

更多资源v: 2938843375

更多资源v: 2938843375

慢查询诊断问题

更多资源v: 2938843375

更多资源v: 2938843375

目录

1 慢查询相关参数和建议配置

2 如何判断 SQL 的主要消耗阶段

3 调优思路

4 案例分析

慢查询分析思路

1. 确认是锁还是执行消耗
2. 对于执行消耗，分析执行过程

更多资源v: 2938843375

更多资源v: 2938843375

基础知识回顾

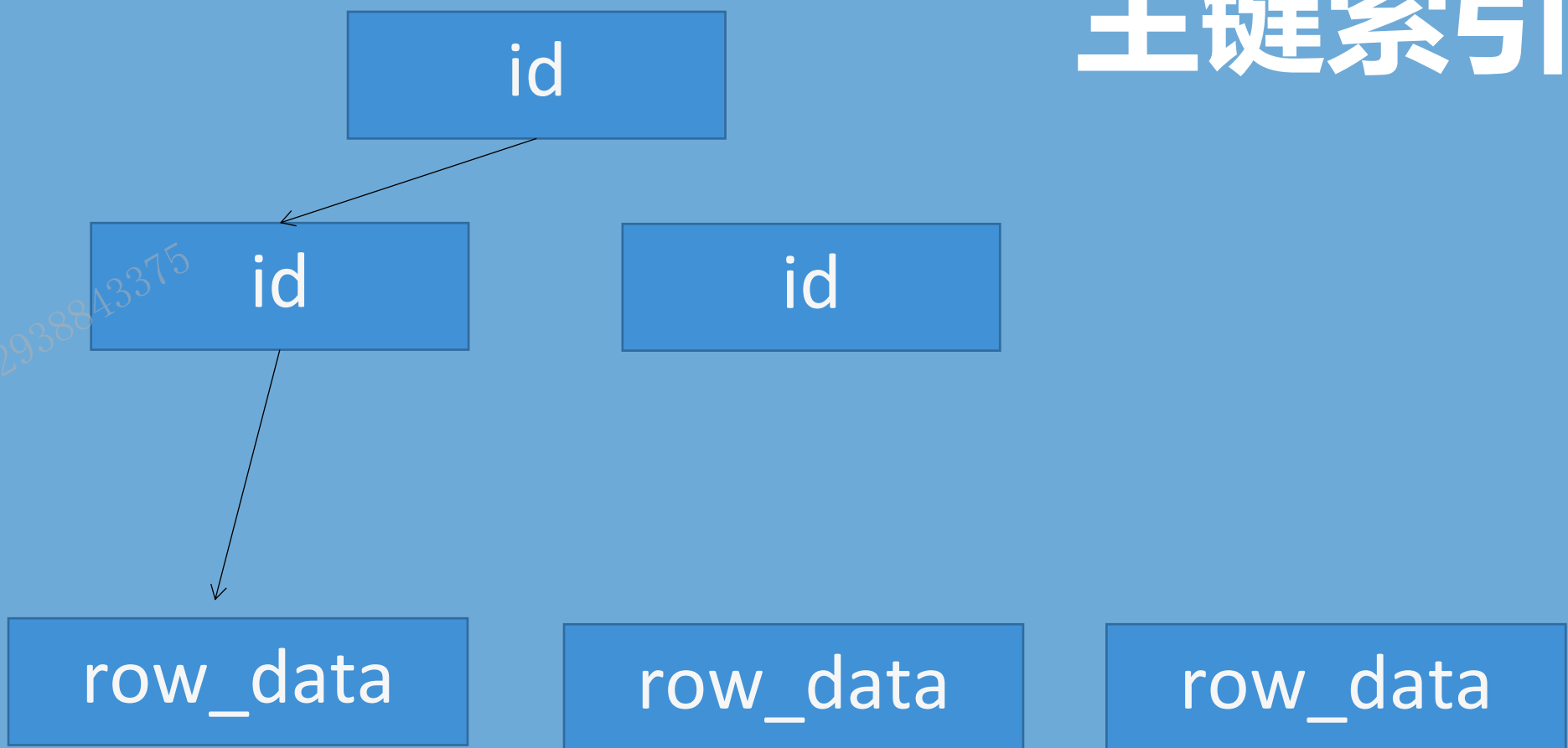
更多资源v: 2938843375

更多资源v: 2938843375

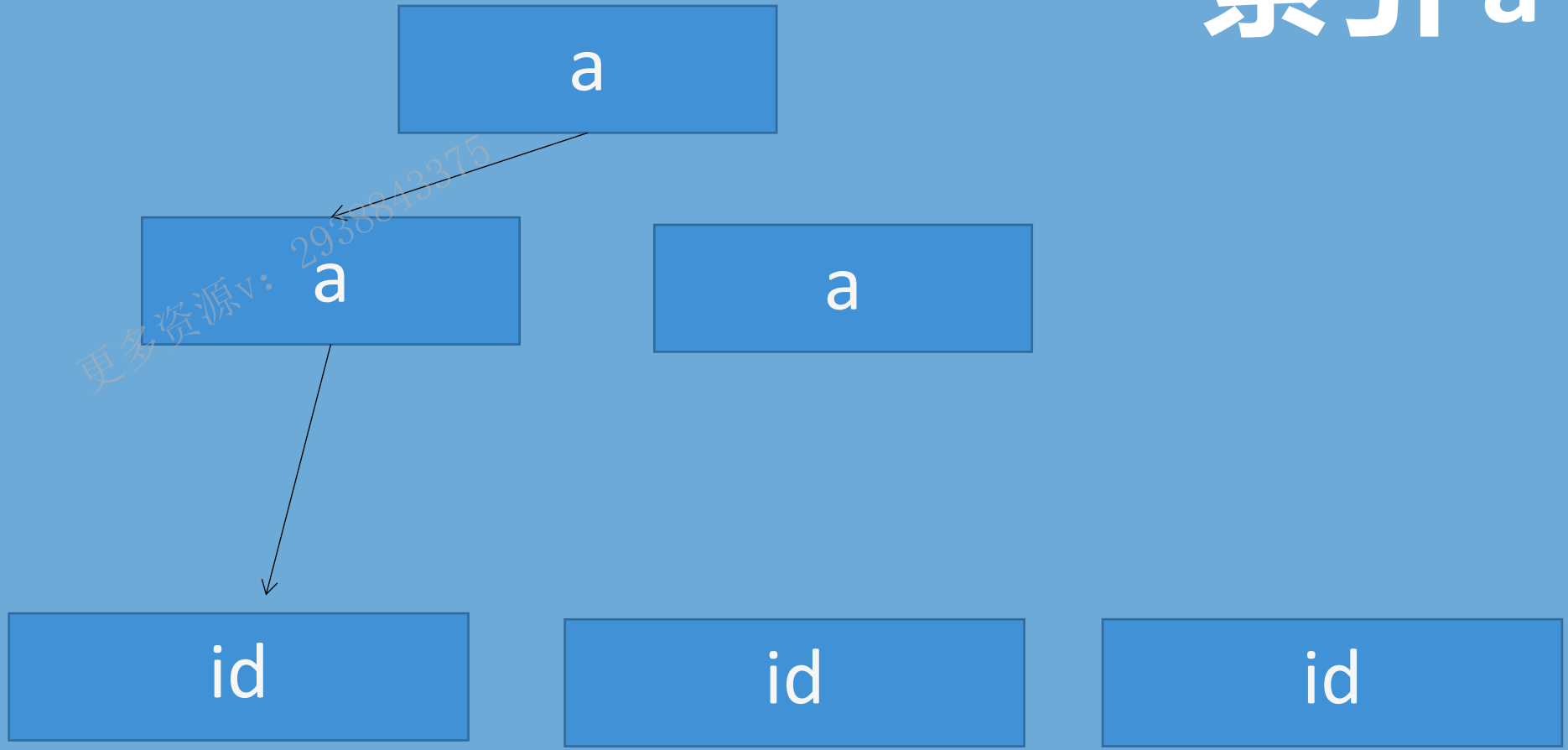
InnoDB 聚簇索引结构

```
t1 | CREATE TABLE `t1` (  
  `id` int NOT NULL AUTO_INCREMENT,  
  `a` int DEFAULT NULL,  
  `b` int DEFAULT NULL,  
  `c` int DEFAULT NULL,  
  PRIMARY KEY (`id`),  
  KEY `a` (`a`),  
  KEY `bc` (`b`,`c`)  
  ENGINE=InnoDB DEFAULT CHARSET=utf8mb4  
  )
```

