LOGGING OR NOLOGGING THAT IS THE QUESTION

By:

Francisco Munoz Alvarez

dbis™ DATABASE INTEGRATED SOLUTIONS

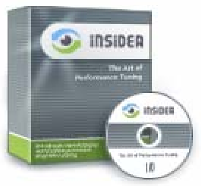
LOGGING OR NOLOGGING : THAT IS THE QUESTION

Francisco Munoz Alvarez

Oracle ACE Director
CLOUG (Chilean Oracle Users Group) President
LAOUC (Latin American Oracle Users Group Council) President
8/9/10g/11g OCP, RAC OCE, AS OCA, E-Business OCP, SQL/PLSQL OCA, Oracle 7 OCM
Oracle 7 & 11GR2 Beta Tester
ITIL Certified

Blog: www.oraclenz.com - Email: franciscoa@dbisonline.com – Twitter : fcomunoz

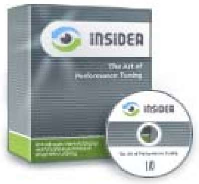
Database Director at DBIS ™
Database Integrated Solutions
www.dbisonline.com
www.dbis.co.nz



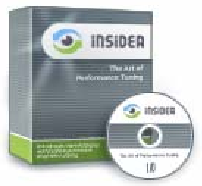
“Redo generation is a vital part of the Oracle recovery mechanism. Without it, an instance will not recover when it crashes and will not start in a consistent state. By other side, excessive redo generation is the result of excessive work on the database.“

Common Questions?

- *Does creating a table with the NOLOGGING option means there is “no generation of redo ever”, or just that the initial creation operation has no redo generation, but that DML down the road generates redo?*



- *How and when can the NOLOGGING option be employed?*

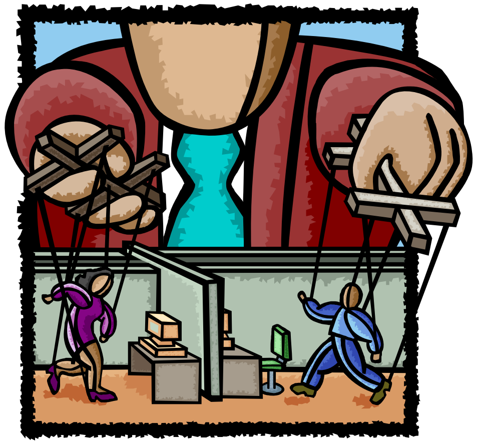


The Rule:

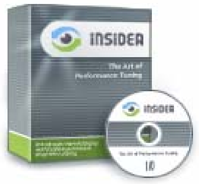
“The most important rule with respect to data is to never put yourself into an unrecoverable situation.”

The importance of this guideline cannot be stressed enough, but it does not mean that you can never use time saving or performance enhancing options.

Topics



- ✓ What is Redo?
- ✓ Redo Generation and Recoverability
- ✓ Redo and Undo
- ✓ Important Points about Logging and NoLogging
- ✓ Disabling Redo Generation (NoLogging)
- ✓ Tips
- ✓ Common Problems
- ✓ How to detect Redo Generation



What is Redo? (Long Answer)

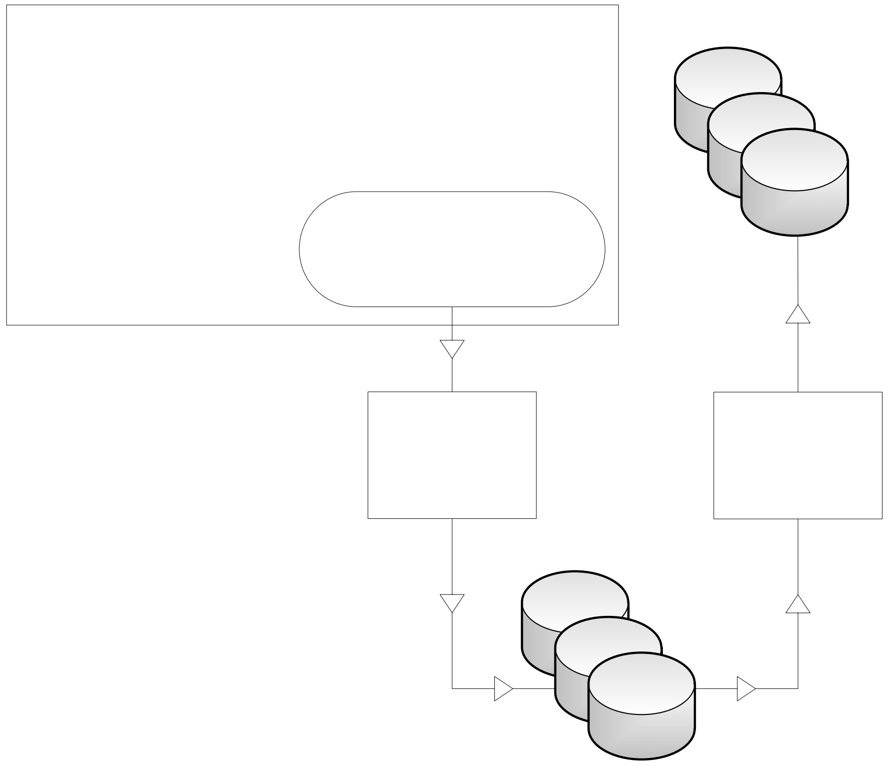
When Oracle blocks are changed, including undo blocks, oracle records the changes in a form of vector changes which are referred to as redo entries or redo records. The changes are written by the server process to the redo log buffer in the SGA. The redo log buffer is then flushed into the online redo logs in near real time fashion by the log writer (LGWR).

Short Answer?

In other words:

Redo = Transactions

How it Works



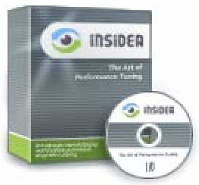
SG

When Redo is flushed?

The redo are flushed from Log Buffer by the LGWR:

- ✓ *When a user issue a commit.*
- ✓ *When the Log Buffer is 1/3 full.*
- ✓ *When the amount of redo entries is 1MB.*
- ✓ *Every three seconds*
- ✓ *When a database checkpoint takes place.*

The redo entries are written before the checkpoint to ensure recoverability.



Redo Generation and Recoverability

“The main purpose of redo generation is to ensure recoverability. “

This is the reason why, Oracle does not give the DBA or the Developer a lot of control over redo generation. If the instance crashes, then all the changes within SGA will be lost. Oracle will then use the redo entries in the online redo files to bring the database to a consistent state.

Some Frequent Questions:

- *Why I have excessive Redo Generation during an Online Backup?*
- *Why Oracle generates redo and undo for DML?*
- *Does temporary tables generate Redo?*
- *Can Redo Generation be Disabled During Materialized View Refresh?*
- *Why my table on NoLogging still Generating Redo?*

Why I have excessive Redo Generation during an Online Backup?

