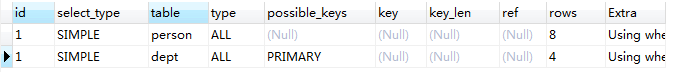
MYSQL EXPLAIN 详解

EXPLAIN 命令用于SQL语句的**查询执行计划**，这条命令的输出结果能让我们了解MySQL优化器是如何执行SQL语句的。这条命令没有提供任何调整建议，但它能够提供的信息帮助你做出调优决策。

实例：执行一条sql语句

EXPLAIN SELECT  \* FROM person,dept WHERE person.dept\_id = dept.did and person.salary >20000



下面解释查询计划结果列的各项内容。

**id:查询序列号**

查询序列号就是sql语句的执行顺序。

如上图，从2个表中查询，对应输出2行，每行对应一个表，id越大越先执行

**select\_type：查询类型**

提供了对表的查询类型。最常见的包括SIMPLE,PRIMARY,DERIVE和UNION RESULT,SUBQUERY等等。

1.simple简单查询

不包含子查询和其他复杂语法的查询，是最常见的类型。

EXPLAIN SELECT \* FROM person;

https://images2017.cnblogs.com/blog/1284211/201801/1284211-20180112114322207-2091487127.png

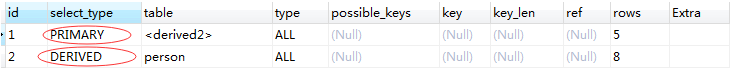
2.primary最外层查询（存在于子查询的语句中，最外面的select查询就是primary）

这是为更复杂查询而创建的首要表（也就是最外层的表）。

这个类型通常可以在DERIVED和UNION混合使用时见到。

3.derived子查询（在FROM中包含的子查询）

EXPLAIN SELECT \*FROM (SELECT\* FROM person LIMIT 5) AS s



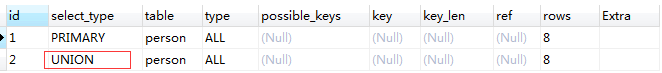
4.subquery 映射为子查询

EXPLAIN SELECT person.\*,(select 2 from person as p2) FROM person where dept\_id = (select did from dept where dname='python');

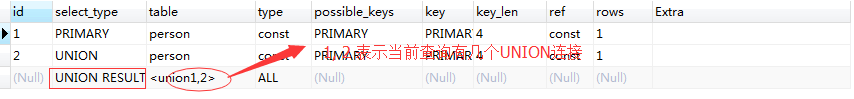


5.union联合

EXPLAIN SELECT \* FROM person union all select \* from person ;



6. union result 使用联合的结果

EXPLAIN SELECT \* FROM person union  select \* from person ; 

**table:输出的行所用的表**

**type:显示连接使用了何种类型。从最好到最差依次为const, eq\_reg, ref, range, index 和 all.**

* **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
* **eq\_reg**: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。
* **ref**: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。
* **range**:给定范围内的检索，使用一个索引来检查行。
* **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）
* **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。

**possible\_keys**：**显示可能应用在这张表中的索引。如果为空，没有可能的索引。可以为相关的域从where语句中选择一个合适的语句**

**key： 实际使用的索引。如果为null，则没有使用索引。很少的情况下，mysql会选择优化不足的索引。这种情况下，可以在select语句中使用use index（indexname）来强制使用一个索引或者用ignore index（indexname）来强制mysql忽略索引**

**key\_len：使用的索引的长度。在不损失精确性的情况下，长度越短越好**

**ref：显示索引的哪一列被使用了，如果可能的话，是一个常数**

**rows：mysql认为必须检查的用来返回请求数据的行数,数值越大越不好，说明没有用好索引**

**extra**：**关于mysql如何解析查询的额外信息。using temporary和using filesort是最差的情况，意思mysql根本不能使用索引，结果是检索会很慢**

distinct:一旦mysql找到了与行相联合匹配的行，就不再搜索了

not exists: mysql优化了left join，一旦它找到了匹配left join标准的行，就不再搜索了

range checked for each record（index map:#）:没有找到理想的索引，因此对于从前面表中来的每一个行组合，mysql检查使用哪个索引，并用它来从表中返回行。这是使用索引的最慢的连接之一

using filesort: 看到这个的时候，查询就需要优化了。mysql需要进行额外的步骤来发现如何对返回的行排序。它根据连接类型以及存储排序键值和匹配条件的全部行的行指针来排序全部行

using index: 列数据是从仅仅使用了索引中的信息而没有读取实际的行动的表返回的，这发生在对表的全部的请求列都是同一个索引的部分的时候

using temporary 看到这个的时候，查询需要优化了。这里，mysql需要创建一个临时表来存储结果，这通常发生在对不同的列集进行order by上，而不是group by上

where used 使用了where从句来限制哪些行将与下一张表匹配或者是返回给用户。如果不想返回表中的全部行，并且连接类型all或index，这就会发生，或者是查询有问题不同连接类型的解释（按照效率高低的顺序排序）

system 表只有一行：system表。这是const连接类型的特殊情况

const:表中的一个记录的最大值能够匹配这个查询（索引可以是主键或惟一索引）。因为只有一行，这个值实际就是常数，因为mysql先读这个值然后把它当做常数来对待

eq\_ref:在连接中，mysql在查询时，从前面的表中，对每一个记录的联合都从表中读取一个记录，它在查询使用了索引为主键或惟一键的全部时使用

ref:这个连接类型只有在查询使用了不是惟一或主键的键或者是这些类型的部分（比如，利用最左边前缀）时发生。对于之前的表的每一个行联合，全部记录都将从表中读出。这个类型严重依赖于根据索引匹配的记录多少—越少越好

range:这个连接类型使用索引返回一个范围中的行，比如使用>或<查找东西时发生的情况

index: 这个连接类型对前面的表中的每一个记录联合进行完全扫描（比all更好，因为索引一般小于表数据）

all:这个连接类型对于前面的每一个记录联合进行完全扫描，这一般比较糟糕，应该尽量避免