数据库设计

数据库优化的理由

收益：阿姆达尔定理

数据库单点压力：

积累压力：

数据库设计

1。先设计数据表的关系，再推导使用对象。

2。反之。

设计别扭的时候。1-2，3-2。

数据冗余：

数据不一致：

插入异常：删除异常。

数据库范式。等级越高、越严格。

反范式，建议 在范式的基础上进行修正，减少范式带来的负面影响。

第一范式：每个字段都是原子的，不能再拆分。

反例：json。先看设计是否合理，再选合适的存储组件。

第二范式：必须有主键（可以是一个，或 多个字段），非主属性，必须依赖于主键。

反例：好友关系列表。主键：关注人id，被关注人Id，关注人头像。

第三范式：没有传递依赖，非主属性直接依赖于主键，而不能间接依赖于主键。

反例:员工表：员工id。部门id，部门名称。部门简介。

反范式：字段冗余。

如果你的系统是重业务的系统，对性能和并发的要求不高，最好用第三方范式。

如果系统对某些查询较频繁，可以考虑冗余，反范式。

巨量数据的优化

数据量增加---》响应时间的增加。

常用的方法：建索引。mysql性能调优。

表分区->分库分表->读写分离

表分区

表。每个表对应磁盘上的一个文件。ibd。data目录下

分流：对一个ibd文件的请求，分发到对多个ibd文件的请求。

将数据 多个物理表进行存储，逻辑上，还是一张表。

好处：

1。 当查询条件 可以判定 某个数据位于哪个分区，那么直接在分区中查询，不用扫描整个表。

2。业务代码不用改。

3。分区进行单独管。备份，恢复。

range。0-10，11-20。

partition by range

list: partition by list xxxxxxxxxxx values in (1,2)

hash:partion by hash

key

做分区的注意事项：

1。 做分区时，要结合查询规则。尽量保证常用查询落到一个分区中。

举例：大数据量： 学生姓名，老师姓名。如果以学生姓名作为条件查询较多。以学生姓名分区。

2。分区条件放到where语句中。比如上面的学生姓名。

坏处：如果出现跨区查询，适得其反。

分库分表

目的：

1。业务拆分。

2。通过分库分表应对高并发。需要看场景：读多写少（从库，缓存）？读少写多？

3。数据隔离。将核心业务和非核心业务进行拆分。

分库分表的设计

一个库里有两张表a，b。

如何拆？

1。不破坏表。a放一个库，b放一个库。

2。破坏表。垂直分 和 水平分

垂直分：

字段1，字段2，字段3，字段4。

数据条数不变，字段数变少。

场景：

水平分：

字段1，字段2，字段3，字段4。用户表：性别，年龄，姓名，介绍。

1

2

3

4

婚恋。

额外的操作：

路由操作：

拼接操作：

开发中：遇到问题，先用简单办法解决，解决不了，再复杂。

分库分表的方法

范围路由：选取 有序的列，按照一定的范围去划分。

分段小：导致子表数量多，增加维护难度，分段大：有可能，单表依然存在性能问题。

分的依据：分表后，表的各方面性能，能否满足系统要求。

优点：可以平滑的扩充新表，只需增加子表的数据量，原有的数据不用动。

缺点：数据分布不均匀。

hash路由：选取 列 ，进行hash运算。 %10 = 。

优点：相反，

缺点：相反。

10， 11 --1 。

11，11--0

分库分表的问题

分布式id的问题：

需要 全局id生成服务。

注意点：全局唯一（时间戳，机器号，序列号）；高性能；高可用；易使用，好接入；

uuid，数据库自增，号段，redis自增，雪花算法，TinyID，UIDGenerator。leaf。

有独立的课。

拆分维度的问题：电商系统-订单表：，用户id，订单id，商品id。

依据：自己业务的情况。看那个字段用的多？

索引表、映射表：（过渡方案）

dts，datax：增量同步。

join问题：

商品、订单、用户。单库join没问题，多库join失败。

1。在代码层面做join。

2。es。canal --》es。

目前：数据库越简单越好。 禁止3张表 做关联，禁用存储过程，禁用触发器。

事务事务：

XA，jar包。不好用。shardingsphere.

成本问题：

非必要 不分库。不要过度设计。

1亿 1张表，10ms。100w1张表 1min。

读写分离

1个库。

2年后：10个库。

4年后：又慢了。

10个库里的每一个 做 读写分离。（ 前提 ）

分流：

数据库锁（分）对数据库并发有影响。

X锁-写锁，只能我一人，读、写 干不了。

S锁-读锁，

场景：读多写少。不适用：读少写多（反而用途不大）。

防止阻塞。

一主（写）多从（读）

路由问题：select S 锁 从库 ; create update delete add. X 锁 写库。 mybatis插件。

主从复制的问题：

1 写主库，如何 高效 同步到从库。

从库用sql命令写，会失去读写分离的意义。因为 回到了从前。

主库：binlog，传给从库，写入从库：的relayLog，解析relaylog。重现数据。

base：

2 会有时间差。

（一）

编造业务场景：注册（主库），登录（查 从库）。

将注册后的第一次 查询 ，指到 主库上。

比如说：注册完，写个redis，key 用户id 1。删掉key。

和业务耦合大。

lich：我是先从库查，返回空后，再查一次主库，如果没有的话，那就真的没有了。-- 个别业务用。

（二）

主业务用主库，非主业务用从库。

用户系统，注册登录，都走主库， 用户介绍、xxx 走从库。

实现

分库分表，读写分离。

读写分离：将读和写，分开。

分库分表：将 数据 分库。

分-

分配机制-

将不同的sql 分发到不同的机器上。

select ----某台机器上

update --某台机器上。

where id = 1 某台机器上

where id = 2 某台机器上。

拦截，判断，分发。

代码封装：dao抽象一层。夹层。tddl(Taobao Distributed Data Layer) 头都大了。

Shardingjdbc。需要改代码。

中间件封装：mycat。