Replicating Rdb Databases to SQL Server 2016

Keith Hare, Tom Musson,
Jeff Jalbert, Cheryl Jalbert

JCC Consulting, Inc.

Abstract

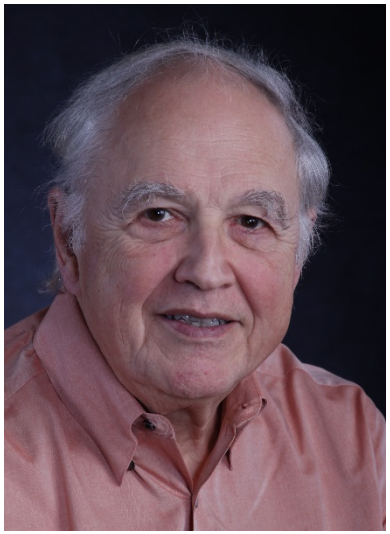
The JCC LogMiner Loader supports replicating committed transactions from Rdb databases to other databases using a variety of transports including Rdb, Oracle OCI, and JDBC drivers.

This session provides a detailed example of setting up replication from an Rdb database to an SQL Server 2016 database on a Windows 2016 server.

Who are we?

- JCC Consulting, Inc., began in June, 1984
- JCC first deployed the JCC LogMiner Loader in 2001
- The Loader team all do architecture, training, and ….

Tom Musson
Developer



Jeff Jalbert
Testing

Cheryl Jalbert
Documentation

Keith Hare
Installation

Introduction

- Data Replication Goals
- The Big Picture
- Preparing the source database
- Preparing the target database
- Setting up the JCC LogMiner Loader
- Monitoring Replication
- But what about …?
- Summary

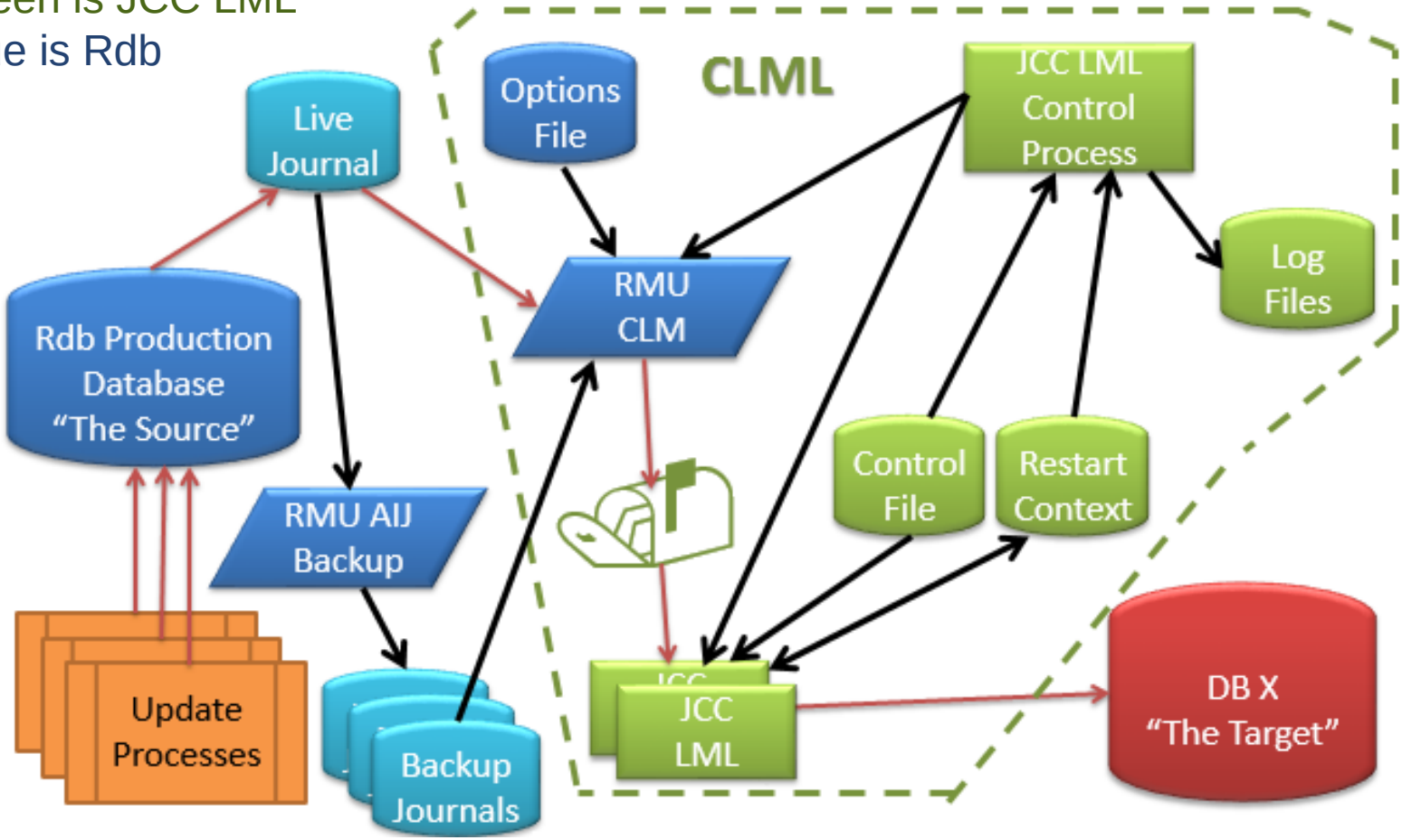
Data Replication Goals

Make Rdb data available another environment

- Near “real time”
- Transactionally consistent
- Reporting, Ad-Hoc queries, web site, testing, Audit, etc.

