

Oracle 索引的监控

1.1 BLOG 文档结构图

Oracle 索引的监控
1.1 BLOG 文档结构图
1.2 前言部分
1.2.1 导读和注意事项
1.3 相关知识点扫盲(摘自网络)
1.4 索引监控的方法
1.4.1 方法一：开启监控功能
1.4.1.1 个人实验
1.4.1.2 实验中用到的 SQL
1.4.2 方法二：查看历史的执行计划进行分析
About Me

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(n_n)o~：

① 掌握 oracle 中索引的监控方法

② sys.col_usage\$的初步了解

Tips：

① 本文在 ITpub (<http://blog.itpub.net/26736162>) 和博客园 (<http://www.cnblogs.com/lhrbest>) 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/>

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。

1.3 相关知识点扫盲(摘自网络)

合理的为数据库表上创建战略性索引，可以极大程度的提高查询性能。但事实上日常中所创建的索引并非战略性索引，恰恰是大量冗余或是根本没有用到的索引耗用了大量的存储空间，导致 DML 性能低下。应用程序在开发时，可能会建立众多索引，但是这些索引的使用到底怎么样，是否有些索引一直都没有用到过，这要求我们对这些索引进行监控，以便确定他们的使用情况，并为是否可以清除它们给出依据。

冗余索引的弊端：

大量冗余和无用的索引导致整个数据库性能低下，耗用了大量的 CPU 与 I/O 开销，具体表现如下：

- a、浪费大量的存储空间，尤其是大表的索引，浪费的存储空间尤其可观 (索引段的维护与管理)
- b、增加了 DML 操作 (UPDATE、INSERT、DELETE) 的开销
- c、耗用大量统计信息 (索引) 收集的时间
- d、结构性验证时间
- f、增加了恢复所需的时间

本文介绍两种方式：

第一：开启监控功能；

第二：查看历史的执行计划，进行分析；

1.4 索引监控的方法

1.4.1 方法一：开启监控功能

1、单个索引监控

- a、对于单个索引的监控，可以使用下面的命令来完成

```
alter index <INDEX_NAME> monitoring usage;
```

- b、关闭索引监控

```
alter index <INDEX_NAME> nomonitoring usage;
```

- c、观察监控结果 (查询 v\$object_usage 视图)

```
select * from v$object_usage;
```

2、schema 级别索引监控

如果我们想在系统中监控所有的索引，那么我们可以通过下面脚本实现监控数据库所有的索引。注意我们要排除一些系统表的索引、以及 LOB indexes。原因有下面两个：

1 LOB indexes 不能修改，否则会报 ORA-22864 错误 (ORA-22864: cannot ALTER or DROP LOB indexes)。

```

2:ORA-00701: object necessary for warmstarting database cannot be altered

ORA-00701: object necessary for warmstarting database cannot be altered
00701. 00000 - "object necessary for warmstarting database cannot be altered"
*Cause: Attempt to alter or drop a database object (table, cluster, or
index) which are needed for warmstarting the database.
*Action: None.

```

直接执行脚本来开启索引监控，当然监控索引时长非常重要，太短的话有可能导致查询出来的数据有问题，一般

建议监控一周后即可，OLAP 系统则需要适当延长监控的时间。

```

SELECT 'ALTER INDEX ' || owner || '.' || index_name || ' MONITORING USAGE;'
enable_monitor,
      'ALTER INDEX ' || owner || '.' || index_name ||
      ' NOMONITORING USAGE;' disable_monitor
FROM dba_indexes
WHERE INDEX_TYPE != 'LOB'
and owner IN
      (SELECT username FROM dba_users WHERE account_status = 'OPEN')
AND owner NOT IN ('SYS',
                  'SYSTEM',
                  'PERFSTAT',
                  'MGMT_VIEW',
                  'MONITOR',
                  'SYSMAN',
                  'DBSNMP')
AND owner not like '%SYS%';

```

监控一个月就大概可以知道那些是无用的索引了。

虽然 v\$object_usage 表能记录索引监控和使用的状态，但它不能统计索引被使用的次数和频率，只记录了在开启索引监控的时间段索引是否被使用过，这一点要值的注意。

另外需要注意的 2 点：

- ① 10g 在收集统计信息时会导致索引被监控、这并非 SQL 语句产生、而在 11g 则不会出现这种情况了
- ② 外键索引不会因为主表的 DML 操作而被监控到、不要因为该索引没用而将它给删了

