEK MONTH QUA TER YEAR SECOND_MICROSECOND MINUTE_MICROSECOND MINUTE_SECOND HOUR_MICROSECOND HOUR_SECOND HOUR_MINUTE DAY_MICROSECOND DAY_SECOND DAY_MINUTE

参数	说明
	DAY_HOUR
	YEAR_MONTH

示例

```
create table t2(orderid int, productname varchar(20), orderdate
datetime);
insert into t2 values ('1','Jarl','2008-11-11 13:23:44.657');

mysql> SELECT OrderId,DATE_ADD(OrderDate,INTERVAL 45 DAY) AS OrderPayDate
FROM t2;
+---------+---------------------+
| orderid | orderpaydate |
+---------+---------------------+
| 1 | 2008-12-26 13:23:45 |
+---------+---------------------+
```

限制

目前 date 格式只支持 `yyyy-mm-dd` 和 `yyyymmdd` 的数据格式。

DATE_FORMAT()

函数说明

根据格式字符串格式化日期值。如果任一参数为 `NULL`，则函数返回 `NULL`。

`DATE_FORMAT()` 返回一个字符串，其中包含由 `character_set_connection` 和 `collation_connection` 给出的字符集和排序规则，以便它可以返回包含非 ASCII 字符的月份和工作日名称。

函数语法

```
> DATE_FORMAT(date,format)
```

参数释义

参数	说明
date	必要参数。date 参数是合法的日期表达式。
format	必要参数。Required. format 可用的说明符可以参加下表详情。

Format 说明符

> **⚠ 注意**
>
> 下表中显示的说明符可用于格式字符串。在格式说明符字符之前需要加 `%` 字符。说明符也适用于函数 `UNIX_TIMESTAMP()`。

说明符	描述
%a	Abbreviated weekday name (Sun..Sat)
%b	Abbreviated month name (Jan..Dec)

说明

符	描述
%c	Month, numeric (0..12)
%D	Day of the month with English suffix (0th, 1st, 2nd, 3rd, ...)
%d	Day of the month, numeric (00..31)
%e	Day of the month, numeric (0..31)
%f	Microseconds (000000..999999)
%H	Hour (00..23)
%h	Hour (01..12)
%I	Hour (01..12)
%i	Minutes, numeric (00..59)
%j	Day of year (001..366)
%k	Hour (0..23)
%l	Hour (1..12)
%M	Month name (January..December)
%m	Month, numeric (00..12)
%p	AM or PM
%r	Time, 12-hour (hh:mm followed by AM or PM)
%S	Seconds (00..59)
%s	Seconds (00..59)
%T	Time, 24-hour (hh:mm)
%U	Week (00..53), where Sunday is the first day of the week; WEEK() mode 0
%u	Week (00..53), where Monday is the first day of the week; WEEK() mode 1
%V	Week (01..53), where Sunday is the first day of the week; WEEK() mode 2; used with %X
%v	Week (01..53), where Monday is the first day of the week; WEEK() mode 3; used with %x

说明

符	描述
%W	Weekday name (Sunday..Saturday)
%w	Day of the week (0=Sunday..6=Saturday)
%X	Year for the week where Sunday is the first day of the week, numeric, four digits; used with %V
%x	Year for the week, where Monday is the first day of the week, numeric, four digits; used with %v
%Y	Year, numeric, four digits
%y	Year, numeric (two digits)
%%	A literal % character
%x	x, for any “x” not listed above

示例

```
mysql> SELECT DATE_FORMAT('2009-10-04 22:23:00', '%W %M %Y');
+-------------------------------------------+
| date_format(2009-10-04 22:23:00, %W %M %Y) |
+-------------------------------------------+
| Sunday October 2009 |
+-------------------------------------------+
1 row in set (0.01 sec)

mysql> SELECT DATE_FORMAT('2007-10-04 22:23:00', '%H:%i:%s');
+-------------------------------------------+
| date_format(2007-10-04 22:23:00, %H:%i:%s) |
+-------------------------------------------+
| 22:23:00 |
+-------------------------------------------+
1 row in set (0.02 sec)

mysql> SELECT Date_format('1900-10-04 22:23:00', '%D %y %a %d %m %b %j');
+-----------------------------------------------------------+
| date_format(1900-10-04 22:23:00, %D %y %a %d %m %b %j) |
+-----------------------------------------------------------+
| 4th 00 Thu 04 10 Oct 277 |
+-----------------------------------------------------------+
1 row in set (0.01 sec)

mysql> SELECT DATE_FORMAT('1997-10-04 22:23:00', '%H %k %I %r %T %S %w');
+-----------------------------------------------------------+
| date_format(1997-10-04 22:23:00, %H %k %I %r %T %S %w) |
+-----------------------------------------------------------+
| 22 22 10 10:23:00 PM 22:23:00 00 6 |
+-----------------------------------------------------------+
1 row in set (0.01 sec)

mysql> SELECT DATE_FORMAT('1999-01-01', '%X %V');
+---------------------------------+
| date_format(1999-01-01, %X %V) |
+---------------------------------+
| 1998 52 |
+---------------------------------+
1 row in set (0.00 sec)
```

<!—SELECT DATE_FORMAT('2006-06-00', '%d'); ERROR 20301 (HY000): invalid input: invalid datatime value 2006-06-00—>

```
CREATE TABLE t2 (f1 DATETIME);
INSERT INTO t2 (f1) VALUES ('2005-01-01');
INSERT INTO t2 (f1) VALUES ('2005-02-01');

mysql> SELECT Date_format(f1, "%m") AS d1,
Date_format(f1, "%m") AS d2
FROM t2
ORDER BY Date_format(f1, "%m");
+------+------+
| d1 | d2 |
+------+------+
| 01 | 01 |
| 02 | 02 |
+------+------+
2 rows in set (0.00 sec)
```

```
CREATE TABLE t5 (a int, b date);
INSERT INTO t5
VALUES (1,
'2000-02-05'),
(2,
'2000-10-08'),
(3,
'2005-01-03'),
(4,
'2007-09-01'),
(5,
'2022-01-01');

mysql> SELECT * FROM t5
WHERE b = Date_format('20000205', '%Y-%m-%d');
+------+------------+
| a | b |
+------+------------+
| 1 | 2000-02-05 |
+------+------------+
1 row in set (0.01 sec)

mysql> SELECT * FROM t5
WHERE b != Date_format('20000205', '%Y-%m-%d');
+------+------------+
| a | b |
+------+------------+
| 2 | 2000-10-08 |
| 3 | 2005-01-03 |
| 4 | 2007-09-01 |
| 5 | 2022-01-01 |
+------+------------+
4 rows in set (0.01 sec)

mysql> SELECT DATE_FORMAT("2009-01-01",'%W %d %M %Y') as valid_date;
+-------------------------+
| valid_date |
+-------------------------+
| Thursday 01 January 2009 |
+-------------------------+
1 row in set (0.00 sec)
```

限制

目前 date 格式只支持 `yyyy-mm-dd` 和 `yyyymmdd` 的数据格式。

DATE_SUB()

函数说明

`DATE_SUB()` 都用于执行日期运算：`DATE_SUB()` 函数从日期减去指定的时间间隔。如果 `date` 为 `NULL`，函数返回 `NULL`。

函数语法

```
DATE_SUB(date,INTERVAL expr unit)
```

参数释义

参数	说明
date	必要参数。date 参数是合法的日期表达式。
expr	必要参数。expr 参数是需要添加进 date 的时间间隔，如果 expr 为负数，那么可以以“-”开头。
unit	必要参数。unit 参数可以是下列值： MICROSECOND SECOND MINUTE HOUR DAY WEEK MONTH QUA TER YEAR SECOND_MICROSECOND MINUTE_MICROSECOND MINUTE_SECOND HOUR_MICROSECOND HOUR_SECOND HOUR_MINUTE DAY_MICROSECOND DAY_SECOND DAY_MINUTE

参数	说明
	DAY_HOUR
	YEAR_MONTH

示例

```
create table t2(orderid int, productname varchar(20), orderdate
datetime);
insert into t2 values ('1','Jarl','2008-11-11 13:23:44.657');

mysql> SELECT OrderId,DATE_SUB(OrderDate,INTERVAL 5 DAY) AS SubtractDate
FROM t2;
+---------+---------------------+
| orderid | subtractdate |
+---------+---------------------+
| 1 | 2008-11-06 13:23:45 |
+---------+---------------------+
1 row in set (0.01 sec)
```

限制

目前 date 格式只支持 `yyyy-mm-dd` 和 `yyyymmdd` 的数据格式。

DATEDIFF()

函数说明

`DATEDIFF()` 函数返回两个日期之间的天数。

函数语法

```
> DATEDIFF(expr1,expr2)
```

参数释义

参数	说明
expr1, expr2	必要参数。expr1 和 expr2 参数是合法的日期或日期/时间表达式。只有值的日期部分参与计算。

示例

```
mysql> SELECT DATEDIFF('2007-12-31 23:59:59','2007-12-30');
+-----------------------------------------+
| datediff(2007-12-31 23:59:59, 2007-12-30) |
+-----------------------------------------+
| 1 |
+-----------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT DATEDIFF('2010-11-30 23:59:59','2010-12-31');
+-----------------------------------------+
| datediff(2010-11-30 23:59:59, 2010-12-31) |
+-----------------------------------------+
| -31 |
+-----------------------------------------+
1 row in set (0.00 sec)
```

```
create table t1(a INT, b date);
insert into t1 values(1, "2012-10-11");
insert into t1 values(2, "2004-04-24");
insert into t1 values(3, "2008-12-04");
insert into t1 values(4, "2012-03-23");
insert into t1 values(5, "2000-03-23");
insert into t1 values(6, "2030-03-23");
insert into t1 values(7, "2040-03-23");

mysql> SELECT a, DATEDIFF('2022-10-9', b) from t1;
+------+------------------------+
| a | datediff(2022-10-9, b) |
+------+------------------------+
| 1 | 3650 |
| 2 | 6742 |
| 3 | 5057 |
| 4 | 3852 |
| 5 | 8235 |
| 6 | -2722 |
| 7 | -6375 |
+------+------------------------+
7 rows in set (0.01 sec)
```

DAY()

函数说明

返回日期，即返回月份的第几号，范围为 1 到 31；对于诸如 `0000-00-00` 或 `2008-00-00` 等包含 0 部分的日期，则返回 0。如果日期为 `NULL` 则返回 `NULL`。

函数语法

```
> DAY(date)
```

参数释义

参数	说明
date	必要参数。date 参数是合法的日期表达式。

示例

```
mysql> SELECT day('2007-02-03');
+-----------------+
| day(2007-02-03) |
+-----------------+
| 3 |
+-----------------+
1 row in set (0.01 sec)
```

```
CREATE TABLE t3(c1 TIMESTAMP NOT NULL);
INSERT INTO t3 VALUES('2000-01-01');
INSERT INTO t3 VALUES('1999-12-31');
INSERT INTO t3 VALUES('2000-01-01');
INSERT INTO t3 VALUES('2006-12-25');
INSERT INTO t3 VALUES('2008-02-29');

mysql> SELECT day(c1) from t3;
+---------+
| day(c1) |
+---------+
| 1 |
| 31 |
| 1 |
| 25 |
| 29 |
+---------+
5 rows in set (0.01 sec)
```

DAYOFYEAR()

函数说明

返回日期所对应在一年中的天数，返回值在 1-366 之间。

函数语法

```
> DAYOFYEAR(expr)
```

参数释义

参数	说明
expr	必要参数。需要提取天数的 date 格式的输入值

示例

```
drop table if exists t1;
create table t1(a date, b datetime,c varchar(30));
insert into t1 values('2022-01-01','2022-01-01 01:01:01','2022-01-01
01:01:01');
insert into t1 values('2022-01-01','2022-01-01 01:01:01','2022-01-01
01:01:01');
insert into t1 values(20220101,'2022-01-01 01:01:01','2022-13-13
01:01:01');
insert into t1 values('2022-01-02','2022-01-02 23:01:01','2022-01-01
23:01:01');
insert into t1 values('2021-12-31','2021-12-30 23:59:59','2021-12-30
23:59:59');
insert into t1 values('2022-06-30','2021-12-30 23:59:59','2021-12-30
23:59:59');

mysql> select distinct dayofyear(a) as dya from t1;
+------+
| dya |
+------+
| 1 |
| 2 |
| 365 |
| 181 |
+------+
4 rows in set (0.00 sec)

mysql> select * from t1 where dayofyear(a)>120;
+------------+---------------------+---------------------+
| a | b | c |
+------------+---------------------+---------------------+
| 2021-12-31 | 2021-12-30 23:59:59 | 2021-12-30 23:59:59 |
| 2022-06-30 | 2021-12-30 23:59:59 | 2021-12-30 23:59:59 |
+------------+---------------------+---------------------+
2 rows in set (0.01 sec)

mysql> select * from t1 where dayofyear(a) between 1 and 184;
+------------+---------------------+---------------------+
| a | b | c |
+------------+---------------------+---------------------+
| 2022-01-01 | 2022-01-01 01:01:01 | 2022-01-01 01:01:01 |
| 2022-01-01 | 2022-01-01 01:01:01 | 2022-01-01 01:01:01 |
| 2022-01-01 | 2022-01-01 01:01:01 | 2022-13-13 01:01:01 |
```

```
| 2022-01-02 | 2022-01-02 23:01:01 | 2022-01-01 23:01:01 |
| 2022-06-30 | 2021-12-30 23:59:59 | 2021-12-30 23:59:59 |
+------------+---------------------+---------------------+
4 rows in set (0.00 sec)
```

限制

- 目前 `DAYOFYEAR()` 只支持 `date` 类型。
- 目前 `date` 格式只支持 `yyyy-mm-dd` 和 `yyyymmdd` 的数据格式。

EXTRACT()

函数说明

`EXTRACT()` 函数是从日期中提取部分内容。如果日期是 `NULL` 则返回 NULL。

函数语法

```
> EXTRACT(unit FROM date)
```

参数释义

参数	说明
date	必要参数。date 参数是合法的日期表达式。
unit	必要参数。unit 参数可以是下列值： MICROSECOND SECOND MINUTE HOUR DAY WEEK MONTH QUA TER YEAR SECOND_MICROSECOND MINUTE_MICROSECOND MINUTE_SECOND HOUR_MICROSECOND HOUR_SECOND HOUR_MINUTE DAY_MICROSECOND DAY_SECOND DAY_MINUTE DAY_HOUR YEAR_MONTH

示例

```
create table t2(orderid int, productname varchar(20), orderdate
datetime);
insert into t2 values ('1','Jarl','2008-11-11 13:23:44.657');

mysql> SELECT EXTRACT(YEAR FROM OrderDate) AS OrderYear, EXTRACT(MONTH
FROM OrderDate) AS OrderMonth FROM t2 WHERE OrderId=1;
+-----------+------------+
| orderyear | ordermonth |
+-----------+------------+
| 2008 | 11 |
+-----------+------------+
1 row in set (0.01 sec)
```

限制

目前 date 格式只支持 `yyyy-mm-dd` 和 `yyyymmdd` 的数据格式。

FROM_UNIXTIME()

函数说明

`FROM_UNIXTIME()` 函数把内部 UNIX 时间戳值转换为普通格式的日期时间值，以 *YYYY-MM-DD HH:MM*

或 *YYYYMMDDHHMMSS* 格式来显示。与 **`UNIX_TIMESTAMP ()`** 函数互为反函数。

函数语法

```
> FROM_UNIXTIME(unix_timestamp[,format])
```

参数释义

参数	说明
format	可选参数。表示返回值格式的格式字符串。 如果省略 format，则返回一个 `DATETIME` 值。 如果 format 为空，则返回 `NULL` 。 format 用于格式化结果，其方式与 `DATE_FORMAT()` 函数使用的格式字符串相同。如果 format 已存在指定格式，则返回值为 `VARCHAR` 。
unix_timestamp	必要参数。时间戳，可以用数据库里的存储时间数据的字段。 如果 unix_timestamp 为空，则返回 `NULL` 。 如果 unix_timestamp 是一个整数，则 `DATETIME` 的小数秒精度为零。当 unix_timestamp 是十进制值时， `DATETIME` 的小数秒精度与十进制值的精度相同，最多可达 6 秒。当 `unix_timestamp` 是浮点数时， `datetime` 的分秒精度为 6。

示例

```
mysql> SELECT FROM_UNIXTIME(1447430881);
+---------------------------+
| from_unixtime(1447430881) |
+---------------------------+
| 2015-11-14 00:08:01 |
+---------------------------+
1 row in set (0.00 sec)

mysql> SELECT FROM_UNIXTIME(1447430881, '%Y %D %M %h:%i:%s %x');
+-------------------------------------------------+
| from_unixtime(1447430881, %Y %D %M %h:%i:%s %x) |
+-------------------------------------------------+
| 2015 14th November 12:08:01 2015 |
+-------------------------------------------------+
1 row in set (0.00 sec)
```

限制

目前 date 格式只支持 `yyyy-mm-dd` 和 `yyyymmdd` 的数据格式。

MONTH()

函数说明

返回日期的月份，对于 1 月到 12 月，返回范围为 1 到 12；对于 “0000-00-00” 或 “2008-00-00” 等日期，返回 0 月份。如果 `date` 为 `NULL` 则返回 `NULL`。

函数语法

```
> MONTH(date)
```

参数释义

参数	说明
date	必要参数。date 参数是合法的日期表达式。

示例

- 示例 1：

```
mysql> SELECT MONTH('2008-02-03');
+-------------------+
| month(2008-02-03) |
+-------------------+
| 2 |
+-------------------+
1 row in set (0.02 sec)
```

- 示例 2：

```
drop table if exists t1;
create table t1 (id int,d date, dt datetime,c char(10),vc varchar(20));
insert into t1 values (1,"2021-01-13", "2021-01-13 13:00:00", "2021-12-15", "2021-12-16");
insert into t1 values (1,"2021-01-31", "2021-01-31 13:00:00", "2021-12-15", "2021-12-16");
insert into t1 values (2,"2022-02-15", "2022-02-15 18:54:29", "2021-02-15", "2021-02-15");
insert into t1 values (2,"2022-02-28", "2022-02-28 18:54:29", "2021-02-15", "2021-02-15");
insert into t1 values (3,"2000-02-29", "2000-02-29 18:54:29", "2021-02-15", "2021-02-15");
insert into t1 values (4,"2023-03-17", "2021-02-17 23:54:59", "2021-03-17", "2021-03-17");
insert into t1 values (5,"1985-04-18", "1985-04-18 00:00:01", "1985-04-18", "1985-04-18");
insert into t1 values (6,"1987-05-20", "1987-05-20 22:59:59", "1987-05-20", "1987-05-20");
insert into t1 values (7,"1989-06-22", "1989-06-22 15:00:30", "1989-06-22", "1989-06-22");
insert into t1 values (8,"1993-07-25", "1987-07-25 03:04:59", "1993-07-25", "1993-07-25");
insert into t1 values (9,"1995-08-27", "1987-08-27 04:32:33", "1995-08-27", "1995-08-27");
insert into t1 values (10,"1999-09-30", "1999-09-30 10:11:12", "1999-09-30", "1999-09-30");
insert into t1 values (11,"2005-10-30", "2005-10-30 18:18:59", "2005-10-30", "2005-10-30");
insert into t1 values (12,"2008-11-30", "2008-11-30 22:59:59", "2008-11-30", "2008-11-30");
insert into t1 values (13,"2013-12-01", "2013-12-01 22:59:59", "2013-12-01", "2013-12-01");
insert into t1 values (14,null, null, null, null);

mysql> select month(d),month(dt) from t1;
+----------+-----------+
| month(d) | month(dt) |
+----------+-----------+
| 1 | 1 |
| 1 | 1 |
| 2 | 2 |
| 2 | 2 |
| 2 | 2 |
| 3 | 2 |
```

```
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |
| 8 | 8 |
| 9 | 9 |
| 10 | 10 |
| 11 | 11 |
| 12 | 12 |
| NULL | NULL |
+----------+-----------+
16 rows in set (0.01 sec)

mysql> select month(c),month(vc) from t1;
+----------+-----------+
| month(c) | month(vc) |
+----------+-----------+
| 12 | 12 |
| 12 | 12 |
| 2 | 2 |
| 2 | 2 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |
| 8 | 8 |
| 9 | 9 |
| 10 | 10 |
| 11 | 11 |
| 12 | 12 |
| NULL | NULL |
+----------+-----------+
16 rows in set (0.01 sec)
```

TIMEDIFF()

函数说明

`TIMEDIFF()` 返回两个 `TIME` 或 `DATETIME` 值之间的差值。`TIMEDIFF()` 函数的两个表达式必须为相同类型的参数，即 `TIME` 或 `DATETIME`。`TIMEDIFF` 函数返回表示为时间值的 `expr1 - expr2` 的结果。

函数语法

```
> TIMEDIFF(expr1,expr2)
```

参数释义

参数	说明
expr1, expr2	必要参数。expr1 和 expr2 表达式需要具有相同的类型。expr1 和 expr2 是转换为 TIME 或 DATETIME 表达式的字符串。如果 expr1 或 expr2 为 NULL 则返回 NULL。

示例

```
mysql> select timediff("22:22:22", "11:00:00");
+------------------------------+
| timediff(22:22:22, 11:00:00) |
+------------------------------+
| 11:22:22.000000 |
+------------------------------+
1 row in set (0.01 sec)

mysql> select timediff(cast('22:22:22' as time), null);
+--------------------------------------------+
| timediff(cast(22:22:22 as time(26)), null) |
+--------------------------------------------+
| NULL |
+--------------------------------------------+
1 row in set (0.00 sec)

mysql> select timediff(CAST('2017-08-08 22:22:22' as datetime),
CAST('2000-01-02 11:00:00' as datetime));
+------------------------------------------------------------------------
------------------------+
| timediff(cast(2017-08-08 22:22:22 as datetime(26)), cast(2000-01-02
11:00:00 as datetime(26))) |
+------------------------------------------------------------------------
------------------------+
| 154283:22:22 |
+------------------------------------------------------------------------
------------------------+
1 row in set (0.00 sec)
```

```
create table time_01(t1 time,t2 time,t3 time);
insert into time_01 values("-838:59:59.0000","838:59:59.00","22:00:00");
insert into time_01 values("0:00:00.0000","0","0:00");
insert into time_01 values(null,NULL,null);
insert into time_01 values("23","1122","-1122");
insert into time_01 values("101412","4","-101219");
insert into time_01
values("24:59:09.932823","24:02:00.93282332424","24:20:34.00000000");
insert into time_01 values("2022-09-08 12:00:01","019","23403");

mysql> select * from time_01;
+------------+-----------+-----------+
| t1 | t2 | t3 |
+------------+-----------+-----------+
| -838:59:59 | 838:59:59 | 22:00:00 |
| 00:00:00 | 00:00:00 | 00:00:00 |
| NULL | NULL | NULL |
| 00:00:23 | 00:11:22 | -00:11:22 |
| 10:14:12 | 00:00:04 | -10:12:19 |
| 24:59:10 | 24:02:01 | 24:20:34 |
| 12:00:01 | 00:00:19 | 02:34:03 |
+------------+-----------+-----------+
7 rows in set (0.00 sec)

mysql> select timediff(t1,t2) from time_01;
+------------------+
| timediff(t1, t2) |
+------------------+
| -1677:59:58 |
| 00:00:00 |
| NULL |
| -00:10:59 |
| 10:14:08 |
| 00:57:09 |
| 11:59:42 |
+------------------+
7 rows in set (0.00 sec)
```

TIMESTAMP()

函数说明

单个参数时，`TIMESTAMP()` 将日期或日期时间表达式 expr 作为日期时间值返回。两个参数时，`TIMESTAMP()` 将时间表达式 expr2 添加到日期或 datetime 表达式 expr1 中，并将结果作为 datetime 值返回。如果 expr、expr1 或 expr2 为 `NULL`，则返回 `NULL`。

函数语法

```
> TIMESTAMP(expr), TIMESTAMP(expr1,expr2)
```

参数释义

参数	说明
expr	必要参数。expr 参数是需要添加进 date 的时间间隔，如果 expr 为负数，那么可以以“-”开头。

示例

```
mysql> SELECT TIMESTAMP('2003-12-31');
+----------------------------+
| timestamp(2003-12-31) |
+----------------------------+
| 2003-12-31 00:00:00.000000 |
+----------------------------+
1 row in set (0.00 sec)
```

```
CREATE TABLE t1(c1 DATE NOT NULL);
INSERT INTO t1 VALUES('2000-01-01');
INSERT INTO t1 VALUES('1999-12-31');
INSERT INTO t1 VALUES('2000-01-01');
INSERT INTO t1 VALUES('2006-12-25');
INSERT INTO t1 VALUES('2008-02-29');

mysql> SELECT TIMESTAMP(c1) FROM t1;
+----------------------------+
| timestamp(c1) |
+----------------------------+
| 2000-01-01 00:00:00.000000 |
| 1999-12-31 00:00:00.000000 |
| 2000-01-01 00:00:00.000000 |
| 2006-12-25 00:00:00.000000 |
| 2008-02-29 00:00:00.000000 |
+----------------------------+
5 rows in set (0.00 sec)
```

限制

`TIMESTAMP()` 暂不支持双参数，即暂不支持 `TIMESTAMP(expr1,expr2)`。

TO_DATE()

函数说明

`TO_DATE()` 函数按照指定日期或时间显示格式，将字符串转换为日期或日期时间类型。

格式字符串可以包含文字字符和以%开头的格式说明符。format 中的字面字符必须匹配 str 中的字面字符。format 中的格式说明符必须匹配 str 中的日期或时间部分。

函数语法

```
> TO_DATE(str,format)
```

参数释义

Arguments	Description
str	Required. 如果 `str` 为 `NULL`，则函数返回 `NULL`。 如果从 `str` 中的 `date` 或 `datetime` 值不合法，则 `STR_TO_DATE()` 将返回 `NULL` 并产生警告。
format	可选参数。表示返回值格式的格式字符串。 如果省略 format，则返回一个 `DATETIME` 值。 如果 format 为空，则返回 `NULL`。 如果 format 已存在指定格式，则返回值为 `VARCHAR`。

说明：格式字符串可以包含文字字符和以 % 开头的格式说明符。`format` 中的字面字符必须匹配 `str` 中的字面字符。`format` 中的格式说明符必须匹配 `str` 中的日期或时间部分。

示例

```
mysql> SELECT TO_DATE('2022-01-06 10:20:30','%Y-%m-%d %H:%i:%s') as
result;
+---------------------+
| result |
+---------------------+
| 2022-01-06 10:20:30 |
+---------------------+
1 row in set (0.00 sec)
```

限制

目前 date 格式只支持 `yyyy-mm-dd` 和 `yyyymmdd` 的数据格式。

UNIX_TIMESTAMP()

函数说明

`UNIX_TIMESTAMP()` 返回自 *1970-01-01 00:00:00* UTC 至当前时间的秒数。

`UNIX_TIMESTAMP(date)` 将参数的值返回为 *1970-01-01 00:00:00* UTC 至 *date* 指定时间的秒数。

如果日期超出范围传递给 `UNIX_TIMESTAMP()`，它将返回 0。如果 `date` 为 `NULL`，则返回 `NULL`。

如果没有给出参数或参数不包含小数秒部分，则返回值为整数；如果给出参数包含小数秒部分，则返回值为 `DECIMAL`。

函数语法

```
> UNIX_TIMESTAMP([date])
```

参数释义

参数	说明
date	可选参数。date 参数是合法的日期表达式。 date 参数可以是 `DATE`、`DATETIME` 或 `TIMESTAMP` 字符串，也可以是 *YYMMDD*、*YYMMDDhhmmss*、*YYYYMMDD* 或 *YYYYMMDDhhmmss* 格式的数字。如果 date 参数包含时间部分，则它有选择地包含秒的小数部分。 当 date 参数是 `TIMESTAMP` 时，`UNIX_TIMESTAMP()` 直接返回内部时间戳值，而不进行隐含的 *string-to-Unix-timestamp* 转换。

非 UTC 时区的值和 Unix 时间戳值之间的转换

如果使用 `UNIX_TIMESTAMP()` 和 `FROM_UNIXTIME()` 在非 UTC（Coordinated Universal Time，协调世界时）时区的值和 Unix 时间戳值之间进行转换，则转换是有损的，因为映射在两个方向上不是一对一的。例如，由于诸如夏令时 (DST) 等本地时区更改的约定，`UNIX_TIMESTAMP()` 可以将在非 UTC 时区中不同的两个值映射到相同的 Unix 时间戳值。`FROM_UNIXTIME()` 将该值仅映射回原始值之一。下面的示例，即，使用在 `MET` 时区不同的值：

```
mysql> SET time_zone = 'MET';
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT UNIX_TIMESTAMP('2005-03-27 03:00:00');
+-------------------------------------+
| unix_timestamp(2005-03-27 03:00:00) |
+-------------------------------------+
| 1111885200 |
+-------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT UNIX_TIMESTAMP('2005-03-27 02:00:00');
+-------------------------------------+
| unix_timestamp(2005-03-27 02:00:00) |
+-------------------------------------+
| 1111885200 |
+-------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT FROM_UNIXTIME(1111885200);
+---------------------------+
| from_unixtime(1111885200) |
+---------------------------+
| 2005-03-27 03:00:00 |
+---------------------------+
1 row in set (0.00 sec)
```

