原 MYSQL explain详解

标签： WHERE子句用于限制哪一个行匹配下一个 如果Extra值不为Using wher 查询可能会有一些错误 如果想

分类： mysql (13)

explain显示了mysql如何使用索引来处理select语句以及连接表。可以帮助选择更好的索引和写出更优化的查询语句。

先解析一条sql语句，看出现什么内容

```
EXPLAINSELECTs.uid,s.username,s.name,f.email,f.mobile,f.phone,f.postalcode,f.address
FROM uchome_space ASs,uchome_spacefieldASf
WHERE 1
AND s.groupid=0
AND s.uid=f.uid
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	s	ALL	NULL	NULL	NULL	NULL	17022	Using where
1	SIMPLE	f	ALL	NULL	NULL	NULL	NULL	17039	Using where; Using join buffer

1. id

SELECT识别符。这是SELECT查询序列号。这个不重要,查询序号即为sql语句执行的顺序，看下面这条sql

```
EXPLAINSELECT*FROM(SELECT* FROMuchome_space LIMIT10)ASs
```

它的执行结果为

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	PRIMARY	<derived2>	ALL	NULL	NULL	NULL	NULL	10	
2	DERIVED	uchome_space	ALL	NULL	NULL	NULL	NULL	17022	

可以看到这时的id变化了

2.select_type

select类型，它有以下几种值

2.1 simple 它表示简单的select,没有union和子查询

2.2 primary 最外面的select,在有子查询的语句中，最外面的select查询就是primary,上图中就是这样

2.3 union union语句的第二个或者说是后面那一个.现执行一条语句，explain
select * from uchome_space limit 10 union select * from uchome_space limit 10,10

框架CI与YII之个人见解1 (3406)
jquery之闭包 实现无刷新 (1142)
php面试题（实时更新） (916)
MySQL的日常管理 (877)
mysql--复杂sql语句解析 (831)
MYSQL 优化数据库结构 (792)
php程序员之面试宝典（ (725)
linux常用实用命令详解 (528)
数据结构 (507)

评论排行

MYSQL explain详解 (5)
框架CI与YII之个人见解1 (1)
php 性能优化 (1)
JQUERY之传值 (0)
每日PHP函数1 (0)
Mysql 数据库优化6 (0)
Mysql 数据库优化5 (0)
Mysql 数据库优化4 (0)
Mysql 数据库优化2 (0)
linux常用命令3(进程管理 (0)

推荐文章

*Networking Named Content 全文翻译
* 边缘检测与图像分割
* 数据库性能优化之SQL语句优化
*阿里巴巴发布《2015移动安全漏洞年报》
* Java经典设计模式之七大结构型模式（附实例和详解）
*网络性能评价方法

最新评论

MYSQL explain详解
chenxicigema: 4.3.1 为什么是只有uchome_space一个表用到了eq_ref,并且sql语句如果变成 ...
MYSQL explain详解
lgslgs123456: 太多了 一时还看不太懂 先收藏起来 慢慢咀嚼
MYSQL explain详解
zhangxyzhangxy: 关于4.3.1的疑问：因为是两个表做join，所以肯定要遍历其中的一个表，然后用uchome_spa...
MYSQL explain详解
qq_20991785: 非常有帮助，对于各种情况讲的都很到位。
php 性能优化
yongsheng_xia: 第一条就错，php用的是写时复制，赋值并不会真的分配内存。另外PHP的opcode缓存真能提升那么高...
MYSQL explain详解
iaiti: 好详细。
