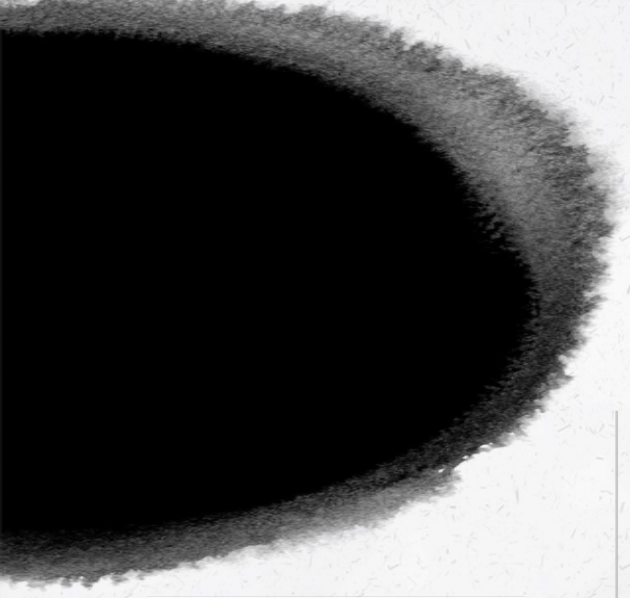


MySQL架构演变与性能优化

信海龙

2017-08-26

简介

RESUME

一个不务正业，经历丰富的老程序猿

博学无忧，www.bo56.com

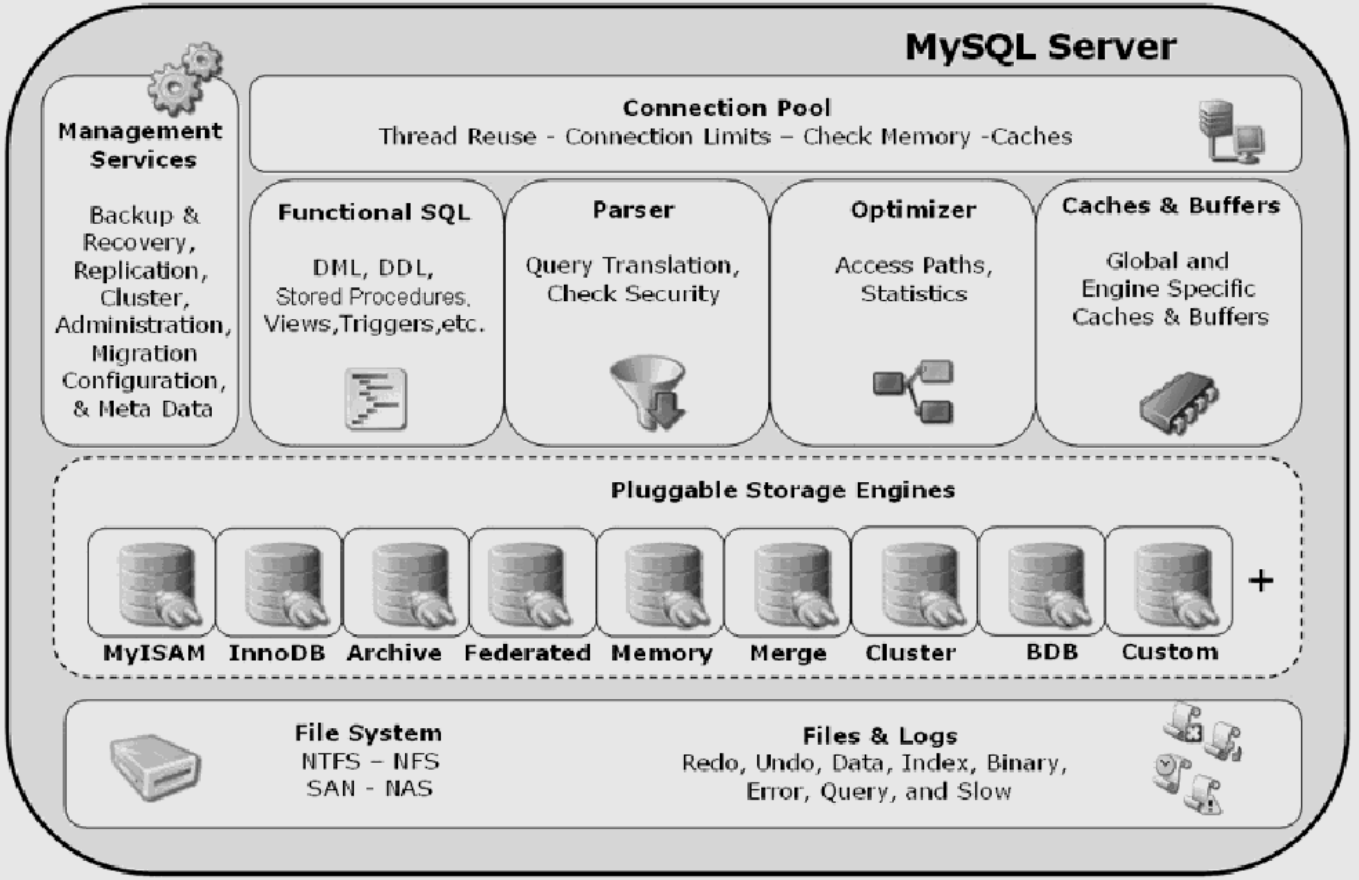
目录

CONTENTS

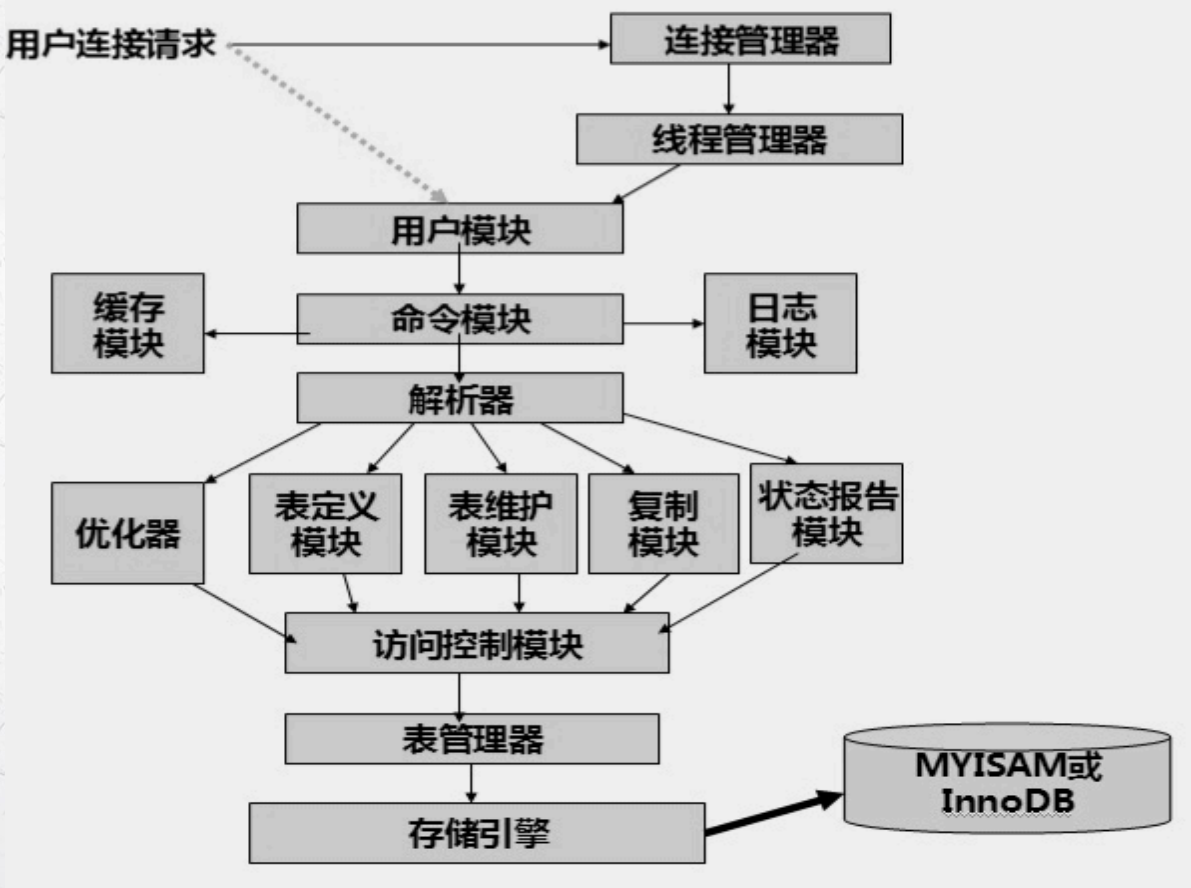
目录

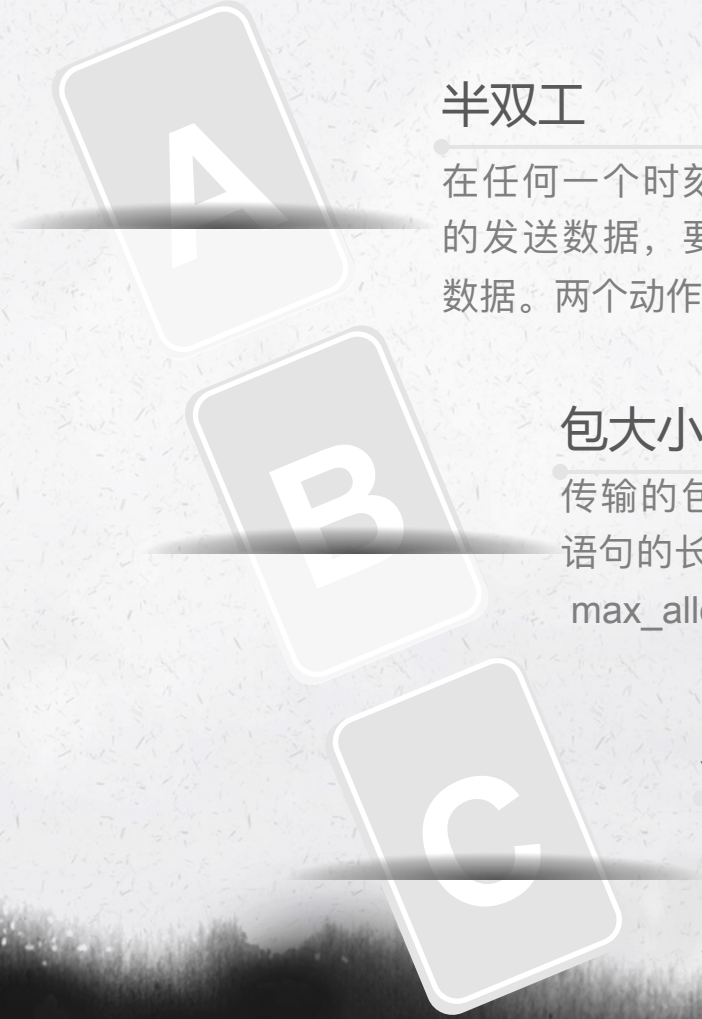
CONTENTS

MySQL 架构图


MySQL Server
Management Services
Backup & Recovery, Replication, Cluster, Administration, Migration Configuration, & Meta Data
Connection Pool
Thread Reuse - Connection Limits – Check Memory -Caches
Functional SQL
DML, DDL, Stored Procedures, Views,Triggers,etc.
Parser
Query Translation, Check Security
Optimizer
Access Paths, Statistics
Caches & Buffers
Global and Engine Specific Caches & Buffers
Pluggable Storage Engines
MyISAM
InnoDB
Archive
Federated
Memory
Merge
Cluster
BDB
Custom
+
File System
NTFS – NFS
SAN - NAS
Files & Logs
Redo, Undo, Data, Index, Binary, Error, Query, and Slow

MySQL 流程图


用户连接请求
连接管理器
线程管理器
用户模块
缓存模块
命令模块
日志模块
解析器
优化器
表定义模块
表维护模块
复制模块
状态报告模块
访问控制模块
表管理器
存储引擎
MYISAM或InnoDB



