

# 性能优化

# 目录

1

ISS

2

ICP

3

索引失效

# 上期问题续: 用短的停机时间, 删除主键

```
CREATE TABLE `f` (  
  `id` varchar(36) NOT NULL ,  
  `a` varchar(36) DEFAULT ,  
  `b` varchar(36) DEFAULT ,  
  `c` varchar(36) DEFAULT ,  
  `d` varchar(36) DEFAULT ,  
  `e` varchar(50) DEFAULT ,  
  `f` varchar(100) DEFAULT ,  
  `g` datetime DEFAULT NULL COMMENT '行为操作时间',  
  `h` varchar(50) DEFAULT ,  
  `i` char(1) DEFAULT ,  
  `j` varchar(2500) DEFAULT ,  
  `k` varchar(500) DEFAULT ,  
  `l` varchar(500) DEFAULT ,  
  `m` varchar(30) DEFAULT ,  
  `n` varchar(255) DEFAULT ,  
  PRIMARY KEY (`id`) USING BTREE,  
  KEY `ind_d` (`d`) USING BTREE ,  
  KEY `ind_hx` (h,n) USING BTREE ,  
  KEY `file_operate_time` (h) USING BTREE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8  
ROW_FORMAT=DYNAMIC;
```

说明: 业务可以接受停机时间,要求是越短越好,  
因此不考虑使用 ost 工具

## 方案3：先并发写主键索引，后续补普通索引

### 创建空表

```
CREATE TABLE `new_f` (  
  `id` bigint unsigned auto_increment ,  
  `a` varchar(36) DEFAULT ,  
  `b` varchar(36) DEFAULT ,  
  `c` varchar(36) DEFAULT ,  
  `d` varchar(36) DEFAULT ,  
  `e` varchar(50) DEFAULT ,  
  `f` varchar(100) DEFAULT ,  
  `g` datetime DEFAULT NULL ,  
  `h` varchar(50) DEFAULT ,  
  `i` char(1) DEFAULT ,  
  `j` varchar(2500) DEFAULT ,  
  `k` varchar(500) DEFAULT ,  
  `l` varchar(500) DEFAULT ,  
  `m` varchar(30) DEFAULT ,  
  `n` varchar(255) DEFAULT ,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8  
ROW_FORMAT=DYNAMIC;
```

# 方案3：先并发写主键索引，后续补普通索引

## 1. 准备数据

```
select * from f order by id limit 5000000 into outfile file1;  
//tail file1 最后一行得到最大id M1  
select * from f order by id where id>M1 limit 5000000 into outfile file2;  
//tail file2 最后一行得到最大id M2  
....  
select * from f order by id where id>M6 limit 5000000 into outfile file2;  
  
// 修改file1文件, 把第一列修改成数字1~500W  
//修改file2文件, 把第一列修改成数字5000001~1000W  
.....
```

## 2. 多线程执行,每个线程执行 load

```
load data file1 into ..  
load data file2 into ..  
...
```

## 3. 全部 insert 执行完成后

```
alter table new_f add KEY `ind_d` (`d`) USING BTREE ,  
add KEY `ind_hx` (h,n) USING BTREE ;
```



# 课堂练习：方案对比

1. 如果建表的时候就把 new\_f 里,这两个索引都先加上,再 load, 会出现什么情况?

```
add KEY `ind_d` (`d`) USING BTREE ,  
add KEY `ind_hx` (h,n) USING BTREE ;
```

2. 如果不设置 csv 表的 id 值,用  
load data file1 into new\_f(a,b,c...) values(...)  
有什么不同

相关知识点:

innodb\_autoinc\_lock\_mode=2 时, 所有的申请自增主键的动作都是申请后就释放锁

# 课堂练习

什么情况下会出现：一个语句把 order by desc 改成 asc 就慢很多？

# 参考回答

```
create table sample(id int primary key, c int, d int, e int,
f int, index(c), index(d,e));
insert into
sample(1,1,1,1,1),(2,2,2,2,2)...(10w,10w,10w,10w,10w);
select * from sample where c<2w and f>9w order by c
desc limit 10;
```

```
select d from sample order by d desc, e asc;
```

where c<2w order by d desc这个需要排序  
吗

where c<2w order by d desc , e desc 这个后面列需要排序吗  
(c,d,e)

where c<2w order by c desc, d desc , e desc 这个后面列需要排序吗  
(c,d,e)

(cde)(cde)