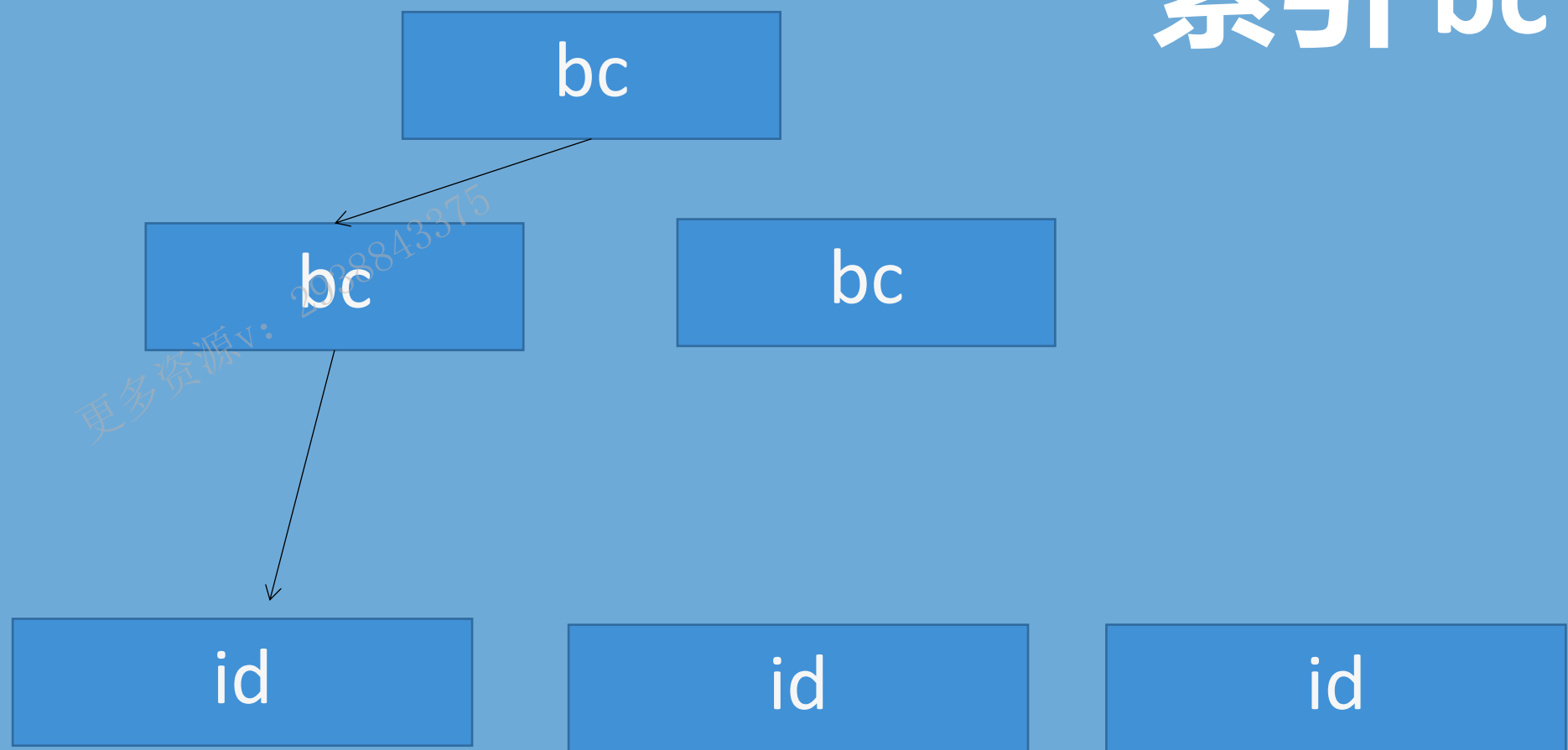
主键索引



索引 a



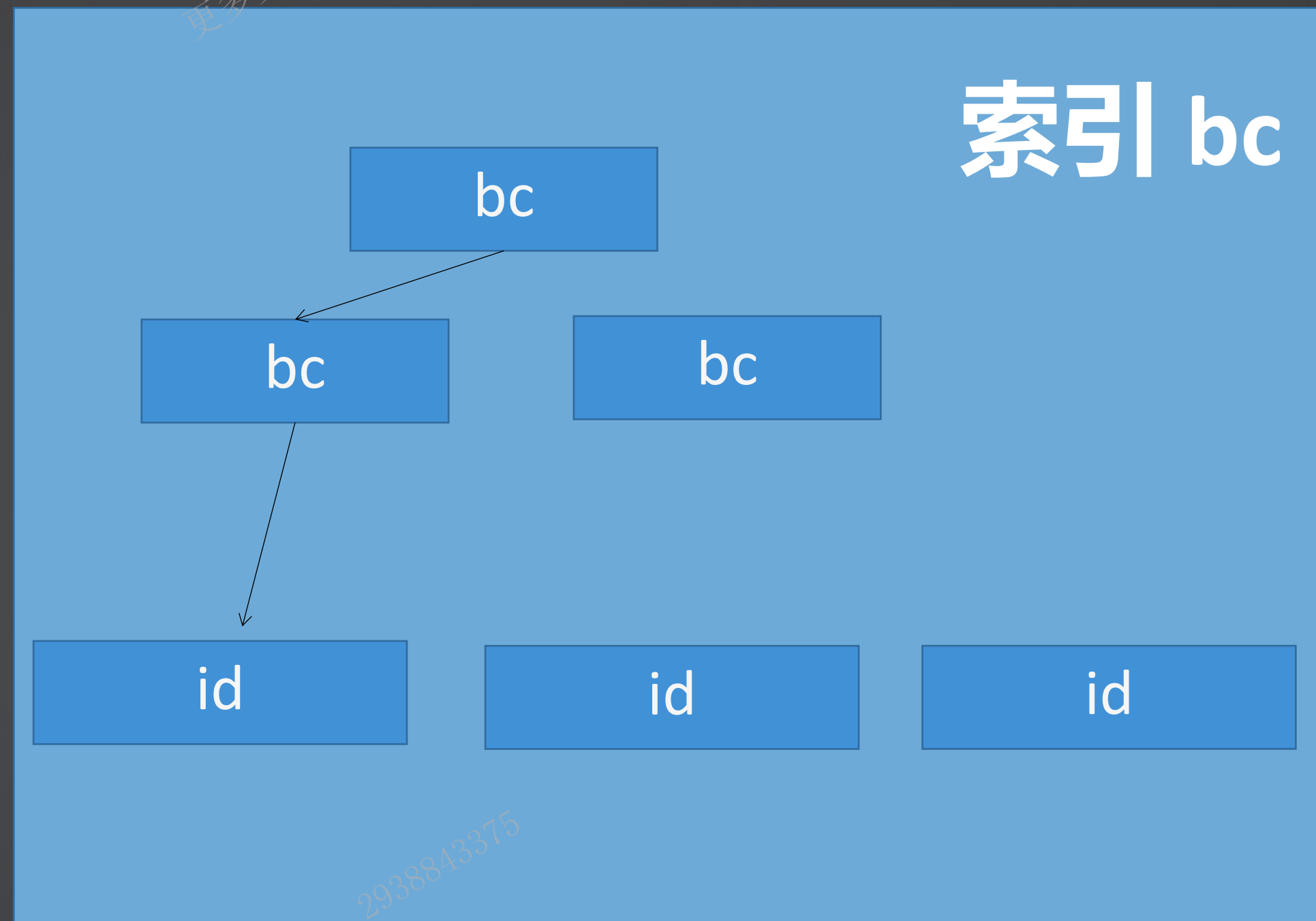
索引 bc



执行器单元操作演示及相应的数据结构

B+ 树上的操作：

- index lookup
- index range scan
- table scan
- index skip scan



执行器单元操作演示及相应的数据结构

filter

temporary/materialize

aggregate:

- group by
- distinct
- sum/avg

group by

c	count(*)
n1	X
n2	Y
n3	Z

distinct

c
n1
n2
n3

执行器单元操作演示及相应的数据结构

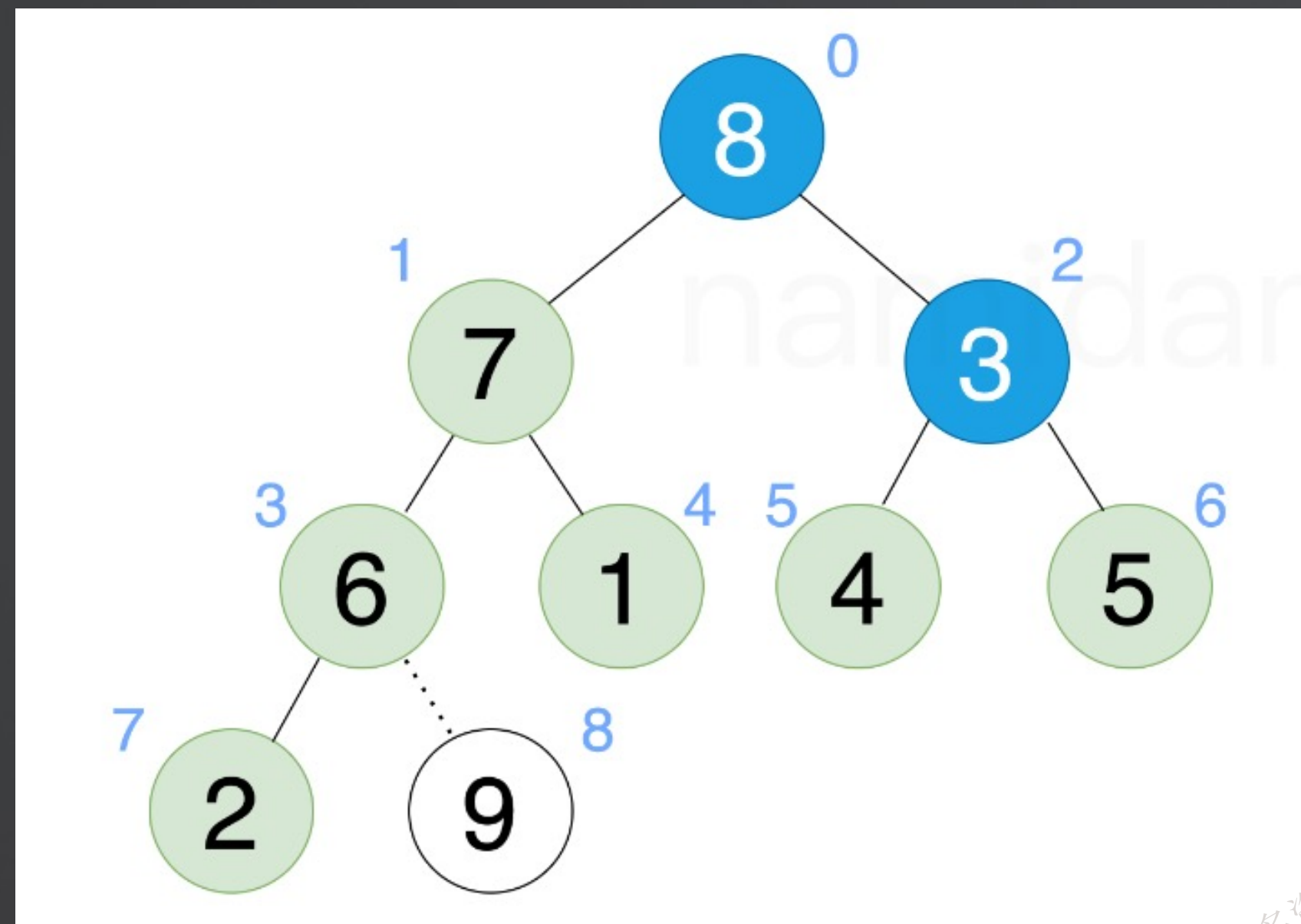
排序

- qsort
- priority queue
- merge sort

$M * \log_2(M)$

优先队列排序（堆排序）

where ... limit N

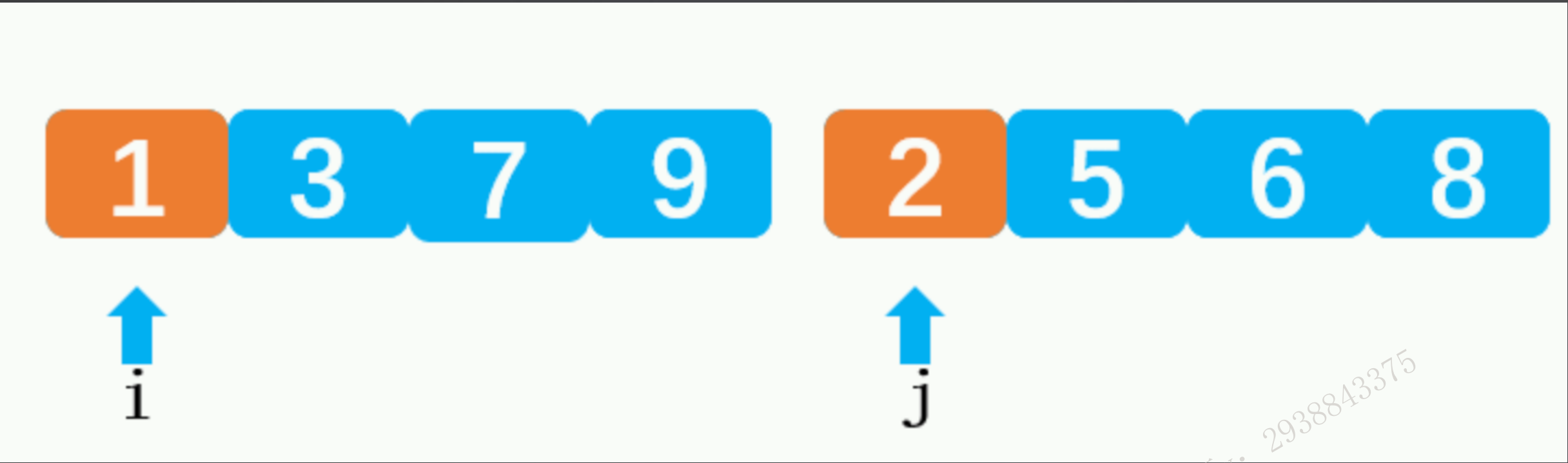


$\lg_2(N)$

range scan (M) \longrightarrow 大顶堆 $M * \lg_2(N)$

归并排序

1 2 3 5



更多资源v: 2938843375

更多资源v: 2938843375

```
create table t1(id int primary key auto_increment, a int , b int,  
c int , index (a), index bc(b,c))engine=innodb;
```

```
insert into t1(a,b,c) values(1,1,1),(2,2,2),(3,3,3),(4,4,4);
```

```
insert into t1(a,b,c) select a,b,c from t1; //重复插入到16384行
```

```
insert into t1(a,b,c) values(5,5,5);
```

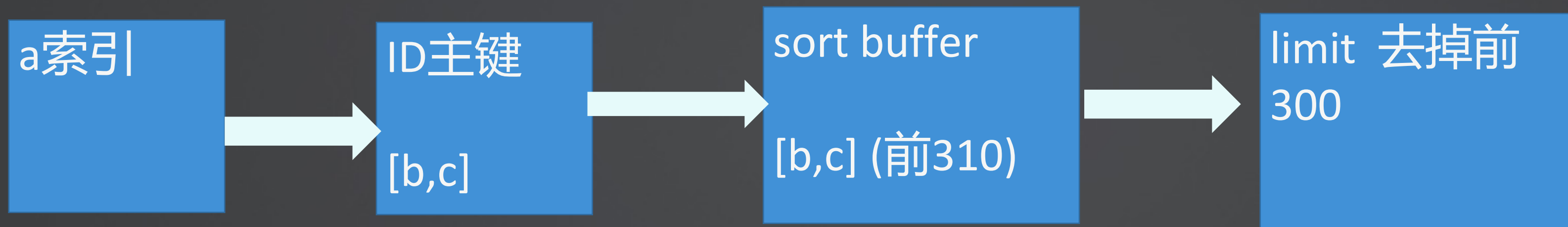
更多资源v: 2938843375

更多资源v: 2938843375

```
insert into t1(a,b,c) select a,b,c from t1;
```

```
insert into t1(a,b,c) select a,b,c from t1; //总共65540行
```

select c from t1 where a=3 and b>1 order by b limit 300,10\G



更多资源v: 2938843375

更多资源v: 2938843375

explain analyze select c from t1 where a=3 and b>1 order by b limit 300,10\G

```
mysql> explain analyze select c from t1 where a=3 and b>1 order by b limit 300,10\G
***** 1. row *****
EXPLAIN: -> Limit/Offset: 10/300 row(s) (cost=1631 rows=10) (actual time=24.1..24.1 rows=10 loops=1)
