私塾在线《高级软件架构师实战培训阶段二》

——跟着cc学架构系列精品教程

MySQL的分区-1

- 什么是分区

所谓分区就是将一个表分解成多个区块进行操作和保存，从而降低每次操作的数据，提高性能。而对应用来说是透明的，从逻辑上看是只有一个表（这里跟分库分表的访问不一样），但在物理上这个表可能是由多个物理分区组成的，每个分区都是一个独立的对象，可以进行独立处理。

- 分区能干什么

1：进行逻辑数据分割，分割数据能够有多个不同的物理文件路径
2：可以存储更多的数据，突破系统单个文件最大限制
3：提升性能，提高每个分区的读写速度，提高分区范围查询的速度
4：可以通过删除相关分区来快速删除数据
5：通过跨多个磁盘来分散数据查询，从而提高磁盘I/O的性能
6：涉及到例如SUM()和COUNT()这样聚合函数的查询，可以很容易地进行并行处理
7：可以备份和恢复独立的分区，这对大数据量很有好处

- 分区能支持的引擎

MySQL支持大部分的存储引擎创建分区，如MyISAM、InnoDB等；不支持MERGE和CSV等来创建分区。同一个分区表中的所有分区必须是同一个存储引擎。

MySQL的分区-2

- 确认MySQL支持分区

从MySQL5.1开始引入分区功能，可以如下方式查看是否支持：

1：“老”的版本用：SHOW VARIABLES LIKE '%partition%';

2：新的版本用：show plugins;

- 分区类型

1：RANGE分区：基于属于一个给定连续区间的列值，把多行分配给分区

2：LIST分区：类似于按RANGE分区，LIST是列值匹配一个离散值集合中的某个值来进行选择

3：HASH分区：基于用户定义的表达式的返回值来进行选择的分区，该表达式使用将要插入到表中的这些行的列值进行计算，这个函数必须产生非负整数值

4：KEY分区：类似于按HASH分区，由MySQL服务器提供其自身的哈希函数

- 但是不论什么类型的分区，都要注意以下问题：

1，如果表中存在primary key或者unique key时，分区的列必须是primary key或者unique key的一个组成部分，也就是说，分区函数的列只能从pk或者uk这些key中取子集

2，如果表中不存在任何的primary key或者unique key，则可以指定任何一个列作为分区列

3：5.5版本前的Range、List、Hash分区要求分区键必须是int；MySQL5.5及以上，支持非整型的Range和List分区，即：range columns 和 list columns。

MySQL的分区-3

- 分区命名

分区的名字基本上遵循其他MySQL 标识符应当遵循的原则，例如用于表和数据库名字的标识符。但是应当注意，分区的名字是不区分大小写的。

无论使用何种类型的分区，分区总是在创建时就自动的顺序编号，且从0开始记录。

- 创建分区

1：RANGE分区

```
CREATE TABLE tbl_users (
    uuid INT NOT NULL,
    customerId VARCHAR(20),
    pwd VARCHAR(20),
    showName VARCHAR(100),
    trueName VARCHAR(100),
    registerTime VARCHAR(100)
)
PARTITION BY RANGE (uuid) (
    PARTITION p0 VALUES LESS THAN (5),
    PARTITION p1 VALUES LESS THAN (10),
    PARTITION p2 VALUES LESS THAN (15),
    PARTITION p3 VALUES LESS THAN MAXVALUE
);
```

```
CREATE TABLE tbl_users7(
    uuid INT NOT NULL,
    customerId VARCHAR(20),
    pwd VARCHAR(20),
    showName VARCHAR(100),
    trueName VARCHAR(100),
    registerTime  Date
)
PARTITION BY RANGE(YEAR(registerTime))
    SUBPARTITION BY HASH(TO_DAYS(registerTime))
    (
        PARTITION p0 VALUES LESS THAN (2008)(
            SUBPARTITION s0,
            SUBPARTITION s1
        ),
        PARTITION p1 VALUES LESS THAN (2015)(
            SUBPARTITION s2,
            SUBPARTITION s3
        ),
        PARTITION p2 VALUES LESS THAN MAXVALUE(
            SUBPARTITION s4,
SUBPARTITION s5
        )
    );
```

MySQL的分区-4

（1）到存放数据的地方查看文件，路经配置在/usr/bin/mysql_config里面的ldata。

（2）可以通过使用形如：select * from information_schema.partitions where table_schema=‘arch1’ and table_name=‘tbl_users’ \G; 的语句来查看分区信息

