

Oracle Rdb LogMiner and JCC Loader

An On-Line Demonstration

Abstract

The Rdb LogMiner and the JCC LogMiner Loader provide another opportunity to reflect Rdb data into multiple independent databases. Unlike the Hot Standby option, the target database may be very different in architecture from the source database, and can even be an Oracle 9i database or data can even be sent to your own API in XML format. The recent delivery of Continuous LogMiner and version 1.2 of the Loader supports this transport in near real time. The technology can also be used for other purposes such as database rebuilds and even for database tuning.

While setting up a LogMiner and Loader session can seem to be complicated, in fact it is quite straightforward with the procedures supplied in the Loader kit. This presentation will demonstrate how easy it is to set up a LogMiner Loader session and manage the process and will demonstrate the workings of the product set.

Performance data will also be presented.

Agenda

- Review LogMiner Concepts
- Architecture of static LogMining
- Architecture of continuous LogMining
- Architecture of JCC Loader
- Demo of LogMining and Loading
- Performance review

Demonstration

- The demonstrations will require two databases
 - Create an MF-personnel database (Our “transaction processing” database)
 - Create a single file personnel database (Our “reporting” database)

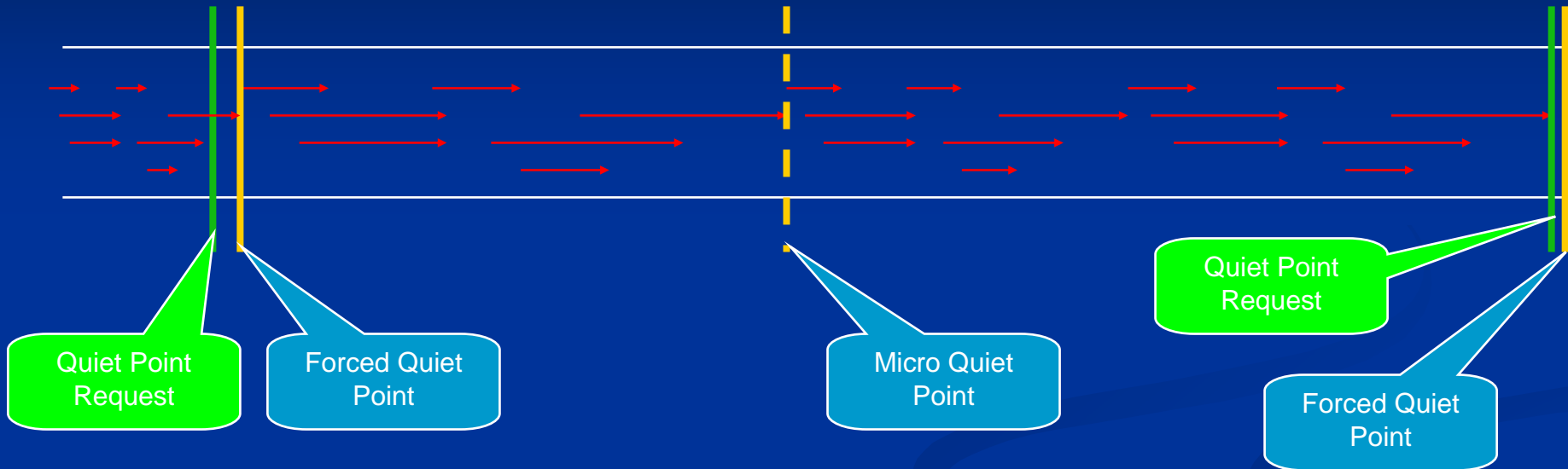
AIJ Files and Rdb

- An AIJ file or set of AIJ files is associated with an Rdb database.
- From that point in time onward, almost all changes to the database are also reflected in the Journal.
 - Some database properties are not reflected in the Journal, e.g. having the database enabled for LogMiner.
 - The number of these is diminishing over time.
 - Does not represent application data anyway
- Conceptually the journal is a continuous record of changes to the database that never terminates.
 - Is partitioned for management purposes into individual files.
 - Which can be emptied out periodically to provide space for additional journal data.