Middleware

Green is JCC LML
Blue is Rdb



Sources and Targets

- Source
  - Oracle Rdb (any version that supports the LogMiner)
- Target
  - Oracle Rdb (any version that supports multi-statement procedures)
  - Oracle (requires SQL*net client on the system running the Loader)
    - 12.1
    - 11.0, 11.1
    - 10.1, 10.2
    - 9.0, 9.2
    - 8.1 - deprecated
    - Oracle client-server compatibility (see Oracle matrix)
  - JDBC Class 4 driver
  - Tuxedo
  - XML (to your own API)
  - File

Sources and Targets (cont.)

- Format of the target can be completely different from format of the source database.
  - Logically
  - Physically
- Tuning of the target can be different.
  - Indices
  - Placement
  - Buffering
  - Caching

JCC LogMiner Loader Example

- Source Database
  - Oracle Rdb V7.3-210
  - OpenVMS IA64 V8.4-2L1
  - HP Integrity rx2800 i4 (2.39GHz/32.0MB)
    - 8 CPUs
    - 128 GB memory
- Target Database
  - SQL Server 2016
  - Windows Server 2016
  - Virtual X86
    - 4 CPUs
    - 16 GB memory

Preparing the source database

- Oracle Rdb V7.0.6.5 or later
  - OpenVMS V7.1 or later on Alpha or Itanium
  - More recent versions of Rdb and OpenVMS support more options
- Circular AIJs enabled
- Continuous Logminer enabled
- Immutable primary keys

Rdb Sample Database

Start with the Rdb multi-file personnel database to create a repeatable example:

```
ares > @sql$sample:personnel sql m nocdd

This command file builds the Oracle Rdb sample database, MF_PERSONNEL.
It requires approximately 4000 blocks of disk space.

You may enter the name of a VMS directory in which to create the database
or use your current default directory.
...
NOTE: If a copy of MF_PERSONNEL already exists in the location you specify,
this command procedure will stop.

     Enter VMS directory specification for the database files

Directory; default is JCC_ROOT:[KEITH.SQL_CLASS.MF_V73]:
...
```

Rdb Create Journal Example

```
$ define mfp_db JCC_ROOT:[KEITH.SQL_CLASS.MF_V73]mf_personnel
$ rmu/set after /reserve=20 mfp_db
$ rmu/set after mfp_db          -
/add=(name=aij_01       -
     ,file=JCC_USER:[KEITH.SQL_CLASS.mf_v73.aij]MFP_AIJ_01 -
     ,allocation=20000                                      -
    )                                                      -
...
/add=(name=aij_04                                          -
     ,file=JCC_USER:[KEITH.SQL_CLASS.mf_v73.aij]MFP_AIJ_04 -
     ,allocation=20000                                     -
    )
$!
$ rmu/set after/server=automatic mf_personnel
$ rmu/set logminer/enabled/continuous mf_personnel
$ sql
alter database filename mf_personnel
    journal is enabled
      (fast commit is enabled
      ,backup filename
          jcc_root:[keith.sql_class.mf_v73.backups]mfp_aij.aij
          (edit string is '_' + sequence)
      );
exit
$!
```

VMS_FUNCTIONS.SQL

- Optionally add functions to source database

```
ares > @jcc_tool_com:jcc_lml_user 3.5
ares > sql
SQL> attach 'filename mf_personnel';
SQL> @jcc_tool_sql:vms_functions.sql
%SQL-F-FCNNOTDEF, function JCC$SET_SYMBOL is not defined
%SQL-F-MODNOTDEF, module JCC$GET_SYMBOL_DATE is not defined
%SQL-F-PROCNOTDEF, procedure JCC$GET_SYMBOL is not defined
%SQL-F-MODNOTDEF, module CVT_DBKEY is not defined
%SQL-F-PROCNOTDEF, procedure EXT_CVT_DBKEY_TO_QUAD is not defined
%SQL-F-PROCNOTDEF, procedure EXT_CVT_QUAD_TO_DBKEY is not defined
SQL> SHOW FUNCTION
Functions in database with filename mf_personnel
     CVT_DBKEY
     JCC$GET_DATE_VALUE
 Function is Not Deterministic (variant)
     JCC$GET_SYMBOL_VALUE
 Function is Not Deterministic (variant)
     JCC$SET_SYMBOL                  An external function.
SQL>
```

Immutable primary keys

- Does not have to be defined as the source primary key
- Must not be updated
- Used for the target table primary key
- Can add SQL identity column to source tables if nothing else available

Target

- Creating tables
  - Primary Key and indexes
- Querying Data
- Describe a table
- This example uses SQL Server on Windows 2016

Preparing the target database

- Create tables
- Primary Key constraints as appropriate
- Create indexes as needed
  - Unique indexes for Primary Key
  - Other indexes should allow duplicates
- Do not define foreign key or referential constraints
- Do not define triggers that replicate work done on the source database

