# 第五次实验

# 22371437 张智威

Q1

alter table bdcopy1 add index Q1(v)

#### 等值查询语句为:

select \* from bdcopy1(2) where v = 1001

#### 从bdcopy1表中查询

 时间
 类型
 文本
 持续时间(毫秒)
 行
 结果

 4月-25 19:39:19
 SQL / User
 select \* from bdcopy1¶where v = 1001¶LIMIT 0, 800000
 90
 9
 成功

# 从bdcopy2表中查询

 时间
 类型
 文本
 持续时间(毫秒)
 行
 结果

 4月-25 19:40:35
 SQL / User
 select \* from bdcopy2¶where v = 1001¶LIMIT 0, 800000
 349
 9
 成功

#### 范围查询语句为:

select \* from bdcopy1(2) where v>10000

#### 从bdcopy1表中查询

 时间
 类型
 文本
 持续时间(毫秒)
 行
 结果

 4月-25 19:38:16
 SQL / User
 select \* from bdcopy1 where v>10000¶LIMIT 0, 800000
 537
 719382
 成功

# 从bdcopy2表中查询

 时间
 类型
 文本
 持续时间 (毫秒)
 行
 结果

 4月-25 19:37:08
 SQL / User
 select \* from bdcopy2 where v>10000¶LIMIT 0, 800000
 6,377
 719382
 成功

对于B+树索引,无论是等值查找还是范围查找,其速度均显著快于不使用索引, 这是由于对于两种查找方式,不使用索引都需要扫描表中每一行数据来逐一判断是否 满足要求,而使用了B+树索引只需要根据树的每一层节点值的划分情况,来根据需要

寻找所满足的叶子节点,不需要对每一行数据都进行查询,节省了大量存储1/0次数,从而节约了大量时间。

Q2

# 等值查询语句为:

select \* from bdcopy1(3)
where v = 1001

#### 从bdcopy1表中查询

时间	类型	文本	持续时间(毫秒)	行	结果
4月-25 20:20:56	SQL / User	select * from bdcopy1¶where v = 1001¶LIMIT 0, 800000	83	9	成功

#### 从bdcopy3表中查询

时间	类型	文本	持续时间(毫秒)	行	结果
4月-25 20:06:12	SQL / User	select * from bdcopy3¶where v = 1001¶LIMIT 0, 800000	82	9	成功

# 从bdcopy4表中查询

时间	类型	文本	持续时间(毫秒)	行	结果
4月-25 20:18:56	SQL / User	select * from bdcopy4¶where v = 1001¶LIMIT 0, 800000	95	2	成功

#### 从bdcopy5表中查询

时间	类型	文本	持续时间 (毫秒)	行	结果
4月-25 20:19:18	SQL / User	select * from bdcopv5¶where v = 1001¶LIMIT 0, 800000	171	2	成功

bdcopy1查询时间与bdcopy3中近乎相等,这是由于MySql默认的InnoDB引擎不支持hash索引,此处实际建立了B+索引,两个表的索引相同。而bdcopy4查询时间显著短于bdcopy5,这是由于在相同数据的情况下,由于有Hash索引的存在,只需要计算哈希值然后在Hash桶中进行判断即可,不需要所有数据全部查询,大大节省了时间。

Q3

# 等值查询语句为:

select \* from bdcopy1(23) where v = 399998

# 从bdcopy1表中查询

时间	类型	文本	持续时间(毫秒)	行	结果
4月-25 20:33:07	SQL / User	select * from bdcopy1¶where bid = 399998¶LIMIT 0, 800000	88	1	成功

#### 从bdcopy2表中查询

时间	类型	文本	持续时间 (毫秒)	行	结果
4月-25 20:33:58	SQL / User	select * from bdcopy2¶where bid = 399998¶LIMIT 0, 800000	89	1	成功

# 范围查询语句为:

```
select * from bdcopy1(2)
where bid between 10000 and 120000
```

# 从bdcopy1表中查询

时间	类型	文本	持续时间 (毫秒)	行	结果
4月-25 20:40:38	SQL / User	select * from bdcopy1¶where bid between 10000 and 12000	1,213	110001	成功

# 从bdcopy2表中查询

时间	类型	文本	持续时间(毫秒)	行	结果
4月-25 20:45:42	SQL / User	select * from bdcopy2¶where bid between 10000 and 12000	911	110001	成功

等值查找时,聚簇索引和B+树索引时间差距不大,而在进行范围查找时,由于聚 簇索引数据按照物理顺序排放,在聚簇索引的列上进行范围查找时只需要查找少数几个页,使得聚簇索引速度显著大于B+树索引。

Q4

#### Test1

```
select * from bdcopy1(2)
where v > 10001 and s = 7
```

# 从bdcopy1表中查询

时间	类型	文本	持续时间(毫秒)	行	结果
4月-25 21:06:22	SQL / User	select * from bdcopy1¶where v > 10001 and s = 7¶LIMIT 0, 8	875	72058	成功

# 从bdcopy2表中查询

时间	类型	文本	持续时间(毫秒)	行	结果
4月-25 21:07:07	SQL / User	select * from bdcopy2¶where v > 10001 and s = 7¶LIMIT 0, 8	714	72058	成功

#### Test2

```
select * from bdcopy1(2)
where v > 10001
```

#### 从bdcopy1表中查询

时间	类型	文本	持续时间 (毫秒)	行	结果
4月-25 21:08:22	SQL / User	select * from bdcopy1¶where v > 10001¶LIMIT 0, 800000	6,093	719375	成功

#### 从bdcopy2表中查询

时间	类型	文本	持续时间 (毫秒)	行	结果
4月-25 21:08:01	SQL / User	select * from bdcopy2¶where v > 10001¶LIMIT 0, 800000	5,926	719375	成功

#### Test3

```
select * from bdcopy1(2)
where s = 7
```

#### 从bdcopy1表中查询

时间	类型	文本	持续时间(毫秒)	行	结果
4月-25 21:20:17	SQL / User	select * from bdcopy1¶where s = 7¶LIMIT 0, 800000	800	80140	成功

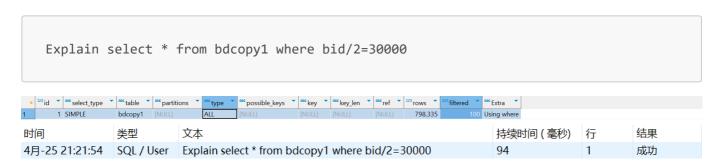
#### 从bdcopy2表中查询

时间	类型	文本	持续时间(毫秒)	行	结果
4月-25 21:09:40	SQL / User	select * from bdcopy2¶where s = 7¶LIMIT 0, 800000	807	80140	成功

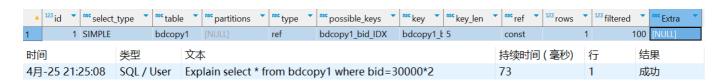
Test1中,对两列同时进行条件判断查找,联合索引由于最左前缀的优势,查询速度较快。Test2中,仅对联合索引中最左列进行查找,联合索引等同于一个B+树索引,两张表查询速度等同,Test3中对非联合索引最左列进行查找,两个表相当于在此列中均未设立索引,所以查询速度相当。

Q5

#### Test1



Explain select \* from bdcopy1 where bid=30000\*2



Test1中,查询type为all,即实际上全部数据都查询了一次,Test2中,查询type为ref,表示使用了索引进行扫描。原因是第一种写法需要对每一行数据的bid进行除以2的操作再与30000比较,因此无法使用B+树索引,而第二个写法就相当于与60000比较,可以使用B+树索引,因此第二个的时间也比第一个的时间短。