半双工

在任何一个时刻。要么客户端向服务端的发送数据，要么服务端向客户端发送数据。两个动作不能同时发生。

包大小

传输的包有大小限制。所以要注意SQL语句的长度。过长会被抛异常。可以调整 max_allowed_packet 参数控制包大小。

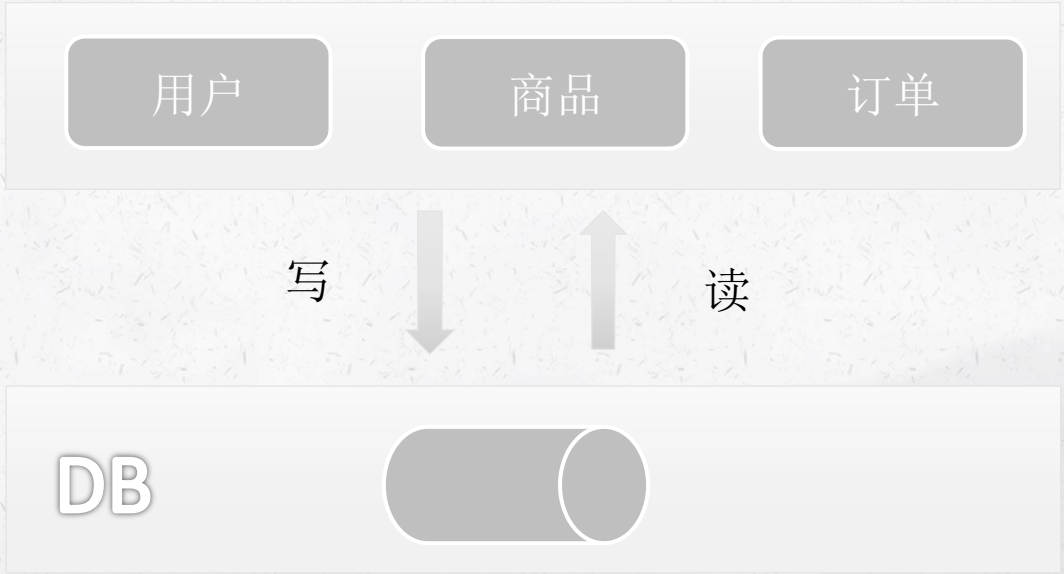
客户端缓存

客户端一般会全部接收返回的结果数据，然后缓存到内存中。尽早的释放MySQL服务器资源。

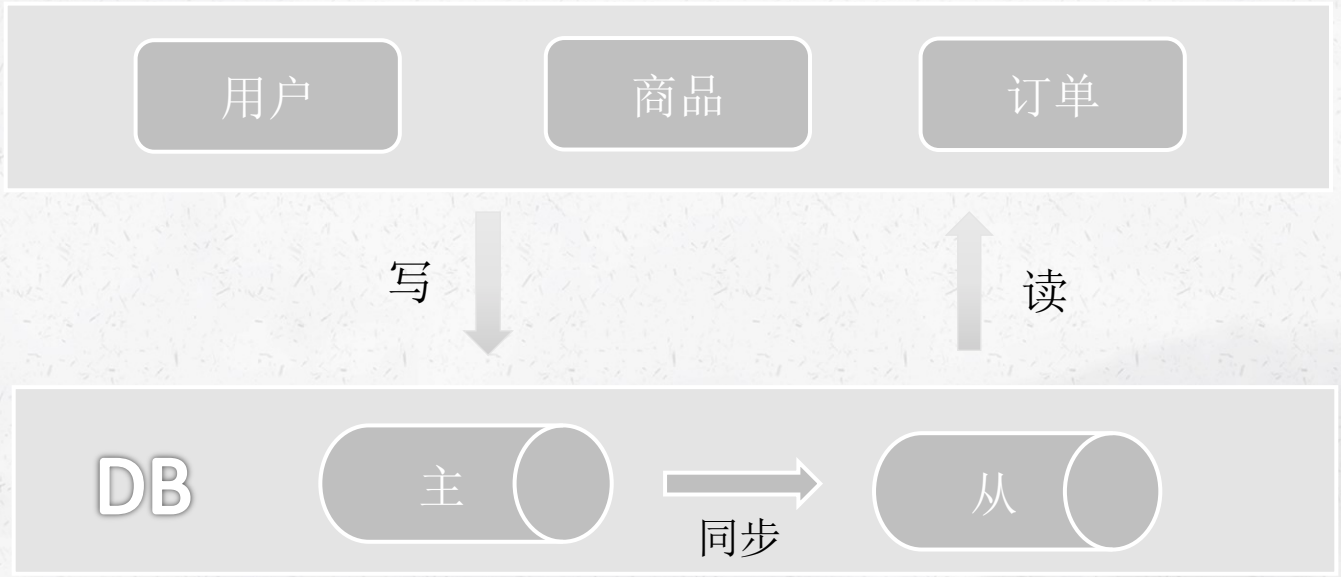
目录

CONTENTS

单表单库



读写分离



垂直分库

水平分表

目录

CONTENTS

论坛帖子分表

id	user_id	title	content	...
1	2	博学无忧	技术分析	
2	3	最好的语言	不服来战	
3	4	语言新秀	异军突起	
4	5	说点啥呢？	沉默是金	

数据垂直拆分

id	user_id	title	...
1	2	博学无忧	
2	3	最好的语言	
3	4	语言新秀	
4	5	说点啥呢？	

id	content
1	技术分析
2	不服来战
3	异军突起
4	沉默是金

按ID水平拆分

table_id_00

id	user_id	title	. ...
2	3	最好的语言	
4	5	说点啥呢？	

table_id_01

id	user_id	title	. ...
1	2	博学无忧	
3	4	语言新秀	

按用户ID水平拆分

table_user_id_00

id	user_id	title	...
1	2	博学无忧	
3	4	语言新秀	

table_user_id_01

id	user_id	title	...
2	3	最好的语言	
4	5	说点啥呢？	

数据主键生成(单表)

```
CREATE TABLE `sequence` (
 `tablename` varchar(30) NOT NULL,
 `nextid` bigint(20) NOT NULL,
 PRIMARY KEY (`tablename`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

```
# 获取最新 nextid值 , 如 202
select nextid from table where tablename='table';
# nextid值加1后更新
update sequence set nextid = 203 where tablename = 'table' and nextid < 203;
```

数据主键生成（flicker）

```
CREATE TABLE `sequence` (
  `id` bigint(20) unsigned NOT NULL auto_increment,
  `stub` char(1) NOT NULL default '',
  PRIMARY KEY  (`id`),
  UNIQUE KEY `stub` (`stub`)
) ENGINE=MyISAM;
```