示例

```
mysql> SELECT UNIX_TIMESTAMP("2016-07-11");
+----------------------------+
| unix_timestamp(2016-07-11) |
+----------------------------+
| 1468188000 |
+----------------------------+
1 row in set (0.00 sec)

mysql> SELECT UNIX_TIMESTAMP('2015-11-13 10:20:19');
+-------------------------------------+
| unix_timestamp(2015-11-13 10:20:19) |
+-------------------------------------+
| 1447406419 |
+-------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT UNIX_TIMESTAMP('2015-11-13 10:20:19.012');
+----------------------------------------+
| unix_timestamp(2015-11-13 10:20:19.012) |
+----------------------------------------+
| 1447406419.012000 |
+----------------------------------------+
1 row in set (0.00 sec)
```

限制

目前 date 格式只支持 `yyyy-mm-dd` 和 `yyyymmdd` 的数据格式。

UTC_TIMESTAMP()

函数说明

将当前 UTC 时间以 `YYYY-MM-DD hh:mm:ss` 或 `YYYYMMDDhhmmss` 的格式返回，返回格式取决于函数是在字符串还是数字。

函数语法

```
> UTC_TIMESTAMP, UTC_TIMESTAMP([fsp])
```

参数释义

参数	说明
fsp	可选。参数 `fsp` 参数用于指定分秒精度，有效值为 0 到 6 之间的整数。

示例

- 示例 1:

```
mysql> SELECT UTC_TIMESTAMP();
+---------------------+
| utc_timestamp() |
+---------------------+
| 2022-09-16 03:37:40 |
+---------------------+
1 row in set (0.01 sec)

mysql> select unix_timestamp(utc_timestamp());
+--------------------------------+
| unix_timestamp(utc_timestamp()) |
+--------------------------------+
| 1663282842 |
+--------------------------------+
1 row in set (0.02 sec)
```

- 示例 2：

```
create table t1 (ts timestamp);
set time_zone='+00:00';

mysql> select unix_timestamp(utc_timestamp())-
unix_timestamp(utc_timestamp());
+-----------------------------------------------------------------------+
| unix_timestamp(utc_timestamp()) - unix_timestamp(utc_timestamp()) |
+-----------------------------------------------------------------------+
| 0 |
+-----------------------------------------------------------------------+
1 row in set (0.00 sec)

insert into t1 (ts) values ('2003-03-30 02:30:00');
set time_zone='+10:30';

mysql> select unix_timestamp(utc_timestamp())-
unix_timestamp(utc_timestamp());
+-----------------------------------------------------------------------+
| unix_timestamp(utc_timestamp()) - unix_timestamp(utc_timestamp()) |
+-----------------------------------------------------------------------+
| 0 |
+-----------------------------------------------------------------------+
1 row in set (0.01 sec)

insert into t1 (ts) values ('2003-03-30 02:30:00');
set time_zone='-10:00';

mysql> select unix_timestamp(utc_timestamp())-
unix_timestamp(current_timestamp());
+-------------------------------------------------------------------------+
| unix_timestamp(utc_timestamp()) - unix_timestamp(current_timestamp()) |
+-------------------------------------------------------------------------+
| 36000 |
+-------------------------------------------------------------------------+
1 row in set (0.00 sec)

insert into t1 (ts) values ('2003-03-30 02:30:00');

mysql> select * from t1;
+---------------------+
| ts |
+---------------------+
| 2003-03-29 16:30:00 |
| 2003-03-29 06:00:00 |
```

```
| 2003-03-30 02:30:00 |
+---------------------+
3 rows in set (0.00 sec)
```

- 示例 3：

```
DROP TABLE IF EXISTS t1;
CREATE TABLE t1 (a TIMESTAMP);
INSERT INTO t1 select (utc_timestamp());
INSERT INTO t1 select (utc_timestamp());
INSERT INTO t1 select (utc_timestamp());

mysql> SELECT year(a) FROM t1 WHERE a > '2008-01-01';
+---------+
| year(a) |
+---------+
| 2022 |
| 2022 |
| 2022 |
+---------+
3 rows in set (0.04 sec)
```

```
DROP TABLE if exists t1;
create table t1 (a int primary key, b int, c int, d timestamp);
insert into t1 select 1,1,1,utc_timestamp();
insert into t1 select 2,0,0,null;

mysql> select a,b,c,year(d) from t1;
+------+------+------+---------+
| a | b | c | year(d) |
+------+------+------+---------+
| 1 | 1 | 1 | 2022 |
| 2 | 0 | 0 | NULL |
+------+------+------+---------+
2 rows in set (0.01 sec)
```

```
DROP TABLE if exists t1;
CREATE TABLE t1 (a TIMESTAMP);
INSERT INTO t1 select (utc_timestamp());
INSERT INTO t1 select (utc_timestamp());

mysql> SELECT 1 FROM t1 ORDER BY 1;
+------+
| 1 |
+------+
| 1 |
| 1 |
+------+
2 rows in set (0.01 sec)
```

限制

暂不支持操作符 `+` 或 `-` 对 `DATETIME BIGINT` 参数进行运算。

WEEKDAY()

函数说明

返回日期的工作日索引（0 = 星期一，1 = 星期二，... 6 = 星期日）。如果日期为 `NULL`，则返回 `NULL`。

函数语法

```
> WEEKDAY(date)
```

参数释义

参数	说明
date	必要参数。date 参数是合法的日期表达式。

示例

- 示例 1：

```
mysql> SELECT WEEKDAY('2008-02-03 22:23:00');
+------------------------------+
| weekday(2008-02-03 22:23:00) |
+------------------------------+
| 6 |
+------------------------------+
1 row in set (0.03 sec)
```

- 示例 2：

```
drop table if exists t1;
create table t1 (id int,d date, dt datetime,c char(10),vc varchar(20));
insert into t1 values (1,"2021-01-13", "2021-01-13 13:00:00", "2021-12-15", "2021-12-16");
insert into t1 values (1,"2021-01-31", "2021-01-31 13:00:00", "2021-12-15", "2021-12-16");
insert into t1 values (2,"2022-02-15", "2022-02-15 18:54:29", "2021-02-15", "2021-02-15");
insert into t1 values (2,"2022-02-28", "2022-02-28 18:54:29", "2021-02-15", "2021-02-15");
insert into t1 values (3,"2000-02-29", "2000-02-29 18:54:29", "2021-02-15", "2021-02-15");
insert into t1 values (4,"2023-03-17", "2021-02-17 23:54:59", "2021-03-17", "2021-03-17");
insert into t1 values (5,"1985-04-18", "1985-04-18 00:00:01", "1985-04-18", "1985-04-18");
insert into t1 values (6,"1987-05-20", "1987-05-20 22:59:59", "1987-05-20", "1987-05-20");
insert into t1 values (7,"1989-06-22", "1989-06-22 15:00:30", "1989-06-22", "1989-06-22");
insert into t1 values (8,"1993-07-25", "1987-07-25 03:04:59", "1993-07-25", "1993-07-25");
insert into t1 values (9,"1995-08-27", "1987-08-27 04:32:33", "1995-08-27", "1995-08-27");
insert into t1 values (10,"1999-09-30", "1999-09-30 10:11:12", "1999-09-30", "1999-09-30");
insert into t1 values (11,"2005-10-30", "2005-10-30 18:18:59", "2005-10-30", "2005-10-30");
insert into t1 values (12,"2008-11-30", "2008-11-30 22:59:59", "2008-11-30", "2008-11-30");
insert into t1 values (13,"2013-12-01", "2013-12-01 22:59:59", "2013-12-01", "2013-12-01");
insert into t1 values (14,null, null, null, null);

mysql> select weekday(d),weekday(dt) from t1;
+------------+-------------+
| weekday(d) | weekday(dt) |
+------------+-------------+
| 2 | 2 |
| 6 | 6 |
| 1 | 1 |
| 0 | 0 |
| 1 | 1 |
| 4 | 2 |
```

```
| 3 | 3 |
| 2 | 2 |
| 3 | 3 |
| 6 | 5 |
| 6 | 3 |
| 3 | 3 |
| 6 | 6 |
| 6 | 6 |
| 6 | 6 |
| NULL | NULL |
+------------+-------------+
16 rows in set (0.01 sec)

mysql> select weekday(c),weekday(vc) from t1;
+------------+-------------+
| weekday(c) | weekday(vc) |
+------------+-------------+
| 2 | 3 |
| 2 | 3 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 2 | 2 |
| 3 | 3 |
| 2 | 2 |
| 3 | 3 |
| 6 | 6 |
| 6 | 6 |
| 3 | 3 |
| 6 | 6 |
| 6 | 6 |
| 6 | 6 |
| NULL | NULL |
+------------+-------------+
16 rows in set (0.02 sec)
```

YEAR()

函数说明

`YEAR()` 和 `TOYEAR()` 函数返回了给定日期的年份（从 1000 到 9999）。

函数语法

```
> YEAR(date)
> TOYEAR(date)
```

参数释义

参数	说明
date	必要参数，需要提取年份的日期

示例

```
drop table if exists t1;
create table t1(a date, b datetime);
insert into t1 values('20211223','2021-10-22 09:23:23');
insert into t1 values('2021-12-23','2021-10-22 00:23:23');

mysql> select year(a) from t1;
+---------+
| year(a) |
+---------+
| 2021 |
| 2021 |
+---------+
2 rows in set (0.00 sec)
```

```
DROP TABLE IF EXISTS t3;
CREATE TABLE t3(c1 DATE NOT NULL);
INSERT INTO t3 VALUES('2000-01-01');
INSERT INTO t3 VALUES('1999-12-31');
INSERT INTO t3 VALUES('2000-01-01');
INSERT INTO t3 VALUES('2006-12-25');
INSERT INTO t3 VALUES('2008-02-29');
mysql> SELECT YEAR(c1) FROM t3;
+----------+
| year(c1) |
+----------+
| 2000 |
| 1999 |
| 2000 |
| 2006 |
| 2008 |
+----------+
5 rows in set (0.01 sec)
```

限制

目前只支持 `yyyy-mm-dd` 和 `yyyymmddd` 的数据格式。

BIN()

函数说明

`BIN()` 将 *N* 转换为二进制的字符串形式。其中 *N* 是一个 `longlong (BIGINT)` 数字。如果 *N* 为 *NULL*，则返回 `NULL`。

语法

```
> BIN(N)
```

参数释义

参数	说明
N	必要参数。UINT 类型

示例

```
> SELECT bin(1314);
+-------------+
| bin(1314) |
+-------------+
| 10100100010 |
+-------------+
1 row in set (0.01 sec)

> select bin(2e5);
+--------------------+
| bin(2e5) |
+--------------------+
| 110000110101000000 |
+--------------------+
1 row in set (0.00 sec)
```

BIT_LENGTH()

函数说明

返回字符串 str 的长度，单位为 bit。如果 str 为 `NULL` 则返回 `NULL`。

语法说明

```
> BIT_LENGTH(str)
```

参数释义

参数	说明
str	必要参数，想要计算长度的字符串

示例

```
> SELECT BIT_LENGTH('text');
+------------------+
| bit_length(text) |
+------------------+
| 32 |
+------------------+
1 row in set (0.00 sec)
```

CHAR_LENGTH()

函数说明

`CHAR_LENGTH` 以字符为单位返回字符串 `str` 的长度，一个多字节字符算作一个字符。一个汉字所对应的字符长度是 1。

函数语法

```
> CHAR_LENGTH(str)
```

参数释义

参数	说明
str	必要参数，想要计算长度的字符串

> **“<font size=4>note</font>”**
>
> <font size=3> `CHAR_LENGTH` 也可以写为 `lengthUTF8()` 。</font>

示例

```
> drop table if exists t1;
> create table t1(a varchar(255),b varchar(255));
> insert into t1 values('nihao','你好');
> select char_length(a), char_length(b) from t1;
+---------------+---------------+
| lengthutf8(a) | lengthutf8(b) |
+---------------+---------------+
| 5 | 2 |
+---------------+---------------+
```

CONCAT()

函数说明

`CONCAT()` 将多个字符串（或仅含有一个字符串）连接成一个字符串。如果所有参数都是非二进制字符串，则结果是非二进制字符串。如果参数包含任何二进制字符串，则结果为二进制字符串。

`CONCAT()` 中如果有任何一个参数为 `NULL`，则返回 `NULL`。

语法说明

```
>
CONCAT(str1,str2,...)
```

参数释义

参数	说明
str1,str2,...	必要参数。将 `str1,str2,..` 连接成一个字符串。 说明：如果有任何一个参数为 `NULL`，则返回 `NULL`。

示例

```
mysql> SELECT CONCAT('My', 'S', 'QL');
+------------------+
| concat(My, S, QL) |
+------------------+
| MySQL |
+------------------+
1 row in set (0.01 sec)

mysql> SELECT CONCAT('My', NULL, 'QL');
+---------------------+
| concat(My, null, QL) |
+---------------------+
| NULL |
+---------------------+
1 row in set (0.00 sec)
```

限制

`CONCAT()` 当前还不支持带引号的字符串参数和数字参数。

CONCAT_WS()

函数说明

`CONCAT_WS()` 代表 Concatenate With Separator，是 `CONCAT()` 的一种特殊形式。第一个参数是其它参数的分隔符。分隔符的位置放在要连接的两个字符串之间。分隔符可以是字符串，也可以是其他参数。如果分隔符为 `NULL`，则结果为 `NULL`。函数会忽略任何分隔符参数后的 `NULL` 值。

函数语法

- 函数语法 1

```
> CONCAT_WS(separator,str1,str2,...)
```

- 函数语法 2

```
> CONCAT_WS(separator,str1,NULL,str1,...);
```

参数释义

参数	说明
Str	必要参数。需要翻转的字符串。CHAR 与 VARCHAR 类型均可。

示例

```
SELECT CONCAT_WS(',','First name','Second name','Last Name');
+-------------------------------------------------+
| concat_ws(,, First name, Second name, Last Name) |
+-------------------------------------------------+
| First name,Second name,Last Name |
+-------------------------------------------------+
1 row in set (0.01 sec)
> SELECT CONCAT_WS(',','First name',NULL,'Last Name');
+-----------------------------------------+
| concat_ws(,, First name, null, Last Name) |
+-----------------------------------------+
| First name,Last Name |
+-----------------------------------------+
1 row in set (0.01 sec)
```

EMPTY()

函数说明

判断输入的字符串是否为空。如果包含至少一个字节则不为空，即使是一个空格或者 NULL。

函数语法

```
> EMPTY(str)
```

参数释义

参数	说明
str	必要参数，CHAR 与 VARCHAR 类型均可。

返回值

空字符串返回 1，非空字符串返回 0。

示例

```
> drop table if exists t1;
> create table t1(a varchar(255),b varchar(255));
> insert into t1 values('', 'abcd');
> insert into t1 values('1111', '');
> select empty(a),empty(b) from t1;
+----------+----------+
| empty(a) | empty(b) |
+----------+----------+
| 1 | 0 |
| 0 | 1 |
+----------+----------+
```

ENDSWITH()

函数说明

检查是否以指定后缀结尾。字符串如果以指定后缀结尾返回 1，否则则返回 0。该函数是对大小写敏感的。

函数语法

```
> ENDSWITH(str,suffix)
```

参数释义

参数	说明
str	必要参数。CHAR 和 VARCHAR 类型都支持。
suffix	必要参数。CHAR 和 VARCHAR 类型都支持。

返回值

- 1，如果字符串是以指定后缀结尾的。
- 0，如果字符串不以指定后缀结尾的。

示例

```
> drop table if exists t1;
> create table t1(a int,b varchar(100),c char(20));
> insert into t1 values
(1,'Ananya Majumdar', 'XI'),
(2,'Anushka Samanta', 'X'),
(3,'Aniket Sharma', 'XI'),
(4,'Anik Das', 'X'),
(5,'Riya Jain', 'IX'),
(6,'Tapan Samanta', 'XI');
> select a,endsWith(b,'a') from t1;
+------+----------------+
| a | endswith(b, a) |
+------+----------------+
| 1 | 0 |
| 2 | 1 |
| 3 | 1 |
| 4 | 0 |
| 5 | 0 |
| 6 | 1 |
+------+----------------+
> select a,b,c from t1 where endswith(b,'a')=1 and endswith(c,'I')=1;
+------+---------------+------+
| a | b | c |
+------+---------------+------+
| 3 | Aniket Sharma | XI |
| 6 | Tapan Samanta | XI |
+------+---------------+------+
```

FIELD()

函数说明

`FIELD()` 函数返回第一个字符串 str 在字符串列表 (str1,str2,str3,...) 中的位置。

语法说明

```
> FIELD(str,str1,str2,str3,...)
```

参数释义

参数	说明
str	必要参数。要在列表中查找的值。
str1,str2,str3,...	必要参数。被搜索的列表中的各个元素。

返回值

如果 `FIELD()` 的所有参数都是 string 类型，则所有参数都作为 string 类型进行比较。如果所有参数都是数字，则将它们作为数字进行比较。如果所有参数都是 double 类型，则将它们作为 double 类型进行比较。

- 如果在列表中找到指定的值，`FIELD()` 函数返回对应的位置索引。`FIELD()` 函数返回的索引的值从 1 开始。
- 如果在列表中找不到指定的值，`FIELD()` 函数返回 0。
- 如果要查找的值为 `NULL`，`FIELD()` 函数返回 0。

示例

- 示例 1：

```
mysql> SELECT FIELD('Bb', 'Aa', 'Bb', 'Cc', 'Dd', 'Ff');
+-------------------------------+
| field(Bb, Aa, Bb, Cc, Dd, Ff) |
+-------------------------------+
| 2 |
+-------------------------------+
1 row in set (0.00 sec)

mysql> SELECT FIELD('Gg', 'Aa', 'Bb', 'Cc', 'Dd', 'Ff');
+-------------------------------+
| field(Gg, Aa, Bb, Cc, Dd, Ff) |
+-------------------------------+
| 0 |
+-------------------------------+
1 row in set (0.00 sec)
```

- 示例 2：

```
drop table if exists t;
create table t(
i int,
f float,
d double
);
insert into t() values (1, 1.1, 2.2), (2, 3.3, 4.4), (0, 0, 0), (0, null,
0);

mysql> select * from t;
+------+------+------+
| i | f | d |
+------+------+------+
| 1 | 1.1 | 2.2 |
| 2 | 3.3 | 4.4 |
| 0 | 0 | 0 |
| 0 | NULL | 0 |
+------+------+------+
4 rows in set (0.01 sec)

mysql> select field(1, i, f, d) from t;
+-------------------+
| field(1, i, f, d) |
+-------------------+
| 1 |
| 0 |
| 0 |
| 0 |
+-------------------+
4 rows in set (0.01 sec)

mysql> select field(i, f, d, 0, 1, 2) from t;
+-------------------------+
| field(i, f, d, 0, 1, 2) |
+-------------------------+
| 4 |
| 5 |
| 1 |
| 2 |
+-------------------------+
4 rows in set (0.01 sec)

mysql> select field('1', f, d, 0, 1, 2) from t;
+-------------------------+
```

```
| field(1, f, d, 0, 1, 2) |
+-------------------------+
| 4 |
| 4 |
| 4 |
| 4 |
+-------------------------+
4 rows in set (0.01 sec)
```

FIND_IN_SET()

函数说明

- str 为要查询的字符串
- strList 为字段名，参数以 “,” 分隔，如 (1,2,6,8)
- 查询字段 (strList) 中包含的结果，返回结果 NULL 或记录。

如果字符串 *str* 在由 N 个子字符串组成的字符串列表 *strlist* 中，则返回值的范围在 *1* 到 *N* 之间。一个字符串列表就是由 ‘,’ 符号分开的子字符串组成。如果第一个参数是常量字符串，第二个参数是 `SET` 类型的列，那么 `FIND_IN_SET()` 函数将优化为使用位运算。如果 *str* 不在 *strlist* 中或者 *strlist* 是空字符串，则返回 *0*。如果任一参数为 *NULL* 则返回 `NULL`。如果第一个参数包含逗号 `(,)` 字符，此函数将无法正常运行。

函数语法

```
> FIND_IN_SET(str,strlist)
```

参数释义

参数	说明
str	必要参数。CHAR 和 VARCHAR 类型都支持。
strlist	必要参数。

示例

```
select find_in_set('b','a,b,c,d');
+-------------------------+
| find_in_set(b, a,b,c,d) |
+-------------------------+
| 2 |
+-------------------------+
```

FORMAT()

函数说明

`FORMAT` 函数用于将数字格式设置为 "#,###,###.##" 格式，并四舍五入到小数点后一位。格式化数字后，它将以字符串的形式返回值。

语法

```
> FORMAT(X,D[,locale])
```

参数释义

参数	说明
X	必要参数。X 是要格式化的数字。如果 X 是 NULL，函数将返回 NULL。
D	必要参数。D 是要舍入小数位数 如果 D 是 0，则结果没有小数点或没有小数部分。 如果 D 为 NULL，函数返回 NULL。
[,locale]	可选参数。可选参数 [,locale] 指定要使用的语言环境，并且用于确定千个分隔符和分隔符之间的分组。如果 [locale] 设置为 NULL 或未指定，则默认语言环境为 'en_US'。

示例

```
mysql> SELECT FORMAT(12332.123456, 4);
+-------------------------+
| format(12332.123456, 4) |
+-------------------------+
| 12,332.1235 |
+-------------------------+
1 row in set (0.01 sec)

mysql> SELECT FORMAT(12332.1,4);
+--------------------+
| format(12332.1, 4) |
+--------------------+
| 12,332.1000 |
+--------------------+
1 row in set (0.00 sec)

mysql> SELECT FORMAT(12332.2,0);
+--------------------+
| format(12332.2, 0) |
+--------------------+
| 12,332 |
+--------------------+
1 row in set (0.00 sec)

mysql> SELECT FORMAT(12332.2,2,'de_DE');
+---------------------------+
| format(12332.2, 2, de_DE) |
+---------------------------+
| 12.332,20 |
+---------------------------+
1 row in set (0.00 sec)

mysql> SELECT FORMAT(19999999.999999999,4);
+------------------------------+
| format(19999999.999999999, 4) |
+------------------------------+
| 20,000,000.0000 |
+------------------------------+
1 row in set (0.01 sec)

mysql> SELECT FORMAT("-.12334.2","2", "en_US");
+----------------------------+
```

```
| format(-.12334.2, 2, en_US) |
+-----------------------------+
| -0.12 |
+-----------------------------+
1 row in set (0.00 sec)

mysql> SELECT FORMAT("-.12334.2","2", "de_CH");
+-----------------------------+
| format(-.12334.2, 2, de_CH) |
+-----------------------------+
| -0.12 |
+-----------------------------+
1 row in set (0.01 sec)
```

HEX()

函数说明

对于字符串参数 str，`HEX()` 返回 str 的十六进制字符串形式，其中 str 中每个字符的每个字节都转换为两个十六进制数字。(多字节字符转换成两个以上的数字。)

对于数值参数 N，`HEX()` 返回 N 值的十六进制字符串形式，且按照整数进行转换。这等同于 `CONV(N,10,16)`，相反的操作执行 `CONV(HEX(N),16,10)`。

对于 `NULL` 参数，此函数返回 `NULL`。

函数语法

```
> HEX(str), HEX(N)
```

参数释义

参数	说明
N	必要参数。要转换为十六进制的数字。
str	必要参数。字符串，其每个字符都要转换为两个十六进制数字。

示例

```
> SELECT HEX('abc');
+----------+
| hex(abc) |
+----------+
| 616263 |
+----------+
1 row in set (0.00 sec)

> SELECT HEX(255);
+----------+
| hex(255) |
+----------+
| FF |
+----------+
1 row in set (0.00 sec)
```

LEFT()

函数说明

`LEFT()` 函数返回 *str* 字符串中最左边的长度字符。如果 *str* 或 *len* 参数为 `NULL`，则返回 `NULL` 值。

函数语法

```
> LEFT(str,len)
```

参数释义

参数	说明
str	必要参数。要提取子字符串的字符串。
len	必要参数。是一个正整数，指定将从左边返回的字符数。 如果 *len* 为 0 或为负，则 LEFT 函数返回一个空字符串。 如果 *len* 大于 *str* 字符串的长度，则 LEFT 函数返回整个 *str* 字符串。

示例

```
mysql> select left('abcde', 3) from dual;
+----------------+
| left(abcde, 3) |
+----------------+
| abc |
+----------------+
1 row in set (0.00 sec)

drop table if exists t1;
CREATE TABLE t1 (str VARCHAR(100) NOT NULL, len INT);
insert into t1 values('abcdefghijklmn',3);
insert into t1 values(' ABCDEFGH123456', 3);
insert into t1 values('ABCDEF GHIJKLMN', 20);
insert into t1 values('ABCDEFGHijklmn ', -1);
insert into t1 values('ABCDEFGH123456', -35627164);
insert into t1 values('', 3);
mysql> select left(str, len) from t1;
+------------------+
| left(str, len) |
+------------------+
| abc |
| A |
| ABCDEF GHIJKLMN |
| |
| |
| |
+------------------+
6 rows in set (0.01 sec)

mysql> select left('sdfsdfsdfsdf', len) from t1;
+-------------------------+
| left(sdfsdfsdfsdf, len) |
+-------------------------+
| sdf |
| sdf |
| sdfsdfsdfsdf |
| |
| |
| sdf |
+-------------------------+
6 rows in set (0.01 sec)
```

LENGTH()

函数说明

`length()` 函数返回了字符串的长度。

函数语法

```
> LENGTH(str)
```

参数释义

参数	说明
str	必要参数，想要计算长度的字符串

示例

```
> select a,length(a) from t1;
a length(a)
a 1
ab 2
abc 3
```

LPAD()

函数说明

函数 LPAD(str,len,padstr) 在字符串 *str* 左侧使用 *padstr* 进行填充，直至总长度为 *len* 的字符串，最后返回填充后的字符串。如果 *str* 的长度大于 *len*，那么最后的长度将缩减至 *len*。
若 *len* 为负数，则返回 NULL。

函数语法

```
> LPAD(str,len,padstr)
```

参数释义

参数	说明
str	必要参数，被填充的字符串。CHAR 与 VARCHAR 类型均可。
len	必要参数，需要填充到的总长度。
padstr	必要参数，用于填充的字符串。CHAR 与 VARCHAR 类型均可。

示例

```
> drop table if exists t1;
> CREATE TABLE t1(Student_id INT,Student_name VARCHAR(100),Student_Class
CHAR(20));
> INSERT INTO t1
VALUES
(1,'Ananya Majumdar', 'IX'),
(2,'Anushka Samanta', 'X'),
(3,'Aniket Sharma', 'XI'),
(4,'Anik Das', 'X'),
(5,'Riya Jain', 'IX'),
(6,'Tapan Samanta', 'X');
> SELECT Student_id, Student_name,LPAD(Student_Class, 10, ' _') AS
LeftPaddedString FROM t1;
+------------+-----------------+------------------+
| Student_id | Student_name | LeftPaddedString |
+------------+-----------------+------------------+
| 1 | Ananya Majumdar | _ _ _ _IX |
| 2 | Anushka Samanta | _ _ _ _ X |
| 3 | Aniket Sharma | _ _ _ _XI |
| 4 | Anik Das | _ _ _ _ X |
| 5 | Riya Jain | _ _ _ _IX |
| 6 | Tapan Samanta | _ _ _ _ X |
+------------+-----------------+------------------+
> SELECT Student_id, lpad(Student_name,4,'new') AS LeftPaddedString FROM
t1;
+------------+------------------+
| Student_id | LeftPaddedString |
+------------+------------------+
| 1 | Anan |
| 2 | Anus |
| 3 | Anik |
| 4 | Anik |
| 5 | Riya |
| 6 | Tapa |
+------------+------------------+
> SELECT Student_id, lpad(Student_name,-4,'new') AS LeftPaddedString FROM
t1;
+------------+------------------+
| Student_id | LeftPaddedString |
+------------+------------------+
| 1 | NULL |
| 2 | NULL |
```

```
| 3 | NULL |
| 4 | NULL |
| 5 | NULL |
| 6 | NULL |
+------------+------------------+
> SELECT Student_id, lpad(Student_name,0,'new') AS LeftPaddedString FROM
t1;
+------------+------------------+
| Student_id | LeftPaddedString |
+------------+------------------+
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
+------------+------------------+
```

LTRIM()

函数说明

LTRIM() 将输入字符串的前部空格去除，返回处理后的字符。

函数语法

```
> LTRIM(char)
```

参数释义

参数	说明
char	必要参数，CHAR 与 VARCHAR 均可

示例

```
> drop table if exists t1;
> create table t1(a char(8),b varchar(10));
> insert into t1 values(' matrix',' matrixone');
> select ltrim(a),ltrim(b) from t1;

+----------+-----------+
| ltrim(a) | ltrim(b) |
+----------+-----------+
| matrix | matrixone |
+----------+-----------+
```

