引言：大家都知道“效率”是数据库中非常重要的一个指标，如何提高效率大家可能都会想起索引，但索引又这么多种，什么场合应该使用什么索引呢？哪种索引可以提高我们的效率，哪种索引可以让我们的效率大大降低（有时还不如全表扫描性能好）下面要讲的“索引”如何成为我们的利器而不是灾难！多说一点，由于不同索引的存储结构不同，所以应用在不同组织结构的数据上，本篇文章重点就是：理解不同的技术都适合在什么地方应用！  
  
B-Tree索引  
场合：非常适合数据重复度低的字段 例如 身份证号码  手机号码  QQ号等字段，常用于主键 唯一约束，一般在在线交易的项目中用到的多些。  
原理：一个键值对应一行（rowid）  格式： 【索引头|键值|rowid】  
优点：当没有索引的时候，[oracle](http://www.itpub.net/tree/index_1/)只能全表扫描where qq=40354446 这个条件那么这样是灰常灰常耗时的，当数据量很大的时候简直会让人崩溃，那么有个B-tree索引我们就像翻书目录一样，直接定位rowid立刻就找到了我们想要的数据，实质减少了I/O操作就提高速度，它有一个显著特点查询性能与表中数据量无关，例如 查2万行的数据用了3 consistent get,当查询1200万行的数据时才用了4 consistent gets。  
当我们的字段中使用了主键or唯一约束时，不用想直接可以用B-tree索引  
缺点：不适合键值重复率较高的字段上使用，例如 第一章 1-500page 第二章 501-1000page  
实验：  
alter system flush shared\_pool;   清空共享池  
alter system flush buffer\_cache;  清空数据库缓冲区，都是为了实验需要  
创建leo\_t1  leo\_t2 表  
leo\_t1 表的object\_id列的数据是没有重复值的，我们抽取了10行数据就可以看出来了。  
[LS@LEO](mailto:LS@LEO" \t "_blank)> create table leo\_t1 as select object\_id,object\_name from dba\_objects;  
[LS@LEO](mailto:LS@LEO)> select count(\*) from leo\_t1;  
  COUNT(\*)  
----------  
      9872  
[LS@LEO](mailto:LS@LEO)>  select \* from leo\_t1 where rownum <= 10;  
OBJECT\_ID OBJECT\_NAME  
---------- -----------  
        20 ICOL$  
        44 I\_USER1  
        28 CON$  
        15 UNDO$  
        29 C\_COBJ#  
         3 I\_OBJ#  
        25 PROXY\_ROLE\_DATA$  
        39 I\_IND1  
        51 I\_CDEF2  
        26 I\_PROXY\_ROLE\_DATA$\_1  
leo\_t2 表的object\_id列我们是做了取余操作，值就只有0,1两种，因此重复率较高，如此设置为了说明重复率对B树索引的影响  
[LS@LEO](mailto:LS@LEO)> create table leo\_t2 as select mod(object\_id,2) object\_ID ,object\_name from dba\_objects;  
[LS@LEO](mailto:LS@LEO)> select count(\*) from leo\_t2;  
  COUNT(\*)  
----------  
      9873  
[LS@LEO](mailto:LS@LEO)> select \* from leo\_t2 where rownum <= 10;  
OBJECT\_ID OBJECT\_NAME  
---------- -----------  
         0 ICOL$  
         0 I\_USER1  
         0 CON$  
         1 UNDO$  
         1 C\_COBJ#  
         1 I\_OBJ#  
         1 PROXY\_ROLE\_DATA$  
         1 I\_IND1  
         1 I\_CDEF2  
         0 I\_PROXY\_ROLE\_DATA$\_1  
[LS@LEO](mailto:LS@LEO)> create index leo\_t1\_index on leo\_t1(object\_id);   创建B-tree索引，说明 默认创建的都是B-tree索引  
Index created.  
[LS@LEO](mailto:LS@LEO)> create index leo\_t2\_index on leo\_t2(object\_ID);   创建B-tree索引  
Index created.  