# Index skip scan

```
CREATE TABLE t1 (f1 INT NOT NULL, f2 INT NOT NULL,  
PRIMARY KEY(f1, f2));
```

```
INSERT INTO t1 VALUES (1,1), (1,2), (1,3), (1,4), (1,5), (2,1), (2,2),  
(2,3), (2,4), (2,5);
```

```
INSERT INTO t1 SELECT f1, f2 + 5 FROM t1;
```

```
INSERT INTO t1 SELECT f1, f2 + 10 FROM t1;
```

```
INSERT INTO t1 SELECT f1, f2 + 20 FROM t1;
```

```
INSERT INTO t1 SELECT f1, f2 + 40 FROM t1;
```

```
ANALYZE TABLE t1;
```

```
mysql> select f1, count(*) as c from t1 group by f1;
```

f1	c
1	80
2	80

```
2 rows in set (0.00 sec)
```

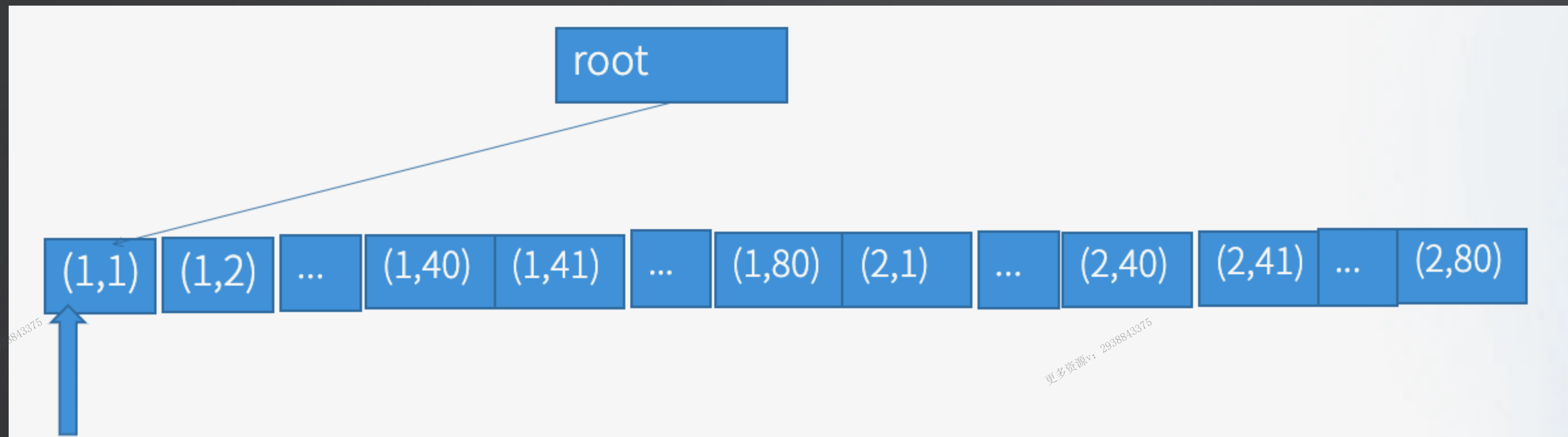
# 执行计划

EXPLAIN SELECT f1, f2 FROM t1 WHERE f2 > 40;

```
mysql> EXPLAIN SELECT f1, f2 FROM t1 WHERE f2 > 40\G
***** 1. row *****
      id: 1
  select_type: SIMPLE
        table: t1
  partitions: NULL
         type: range
possible_keys: PRIMARY
          key: PRIMARY
       key_len: 8
         ref: NULL
        rows: 53
   filtered: 100.00
      Extra: Using where; Using index for skip scan
1 row in set, 1 warning (0.00 sec)
```

# 执行流程

SELECT f1, f2 FROM t1 WHERE f2 > 40;



# Index condition pushdown

```
create table sample(id int primary key, c int, d int, e int, f
int, index(c), index(d,e));
insert into
sample(1,1,1,1,1),(2,2,2,2,2)...(10w,10w,10w,10w,10w);
explain select * from sample where d>5w and e>9w;
```

# 索引失效

课堂练习： 以下哪些情况会导致查询条件中的索引失效？

- A. where  $f+1 > 100$
- B. where  $f = \text{cast}(100, \text{char})$ ; (f是varchar类型)
- C. where  $\text{cast}(f, \text{char}) = 'a'$ ; (f是big unsigned)
- D. join 字段,一个是 utf8 一个是utf8mb4



# 讨论：5.7 升级到 8.0, utf-8 怎么处理？

是否需要重建包含 `varchar` 的表？

# 内存命中率

课堂练习:

```
create table sample(id int primary key, c int, d int, e int, f int, index(c), index(d,e));
```

一、以下关于select count(\*) from sample force index(c);说法错误的是:

- A. 这个语句可能走主键索引
- B. 这个语句会访问索引 c 的所有 page
- C. 这个语句可以使用覆盖索引
- D. 可以用定时执行这个语句的方式,让索引 c 的数据页有更多的机会常驻内存

二、如果表有 1000w 行,为了实现上题 D 的目的,同时避免这个语句执行期间占用大量 CPU,是否可以使用 select sleep(0.000001), id from sample force index(c),让每查询一行暂停1 微秒来实现?

