MySQL介绍和优化分享

作者：heiyeluren(黑夜路人)

博客：http://blog.csdn.net/heiyeshuwu

2009-07

目录索引

- MySQL基本介绍
- MySQL优化方式
- MySQL技巧分享
- Q & A

MySQL基本介绍

什么是MySQL

MySQL是一个小型关系型数据库管理系统，开发者为瑞典**MySQL AB**公司。目前**MySQL**被广泛地应用在**Internet**上的中小型网站中。由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，许多中小型网站为了降低网站总体拥有成本而选择了**MySQL**作为网站数据库。

MySQL官方网站：**http://www.mysql.com**

谁在用MySQL


MySQL


YAHOO!


Google
谷歌


Bai du 百度


sina 新浪网
sina.com.cn


搜狐
SOHU.com


facebook


开心网


校内 xiaonei


豆瓣 douban


猫扑
mop.com


腾讯网
QQ.com

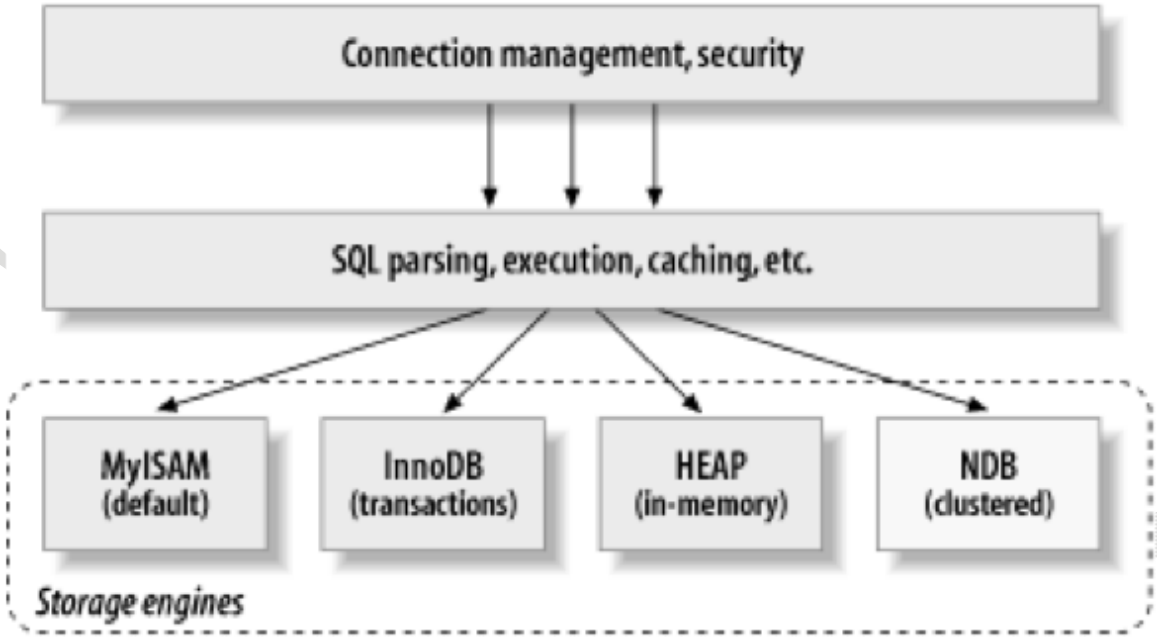
MySQL历史

- 1979年， 报表工具，数据引擎
- 1996年，MySQL 1.0 (3.11.1) 发布，支持SQL
- 2000年，成立 MySQL AB 公司
- 2008年1月，Sun公司以10亿美元收购MySQL AB公司
- 2009年4月，Oracle公司以74亿美元收购Sun公司

MySQL里程碑

- 3.11.1 First public release
- 3.23 集成Berkeley DB, 支持事务，抽象出Storage Engine
- 4.0 集成InnoDB
- 4.1 重大改进，子查询、unicode、c/s通信协议
- 5.0 stored procedure、view、triggers、query optimizer
- 5.1 File NDB、record replication.....