（3）可以通过形如select * from tbl_users partition(p0);的语句来查看分区上的数据

（4）可以使用形如explain partitions select * from tbl_users where uuid=2;的语句来查看MySQL会操作的分区

2：List分区

```
PARTITION BY List (uuid) (
    PARTITION p0 VALUES in (1,2,3,5),
    PARTITION p1 VALUES in (7,9,10),
    PARTITION p2 VALUES in (11,15)
);
```

（1）如果试图操作的列值不在分区值列表中时，那么会失败并报错。要注意的是，LIST分区没有类似如“VALUES LESS THAN MAXVALUE”这样的包含其他值在内的定义，将要匹配的任何值都必须在值列表中找到。

（2）LIST分区除了能和RANGE分区结合起来生成一个复合的子分区，与HASH和KEY分区结合起来生成复合的子分区也是可以的。

MySQL的分区-5

3：Hash分区

HASH分区主要用来确保数据在预先确定数目的分区中平均分布。在RANGE和LIST分区中，必须明确指定一个给定的列值或列值集合以指定应该保存在哪个分区中；而在HASH分区中，MySQL自动完成这些工作，要做的只是基于将要被哈希的列值指定一个表达式，以及指定被分区的表将要被分割成的分区数量，如：

```
PARTITION BY HASH (uuid)
PARTITIONS 3;
```

（1）由于每次插入、更新、删除一行，这个表达式都要计算一次；这意味着非常复杂的表达式可能会引起性能问题，尤其是在执行同时影响大量行的运算（例如批量插入）的时候。

（2）最有效率的哈希函数是只对单个表列进行计算，并且它的值随列值进行一致地增大或减小，因为这考虑了在分区范围上的“修剪”。也就是说，表达式值和它所基于的列的值变化越接近，就能越有效地使用该表达式来进行HASH分区。

3.1：线性Hash分区

线性哈希分区在“PARTITION BY” 子句中添加“LINEAR”关键字。

线性哈希分区的优点在于增加、删除、合并和拆分分区将变得更加快捷，有利于处理含有极其大量数据的表。它的缺点在于，各个分区间数据的分布不大可能均衡。

MySQL的分区-6

4：Key分区

类似于按照HASH分区，Hash分区允许用户自定义的表达式，而Key分区不允许使用用户自定义的表达式； Hash分区只支持整数分区，Key分区支持除了blob或text类型之外的其他数据类型分区。

与Hash分区不同，创建Key分区表的时候，可以不指定分区键，默认会选择使用主键或唯一键作为分区键，没有主键或唯一键，就必须指定分区键。

```
CREATE TABLE tbl_users4 (
    uuid INT NOT NULL,
    customerId VARCHAR(20),
    pwd VARCHAR(20),
    showName VARCHAR(100),
    trueName VARCHAR(100),
    registerTime VARCHAR(100)
)
PARTITION BY LINEAR Key (uuid)
PARTITIONS 3;
```

MySQL的分区-7

5：子分区

子分区是分区表中每个分区的再次分割，适合保存非常大量的数据。

```
CREATE TABLE tbl_users5 (
    ......
    registerTime Date )
PARTITION BY RANGE(YEAR(registerTime))
    SUBPARTITION BY HASH(TO_DAYS(registerTime))
    SUBPARTITIONS 2
    (
        PARTITION p0 VALUES LESS THAN (2008),
        PARTITION p1 VALUES LESS THAN (2015),
        PARTITION p2 VALUES LESS THAN MAXVALUE
    );
```

（1）在MySQL 5.1中，对于已经通过RANGE或LIST分区了的表再进行子分区是可能的。子分区既可以使用HASH希分区，也可以使用KEY分区。这也被称为复合分区

（2）每个分区必须有相同数量的子分区

（3）如果在一个分区表上的任何分区上使用SUBPARTITION 来明确定义任何子分区，那么就必须定义所有的子分区

（4）每个SUBPARTITION 子句必须包括 (至少)子分区的一个名字

MySQL的分区-8

（5）在每个分区内，子分区的名字必须是唯一的，目前在整个表中，也要保持唯一。 例如：

```
PARTITION BY RANGE(YEAR(registerTime))
    SUBPARTITION BY HASH(TO_DAYS(registerTime))
    (
        PARTITION p0 VALUES LESS THAN (2008)(
          SUBPARTITION s0,
          SUBPARTITION s1
        ),
        PARTITION p1 VALUES LESS THAN (2015)(
          SUBPARTITION s2,     SUBPARTITION s3
        ),
        PARTITION p2 VALUES LESS THAN MAXVALUE(
          SUBPARTITION s4,     SUBPARTITION s5
        )
    );
```