Quiet Points

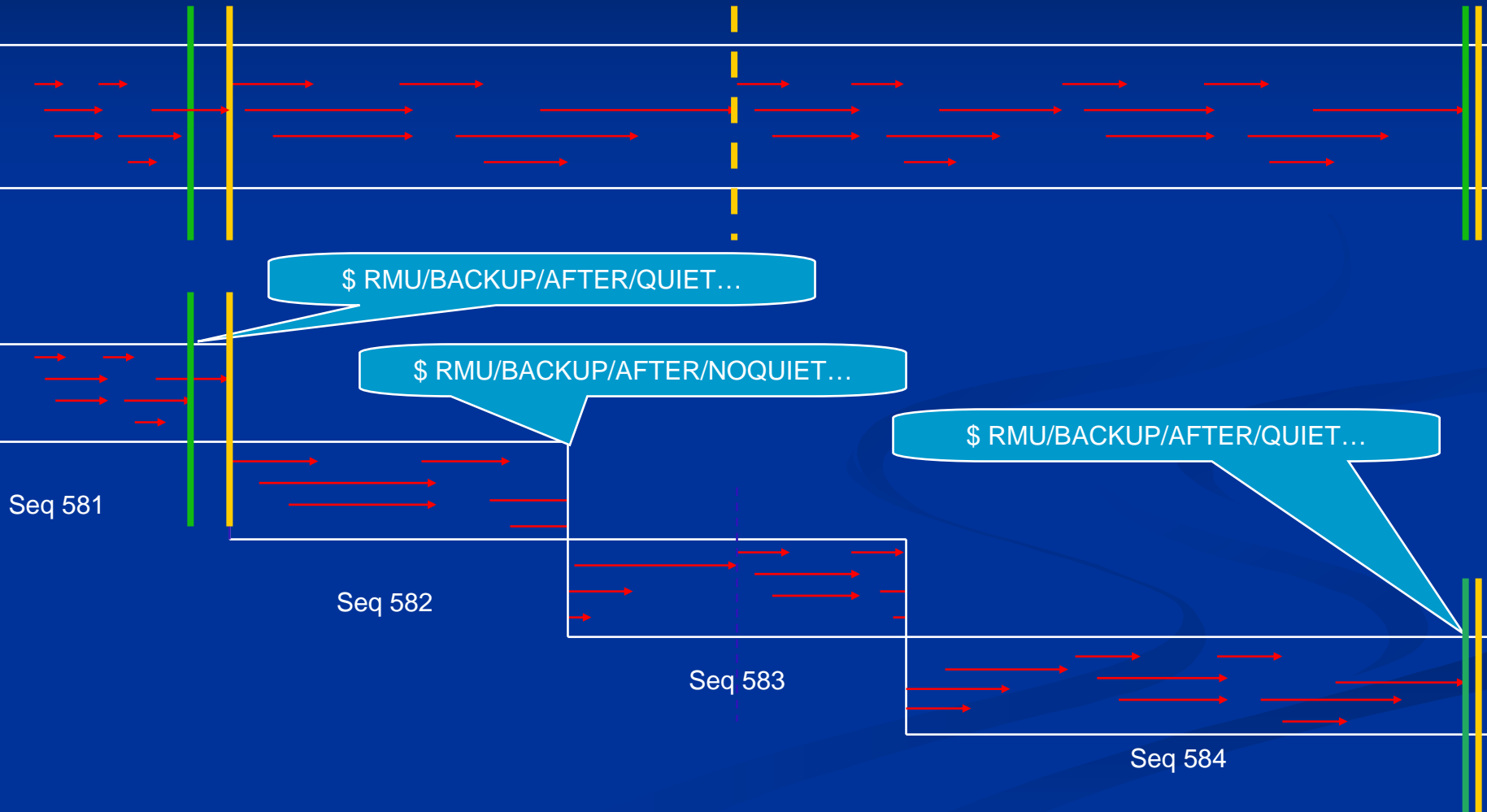
- There are special locations within the Journal, called quiet points, at which all transactions *that have written to the Journal* have committed.
- Each AIJ file contains a header which can record whether this quiet point has occurred *at the time the previous AIJ backup has occurred*.
- Some quiet points are forced by the database administrator or the Rdb engine requesting quiet points.
- Some quiet points are spontaneous, just because.
 - Called “micro” quiet points.
 - Are surprisingly frequent.

