课堂主题

MySQL集群、主从复制、读写分离、分库分表概念

课堂目标

理解主从复制的原理

会配置主从复制

会对主从复制进行修复

会使用Mysql-proxy进行读写分离

理解垂直切分和水平切分

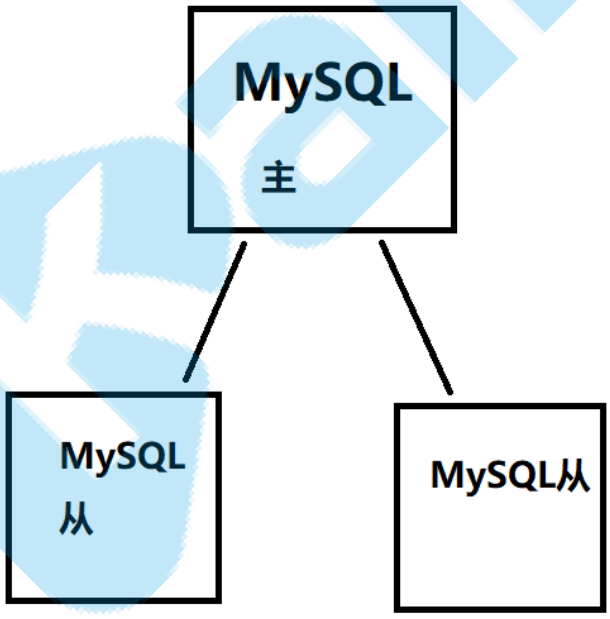
了解分库分表的问题和解决方案

MySQL集群篇

集群搭建之主从复制

主从复制原理

