Oracle 之不可见索引

1.1 BLOG 文档结构图

1.2 前言部分

1.2.1 导读和注意事项

各位技术爱好者，看完本文后，你可以掌握如下的技能，也可以学到一些其它你所不知道的知识，~O(∩_∩)O~：

① Oracle 不可见索引的使用

Tips：

① 本文在 ITpub（http://blog.itpub.net/26736162）、博客园（http://www.cnblogs.com/lhrbest）和微信公众号（xiaomaimiaolhr）有同步更新

② 文章中用到的所有代码，相关软件，相关资料请前往小麦苗的云盘下载（http://blog.itpub.net/26736162/viewspace-1624453/）

③ 若文章代码格式有错乱，推荐使用搜狗、360 或 QQ 浏览器，也可以下载 pdf 格式的文档来查看，pdf 文档下载地址：http://blog.itpub.net/26736162/viewspace-1624453/，另外 itpub 格式显示有问题，可以去博客园地址阅读

④ 本篇 BLOG 中命令的输出部分需要特别关注的地方我都用灰色背景和粉红色字体来表示，比如下边的例子中，

thread 1 的最大归档日志号为 33，thread 2 的最大归档日志号为 43 是需要特别关注的地方；而命令一般使用黄色背景和红色字体标注；对代码或代码输出部分的注释一般采用蓝色字体表示。

```
  List of Archived Logs in backup set 11
  Thrd Seq     Low SCN    Low Time            Next SCN   Next Time
  ---- ------- ---------- ------------------- ---------- ---------
  1    32      1621589    2015-05-29 11:09:52 1625242    2015-05-29 11:15:48
  1    33      1625242    2015-05-29 11:15:48 1625293    2015-05-29 11:15:58
  2    42      1613951    2015-05-29 10:41:18 1625245    2015-05-29 11:15:49
  2    43      1625245    2015-05-29 11:15:49 1625253    2015-05-29 11:15:53

[ZHLHRDB1:root]:/>lsvg -o
T_XDESK_APP1_vg
rootvg
[ZHLHRDB1:root]:/>
00:27:22 SQL> alter tablespace idxtbs read write;

====》2097152*512/1024/1024/1024=1G
```

本文如有错误或不完善的地方请大家多多指正，ITPUB 留言或 QQ 皆可，您的批评指正是我写作的最大动力。

1.2.2 相关参考文章连接

Oracle 之虚拟索引：http://blog.itpub.net/26736162/viewspace-2123687/

Oracle 索引的监控：http://blog.itpub.net/26736162/viewspace-2120752/

oracle 如何预估将要创建的索引的大小：http://blog.itpub.net/26736162/viewspace-1381160/

1.3 不可见索引(Invisible Indexes)

您常常感到疑惑，索引是否真的有利于用户的查询？它可能有利于一个查询，但会影响 10 个其他查询。索引肯定会对 INSERT 语句造成负面影响，也会执行潜在的删除和更新操作，这取决于 WHERE 条件是否在索引中包括该列。一个相关的问题是，使用索引时，如果该索引被删除，会对查询性能造成什么影响？当然，您可以删除索引并查看对查询的影响，但说起来容易做起来难。索引实际上如何有助于查询？您必须重新定义索引，为此，需要进行重新创建。完全重新创建之后，就没有人能使用它了。重新创建索引也是一个昂贵的过程；它会占用许多有用的数据库资源。

我们经常在数据库上建索引或删除索引，由于索引对 SQL 的执行性能影响非常大，有可能变得很好，也有可能变得很差，在线下开发环境我们可以充分测试，对于创建或删除索引没什么问题。但是在线上环境，由于高并发的访问，如果我们删除了一个重要的大索引(GB 以上)，删除后才发现大量 SQL 性能变差，很快主机就 LOAD 飙升，系统无法运行了，由于索引已经删除，并且很大，要当场重建基本不可能，因为这个索引巨大，创建估计要几分钟甚至几个小时，况且这时主机已经基本没有响应，IO 全部用光，只能把应用停了，等索引建好后再开始打开应用，等发生这样的事才会为自己的失误而后悔。那我们有没有办法让删除索引的风险降低呢，答案是有！即本文介绍的不可见索引。

