收获，不止 SQL 优化

第 十 章

且慢，其他索引应用让 SQL 飞

E-Mail:45240040@qq.com

目录

1.说说位图索引的优势和不适合使用的场景

1.1 位图索引的优势

位图索引与其他索引不同，它不是存储的索引列的列值，而是以比特位 0、1 的形式存储，所以在空间上它占的空间比较小，相应的一致性查询所使用的数据块也比较小，查询的效率就会比较高。

1.1.1 高效即席查询

```
SQL> create table t
  2  (name_id,
  3  gender not null,
  4  location not null,
  5  age_group not null,
  6  data
  7  )  as
  8  select rownum,
  9  decode(ceil(dbms_random.value(0,2)),
 10  1,'M',
 11  2,'F')gender,
 12  ceil(dbms_random.value(1,50)) location,
 13  decode(ceil(dbms_random.value(0,3)),
 14  1,'child',
 15  2,'young',
 16  3,'middle_age',
 17  4,'old'),
 18  rpad('*',400,'*')
 19  from dual
 20  connect by rownum<=100000;

Table created.

SQL> begin
  2  dbms_stats.gather_table_stats(ownname =>'SCOTT',
  3  tabname => 'T',
  4  estimate_percent => 10,
  5  method_opt=> 'for all indexed columns',
  6  cascade=>TRUE);
  7  end;
  8  /

PL/SQL procedure successfully completed.
```

```
SQL> begin
  2  dbms_stats.gather_table_stats(ownname =>'SCOTT',
  3  tabname => 'T',
  4  estimate_percent => 10,
  5  method_opt=> 'for all indexed columns',
  6  cascade=>TRUE);
  7  end;
  8  /

PL/SQL procedure successfully completed.

SQL> set lines 1000
SQL> set autotrace traceonly
SQL> select *  from t where gender='M' and location in (1,10,30) and age_group='child';

647 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 306189815

-------------------------------------------------------------------------------------------
| Id  | Operation                     | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |            |   695 |   154K|   685   (0)| 00:00:09 |
|   1 |  INLIST ITERATOR              |            |       |       |            |          |
|   2 |   TABLE ACCESS BY INDEX ROWID| T          |   695 |   154K|   685   (0)| 00:00:09 |
|*  3 |    INDEX RANGE SCAN           | IDX_UNION  |   695 |       |     4   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------
```

```
Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        680  consistent gets
          0  physical reads
          0  redo size
     282451  bytes sent via SQL*Net to client
        857  bytes received via SQL*Net from client
         45  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
        647  rows processed
```

采用位图索引查看效率

```
SQL> create bitmap index gender_idx on t(gender);

Index created.

SQL> create bitmap index location_idx on t(location);

Index created.

SQL> create bitmap index age_group_idx on t(age_group);

Index created.

SQL> set autot traceonly
SQL> select * from t where gender='M' and location in (1,10,30)  and age_group='child';

647 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 4134214960

-------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |               |   695 |   154K|   202   (0)| 00:00:03 |
|   1 |  TABLE ACCESS BY INDEX ROWID  | T             |   695 |   154K|   202   (0)| 00:00:03 |
|   2 |   BITMAP CONVERSION TO ROWIDS |               |       |       |            |          |
|   3 |    BITMAP AND                 |               |       |       |            |          |
|*  4 |     BITMAP INDEX SINGLE VALUE | AGE_GROUP_IDX |       |       |            |          |
|   5 |     BITMAP OR                 |               |       |       |            |          |
|*  6 |      BITMAP INDEX SINGLE VALUE| LOCATION_IDX  |       |       |            |          |
|*  7 |      BITMAP INDEX SINGLE VALUE| LOCATION_IDX  |       |       |            |          |
|*  8 |      BITMAP INDEX SINGLE VALUE| LOCATION_IDX  |       |       |            |          |
|*  9 |     BITMAP INDEX SINGLE VALUE | GENDER_IDX    |       |       |            |          |
-------------------------------------------------------------------------------------------------
```

```
Predicate Information (identified by operation id):
---------------------------------------------------

   4 - access("AGE_GROUP"='child')
   6 - access("LOCATION"=1)
   7 - access("LOCATION"=10)
   8 - access("LOCATION"=30)
   9 - access("GENDER"='M')


Statistics
----------------------------------------------------------
         1  recursive calls
         0  db block gets
       637  consistent gets
        40  physical reads
         0  redo size
    282451  bytes sent via SQL*Net to client
       857  bytes received via SQL*Net from client
        45  SQL*Net roundtrips to/from client
         0  sorts (memory)
         0  sorts (disk)
       647  rows processed
```