```
auto-increment-increment = 2
auto-increment-offset = 1
```

```
auto-increment-increment = 2
auto-increment-offset = 2
```

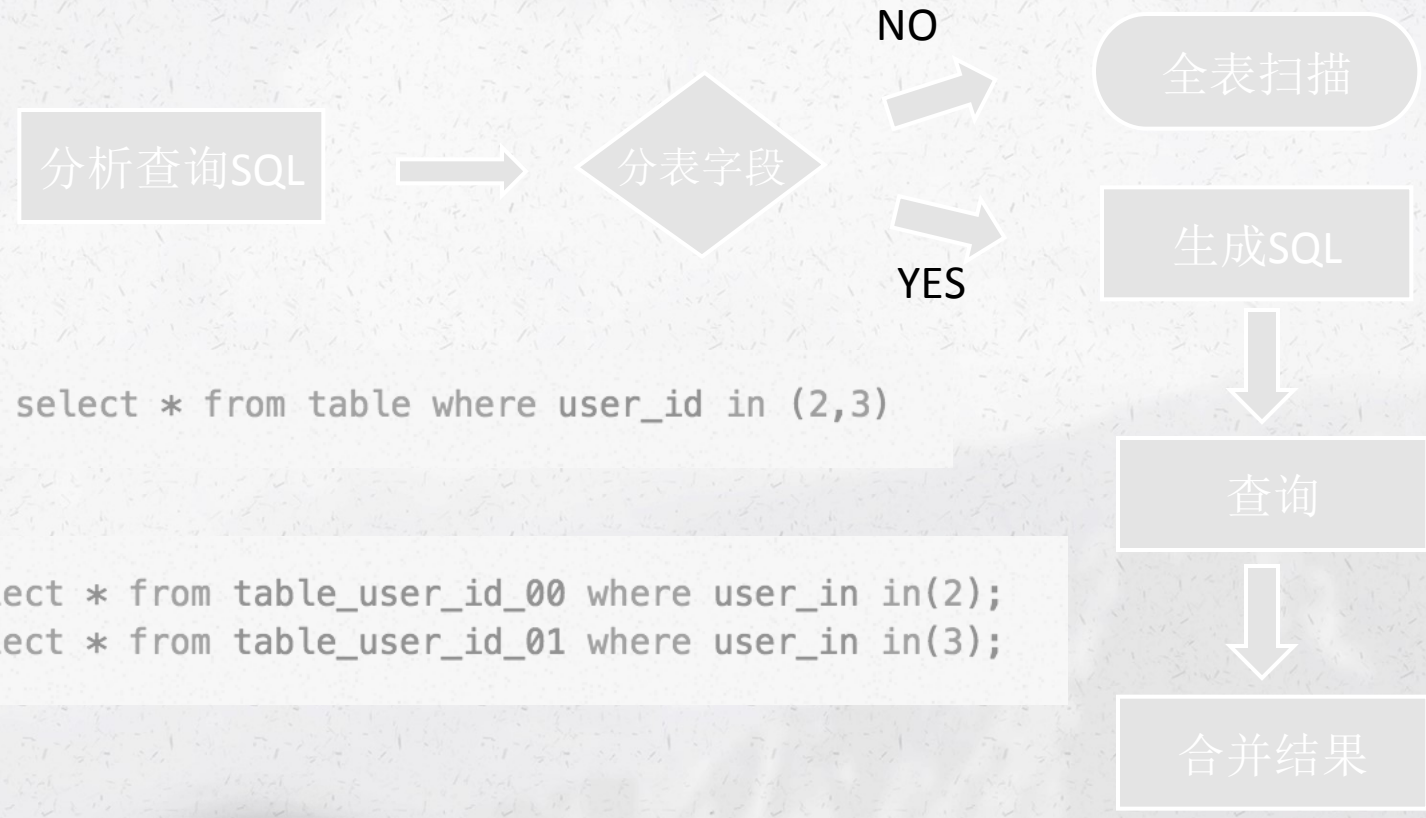
```
REPLACE INTO uid_sequence (stub) VALUES ('a');
SELECT LAST_INSERT_ID();
```

数据插入问题


```
insert into table(id, user_id, title) value(6,7,'new');
```

```
insert into table_id_00(id, user_id, title) value(6,7,'new');
insert into table_user_id_01(id, user_id, title) value(6,7,'new');
```

数据查询问题




```
select * from table where user_id in (2,3)
```

```
select * from table_user_id_00 where user_in in(2);
select * from table_user_id_01 where user_in in(3);
```

数据更新流程

分析SQL → 查询数据 → 生成SQL → 更新数据