索引维护是 DBA 的一项重要工作。当一个系统运行很长一段时间，经过需求变更、结构设计变化后，系统中就可能会存在一些不会被使用的索引，或者使用效率很低的索引。这些索引的存在，不仅占用系统空间，而且会降低事务效率，增加系统的 waits。因此，我们需要找出那些无用或低效索引的索引并删除它们（找出无用索引可以通过索引监控的方法）。但是，直接删除索引还是存在一定风险的。例如，某些索引可能只是在一些周期的作业中被使用到，而如果监控周期没有覆盖到这些作业的触发点，就会认为索引是无用的而被删除。当作业启动后，可能就会对系统性能造成冲击。这时，可能就会手忙脚乱的去找回索引定义语句、重建索引。11G 之前，我们可以先不删除索引，而将其修改为 unusable。这样的话，索引的定义并未删除，只是索引不能再被使用也不会随着表数据的更新而更新。当需要重新使用该索引时，需要用 rebuild 语句重建、然后更新统计信息。对于一些大表来说，这个时间可能就非常长。在 11g 里，Oracle 提供了一个新的特性来降低直接删除索引或者禁用索引的风险，那就是索引不可见（Index Invisible）。

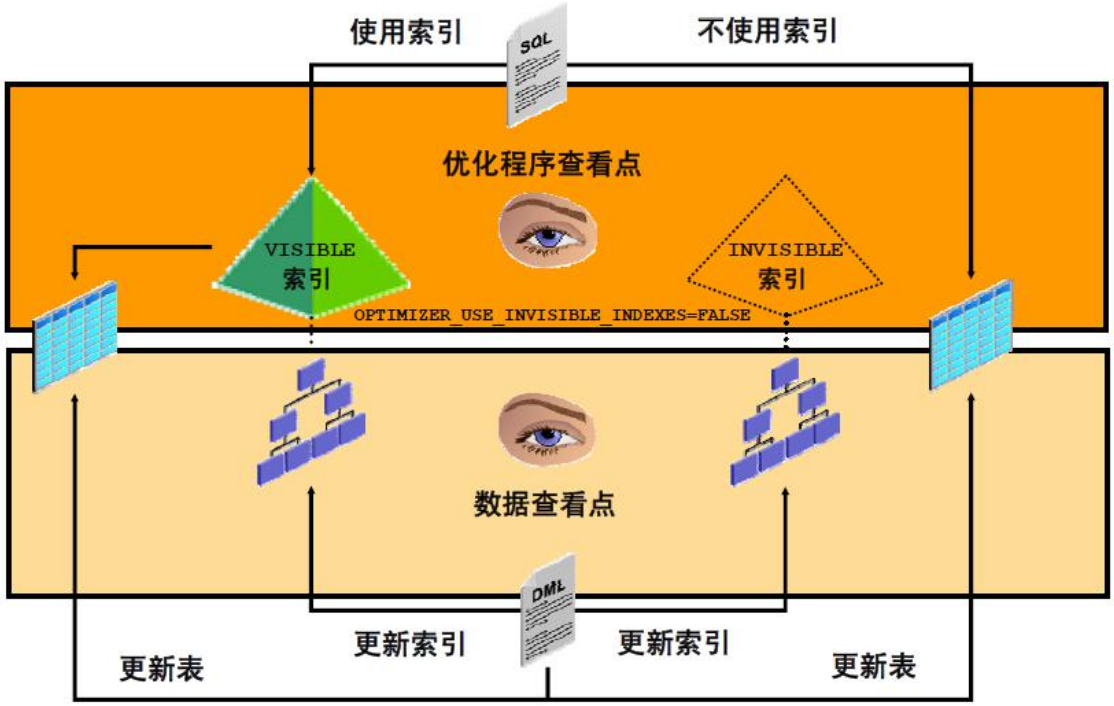
从版本 11g 开始，可以创建不可见的索引。优化程序会忽略不可见的索引，除非在会话或系统级别上将 OPTIMIZER_USE_INVISIBLE_INDEXES 初始化参数显式设置为 TRUE。**此参数的默认值是 FALSE。**