After Image Journal



Logical view – continuous record of database changes

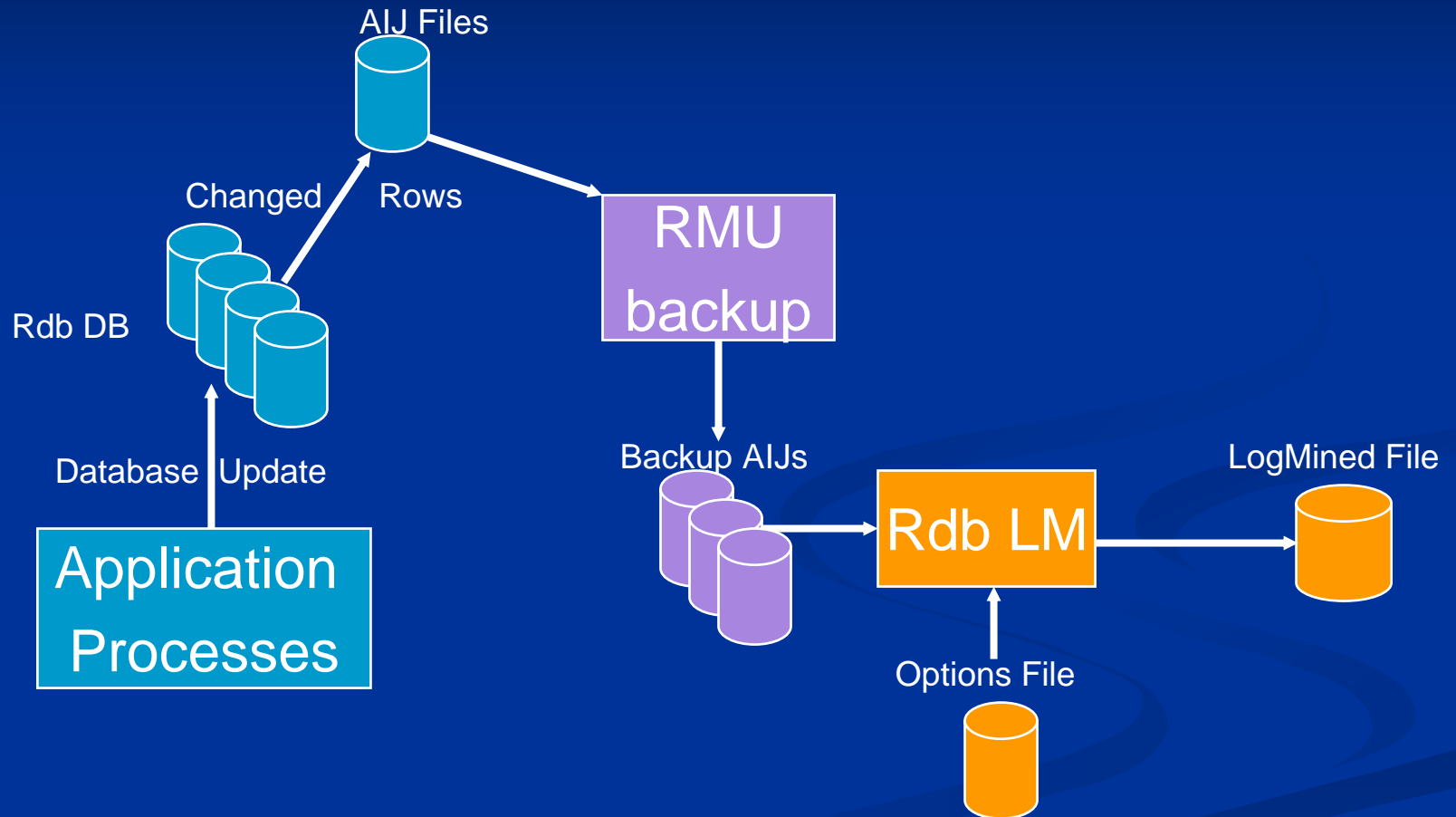
After Image Journal Backups



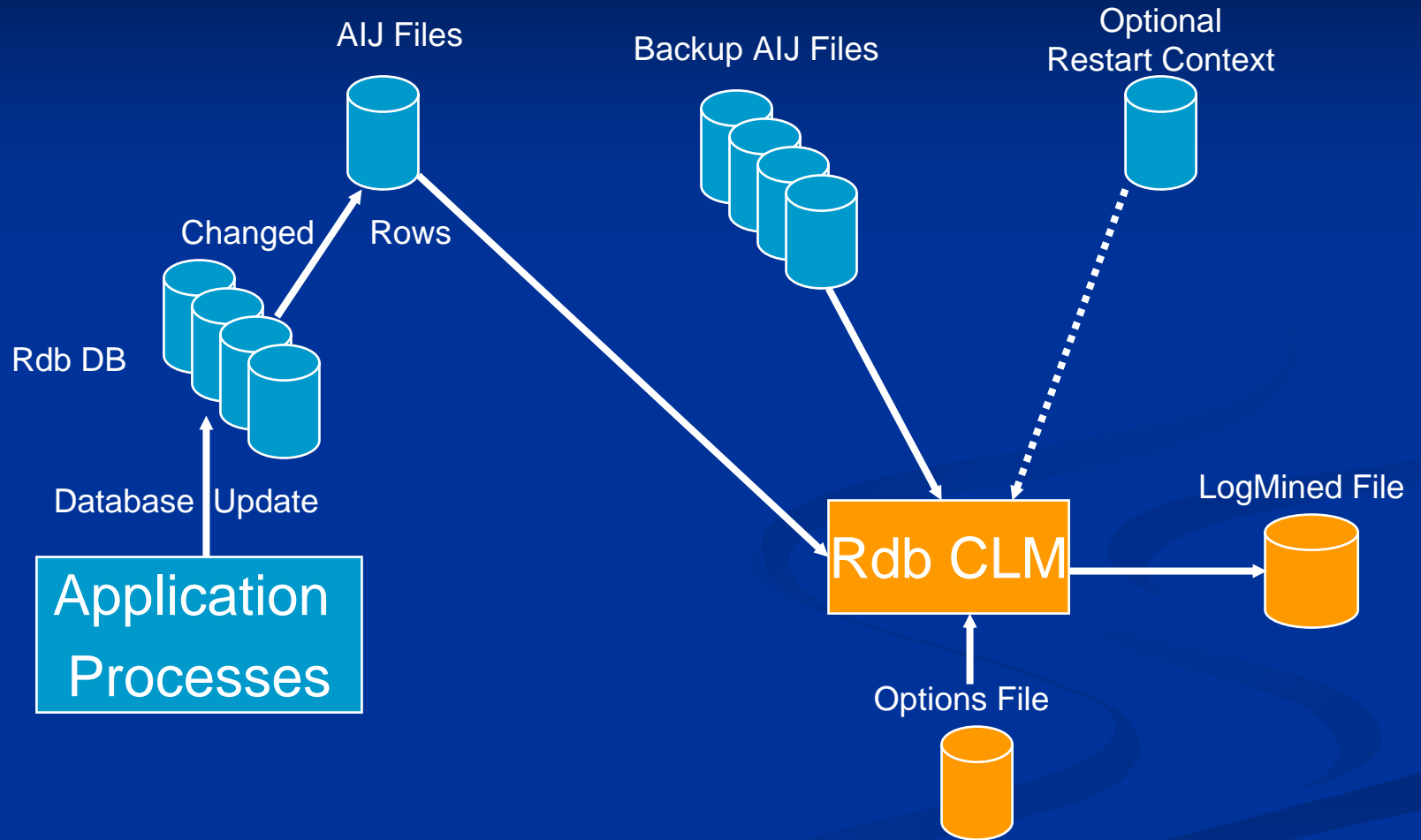
What Rdb LogMiner Does

- Extracts records for specified tables written to the AIJ.
- Writes requested records to specified output files.
 - Disk files.
 - Mailbox (AKA pipe).
 - Contents of each output file is records for specific tables.
 - JCC LogMiner Loader expects all records to be in a single file.
- If continuous LogMining is in progress can also write commit records into the output file.
 - Must be explicitly requested.
 - `/include=action=commit.`
 - Required for LogMiner Loader.
 - If CLM is in progress, can restart at some context specified in commit record.
 - Even if context has been moved to backup AIJ file.

Static LogMining



Continuous LogMining



LogMiner Output Format

- Each record in the AIJ is formatted separately.
 - Action.
 - “M” indicates row was updated.
 - “D” indicates row was deleted. Requires database to be enabled for LogMiner.
 - Table name.
 - Record type.
 - Data length.
 - Null bit vector length.
 - Db-key of underlying row.
 - Transaction start timestamp.
 - Transaction end timestamp.
 - TSN of transaction.
 - Row version number.
 - Data in its internal format.
 - Null bit vector, one bit per column.

LogMiner Output Format

- The commit record for the continuous LogMiner is somewhat different from the data records. It is present only when requested.
 - Action – “C.”
 - Table name – Empty.
 - Table type – Empty.
 - Data Length is the length of the fields at the end of the record.
 - Null bit vector length – Zero.
 - Logical DB-key – zero.
 - Transaction start date time.
 - Transaction end date time.
 - TSN of the transaction.
 - RM_TID_LEN – Length of the global TID (VMS TID).
 - AERCP_LEN – Length of the AERCP information.
 - RM_TID – The actual global TID for the transaction.
 - AERCP – AIJ Recovery Control Point (where to restart in the AIJ for this transaction).

