# 数据库篇

## 数据库的主从同步读写分离:

当时我们给数据库也做了优化，配置了主从同步和通过mycat进行读写分离；MYSQL主从同步架构是目前使用最多的数据库架构之一，尤其是负载比较大的网站。  
原理是：从服务器的IO线程到主服务器获取二进制日志，并在本地保存为中继日志，然后通过SQL线程来在从库上执行中继日志中的内容，从而使从库和主库保持一致。配置好主从同步之后我们还通过mycat来配置了读写分离；读写分离：数据分布,负载均衡,高可用性和容错,备份.  
使用：就是我们搭建两台数据库服务器，一台作为主服务器master，一台作为从服务器slave,  
1.要更改mysql数据库中的my-defult.ini文件，将文件名改为my.ini,然后更改里面的配置，包括数据库路径，端口号。编码集、log-bin文件名，开放的数据库等信息。  
2.在主数据库，新增一个用户，并指定replication权限  
3.在主数据库里面运行show master status;记下file和position字段对应的参数。  
4.在从库设置它的master  
5. stop slave; #停止服务再输入#指定用户名密码，将里面的master\_log\_file和master\_log\_pos对应刚才show master status记下的参数  
6.开启服务，start slave   
7. 通过show slave status来查看同步的状态  
这样就实现了主从同步。

## 主从同库延迟问题？

1.主库和同库尽可能在同一个内，交换机网卡采用千兆网卡。  
2.主数据库更新完成之后产生的操作日志不是瞬间产生的，我们可以通过设置sync\_binlog=1, 让它瞬间产生磁盘日志，（n=1指主数据库只要操作一次，就产生一次磁盘日志，n=10，就是操作10次，产生一次），从数据库可以依赖磁盘日志，瞬间产生同步，可以达到减低延迟的效果。  
3. 设置主库和从库读取日志失败之后，及时重新建立连接，延迟缩短。

## Sql优化:

避免在 where 子句中对有索引的字段进行运算,这会导致索引失效，从而进行全表扫描。外键必须加索引

在 where 及 order by 涉及的列上建立索引，要尽量避免全表扫描。

在设计表时要避免表中字段出现null的情况，通常要为其设置默认值。

避免在查找时放弃使用索引而进行全表扫描。

Sql语句字段尽量大写。

SELECT语句中避免使用'\*’，只查询需要返回的字段 ，这样可以减少oracle解析sql语句的时间。

## 索引、存储过程、函数语法及各自的优缺点：

索引

索引是表的索引目录，在查找内容之前先查目录中查找索引位置，从而快速定位查询数据；

a、普通索引

# 创建表 + 索引

create table in1(

nid int not null auto\_increment primary key,

name varchar(32) not null,

email varchar(64) not null,

extra text,

index ix\_name (name)

)

# 创建索引

create index index\_name on table\_name(column\_name)

# 删除索引

drop index\_name on table\_name;

# 查看索引

show index from table\_name;

#注意：对于创建索引时如果是BLOB 和 TEXT 类型，必须指定length。

create index ix\_extra on in1(extra(32));

如何正确使用索引

# like '%xx'，避免%\_写在开头

select \* from tb1 where name like '%n';

存储过程：

存储过程是一个SQL语句集合，类似函数，需要主动调用。

1创建存储过程

# 无参数存储过程

# 创建存储过程

delimiter //

create procedure p1()

BEGIN

select \* from t1;

END//

delimiter ;

# 执行存储过程

call p1()

都说了类似函数，那必须得可以接收参数，且参数有三类：

in 仅用于传入参数用

out 仅用于返回值用

inout 既可以传入又可以当作返回值

# 有参数存储过程

# 创建存储过程

delimiter \\ # 结尾分号改为\\

create procedure p1(

in i1 int,

in i2 int,

inout i3 int,

out r1 int

)

BEGIN

DECLARE temp1 int; # 创建申明局部变量

DECLARE temp2 int default 0;

set temp1 = 1;

set r1 = i1 + i2 + temp1 + temp2;

set i3 = i3 + 100;

end\\

delimiter ;

# 执行存储过程

DECLARE @t1 INT default 3;