让我们看一下leo\_t1与leo\_t2的重复情况  
[LS@LEO](mailto:LS@LEO)> select count(distinct(object\_id)) from leo\_t1;    让我们看一下leo\_t1与leo\_t2的重复情况，leo\_t1没有重复值，leo\_t2有很多  
COUNT(DISTINCT(OBJECT\_ID))  
--------------------------  
                      9872  
[LS@LEO](mailto:LS@LEO)> select count(distinct(object\_ID)) from leo\_t2;  
COUNT(DISTINCT(OBJECT\_ID))  
--------------------------  
                         2  
收集2个表统计信息   
[LS@LEO](mailto:LS@LEO)> execute dbms\_stats.gather\_table\_stats(ownname=>'LS',tabname=>'LEO\_T1',method\_opt=>'for all indexed columns size 2',cascade=>TRUE);  
[LS@LEO](mailto:LS@LEO)> execute dbms\_stats.gather\_table\_stats(ownname=>'LS',tabname=>'LEO\_T2',method\_opt=>'for all indexed columns size 2',cascade=>TRUE);  
参数详解：  
method\_opt=>'for all indexed columns size 2'  size\_clause=integer 整型 ，范围 1~254 ，使用柱状图[ histogram analyze ]分析列数据的分布情况  
cascade=>TRUE                          收集表的统计信息的同时收集B-tree索引的统计信息  
显示执行计划和统计信息+设置autotrace简介  
序号  命令                          解释  
1    SET AUTOTRACE OFF             此为默认值，即关闭Autotrace   
2    SET AUTOTRACE ON EXPLAIN      只显示执行计划  
3    SET AUTOTRACE ON STATISTICS   只显示执行的统计信息  
4    SET AUTOTRACE ON              包含2,3两项内容  
5    SET AUTOTRACE TRACEONLY       与ON相似，但不显示语句的执行结果  
结果键值少的情况  
set autotrace trace exp stat;  （SET AUTOTRACE OFF 关闭执行计划和统计信息）  
[LS@LEO](mailto:LS@LEO)> select \* from leo\_t1 where object\_id=1;  
no rows selected  
Execution Plan 执行计划  
----------------------------------------------------------  
Plan hash value: 3712193284  
--------------------------------------------------------------------------------------------  
| Id  | Operation                   | Name         | Rows  | Bytes | Cost (%CPU)| Time     |  
--------------------------------------------------------------------------------------------  
|   0 | SELECT STATEMENT            |              |     1 |    21 |     2   (0)| 00:00:01 |  
|   1 |  TABLE ACCESS BY INDEX ROWID| LEO\_T1       |     1 |    21 |     2   (0)| 00:00:01 |  
|\*  2 |   INDEX RANGE SCAN索引扫描  | LEO\_T1\_INDEX |     1 |       |     1   (0)| 00:00:01 |  
--------------------------------------------------------------------------------------------  
Predicate Information (identified by operation id):  
---------------------------------------------------  
   2 - access("OBJECT\_ID"=1)  
Statistics  统计信息  
----------------------------------------------------------  
          0  recursive calls  
          0  db block gets  
          2  consistent gets  我们知道leo\_t1表的object\_id没有重复值，因此使用B-tree索引扫描只有2次一致性读  
          0  [PHYSICAL READS[http://cdncache-a.akamaihd.net/items/it/img/arrow-10x10.png](http://www.itpub.net/)](http://www.itpub.net/)  
          0  redo size  
        339  bytes sent via SQL\*Net to client  
        370  bytes received via SQL\*Net from client  
          1  SQL\*Net roundtrips to/from client  
          0  sorts (memory)  
          0  sorts (disk)  
          0  rows processed  
结果键值多的情况  
[LS@LEO](mailto:LS@LEO)> select \* from leo\_t2 where object\_ID=1; （select /\*+full(leo\_t2) \*/ \*  from leo\_t2 where object\_ID=1;hint方式强制全表扫描）  
4943 rows selected.      