主对外工作，从对内备份。



单机MySQL的问题

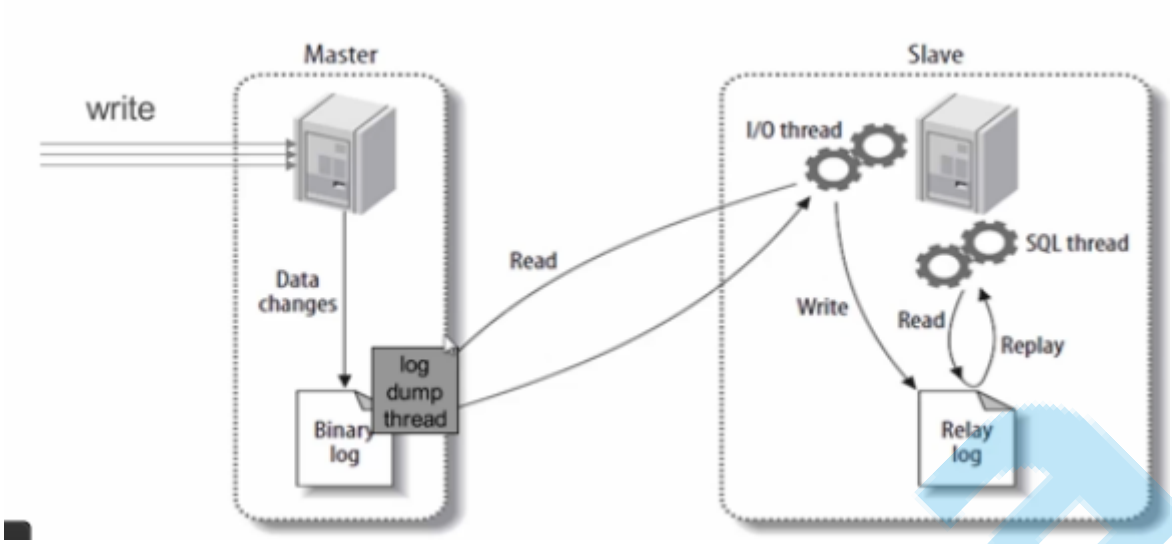
1、不能高可用

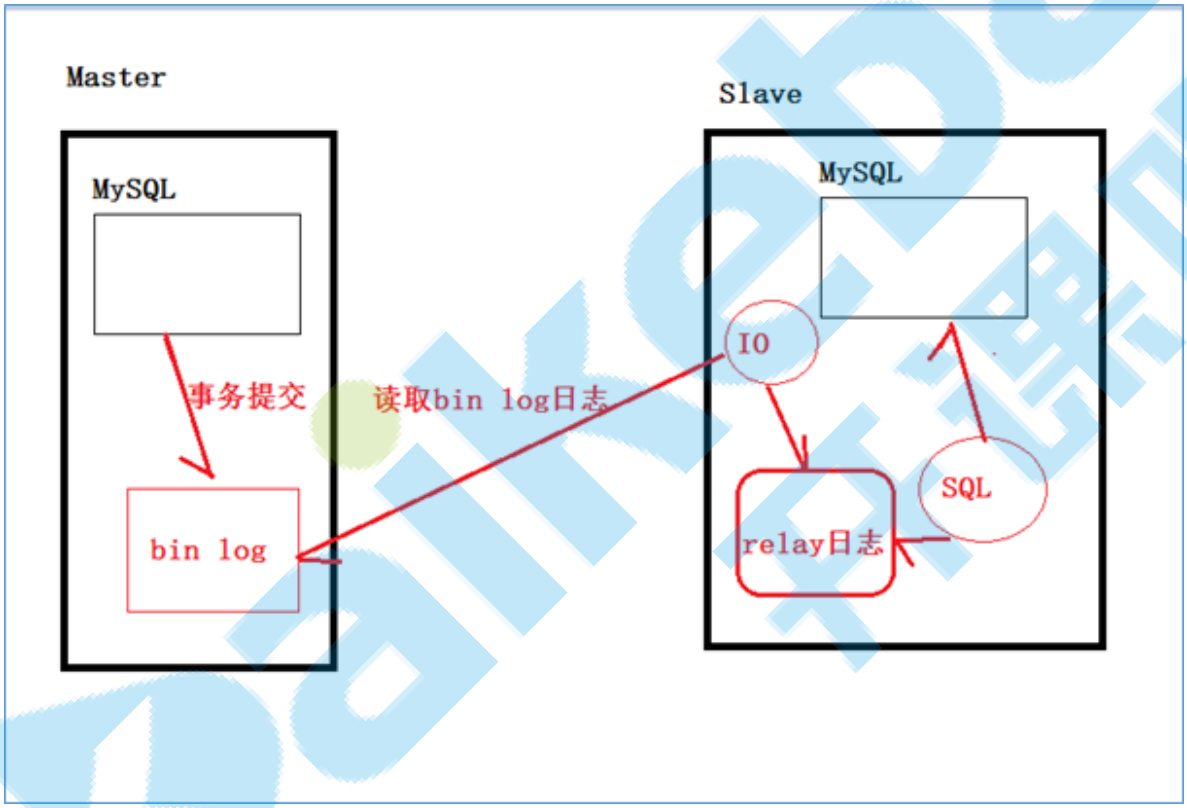
2、不能高并发

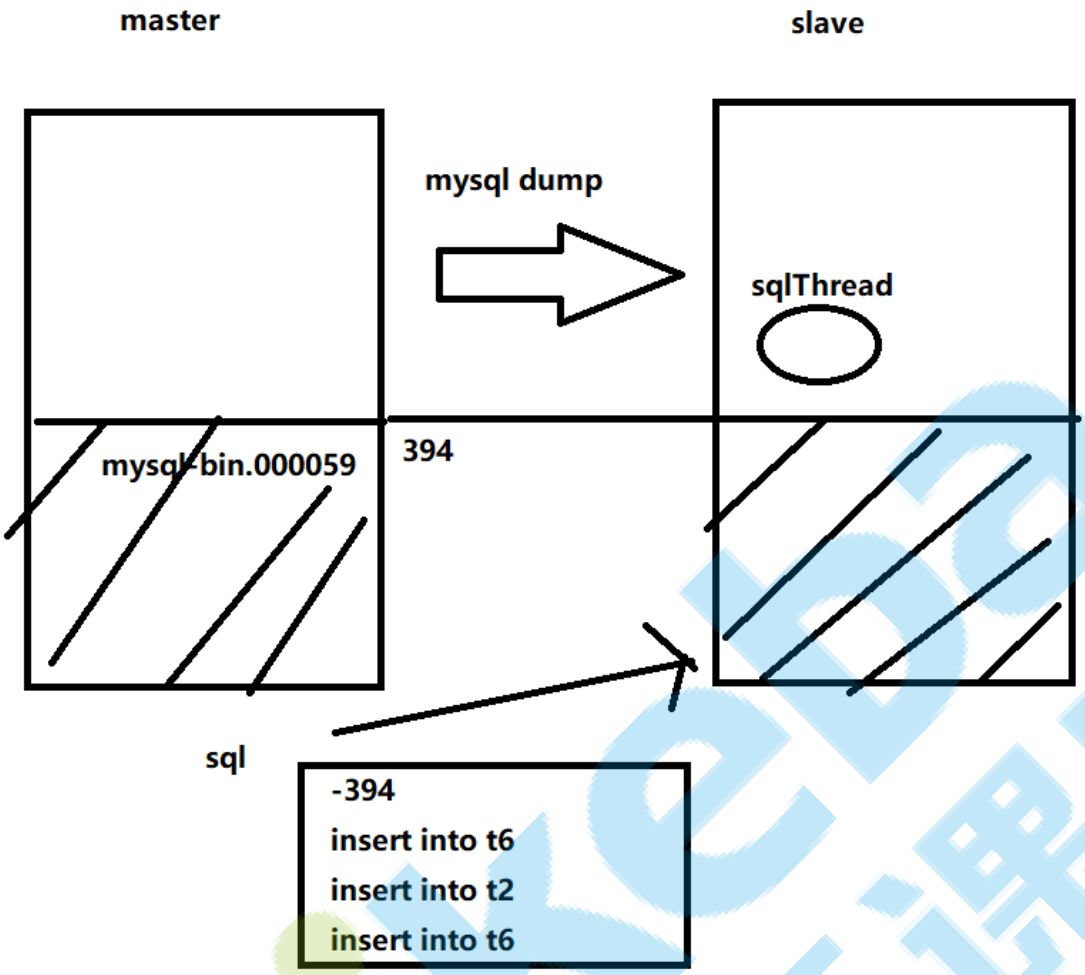
3、不能处理海量数据(5千万)

