Data Transformations Using JCC LogMiner Loader MapResult

Tom Musson, Keith Hare,
Jeff Jalbert, Cheryl Jalbert

JCC Consulting, Inc.

Abstract

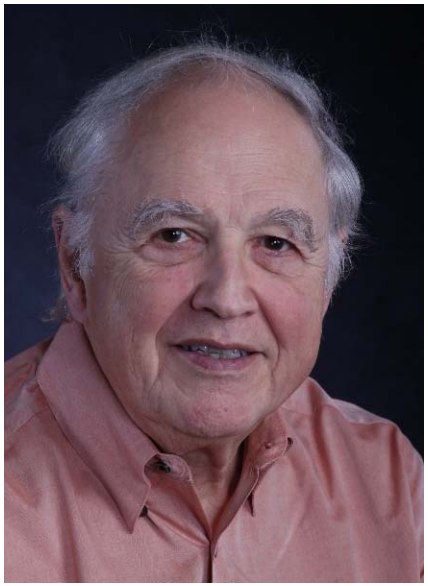
JCC LogMiner Loader V3.5 supports a new control file directive MapResult. MapResult directives allow data to be transformed using both built-in and user defined Rdb functions.

This session provides examples of how this capability can be used and discusses the performance implications.

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
Development



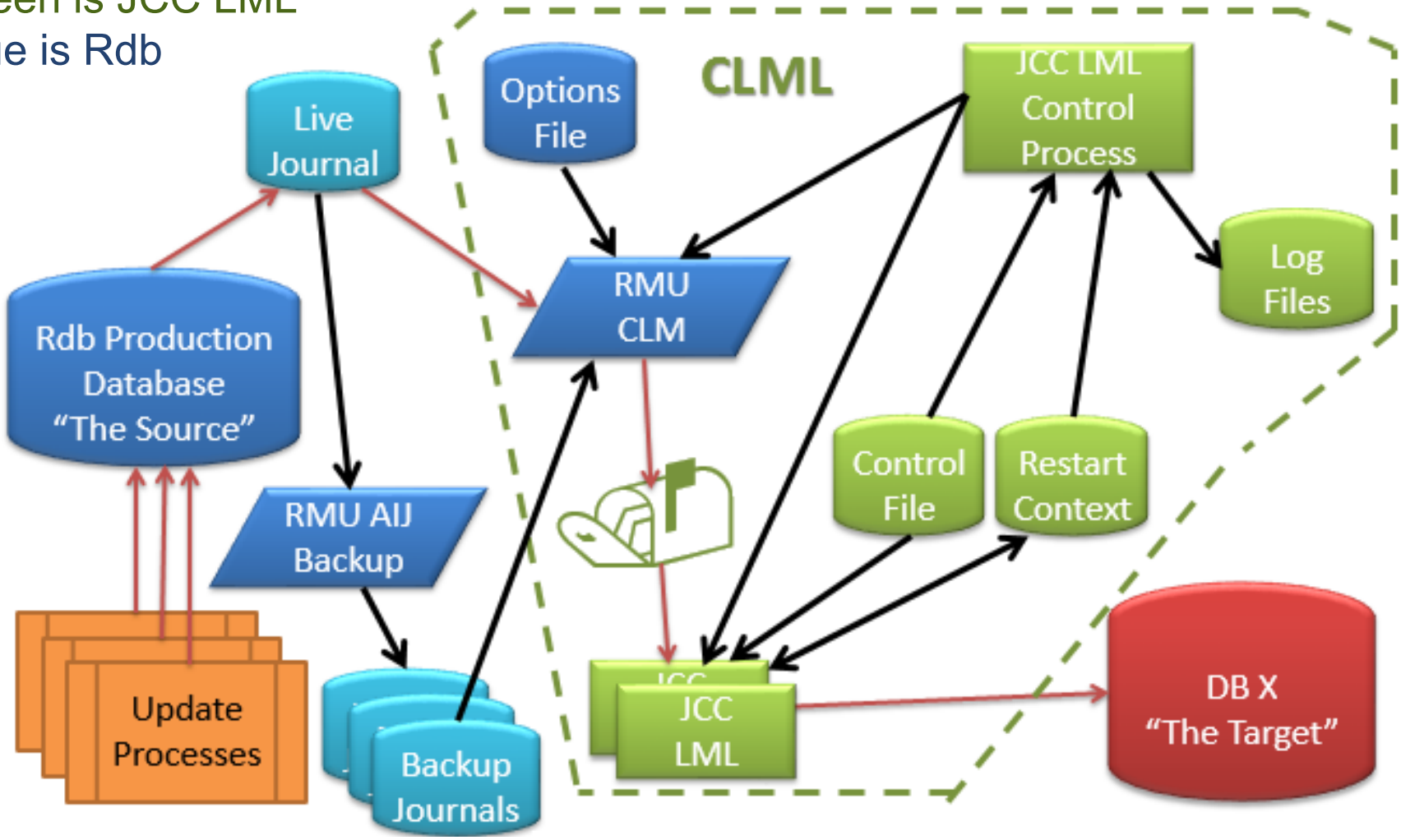
Jeff Jalbert
Testing

Cheryl Jalbert
Documentation

Keith Hare
Installation

JCC LogMiner Loader

Green is JCC LML
Blue is Rdb