SQL Server Create Table Example

- Script created with

```
$ rmu/extract/item=table,index,constraint) –
  /language=ansi
  /output=mfp_tables.sql mf_personnel
```

- Some editing required

```
create table EMPLOYEES (
    EMPLOYEE_ID
        CHAR (5)
        constraint EMPLOYEES_PRIMARY_EMPLOYEE_ID
            primary key,
    LAST_NAME          CHAR (14),
    FIRST_NAME         CHAR (10),
    ...
    BIRTHDAY           DATETIME,
    STATUS_CODE        CHAR (1)          default 'N');
```

String Datatypes

Rdb	SQL Server
Char(10)	Char(10)
Varchar(10)	Varchar(10)
Nchar(10)	Nchar(10)
Nchar Varying (10)	Nvarchar(10)
Char(1500)	Char(1500), Text

Numeric Datatypes

Rdb	SQL Server
Tinyint (range is -128 to 127)	Tinyint (range is 0 – 255) or Smallint
Smallint	Smallint
Smallint(3)	Decimal(5,3)
Integer	Integer
Integer(2)	Decimal(10,2)
Bigint	Bigint
Bigint(4)	Decimal(20,4)
Real	Real
Double	Double

Date Time Datatypes

Rdb	SQL Server
Date VMS	Datetime
Date ANSI	Datetime
Time	Datetime
Timestamp(2)	Datetime

SQL Server User Interfaces

- Oracle SQL Developer
- Microsoft SQL Management Studio
- DBVisualizer
- Etc.

Setting up the JCC LogMiner Loader

- JDBC Driver
- Option File
- Configuration File
- Command procedure

jTDS SQL Server JDBC Drivers

- jTDS – SourceForge project
  - jtds-1.2.7.jar – Java 1.5
  - Jtds-1.2.8.jar – Java 1.6
  - jtds-1.3.1.jar – Java 1.8

The jTDS JDBC drivers can be downloaded from the jTDS project page

http://jtds.sourceforge.net/

Microsoft JDBC Drivers

- SQL Server 2012 JDBC Drivers
  - sqljdbc.jar – Java V1.5
  - sqljdbc4.jar – Java V1.6
  - sqljdbc41.jar – Java V1.7
  - sqljdbc42.jar – Java V1.8
- SQL Server 2016 JDBC Drivers
  - mssql-jdbc-6.2.0.jre7.jar – Java V1.7
  - mssql-jdbc-6.2.0.jre8.jar – Java V1.8

The Microsoft JDBC drivers can be downloaded from the appropriate Microsoft web site.

FTP JDBC Driver to OpenVMS

- Volume is ODS-5 so supports lowercase names

```
ares > di
Directory JCC_ROOT:[KEITH.SQL_CLASS.SQLSERVER.JTDS]
jtds-1-2-7.jar;1      595/595    5-JUL-2017 11:08:53.24
(RWED,RWED,RE,)
jtds-1-2-8.jar;1      590/590    6-JUL-2017 13:17:21.20
(RWED,RWED,RE,)
jtds-1-3-1.jar;1      621/625    5-JUL-2017 10:50:25.55
(RWED,RWED,RE,)
Total of 3 files, 1806/1810 blocks.
ares >
```

- Must reset the file attributes

```
ares > set file/attr=(rfm:stmlf,rat:cr,lrl:0,mr:0) *.jar;*/log
%SET-I-MODIFIED, JCC_ROOT:[KEITH.SQL_CLASS.SQLSERVER.JTDS]jtds-1-2-
7.jar;1 modified
%SET-I-MODIFIED, JCC_ROOT:[KEITH.SQL_CLASS.SQLSERVER.JTDS]jtds-1-2-
8.jar;1 modified
%SET-I-MODIFIED, JCC_ROOT:[KEITH.SQL_CLASS.SQLSERVER.JTDS]jtds-1-3-
1.jar;1 modified
ares >
```

Which JDBC Driver?

- Java 1.8 – IPF VMS
  - jtds-1-3-1.jar
  - sqljdbc42.jar
  - mssql-jdbc-6-2-0-jre8.jar
- Java 1.6 – IPF VMS
  - jtds-1-2-8.jar
  - sqljdbc4.jar
- Java 1.5 – Alpha VMS
  - jtds-1-2-7.jar
  - sqljdbc.jar

Sidetrack – Java on OpenVMS

- Alpha OpenVMS V8.4

```
dba08 > java -version
java version "1.5.0"
Java(TM) 2 Runtime Environment, Standard Edition
Fast VM (build 1.5.0-8, build J2SDK.v.1.5.0:02/08/2012-17:12, native threads,
jit_150)
dba08 >
```

- IPF OpenVMS V8.4-1H1 & V8.4.2

```
ares > @sys$common:[java$60.com]JAVA$60_SETUP.COM
ares > java -version
java version "1.6.0"
Java(TM) SE Runtime Environment "1.6.0-6"
Java HotSpot(TM) 64-Bit Server VM (build 20.6-b06, mixed mode)

ares > @sys$common:[java$80.com]java$setup.com
ares > java -version
java version "1.8.0.03-OpenVMS"
Java(TM) SE Runtime Environment (build 1.8.0.03-vms-rc1)
Java HotSpot(TM) 64-Bit Server VM (build 25.51-b02, mixed mode)
ares >
```

