# 数据库设计规范回顾

### 三范式

可以做到节省磁盘空间(防止数据冗余)，牺牲查询时间

* + 背景：早期一种使用的范式设计方式，硬盘空间资源比较稀缺，而又不需要提升用户体验。考虑一种设计模式，节省空间磁盘资源

#### 1NF(数据库表格字段不可拆分)

下列中，用户信息字段可以拆分，不满足1NF

|  |  |
| --- | --- |
| name | 用户信息 |
| 王翠花 | 18716531678, 69887653, 北京 |

#### 2NF(非主键字段依赖全部主键字段，不能部分依赖)

下列中 学生名称只能是学生id决定，和学科id无关，即学生名称部分依赖于学科学生id主键

|  |  |  |  |
| --- | --- | --- | --- |
| 学生id | 学科id | 学生成绩 | 学生名称 |
| 1 | a | 98 | 王翠花 |
| 2 | a | 88 | 刘首付 |
| 3 | b | 99 | 吴有才 |
| 1 | b | 100 | 王翠花 |

完成2NF设计

|  |  |
| --- | --- |
| 学生id | 学生姓名 |
| 1 | 王翠花 |
| 2 | 刘首付 |
| 3 | 吴有才 |

|  |  |  |
| --- | --- | --- |
| 学生id | 学科id | 学生成绩 |
| 1 | a | 98 |
| 2 | a | 88 |
| 3 | b | 99 |
| 1 | b | 100 |

#### 3NF

表格中,非主键字段必须依赖主键字段不能依赖其他的非主键字段

学院名称依赖于非主键字段学院id，不满足3NF

|  |  |  |  |
| --- | --- | --- | --- |
| 学生id | 学院id | 学生姓名 | 学院名称 |
| 1 | a | 王翠花 | 物理系 |
| 2 | b | 刘首付 | 化学系 |

### 反范式

可以节省查询时间，牺牲磁盘空间

* + 背景：用户增加，硬件资源成本变低。提升查询速度。

违反三范式的设计,出现了冗余字段,节省查询时间,系统磁盘空间

# 数据库的主从复制(高可用)

### 结构

计算机生成了可选文字:
masterl
访问集群数据库客户端软件
数据分片计算
m2
m3

### 原理

计算机生成了可选文字:
clientl
client2
client
MASTER
select
upae
delete
insert
drop
create。
p=387
p=467
mon
fetch
SLAVE
中继日志
p=123
0
p=239
0
SQL

**主节点（master）**：开启二进制日志文件，记录所有master客户端在master执行的写(insert、update、delete等)操作命令，并且在二进制文件中以Position值记录最后的位置。

**从节点（slave）**：开启IO线程、SQL线程、中继日志

**IO线程**：登录到主节点（master），监听具体的日志文件，通过上次抓取后的记录Position判断文件中是否有更新内容，如果有，则抓取更新数据，存放到本地中继日志

**中继日志**：存放IO线程更新抓取过来的所有主节点二进制命令数据

**SQL线程：**监听本地中继日志，一旦有更新，拿到更新命令在从节点执行**最终主从节点由于上述的同步逻辑保持数据的同步复制**

# 数据库的主从复制实现(高可用)

### 介绍一下基本命令(Linux)

#### service mysql start：启动mysql

#### service mysql stop：停止mysql

#### service mysql restart：重新启动mysql

### 主从数据挂接

#### 修改uuid

如果二台主机的mysql是通过镜像拷贝的，那么uuid一样肯定不能搭建主从集群，所以需要修改

保证mysql服务是关闭的：

|  |
| --- |
| [root@10-42-175-170 redis-3.2.11]# vim /var/lib/mysql/auto.cnf |
|  |

修改的uuid必须是计算方法生成的正确值,不能随便改动,可以从数据库表格使用各种id的值(JAVA UUID生成的)

重启mysql服务,使得新的uuid生效,否则相同uuid无法使用.

|  |
| --- |
| service mysql restart |

#### 主节点开启二进制日志

打开配置文件

|  |
| --- |
| [root@10-42-175-170 redis-3.2.11]# vim /etc/my.cnf |
|  |

添加二行，主从server-id不能一样，easymall-log二进制日志文件名称

|  |
| --- |
| #数据库主从复制配置  server-id=1  log-bin=easymall-log #记录操作日志 |
|  |

重启主节点Mysql

|  |
| --- |
| service mysql restart |

查看配置信息

|  |
| --- |
| show master status; |
|  |

**file**：mysql-log.000001 二进制日志文件名称,后缀是一个自增的数字,当二进制文件非常大,滚动成多个000002/000003.重启服务时会滚动数字.

**position**: 初始化的指针值是120,对当前数据库执行一些写的命令时,postion变动，这里为530，说明做了些非查询操作

**---至此已完成主节点配置**

#### 从节点

##### 配置文件

打开配置文件

|  |
| --- |
| [root@10-42-147-110 redis-3.2.11]# vim /etc/my.cnf |

添加二行，主从server-id不能一样，从节点不需要开启二进制日志文件

|  |
| --- |
| #数据库主从复制配置  server-id=2 |
|  |

重启mysql

|  |
| --- |
| [root@10-9-48-69 redis-3.2.11]# service mysql restart  Shutting down MySQL (Percona Server).... SUCCESS!  Starting MySQL (Percona Server). SUCCESS! |