Execution Plan  执行计划  
----------------------------------------------------------  
Plan hash value: 3657048469  
----------------------------------------------------------------------------  
| Id  | Operation         | Name   | Rows  | Bytes | Cost (%CPU)| Time     |  
----------------------------------------------------------------------------  
|   0 | SELECT STATEMENT  |        |  4943 | 98860 |    12   (0)| 00:00:01 |  
|\*  1 |  TABLE ACCESS FULL| LEO\_T2 |  4943 | 98860 |    12   (0)| 00:00:01 | sql结果是4943row，那么全表扫描也是4943row  
----------------------------------------------------------------------------  
Predicate Information (identified by operation id):  
---------------------------------------------------  
   1 - filter("OBJECT\_ID"=1)   
Statistics  统计信息  
----------------------------------------------------------  
          1  recursive calls  
          0  db block gets  
        366  consistent gets   导致有366次一致性读  
          0  [PHYSICAL READS[http://cdncache-a.akamaihd.net/items/it/img/arrow-10x10.png](http://www.itpub.net/)](http://www.itpub.net/)  
          0  redo size  
     154465  bytes sent via SQL\*Net to client  
       4000  bytes received via SQL\*Net from client  
        331  SQL\*Net roundtrips to/from client  
          0  sorts (memory)  
          0  sorts (disk)  
       4943  rows processed  
大家肯定会疑惑，为什么要用全表扫描而不用B-tree索引呢，这是因为oracle基于成本优化器CBO认为使用全表扫描要比使用B-tree索引性能更好更快，由于我们结果重复率很高，导致有366次一致性读，从cup使用率12%上看也说明了B-tree索引不适合键值重复率较高的列  
我们在看一下强制使用B-tree索引时，效率是不是没有全表扫描高呢？  
[LS@LEO](mailto:LS@LEO)> select /\*+index(leo\_t2 leo\_t2\_index) \*/ \* from leo\_t2 where object\_ID=1; hint方式强制索引扫描  
4943 rows selected.  
Execution Plan  执行计划  
----------------------------------------------------------  
Plan hash value: 321706586  
--------------------------------------------------------------------------------------------  
| Id  | Operation                   | Name         | Rows  | Bytes | Cost (%CPU)| Time     |  
--------------------------------------------------------------------------------------------  
|   0 | SELECT STATEMENT            |              |  4943 | 98860 |    46   (0)| 00:00:01 |  
|   1 |  TABLE ACCESS BY INDEX ROWID| LEO\_T2       |  4943 | 98860 |    46   (0)| 00:00:01 |  
|\*  2 |   INDEX RANGE [SCAN[http://cdncache-a.akamaihd.net/items/it/img/arrow-10x10.png](http://www.itpub.net/)](http://www.itpub.net/)          | LEO\_T2\_INDEX |  4943 |       |    10   (0)| 00:00:01 |  
--------------------------------------------------------------------------------------------  
Predicate Information (identified by operation id):  
---------------------------------------------------  
   2 - access("OBJECT\_ID"=1)  
Statistics  统计信息  
----------------------------------------------------------  
          1  recursive calls  
          0  db block gets  
        704  consistent gets  使用B-tree索引704次一致性读 > 全表扫描366次一致性读，而且cpu使用率也非常高，显然效果没有全表扫描高  
          0  [PHYSICAL READS[http://cdncache-a.akamaihd.net/items/it/img/arrow-10x10.png](http://www.itpub.net/)](http://www.itpub.net/)  
          0  redo size  
     171858  bytes sent via SQL\*Net to client  
       4000  bytes received via SQL\*Net from client  
        331  SQL\*Net roundtrips to/from client  
          0  sorts (memory)  
          0  sorts (disk)  
       4943  rows processed  
小结：从以上的测试我们可以了解到，B-tree索引在什么情况下使用跟键值重复率高低有很大关系的，之间没有一个明确的分水岭，只能多测试分析执行计划后来决定。  
  
位图索引   Bitmap index  
场合：列的基数很少，可枚举，重复值很多，数据不会被经常更新  
原理：一个键值对应很多行（rowid）， 格式：键值  start\_rowid   end\_rowid  位图  
优点：OLAP 例如报表类数据库 重复率高的数据 特定类型的查询例如count、or、and等逻辑操作因为只需要进行位运算即可得到我们需要的结果  
缺点：不适合重复率低的字段，还有经常DML操作（insert，update，delete），因为位图索引的锁代价极高，修改一个位图索引段影响整个位图段，例如修改  
一个键值，会影响同键值的多行，所以对于OLTP 系统位图索引基本上是不适用的  
实验：位图索引和B-tree索引的性能比较  
set pagesize 100;   设置页大小  
利用dba\_objects数据字典创建一个15万行的表  
[LS@LEO](mailto:LS@LEO)> create table leo\_bm\_t1 as select \* from dba\_objects;  
Table created.  
[LS@LEO](mailto:LS@LEO" \t "_blank)> insert into leo\_bm\_t1 select \* from leo\_bm\_t1;  翻倍插入  
9876 rows created.  
