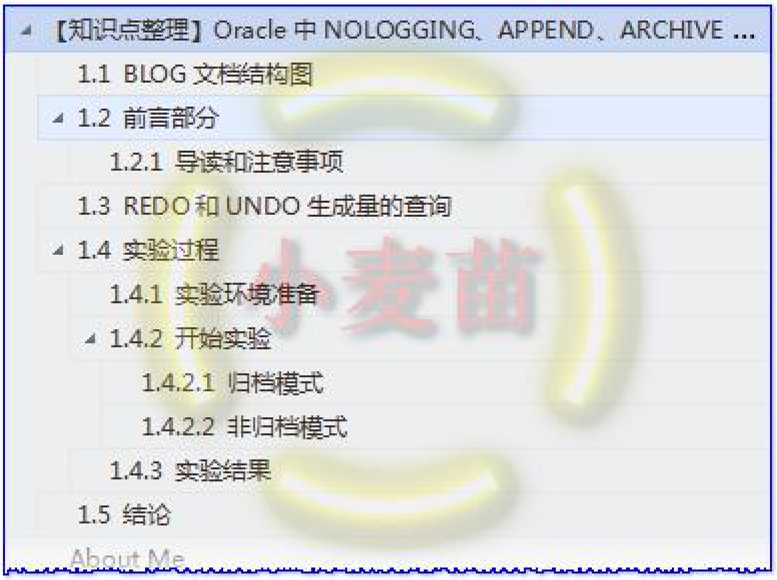
【知识点整理】Oracle 中 NOLOGGING、APPEND、ARCHIVE 和 PARALLEL 下，REDO、UNDO 和执行速度的比较

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 系统和会话级别的 REDO 和 UNDO 量的查询

② NOLOGGING、APPEND、ARCHIVE 和 PARALLEL 下，REDO、UNDO 和执行速度的比较（重点）

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 若网页文章代码格式有错乱，推荐使用 360 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，也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_XLHRD_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 REDO 和 UNDO 生成量的查询

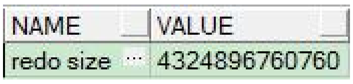
说明：反映 UNDO、REDO 占用量的统计指标是：

UNDO:undo change vector size

REDO:redo size

1、查看全局数据库 REDO 生成量，可以通过 V$SYSSTAT 视图查询

```
SELECT NAME,
       VALUE
FROM   V$SYSSTAT
WHERE  NAME = 'redo size';
```



NAME	VALUE
redo size ···	4324896760760

2、查看当前会话的 REDO 生成量，可以通过 V$MYSTAT 或 V$SESSTAT 视图查询

```
create or replace view redo_size as
SELECT VALUE
FROM   v$mystat   my,
       v$statname  st
```

```
WHERE  my.statistic# =st.STATISTIC#
AND    st.name = 'redo size';
```

----下边的实验将用到这个视图

```
CREATE OR REPLACE VIEW VW_REDO_UNDO_LHR AS
SELECT (SELECT NB.VALUE
         FROM V$MYSTAT NB, V$STATNAME ST
        WHERE NB.STATISTIC# = ST.STATISTIC#
          AND ST.NAME = 'redo size') REDO,
       (SELECT NB.VALUE
         FROM V$MYSTAT NB, V$STATNAME ST
        WHERE NB.STATISTIC# = ST.STATISTIC#
          AND ST.NAME = 'undo change vector size') UNDO
  FROM DUAL;
```

或：

```
CREATE OR REPLACE VIEW VW_REDO_UNDO_LHR AS
SELECT  (SELECT NB.VALUE
          FROM v$sesstat NB, V$STATNAME ST
         WHERE NB.STATISTIC# = ST.STATISTIC#
           AND ST.NAME = 'redo size'
                    AND NB.SID=USERENV('SID')) REDO,
       (SELECT NB.VALUE
          FROM v$sesstat NB, V$STATNAME ST
         WHERE NB.STATISTIC# = ST.STATISTIC#
           AND ST.NAME = 'undo change vector size'
                    AND NB.SID=USERENV('SID')) UNDO
  FROM DUAL;
```

1.4 实验过程

1.4.1 实验环境准备

```
--记录 REDO 和 UNDO 量的视图
CREATE OR REPLACE VIEW VW_REDO_UNDO_LHR AS
SELECT (SELECT NB.VALUE
         FROM V$MYSTAT NB, V$STATNAME ST
        WHERE NB.STATISTIC# = ST.STATISTIC#
          AND ST.NAME = 'redo size') REDO,
       (SELECT NB.VALUE
         FROM V$MYSTAT NB, V$STATNAME ST
        WHERE NB.STATISTIC# = ST.STATISTIC#
          AND ST.NAME = 'undo change vector size') UNDO
  FROM DUAL;
```

```
--准备中间表，T_A 为 500W，T_B 为 500W 的数据量，T_A 表删掉少量数据
DROP TABLE  T_A PURGE;
DROP TABLE  T_B PURGE;
CREATE TABLE T_A AS SELECT * FROM DBA_OBJECTS;
CREATE TABLE T_B AS SELECT * FROM DBA_OBJECTS;

INSERT INTO T_A SELECT * FROM T_A;
INSERT INTO T_A SELECT * FROM T_A;
INSERT INTO T_A SELECT * FROM T_A;
INSERT INTO T_A SELECT * FROM T_A;
INSERT INTO T_A SELECT * FROM T_A;
INSERT INTO T_A SELECT * FROM T_A;
COMMIT;
INSERT INTO T_B SELECT * FROM T_A;
DELETE FROM T_A WHERE OBJECT_ID>=90000;
COMMIT;

SELECT COUNT(1) FROM T_A;     --5548800
SELECT COUNT(1) FROM T_B;     --5668976
```

