

oracle 中默认的索引类型是 B 树索引。还有位图索引，反向键索引，hash 索引，基于函数的索引。  
本篇主要介绍 B 树索引，通过转储分析。对于索引的扫描类型，索引的基本操作不做详细的介绍。

系统信息：

```
[oracle@localhost ~]$ cat /etc/issue
Enterprise Linux Enterprise Linux Server release 5.5 (Carthage)
Kernel \r on an \m
```

数据库版本：

```
SQL> select * from v$version where rownum =1 ;
```

BANNER

```
-----
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production
```

```
SQL> show user;
```

USER 为 "HR"

```
SQL> desc tt;
```

名称	是否为空? 类型
ID	NUMBER
NAME	VARCHAR2(10)

```
SQL> select count(rownum) from tt;
```

```
COUNT (ROWNUM)
```

```
-----
3670016
```

基于 ID 创建索引 index\_t

```
SQL> create index index_t on tt(id) tablespace users;
```

索引已创建。

```
SQL> select object_id from dba_objects where object_name=
```

```
2 'INDEX_T' ;
```

数据库中 segment 有数据段，索引段，undo 段，它们和表名，索引名不是同一概念，但是名字是相同的。

```
OBJECT_ID
```

76332

转储索引:

SQL> alter session set events 'immediate trace name treedump level 76332';

会话已更改。

----- begin tree dump

branch: 0x10038ab 16791723 (0: nrow: 15, level: 2)

branch: 0x100540b 16798731 (-1: nrow: 503, level: 1)

leaf: 0x10038ac 16791724 (-1: nrow: 512 rrow: 512)

leaf: 0x10038ad 16791725 (0: nrow: 512 rrow: 512)

leaf: 0x10038ae 16791726 (1: nrow: 512 rrow: 512)

leaf: 0x10038af 16791727 (2: nrow: 512 rrow: 512)

leaf: 0x10038b0 16791728 (3: nrow: 512 rrow: 512)

leaf: 0x10038b1 16791729 (4: nrow: 512 rrow: 512)

leaf: 0x10038b2 16791730 (5: nrow: 512 rrow: 512)

leaf: 0x10038b3 16791731 (6: nrow: 512 rrow: 512)

leaf: 0x10038b4 16791732 (7: nrow: 512 rrow: 512)

leaf: 0x10038b5 16791733 (8: nrow: 512 rrow: 512)

leaf: 0x10038b6 16791734 (9: nrow: 512 rrow: 512)

leaf: 0x10038b7 16791735 (10: nrow: 512 rrow: 512)

leaf: 0x10038b9 16791737 (11: nrow: 512 rrow: 512)

leaf: 0x10038ba 16791738 (12: nrow: 512 rrow: 512)

leaf: 0x10038bb 16791739 (13: nrow: 512 rrow: 512)

leaf: 0x10038bc 16791740 (14: nrow: 512 rrow: 512)

leaf: 0x10038bd 16791741 (15: nrow: 512 rrow: 512)

leaf: 0x10038be 16791742 (16: nrow: 512 rrow: 512)

leaf: 0x10038bf 16791743 (17: nrow: 512 rrow: 512)

leaf: 0x10038c0 16791744 (18: nrow: 512 rrow: 512)

leaf: 0x10038c1 16791745 (19: nrow: 512 rrow: 512)

leaf: 0x10038c2 16791746 (20: nrow: 512 rrow: 512)

leaf: 0x10038c3 16791747 (21: nrow: 512 rrow: 512)

leaf: 0x10038c4 16791748 (22: nrow: 512 rrow: 512)