When a tablespace is put in backup mode the redo generation behaviour changes but there is **not excessive** redo being generated, there is additional information logged into the online redo log during a hot backup the first time a block is modified in a tablespace that is in hot backup mode.

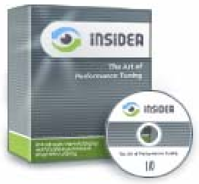
The datafile headers which contain the SCN of the last completed checkpoint are NOT updated while a file is in hot backup mode. DBWR constantly write to the datafiles during the hot backup. The SCN recorded in the header tells us how far back in the redo stream one needs to go to recover that file.

Why Oracle generates redo and undo for DML?

When you issue an insert, update or delete, Oracle actually makes the change to the data blocks that contain the affected data even though you have not issued a commit. To ensure database integrity, Oracle must write information necessary to reverse the change (UNDO) into the log to handle transaction failure or rollback. Recovery from media failure is ensured by writing information necessary to re-play database changes (REDO) into the log. So, UNDO and REDO information logically MUST be written into the transaction log of the RDBMS

To clear this question we have this table:

	UNDO	**REDO**
Record of	How to undo a change	How to reproduce a change
Used for	Rollback, Read-Consistency	Rolling forward DB Changes
Stored in	Undo segments	Redo log files
Protect Against	Inconsistent reads in multiuser systems	Data loss



Does temporary tables generate Redo?

The amount of log generation for temporary tables should be approximately 50% of the log generation for permanent tables.

However, you must consider that an INSERT requires only a small amount of "undo" data, whereas a DELETE requires a small amount of "redo" data.

If you tend to insert data into temporary tables and if you don't delete the data when you're done, the relative log generation rate may be much lower for temporary tables that 50% of the log generation rate for permanent tables.

Can Redo Generation Be Disabled During Materialized View Refresh?

There is no way to turn off redo generation when refreshing materialized views.

Setting the NOLOGGING option during the materialized view creation does not affect this, as the option only applies during the actual creation and not to any subsequent actions on the materialized view.

The amount of redo generated can be reduced by setting ATOMIC_REFRESH=FALSE in the DBMS_MVIEW.REFRESH options.

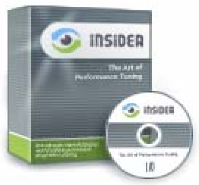
Why my table on NoLogging mode still Generating Redo?

The *NOLOGGING* attribute tells the Oracle that the operation being performed does not need to be recoverable in the event of a failure.

In this case Oracle will generate a minimal number of redo log entries in order to protect the data dictionary, and the operation will probably run faster.

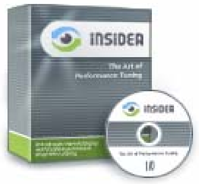
Oracle is relying on the user to recover the data manually in the event of a media failure.

It is important to note that just because an index or a table was created with NOLOGGING does not mean that redo generation has been stopped for this table or index. NOLOGGING is active in the following situations and while running one of the following commands but not after that.



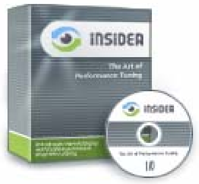
This is a partial list:

- *DIRECT LOAD (SQL*Loader)*
- *DIRECT LOAD INSERT (using APPEND hint)*
- *CREATE TABLE ... AS SELECT*
- *CREATE INDEX*
- *ALTER TABLE MOVE*
- *ALTER TABLE ... MOVE PARTITION*
- *ALTER TABLE ... SPLIT PARTITION*
- *ALTER TABLE ... ADD PARTITION (if HASH partition)*



(continuation...)

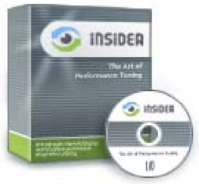
- *ALTER TABLE ... MERGE PARTITION*
- *ALTER TABLE ... MODIFY PARTITION*
    - *ADD SUBPARTITON*
    - *COALESCE SUBPARTITON*
    - *REBUILD UNUSABLE INDEXES*
- *ALTER INDEX ... SPLIT PARTITION*
- *ALTER INDEX ... REBUILD*
- *ALTER INDEX ... REBUILD PARTITION*



Logging is stopped only while one of the commands in the previous slides is running, so if a user runs this:

```
SQL> ALTER INDEX new_index NOLOGGING.
SQL>ALTER INDEX new_index REBUILD;
```

The actual rebuild of the index does not generate redo (all data dictionary changes associated with the rebuild will do) but after that any DML on the index will generate redo this includes direct load insert on the table which the index belongs to.



Here is another example to make this point more clear:

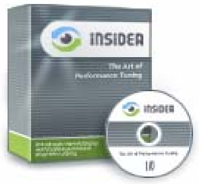
SQL>CREATE TABLE new_table_nolog_test NOLOGGING(....);

All the following statements will generate redo despite the fact the table is in *NOLOGGING* mode:

SQL> INSERT INTO new_table_nolog_test ...,
SQL> UPDATE new_table_nolog_test SET …,
SQL> DELETE FROM new_table_nolog_test ..

The following will not generate redo (except from dictionary changes and indexes):

- INSERT /*+APPEND+/ …
- *ALTER TABLE new_table_nolog_test MOVE …*
- *ALTER TABLE new_table_nolog_test MOVE PARTITION …*



Consider the following example:

```
SQL> select name,value from v$sysstat where name like '%redo size%';

NAME                                                    VALUE
 ------------------------------------------------------ ----------
redo size                                                 27.556.720

SQL> insert into scott.redo1  select * from scott.redotesttab;

50000 rows created.

SQL> select name,value from v$sysstat where name like '%redo size%';

NAME                                                    VALUE
 ------------------------------------------------------ ----------
redo size                                                28.536.820  => 980.100 bytes

SQL> insert /*+ APPEND */ into scott.redo1  select * from scott.redotesttab;

50000 rows created.

SQL> select name,value from v$sysstat where name like '%redo size%';

NAME                                                    VALUE
------------------------------------------------------ ----------
redo size                                                28.539.944 => 3.124 bytes
```

To activate the *NOLOGGING* for one of the *ALTER* commands add the *NOLOGGING* clause after the end of the *ALTER* command.

For example:

SQL> ALTER TABLE new_table_nolog_test NOLOGGING;

The same applies for *CREATE INDEX* but for *CREATE TABLE* the *NOLOGGING* should come after the table name.

Example:

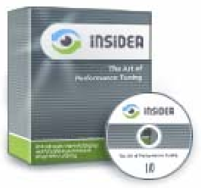
*SQL> CREATE TABLE new_table_nolog_test NOLOGGING AS SELECT * FROM big_table;*

"It is a common mistake to add the NOLOGGING option at the end of the SQL (Because oracle will consider it an alias and the table will generate a lot of logging)."