Java V1.6 and SSL

- Microsoft JDBC Drivers use SSL to transmit password
- SSL connection generates exception:

```
Exception in thread "main" com.microsoft.sqlserver.jdbc.SQLServerException:
The driver could not establish a secure connection to SQL Server by using Secure
Sockets Layer (SSL) encryption.
Error: "java.lang.RuntimeException: Could not generate DH keypair".
Caused by: java.security.InvalidAlgorithmParameterException: Prime size must be
multiple of 64, and can only range from 512 to 1024 (inclusive)
```

- Known issue in Java 1.6 and 1.7 (See JDK-6521495)
- Work around is to patch BouncyCastle's JCE implementation into your Java environment
  - I am **NOT** doing this on an OpenVMS server.
- For Java 1.6, use jtds-1-2-8.jar

Generate Options File

Tells RMU/Unload/After/Continuous which tables to unload

```
ares > JCC_CREATE_LOG_MINER_OPT_FILE mfp_v73
...
```

- Options file only needs to be regenerated when tables added
- If VMS_FUNCTIONS have not been added, exception message

```
dba08 > JCC_CREATE_LOG_MINER_OPT_FILE mfp_v73
%SQL-F-RTNNOTDEF, function or procedure JCC$SET_SYMBOL is not defined
JCC-I-VMS_FUNCTIONS, VMS_FUNCTIONS not in database.  Exclude capability
disabled.
...
```

- This example does not use the exclude capability

Options File Example

- Generated by the previous example

```
!
!  Continuous Logminer Options file
!  Generated at 2015-09-16 19:40:15
!
table=CANDIDATES,output=rdb_logminer_output_file
table=COLLEGES,output=rdb_logminer_output_file
table=DEGREES,output=rdb_logminer_output_file
table=DEPARTMENTS,output=rdb_logminer_output_file
table=EMPLOYEES,output=rdb_logminer_output_file
table=JOBS,output=rdb_logminer_output_file
table=JOB_HISTORY,output=rdb_logminer_output_file
!table=RESUMES,output=rdb_logminer_output_file
table=SALARY_HISTORY,output=rdb_logminer_output_file
table=WORK_STATUS,output=rdb_logminer_output_file
```

- RESUMES contains a segmented string

Generate Table and MapTable

```
ares > JCC_LML_CREATE_CONTROL_FILE mfp_v73 all
...
```

- Creates three files
  - Table File – MF_PERSONNEL_TABLE.INI
    - Describes source tables
    - Must be regenerated if source tables changed
  - MapTable – MF_PERSONNEL_MAPTABLE.INI
    - Maps source tables to target tables
  - DateFilter – MF_PERSONNEL_DATEFILTER.INI
    - Useful if source might have bad dates

Table File

- Describes source data

```
!
!  JCC LogMiner Loader Metadata (DDL) Control File
!  Generated at 2017-07-25 16:50:17
!
Table~CANDIDATES~1~~NoMapTable
Column~CANDIDATES~LAST_NAME~1~14~0~14~0
Column~CANDIDATES~FIRST_NAME~2~10~0~14~0
Column~CANDIDATES~MIDDLE_INITIAL~3~1~0~14~0
Column~CANDIDATES~CANDIDATE_STATUS~4~255~0~37~0
!
...
```

- Table name and version
- Columns with datatype information
- Do not generate by hand

MapTable – Candidates

```
...
MapTable~CANDIDATES~CANDIDATES~Replicate
!
!Table_"CANDIDATES"_has_neither_a_primary_key_nor_a_unique_index.
!
MapColumn~CANDIDATES~LAST_NAME
MapColumn~CANDIDATES~FIRST_NAME
MapColumn~CANDIDATES~MIDDLE_INITIAL
MapColumn~CANDIDATES~CANDIDATE_STATUS
!
!Originating_DBKey_column_required_due_to_no_key.
!
MapColumn~CANDIDATES~originating_dbkey
MapKey~CANDIDATES~originating_dbkey
...
```

- Primary Key is originating_dbkey

MapTable – Employees

```
MapTable~EMPLOYEES~EMPLOYEES~Replicate
!
!Using_defined_primary_key_constraint.
!
MapColumn~EMPLOYEES~EMPLOYEE_ID
MapColumn~EMPLOYEES~LAST_NAME
MapColumn~EMPLOYEES~FIRST_NAME
MapColumn~EMPLOYEES~MIDDLE_INITIAL
MapColumn~EMPLOYEES~ADDRESS_DATA_1
MapColumn~EMPLOYEES~ADDRESS_DATA_2
MapColumn~EMPLOYEES~CITY
MapColumn~EMPLOYEES~STATE
MapColumn~EMPLOYEES~POSTAL_CODE
MapColumn~EMPLOYEES~SEX
MapColumn~EMPLOYEES~BIRTHDAY
MapColumn~EMPLOYEES~STATUS_CODE
MapKey~EMPLOYEES~EMPLOYEE_ID
```

MapTable Comments

- Often requires editing
  - Correct primary key
  - Table or Column names different in target
- MapKey specifies logical primary key
  - Used when Updating/Inserting target
  - Generated based on the first found of:
    - Primary key constraint on source
    - A unique index
    - If none of the above, add OriginatingDBKey column

