JCC LogMiner Loader Version 2.1

Jeffrey S. Jalbert, JCC Consulting, Inc.

Announcement

JCC is announcing version 2.1 of the JCC LogMiner Loader product. This version adds a large number of features to the Loader including the ability to apply advanced filtering to rows and to send a single row to multiple target tables. This builds on the 2.0 features which supported high update databases.

Presentation Agenda

- What the Loader is used for today.
- New Features in version 2.1.
- Performance data.
- Additional examples.
- Futures.
- Your questions.

JCC LogMiner Loader

- The Loader publishes database changes to a target or targets.
- The Loader is used in mission-critical architectures.
- Targets can diverge significantly from the source.
- The source database is, generally, a production database that must run unchanged with little effect by the presence of the Loader.
- Some applications are extremely intensive, involving millions of customers or thousands of transactions per second.

Where to Use

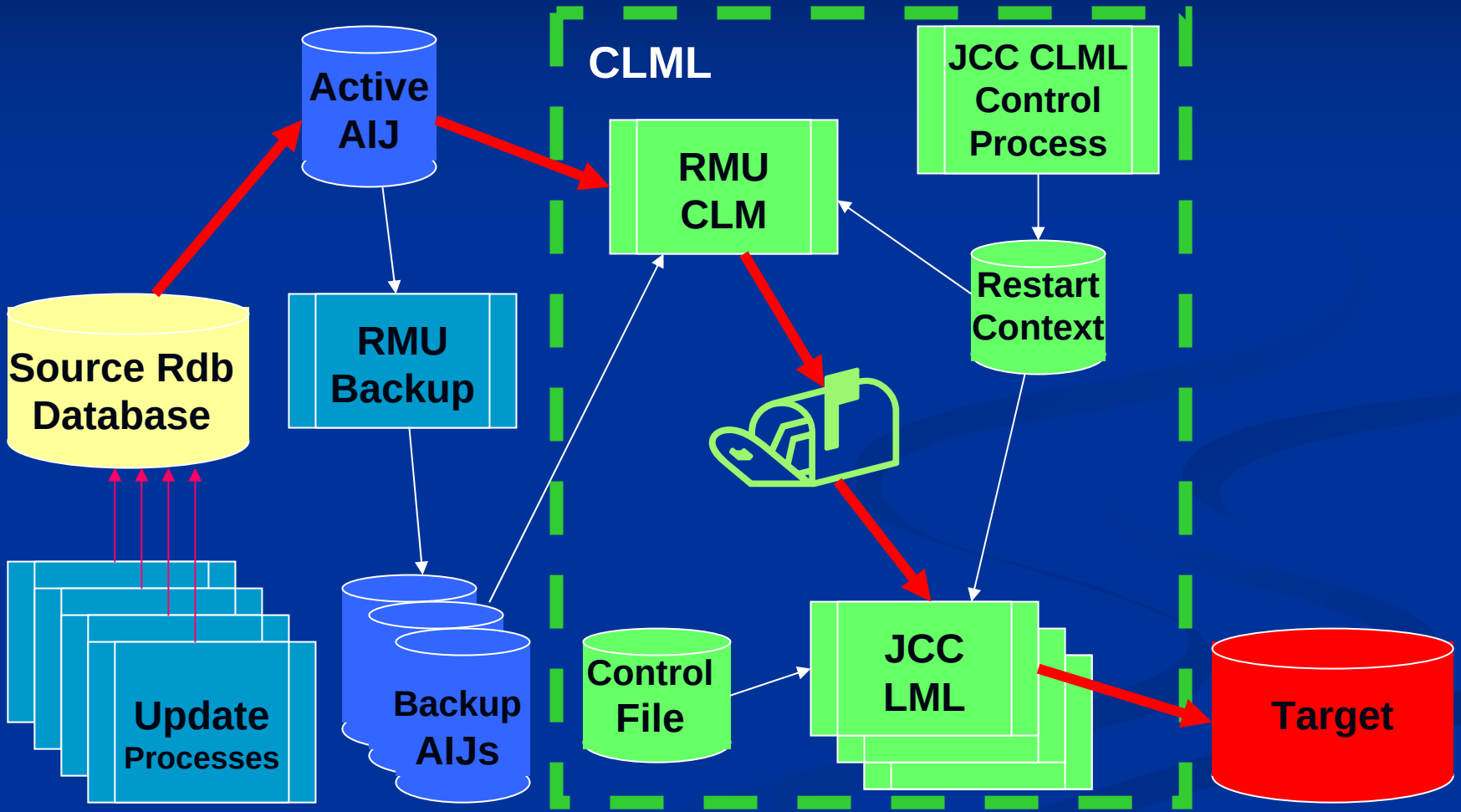
- Replication *
- Rollup *
- Archival
- Audit
- Data Warehousing
- Operational Data Stores (ODS)
- Web Browsing *
- Extract Transform Store (ETS)
- Publish changes to trigger actions
- Database Reorganization *
- Disaster Recovery
- Conversion from one database platform to another
- Capture of real world scenarios for regression testing
- Test environments for tuning alternatives
- Specialization for remote or departmental data subsets, whether for convenience, conservation of resources, or security