1.4.1.1 个人实验

新建 1 个表 TB_LHR_20160622 , 并创建 2 个索引 :

```
SYS@rac1hr2> select * from v$version;

BANNER
-----
Oracle Database 11g Enterprise Edition Release 11.2.0.4.0 - 64bit Production
PL/SQL Release 11.2.0.4.0 - Production
CORE 11.2.0.4.0 Production
TNS for IBM/AIX RISC System/6000: Version 11.2.0.4.0 - Production
NLSRTL Version 11.2.0.4.0 - Production
SYS@rac1hr2> Create Table TB_LHR_20160622 nologging As select * from dba_objects;

Table created.

SYS@rac1hr2> create index ind_TB_LHR_20160622_id on TB_LHR_20160622(object_id);

Index created.

SYS@rac1hr2> create index ind_TB_LHR_20160622_name on TB_LHR_20160622(object_name);

Index created.
```

查询 v\$object_usage 视图 , 收集统计信息 :

```
SYS@rac1hr2> select * from v$object_usage;

no rows selected

SYS@rac1hr2> BEGIN
2      dbms_stats.gather_table_stats(USER,
3                                     'TB_LHR_20160622',
4                                     cascade      => TRUE,
5                                     degree       => 8);
6 END;
7 /

PL/SQL procedure successfully completed.

SYS@rac1hr2> select * from v$object_usage;

no rows selected
```

开启索引的监控 :

```
SYS@rac1hr2> alter index ind_TB_LHR_20160622_id monitoring usage;

Index altered.

SYS@rac1hr2> COL INDEX_NAME FOR A25
SYS@rac1hr2> COL TABLE_NAME FOR A20
SYS@rac1hr2> COL MONITORING FOR A10
SYS@rac1hr2> COL USED FOR A10
SYS@rac1hr2> COL START_MONITORING FOR A20
SYS@rac1hr2> COL END_MONITORING FOR A20
SYS@rac1hr2> select * from v$object_usage;

INDEX_NAME          TABLE_NAME          MONITORING USED          START_MONITORING          END_MONITORING
-----
```

```
IND_TB_LHR_20160622_ID    TB_LHR_20160622    YES    NO    06/22/2016 15:15:54
```

```
SYS@rac1hr2> alter index ind_TB_LHR_20160622_name monitoring usage;
```

```
Index altered.
```

```
SYS@rac1hr2> select count(1) from TB_LHR_20160622 t where t.object_id=88;
```

```
COUNT(1)
```

```
1
```

```
SYS@rac1hr2> explain plan for select count(1) from TB_LHR_20160622 t where t.object_id=88;
```

```
Explained.
```

```
SYS@rac1hr2> select * from table(dbms_xplan.display());
```

```
PLAN_TABLE_OUTPUT
```

```
Plan hash value: 2688591802
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	5	1 (0)	00:00:01
1	SORT AGGREGATE		1	5		
* 2	INDEX RANGE SCAN	IND_TB_LHR_20160622_ID	1	5	1 (0)	00:00:01

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

```
2 - access("T"."OBJECT_ID"=88)
```

```
14 rows selected.
```

```
SYS@rac1hr2> COL INDEX_NAME FOR A25
```

```
SYS@rac1hr2> COL TABLE_NAME FOR A20
```

```
SYS@rac1hr2> COL MONITORING FOR A10
```

```
SYS@rac1hr2> COL USED FOR A10
```

```
SYS@rac1hr2> COL START_MONITORING FOR A20
```

```
SYS@rac1hr2> COL END_MONITORING FOR A20
```

```
SYS@rac1hr2> select * from v$object_usage;
```

INDEX_NAME	TABLE_NAME	MONITORING	USED	START_MONITORING	END_MONITORING
IND_TB_LHR_20160622_ID	TB_LHR_20160622	YES	YES	06/22/2016 15:15:54	
IND_TB_LHR_20160622_NAME	TB_LHR_20160622	YES	NO	06/22/2016 15:16:17	

注意：SELECT * FROM V\$OBJECT_USAGE；只能查看当前用户下被监控的索引信息。即使 sys、system 用

户也不能查看其它用户的信息，如下，但我们可以创建一个视图来解决这个问题。