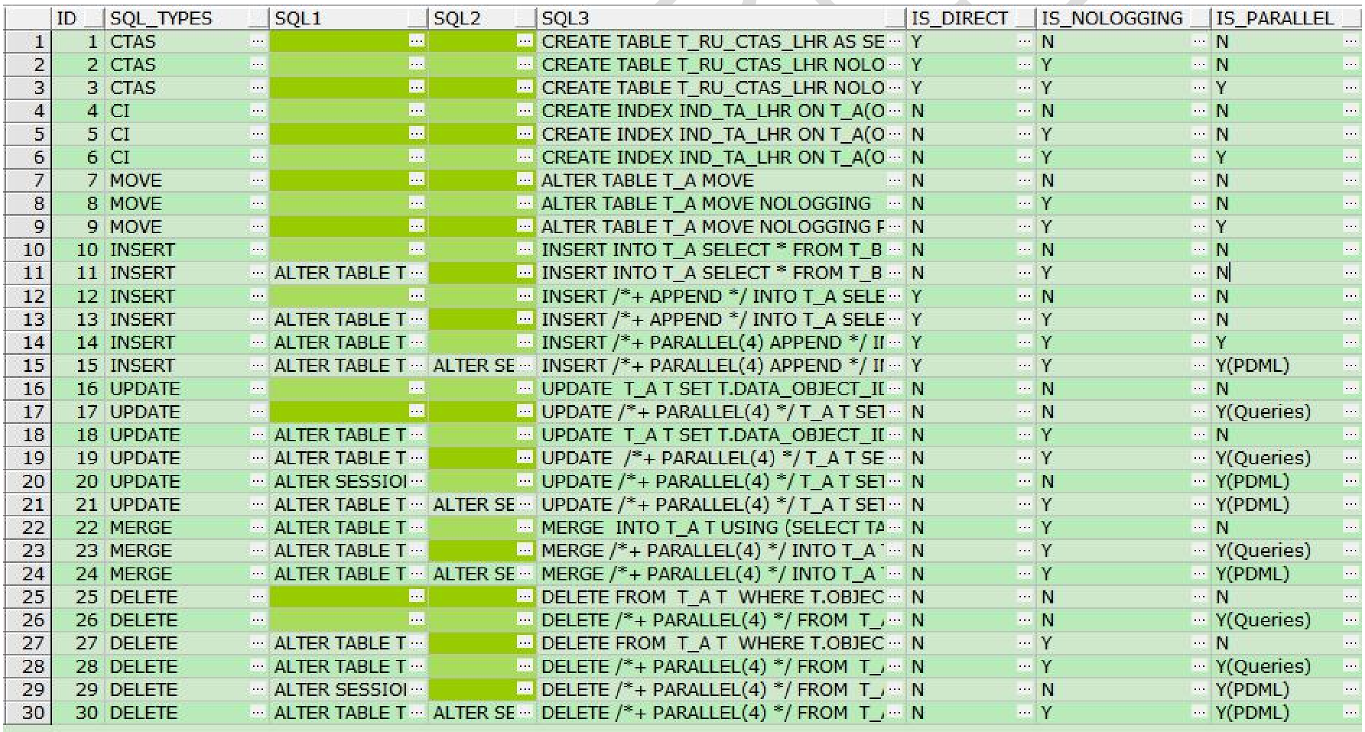
```
--记录测试结果
DROP TABLE T_RU_160929_LHR;
CREATE TABLE T_RU_160929_LHR (
   ID NUMBER PRIMARY KEY,
        SQL_TYPES VARCHAR2(255),
        SQL1 VARCHAR2(255),
        SQL2 VARCHAR2(255),
        SQL3 VARCHAR2(4000),
        IS_DIRECT VARCHAR2(20),
        IS_NOLOGGING VARCHAR2(20),
        IS_PARALLEL VARCHAR2(20),
        ARCH_REDO NUMBER,
        ARCH_UNDO NUMBER,
        NOARCH_REDO NUMBER,
        NOARCH_UNDO NUMBER,
        ARCH_USE_TIME NUMBER,
        NOARCH_USE_TIME NUMBER,
   SQL_EXPLAIN CLOB,
   COMMENTS VARCHAR2(255)
);

--插入要执行的 SQL 语句

INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(1, 'CTAS', NULL, NULL, 'CREATE TABLE T_RU_CTAS_LHR AS SELECT * FROM T_B', 'Y', 'N', 'N');
INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(2, 'CTAS', NULL, NULL, 'CREATE TABLE T_RU_CTAS_LHR NOLOGGING AS SELECT * FROM T_B', 'Y', 'Y', 'N');
INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(3, 'CTAS', NULL, NULL, 'CREATE TABLE T_RU_CTAS_LHR NOLOGGING PARALLEL 4 AS SELECT * FROM T_B', 'Y', 'Y',
'Y');

INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(4, 'CI', NULL, NULL, 'CREATE INDEX IND_TA_LHR ON T_A(OBJECT_ID)', 'N', 'N', 'N');
INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(5, 'CI', NULL, NULL, 'CREATE INDEX IND_TA_LHR ON T_A(OBJECT_ID) NOLOGGING', 'N', 'Y', 'N');
INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(6, 'CI', NULL, NULL, 'CREATE INDEX IND_TA_LHR ON T_A(OBJECT_ID) NOLOGGING PARALLEL 4', 'N', 'Y', 'Y');

INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(7, 'MOVE', NULL, NULL, 'ALTER TABLE T_A MOVE', 'N', 'N', 'N');
INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(8, 'MOVE', NULL, NULL, 'ALTER TABLE T_A MOVE NOLOGGING', 'N', 'Y', 'N');
INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
```

```
(9, 'MOVE', NULL, NULL, 'ALTER TABLE T_A MOVE NOLOGGING PARALLEL 4', 'N', 'Y', 'Y');

INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(10, 'INSERT', NULL, NULL, 'INSERT INTO T_A SELECT * FROM T_B', 'N', 'N', 'N');
INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(11, 'INSERT', 'ALTER TABLE T_A NOLOGGING', NULL, 'INSERT INTO T_A SELECT * FROM T_B', 'N', 'Y', 'N');
INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(12, 'INSERT', NULL, NULL, 'INSERT /*+ APPEND */ INTO T_A SELECT * FROM T_B', 'Y', 'N', 'N');
INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(13, 'INSERT', 'ALTER TABLE T_A NOLOGGING', NULL, 'INSERT /*+ APPEND */ INTO T_A SELECT * FROM T_B', 'Y',
'Y', 'N');
INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(14, 'INSERT', 'ALTER TABLE T_A NOLOGGING', NULL, 'INSERT /*+ PARALLEL(4) APPEND */ INTO T_A SELECT * FROM
T_B', 'Y', 'Y', 'Y');
INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(15, 'INSERT', 'ALTER TABLE T_A NOLOGGING', 'ALTER SESSION ENABLE PARALLEL DML', 'INSERT /*+ PARALLEL(4)
APPEND */ INTO T_A SELECT * FROM T_B', 'Y', 'Y', 'Y(PDML)');

INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(16, 'UPDATE', NULL, NULL, 'UPDATE  T_A T SET T.DATA_OBJECT_ID =(SELECT TB.DATA_OBJECT_ID FROM T_B TB WHERE
TB.OBJECT_ID = T.OBJECT_ID AND ROWNUM=1) WHERE T.OBJECT_ID <= 1000', 'N', 'N', 'N');
INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(17, 'UPDATE', NULL, NULL, 'UPDATE /*+ PARALLEL(4) */ T_A T SET T.DATA_OBJECT_ID =(SELECT TB.DATA_OBJECT_ID
FROM T_B TB WHERE TB.OBJECT_ID = T.OBJECT_ID AND ROWNUM=1) WHERE T.OBJECT_ID <= 1000', 'N', 'N', 'Y(Queries)');
INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(18, 'UPDATE', 'ALTER TABLE T_A NOLOGGING', NULL, 'UPDATE  T_A T SET T.DATA_OBJECT_ID =(SELECT
TB.DATA_OBJECT_ID FROM T_B TB WHERE TB.OBJECT_ID = T.OBJECT_ID AND ROWNUM=1) WHERE T.OBJECT_ID <= 1000',
'N', 'Y', 'N');
INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(19, 'UPDATE', 'ALTER TABLE T_A NOLOGGING', NULL, 'UPDATE  /*+ PARALLEL(4) */ T_A T SET T.DATA_OBJECT_ID
=(SELECT TB.DATA_OBJECT_ID FROM T_B TB WHERE TB.OBJECT_ID = T.OBJECT_ID AND ROWNUM=1) WHERE T.OBJECT_ID <=
1000', 'N', 'Y', 'Y(Queries)');
INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(20, 'UPDATE', 'ALTER SESSION ENABLE PARALLEL DML', NULL, 'UPDATE /*+ PARALLEL(4) */ T_A T SET T.DATA_OBJECT_ID
=(SELECT TB.DATA_OBJECT_ID FROM T_B TB WHERE TB.OBJECT_ID = T.OBJECT_ID AND ROWNUM=1) WHERE T.OBJECT_ID <=
1000', 'N', 'N', 'Y(PDML)');
INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(21, 'UPDATE', 'ALTER TABLE T_A NOLOGGING', 'ALTER SESSION ENABLE PARALLEL DML', 'UPDATE /*+ PARALLEL(4)
*/ T_A T SET T.DATA_OBJECT_ID =(SELECT TB.DATA_OBJECT_ID FROM T_B TB WHERE TB.OBJECT_ID = T.OBJECT_ID AND
ROWNUM=1) WHERE T.OBJECT_ID <= 1000', 'N', 'Y', 'Y(PDML)');

INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(22, 'MERGE', 'ALTER TABLE T_A NOLOGGING', NULL, 'MERGE  INTO T_A T USING (SELECT TA.ROWID ROWIDS,
MAX(TB.DATA_OBJECT_ID) DATA_OBJECT_ID FROM T_B TB, T_A TA WHERE TB.OBJECT_ID = TA.OBJECT_ID AND TA.OBJECT_ID
<= 1000  GROUP BY TA.ROWID) T1 ON (T.ROWID = T1.ROWIDS)WHEN MATCHED THEN UPDATE SET T.DATA_OBJECT_ID =
T1.DATA_OBJECT_ID', 'N', 'Y', 'N');
INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(23, 'MERGE', 'ALTER TABLE T_A NOLOGGING', NULL, 'MERGE /*+ PARALLEL(4) */ INTO T_A T USING (SELECT TA.ROWID
ROWIDS, MAX(TB.DATA_OBJECT_ID) DATA_OBJECT_ID FROM T_B TB, T_A TA WHERE TB.OBJECT_ID = TA.OBJECT_ID AND
TA.OBJECT_ID <= 1000  GROUP BY TA.ROWID) T1 ON (T.ROWID = T1.ROWIDS)WHEN MATCHED THEN UPDATE SET
T.DATA_OBJECT_ID = T1.DATA_OBJECT_ID', 'N', 'Y', 'Y(Queries)');
INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(24, 'MERGE', 'ALTER TABLE T_A NOLOGGING', 'ALTER SESSION ENABLE PARALLEL DML', 'MERGE /*+ PARALLEL(4) */
INTO T_A T USING (SELECT TA.ROWID ROWIDS, MAX(TB.DATA_OBJECT_ID) DATA_OBJECT_ID FROM T_B TB, T_A TA WHERE
TB.OBJECT_ID = TA.OBJECT_ID AND TA.OBJECT_ID <= 1000  GROUP BY TA.ROWID) T1 ON (T.ROWID = T1.ROWIDS)WHEN
MATCHED THEN UPDATE SET T.DATA_OBJECT_ID = T1.DATA_OBJECT_ID', 'N', 'Y', 'Y(PDML)');

INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(25, 'DELETE', NULL, NULL, 'DELETE FROM  T_A T  WHERE T.OBJECT_ID IN  ( SELECT TB.OBJECT_ID FROM T_B TB) AND
T.OBJECT_ID <= 1000', 'N', 'N', 'N');
INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(26, 'DELETE', NULL, NULL, 'DELETE /*+ PARALLEL(4) */ FROM  T_A T  WHERE T.OBJECT_ID IN  ( SELECT TB.OBJECT_ID
FROM T_B TB) AND T.OBJECT_ID <= 1000', 'N', 'N', 'Y(Queries)');
INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
```

```
(27, 'DELETE', 'ALTER TABLE T_A NOLOGGING', NULL, 'DELETE FROM  T_A T  WHERE T.OBJECT_ID IN  ( SELECT
TB.OBJECT_ID FROM T_B TB) AND T.OBJECT_ID <= 1000', 'N', 'Y', 'N');
INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(28, 'DELETE', 'ALTER TABLE T_A NOLOGGING', NULL, 'DELETE /*+ PARALLEL(4) */ FROM  T_A T  WHERE T.OBJECT_ID
IN  ( SELECT TB.OBJECT_ID FROM T_B TB) AND T.OBJECT_ID <= 1000', 'N', 'Y', 'Y(Queries)');
INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(29, 'DELETE', 'ALTER SESSION ENABLE PARALLEL DML', NULL, 'DELETE /*+ PARALLEL(4) */ FROM  T_A T  WHERE
T.OBJECT_ID IN  ( SELECT TB.OBJECT_ID FROM T_B TB) AND T.OBJECT_ID <= 1000', 'N', 'N', 'Y(PDML)');
INSERT INTO T_RU_160929_LHR (ID, SQL_TYPES, SQL1, SQL2, SQL3, IS_DIRECT, IS_NOLOGGING, IS_PARALLEL) VALUES
(30, 'DELETE', 'ALTER TABLE T_A NOLOGGING', 'ALTER SESSION ENABLE PARALLEL DML', 'DELETE /*+ PARALLEL(4)
*/ FROM  T_A T  WHERE T.OBJECT_ID IN  ( SELECT TB.OBJECT_ID FROM T_B TB) AND T.OBJECT_ID <= 1000', 'N', 'Y',
'Y(PDML)');
COMMIT;
```