使索引不可见是使索引不可用或删除索引的一种替代办法。使用不可见的索引，可完成以下操作：

(1) 在删除索引之前测试对索引的删除。

(2) 对应用程序的特定操作或模块使用临时索引结构，这样就不会影响整个应用程序。

注意：与不可用的索引不同，不可见的索引在使用 DML 语句期间仍会得到维护。



当索引不可见时，优化程序生成的计划不会使用该索引。如果未发现性能下降，则可以删除该索引。还可以创建最初不可见的索引，执行测试，然后确定是否使该索引可见。

可以查询*_INDEXES 数据字典视图的 VISIBILITY 列来确定该索引是 VISIBLE 还是 INVISIBLE。

```
SQL> SELECT VISIBILITY FROM DBA_INDEXES WHERE INDEX_NAME='IDX_ID';
VISIBILIT
---------
VISIBLE
```

--创建不可见索引：

```
CREATE INDEX INDEX_NAME ON TABLE_NAME(COLUMN_NAME) INVISIBLE;
```

--修改索引是否可见：

```
ALTER INDEX INDEX_NAME INVISIBLE;
ALTER INDEX INDEX_NAME VISIBLE;
```

特点总结：

1、当索引变更为不可见的时候，只是对 oracle 的优化器不可见。

2、不可见索引在 DML 操作的时候也会被维护。

3、加 HNIT 对不可见索引无效。

4、可以通过修改 system 级别和 session 级别参数来使用不可见索引。

1.3.1 我的示例

创建表，不可见索引，并收集统计信息：

```
SYS@lhrdb> CREATE TABLE T_II_20160819_01_LHR AS SELECT * FROM DBA_OBJECTS;

Table created.

SYS@lhrdb> CREATE INDEX IDX_II_20160819 ON T_II_20160819_01_LHR(OBJECT_ID) INVISIBLE;

Index created.


SYS@lhrdb> SELECT VISIBILITY FROM DBA_INDEXES WHERE INDEX_NAME='IDX_II_20160819';

VISIBILIT
---------
INVISIBLE

SYS@lhrdb> EXEC DBMS_STATS.GATHER_TABLE_STATS(OWNNAME
=>USER,TABNAME=>'T_II_20160819_01_LHR',DEGREE=>2,CASCADE => TRUE);

PL/SQL procedure successfully completed.
```

--带 where 条件查询：

```
SYS@lhrdb> SHOW PARAMETER OPTIMIZER_USE_INVISIBLE_INDEXES

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
optimizer_use_invisible_indexes      boolean     FALSE
SYS@lhrdb> set line 9999
SYS@lhrdb> set autot traceonly exp
SYS@lhrdb> SELECT * FROM T_II_20160819_01_LHR WHERE OBJECT_ID=1;

Execution Plan
----------------------------------------------------------
Plan hash value: 700947541

-------------------------------------------------------------------------------------------
| Id  | Operation         | Name                 | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |                      |     1 |    98 |   343   (2)| 00:00:05 |
|*  1 |  TABLE ACCESS FULL| T_II_20160819_01_LHR |     1 |    98 |   343   (2)| 00:00:05 |
-------------------------------------------------------------------------------------------

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

   1 - filter("OBJECT_ID"=1)
```