MySQL架构


Connection management, security
SQL parsing, execution, caching, etc.
MyISAM
(default)
InnoDB
(transactions)
HEAP
(in-memory)
NDB
(clustered)
Storage engines

MySQL存储引擎比较

Attribute	MyISAM	HEAP	BDB	InnoDB
Transactions	No	No	Yes	Yes
Lock granularity	Table	Table	Page(8K)	Row
Storage	Split files	In-memory	Single file per table	Tablespace
Isolation levels	None	None	Read committed	All
Portable format	Yes	N/A	No	Yes
Referential integrity	No	No	No	Yes
Primary key with data	No	No	Yes	Yes
Caches data records	No	Yes	Yes	Yes
Availability	All	All	MySQL-Max	All

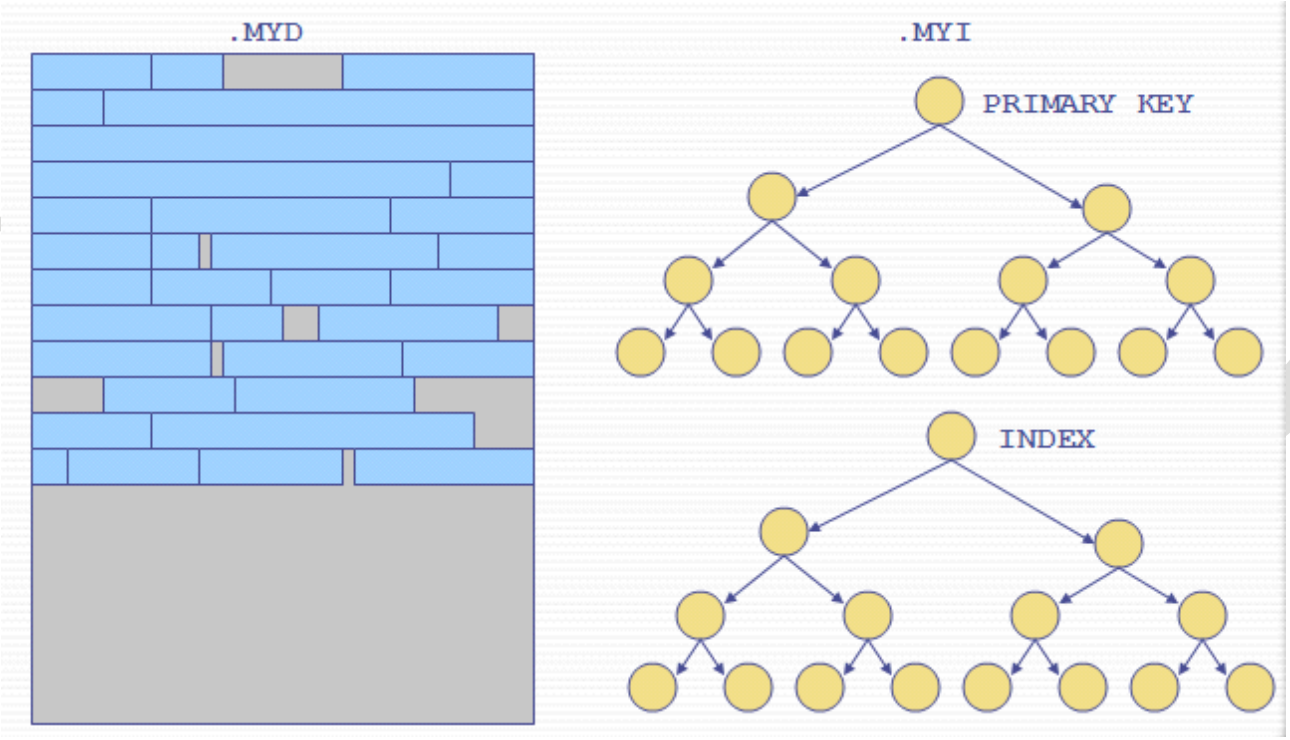
MyISAM vs InnoDB

MyISAM 特点

- 数据存储方式简单，使用 **B+ Tree** 进行索引
- 使用三个文件定义一个表：**.MYI .MYD .frm**
- 少碎片、支持大文件、能够进行索引压缩
- 二进制层次的文件可以移植 **(Linux → Windows)**
- 访问速度飞快，是所有**MySQL**文件引擎中速度最快的
- 不支持一些数据库特性，比如 事务、外键约束等
- **Table level lock**，性能稍差，更适合读取多的操作
- 表数据容量有限，一般建议单表数据量介于 **50w－200w**
- 增删查改以后要使用 **myisamchk** 检查优化表