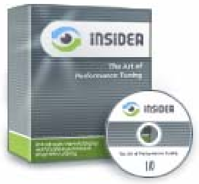
LOGGING and NOLOGGING

Despite the importance of the redo entries, Oracle gave users the ability to limit redo generation on tables and indexes by setting them in *NOLOGGING* mode.

NOLOGGING affect the recoverability. Before going into how to limit the redo generation, it is important to clear the misunderstanding that *NOLOGGING* is the way out of redo generation, this are some points regarding it:



- *NOLOGGING* is designed to handle bulk inserts of data which can be easy re-produced.
- Regardless of *LOGGING* status, writing to undo blocks causes generation of redo.
- *LOGGING* should not be disabled on a primary database if it has one or more standby databases. For this reason oracle introduced the ALTER *DATABASE FORCE LOGGING* command in Oracle 9i R2. (*Means that the NOLOGGING attribute will not have any effect on the segments*) If the database is in *FORCE LOGGING MODE*. *NOLOGGING* can be also override at tablespace level using *ALTER TABLESPACE … FORCE LOGGING.*

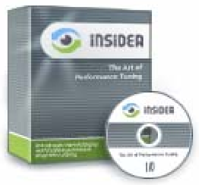


DATABASE INTEGRATED SOLUTIONS

- Any change to the database dictionary will cause redo generation. This will happen to protect the data dictionary.

An example:

if we allocated a space above the HWM for a table, and the system fail in the middle of one *INSERT /*+ APPEND */* , the Oracle will need to rollback that data dictionary update. There will be redo generated but it is to protect the data dictionary, not your newly inserted data.



- The data which are not logged will not be able to recover. The data should be backed up after the modification.
- Tables and indexes should be set back to *LOGGING* mode when the *NOLOGGING* is no longer needed.
- *NOLOGGING* is not needed for Direct Path Insert if the database is in *NO ARCHIVE LOG MODE.*

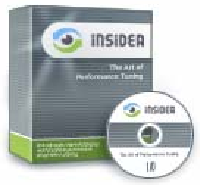
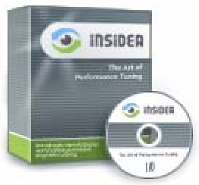
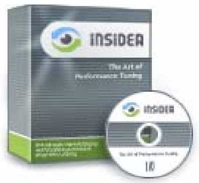


Table Mode	Insert Mode	ArchiveLog Mode	Result
LOGGING	APPEND	ARCHIVE LOG	REDO GENERATED
NOLOGGING	APPEND	ARCHIVE LOG	**NO REDO**
LOGGING	NO APPEND	ARCHIVE LOG	REDO GENERATED
NOLOGGING	NO APPEND	ARCHIVE LOG	REDO GENERATED
LOGGING	APPEND	NO ARCHIVE LOG	**NO REDO**
NOLOGGING	APPEND	NO ARCHIVE LOG	**NO REDO**
LOGGING	NO APPEND	NO ARCHIVE LOG	REDO GENERATED
NOLOGGING	NO APPEND	NO ARCHIVE LOG	REDO GENERATED

- The data which is not able to reproduce should not use the *NOLOGGING* mode. If data which can not be reloaded was loaded using *NOLOGGING.* The data cannot be recovered when the database crashes before backing the data.
- *NOLOGGING* does not apply to *UPDATE*, *DELETE*, and *INSERT*.



- *NOLOGGING* will work during certain situations but subsequent DML will generate redo. Some of these situations are:
  - direct load *INSERT* (using *APPEND* hint),
  - *CREATE TABLE ... AS SELECT,*
  - *CREATE INDEX.*
- If the *LOGGING* or *NOLOGGING* clause is not specified when creating a table, partition, or index the default to the *LOGGING* attribute, will be the *LOGGING* attribute of the tablespace in which it resides.

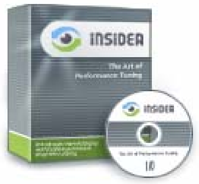


Some Tips and Directions when using **Logging** Mode (DEFAULT)

While Backing Up

RMAN does not need to write the entire block to redo because it knows when the block is being copied. If the user needs to use the user managed backup then they can follow these steps to reduce redo generation:

- Do not back up all the tablespaces in one go. This will put every tablespace in backup mode for longer than it needs to be and therefore generates redo for longer than it needs to do.
- Automatic backup on the busy tablespaces
- Backup a tablespace during a time when it is least busy in terms of DML.



Bulk Inserts

By bulk we mean a large percentage compared to the existing data

To reduce the amount of redo generation in a bulk data load, the user needs to disable the indexes (when making a direct load to a table that have indexes, the indexes will produce redo) before the load then re-build them again as follow:

- *Alter index index_name unusable ; # Do this for every index*
- *Alter session set skip_unusable_indexes=true ; (*)*
- *Insert into table_name select …*
- *Alter index index_name rebuild;*

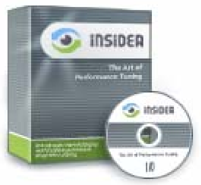
(*)skip_unusable_indexes is an instance initialization parameter in 10g and it default to true. Before 10g, skip_unusable_indexes needs to be set in a session or the user will get an error. It is a good practice to set it in a session, regardless of the database version, when the above steps is done.

Bulk Delete

1. *Create table new_table with logging*
2. *Insert into new_table select the records you want to keep from current_table.*
3. *Create the indexes on the new_table (*)*
4. *Create constraints, grants etc.*
5. *Drop current_table.*
6. *Rename new_table to current.*

(*) If the data left is so small or there are a lot of dependencies on the table (views, procedures, functions, etc) the following steps can be used instead of 3-6 above:

3. *Disable constrains on current_table.*
4. *Truncate current_table;*
5. *Insert into current_table select * from new_table ;*
6. *commit;*
7. *enable constraints*
8. *drop table new_table;*



Bulk Update

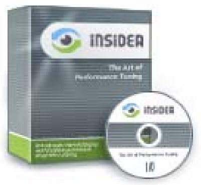
Use this method if indexes are going to be affected by the update. This is because mass updating indexes is more expensive than re-building them.

If a small portion of the data is updated then use this method:

1. *Disable constraints.*
2. *Alter index index_name unusable ;*
3. *Alter session set skip_unusable_indexes=true ;*
4. *Update the table.*
5. *Commit;*
6. *Alter index index_name rebuild ;*
7. *Enable constraints.*

If the update causes a good portion of the data to be updated then follow this method:

1. *Create new_table as select (updating statement)*
2. *Create indexes on the new_table,*
3. *Create grants, constraints etc on the new_table*
4. *Drop current table*
5. *Rename the new_table to current_table.*



Tips For Developers

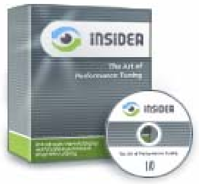
Run the DML in as few SQL statements as you can. This will reduce the generation of undo and block header update and therefore reduces redo generation.