[LS@LEO](mailto:LS@LEO)> /  
19752 rows created.  
[LS@LEO](mailto:LS@LEO)> /  
39504 rows created.  
[LS@LEO](mailto:LS@LEO)> /  
79008 rows created.  
[LS@LEO](mailto:LS@LEO)> /  
158016 rows created.  
因object\_type字段重复值较高，顾在此字段上创建bitmap索引  
[LS@LEO](mailto:LS@LEO)> create bitmap index leo\_bm\_t1\_index on leo\_bm\_t1(object\_type);  
Index created.  
创建一个和leo\_bm\_t1表结构一模一样的表leo\_bm\_t2，并在object\_type列上创建一个B-tree索引（15万行记录）  
[LS@LEO](mailto:LS@LEO)> create table leo\_bm\_t2 as select \* from leo\_bm\_t1;  
Table created.  
[LS@LEO](mailto:LS@LEO" \t "_blank)> create index leo\_bm\_t2\_bt\_index on leo\_bm\_t2(object\_type);  
Index created.  
对比位图索引和B-tree索引所占空间大小,很明显位图要远远小于B-tree索引所占用的空间，节约空间特性也是我们选择位图的理由之一  
[LS@LEO](mailto:LS@LEO)> select segment\_name,bytes from user\_segments where segment\_type='INDEX';  
SEGMENT\_NAME                                                                           BYTES  
--------------------------------------------------------------------------------- ----------  
LEO\_BM\_T1\_INDEX                                                                       327680(327K)  
LEO\_BM\_T2\_BT\_INDEX                                                                   7340032(7M)  
显示执行计划和统计信息  
set autotrace trace exp stat;  
在创建有位图索引的表上做count操作对比执行计划  
[LS@LEO](mailto:LS@LEO)> select count(\*) from leo\_bm\_t1 where object\_type='TABLE';  
Execution Plan  执行计划  
----------------------------------------------------------  
Plan hash value: 3251686305  
-----------------------------------------------------------------------------------------------  
| Id  | Operation                   | Name            | Rows  | Bytes | Cost (%CPU)| Time     |  
-----------------------------------------------------------------------------------------------  
|   0 | SELECT STATEMENT            |                 |     1 |    11 |     4   (0)| 00:00:01 |  
|   1 |  SORT AGGREGATE             |                 |     1 |    11 |            |          |  
|   2 |   BITMAP CONVERSION COUNT   |                 | 36315 |   390K|     4   (0)| 00:00:01 |  
|\*  3 |    BITMAP INDEX SINGLE VALUE| LEO\_BM\_T1\_INDEX |       |       |            |          |  
-----------------------------------------------------------------------------------------------  
位图索引上只扫描了一个值  
Predicate Information (identified by operation id):  
---------------------------------------------------  
   3 - access("OBJECT\_TYPE"='TABLE')  
Note  
-----  
   - dynamic [SAMPLING[http://cdncache-a.akamaihd.net/items/it/img/arrow-10x10.png](http://www.itpub.net/)](http://www.itpub.net/) used for this statement 动态采样  
Statistics   统计信息  
----------------------------------------------------------  
          9  recursive calls  
          0  db block gets  
         93  consistent gets  oracle选择使用位图索引访问数据，导致93次一致性读  
          7  physical reads  
          0  redo size  
        413  bytes sent via SQL\*Net to client  
        381  bytes received via SQL\*Net from client  
          2  SQL\*Net roundtrips to/from client  
          0  sorts (memory)  
          0  sorts (disk)  
          1  rows processed  
在创建有B-tree索引的表上做count操作对比执行计划  
[LS@LEO](mailto:LS@LEO)> select count(\*) from leo\_bm\_t2 where object\_type='TABLE';  
Execution Plan   执行计划  
----------------------------------------------------------  
Plan hash value: 613433245  
----------------------------------------------------------------------------------------  
| Id  | Operation         | Name               | Rows  | Bytes | Cost (%CPU)| Time     |  
----------------------------------------------------------------------------------------  
|   0 | SELECT STATEMENT  |                    |     1 |    11 |    59   (0)| 00:00:01 |  
|   1 |  SORT AGGREGATE   |                    |     1 |    11 |            |          |  
|\*  2 |   INDEX RANGE SCAN| LEO\_BM\_T2\_BT\_INDEX | 25040 |   268K|    59   (0)| 00:00:01 |  
----------------------------------------------------------------------------------------  
B-tree索引上全部扫描，cpu使用率达到了59%，比位图索引cpu使用率4%高出许多  
Predicate Information (identified by operation id):  
---------------------------------------------------  
   2 - access("OBJECT\_TYPE"='TABLE')  
Note  
-----  
   - dynamic sampling used for this statement 动态采样  
  
Statistics   统计信息  
----------------------------------------------------------  
         32  recursive calls  
          0  db block gets  
        161  consistent gets  B-tree索引表上发生了161次一致性读要远远高于位图索引表上93次一致性读，因此还是位图索引效率高  
         74  physical reads  
          0  redo size  
        413  bytes sent via SQL\*Net to client  
        381  bytes received via SQL\*Net from client  
          2  SQL\*Net roundtrips to/from client  
          0  sorts (memory)  
          0  sorts (disk)  
          1  rows processed  
我们再看看等值查找where object\_type='TABLE'情况下位图索引和B-tree索引的性能对比  
[LS@LEO](mailto:LS@LEO)> select \* from leo\_bm\_t1 where object\_type='TABLE' ;  
28512 rows selected.  