leaf: 0x10038c5 16791749 (23: nrow: 512 rrow: 512)

```

leaf: 0x10038c6 16791750 (24: nrow: 512 rrow: 512)
leaf: 0x10038c7 16791751 (25: nrow: 512 rrow: 512)
leaf: 0x10038c9 16791753 (26: nrow: 512 rrow: 512)
leaf: 0x10038ca 16791754 (27: nrow: 512 rrow: 512)
leaf: 0x10038cb 16791755 (28: nrow: 512 rrow: 512)
leaf: 0x10038cc 16791756 (29: nrow: 512 rrow: 512)
leaf: 0x10038cd 16791757 (30: nrow: 512 rrow: 512)
leaf: 0x10038ce 16791758 (31: nrow: 512 rrow: 512)
leaf: 0x10038cf 16791759 (32: nrow: 512 rrow: 512)
leaf: 0x10038d0 16791760 (33: nrow: 512 rrow: 512)
leaf: 0x10038d1 16791761 (34: nrow: 512 rrow: 512)
leaf: 0x10038d2 16791762 (35: nrow: 512 rrow: 512)
leaf: 0x10038d3 16791763 (36: nrow: 512 rrow: 512)
leaf: 0x10038d4 16791764 (37: nrow: 512 rrow: 512)
leaf: 0x10038d5 16791765 (38: nrow: 512 rrow: 512)
leaf: 0x10038d6 16791766 (39: nrow: 512 rrow: 512)
leaf: 0x10038d7 16791767 (40: nrow: 512 rrow: 512)
leaf: 0x10038d9 16791769 (41: nrow: 512 rrow: 512)
leaf: 0x10038da 16791770 (42: nrow: 512 rrow: 512)
leaf: 0x10038db 16791771 (43: nrow: 512 rrow: 512)
leaf: 0x10038dc 16791772 (44: nrow: 512 rrow: 512)
leaf: 0x10038dd 16791773 (45: nrow: 512 rrow: 512)
leaf: 0x10038de 16791774 (46: nrow: 512 rrow: 512)
.....
....
....
....
----- end tree dump

```

这是一棵平衡树，因为平衡树的查找效率很高，根节点到所有的叶子节点的高度相同。

branch 表示的是根节点。以上选取了一部分，已经按从左向右拍好序了。

leaf: 0x10038ac 16791724 (-1: nrow: 512 rrow: 512) 我们选取这一列：

把十六进制，和十进制数相互转换：

```
SQL> select to_number('10038ac','xxxxxxxxxxxxxxxx') from dual;
```

```
TO_NUMBER('10038AC','XXXXXXXXXXXXXXXXX')
```

```
-----
```

16791724

```
SQL> select to_char('16791724','XXXXXXXXXXXXXXXXX') from dual;
```

```
TO_CHAR('16791724','XXXXXXXXXXXXXXXXX')
```

```
-----
```

10038ac

我们利用 oracle 中提供的一个包可以求得索引所在的文件号，块号：

```
SQL> select dbms_utility.data_block_address_file(16791724) from dual;
```

```
DBMS_UTILITY.DATA_BLOCK_ADDRESS_FILE(16791724)
```

```
-----
```

4

```
SQL> select dbms_utility.data_block_address_block(16791724) from dual;
```

```
DBMS_UTILITY.DATA_BLOCK_ADDRESS_BLOCK(16791724)
```

```
-----
```

14508

通过查视图 dba\_extends, 索引存储在数据文件 4, 块 14508 在起始块范围内。

SQL 窗口 - select \* from dba\_extends where segment\_name='INDEX\_T';

SQL | 输出 | 统计表

select \* from dba\_extends where segment\_name='INDEX\_T';

	OWNER	SEGMENT_NAME	PARTITION_NAME	SEGMENT_TYPE	TABLESPACE_NAME	EXTENT_ID	FILE_ID	BLOCK_ID	BYTES	BLOCKS	RELATIVE_FNO
1	HR	INDEX_T		INDEX	USERS	0	4	14504	65536	8	
2	HR	INDEX_T		INDEX	USERS	1	4	14512	65536	8	
3	HR	INDEX_T		INDEX	USERS	2	4	14520	65536	8	
4	HR	INDEX_T		INDEX	USERS	3	4	14528	65536	8	
5	HR	INDEX_T		INDEX	USERS	4	4	14536	65536	8	
6	HR	INDEX_T		INDEX	USERS	5	4	14544	65536	8	
7	HR	INDEX_T		INDEX	USERS	6	4	14552	65536	8	
8	HR	INDEX_T		INDEX	USERS	7	4	14560	65536	8	
9	HR	INDEX_T		INDEX	USERS	8	4	14568	65536	8	
10	HR	INDEX_T		INDEX	USERS	9	4	14576	65536	8	

1:56 0:01 sys@55 AS SYSDBA 10 行被选择, 耗时 1.03 秒 (更多...)