子分区可以用于特别大的表，可以在多个磁盘间分配数据和索引。 例如：

```
    SUBPARTITION s0
                DATA DIRECTORY = '/disk0/data'
                INDEX DIRECTORY = '/disk0/idx',
            SUBPARTITION s1
                DATA DIRECTORY = '/disk1/data'
                INDEX DIRECTORY = '/disk1/idx'
```

MySQL的分区-9

- MySQL分区处理NULL值的方式

  MySQL中的分区在禁止空值NULL上没有进行处理，无论它是一个列值还是一个用户定义表达式的值，一般而言，在这种情况下MySQL把NULL视为0。如果你希望回避这种做法，你应该在设计表时声明列“NOT NULL”

- 分区管理概述

  可以对分区进行添加、删除、重新定义、合并或拆分等管理操作。

- RANGE和LIST分区的管理

1：删除分区语句如：alter table tbl_users drop partition p0;

（1）当删除了一个分区，也同时删除了该分区中所有的数据

（2）可以通过show create table tbl_users;来查看新的创建表的语句

（3）如果是List分区的话，删除的数据不能新增进来，因为这些行的列值包含在已经删除了的分区的值列表中

MySQL的分区-10

2：添加分区语句如：alter table tbl_users add partition(partition p3 values less than(50));

（1）对于RANGE分区的表，只可以添加新的分区到分区列表的高端

（2）对于List分区的表，不能添加已经包含在现有分区值列表中的任意值

3：如果希望能不丢失数据的条件下重新定义分区，可以使用如下语句：

ALTER TABLE tbl_name REORGANIZE PARTITION partition_list INTO (partition_definitions)

（1）拆分分区如：alter table tbl_users REORGANIZE PARTITION p1 INTO(partition s0 values less than(5), partition s1 values less than(10));

或者如：alter table tbl_users2 REORGANIZE PARTITION p0 INTO(partition s0 values in(1,2,3), partition s1 values in(4,5));

（2）合并分区如：alter table tbl_users2 REORGANIZE PARTITION s0,s1 INTO(partition p0 values in(1,2,3,4,5));

4：删除所有分区，但保留数据，形如： alter table tbl_users remove partitioning;

MySQL的分区-11

- HASH和KEY分区的管理

1：减少分区数量语句如： alter table tbl_users3 COALESCE PARTITION 2;

2：添加分区数量语句如：alter table tbl_users3 add PARTITION partitions 2;

- 其它分区管理语句

1：重建分区：类似于先删除保存在分区中的所有记录，然后重新插入它们，可用于整理分区碎片。如：alter table tbl_users REBUILD PARTITION p2,p3;

2：优化分区：如果从分区中删除了大量的行，或者对一个带有可变长度的行（也就是说，有VARCHAR，BLOB，或TEXT类型的列）作了许多修改，可以使用“ALTER TABLE ... OPTIMIZE PARTITION”来收回没有使用的空间，并整理分区数据文件的碎片。如：alter table tbl_users OPTIMIZE PARTITION p2,p3;

3：分析分区：读取并保存分区的键分布，如：alter table tbl_users ANALYZE PARTITION p2,p3;

4：检查分区：检查分区中的数据或索引是否已经被破坏，如：alter table tbl_users CHECK PARTITION p2,p3;

5：修补分区： 修补被破坏的分区，如：alter table tbl_users REPAIR PARTITION p2,p3;

MySQL的分区-12

- 其它

1：最大分区数目不能超过1024，一般建议对单表的分区数不要超过150个
2：如果含有唯一索引或者主键，则分区列必须包含在所有的唯一索引或者主键在内
3：不支持外键
4：不支持全文索引，对分区表的分区键创建索引，那么这个索引也将被分区
5：按日期进行分区很合适，因为很多日期函数可以用。但是对于字符串来说合适的分区函数不太多
6：只有RANG和LIST分区能进行子分区，HASH和KEY分区不能进行子分区
7：临时表不能被分区
8：分区表对于单条记录的查询没有优势
9：要注意选择分区的成本，每插入一行数据都需要按照表达式筛选插入的分区
10：分区字段尽量不要可以为null