Agenda

- Mapping Source to Target
- Metadata changes in the Source
- Metadata changes in the Target
- Data transforms and MapResult

Mapping

- For most uses the default mapping is sufficient
  - Source table X -> Target table X
  - Source column X.Y -> Target column X.Y
  - Kit procedures generate
- Generally not what this talk is about
  - See documentation for more details

Source Metadata Changes

- Need for Source database changes are driven by the business
- Included here for completeness

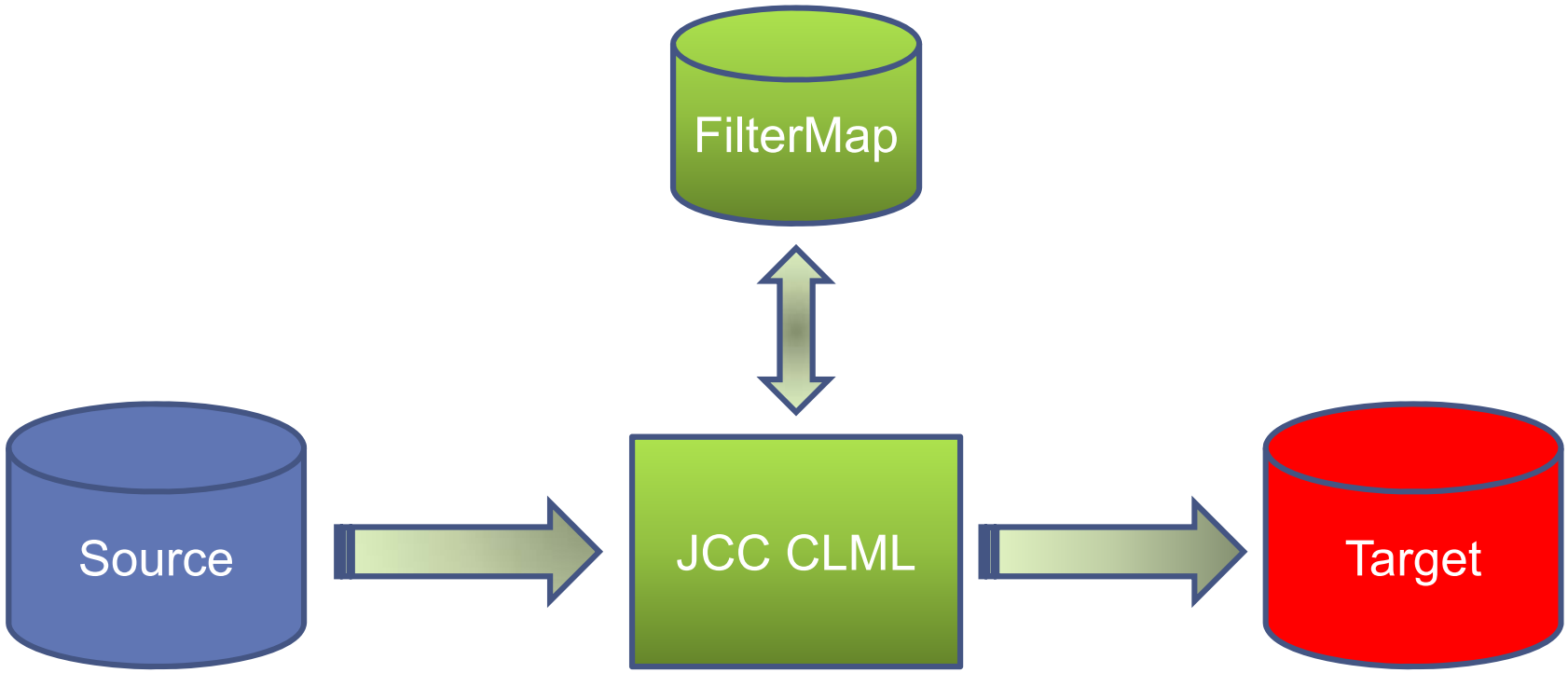
- JCC LogMiner Loader
  - Stop Source updates
  - Shutdown Loader after all updates replicated
  - Make Source metadata changes
  - Generate new Source TABLE control file