That's how it should be done:

```
SQL> set autotrace on statistics
SQL> insert into test select rownum from dba_objects;

93,244 rows created.

Statistics
---------------------------------------------------------------
...  912,326 redo size
... 93,244   rows processed
```



That’s how it should NOT be done:

```
SQL> set autotrace on statistics
SQL> declare
  2  cursor c1 is
  3  select rownum r from dba_objects;
  4  begin
  5         for v in c1
  6         loop
  7                 insert into test values( v.r) ;
  8         end loop ;
  9  end;
 10  /

PL/SQL procedure successfully completed.

Statistics
----------------------------------------------------------
...  16,112,247 redo size
```

UFFF - 912,326 redo size against 16,112,247 redo size

Do not commit more than you need.

By issuing the commit command you are forcing Oracle to do some internal updates which produces redo.

I ran the PL/SQL code used in the previous example with the command COMMIT; inserted after line 7.

The redo generated was: 28,642,216.

I also ran the script again with the commit at the end followed by a "select * from test" statement to force committed block cleaning the redo generated, and the result was 13,216,188.

You can see that using a lot of committing to insert the same amount of data has produced far more redo. By reducing commits you also will reduce the strain on the LGWR process.

DEMO

“Let see if it’s true”

Some Tips and Directions Now using **NoLogging** Mode

DIRECT PATH INSERT

When direct path insert is used oracle does the following:

- Format the data to be inserted as oracle blocks.
- Insert the blocks above the High Water Mark (HWM)
- When commit takes place the HWM is moved to the new place (The process is done bypassing the buffer cache).

It is very important to understand how Direct Path Inserts affects redo generation. As mentioned above it does not affect indexes but it is affected by the following factors:

- The database Archivelog mode.
- Using the */*+ APPEND */* hint.
- The *LOGGING* mode of the table.
- The *FORCE LOGGING* mode of the database (from 9i R2).

If the database is in *FORCE LOGGING* mode then Oracle will treat the table as if it was in *LOGGING* mode regardless of its mode.

This table will show the relation between *ARCHIVELOG* mode and having the table in LOGGING mode when the */*+ APPEND */* hint is used. This does not include index and data dictionary changes redo generation.

LOGGING MODE	ARCHIVELOG	NOARCHIVELOG
LOGGING	Redo	No Redo
NOLOGGING	No Redo	No Redo

Bulk Inserts

To load bulk data using Direct Path.

- *Alter table table_name nologging;*
- *Alter index index_name unusable ;*
- *Alter session set skip_unusable_indexes=true ;(*)*
- *Insert /*+ APPEND */ into table_name select …*
- *Alter index index_name rebuild nologging;*
- *Alter table table_name logging ;*
- *Alter index index_name logging ;*
- Backup the data.

(*)skip_unusable_indexes is an instance initialization parameter in 10g and defaulted to true. Before 10g, skip_unusable_indexes needs to be set in a session or the user will get an error. It is a good practice to set it in a session, regardless of the database version, when the above is done.

Bulk Delete

1. Create a new_table with no logging.
2. *Insert /*+ Append */ into new_table select* the records you want to keep from current_table.
3. Create the indexes on the new table with *NOLOGGING* (*)
4. Create constraints, grants etc.
5. Drop current_table.
6. Rename new_table to current.
7. Alter new_table and indexes logging.
8. Backup the data.

(*) If the data left is so small or there are a lot of dependencies on the table (views, procedures, functions) the following steps can be used instead of 3-6 in the previous slide:

3. Disable constrains on current_table;
4. Truncate current_table;
5. make indexes unusable;
6. *alter current table NOLOGGING ;*
7. *Insert /*+ APPEND */ into current_table select * from new_table ;*
8. commit;
9. rebuild indexes with *NOLOGGING*;
10. enable constraints
11. Put current table and indexes in *LOGGING* mode
12. backup the data
13. drop table new_table;

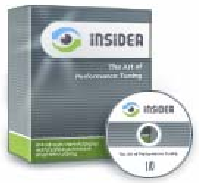
Bulk Update

Follow the steps for bulk Delete but integrate the update within the select statement. Lets say that you want to update the value column in the goods table by increasing it by 10% the statement will be like:

1. Create a new_table with no logging.
2. *Insert /*+ Append */ into new_table select (update statement eg: col1, col2* 1.1,…)*
3. Create the indexes on the new table with *NOLOGGING* (*)
4. Create constraints, grants etc.
5. Drop current_table.
6. Rename new_table to current.
7. Alter new_table and indexes logging.
8. Backup the data.

Some Common Problems

- *Block Corruption due to NoLogging (Standby DB)*
- *Recover problems (NoLogging Data)*
- *Excessive Log Swiches on Bulk Transactions (Logging)*
- ***'log file parallel write'***
- ***'log file sync'***



ORA-01578: ORACLE data block corrupted (file # 3, block # 2527)
ORA-01110: data file 1: '/u1/oracle/dbs/stdby/tbs_nologging_1.dbf‘
ORA-26040: Data block was loaded using the NOLOGGING option“

How to Detect Redo

Just Examine the amount of undo generated. When a transaction generates undo, it will automatically generate redo as well.

1) Query V$SESS_IO. This view contains the column BLOCK_CHANGES which indicates how much blocks have been changed by the session. High values indicate a session generating lots of redo.

The query you can use is:

```
SQL> SELECT s.sid, s.serial#, s.username, s.program,
  2  i.block_changes
  3  FROM v$session s, v$sess_io i
  4  WHERE s.sid = i.sid
  5  ORDER BY 5 desc, 1, 2, 3, 4;
```

Run the query multiple times and examine the delta between each occurrence of BLOCK_CHANGES. Large deltas indicate high redo generation by the session.

Detecting Redo (Part II)

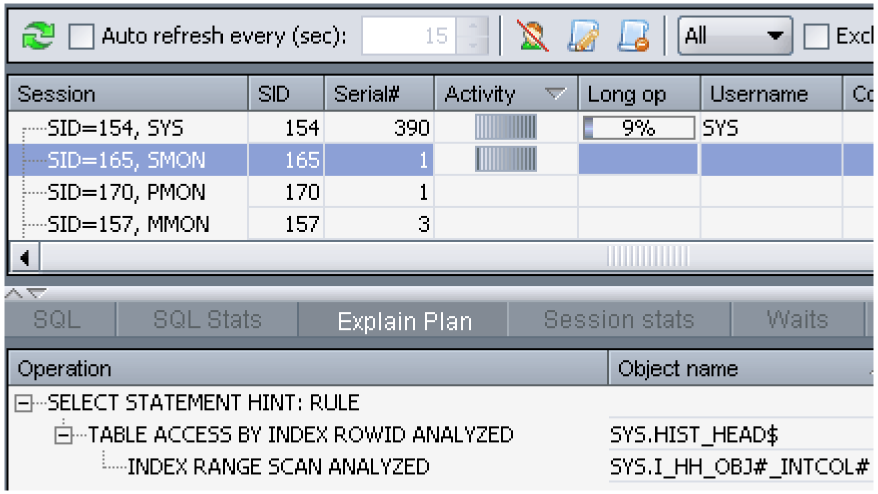
2) Query V$TRANSACTION. These view contains information about the amount of undo blocks and undo records accessed by the transaction (as found in the USED_UBLK and USED_UREC columns).