AIJ Recovery Point (AERCPC)

- Used during restart of LogMiner.
 - Together with the TSN of the last transaction LogMined.
- The format of this field is documented in the Rdb release notes.
- Is not guaranteed across Rdb releases.
- Is parsed by the JCC LogMiner Loader real-time monitor to report and checkpoint where the Loader is relative to the current AIJ files.
 - Which AIJ is being processed.
 - What TSN is being mined.

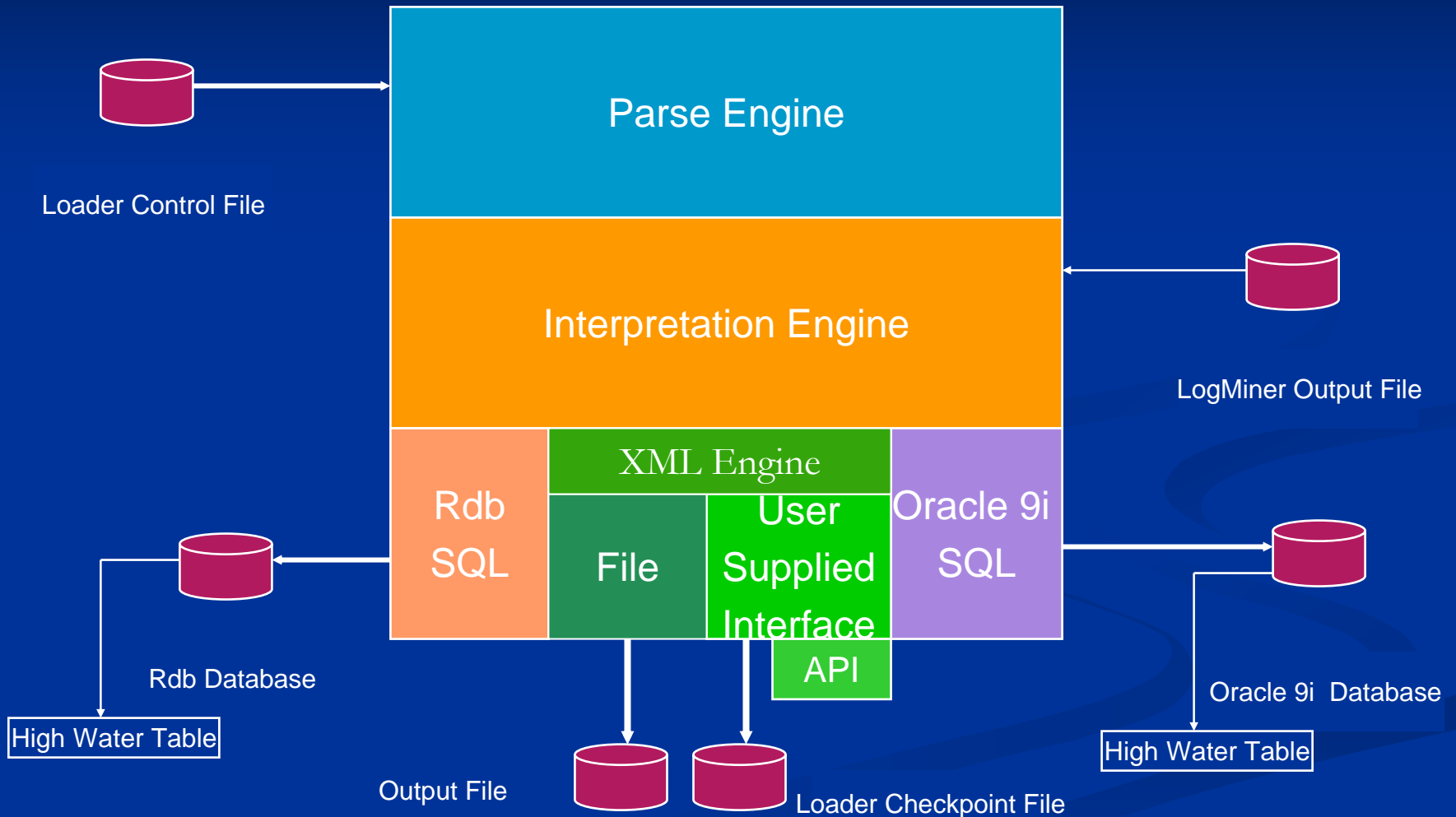
Database Access

- Static LogMining requires access to the metadata in the database.
 - By default will open the database to read this.
 - Can be directed to use an extracted “metadata” file.
- Continuous LogMining will require the LogMiner to *attach* to the database.
 - To extract metadata.
 - To discover when records are written to the AIJ.
 - When mining the backup AIJs, the CLM will not be attached to the database.