DECLARE @t2 INT;

CALL p1 (1, 2 ,@t1, @t2);

SELECT @t1,@t2;

函数

函数，与存储过程不同的是有return值。

1自定义函数

delimiter \\

create function f1(

i1 int,

i2 int)

returns int

BEGIN

declare num int;

set num = i1 + i2;

return(num);

END \\

delimiter ;

索引优缺点：

优点： 1.创建唯一索引，保证数据库表中每一行数据的唯一性

2.大大加快数据的检索速度，这也是创建索引的最主要原因

3.减少磁盘IO(像字典一样可以直接定位)

缺点: 1.创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加

2.索引需要占用额外的物理空间

3.当对表中的数据进行增加、删除和修改的时候，索引也要动态维护，降低了数据的维护速度

存储过程 优缺点

优点： 1.简化了复杂的业务逻辑，根据需要可以重复使用

2.屏蔽了底层细节，不暴露表结构即可完成操作

3.降低网络通信量，多条语句可以封装成一个存储过程来执行

4.设置访问权限来提高安全性

5.提高执行效率，因为它是预编译的以及存储在数据库中

缺点； 1.可移植性差，相同的存储过程并不能跨多个数据库操作

2.大量使用存储过程后，首先会造成服务器压力大，而且维护难度页逐渐增加

函数 优缺点:

优点： 1.函数允许标准组件式编程，提高了SQL语句的重用性、共享性和可移植性。

2.函数可以被作为一种安全机制来利用。

3.函数能够实现较快的执行速度，能够减少网络流量。

缺点： 1.函数的编写比单句SQL语句复杂。

2.在编写函数时，需要创建这些数据库对象的权限。

## 分区和分表的区别：

分表就是按照一定的规则把一张大表给分成多张小表，在数据库中看到的是几张不同的表，在硬盘上也是不同的文件，因此在进行增删改查操作的时候要根据同样的规则去找到具体操作的表。

分区是按照一定的规则把一张大表分成不同的区块，在数据库中看到的还是一张表，只不过在硬盘上存储的时候分成了几个不同的文件。这样在进行增删改查操作的时候就像操作一张普通的表一样简单方便。

他们都是为了处理单张表的大数据量问题，都能提高性能。

## 防sql注入

针对防sql注入，我们通常是这样做的：

首先在前台页面对用户输入信息进行js验证，对一些特殊字符进行屏蔽， 比如：or ,单引号，--，= ，还有就是限制用户名输入的长度，我们一般将其限制在6---13位。另外，对于用户的敏感信息我们进行Md5加密，还有，为了增加用户体验度和用户友好度，为了不使用户看到一些详细的异常信息我们会进行错误信息页面的定制，像404,500错误。另一个我层面讲，这样做也是为了保护我们的一些重要信息。此外，我们会给特定的人分配定定的权限，而不是给其分配管理员权限！

## 数据库连接池：

数据库连接池的优点运行原理:

在我们不使用数据库连接池的时候，每次访问数据库都需要创建连接，

使用完成之后需要释放关闭连接，而这样是很耗费资源的。当我们使用

数据库连接池的时候，在tomcat启动的时候就创建了指定数量的连接，

之后当我们程序使用的时候就直接从连接池里面取，而不需要创建，同理，

当我们使用完的时候也不需要关闭连接，而是将连接返回到连接池中，供其他请求继续使用。DBCP：比较稳定。C3P0: 性能比较高。

## 索引概述

索引的优点

　1.创建唯一性索引，保证数据库表中每一行数据的唯一性

　2.大大加快数据的检索速度

　　 3.减少磁盘IO（向字典一样可以直接定位）

索引的缺点

1.创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加

2.索引需要占用额外的物理空间

3.当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，降低了数据的维护速度

## 数据库三范式：

第一范式（1NF）：强调的是列的原子性，即列不能够再分成其他几列。

第二范式（2NF）: 首先是满足第一范式，另外包含两部分内容，一是表必须有一个主键，二是没有包含在主键中的列必须完全依赖于主键，而不是部分依赖。

第三范式（3NF）: 首先满足第二范式，非主键列直接依赖于主键，消除传递依赖。