框架CI与YII之个人见解1
guojikai: 确实如此，每个框架都有其操蛋的地方。CI的设计思路非常清晰，就是功能是在太弱了。整天都在造轮子了。

会有如下结果

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	PRIMARY	uchome_space	ALL	NULL	NULL	NULL	NULL	17022	
2	UNION	uchome_space	ALL	NULL	NULL	NULL	NULL	17022	
NULL	UNION RESULT	<union1,2>	ALL	NULL	NULL	NULL	NULL	NULL	

第二条语句使用了union

2.4 dependent union　UNION中的第二个或后面的SELECT语句，取决于外面的查询

2.5 union result　　UNION的结果,如上面所示

还有几个参数，这里就不说了，不重要

3 table

输出的行所用的表，这个参数显而易见，容易理解

4 type

连接类型。有多个参数，先从最佳类型到最差类型介绍 **重要且困难**

4.1 system

表仅有一行，这是const类型的特列，平时不会出现，这个也可以忽略不计

4.2 const

表最多有一个匹配行，const用于比较primary key 或者unique索引。因为只匹配一行数据，所以很快

记住一定是用到primary key 或者unique，并且只检索出两条数据的 情况下才会是const,看下面这条语句

explain SELECT * FROM `asj_admin_log` limit 1,结果是

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	asj_admin_log	ALL	NULL	NULL	NULL	NULL	72304	

虽然只搜索一条数据，但是因为没有用到指定的索引，所以不会使用const.继续看下面这个

explain SELECT * FROM `asj_admin_log` where log_id = 111

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	asj_admin_log	const	PRIMARY	PRIMARY	4	const	1	

log_id是主键，所以使用了const。所以说可以理解为const是最优化的

4.3 eq_ref

对于eq_ref的解释，mysql手册是这样说的:"对于每个来自于前面的表的行组合，从该表中读取一行。这可能是最好的联接类型，除了const类型。它用在一个索引的所有部分被联接使用并且索引是UNIQUE或PRIMARY KEY"。eq_ref可以用于使用=比较带索引的列。看下面的语句

explain select * from uchome_spacefield,uchome_space where uchome_spacefield.uid = uchome_space.uid

得到的结果是下图所示。很明显，mysql使用eq_ref联接来处理uchome_space表。

	id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extr
☐	1	SIMPLE	uchome_spacefield	ALL	PRIMARY	(NULL)	(NULL)	(NULL)	17039	
☐	1	SIMPLE	uchome_space	eq_ref	PRIMARY	PRIMARY	4	uspace_uchome.uchome_spacefield.uid	1	

目前的疑问：

4.3.1 为什么是只有uchome_space一个表用到了eq_ref,并且sql语句如果变成

explain select * from uchome_space,uchome_spacefield where uchome_space.uid = uchome_spacefield.uid

结果还是一样，需要说明的是uid在这两个表中都是primary

4.4 ref 对于每个来自于前面的表的行组合，所有有匹配索引值的行将从这张表中读取。如果联接只使用键的最左边的前缀，或如果键不是UNIQUE或PRIMARY KEY（换句话说，如果联接不能基于关键字选择单个行的话），则使用ref。如果使用的键仅仅匹配少量行，该联接类型是不错的。