此时我们 dump 数据文件 4, 块 14508:

SQL> alter system dump datafile 4 block 14508;

系统已更改。

```
row#0[8020] flag: -----, lock: 0, len=12
col 0; len 2; (2):  c1 02
col 1; len 6; (6):  01 00 1b ab 00 01
row#1[8008] flag: -----, lock: 0, len=12
col 0; len 2; (2):  c1 02
col 1; len 6; (6):  01 00 1b ab 00 03
row#2[7996] flag: -----, lock: 0, len=12
col 0; len 2; (2):  c1 02
col 1; len 6; (6):  01 00 1b ab 00 08
row#3[7984] flag: -----, lock: 0, len=12
col 0; len 2; (2):  c1 02
col 1; len 6; (6):  01 00 1b ab 00 0a
row#4[7972] flag: -----, lock: 0, len=12
col 0; len 2; (2):  c1 02
col 1; len 6; (6):  01 00 1b ab 00 0f
row#5[7960] flag: -----, lock: 0, len=12
col 0; len 2; (2):  c1 02
col 1; len 6; (6):  01 00 1b ab 00 11
row#6[7948] flag: -----, lock: 0, len=12
```

col 0; len 2; (2): c1 02  
col 1; len 6; (6): 01 00 1b ab 00 16  
row#7[7936] flag: -----, lock: 0, len=12  
col 0; len 2; (2): c1 02  
col 1; len 6; (6): 01 00 1b ab 00 18  
row#8[7924] flag: -----, lock: 0, len=12  
col 0; len 2; (2): c1 02  
col 1; len 6; (6): 01 00 1b ab 00 1d  
row#9[7912] flag: -----, lock: 0, len=12  
col 0; len 2; (2): c1 02  
col 1; len 6; (6): 01 00 1b ab 00 1f  
row#10[7900] flag: -----, lock: 0, len=12  
col 0; len 2; (2): c1 02  
col 1; len 6; (6): 01 00 1b ab 00 24  
row#11[7888] flag: -----, lock: 0, len=12  
col 0; len 2; (2): c1 02  
col 1; len 6; (6): 01 00 1b ab 00 26  
row#12[7876] flag: -----, lock: 0, len=12  
col 0; len 2; (2): c1 02  
col 1; len 6; (6): 01 00 1b ab 00 2b  
row#13[7864] flag: -----, lock: 0, len=12  
col 0; len 2; (2): c1 02  
col 1; len 6; (6): 01 00 1b ab 00 2d  
row#14[7852] flag: -----, lock: 0, len=12  
col 0; len 2; (2): c1 02  
col 1; len 6; (6): 01 00 1b ab 00 32  
row#15[7840] flag: -----, lock: 0, len=12  
col 0; len 2; (2): c1 02  
col 1; len 6; (6): 01 00 1b ab 00 34  
row#16[7828] flag: -----, lock: 0, len=12  
col 0; len 2; (2): c1 02  
col 1; len 6; (6): 01 00 1b ab 00 39  
row#17[7816] flag: -----, lock: 0, len=12  
col 0; len 2; (2): c1 02  
col 1; len 6; (6): 01 00 1b ab 00 3b  
row#18[7804] flag: -----, lock: 0, len=12  
col 0; len 2; (2): c1 02  
col 1; len 6; (6): 01 00 1b ab 00 40  
row#19[7792] flag: -----, lock: 0, len=12  
col 0; len 2; (2): c1 02  
col 1; len 6; (6): 01 00 1b ab 00 42

```
row#20[7780] flag: -----, lock: 0, len=12
.....
.....
```

选取这一行为例：

```
row#0[8020] flag: -----, lock: 0, len=12
col 0; len 2; (2):  c1 02
col 1; len 6; (6):  01 00 1b ab 00 01
col 0 表示第一列，长度为2，c1 02 表示是多少呢？
```

```
SQL> select * from tt where rownum<4 order by id;
```

```

          ID NAME
-----
          1 wO
          2 wang
          6 hong

```

```
SQL> select dump(1,16) from dual;
```

```
DUMP(1,16)
-----
Typ=2 Len=2: c1,2      ==>这里是 c1 02, 0 省略了
```