JCC LogMiner Loader Overview

- The JCC LogMiner Loader (the Loader) reads the binary files produced by the Rdb LogMiner and emits them to a target data store.
- The target data store may be one of the following:
 - Rdb database.
 - Oracle 9i database.
 - User-supplied API where one may generate an XML document per source transaction.
 - Disk file.
- The Loader does its work within a recoverable and restartable context.
 - Maps one or more source database transactions into a single output transaction.
 - Stores checkpoint data in either the target database or in a local checkpoint file.

JCC LogMiner Loader Architecture



JCC LogMiner Loader Overview

- The Loader's actions are specified in a special Loader control file.
 - Type of target.
 - Tables and columns to transmit to the target.
 - Columns to materialize in the target.
- Each instance of the Loader is named.
 - Multiple named Loaders may access the same target, but the Loader names within a single database target must be unique, absolutely.
 - Within the first 11 characters if this is a continuous Loading session.
 - Name of Loader is included in XML header. Causes header to vary in size.
 - API may require fixed width for this value.
 - Loaders running on the same system must be unique.

LogMiner Loader Overview

- The Loader reads and interprets the binary data presented by the LogMiner.
- The Loader must understand the database structure that is being LogMined.
 - Requires knowledge of the metadata.
- No table version changes allowed within a session.

LogMiner Loader Control File

- Control file contains a number of <keyword, value> combinations.
- The order of keywords is reasonably flexible.
 - LoaderName must be first keyword.
 - Columns and tables may not be excluded before they are defined.
 - Filters and virtual columns may not be defined before the associated tables are specified.
- Special keyword `include_file` enables structured construction of control files by inclusion of Loader configuration script as defined in another file.
 - Top level file contains LoaderName and other elements specific to Loader instance.
 - Include files contain the remainder of the definition.
 - Allows changeable information such as database metadata to be segregated to its own file.

LogMiner Loader Keywords

- The control file permits you to specify a wide range of actions.
 - Target database and type of database.
 - Accuracy and performance options.
 - What to do with tables and renaming tables and columns.
 - Replication.
 - Rollup.
 - Other.

Loader Control File

- Most database definitions will be long and complex.
 - Can automatically generate the metadata portion of the Loader control file with supplied utility.
 - “Guesses” primary keys.
 - Must regenerate whenever the database metadata is changed.
- The control file can be used to exclude certain tables and columns from being sent to the target.
 - The LogMiner can also be used to exclude tables
 - Cannot exclude what has not been defined.

Identifying Rows in The Target Database

- The Loader can be used to slave rows in a target database.
- The result will be that rows are exactly the same as they are in the production database.
 - Insert.
 - Update.
 - Delete.
- Requires a *primary key* that does not change.
- For applications that do not satisfy this, the Loader can materialize a column called `originating_dbkey`.
 - Guaranteed to be unique.
 - But DB-keys can be reused so table cannot be reorganized.

Columns in Tables

- The *datatypes* in the target database must be consistent with the *data* in the source database.
- Selected columns can be suppressed when going to the target database.
- Columns in the target database may be materialized.
 - Loader name.
 - Date timestamp of commit.
 - Etc.

Rows in the Target Database

- Transmitted of rows may be filtered on the basis of column values.
- May be applied to different columns simultaneously.
- Matches are always “equality” matches and are case sensitive for textual data.
- Filtering may be applied to materialized columns.

