**MySQL数据库优化专题**

# 存储过程

## 什么是存储过程

简单的说，就是一组SQL语句集，功能强大，可以实现一些比较复杂的逻辑功能，类似于JAVA语言中的方法；

ps:存储过程跟触发器有点类似，都是一组SQL集，但是存储过程是主动调用的，且功能比触发器更加强大，触发器是某件事触发后自动调用；

## 存储过程有哪些特性

有输入输出参数，可以声明变量，有if/else, case,while等控制语句，通过编写存储过程，可以实现复杂的逻辑功能；

函数的普遍特性：模块化，封装，代码复用；

速度快，只有首次执行需经过编译和优化步骤，后续被调用可以直接执行，省去以上步骤；

## 创建一个存储过程

|  |
| --- |
| create procedure user\_porced ()  begin  select name from users;  end; |

## 调用存储过程

|  |
| --- |
| call porcedureName (); |

## 传参存储过程

|  |
| --- |
| create PROCEDURE user\_porcedPa(  in a int(10)  )  BEGIN  select \* from users where age>a;  END;  call user\_porcedPa(10); |

## 存储过程优缺点

优点

1.在生产环境下，可以通过直接修改存储过程的方式修改业务逻辑（或bug），而不用重启服务器。但这一点便利被许多人滥用了。有人直接就在正式服务器上修改存储过程，而没有经过完整的测试，后果非常严重。

2.执行速度快。存储过程经过编译之后会比单独一条一条执行要快。但这个效率真是没太大影响。如果是要做大数据量的导入、同步，我们可以用其它手段。

3.减少网络传输。存储过程直接就在数据库服务器上跑，所有的数据访问都在服务器内部进行，不需要传输数据到其它终端。但我们的应付服务器通常与数据库是在同一内网，大数据的访问的瓶颈会是硬盘的速度，而不是网速。

4.能够解决presentation与数据之间的差异，说得文艺青年点就是解决OO模型与二维数据持久化之间的阻抗。领域模型和数据模型的设计可能不是同一个人（一个是SA，另一个是DBA），两者的分歧可能会很大——这不奇怪，一个是以OO的思想来设计，一个是结构化的数据来设计，大家互不妥协——你说为了软件的弹性必须这么设计，他说为了效率必须那样设计，为了抹平鸿沟，就用存储过程来做数据存储的逻辑映射（把属性映射到字段）。好吧，台下已经有同学在叨咕ORM了。

5.方便DBA优化。所有的SQL集中在一个地方，DBA会很高兴。这一点算是ORM的软肋。不过按照CQRS框架的思想，查询是用存储过程还是ORM，还真不是问题——DBA对数据库的优化，ORM一样会受益。况且放在ORM中还能用二级缓存，有些时候效率还会更高。

缺点

1.SQL本身是一种结构化查询语言，加上了一些控制（赋值、循环和异常处理等），但不是OO的，本质上还是过程化的，面对复杂的业务逻辑，过程化的处理会很吃力。这一点算致命伤。

2.不便于调试。基本上没有较好的调试器，很多时候是用print来调试，但用这种方法调试长达数百行的存储过程简直是噩梦。好吧，这一点不算啥，C#/java一样能写出噩梦般的代码。

3.没办法应用缓存。虽然有全局临时表之类的方法可以做缓存，但同样加重了数据库的负担。如果缓存并发严重，经常要加锁，那效率实在堪忧。

4.无法适应数据库的切割（水平或垂直切割）。数据库切割之后，存储过程并不清楚数据存储在哪个数据库中。

5.精通SQL的新手越来越少——不要笑，这是真的，我面试过N多新人，都不知道如何创建全局临时表、不知道having、不知道聚集索引和非聚集索引，更别提游标和提交叉表查询了。好吧，这个缺点算是凑数用的，作为屌丝程序员，我们的口号是：没有不会的，只有不用的。除了少数有语言洁癖的人，我相信精通SQL只是时间问题。

# MySQL如何优化

表的设计合理化(符合3NF)

添加适当索引(index) [四种: 普通索引、主键索引、唯一索引unique、全文索引]

SQL语句优化

分表技术(水平分割、垂直分割)

读写[写: update/delete/add]分离

存储过程 [模块化编程，可以提高速度]

对mysql配置优化 [配置最大并发数my.ini, 调整缓存大小 ]

mysql服务器硬件升级

定时的去清除不需要的数据,定时进行碎片整理(MyISAM)

# 数据库设计

## 什么是数据库范式

为了建立冗余较小、结构合理的数据库，设计数据库时必须遵循一定的规则。在关系型数据库中这种规则就称为范式。范式是符合某一种设计要求的总结。要想设计一个结构合理的关系型数据库，必须满足一定的范式。

## 数据库三大范式

第一范式：1NF是对属性的原子性约束，要求属性(列)具有原子性，不可再分解；(只要是关系型数据库都满足1NF)

第二范式：2NF是对记录的惟一性约束，表中的记录是唯一的, 就满足2NF, 通常我们设计一个主键来实现，主键不能包含业务逻辑。

第三范式：3NF是对字段冗余性的约束，它要求字段没有冗余。 没有冗余的数据库设计可以做到。

但是，没有冗余的数据库未必是最好的数据库，有时为了提高运行效率，就必须降低范式标准，适当保留冗余数据。具体做法是： 在概念数据模型设计时遵守第三范式，降低范式标准的工作放到物理数据模型设计时考虑。降低范式就是增加字段，允许冗余。