The query you can use is:

```
SQL> SELECT s.sid, s.serial#, s.username, s.program,
  2  t.used_ublk, t.used_urec
  3  FROM v$session s, v$transaction t
  4  WHERE s.taddr = t.addr
  5  ORDER BY 5 desc, 6 desc, 1, 2, 3, 4;
```

Run the query multiple times and examine the delta between each occurrence of USED_UBLK and USED_UREC. Large deltas indicate high redo generation by the session.

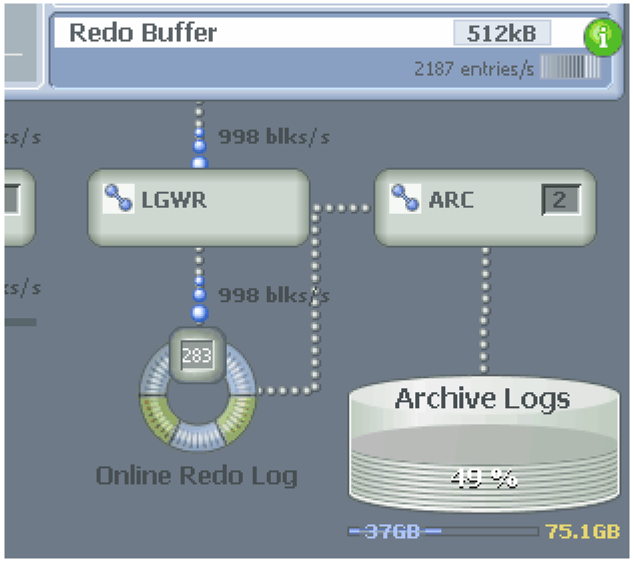
You use the first query when you need to check for programs generating lots of redo when these programs activate more than one transaction. The latter query can be used to find out which particular transactions are generating redo.

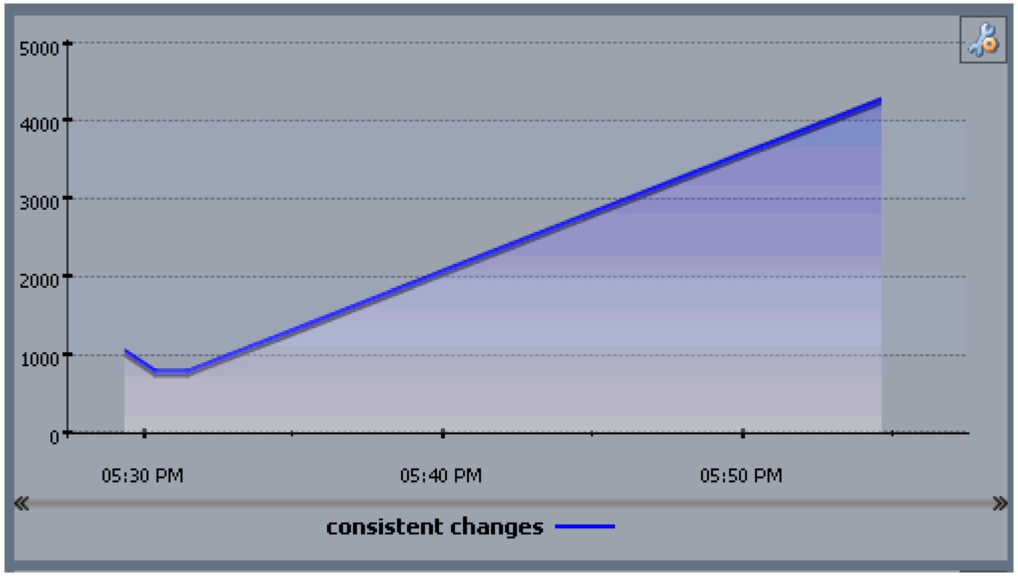


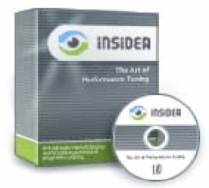
Auto refresh every (sec): 15 | All | Excl

Session	SID	Serial#	Activity	Long op	Username
SID=154, SYS	154	390		9%	SYS
SID=165, SMON	165	1			
SID=170, PMON	170	1			
SID=157, MMON	157	3			

Operation	Object name
SELECT STATEMENT HINT: RULE	
TABLE ACCESS BY INDEX ROWID ANALYZED	SYS.HIST_HEAD$
INDEX RANGE SCAN ANALYZED	SYS.I_HH_OBJ#_INTCOL#