从一般是作为主库的备份

读写分离后，从可以读


Master
write
Data changes
log dump thread
Binary log
Read
Slave
I/O thread
SQL thread
Write
Read
Replay
Relay log


Master
MySQL
事务提交
bin log
读取bin log日志
Slave
MySQL
IO
SQL
relay日志



binlog介绍和relay日志

关闭主从机器的防火墙

```
systemctl stop iptables（需要安装iptables服务）
systemctl stop firewalld（默认）
systemctl disable firewalld.service（设置开启不启动）
```

主服务器配置

第一步：修改my.cnf文件

在[mysqld]段下添加:

```
#启用二进制日志
log-bin=mysql-bin
#服务器唯一ID，一般取IP最后一段
server-id=133
#指定复制的数据库(可选)
binlog-do-db=kkb2
binlog-ignore-db=kkb
#指定不复制的数据库(可选,, mysql5.7)
replicate-ignore-db=kkb
#指定忽略的表（可选，mysql5.7）
replicate-ignore-table = db.table1
```

第二步：重启mysql服务

```
systemctl  restart  mysqld
```

第三步：主机给从机授备份权限

```
GRANT REPLICATION SLAVE ON *.* TO 'root'@'%' identified by 'root';

GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' identified by 'root';
```

第四步：刷新权限

```
FLUSH PRIVILEGES;
```

第五步：查询master的状态

```
mysql> show master status;
+------------------+----------+--------------+------------------+--------------
----+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
Executed_Gtid_Set |
+------------------+----------+--------------+------------------+--------------
----+
| mysql-bin.000009 |      871 |              | kkb_2            |
    |
+------------------+----------+--------------+------------------+--------------
----+
1 row in set (0.00 sec)
```

主从备份： mysqldump

从服务器配置

第一步：修改my.cnf文件

```
[mysqld]
server-id=135
```

第二步：重启并登录到MySQL进行配置从服务器

以前是启动状态

show slave status \G;

先关闭 stop slave

同步初始化 master_log_file 、 master_log_pos 以主机状态为主 show master status

```
mysql>change master to
 master_host='192.168.56.101',
 master_port=3306,
 master_user='root',
 master_password='root',
 master_log_file='mysql-bin.000059',
 master_log_pos=394;
```

第四步：启动从服务器复制功能

```
mysql>start slave;
```

第五步：检查从服务器复制功能状态

```
mysql> show slave status \G;

.....(省略部分)
Slave_IO_Running: Yes //此状态必须YES
Slave_SQL_Running: Yes //此状态必须YES
.....(省略部分)
```

测试

搭建成功之后，往主机中插入数据，看看从机中是否有数据

```
注：  如果出现复制不成功，可以使用

set global sql_slave_skip_counter =1;  # 忽略一个错误
start slave
```