##### 挂接到主节点

登录到从节点mysql

执行命令：相关信息可登录主节点mysql 执行show master status查看

|  |
| --- |
| CHANGE MASTER TO  MASTER\_HOST='10.9.39.13',#主节点ip  MASTER\_PORT=3306, #主节点mysql端口  MASTER\_USER='root', #主节点mysql用户  MASTER\_PASSWORD='root', #主节点mysql密码  MASTER\_LOG\_FILE='easymall-log.000001', #主节点二进制日志文件名 由从节点IO线程监听  MASTER\_LOG\_POS=120#主节点当前日志文件的Position |

查看挂接状态 \G表示竖向显示

|  |
| --- |
| mysql>show slave status\G; |
|  |

可以看到IO线程和SQL线程为关闭状态，执行下面命令开启

|  |
| --- |
| mysql>start slave; |
|  |

#### 常见问题

**启动从节点没有成功,IO线程么有成功**

**原因**:

Fatal error: The slave I/O thread stops because master and slave have equal MySQL server UUIDs; these UUIDs must be different for replication to work.

当前2个服务器的mysql服务,uuid(id)生成是相同的.所以不同添加到同一个主从中.

镜像安装完整的mysql服务,uuid安装镜像就确定,复制镜像给所有人同步镜像内容,uuid相同.

**解决办法**:

将2个服务器的uuid值,变的不一样

**io状态显示Connecting to master**

**原因**:

连接主节点提供的属性(从节点挂接主节点)ip:port user password是不正确,修改正确.

* 停止从节点线程io和sql mysql>stop slave;
* 重新运行change master to 的命令确保参数正确
* 重新mysql> start slave

### 主从复制测试

测试主从数据复制,通过非法操作,理解主从同步原理.

#### 主节点写数据测试，观察从节点是否同步

主节点测试创建数据库、表、插入数据等从节点也会相应操作

#### 非法操作、理解原理

从节点插入数据 id 为主键，从节点插入数据肯定不会同步到主节点

|  |
| --- |
| mysql> insert into test values(1,'shu');  Query OK, 1 row affected (0.00 sec)  mysql> select \* from test;  +------+------+  | id | name |  +------+------+  | 1 | shu |  +------+------+  1 row in set (0.00 sec)  mysql> |

主节点插入相同主键数据，主节点新增成功，没有错误提示。从节点已有数据肯定不成功

|  |
| --- |
| mysql> insert into test values(1,'shu');  Query OK, 1 row affected (0.00 sec)  mysql> |

再次在主节点插入正常数据

|  |
| --- |
| mysql> insert into test values(2,'shu');  Query OK, 1 row affected (0.00 sec)  mysql> |

主节点是否有数据？没有

|  |
| --- |
| mysql> select \* from test;  +----+------+  | id | name |  +----+------+  | 1 | shu |  +----+------+  1 row in set (0.00 sec) |

为什么正常数据也不能插入了？

|  |
| --- |
| 计算机生成了可选文字: master insert4陈老师 insert5吴老师 inset6刘老师 insert4陈老师 insert5吴老师 inset6刘老师 Slave 4王老师 insert4陈老师 insert5吴老师 inset6刘老师 insert SQL insert4王老师 duplicatentry4 主键重复 |

**原因**：从节点由于写了数据存在一个 id=1的行数据,主节点操作时同步过来的insert语句id也等于1，然后无法正确执行,导致SQL线程运行停止(可show slave status查看为no).后续在继续同步过来的sql也无法执行

**解决**：将从节点的错误数据id=1的行数据删除,重启sql线程(stop slave start slave)

**结论**:主从mysql结构中,虽然mysql从技术角度不拦着你在从节点写数据,但是从应用角度不能再从节点执行写逻辑,只能进行读操作.

# 高可用数据库集群

### 故障转移

redis就接触过故障转移.数据库中没有哨兵,将主从角色进行转化.

通过代码逻辑,将客户端连接主节点进行写操作,一旦节点宕机不可达,客户端重新连接到从节点---故障转移.

### 读写分离

如果搭建了复杂的主从分布式高可用用集群,不断的通过对主节点操作,实现数据的增删查改,从节点显得有些冗余了,浪费资源.很多客户端软件都能实现高可用数据库集群中的读写分离.有效的提升数据库集群的使用效率.

主节点:主要负责写,也可以负责一部分的读操作

从节点:只能负责读操作

### 双击备份

实现单向主从备份结构中,并不能在实际的应用场景提供多次的故障转移.

|  |
| --- |
| 计算机生成了可选文字: master lo-bin Slave y-log re Slave 王五 master 王五 log-bin |

将原有主从进行反向的主从挂接就可以了.(原有的主节点同时作为从节点挂接到原有的从节点，参考主从复制实现)

* 主节点开启二进制,查看二进制详细信息 名字和pos
* 从节点change master to挂接主节点
* 从节点启动2个线程start slave

即使,我们搭建出了一个复杂的高可用数据库集群,对于使用这个集群的客户端系统(比如EASYMALL)

必须考虑如何实现使用故障转移,如何实现数据的分片计算.

--管理一个高可用分布式的数据库集群,引入新的技术--数据库中间件(mycat)