  -> Sort: t1.b, limit input to 310 row(s) per chunk (cost=1631 rows=30206) (actual time=24.1..24.1 rows=310 loops=1)
    -> Filter: (t1.b > 1) (cost=1631 rows=30206) (actual time=0.186..22.5 rows=16384 loops=1)
      -> Index lookup on t1 using a (a=3) (cost=1631 rows=30206) (actual time=0.185..21.4 rows=16384 loops=1)

1 row in set (0.03 sec)
```

更多资源v: 2938843375

更多资源v: 2938843375

更多资源v: 2938843375

```
{
  "join_execution": {
    "select#": 1,
    "steps": [
      {
        "sorting_table": "t1",
        "filesort_information": [
          {
            "direction": "asc",
            "expression": "`t1`.`b`"
          }
        ],
        "filesort_priority_queue_optimization": {
          "limit": 310,
          "chosen": true
        },
        "filesort_execution": [
        ],
        "filesort_summary": {
          "memory_available": 262144,
          "key_size": 9,
          "row_size": 22,
          "max_rows_per_buffer": 311,
          "num_rows_estimate": 29006,
          "num_rows_found": 16384,
          "num_initial_chunks_spilled_to_disk": 0,
          "peak_memory_used": 9330,
          "sort_algorithm": "std::stable_sort",
          "unpacked_addon_fields": "using_priority_queue",
          "sort_mode": "<fixed_sort_key, additional_fields>"
        }
      ]
    }
  }
}
```