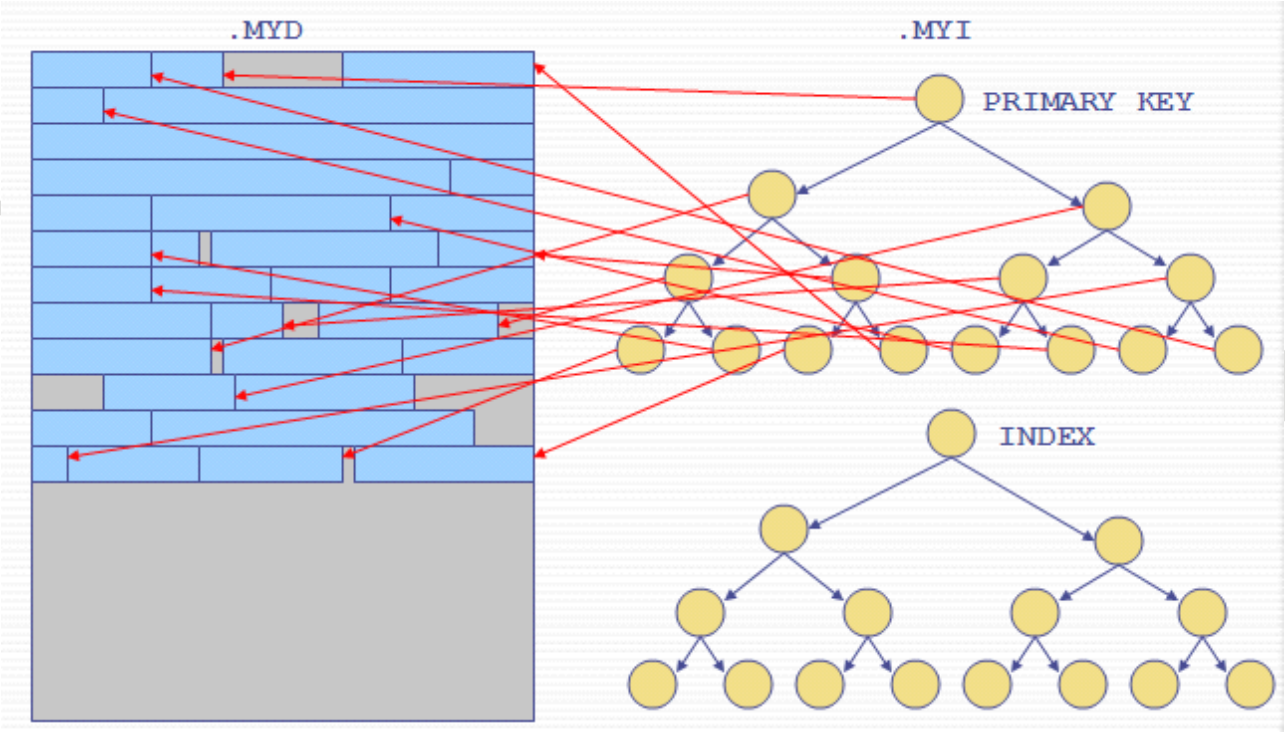
MyISAM vs InnoDB

MyISAM 存储结构



MyISAM vs InnoDB

MyISAM 索引结构



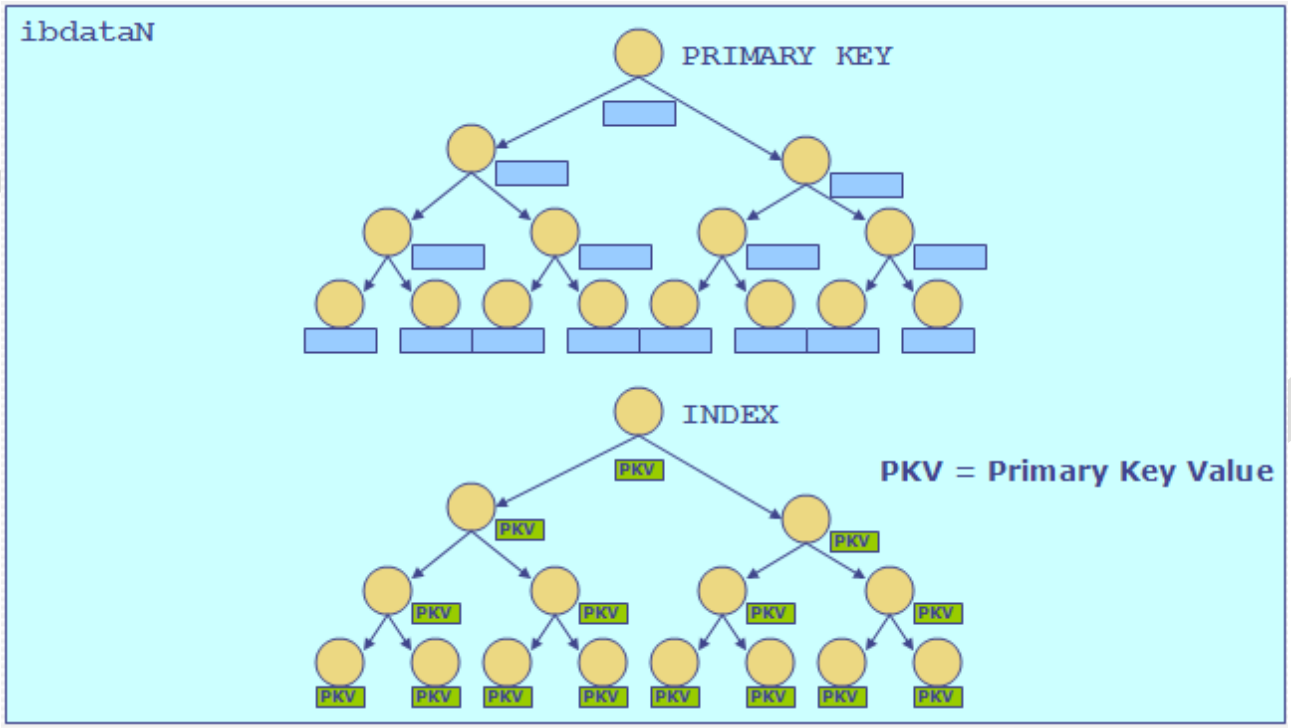
MyISAM vs InnoDB

InnoDB 特点

- 使用 **Table Space** 的方式来进行数据存储 **(ibdata1, ib_logfile0)**
- 支持 事务、外键约束等数据库特性
- **Rows level lock ,** 读写性能都非常优秀
- 能够承载大数据量的存储和访问
- 拥有自己独立的缓冲池，能够缓存数据和索引
- 在关闭自动提交的情况下，与**MyISAM**引擎速度差异不大

MyISAM vs InnoDB

InnoDB 数据结构



MyISAM vs InnoDB 性能测试

数据量/单位:万	MyISAM	InnoDB	备注: my.cnf 特殊选项
插入：1w	3秒	219秒	innodb_flush_log_at_trx_commit=1
插入：10w	29 秒	2092秒	innodb_flush_log_at_trx_commit=1
插入：100w	287秒	N/A	innodb_flush_log_at_trx_commit=1
插入：1w	3秒	3秒	innodb_flush_log_at_trx_commit=0