* Specific examples included.

Loader Sources and Targets

- Source
  - Any Rdb version that supports LogMiner
  - … <stay tuned>
- Targets
  - Rdb (any version that supports multi-statement procedures)
  - Oracle (via 8.1.7, 9.0 and 9.2 interfaces on VMS, any target Oracle version that can communicate with these SQL*net interfaces)
  - Tuxedo (from BEA, writing to whatever data store is supported and chosen)
  - XML (to a user-supplied API)
  - … <stay tuned>
- Note that the format of the target can be completely different than the format of the source database.
- Tuning of the target can be different.

How The Loader Does It



Configurable Loader Behavior

- The Loader actions are determined by a "Control File".
- Text editable format.
- Using keywords you define:
  - Source tables and columns
  - Mapping to the target
  - Actions to take on source actions (insert, update, delete)
  - Sorts, filters, values for null
  - Columns and tables to materialize
  - Performance and monitoring characteristics
  - Etc.
- Keywords have default values and many will not need to be specified.

Identifying Rows in the Target

- Changes are applied to target databases (Oracle or Rdb) on the basis of "Key" values.
- Keys are data columns in target.
- For databases modified by unusual mechanisms, the Loader supports using the row DB-key.
  - When data values change for a column in the primary key
  - When there is no primary key

Multi Version Support

- Multiple versions of the Loader may be run simultaneously on the same system.
  - Provides a way to test new versions while older versions are in production.
- Standard version could be an alternate dialect of the 2.1 Loader.
  - Unthreaded.
  - Older VMS support.

New Features in V 2.1

The focus of 2.1 development has been on:

- Reliability
- Performance
- Advanced mapping and multiple targets for single row
- Advanced Filtering
- Materialized columns as part of the Key
- Materialized table
- Real-time Throttle
- Enhanced Monitoring and DBA support
- Backward support for VMS 7.1 and 7.2
- Oracle 9.2 SQL*net on VMS.
- Data Pump

Reliability

- Regression testing
  - Extended to include back versions of VMS and multiple concurrent Oracle interface versions. (Different kits are required for old VMS.)
  - Random workload.
  - All capabilities exercised at the same time.
  - Loader families are "attacked" to validate recovery and restart.
  - Complete differences run between source and all targets.
  - Inter-target comparisons to ensure same results in all targets.
  - Cycles every ½ hour and runs for days (exercises the reliability of the targets and scheduling software.)
- Results
  - No serious errors have been uncovered after any release.

Throughput – Multiple Threads

- Support for update intensive databases via multiple (up to 32) concurrent processes (called threads).
  - Thread count may be dynamic to support varying workload.
  - DBA can force changes to thread counts.
- Mode is configurable.
  - Can be configured to avoid burying updates to target databases.
    - Use VMS Lock manager too ensure correct behavior.
  - Threads can run unconstrained and materialize the Loader Sequence Number column to mark relative age of rows in the target.

Performance – Commit Intervals and Checkpointing

- Commit interval defines how many source transactions to bundle into a single target transaction.
  - Can lead to efficiencies in processing.
- Continuing enhancements to provide tuning options.
  - By default, exclude NoWork transactions from checkpoints thereby reducing the workload in the target.
  - Timeouts possible if source database becomes dormant in the midst of a commit interval.
  - Dynamic adjustment of checkpoint interval depending on the volume of data in transactions.