更多资源v: 2938843375

更多资源v: 2938843375

更多资源v: 2938843375

更多资源v: 2938843375

更多资源v: 2938843375

双表

更多资源v: 2938843375

更多资源v: 2938843375

基础知识：Inner / left / right / outter join 语义和流程图

```
CREATE TABLE `t1` (  
  `id` int(11) NOT NULL,  
  `a` int(11) DEFAULT NULL,  
  `b` int(11) DEFAULT NULL,  
  PRIMARY KEY (`id`),  
  KEY `a` (`a`)  
) ENGINE=InnoDB;
```

```
create table t2 like t1;  
insert into t1  
values(2,2,2),(4,4,4),(6,6,6),(8,8,8);  
insert into t2  
values(3,3,3),(6,6,6),(9,9,9);
```

基础知识：Inner / left / right / outter 语义

```
mysql> select * from t1 join t2 on t1.id=t2.id;
```

id	a	b	id	a	b
6	6	6	6	6	6

1 row in set (0.00 sec)

```
mysql> select * from t1 left join t2 on t1.id=t2.id;
```

id	a	b	id	a	b
2	2	2	NULL	NULL	NULL
4	4	4	NULL	NULL	NULL
6	6	6	6	6	6
8	8	8	NULL	NULL	NULL

4 rows in set (0.01 sec)

```
mysql> select * from t1 right join t2 on t1.id=t2.id;
```

id	a	b	id	a	b
NULL	NULL	NULL	3	3	3
6	6	6	6	6	6
NULL	NULL	NULL	9	9	9

3 rows in set (0.00 sec)

基础知识：Inner / left / right / outter 语义

```
mysql> select * from t1 left join t2 on t1.id=t2.id union select * from t1 right join t2 on t1.id=t2.id;
```

id	a	b	id	a	b
2	2	2	NULL	NULL	NULL
4	4	4	NULL	NULL	NULL
6	6	6	6	6	6
8	8	8	NULL	NULL	NULL
NULL	NULL	NULL	3	3	3
NULL	NULL	NULL	9	9	9

6 rows in set (0.00 sec)

```
mysql> select * from t1 left join t2 using(id);
```

id	a	b	a	b
2	2	2	NULL	NULL
4	4	4	NULL	NULL
6	6	6	6	6
8	8	8	NULL	NULL

4 rows in set (0.00 sec)

资源v: 2938845465

更多资源v: 29388493

更多资源v: 29388493

更多资源v: 29388493

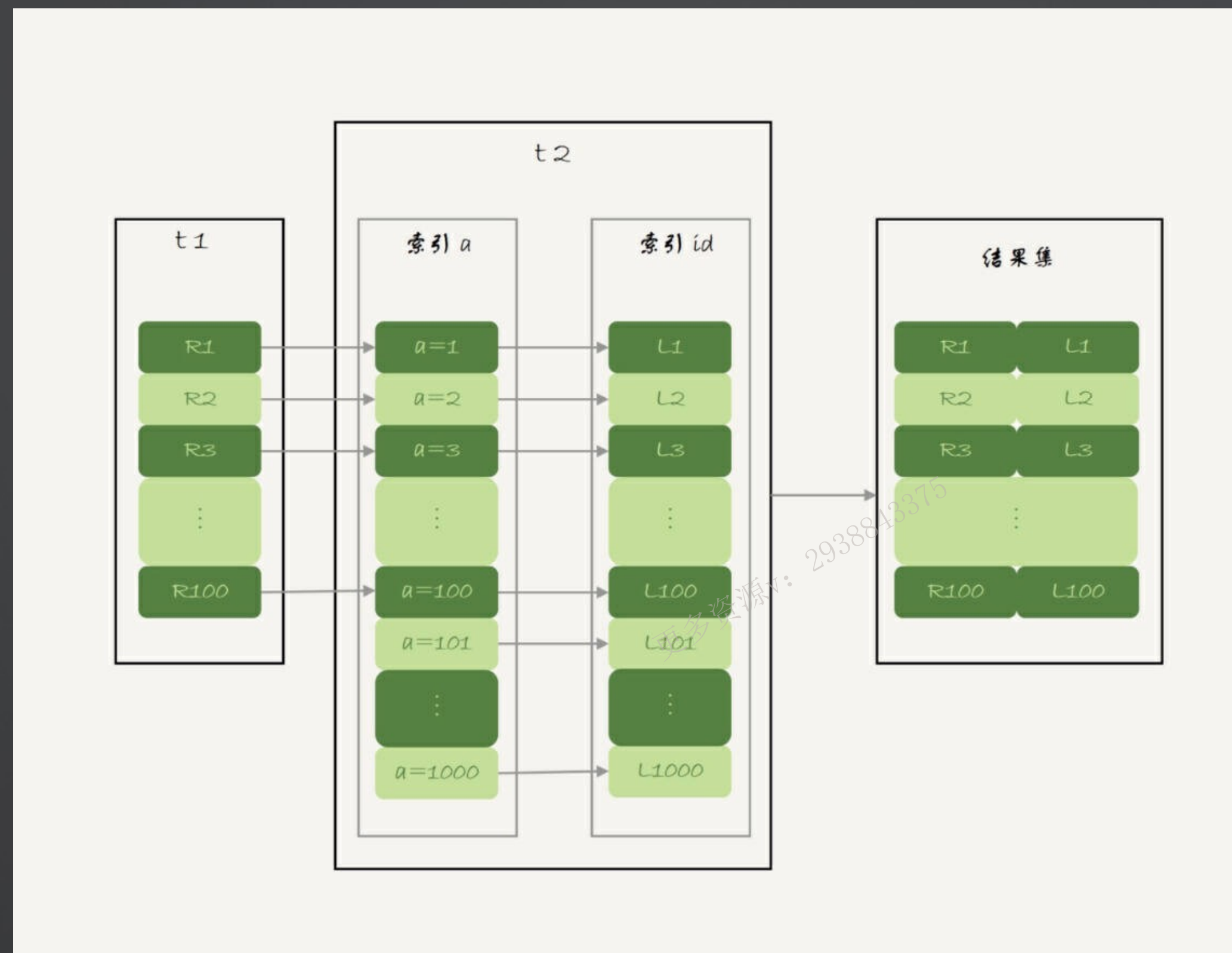
更多资源v: 29388493

更多资源v: 29388493

更多资源v: 29388493

基础知识：Nested Loop join 算法流程和代价分析

```
select * from t1 join t2 using (a);
```



基础知识：straight_join

```
select * from t2 join t1 using (a);
```

```
mysql> explain select * from t2 join t1 using (a);
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	t1	NULL	ALL	a	NULL	NULL	NULL	100	100.00	Using where
1	SIMPLE	t2	NULL	ref	a	a	5	test.t1.a	1	100.00	NULL