```
SYS@rac1hr2> conn scott/tiger
```

```
Connected.
```

```
SCOTT@rac1hr2> select * from v$object_usage;
```

```
no rows selected
```

```
SCOTT@rac1hr2> conn / as sysdba
```

```
Connected.
```

```
SYS@rac1hr2> create or replace view vw_INDEX_USAGE_lhr AS
```

```
2 SELECT U.NAME OWNER,
3 IO.NAME INDEX_NAME,
4 T.NAME TABLE_NAME,
```

```

5      DECODE(BITAND(I.FLAGS, 65536), 0, 'NO', 'YES') MONITORING,
6      DECODE(BITAND(OU.FLAGS, 1), 0, 'NO', 'YES') USED,
7      OU.START_MONITORING START_MONITORING,
8      OU.END_MONITORING END_MONITORING
9  FROM SYS.USER$      U,
10     SYS.OBJ$        IO,
11     SYS.OBJ$        T,
12     SYS.IND$        I,
13     SYS.OBJECT_USAGE OU
14 WHERE I.OBJ# = OU.OBJ#
15     AND IO.OBJ# = OU.OBJ#
16     AND T.OBJ# = I.BO#
17     AND U.USER# = IO.OWNER#;

```

View created.

```
SYS@rac1hr2> create or replace public synonym syn_INDEX_USAGE_1hr for sys.vw_INDEX_USAGE_1hr;
```

Synonym created.

```
SYS@rac1hr2> grant select on sys.vw_INDEX_USAGE_1hr to public;
```

Grant succeeded.

```
SYS@rac1hr2> conn scott/tiger
```

Connected.

```
SCOTT@rac1hr2> set line 9999 pagesize 9999
```

```
SCOTT@rac1hr2> col owner format A10
```

```
SCOTT@rac1hr2> COL INDEX_NAME FOR A25
```

```
SCOTT@rac1hr2> COL TABLE_NAME FOR A20
```

```
SCOTT@rac1hr2> COL MONITORING FOR A10
```

```
SCOTT@rac1hr2> COL USED FOR A10
```

```
SCOTT@rac1hr2> COL START_MONITORING FOR A20
```

```
SCOTT@rac1hr2> COL END_MONITORING FOR A20
```

```
SCOTT@rac1hr2> SELECT * FROM syn_INDEX_USAGE_1hr;
```

OWNER	INDEX_NAME	TABLE_NAME	MONITORING	USED	START_MONITORING	END_MONITORING
SYS	IND_TB_LHR_20160622_ID	TB_LHR_20160622	YES	YES	06/22/2016 15:15:54	
SYS	IND_TB_LHR_20160622_NAME	TB_LHR_20160622	YES	NO	06/22/2016 15:16:17	

取消索引的监控：

```
SCOTT@rac1hr2> CONN / AS SYSDBA
```

Connected.

```
SYS@rac1hr2> alter index ind_TB_LHR_20160622_id nomonitoring usage;
```

Index altered.

```
SYS@rac1hr2> SELECT * FROM syn_INDEX_USAGE_1hr;
```

OWNER	INDEX_NAME	TABLE_NAME	MONITORING	USED	START_MONITORING	END_MONITORING
SYS	IND_TB_LHR_20160622_ID	TB_LHR_20160622	NO	YES	06/22/2016 15:15:54	06/22/2016 15:22:30
SYS	IND_TB_LHR_20160622_NAME	TB_LHR_20160622	YES	NO	06/22/2016 15:16:17	

```
SYS@rac1hr2> alter index ind_TB_LHR_20160622_name nomonitoring usage;
```

Index altered.

```
SYS@rac1hr2> SELECT * FROM syn_INDEX_USAGE_1hr;
```

OWNER	INDEX_NAME	TABLE_NAME	MONITORING	USED	START_MONITORING	END_MONITORING
-------	------------	------------	------------	------	------------------	----------------

SYS	IND_TB_LHR_20160622_ID	TB_LHR_20160622	NO	YES	06/22/2016 15:15:54	06/22/2016 15:22:30
SYS	IND_TB_LHR_20160622_NAME	TB_LHR_20160622	NO	NO	06/22/2016 15:22:45	06/22/2016 15:23:12

1.4.1.2 实验中用到的 SQL

```
drop table TB_LHR_20160622 purge;
Create Table TB_LHR_20160622 nologging As select * from dba_objects;
create index ind_TB_LHR_20160622_id on TB_LHR_20160622(object_id);
create index ind_TB_LHR_20160622_name on TB_LHR_20160622(object_name);
```

```
select * from v$object_usage;
```

```
BEGIN
```

```
    dbms_stats.gather_table_stats(USER,
                                   'TB_LHR_20160622',
                                   cascade      => TRUE,
                                   degree       => 8);
```