--这里使用了全表扫描，根据唯一性，这里应该走索引的，我们加上 hint 试试

```
SYS@lhrdb> SELECT /*+index(T,IDX_II_20160819)*/ * FROM T_II_20160819_01_LHR T WHERE OBJECT_ID=1;

Execution Plan
----------------------------------------------------------
Plan hash value: 700947541

-------------------------------------------------------------------------------------------
```

```
| Id  | Operation         | Name                 | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |                      |     1 |    98 |   343   (2)| 00:00:05 |
|*  1 |  TABLE ACCESS FULL| T_II_20160819_01_LHR |     1 |    98 |   343   (2)| 00:00:05 |
-------------------------------------------------------------------------------------

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

   1 - filter("OBJECT_ID"=1)
```

--对于 invisible 的 index，使用 hint 也没有用。

--修改 OPTIMIZER_USE_INVISIBLE_INDEXES 参数为 TRUE，再次查询：

```
SYS@lhrdb> ALTER SESSION SET OPTIMIZER_USE_INVISIBLE_INDEXES=TRUE;

Session altered.

SYS@lhrdb> SELECT * FROM T_II_20160819_01_LHR WHERE OBJECT_ID=1;

Execution Plan
----------------------------------------------------------
Plan hash value: 2544197461

-------------------------------------------------------------------------------------------------
| Id  | Operation                   | Name                 | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |                      |     1 |    98 |     2   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| T_II_20160819_01_LHR |     1 |    98 |     2   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN          | IDX_II_20160819      |     1 |       |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------

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

   2 - access("OBJECT_ID"=1)
```

--这次使用了索引。关闭 OPTIMIZER_USE_INVISIBLE_INDEXES 参数，将索引改成 visible，再测试：

```
SYS@lhrdb> ALTER SESSION SET OPTIMIZER_USE_INVISIBLE_INDEXES=FALSE;

Session altered.

SYS@lhrdb> ALTER INDEX IDX_II_20160819 VISIBLE;

Index altered.

SYS@lhrdb> SELECT * FROM T_II_20160819_01_LHR WHERE OBJECT_ID=1;

Execution Plan
----------------------------------------------------------
Plan hash value: 2544197461

-------------------------------------------------------------------------------------------------
| Id  | Operation                   | Name                 | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |                      |     1 |    98 |     2   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| T_II_20160819_01_LHR |     1 |    98 |     2   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN          | IDX_II_20160819      |     1 |       |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------

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

   2 - access("OBJECT_ID"=1)
```

--这次又正常了。

以上是 Oracle11g 的处理方法，但是在 Oracle9i 或 Oracle10g 中索引没有 invisible 的功能，我们如何处理呢？现在 Oracle 数据库一般都采用基于成本的计算方法来生成执行计划，只要索引的成本更低，ORACLE 就会选择使用索引，OK，那我们只要告诉 ORACLE 使用这个索引成本很高，它就不会使用这个索引，这样就达到了暂时让索引不可用的效果。相信很多人都知道 ORACLE 提供了 dbms_stats 包来管理对像的统计信息，通过 DBMS_STATS.SET_INDEX_STATS 函数我们可以强制设置统计信息，现在我们只要把索引的成本设置成非常大即可，如下所示。

--设置非常离谱的统计信息，让 ORACLE 认为使用索引的成本很高

```
SYS@lhrdb> SELECT A.OWNER,A.INDEX_NAME,A.BLEVEL,A.LEAF_BLOCKS,A.NUM_ROWS FROM DBA_INDEXES A WHERE
INDEX_NAME='IDX_II_20160819';

OWNER                          INDEX_NAME                       BLEVEL LEAF_BLOCKS   NUM_ROWS
------------------------------ ------------------------------ ---------- ----------- ----------
SYS                            IDX_II_20160819                         1         193      87133
SYS@lhrdb> EXEC DBMS_STATS.SET_INDEX_STATS(OWNNAME => user,INDNAME => 'IDX_II_20160819',INDLEVEL =>
10,NUMLBLKS => 1000000000,NUMROWS => 100000000000,NO_INVALIDATE => FALSE );

PL/SQL procedure successfully completed.
SYS@lhrdb> col NUM_ROWS format 999999999999999
SYS@lhrdb> SELECT A.OWNER,A.INDEX_NAME,A.BLEVEL,A.LEAF_BLOCKS,A.NUM_ROWS FROM DBA_INDEXES A WHERE
INDEX_NAME='IDX_II_20160819';

OWNER                          INDEX_NAME                       BLEVEL LEAF_BLOCKS         NUM_ROWS
------------------------------ ------------------------------ ---------- ----------- ----------------
SYS                            IDX_II_20160819                        10  1000000000     100000000000
```