主从延时

1、因为SQLThread和IOThread是默认单线程，当主机的tps(每秒事务处理数)高于从机的Thread所能承受范围，则会出现从机复制延时

解决方案： 将thread改成多线程模式 MySQL5.6改表，MySQL5.7改GTID

2、网络延时

解决方案： 主和从在一个网内

3、IO延时

slave server硬件升级

判断延时：

show slave status中Seconds_Behind_Master=0则不延时

建表加时间戳(timestamp)，看时间差

解决方案： 利用分库分表中间件 Mycat、 sharding JDBC

强制读取主库

集群搭建之读写分离

读写分离的理解

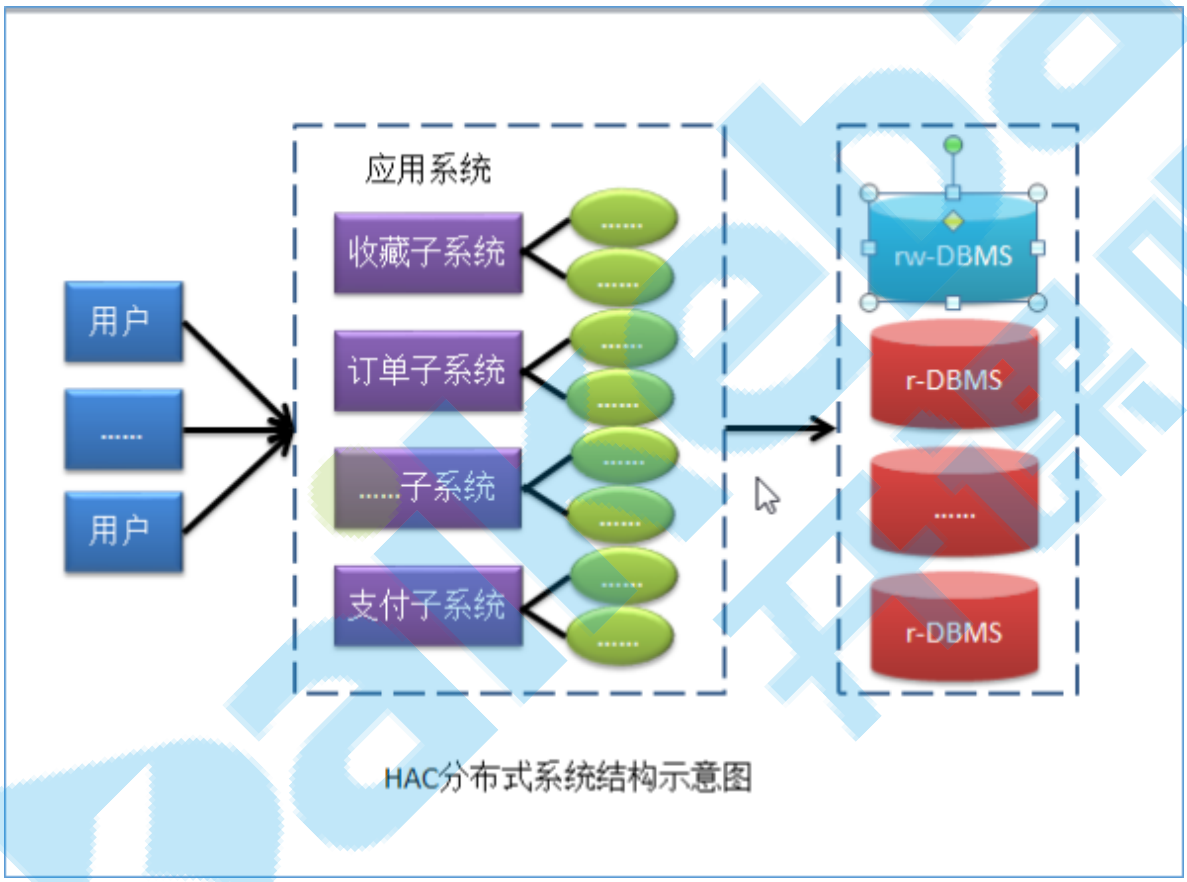
为什么要有读写分离集群？

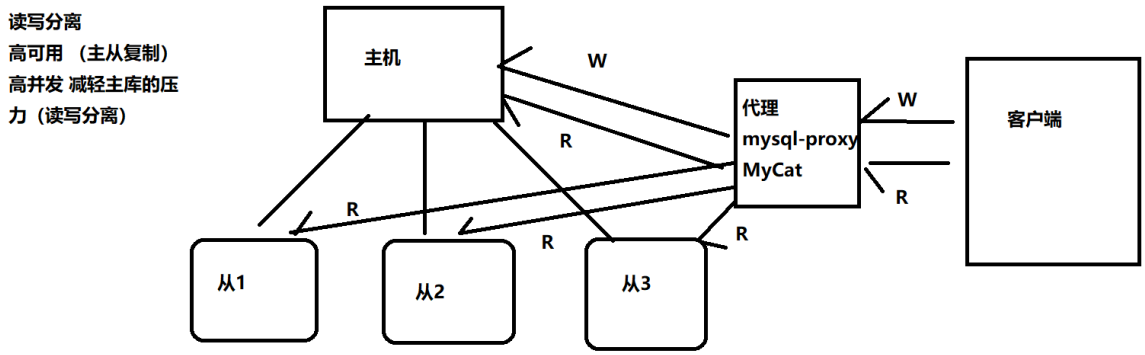
主从集群的问题：只有主对外工作，从不对外工作。主既要负责写操作，也要负责读操作。

对于主从集群来说，只是保证了数据的安全备份。

主：负责部分读 、写

从：负责读





名词解释：

```
HAC：High Availability Cluster，高可用集群
```

注意事项：

```
MySQL的主从复制，只会保证主机对外提供服务，而从机是不对外提供服务的，只是在后台为主机进行备份。
读写分离后，主负责写和部分读，从负责读，高性能高可用的数据库集群
```

读写分离演示需求

```
MySQL master：192.168.56.101

MySQL slave ：192.168.56.102

MySQL proxy ：192.168.56.102
```

MySQL-Proxy安装

- 下载

```