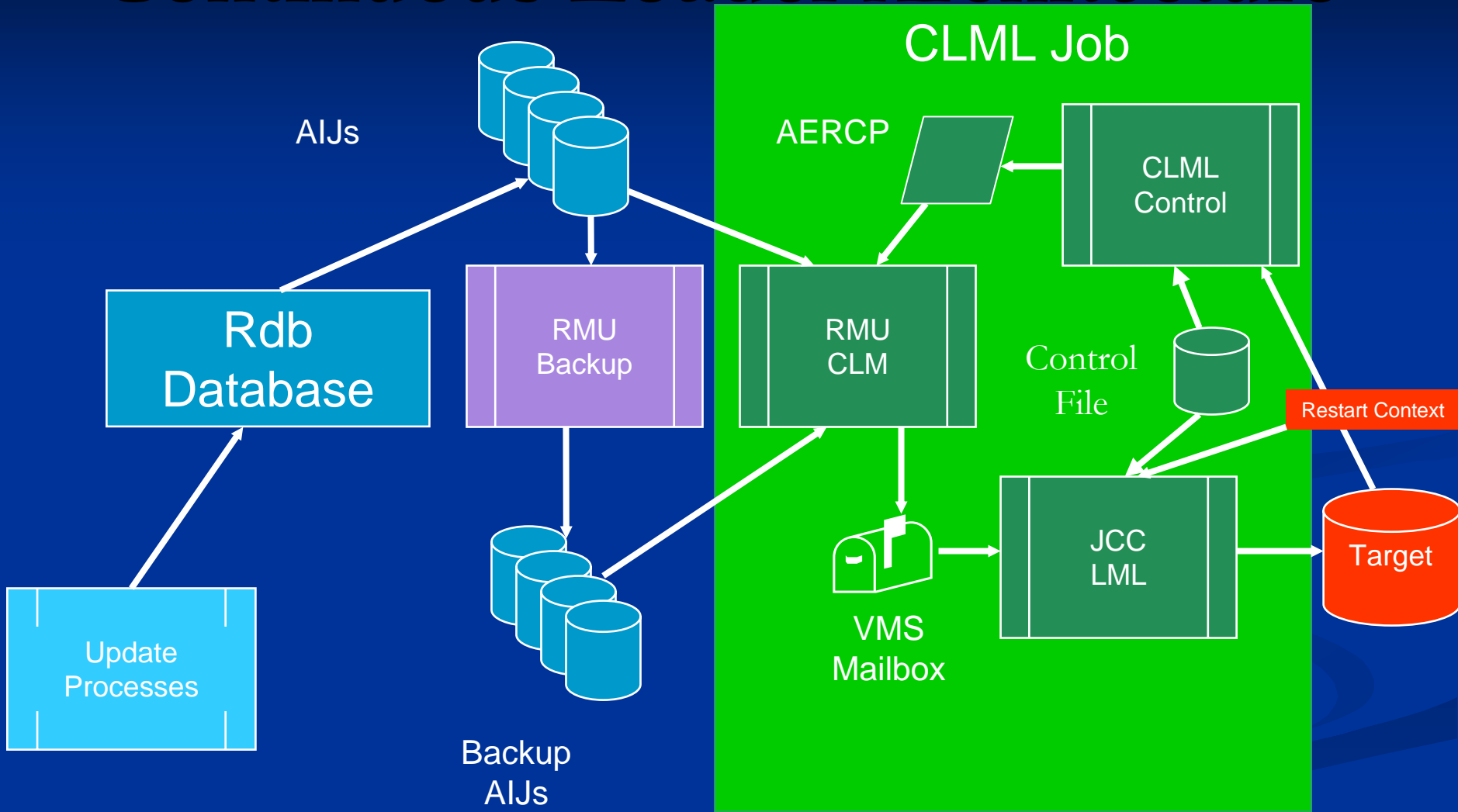
Logging

- The Loader may be directed to log its actions.
 - At multiple levels.
 - Can become extremely verbose.
- Logging levels cannot be adjusted in real-time.
 - Loader must be stopped and restarted.
 - Restart will pick up where left off.
 - Only a small penalty to do this restart.

Checkpoints

- The loader applies its changes in transactions.
- For performance reasons, multiple source database transactions may be grouped in a single target database transaction.
- The Loader retains restart context in the target database.
 - Requires a new table.
 - The TSN of the last transaction committed.
 - The AERCPC.
- For non-database output the Loader writes to a special checkpoint file.

Continuous Loader Architecture



Near Real-Time

- The LogMiner does not emit any records for a transaction until it finds a commit record.
- All records for that transaction are emitted in a single group.
- Therefore the LogMiner and Loader will always be “behind” the activities in the production database.