OCT(N)

函数说明

函数 `OCT(N)` 返回 *N* 的八进制值的字符串，其中 *N* 是一个 longlong(BIGINT) 类型的数字，即将一个数字从十进制数字基数系统转换到八进制数字基数系统。 若 *N* 为 *NULL*，则返回 `NULL`。

函数语法

```
> OCT(N)
```

参数释义

参数	说明
N	必要参数。UINT 类型

示例

```
SELECT OCT(12);
+---------+
| oct(12) |
+---------+
| 14.0000 |
+---------+
1 row in set (0.00 sec)
```

REVERSE()

函数说明

将 str 字符串中的字符顺序翻转输出。

函数语法

```
> REVERSE(str)
```

参数释义

参数	说明
Str	必要参数。需要翻转的字符串。CHAR 与 VARCHAR 类型均可。

示例

```
> drop table if exists t1;
> create table t1(a varchar(12),c char(30));
> insert into t1 values('sdfad ','2022-02-02 22:22:22');
> insert into t1 values(' sdfad ','2022-02-02 22:22:22');
> insert into t1 values('adsf sdfad','2022-02-02 22:22:22');
> insert into t1 values(' sdfad','2022-02-02 22:22:22');
> select reverse(a),reverse(c) from t1;
+-------------+---------------------+
| reverse(a) | reverse(c) |
+-------------+---------------------+
| dafds | 22:22:22 20-20-2202 |
| dafds | 22:22:22 20-20-2202 |
| dafds fsda | 22:22:22 20-20-2202 |
| dafds | 22:22:22 20-20-2202 |
+-------------+---------------------+
> select a from t1 where reverse(a) like 'daf%';
+-------------+
| a |
+-------------+
| adsf sdfad |
| sdfad |
+-------------+
> select reverse(a) reversea,reverse(reverse(a)) normala from t1;
+-------------+-------------+
| reversea | normala |
+-------------+-------------+
| dafds | sdfad |
| dafds | sdfad |
| dafds fsda | adsf sdfad |
| dafds | sdfad |
+-------------+-------------+
```

RPAD()

函数说明

函数 RPAD(str,len,padstr) 在字符串 *str* 右侧使用 *padstr* 进行填充，直至总长度为 *len* 的字符串，最后返回填充后的字符串。如果 *str* 的长度大于 *len*，那么最后的长度将缩减至 *len*。
若 *len* 为负数，则返回 NULL。

函数语法

```
> RPAD(str,len,padstr)
```

参数释义

参数	说明
str	必要参数，被填充的字符串。CHAR 与 VARCHAR 类型均可。
len	必要参数，需要填充到的总长度。
padstr	必要参数，用于填充的字符串。CHAR 与 VARCHAR 类型均可。

示例

```
> drop table if exists t1;
> CREATE TABLE t1(Student_id INT,Student_name VARCHAR(100),Student_Class
CHAR(20));
> INSERT INTO t1
VALUES
(1,'Ananya Majumdar', 'IX'),
(2,'Anushka Samanta', 'X'),
(3,'Aniket Sharma', 'XI'),
(4,'Anik Das', 'X'),
(5,'Riya Jain', 'IX'),
(6,'Tapan Samanta', 'X');
> SELECT Student_id, Student_name,RPAD(Student_Class, 10, ' _') AS
LeftPaddedString FROM t1;
+------------+-----------------+------------------+
| Student_id | Student_name | LeftPaddedString |
+------------+-----------------+------------------+
| 1 | Ananya Majumdar | IX _ _ _ _ |
| 2 | Anushka Samanta | X _ _ _ _ |
| 3 | Aniket Sharma | XI _ _ _ _ |
| 4 | Anik Das | X _ _ _ _ |
| 5 | Riya Jain | IX _ _ _ _ |
| 6 | Tapan Samanta | X _ _ _ _ |
+------------+-----------------+------------------+
> SELECT Student_id, rpad(Student_name,4,'new') AS LeftPaddedString FROM
t1;
+------------+------------------+
| Student_id | LeftPaddedString |
+------------+------------------+
| 1 | Anan |
| 2 | Anus |
| 3 | Anik |
| 4 | Anik |
| 5 | Riya |
| 6 | Tapa |
+------------+------------------+
> SELECT Student_id, rpad(Student_name,-4,'new') AS LeftPaddedString FROM
t1;
+------------+------------------+
| Student_id | LeftPaddedString |
+------------+------------------+
| 1 | NULL |
| 2 | NULL |
```

```
| 3 | NULL |
| 4 | NULL |
| 5 | NULL |
| 6 | NULL |
+------------+------------------+
> SELECT Student_id, rpad(Student_name,0,'new') AS LeftPaddedString FROM
t1;
+------------+------------------+
| Student_id | LeftPaddedString |
+------------+------------------+
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
+------------+------------------+
```

RTRIM()

函数说明

RTRIM() 将输入字符串的后方空格去除，返回处理后的字符。

函数语法

```
> RTRIM(char)
```

参数释义

参数	说明
char	必要参数，CHAR 与 VARCHAR 均可

示例

```
> drop table if exists t1;
> create table t1(a char(8),b varchar(10));
> insert into t1 values('matrix ','matrixone ');
> select rtrim(a),rtrim(b) from t1;

+----------+-----------+
| rtrim(a) | rtrim(b) |
+----------+-----------+
| matrix | matrixone |
+----------+-----------+
```

SPACE()

函数说明

SPACE(N) 返回 N 个空格组成的字符串。

语法

```
> SPACE(N)
```

参数释义

参数	说明
N	必要参数。UINT 类型

示例

```
> drop table if exists t1;
> CREATE TABLE t1
(
Employee_name VARCHAR(100) NOT NULL,
Joining_Date DATE NOT NULL
);
> INSERT INTO t1
(Employee_name, Joining_Date )
VALUES
(' Ananya Majumdar', '2000-01-11'),
(' Anushka Samanta', '2002-11-10' ),
(' Aniket Sharma ', '2005-06-11' ),
(' Anik Das', '2008-01-21' ),
(' Riya Jain', '2008-02-01' ),
(' Tapan Samanta', '2010-01-11' ),
(' Deepak Sharma', '2014-12-01' ),
(' Ankana Jana', '2018-08-17'),
(' Shreya Ghosh', '2020-09-10') ;
> INSERT INTO t1
(Employee_name, Joining_Date ) values(' ','2014-12-01');
> select * from t1 where Employee_name=space(5);
+---------------+--------------+
| Employee_name | Joining_Date |
+---------------+--------------+
| | 2014-12-01 |
+---------------+--------------+
```

STARTSWITH()

函数说明

字符串如果以指定前缀开始返回 1，否则则返回 0。该函数是对大小写敏感的。

函数语法

```
> STARTSWITH(str,prefix)
```

参数释义

参数	说明
str	必要参数。CHAR 和 VARCHAR 类型都支持。
prefix	必要参数。CHAR 和 VARCHAR 类型都支持。

返回值

- 1，如果字符串是以指定前缀开始的。
- 0，如果字符串不以指定前缀开始的。

示例

```
> drop table if exists t1;
> create table t1(a int,b varchar(100),c char(20));
> insert into t1 values
(1,'Ananya Majumdar', 'IX'),
(2,'Anushka Samanta', 'X'),
(3,'Aniket Sharma', 'XI'),
(4,'Anik Das', 'X'),
(5,'Riya Jain', 'IX'),
(6,'Tapan Samanta', 'X');
> select a,startswith(b,'An') from t1;
+------+-------------------+
| a | startswith(b, An) |
+------+-------------------+
| 1 | 1 |
| 2 | 1 |
| 3 | 1 |
| 4 | 1 |
| 5 | 0 |
| 6 | 0 |
+------+-------------------+
> select a,b,c from t1 where startswith(b,'An')=1 and
startswith(c,'I')=1;
+------+-----------------+------+
| a | b | c |
+------+-----------------+------+
| 1 | Ananya Majumdar | IX |
+------+-----------------+------+
```

SUBSTRING()

函数说明

SUBSTR() 是 SUBSTRING() 的同义词。 不带 len 参数的写法会返回一个从 pos 位置开始的子字符串。带 len 参数的写法会返回一个从 pos 位置开始的长度为 len 的子字符串。

语法

```
> SUBSTRING(str,pos)
> SUBSTR(str,pos,len)
```

参数释义

参数	说明
str	必要参数，母字符串。CHAR 与 VARCHAR 类型均可。
pos	必要参数，开始位置
len	可选参数，返回子字符串长度

示例

```
> CREATE TABLE IF NOT EXISTS t1 (
pub_id varchar(8) COLLATE latin1_general_ci NOT NULL DEFAULT '',
pub_name varchar(50) COLLATE latin1_general_ci NOT NULL DEFAULT '',
pub_city varchar(25) COLLATE latin1_general_ci NOT NULL DEFAULT '',
country varchar(25) COLLATE latin1_general_ci NOT NULL DEFAULT '',
country_office varchar(25) COLLATE latin1_general_ci NOT NULL DEFAULT '',
no_of_branch int NOT NULL DEFAULT 0,
estd date NOT NULL DEFAULT '2000-01-01'
);

> INSERT INTO t3 (pub_id, pub_name, pub_city, country, country_office,
no_of_branch, estd) VALUES
('P001', 'Jex Max Publication', 'New York', 'USA', 'New York', 15, '1969-
12-25'),
('P002', 'BPP Publication', 'Mumbai', 'India', 'New Delhi', 10, '1985-10-
01'),
('P003', 'New Harrold Publication', 'Adelaide', 'Australia', 'Sydney', 6,
'1975-09-05'),
('P004', 'Ultra Press Inc.', 'London', 'UK', 'London', 8, '1948-07-10'),
('P005', 'Mountain Publication', 'Houstan', 'USA', 'Sun Diego', 25,
'1975-01-01'),
('P006', 'Summer Night Publication', 'New York', 'USA', 'Atlanta', 10,
'1990-12-10'),
('P007', 'Pieterson Grp. of Publishers', 'Cambridge', 'UK', 'London', 6,
'1950-07-15'),
('P008', 'Novel Publisher Ltd.', 'New Delhi', 'India', 'Bangalore', 10,
'2000-01-01');

> SELECT pub_name, SUBSTR(pub_name,4,5) FROM t1 WHERE country='USA';
+--------------------------+-----------------------+
| pub_name | substr(pub_name, 4, 5) |
+--------------------------+-----------------------+
| Jex Max Publication | Max |
| Mountain Publication | ntain |
| Summer Night Publication | mer N |
+--------------------------+-----------------------+
3 rows in set (0.04 sec)

> SELECT pub_name, SUBSTR(pub_name,5) FROM t1 WHERE country='USA';
+--------------------------+----------------------+
| pub_name | substr(pub_name, 5) |
+--------------------------+----------------------+
```

```
| Jex Max Publication | Max Publication |
| Mountain Publication | tain Publication |
| Summer Night Publication | er Night Publication |
+--------------------------+----------------------+
3 rows in set (0.03 sec)
```

限制

Substring 函数目前不支持在函数中使用 FROM 与 FOR 操作符。

SUBSTRING_INDEX()

函数说明

此函数 `SUBSTRING_INDEX()` 以分隔符为索引，获取不同索引位的字符。

如果 count 为正，则返回最后一个分隔符左侧（从左侧开始计数）的所有内容。

如果 count 为负数，则返回最后一个分隔符右侧（从右侧开始计数）的所有内容。

如果参数为 `NULL`， `SUBSTRING_INDEX()` 将返回 `NULL`。

语法说明

```
> SUBSTRING_INDEX(str,delim,count)
```

即，substring_index（“待截取有用部分的字符串”，“截取数据依据的字符”，截取字符的位置 N）

参数释义

参数	说明
str	字符串
delim	分隔符
count	表示 delim 出现次数的整数。

示例

```
mysql> SELECT SUBSTRING_INDEX('www.mysql.com', '.', 2);
+-------------------------------------+
| substring_index(www.mysql.com, ., 2) |
+-------------------------------------+
| www.mysql |
+-------------------------------------+
1 row in set (0.03 sec)

mysql> select substring_index('xyz', 'abc', 9223372036854775808);
+-----------------------------------------------+
| substring_index(xyz, abc, 9223372036854775808) |
+-----------------------------------------------+
| xyz |
+-----------------------------------------------+
1 row in set (0.02 sec)

mysql> SELECT SUBSTRING_INDEX('www.mysql.com', '.', -2);
+-------------------------------------+
| substring_index(www.mysql.com, ., -2) |
+-------------------------------------+
| mysql.com |
+-------------------------------------+
1 row in set (0.02 sec)

mysql> SELECT SUBSTRING_INDEX(SUBSTRING_INDEX('192,168,8,203', ',', 2),
',',-1);
+-----------------------------------------------------------------+
| substring_index(substring_index(192,168,8,203, ,, 2), ,, -1) |
+-----------------------------------------------------------------+
| 168 |
+-----------------------------------------------------------------+
1 row in set (0.02 sec)

create table test(a varchar(100), b varchar(20), c int);
insert into test values('www.mysql.com', '.', 0);
insert into test values('www.mysql.com', '.', 1);
insert into test values('www.mysql.com', '.', 2);
insert into test values('www.mysql.com', '.', 3);
insert into test values('www.mysql.com', '.', 9223372036854775808);
insert into test values('www.mysql.com', '.', -1);
insert into test values('www.mysql.com', '.', -2);
insert into test values('www.mysql.com', '.', -3);
```

```
mysql> select SUBSTRING_INDEX(a, b, c) from test;
+--------------------------+
| substring_index(a, b, c) |
+--------------------------+
| |
| www |
| www.mysql |
| www.mysql.com |
| com |
| mysql.com |
| www.mysql.com |
+--------------------------+
7 rows in set (0.02 sec)
```

TRIM()

函数说明

`TRIM()` 函数返回一个字符串，删除不需要的字符。

你可以使用 `LEADING`， `TRAILING` 或 `BOTH` 选项明确指示 `TRIM()` 函数从字符串中删除前导，尾随或前导和尾随的不必要的字符。

如果你没有指定任何内容， `TRIM()` 函数默认使用 `BOTH` 选项。

`[remstr]` 是要删除的字符串。默认是一个空格。即如果不指定特定的字符串，则 `TRIM()` 函数仅删除空格。

函数语法

```
> TRIM([{BOTH | LEADING | TRAILING} [remstr] FROM] str), TRIM([remstr
FROM] str)
```

参数释义

参数	说明
str	必要参数。是要删除子字符 `remstr` 的字符串。

示例

```
mysql> select trim(' abc '), trim('abc '), trim(' abc'), trim('abc');
+-------------+------------+------------+-----------+
| trim( abc ) | trim(abc ) | trim( abc) | trim(abc) |
+-------------+------------+------------+-----------+
| abc | abc | abc | abc |
+-------------+------------+------------+-----------+
1 row in set (0.00 sec)

drop table if exists t1;
create table t1(a varchar(100), b varchar(100));
insert into t1 values('abc', 'abc');
insert into t1 values('啊abc哦', '啊abc哦');
insert into t1 values('啊啊o', 'o');
insert into t1 values('啊啊o', '啊');
insert into t1 values('啊啊o', 'o啊');
mysql> select trim(both a from b) from t1;
+---------------------+
| trim(both a from b) |
+---------------------+
| |
| |
| o |
| 啊 |
| o啊 |
+---------------------+
5 rows in set (0.00 sec)

mysql> select trim(leading a from b) from t1;
+------------------------+
| trim(leading a from b) |
+------------------------+
| |
| |
| o |
| 啊 |
| o啊 |
+------------------------+
5 rows in set (0.01 sec)

mysql> select trim(trailing a from b) from t1;
+------------------------+
| trim(trailing a from b) |
```

```
+-------------------------+
| |
| |
| o |
| 啊 |
| o啊 |
+-------------------------+
5 rows in set (0.00 sec)
```

通用参数配置

在 *matrixone/etc/launch-tae-CN-tae-DN/* 目录有三个配置文件 *cn.toml*、*dn.toml* 和 *log.toml*。

各个配置文件中所含参数解释如下：

cn.toml

参数	参数解释
service-type = “CN”	节点类型
data-dir = “./mo-data”	默认数据目录
[log]	
level = “info”	日志级别，可修改为 info/debug/error/faltal
format = “console”	日志格式
max-size = 512	日志默认长度
[hakeeper-client]	HAkeeper 默认地址与端口，不建议更改
service-addresses = [“127.0.0.1:32001”,]	
[[fileservice]]	fileservice 配置，不建议更改
name = “LOCAL”	fileservice 存储类型，本地存储
backend = “DISK”	fileservice 后端介质，磁盘
[[fileservice]]	
name = “SHARED”	fileservice 存储类型，S3
backend = “DISK”	fileservice 后端介质，磁盘
data-dir = “mo-data/s3”	s3 存储数据路径
[fileservice.cache]	
memory-capacity = “512MB”	fileservice 使用的 cache 内存大小
disk-capacity = “8GB”	fileservice 使用的 cache 磁盘大小

参数	参数解释
disk-path = “mo-data/file-service-cache”	fileservice 的磁盘 cache 路径
[[fileservice]]	
name = “ETL”	fileservice 存储类型，ETL
backend = “DISK-ETL”	fileservice 后端介质，DISK-ETL
[observability]	可观测性参数，默认不开启
disableTrace = true	
disableMetric = true	
[cn]	cn 节点的编号，不可修改
uuid = “dd1dccb4-4d3c-41f8-b482-5251dc7a41bf”	
[cn.Engine]	cn 节点的存储引擎，分布式 tae，不可修改
type = “distributed-tae”	

dn.toml

参数	参数解释
service-type = “DN”	节点类型
data-dir = “./mo-data”	默认数据目录
[log]	
level = “info”	日志级别，可修改为info/debug/error/faltal
format = “console”	日志格式
max-size = 512	日志默认长度
[hakeeper-client]	HAkeeper 默认地址与端口，不建议更改
service-addresses = [“127.0.0.1:32001”,]	
[[fileservice]]	fileservice 配置，不建议更改
name = “LOCAL”	fileservice 存储类型，本地存储

参数	参数解释
backend = “DISK”	fileservice 后端介质，磁盘
[[fileservice]]	
name = “SHARED”	fileservice 存储类型，S3
backend = “DISK”	fileservice 后端介质，磁盘
data-dir = “mo-data/s3”	s3 存储数据路径
[fileservice.cache]	
memory-capacity = “512MB”	fileservice 使用的 cache 内存大小
disk-capacity = “8GB”	fileservice 使用的 cache 磁盘大小
disk-path = “mo-data/file-service-cache”	fileservice 的磁盘 cache 路径
[[fileservice]]	
name = “ETL”	fileservice 存储类型，ETL
backend = “DISK-ETL”	fileservice 后端介质，DISK-ETL
[dn]	
uuid = “dd4dccb4-4d3c-41f8-b482-5251dc7a41bf”	dn 的 uuid，不可修改
[dn.Txn.Storage]	dn 事务后端的存储引擎，不可修改
backend = “TAE”	
log-backend = “logservice”	
[dn.Ckp]	dn 的 checkpoint 相关参数，不建议更改
flush-interval = ”60s”	内部刷新间隔
min-count = 100	checkpoint 最小个数
scan-interval = “5s”	内部扫描间隔
incremental-interval = “180s”	checkpoint 自增间隔
global-min-count = 60	全局最小的 dn checkpoint 个数
[dn.LogtailServer]	
listen-address = “0.0.0.0:32003”	logtail 监听端口

参数	参数解释
service-address = “127.0.0.1:32003”	logtail 内部访问地址
rpc-max-message-size = “16KiB”	logtail 使用的最大 rpc 消息大小
rpc-payload-copy-buffer-size = “16KiB”	rpc 复制 buffer 的大小
rpc-enable-checksum = true	是否开启 rpc checksum
logtail-collect-interval = “2ms”	logtail 的统计收集时间间隔
logtail-response-send-timeout = ”10s”	logtail 发送的超时时间
max-logtail-fetch-failure = 5	获取 logtail 允许的最大失败次数

log.toml

参数	参数解释
service-type = “LOG”	节点类型
data-dir = “./mo-data”	默认数据目录
[log]	
level = “info”	日志级别，可修改为info/debug/error/faltal
format = “console”	日志格式
max-size = 512	日志默认长度
[[fileservice]]	fileservice 配置，不建议更改
name = “LOCAL”	fileservice 存储类型，本地存储
backend = “DISK”	fileservice 后端介质，磁盘
[[fileservice]]	
name = “SHARED”	fileservice 存储类型，S3
backend = “DISK”	fileservice 后端介质，磁盘
data-dir = “mo-data/s3”	s3 存储数据路径
[fileservice.cache]	
memory-capacity = “512MB”	fileservice 使用的 cache 内存大小

参数	参数解释
disk-capacity = “8GB”	fileservice 使用的 cache 磁盘大小
disk-path = “mo-data/file-service-cache”	fileservice 的磁盘 cache 路径
[[fileservice]]	
name = “ETL”	fileservice 存储类型，ETL
backend = “DISK-ETL”	fileservice 后端介质，DISK-ETL
[observability]	监控相关参数
statusPort = 7001	预留普罗米修斯的监控端口
enableTraceDebug = false	开启 trace 功能的 dbug 模式
[hakeeper-client]	HAkeeper 默认地址与端口，不建议更改
service-addresses = [“127.0.0.1:32001”,]	
[logservice]	logservice 的相关参数，不可修改
deployment-id = 1	logservice 的部署 id
uuid = “7c4dccb4-4d3c-41f8-b482-5251dc7a41bf”	logservice 的节点 id
raft-address = “127.0.0.1:32000”	raft 协议使用的地址
logservice-address = “127.0.0.1:32001”	logservice 服务地址
gossip-address = “127.0.0.1:32002”	gossip 协议的地址
gossip-seed-addresses = [“127.0.0.1:32002”,]	gossip 协议的种子节点地址
gossip-allow-self-as-seed = true	是否允许 gossip 协议用本节点做种子节点
[logservice.BootstrapConfig]	bootstrap 相关参数，不可修改
bootstrap-cluster = true	bootstrap 是否集群启动
num-of-log-shards = 1	logservice 的分片数
num-of-dn-shards = 1	dn 的分片数
num-of-log-shard-replicas = 1	logservice 分片的副本数

参数	参数解释
init-hakeeper-members = [“131072:7c4dccb4-4d3c-41f8-b482-5251dc7a41bf”,]	初始化 hakeeper 的成员

时区支持

MatrixOne 使用的时区取决于三个系统变量：`global.time_zone`，`session.time_zone` 和 `global.system_time_zone`。

- `global.system_time_zone` 表示服务器系统时区。当服务器启动时，它会尝试确定主机的时区，并使用它来设置系统时区（`system_time_zone`）。
- `global.time_zone` 表示服务器当前时区。初始 `time_zone` 值为 `SYSTEM`，表示服务器时区与系统时区相同。

你可以使用以下语句在运行时设置全局服务器时区，设置完成后无法在当前会话中生效，你需要先退出当前会话，再次重新连接 MatrixOne 时才会生效。

```
> SET GLOBAL time_zone = timezone;
```

- 每个会话都有自己的会话时区，由当时的会话 `time_zone` 变量决定。最初，会话时区的变量值从全局 `time_zone` 变量中获得，但会话客户端可以更改自己的时区。但是这个这个设置只会在当前会话期间有效。

```
SET time_zone = timezone;
```

使用以下 SQL 语句查看当前全局时区、客户端时区和系统时区的值：

```
> SELECT @@global.time_zone, @@session.time_zone,
@@global.system_time_zone;
+-------------+-------------+--------------------+
| @@time_zone | @@time_zone | @@system_time_zone |
+-------------+-------------+--------------------+
| timezone | +08:00 | |
+-------------+-------------+--------------------+
1 row in set (0.00 sec)
```

设置 time_zone 的值的格式：

- 值 `SYSTEM` 表示时区应与服务器系统时区相同。

- 值 `UTC` 表示时区设置为 UTC（Coordinated Universal Time，协调世界时）。仅支持 “UTC” 缩写作为时区使用。
- 该值可以作为字符串给出，表示 UTC 时间的偏移，格式为 “HH

  ”，带有 + 或 -，例如 `+10:00` 或者 `-6:00`。允许的范围是 “-13:59” 到 “+14:00”。

当前会话时区设置会影响时区敏感时间值的显示和存储。即会影响执行 `NOW()` 等函数查询到的值以及存储在 `TIMESTAMP` 列中和从 `TIMESTAMP` 列中查询到的值。

会话时区设置不影响 `UTC_TIMESTAMP()` 等函数显示的值或 `DATE`、`TIME` 或 `DATETIME` 列中的值。

> **注意**
>
> 只有 Timestamp 数据类型的值是受时区影响的。可以理解为，Timestamp 数据类型的实际表示使用的是（字面值 + 时区信息）。其它时间和日期类型，比如 Datetime/Date/Time 是不包含时区信息的，所以也不受到时区变化的影响。

```
> SELECT @@global.time_zone, @@session.time_zone,
@@global.system_time_zone;
+-------------+-------------+--------------------+
| @@time_zone | @@time_zone | @@system_time_zone |
+-------------+-------------+--------------------+
| SYSTEM | SYSTEM | |
+-------------+-------------+--------------------+
1 row in set (0.01 sec)

> create table t (ts timestamp, dt datetime);
Query OK, 0 rows affected (0.02 sec)

mysql> set @@time_zone = 'UTC';
Query OK, 0 rows affected (0.00 sec)

mysql> insert into t values ('2017-09-30 11:11:11', '2017-09-30
11:11:11');
Query OK, 1 row affected (0.02 sec)

mysql> set @@time_zone = '+08:00';
Query OK, 0 rows affected (0.00 sec)

mysql> select * from t;
+---------------------+---------------------+
| ts | dt |
+---------------------+---------------------+
| 2017-09-30 19:11:11 | 2017-09-30 11:11:11 |
+---------------------+---------------------+
1 row in set (0.00 sec)
```

上面的例子中，无论怎么调整时区的值，Datetime 类型字段的值是不受影响的，而 Timestamp 则随着时区改变，显示的值会发生变化。其实 Timestamp 持久化到存储的值始终没有变化过，只是根据时区的不同显示值不同。

> **注意**
>
> Timestamp 类型和 Datetime 等类型的值，两者相互转换的过程中，会涉及到时区。这种情况一律基于当前 time_zone 时区处理。

修改 MatrixOne 时区

1. 查看当前时间或时区：

```
> select now();
+----------------------------+
| now() |
+----------------------------+
| 2022-10-14 18:38:27.876181 |
+----------------------------+
1 row in set (0.00 sec)


> show variables like "%time_zone%";
+------------------+--------+
| Variable_name | Value |
+------------------+--------+
| system_time_zone | |
| time_zone | SYSTEM |
+------------------+--------+
2 rows in set (0.00 sec)
```

- `time_zone`：使用 system 的时区。
- `system_time_zone` 说明 system 使用服务器系统时区。

2. 修改当前时区：

```
set global time_zone = '+08:00';
set time_zone = '+08:00';
```

- `set global time_zone = '+08:00';`：修改 mysql 全局时区为北京时间，即我们所在的东 8 区。
- `set time_zone = '+08:00';`：修改当前会话时区。

限制

1. `system_time_zone` 值暂不存在。
2. 仅支持 `(+/-)HH:MM` 格式和 `UTC` 来设置 `time_zone` 的值。

MatrixOne 系统数据库和表

MatrixOne 系统数据库和表是 MatrixOne 存储系统信息的地方。你可以通过它们访问系统信息。MatrixOne 在初始化时创建了 6 个系统数据库：`mo_catalog`、`information_schema`、`system_metrcis`、`system`、`mysql` 和 `mo_task`。`mo_task` 当前正在开发中，暂时对你所进行的操作不会产生直接影响。本文档中描述了其他系统数据库和表函数。

系统只能修改系统数据库和表，你仅能从中进行读取操作。

`mo_catalog` 数据库

`mo_catalog` 用于存储 MatrixOne 对象的元数据，如：数据库、表、列、系统变量、租户、用户和角色。

从 MatrixOne 0.6 版本即引入了多租户的概念，默认的 `sys` 租户和其他租户的行为略有不同。服务于多租户管理的系统表 `mo_account` 仅对 `sys` 租户可见；其他租户看不到此表。

mo_database table

列属性	类型	描述
dat_id	bigint unsigned	主键
datname	varchar(100)	数据库名称
dat_catalog_name	varchar(100)	数据库 catalog 名称，默认 `def`
dat_createsql	varchar(100)	创建数据库 SQL 语句
owner	int unsigned	角色 ID ID
creator	int unsigned	用户 ID ID
created_time	timestamp	创建时间
account_id	int unsigned	租户 ID ID

mo_tables table

列属性	类型	描述
rel_id	bigint unsigned	主键

列属性	类型	描述
relname	varchar(100)	表、索引、视图等的名称
reldatabase	varchar(100)	包含此关系的数据库，参考 mo_database.datname
reldatabase_id	bigint unsigned	包含此关系的数据库 ID，参考 mo_database.datid
relpersistence	varchar(100)	p = 永久表 t = 临时表
relkind	varchar(100)	r = 普通表 e = 外部表 i = 索引 S = 序列 v = 视图 m = 物化视图
rel_comment	varchar(100)	
rel_createsql	varchar(100)	创建表 SQL 语句
created_time	timestamp	创建时间
creator	int unsigned	创建者 ID
owner	int unsigned	创建者的默认角色 ID
account_id	int unsigned	租户 id
partitioned	blob	按语句分区
viewdef	blob	视图定义语句

mo_columns table

列属性	类型	描述
att_uniq_name	varchar(256)	主键。隐藏的复合主键，格式类似于 " ${att_relname_id}-${attname}"
account_id	int unsigned	租户 ID
att_database_id	bigint unsigned	数据库 ID
att_database	varchar(256)	数据 Name