插入完成后查询结果：

```
SELECT ID,
       SQL_TYPES,
       SQL1,
       SQL2,
       SQL3,
       IS_DIRECT,
       IS_NOLOGGING,
       IS_PARALLEL
  FROM T_RU_160929_LHR D
 ORDER BY D.ID;
```



	ID	SQL_TYPES	SQL1	SQL2	SQL3	IS_DIRECT	IS_NOLOGGING	IS_PARALLEL
1	1	CTAS			CREATE TABLE T_RU_CTAS_LHR AS SE	Y	N	N
2	2	CTAS			CREATE TABLE T_RU_CTAS_LHR NOLO	Y	Y	N
3	3	CTAS			CREATE TABLE T_RU_CTAS_LHR NOLO	Y	Y	Y
4	4	CI			CREATE INDEX IND_TA_LHR ON T_A(O	N	N	N
5	5	CI			CREATE INDEX IND_TA_LHR ON T_A(O	N	Y	N
6	6	CI			CREATE INDEX IND_TA_LHR ON T_A(O	N	Y	Y
7	7	MOVE			ALTER TABLE T_A MOVE	N	N	N
8	8	MOVE			ALTER TABLE T_A MOVE NOLOGGING	N	Y	N
9	9	MOVE			ALTER TABLE T_A MOVE NOLOGGING F	N	Y	Y
10	10	INSERT			INSERT INTO T_A SELECT * FROM T_B	N	N	N
11	11	INSERT	ALTER TABLE T		INSERT INTO T_A SELECT * FROM T_B	N	Y	N
12	12	INSERT			INSERT /*+ APPEND */ INTO T_A SELE	Y	N	N
13	13	INSERT	ALTER TABLE T		INSERT /*+ APPEND */ INTO T_A SELE	Y	Y	N
14	14	INSERT	ALTER TABLE T		INSERT /*+ PARALLEL(4) APPEND */ II	Y	Y	Y
15	15	INSERT	ALTER TABLE T	ALTER SE	INSERT /*+ PARALLEL(4) APPEND */ II	Y	Y	Y(PDML)
16	16	UPDATE			UPDATE T_A T SET T.DATA_OBJECT_II	N	N	N
17	17	UPDATE			UPDATE /*+ PARALLEL(4) */ T_A T SET	N	N	Y(Queries)
18	18	UPDATE	ALTER TABLE T		UPDATE T_A T SET T.DATA_OBJECT_II	N	Y	N
19	19	UPDATE	ALTER TABLE T		UPDATE /*+ PARALLEL(4) */ T_A T SE	N	Y	Y(Queries)
20	20	UPDATE	ALTER SESSIOI		UPDATE /*+ PARALLEL(4) */ T_A T SET	N	N	Y(PDML)
21	21	UPDATE	ALTER TABLE T	ALTER SE	UPDATE /*+ PARALLEL(4) */ T_A T SET	N	Y	Y(PDML)
22	22	MERGE	ALTER TABLE T		MERGE INTO T_A T USING (SELECT TA	N	Y	N
23	23	MERGE	ALTER TABLE T		MERGE /*+ PARALLEL(4) */ INTO T_A	N	Y	Y(Queries)
24	24	MERGE	ALTER TABLE T	ALTER SE	MERGE /*+ PARALLEL(4) */ INTO T_A	N	Y	Y(PDML)
25	25	DELETE			DELETE FROM T_A T WHERE T.OBJEC	N	N	N
26	26	DELETE			DELETE /*+ PARALLEL(4) */ FROM T_	N	N	Y(Queries)
27	27	DELETE	ALTER TABLE T		DELETE FROM T_A T WHERE T.OBJEC	N	Y	N
28	28	DELETE	ALTER TABLE T		DELETE /*+ PARALLEL(4) */ FROM T_	N	Y	Y(Queries)
29	29	DELETE	ALTER SESSIOI		DELETE /*+ PARALLEL(4) */ FROM T_	N	N	Y(PDML)
30	30	DELETE	ALTER TABLE T	ALTER SE	DELETE /*+ PARALLEL(4) */ FROM T_	N	Y	Y(PDML)