看下面这条语句 explain select * from uchome_space where uchome_space.friendnum = 0，得到结果如下，这条语句能搜出1w条数据

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	uchome_space	ref	friendnum	friendnum	4	const	17006	

4.5 ref_or_null 该联接类型如同ref，但是添加了MySQL可以专门搜索包含NULL值的行。在解决子查询中经常使用该联接类型的优化。

上面这五种情况都是很理想的索引使用情况

4.6 index_merge 该联接类型表示使用了索引合并优化方法。在这种情况下，key列包含了使用的索引的清单，key_len包含了使用的索引的最长的关键元素。

4.7 unique_subquery

4.8 index_subquery

4.9 range 给定范围内的检索，使用一个索引来检查行。看下面两条语句

explain select * from uchome_space where uid in (1,2)

explain select * from uchome_space where groupid in (1,2)

uid有索引，groupid没有索引，结果是第一条语句的联接类型是range,第二个是ALL.以为是一定范围所以说像between也可以这种联接,很明显

explain select * from uchome_space where friendnum = 17

这样的语句是不会使用range的，它会使用更好的联接类型就是上面介绍的ref

4.10 index　该联接类型与ALL相同，除了只有索引树被扫描。这通常比ALL快，因为索引文件通常比数据文件小。（也就是说虽然all和Index都是读全表，但index是从索引中读取的，而all是从硬盘中读的）

当查询只使用作为单索引一部分的列时，MySQL可以使用该联接类型。

4.11 ALL 对于每个来自于先前的表的行组合，进行完整的表扫描。如果表是第一个没标记const的表，这通常不好，并且通常在它情况下***很***差。通常可以增加更多的索引而不要使用ALL，使得行能基于前面的表中的常数值或列值被检索出。

5 possible_keys 提示使用哪个索引会在该表中找到行，不太重要

6 keys MYSQL使用的索引，简单且重要

7 key_len MYSQL使用的索引长度

8 ref　ref列显示使用哪个列或常数与key一起从表中选择行。

9 rows 显示MYSQL执行查询的行数，简单且重要，数值越大越不好，说明没有用好索引

10 Extra 该列包含MySQL解决查询的详细信息。

10.1 Distinct　MySQL发现第1个匹配行后，停止为当前的行组合搜索更多的行。一直没见过这个值

10.2 Not exists

10.3 range checked for each record

没有找到合适的索引

10.4 using filesort

MYSQL手册是这么解释的"MySQL需要额外的一次传递，以找出如何按排序顺序检索行。通过根据联接类型浏览所有行并为所有匹配WHERE子句的行保存排序关键字和行的指针来完成排序。然后关键字被排序，并按排序顺序检索行。"目前不太明白

10.5 using index 只使用索引树中的信息而不需要进一步搜索读取实际的行来检索表中的信息。这个比较容易理解，就是说明是否使用了索引

explain select * from ucspace_uchome where uid = 1的extra为using index（uid建有索引）

explain select count(*) from uchome_space where groupid=1 的extra为using where(groupid未建立索引)

10.6 using temporary

为了解决查询，MySQL需要创建一个临时表来容纳结果。典型情况如查询包含可以按不同情况列出列的GROUP BY和ORDER BY子句时。

出现using temporary就说明语句需要优化了，举个例子来说

EXPLAIN SELECT ads.id FROM **ads**, city WHERE city.city_id = 8005 AND ads.status = 'online' AND city.ads_id=ads.id **ORDER BY ads.id desc**