列属性	类型	描述
att_relname_id	bigint unsigned	表 ID
att_relname	varchar(256)	此列所属的表。（参考 mo_tables.relname）
attname	varchar(256)	列名
atttyp	varchar(256)	此列的数据类型 (删除的列为 0)。
attnum	int	列数。普通列从 1 开始编号。
att_length	int	类型的字节数
attnotnull	tinyint(1)	表示一个非空约束。
atthasdef	tinyint(1)	此列有默认表达式或生成表达式。
att_default	varchar(1024)	默认表达式
attisdropped	tinyint(1)	此列已删除，不再有效。删除的列仍然物理上存在于表中，但解析器会忽略它，因此不能通过 SQL 访问它。
att_constraint_type	char(1)	p = 主键约束 n=无约束
att_is_unsigned	tinyint(1)	是否未署名
att_is_auto_increment	tinyint(1)	是否自增
att_comment	varchar(1024)	注释
att_is_hidden	tinyint(1)	是否隐藏
attr_has_update	tinyint(1)	此列含有更新表达式
attr_update	varchar(1024)	更新表达式

mo_account table (仅 `sys` 租户可见)

列属性	类型	描述
account_id	int unsigned	租户 ID
account_name	varchar(100)	租户名
status	varchar(100)	开启/暂停
created_time	timestamp	创建时间

列属性	类型	描述
comment	varchar(256)	注释
suspended_time	TIMESTAMP	修改租户状态的时间

mo_role table

列属性	类型	描述
role_id	int unsigned	角色 ID
role_name	varchar(100)	角色名称
creator	int unsigned	用户 ID
owner	int unsigned	MatrixOne 管理员/租户管理员拥有者 ID
created_time	timestamp	创建时间
comment	text	注释

mo_user table

列属性	类型	描述
user_id	int	用户 ID
user_host	varchar(100)	用户主机地址
user_name	varchar(100)	用户名
authentication_string	varchar(100)	密码加密的认证字符串
status	varchar(8)	开启、锁定、失效
created_time	timestamp	用户创建时间
expired_time	timestamp	用户过期时间
login_type	varchar(16)	ssl/密码/其他
creator	int	创建此用户的创建者 ID
owner	int	此用户的管理员 ID
default_role	int	此用户的默认角色 ID

mo_user_grant table

列属性	类型	描述
role_id	int unsigned	角色 ID
user_id	int unsigned	用户 ID
granted_time	timestamp	授权时间
with_grant_option	bool	是否允许授权

mo_role_grant table

列属性	类型	描述
granted_id	int	被授予的角色 ID
grantee_id	int	要授予其他角色的角色 ID
operation_role_id	int	操作角色 ID
operation_user_id	int	操作用户 ID
granted_time	timestamp	授权时间
with_grant_option	bool	是否允许授权

mo_role_privs table

列属性	类型	描述
role_id	int unsigned	角色 ID
role_name	varchar(100)	角色名：accountadmin/public
obj_type	varchar(16)	对象类型：account/database/table
obj_id	bigint unsigned	对象 ID
privilege_id	int	权限 ID
privilege_name	varchar(100)	权限名：权限列表
privilege_level	varchar(100)	权限级别
operation_user_id	int unsigned	操作用户 ID
granted_time	timestamp	授权时间

列属性	类型	描述
with_grant_option	bool	是否允许授权

`system_metrics` 数据库

`system_metrics` 收集 SQL 语句、CPU 和内存资源使用的状态和统计信息。

`system_metrics` 表一些相同的列类型，这些表中的字段描述如下：

- collecttime：收集时间。
- value：采集 `metrics` 的值。
- node：表示 MatrixOne 节点的 uuid。
- role：MatrixOne 节点角色，包括 CN、DN 和 Log。
- account：默认为 “sys” 租户，即触发 SQL 请求的账户。
- type：SQL 类型，可以是 `select`，`insert`，`update`，`delete`，`other` 类型。

`metric` 表

列属性	类型	描述
metric_name	VARCHAR(128)	指标名称，例如：sql_statement_total，server_connections，process_cpu_percent，sys_memory_used 等
collecttime	DATETIME	指标数据收集时间
value	DOUBLE	指标值
node	VARCHAR(36)	MatrixOne 节点 uuid
role	VARCHAR(32)	MatrixOne 节点角色
account	VARCHAR(128)	租户名称，默认 `sys`
类型	VARCHAR(32)	SQL 类型，例如：INSERT，SELECT，UPDATE

以下表为 `metric` 表的视图：

- `process_cpu_percent` 表：CPU 进程繁忙百分比。
- `process_open_fs` 表：打开的文件描述符的数量。
- `process_resident_memory_bytes` 表：驻留内存量，单位为字节。
- `server_connection` 表：服务器连接数。
- `sql_statement_errors` 表：执行错误的 SQL 语句的计数器。
- `sql_statement_total` 表：执行 SQL 语句的计数器。
- `sql_transaction_errors` 表：错误执行的事务性语句的计数器。
- `sql_transaction_total` 表：事务性 SQL 语句的计数器。
- `sys_cpu_combined_percent` 表：系统 CPU 繁忙百分比，所有逻辑核的平均值。
- `sys_cpu_seconds_total` 表：系统 CPU 时间，以秒为单位，由核数标准化
- `sys_disk_read_bytes` 表：以字节为单位读取系统盘。
- `sys_disk_write_bytes` 表：以字节为单位写入系统盘。
- `sys_memory_available` 表：以字节为单位的可用系统内存。
- `sys_memory_used` 表：以字节为单位已使用的系统内存。
- `sys_net_recv_bytes` 表：以字节为单位接收的系统网络。

- `sys_net_sent_bytes` 表：以字节为单位发送的系统网络。

`system` 数据库

`System` 数据库存储 MatrixOne 历史 SQL 语句、系统日志、错误信息。

`statement_info` 表

`statement_info` 表记录用户和系统的 SQL 语句和详细信息。

列属性	类型	描述
statement_id	VARCHAR(36)	声明语句唯一 ID
transaction_id	VARCHAR(36)	事务唯一 ID
session_id	VARCHAR(36)	账户唯一 ID
account	VARCHAR(1024)	租户名称
user	VARCHAR(1024)	用户名称
host	VARCHAR(1024)	用户客户端 IP
database	VARCHAR(1024)	数据库当前会话停留处
statement	TEXT	SQL 语句
statement_tag	TEXT	语句中的注释标签 (保留)
statement_fingerprint	TEXT	语句中的注释标签 (保留)
node_uuid	VARCHAR(36)	节点 uuid，即生成数据的某个节点
node_type	VARCHAR(64)	在 MatrixOne 内，var 所属的 DN/CN/Log 的节点类型
request_at	DATETIME	请求接受的 datetime
response_at	DATETIME	响应发送的 datetime
duration	BIGINT	执行时间，单位：ns
status	VARCHAR(32)	SQL 语句执行状态：Running, Success, Failed
err_code	VARCHAR(1024)	错误码

列属性	类型	描述
error	TEXT	错误信息
exec_plan	JSON	语句执行计划
rows_read	BIGINT	读取总行数
bytes_scan	BIGINT	扫描总字节数
stats	JSON	global stats info in exec_plan
statement_type	VARCHAR(1024)	statement type, val in [Insert, Delete, Update, Drop Table, Drop User, ...]
query_type	VARCHAR(1024)	query type, val in [DQL, DDL, DML, DCL, TCL]
role_id	BIGINT	role id
sql_source_type	TEXT	sql statement source type

`rawlog` 表

`rawlog` 表记录了非常详细的系统日志。

列属性	类型	描述
raw_item	VARCHAR(1024)	原日志项
node_uuid	VARCHAR(36)	节点 uuid，即生成数据的某个节点
node_type	VARCHAR(64)	在 MatrixOne 内，var 所属的 DN/CN/Log 的节点类型
span_id	VARCHAR(16)	span 的唯一 ID
statement_id	VARCHAR(36)	声明语句唯一 ID
logger_name	VARCHAR(1024)	日志记录器的名称
timestamp	DATETIME	时间戳的动作
level	VARCHAR(1024)	日志级别，例如：debug, info, warn, error, panic, fatal
caller	VARCHAR(1024)	产生 Log 的地方：package/file.go:123
message	TEXT	日志消息
extra	JSON	日志动态字段
err_code	VARCHAR(1024)	错误日志

列属性	类型	描述
error	TEXT	错误信息
stack	VARCHAR(4096)	
span_name	VARCHAR(1024)	span 名称，例如：step name of execution plan, function name in code, ...
parent_span_id	VARCHAR(16)	父级 span 唯一的 ID
start_time	DATETIME	
end_time	DATETIME	
duration	BIGINT	执行时间，单位：ns
resource	JSON	静态资源信息

其他 3 个表（`log_info`、`span_info` 和 `error_info`）是 `statement_info` 和 `rawlog` 表的视图。

`information_schema` 数据库

Information Schema 提供了一种 ANSI 标准方式，用于查看系统的元数据。MatrixOne 除了为 MySQL 兼容性而包含的表之外，还提供了许多自定义的 `information_schema` 表。

许多 `INFORMATION_SCHEMA` 表都有相应的 `SHOW` 命令。查询 `INFORMATION_SCHEMA` 可以在表之间进行连接。

MySQL 兼容性表

表名称	描述
CHARACTER_SETS	提供了服务器支持的字符集列表。
COLUMNS	提供了所有表的列列表。
ENGINES	提供了支持的存储引擎列表。
KEY_COLUMN_USAGE	描述了列的键约束，例如主键约束。
PROCESSLIST	提供了与执行命令 `SHOW PROCESSLIST` 类似的信息。
SCHEMATA	提供了与执行 `SHOW DATABASES` 类似的信息。

表名称	描述
TABLES	提供了当前用户可以查看的表列表。类似于执行 `SHOW TABLES` 。
TRIGGERS	提供了与执行 `SHOW TRIGGERS` 类似的信息。Provides similar information to `SHOW TRIGGERS`.
USER_PRIVILEGES	列举了与当前用户关联的权限。

`CHARACTER_SETS` 表

`CHARACTER_SETS` 表中的列描述如下：

- `CHARACTER_SET_NAME`： 字符集的名称。
- `DEFAULT_COLLATE_NAME`： 字符集的默认排序规则名称。
- `DESCRIPTION`： 字符集的描述。
- `MAXLEN`： 在此字符集中存储字符所需的最大长度。

`COLUMNS` 表

`COLUMNS` 表中的列描述如下：

- `TABLE_CATALOG`： 含有该列的表所属的目录的名称。该值始终为 `def` 。
- `TABLE_SCHEMA`： 含有列的表所在的模式的名称。
- `TABLE_NAME`： 包含列的表的名称。
- `COLUMN_NAME`： 列的名称。
- `ORDINAL_POSITION`： 表中列的位置。

- `COLUMN_DEFAULT`：列的默认值。如果显式默认值为 `NULL`，或者如果列定义不包含 `default` 子句，则此值为 `NULL`。
- `IS_NULLABLE`：列是否可以为空。如果该列可以存储空值，则该值为 `YES`；否则为 `NO`

。
- `DATA_TYPE`：列中的数据类型。
- `CHARACTER_MAXIMUM_LENGTH`：对于字符串列，字符的最大长度。
- `CHARACTER_OCTET_LENGTH`：对于字符串列，最大长度（以字节为单位）。
- `NUMERIC_PRECISION`：数字类型列的数字精度。
- `NUMERIC_SCALE`：数字类型列的数字比例。
- `DATETIME_PRECISION`：对于时间类型列，小数秒精度。
- `CHARACTER_SET_NAME`：字符串列的字符集名称。
- `COLLATION_NAME`：字符串列的排序规则的名称。
- `COLUMN_TYPE`：列类型。
- `COLUMN_KEY`：该列是否被索引。该字段可能具有以下值：
  - `Empty`：此列未编入索引，或者此列已编入索引并且是多列非唯一索引中的第二列。
  - `PRI`：此列是主键或多个主键之一。
  - `UNI`：此列是唯一索引的第一列。
  - `MUL`：该列是非唯一索引的第一列，其中允许给定值多次出现。

- `EXTRA`：给定列的任何附加信息。
- `PRIVILEGES`：当前用户所拥有的对该列的权限。
- `COLUMN_COMMENT`：列定义中包含的描述。
- `GENERATION_EXPRESSION`：对于生成的列，此值显示用于计算列值的表达式。对于非生成列，该值为空。
- `SRS_ID`：此值适用于空间列。它包含列 `SRID` 值，该值表示为存储在该列中的值提供一个空间参考系统。

`ENGINES` 表

`ENGINES` 表中的列描述如下：

- `ENGINES`：存储引擎的名称。
- `SUPPORT`：服务器对存储引擎的支持级别。
- `COMMENT`：对存储引擎的简短评论。
- `TRANSACTIONS`：存储引擎是否支持事务。
- `XA`：存储引擎是否支持 XA 事务。
- `SAVEPOINTS`：存储引擎是否支持 `savepoints`。

`PROCESSLIST` 表

`PROCESSLIST` 表中的字段描述如下：

- `ID`：用户连接的 ID。
- `USER`：正在执行 `PROCESS` 的用户名。
- `HOST`：用户连接的地址。
- `DB`：当前连接的默认数据库的名称。
- `COMMAND`：`PROCESS` 正在执行的命令类型。
- `TIME`：`PROCESS` 的当前执行时长，以秒为单位。
- `STATE`：当前连接状态。
- `INFO`：正在处理的请求语句。

`SCHEMATA` 表

`SCHEMATA` 表提供有关数据库的信息。表数据等同于 `SHOW DATABASES` 语句的结果。`SCHEMATA` 表中的字段描述如下：

- `CATALOG_NAME`：数据库所属的目录。
- `SCHEMA_NAME`：数据库名称。
- `DEFAULT_CHARACTER_SET_NAME`：数据库的默认字符集。
- `DEFAULT_COLLATION_NAME`：数据库的默认排序规则。
- `SQL_PATH`：此项的值始终为 `NULL`。

- `DEFAULT_TABLE_ENCRYPTION`：定义数据库和通用表空间的 *default encryption* 设置。

`TABLES` 表

`TABLES` 表中列的描述如下：

- `TABLE_CATALOG`：表所属目录的名称。该值始终为 `def`。
- `TABLE_SCHEMA`：表所属的模式的名称。
- `TABLE_NAME`：表的名称。
- `TABLE_TYPE`：表的类型。
- `ENGINE`：存储引擎的类型。
- `VERSION`：版本。默认值为 `10`。
- `ROW_FORMAT`：行格式。该值当前为 `Compact`。
- `TABLE_ROWS`：统计表中的行数。
- `AVG_ROW_LENGTH`：表的平均行长。`AVG_ROW_LENGTH` = `DATA_LENGTH` / `TABLE_ROWS`。
- `DATA_LENGTH`：数据长度。`DATA_LENGTH` = `TABLE_ROWS` * 元组中列的存储长度之和。不考虑 TiKV 的副本。
- `MAX_DATA_LENGTH`：最大数据长度。该值当前为 `0`，表示数据长度没有上限。
- `INDEX_LENGTH`：索引长度。`INDEX_LENGTH` = `TABLE_ROWS` * 索引元组中列的长度总和。

- `DATA_FREE`：数据片段。该值当前为 `0`。
- `AUTO_INCREMENT`：自增主键的当前步长。
- `CREATE_TIME`：创建表的时间。
- `UPDATE_TIME`：表更新的时间。
- `CHECK_TIME`：检查表的时间。
- `TABLE_COLLATION`：表中字符串的排序规则。
- `CHECKSUM`：校验和。
- `CREATE_OPTIONS`：创建选项。
- `TABLE_COMMENT`：表格的注释和注释。

`USER_PRIVILEGES` 表

`USER_PRIVILEGES` 表提供了关于全局权限的信息。

`USER_PRIVILEGES` 表中的字段描述如下：

- `GRANTEE`：授权用户名，格式为 `'user_name'@'host_name'`。
- `TABLE_CATALOG`：表所属的目录的名称。值为 `def`。
- `PRIVILEGE_TYPE`：要授予的权限类型。每行只显示一种权限类型。

- `IS_GRANTABLE`：如果你有 `GRANT OPTION` 权限，该值为 `YES`，没有 `GRANT OPTION` 权限，该值为 `NO`。

`mysql` 数据库

授权系统表

授权系统表包含了关于用户帐户及其权限信息：

- `user` 用户帐户、全局权限和其他非权限列。
- `db`：数据库级权限。
- `tables_priv`：表级权限。
- `columns_priv`：列级权限。
- `procs_priv`：存储过程和存储函数的权限。

MatrixOne 权限分类

本篇文章主要介绍 MatrixOne 中的权限分类。

MatrixOne 的访问控制权限分为**系统权限**和**对象权限**，在授予角色权限时可做参考。

系统权限

系统权限为集群管理员（默认用户名为 root）的权限，它可以初始化所拥有的权限。集群管理员可以创建、删除其他租户，并管理租户；集群管理员不能管理其他租户下的其他资源。

权限	含义
CREATE ACCOUNT	创建租户，仅集群管理员拥有
DROP ACCOUNT	删除租户，仅集群管理员拥有
ALTER ACCOUNT	管理租户资源，仅集群管理员拥有

对象权限

对象权限可以按照赋权的对象细分为**租户权限**、**用户权限**、**角色权限**、**数据库权限**、**表权限**。

租户权限

拥有*租户权限*的对象可以拥有以下权限：

权限	含义
CREATE USER	创建用户
DROP USER	删除用户
ALTER USER	修改用户
CREATE ROLE	创建角色
DROP ROLE	删除角色
CREATE DATABASE	创建数据库

权限	含义
DROP DATABASE	删除数据库
SHOW DATABASES	查看当前租户下所有数据库
CONNECT	允许使用 `use [database
MANAGE GRANTS	权限管理。包括角色授权、角色继承的权限
ALL [PRIVILEGES]	Account 的所有权限
OWNERSHIP	Account 的所有权限，可以通过 `WITH GRANT OPTION` 设置权限

用户权限

拥有*用户权限*的对象可以拥有以下权限：

权限	含义
Ownership	管理用户所有的权限，包括修改用户信息、密码、删除用户，且可以将这些权限传递给其他角色。

角色权限

拥有*角色权限*的对象可以拥有以下权限：

权限	含义
Ownership	管理角色的所有权限，包括修改角色名称、描述、删除角色，且可以将这些权限传递给其他角色。

数据库权限

拥有*数据库权限*的对象可以拥有以下权限：

权限	含义
SHOW TABLES	查看当前数据库下所有表
CREATE TABLE	建表权限
DROP TABLE	删表权限
CREATE VIEW	创建视图权限，无对应表权限时创建视图无法查询

权限	含义
DROP VIEW	删除视图
ALTER TABLE	修改表权限
ALTER VIEW	修改视图权限，无对应表权限时创建视图无法查询
ALL [PRIVILEGES]	数据库的所有权限
OWNERSHIP	数据库的所有权限，附加 `WITH GRANT OPTION`

表权限

拥有*表权限*的对象可以拥有以下权限：

权限	含义
SELECT	对表执行 `SELECT` 命令
INSERT	对表执行 `INSERT` 命令
UPDATE	对表执行 `UPDATE` 命令
TRUNCATE	对表执行 `TRUNCATE TABLE` 命令
DELETE	对表执行 `DELETE` 命令
REFERENCE	允许将表引用为外间约束的唯一/主键表。通过 `DESCRIBE` 或 `SHOW` 命令查看表的结构
INDEX	创建删除 INDEX
ALL	指定表的所有权限
OWNERSHIP	指定表的所有权限，附加 `WITH GRANT OPTION`

表执行权限

拥有*表执行权限*的对象可以拥有以下权限：

权限	含义
EXECUTE	允许执行函数或存储过程的权限

MatrixOne 的 JDBC 功能支持列表

使用 JDBC 开发应用，MatrixOne 支持以下类和对象：

1. Connection（类）：获取数据库连接对象

类中的方法

1. 获取执行 Sql 对象 Statement
    - Statement createStatement();
    - Statement prepareStatement(String sql);
2. 管理事务
    - 开启事务：setAutoCommit(boolean autoCommit)：调用该方法设置参数为 false，即开启事务
    - 提交事务：void commit()
    - 回滚事务：void rollback()

connection 接口中的方法	支持（Y）/不支持（N）
createStatement()	Y
prepareStatement(String sql)	Y
prepareCall(String sql)	Y
nativeSQL(String sql)	Y
setAutoCommit(boolean autoCommit)	Y
getAutoCommit()	Y
commit()	Y
rollback()	Y
close()	Y
isClosed()	Y

connection 接口中的方法	支持（Y）/不支持（N）
getMetaData()	Y
setReadOnly(boolean readOnly)	Y
isReadOnly()	Y
setCatalog()	Y
getCatalog()	Y
setTransactionIsolation(int level)	N
getTransactionIsolation()	N
getWarnings()	N
clearWarnings()	N
createStatement(int resultSetType, int resultSetConcurrency)	Y
prepareStatement(String sql, int resultSetType, int resultSetConcurrency)	Y
prepareCall(String sql, int resultSetType, int resultSetConcurrency)	Y
getTypeMap()	N
setTypeMap(java.util.Map<String,Class<?>> map)	N
setHoldability(int holdability)	N
getHoldability()	N
setSavepoint()	N
setSavepoint(String name)	N
rollback(Savepoint savepoint)	N
releaseSavepoint(Savepoint savepoint)	N
createStatement(int resultSetType, int resultSetConcurrency, int resultSetHoldability)	Y
prepareStatement(String sql, int resultSetType, int resultSetConcurrency, int resultSetHoldability)	Y
prepareCall(String sql, int resultSetType, int resultSetConcurrency, int resultSetHoldability)	Y

connection 接口中的方法	支持（Y）/不支持（N）
prepareStatement(String sql, int autoGeneratedKeys)	Y
prepareStatement(String sql, int columnIndexes[])	Y
prepareStatement(String sql, String columnNames[])	Y
createClob()	N
createBlob()	N
createNClob()	N
createSQLXML()	N
isValid()	Y
setClientInfo(String name, String value)	N
setClientInfo(Properties properties)	N
getClientInfo(String name)	N
getClientInfo()	N
createArrayOf(String typeName, Object[] elements)	N
createStruct(String typeName, Object[] attributes)	N
setSchema(String schema)	N
getSchema()	N
abort(Executor executor)	N
setNetworkTimeout(Executor executor, int milliseconds)	Y
getNetworkTimeout()	Y

2. Statement 类中的方法

Statement 类中的方法	支持（Y）/不支持（N）
executeQuery(String sql)	Y
executeUpdate(String sql)	Y

Statement 类中的方法	支持（Y）/不支持（N）
close()	Y
getMaxFieldSize()	Y
setMaxFieldSize()	Y
getMaxRows()	Y
setMaxRows()	Y
setEscapeProcessing()	N
getQueryTimeout()	Y
setQueryTimeout(int seconds)	Y
cancel()	Y
getWarnings()	N
clearWarnings()	N
setCursorName(String name)	N
execute(String sql)	Y
getResultSet()	Y
getUpdateCount()	Y
getMoreResults()	Y
setFetchDirection(int direction)	Y
getFetchDirection()	N
setFetchSize(int rows)	Y
getFetchSize()	Y
getResultSetConcurrency()	Y
getResultSetType()	Y
addBatch(String sql)	Y
clearBatch()	Y
executeBatch()	Y

Statement 类中的方法	支持（Y）/不支持（N）
getConnection()	Y
getMoreResults(int current)	Y
getGeneratedKeys()	Y
executeUpdate(String sql, int autoGeneratedKeys)	Y
executeUpdate(String sql, int columnIndexes[])	Y
executeUpdate(String sql, String columnNames[])	Y
execute(String sql, int autoGeneratedKeys)	Y
execute(String sql, int columnIndexes[])	Y
execute(String sql, String columnNames[])	Y
getResultSetHoldability()	Y
isClosed()	Y
setPoolable(boolean poolable)	N
isPoolable()	N
closeOnCompletion()	Y
isCloseOnCompletion()	Y

3. ResultSet interface 中的方法

ResultSet 类中的方法	支持（Y）/不支持（N）
next()	Y
close()	Y
wasNull()	Y
getString(int columnIndex)	Y
getBoolean(int columnIndex)	Y
getByte(int columnIndex)	Y
getShort(int columnIndex)	Y

ResultSet 类中的方法	支持（Y）/不支持（N）
getInt(int columnIndex)	Y
getLong(int columnIndex)	Y
getFloat(int columnIndex)	Y
getDouble(int columnIndex)	Y
getBigDecimal(int columnIndex, int scale)	Y
getBytes(int columnIndex)	Y
getDate(int columnIndex)	Y
getTime(int columnIndex)	Y
getTimestamp(int columnIndex)	Y
getAsciiStream(int columnIndex)	Y
getUnicodeStream(int columnIndex)	Y
getBinaryStream(int columnIndex)	Y
getWarnings()	N
clearWarnings()	N
getCursorName()	N
getMetaData()	Y
getObject()	N
findColumn()	Y
getCharacterStream()	Y
isBeforeFirst()	Y
isAfterLast()	Y
isFirst()	Y
isLast()	Y
beforeFirst()	Y
afterLast()	Y

ResultSet 类中的方法	支持（Y）/不支持（N）
first()	Y
last()	Y
getRow()	Y
absolute()	Y
relative()	Y
previous()	Y
setFetchDirection()	Y
getFetchDirection()	Y
setFetchSize()	Y
getFetchSize()	Y
getType()	Y
getConcurrency()	Y
rowUpdated()	Y
rowInserted()	Y
rowDeleted()	Y
update()(一连串数据类型)	Y
updateNull()	Y

4. ResultSetMetaData 中的方法

ResultSetMetaData 类中的方法	支持（Y）/不支持（N）
getColumnCount()	Y
isAutoIncrement()	Y
isCaseSensitive()	Y
isSearchable()	Y
isCurrency()	Y

ResultSetMetaData 类中的方法	支持（Y）/不支持（N）
isNullable()	Y
isSigned()	Y
getColumnDisplaySize()	Y
getColumnLabel()	Y
getColumnName()	Y
getSchemaName()	N
getPrecision()	Y
getScale()	Y
getTableName()	Y
getCatalogName()	Y
getColumnType()	Y
getColumnTypeName()	Y
isReadOnly()	N
isWritable()	N
isDefinitelyWritable()	N
getColumnClassName()	Y

Mysql 各数据类型 DisplaySize、Prec、Scale 统计

数据类型	DisplaySize	Prec	Scale
TINYINT	4	4	0
SMALLINT	6	6	0
INT	11	11	0
BIGINT	20	20	0
TINYINT UNSIGNED	3	3	0
SMALLINT UNSIGNED	5	5	0
INT UNSIGNED	10	10	0

数据类型	DisplaySize	Prec	Scale
BIGINT UNSIGNED	20	20	0
DECIMAL64(根据实际情况)	17	15	2
DECIMAL128(根据实际情况)	23	21	3
FLOAT	12	12	31
DOUBLE	22	22	31
VARCHAR(根据实际情况)	100	100	0
CHAR(根据实际情况)	100	100	0
DATE	10	10	0
DATETIME	19	19	0
TIMESTAMP	19	19	0
JSON	2147483647	2147483647	0

MatrixOne DDL 语句分区支持的说明

1. MatrixOne 支持的分区类型

目前 MatrixOne 的 DDL 语句支持的 6 种分区类型，与 MySQL 官网基本一致，具体如下：

- KEY Partitioning
- HASH Partitioning
- RANGE Partitioning
- RANGE COLUMNS partitioning
- LIST Partitioning
- LIST COLUMNS partitioning