下边的存过可以测试 REDO 和 UNDO 的量，至于该存过的算法大家自己看吧。

```
--创建存储过程，用来测试 REDO 量
CREATE OR REPLACE PROCEDURE PRO_TEST_RU_LHR AS

  V_REDO       NUMBER := 0;
  V_UNDO       NUMBER := 0;
  V_REDO1      NUMBER := 0;
  V_UNDO1      NUMBER := 0;
  V_ARCH       VARCHAR2(30);
  V_START_TIME NUMBER := 0;
  V_END_TIME   NUMBER := 0;
```

```
BEGIN

  SELECT D.LOG_MODE INTO V_ARCH FROM V$DATABASE D;

  FOR CUR IN (SELECT D.ID, D.SQL1, D.SQL2, D.SQL3
               FROM T_RU_160929_LHR D
              ORDER BY D.ID) LOOP

    BEGIN
      EXECUTE IMMEDIATE CUR.SQL1;
    EXCEPTION
      WHEN OTHERS THEN
        NULL;
    END;
    BEGIN
      EXECUTE IMMEDIATE CUR.SQL2;
    EXCEPTION
      WHEN OTHERS THEN
        NULL;
    END;
    SELECT DBMS_UTILITY.GET_TIME INTO V_START_TIME FROM DUAL;
    SELECT V.REDO, V.UNDO INTO V_REDO, V_UNDO FROM VW_REDO_UNDO_LHR V;
    EXECUTE IMMEDIATE CUR.SQL3;
    SELECT V.REDO, V.UNDO INTO V_REDO1, V_UNDO1 FROM VW_REDO_UNDO_LHR V;
    SELECT DBMS_UTILITY.GET_TIME INTO V_END_TIME FROM DUAL;
    ROLLBACK;
    IF V_ARCH = 'ARCHIVELOG' THEN
      UPDATE T_RU_160929_LHR T
         SET T.ARCH_REDO     = V_REDO1 - V_REDO,
             T.ARCH_UNDO     = V_UNDO1 - V_UNDO,
             T.ARCH_USE_TIME =
             (V_END_TIME - V_START_TIME) / 100,
             T.COMMENTS      = T.COMMENTS || 'ARCHIVELOG:' ||
                               (SELECT COUNT(1) FROM T_A) || '  '
       WHERE T.ID = CUR.ID;

    ELSE
      UPDATE T_RU_160929_LHR T
         SET T.NOARCH_REDO     = V_REDO1 - V_REDO,
             T.NOARCH_UNDO     = V_UNDO1 - V_UNDO,
             T.NOARCH_USE_TIME =
             (V_END_TIME - V_START_TIME) / 100,
             T.COMMENTS        = T.COMMENTS || 'NOARCHIVELOG:' ||
                                 (SELECT COUNT(1) FROM T_A) || '  '
       WHERE T.ID = CUR.ID;

    END IF;
    COMMIT;
    EXECUTE IMMEDIATE 'ALTER TABLE T_A LOGGING';
    EXECUTE IMMEDIATE 'ALTER SESSION DISABLE PARALLEL DML';
    EXECUTE IMMEDIATE 'ALTER SYSTEM FLUSH BUFFER_CACHE';
    BEGIN
      EXECUTE IMMEDIATE 'DROP INDEX IND_TA_LHR';
    EXCEPTION
      WHEN OTHERS THEN
        NULL;
    END;
    BEGIN
      EXECUTE IMMEDIATE 'DROP TABLE T_RU_CTAS_LHR PURGE';
    EXCEPTION
      WHEN OTHERS THEN
        NULL;
    END;
  END LOOP;

END;
```

1.4.2 开始实验

1.4.2.1 归档模式

增加日志组的个数，避免因为日志切换导致的等待。

```
SYS@lhrdb> 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@lhrdb> select GROUP#,BYTES,STATUS from v$log;

    GROUP#      BYTES STATUS
---------- ---------- ----------------
         1  104857600 ACTIVE
         2  104857600 ACTIVE
         3  104857600 ACTIVE
         4  104857600 CURRENT
         5  104857600 ACTIVE
         6  104857600 ACTIVE

6 rows selected.

SYS@lhrdb> archive log list;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence     401
Next log sequence to archive   406
Current log sequence           406
SYS@lhrdb> SET TIMING ON
SYS@lhrdb> exec PRO_TEST_RU_LHR;

PL/SQL procedure successfully completed.

Elapsed: 00:12:49.83
SYS@lhrdb>
```