```
update table set title='update' where id=2;
```

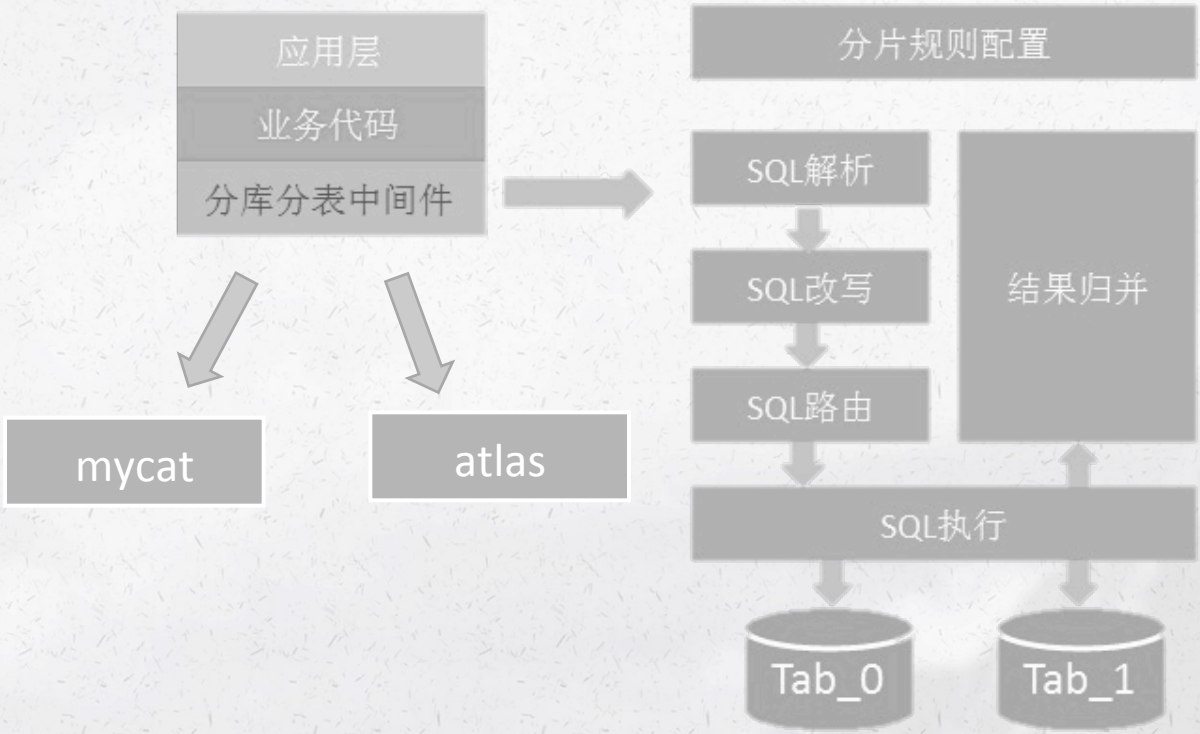
```
update table_id_00 set title='update' where id=2;
update table_user_id_01 set title='update' where id=2;
```

数据迁移问题

id	user_id	title	...
2	3	最好的语言	
4	5	说点啥呢？	

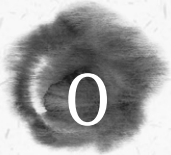
```
insert into table_id_02(id, user_id...) value(2,3...)
insert into table_user_id_03(id, user_id...) value(2,3...)
insert into table_id_00(id, user_id...) value(4,5...)