wget https://downloads.mysql.com/archives/get/file/mysql-proxy-0.8.5-linux-el6-x86-64bit.tar.gz
```

- 解压缩

```
tar -xf mysql-proxy-0.8.5-linux-el6-x86-64bit.tar.gz -C /kkb
```

MySQL-Proxy配置

- 创建mysql-proxy.cnf文件

```
[mysql-proxy]
user=root
admin-username=root
admin-password=root
proxy-address=192.168.10.137:4040
proxy-backend-addresses=192.168.10.135:3306
proxy-read-only-backend-addresses=192.168.10.136:3306
proxy-lua-script=/root/mysql-proxy/share/doc/mysql-proxy/rw-splitting.lua
log-file=/root/mysql-proxy/logs/mysql-proxy.log
log-level=debug
keepalive=true
daemon=true
```

- 修改mysql-proxy.cnf文件的权限

```
chmod 660 mysql-proxy.cnf    #可读写
```

- 修改rw-splitting.lua脚本

MySQL-Proxy启动域测试

- 启动命令

```
./mysql-proxy  --defaults-file=mysql-proxy.cnf配置文件的地址
```

注意事项：

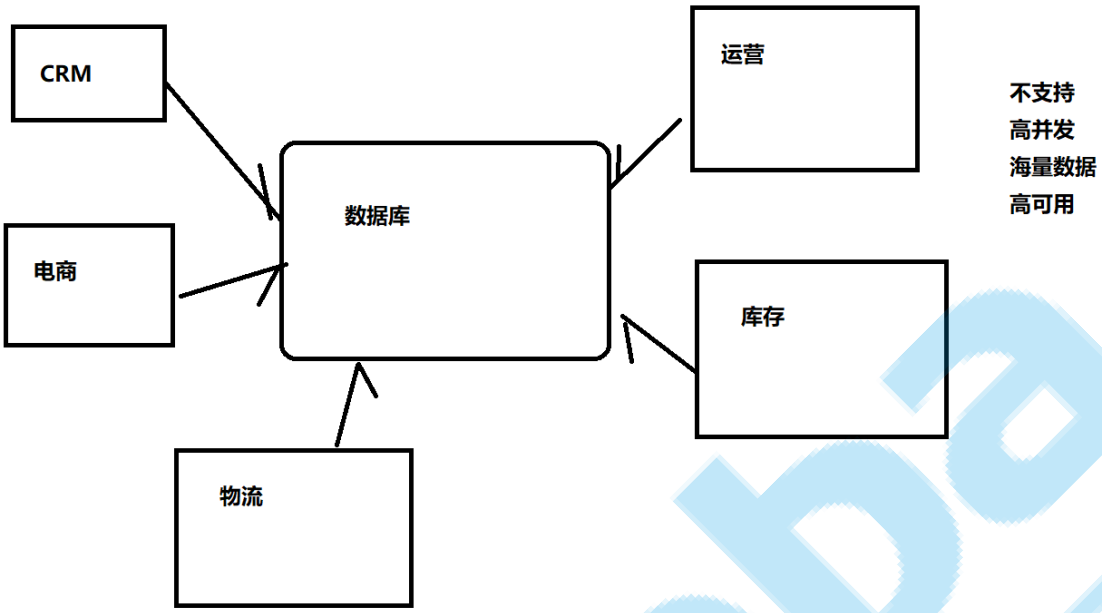
如果没有配置profile文件的环境变量，则需要去拥有mysql-proxy命令的目录通过./mysql-proxy进行启动。