在 PL/SQL DEVELOPER 中查询结果：

```
SELECT D.*
  FROM T_RU_160929_LHR D
 ORDER BY D.ID;
```

1.4.2.2 非归档模式

```
SYS@lhrdb> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SYS@lhrdb> startup mount
ORACLE instance started.

Total System Global Area 1720328192 bytes
Fixed Size               2247072 bytes
Variable Size           486540896 bytes
Database Buffers        1224736768 bytes
Redo Buffers             6803456 bytes
Database mounted.

SYS@lhrdb> alter database noarchivelog;

Database altered.

SYS@lhrdb> alter database open;

Database altered.

SYS@lhrdb> archive log list;
Database log mode            No Archive Mode
Automatic archival           Disabled
Archive destination          USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence    419
Current log sequence         424
SYS@lhrdb>


SYS@lhrdb> set timing on
SYS@lhrdb> exec PRO_TEST_RU_LHR;

PL/SQL procedure successfully completed.

Elapsed: 00:13:31.67
```

在 PL/SQL DEVELOPER 中查询结果：

```
SELECT D.*
  FROM T_RU_160929_LHR D
 ORDER BY D.ID;
```

以上测试过程，可以多做几次，然后取其平均值，多次测试前将结果表清空：

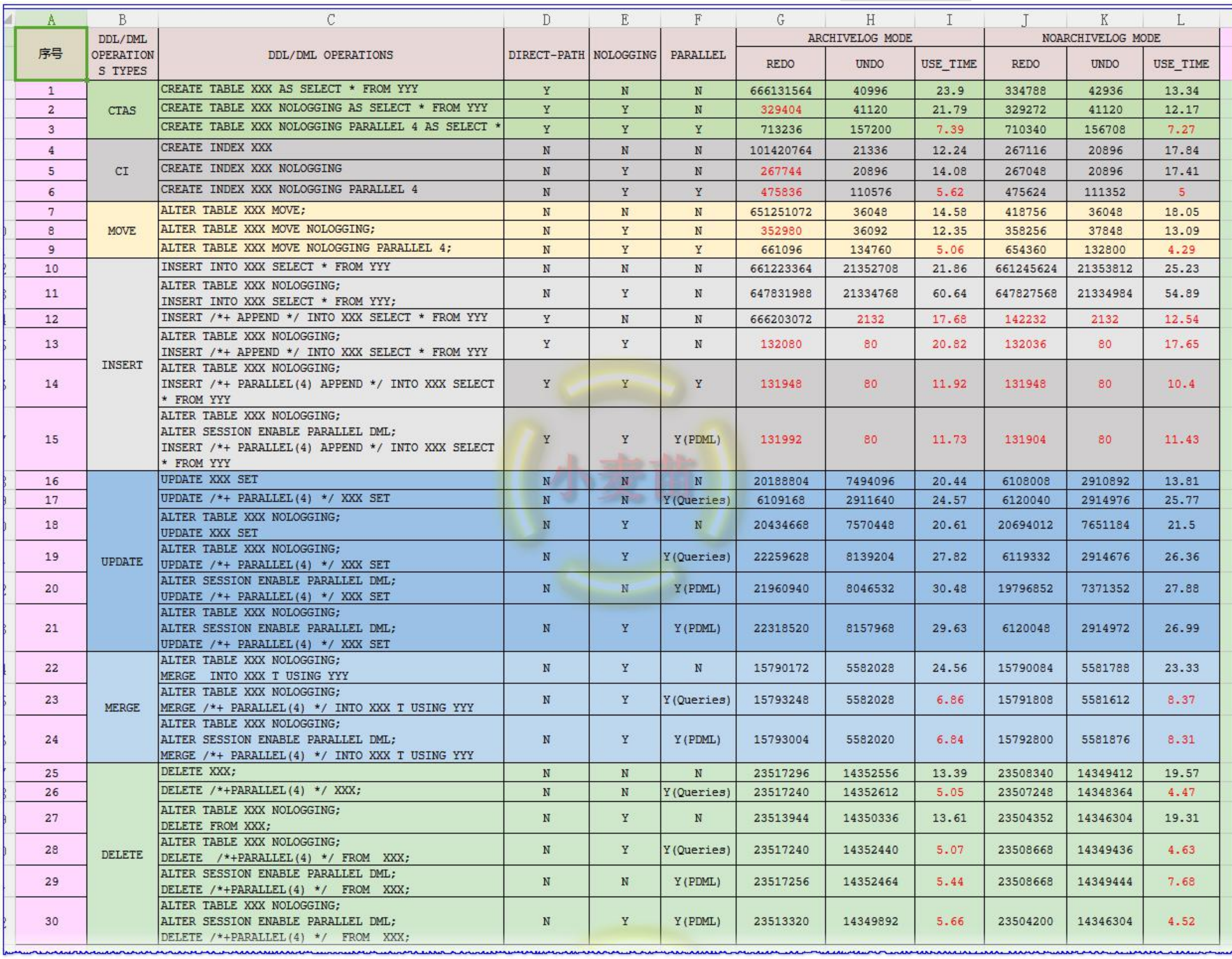
```
UPDATE T_RU_160929_LHR T
   SET T.ARCH_REDO      = '',
       T.ARCH_UNDO      = '',
       T.ARCH_USE_TIME  = '',
       T.NOARCH_REDO    = '',
       T.NOARCH_UNDO    = '',
       T.NOARCH_USE_TIME = '',
       T.COMMENTS       = '';
```

```
    COMMIT;
```

1.4.3 实验结果

NOLOGGING、APPEND、ARCHIVE和PARALLEL下，REDO、UNDO和执行速度的比较_实验结果_LHR.zip

根据以上的实验可以得到一些结论：关于表日志模式(LOGGING/NOLOGGING)、插入模式(APPEND/NOAPPEND)、数据库运行模式(归档/非归档)和并行模式下，REDO、UNDO 和执行速度的情况大约如下表所示：