# SQL优化

如何从一个大项目中，迅速的定位执行速度慢的语句. (定位慢查询)

### show status

使用show status使用show status查看MySQL服务器状态信息

常用命令

|  |
| --- |
| --mysql数据库启动了多少时间  show status like 'uptime'; |
| show stauts like 'com\_select' show stauts like 'com\_insert' ...类推 update delete（显示数据库的查询，更新，添加，删除的次数） |
| show [session|global] status like .... 如果你不写 [session|global] 默认是session 会话，指取出当前窗口的执行，如果你想看所有(从mysql 启动到现在，则应该 global) |
| //显示到mysql数据库的连接数  show status like 'connections '; |
| //显示慢查询次数  show status like 'slow\_queries'; |

### 慢查询

#### 什么是慢查询

MySQL默认10秒内没有响应SQL结果,则为慢查询

可以去修改MySQL慢查询默认时间

#### 如何修改慢查询

|  |
| --- |
| --查询慢查询时间  show variables like 'long\_query\_time';  --修改慢查询时间  set long\_query\_time=1; ---但是重启mysql之后，long\_query\_time依然是my.ini中的值 |

#### 如何定位慢查询

#### 初始化测试数据

##### 创建表结构

|  |
| --- |
| /\*部门表\*/  CREATE TABLE dept(  deptno MEDIUMINT UNSIGNED NOT NULL DEFAULT 0, /\*编号\*/  dname VARCHAR(20) NOT NULL DEFAULT "", /\*名称\*/  loc VARCHAR(13) NOT NULL DEFAULT "" /\*地点\*/  ) ENGINE=MyISAM DEFAULT CHARSET=utf8 ;  /\*员工表\*/  CREATE TABLE emp  (empno MEDIUMINT UNSIGNED NOT NULL DEFAULT 0, /\*编号\*/  ename VARCHAR(20) NOT NULL DEFAULT "", /\*名字\*/  job VARCHAR(9) NOT NULL DEFAULT "",/\*工作\*/  mgr MEDIUMINT UNSIGNED NOT NULL DEFAULT 0,/\*上级编号\*/  hiredate DATE NOT NULL,/\*入职时间\*/  sal DECIMAL(7,2) NOT NULL,/\*薪水\*/  comm DECIMAL(7,2) NOT NULL,/\*红利\*/  deptno MEDIUMINT UNSIGNED NOT NULL DEFAULT 0 /\*部门编号\*/  )ENGINE=MyISAM DEFAULT CHARSET=utf8 ;  /\*薪水\*/  CREATE TABLE salgrade  (  grade MEDIUMINT UNSIGNED NOT NULL DEFAULT 0,  losal DECIMAL(17,2) NOT NULL,  hisal DECIMAL(17,2) NOT NULL  )ENGINE=MyISAM DEFAULT CHARSET=utf8;  /\*测试数据\*/  INSERT INTO salgrade VALUES (1,700,1200);  INSERT INTO salgrade VALUES (2,1201,1400);  INSERT INTO salgrade VALUES (3,1401,2000);  INSERT INTO salgrade VALUES (4,2001,3000);  INSERT INTO salgrade VALUES (5,3001,9999); |

##### 创建函数

|  |
| --- |
| create function rand\_string(n INT)  returns varchar(255) #该函数会返回一个字符串  begin  #chars\_str定义一个变量 chars\_str,类型是 varchar(100),默认值'abcdefghijklmnopqrstuvwxyzABCDEFJHIJKLMNOPQRSTUVWXYZ';  declare chars\_str varchar(100) default  'abcdefghijklmnopqrstuvwxyzABCDEFJHIJKLMNOPQRSTUVWXYZ';  declare return\_str varchar(255) default '';  declare i int default 0;  while i < n do  set return\_str =concat(return\_str,substring(chars\_str,floor(1+rand()\*52),1));  set i = i + 1;  end while;  return return\_str;  end  create FUNCTION rand\_num()  RETURNS int(5)  BEGIN  DECLARE i int default 0;  set i =floor(10+RAND()\*500);  return i;  END |

##### 创建存储过程

|  |
| --- |
| delimiter $$  create procedure insert\_emp(in start int(10),in max\_num int(10))  begin  declare i int default 0;  #set autocommit =0 把autocommit设置成0  set autocommit = 0;  repeat  set i = i + 1;  insert into emp values ((start+i) ,rand\_string(6),'SALESMAN',0001,curdate(),2000,400,rand\_num());  until i = max\_num  end repeat;  commit;  end $$  执行存储过程  call insert\_emp (100001,40000000); |

#### 如何将慢查询定位到日志中

在默认情况下，我们的mysql不会记录慢查询，需要在启动mysql时候，指定记录慢查询才可以

bin\mysqld.exe --safe-mode --slow-query-log [mysql5.5 可以在my.ini指定]（安全模式启动，数据库将操作写入日志，以备恢复）

bin\mysqld.exe –log-slow-queries=d:/abc.log [低版本mysql5.0可以在my.ini指定]

先关闭mysql,再启动, 如果启用了慢查询日志，默认把这个文件放在

my.ini 文件中记录的位置

#Path to the database root

datadir=" C:/ProgramData/MySQL/MySQL Server 5.5/Data/"