插入：10w	30秒	29秒	innodb_flush_log_at_trx_commit=0
插入：100w	273秒	423秒	innodb_flush_log_at_trx_commit=0
插入：1w	N/A	3秒	innodb_flush_log_at_trx_commit=0 innodb_buffer_pool_size=256M
插入：10W	N/A	26秒	innodb_flush_log_at_trx_commit=0 innodb_buffer_pool_size=256M
插入：100W	N/A	379秒	innodb_flush_log_at_trx_commit=0 innodb_buffer_pool_size=256M

MyISAM vs InnoDB 性能测试

测试结果

可以看出在**MySQL 5.0**里面，**MyISAM**和**InnoDB**存储引擎性能差别并不是很大，针对**InnoDB**来说，影响性能的主要是

innodb_flush_log_at_trx_commit 这个选项，如果设置为**1**的话，那么每次插入数据的时候都会自动提交，导致性能急剧下降，应该是跟刷新日志有关系，设置为**0**效率能够看到明显提升，当然，同样你可以**SQL**中提交“**SET AUTOCOMMIT = 0**”来设置达到好的性能。

同时也可以看出值得使用 **InnoDB** 来替代 **MyISAM** 引擎来进行开发，毕竟 **InnoDB** 有多数据库特性、更良好的数据存储性能和查询性能