A	B	C	D	E	F	G	H	I	J	K	L
序号	DDL/DML OPERATIONS TYPES	DDL/DML OPERATIONS	DIRECT-PATH	NOLOGGING	PARALLEL	ARCHIVELOG MODE			NOARCHIVELOG MODE		
						REDO	UNDO	USE_TIME	REDO	UNDO	USE_TIME
1	CTAS	CREATE TABLE XXX AS SELECT * FROM YYY	Y	N	N	666131564	40996	23.9	334788	42936	13.34
2		CREATE TABLE XXX NOLOGGING AS SELECT * FROM YYY	Y	Y	N	329404	41120	21.79	329272	41120	12.17
3		CREATE TABLE XXX NOLOGGING PARALLEL 4 AS SELECT *	Y	Y	Y	713236	157200	7.39	710340	156708	7.27
4	CI	CREATE INDEX XXX	N	N	N	101420764	21336	12.24	267116	20896	17.84
5		CREATE INDEX XXX NOLOGGING	N	Y	N	267744	20896	14.08	267048	20896	17.41
6		CREATE INDEX XXX NOLOGGING PARALLEL 4	N	Y	Y	475836	110576	5.62	475624	111352	5
7	MOVE	ALTER TABLE XXX MOVE;	N	N	N	651251072	36048	14.58	418756	36048	18.05
8		ALTER TABLE XXX MOVE NOLOGGING;	N	Y	N	352980	36092	12.35	358256	37848	13.09
9		ALTER TABLE XXX MOVE NOLOGGING PARALLEL 4;	N	Y	Y	661096	134760	5.06	654360	132800	4.29
10	INSERT	INSERT INTO XXX SELECT * FROM YYY	N	N	N	661223364	21352708	21.86	661245624	21353812	25.23

11		ALTER TABLE XXX NOLOGGING;
		INSERT INTO XXX SELECT * FROM YYY;
			N	Y	N	647831988	21334768	60.64	647827568	21334984	54.89

12		INSERT /*+ APPEND */ INTO XXX SELECT * FROM YYY	Y	N	N	666203072	2132	17.68	142232	2132	12.54

13		ALTER TABLE XXX NOLOGGING;
		INSERT /*+ APPEND */ INTO XXX SELECT * FROM YYY
			Y	Y	N	132080	80	20.82	132036	80	17.65

14		ALTER TABLE XXX NOLOGGING;
		INSERT /*+ PARALLEL(4) APPEND */ INTO XXX SELECT * FROM YYY
			Y	Y	Y	131948	80	11.92	131948	80	10.4

15		ALTER TABLE XXX NOLOGGING;
		ALTER SESSION ENABLE PARALLEL DML;
		INSERT /*+ PARALLEL(4) APPEND */ INTO XXX SELECT * FROM YYY
			Y	Y	Y(PDML)	131992	80	11.73	131904	80	11.43

16	UPDATE	UPDATE XXX SET	N	N	N	20188804	7494096	20.44	6108008	2910892	13.81
17		UPDATE /*+ PARALLEL(4) */ XXX SET	N	N	Y(Queries)	6109168	2911640	24.57	6120040	2914976	25.77

18		ALTER TABLE XXX NOLOGGING;
		UPDATE XXX SET
			N	Y	N	20434668	7570448	20.61	20694012	7651184	21.5

19		ALTER TABLE XXX NOLOGGING;
		UPDATE /*+ PARALLEL(4) */ XXX SET
			N	Y	Y(Queries)	22259628	8139204	27.82	6119332	2914676	26.36

20		ALTER SESSION ENABLE PARALLEL DML;
		UPDATE /*+ PARALLEL(4) */ XXX SET
			N	N	Y(PDML)	21960940	8046532	30.48	19796852	7371352	27.88

21		ALTER TABLE XXX NOLOGGING;
		ALTER SESSION ENABLE PARALLEL DML;
		UPDATE /*+ PARALLEL(4) */ XXX SET
			N	Y	Y(PDML)	22318520	8157968	29.63	6120048	2914972	26.99

22	MERGE	ALTER TABLE XXX NOLOGGING; MERGE INTO XXX T USING YYY	N	Y	N	15790172	5582028	24.56	15790084	5581788	23.33
23		ALTER TABLE XXX NOLOGGING; MERGE /*+ PARALLEL(4) */ INTO XXX T USING YYY	N	Y	Y(Queries)	15793248	5582028	6.86	15791808	5581612	8.37

24		ALTER TABLE XXX NOLOGGING;
		ALTER SESSION ENABLE PARALLEL DML;
		MERGE /*+ PARALLEL(4) */ INTO XXX T USING YYY
			N	Y	Y(PDML)	15793004	5582020	6.84	15792800	5581876	8.31

25	DELETE	DELETE XXX;	N	N	N	23517296	14352556	13.39	23508340	14349412	19.57
26		DELETE /*+PARALLEL(4) */ XXX;	N	N	Y(Queries)	23517240	14352612	5.05	23507248	14348364	4.47

27		ALTER TABLE XXX NOLOGGING;
		DELETE FROM XXX;
			N	Y	N	23513944	14350336	13.61	23504352	14346304	19.31

28		ALTER TABLE XXX NOLOGGING;
		DELETE /*+PARALLEL(4) */ FROM XXX;
			N	Y	Y(Queries)	23517240	14352440	5.07	23508668	14349436	4.63

29		ALTER SESSION ENABLE PARALLEL DML;
		DELETE /*+PARALLEL(4) */ FROM XXX;
			N	N	Y(PDML)	23517256	14352464	5.44	23508668	14349444	7.68