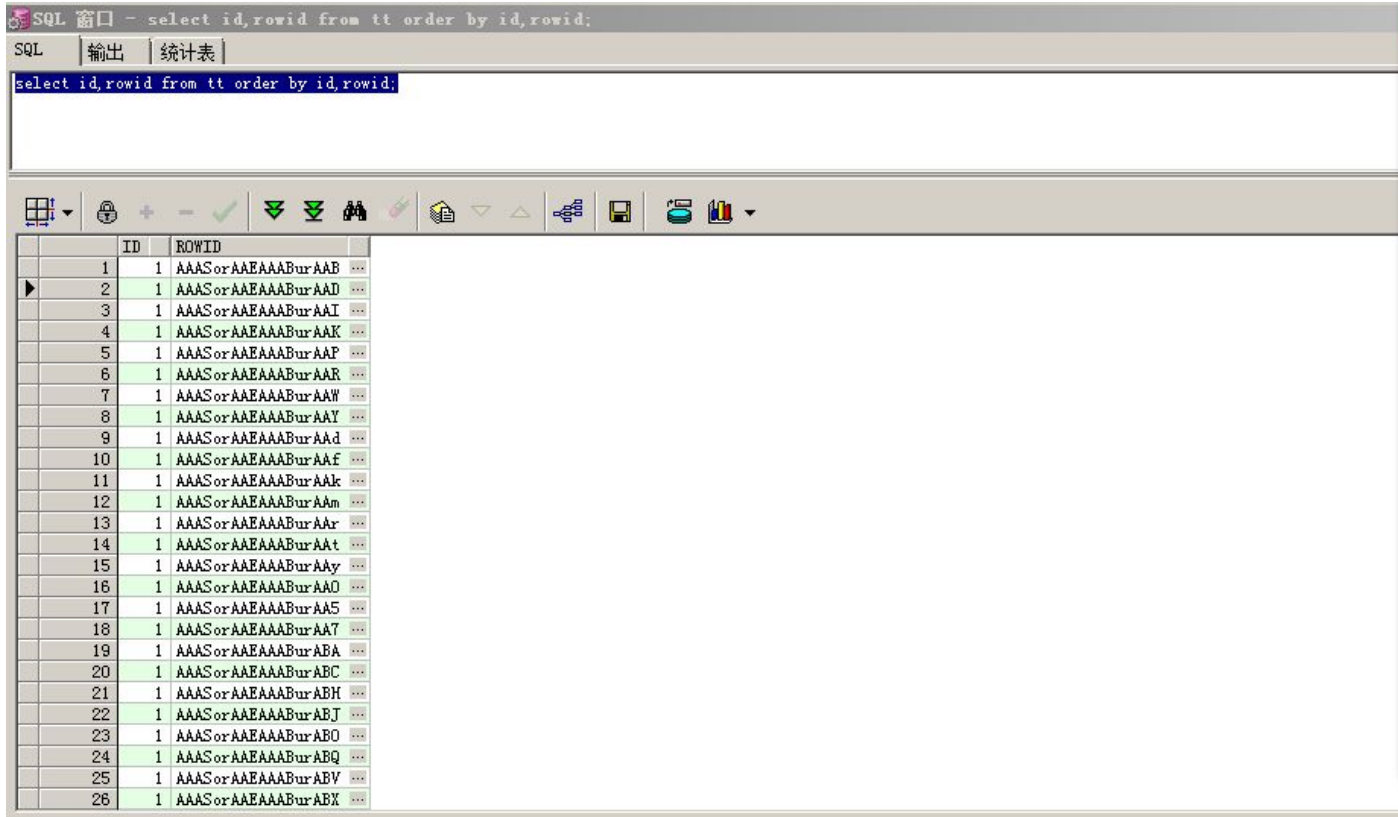
这下清楚了吧，第一行第一列存储的是 1，oracle 存储数据的方法很复杂。

col 1 表示的是第二列，长度是 6， 01 00 1b ab 00 01 就是索引的值，是十六进制数，我们可以转化为二进制数：

```
00000001      00000000      00011011      10101011      00000000      0
00000001
```

```
-----
00000001      00      ==>1x2x2=4      前 10 位表示了数据文件号
000000      00011011      10101011      ==>4096+2048+512+256+128+32+8+2+1=7083
      这里的 22 位表示了块号
00000000      00000001==>1      16 位代表着行
号
```