Restart From Shutdown

- The Loader sends data on transaction boundaries.
- Loader is tolerant of environmental and downstream difficulties and interruptions.
- Loader maintains "checkpoint" information to ensure restartability.
- DBA may backup journals:
  - While Loader is running.
  - When Loader is down.
    - Unprocessed backup AIJs *must* be available on restart.

Other Performance Features

- Control placement of work files.
- Mechanism to define logical names within context of different processes.
- On restart, detects whether the last checkpoint is in live AIJs and restarts there rather than in backups.
- Additional materialized columns to facilitate physical organization of target database (Random value, Modulo, Thread number.)

Advanced Source to Target Mapping

- Single source row with subsets of columns sent to different tables.
  - No restrictions on mapping; single target row may receive updates from multiple source tables.
  - Key definitions for the mappings may be different.
- Advanced filtering supports any SQL restriction that only operates on the row under consideration.
- Values to use for nulls can be defined.

Materialized Data

- Loader can materialize columns.
  - Timestamps for start transaction, commit, etc.
  - Values to aid partitioning
  - Etc.
- Loader can also materialize an entire table.
  - Derived from the commit records received from the LogMiner.
  - Columns can be almost all materialized columns.
  - Application can use triggers in the target to cause additional actions.

Materialized Columns in Key

- The Loader may materialize columns such as
  - Loader name.
  - Constant.
- These columns can participate in the key of the target database.
  - Allows rollup of multiple source databases which have the same primary key values.
  - Example, rollup regional databases.

Throttle

- In some circumstances the Loader moves data too quickly. ☺
- Solution is to throttle the Loader performance
  - Realtime: Speed emulates real time transaction rate.
    - Best results if commit interval is 1.
  - Fixed: Commits to target spaced at a specific interval.
- Can be used to replay workloads in tuning efforts.

Oracle Target Support

- Loader is linked with Oracle SQL*net libraries for
  - 8.1.7
  - 9.0
  - 9.2
- Dialect selected by DBA when running.
- Requires that the Loader be linked with the VMS threads library.
  - Loader does not use threads, but Oracle interface does.
  - Older VMS versions require alternate linking.
- If not using Oracle, may start a non-threaded set of images for performance.

Monitoring

- Live: Configure which style and how frequently the statistics are taken for on-line displays.
- Logging: Configure what goes in the log, including more than should be included except for debugging and development.
- Locking: Display the current Loader locking tree for all families or a single family.
- Checkpointing: Display current checkpoint information.
  - Place in DCL symbols.
  - Useful for restart when combined with RMU.
  - Or if you have some sort of disaster.

Enhanced Monitoring

- Enhanced Logging by Loader threads.
  - Can provide periodic updates of work accomplished.
- Enhanced monitoring.
  - Full Report now displays costs for each stage of operation.
  - Detailed Report provides 24 line display of what is going on.
  - Comma Separated Values [CSV] report is suitable for loading into a database or spreadsheet in real time.

Other DBA Tools

- Control of Loader families cluster wide.
  - Requires DECnet proxies.
- Directed opcom messages.
  - Configurable with the Control File.
- Alerts from latency in excess of a threshold.
  - Makes it possible for the Loader to warn of trouble in the environment or downstream.

Data Pump

- Data Pump is included in the Loader kit.
- Data Pump is *not* required to run the Loader.
- Data Pump pumps more than the normal changes to the source.
- Data Pump makes no change updates to the source to "pump" the data through the Loader.

Data Pump

- One way to initially populate target table.
  - Rdb and the Loader kit include alternatives.
- Can be used to "repair" target if it were damaged by unintended updates by external processes.
  - Development errors
  - Production time frame errors
- Can generate a great deal of data and must be used wisely.

Data Pump

- Provides a way to force selected rows to the journal.
  - No-change updates.
  - Extremely low intrusion on source database.
    - Uses read-only transactions combined with configurable commit intervals for updates.
  - Minimize locking by using DBKeys during update phase.
  - Uses system and database resources efficiently.
- Structure file (SQL expressions) contains specification of parent–child relationships to pump entire structures.
- Driver file contains column values to be used in selecting rows.