Target Metadata Changes

- Need for Target database changes are driven by the business
- Included here for completeness

- JCC LogMiner Loader
  - Stop Source updates
  - Shutdown Loader after all updates replicated
  - Make Target metadata changes
  - Modify Target MAPTABLE control file

FilterMap Database



MapResult

- Provides more complex capabilities

- Syntax

```
MapResult                          \
      ~ <MapTable name>            \
      ~ <column name>              \
      ~ <sql expression>
```

So Many Slides, So Little Time

- Using Oracle Rdb Sample Personnel database
- Setup presentation can be downloaded at

  http://jcc.com/dl/lml_setup_example

JCC LogMiner Loader Example

- Source Database
  - Oracle Rdb V7.3-210
  - OpenVMS IA64 V8.4-2L1
  - Itanium RX2800, 8 CPU, 128 GB memory

- Target Database…

Targets

- Rdb target for these examples
- Oracle targets
- JDBC Targets increases possibilities
- API or FILE XML documents

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

- Modify data
- Generate new data

Example #1 – Simple

- Use Rdb built-in functions
- CANDIDATES table
- No new columns for target database
  - Uppercase LAST_NAME
  - Lowercase FIRST_NAME

Wait…What?

- Oracle Rdb to Oracle Rdb
  - Why use an Oracle Rdb built-in function in the Loader?
  - Could create a Computed By or an Automatic column on the target
  - Adds latency to the replication
- Flexibility
  - Not all Targets support Computed By or Automatic columns
- Functionality
  - Some data cannot be sent to target for security reasons
  - Trivial examples for demonstration purposes only

Implementation Steps

- Shutdown Loader THMRDB
  `Ares> jcc_clml_shutdown thmrdb`
- Modify MAPTABLE.INI changed column(s)
- Restart Loader
  `Ares> submit/noti/nopri/keep/log=[] CLML_THMRDB.COM`
- Update source data to populate new column(s)

Target Modifications

- In the MAPTABLE.INI

```
MapTable~CANDIDATES~CANDIDATES,candidates~Replicate
! 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
!
MapResult~CANDIDATES~FIRST_NAME~ LOWER(FIRST_NAME)
MapResult~CANDIDATES~LAST_NAME~  UPPER(LAST_NAME)
```

MapResult In Action

- Rdb source database update:

```
SQL> SELECT LAST_NAME, FIRST_NAME FROM CANDIDATES;
 LAST_NAME         FIRST_NAME
 Wilson            Oscar
 Schwartz          Trixie
 Boswick           Fred
3 rows selected
SQL> update candidates set middle_initial = middle_initial;
3 rows updated
SQL> commit;
```

- Target database query

```
SQL> SELECT LAST_NAME, FIRST_NAME FROM CANDIDATES;
 LAST_NAME         FIRST_NAME
 WILSON            oscar
 SCHWARTZ          trixie
 BOSWICK           fred
3 rows selected
```

Example #2 – Synthesize Data

- Call user defined function
- EMPLOYEES table
- Problem with sort order for reporting
  - “D'Amico” sorted before “Dallas”
  - Different languages have different sort order
    - Sample code is one sorting methodology
- Add SORT_NAME