Execution Plan  执行计划  
----------------------------------------------------------  
Plan hash value: 4228542614  
------------------------------------------------------------------------------------------------  
| Id  | Operation                    | Name            | Rows  | Bytes | Cost (%CPU)| Time     |  
------------------------------------------------------------------------------------------------  
|   0 | SELECT STATEMENT             |                 | 36315 |  6277K|   562   (0)| 00:00:07 |  
|   1 |  TABLE ACCESS BY INDEX ROWID | LEO\_BM\_T1       | 36315 |  6277K|   562   (0)| 00:00:07 |  
|   2 |   BITMAP CONVERSION TO ROWIDS| 位图映像->rowids |       |       |            |          |  
|\*  3 |    BITMAP INDEX SINGLE VALUE | LEO\_BM\_T1\_INDEX |       |       |            |          |  
------------------------------------------------------------------------------------------------  
Predicate Information (identified by operation id):  
---------------------------------------------------  
   3 - access("OBJECT\_TYPE"='TABLE')  
Note  
-----  
   - dynamic sampling used for this statement动态采样  
Statistics   统计信息  
----------------------------------------------------------  
          7  recursive calls  
          0  db block gets  
       4407  consistent gets  使用位图索引发生了4407次一致性读  
          0  physical reads  
          0  redo size  
    2776927  bytes sent via SQL\*Net to client  
      21281  bytes received via SQL\*Net from client  
       1902  SQL\*Net roundtrips to/from client  
          0  sorts (memory)  
          0  sorts (disk)  
      28512  rows processed  
leo\_bm\_t2表上使用B-tree索引得到执行计划  
[LS@LEO](mailto:LS@LEO)> select /\*+index(leo\_bm\_t2 leo\_bm\_t2\_bt\_index) \*/ \* from leo\_bm\_t2 where object\_type='TABLE' ;  
28512 rows selected.  我们强制使用B-tree索引扫描等值条件  
Execution Plan  执行计划  
----------------------------------------------------------  
Plan hash value: 1334503202  
--------------------------------------------------------------------------------------------------  
| Id  | Operation                   | Name               | Rows  | Bytes | Cost (%CPU)| Time     |  
--------------------------------------------------------------------------------------------------  
|   0 | SELECT STATEMENT            |                    | 25040 |  4328K|  2063   (1)| 00:00:25 |  
|   1 |  TABLE ACCESS BY INDEX ROWID| LEO\_BM\_T2          | 25040 |  4328K|  2063   (1)| 00:00:25 |  
|\*  2 |   INDEX RANGE SCAN          | LEO\_BM\_T2\_BT\_INDEX | 25040 |       |    59   (0)| 00:00:01 |  
--------------------------------------------------------------------------------------------------  
B-tree索引上全部扫描，cpu使用率达到了2063%，比位图索引cpu使用率562%高出许多  
Predicate Information (identified by operation id):  
---------------------------------------------------  
   2 - access("OBJECT\_TYPE"='TABLE')  
Note  
-----  
   - dynamic sampling used for this statement  
  
Statistics  
----------------------------------------------------------  
          7  recursive calls  
          0  db block gets  
       6621  consistent gets    
          0  physical reads  
          0  redo size  
    2776927  bytes sent via SQL\*Net to client  
      21281  bytes received via SQL\*Net from client  
       1902  SQL\*Net roundtrips to/from client  
          0  sorts (memory)  
          0  sorts (disk)  
      28512  rows processed  
小结：在等值查找中我们可以看出位图索引的效率依言高于B-tree索引  
  
全文索引 Text index  
定义：全文索引就是通过将文字按照某种语言进行词汇拆分，重新将数据组合存储，来达到快速检索的目的  
场合：当字段里存储的都是文本时适合用全文索引，常用于搜索文字  
优点：全文索引不是按照键值存储的，而是按照分词重组数据，常用于模糊查询Where name like '%leonarding%'效率比全表扫描高很多，适用OLAP系统，  
OLTP系统里面用到的并不多。  
缺点：全文索引会占用大量空间有时比原表本身占的空间还多，bug较多，维护困难。  
实验：全文索引性能优势  
创建一个表包含2个字段  
[LS@LEO](mailto:LS@LEO)> create table leo\_text\_t1 (id int,name varchar(10));  
Table created.  