```
id select_type table type possible_keys key key_len ref rows filtered
Extra
------ ----------- ------ ------ -------------- ------- ------- -------------------- ------ -------- ----------------------------
-
   1 SIMPLE city ref ads_id,city_id city_id 4 const 2838 100.00
Using temporary; Using filesort
   1 SIMPLE ads eq_ref PRIMARY PRIMARY 4 city.ads_id 1 100.00
Using where
```

这条语句会使用using temporary,而下面这条语句则不会

EXPLAIN SELECT ads.id FROM **ads**, city WHERE city.city_id = 8005 AND ads.status = 'online' AND city.ads_id=ads.id **ORDER BY city.ads_id desc**

```
id select_type table type possible_keys key key_len ref rows filtered
Extra
------ ----------- ------ ------ -------------- ------- ------- -------------------- ------ -------- --------------------------
   1 SIMPLE city ref ads_id,city_id city_id 4 const 2838 100.00
Using where; Using filesort
   1 SIMPLE ads eq_ref PRIMARY PRIMARY 4 city.ads_id 1 100.00
Using where
```

这是为什么呢？他俩之间只是一个order by不同，MySQL 表关联的算法是 Nest Loop Join，是通过驱动表的结果集作为循环基础数据，然后一条一条地通过该结果集中的数据作为过滤条件到下一个表中查询数据，然后合并结果。EXPLAIN 结果中，第一行出现的表就是驱动表（Important!）以上两个查询语句，驱动表都是 city，如上面的执行计划所示！

对驱动表可以直接排序，对非驱动表（的字段排序）需要对循环查询的合并结果（临时表）进行排序（Important!）

因此，order by ads.id desc 时，就要先 using temporary 了！

驱动表的定义

wwh999 在 2006年总结说，当进行多表连接查询时，[驱动表] 的定义为：

1）指定了联接条件时，满足查询条件的记录行数少的表为[驱动表]；

2）未指定联接条件时，行数少的表为[驱动表]（Important!）。

永远用小结果集驱动大结果集

今天学到了一个很重要的一点：当不确定是用哪种类型的join时，让mysql优化器自动去判断，我们只需写select * from t1,t2 where t1.field = t2.field

10.7 using where

WHERE子句用于限制哪一个行匹配下一个表或发送到客户。除非你专门从表中索取或检查所有行，如果Extra值不为Using where并且表联接类型为ALL或index，查询可能会有一些错误。（这个说明不是很理解，因为很多很多语句都会有where条件，而type为all或index只能说明检索的数据多，并不能说明错误，useing where不是很重要，但是很常见）

如果想要使查询尽可能快，应找出Using filesort 和Using temporary的Extra值。

10.8 Using sort_union(...), Using union(...),Using intersect(...)

这些函数说明如何为index_merge联接类型合并索引扫描

10.9 Using index for group-by

类似于访问表的Using index方式，Using index for group-by表示MySQL发现了一个索引，可以用来查询GROUP BY或DISTINCT查询的所有列，而不要额外搜索硬盘访问实际的表。并且，按最有效的方式使用索引，以便对于每个组，只读取少量索引条目。

实例讲解

通过相乘EXPLAIN输出的rows列的所有值，你能得到一个关于一个联接如何的提示。这应该粗略地告诉你MySQL必须检查多少行以执行查询。当你使用max_join_size变量限制查询时，也用这个乘积来确定执行哪个多表SELECT语句。

上一篇 linux常用实用命令详解
下一篇 MYSQL 优化数据库结构

我的同类文章

mysql（13）

MYSQL 优化数据库结构	2013-12-05 阅读 790	MySQL的日常管理	2013-10-26 阅读 877
mysql 数据目录	2013-10-22 阅读 280	高性能mysql基础知识3	2012-10-25 阅读 233
高性能mysql基础知识2	2012-10-22 阅读 209	高性能mysql基础知识1	2012-10-18 阅读 225
mysql question	2012-09-08 阅读 207	mysql--复杂sql语句解析	2012-09-07 阅读 830
Mysql 数据库优化6	2012-09-07 阅读 219		

更多文章

猜你在找

- MySQL SQL优化及高可用公开课视频分享

- MySQL数据库管理
- 《C语言/C++学习指南》数据库篇(MySQL& sqlite)
- 淘宝丁奇：如何解决影响MySQL使用的9大问题？
- 深入浅出MySQL入门必备

查看评论

5楼 chenxicigema 2015-11-18 15:29发表

4.3.1 为什么是只有uchome_space一个表用到了eq_ref,并且sql语句如果变成

explain select * from uchome_space,uchome_spacefield where uchome_space.uid = uchome_spacefield.uid

经测试，记录条数数量少的表，会用All扫，多的走eq_ref。这个也很好理解，用少的扫全表，用记录多的走索引。

4楼 lgslgs123456 2015-11-09 22:17发表

太多了 一时还看不太懂 先收藏起来 慢慢咀嚼

3楼 zhangxyzhangxy 2015-05-06 17:59发表

关于4.3.1的疑问：
因为是两个表做join，所以肯定要遍历其中的一个表，然后用uchome_spacefield中的uid去uchome_space做select，遍历uchome_spacefield的type肯定是all，在uchome_space做select的过程是eq_ref。

2楼 qq_20991785 2015-03-25 20:46发表

非常有帮助，对于各种情况讲的都很到位。

1楼 iaiti 2014-10-06 11:31发表


好详细。

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack
VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery BI
HTML5 Spring Apache .NET API HTML SDK IIS Fedora XML LBS Unity Splashtop
UML components Windows Mobile Rails QEMU KDE Cassandra CloudStack FTC coremail
OPhone CouchBase 云计算 iOS6 Rackspace Web App 关闭 Maemo Compuware 大数据
aptech Perl Tornado Ruby Hibernate ThinkPHP HBase Pure Solr Angular Cloud Foundry
Redis Scala Django