Poor Man’s Auditing  
with  
Oracle Log Miner  

Caleb Small, BSc, ISP  
Caleb@Caleb.com  
www.Caleb.com/dba
Who is Caleb?

- Lifelong IT career
- Oracle DBA since v7.0
- Former Instructor for Oracle Corp.
- Independent consultant
- Faculty appointment Camosun College
- U of W night courses
- Vice-president, VicOUG
- Director, PSOUG
What does he know about Oracle?

• Installing Oracle Databases for 15yrs
• Studied RAC on 9i
• Re-wrote RAC curriculum for 10gR2
• Teaching RAC workshop for PSOUG
• Various RAC & DataGuard installations
• Setup PSOUG and VicOUG RAC labs
Case Study

- Software development company
- Medical research application
- Audit requirement driven by customer research funding rules
- Also desirable to track erroneous entries and changes
- Data access not an issue
- “Audit all changes to the database”
- Implement with least amount of resources, preferable no code changes
Architecture

- Typical middle tier Java application
- Hibernate/C3PO connection pool
- Uses database as back end store
- User authentication within the app
- Single point of interface with the DB
Possible Solutions

- **Audit Vault** — too complex, infrastructure requirements
- **Standard auditing** — does not capture values
- **Fine Grained Auditing** — too focused, not broad based
- **Trigger based** — code generation & maintenance
- **Application auditing** — don’t want to write code
- **Log Miner**
Advantages of Log Miner

- Mature technology
- Built in
- Simple implementation
- Broad based – all tables
- Captures before & after values
- Captures user & date
- Searchable
- No additional audit trail
Pre-requisites

- DB must be in archivelog mode
- Procedures *must* be in place to protect archive and online redo logs
- Archive logs *must* be kept for duration of audit period
- Gaps *may* be tolerable but represent lost audit information

Most production systems already meet these requirements
Additional Requirements

• Enable Supplemental Logging
  – Every txn writes addn’l info to redo log to identify the row being modified
  – Slight system overhead, slightly larger log files

• Periodically save the data dictionary
  – Data dictionary changes over time
  – Changes can be tracked but still require a starting point
  – RESETLOGS requires a new dictionary
Supplemental Logging

- **Minimal** (not recommended)
  - Stores physical ROWID
  - Only valid on source database, if row has not moved

- **Primary Key**
  - Stores PK, or alternately UK or all columns

- **Unique, Foreign Key, All Column**
  - More overhead
  - Useful if before and after image of entire row is required (not just columns that change)

- Can be confined to specific tables rather than whole database
Data Dictionary

1. On-line catalogue
   - Uses dictionary in source database
   - Simple and easy
   - Cannot reconstruct SQL across DDL changes!

2. Dictionary written to redo logs
   - ~10M addn’l redo
   - Provides starting point for mining sessions
   - DDL changes can be tracked seamlessly
   - Use DB job to write once/day or week
   - *Must* be re-written after RESETLOGS

3. Dictionary written to flat file
   - Deprecated
Setup Complete

With these simple steps complete,
Log Miner is already working!
Starting LogMiner

• Can reconstruct REDO and UNDO SQL
• Can mine on-line and archived redo
• Can be run on original source database or a different database
• Different database must:
  – Have the same, or a superset of, the character set
  – Be the same hardware platform
  – Be the same or a higher database version
Starting LogMiner – mining period

1. **SCNs** - only truly accurate measurement of time in the database, but difficult to determine after the fact.
2. **Timestamps (date and time)** - map to SCNs, but have a granularity of at least 3 seconds, possibly much more. Not always be possible for older data.
3. **Log sequence numbers** - most coarse time period specification, but may be the only option for older logs that have aged out of the control file.

- The time period must include a copy of the dictionary.
- Consider RMAN recovery catalogue for long term retention of archive log records
Starting LogMiner – Basic Steps

1. If necessary, add individual archived log files to the session
2. Start LogMiner with various options, including time or SCN range
3. Perform analysis by querying `v$logmnr_contents`
4. Optionally, restart LogMiner with different options or time/SCN range
5. End the LogMiner session
Supplemental Logging - Options

- **DICT_FROM_ONLINE_CATALOG**
  - use the online catalog, only valid if no DDL has been done

- **DICT_FROM_REDO_LOGS**
  - scan the redo logs for a copy of the dictionary

- **CONTINUOUS_MINE**
  - automatically locate redo logs for requested time/SCN period