目前支持子分区（Subpartitioning）语法，但是不支持计划构建。

2. 关于分区键的说明

Partition Keys, Primary Keys 和 Unique Keys 的关系

分区键（Partition Keys）、主键（Primary Keys）和唯一键（Unique Keys）的关系规则可以概括为：

分区表的分区表达式中使用的所有列必须是该表可能具有的每个唯一键的一部分。

> **注意**
>
> 唯一键包括 PrimaryKey 和 Unique KEY。

也就是说，表上的每个唯一键必须使用表的分区表达式中的每一列。唯一键也包括表的主键，因为根据定义，表的主键也是唯一键。

示例说明

例如，以下每个表创建语句都无效：

```
> CREATE TABLE t1 (
col1 INT NOT NULL,
col2 DATE NOT NULL,
col3 INT NOT NULL,
col4 INT NOT NULL,
UNIQUE KEY (col1, col2)
)
PARTITION BY HASH(col3)
PARTITIONS 4;
ERROR 1503 (HY000): A PRIMARY KEY must include all columns in the table's
partitioning function

> CREATE TABLE t2 (
col1 INT NOT NULL,
col2 DATE NOT NULL,
col3 INT NOT NULL,
col4 INT NOT NULL,
UNIQUE KEY (col1),
UNIQUE KEY (col3)
)
PARTITION BY HASH(col1 + col3)
PARTITIONS 4;

ERROR 1503 (HY000): A PRIMARY KEY must include all columns in the table's
partitioning function
```

关于 KEY 分区分区键为 NULL

1. KEY 只接受零个或多个列名的列表。在表有主键的情况下，用作分区键的任何列必须包含表主键的一部分或全部。

   如果没有指定列名作为分区键，则使用表的主键（如果有）。例如，以下 `CREATE TABLE` 语句在 MySQL 中有效。

2. 如果没有主键，但有 UNIQUE KEY，那么 UNIQUE KEY 用于分区键。

例如，以下建表语句中，KEY 分区分区键为 NULL，没有定义主键，但是含有唯一键，构建分区表达式时则使用唯一键作为分区键：

```
CREATE TABLE t1 (
col1 INT NOT NULL,
col2 DATE NOT NULL,
col3 INT NOT NULL,
col4 INT NOT NULL,
UNIQUE KEY (col1, col2)
)
PARTITION BY KEY()
PARTITIONS 4;
```

> **注意**
>
> 其他分区规则与 MySQL 基本保持一致。

3. 关于 MatrixOne 分区表达式的说明

在 DDL 语句构建分区表时，会回针对每一种分区定义生成一个分区表达式，该分区表达式可用于计算数据的所属的分区。

在计划构建阶段对 DDL 语句中的分区信息数据结构为 plan.PartitionInfo：

```
type PartitionInfo struct \{
Type PartitionType
Expr *Expr
PartitionExpression *Expr
Columns []*Expr
PartitionColumns []string
PartitionNum uint64
Partitions []*PartitionItem
Algorithm int64
IsSubPartition bool
PartitionMsg string
}
```

其中 `PartitionExpression` 即为分区表达式。分区表达式为 MatrixOne 把分区子句转换为一个表达式进行处理的方式，对每一种分区表达式的构建方式如下：

KEY Partitioning

KEY 分区会根据分区键和分区数量，构建一个分区表达式，分区表达式的计算结果为一个大于等于 0 的整数，代表分区序号，从零开始依次递增。

SQL 示例如下：

```
CREATE TABLE t1 (
col1 INT NOT NULL,
col2 DATE NOT NULL,
col3 INT NOT NULL,
col4 INT NOT NULL,
PRIMARY KEY (col1, col2)
)
PARTITION BY KEY(col1)
PARTITIONS 4;
```

HASH Partitioning

与 KEY 分区类似，HASH 分区会根据分区函数和分区数量，构建一个分区表达式，分区表达式的计算结果为一个大于等于 0 的整数，代表分区序号，从零开始依次递增。

SQL 示例如下：

```
CREATE TABLE t1 (
col1 INT,
col2 CHAR(5),
col3 DATE
)
PARTITION BY LINEAR HASH( YEAR(col3))
PARTITIONS 6;
```

RANGE Partitioning

RANGE 分区会根据分区函数，分区数量和分区项的定义，构建一个分区表达式，分区表达式的计算结果是一个整数，代表分区序号，正常的分区序号从零开始依次递增，如果分区表达式的计算结果为 -1，表示当前该数据不属于任何已定义分区，依 MySQL 语法，需要执行器报错：`Table has no partition for value xxx`。

SQL 示例如下：

```
CREATE TABLE employees (
id INT NOT NULL,
fname VARCHAR(30),
lname VARCHAR(30),
hired DATE NOT NULL DEFAULT '1970-01-01',
separated DATE NOT NULL DEFAULT '9999-12-31',
job_code INT NOT NULL,
store_id INT NOT NULL
)
PARTITION BY RANGE (store_id) (
PARTITION p0 VALUES LESS THAN (6),
PARTITION p1 VALUES LESS THAN (11),
PARTITION p2 VALUES LESS THAN (16),
PARTITION p3 VALUES LESS THAN MAXVALUE
);
```

RANGE COLUMNS partitioning

RANGE COLUMNS 分区会根据键列表，分区数量和分区项的定义，构建一个分区表达式，分区表达式的计算结果是一个整数，代表分区序号，正常的分区序号从零开始依次递增，如果分区表达式的计算结果为 -1，表示当前该数据不属于任何已定义分区，依 M 有 SQL 语法，需要执行器报错：`Table has no partition for value xxx`。

SQL 示例如下：

```
CREATE TABLE rc (
a INT NOT NULL,
b INT NOT NULL
)
PARTITION BY RANGE COLUMNS(a,b) (
PARTITION p0 VALUES LESS THAN (10,5) COMMENT = 'Data for LESS THAN
(10,5)',
PARTITION p1 VALUES LESS THAN (20,10) COMMENT = 'Data for LESS THAN
(20,10)',
PARTITION p2 VALUES LESS THAN (50,MAXVALUE) COMMENT = 'Data for LESS THAN
(50,MAXVALUE)',
PARTITION p3 VALUES LESS THAN (65,MAXVALUE) COMMENT = 'Data for LESS THAN
(65,MAXVALUE)',
PARTITION p4 VALUES LESS THAN (MAXVALUE,MAXVALUE) COMMENT = 'Data for
LESS THAN (MAXVALUE,MAXVALUE)'
);
```

LIST Partitioning

LIST 分区会根据分区键，分区数量和分区项的定义，构建一个分区表达式，分区表达式的计算结果是一个整数，代表分区序号，正常的分区序号从零开始依次递增，如果分区表达式的计算结果为 -1，表示当前该数据不属于任何已定义分区，依 MySQL 语法，需要执行器报错：`Table has no partition for value xxx`。

SQL 示例如下：

```
CREATE TABLE client_firms (
id INT,
name VARCHAR(35)
)
PARTITION BY LIST (id) (
PARTITION r0 VALUES IN (1, 5, 9, 13, 17, 21),
PARTITION r1 VALUES IN (2, 6, 10, 14, 18, 22),
PARTITION r2 VALUES IN (3, 7, 11, 15, 19, 23),
PARTITION r3 VALUES IN (4, 8, 12, 16, 20, 24)
);
```

LIST COLUMNS partitioning

LIST COLUMNS 分区会根据分区键列表，分区数量和分区项的定义，构建一个分区表达式，分区表达式的计算结果是一个整数，代表分区序号，正常的分区序号从零开始依次递增，如果分区表达式的计算结果为 -1，表示当前该数据不属于任何已定义分区，依 M 有 SQL 语法，需要执行器报错：`Table has no partition for value xxx`。

SQL 示例如下：

```
CREATE TABLE lc (
a INT NULL,
b INT NULL
)
PARTITION BY LIST COLUMNS(a,b) (
PARTITION p0 VALUES IN( (0,0), (NULL,NULL) ),
PARTITION p1 VALUES IN( (0,1), (0,2) ),
PARTITION p2 VALUES IN( (1,0), (2,0) )
);
```

MatrixOne 目录结构

完成 MatrixOne 搭建和连接后，首次执行时，MatrixOne 会自动生成以下目录，用于存放各类数据文件或元数据信息。

进入 *matrixone* 目录执行 `ls` 查看目录结构，相关目录结构以及用途如下：

matrixone ***//MatrixOne 主目录***
|— etc ***//配置文件目录***
| |— quickstart ***//配置文件目录***
|— mo-data ***//数据文件目录***
| |— local ***//本地 fileservice 目录***
| | |— cnservice ***//cn 节点信息目录***
| | |— dnservice ***//dn 节点信息目录***
| |— etl ***//外部表目录***
| | |— sys ***//外部表信息归属于哪个租户***
| | |— logs ***//统计信息的类型***
| | |— 2022 ***//统计信息的年份***
| | |— 10 ***//统计信息的月份***
| | |— 27 ***//统计信息的天数***
| | |— metric ***//统计信息对应表的存储目录***
| | |— rawlog ***//统计信息对应表的存储目录***
| | |— statement_info ***//统计信息对应表的存储路目录***
| | |— merged ***//过往外部表信息的合并记录***
| | |— 2022 ***//统计信息的年份***
| | |— 10 ***//统计信息的月份***
| | |— 27 ***//统计信息的天数***
| | |— metric ***//统计信息对应表的存储目录***
| | |— rawlog ***//统计信息对应表的存储目录***
| | |— statement_info ***//统计信息对应表的存储路目录***
| |— logservice ***//logservice 目录***
| |— 7c4dccb4-4d3c-41f8-b482-5251dc7a41bf ***//logservice 节点对应目录（随机生成 uuid）***
| |—hostname ***//MatrixOne 的服务器域名***
| | |— 00000000000000000001 ***//快照保存目录***
| | |— exported-snapshot ***//导出快照目录***
| | |— snapshot-part-n ***//快照分部保存目录***

```
|   |  |— tandb        //bootstrap 信息保存目录
|— s3        //数据保存目录
```

慢查询

慢查询，即在日志中记录运行比较慢的 SQL 语句。慢查询记录在慢查询日志中，通过慢查询日志，可以查找出哪些查询语句的执行效率低，以便进行优化。

当前 MatrixOne 的慢查询是超过 1000 毫秒的查询，暂不支持定向输出到对应的日志文件中，需要通过创建视图的方式进行过滤获取。

开启慢查询

慢查询日志默认关闭，要使用慢查询日志功能，首先要开启慢查询日志功能。

MatrixOne 的慢查询功能在版本 0.7.0 中提供了如下几个基础信息：

- `statement`：即 SQL 文本，用于提供完整的 SQL 语句。
- `request_at`：SQL 语句的的起始时间。
- `duration_second`：SQL 语句的实际执行时间。
- `exec_plan`：SQL 语句的详细执行计划。

执行如下内容的脚本，开启慢查询：

```
drop database if exists mo_ts;
create database mo_ts;
use mo_ts;
create view slow_query as select statement,request_at,duration/1000000000
as duration_second from system.statement_info where statement like
'select%' and duration/1000000000>1 order by request_at desc;
create view slow_query_with_plan as select
statement,request_at,duration/1000000000 as duration_second,exec_plan
from system.statement_info where statement like 'select%' and
duration/1000000000>1 order by request_at desc;
```

对于所有超过 1 秒的查询，可以执行如下语句：

```
mysql> select * from mo_ts.slow_query;
mysql> select * from mo_ts.slow_query_with_plan;
```

语句解释

- `select * from mo_ts.slow_query;`：不带执行计划。
- `select * from mo_ts.slow_query_with_plan;`：带执行计划。

错误日志

在开启了慢查询的情况下，可以开启错误日志，检查日志，定位错误信息。

开启错误日志

执行如下内容脚本：

```
create database mo_ts if not exists mo_ts;
use mo_ts;
create view error_message as select timestamp,message from
system.log_info where level in ('error','panic','faltal');
create view error_sql as select si.request_at
time_stamp,si.statement,si.error as SQL from system.statement_info si
where si.user<>'internal' and si.status='Failed' ;
```

查询数据库服务错误

查询数据库服务错误，执行如下 SQL：

```
mysql> select * from mo_ts.error_message;
```

查询结果示例如下：

```
+----------------------------+-----------------------------------------
------------------------------+
| timestamp | message |
+----------------------------+-----------------------------------------
------------------------------+
| 2022-11-28 14:47:31.324762 | error: SQL parser error: table "error_sql"
does not exist |
| 2022-11-28 14:47:31.324837 | SQL parser error: table "error_sql" does
not exist |
| 2022-11-28 14:47:31.324872 | query trace status |
| 2022-11-28 14:40:06.579795 | read loop stopped |
| 2022-11-28 14:40:06.585220 | gc inactive backends task stopped |
| 2022-11-28 14:40:06.591082 | error: cannot locate ha keeper |
| 2022-11-28 14:40:08.442515 | failed to propose initial cluster info |
| 2022-11-28 14:40:08.442667 | failed to set initial cluster info |
| 2022-11-28 14:40:09.411286 | error: timeout, converted to code 20429 |
| 2022-11-28 14:40:09.411508 | read loop stopped |
| 2022-11-28 14:40:09.416557 | gc inactive backends task stopped |
| 2022-11-28 14:40:10.052585 | error: internal error: failed to get task
service |
| 2022-11-28 14:40:10.052630 | failed to create init tasks |
| 2022-11-28 14:40:11.053926 | error: internal error: failed to get task
service |
| 2022-11-28 14:40:11.054059 | failed to create init tasks |
| 2022-11-28 14:40:12.054578 | error: internal error: failed to get task
service |
| 2022-11-28 14:40:12.054630 | failed to create init tasks |
| 2022-11-28 14:40:13.055828 | error: internal error: failed to get task
service |
| 2022-11-28 14:40:13.055896 | failed to create init tasks |
| 2022-11-28 14:40:14.057102 | error: internal error: failed to get task
service |
| 2022-11-28 14:40:14.057208 | failed to create init tasks |
| 2022-11-28 14:40:15.058425 | error: internal error: failed to get task
service |
| 2022-11-28 14:40:15.058563 | failed to create init tasks |
| 2022-11-28 14:40:16.059867 | error: internal error: failed to get task
service |
| 2022-11-28 14:40:16.060031 | failed to create init tasks |
| 2022-11-28 14:40:16.443234 | read loop stopped |
| 2022-11-28 14:40:16.443162 | read from backend failed |
| 2022-11-28 14:40:16.448858 | gc inactive backends task stopped |
| 2022-11-28 14:40:16.457276 | error: file dnservice/dd4dccb4-4d3c-41f8-
b482-5251dc7a41bf is not found |
```

```
| 2022-11-28 14:40:17.061260 | error: internal error: failed to get task
service |
| 2022-11-28 14:40:17.061323 | failed to create init tasks |
| 2022-11-28 14:40:18.062165 | error: internal error: failed to get task
service |
| 2022-11-28 14:40:18.062249 | failed to create init tasks |
| 2022-11-28 14:40:18.642097 | error: dn shard uuid , id 2 not reported |
| 2022-11-28 14:40:19.062775 | error: internal error: failed to get task
service |
| 2022-11-28 14:40:19.062937 | failed to create init tasks |
| 2022-11-28 14:40:20.063237 | error: internal error: failed to get task
service |
| 2022-11-28 14:40:20.063252 | failed to create init tasks |
| 2022-11-28 14:40:21.064529 | failed to create init tasks |
| 2022-11-28 14:40:21.064457 | error: internal error: failed to get task
service |
| 2022-11-28 14:40:21.463193 | read loop stopped |
| 2022-11-28 14:40:21.468423 | gc inactive backends task stopped |
| 2022-11-28 14:40:21.474688 | error: file cnservice/dd1dccb4-4d3c-41f8-
b482-5251dc7a41bf is not found |
| 2022-11-28 15:24:56.210577 | error: SQL parser error: table "error_sql"
does not exist |
| 2022-11-28 15:24:56.210773 | SQL parser error: table "error_sql" does
not exist |
| 2022-11-28 15:24:56.210898 | query trace status |
| 2022-11-28 14:40:22.065723 | error: internal error: failed to get task
service |
| 2022-11-28 14:40:22.065838 | failed to create init tasks |
| 2022-11-28 14:40:22.478229 | error: invalid state no cn in the cluster
|
| 2022-11-28 14:40:22.478846 | failed to refresh task storage |
| 2022-11-28 14:40:23.090160 | error: invalid database mo_task |
| 2022-11-28 14:40:23.090274 | invalid database mo_task |
| 2022-11-28 14:40:23.090604 | query trace status |
| 2022-11-28 15:32:30.354364 | error: SQL parser error: table
"slow_query" does not exist |
| 2022-11-28 15:32:30.354485 | SQL parser error: table "slow_query" does
not exist |
| 2022-11-28 15:32:30.354605 | query trace status |
| 2022-11-28 15:26:59.639892 | error: SQL parser error: table "error_sql"
does not exist |
| 2022-11-28 15:26:59.640039 | SQL parser error: table "error_sql" does
not exist |
| 2022-11-28 15:26:59.640208 | query trace status |
```

```
| 2022-11-28 15:37:29.289457 | error: table slow_query already exists |
| 2022-11-28 15:37:29.289486 | table slow_query already exists |
| 2022-11-28 15:37:29.289518 | query trace status |
| 2022-11-28 15:37:45.773829 | error: table slow_query_with_plan already
exists |
| 2022-11-28 15:37:45.773856 | table slow_query_with_plan already exists
|
| 2022-11-28 15:37:45.773888 | query trace status |
| 2022-11-28 14:45:48.821324 | error: not supported: function or operator
'interval' |
| 2022-11-28 14:45:48.823261 | error: not supported: function or operator
'interval' |
| 2022-11-28 14:45:48.823426 | error: not supported: function or operator
'interval' |
| 2022-11-28 14:45:48.823525 | error: not supported: function or operator
'interval' |
| 2022-11-28 14:47:14.513831 | error: SQL parser error: table
"statement_info" does not exist |
| 2022-11-28 14:47:14.513929 | SQL parser error: table "statement_info"
does not exist |
| 2022-11-28 14:47:14.513962 | query trace status |
+----------------------------+-----------------------------------------
-----------------------------+
72 rows in set (0.13 sec)
```

查询 SQL 错误

查询 SQL 错误，执行如下命令：

```
mysql> select * from mo_ts.error_sql;
```

查询结果示例如下：

```
+----------------------------+-------------------------------------------
-------------------------------------------------------------------------
-------------------------------------------------------------------------
-----------------------------------------+-----------------------------
----------------------------+
| time_stamp | statement | sql |
+----------------------------+-------------------------------------------
-------------------------------------------------------------------------
-------------------------------------------------------------------------
-----------------------------------------+-----------------------------
----------------------------+
| 2022-11-28 14:40:23.073188 | use mo_task | invalid database mo_task |
| 2022-11-28 15:26:59.637130 | select * from mo_ts.error_sql | SQL parser
error: table "error_sql" does not exist |
| 2022-11-28 15:37:29.283683 | create view slow_query as select
statement, request_at, duration / 1000000000 as duration_second from
system.statement_info where statement like "select%" and duration /
1000000000 > 1 order by request_at desc | table slow_query already exists
|
| 2022-11-28 15:37:45.765394 | create view slow_query_with_plan as select
statement, request_at, duration / 1000000000 as duration_second,
exec_plan from system.statement_info where statement like "select%" and
duration / 1000000000 > 1 order by request_at desc | table
slow_query_with_plan already exists |
| 2022-11-28 15:32:30.351695 | select * from mo_ts.slow_query | SQL
parser error: table "slow_query" does not exist |
| 2022-11-28 14:47:14.510060 | create view error_sql as select
si.request_at as time_stamp, si.statement as sql, el.err_code from
statement_info as si cross join error_info as el where si.statement_id =
el.statement_id and user != "internal" | SQL parser error: table
"statement_info" does not exist |
| 2022-11-28 14:47:31.323884 | select * from mo_ts.error_sql | SQL parser
error: table "error_sql" does not exist |
| 2022-11-28 15:24:56.208171 | select * from mo_ts.error_sql | SQL parser
error: table "error_sql" does not exist |
+----------------------------+-------------------------------------------
-------------------------------------------------------------------------
-------------------------------------------------------------------------
-----------------------------------------+-----------------------------
----------------------------+
8 rows in set (0.14 sec)
```

常用统计数据查询

统计数据是数据库在运维使用的过程中周期性进行的常用查询，可以帮助数据库用户较为直观准确地掌握当前数据库的状态以及健康程度。

在 MatrixOne 中，统计数据包含了如下几方面的内容：

- 元数据（Meatadata）：描述数据库的数据，包含数据库信息、表信息、列信息。
- SQL 统计：在特定时间范围内，SQL 的执行成功与否、执行用户、起始与停止时间。
- 角色与权限信息：通过查询，获取到 MatrixOne 下所有角色的授权、权限、继承信息，以及执行时间与授权人。

查看当前租户下所有数据库基本信息

```
> select md.datname as database_name,md.created_time as
created_time,mu.user_name as creator, mr.role_name as owner_role,
count(mt.reldatabase) as total_tables
from mo_catalog.mo_database md,mo_catalog.mo_role mr, mo_catalog.mo_user
mu, mo_catalog.mo_tables mt
where md.creator=mu.user_id and md.owner=mr.role_id and
mt.reldatabase_id=md.dat_id
group by
mt.reldatabase,md.datname,md.created_time,mu.user_name,mr.role_name
order by md.created_time asc;
```

查看所有的自增列相关信息

```
> select att_database as database_name,att_relname as table_name,attname
as column_name
from mo_catalog.mo_columns
where att_is_auto_increment=1
order by att_database, att_relname asc;
```

查看所有视图

```
> select mt.relname as view_name, mt.reldatabase as
database_name,mu.user_name as created_user,mr.role_name as
owner_role,mt.created_time
from mo_catalog.mo_tables mt, mo_catalog.mo_user mu, mo_catalog.mo_role
mr
where mt.relkind='v' and mt.creator=mu.user_id and mt.owner=mr.role_id
order by 1,2 asc;
```

查看所有外部表

```
> select mt.relname as view_name, mt.reldatabase as
database_name,mu.user_name as created_user,mr.role_name as
owner_role,mt.created_time
from mo_catalog.mo_tables mt, mo_catalog.mo_user mu, mo_catalog.mo_role
mr
where mt.relkind='e' and mt.creator=mu.user_id and mt.owner=mr.role_id
order by 1,2 asc;
```

查看所有表的主键

```
> select att_database as database_name,att_relname as table_name,attname
as column_name
from mo_catalog.mo_columns
where att_constraint_type='p' and att_relname not like '%!%'
order by att_database, att_relname asc;
```

查看所有没有主键的表

```
> select distinct att_database as database_name,att_relname as table_name
from mo_catalog.mo_columns
minus
select att_database as database_name,att_relname as table_name
from mo_catalog.mo_columns
where att_constraint_type='p'
order by database_name,table_name asc;
```

查看过去 24 小时内的 sql 统计（非 sys 租户暂不支持）

```
> select user,host,status,count(status) as count, date_sub(now(),
interval 24 hour) as start_time, now() as end_time
from system.statement_info
where status in ('Success','Failed') and user <> 'internal'
and request_at between date_sub(now(), interval 24 hour) and now()
group by status,user,host
order by user,status asc;
```

查看所有角色授予用户信息

```
> select mu.user_name as user_name,mr.role_name as
role_name,mug.with_grant_option
from mo_catalog.mo_user mu, mo_catalog.mo_role mr,
mo_catalog.mo_user_grant mug
where mu.user_id=mug.user_id and mr.role_id=mug.role_id
order by mu.user_name,mr.role_name asc;
```

查看所有角色权限信息

```
> select
mrp.role_name,mrp.privilege_name,mrp.obj_type,mrp.privilege_level,md.datname
as object_name,with_grant_option
from mo_catalog.mo_role_privs mrp, mo_catalog.mo_database md
where mrp.obj_id=md.dat_id and mrp.obj_type='database'
union
select
mrp.role_name,mrp.privilege_name,mrp.obj_type,mrp.privilege_level,'*',with_g
from mo_catalog.mo_role_privs mrp
where obj_id=0
union
select
mrp.role_name,mrp.privilege_name,mrp.obj_type,mrp.privilege_level,mt.relname
as object_name,with_grant_option
from mo_catalog.mo_role_privs mrp, mo_catalog.mo_tables mt
where mrp.obj_id=mt.rel_id and mrp.obj_type='table'
order by 1,2 asc;
```

查看所有角色继承信息

```
> select mr1.role_name as inheritor_role,mr2.role_name as
inheritee_role,mu.user_name as
operator_user,mrg.granted_time,mrg.with_grant_option
from mo_catalog.mo_user mu, mo_catalog.mo_role mr1, mo_catalog.mo_role
mr2,mo_catalog.mo_role_grant mrg
where mu.user_id=mrg.operation_user_id and mr1.role_id=mrg.grantee_id and
mr2.role_id=mrg.granted_id
order by mr1.role_name,mr2.role_name asc;
```

参考文档

本篇文章中所查询的 MatrixOne 系统数据库和表是 MatrixOne 存储系统信息的地方。如果你想了解更多关于系统数据库和表的详细信息，请参见 [MatrixOne 系统数据库和表](#)。

MatrixOne 数据库统计信息

MatrixOne 数据库统计信息是指数据库通过采样、统计出来的表、列的相关信息，例如，表的个数、表的列数、表所占的存储空间等。MatrixOne 数据库在生成执行计划时，需要根据统计信息进行估算，计算出最优的执行计划。

MatrixOne 数据库的统计信息维度如下：

查看数据库下表的个数

通过该命令，可以查看指定数据库下表的总数。

语法结构：

```
SHOW TABLE_NUMBER FROM {DATABASE_NAME}
```

示例

- **示例 1**：查看系统数据库 mo_catalog 下的表的总数：

```
mysql> show table_number from mo_catalog;
+--------------------------------+
| Number of tables in mo_catalog |
+--------------------------------+
| 11 |
+--------------------------------+

-- 验证一下是哪些表
mysql> use mo_catalog;
mysql> show tables;
+----------------------------+
| Tables_in_mo_catalog |
+----------------------------+
| mo_user |
| mo_account |
| mo_role |
| mo_user_grant |
| mo_role_grant |
| mo_role_privs |
| mo_user_defined_function |
| mo_columns |
| mo_mysql_compatbility_mode |
| mo_tables |
| mo_database |
+----------------------------+
11 rows in set (0.01 sec)
```

- **示例 2**：创建数据库，并创建了新的表，查询指定数据库下表的总数：

```
create database demo_1;
use demo_1;
-- 创建三个新的表
CREATE TABLE t1(a bigint, b varchar(10), c varchar(10));
CREATE TABLE t2(a bigint, b int);
CREATE TABLE t3(a int, b varchar(10), c varchar(10));

-- 查询出数据库 demo_1 中有三个表
mysql> show table_number from demo_1;
+----------------------------+
| Number of tables in demo_1 |
+----------------------------+
| 3 |
+----------------------------+
1 row in set (0.01 sec)
```

查看表拥有的列数

通过该命令，可以查看指定表的总列数。

语法结构：

```
SHOW COLUMN_NUMBER FROM {[DATABASE_NAME.]TABLE_NAME}
```

示例

```
use mo_catalog;
use mo_user;
mysql> show column_number from mo_user;
+-----------------------------+
| Number of columns in mo_user |
+-----------------------------+
| 11 |
+-----------------------------+

-- 或者使用下面的命令
mysql> show column_number from mo_catalog.mo_user;
+-----------------------------+
| Number of columns in mo_user |
+-----------------------------+
| 11 |
+-----------------------------+

-- 查看验证有哪些列
mysql> desc mo_catalog.mo_user;
+----------------------+--------------+------+------+---------+-------+-
--------+
| Field | Type | Null | Key | Default | Extra | Comment |
+----------------------+--------------+------+------+---------+-------+-
--------+
| user_id | INT | YES | | NULL | | |
| user_host | VARCHAR(100) | YES | | NULL | | |
| user_name | VARCHAR(300) | YES | | NULL | | |
| authentication_string | VARCHAR(100) | YES | | NULL | | |
| status | VARCHAR(8) | YES | | NULL | | |
| created_time | TIMESTAMP | YES | | NULL | | |
| expired_time | TIMESTAMP | YES | | NULL | | |
| login_type | VARCHAR(16) | YES | | NULL | | |
| creator | INT | YES | | NULL | | |
| owner | INT | YES | | NULL | | |
| default_role | INT | YES | | NULL | | |
+----------------------+--------------+------+------+---------+-------+-
--------+
11 rows in set (0.01 sec)
```

查看表中所有列包含的最大与最小值

通过该命令，可以查看指定表中的每一列的最大值与最小值。

Note: 如果指定表中各列值的数据类型不一致，则排序规则为：按照数字的大小排序；日期按照时间先后排序；字符类按照 ASCII 码排序；当几种数据类型混合排序时，非字符类型的则先转换为字符类型，然后按 ASCII 码排序。

语法结构：

```
SHOW TABLE_VALUES FROM {[DATABASE_NAME.]TABLE_NAME}
```

示例

```
create table t1(
col1 int,
col2 float,
col3 varchar
);
insert into t1 values(1,1.11,'1.111'),(2,2.22,'1.222'),(3,0,'abc');

mysql> show table_values from t1;
+-----------+-----------+-----------+-----------+-----------+-----------+
| max(col1) | min(col1) | max(col2) | min(col2) | max(col3) | min(col3) |
+-----------+-----------+-----------+-----------+-----------+-----------+
| 3 | 1 | 2.22 | 0 | abc | 1.111 |
+-----------+-----------+-----------+-----------+-----------+-----------+
```