# 参考回答：sleep(t) 里 t 最小有效值的问题


```
/*
  On 64-bit OSX mysql_cond_timedwait() waits forever
  if passed abstime time has already been exceeded by
  the system time.
  When given a very short timeout (< 10 mcs) just return
  immediately.
  We assume that the lines between this test and the call
  to mysql_cond_timedwait() will be executed in less than 0.00001 sec.
*/
if (timeout < 0.00001) return 0;

timed_cond.set_timeout((ulonglong)(timeout * 1000000000.0));

mysql_cond_init(key_item_func_sleep_cond, &cond);
mysql_mutex_lock(&LOCK_item_func_sleep);

thd->ENTER_COND(&cond, &LOCK_item_func_sleep, &stage_user_sleep, nullptr);

error = 0;
thd_wait_begin(thd, THD_WAIT_SLEEP);
while (!thd->killed) {
    error = timed_cond.wait(&cond, &LOCK_item_func_sleep);
    if (is_timeout(error)) break;
    error = 0;
}
```



# 参考回答：sleep(t) 里 t 最小有效值的问题

```
/*
On 64-bit OSX mysql_cond_timedwait() waits forever
if passed abstime time has already been exceeded by
the system time.
When given a very short timeout (< 10 mcs) just return
immediately.
We assume that the lines between this test and the call
to mysql_cond_timedwait() will be executed in less than 0.00001 sec.
*/
if (timeout < 0.00001) return 0;
timed_cond.set_timeout((ulonglong)(timeout * 1000000000.0));

mysql_cond_init(key_item_func_sleep_cond, &cond);
mysql_mutex_lock(&LOCK_item_func_sleep);

thd->ENTER_COND(&cond, &LOCK_item_func_sleep, &stage_user_sleep, nullptr);

error = 0;
thd_wait_begin(thd, THD_WAIT_SLEEP);
while (!thd->killed) {
    error = timed_cond.wait(&cond, &LOCK_item_func_sleep);
    if (is_timeout(error)) break;
    error = 0;
}
```



# 讨论

“身份证号码”要不要单独创建一个 hash 字段做索引用？

```
ha=crc32(people_id)  
key ha(ha)
```



# 索引长度

手机号码、身份证号码、邮箱地址、居住地址等一类的字段, 索引怎么创建

```
select  
  count(distinct left(email,4)) as L4,  
  count(distinct left(email,5)) as L5,  
  count(distinct left(email,6)) as L6,  
  count(distinct left(email,7)) as L7,  
from speople;
```

email(7), address(10)

```
select count(*) from speople where email like 'abx%' and xxx=...;
```

# 模糊匹配

```
select * from where f1 like '%abc%' and f2 between '2024-01-01' and '2025-1-1'
```

有优化空间吗?

```
select * from where f1 like '%abc%' and f2 >= '2024-01-01' and f2< '2024-2-1'
```

```
select * from where f1 like '%abc%' and f2 >= '2024-02-01' and f2< '2024-3-1'
```

...

# 全实例mysqldump, 指定库restore

有哪些跳过不需要的库的方案

1. 文本处理
2. 权限方案
3. blackhole

a - b1(blackhole) - c1

- c2

- b2(blackhole) - c3

- c4

# 课堂练习: 停机时间本地拆表

表 t 有一个 shopid 字段, 约 1 亿行, 现在要按照奇偶分成两张表 t1、t2, 为以后拆库做准备。

可以跟业务一起停机, 业务修改代码, db 做变更,

问: db 端最快的操作方案是什么样的?

设计一个比以下方案速度快的方案:

```
create table t2 like t;
```

```
insert into t2 select * from t where shopid %2 = 0;
```

```
delete from t where shopid %2 = 0;
```

```
rename table t to t1;
```

# 勘误

```
mysql> show create table t2\G
***** 1. row *****
      Table: t2
Create Table: CREATE TABLE `t2` (
  `id` int NOT NULL,
  `e` int DEFAULT NULL,
  `f` int DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
1 row in set (0.00 sec)
```



# 勘误

```
mysql> explain select * from t2 where id>'1a';
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	t2	NULL	range	PRIMARY	PRIMARY	4	NULL	2	100.00	Using where

```
1 row in set, 2 warnings (0.00 sec)
```

```
mysql> show warnings;
```

Level	Code	Message
Warning	1292	Truncated incorrect DOUBLE value: '1a'
Note	1003	/* select#1 */ select `test`.`t2`.`id` AS `id`,`test`.`t2`.`e` AS `e`,`test`.`t2`.`f` AS `f` from `test`.`t2` where (`test`.`t2`.`id` > 1)

2 rows in set (0.01 sec)

# Q&A

THANKS