insert into table_user_id_01(id, user_id...) value(4,5...)
```

MySQL中间件

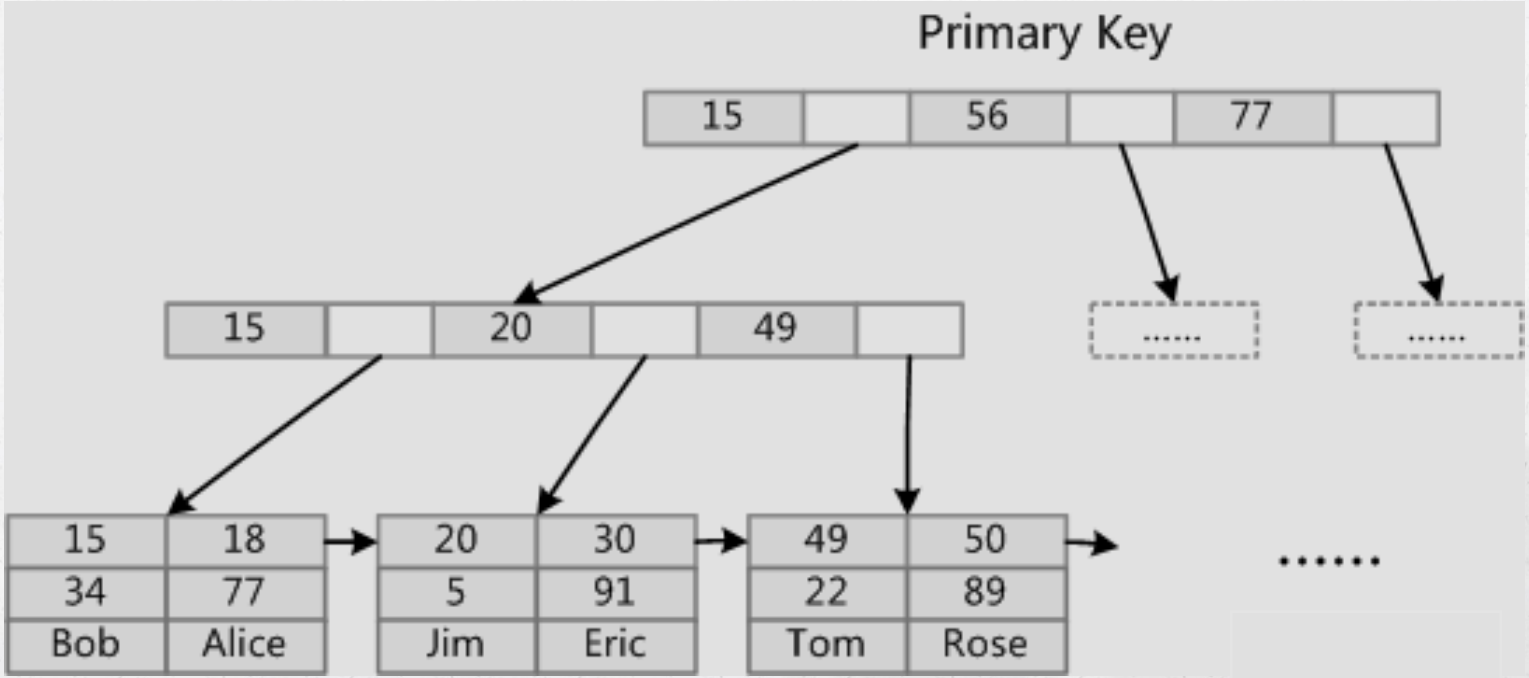

应用层
业务代码
分库分表中间件
mycat
atlas
分片规则配置
SQL解析
SQL改写
SQL路由
结果归并
SQL执行
Tab_0
Tab_1

目录

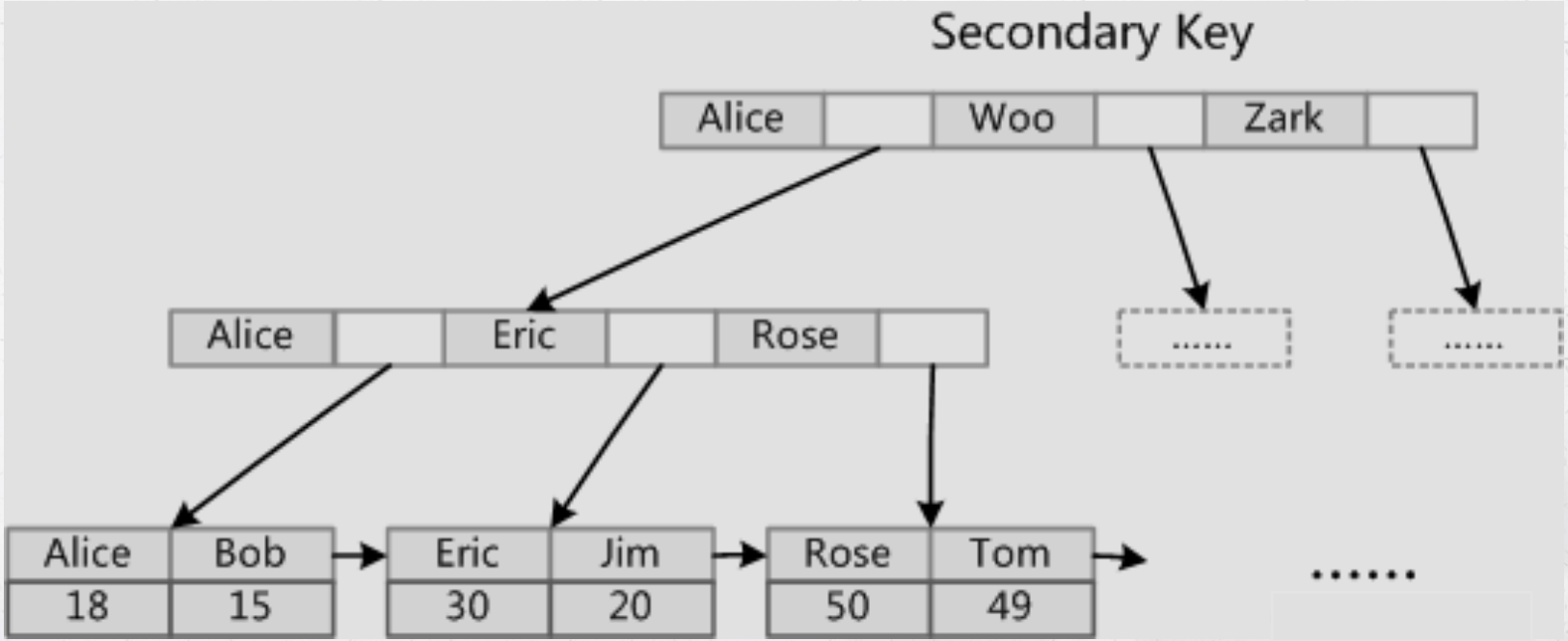
CONTENTS



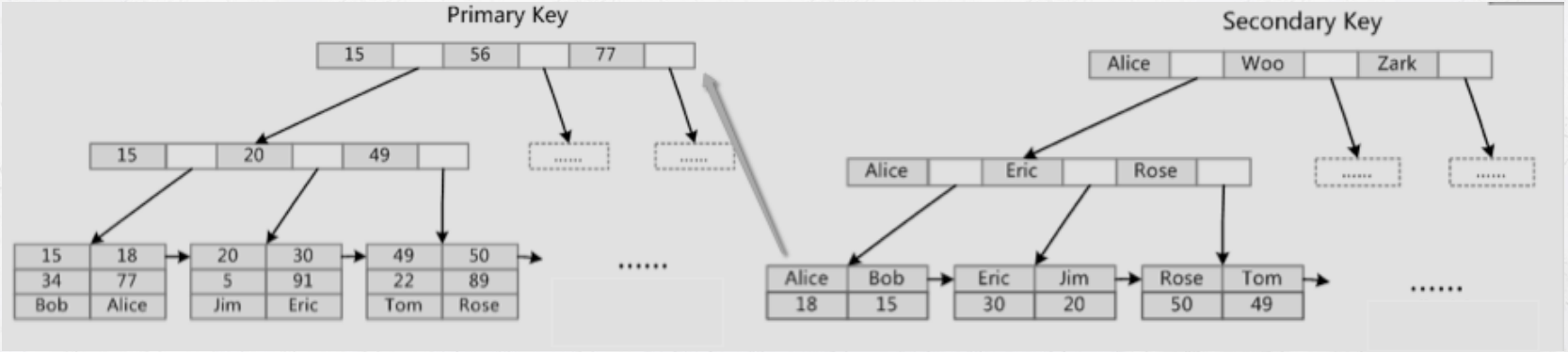
B+Tree 一级索引


Primary Key
15
56
77
15
20
49
.....
.....
15
18
34
77
Bob
Alice
20
30
5
91
Jim
Eric
49
50
22
89
Tom
Rose
.....

B+Tree二级索引


Secondary Key
Alice
Woo
Zark
Alice
Eric
Rose
.....
.....
Alice
Bob
18
15
Eric
Jim
30
20
Rose
Tom
50
49
.....

二级索引查询


Primary Key
15
56
77
15
20
49
15
18
34
77
Bob
Alice
20
30
5
91
Jim
Eric
49
50
22
89
Tom
Rose
Secondary Key
Alice
Woo
Zark
Alice
Eric
Rose
Alice
Bob
18
15
Eric
Jim
30
20
Rose
Tom
50
49

如何发现慢SQL

slow_query_log

这个参数设置为ON，可以捕获执行时间超过一定数值的SQL语句。

日志

slow_query_log_file

记录日志的文件名。

long_query_time

当SQL语句执行时间超过此数值时，就会被记录到日志中，建议设置为1或者更短。

explain