no_invalidate=false 表示让 CACHE 中的执行计划立即失效，重新按现在的统计信息生成 SQL 执行计划。验证一下是否生效：

```
SYS@lhrdb> set autot on9
SYS@lhrdb> SELECT * FROM T_II_20160819_01_LHR WHERE OBJECT_ID=1;

no rows selected


Execution Plan
----------------------------------------------------------
Plan hash value: 700947541

-------------------------------------------------------------------------------------------
| Id  | Operation         | Name                 | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |                      |     1 |    98 |   343   (2)| 00:00:05 |
|*  1 |  TABLE ACCESS FULL| T_II_20160819_01_LHR |     1 |    98 |   343   (2)| 00:00:05 |
-------------------------------------------------------------------------------------------

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

   1 - filter("OBJECT_ID"=1)


Statistics
```

```
-------------------------------------------------------------
         0  recursive calls
         0  db block gets
      2491  consistent gets
         0  physical reads
         0  redo size
      1343  bytes sent via SQL*Net to client
       509  bytes received via SQL*Net from client
         1  SQL*Net roundtrips to/from client
         0  sorts (memory)
         0  sorts (disk)
         0  rows processed
```

1.3.2 OCP

An index called ORD_CUSTNAME_IX has been created on the CUSTNAME column in the ORDERS table using the following command:

SQL>CREATE INDEX ord_custname_ix ON orders(custname);

The ORDERS table is frequently queried using the CUSTNAME column in the WHERE clause.You want to check the impact on the performance of the queries if the index is not available.You do not want the index to be dropped or rebuilt to perform this test.

Which is the most efficient method of performing this task?

A.disabling the index

B.making the index invisible

C.making the index unusable

D.using the MONITORING USAGE clause for the index

Answer:B

答案解析：题目要求在不能删除和重建的情况下来测试索引的性能。

对于选项 A，索引不能被禁用。所以，选项 A 错误。

对于选项 B，让索引不可用，为正确选项。所以，选项 B 正确。

对于选项 C，让索引不可用之后还是得重建索引。所以，选项 C 错误。

对于选项 D，监控索引并不能测试索引在不可用的情况下对系统的性能影响。所以，选项 D 错误。

所以，本题的答案为 B。

About Me

- 本文作者：小麦苗，只专注于数据库的技术，更注重技术的运用

- 本文在 ITpub（http://blog.itpub.net/26736162）、博客园(http://www.cnblogs.com/lhrbest)和个人微信公众号（xiaomaimiaolhr）上有同步更新
- QQ 群：230161599 微信群：私聊
- 本文 itpub 地址：http://blog.itpub.net/26736162/viewspace-2124044/ 本文博客园地址：http://www.cnblogs.com/lhrbest/articles/5808049.html
- 本文 pdf 版：http://yunpan.cn/cdEQedhCs2kFz （提取码：ed9b）
- 小麦苗分享的其它资料：http://blog.itpub.net/26736162/viewspace-1624453/
- 联系我请加 QQ 好友(642808185)，注明添加缘由
- 于 2016-08-19 09:00~ 2016-08-19 19:00 在中行完成
- 【版权所有，文章允许转载，但须以链接方式注明源地址，否则追究法律责任】

.....

长按识别二维码或微信客户端扫描下边的二维码来关注小麦苗的微信公众号：xiaomaimiaolhr，学习最实用的数据库技术。