Implementation Steps

- Shutdown Loader THMRDB
  `Ares> jcc_clml_shutdown thmrdb`

- Modify MAPTABLE.INI to add new column(s)
- Modify Target table(s) (as necessary)
- Modify FilterMap database (as necessary)

- Restart Loader
  `Ares> submit/noti/nopri/keep/log=[] CLML_THMRDB.COM`

- Update source data to populate new column(s)

Target Modifications

- In the MAPTABLE.INI

```
MapTable~EMPLOYEES~EMPLOYEES~Replicate
MapColumn~EMPLOYEES~EMPLOYEE_ID
MapColumn~EMPLOYEES~LAST_NAME
MapColumn~EMPLOYEES~FIRST_NAME
MapColumn~EMPLOYEES~MIDDLE_INITIAL
MapColumn~EMPLOYEES~ADDRESS_DATA_1
...
MapColumn~EMPLOYEES~STATUS_CODE
MapKey~EMPLOYEES~EMPLOYEE_ID
!
MapResult~EMPLOYEES~SORT_NAME~                                    \
          jcc$sort_name(FIRST_NAME,MIDDLE_INITIAL,LAST_NAME)
```

- In Target

```
alter table EMPLOYEES
  add SORT_NAME CHAR(25);
```

FilterMap Database Modifications

```
SQL> create domain jcc$text32 varchar(32);
SQL> ...
SQL> create module JCC$UTILITIES
cont>     language SQL
cont>     ...
cont>     function jcc$sort_name (:inFirst jcc$text32, :inInitial jcc$text32,:inLast jcc$text32)
cont>   returns jcc$text32
cont>   deterministic;
cont>     begin
cont>
cont>     return(trim(translate(COALESCE(:inLast,'')
cont>       ,'AaEeHhIiOoUuWwYyBbFfPpVvCcGgJjKkQqSsXxZzDdTtLlMmNnRr'' -~!@#$%^&*()_+={}[]|\/?,.<>:;"`'
cont>       ,'AAEEHHIIOOUUWWYYBBFFPPVVCCGGJJKKQQSSXXZZDDTTLLMMNNRR'))
cont>    || ' '
cont>    || trim(translate(COALESCE(:inFirst,'')
cont>       ,'AaEeHhIiOoUuWwYyBbFfPpVvCcGgJjKkQqSsXxZzDdTtLlMmNnRr'' -~!@#$%^&*()_+={}[]|\/?,.<>:;"`'
cont>       ,'AAEEHHIIOOUUWWYYBBFFPPVVCCGGJJKKQQSSXXZZDDTTLLMMNNRR'))
cont>    || ' '
cont>    || trim(translate(COALESCE(:inInitial,'')
cont>       ,'AaEeHhIiOoUuWwYyBbFfPpVvCcGgJjKkQqSsXxZzDdTtLlMmNnRr'' -~!@#$%^&*()_+={}[]|\/?,.<>:;"`'
cont>       ,'AAEEHHIIOOUUWWYYBBFFPPVVCCGGJJKKQQSSXXZZDDTTLLMMNNRR'))
cont>           );
cont>
cont>     end;
cont>   ...
cont> end module;
SQL> grant execute on module JCC$UTILITIES to public;
SQL> commit;
```

MapResult In Action

- Rdb source database update:

```
SQL> update EMPLOYEES set middle_initial = middle_initial;
100 rows updated
SQL> commit;
SQL> select LAST_NAME,FIRST_NAME,MIDDLE_INITIAL from EMPLOYEES
cont> where last_name starting with 'D'
cont> order by LAST_NAME,FIRST_NAME,MIDDLE_INITIAL limit to 3 rows;
 LAST_NAME        FIRST_NAME   MIDDLE_INITIAL
 D'Amico          Aruwa        NULL
 Dallas           Meg          NULL
 Danzig           Chris        NULL
3 rows selected
```

- Target database query

```
SQL> select LAST_NAME,FIRST_NAME,MIDDLE_INITIAL,SORT_NAME from EMPLOYEES
cont> where last_name starting with 'D'
cont> order by SORT_NAME limit to 3 rows;
 LAST_NAME        FIRST_NAME   MIDDLE_INITIAL   SORT_NAME
 Dallas           Meg          NULL             DALLAS MEG
 D'Amico          Aruwa        NULL             DAMICO ARUWA
 Danzig           Chris        NULL             DANZIG CHRIS
3 rows selected
```

