MYSQL explain详解

原创　2013年11月24日 17:55:55

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　150657

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。
虽然这篇文章我写的很长，但看起来真的不会困啊，真的都是干货啊！！！！
先解析一条sql语句，看出现什么内容

```
EXPLAIN SELECT s.uid,s.username,s.name,f.email,f.mobile,f.phone,f.postalcode,f.address
FROM uchome_space AS s,uchome_spacefield AS f
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 LIMIT 10)AS s
```

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

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 BYcity.ads_id desc

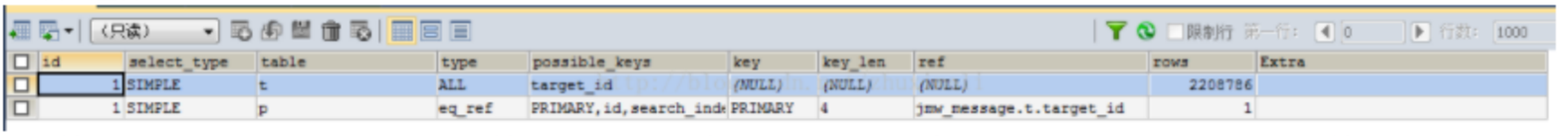
id	select_type	table	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	city	ref	ads_id,city_id	city_id	4	const	2838	100.00	**Using where; Using filesort**
1	SIMPLE	ads	eq_ref	PRIMARY	PRIMARY	4	city.ads_id	1	100.00	Using where

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

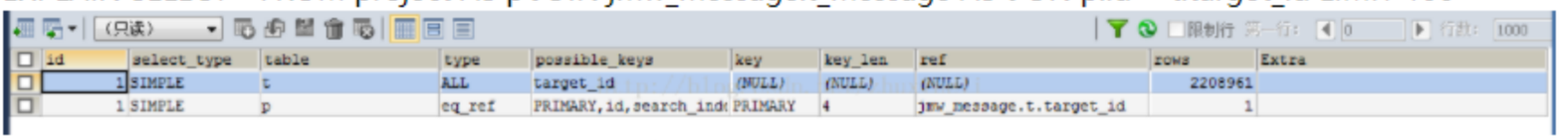
通过相乘EXPLAIN输出的rows列的所有值，你能得到一个关于一个联接如何的提示。这应该粗略地告诉你MySQL必须检查多少行以执行查询。当你使用max_join_size变量限制查询时，也用这个乘积来确定执行哪个多表SELECT语句。

2017年1.26的拓展　我是无所不能的coder的分界线
回头看看几年前写的这篇博客，真的也是很浅显，只是简单的介绍了explain后每个选项的概念，对于实例没有太多的讲解，而且最重要的是没有指出那种情况下的选项（结合实际情况）才是最优化的，ok,start again
很明显，在所有explain的结果中最重要的要数type/key/rows/extra这4个字段了,那接下来我着重在说一下这四个字段代表的意思及如何优化
现有两个表，一个项目表（project），一个留言表(t_message)，用户可以针对不同的项目进行流行操作。
现有一个最基本的联表操作，
EXPLAIN SELECT * FROM project AS p JOIN jmw_message.t_message AS t ON p.id = t.target_id
结果是这样的



id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	t	ALL	target_id	(NULL)	(NULL)	(NULL)	2208786	
1	SIMPLE	p	eq_ref	PRIMARY,id,search_inde	PRIMARY	4	jmw_message.t.target_id	1	

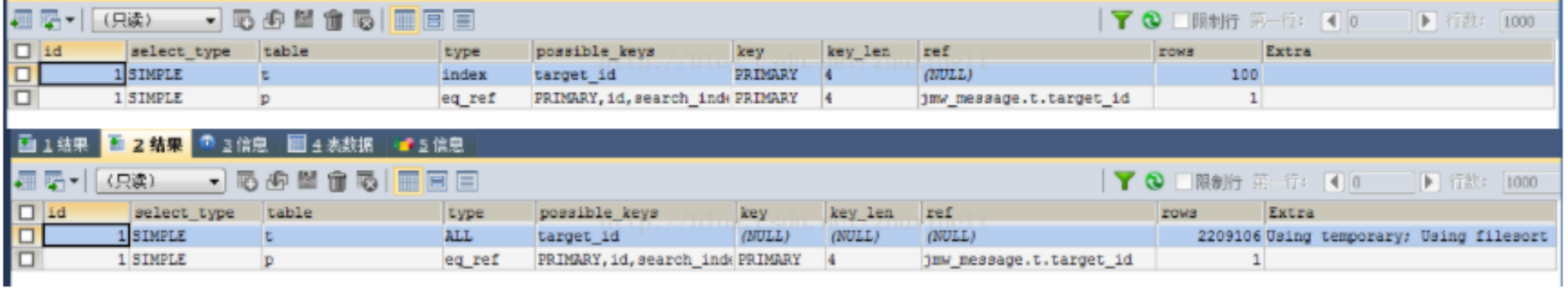
出现这种情况是最容易理解的了，因为这只是简单联表查询，没有加任何条件，在实际情况下是不会出现这种sql的。从上图的结果中可以看出mysql对t_message表进行了全表扫描，对project表使用了eq_ref,这符合了mysql对什么情况下会使用到eq_ref的定义，这是非常理想的一种连接类型。
下面我们讨论一个实际情况下会遇到的例子，我们联表取前100条数据，
EXPLAIN SELECT * FROM project AS p JOIN jmw_message.t_message AS t ON p.id = t.target_id LIMIT 100