1.1.2 快速统计条数

```
SQL> create table t as select * from dba_objects;

Table created.

SQL> insert into t select * from t;

9470 rows created.

SQL> /

18940 rows created.

SQL> /

37880 rows created.

SQL> /

75760 rows created.

SQL> /

151520 rows created.

SQL> /

303040 rows created.

SQL> update t set object_id=rownum;

606080 rows updated.

SQL> commit;

Commit complete.
```

```
SQL> create index idx_t_obj on t(object_id);

Index created.

SQL> alter table T modify object_id not null;

Table altered.

SQL> set autotrace traceonly
SQL> select count(*) from t;


Execution Plan
----------------------------------------------------------
Plan hash value: 278572740

--------------------------------------------------------------------------------
| Id  | Operation             | Name       | Rows  | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |            |     1 |   376   (3)| 00:00:05 |
|   1 |  SORT AGGREGATE       |            |     1 |            |          |
|   2 |   INDEX FAST FULL SCAN| IDX_T_OBJ  |   661K|   376   (3)| 00:00:05 |
--------------------------------------------------------------------------------
```

同样条件下普通索引的效率

```
Statistics
----------------------------------------------------------
        205  recursive calls
          0  db block gets
       1473  consistent gets
       1352  physical reads
          0  redo size
        413  bytes sent via SQL*Net to client
        384  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          6  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

```
SQL> create bitmap index idx_bitm_t_status on t(status);

Index created.

SQL> select count(*) from t;


Execution Plan
----------------------------------------------------------
Plan hash value: 4272013625

--------------------------------------------------------------------------------------------
| Id  | Operation                    | Name             | Rows  | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                  |     1 |    14   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE              |                  |     1 |            |          |
|   2 |   BITMAP CONVERSION COUNT    |                  |   661K|    14   (0)| 00:00:01 |
|   3 |    BITMAP INDEX FAST FULL SCAN| IDX_BITM_T_STATUS |       |            |          |
--------------------------------------------------------------------------------------------
```

同样条件下位图索引的查询效率

```
Statistics
----------------------------------------------------------
          5  recursive calls
          0  db block gets
        102  consistent gets
         15  physical reads
          0  redo size
        413  bytes sent via SQL*Net to client
        384  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

1.2 不适合的场景

由于位图索引本身存储特性的限制，在值重复率高的列或需要经常更新的列是不适合建立位图索引的。

1.2.1 列修改容易造成锁表

```
SQL> create table test(id number,name varchar2(30),sex varchar2(10),age number);

Table created.

SQL> create bitmap index idx_test_bitmap on test(sex);

Index created.

SQL> select sid from v$mystat where rownum=1;

       SID
----------
        37

SQL> insert into test values(1,'renqinglei','man',30);

1 row created.

SQL>
```

暂未提交

```
[oracle@std ~]$ sqlplus scott/tiger

SQL*Plus: Release 10.2.0.1.0 - Production on Thu Apr 10 10:51:11 2014

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options

SQL> select sid from v$mystat where rownum=1;

       SID
----------
        31

SQL> insert into test values(2,'beijing','man',20);
```

插入等待

```
SQL> select sid,type,id1,id2,lmode,block from v$lock where sid in (31,37);

       SID TY        ID1        ID2      LMODE      BLOCK
---------- -- ---------- ---------- ---------- ----------
        37 TO       8693          1          3          0
        31 TX    1048611         55          0          0
        37 TM      17155          0          3          0
        31 TM      17155          0          3          0
        37 TX    1048611         55          6          1
        31 TX    1310731         44          6          0

6 rows selected.
```

1.2.2 列值重复度低效率低

```
SQL> create table t as select * from dba_objects;

Table created.

SQL> insert into t select * from t;

9472 rows created.

SQL> /

18944 rows created.

SQL> /

37888 rows created.

SQL> /

75776 rows created.

SQL> /

151552 rows created.

SQL> commit;

Commit complete.
```

```
SQL> create index idx_t on t(object_id);

Index created.

SQL> set autotrace traceonly
SQL> alter table t modify object_id not null;

Table altered.

SQL> select /*+index(t idx_t)*/  count(*) from t;
```