```
+------+--------------------+------------+--------+----------------------+---------+---------+------+--------+-------------+
| id   | select_type        | table      | type   | possible_keys        | key     | key_len | ref  | rows   | Extra       |
+------+--------------------+------------+--------+----------------------+---------+---------+------+--------+-------------+
|    1 | PRIMARY            | e1         | ALL    | NULL                 | NULL    | NULL    | NULL | 299423 | Using where |
|    2 | DEPENDENT SUBQUERY | e2         | eq_ref | PRIMARY,i_first_name | PRIMARY | 4       | func |      1 | Using where |
|    3 | DEPENDENT UNION    | e3         | eq_ref | PRIMARY              | PRIMARY | 4       | func |      1 | Using where |
| NULL | UNION RESULT       | <union2,3> | ALL    | NULL                 | NULL    | NULL    | NULL |   NULL |             |
+------+--------------------+------------+--------+----------------------+---------+---------+------+--------+-------------+
4 rows in set (0.01 sec)
```

如何判断索引好坏

```
root@test 02:19:27>show indexes from mq_biz_order_0000;
+-----------------+------------+-----------+--------------+--------------+-----------+-------------+----------+--------+------+------------+
| Table           | Non_unique | Key_name  | Seq_in_index | Column_name  | Collation | Cardinality | Sub_part | Packed | Null | Index_t
+-----------------+------------+-----------+--------------+--------------+-----------+-------------+----------+--------+------+------------+
| mq_biz_order_0000 |          0 | PRIMARY   |            1 | biz_order_id | A         |        5272 |     NULL | NULL   |      | BTREE
| mq_biz_order_0000 |          1 | idx_st_bt |            1 | status       | A         |           4 |     NULL | NULL   | YES  | BTREE
| mq_biz_order_0000 |          1 | idx_st_bt |            2 | biz_type     | A         |          28 |     NULL | NULL   |      | BTREE
| mq_biz_order_0000 |          1 | idx_bid_pt |           1 | buyer_id     | A         |        5272 |     NULL | NULL   |      | BTREE
| mq_biz_order_0000 |          1 | idx_bid_pt |           2 | pay_time     | A         |        5272 |     NULL | NULL   | YES  | BTREE
+-----------------+------------+-----------+--------------+--------------+-----------+-------------+----------+--------+------+------------+
5 rows in set (0.00 sec)

root@test 02:19:32>select count(*),count(distinct status),count(distinct biz_type),count(distinct buyer_id),count(distinct pay_time) from
+----------+-----------------------+-----------------------+------------------------+------------------------+
| count(*) | count(distinct status) | count(distinct biz_type) | count(distinct buyer_id) | count(distinct pay_time) |
+----------+-----------------------+-----------------------+------------------------+------------------------+
|     6063 |                     2 |                    13 |                   2832 |                   3260 |
+----------+-----------------------+-----------------------+------------------------+------------------------+
1 row in set (0.01 sec)
```

最左前缀原则

对于索引的匹配，是从左到右的。即，索引字段的值，只能先完全匹配左边后，才能匹配后边的值。

```
# title字段建立索引
select * from table where title like 'abc%'; # 能用到索引
select * from table where title like '%abc'; # 不能用到索引
```

```
# user_id 和 title字段建立联合索引
select * from table where user_id=1 and title like 'abc%'; # 能用到索引
select * from table where title like 'abc%'; # 不能用到索引
```

覆盖索引

当一个查询SQL中所有使用的字段，在索引中全部有时，就不会再跟进索引读取数据文件。

```
# user_id 和 title字段建立联合索引
select user_id,title from table where user_id=1 and title like 'abc%'; # 能用到索引
```

使用索引排序

当索引的列顺序和order by子句的顺序完全一致，并且所有列的排序方向都一样时，mysql才能使用索引来对结果做排序，

如果查询需要关联多张表，则只有当order by子句引用的字段全部为第一个表时，才能使用索引做排序。

```
# user_id 和 title字段建立联合索引
select * from table where user_id=1 order by title desc; # 能用到索引
```

Innodb主键优化

1. 主键要自增
2. 主键要尽量小
3. 尽量不要对主键修改
4. 显示指定主键

批量插入

批量插入，插入速度更快

```
# 逐条插入
insert into table (id, user_id, title) value (1, 2, 'a');
insert into table (id, user_id, title) value (2, 3, 'b');
```

```
# 批量插入
insert into table (id, user_id, title) values (1, 2, 'a'), (2, 3, 'b');
```

谢谢

THANK YOU