查看表中的数据总行数

通过调用该函数，即可获得数据库中某张表的数据总行数。

语法结构：

```
MO_TABLE_ROWS({DATABASE_NAME},{TABLE_NAME})
```

示例

```
-- 查询数据库 mo_catalog 下表 mo_tables 的总行数
mysql> select mo_table_rows('mo_catalog','mo_tables');
+--------------------------------------+
| mo_table_rows(mo_catalog, mo_tables) |
+--------------------------------------+
| 64 |
+--------------------------------------+
```

查看表在存储中占用的空间

通过调用该函数，即可获得数据库中某张表占用的存储空间，单位是字节数。

语法结构：

```
MO_TABLE_SIZE({DATABASE_NAME},{TABLE_NAME})
```

示例

```
-- 查询数据库 mo_catalog 下表 mo_tables 占用的存储空间
mysql> select mo_table_size('mo_catalog','mo_tables');
+--------------------------------------+
| mo_table_size(mo_catalog, mo_tables) |
+--------------------------------------+
| 16128 |
+--------------------------------------+
```

审计

概述

审计是用来记录数据库用户行为以及数据库内部重要事件的功能，它记录了所有用户在登录数据库后做出的所有数据库操作以及数据内部的重大事件。也是很多企业级数据库必备的功能之一。在 MatrixOne 0.7.0 的版本中，审计功能已经具备了最初的形态，可供用户进行测试体验。为了方便用户能够更方便地获取自己所需要的相关信息，可以通过本文档进行开启。

在日常的数据库运维中，为了确保数据库用户的所有行为合规合法，审计是非常有效的手段。在数据库发生重要事件时，例如启停、节点宕机等，审计内容可以非常方便地追踪到前后时段的护具库行为。

对于重要的业务信息表或系统配置表需要进行有效完整的行为监控时，数据库审计的开启十分有必要。例如监控对用户 A 在数据库中所有行为，以便于及时发现违规的数据修改或删除来源。对于数据库内部重大事件的监控，可以第一时间排查故障，并且追溯事故产生的根本原因。

开启审计

执行如下内容脚本，开启审计功能：

```
drop database if exists mo_audits;
create database mo_audits;
use mo_audits;
create view mo_user_action as select
request_at,user,host,statement,status from system.statement_info where
user in (select distinct user_name from mo_catalog.mo_user) and statement
not like '______internal_%' order by request_at desc;
create view mo_events as select timestamp,level,message from
system.log_info where level in ('error','panic','fatal') order by
timestamp desc;
```

审计查询

对用户行为进行审计时，执行下面的 SQL 语句进行查看：

```
mysql> select * from mo_audits.mo_user_action;
```

查询示例结果如下：

```
+----------------------------+------+---------+-------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
-------------------------------------------------------------------------+----
-----+
| request_at | user | host | statement | status |
+----------------------------+------+---------+-------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
-------------------------------------------------------------------------+----
-----+
| 2023-02-10 19:54:28.831970 | root | 0.0.0.0 | create view
mo_user_action as select request_at, user, host, statement, status from
system.statement_info where user in (select distinct user_name from
mo_catalog.mo_user) and statement not like "______internal_%" order by
request_at desc | Success |
| 2023-02-10 19:54:14.079939 | root | 0.0.0.0 | show tables | Success |
| 2023-02-10 19:54:14.076260 | root | 0.0.0.0 | show databases | Success
|
| 2023-02-10 19:54:14.071728 | root | 0.0.0.0 | use mo_audits | Success |
| 2023-02-10 19:54:14.071108 | root | 0.0.0.0 | select database() |
Success |
| 2023-02-10 19:54:01.007241 | root | 0.0.0.0 | create database mo_audits
| Success |
| 2023-02-10 19:53:48.924819 | root | 0.0.0.0 | drop database if exists
mo_audits | Success |
| 2023-02-10 19:30:59.668646 | root | 0.0.0.0 | show triggers | Success |
| 2023-02-10 19:30:53.438212 | root | 0.0.0.0 | show locks | Success |
| 2023-02-10 19:30:44.258894 | root | 0.0.0.0 | show index from t |
Success |
| 2023-02-10 19:30:43.662063 | root | 0.0.0.0 | create table t (a int, b
int, c int, primary key (a)) | Success |
| 2023-02-10 19:30:23.104830 | root | 0.0.0.0 | show triggers | Success |
| 2023-02-10 19:30:20.062010 | root | 0.0.0.0 | show tables | Success |
| 2023-02-10 19:30:20.060324 | root | 0.0.0.0 | show databases | Success
|
| 2023-02-10 19:30:20.055515 | root | 0.0.0.0 | use aab | Success |
| 2023-02-10 19:30:20.055186 | root | 0.0.0.0 | select database() |
Success |
| 2023-02-10 19:30:17.152087 | root | 0.0.0.0 | create database aab |
Success |
| 2023-02-10 19:30:10.621294 | root | 0.0.0.0 | create aab | Failed |
| 2023-02-10 19:29:59.983433 | root | 0.0.0.0 | show databases | Success
|
```

```
| 2023-02-10 19:29:45.370956 | root | 0.0.0.0 | show index from t |
Failed |
| 2023-02-10 19:29:44.875580 | root | 0.0.0.0 | create table t (a int, b
int, c int, primary key (a)) | Failed |
| 2023-02-10 19:29:44.859588 | root | 0.0.0.0 | drop table if exists t |
Success |
| 2023-02-10 19:29:19.974775 | root | 0.0.0.0 | show index | Failed |
| 2023-02-10 19:29:11.188286 | root | 0.0.0.0 | show locks | Success |
| 2023-02-10 19:29:06.618778 | root | 0.0.0.0 | show node list | Success
|
| 2023-02-10 19:19:11.319058 | root | 0.0.0.0 | show triggers | Failed |
| 2023-02-10 19:19:06.809302 | root | 0.0.0.0 | show databases | Success
|
| 2023-02-10 19:18:52.840282 | root | 0.0.0.0 | show triggers | Failed |
| 2023-02-10 10:54:09.892254 | root | 0.0.0.0 | show databases | Success
|
| 2023-02-10 10:54:04.468721 | root | 0.0.0.0 | select @@version_comment
limit 1 | Success |
+----------------------------+------+---------+-------------------------
-------------------------------------------------------------------------
-------------------------------------------------------------------------
-----------------------------------------------------------------+----
-----+
30 rows in set (0.81 sec)
```

查询数据库内部状态变更查询，执行下面的 SQL 语句进行查看：

```
mysql> select * from mo_events;
```

查询示例结果如下：

```
|
| 2022-10-18 15:26:20.293735 | error | error: timeout, converted to code
20429 |
| 2022-10-18 15:26:20.293725 | error | failed to propose initial cluster
info |
| 2022-10-18 15:26:20.288695 | error | failed to set initial cluster info
|
| 2022-10-18 15:26:20.288559 | error | failed to propose initial cluster
info |
| 2022-10-18 15:26:20.285384 | error | failed to set initial cluster info
|
| 2022-10-18 15:26:20.285235 | error | failed to propose initial cluster
info |
| 2022-10-18 15:26:18.473472 | error | failed to join the gossip group, 1
error occurred:
* Failed to join 127.0.0.1:32022: dial tcp 127.0.0.1:32022: connect:
connection refused |
| 2022-10-18 15:26:18.469029 | error | failed to join the gossip group, 1
error occurred:
* Failed to join 127.0.0.1:32012: dial tcp 127.0.0.1:32012: connect:
connection refused
```

关闭审计

执行下面的 SQL 语句，关闭审计：

```
> drop database if exists mo_audits;
```

错误码

在 MatrixOne 中，错误信息是根据错误编码进行分类，某一类的错误编码会统一至一个确定的错误编码中，以便于用户进行排查。MatrixOne 数据库服务所发出的所有消息都分配有五个字符的错误代码，既包含了错误编码的种类，又包含了错误编码的具体类别。

实际使用过程中，你可以优先查看错误编码，搭配错误编码附带的文本错误消息，来确定发生了哪种错误情况。错误编码不会发生跨版本更改，并且也不会因错误消息的本地化而发生更改。

> **注意**
>
> MatrixOne 生成的一部分错误编码是由 SQL 标准定义的；一些未被标准定义的条件的额外错误编码是从其他数据库中发明或借用的。

错误编码开头	类型
201	内部错误
202	数字与函数错误
203	无效操作
204	未知错误或 IO 错误
205	RPC 超时
206	事务错误或存储引擎错误

详细错误码对应的错误信息及错误详情请参见下表：

错误码	错误信息	错误详情	错误分类
20100	ErrStart	internal error code start	内部错误
20101	ErrInternal	Internal error	内部错误
20103	ErrNYI	not yet implemented	内部错误
20104	ErrOOM	out of memory	内部错误
20105	ErrQueryInterrupted	query interrupted	内部错误

错误码	错误信息	错误详情	错误分类
20106	ErrNotSupported	not supported	内部错误
20200	ErrDivByZero	division by zero	数字与函数
20201	ErrOutOfRange	data out of range	数字与函数
20202	ErrDataTruncated	data truncated	数字与函数
20203	ErrInvalidArg	invalid argument	数字与函数
20204	ErrTruncatedWrongValueForField	truncated wrong value for column	数字与函数
20300	ErrBadConfig	invalid configuration	无效操作
20301	ErrInvalidInput	invalid input	无效操作
20302	ErrSyntaxError	SQL syntax error	无效操作
20303	ErrParseError	SQL parser error	无效操作
20304	ErrConstraintViolation	constraint violation	无效操作
20305	ErrDuplicate	tae data duplicated	无效操作
20306	ErrRoleGrantedToSelf	cannot grant role	无效操作
20307	ErrDuplicateEntry	duplicate entry for key	无效操作
20400	ErrInvalidState	invalid state	未知状态或 I/O 错误
20401	ErrLogServiceNotReady	log service not ready	未知状态或 I/O 错误
20402	ErrBadDB	invalid database	未知状态或 I/O 错误
20403	ErrNoSuchTable	no such table	未知状态或 I/O 错误
20404	ErrEmptyVector	empty vector	未知状态或 I/O 错误
20405	ErrFileNotFound	file is not found	未知状态或 I/O 错误
20406	ErrFileAlreadyExists	file alread exists	未知状态或 I/O 错误

错误码	错误信息	错误详情	错误分类
20407	ErrUnexpectedEOF	unexpteded end of file	未知状态或 I/O 错误
20408	ErrEmptyRange	empty range of file	未知状态或 I/O 错误
20409	ErrSizeNotMatch	file size does not match	未知状态或 I/O 错误
20410	ErrNoProgress	file has no io progress	未知状态或 I/O 错误
20411	ErrInvalidPath	invalid file path	未知状态或 I/O 错误
20412	ErrShortWrite	file io short write	未知状态或 I/O 错误
20413	ErrInvalidWrite	file io invalid write	未知状态或 I/O 错误
20414	ErrShortBuffer	file io short buffer	未知状态或 I/O 错误
20415	ErrNoDB	not connect to a database	未知状态或 I/O 错误
20416	ErrNoWorkingStore	no working store	未知状态或 I/O 错误
20417	ErrNoHAKeeper	cannot locate ha keeper	未知状态或 I/O 错误
20418	ErrInvalidTruncateLsn	invalid truncate lsn, shard already truncated	未知状态或 I/O 错误
20419	ErrNotLeaseHolder	not lease holder, current lease holder ID xxx	未知状态或 I/O 错误
20420	ErrDBAlreadyExists	database already exists	未知状态或 I/O 错误
20421	ErrTableAlreadyExists	table already exists	未知状态或 I/O 错误
20422	ErrNoService	service not found	未知状态或 I/O 错误

错误码	错误信息	错误详情	错误分类
20423	ErrDupServiceName	duplicate service name	未知状态或 I/O 错误
20424	ErrWrongService	wrong service, expecting A, got B	未知状态或 I/O 错误
20425	ErrBadS3Config	bad s3 config	未知状态或 I/O 错误
20426	ErrBadView	invalid view	未知状态或 I/O 错误
20427	ErrInvalidTask	invalid task	未知状态或 I/O 错误
20428	ErrInvalidServiceIndex	invalid service idx	未知状态或 I/O 错误
20429	ErrDragonboatTimeout	Dragonboat timeout	未知状态或 I/O 错误
20430	ErrDragonboatTimeoutTooSmall	Dragonboat timeout too small	未知状态或 I/O 错误
20431	ErrDragonboatInvalidDeadline	Dragonboat invalid deadline	未知状态或 I/O 错误
20432	ErrDragonboatRejected	Dragonboat rejected	未知状态或 I/O 错误
20433	ErrDragonboatInvalidPayloadSize	invalid payload size	未知状态或 I/O 错误
20434	ErrDragonboatShardNotReady	shard not ready	未知状态或 I/O 错误
20435	ErrDragonboatSystemClosed	Dragonboat system closed	未知状态或 I/O 错误
20436	ErrDragonboatInvalidRange	Dragonboat invalid range	未知状态或 I/O 错误
20437	ErrDragonboatShardNotFound	shard not found	未知状态或 I/O 错误
20438	ErrDragonboatOtherSystemError	other system error	未知状态或 I/O 错误

错误码	错误信息	错误详情	错误分类
20439	ErrDropNonExistsDB	Can't drop database ; database doesn't exist	未知状态或 I/O 错误
20500	ErrRPCTimeout	rpc timeout	RPC 超时
20501	ErrClientClosed	client closed	RPC 超时
20502	ErrBackendClosed	backend closed	RPC 超时
20503	ErrStreamClosed	stream closed	RPC 超时
20504	ErrNoAvailableBackend	no available backend	RPC 超时
20600	ErrTxnClosed	the transaction has been committed or aborted	事务
20601	ErrTxnWriteConflict	transaction write conflict	事务
20602	ErrMissingTxn	missing transaction	事务
20603	ErrUnresolvedConflict	unresolved conflict	事务
20604	ErrTxnError	transaction error	事务
20605	ErrDNShardNotFound	dn shard not found	事务
20606	ErrShardNotReported	dn shard not reported	事务
20607	ErrTAEError	tae error	TAE 错误
20608	ErrTAERead	tae read error	TAE 错误
20609	ErrRpcError	rpc error	TAE 错误
20610	ErrWaitTxn	transaction wait error	TAE 错误
20611	ErrTxnNotFound	transaction not found	TAE 错误
20612	ErrTxnNotActive	transaction not active	TAE 错误
20613	ErrTAEWrite	tae write error	TAE 错误
20614	ErrTAECommit	tae commit error	TAE 错误
20615	ErrTAERollback	tae rollback error	TAE 错误
20616	ErrTAEPrepare	tae prepare error	TAE 错误
20617	ErrTAEPossibleDuplicate	tae possible duplicate	TAE 错误

错误码	错误信息	错误详情	错误分类
20618	ErrTxnRWConflict	r-w conflict	TAE 错误
20619	ErrTxnWWConflict	w-w conflict	TAE 错误
20620	ErrNotFound	transaction not found	TAE 错误
20621	ErrTxnInternal	transaction internal error	TAE 错误
20622	ErrTxnReadConflict	transaction read conflict	TAE 错误
20623	ErrPrimaryKeyDuplicated	duplicated primary key	TAE 错误
20624	ErrAppendableSegmentNotFound	appendable segment not found	TAE 错误
20625	ErrAppendableBlockNotFound	appendable block not found	TAE 错误
20626	ErrTAEDebug	TAE debug	TAE 错误

产品常见问题

- **什么是 MatrixOne?**

MatrixOne 是一款面向未来的超融合异构云原生数据库，通过超融合数据引擎支持事务/分析/流处理等混合工作负载，通过异构云原生架构支持跨机房协同/多地协同/云边协同。
MatrixOne 希望简化数据系统开发和运维的成本，消减复杂系统间的数据碎片，打破数据融合的各种边界。
想了解更多关于 MatrixOne 的信息，您可以浏览 MatrixOne 简介。

- **MatrxOne 支持哪些应用?**

  MatrixOne 为用户提供了极致的 HTAP 服务，MatrixOne 可以被应用在企业数据中台，大数据分析等场景中。

- **MatrixOne 是基于 MySQL 或者其他数据库开发的吗?**

MatrixOne 是一个从零打造的全新数据库。MatrixOne 兼容 MySQL 的部分语法与语义，并且在未来将会产生更多与 MySQL 不同的语义，以便我们将之打造为一款更强大的超融合数据库。 关于与 MySQL 的兼容性，您可参见 MySQL 兼容性。

- **MatrixOne 使用什么编程语言开发的?**

  目前，我们主要使用 **Golang** 作为最主要的编程语言。

- **MatrixOne 可以在什么操作系统上部署?**

  MatrixOne 支持在 Linux 与 MacOS 系统上部署。更多信息，参见部署常见问题.

- **MatrixOne 都支持哪些数据类型?**

  有关数据类型的更多信息，参见 MatrixOne 数据类型。

- **我可以在哪里部署 MatrixOne?**

  MatrixOne 可以本地部署、公共云、私有云或 kubernetes 上。

- **可以参与贡献 MatrixOne 项目吗?**

  MatrixOne 是一个完全在 Github 上进行的开源项目，欢迎所有开发者的贡献。更多信息，参见我们的贡献指南。

- **除了官方文档，是否还有其他 MatrixOne 知识获取途径?**

目前，MatrixOne 文档是获取 MatrixOne 相关知识最重要、最及时的途径。此外，我们在 Slack 和微信还有一些技术交流群。如有任何需求，请联系 opensource@matrixorigin.io。

部署常见问题

操作系统要求

- **部署 MatrixOne 所需的操作系统版本是什么？**
- 单机推荐配置：MatrixOne 当前支持下表中操作系统。

Linux OS	版本
Red Hat Enterprise Linux	7.3 or later 7.x releases
CentOS	7.3 or later 7.x releases
Oracle Enterprise Linux	7.3 or later 7.x releases
Ubuntu LTS	22.04 or later

- MatrixOne 也支持 macOS 操作系统，当前仅建议在测试和开发环境运行。

macOS	版本
macOS	Monterey 12.3 or later

硬件要求

MatrixOne 对部署硬件的配置要求如何？**

单机安装情况下，MatrixOne 当前可以运行在 Intel x86-64 架构的 64 位通用硬件服务器平台上。

对于开发、测试和生产环境的服务器硬件配置要求和建议如下：

开发和测试环境要求

CPU	内存	本地存储
4 core+	16 GB+	SSD/HDD 200 GB+

ARM 架构的 Macbook M1/M2 也适合开发环境。

生产环境要求

CPU	内存	本地存储
16 core+	64 GB+	SSD/HDD 500 GB+

安装和部署

安装时需要更改什么设置吗?

通常情况下，安装时，你无需更改任何设置。`launch.toml` 默认设置完全可以直接运行 MatrixOne。但是如果你需要自定义监听端口、IP 地址、存储数据文件路径，你可以修改相应的 `cn.toml`、`dn.toml` 或 `log.toml`，这些文件内参数配置详情可参考[通用参数配置]

当我安装完成 MySQL 客户端后，打开终端运行 `mysql` 产生报错 `command not found: mysql`，我该如何解决?

产生这个报错是环境变量未设置的原因，打开一个新的终端，执行以下命令：

=== "Linux 环境"

```
```
cd ~
sudo vim /etc/profile
password:
```

回车执行上面的命令后，需要输入 root 用户密码，即你在安装 MySQL 客户端时，你在安装窗口设置的 root 密码；如果没有设置密码，则直接回车跳过即可。

输入/跳过 root 密码后，即进入了 *profile* 文件，点击键盘上的 *i* 进入 insert 状态，即可在文件下方输入如下命令：

```
export PATH=/software/mysql/bin:$PATH
```

输入完成后，点击键盘上的 esc 退出 insert 状态，并在最下方输入 `:wq` 保存退出。继续输入 `source /etc/profile`，回车执行，运行环境变量。
```

=== "MacOS 环境"

```
cd ~
sudo vim .bash_profile
Password:
```

回车执行上面的命令后，需要输入 root 用户密码，即你在安装 MySQL 客户端时，你在安装窗口设置的 root 密码；如果没有设置密码，则直接回车跳过即可。

输入/跳过 root 密码后，即进入了 *profile* 文件，点击键盘上的 *i* 进入 insert 状态，即可在文件下方输入如下命令：

```
export PATH=${PATH}:/usr/local/mysql/bin
```

输入完成后，点击键盘上的 esc 退出 insert 状态，并在最下方输入 `:wq` 保存退出。继续输入 `source .bash_profile`，回车执行，运行环境变量。

当我安装选择从源代码安装构建 MatrixOne 时，产生了以下错误或构建失败提示，我该如何继续?

报错：

```
Get "https://proxy.golang.org/........": dial tcp 142.251.43.17:443: i/o timeout
```

由于 MatrixOne 需要许多 GO 库作为依赖项，所以它在构建时，会同时下载 GO 库。上述所示的报错是下载超时的错误，主要原因是网络问题。

- 如果你使用的是中国大陆的网络，你需要设置你的 GO 环境到一个中国的镜像网站，以加快 GO 库的下载。
- 如果你通过 `go env` 检查你的 GO 环境，你可能会看到 `GOPROXY="https://proxy.golang.org,direct"`，那么你需要进行下面设置：

```
go env -w GOPROXY=https://goproxy.cn,direct
```

设置完成后，`make build` 应该很快就能完成。

当我想将 MatrixOne 的数据目录保存到我指定的文件目录中，我应该如何操作?

当你使用 Docker 启动 MatrixOne，你可以将你指定的数据目录挂载至 Docker 容器，参见挂载目录到 Docker 容器。

当你使用源码或二进制包编译并启动 MatrixOne，你可以通过修改配置文件中的默认数据目录路径：打开 MatrixOne 源码文件目录 `matrixone/etc/launch-tae-CN-tae-DN`，修改 `cn.toml`、`dn.toml` 和 `log.toml` 三个文件内的配置参数 `data-dir = "./mo-data"` 为 `data-dir = "your_local_path"`，保存后重启 MatrixOne 即可生效。

当我通过 MO-Tester 对 MatrixOne 进行测试时，我如何解决产生的 `too many open files` 错误?

为了对 MatrixOne 进行测试，MO-Tester 会快速地打开和关闭许多 SQL 文件，于是很快就达到 Linux 和 MacOS 系统的最大打开文件限制，这就是导致 `too many open files` 错误的原因。

- 对于 MacOS 系统，你可以通过一个简单的命令设置打开文件的限制：

```
ulimit -n 65536
```

- 对于 Linux 系统，请参考详细的指南，将 *ulimit* 设置为 100000。

设置完成后，将不会出现 `too many open files` 错误。

我的 PC 是 M1 芯片，当我进行 SSB 测试时，发现无法编译成功 ssb-dbgen

硬件配置为 M1 芯片的 PC 在编译 `ssb-dbgen` 之前，还需要进行如下配置：

1. 下载并安装 GCC11。
2. 输入命令，确认 gcc-11 是否成功：

   ```
   gcc-11 -v
   ```

   如下结果，表示成功：

```
Using built-in specs.
COLLECT_GCC=gcc-11
COLLECT_LTO_WRAPPER=/opt/homebrew/Cellar/gcc@11/11.3.0/bin/../libexec/gc
apple-darwin21/11/lto-wrapper
Target: aarch64-apple-darwin21
Configured with: ../configure --prefix=/opt/homebrew/opt/gcc@11 --
libdir=/opt/homebrew/opt/gcc@11/lib/gcc/11 --disable-nls --enable-
checking=release --with-gcc-major-version-only --enable-
languages=c,c++,objc,obj-c++,fortran --program-suffix=-11 --with-
gmp=/opt/homebrew/opt/gmp --with-mpfr=/opt/homebrew/opt/mpfr --with-
mpc=/opt/homebrew/opt/libmpc --with-isl=/opt/homebrew/opt/isl --with-
zstd=/opt/homebrew/opt/zstd --with-pkgversion='Homebrew GCC 11.3.0' --
with-bugurl=https://github.com/Homebrew/homebrew-core/issues --
build=aarch64-apple-darwin21 --with-system-zlib --with-
sysroot=/Library/Developer/CommandLineTools/SDKs/MacOSX12.sdk
Thread model: posix
Supported LTO compression algorithms: zlib zstd
gcc version 11.3.0 (Homebrew GCC 11.3.0)
```

3. 手动修改 *ssb-dbgen* 目录下的 *bm_utils.c* 配置文件：
   - 将第 41 行的 `#include <malloc.h>` 修改为 `#include <sys/malloc.h>`
   - 将第 398 行的
     ```
     open(fullpath, ((*mode == 'r')?
     O_RDONLY:O_WRONLY)|O_CREAT|O_LARGEFILE,0644);
     ```
     修改为
     ```
     open(fullpath, ((*mode == 'r')?O_RDONLY:O_WRONLY)|O_CREAT,0644);
     ```
4. 手动修改 *ssb-dbgen* 目录下的 *varsub.c* 配置文件：
   - 将第 5 行的 `#include <malloc.h>` 修改为 `#include <sys/malloc.h>`
5. 手动修改 *ssb-dbgen* 目录下的 *makefile* 配置文件：
   - 将第 5 行的 `CC = gcc` 修改为 `CC = gcc-11`
6. 再次进入 *ssb-dbgen* 目录，进行编译：

```
cd ssb-dbgen
make
```

7. 查看 *ssb-dbgen* 目录，生成 *dbgen* 可执行文件，表示编译成功。

我先在 main 分支构建了 MatrixOne，现在切换到其他版本再进行构建出现 panic

如果你选择某个版本的代码并 `make build` 编译构建了 MatrixOne，则会产生 *mo-data* 的数据文件目录，此时如果你需要切换版本（即 `git checkout version-name`），由于版本不兼容，你需要先清理 *mo-data*（即 `rm -rf mo-data`），再构建 MatrixOne。代码示例：

```
[root ~]# cd matrixone // 进入 matrixone 文件目录
[root ~]# git branch // 查看当前分支
* main
[root ~]# make build // 构建 matrixone
... // 此处省略构建过程代码。如果你此时想要切换到其他版本，例如 0.7.0 版本
[root ~]# git checkout 0.7.0 // 切换到 0.7.0 版本
[root ~]# rm -rf mo-data // 清理数据目录
[root ~]# make build // 构建 matrixone
... // 此处省略构建过程代码
[root ~]# ./mo-service --daemon --launch ./etc/quickstart/launch.toml &>
test.log & // 在终端的后台启动 MatrixOne 服务
```

SQL 常见问题

- **MatrixOne 中的函数和关键字是否区分大小写?**

  不区分大小写。

  在 MatrixOne 中，只有一种情况需要区分大小写：如果你创建的表和属性带有 “，“ 中的名称需要注意大小写。查询这个表名或属性名，那么表名和属性名也需要被包含在“里。

- **如何将数据从 MatrixOne 导出到文件?**

你可以使用 `SELECT INTO OUTFILE` 命令来将数据导出为 **csv** 文件（只能导出到服务器主机，无法到远程客户端）。
关于该命令的更多信息，参见 [SELECT 参考指南](#)。

- **MatrixOne 事务大小的限制是什么?**

  事务大小受限于硬件环境的内存大小。

- **MatrixOne 支持什么类型的字符集?**

  MatrixOne 默认支持 UTF-8 字符集，且目前只支持 UTF-8。

- **MatrixOne 中的 `sql_mode` 是什么?**

  MatrixOne 默认的 `sql_mode` 是 MySQL 中的 `only_full_group_by`。目前 MatrixOne 不支持修改 `sql_mode`。

- **我如何批量将数据加载到 MatrixOne?**

  MatrixOne 提供了两种批量加载数据的方法：

  - 在 shell 中使用 `source filename` 命令，你可以加载包含所有 DDL 的 SQL 文件并插入数据语句。
  - 使用 `load data infile...into table...` 命令，你可以加载一个现有的*. csv* 文件到 MatrixOne。

- **我怎么知道我的查询是如何执行的?**

  要查看 MatrixOne 对给定查询的执行情况，可以使用 `EXPLAIN` 语句，它将打印出查询计划。

```
EXPLAIN SELECT col1 FROM tbl1;
```

MatrixOne 版本发布历史记录

本文列出了所有已发布的 MatrixOne 版本，按发布时间倒序排列呈现。

版本	发布日期
v0.7.0	2023/2/23
v0.6.0	2022/11/29
v0.5.1	2022/8/19
v0.5.0	2022/7/18
v0.4.0	2022/5/5
v0.3.0	2022/3/10
v0.2.0	2022/1/6
v0.1.0	2021/10/24

MatrixOne v0.7.0 发布报告

热烈祝贺 MatrixOne 的 v0.7.0 版本于 2023 年 2 月 23 日正式发布！在这个版本中，MatrixOne 在云原生架构和完整数据库功能形态下，版本 0.7.0 进行了稳定性和性能的专项优化：