```
2 rows in set, 1 warning (0.00 sec)
```

```
mysql> explain select * from t2 straight_join t1 using (a);
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	t2	NULL	ALL	a	NULL	NULL	NULL	1000	100.00	Using where
1	SIMPLE	t1	NULL	ref	a	a	5	test.t2.a	1	100.00	NULL

```
2 rows in set, 1 warning (0.00 sec)
```

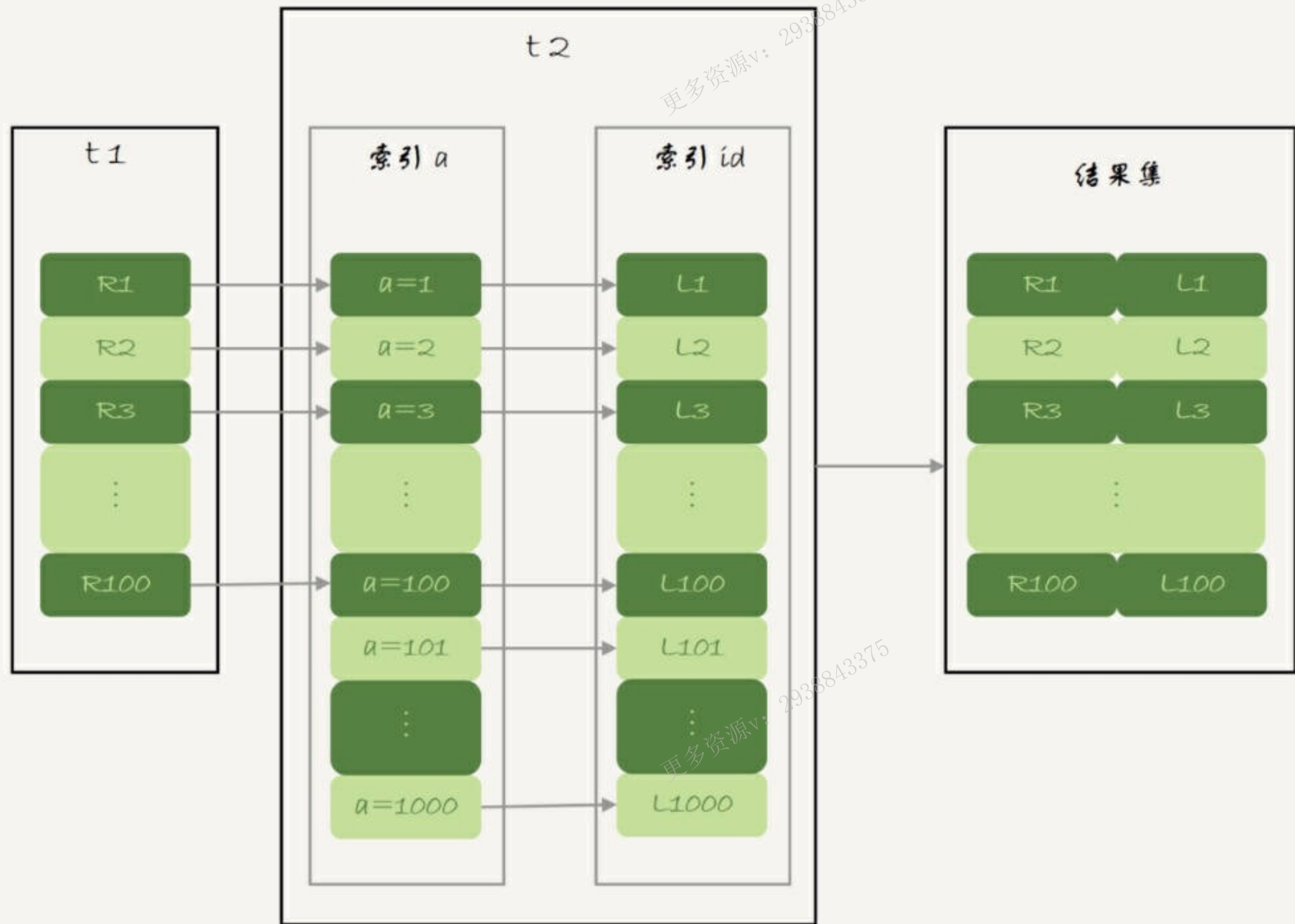
基础知识：semijoin

explain analyze select * from t1 where a in (select a from t2)\G

```
mysql> explain select * from t1 where a in (select a from t2);
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | t1 | NULL | ALL | a | NULL | NULL | NULL | 100 | 100.00 | Using where |
| 1 | SIMPLE | t2 | NULL | ref | a | a | 5 | test.t1.a | 1 | 100.00 | Using index; FirstMatch(t1) |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

2 rows in set, 1 warning (0.00 sec)

mysql> show warnings\G
***** 1. row *****
Level: Note
Code: 1003
Message: /* select#1 */ select `test`.`t1`.`id` AS `id`,`test`.`t1`.`a` AS `a`,`test`.`t1`.`b` AS `b` from `test`.`t1` semi join (`test`.`t2`
) where (`test`.`t2`.`a` = `test`.`t1`.`a`)
1 row in set (0.00 sec)
```

基础知识：anti join

`select * from t1 where id not in (select id from t2);`

```
mysql> explain analyze select * from t1 where id not in (select id from t2)\G
***** 1. row *****
EXPLAIN: -> Nested loop antijoin (cost=45.2 rows=100) (actual time=9.1..9.1 rows=0 loops=1)
    -> Table scan on t1 (cost=10.2 rows=100) (actual time=0.547..0.806 rows=100 loops=1)
    -> Single-row covering index lookup on t2 using PRIMARY (id=t1.id) (cost=0.251 rows=1) (actual time=0.0823..0.0823 rows=1 loops=100)

1 row in set (0.01 sec)
```