MapTable – Originating DBKey

- Originating DBKey can be used as primary key for target table
- Rdb DBKey is physical address of source record
- Use 8-byte integer (BIGINT) in the target DB
- If source table is reorganized (unload/load) target table must be completely reloaded

Date Filter File

```
!
!  JCC LogMiner Loader DateFilter Control File Template
!  Generated at 2017-07-25 16:50:18
!
FilterMap~EMPLOYEES~where \
COALESCE(BIRTHDAY,date vms'17-NOV-1858') between date vms'17-NOV-1858'
and date vms'31-DEC-9999'
...
```

- OpenVMS Date datatype supports dates well past the year 9999
- Date Formatting functions support 4 digit years
- Eliminate rows with bad dates
- Optional but useful for some environments

Datatypes

- Most Rdb datatypes have corresponding SQL Server Datatypes
- Some Rdb datatypes need to use different SQL Server datatypes
  - For Scaled Integers (INTEGER(M)) use
    - Number(N,M)
    - Money
  - For VMS Date use
    - Datetime

Procedure to put it together

MFP_JDBC_CLM_THRUD.COM – my naming

- Sets up environment
  - JCC_LML_USER and JCC_LML_JDBC_USER
  - Various Logical Names
- Invokes **JCC_RUN_CLM_LML** with 3 parameters
  - **MFP_V73 – Logical** name pointing to the source DB
  - **MF_PERSONNEL_LM_UNL.OPT** – Lists tables for RMU/UNLOAD/AFTER/Continuous to extract
  - **MF_PERSONNEL_JDBC_THRUD.INI** – main configuration file for the loader session. Includes additional files:
    - **MF_PERSONNEL_JDBC_TABLE.INI** – describe source
    - **MF_PERSONNEL_JDBC_MAPTABLE.INI** – maps to target db

DCL Procedure – MFP_JDBC_THRUD.COM

```
$ set verify
$ $ set defa jcc_root:[keith.sql_class.lml_sqlserver]
$ @jcc_tool_com:jcc_lml_user 3.5
$ jcc_lml_jdbc_user 6.0 sys$common:[java$60.com]java$60_setup.com
$ java –version
$ jcc_version
$ define jcc_lml_jdbc_batch_size 20
$ define decc$fd_locking true
$ define JCC_LML_JAVA_COMMAND_LINE " -Xmx256m -Xss1m"
$ define JCC_LogMiner_Loader_Name                KWH_THRUD
$ define JCC_LogMiner_Loader_HW_Response         CREATE
$ define loader_sqlsrv_jdbc_chkpt –
      []loader_sqlsrv_thrud_chkpt.JCCLML_CHECKPOINT
$ define JCC_LML_CASE_SENSITIVE_TARGET 1
$ define nls_lang "AMERICAN_AMERICA.WE8ISO8859P1"
$ define jcc_aij_backup_spec      -
       JCC_ROOT:[KEITH.SQL_CLASS.MF_V73.backups]*.aij;*
$ define JCC_ADD_CLM_IGNORE_OLD_VERSION_TABLES "*"
$ jcc_run_clm_lml source_db  MF_PERSONNEL_LM_UNL.OPT –
                             mf_personnel_jdbc_thrud.ini
```

Main Configuration File – MF_PERSONNEL_JDBC_THRUD.INI

```
logging~initialization
logging~statistics~runtime,timer,commit
logging~input~ignore_unknown_tables,log_restart
sort~by_record
input~ipc
checkpoint~25~lml_internal~asynch~loader_sqlsrv_jdbc_chkpt
input_failure~10
output~jdbc~synch~jdbc:jtds:sqlserver://thrud:1433;DatabaseName=Personnel
validation~test~<password>
jdbc~driver~net.sourceforge.jtds.jdbc.Driver
jdbc~classpath~/jcc_root/keith/sql_class/sqlserver/jtds/jtds-1-2-8.jar
output_failure~1~10
parallel~1~1~constrained
thread~startup~10
thread~shutdown~1
include_file~mf_personnel_jdbc_table.ini
include_file~mf_personnel_jdbc_maptable.ini
```

SQL Server Connect String

```
output~jdbc~synch~jdbc:jtds:sqlserver://
        thrud:1433;DatabaseName=Personnel
```

- jdbc:jtds:sqlserver: from jTDS documentation
- SQL Server installed on node Thrud
  - Windows 2016 Server in JCC internal network
- SQL Server default port: 1433
- Database name: Personnel
- Sidetrack – why is the server named Thrud?
  - Thor was our original SQL Server 2000 machine
  - In Norse Mythology, Thrud is the daughter of Thor

JDBC Driver and Classpath

- JDBC Driver

  `jdbc~driver~net.sourceforge.jtds.jdbc.Driver`

  - Driver specification is found on the jTDS web page

- JDBC Classpath

  `jdbc~classpath~/jcc_root/keith/sql_class/sqlserver/jtds/jtds-1-2-8.jar`

  - VMS filename is:

    `JCC_ROOT:[KEITH.SQL_CLASS.SQLSERVER.JTDS]jtds-1-2-8.jar`

Starting the Loader Session

- Submit the command procedure as a batch job

```
ares > submit/noprint/notify/log=[] MFP_JDBC_CLM_THRUD.COM
```