- 在稳定性方面，一方面我们优化了整体的内存管理机制，极大的降低了内存泄漏发生的概率；另一方面调整了部分模块的超时机制，使得一些极端情况下的导入数据及重启不会导致系统停止响应。同时对分布式架构单一模块失效导致整体崩溃的极端情况进行了修复。
- 在性能方面，此次迭代针对读和写分别做了大量优化。在读的方面，实现了基于

  Zonemap 的统计信息，并给出更优的执行计划。实现了在建表时增加 `Cluster by` 字段，可以预先将数据做好排布，以方便更快的获取。另外也优化了 `IN` 操作符、类型转换、谓词过滤等众多性能卡点。在写的方面，这个迭代实现了直接通过 `CN` 节点大批量

  并行写入共享存储，从而获得了加载性能的大幅提升。

Docker

```
docker pull matrixorigin/matrixone:0.7.0
```

最新特性

- 新增并行加载数据模式。
- 新增外键 `Foreign Key`。
- 新增建表 `Cluster by` 字段。
- 新增 `MYSQL_COMPATBILITY_MODE` 参数管理部分 MySQL 专属兼容行为。
- 修改系统默认隔离级别参数为 `REPEATABLE_READ` 实现更优的 MySQL 兼容。

- `Unique Index` 实现完整唯一约束。
- 支持 Load 从客户端本地导入数据。
- 新增 `Alter View`。
- 新增查询 SQL 结果函数。
- 新增租户暂停使用功能。
- 多租户新增集群表功能（管理员写数据写给某个租户）。
- 新增聚合函数 `group_concat`。
- 新增 `format`, `replace`, `curdate`, `field`, `substring_index` 等系统函数。

仍存在的已知问题

- 0.7.0 版本的存储数据格式与以前版本不兼容，无法直接升级。
- OLTP 类负载并发性能不稳定。
- 次级索引加速功能尚未实现。
- 100GB 规模的 TPCH Benchmark 测试不稳定。
- 长时间频繁压力测试会导致内存溢出等问题。
- 分布式集群中多个 GB 级大表加载会导致 OOM 问题。

贡献者们

截止本次发布，共有 43 位贡献者为 MatrixOne 的开发作出贡献，共产生了 656 次提交记录。

欢迎新加入的贡献者

- @sourcelliu
- @iceTTTT
- @chrisxu333

- @songjiayang
- @dr-lab
- @arjunsk
- @Morranto

我们感谢您的贡献，欢迎来到 MatrixOne 社区！

更详细的更新日志

https://github.com/matrixorigin/matrixone/compare/v0.6.0...v0.7.0

MatrixOne v0.6.0 发布报告

热烈祝贺 MatrixOne 的 v0.6.0 版本于 2022 年 11 月 29 日正式发布！在这个版本中，MatrixOne 已升级为存算分离、读写分离、冷热分离、事务和分析能力分离、并拥有极致扩展能力和完整功能的云原生 HTAP 数据库。干杯！

Docker

```
docker pull matrixorigin/matrixone:0.6.0
```

最新特性

- 支持具有快照隔离级别的分布式 ACID 事务。
- 支持 `TEXT`、`BLOB`、`TIME`、`JSON` 数据类型。
- 支持权限管控相关的 `CREATE/GRANT/SET ROLE/REVOKE` 等操作。
- 支持 `VIEW`。
- 支持 Java、Python、Golang 连接器与 Mybatis、Spring JPA、SQLAlchemy ORM 连接。
- 支持从本地文件系统和云端 S3 存储导入 `CSV` 和 `JSON` 数据。
- 实施了 MatrixOne 专用备份工具 `modump`。
- 支持复合主键、唯一键和自增约束。
- 在目录中添加 `system_metrics` 数据库以监控实例状态。
- 目录中增加 `system` 数据库，用于记录用户语句和系统日志。

- 增加对时区 `timezone` 的支持。
- 支持 TLS 加密传输。
- 支持预编译语句 `PREPARE`、`EXECUTE`、`DEALLOCATE`。
- 支持 `Explain Analyze` 详细查询计划分析。
- 支持 `UNION`、`UNION ALL`、`INTERSECT`、`MINUS` 运算符。
- 支持 `CREATE TEMPORARY TABLE` 临时表。
- 支持 `CREATE EXTERNAL TABLE` 外部表。
- 增加了分区 `Partition by` 能力。
- 增加了大量系统配置参数，函数及系统表以保持对 MySQL 的兼容。
- 增加了新的 `JSON` 以及大量 `Datetime` 类型函数。
- 增加 `UUID` 函数。
- 支持位操作符 `&, |, ^, ~, <<, >>`。
- 支持 `DDL` 中带 `comment` 功能。
- 支持 `SET` 自定义变量功能。

仍存在的已知问题

- 0.6 版本的存储数据格式与以前版本不兼容，无法直接升级。
- 在点查与并发等 OLTP 性能上与主流数据库仍有差距。

- 复合主键，唯一键，次级索引无法起到加速查询作用。 #6028
- Insert/Update into select 超过 100MB 数据在分布式环境下会失败。 #6780
- 持续循环 Load 数据会出现内存不足问题。 #6793
- 分布式环境中会出现 Data Race 问题。 #6855, #6926
- 后台任务会有较低概率与用户正在执行的事务产生读写冲突，从而终止用户事务。#6049
- 写入二进制文件到 `BLOB` 中会出现失败。#6302

贡献者们

截止本次发布，共有 97 位贡献者为 MatrixOne 的开发作出贡献，共产生了 1520 次提交记录。

欢迎新加入的贡献者

- @lokax
- @triump2020
- @Abirdcfly
- @yjw1268
- @Juneezee
- @ZoranPandovski
- @Toms1999
- @xy2398437254
- @goodMan-code
- @DanielZhangQD
- @taofengliu
- @TszKitLo40
- @TheR1sing3un
- @qqIsAProgrammer

我们感谢您的贡献，欢迎来到 MatrixOne 社区！

更详细的更新日志

https://github.com/matrixorigin/matrixone/compare/v0.5.1...v0.6.0

MatrixOne v0.5.1 发布报告

热烈祝贺 MatrixOne 的 v0.5.1 版本于 2022 年 8 月 19 日正式发布！在这个版本中，MatrixOne 解决了一些日志回放和存储垃圾收集（GC，Garbage Collection）问题。

功能优化

- 优化了回放检查点。 #4214
- 修复了 block 和 index 文件引用计数不释放的缺陷问题。 #4052
- 修复了日志未满时重放死循环的缺陷问题。 #4051

修复这些错误后，MatrixOne 实例稳定性得以提升。

更详细的更新日志

https://github.com/matrixorigin/matrixone/compare/v0.4.0...v0.5.1

MatrixOne v0.5.0 发布报告

热烈祝贺 MatrixOne 的 v0.5.0 版本于 2022 年 7 月 18 日正式发布！在这个版本中，MatrixOne 拥有一个独立的列式存储引擎，可以支持 HTAP 工作负载。干杯！

Docker

可以使用 docker 拉取 MatrixOne 0.5.0 版本。

```
docker pull matrixorigin/matrixone:0.5.0
```

最新特性

- 支持带有快照隔离级别的 ACID 事务。
- 支持 `UPDATE`， `DELETE` 和 `INSERT INTO ... SELECT ...` 语法。 — 支持 `BOOL` 和 `Timestamp` 数据类型。
- 支持 `LEFT/RIGHT/OUTER/NATURAL JOIN`
- 支持 `Having` 表达。
- 支持子查询。
- 支持公共表表达式。
- 支持 `CASE ... WHEN` 表达式。
- 支持 `Interva` 表达式。
- 支持 `Explain` 计划树。
- 支持新增聚合函数 `Any_value`。

- 新增其他大量功能。

仍存在的已知问题

- 0.5.0-Hotfix 可能导致部分数据格式不兼容。
- 当你频繁进行插入，更新，删除数据，创建或删除表操作时，可能会出现 *Too many open files* 错误。你需要增加 *max open files* 来解决这个问题。
- 在内存小于 64GB 的情况下运行 1GB 的 TPCH 基准测试可能会导致内存不足，产生报错。
- 加载大于 100GB 的*. CSV* 文件可能会导致系统挂起。 #3858
- TP 和 AP 两种工作负载长时间混合运行，可能导致系统内核错误。 #3947 #3961

贡献者们

截止本次发布，共有 73 位贡献者为 MatrixOne 的开发作出贡献，共产生了 811 次提交记录。

欢迎新加入的贡献者

- @lawrshen
- @lyfer233
- @wuliuqii
- @ericsyh
- @dongdongyang33
- @aylei
- @richelleguice
- @aressu1985
- @mklzl

我们感谢您的贡献，欢迎来到 MatrixOne 社区！

更详细的更新日志

https://github.com/matrixorigin/matrixone/compare/v0.4.0...v0.5.0

MatrixOne v0.4.0 发布报告

热烈祝贺 MatrixOne 的 v0.4.0 版本于 2022 年 5 月 5 日正式发布！以下我们对版本最新的更新内容进行简要说明。

Docker

可以使用 docker 拉取 MatrixOne0.4.0 版本。

```
docker pull matrixorigin/matrixone:0.4.0
```

最新特性

- 支持等式运算符 Inner Join
- 支持 From 子查询
- 新增 decimal 类型
- 新增以下系统函数：
  - 数学类：Abs, Log, Ln, Ceil, Exp, Power, Pi, Sin, Sinh, Cos, ACos, Tan, ATan, Cot
  - 时间日期类：Month, Weekday, Date, DayOfYear
  - 字符串类：Space, Reverse, Substring, Ltrim, Rtrim, StartsWith, EndsWith, Lpad, Rpad, Empty, LengthUTF8
- 新增以下聚合函数：
  - Bit_and, Bit_or, Bit_xor, Stddev_pop, Var

仍存在的已知问题

- 当在脚本中进行一连续的建表操作时，集群中的 `pre-allocate-group-num` 参数应该设置为更大的值。否则，建表过程中会报错 “no available raft group”。

贡献者们

截止本次发布，共有 50 位贡献者为 MatrixOne 的开发作出贡献，共产生了 253 次提交记录。

欢迎新加入的贡献者

- @BePPPower
- @JackTan25
- @Charlie17Li
- @domingozhang
- @Fungx
- @JasonPeng1310
- @jiajunhuang
- @NTH19
- @noneback
- @RinChanNOWWW
- @chaixuqing
- @Y7n05h
- @yuxubinchen
- @adlternative
- @ajian2002
- @bxiiiiii
- @coderzc
- @e11jah
- @fengttt
- @florashi181
- @hiyoyolumi
- @jinfuchiang
- @ouyuanning
- @qingxinhome

- @supermario1990
- @whileskies
- @xiw5
- @yclchuxue
- @ZtXavier

我们感谢您的贡献，欢迎来到 MatrixOne 社区！

更详细的更新日志

https://github.com/matrixorigin/matrixone/compare/v0.2.0...v0.3.0

MatrixOne v0.3.0 发布报告

热烈祝贺 MatrixOne 的 v0.3.0 版本于 2022 年 3 月 10 日正式发布！以下我们将对版本最新的更新内容进行陈列。

Docker

可以使用 docker 拉取 MatrixOne0.3.0 版本。

最新特性

- 将数据导出至 CSV 文件
- 引入了并行执行，提升了查询速度
- 引入了 `IN` 运算符，在 `WHERE` 子句中指定多个值
- 在 `GROUP BY` 语句中支持 `NULLABLE` 列
- 引入了函数 round() 与 floor()
- 在分布式系统中引入了 `Chaos` 测试框架

已发现的问题

- 当在脚本中进行一连续的建表操作时，集群中的 `pre-allocate-group-num` 参数应该设置为更大的值。否则，过程中会报错 “no available raft group”。

贡献者们

截止本次发布，共有 21 位贡献者为 MatrixOne 的开发作出贡献，共产生了 157 次提交记录。

- broccoliSpicy (@broccoliSpicy)

- Chen Mingsong (@m-schen)
- hanfang (@aptend)
- O2 (@ikenchina)
- Jin Hai (@JinHai-CN)
- Jiang xinmeng (@jiangxinmeng1)
- Lin Junhong (@iamlinjunhong)
- Long Ran (@aunjgr)
- Nan Deng (@dengn)
- Otter (@WenhaoKong2001)
- Peng Zhen (@daviszhen)
- Qin Shuqi (@sukki37)
- Sundy Li (@sundy-li)
- Shen JiangWei (@LeftHandCold)
- Jian Wang (@jianwan0214)
- Wan Hanbo (@wanhanbo)
- Xu Peng (@XuPeng-SH)
- Yan Wenze (@nnsgmsone)
- Yuesheng Li (@reusee)
- Zilong Zhou (@zzl200012)
- Zhang Yingfeng (@yingfeng)

欢迎新人

- @wanhanbo 首次贡献于 https://github.com/matrixorigin/matrixone/pull/1600
- @ikenchina 首次贡献于 https://github.com/matrixorigin/matrixone/pull/1685
- @sundy-li 首次贡献于 https://github.com/matrixorigin/matrixone/pull/1704
- @WenhaoKong2001 首次贡献于 https://github.com/matrixorigin/matrixone/pull/1838

我们期待您的贡献，欢迎来到 MatrixOne 社区！

更详细的更新日志

https://github.com/matrixorigin/matrixone/compare/v0.2.0...v0.3.0

MatrixOne v0.2.0 发布报告

热烈祝贺 MatrixOne 的 v0.2.0 版本于 2022 年 1 月 6 日正式发布！以下我们将对版本最新的更新内容进行陈列。

Docker

可以使用 docker 拉取 MatrixOne0.2.0 版本。

最新特性

- 在 AOE 列存引擎中支持 **automatic rebalancing**
- 引入全新的 SQL parser
- 引入 SQL 的因子化执行，实现了 GO 编写的最快 MPP
- 支持 CREATE / DROP INDEX 语句
- 在建表时支持创建 PRIMARY KEY
- 为 SQL 的二进制操作符提供更多数据类型
- 支持在 `group by` 或聚合语句中使用 equi join
- 支持新数据类型 DATE 与 DATETIME

已发现的问题

- 当在脚本中进行一连续的建表操作时，集群中的 `pre-allocate-group-num` 参数应该设置为更大的值。否则，过程中会报错 “no available raft group”。

贡献者们

截止本次发布，共有 21 位贡献者为 MatrixOne 的开发作出贡献，共产生了 243 次提交记录。

- BingLin Chang (@decster)
- Chen Mingsong (@m-schen)
- Nan Deng (@dengn)
- Jin Hai (@JinHai-CN)
- Jiang xinmeng (@jiangxinmeng1)
- Li Yang (@lignay)
- Lin Junhong (@iamlinjunhong)
- lni (@lni)
- Long Ran (@aunjgr)
- Peng Zhen (@daviszhen)
- Qin Shuqi (@sukki37)
- Shen JiangWei (@LeftHandCold)
- Jian Wang (@jianwan0214)
- broccoliSpicy (@broccoliSpicy)
- Ryan Wang (@wanglei4687)
- Xiong Jingjuan (@anitajjx)
- Xu Peng (@XuPeng-SH)
- Yan Wenze (@nnsgmsone)
- Yuesheng Li (@reusee)
- Zilong Zhou (@zzl200012)
- Zhang Yingfeng (@yingfeng)

MatrixOne v0.1.0 Release Notes

热烈祝贺 MatrixOne 的 v0.1.0 版本于 2021 年 10 月 24 日正式发布！

Docker

可以使用 docker 拉取 MatrixOne 0.1.0 版本。

核心特性

SQL

本次发布的版本支持一下 SQL 语句：

DDL

- CREATE / DROP DATABASE
- CREATE / DROP TABLE

DML

- INSERT
- LOAD DATA
- SELECT
  - WHERE
  - GROUP BY
  - ORDER BY

    - LIMIT, OFFSET
- SHOW
    - DATABASES
    - TABLES
- USE

数据类型

- TINYINT / SMALLINT / INT / BIGINT, SIGNED / UNSIGNED
- FLOAT / DOUBLE
- CHAR / VARCHAR

运算符

- <, >=, <, <=, <>, !=, =
- BETWEEN ... AND ..., NOT BETWEEN ... AND ...
- AND, &&, OR, ||
- +, -, *, /, %, MOD, DIV, NEG
- CAST

聚合函数

- COUNT
- SUM
- AVG
- MAX
- MIN

数据引擎

- 支持 MySQL 语法
- 内置的 AOE 列存引擎可作为融合性数据库引擎

- 支持实时分析查询
- 提供了基于 MPP 的向量化查询引擎
- 通过 SIMD 指令实现了部分执行向量化
- 实现了在 RAFT 协议下的强一致性分布式 AOE 引擎
- 通过对 RAFT 日志和 Write-Ahead 日志的复用设计，使复制状态机的性能得到了极大的提高

贡献者们

截止本次发布，共有 16 位贡献者为 MatrixOne 的开发作出贡献，共产生了 453 次提交记录。此外，我们格外感谢首次为 MatrixOne 作出贡献的开发者！

Yan Wenze (@nnsgmsone)
Chen Mingsong (@m-schen)
Jin Hai (@JinHai-CN)
Jiang xinmeng (@jiangxinmeng1)
Li Yang (@lignay)
Lin Junhong (@iamlinjunhong)
lni (@lni)
Long Ran (@aunjgr)
Peng Zhen (@daviszhen)
Qin Shuqi (@sukki37)
Shen JiangWei (@LeftHandCold)
Wei Ziran (@w-zr)
Xiong Jingjuan (@anitajjx)
Xu Peng (@XuPeng-SH)
Yan Wenze (@nnsgmsone)
Zilong Zhou (@zzl200012)
Zhang Yingfeng (@yingfeng)

术语表

术语

阅读以下对相关词汇的概念解释或许有助于你理解我们的整体架构。

术语	定义
A	
AST	AST 即抽象语法树，是代码的树结构表示形式，是组成编译器工作模式的基本部分
C	
Cluster	MatrixOne 的分布式部署形式，由多台主机组成，在逻辑上构成一个整体。
E	
Explicit Transactions	显式事务，即是一种指定的事务，这种事务需要由你自己决定哪批工作必须成功完成，否则所有部分都不完成。可以使用 `BEGIN TRANSACTION` 和 `ROLLBACK TRANSACTION` 或 `COMMIT TRANSACTION` 关键字进行控制。
I	
Implicit transactions	隐式事务，即自动提交事务。
S	
Snapshot Isolation (SI)	Snapshot Isolation 是一种在实践中广泛应用的多版本并发控制技术，MatrixOne 支持 Snapshot 隔离级别的分布式事务。

重要概念

概念	定义
A	
Auto-Rebalance	在分布式系统中，多个服务器的存储量、读写负载的自动平衡过程称之为 Auto-Rebalance。
C	
Consistency	MatrixOne 支持强一致性，保证了在成功写入数据后，无论在哪个 Store(节点) 上都能读取到最新的数据。
E	

概念	定义
Execution Plan	数据库中的执行计划是查询优化器生成的查询操作的图形表示，可以得到执行该操作的最高效方法
F	
Fault-Tolerance	Fault-Tolerance（容错性）意味着系统在其中一个或多个组件发生故障后仍然可以继续运行的能力。
M	
Monolitic Engine	Monolitic Engine 即超融合引擎，可支持 TP、AP、时序、机器学习等混合工作负载。
Materialized View	Materialized View 即物化视图，是预先被计算好的数据集，存储下来以便后续使用，通常可以提升查询的运行效率。
Metadata	Metadata 即元数据，是用于描述数据库中数据的结构信息的数据。
P	
Paxos	Paxos 是一种一致性算法，保持一组异步网络通信的分布式计算机之间的一致性。
R	
Raft	Raft 是一种易于理解的一致性协议算法，在容错性与性能上与 Paxos 相当。
Raft Group and Leader	Raft 在一组中定义了一个 leader 以及许多 followers。一个组代表一个复制状态机，只有 leader 才可以响应客户端请求，然后将传达给 followers。
S	
SIMD instruction	SIMD 是 Single Instruction/Multiple Data 的简写，即单指令多数据流，SIMD 操作一般指一种使用一条指令即可处理多条数据的计算方法。
T	
Transaction	在数据库中执行的一系列满足 ACID 基本要求的操作。
V	
Vectorized Execution	通过有效利用 CPU 的缓存，向量化执行提高了分析查询引擎的速度。Arrow 的列式格式可以使用轻量级的架构，如 dictionary encoding，bit packing 以及 run length encoding，这都进一步了提升了查询效率。

快速贡献

MatrixOne 社区欢迎所有开发者的加入和贡献！本章节旨在帮助您快速完成首次贡献。

如何贡献?

在哪些方面可以大展拳脚呢？详情请参见贡献种类。 如果您是完全的新手，您可以从以下两种类别中选择一种进行尝试，这些类别的问题对您技术背景的要求很少，所以不必担心！

- 报告代码中出现的 Bug
- 完善 MatrixiOne 文档

在开始处理问题之前，建议您先将找到的问题作为 GitHub 上的一项 Issue 提出。此外，我们准备了一系列带有 `good-first-issue` 标签的 Issue，它们包含了明晰的实现步骤和预期结果，您可以此作为突破口！

认领任务

当您提出 Issue 之后或者是在浏览 `good-first-issue` 后决定上手解决，您需要认领这次 Issue。在相应 Issue 的评论中输入如 “I'd like to work on this issue”，社区人员会将任务分配给您，此时，您可以在右侧的 Assignees 板块看见自己，接下来便可以正式着手解决问题。

前置准备

请确保您至少安装了单机版 MatrixOne 并部署了相关开发环境。具体请参考准备工作。

工作流程

步骤 1：Fork 项目仓库

首先前往 Github 上的 matrixorigin/matrixone 仓库。 在页面右上角处，点击 `Fork` 按键，创建主库的分叉，并作为您之后主要工作的仓库。

Fork our repo

步骤 2：将仓库克隆至本地

前往刚才您创建的 Fork 仓库，点击 `Code`，然后再点击**复制**图标，将库的网址复制到剪贴板。

Clone your fork

然后，在您本地挑选一个合适的工作目录，打开命令行输入以下 Git 命令将文件克隆至您本地的目录：

```
git clone \<content you just copied>
```

例如：

```
git clone git@github.com:\<yourname>/matrixone.git
```

`\<yourname>` 是您的 Github 账号名，换言之，您要用您自己的账号名替换它。

步骤 3：添加 matrixone 仓库作为本地的远程仓库

您可以将 matrixorigin/matrixone 添加为本地的远程仓库，以便后续步骤进行操作：

```
git remote add upstream https://github.com/matrixorigin/matrixone.git
```

其中，`upstream` 是该远程仓库的名字，您可以自行替换，但注意后续对该词也应该一并替换。

步骤 4：修改、试运行

修改

克隆代码库之后您就可以开始您的开发过程，可以在您所需要的地方进行任何修改。

试运行

如果您在修改完成后想知道改动是否有效，能否解决最初的问题，或者是否影响程序运行，您可以运行 MatrixOne 进行简单的预览和测试。 当然，请确保您已经按照教程完成了[单机部署 MatrixOne]。

步骤 5：提交修改

当完成以上修改和测试后，您就可以开始提交修改。首先使用将您所更改的文件添加至 git 所管理的目录中：

```
git add \<filename>
```

`\<filename>` 是您所修改的文件的名称。 您也可以使用如下命令，直接将当前文件夹中的所有文件都添加至管理目录：

```
git add .
```

之后，您可以提交您的修改

```
git commit -m "\<commit message>" -s
```

`\<commit message>` 是您对本次修改的简单总结和描述，试着做到简明扼要。

步骤 6：代码推送

提交修改后，您需要将本地的提交推送至远程仓库———我们强烈推荐您推送至目标仓库的一个新分支：

```
git push origin main:NEW_BRANCH
```

`NEW_BRANCH` 您创建并推送至的新分支名，您也可以随意替换它，但也记得后续一并修改。

步骤 7：创建 PR

推送后，可以在您所 Fork 的仓库中看到相关提示信息，点击 **Compare & Pull Request** 按键来创建一个 PR（该 PR 应该是从个人仓库的 `NEW_BRANCH` 分支到主库的 `main` 分支）。

注意

建议按照 PR 中所给出的模板要求，撰写相关信息，如此可以准确表达您的问题以及所做的修改，从而提高审查效率。

Pull Request

Pull Request Template

您的 PR 一旦创建，就有社区资深的开发人员审核您的代码，他将会检查您做的修改并且进行回复，请及时与您沟通，然后按照要求进行修改。

步骤 8：同步代码库

当进行到这一步时，恭喜您的修改已经被接受并且 Merge 进入项目中，感谢您做出的贡献！但工作还没有结束，还有一些工作要做（这些工作有助于保证提交记录的干净，有利于项目进一步发展）。 可以将远端仓库的代码拉取下来覆盖本地，以保持同步：

```
git pull --force upstream main:main
```

最后，再将代码推送到您的仓库中的 `main` 分支，这样就能保证三个仓库代码均一致。

```
git push --force origin main:main
```

贡献类型

对 MatrixOne 的贡献绝不仅限于代码。以下为您展示了参与 MatrixOne 项目并参与我们的开源社区的各种方式：

代码贡献

类型	说明
报告 Bug	您可以在 GitHub 上提出 Issue 来报告您在运行或开发 MatrixOne 时所遇见的 Bug。
提交代码	同样，您也可以修复已有的 Bug 或者解决 GitHub 上提出的关键性问题，也可以提出新的功能需求以及设计构思。请参见代码贡献来了解具体的规范与步骤。
审查	您也可以以审查者的身份对 MatrixOne 中的 PR 进行评论、审查，为他人提供建议和帮助。

完善 MatrixOne 文档

类型	说明
报告或修复技术文档的问题	您可以点击文档网站上的 **Edit the page** 按键，可以直接在当前页面进行修改并提交英文文档 PR 或提交中文文档 PR。当然您可以可以在 GitHub 的[Issue]中提出问题，并描述相关信息： 英文文档 Issue 中文文档 Issue
提交一份新的文档	在撰写一份完整的文档之前，请阅读文档贡献指南。

其他

类型	说明
分享用例和解决方案	向用户展示如何在特定场景下使用 MatrixOne，可以将您的用例发送到 dengnan@matrixorigin.io，我们将在社区中发布。
撰写博文、故事墙、白皮书	撰写高质量的文章，清晰准确地解释有关 MatrixOne 的技术细节。

类型	说明
做一场技术演讲或现场直播	发表演讲或现场直播可以帮助提高各界朋友对 MatrixOne 的认识。
互相支持	帮助 GitHub 上的用户解决使用 MatrixOne 时遇到的问题。
Issue 分类	在 GitHub Issues 中为各个 Issue 添加合适的标签，可以很快地把问题分配给对应负责人，提高问题的解决效率。这是对 MatrixOne 工作开发流程的巨大贡献。
其他任何贡献	我们欢迎任何对 MatrixOne 及其开源社区的发展起到积极作用的贡献。

准备工作

非常欢迎您参与到 MatrixOne（以下简称 MO）项目的建设中来！无论你是初识 MatrixOne，还是已经迫切地想参与到开发工作中来，亦或是在阅读文档、使用产品的过程中发现了一些问题，都欢迎你提出意见和建议，与我们共同打造更加强大、稳定的 MatrixOne！ 当然，在您向 MatrixOne 项目提出改进之前，我们需要提前说明一些基本规范与流程，以提高整个贡献过程的质量与流畅性，同时也能保障 MO 的稳定性与安全性。 此外，在本章中，我们为尚未熟知 MO 的贡献者们提供一些了解我们的渠道，希望能为你们带来一些便利！

了解 MatrixOne

特点与框架

在 MatrixOne 简介中您可以了解到 MO 的超融合、云边协同特性以及其所创造的优异表现与独特价值。 此外，在 MatrixOne 框架中您可以详细地了解 MatrixOne 的整体架构，以及存储层、日志层等具体组成情况。 同时，你也可以查阅 MatrixOne 术语表来了解一些复杂的词汇。 在技术层面，SQL 参考指南为您提供了详细的 SQL 语言的参考，其中对语法和示例都有详细解释；同样，自定义函数提供了 MO 中自定义函数的相关解释。

建设情况

目前，MatrixOne v0.7.0 已经发布了，您可以通过版本发布指南来了解最新的发布信息，其中包含了最新的修改与优化。 同时，我们当前正在开发 v0.8.0 版本，对应的工作任务在 GitHub 的 Milestone 0.8.0 中列出。 关于长期的项目规划，请参阅 MatrixOne Roadmap。

你可以做些什么?