30		ALTER TABLE XXX NOLOGGING;
		ALTER SESSION ENABLE PARALLEL DML;
		DELETE /*+PARALLEL(4) */ FROM XXX;
			N	Y	Y(PDML)	23513320	14349892	5.66	23504200	14346304	4.52

1.5 结论

（一）关于效率的结论：

1、INSERT INTO：在 APPEND 提示的情况下，NOLOGGING 或 NOARCHIVELOG 满足一个即产生少量的 REDO 和 UNDO;另外 PARALLEL 默认是以 DIRECT 的方式进行加载数据的，一般在并行情况下 SQL 执行速度提高。

2、**CTAS：**CTAS 本身就是一种 DIRECT 的操作，归档模式+NOLOGGING 模式产生少量 REDO；并行模式下时间大幅度减少，但生成的 REDO 和 UNDO 成倍增长。

3、**ALTER TABLE ... MOVE**：ARCHIVELOG+NOLOGGING 模式产生少量 REDO；并行模式下时间大幅度减少，但生成的 REDO 和 UNDO 成倍增长 。

4、**CREATE INDEX**：ARCHIVELOG+NOLOGGING 模式产生少量 REDO；并行模式下时间大幅度减少，但生成的 REDO 和 UNDO 成倍增长。

5、**UPDATE**：任何组合都会生成大量 UNDO、大量 REDO；有关并行的性能需要查询执行计划再做定夺。

6、**DELETE：**任何组合都会生成大量 UNDO、大量 REDO；加上并行可以大幅度提高 SQL 的执行速度。

7、**MERGE:**在关联更新的情况下，MERGE 语句的非关联形式的性能比 UPDATE 要高，若加上并行性能更好。

8、总体而言，非归档比归档模式下性能高

（二）关于属性 NOLOGGING 和并行度的结论：

1、对于形如：CREATE TABLE TT NOLOGGING PARALLEL **4** AS SELECT * FROM DBA_OBJECTS;或 CREATE INDEX IDNX11 ON TT(OBJECT_ID) NOLOGGING PARALLEL **4**;的 SQL 语句而言，创建的表或索引的并行度是 4，日志模式是 NOLOGGING，所以，生产库上对于重要的表和索引需要修改为 LOGGING，并行度可以根据需要来修改，ALTER TABLE TT LOGGING NOPARALLEL;或 ALTER INDEX IDNX11 LOGGING NOPARALLEL;

2、对于形如：ALTER TABLE TT MOVE NOLOGGING PARALLEL **4**;或 ALTER INDEX IDNX11 REBUILD NOLOGGING PARALLEL **4**;的 SQL 语句而言，修改后的表的并行度依然为原来的并行度，但是索引的并行度是 4，而日志模式都是 NOLOGGING，所以，生产库上对于重要的表和索引需要修改为 LOGGING，并行度可以根据需要来修改，ALTER TABLE TT LOGGING NOPARALLEL;或 ALTER INDEX IDNX11 LOGGING NOPARALLEL;

总之一句话，若执行了上边形式的 SQL 语句后，最好都修改一下表或索引的并行度及其日志模式。

（三）APPEND 使用注意事项：

1、建议不要经常使用 APPEND，这样表空间会一直在高水位上，除非你这个表只插不删。

2、以 APPEND 方式插入记录后，要执行 COMMIT，才能对表进行查询。否则会出现错误：ORA-12838: 无法在并行模式下修改之后读/修改对象。

3、APPEND 对 INSERT INTO ... VALUES 语句不起作用，需要使用 11gR2 的 APPEND_VALUES 来提示才可以直接路径加载，注意：APPEND_VALUES 对 INSERT INTO ... SELECT 也起作用。

4、APPEND 使用 HWM 之上的块，减少了搜索 FREELIST 上块的时间。

5、在归档模式下：NOLOGGING+APPEND 才会显著减少 REDO 数量；在非归档模式下：单独 APPEND 即可减少 REDO 数量。

6、APPEND 不会减少相关表的索引上产生的 REDO 数量。

7、APPEND 的插入操作是给表加上 6 级排它锁，会阻塞表上的所有 DML 语句。

8、每提交一次，就会取一个新的 BLOCK 存放，高水位就上推一个 BLOCK，若在 LOOP 循环中，外部循环 100W 次，但是每循环一次只有一行符合条件的数据插入，这样，大量单条/*+APPEND*/插入，就会使得表急剧增大，除对 INSERT 本身造成性能影响之外，对以后的 SELECT、UPDATE、DELETE 更是带来更巨大的性能影响。

（四）NOLOGGING 使用注意事项：

1、NOLOGGING 插完后最好对表做个备份。生产上重要的表不建议设置 NOLOGGING 属性。

2、如果库处在 FORCE LOGGING 模式下，此时的 NOLOGGING 方式是无效的。

（五）PDML 使用注意事项：

1、必须使用 ALTER SESSION ENABLE PARALLEL DML;才可以启动 PDML。

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
- 本文 itpub 地址：http://blog.itpub.net/26736162/viewspace-2125815/
- 本文博客园地址：http://www.cnblogs.com/lhrbest/p/5924743.html
- 本文 pdf 版：http://yunpan.cn/cdEQedhCs2kFz （提取码：ed9b）
- 小麦苗云盘地址：http://blog.itpub.net/26736162/viewspace-1624453/
- QQ 群：230161599　　　微信群：私聊
- 联系我请加 QQ 好友(642808185)，注明添加缘由
- 于 2016-09-27 10:00 ~ 2016-09-30 19:00 在中行完成
- 文章内容来源于小麦苗的学习笔记，部分整理自网络，若有侵权或不当之处还请谅解！
- 【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】

....

手机长按下图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，免费学习