Performance Example 1

- Loader being used to roll up 28 source databases to a single target [Tuxedo].
  - The architecture called for 28 because of the data volumes.
- Most of time spent waiting for Tuxedo servers.
  - Target servers are a pair of Himalaya systems.
- Maximum throughput in excess of 2,400 rows per second
  - Keeps up with 40 Wildfire 1Ghz CPUs running application

Performance Example 1 - Recovery

- Target Tuxedo servers were down for 3 days
- Examine the catch-up of one database
  - 11 tables being LogMined
  - 173 backup AIJs
  - 13,019,675 blocks of AIJ
  - 644,499 transactions (15,939 were NoWork)
  - 2,970,068 rows sent to the target
  - 1,144,352 rows were “filtered” by the Loader

Performance Example 1 – Catch-up

- Started at 10:25
- Ended at 12:04 – 1 hour and 39 minutes later
- At peak was sending almost 2,000 rows per second
- LogMiner used 30% of 1 CPU
- Loader threads each used 4% of 1 CPU for a total of 120%
- System still had idle time

Detailed Statistics Report

```
Rate:  30.00                        SUBR04_T              26-AUG-2003 11:50:59.72
================================================================================
   Input: 26-AUG-2003 02:24:23.94                Output: 26-AUG-2003 02:22:30.31
--------------------------------                --------------------------------
Transactions                524365              Checkpoints                 23321
Records                    4046142                 Timeout                      4
   Modify                  2869648                 BufferLimit(   250)       3078
   Delete                   652129                 NoWork                       1
   Commit                   524365              Records                   2612034
 Discarded                                         Messages(  7.1K)          251755
   Filtered                 947564                 TablesProcessed              11
   Excluded                      0              Failure                       754
   Unknown                       0              Timeout                         0
   Restart                      51          - Current ---------------- Ave
   NoWork                    13419          Checkpoints          302         10.07
   Heartbeat                     0          Records            54795       1826.44
Timeout                         32          Source(sec)       650.75         21.69
--- Restart Context -----                      Rate           2169.10%
M|AIJ#                  138560                - Current ------- Input ------ Output -
Q|VBN                    32204                Trail(sec)      33995.78      34109.41
P|TSN                 517499869
 CTSN                 517507220          - Loaders - 0123456789abcdefghijklmnopqrstuv
  LSN                  52209435          - States  - >>>>>>><R>>>>>>>>>>>>>>>>>>>>>>
```

Performance Example 2

- New installation: Replication to Oracle & Rdb was just being set up.
  - Source database on 6 CPU (1Ghz) Wildfire.
    - DB peaks at > 2,200 TPS for sustained intervals.
      - Not all of which do updates.
  - Target on a Windows server system and Wildfire
  - Target load from backup database completed at 4:30 PM.
  - Only necessary Primary key indexes were built.
- For Oracle, Loader family was started while other indexes were still being built.

Performance Example 2

- Loader runs 4x faster than production.
  - Minimum of 2 threads.
  - Maximum of 8 threads.
  - Using constrained mode to avoid buried updates.
  - Commit interval of 50.
- When “large” transaction occurs (>20,000 rows updated):
  - CLM would require time to digest.
  - Loader threads would shut down and restart afterwards.
    - Suggests altering thread dynamics parameters.
  - Final transmission of big transaction took 30 seconds.

Performance Example 2

- Update rate to target averaged > 1,000 rows per second.
- Reducing commit interval to 1 resulted in Loaders remaining no more than 1 second behind real-time.
  - Except for the occasional 20K row transaction.
  - And catch-up was prompt.