普通索引的开销

```
Execution Plan
----------------------------------------------------------
Plan hash value: 3792760506

--------------------------------------------------------------------------
| Id  | Operation         | Name  | Rows  | Cost (%CPU)| Time     |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |       |     1 |   645   (1)| 00:00:08 |
|   1 |  SORT AGGREGATE   |       |     1 |            |          |
|   2 |   INDEX FULL SCAN| IDX_T |   337K|   645   (1)| 00:00:08 |
--------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
        205  recursive calls
          0  db block gets
        746  consistent gets
        638  physical reads
```

```
SQL> set autot traceonly
SQL> select /* +index(t idx_t_bitmap)*/count(*) from t;


Execution Plan
----------------------------------------------------------
Plan hash value: 435700843

-----------------------------------------------------------------------------------------
| Id  | Operation                    | Name         | Rows  | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |              |     1 |   603   (1)| 00:00:08 |
|   1 |  SORT AGGREGATE              |              |     1 |            |          |
|   2 |   BITMAP CONVERSION COUNT    |              |  1062K|   603   (1)| 00:00:08 |
|   3 |    BITMAP INDEX FAST FULL SCAN| IDX_T_BITMAP |       |            |          |
-----------------------------------------------------------------------------------------

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
       1327  consistent gets
          0  physical reads
      49364  redo size
```

2.简要说说函数索引的原理及好处

2.1 函数索引的原理

函数索引是把通过函数计算过的值存储在索引中，当利用函数进行查询时，利用计算出的函数值查询函数索引，进而查询与其函数索引值对应的 ROWID，直接通过 rowid 取出对应的记录，提高了查询效率。

2.2 使用函数索引的好处

2.2.1 对部分记录建立函数索引

```
SQL> create table t (id int ,status varchar2(2));

Table created.

SQL> create index id_normal on t(status);

Index created.

SQL> insert into t select rownum ,'Y' from dual connect by rownum<=1000000;

1000000 rows created.

SQL> commit;

Commit complete.

SQL> analyze table t compute statistics for table for all indexes for all indexed columns;

Table analyzed.
```

```
SQL> select * from t where status='N';

no rows selected
```

普通索引的开销

```
Execution Plan
----------------------------------------------------------
Plan hash value: 2252729315

-------------------------------------------------------------------------------------------
| Id  | Operation                   | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |           |  6667 | 66670 |    29   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| T         |  6667 | 66670 |    29   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN          | ID_NORMAL |  6667 |       |    18   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("STATUS"='N')


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
          3  consistent gets
          0  physical reads
          0  redo size
        327  bytes sent via SQL*Net to client
        373  bytes received via SQL*Net from client
```

```
SQL> set autotrace off
SQL> analyze index id_normal validate structure;

Index analyzed.
```

普通索引的结构

```
SQL> select name,btree_space,lf_rows,height from index_stats;

NAME                           BTREE_SPACE    LF_ROWS     HEIGHT
------------------------------ ----------- ---------- ----------
ID_NORMAL                         19230668    1000000          3
```

```
SQL> drop index id_normal;

Index dropped.

SQL> create index id_status on  t (Case when status= 'N' then 'N' end);

Index created.

SQL> analyze table t compute statistics for table for all indexes for all indexed columns;

Table analyzed.
```

```
SQL> select * from t where (case when status='N' then 'N' end)='N';

no rows selected

Execution Plan
----------------------------------------------------------
Plan hash value: 1835552001

-------------------------------------------------------------------------------------------
| Id  | Operation                    | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |           |     1 |    10 |     1   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| T         |     1 |    10 |     1   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN           | ID_STATUS |     1 |       |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access(CASE "STATUS" WHEN 'N' THEN 'N' END ='N')


Statistics
----------------------------------------------------------
          8  recursive calls
          0  db block gets
          3  consistent gets
```

位图索引的效率

```
SQL> analyze index id_status validate structure;

Index analyzed.

SQL> select name,btree_space,lf_rows,height from index_stats;

NAME                           BTREE_SPACE    LF_ROWS     HEIGHT
------------------------------ ------------ ---------- ----------
ID_STATUS                             7996          0          1
```

位图索引的结构

2.2.1 减少递归调用

```
SQL> create table t1 (first_name varchar2(200),last_name varchar2(200),id number);

Table created.

SQL> create table t2 as select * from dba_objects where rownum<=1000;

Table created.

SQL> insert into t1 (first_name,last_name,id) select object_name,object_type,rownum
  2  from dba_objects where rownum<=1000;

1000 rows created.

SQL> commit;

Commit complete.

SQL> create or replace function get_obj_name(p_id t2.object_id%type)
  2  return t2.object_name%type DETERMINISTIC is
  3  v_name t2.object_name%type;
  4  begin
  5  select object_name into v_name from t2
  6  where object_id=p_id;
  7  return v_name;
  8  end;
  9  /

Function created.
```

```
SQL> select *   from t1 where get_obj_name(id)='TEST';

no rows selected

                    没走函数索引递归调用

Statistics
----------------------------------------------------------
       1046  recursive calls
          0  db block gets
      15007  consistent gets
          0  physical reads
          0  redo size
        390  bytes sent via SQL*Net to client
        373  bytes received via SQL*Net from client
          1  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          0  rows processed
```

```
SQL> create index idx_func_id on t1(get_obj_name(id));

Index created.

SQL> select * from t1 where get_obj_name(id)='TEST';

no rows selected
```