```
END;
```

```
/
```

```
alter index ind_TB_LHR_20160622_id monitoring usage;
alter index ind_TB_LHR_20160622_name monitoring usage;
```

```
select count(1) from TB_LHR_20160622 t where t.object_id=88;
```

```
set line 9999 pagesize 9999
col owner format A10
COL INDEX_NAME FOR A25
COL TABLE_NAME FOR A20
COL MONITORING FOR A10
COL USED FOR A10
COL START_MONITORING FOR A20
COL END_MONITORING FOR A20
select * from v$object_usage;
```

注意：SELECT * FROM V\$OBJECT_USAGE；只能查看当前用户下被监控的索引信息。即使 sys、

system 用户也不能查看其它用户的信息。

	INDEX_NAME	TABLE_NAME	MONITORING	USED	START_MONITORING	END_MONITORING
1	IND_TB_LHR_20160622_ID	TB_LHR_20160622	YES	YES	06/22/2016 13:35:35	
2	IND_TB_LHR_20160622_NAME	TB_LHR_20160622	YES	NO	06/22/2016 13:35:41	

```
alter index ind_TB_LHR_20160622_id nomonitoring usage;
alter index ind_TB_LHR_20160622_name nomonitoring usage;
```


--- drop table i purge; 表删掉后 v\$object_usage 中关于监控的信息也删除了

----切换用户后查询 select * from v\$object_usage; 查询不到数据，下边这个 SQL 可以查询任何用户下的索引使用情况

```
create or replace view vw_INDEX_USAGE_lhr      AS
SELECT U.NAME OWNER,
       IO.NAME INDEX_NAME,
       T.NAME TABLE_NAME,
       DECODE(BITAND(I.FLAGS, 65536), 0, 'NO', 'YES') MONITORING,
       DECODE(BITAND(OU.FLAGS, 1), 0, 'NO', 'YES') USED,
       OU.START_MONITORING START_MONITORING,
       OU.END_MONITORING END_MONITORING
FROM   SYS.USER$      U,
       SYS.OBJ$       IO,
       SYS.OBJ$       T,
       SYS.IND$       I,
       SYS.OBJECT_USAGE OU
WHERE  I.OBJ# = OU.OBJ#
      AND IO.OBJ# = OU.OBJ#
      AND T.OBJ# = I.BO#
      AND U.USER# = IO.OWNER#;
```

```
create or replace public synonym syn_INDEX_USAGE_lhr for
sys.vw_INDEX_USAGE_lhr;
```

```
set line 9999 pagesize 9999
col owner format A10
col INDEX_NAME FOR A25
col TABLE_NAME FOR A20
col MONITORING FOR A10
col USED FOR A10
col START_MONITORING FOR A20
col END_MONITORING FOR A20
SELECT * FROM syn_INDEX_USAGE_lhr;
```

批量监控系统的所有索引：

```
SELECT 'ALTER INDEX ' || owner || '.' || index_name || ' MONITORING USAGE;'
enable_monitor,
       'ALTER INDEX ' || owner || '.' || index_name ||
       ' NOMONITORING USAGE;' disable_monitor
FROM   dba_indexes
WHERE  INDEX_TYPE != 'LOB'
      and owner IN
```

```
(SELECT username FROM dba_users WHERE account_status = 'OPEN')
AND owner NOT IN ('SYS',
                  'SYSTEM',
                  'PERFSTAT',
                  'MGMT_VIEW',
                  'MONITOR',
                  'SYSMAN',
                  'DBSNMP')
AND owner not like '%SYS%';
```

1.4.2 方法二：查看历史的执行计划进行分析

虽然 v\$object_usage 表能记录索引监控和使用的状态，但它不能统计索引被使用的次数和频率，只记录了在开启索引监控的时间段索引是否被使用过，因此想详细了解索引的使用情况我们可以利用 AWR 的一些视图 dba_hist_sql_plan 和 dba_hist_sqlstat 来弄清楚数据库访问某个索引的次数、索引访问的类型，如索引范围扫描或索引唯一扫描。

```
WITH tmp1 AS
(SELECT i.OWNER INDEX_OWNER,
       i.table_owner,
       TABLE_NAME,
       INDEX_NAME,
       INDEX_TYPE,
       (select nb.created
        from dba_objects nb
        WHERE nb.owner = i.owner
              and nb.object_name = i.index_name
              and nb.subobject_name is null) created,
       (SUM(S.bytes) / 1024 / 1024) INDEX_MB
FROM DBA_SEGMENTS S, DBA_INDEXES I
WHERE i.INDEX_NAME = s.SEGMENT_NAME
      and i.owner = s.owner
      and s.owner not like '%SYS%'
/*and s.owner = 'FUNDZ'*/
GROUP BY i.OWNER, i.table_owner, TABLE_NAME, INDEX_NAME, INDEX_TYPE
HAVING SUM(S.BYTES) > 1024 * 1024),
tmp2 as
(SELECT index_owner,
       index_name,
       plan_operation,
```