更多资源v: 2938843375

更多资源v: 2938843375

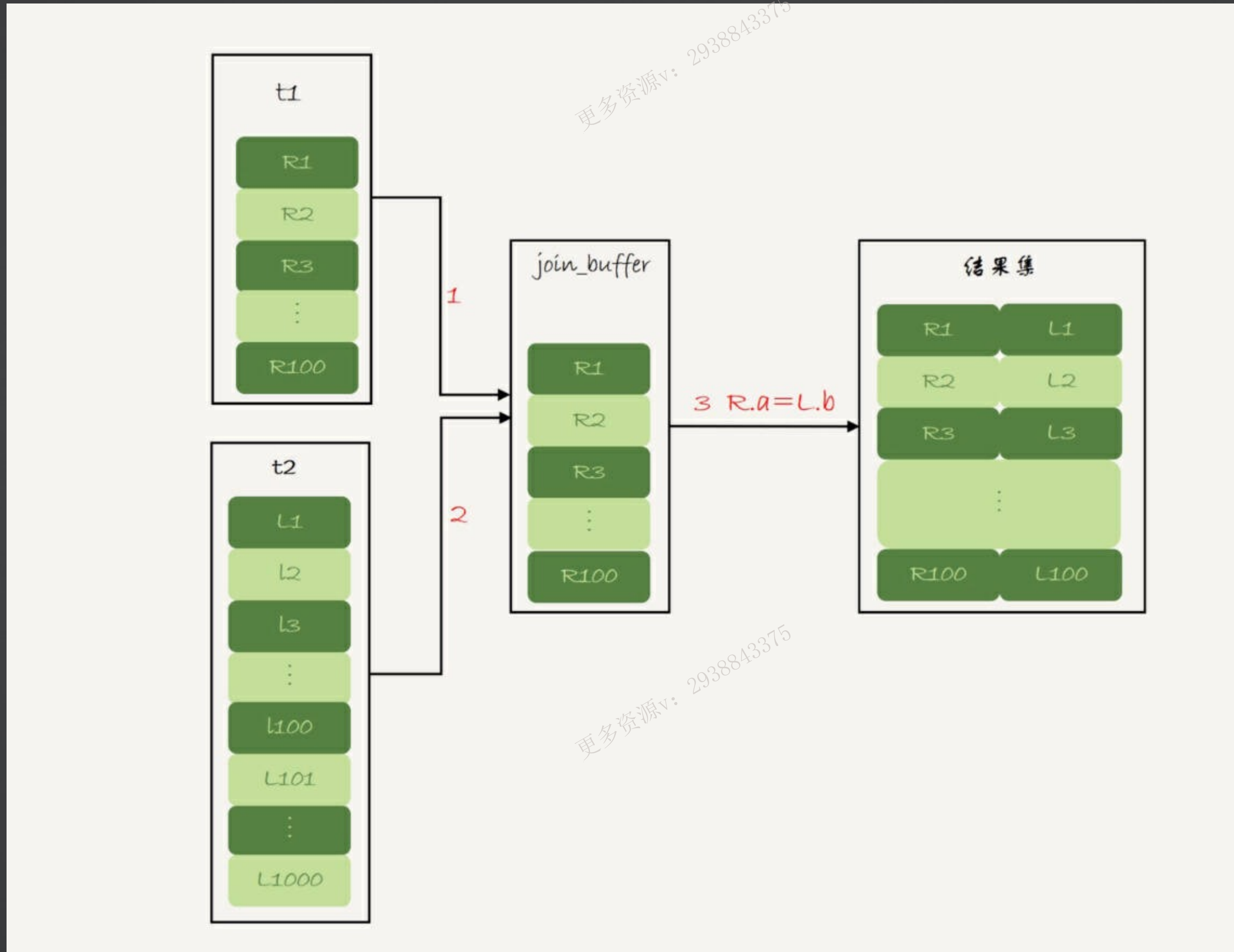
问个问题: join_buffer_size 怎么还没上场?