Example #3 – Synthesize Data

- Call user defined function
- EMPLOYEES table
- Problem looking up employees by name
  - Correct spelling necessary for exact match lookup
- Add SOUNDEX_NAME

Implementation Steps

- Shutdown Loader THMRDB
  `Ares> jcc_clml_shutdown thmrdb`

- Modify MAPTABLE.INI to add new column(s)
- Modify Target table(s) (as necessary)
- Modify FilterMap database (as necessary)

- Restart Loader
  `Ares> submit/noti/nopri/keep/log=[] CLML_THMRDB.COM`

- Update source data to populate new column(s)

Target Modifications

- In the MAPTABLE.INI

```
MapTable~EMPLOYEES~EMPLOYEES~Replicate
MapColumn~EMPLOYEES~EMPLOYEE_ID
MapColumn~EMPLOYEES~LAST_NAME
MapColumn~EMPLOYEES~FIRST_NAME
MapColumn~EMPLOYEES~MIDDLE_INITIAL
MapColumn~EMPLOYEES~ADDRESS_DATA_1
...
MapColumn~EMPLOYEES~STATUS_CODE
MapKey~EMPLOYEES~EMPLOYEE_ID
!
MapResult~EMPLOYEES~SOUNDEX_NAME~ jcc$soundex_lite(LAST_NAME)
```

- In Target

```
alter table EMPLOYEES
  add SOUNDEX_NAME CHAR(4);
```

FilterMap Database Modifications

```
create domain jcc$sdex char(4);
SQL> ...
SQL> create module JCC$UTILITIES
cont>     language SQL
cont>     ...
cont>     function jcc$soundex_lite (:instring jcc$text255)
cont>     returns jcc$sdex
cont>     deterministic;
cont>     begin
cont>     declare :retcode jcc$sdex = 'Z000';
cont>     declare :sdex jcc$text255;
cont>     declare :firstchar, :tstchr char(1);
cont>     declare :st, :en integer;
cont>
cont>   set :sdex = translate(
cont>        :instring
cont>       ,'AaEeHhIiOoUuWwYyBbFfPpVvCcGgJjKkQqSsXxZzDdTtLlMmNnRr'' -~!@#$%^&*()_+={}[]|\/?,.<>:;"`'
cont>       ,'AaEeHhIiOoUuWwYyBbFfPpVvCcGgJjKkQqSsXxZzDdTtLlMmNnRr');
cont>   if (length(:sdex) > 0)
cont>   then
cont>       set :firstchar = substring(:sdex from 1 for 1 using characters);
cont>   ...
cont>       set :sdex = translate(substring(:sdex from 2 for length(:sdex)), '12345609'
cont>                                                                     , '123456');
cont>       set :retcode = substring(:firstchar||:sdex||'000' from 1 for 4);
cont>   end if;
cont>   return(:retcode);
cont>   ...
cont> end module;
SQL> grant execute on module JCC$UTILITIES to public;
SQL> commit;
```

MapResult In Action

- Rdb source database update:

```
SQL> update EMPLOYEES set middle_initial = middle_initial;
100 rows updated
SQL> commit;
SQL> select select EMPLOYEE_ID, LAST_NAME from EMPLOYEES
cont> where last_name starting with 'D'
cont> order by EMPLOYEE_ID limit to 3 rows;
EMPLOYEE_ID   LAST_NAME
 00166         Dietrich
 00171         D'Amico
 00179         Dallas
3 rows selected
```

- Target database query

```
SQL> select EMPLOYEE_ID, LAST_NAME, SOUNDEX_NAME from EMPLOYEES
cont> where last_name starting with 'D'
cont> order by EMPLOYEE_ID limit to 3 rows;
 EMPLOYEE_ID   LAST_NAME         SOUNDEX_NAME
 00166         Dietrich          D362
 00171         D'Amico           D520
 00179         Dallas            D420
3 rows selected
```

Example #4 – Translate Data

- CANDIDATES table requires special attention
- ORIGINATING_DBKEY column in target is CHAR(8) containing binary data
  - Used as Loader MapKey
  - Determining Source database dbkey requires a program to read and format
  - Need easier way to find row in source database that matches the target
- Add formatted TEXT_DBKEY to target