- **COMMITTED_DATA_ONLY**
  - show only committed transactions

- **SKIP_CORRUPTION**
  - skip corrupt redo blocks, rather than terminate select

- **NO_SQL_DELIMITER**
  - format the appearance of reconstructed SQL

- **PRINTPRETTY_SQL**
  - format the appearance of reconstructed SQL

- **NO_ROWID_IN_STMT**
  - omit the ROWID in reconstructed SQL, use at least primary key supplemental logging with this option

- **DDL_DICT_TRACKING**
  - seamlessly track and report DDL changes
EXECUTE SYS.DBMS_LOGMNR.START_LOGMNR( -
  STARTTIME => TO_DATE('15-DEC-2007 17:00:00', 'DD-MON-YYYY hh24:mi:ss'), -
  ENDTIME => TO_DATE('16-DEC-2007 09:00:00', 'DD-MON-YYYY hh24:mi:ss'), -
  OPTIONS => SYS.DBMS_LOGMNR.DICT_FROM_REDO_LOGS + -
              SYS.DBMS_LOGMNR.DDL_DICT_TRACKING + -
              SYS.DBMS_LOGMNR.COMMITTED_DATA_ONLY + -
              SYS.DBMS_LOGMNR.NO_ROWID_IN_STMT + -
              SYS.DBMS_LOGMNR.CONTINUOUS_MINE);
Starting LogMiner – Perform Analysis

Select from v$logmnr_contents to perform analysis
Useful to store results in temp table for repeated query

CREATE TABLE lmtemp AS
    SELECT scn, timestamp, tx_name, seg_name, seg_type,
        operation, sql_redo,
        SYS.DBMS_LOGMNR.MINE_VALUE( UNDO_VALUE, 'LMUSER.EMP.SAL' )
        as oldsal,
        SYS.DBMS_LOGMNR.MINE_VALUE( REDO_VALUE, 'LMUSER.EMP.SAL' )
        as newsal
    FROM v$logmnr_contents
    WHERE seg_owner='LMUSER'
    AND SYS.DBMS_LOGMNR.COLUMN_PRESENT
        ( UNDO_VALUE,'LMUSER.EMP.SAL' ) = 1;
**Re-Start LogMiner Session**

`DBMS_LOGMNR.START_LOGMNR` can be called repeatedly within a given session without ending the current session.

- Useful to refine the mining period or to specify other mining options, without having to reload the dictionary.
- Loading the dictionary generates significant overhead as internal LogMiner objects are rebuilt - 90M of additional redo on a test database.
End the LogMiner Session

- Release resources
- Perform cleanup

EXEC DBMS_LOGMNR.END_LOGMNR();
If’s, and’s & but’s

The following data types are not supported:

- BFILE datatype
- Simple and nested abstract datatypes (ADTs)
- Collections (nested tables and VARRAYs)
- Object refs
- XMLTYPE datatype
- Tables using table compression
If’s, and’s & but’s

Middle Tier application does not set Oracle username!

SET_CLIENT_INFO does not help – not supported by LogMiner

Use TRANSACTION NAME instead
If’s, and’s & but’s

Running LogMiner on source database can generate a lot of redo

Use GLOBAL TEMPORARY table to store results of query to v$logmnr_contents

Consider mining from a separate database less flexibility (eg. Log seq #), but no addn’d redo
If’s, and’s & but’s

Record dictionary writes in a permanent table for future reference

Consider a DDL trigger to enforce dictionary write after RESETLOGS
References

Oracle Database Utilities, ch. 17 Using LogMiner to Analyze Redo Log Files

Oracle Database PL/SQL Packages and Types Reference, ch. 54 DBMS_LOGMNR and ch. 55 DBMS_LOGMNR_D

Package header files in $ORACLE_HOME/rdbms/admin: dbmslmd.sql & dbmslm.sql

Spoofing Oracle Session Information, Stephen Kost and Jack Kanter, Integrity Corporation, Chicago, Illinois, 2006

Oracle’s Log Miner Part 2, Darryl Hurley, Mobile Data Solutions, March, 2000
Crash and Burn Demo

Mining Period

- CREATE TABLE
- DML 500
- DDL
- DML 1000
- Write dictionary
- DML 2000
- DDL
- DML 3000

Puget Sound Oracle Users Group

Caleb@Caleb.com
Crash and Burn Demo

CREATE TABLE
500
DML
1000
DDL
Write dictionary
2000
DML
3000
DDL

Mining Period

Puget Sound Oracle Users Group
Caleb@Caleb.com
Crash and Burn Demo

Mining Period

2

CREATE TABLE

3

DDL

DML

500

1

DML

1000

1

Write dictionary

DML

2000

2

DDL

DML

3000

Puget Sound Oracle Users Group

Caleb@Caleb.com