```

(SELECT min(to_char(nb.begin_interval_time, 'YYYY-MM-DD
HH24:MI:SS'))
      FROM dba_hist_snapshot nb
      where nb.snap_id = v.min_snap_id) min_date,
(SELECT max(to_char(nb.end_interval_time, 'YYYY-MM-DD
HH24:MI:SS'))
      FROM dba_hist_snapshot nb
      where nb.snap_id = v.max_snap_id) max_date,
counts
FROM (SELECT d.object_owner index_owner,
            d.object_name index_name,
            d.operation || ' ' || d.options plan_operation,
            min(h.snap_id) min_snap_id,
            max(h.snap_id) max_snap_id,
            COUNT(1) counts
      FROM dba_hist_sql_plan d, dba_hist_sqlstat h
      WHERE /*d.object_owner = 'FUNDZ'
            AND */
            d.operation LIKE '%INDEX%'
      AND d.sql_id = h.sql_id
      GROUP BY d.object_owner, d.object_name, d.operation, d.options)
v)
SELECT a.table_owner,
       a.TABLE_NAME,
       a.index_owner,
       a.index_name,
       a.created,
       a.INDEX_TYPE,
       a.INDEX_MB,
       b.plan_operation,
       min_date,
       max_date,
       counts
from tmp1 a
left outer join tmp2 b
  on (a.index_owner = b.index_owner and a.index_name = b.index_name);

```

TABLE	TABLE_NA	INDEX	INDEX_NAME	CREATED	INDEX_T	INDEX	PLAN_OPERATION	MIN_DATE	MAX_DATE	COUNTS			
AI	C	WWV_FLO	AI	C	WWV_FLOW	2011/10/21 4	NORMAL	5	2016-06-13 22:00	2016-06-22 15:00			
FI	Z	FZ	OG	F	Z	FZIFLOG_ID	2016/5/13 15	NORMAL	9	2016-06-13 22:00	2016-06-22 15:00		
FI	Z	FZ	OG	F	Z	FZACLOG_ID	2016/5/13 15	NORMAL	9	2016-06-13 22:00	2016-06-22 15:00		
F	Z	FZ	OG	F	Z	FZACLOG_PK	2016/5/13 15	NORMAL	16	INDEX FAST FULL SC	2016-06-14 11:37	2016-06-14 13:00	1
F	Z	FZ	OG	F	Z	FZACLOG_PK	2016/5/13 15	NORMAL	16	INDEX SKIP SCAN	2016-06-14 10:00	2016-06-17 22:00	111
F	Z	FZ	OG	F	Z	FZACLOG_PK	2016/5/13 15	NORMAL	16	INDEX UNIQUE SCAL	2016-06-14 10:00	2016-06-17 22:00	208
F	Z	FZ	OG	F	Z	FZIFLOG_PK	2016/5/13 15	NORMAL	27	INDEX FULL SCAN	2016-06-14 11:37	2016-06-17 23:00	3
F	Z	FZ	OG	F	Z	FZ11F_IDX1	2016/5/13 15	NORMAL	32	INDEX FULL SCAN	2016-06-15 15:40	2016-06-17 23:00	3
F	Z	FZ	OG	F	Z	FZ11F_PK	2016/5/13 15	NORMAL	38	INDEX RANGE SCAN	2016-06-14 10:00	2016-06-21 23:00	66
F	Z	FZ	OG	F	Z	FZ11F_PK	2016/5/13 15	NORMAL	38	INDEX SKIP SCAN	2016-06-14 09:00	2016-06-17 22:00	805
F	Z	FZ	OG	F	Z	FZ11F_PK	2016/5/13 15	NORMAL	38	INDEX FULL SCAN	2016-06-17 20:35	2016-06-21 22:00	4
F	Z	FZ	OG	F	Z	FZ11F_PK	2016/5/13 15	NORMAL	38	INDEX UNIQUE SCAL	2016-06-14 09:00	2016-06-17 22:00	147
F	Z	FZ	OG	F	Z	FZ11F_PK	2016/5/13 15	NORMAL	38	INDEX SAMPLE FAST	2016-06-21 21:00	2016-06-21 22:00	1