MySQL优化方式

MySQL优化方式

- 系统优化：硬件、架构
- 服务优化
- 应用优化

系统优化

- 使用好的硬件，更快的硬盘、大内存、多核CPU，专业的存储服务器（NAS、SAN）

- 设计合理架构，如果 MySQL 访问频繁，考虑 Master/Slave 读写分离；数据库分表、数据库切片（分布式），也考虑使用相应缓存服务帮助 MySQL 缓解访问压力

服务优化

MySQL配置原则

- 配置合理的MySQL服务器，尽量在应用本身达到一个MySQL最合理的使用
- 针对 MyISAM 或 InnoDB 不同引擎进行不同定制性配置
- 针对不同的应用情况进行合理配置
- 针对 my.cnf 进行配置，后面设置是针对内存为2G的服务器进行的合理设置

服务优化

公共选项

选项	缺省值	推荐值	说明
max_connections	100	1024	MySQL服务器同时处理的数据库连接的最大数量
query_cache_size	0 (不打开)	16M	查询缓存区的最大长度，按照当前需求，一倍一倍增加，本选项比较重要
sort_buffer_size	512K	16M	每个线程的排序缓存大小，一般按照内存可以设置为2M以上，推荐是16M，该选项对排序order by，group by起作用
record_buffer	128K	16M	每个进行一个顺序扫描的线程为其扫描的每张表分配这个大小的一个缓冲区，可以设置为2M以上
table_cache	64	512	为所有线程打开表的数量。增加该值能增加mysqld要求的文件描述符的数量。MySQL对每个唯一打开的表需要2个文件描述符。

服务优化

MyISAM 选项

选项	缺省值	推荐值	说明
key_buffer_size	8M	256M	用来存放索引区块的缓存值, 建议128M以上，不要大于内存的30%
read_buffer_size	128K	16M	用来做MyISAM表全表扫描的缓冲大小. 为从数据表顺序读取数据的读操作保留的缓存区的长度
myisam_sort_buffer_size	16M	128M	设置,恢复,修改表的时候使用的缓冲大小，值不要设的太大

服务优化

InnoDB 选项

选项	缺省值	推荐值	说明
innodb_buffer_pool_size	32M	1G	InnoDB使用一个缓冲池来保存索引和原始数据, 这里你设置越大,你在存取表里面数据时所需要的磁盘I/O越少，一般是内存的一半，不超过2G，否则系统会崩溃，这个参数非常重要
innodb_additional_mem_pool_size	2M	128M	InnoDB用来保存 metadata 信息, 如果内存是4G，最好本值超过200M
innodb_flush_log_at_trx_commit	1	0	0 代表日志只大约每秒写入日志文件并且日志文件刷新到磁盘; 1 为执行完没执行一条SQL马上commit; 2 代表日志写入日志文件在每次提交后,但是日志文件只有大约每秒才会刷新到磁盘上. 对速度影响比较大，同时也关系数据完整性
innodb_log_file_size	8M	256M	在日志组中每个日志文件的大小, 一般是innodb_buffer_pool_size的25%，官方推荐是innodb_buffer_pool_size 的 40-50%, 设置大一点来避免在日志文件覆写上不必要的缓冲池刷新行为
innodb_log_buffer_size	128K	8M	用来缓冲日志数据的缓冲区的大小. 推荐是8M，官方推荐该值小于16M，最好是 1M-8M 之间

应用优化

应用优化方式

- 设计合理的数据表结构：适当的数据冗余
- 对数据表建立合适有效的数据库索引
- 数据查询：编写简洁高效的SQL语句

应用优化

表结构设计原则

- 选择合适的数据类型：如果能够定长尽量定长
- 不要使用无法加索引的类型作为关键字段，比如 **text**类型
- 为了避免联表查询，有时候可以适当的数据冗余，比如邮箱、姓名这些不容易更改的数据
- 选择合适的表引擎，有时候 **MyISAM** 适合，有时候 **InnoDB**适合
- 为保证查询性能，最好每个表都建立有 **auto_increment** 字段， 建立合适的数据库索引
- 最好给每个字段都设定 **default** 值