- 在其他客户端，通过mysql命令去连接MySQL Proxy机器

```
mysql -uroot -proot -h192.168.56.102 -P4040;
注：  关闭防火墙
```

MySQL分库分表篇

传统项目结构



数据库性能瓶颈：

1、数据库连接数有限

MySQL数据库默认100个连接、单机最大1500连接。

2、表数据量

单机 表数量多，成百上千

单表数据，千万级别 5千万 (不超过100字节)

查询问题 索引，命中率问题，索引存磁盘，占空间

3、硬件问题

数据库性能优化

1、参数优化

2、缓存、索引

3、读写分离

4、分库分表 （最终方案）

MySQL （关系型数据库） ------> NoSQL (Redis、MongoDB)-------->NewSQL 分布式关系型数据库 TiDB

分库分表介绍

使用背景

当【表的数量】达到了几百上千张表时，众多的业务模块都访问这个数据库，压力会比较大，考虑对其进行分库。

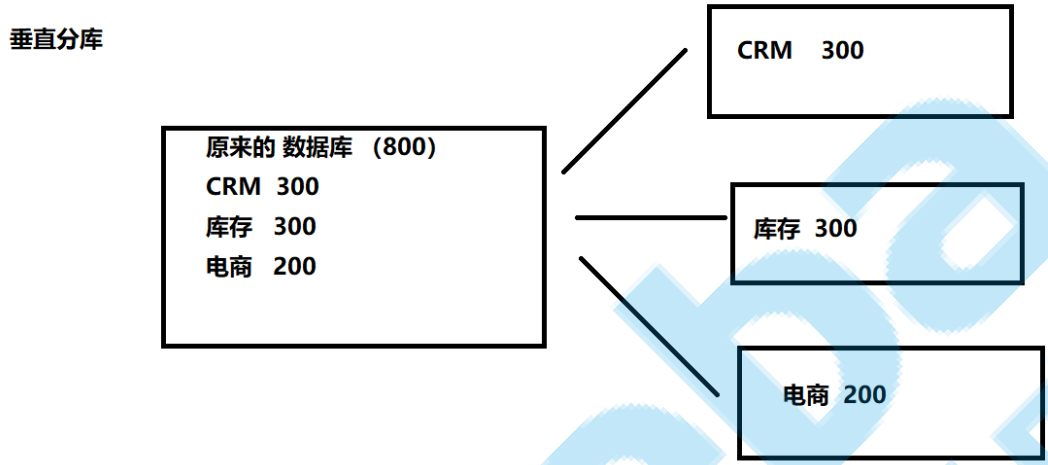
当【表的数据】达到了几千万级别，在做很多操作都比较吃力,所以，考虑对其进行分库或者分表