The CLML Control Program

- Reads the target database or local checkpoint file to obtain restart information.
- Creates a sub-process to run the Loader.
- Creates a sub-process to run the continuous LogMiner.
- Monitors the health of the two sub-processes.
- Acts as a logging sink for the log files for the two sub processes (uses more mailboxes for this).
 - Procedure supplied to cause log files to be reopened.
- Sends out operator alerts if something goes wrong.
- Listens for shutdown request.
- Drains out mailbox during shutdown.

Monitoring a Running Session

- As the Loader runs, it logs its activities in a global section.
- The contents of this global section may be displayed with the real time monitor provided with the kit.
- The monitor is not screen oriented.
- It reports activity at specific intervals as specified on command line.
- Two monitoring modes provided.
 - Full, displays a great deal of information.
 - Brief, provides simple summary of work to date.

OPCOM Messages

- The Loader will report various events via OPCOM.
- Loader: Indicates that the image has terminated with a problem in the control file.
- CLML Control: Indicates that the image has terminated with an exception.
 - A sub-process has gone away.
 - Something has gone wrong with the Control process itself.
 - Can't reopen log files etc.
- Real-time monitor process.
 - Indicating that the Loader is behind production by some threshold (is tardy).
- All of these messages (26 of them) are outlined in the documentation.

Demonstration

- Create an MF-personnel database (Our “transaction processing” database)
- Create a single file personnel database (Our “reporting” database)
- Run loader build scripts to generate:
 - Loader control file
 - LogMiner options file
 - Loader high water table in the target db
- Add AIJ files to TP database.
- Enable the TP database for Continuous LogMining
- Run CLML.
- Execute some transactions and watch them flow from source to target.
 - Via RMU.
 - Via Loader Real-time monitor.

Performance Experience # 1

- Database with normal transaction rate of 500 TPS.
 - About 2% of them update transactions.
- Loader database used for read only transactions in ACMS application.
- Target database is Rdb database.
- Maintains synchronization with one loader thread within a few (0-5) seconds.
 - Required commit interval of 50 because of bursts of do-nothing read-write transactions on night time batch runs.
- Average transaction duration reduced by 50%

Performance Experience #2

- Target database is Oracle 8i on Tru64 Unix.
- Single loader stream sends up to 18,000 records per minute per loader stream.
- Source database, at peak, produces 2,100 rows per second across 9 source databases.
- A collection of 4 loader streams per database maintains synchronization delay of no more than:
 - A few (3-5) minutes at peak for busiest tables.
 - A few seconds off-peak.
- Pegged the CPU resources on a 2 QBB GS320 (8 CPUs)

Performance Experience #3

- Reorganize 50Gb database in near-on-line mode.
 - Reorganize off line.
 - Apply AIJ iteratively to catch up.
 - One last AIJ application with database down.
- Minimal down time.
- Database now has architecture to support growth.
- Jobs run in $\frac{1}{2}$ of the time.

Performance Experience #3

Import Empty database	25 Minutes
Restore backup DB	120 Minutes
Unload Load all tables	420 Minutes
Add DBKey Indices	106 Minutes
LogMine 1 day of AIJ	7 Minutes
Run JCC Loader	60 Minutes
Apply indexes	180 Minutes
Mine and rerun Loader last time	30 Minutes
Add triggers and constraints, remove DB-Key indexes	2 Minutes

Production
was
Running
For
This
½
Day

Production
was
down

Product History

- Version 1.0 September 2001 Static LogMine Loading to an Rdb database
- Version 1.1 December 2001 Static LogMine Loading to Oracle 8i
- Version 1.2 June 2002 Continuous LogMine Loading and XML and File outputs. Validated on Oracle 9i, VMS 7.3 and 7.2-2, and Rdb 7.0.6.3, 7.0.6.4 and 7.1
- Version 2.0 [Fall 2002] Multithreading and other enhancements, TUXEDO interface.

Obtaining Evaluation Licenses

- 90-day evaluation licenses are available.
- Fill out form on JCC web page.
 - Include your name, mailing address, telephone number, organization and context information.
- Distribution kit is available on the JCC FTP site:
<ftp.jcc.com\outgoing\LogLoader>
 - Requires key to run.

Obtaining Documentation

- You may download documentation from:
- For HTTP access www.jcc.com and follow links to the loader.
- For FTP access: [ftp.jcc.com](ftp://ftp.jcc.com) and traverse to outgoing\logloader
- We are working on making a video of a training course available on-line.

Join the International Rdb Discussion Group



Send Email to: oraclerdb-request@jcc.com

In body of message type:

Subscribe OracleRdb (not case sensitive)

Questions