id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	t	ALL	target_id	(NULL)	(NULL)	(NULL)	2208961	
1	SIMPLE	p	eq_ref	PRIMARY,id,search_inde	PRIMARY	4	jmw_message.t.target_id	1	

可以发现，除了影响的行数稍微多了一点（可以忽略，甚至可以理解为没有不同），其他所有的参数都是相同的，也就是说，这种情况下搜索全部数据和搜索100条数据的耗时是一样的，为什么会这样呢？不应该啊！！
这里需要着重说明的是：上面两条语句explain得到的结果是相同的，是因为他们的索引使用策略是相同的，即都没有很好的使用索引，（因为没有where条件和order by语句）但他们的最终耗时是不同的，很明显传输100条数据肯定要比传送1条数据慢。所以，最终耗时会在sending data（用show profile查看）上消耗的比例最大
那实际情况下，最有可能会遇到什么问题呢？
1 根据项目id作为搜索条件（即使用where条件）
2 根据时间或者id来排序（即使用order by条件）
3 根据以上两个
下面我们开始举栗子
《1》搜索最新的100条留言
《2》搜索出某个项目下最新的10条留言
《3》搜索出某个项目最近一个月每天有多少条留言
《4》搜索出最近一个月每天有多少条留言
《5》搜索某个用户今天留言数量
《6》搜索今天有多少条新增留言
下面我们开始吃栗子
《1》搜索最新的100条留言
EXPLAIN SELECT * FROM project AS p JOIN jmw_message.t_message AS t ON p.id = t.target_id ORDER BY t.id DESC LIMIT 100 ;
EXPLAIN SELECT * FROM project AS p JOIN jmw_message.t_message AS t ON p.id = t.target_id ORDER BY p.id DESC LIMIT 100

以下两条语句都能实现效果，但索引使用情况却完全不同。第一条语句要比第二条优化的多，



id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	t	index	target_id	PRIMARY	4	(NULL)	100	
1	SIMPLE	p	eq_ref	PRIMARY,id,search_ind	PRIMARY	4	jmw_message.t.target_id	1	

1 结果 2 结果 3 信息 4 表数据 5 信息

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	t	ALL	target_id	(NULL)	(NULL)	(NULL)	2209106	Using temporary; Using filesort
1	SIMPLE	p	eq_ref	PRIMARY,id,search_ind	PRIMARY	4	jmw_message.t.target_id	1	

可以看到，第一条语句的type值为index,影响结果即只有100行，也就是说非常合适的使用了索引。正所谓，福无双至祸不单行，当你一个地方出问题的时候，难免其他地方也出问题，因为没有使用合理的索引-->导致全表扫描-->影响结果集太大-->从而导致使用了using temporary和using filesort(这个也很重要)。那这两条语句很明显只有order by条件的一点小小的不同.说实话，我不是很理解为什么会出现这种情况因为这两个条件分表是两个表的主键，都有主键索引，唯一合理的解释可能是因为这时候联表之后t_message是主表（因为他是留言表，一切以他为准），而order by排序当然应该是根据主表的主键拍排序才会使用到索引了，似乎有点牵强，但貌似这么理解没有大毛病

下面着重说一下using temporary和using filesort

using temporary 官方解释：” 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。 “” 很明显就是通过where条件一次性检索出来的结果集太大了，内存放不下了，只能通过家里临时表来辅助处理

using filesort 官方解释： “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”

我这里的理解是：对于order by的字段没有使用到字段，所以使用了using filesort. 这两个问题同时出现的可能性很大啊！！！

《2》搜索出某个项目下最新的10条留言

EXPLAIN SELECT * FROM t_message WHERE target_id = 770 ORDER BY id DESC LIMIT 10;

EXPLAIN SELECT * FROM t_message WHERE target_id = 770 ORDER BY publish_time DESC LIMIT 10

以上两条select语句的执行搜索结果是一样的，但explain分析结果不同，只是因为order by 条件的不同