数据切分（sharding）方案

数据的切分（Sharding）根据其切分规则的类型，可以分为两种切分模式：

- **垂直切分**：按照业务模块进行切分，将不同模块的表切分到不同的数据库中。

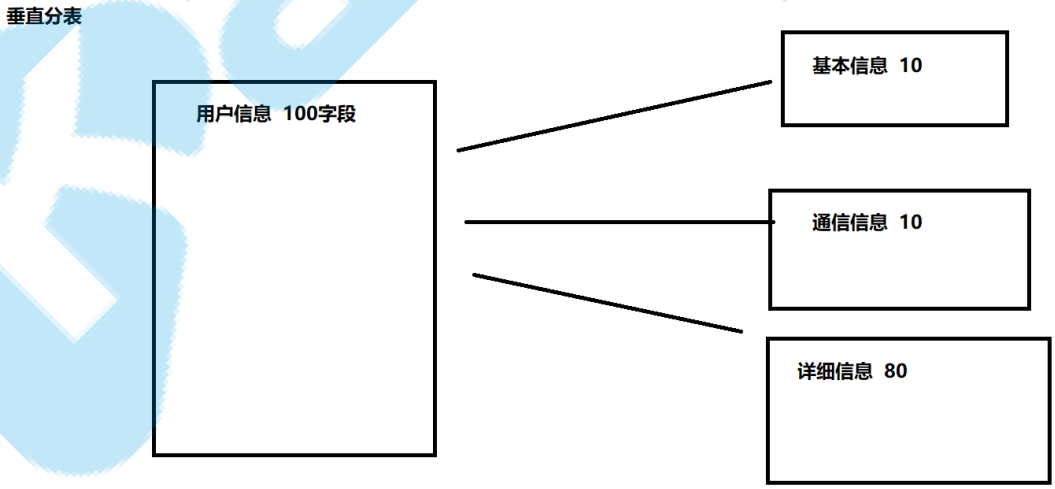
  分库



  分表

  大表拆小表

  Blob Clob : 二进制数据 头像 小图片

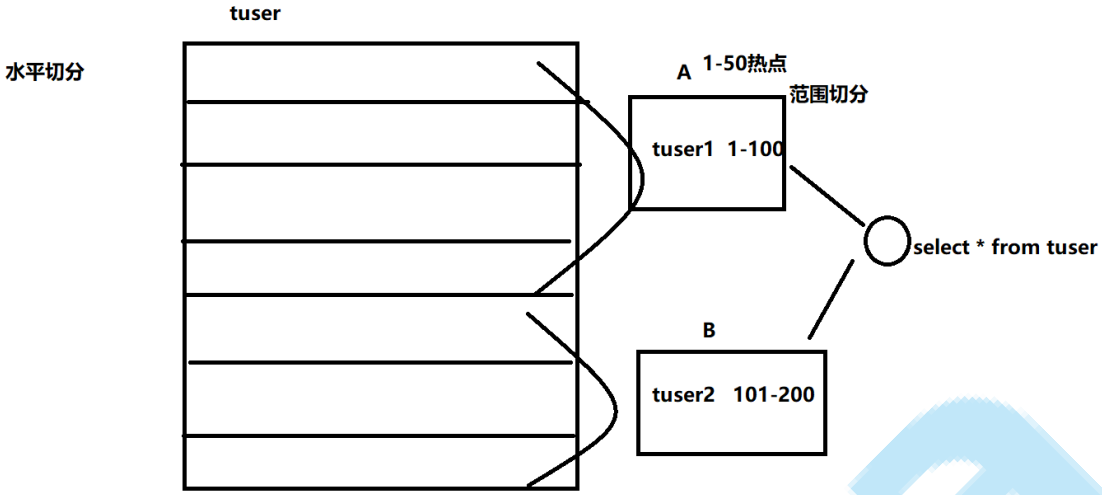


- **水平切分**：将一张大表按照一定的切分规则，按照**行**切分成不同的表或者切分到不同的库中

  范围式拆分

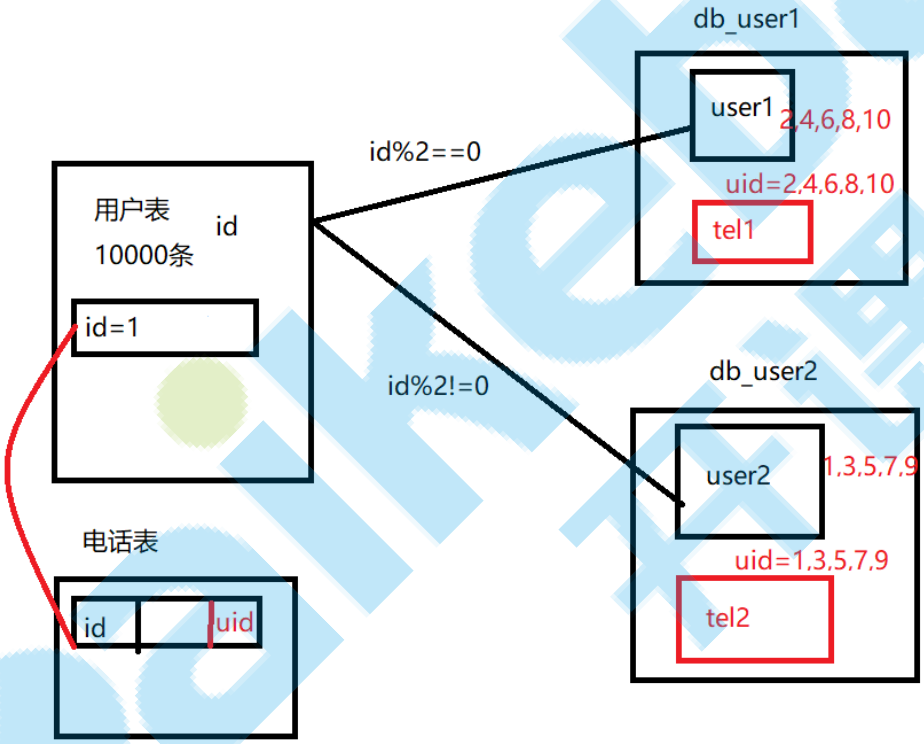
  好处：数据迁移是部分迁移，扩展性好

  坏处：热点数据分布不均，压力不能负载



水平切分

按照行切



hash式拆分

好处：热点数据分布均匀，访问压力能负载

坏处：扩展能力差，数据都要迁移

综合式：

先范围后hash

一致性hash环（mycat录播）

水平切分规则

- **按照ID取模：**对ID进行取模，余数决定该行数据切分到哪个表或者库中
- **按照日期：**按照年月日，将数据切分到不同的表或者库中
- **按照范围：**可以对某一列按照范围进行切分，不同的范围切分到不同的表或者数据库中。

切分原则

第一原则：能不切分尽量不要切分。

第二原则：如果要切分一定要选择合适的切分规则，提前规划好。 （向上取整）

第三原则：数据切分尽量通过数据冗余或表分组（Table Group）来**降低跨库 Join** 的可能。

分库分表需要解决的问题

分布式事务问题

分布式事务问题

```
本地事务：ACID