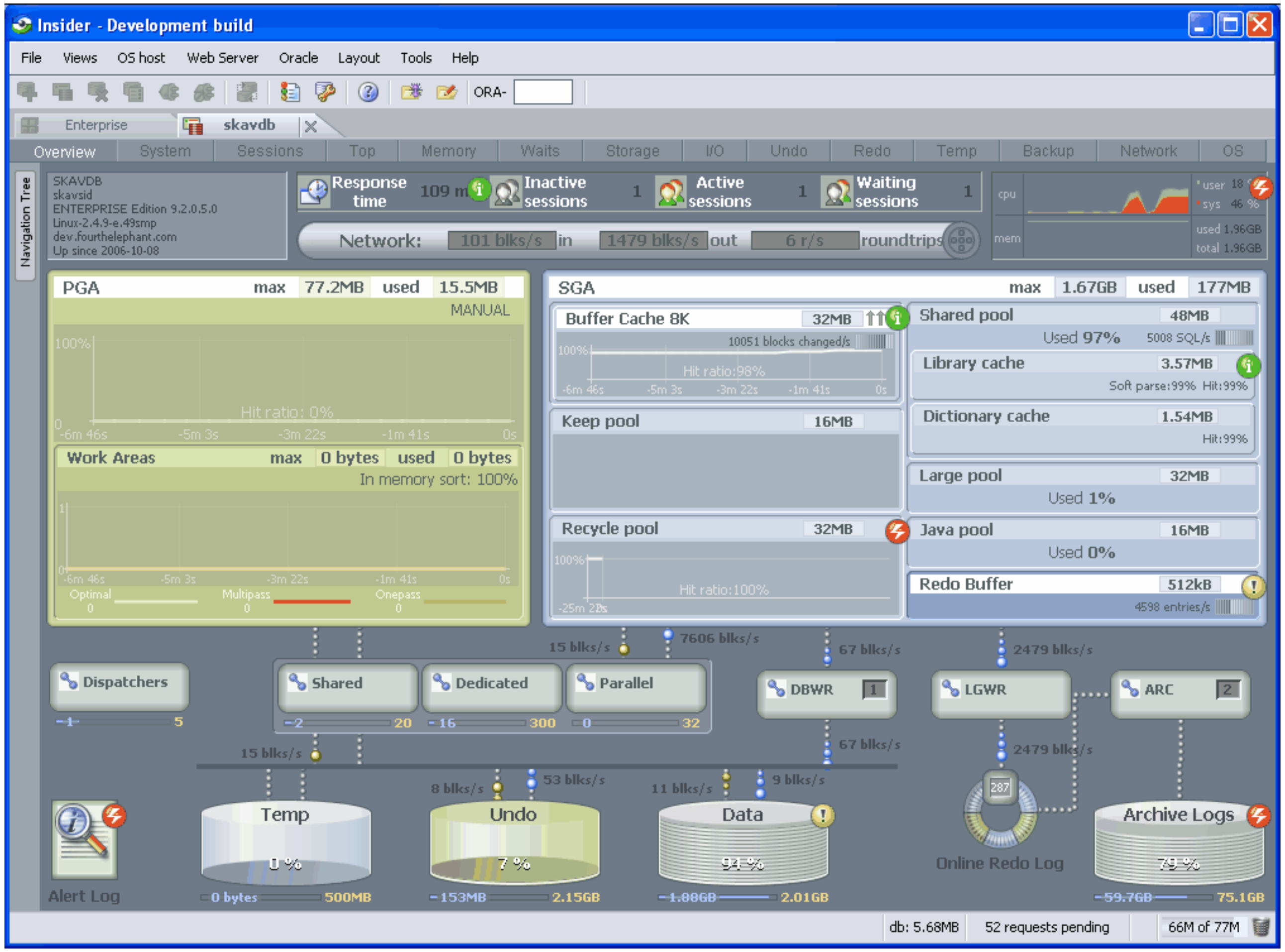


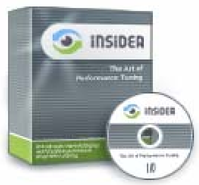




Download Insider here:
http://www.dbisonline.com

HROUG 14





dbis™ DATABASE INTEGRATED SOLUTIONS

Overview

- ✓ What is Redo?
- ✓ Redo Generation and Recoverability
- ✓ Redo and Undo
- ✓ Important Points about Logging and NoLogging
- ✓ Disabling Redo Generation (NoLogging)
- ✓ Tips
- ✓ Common Problems
- ✓ How to detect Redo Generation

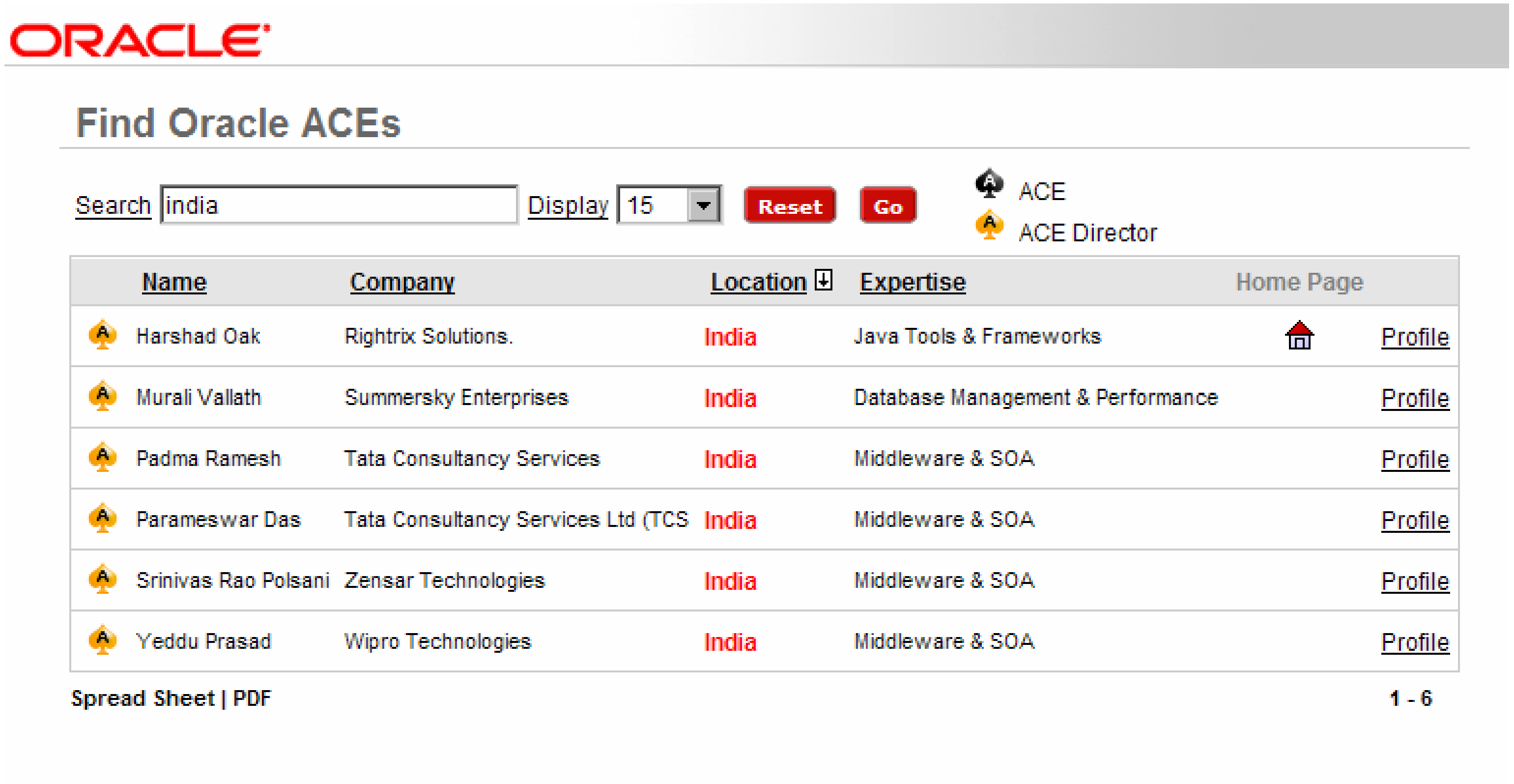
Oracle ACE PROGRAM

The Oracle ACE Program is designed to recognize and reward members of the Oracle Technology and Applications communities for their contributions to those communities. These individuals are technically proficient (when applicable) and willingly share their knowledge and experiences.

The program comprises two levels: **Oracle ACE and Oracle ACE Director.**

The former designation is Oracle's way of saying "thank you" to community contributors for their efforts; we (and the community) appreciate their enthusiasm. The latter designation is for community enthusiasts who not only share their knowledge (usually in extraordinary ways), but also want to increase their community advocacy and work more proactively with Oracle to find opportunities for the same. In this sense, Oracle ACE is "backward looking" and Oracle ACE Director is "forward looking."