- Sets up three (or more) processes

```
ares > show sys/proc=*kwh_thrud*
OpenVMS V8.4-2L1  on node ARES   26-JUL-2017 14:26:50.77  Uptime  2 02:07:25
  Pid    Process Name    State  Pri    I/O       CPU       Page flts  Pages
006010DD CTL KWH_THRUD   LEF      6  81793   0 00:00:02.51    5834   1309  B
006010DE ||0 KWH_THRUD   LEF      6  11622   0 00:00:01.38    6634   9458  S
006010DF CLM KWH_THRUD   HIB      6   2369   0 00:00:00.34    4919   1619  S
ares >
```

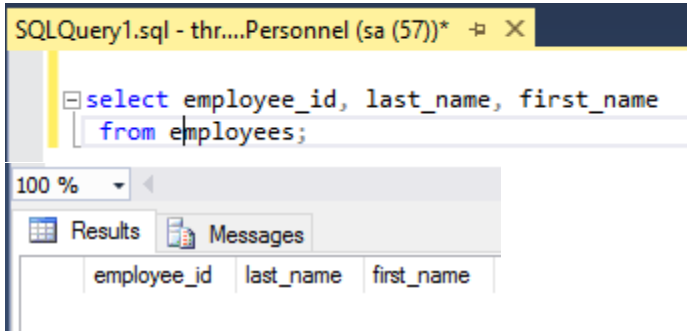
  - CTL process controls and logs
  - CLM process reads from the source DB
  - Thread (||0) process writes to the target DB

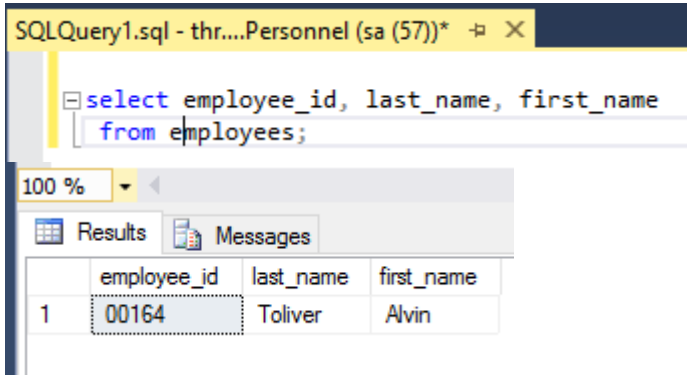
Loader Session Running

Rdb

```
SQL> select employee_id, last_name,
cont>   first_name
cont>  from employees
cont> where employee_id = '00164';
 EMPLOYEE_ID   LAST_NAME
FIRST_NAME
 00164         Toliver          Alvin
1 row selected
SQL> update employees
cont> set last_name = last_name
cont> where employee_id = '00164';
1 row updated
SQL> commit;
SQL>
```

SQL Server





Log Files

- Log files are created for each of the processes
  - MFP_JDBC_CLM_THRUD.LOG
  - JCC_TOOL_LOGS: jcc_run_clm-KWH_THRUD.log
  - JCC_TOOL_LOGS: jcc_run_lml-KWH_THRUD_0.log
- Contain logging information useful for understanding what happened
- Look for the first exception in a log
- JCC LML Support will usually ask for all logs
- Additional logging can be enabled

Monitoring Replication

- JCC_LML_STATISTICS Parameters
  - loader-family-name
  - [Refresh seconds]
  - [Type of logging]
    - brief|full|detail|csv|t4
  - [tardy threshold [operator class]]
- Example

```
ares > jcc_lml_statistics KWH_thrud 6 d
```

JCC_LML_STATISTICS Detail

```
Rate:   6.00                          KWH_THRUD              26-JUL-2017 14:52:00.08
=============================================================================================
   Input: 26-JUL-2017 14:51:46.52                Output: 26-JUL-2017 14:51:46.47
--[Trail:   13.56]--------------              ---[Trail:   13.61]--------------
Transactions                 29795            Checkpoints                   1197
Records                      79867               Timeout                       4
   Modify                    29872               BufferLimit(   250)           2
   Delete                    20200               NoWork                        0
   Commit                    29795            Records(   1)                50067
 Discarded                                        Messages(  N/A )             N/A
   Filtered                      0               Filtered                      0
   Excluded                      0            Failure                          0
   Unknown                       0            Timeout                          0
   Restart                       0         - Current ---------------- Ave/Second -
   NoWork                       20         Checkpoints             45          7.50
   Heartbeat                     0         Records               1125        187.47
Timeout                      7831         Rate                 56.99%
--- Restart Context ------               - Latency(sec) ------ LML detail ------
M|AIJ#                          9 |        CLM   11.99 |  Inpt   0.8%  Cnvt   2.5%
Q|VBN                    14281 |           ------------   Sort   0.0%  Trgt  96.7%
P|TSN                    32754 |           LML    0.13    Sync   0.0%  Ckpt   0.0%
 CTSN                    32754 |       - Loaders - 0 ------------------------------
  LSN                    29750 |       - States  - >
```

Statistics Description

- Input
  - 29,795 Source Transactions
  - 79,867 Records
    - 29,872 rows Modify (Insert, Update)
    - 20,200 rows Deleted
    - 29,795 Commit records
- Output
  - 1,197 Checkpoints (Commits on target)
  - 50,067 Records sent to target