分库分表-1

- 为什么要分库分表

数据库的复制能解决访问问题，并不能解决大规模的并发写入问题，由于无法进行分布式部署，而一台服务器的资源（CPU、磁盘、内存、IO等）是有限的，最终数据库所能承载的数据量、数据处理能力都将遭遇瓶颈。

要解决这个问题就要考虑对数据库进行分库分表了，它有如下好处：

1：解决磁盘系统最大文件限制，比如常见的有：
FAT16(最大分区2GB,最大文件2GB)
FAT32(最大分区32GB，最大容量2TB，最大文件32G)
NTFS(最大分区2TB，最大容量，最大文件2TB)
EXT3(最大文件大小：2TB，最大文件极限：仅受文件系统大小限制，最大分区/文件系统大小：4TB，最大文件名长度：255 字符)

2：减少增量数据写入时的锁对查询的影响，减少长时间查询造成的表锁，影响写入操作等锁竞争的情况，节省排队的时间开支，增加吞吐量。

3：由于单表数量下降，常见的查询操作由于减少了需要扫描的记录，使得单表单次查询所需的检索行数变少，减少了磁盘IO，时延变短。

分库分表-2

- 什么是分库

分库又叫垂直切分，就是把原本存储于一个库的表拆分存储到多个库上，通常是将表按照功能模块、关系密切程度划分出来，部署到不同的库上。

如果数据库是因为表太多而造成海量数据，并且项目的各项业务逻辑划分清晰、低耦合，那么规则简单明了、容易实施的首选就是分库。

分库的优点是：实现简单，库与库之间界限分明，便于维护，缺点是不利于频繁跨库操作，单表数据量大的问题解决不了。

- 什么是分表

分表又叫水平切分，是按照一定的业务规则或逻辑，将一个表的数据拆分成多份，分别存储在多个表结构一样的表中，这多个表可以存在一到多个库中。分表又分成垂直分表和水平分表：

垂直分表：将本来可以在同一个表的内容，人为划分为多个表。（所谓的本来，是指按照关系型数据库的第三范式要求，是应该在同一个表的。）

水平分表，也被称为数据分片：是把一个表复制成同样表结构的不同表，然后把数据按照一定的规则划分，分别存储到这些表中，从而保证单表的容量不会太大，提升性能；当然这些结构一样的表，可以放在一个或多个数据库中。

分库分表-3

分表的优点是：能解决分库的不足点，但是缺点是实现起来比较复杂，特别是分表规则的划分，程序的编写，以及后期的数据库拆分移植维护。

一般都是先分库再分表，两者结合使用，取长补短，这样能发挥扩展的最大优势，但是缺点是架构很大，很复杂，应用程序的编写也比较复杂。

- 如何分库

基本的思路就是分析业务功能，以及表间的聚合关系，把关系紧密的表放在一起。

分库的粒度指的是在做切分时允许几级的关联表放在一起，这个问题对应用程序实现有着很大的影响。关联打断的越多，则受影响的join操作越多，应用程序为此做出的妥协就越大，但单表的路由会越简单，与业务的关联性会越小，就越容易使用统一机制处理。

实际的粒度掌控需要结合“业务紧密程度”和“表的数据量”两个因素综合考虑，一般来说：若划归到一起的表关系紧密，且数据量并不大，增速也非常缓慢，则适宜放在一起，不需要再进行水平切分；若划归到一起的表的数据量巨大且增速迅猛，则势必要在分库的基础上再进行分表，这就意味着原单一的库还可能会被拆分成多个库，这会导致更多的复杂性，一开始最好就要考虑进去。

分库分表-4

- 如何分表

对于垂直分表，通常是按照业务功能的使用频次，把主要的、热门的字段放在一起做为主要表；然后把不常用的，按照各自的业务属性进行聚集，拆分到不同的次要表中；主要表和次要表的关系一般都是一对一的。