Oracle ACE PROGRAM



Find Oracle ACEs

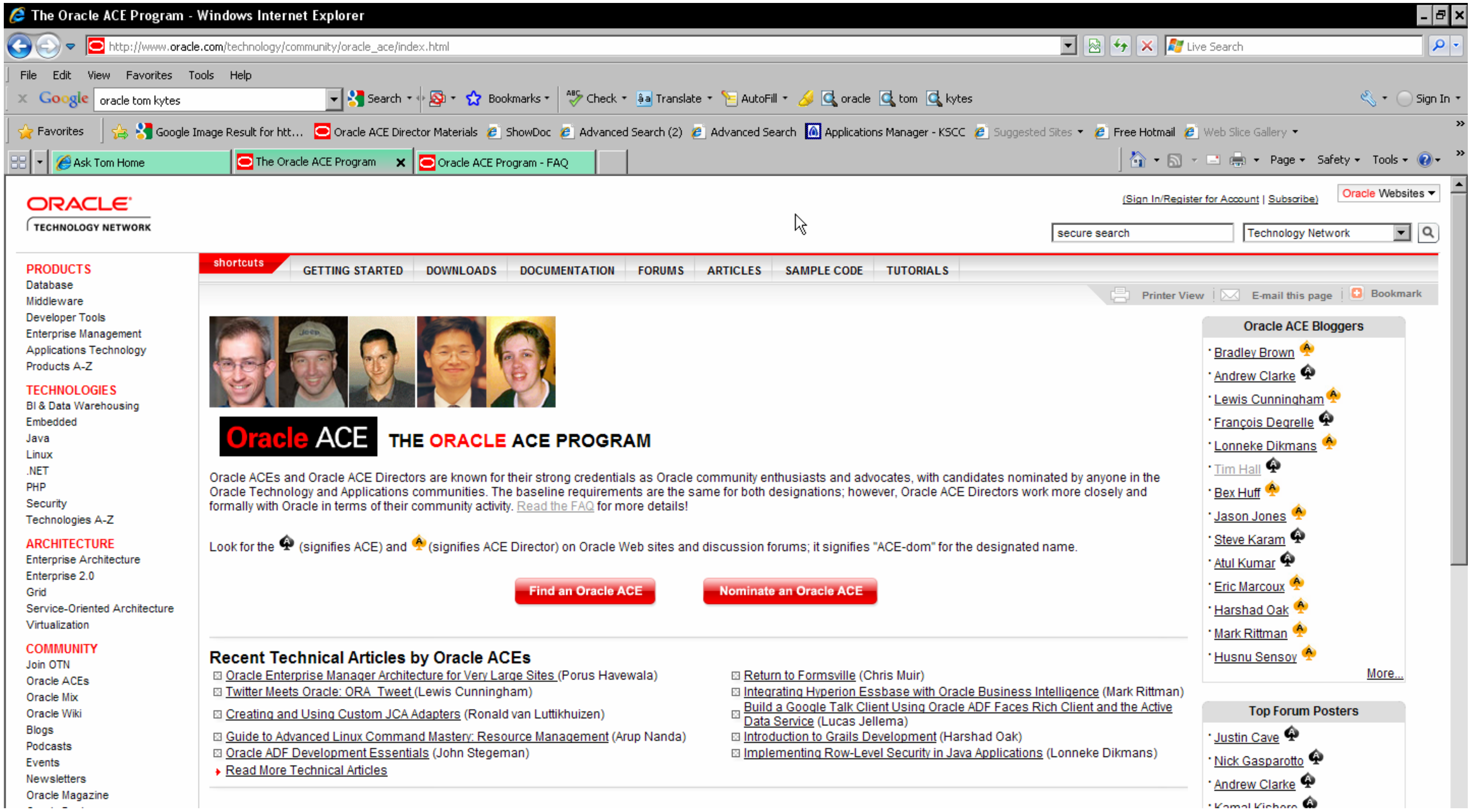
Search: india Display: 15 Reset Go

ACE
ACE Director

Name	Company	Location	Expertise	Home Page	
Harshad Oak	Rightrix Solutions.	India	Java Tools & Frameworks		Profile
Murali Vallath	Summersky Enterprises	India	Database Management & Performance		Profile
Padma Ramesh	Tata Consultancy Services	India	Middleware & SOA		Profile
Parameswar Das	Tata Consultancy Services Ltd (TCS	India	Middleware & SOA		Profile
Srinivas Rao Polsani	Zensar Technologies	India	Middleware & SOA		Profile
Yeddu Prasad	Wipro Technologies	India	Middleware & SOA		Profile

Spread Sheet | PDF 1 - 6

Oracle ACE PROGRAM



The Oracle ACE Program - Windows Internet Explorer

http://www.oracle.com/technology/community/oracle_ace/index.html

PRODUCTS
Database
Middleware
Developer Tools
Enterprise Management
Applications Technology
Products A-Z

TECHNOLOGIES
BI & Data Warehousing
Embedded
Java
Linux
.NET
PHP
Security
Technologies A-Z

ARCHITECTURE
Enterprise Architecture
Enterprise 2.0
Grid
Service-Oriented Architecture
Virtualization

COMMUNITY
Join OTN
Oracle ACEs
Oracle Mix
Oracle Wiki
Blogs
Podcasts
Events
Newsletters
Oracle Magazine

Oracle ACE THE ORACLE ACE PROGRAM

Oracle ACEs and Oracle ACE Directors are known for their strong credentials as Oracle community enthusiasts and advocates, with candidates nominated by anyone in the Oracle Technology and Applications communities. The baseline requirements are the same for both designations; however, Oracle ACE Directors work more closely and formally with Oracle in terms of their community activity. Read the FAQ for more details!

Look for the ♠ (signifies ACE) and ♠ (signifies ACE Director) on Oracle Web sites and discussion forums; it signifies "ACE-dom" for the designated name.