对 MatrixOne 的贡献可分为以下几类：

- 报告代码中的 Bug 或文档中的谬误。请在 GitHub 上提出 Issue，并提供问题的详细信息。请记得选取合适的 Issue 模板，并打上标签。
- 提议新的功能。请在 Feature Request 中描述详情并与社区中的开发人员商议。一旦我们的团队认可了您的计划，您就可以按照工作流程进行具体开发。
- 实现某个功能或修复既有问题，请按照工作流程完成开发。如果你需要关于某一特定问题的更多背景信息，请就该问题发表评论。

工作目录与文件介绍

我们将为 *Github* 上 *matrixorigin/matrixone* 的项目目录及其中关键文件进行简单介绍，以期为您的深入了解和开发提供指导性帮助。 matrixone 是 MatrixOne 代码所在的主库，我们将介绍其中的项目目录以及关键文件，以期为您的深入了解和开发提供指导性帮助。

目录	内容
/LICENSES	相关依赖库的许可
/cmd	Go 的可执行文件的 binary entry
optools	测试与部署工具
pkg	MatrixOne 项目的主要代码库

对于不同的技术模块，`/pkg` 喜爱的代码结构如下表所示。

目录	模块
frontend/	SQL 前端
sql/	MPP SQL Execution
sql/parser	SQL 解析
sql/vectorize	SQL 的向量化执行 \
catalog/	存储元数据的 Catalog
vm/engine	存储引擎
vm/engine/aoe	AOE 引擎（分析优化引擎）\
vm/engine/tpe	TPE 引擎（事务处理引擎） \
buildin/	系统的内置函数 |

在文档方面，matrixorigin.io、matrixorigin.io.cn 与 artwork 都是在贡献过程中可能使用的库，详情参见文档贡献。

目录	内容
matrixone/docs/rfcs	MatrixOne 项目的设计文档
matrixorigin.io/docs/MatrixOne	英文文档网站的具体内容文件（.md 文件）

目录	内容
matrixorigin.io.cn/docs/MatrixOne	中文文档网站的具体内容文件（.md 文件）
matrixorigin.io/mkdocs.yml	英文文档网站的配置文件
matrixorigin.io.cn/mkdocs.yml	中文文档网站的配置文件
artwork/docs	文档官网出现的图片和图表

开发环境

MO 主要由 Go 语言编写，因此需要提前安装部署好相关的开发环境，简要的示例流程如下：

1. 安装版本为 1.19 的 Go，您可以通过 Download Go 与 Installation instructions 教程来完成整个过程。
2. 定义环境变量并修改路径，您可以遵循以下示例流程：

```
export GOPATH=$HOME/go
export PATH=$PATH:$GOPATH/bin
```

提示

MatrixOne 使用 `Go Modules` 来管理相关依赖。

若您需要补充 Go 语言的相关知识，可以通过 How to Write Go Code 进行了解。

此外，确保您至少已经安装了单机版本的 MatrixOne，具体过程可参照 Install Standalone MatrixOne。

Github & Git

为更好地开发建设 MatrixOne，我们采取了开源运营的方式，通过 Github 为项目维护人员和其他开发者提供了一个协作平台。因此，如果您想参与到 MO 的开发中来，我们强烈建议您采取 Github 的渠道。
若您还未使用过 Github 或缺少相关开发经验，您首先需要熟悉 **GitHub** 上的相关操作，并学习基本的 **git** 命令。
如果您没有 Github 帐户，请在 https://github.com 上完成注册。

如果你没有 SSH 密钥，你可以按照 GitHub 上关于 SSH 的教程来生成、添加密钥。
更多详情请参见 Github Docs。

此外，我们建议您学习并使用 **git** 命令来完成 Github 上的各种流程，因为我们提供的相关工作流程大多通过 **git** 命令完成，这有助于您提高效率。
您可通过 Install git 安装 Git 环境。
并且可以通过以下教程来学习如何使用：

- 简易版
- 详细版

提出问题

您在使用或开发 MatrixOne 过程中遇见的任何问题都能以 Issues 的形式提出来，我们也鼓励您按照我们设定的模板和标签对 Issues 进行详细描述和分类，以便更高效地解决问题。
本节旨在介绍提出问题时需要遵循的模板、标签和注意事项。

避免重复问题

在提出 Issue 之前，尽量先确认其他人是否已经提出过相同或类似的问题，避免重复，您可以使用 search bar 工具帮助您筛选查找。

模板

针对不同种类的问题，MatrixOne 使用了不同的模板对其内容进行刻画，其中大多描述了问题的关键信息，有助于审查者与其他开发者理解并参与其中。

例如，`Bug report` 问题模板包含以下信息：

- **Detail Environment**
  Describe the details about the operating environment including version, hardware parameters, OS type and so on.
- **Steps to Reproduce**
  List steps to reproduce what you encountered.
- **Expected & Actual Behavior**
  Describe what's the observed and your expected behavior respectively.

`Enhancement` 问题模板包含以下信息：

- **What would you like to be added**
  A concise description of what you're expecting/suggesting.
- **Why is this needed**
  A concise description of the reason/motivation.
- **Anything else**
  Anything that will give us more dectail about your issue!

`Feature Request` 问题模板包含以下信息：

- **Is your feature request related to a problem?**
  A clear and concise description of what the problem is and state your reasons why you need this feature.
- **Describe the feature you'd like:**
  A clear and concise description of what you want to happen.
- **Describe alternatives you've considered:**
  A clear and concise description of any alternative solutions or features you've considered.
- **Teachability, Documentation, Adoption, Migration Strategy:**
  If you can, explain some scenarios how users might use this, situations it would be helpful in. Any API designs, mockups, or diagrams are also helpful.

`Performance Question` 问题模板包含以下信息：

- **Detail Environment**
  Describe the details about the operation environment including version, hardware parameters, OS type and so on.
- **Steps to Reproduce**
  List steps detailedly to reproduce the operations to test performance.
- **Expected & Actual Performance**
  Describe what's the observed and your expected performance respectively.
- **Additional context**
  Add any other context about the problem here. For example:
  - Have you compared TiDB with other databases? If yes, what's their difference?

`Documentation Issue` 问题模板包含以下信息：

- **Describe the issue**
  A clear and concise description of what's wrong in documentation.
- **Additional context**
  Add any other context about the problem here.

`Refactoring Request` 问题模板包含以下信息：

- **Is your refactoring request related to a problem?**
  A clear and concise description of what the problem is.

- **Describe the solution you'd like**
  A clear and concise description of the refactoring you want to.
- **Describe alternatives you've considered**
  A clear and concise description of any alternative solutions or refactoring method you've considered.
- **Additional context**
  Add any other context or screenshots about the refactoring request here.

标签

除了描述问题的详细信息外，您还可以根据问题所属的组件以及问题所属版本为其添加适当的标签。当您的 Issue 提交之后，会自动打上 `needs-triage` 的标签，之后项目维护者会详细阅读您的 Issue，然后为之打上合适的标签并分配给合适的开发者。
如果您想自己亲手处理这个问题，可以在评论中提出，社区管理员会给您分配这个 issue。若在 Assignees 部分可以看见您自己，说明操作成功。

Good First Issues

当您首次参与贡献时，您可以选择 `good-first-issue` 标签下的问题着手解决，其下的每个问题都是相对来说容易解决的。
详情请阅读快速上手章节。

代码贡献

MatrixOne 是一个由项目管理者、社区开发者共同维护、改进和扩展的开源项目。 本文档主要描述了开发的准则与一般流程，并提供了在编写代码、提交 PR 过程中需要使用的样式和模板。如果您在参与 MatrixOne 的贡献过程中遇到任何问题或发现一些错误，请在 Github 上提出 issue 或通其他平台联系我们。

前置准备

在正式开发之前，请确保您已经阅读了准备工作，已经对 MatrixOne 的核心理念、基础架构有一定了解，并准备好了开发所需要的相应环境、语言、工具等。

风格规范指南

在对 MatrixOne 进行开发和完善时，应该使代码、代码注释、提交信息 (Committing Message) 和拉取请求 (Pull Request，简称 PR) 保持一致的风格。当您提交 PR 时，我们强烈建议您确保所作出的修改符合我们的一贯风格，这不仅会提高 PR 的通过率，并且也能使 MatrixOne 易于审查、维护和进一步开发。

- **代码规范**

MatrixOne 采用了 Golang 社区建议的编码规范，详情请见 Effective Go。

- **代码注释规范**

关于代码注释，请参考代码注释规范。

- **提交信息 & PR 规范**

可参考 Commit & PR 规范。

一般工作流程

您可以按照以下工作流程来进行开发并在 Github 上提交修改，如果您还需要更加详细的解释，可以查看 Make Your First Contribution

1. 在 Github 上 Fork matrixorigin/matrixone 仓库。

2. 将 Fork 的仓库克隆至本地：

```
git clone git@github.com:<yourname>/matrixone.git
```

并且把 matrixone 仓库添加为远程仓库：

```
git remote add upstream https://github.com/matrixorigin/matrixone.git
```

3. 创建一个新的分支，分支名自定义：

```
git checkout -b topic-branch
```

4. 在本地进行开发，完成相关修改，并完成必要的单元测试，最后进行提交。

5. 将修改推送至仓库的一个新分支：

```
git push origin main:NEW_BRANCH
```

6. 在仓库中的新分支 `NEW_BRANCH` 中创建 Pull Request，并添加相应标签、建立与相关 issue 的关联。

7. PR 通过后，覆盖本地提交历史：

```
git pull --force upstream main:main
```

8. 更新您的仓库的 `main` 分支：

```
git push --force origin main:main
```

如果您仍然有一些困惑，可以参考 GitHub 官方文档寻求帮助；若您发现我们提供的工作流程有错误或想要提出改善的方法，欢迎您提出建议！

代码审阅

当您创建 PR 请求时，您可以指定一些审阅者，或者留空。并且您可以添加一些相关的标签，这样更容易识别 PR 的类型、优先级等。在代码审阅期间，审阅者会对您的代码片段给出意见，您可以相应地在本地修改您的分支上的代码，提交更改，然后推送到 GitHub，新的提交会自动附加到 PR 上。

审阅与评论

对 MatrixOne 来说，对 PR 的审阅和评论是至关重要的：您可以对他人的 PR 进行分类，以便有专家更快的来解决这些问题；您也可以对代码的内容进行审阅，对代码编写的风格、规范等提出建议；哪怕是在评论区留下一个小小的 Idea，这也弥足珍贵。 所有别再犹豫，别再担心您的想法不够完善，无论多么微小的建议都可能会对 MatrixOne 产生深远影响。

当然，在此之前，我们希望您能认真阅读本节，了解基本要求和相关方法。

基本原则

当对一个 PR 进行评论或者审阅时，无论内容如何，我们呼吁所有参与者都保持友好和善的态度，营造一个和谐的社区氛围。

- **尊重他人**
  尊重每一个 PR 请求的发起人和其他审阅者。代码审阅是社区活动的重要部分，因此请遵循社区要求。

- **注意语气** 在与他人沟通交流时，我们鼓励您多使用 “建议” 或者 “提问” 的语气，而不要总是命令他人。换位思考，所有人都希望被温柔以待！

- **赞美**
  不要吝啬对他人的赞美！一个好的想法或者好的成果值得我们夸赞。在很多情况下，鼓励、赞美他人往往比不留情面的批评更有价值！

此外，在具体内容上我们有如下建议：

- **详细而具体**
  我们希望您能在评论中提供更为详细而具体的信息，因为信息越详细，他人就更容易理解，也可以更高效地解决问题。如果您进行了测试，完全可以放上测试环境和结果；如果您提出了一些建议，那不妨说说应该如何落实。

- **客观而公正**
  请避免个人偏见和主观情绪。诚然，每个人的评论或多或少都会带有主观色彩，但是，作为一个成熟的审阅者，您应该注重技术和数据，而不是个人的喜好。

- **灵活而审慎**
  当遇见一些复杂问题时，即使是在综合考量多方因素之后也很难抉择，“接受” 还是 “拒

绝"，是个两难的问题———我们也无法给出一个明确、具体的标准，只能说 "具体情况，具体分析"。但是我们建议您在难以抉择时，寻求他人的帮助。

对 PR 分类

有些 PR 创建者可能并不熟悉 MatrixOne 或相关开发工作流程，因此不确定应该添加何种标签，也不了解应该把问题分配给谁。如果您知道该如何做，我们希望您可以向他们施以援手，为问题补充上标签等信息，这有助于推动问题的解决进程。

检查正误

当您检查代码或其他修改的正误时，务必注意以下几点：

- **聚焦**
  无论处理的问题有多小，都应该聚焦，因此需要注意的是这项 PR 是否有专注于解决一件事情，是否混淆了其他模块的内容。
- **测试**
  代码贡献应该确保代码通过测试（单元测试、集成测试等），并提交一份测试报告，展示所使用的测试用例以及代码覆盖率。
- **完善性**
  审阅代码的完善性主要是关注其是否真的解决了当时提出的问题，与最初的目的是否契合，您可以查看 GitHub Issue 来追溯整个问题的前因后果与开发走向。
- **风格规范**
  PR 中的代码应该遵相应的规范。现有的部分代码可能与风格指南的要求不一致，您应该维护其一致性，或直接提交一个新的 Issue 来更正它。
- **必要的文档**
  如果一个 PR 改变了用户构建、测试、交互或发布代码的方式，您必须检查它是否也更新了相关文档，如 `README.md` 。类似地，如果一个 PR 删除或弃用了一段代码，您必须检查相应的文档是否也应该被删除。
- **性能**
  如果您发现新增的代码可能会影响 MatrixOne 的性能，您可以要求开发者提供一个基准测试（如文档中展示的 SSB、TPCH 测试）。

文档贡献指南

欢迎对 MatrixOne 文档的提出贡献。MatrixOne 社区一直在努力简化整个贡献流程，为此，我们创建本节来一步步地指导您完成文档贡献的过程。

准备工作

开始之前请尽量熟悉基本的 Markdown 语法并阅读行为守则和谷歌开发者文档风格指南，以便您写出更高质量的文档。

文档管理逻辑

MatrixOne 文档通过三个仓库来协调管理：

- matrixorigin.io 仓库包含英文文档的具体内容。（*.md* 文件）
- matrixorigin.io.cn 仓库包含中文文档的具体内容。（*.md* 文件）
- artwork 仓库包含了文档所用到的图像等非结构性文件。图像等非结构化文件则直接引用 `artwork` 仓库的网站链接，如：

```
https://github.com/matrixorigin/artwork/blob/main/docs/overview/overall-architecture.png?raw=true
```

`matrixorigin.io` 和 `matrixorigin.io.cn` 均部署了一个 CI 程序，当有新的代码被合并时将自动触发，将文档发布到我们的官方文档网站。
我们的文档是基于 mkdocs-material 组件进行开发的，您可以在此链接中了解更多信息。

文档内容架构

MatrixOne 文档内容可以分为如下几个模块：

- **Overview**: MatrixOne 的简介，包含了产品特点、架构、设计思路和技术细节。
- **Getting Started**: 介绍如何在单机环境中快速部署和运行 MatrixOne。
- **Developing Guide**: 介绍如何在单机或分布式环境中深度使用 MatrixOne。

- **Deploying**: 介绍如何在部署和运行 MatrixOne 集群。
- **Maintenance**: 介绍如何运维 MatrixOne，包括备份与恢复数据等。
- **Migrating**: 介绍如何将数据从其他数据库迁移至 MatrixOne。
- **Testing**: 介绍如何在使用测试工具完成自测，或者对 MatrixOne 进行性能测试。
- **Troubleshooting**: 介绍如何对 MatrixOne 进行故障诊断。
- **Tuning Performance**: 介绍如何在单机或分布式环境中对 MatrixOne 进行性能调优。
- **Privilege**: 介绍 MatrixOne 集群下多租户管理、账号生命周期管理、授权等。
- **Reference**: 包括 SQL 参考指南、配置参数设置、使用限制等。
- **FAQs**: 关于产品、技术设计、SQL、部署的常见疑难问题。
- **Release Notes**: 所有版本的发布说明。
- **Contribution Guide**: 介绍如何为 MatrixOne 项目做出贡献。
- **Glossary**: 名词释义表。

简易的修改

如果您发现了错别字或语法错误，可以点击本页面的 `Edit this Page` 按键直接进行修改。

一般工作流程

当您需要更改文档的具体内容但不涉及章节顺序、架构组织的调整时，需要对 `matrixorigin.io/tree/main/docs/MatrixOne` 或 `matrixorigin.io.cn/tree/main/docs/MatrixOne` 进行操作。

如果需要对章节顺序、架构组织进行调整时，需要对 `matrixorigin.io/blob/main/mkdocs.yml` 或 `matrixorigin.io.cn/blob/main/mkdocs.yml` 进行操作。

以下流程演示的是对二者均做修改的情况，实际情况可以根据您的需求进行简化。

1. 在 GitHub 上对英文文档提出 Issue 或对中文文档提出 Issue，简单介绍您发现的问题。并且在 Issue 下面评论认领该问题。

2. Fork matrixorigin.io 和 matrixorigin.io.cn 仓库。

3. 克隆 matrixorigin.io 和 matrixorigin.io.cn 仓库。

- 克隆 matrixorigin.io：

```
git clone git@github.com:yourusername/matrixorigin.io.git
```

- 克隆 matrixorigin.io.cn：

```
git clone git@github.com:yourusername/matrixorigin.io.cn.git
```

4. 在您的本地 *matrixorigin.io* 和 *matrixorigin.io.cn* 文件夹中将对应仓库添加为远程仓库。

- 在您的本地 *matrixorigin.io* 文件夹中添加 `matrixorigin.io` 为远程仓库：

```
git remote add upstream
https://github.com/matrixorigin/matrixorigin.io.git
```

- 在您的本地 *matrixorigin.io.cn* 文件夹中添加 `matrixorigin.io.cn` 为远程仓库：

```
git remote add upstream
https://github.com/matrixorigin/matrixorigin.io.cn.git
```

5. 本地的 *matrixorigin.io* 或 *matrixorigin.io.cn* 文件夹中将包含文档所需要的全部文件，因此您可以运行 `mkdocs serve` 命令，然后在 `http://localhost:8000` 网址中预览文档，检查整个项目文件是否可以正常运行，并且后续也可以检查您所做的修改是否正确。

```
mkdocs serve
```

6. 进行文档的修改和完善，如果您想对项目的设置进行改动，或者添加新的 page 来更新 sitemap，或更新 CI&CD 工作流代码，您也可以通过 `http://localhost:8000` 来查看您的修改是否有效。

7. 确认修改无误后，使用 `git add .` 和 `git commit` 命令在本地提交修改，并推送至您 Fork 的远程仓库 `matrixorigin.io` 与 `matrixorigin.io.cn`。
我们建议您推送至远程仓库的新分支：

```
git push origin main:NEW_BRANCH
```

8. 在 Github 上相应仓库的 `NEW_BRANCH` 分支提交 Pull Request。

9. 一旦您的修改通过，CI 工作流将开始运行并更新文档网站，这可能需要一些时间。

10. 最后，还有一些操作可以帮助保持您的远端仓库和本地仓库均保持一致。

覆盖本地提交历史：

```
git pull --force upstream main:main
```

更新 Github 上的 `main` 分支：

```
git push --force origin main:main
```

> ⚠ **注意**
>
> 若您在中英文两个仓库都做了修改，那么以上大部分操作都需要分别针对中英文两个仓库都执行一遍。

写一篇博文

如果您有意写一篇关于 MatrixOne 的博文，请在 GitHub 上提出 Issue，或者将您的想法发送到 dengnan@matrixorigin.io，无论是简单的 Idea 还是完整的草案，我们统统接受。我们会尽快审查所有内容，如果您的文章或想法很契合我们的博客，我们会直接联系您。

提交设计方案

前面章节提到了很多种类的修改，比如 Bug 修复、文档完善，这些都可以通过 GitHub 的 PR 工作流程来实现；但与此不同的是，如果您想要在 MatrixOne 中实现新的功能或增添新的组件，都不仅仅是一个 Idea 这么简单，我们鼓励您提出想法的同时还制定相应的设计方案，将其表达为技术设计文档。
因此，本节的目的正是引导您撰写一份技术设计文档，以期可以为这个新功能提供一个更加权威的、大众化的解释，各方人员可以更深入地了解这个模块的核心理念与发展方向。

准备工作

与其他工作一样，在开始之前，尽量做足准备，这样不仅可以提高您的工作效率，还可以增加方案通过的可能性；相反，一份粗糙而随意的设计文件可能会因为质量太差而吃到闭门羹。
我们鼓励您向有经验的开发人员寻求帮助，通过他们的建议您可以修正设计架构并完善技术细节。 您可以在前往 Github，提交一个 `Feature Request` 或 `Refactoring Request` 来向大家展示您的想法。

一般流程

通常，从头到尾地完成一项技术设计需要以下步骤：

- 在 GitHub 上提出 Issue，描述该功能想要解决的问题、目标、大致解决方案。
- 在得到管理者与其他开发者的回应，听取相关建议，然后对您的想法做进一步修改。
- 按照模板撰写技术设计文档并创建 PR。
- 与 Reviewers 交流，并按要求做相应的修改。
- 当至少有两个 Committer 达成一致且其他 Committer 没有异议时，设计文件即被接受，反之亦然。
- 如果提议被接受，那就请为该提议创建一个 tracking issue，或者将之前讨论的 issue 转换为 tracking issue，后续不断跟踪子任务和开发进度。
- 合并该设计的 Pull Request。
- 开始着手实现。

请及时查看子任务的 Tracking issue 来跟踪任务开发进程。

MatrixOne 行为守则

共同的承诺

我们作为项目与社区的贡献者和维护者，承诺在参与项目以及社区的过程中，致力于彼此帮助、共同成长，维护开源开放、和谐友善的氛围，无论年龄、体型、种族、性别、性取向、表达、经验、教育、社会经济地位、国籍、个人外貌、国家、宗教。

行为准则

我们提倡那些有助于营造美好社区环境的行为：

- 使用礼貌、包容性的语言
- 尊重不同的观点与立场
- 谦虚接受他人的建议和批评
- 专注打造更美好的社区
- 与其他参与者换位思考、相互理解

我们不提倡更不接受以下行为：

- 露骨、色情的言语、图像及其他信息
- 一系列侮辱、贬损他人的言论，人身攻击或政治敏感性内容
- 公开或私下骚扰
- 未经他人允许而发布他人的个人信息，如电子邮箱或居住地址
- 在专业领域内被普遍认为不恰当的言论

我们的责任

项目维护人员有责任明确值得提倡的行为规范，并对任何不当行为采取适当且公平的处理措施。

项目维护人员有权利也有责任删除、编辑或拒绝不符合上述行为准则的评论、提交、代码、wiki 编辑和 issue，并暂时或永久封禁发表不当言论或做出不当行为的贡献者。

适用范围

本行为守则适用于所有项目空间，以及任何代表项目、社区的公开发言。比如：使用官方的项目电子邮件地址，通过官方社交媒体账户发布信息，或作为指定代表参加线上线下活动。具体条件由项目管理者进一步明确。

具体实施

若您发现任何侮辱、骚扰或其他不合规定的行为都可以报告给项目团队：`hai.jin@matrixorigin.io`。我们将深入审核、调查所有的投诉并且做出及时、合理的反馈。此外，项目团队有义务保护举报人的隐私安全。
更加具体的执行政策将另行公布。

Attribution

本篇行为守则改编自贡献者公约，version 1.4，也可参见 https://www.contributor-covenant.org/version/1/4/code-of-conduct.html。

相关问题的回答可参见：https://www.contributor-covenant.org/faq

代码注释规范

本文描述了 MatrixOne 所使用的代码注释的规范和样式。当你提交代码时，请务必遵循已有的代码注释规范。

为什么注释很重要?

- 加快了代码审查流程
- 有助于维护代码
- 提高 API 文档的可读性
- 提高整个团队的开发效率

什么时候需要注释?

以下类型的代码时很有必要做出注释：

- 关键性代码
- 艰深晦涩的代码
- 复杂却有趣的代码
- 如果代码存在错误但您无法修复，或者只想暂时忽略
- 如果代码并非最优，但你现在没有更好的方法
- 提醒自己或其他人代码中存在缺失的功能或即将开发的功能

以下部分也需要进行注释：

- Package (Go)
- File
- Type
- Constant
- Function
- Method
- Variable
- Typical algorithm

- Exported name
- Test case
- TODO
- FIXME

如何进行注释?

格式

- Go
  - 使用 `//` 来进行单行注释
  - 使用 `/* ... */` 对代码块进行注释
  - 使用 **gofmt** 来格式化你的代码
- 把单行注释、代码块注释放在代码上方
- 折叠多列注释
- 注释中的每行文本不超过 100 个词
- 包含 URL 的注释：
  - 如果文本链接到同一个 GitHub 存储库中的文件，则使用 **relative URL**，
  - 如果带有此注释的代码是从另一个存储库复制来的，则使用 **absolute URL**，

语言

- 单词
  - 请统一使用**美式英语**
    - color, canceling, synchronize (推荐)
    - colour, cancelling, synchronise (不推荐)
  - 注意拼写正确
  - 使用**标准或官方的大写**
    - MatrixOne, Raft, SQL (正确)

        - matrixone, RAFT, sql (错误)
    - 使用一致性的短语和词汇
        - “dead link” vs. “broken link”（在一篇文章或文件中只能出现其中一个）
    - 不要使用冗长的复合词
    - 尽量不使用缩写
    - 用 *We* 用来代指作者和读者
- 句子
    - 使用标准的语法和标点符号
    - 尽量使用短句
- 句子首字母大写，并以句号结尾
    - 如果一个小写的标识符位于句子开头，可以不用大写

```
// enterGame causes Players to enter the
// video game, which is about a romantic
// story in ancient China.
func enterGame() os.Error {
...
}
```

- 当注释用来描述代码时，应该保证使用**描述性语句**而非**祈使句**
    - Opens the file (正确)
    - Open the file (错误)
- 使用 “this” 而非 “the” 来指代当前事物
    - Gets the toolkit for this component (推荐)
    - Gets the toolkit for the component (不推荐)
- 允许使用 Markdown 语法格式
    - Opens the `log` file

Tips

- 尽量在写代码的同时就完成注释，减少事后返工

- 不要假设代码是不证自明的
- 对于简单代码尽量避免使用过多注释
- 有“代入感”地做注释（仿佛在进行对话一般）
- 确保及时更新注释
- 保证注释清楚易懂

感谢您的贡献！

提交信息&PR 规范

本文档描述了应用于 MatrixOrigin 的所有存储库的提交消息 (commit mesage) 和 PR(pull request) 的样式规范。当你提交代码时，务必遵循这种规范，保证提交的质量。

一条好的 commit message 有多重要?

- 加速审阅流程
  - 帮助审阅者更好地理解 PR 内容
  - 可以忽略不重要的信息
- 有助于撰写发布公告
- 帮助其他人了解前因后果

什么是好的 commit message?

我们认为有以下要素：

1. **What is your change? (必要)**

   它可能修复了一个特定的 bug，添加了一个 feature，提高了性能、可靠性或稳定性，或者只是保障安全性而进行的更改。

2. **Why this change was made? (必要)**

   对于简要的补丁，这部分可以省略。

3. **What effect does the commit have? (可选)**

   除了必然会产生的影响之外，可能还包括基准测试性能变化、对安全性的影响等。对于简要的改动，这部分可以省略。

如何写好一条 commit message?

要写出一条优质的 commit message，我们建议您遵循规定的格式，培养良好的习惯并使用规范的语言。

规定的格式

请在提交时遵循以下格式：

```
<subsystem>: <what changed>
<BLANK LINE>
<why this change was made>
<BLANK LINE>
<footer>(optional)
```

- 第一行
  - 不超过 70 个字符
  - 如果该改动影响了两个模块，请使用逗号和（带空格）进行分隔，如 `util/codec, util/types:`。
  - 如果该改动影响了三个及以上的模块，请使用 `*`，如 `*:`。
  - 在冒号后的文本中使用小写字母。例如：“media: **update** the DM architecture image”
  - 不要在最后添加句号。
- 第二行请留白
- 第三行 “why” 部分，如果没有特定的原因，您可以使用以下表述，如 “Improve performance”，“Improve test coverage.”
- 其他行不超过 80 个字符。

良好的习惯

- 进行总结
- 清楚地描述该方案的逻辑，避免 `misc fixes` 等表达
- 叙述当前方案的限制
- 不要以句号结尾
- 注意代码的证明和测试
- 交代前因后果

规范的语言

- 在第一行使用祈使句
- 使用简单的动词 (如 “add” not “added”)
- 保证标准无误的语法
- 前后使用的单词、短语保持一致
- 使用短句
- 不要使用过长的复合词
- 非必要不缩写

Pull Request 规范

关于 Pull Request 中的描述，请参考下面的 Pull Request 模板，涵盖必要信息：

```
**What type of PR is this?**

- [ ] API-change
- [ ] BUG
- [ ] Improvement
- [ ] Documentation
- [ ] Feature
- [ ] Test and CI
- [ ] Code Refactoring

**Which issue(s) this PR fixes:**

issue #

**What this PR does / why we need it:**


**Special notes for your reviewer:**


**Additional documentation (e.g. design docs, usage docs, and so on.):**
```

如果需要，您也可以使用清单来列举内容，Markdown 语法如下：

```
- [x] A checked line, something already done or fulfilled
- [ ] An unchecked line, something not finished yet
```

对于非常简要的 Pull Requests，你可以省略上面的一些信息。 非常感谢您的贡献！