在name字段上创建B-tree索引，但检索的时候并没有用，还是全表扫描  
[LS@LEO](mailto:LS@LEO)> create index leo\_text\_t1\_bt\_index on leo\_text\_t1(name);  
Index created.  
插入4条记录  
insert into leo\_text\_t1 values(1,'Tom');  
insert into leo\_text\_t1 values(2,'Tom Tom');  
insert into leo\_text\_t1 values(1,'Tom');  
insert into leo\_text\_t1 values(2,'Tom Tom');  
commit;  
[LS@LEO](mailto:LS@LEO)> select \* from leo\_text\_t1;  
        ID NAME  
---------- ----------  
         1 Tom  
         2 Tom Tom  
         1 Tom  
         2 Tom Tom  
我们在创建一个表，并在name字段上创建全文索引  
create table leo\_text\_t2 as select \* from leo\_text\_t1;  
创建全文索引的前提  
ORACLE10g 创建全文索引过程：  
1,首先查看ORACLE是否已安装“全文检索工具”  
  通过查看是否存在 CTXSYS 用户，CTXAPP角色即可判断。  
[LS@LEO](mailto:LS@LEO" \t "_blank)> select username from dba\_users;  
USERNAME  
------------------------------  
LS  
CTXSYS   默认是没有的，需要安装2个脚本catctx.sql，drdefus.sql  
2,如果ORACLE没有安装“全文检索工具”，则使用以下步骤手工安装。  
  a)进入ORACLE安装目录  
  cd $ORACLE\_HOME  
  b)使用 DBA 角色登陆数据库  
  sqlplus sys/sys as sysdba  
  c)查看表空间文件存放路径  
  select name from v$datafile;  
  d)为 CTXSYS 用户创建表空间  
  CREATE TABLESPACE ctxsys  
  LOGGING  
  DATAFILE '/u01/app/oracle/oradata/LEO/file1/ctxsys01.dbf'  
  SIZE 32m  
  AUTOEXTEND ON  
  NEXT 32m MAXSIZE 2048m  
  EXTENT MANAGEMENT LOCAL ;  
  e)创建 CTXSYS 用户，创建 CTXAPP 角色  
  @?/ctx/admin/catctx.sql ctxsys ctxsys temp1 nolock  
  --(密码、表空间、临时表空间、用户状态)  
  --如果当前sql脚本无执行权限，请手工添加。  
  f)为 CTXSYS 执行初始化工作，如果没有此操作，后续操作会失败。  
  connect ctxsys/ctxsys;  
  @?/ctx/admin/defaults/drdefus.sql  
3,创建全文索引  
  a)创建词法分析器及相关表  
  --词法分析器  
execute ctx\_ddl.create\_preference('offerProdAddrLexer','CHINESE\_LEXER');  
  --词法  
execute ctx\_ddl.create\_preference('offerProdAddrList', 'BASIC\_WORDLIST');  
execute  ctx\_ddl.set\_attribute('offerProdAddrList','PREFIX\_INDEX','TRUE');  
execute  ctx\_ddl.set\_attribute('offerProdAddrList','PREFIX\_MIN\_LENGTH',1);  
execute  ctx\_ddl.set\_attribute('offerProdAddrList','PREFIX\_MAX\_LENGTH', 5);  
execute  ctx\_ddl.set\_attribute('offerProdAddrList','SUBSTRING\_INDEX', 'YES');  
b)创建全文索引  
[LS@LEO](mailto:LS@LEO)> conn ctxsys/ctxsys  
Connected.  