应用优化

索引建立原则

- 一般针对数据分散的关键字进行建立索引，比如**ID**、**QQ**，像性别、状态值等等建立索引没有意义
- 尽量使用短索引，一般对**int**、**char/varchar**、**date/time** 等类型的字段建立索引
- 需要的时候建立联合索引，但是要注意查询**SQL**语句的编写
- 谨慎建立 **unique** 类型的索引（唯一索引）
- 一般建议每条记录最好有一个能快速定位的独一无二定位的唯一标示（索引）
- 不要过度索引，单表建立的索引不要超过**5**个，否则更新索引将很耗时

应用优化

编写高效的 SQL

- 能够快速缩小结果集的 **WHERE** 条件写在前面，如果有恒量条件，也尽量放在前面
- 尽量避免使用 **GROUP BY**、**DISTINCT** 、**OR**、**IN** 等语句的使用，避免使用联表查询和子查询，因为将使执行效率大大下降
- 能够使用索引的字段尽量进行有效的合理排列，如果使用了联合索引，请注意提取字段的前后顺序
- 针对索引字段使用 **>, >=, =, <, <=, IF NULL**和**BETWEEN** 将会使用索引， 如果对某个索引字段进行 **LIKE** 查询，使用 **LIKE ‘%abc%’** 不能使用索引，使用 **LIKE ‘abc%’** 将能够使用索引
- 如果在**SQL**里使用了**MySQL**部分自带函数，索引将失效，同时将无法使用 **MySQL** 的 **Query Cache**，比如 **LEFT(), SUBSTR(), TO_DAYS() DATE_FORMAT(),** 等，如果使用了 **OR** 或 **IN**，索引也将失效
- 使用 **Explain** 语句来帮助改进我们的**SQL**语句

MySQL技巧分享

MySQL技巧分享

常用技巧

- 使用 **Explain/ DESC** 来分析**SQL**的执行情况
- 使用 **SHOW PROCESSLIST** 来查看当前**MySQL**服务器线程执行情况，是否锁表，查看相应的**SQL**语句
- 设置 **my.cnf** 中的 **long-query-time** 和 **log-slow-queries** 能够记录服务器那些**SQL**执行速度比较慢
- 另外有用的几个查询：**SHOW VARIABLES**、**SHOW STATUS**、**SHOW ENGINES**
- 使用 **DESC TABLE xxx** 来查看表结构，使用 **SHOW INDEX FROM xxx** 来查看表索引
- 使用 **LOAD DATA** 导入数据比 **INSERT INTO** 快多了
- **SELECT COUNT(*) FROM Tbl** 在 **InnoDB** 中将会扫描全表 **MyISAM** 中则效率很高

MySQL 技巧分享

Explain 使用

- 语法：**EXPLAIN SELECT *select_options***

Type: 类型，是否使用了索引还是全表扫描, **const,eg_reg,ref,range,index,ALL**
Key: 实际使用上的索引是哪个字段
Ken_len: 真正使用了哪些索引，不为 **NULL** 的就是真实使用的索引
Ref: 显示了哪些字段或者常量被用来和 **key** 配合从表中查询记录出来
Rows: 显示了**MySQL**认为在查询中应该检索的记录数
Extra: 显示了查询中**MySQL**的附加信息，关心**Using filesort** 和 **Using temporary**，性能杀手

```
mysql> EXPLAIN EXTENDED SELECT u.subId,r.phoneNum,s.serviceId FROM subscribe as u,service as s,register as r WHERE r.regId=u.
regId AND s.servId=u.servId AND (u.subcribeStatus=1 OR u.subcribeStatus>=3) AND u.expireTime <= now() LIMIT 0,20;
+----+-------------+-------+--------+--------------------+-----------+---------+------------+------+-------------+
| id | select_type | table | type   | possible_keys      | key       | key_len | ref        | rows | Extra       |
+----+-------------+-------+--------+--------------------+-----------+---------+------------+------+-------------+
|  1 | SIMPLE      | s     | index  | PRIMARY            | serviceId | 15      | NULL       |    3 | Using index |
|  1 | SIMPLE      | u     | ALL    | FK_REGID,FK_SERVID | NULL      | NULL    | NULL       |    3 | Using where |
|  1 | SIMPLE      | r     | eq_ref | PRIMARY            | PRIMARY   | 4       | sp.u.regId |    1 |             |
+----+-------------+-------+--------+--------------------+-----------+---------+------------+------+-------------+
3 rows in set, 1 warning (0.00 sec)
```