对于水平分表，通常是按照具体的业务规则和数据的格式，选择能够把数据进行合理拆分的业务数据做为拆分标准，以此来对数据进行拆分。

常见的一些拆分方式：按业务属性、按时间、按区间、Hash、按数据的活跃度、按数据量等，不管采用什么方式，都要结合具体的业务场景进行分析和考量。

当然这个过程中要考虑很多问题，比如：预估的数据量大小、数据量增长速度、分表的数量多少、在多表中数据的均衡、多表负载的均衡、扩容、访问表的导航信息等

分表的表现又分成：单库单表、单库多表、多库多表几种。

分库分表-5

- 分库分表后的问题

1：分布式事务的问题，数据的完整性和一致性问题
2：数据操作的维度问题
　　例如保存交易记录，是按照用户的纬度分表保存，还是按照产品的纬度来分表保存。
3：跨库联合查询的问题，可能需要两次查询
4：跨节点的count、order by、group by以及聚合函数问题，可能需要分别在各个节点上得到结果，然后在应用程序端进行合并
5：额外的数据管理负担，如：访问数据表的导航定位
6：额外的数据运算压力，如：需要在多个节点执行，然后再合并计算
7：程序复杂度上升
8：后期维护难度上升，包括：程序的维护和升级、数据库的扩容、数据的迁移等

水平分表的实现-1

- 概述

水平分表的实现面临一系列问题：切分策略、库节点路由、表路由、全局主键生成、跨节点排序/分组/分页/表关联等操作、多数据源事务处理、数据库扩容等。

- 部分相关开源产品一览（排名不分先后）

1：MySQL Fabric：官方产品，非代理方式，目前不太稳定，性能也不够好，但很有前景，综合了HA和水平分表的功能，是未来的首选。

2：Atlas：360开源，代理方式，基于MySQL-Proxy二次开发的，主要支持两个特性：分表和读写分离，但是分表的话只支持单库多表，事实上是不支持分布式分表的

3：Cobar：阿里开源，代理方式，支持分布式分表，但是不支持单库分多表，不支持读写分离，事务支持也比较麻烦

4：TDDL （Taobao Distributed Data Layer ）：阿里部分开源，非代理方式，提供分库分表对应用的透明化，实现异构数据库之间的数据复制，具有主备，读写分离，动态数据库配置等功能。但复杂度相对较高，公布的文档少，核心部分不开源，还需要依赖Diamond

5：MySQL Proxy：官方提供，基于MySQL协议接口，主要提供负载平衡，读写分离，failover等，但性能较差，不支持大数据量的分库分表

6：Amoeba：支持分数据库实例，每个数据相同的表，不支持事务；类似MySQL Proxy，相对更简单

7：Hibernate Shards：支持分数据库实例，比较复杂，需事先规划数据规模，对HQL的支持非常有限

8：mybatis shardbatis：主要通过插件机制来实现分表，但是插件机制控制不到多数据源的连接；离开插件层又失去了对sql进行集中解析和路由的机会

水平分表的实现-2

- 现状——靠天靠地，不如靠自己

- 基本的实现思路

1：解析路由：根据业务功能指定；根据SQL解析等
不管采用何种方式，要获得需要访问的数据源，以及分别要访问的表
2：分别在数据源和表上去执行功能
3：如果涉及到返回结果集的话，就需要做结果集的合并，并按照需要进行二次处理，比如：排序、分页等
4：如果需要事务的话，就得考虑是使用分布式事务，还是自行实现两阶段提交，或者采用补偿性业务处理的方式等

- 可实现的层面

1：DAO层
2：Spring数据访问封装层，介于DAO与JDBC之间
3：JDBC驱动层
4：介于应用服务器与数据库之间的代理服务器

MySQL Fabric-1

- 概述

是一个用于管理MySQL服务器群的可扩展框架，实现了高可用性以及使用数据分片的横向扩展，这两个特性既可以单独使用，也可以结合使用。

在同一个分片内又可以包含多个数据库，并且由Fabric自动挑选一个适合的作为主数据库，其他的数据库配置成从数据库，来做主从复制。在主数据库挂掉时，自动从各个从数据库中挑选一个提升为主数据库，之后，其他的从数据库转向新的主数据库复制数据。

- 基本概念

1：HA group 或 group：是一个关联在一起的服务器集合，可能是一个复制集，也可能是数据分片。
2：group Identifier：组或者其成员的名字
3：global group：存储和维护所有的分片信息
4：node 或 fabric node：Fabric在服务器上运行的一个实例
5：sharding：是Fabric支持的跨越多个服务器进行数据分片的功能
6：shard：是指一个表的数据进行水平拆分
7：primary：主服务器，提供读写服务
8：secondary：从服务器，提供读的服务

