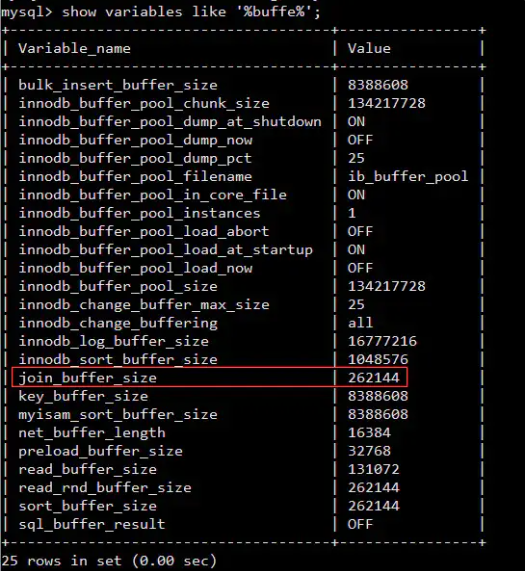
# 原理

join语句是相对比较耗费性能，因为在执行join语句的时候必然要有一个比较的过程。

逐条比较两个表的语句是比较慢的，因此我们可以把两个表中数据依次读进一个内存块中, 以MySQL的InnoDB引擎为例，使用以下语句我们必然可以查到相关的内存区域show variables like '%buffer%'：



join\_buffer\_size的大小将会影响我们join语句的执行性能。

大部分数据库中的数据最终要保存到硬盘上,并且以文件的形式进行存储。我们有多少表要连接就需要读多少个文件，虽然可以利用索引，但还是免不了频繁的移动硬盘的磁头。也就是说频繁的移动磁头会影响性能。

# 分类

SQL标准中Join的类型包括5种：

内连接（INNER）

全外连接（FULL OUTER）

左外连接（LEFT OUTER）

右外连接（RIGHT OUTER）

交叉连接（CROSS）

为了测试不同join的使用场景，以下面表结构为例：

西天取经四人组：

|  |  |
| --- | --- |
| id | user\_name |
| 1 | 唐僧 |
| 2 | 猪八戒 |
| 3 | 孙悟空 |
| 4 | 沙僧 |

孙悟空的朋友们：

|  |  |
| --- | --- |
| id | user\_name |
| 1 | 孙悟空 |
| 2 | 牛魔王 |
| 3 | 蛟魔王 |
| 4 | 鹏魔王 |
| 5 | 狮驼王 |

## 内连接

内连接（inner join）基于连接谓词将两张表（如A和B）的列组合在一起，产生新的结果集（即A和B的公共部分）。



举例SQL：

SELECT a.`user\_name`,`a.`over`,b.`over`

FROM user1 a

INNER JOIN user2 b

ON a.`user\_name`=`b.`user\_name`

;

## 左外连接

左外连接（left outer join）又称左连接，比如A，B两表左连接，则结果以A表为依据，包含A的所有结果。

下面是左连接的两种应用方式：



图1表示包含A的结果集，图2含有where条件，排除属于A但是不属于B的情况（不属于B即为NULL）。

对于图2这种查询也可以使用NOT IN，但是不推荐使用NOT IN子查询，因为NOT IN不会使用索引。

举例SQL：

查询取经四人中哪些人不是悟空的结拜兄弟？

SELECT a.`user\_name`,a.`over`,b,`over`

FROM user` a

LEFT JOIN user2 b

ON a.`user\_name`=b.`user\_name`

WHERE b.user\_name IS NULL

;

## 右外连接

右外连接（right outer join）又称右连接，结果集包含右侧的表。



举例SQL：

查询悟空的结拜兄弟中哪些人没有去取经？

SELECT b.`user\_name`,b.`over`,a.`over`

FROM user1 a

RIGHT JOIN user2 b

ON a.`user\_name`=b.`user\_name`

WHERE a.user\_name IS NULL

;

## 全连接

Full join其实就是左连接和右连接的结合，应用方式包括2种：



图1是全连接，图2含有where条件查询，去除A，B的公共部分。

但是MySQL本身是不直接支持full join的：

SELECT\*

FROMuser1 a

FULL JOIN

User2 b on a.user\_name=b.user\_name

;

客户端报错：

ERROR 1064（42000）：you have an error in your SQL syntax;

check the manual that corresponds to your MySQL server version for that right sytax to user near full join.