MySQL 技巧分享

索引实践

```
mysql> select count(1) from Tbl_ScoreDetail where FDesId = 59747935;
+----------+
| count(1) |
+----------+
|      391 |
+----------+
1 row in set (0.51 sec)

mysql> select count(1) from Tbl_ScoreDetail where FDesId = '59747935';
+----------+
| count(1) |
+----------+
|      391 |
+----------+
1 row in set (0.00 sec)
```

```
mysql> explain select count(1) from Tbl_ScoreDetail where FDesId = 59747935;
+----+-------------+-----------------+-------+---------------+--------+---------+------+--------+--------------------------+
| id | select_type | table           | type  | possible_keys | key    | key_len | ref  | rows   | Extra                    |
+----+-------------+-----------------+-------+---------------+--------+---------+------+--------+--------------------------+
|  1 | SIMPLE      | Tbl_ScoreDetail | index | FDesId        | FDesId | 767     | NULL | 684836 | Using where; Using index |
+----+-------------+-----------------+-------+---------------+--------+---------+------+--------+--------------------------+
1 row in set (0.01 sec)

mysql> explain select count(1) from Tbl_ScoreDetail where FDesId = '59747935';
+----+-------------+-----------------+------+---------------+--------+---------+-------+------+--------------------------+
| id | select_type | table           | type | possible_keys | key    | key_len | ref   | rows | Extra                    |
+----+-------------+-----------------+------+---------------+--------+---------+-------+------+--------------------------+
|  1 | SIMPLE      | Tbl_ScoreDetail | ref  | FDesId        | FDesId | 767     | const |  422 | Using where; Using index |
+----+-------------+-----------------+------+---------------+--------+---------+-------+------+--------------------------+
1 row in set (0.00 sec)
```

MySQL 技巧分享

函数和索引

```
mysql> select count(1) from Tbl_ScoreDetail where FTime < left(now(), 10);
+----------+
| count(1) |
+----------+
|   174055 |
+----------+
1 row in set (0.29 sec)

mysql> select count(1) from Tbl_ScoreDetail where FTime < '2009-07-28';
+----------+
| count(1) |
+----------+
|   174055 |
+----------+
1 row in set (0.08 sec)
```

```
mysql> select count(1) from Tbl_ScoreDetail where FTime like '%2009-07-28%';
+----------+
| count(1) |
+----------+
|     1117 |
+----------+
1 row in set (0.22 sec)

mysql> select count(1) from Tbl_ScoreDetail where FTime like '2009-07-28%';
+----------+
| count(1) |
+----------+
|     1117 |
+----------+
1 row in set (0.10 sec)
```

MySQL 技巧分享

使用 UNION 来取代 IN 和 OR

原SQL：

```
select * from city where id in (1,3,4)  或  select * from city where id = 1 or id = 3 or id = 4
```

***explain* 结果：**

```
id select_type table type possible_keys key key_len ref rows Extra
1  SIMPLE  city ALL  PRIMARY  NULL NULL  NULL 5  Using where
```

修改后SQL：

```
SELECT * FROM city where id = 1
UNION ALL SELECT * FROM city where id = 3
UNION ALL SELECT * FROM city
```

***explain* 结果：**

```
id  select_type table      type possible_keys key     key_len ref   rows Extra
1   PRIMARY     city       const PRIMARY      PRIMARY 4       const 1
2   UNION       city       const PRIMARY      PRIMARY 4       const 1
3   UNION       city       const PRIMARY      PRIMARY 4       const 1

NULL UNION RESULT <union1,2,3> ALL   NULL       NULL    NULL    NULL NULL
```

MySQL 技巧分享

MySQL Slow Log 分析工具

- **mysqldumpslow - mysql官方提供的慢查询日志分析工具**
- **mysqlsla - hackmysql.com推出的一款日志分析工具，功能非常强大**
- **mysql-explain-slow-log - 德国工程师使用Perl开发的把Slow Log 输出到屏幕，功能简单**
- **mysql-log-filter - Google code 上一个开源产品，报表简洁**
- **myprofi - 纯PHP开发的开源log查看工具，功能详细**

MySQL 技巧分享

MySQL优化网站/书籍分享

- MySQL Performance Blog
- MySQL 中文网
- 《MySQL性能调优与架构设计 》
- 《深入浅出MySQL》

Q & A

结束

谢谢大家！