MySQL Fabric-2

- 安装使用的前提

  安装需要：MySQL5.6.10以上，python2.6以上

  使用需要：连接器需要Connector/Python 1.2.1以上，Connector/J 5.1.27以上

- 基本的安装

一：安装python

1：http://python.org/ftp/python/2.7.6/Python-2.7.6.tgz

2：不要卸载旧的2.4.3的版本，直接安装新的版本，解压后，依次./configure、make、make install，可以通过--prefix=“”指定安装的位置

3：修改python命令的链接

（1）先把旧命令备份一下：mv /usr/bin/python /usr/bin/python_bak

（2）然后建立软链：ln -s /usr/common/python2.7.6/bin/python2.7 /usr/bin/python

二：安装Utilities，Fabric包含在里面

1：http://dev.mysql.com/Downloads/MySQLGUITools/mysql-utilities-1.5.4.tar.gz

2：解压后进入文件夹

3：运行python setup.py install

MySQL Fabric-3

- 基本的配置使用

1：建立单独的数据库帐号，包括Backing Store帐号，以及访问各个数据库的帐号，这些帐号名称

2：修改fabric配置文件，默认配置文件：/etc/mysql/fabric.cfg

3：初始化fabric，创建fabric数据库和相关表
先进入命令所在位置：cd /usr/common/python2.7.6/bin，然后执行
./mysqlfabric --config=/etc/mysql/fabric.cfg manage setup --param=storage.user=root --param=storage.password=cc

4：开启mysql fabric Nodes：./mysqlfabric --config=/etc/mysql/fabric.cfg manage start

5：可以在后面增加--daemonize选项来后台启动，日志将到/var/log/fabric.log

6：关闭mysql fabric Nodes：./mysqlfabric --config=/etc/mysql/fabric.cfg manage stop

7：当然你也可以按照提示，把fabric.cfg拷贝到相应位置，这样就不用每次都去指定config了，如：cp /etc/mysql/fabric.cfg /usr/common/python2.7.6/etc/mysql/fabric.cfg，需要提前把文件夹建好

MySQL Fabric-4

- HA配置使用

1：正常启动Fabric的Node

2：创建HA服务器组：./mysqlfabric group create hagroup1

3：添加HA组的成员，先要保证数据库开启了gtid（全局事务标识），如下：

```
log-bin=mysql-bin
gtid-mode=on
enforce-gtid-consistency=true
log_slave_updates=true
```

然后添加成员：./mysqlfabric group add hagroup1 192.168.1.101:3306

4：添加后察看一下组内成员：./mysqlfabric group lookup_servers hagroup1

5：选举一个主库：./mysqlfabric group promote hagroup1

6：配置故障自动切换：./mysqlfabric group activate hagroup1

7：就可以测试一下了

MySQL Fabric-5

- HA配置常用命令

1：mysqlfabric group create hagroup1：创建HA组
2：mysqlfabric group destroy hagroup1：删除HA组
3：mysqlfabric group add hagroup1 192.168.1.101:3306：添加组成员
4：mysqlfabric group remove hagroup1 9f93533a-3f39-11e51：移出组成员
5：mysqlfabric group lookup_servers hagroup1：查看组成员
6：mysqlfabric group promote hagroup1：选举master
7：mysqlfabric groupactivate hagroup1：激活自动故障转移
8：mysqlfabric group deactivatehagroup1：禁用自动故障转移
9：mysqlfabric server set_status server_uuid status：变更服务器状态
（primary,secondary,spare,faulty ）
10：mysqlfabric help manage：manage命令帮助
11：mysqlfabric help group：group命令帮助
12：mysqlfabric help server：server命令帮助

MySQL Fabric-6

- Sharding配置使用

1：创建多个用来分片的组，其中有一个Global的组，方法同前
2：为每个组中添加成员，如果不做HA，就只添加一个
3：选举一个成员作为主库
4：在Global组上定义分片策略：./mysqlfabric sharding create_definition RANGE shgroup-global
5：添加要分片的表及字段：./mysqlfabric sharding add_table 1 test.tbl_t1 uuid
6：给表添加分片信息：./mysqlfabric sharding add_shard 1 "Shgroup1/1, Shgroup2/1000 “--state=ENABLED
7：就可以测试一下了：./mysqlfabric sharding lookup_servers test.tbl_t1 2000