此时排序并查 rowid:



此时用第一列的 rowid,

通过 oracle 提供的一个包，可以求出对象编号，文件号，块号：

执行如下图:



```

SQL> create or replace procedure get_rowid<rid varchar2>
2   is
3   begin
4   for x in
5   < select DBMS_ROWID.ROWID_OBJECT<rid> objno,
6           DBMS_ROWID.ROWID_RELATIVE_FNO<rid> rfno,
7           DBMS_ROWID.ROWID_BLOCK_NUMBER<rid> blockno,
8           DBMS_ROWID.ROWID_ROW_NUMBER<rid> rownumber
9   From dual
10  >
11
12  loop
13      dbms_output.put_line< 'Object No: ' || x.objno >;
14      dbms_output.put_line< 'R Fno: ' || x.rfno >;
15      dbms_output.put_line< 'Block Number: ' || x.blockno >;
16      dbms_output.put_line< 'Row Number: ' || x.rownumber >;
17  end loop;
18  end;
19  /

```

过程已创建。

```

SQL> set serveroutput on
SQL> exec get_rowid<'AAASorAAEAAABurAAB
ERROR:
ORA-01756: 引号内的字符串没有正确结束

```

```

SQL> exec get_rowid<'AAASorAAEAAABurAAB'>;
Object No: 76331
R Fno: 4
Block Number: 7083
Row Number: 1

```

PL/SQL 过程已成功完成。

上面从索引存储的段以及数据段进行分析。

我们知道索引不一定会提高查询效率，往往乱建索引会严重影响查询效率，系统用不用索引，我们不能干预（但是dba可以手动改变），是oracle CBO选择的结果。

下面我们可以做一个小实验：

```
SQL> select count(rowid) from t;
```

COUNT (ROWID)

-----

4718644

```
SQL> select count(rowid) from t where id=1;
```

COUNT (ROWID)

-----

4718592

SQL> select count(rowid) from t where id=2;

COUNT (ROWID)

-----

26

SQL> select count(rowid) from t where id=3;

COUNT (ROWID)

-----

26

SQL> set autotrace traceonly;

SQL> select \* from tt;

执行计划

-----

Plan hash value: 264906180

-----							
Id		Operation		Name		Rows	
-----							
	0	SELECT STATEMENT				3556K	
	1	TABLE ACCESS FULL		TT		3556K	
-----							
				Bytes		Cost (%CPU)	
				Time			

Note

```
-----  
      - dynamic sampling used for this statement (level=2)
```

此时是全表扫描读取，是多块读取，，这样读取比较快，如果此时用索引，则效率会低。  
SQL> select \* from tt where id=5;

执行计划

```
-----  
Plan hash value: 3103123359
```

-----

-----

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Tim
----	-----------	------	------	-------	-------------	-----

e |

-----

-----

0	SELECT STATEMENT		5	100	4 (0)	00:
---	------------------	--	---	-----	-------	-----

00:01 |

1	TABLE ACCESS BY INDEX ROWID	TT	5	100	4 (0)	00:
---	-----------------------------	----	---	-----	-------	-----

00:01 |

|\* 2 | INDEX RANGE SCAN | INDEX\_T | 5 | | 3 (0) | 00:

00:01 |

-----

-----

Predicate Information (identified by operation id):

-----

2 - access("ID"=5)

Note

-----

- dynamic sampling used for this statement (level=2)

此时是索引读取。  
SQL> select \* from tt where id=1;

执行计划

-----

Plan hash value: 264906180

-----

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time	
0	SELECT STATEMENT		2121K	40M	3644 (2)	00:00:44	
* 1	TABLE ACCESS FULL	TT	2121K	40M	3644 (2)	00:00:44	

---

Predicate Information (identified by operation id):

---

1 - filter("ID"=1)

Note

---

- dynamic sampling used for this statement (level=2)

看到了吧，此时是全表扫描读取，数据库是很聪明的吧！