Recent Technical Articles by Oracle ACEs

- Oracle Enterprise Manager Architecture for Very Large Sites (Porus Havewala)
- Twitter Meets Oracle: ORA_Tweet (Lewis Cunningham)
- Creating and Using Custom JCA Adapters (Ronald van Luttikhuizen)
- Guide to Advanced Linux Command Mastery: Resource Management (Arup Nanda)
- Oracle ADF Development Essentials (John Stegeman)
- Read More Technical Articles
- Return to Formsville (Chris Muir)
- Integrating Hyperion Essbase with Oracle Business Intelligence (Mark Rittman)
- Build a Google Talk Client Using Oracle ADF Faces Rich Client and the Active Data Service (Lucas Jellema)
- Introduction to Grails Development (Harshad Oak)
- Implementing Row-Level Security in Java Applications (Lonneke Dikmans)

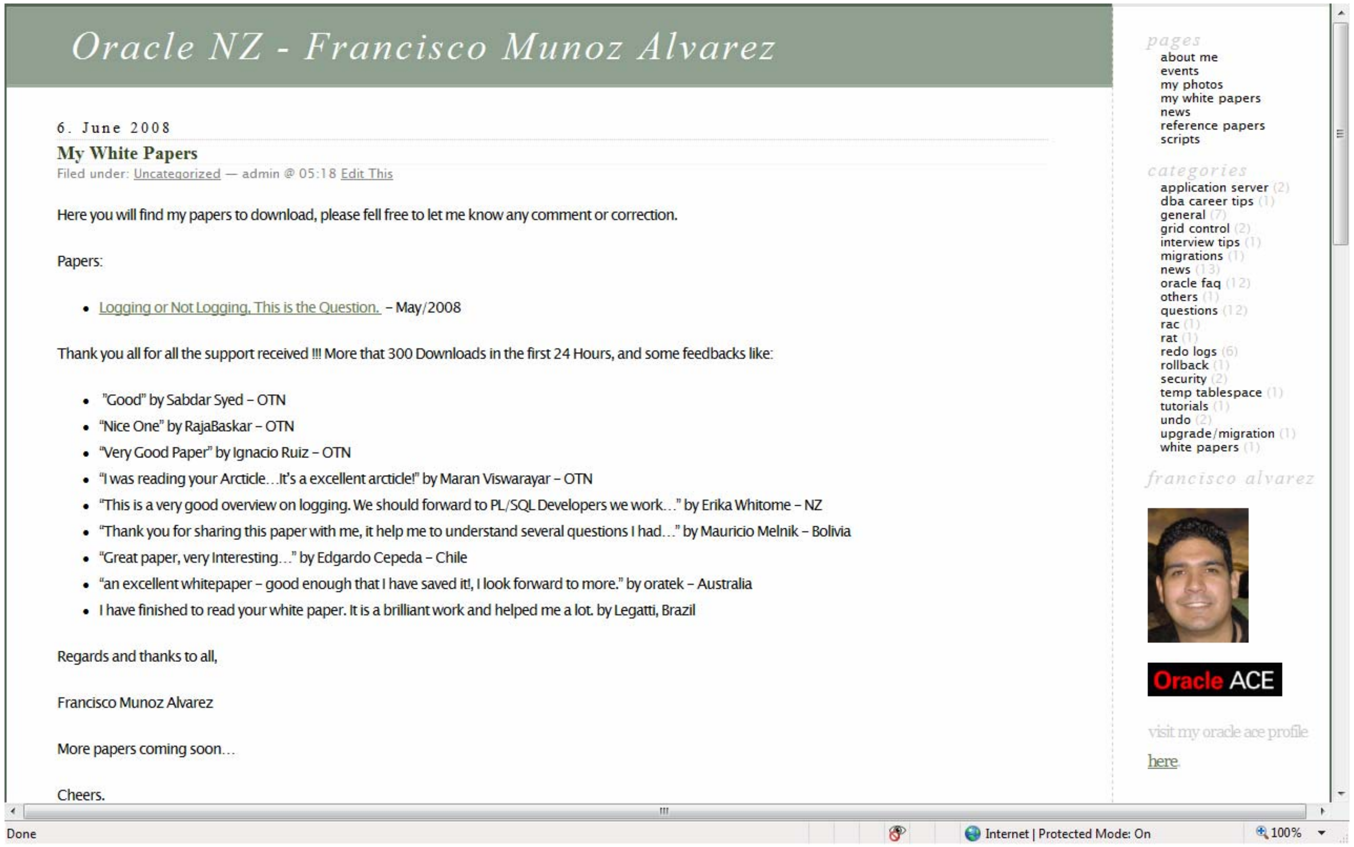
Oracle ACE Bloggers

- Bradley Brown
- Andrew Clarke
- Lewis Cunningham
- François Degrelle
- Lonneke Dikmans
- Tim Hall
- Bex Huff
- Jason Jones
- Steve Karam
- Atul Kumar
- Eric Marcoux
- Harshad Oak
- Mark Rittman
- Husnu Sensoy

More...

Top Forum Posters

- Justin Cave
- Nick Gasparotto
- Andrew Clarke



Oracle NZ - Francisco Munoz Alvarez

6. June 2008

My White Papers

Filed under: Uncategorized — admin @ 05:18 Edit This

Here you will find my papers to download, please fell free to let me know any comment or correction.

Papers:

- Logging or Not Logging, This is the Question. – May/2008

Thank you all for all the support received !!! More that 300 Downloads in the first 24 Hours, and some feedbacks like:

- "Good" by Sabdar Syed – OTN
- "Nice One" by RajaBaskar – OTN
- "Very Good Paper" by Ignacio Ruiz – OTN
- "I was reading your Arcticle…It's a excellent arcticle!" by Maran Viswarayar – OTN
- "This is a very good overview on logging. We should forward to PL/SQL Developers we work…" by Erika Whitome – NZ
- "Thank you for sharing this paper with me, it help me to understand several questions I had…" by Mauricio Melnik – Bolivia
- "Great paper, very Interesting…" by Edgardo Cepeda – Chile
- "an excellent whitepaper – good enough that I have saved it!, I look forward to more." by oratek – Australia
- I have finished to read your white paper. It is a brilliant work and helped me a lot. by Legatti, Brazil

Regards and thanks to all,

Francisco Munoz Alvarez

More papers coming soon…

Cheers.

pages

- about me
- events
- my photos
- my white papers
- news
- reference papers
- scripts

categories

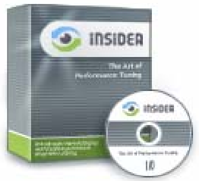
- application server (2)
- dba career tips (1)
- general (7)
- grid control (2)
- interview tips (1)
- migrations (1)
- news (13)
- oracle faq (12)
- others (1)
- questions (12)
- rac (1)
- rat (1)
- redo logs (6)
- rollback (1)
- security (2)
- temp tablespace (1)
- tutorials (1)
- undo (2)
- upgrade/migration (1)
- white papers (1)

francisco alvarez

Oracle ACE

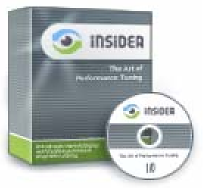
visit my oracle ace profile here.

Done | Internet | Protected Mode: On | 100%



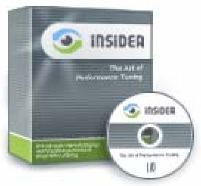
dbis™ DATABASE INTEGRATED SOLUTIONS

QUESTIONS?



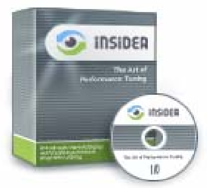
dbis™ DATABASE INTEGRATED SOLUTIONS

Now is your time to take the control of your Redo Generation



DATABASE INTEGRATED SOLUTIONS

Thank you !



dbis™ DATABASE INTEGRATED SOLUTIONS