[CTXSYS@LEO](mailto:CTXSYS@LEO)> create index ls.leo\_text\_t2\_text\_index on ls.leo\_text\_t2(name) indextype is ctxsys.context;  
[CTXSYS@LEO](mailto:CTXSYS@LEO)> conn ls/ls  
Connected.  
[LS@LEO](mailto:LS@LEO)> set autotrace on;  
[LS@LEO](mailto:LS@LEO)> select \* from leo\_text\_t1 where name like '%Tom%';  
        ID NAME  
---------- ----------  
         1 Tom  
         2 Tom Tom  
         1 Tom  
         2 Tom Tom  
Execution Plan   执行计划  
----------------------------------------------------------  
Plan hash value: 3687902158  
---------------------------------------------------------------------------------  
| Id  | Operation         | Name        | Rows  | Bytes | Cost (%CPU)| Time     |  
---------------------------------------------------------------------------------  
|   0 | SELECT STATEMENT  |             |     4 |    80 |     3   (0)| 00:00:01 |  
|\*  1 |  TABLE ACCESS FULL| LEO\_TEXT\_T1 |     4 |    80 |     3   (0)| 00:00:01 |  
---------------------------------------------------------------------------------  
全表扫描，cpu使用率3%  
Predicate Information (identified by operation id):  
---------------------------------------------------  
   1 - filter("NAME" LIKE '%Tom%')  
Note  
-----  
   - dynamic sampling used for this statement动态采样  
Statistics   统计信息  
----------------------------------------------------------  
         56  recursive calls  
          0  db block gets  
         23  consistent gets  全表扫描没有使用B-tree索引，导致23次一致性读  
          0  physical reads  
          0  redo size  
        538  bytes sent via SQL\*Net to client  
        381  bytes received via SQL\*Net from client  
          2  SQL\*Net roundtrips to/from client  
          2  sorts (memory)  
          0  sorts (disk)  
          4  rows processed  
[LS@LEO](mailto:LS@LEO)> select \* from leo\_text\_t2 where contains(name,'Tom')>0;  
        ID NAME  
---------- ----------  
         1 Tom  
         2 Tom Tom  
         1 Tom  
         2 Tom Tom  
Execution Plan   执行计划  
----------------------------------------------------------  
Plan hash value: 2789465217  
------------------------------------------------------------------------------------------------------  
| Id  | Operation                   | Name                   | Rows  | Bytes | Cost (%CPU)| Time     |  
------------------------------------------------------------------------------------------------------  
|   0 | SELECT STATEMENT            |                        |     1 |    32 |     4   (0)| 00:00:01 |  
|   1 |  TABLE ACCESS BY INDEX ROWID| LEO\_TEXT\_T2            |     1 |    32 |     4   (0)| 00:00:01 |  
|\*  2 |   DOMAIN INDEX              | LEO\_TEXT\_T2\_TEXT\_INDEX |       |       |     4   (0)| 00:00:01 |  
------------------------------------------------------------------------------------------------------  
Predicate Information (identified by operation id):  
---------------------------------------------------  
   2 - access("CTXSYS"."CONTAINS"("NAME",'Tom')>0)  
Note  
-----  
   - dynamic sampling used for this statement动态采样  
  
Statistics   统计信息  
----------------------------------------------------------  
         11  recursive calls  
          0  db block gets  
         19  consistent gets     
          0  physical reads  
          0  redo size  
        545  bytes sent via SQL\*Net to client  
        381  bytes received via SQL\*Net from client  
          2  SQL\*Net roundtrips to/from client  
          0  sorts (memory)  
          0  sorts (disk)  
          4  rows processed  
小结：从如上实验来看，当我们检索大量文字的时候使用全文索引要比全表扫描快很多了，有弊就有利，由于全文索引会占用大量空间提前预估全文索引大小保留出足够的空间，context类型全文索引不是基于事务的，无法保证索引和数据实时同步，DML完成后，如果在全文索引中查不到键值时，可以通过手工or定时任务来刷新同步，而B-tree、位图都是实时的。