Implementation Steps

- Shutdown Loader THMRDB
  `Ares> jcc_clml_shutdown thmrdb`

- Modify MAPTABLE.INI to add new column(s)
- Modify Target table(s) (as necessary)
- Modify FilterMap database (as necessary)

- Restart Loader
  `Ares> submit/noti/nopri/keep/log=[] CLML_THMRDB.COM`

- Update source data to populate new column(s)

Target Modifications

- In the MAPTABLE.INI

```
MapTable~CANDIDATES~CANDIDATES,candidates~Replicate
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
!
...
MapResult~CANDIDATES~text_dbkey~ jcc$dbkey2text(originating_dbkey)
```

- In target

```
alter table candidates
  add TEXT_DBKEY CHAR(40);
```

FilterMap Database Modifications

```
SQL> create domain jcc$text255 varchar(255);
SQL> create domain jcc$text32 varchar(32);
SQL> ...
SQL> create procedure quad2dbkeytext (
cont>     in      :dbk   bigint by reference,
cont>     inout   :text  char(255) by reference)
cont>     language SQL;
cont>     external
cont>         name dbkey2text
cont>         location 'jcc_tool_local:jcc$utiities.exe'
cont>             with ALL logical_name translation
cont>         language C
cont>         GENERAL parameter style
cont>     comment is 'Return the text translation of a binary dbkey';
SQL> ...
SQL> create module JCC$UTILITIES
cont>     language SQL
cont>     ...
cont>     function jcc$dbkey2text (in :inbigint bigint)
cont>     returns jcc$text32
cont>     variant
cont>     comment is 'This is a functional wrapper which returns'
cont>     /           'the result as a string';
cont>     begin
cont>     declare :res jcc$text32 = ' ';
cont>
cont>         call quad2dbkeytext (:inbigint,:res);
cont>         return TRIM (:res);
cont>   ...
cont> end module;
SQL> grant execute on module JCC$UTILITIES to public;
SQL> commit;
```

External Function dbkey2text

```
Ares> type JCC$UTILITIES.C
...
typedef struct _DBKEY {
    unsigned short pline;
    unsigned int   ppage;
    unsigned short larea;
} DBKEY;
...
int dbkey2text(DBKEY *,char *);
int decompressdbkey(DBKEY *,DBKEY *);
...
int dbkey2text(inDBKey, outText)
DBKEY *inDBKey;
char *outText;
{
DBKEY dbkey;
int ret;

    ret = decompressdbkey(inDBKey, &dbkey);
    sprintf(outText,"%hu:%u:%hu",dbkey.larea,dbkey.ppage,dbkey.pline);

    return(1);
}
...

Ares> cc JCC$UTILITIES.C
Ares> link/nodeb/share/exe=jcc_tool_local:jcc$utilities.exe JCC$UTILITIES.opt/opt
Ares> shareable_install jcc_tool_local jcc$utilities.exe
```

MapResult In Action

- Rdb source database update:

```
SQL> SELECT LAST_NAME, DBKEY FROM CANDIDATES;
 LAST_NAME                          DBKEY
 Wilson                          90:634:0
 Schwartz                        90:634:1
 Boswick                         90:634:2
3 rows selected
SQL> update candidates set middle_initial = middle_initial;
3 rows updated
SQL> commit;
```

- Target database query

```
SQL> SELECT LAST_NAME, TEXT_DBKEY FROM CANDIDATES;
 LAST_NAME          TEXT_DBKEY
 WILSON             90:634:0
 SCHWARTZ           90:634:1
 BOSWICK            90:634:2
3 rows selected
```

Example #5 – Compress Data

- DEGREES table
  - Originating_dbkey used for Loader MapKey
  - (Hypothetically) large cardinality
  - Compression enabled
- Manipulate Rdb dbkey value
  - Highly compressible
  - Completely reversible
- Add cdbkey
- Change MapKey to cdbkey

Implementation Steps

- Shutdown Loader THMRDB
  `Ares> jcc_clml_shutdown thmrdb`

- Modify MAPTABLE.INI to add new column(s)
- Modify Target table(s) (as necessary)
- Modify FilterMap database (as necessary)