可以采用如下的方式解决（left和right join做union all）：

SELECT a.`user\_name`,a.`over`,b.`over`

FROM user1 a

LEFT JOIN user2 b ON a.`user\_name`=b.`user\_name`

**UNION ALL**

SELECT b.`user\_name`,b.`over`,a.`over`

FROM user1 a

RIGHT JOIN user2 b ON a.`user\_name`=b.`user\_name`

;

## 交叉连接

交叉连接（cross join），又称笛卡尔连接（cartesian join）或叉乘（Product），如果A和B是两个集合，它们的交叉连接就记为：A×B。比如A表4条记录，B表5条记录，则结果为20条记录。

举例SQL：

SELECT a.`user\_name`,a.`over`,b.`user\_name`,b.`over`

FROM user1 a

CROSS JOIN user2 b

;

注：CROSS JOIN是不需要查询条件关键字的。

# JOIN算法

## Nested Loop Join

## Block nested loop

# SQL开发技巧

## 使用join更新表

如何更新使用过滤条件中包含自身的表？

情景：

把同时存在于取经四人组和悟空兄弟表中的记录的人在取经四人组表中的over字段更新为“齐天大圣”。

考虑使用这样的更新SQL：

UPDATE user1

SET over=”齐天大圣”

WHERE user1.`user\_name` IN(

SELECT b.`user\_name`

FROM user1 a INNER JOIN user2 b ON

a.`user\_name`=b.`user\_name`)

;

该语句在oracle和SQL Serve中是支持的，但是在MySQL中客户端报错：ERROR 1093（HY000）：you can`t sepecify target table in from clause.

既然在from子句中不允许这种操作，那么可以这样修改SQL：

UPDATE user1 a JOIN(

SELECT b.`user\_name`

FROM user1 a INNER JOIN user2 b ON

a.`user\_name`=b.`user\_name`

)b ON a.`user\_name`=b.`user\_name`

SET a.`over` = “齐天大圣”

;

## join优化子查询

SELECT a.`user\_name`

,a.`over`

,(SELECT over FROM user2 b

WHERE a.`user\_name`=b.`user\_name`) AS

OVER2

FROM user1 a

;

执行时间 ：



使用join优化子查询：

SELECT a.`user\_name`,a.`over`,b.`over` AS OVER2

FROM user1 a

LEFT JOIN user2 b ON

a.`user\_name`=b.`user\_name`

;

执行时间：



## join优化聚合子查询



问题：如何查询出四人组中打怪最多的日期？

SELECT a.`user\_name`,b.`timestr`,b.`kills`

FROM user1 a JOIN user\_kills b

ON a.id = b.user\_id

WHERE b,kills = (SELECT MAX(c,kills) FROM user\_kills c WHERE

c.user\_id = b.user\_id)

;

使用join优化：

SELECT a.`user\_name`,b.`timestr`,b.`kills`

FROM user1 a

JOIN user\_kills b ON a.id = b.user\_id

GROUP BY a.user\_name,b.timestr,b.kills

HAVING b.kills = MAX(c.kills)

;

# 分组选择数据

场景：

SELECT a.user\_name,b.timestr,b.kills

FROM user1 a JOIN user\_kills b

ON a.id = b.user\_id

WHERE user\_name = “孙悟空”

ORDER BY b.`kills` DESC

LIMIT 2

；

我们可以对每个人分别执行上面的查询。

问题：

1. 如果分类或是用户很多的情况下则需要多次执行同一查询
2. 增加应用程序同数据库的交互次数
3. 增加了数据库执行查询的次数，不符合批处理的原则
4. 增加了网络流量

我们的场景：

SELECT d.user\_name,c.timestr,kills

FROM (

SELECT user\_id,timestr,kills

,(SELECT COUNT(\*) FROM user\_kills b WHERE

b.user\_id=a.user\_id AND a.kills <= b.kills) AS cnt

FROM user\_kills a

GROUP BY user\_id,timestr,kills

) c JOIN user1 d ON c.user\_id=d.id

WHERE cnt<=2

;

# SQL优化

在项目开发中如果需要使用join语句，如何优化提升性能?

分为两种情况，数据规模小的，数据规模大的：

1. 数据规模较小，全部放进内存；
2. 数据规模较大。

可以通过增加索引来优化join语句的执行速度，可以通过冗余信息来减少join的次数，尽量减少表连接的次数，一个SQL语句表连接的次数不要超过5次。

# 注意事项

这个规则超过三张表禁止join ,由于数据量太大的时候，mysql根本查询不出来，导致阿里出了这样一个规定。(其实如果表数据量少，10张表也不成问题,你自己可以试试)而我们公司支付系统朝着大规模高并发目标设计的，所以，遵循这个规定。

在业务层面来讲，写简单sql，把更多逻辑放到应用层，我的需求我会更了解，在应用层实现特定的join也容易得多。