分布式事务：根据百度百科的定义，CAP定理又称CAP原则，指的是在一个分布式系统中，Consistency（一致性）、 Availability（可用性）、Partition tolerance（分区容错性）。一致性是强一致性。CAP理论最多只能同时满足两个。
BASE：基本可用+软状态+最终一致性
```

补偿事务，例如：TCC

利用记录日志等方式

分布式主键ID问题

多个库中表的主键冲突

redis incr命令

数据库（生成主键）

UUID （不好）

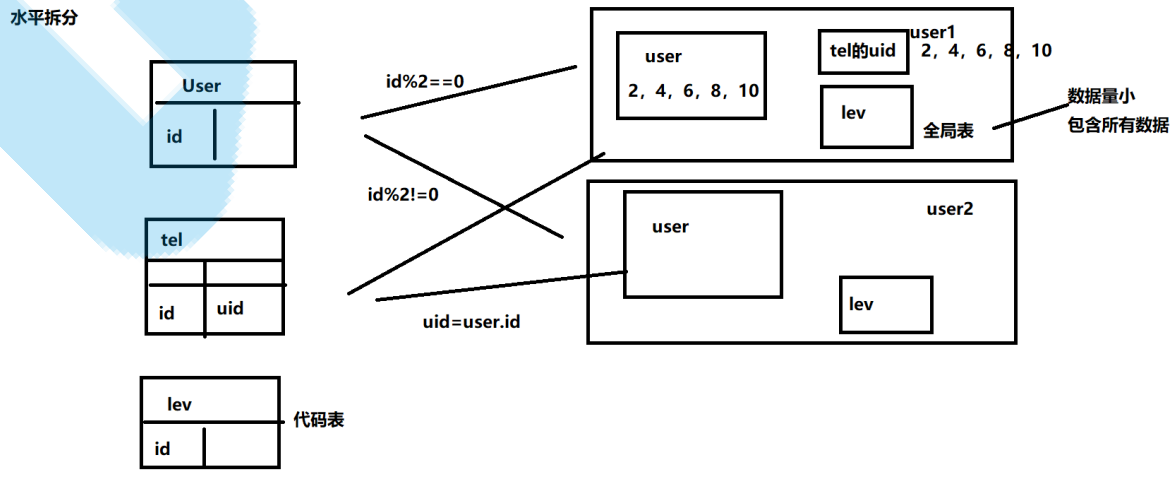
长、不好排序

snowflake算法（https://www.sohu.com/a/232008315_453160）

跨库join问题

建立全局表（每个库都有一个相同的表） 代码表

E-R分片（将有ER关系的记录都存储到一个库中）



最多支持跨两张表跨库的join

分库分表实现技术

阿里的TDDL、Cobar

基于阿里Cobar开发的Mycat

当当网的sharding-jdbc

Sharding案例分析

用户表

uid、name、city、timestamp、sex、age

5亿条数据

x86 64位机

查询维度单一 uid

问:

分几张表?

PartitionKey如何选择?

uid、city、timestamp ?

分表原则

单行数据大于100字节 则 1千万一张表

单行数据小于100字节则5千万一张表

用户表单行数据小于100字节

5亿/5千万=10张 上取整为 16张表

使用哪个PartitionKey?

city、timestamp 会造成热点数据分布不均匀

使用uid

uid%16