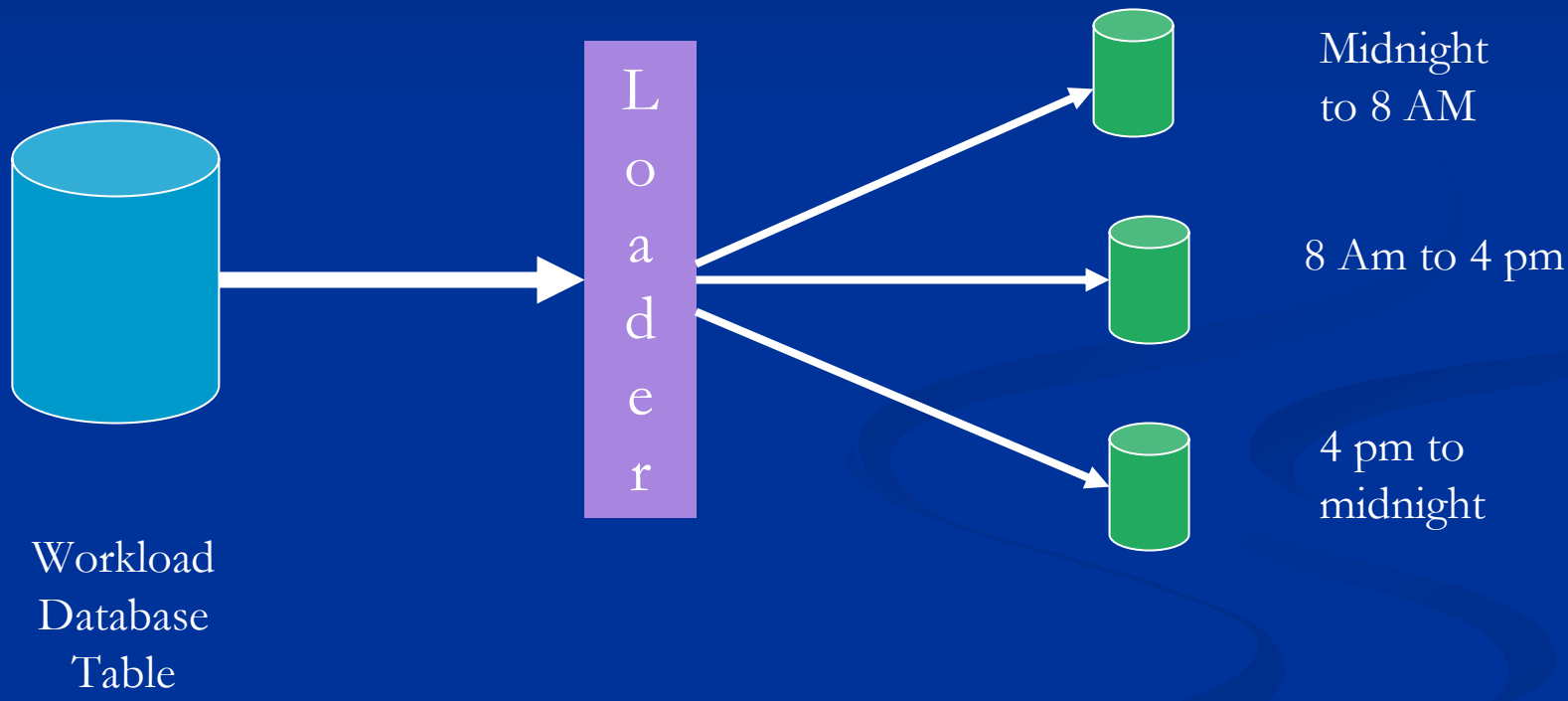
Performance Example 2

- Over the weekend extremely large transactions purge data.
  - 500K rows per transaction.
- The limiting resource in this case was the VMS lock manager.
  - Configured for too few resources.

Example - ETL Capabilities

- The Loader may be looked at as an Extract, Transform and Load tool.
  - A single row *may* have subsets of columns sent to different tables.
  - No restrictions on mapping are applied, a single target table may receive updates from multiple source tables.
  - Key definitions may be different.
  - Some data conversion possible for null columns.
  - Advanced filtering applies any Rdb SQL predicate to rows to determine whether to send them or not.
- In an interesting example: Time-based filters (commit timestamp) and multiple targets allows the Loader to populate multiple work tables alternately.

Filtering and ETL



Other Examples

1. Vital database reorganization was repeatedly deferred because downtime was intolerable.
   - With the Loader, downtime was under half an hour.
2. Company couldn't compete on the web because information could not be retrieved quickly enough.
   - Using the Loader to replicate the database and partitioning the work, the company is competing.
3. Company data is fractured into regional databases.
   - Using the Loader, the data is rolled up in "near real time."

Futures

- Support no-change behavior to avoid an update in target Rdb database.
  - Permits peer databases to be updated and synchronize each other.
- Transaction type selection for Rdb targets.
- Performance.
- For CLM, read metadata from workload database.
- Control order in which tables are sent to target.
- Better DBA control of when tardy alarms are sent.
- More database targets.
- Oracle → Rdb (?).
- Validate with Oracle 10g target db.

Availability

- Kit is available at FTP.JCC.COM
  - Documentation
  - Kit
- Evaluation license available on request
  - Send mail to jeff@jcc.com

Acknowledgements

- Thanks to Rdb engineering for their support and counsel

Questions