- Restart Loader
  `Ares> submit/noti/nopri/keep/log=[] CLML_THMRDB.COM`

- Update source data to populate new column(s)

Target Modifications

- In the MAPTABLE.INI

```
MapTable~DEGREES~DEGREES~Replicate
MapColumn~DEGREES~EMPLOYEE_ID
MapColumn~DEGREES~COLLEGE_CODE
MapColumn~DEGREES~YEAR_GIVEN
MapColumn~DEGREES~DEGREE
MapColumn~DEGREES~DEGREE_FIELD
!Originating_DBKey_column_required_due_to_no_key.
! MapColumn~DEGREES~originating_dbkey
! MapKey~DEGREES~originating_dbkey
!
MapResult~DEGREES~cdbkey~  jcc$compressdbkey(ORIGINATING_DBKEY)
MapKey~DEGREES~cdbkey
```

Target Modifications

- In Target

```
truncate table degrees;

drop index JCC_ODBKEY_003;

alter table degrees
  drop column ORIGINATING_DBKEY
  add cdbkey BIGINT;

create unique index jcc_odbkey_003
    on degrees (cdbkey)
    type is sorted
    node size 1000
    store in jcc_odbkey_u;
```

FilterMap Database Modifications

```
SQL> create procedure compressdbkey (
cont>     in      :dbk   bigint by reference,
cont>     inout   :cdbk  bigint by reference)
cont>     language SQL;
cont>     external
cont>         name compressdbkey
cont>         location 'jcc_tool_local:jcc$utilities.exe'
cont>             with ALL logical_name translation
cont>         language C
cont>         GENERAL parameter style
cont>     comment is 'Return a compressed binary dbkey';
SQL> ...
SQL> create module JCC$UTILITIES
cont>     language SQL
cont>     ...
cont>     function jcc$compressdbkey (in :indbkey bigint)
cont>     returns bigint
cont>     variant
cont>     comment is 'This is a functional wrapper which returns'
cont>     /           'the result as a bigint';
cont>     begin
cont>     declare :res bigint;
cont>
cont>         call compressdbkey(:indbkey,:res);
cont>         return TRIM (:res);
cont>   ...
cont> end module;
SQL> grant execute on module JCC$UTILITIES to public;
SQL> commit;
```

External Function compressdbkey

```
Ares> type JCC$UTILITIES.C
...
typedef struct _DBKEY {
    unsigned short pline;
    unsigned int   ppage;
    unsigned short larea;
} DBKEY;
...
int compressdbkey(DBKEY *, compressDBKEY *);
...
int compressdbkey(DBKEY *inDBKey, compressDBKEY *outDBKey)
{
    memcpy((char *)outDBKey,(char *)inDBKey, sizeof (struct _DBKEY));
    if (outDBKey->area.ca.cmpmbz == 0 && outDBKey->line.cl.clarea == 0
     && outDBKey->page.cp.cparea == 0)
    {
        outDBKey->line.cl.clarea = outDBKey->area.ca.clarea;
        outDBKey->page.cp.cparea = outDBKey->area.ca.cparea;
        outDBKey->area.ca.clarea = 0;
        outDBKey->area.ca.cparea = 0;
    }

    return(1);
}
...

Ares> cc JCC$UTILITIES.C
Ares> link/nodeb/share/exe=jcc_tool_local:jcc$utilities.exe JCC$UTILITIES.opt/opt
Ares> shareable_install jcc_tool_local jcc$utilities.exe
```

MapResult In Action

- Rdb source database update:

```
SQL> update degrees set college_code = college_code;
165 rows updated
SQL> commit;
SQL> SELECT EMPLOYEE_ID, dbkey from degrees
cont> order by EMPLOYEE_ID, YEAR_GIVEN limit to 3 rows;
 EMPLOYEE_ID                    DBKEY
 00164                        93:2:19
 00164                        93:2:20
 00165                        93:2:21
3 rows selected
```

- Target database query

```
SQL> SELECT EMPLOYEE_ID, cdbkey from degrees
cont> order by EMPLOYEE_ID, YEAR_GIVEN limit to 3 rows;
 EMPLOYEE_ID                     CDBKEY
 00164                           154899
 00164                           154900
 00165                           154901
3 rows selected
```