F	Z	FZ	OG	F	Z	FZ11F_IDX2	2016/5/13 15	NORMAL	56	INDEX FULL SCAN	2016-06-15 15:40	2016-06-15 23:00	2
F	Z	FZ	OG	F	Z	FZIFLOGH_ID	2016/5/13 15	NORMAL	216		2016-06-13 22:00	2016-06-22 15:00	
F	Z	FZ	OG	F	Z	FZACLOGH_ID	2016/5/13 15	NORMAL	264		2016-06-13 22:00	2016-06-22 15:00	
F	Z	FZ	OG	F	Z	FZ32F_IDX2	2016/5/13 15	NORMAL	272		2016-06-15 15:40	2016-06-22 15:00	
F	Z	FZ	OG	F	Z	FZ32F_PK	2016/5/13 15	NORMAL	392	INDEX UNIQUE SCAL	2016-06-14 09:00	2016-06-17 22:00	255
F	Z	FZ	OG	F	Z	FZ32F_IDX1	2016/5/13 15	NORMAL	517		2016-06-13 22:00	2016-06-22 15:00	
F	Z	FZ	OG	F	Z	FZ32F_IDX3	2016/5/13 15	NORMAL	599	INDEX RANGE SCAN	2016-06-15 11:05	2016-06-17 22:00	73
DZ	FZ	FZIFLOGH	NDZ	FZIFLOGH_PK	2016/5/13 15	NORMAL	704		2016-06-13 22:00	2016-06-22 15:00			
DZ	FZ	FZACLOGH	NDZ	FZACLOGH_PK	2016/5/13 15	NORMAL	856		2016-06-13 22:00	2016-06-22 15:00			
DZ	FZ	FZ34F	NDZ	FZ34F_PK1	2016/5/13 15	NORMAL	1088		2016-06-13 22:00	2016-06-22 15:00			
DZ	FZ	FZ12F	NDZ	FZ12F_IDX1	2016/5/13 15	NORMAL	1664		2016-06-13 22:00	2016-06-22 15:00			
DZ	FZ	FZ12F	NDZ	FZ12F_PK	2016/5/13 15	NORMAL	3136	INDEX RANGE SCAN	2016-06-17 10:48	2016-06-17 12:00		1	
DZ	FZ	FZ12F	NDZ	FZ12F_IDX2	2016/5/13 15	NORMAL	3600		2016-06-13 22:00	2016-06-22 15:00			
DZ	FZ	FZ10AF	NDZ	FZ10AF_PK	2016/5/13 15	NORMAL	12672	INDEX SAMPLE FAST	2016-06-14 11:37	2016-06-14 13:00		2	
DZ	FZ	FZ10AF	NDZ	FZ10AF_PK	2016/5/13 15	NORMAL	12672	INDEX SKIP SCAN	2016-06-15 11:42	2016-06-17 22:00		102	
DZ	FZ	FZ10AF	NDZ	FZ10AF_PK	2016/5/13 15	NORMAL	12672	INDEX SKIP SCAN D	2016-06-15 11:22	2016-06-17 22:00		88	
DZ	FZ	FZ10AF	NDZ	FZ10AF_PK	2016/5/13 15	NORMAL	12672	INDEX UNIQUE SCAL	2016-06-15 11:42	2016-06-17 22:00		85	

如上图所示，有一个 3.6G 大的索引在 13 号到 22 号从没使用过，接下来，我们可以继续查询该索引是否联合索引，创建是否合理，分析为何不走该索引，从而判断是否可以删除索引。

另外下边的 SQL 可以查询出表上列的使用情况：

```
CREATE OR REPLACE VIEW VW_COLUMN_USAGE_LHR AS
SELECT oo.name          owner,
       o.name           table_name,
       c.name           column_name,
       u.equality_preds,
       u.equijoin_preds,
       u.nonequijoin_preds,
       u.range_preds,
       u.like_preds,
       u.null_preds,
       u.timestamp
FROM sys.col_usage$ u, sys.obj$ o, sys.user$ oo, sys.col$ c
WHERE o.obj# = u.obj#
      AND oo.user# = o.owner#
      AND c.obj# = u.obj#
      AND c.col# = u.intcol#
;
```

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>) 和博客园(<http://www.cnblogs.com/lhrbest>)有同步更新

本文地址 : <http://blog.itpub.net/26736162/viewspace-2120752/>

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 2016-04-06 10:00~ 2016-04-11 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】