更多资源v: 2938843375

更多资源v: 2938843375

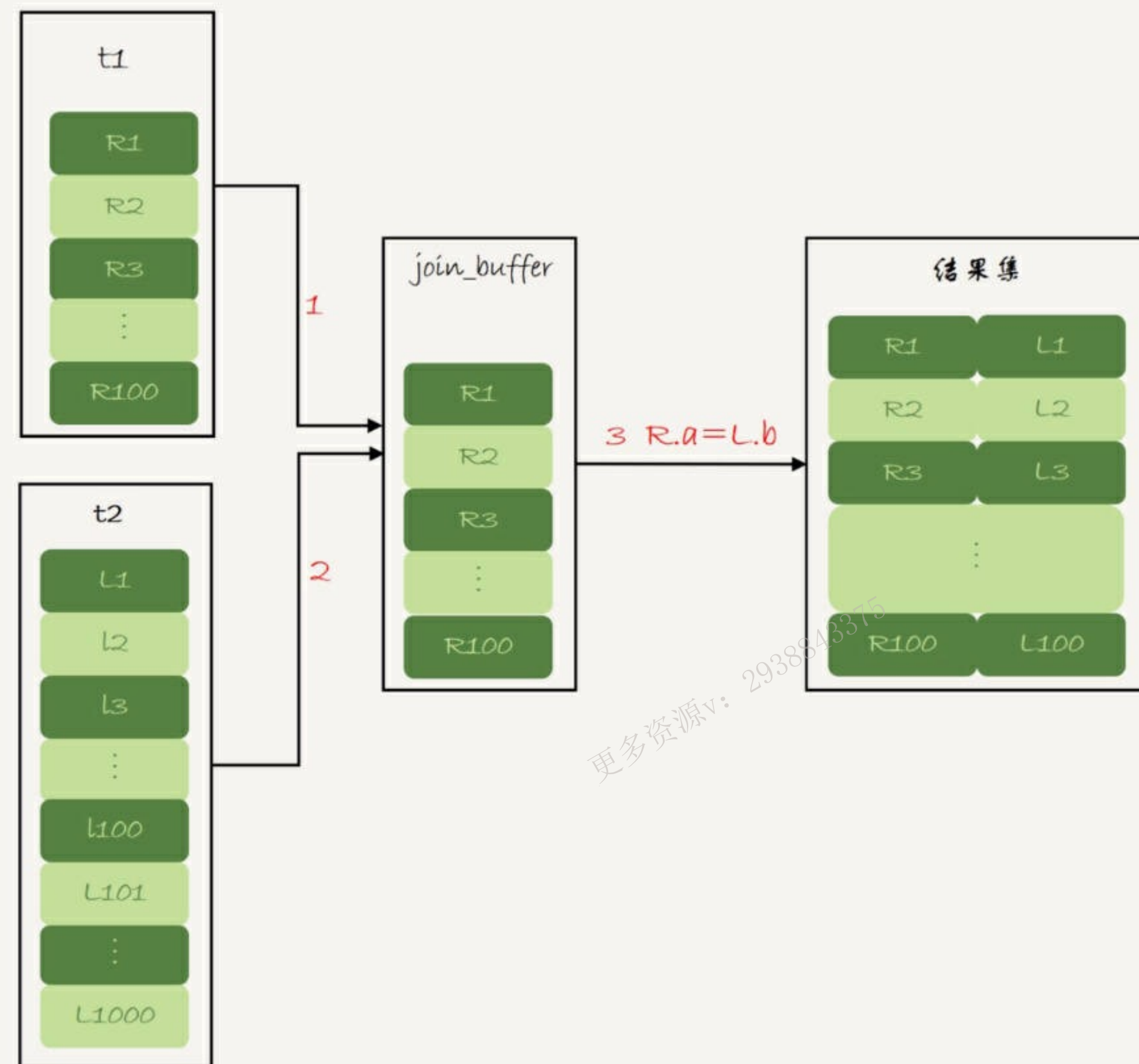
`select * from t1 join t2 using (b);`

MySQL 8.0 之前



基础知识： Hash join 算法流程和代价分析

```
select * from t1 join t2 using (b);
```



更多资源v: 2938843375

更多资源v: 2938843375

```
mysql> explain select * from t2 join t1 using (b);
```

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	t1	NULL	ALL	NULL	NULL	NULL	NULL	100	100.00	NULL
1	SIMPLE	t2	NULL	ALL	NULL	NULL	NULL	NULL	1000	10.00	Using where; Using join buffer (hash join)

2 rows in set, 1 warning (0.00 sec)

更多资源v: 2938843375

更多资源v: 2938843375

更多资源v: 2938843375

更多资源v: 2938843375

案例分析

更多资源v: 2938843375

更多资源v: 2938843375

更多资源v: 2938843375

```
create table users(id int primary key auto_increment, nickname varchar(255));
```

更多资源v: 2938843375

```
create table orders(id int primary key auto_increment, order_id int, customerid  
int, sellerid int, lastmodified datetime, info text, index(sellerid));
```

假设 orders 有100w行，其中sellerid=1的有10w行，现在要查出sellerid=1 的所有订单和买家信息

更多资源v: 2938843375

更多资源v: 2938843375

更多资源v: 2938843375

更多资源v: 2938843375

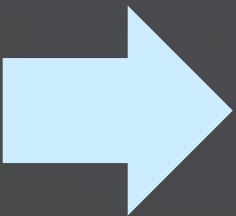
假设 orders 有100w行， 其中sellerid=1的有10w行， 现在要查出sellerid=1 的所有订单和买家信息

```
select * from orders o join user u on o.customerid=u.id where o.sellerid=1;
```

更多资源v: 2938843375

更多资源v: 2938843375

Orders



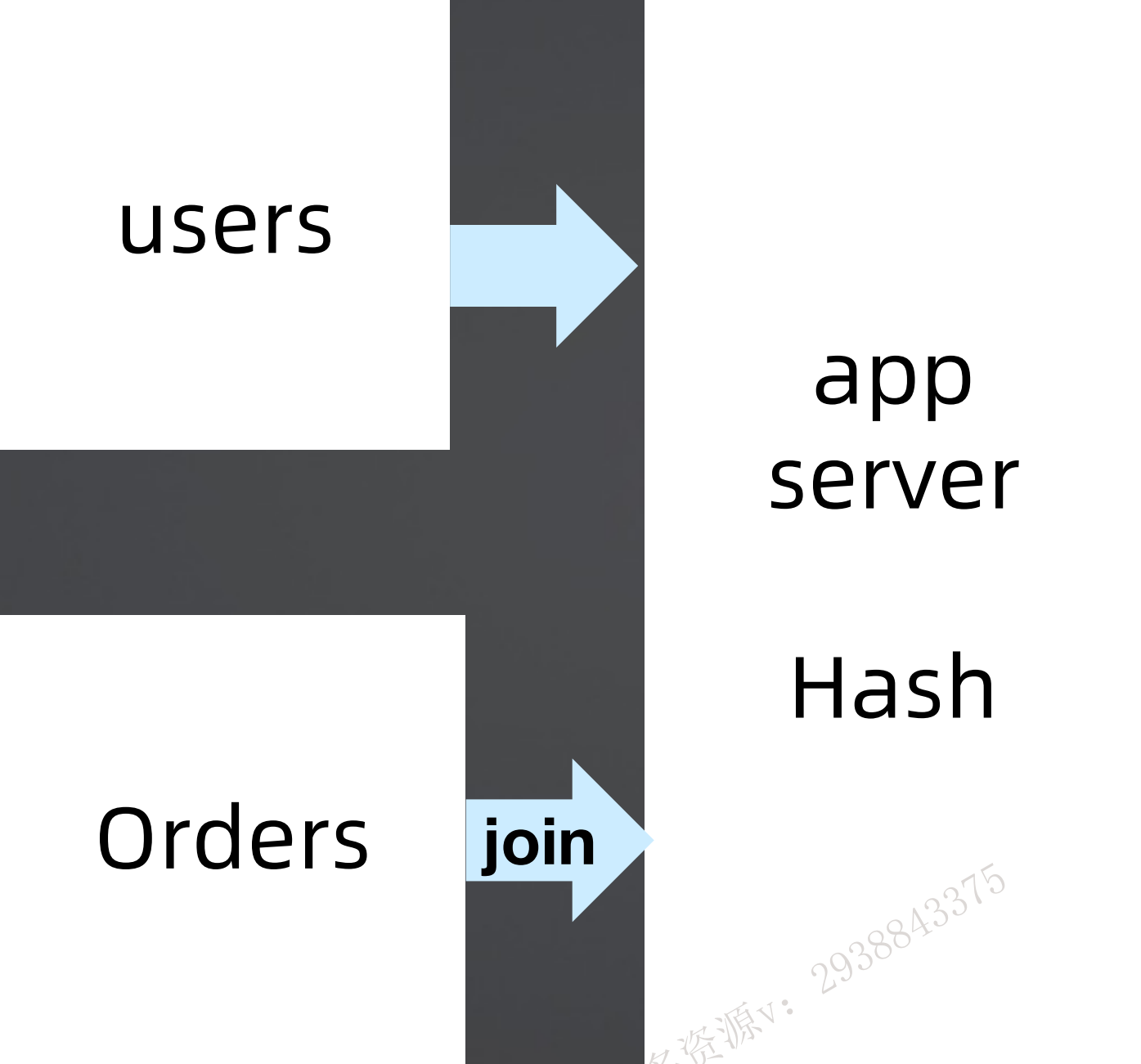
users

更多资源v: 2938843375

更多资源v: 2938843375

更多资源v: 2938843375

更多资源v: 2938843375



更多资源v: 2938843375

更多资源v: 2938843375

Q & A

更多资源v: 2938843375

更多资源v: 2938843375

更多资源v: 2938843375

更多资源v: 2938843375

THANKS

更多资源v: 2938843375

更多资源v: 2938843375