JCC_LML_STATISTICS – Latency

- Bottom right corner of Detail display

```
          - Current ---------------- Ave/Second -
          Checkpoints              45          7.50
          Records                1125        187.47
          Rate                  56.99%
          - Latency(sec) ------ LML detail ------
          CLM   11.99 |  Inpt    0.8%  Cnvt    2.5%
          ------------   Sort    0.0%  Trgt   96.7%
          LML    0.13    Sync    0.0%  Ckpt    0.0%
    - Loaders - 0 ------------------------------
    - States  - >
```

- Current is snapshot of most recent interval
  - 187.47 Rows per second
  - Rate is 56.99% of realtime
- Latency tells where time is spent
  - CLM is 11.99 seconds behind real time
  - LML is 0.13 seconds behind CLM
    - 0.8% (0.00104 seconds) waiting for input
    - 96.7% (0.12571 seconds) waiting for the target
  - Need to understand what is happening in the target

But what about …?

- Shutdown and Restart
- Adding additional information
- Filtering data
- Transforming data
- Using multiple update threads
- Verifying data in target
- Additional targets
- Java V1.8

Shutdown

- jcc_clml_shutdown <loader-name>
- Example:

```
ares > jcc_clml_shutdown kwh_thrud
ares >
Job MFP_JDBC_CLM_THRUD (queue ARES_BATCH, entry 819) completed
ares >
```

Restart

- Restart information written to either:
  - High-water table (Rdb and Oracle OCI only)
  - Checkpoint file on source
- Example

```
ares > di *.*check*

Directory JCC_ROOT:[KEITH.SQL_CLASS.LML_SQLSERVER]

LOADER_SQLSRV_THRUD_CHKPT.JCCLML_CHECKPOINT;1
                    64/65      25-JUL-2017 16:32:27.05  (RWED,RWED,RE,)

Total of 1 file, 64/65 blocks.
ares >
```

- All committed transaction will be replicated
- Restart may resend some transactions

View Restart Information

- jcc_lml_dump_checkpoint <LoaderName> <name> [type]
  - <LoaderName> – LoaderName for the checkpoint
  - <name> – checkpoint filename or target database
  - [type] – checkpoint stream type
    - LML_INTERNAL (Default)
    - OCI
    - RDB

JCC_LML_DUMP_CHECKPOINT Example

```
ares > jcc_lml_dump_checkpoint kwh_thrud  -
_ares > LOADER_SQLSRV_THRUD_CHKPT.JCCLML_CHECKPOINT lml_internal

JCC LML Dump Checkpoint V03.05.00i2 (built 30-JUN-2017 15:05:35.05)

 -- Checkpoint restart information --

--- Parallel mode ---
Write Timestamp:     26-JUL-2017 15:01:22.00
LoaderName:          KWH_THRUD
Completion Flag:     S
Checkpoint Interval: 25
Input Data Source:   LML_CONT_KWH_THRUD
Last Transaction:
        Start Time:  26-JUL-2017 14:51:47.78
        Commit Time: 26-JUL-2017 14:51:47.78
        TSN:         33309
        LSN:         30305
        AERCP:       1-28-9-17779-33309-33309
        RM TID:

ares >
```

Adding additional information

- Virtual Columns
  - ORIGINATING_DBKEY
  - LOADER_SEQUENCE_NUMBER
  - TRANSACTION_START_TIME
  - TRANSACTION_COMMIT_TIME
  - PID – Source transaction OpenVMS Process ID
  - Etc.
- Virtual Tables
  - Map Commit record to target table

Filtering Data

- FilterMap Example

```
FilterMap~EMPLOYEES~where \
COALESCE(BIRTHDAY,date vms'17-NOV-1858') between date vms'17-NOV-1858'
and date vms'31-DEC-9999'
```

- FilterMap statements must be after MapTable statement
- Looks like Rdb SQL
  - Processed using dynamic SQL and FilterMap DB
- Can only reference current row

FilterMap Database

- Rdb database specifically for evaluating filters
- Default is jcc_tool_data:<loadername>.rdb
- Logical Names
  - JCC_LOGMINER_LOADER_FILTER_DIR – specifies Filter DB Location
  - JCC_LOGMINER_LOADER_FILTER_NAME – overrides the <loadername>
- Automatically created when first needed

FilterMap Database

- Create your own if the defaults are not sufficient for your environment
- Example

```
CREATE DATABASE ALIAS filter_db FILENAME jcc_tool_data:<loadername>.rdb
        OPEN IS AUTOMATIC
        NUMBER OF USERS 33
        NUMBER OF CLUSTER NODES 1
        PAGE SIZE IS 4 BLOCKS
        BUFFER SIZE IS 4 BLOCKS
        GLOBAL BUFFERS ARE ENABLED
               (NUMBER IS 330,USER LIMIT IS 10,PAGE TRANSFER VIA DISK)
        DBKEY SCOPE IS TRANSACTION
        PRESTARTED TRANSACTIONS ARE OFF  --(or ON if you wish)
        DICTIONARY IS NOT REQUIRED
        SHARED MEMORY IS SYSTEM;
```

Transforming data

- Map one source table to multiple targets
- Map multiple source tables to one target table
- Turn specific values into NULLs
- Turn NULLs into values
- What if we could add a function to the FilterMap database?
  - More on this in “Data Transformations Using JCC LogMiner Loader MapResult”