创建函数索引后的递归调用

```
Statistics
----------------------------------------------------------
         67  recursive calls
          0  db block gets
         16  consistent gets
          1  physical reads
          0  redo size
        390  bytes sent via SQL*Net to client
        373  bytes received via SQL*Net from client
          1  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          0  rows processed
```

3.说说使用反向索引、全文索引的应用场景

3.1 反向索引的应用场景

在对一个列建立索引的时候，如果这列的值是数值型的且为连续递增型的数据，这种情况下就适合建立反向索引，因为建立普通索引将回导致连续的数据在索引块中过度集中，这样虽然会带来查询上的好处但是也会造成大量的块争用，且当有多个会话修改相连的数据时，容易造成阻塞，影响系统效率。这样情况下建立反向索引后，根据反向索引的特性，在插入索引块之前，系统会把索引列值倒过来插入，从而避免连续的数据插入相同的数据块中，如 123、124、125、126 倒过来以后就会变为 321、421、521、621。反向索引的一大缺点是无法做范围查询，试想倒过来的值已经无法判断他们之间真是的范围了。

3.2 全文索引的应用场景

全文索引适合于在一些大字段类型中查找匹配关键字，例如搜索引擎（谷歌、百度）常会用到。

3.2.1 全文索引环境准备

```
SQL> select role from dba_roles where role='CTXAPP';

no rows selected

SQL> select username from dba_users where username='CTXSYS';

no rows selected
```

```
SQL> create tablespace idx_ctxsys
  2  datafile '/u02/app/oradata/PSDB/disk1/idx_ctxsys01.dbf' size 200m;

Tablespace created.

SQL> @?/ctx/admin/catctx.sql ctxsys idx_ctxsys TEMPTS1 nolock
```

```
SQL> select username,account_status from dba_users where username='CTXSYS';

USERNAME                       ACCOUNT_STATUS
------------------------------ --------------------------------
CTXSYS                         OPEN

SQL> select role from dba_roles where role='CTXAPP';

ROLE
------------------------------
CTXAPP
```

执行脚本后查询用户和角色已创建

```
SQL> conn ctxsys/ctxsys
Connected.
SQL> @?/ctx/admin/defaults/drdefus.sql
Creating lexer preference...

PL/SQL procedure successfully completed.

Creating wordlist preference...

PL/SQL procedure successfully completed.

Creating stoplist...

PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.

Creating default policy...

PL/SQL procedure successfully completed.

SQL>
```

3.2.2 创建语法分析器并设置属性

```
SQL> conn ctxsys/ctxsys
Connected.
SQL> grant execute on ctx_ddl to scott;

Grant succeeded.

SQL> conn scott/tiger
Connected.
SQL> ctx_ddl.create_preference('club_lexer','CHINESE_LEXER');
SP2-0734: unknown command beginning "ctx_ddl.cr..." - rest of line ignored.
SQL> exec ctx_ddl.create_preference('club_lexer','CHINESE_LEXER');

PL/SQL procedure successfully completed.

SQL> exec ctx_ddl.create_preference('mywordlist', 'BASIC_WORDLIST');

PL/SQL procedure successfully completed.

SQL> exec ctx_ddl.set_attribute('mywordlist','PREFIX_INDEX','TRUE');

PL/SQL procedure successfully completed.

SQL> exec ctx_ddl.set_attribute('mywordlist','PREFIX_MIN_LENGTH',1);

PL/SQL procedure successfully completed.

SQL> exec ctx_ddl.set_attribute('mywordlist','PREFIX_MAX_LENGTH', 5);

PL/SQL procedure successfully completed.

SQL> exec ctx_ddl.set_attribute('mywordlist','SUBSTRING_INDEX', 'YES');

PL/SQL procedure successfully completed.
```

3.2.3 创建表及为表创建全文索引

```
SQL> create table test as select * from dba_objects;

Table created.

SQL> update test set object_name='高兴' where rownum<=2;

2 rows updated.

SQL> commit;

Commit complete.

SQL> create index  id_cont_test on TEST (object_name) indextype is ctxsys.context
  2  parameters ('DATASTORE CTXSYS.DIRECT_DATASTORE FILTER
  3  CTXSYS.NULL_FILTER LEXER club_lexer WORDLIST mywordlist');

Index created.

SQL> exec ctx_ddl.sync_index('id_cont_TEST', '20M');

PL/SQL procedure successfully completed.
```

3.2.4 比较全文索引的性能