Multiple Update Threads

- Sometimes benefit in sending multiple threads of data to target
- Example

```
parallel~1~4~constrained
thread~startup~10
thread~shutdown~30
```

- 1 to 4 concurrent update threads
- Startup thread if data is available for more than 10 seconds while all threads are writing
- Shutdown thread if idle more than 30 seconds

Verifying Data in Target

Challenges comparing data in two databases

- Datatype representations
  - CHAR, VARCHAR – trailing spaces, character sets
  - Floating point – G-Float to IEEE precisions
  - Date, Time, Timestamp – precisions
- Data Volume – more rows, more time needed
- Freeze updates while doing a compare?
- For JCC LogMiner Loader Regression Testing
  - Random updates for a period of time
  - Freeze updates
  - Java code to copy target data to parallel tables in source DB
  - Rdb SQL queries to compare data

SQL Server JDBC and Unicode

- jTDS and SQL Server JDBC drivers, by default, convert character strings to Unicode
  - Performance issues if target primary key is character string and is not Unicode
  - EMPLOYEE_ID is a character string
- Use following JDBC URL to send non-unicode characters.

```
output~jdbc~synch~jdbc:jtds:sqlserver://thrud:1433
   ;DatabaseName=Personnel
   ;sendStringParametersAsUnicode=false
```

  - Example of sending parameters to the JDBC driver
  - Punctuation is driver specific
- Could also create target columns as Unicode.

Other targets and JDBC drivers may have similar quirks.

How much of a difference?

```
Rate:   6.00                        KWH_THRUD              27-JUL-2017 10:52:13.47
================================================================================
   Input: 27-JUL-2017 10:52:13.46               Output: 27-JUL-2017 10:52:13.42
--[Trail:    0.01]--------------             ---[Trail:    0.05]--------------
Transactions                15253             Checkpoints                   612
Records                     50698                Timeout                      2
   Modify                   15245                BufferLimit(   250)          2
   Delete                   20200                NoWork                       0
   Commit                   15253             Records(   1)                35422
 Discarded                                      Messages(  N/A )             N/A
   Filtered                     0                Filtered                     0
   Excluded                     0             Failure                         0
   Unknown                      0             Timeout                         0
   Restart                      4          - Current ---------------- Ave/Second -
   NoWork                       8          Checkpoints          139         23.16
   Heartbeat                    0          Records             3455        575.75
Timeout                        23          Rate                99.83%
--- Restart Context ------                 - Latency(sec) ------ LML detail ------
M|AIJ#                    15 |             CLM    0.00 |  Inpt  81.0%  Cnvt   4.8%
Q|VBN                  3102 |             -----------   Sort   0.0%  Trgt  14.1%
P|TSN                 50144 |             LML    0.04    Sync   0.0%  Ckpt   0.1%
 CTSN                 50144 |          - Loaders - 0 -------------------------------
  LSN                 45530 |          - States  - z
```

How much of a difference?

	sendStringParameters AsUnicode=true	sendStringParameters AsUnicode=false
Records/Second	187.47	575.75
Rate	56.99%	99.83%
CLM Latency	11.99 seconds	0.00 Seconds
LML Latency	0.13 Seconds	0.04
Time waiting for Input	0.8% (0.00104 Sec)	81.0% (0.03240 sec)
Time waiting for Output	96.7% (0.12571 Sec)	14.1% (0.00564 sec)

- Workload approximately the same
- Only change is not sending parameters as Unicode
- Near real time!

Additional targets

- Rdb targets are straightforward
- Oracle targets using Oracle Call Interface (OCI)
  - Install Oracle client on OpenVMS
  - Understand datatype differences
- JDBC Targets require a few more steps
  - Install Java on OpenVMS
  - JDBC Driver files to OpenVMS & Reset Jar File characteristics
  - Specify class path in Unix notation
  - Specify appropriate connection string
  - Understand datatype differences

But what about Cloud?

- Vendors offering Database As A Service
  - Microsoft Azure
  - Oracle Database Cloud Service
- Should be straightforward
  - Appropriate licensing
  - Encrypted tunnel
  - Set up test
- Possible future testing

What about Java 1.8?

- HPE & VMS Software Inc. released Java V1.8 in Q2 2017
- Java 1.8 requires a 64 bit interface
- For the JCC LogMiner Loader, this means:
  - New sharable image to instantiate JVM
  - Changes to some command procedures
  - Some changes to Java code
- We are currently testing Java 1.8 support in the next JCC LogMiner Loader release.

Blogs

- www.jcc.com/lml-blog includes a growing number of topics.
  - Managing LogMiner performance
  - Dangerous interaction between RMU AIJ Backup & LogMiner
  - Network Latency
  - … and others

Summary

- Discussed what the JCC LogMiner Loader accomplishes
- Detailed example of setting up replication from Rdb on OpenVMS to SQL Server 2016 on Windows Server 2016
  - First loader setup is the hardest
  - Other sessions and targets use similar steps

Learning More

- The product is The JCC LogMiner Loader.
  - Read about it at http://www.jcc.com/lml
  - Check out the blogs http://www.jcc.com/lml-blog
  - Ask for a temporary license
- Contact us at Info@JCC.com

Join the Conversation

Join the worldwide Rdb community. Send mail to OracleRdb-request@JCC.com.

Include “SUBSCRIBE” in the body of the message.