Example #6 – Sensitive Information

- EMPLOYEES table needs to include social security number
- New column SSN is sensitive

- Target is not secure
  - Accessible via public website
  - Accessible to 3rd party partners

- Data breeches in the U.S.A.
- New regulations in the E.U. (e.g. http://www.eugdpr.org/)

Implementation Steps

- Shutdown Loader THMRDB
  `Ares> jcc_clml_shutdown thmrdb`

- Modify Source table(s) (as necessary)
- Generate new TABLE.INI (as necessary)
- Modify MAPTABLE.INI to add new column(s)
- Modify Target table(s) (as necessary)
- Modify FilterMap database (as necessary)

- Restart Loader
  `Ares> submit/noti/nopri/keep/log=[] CLML_THMRDB.COM`

- Update source data to populate new column(s)

Source Modifications

- In Source

```
alter table EMPLOYEES
  add SSN CHAR(9);
```

- Generate TABLE.INI

```
$ JCC_LML_CREATE_CONTROL_FILE mfp_db TABLE
```

- New Employees description

```
!
Table~EMPLOYEES~2~~NoMapTable
Column~EMPLOYEES~EMPLOYEE_ID~1~5~0~14~0
Column~EMPLOYEES~LAST_NAME~2~14~0~14~0
Column~EMPLOYEES~FIRST_NAME~3~10~0~14~0
Column~EMPLOYEES~MIDDLE_INITIAL~4~1~0~14~0
Column~EMPLOYEES~ADDRESS_DATA_1~5~25~0~14~0
...
Column~EMPLOYEES~STATUS_CODE~12~1~0~14~0
Column~EMPLOYEES~SSN~13~9~0~14~0
!
```

Target Modifications

- In the MAPTABLE.INI

```
MapTable~EMPLOYEES~EMPLOYEES~Replicate
MapColumn~EMPLOYEES~EMPLOYEE_ID
MapColumn~EMPLOYEES~LAST_NAME
MapColumn~EMPLOYEES~FIRST_NAME
MapColumn~EMPLOYEES~MIDDLE_INITIAL
MapColumn~EMPLOYEES~ADDRESS_DATA_1
...
MapColumn~EMPLOYEES~STATUS_CODE
MapKey~EMPLOYEES~EMPLOYEE_ID
!
MapResult~EMPLOYEES~SSN~  jcc$redactssn(SSN)
```

- In Target

```
alter table EMPLOYEES
  add SSN CHAR(9);
```

FilterMap Database Modifications

```
SQL> create domain jcc$ssn char(11);
SQL> ...
SQL> create procedure redactssn (
cont>     in      :ssn   jcc$ssn by reference,
cont>     inout   :rssn  jcc$ssn by reference)
cont>     language SQL;
cont>     external
cont>         name redactssn
cont>         location 'jcc_tool_local:jcc$utilities.exe'
cont>             with ALL logical_name translation
cont>         language C
cont>         GENERAL parameter style
cont>     comment is 'Return a redacted SSN';
SQL> ...
SQL> create module JCC$UTILITIES
cont>     language SQL
cont>     ...
cont>     function jcc$redactssn (in :inssn jcc$ssn)
cont>     returns jcc$ssn
cont>     variant
cont>     comment is 'This is a functional wrapper which returns'
cont>     /           'the redacted SSN';
cont>     begin
cont>     declare :res jcc$ssn;
cont>
cont>         call redactssn(:inssn,:res);
cont>         return(:res);
cont>   ...
cont> end module;
SQL> grant execute on module JCC$UTILITIES to public;
SQL> commit;
```

External Function redactssn

```
Ares> type JCC$UTILITIES.C
...
int redactssn(char *,char *);
...
int redactssn(char *inSSN, char *outRSSN)
{
int inLen = 0, maskLen = 0;
char maskString[16] = "XXXXXXXXXXXXXXX";

    for (inLen=0; isdigit(inSSN[inLen]) || inSSN[inLen] == '-'; inLen++) ;
    if (inLen == 9)       strcpy(maskString,"XXXXX");
    else if (inLen == 10) strcpy(maskString,"XX-XXX");
    else if (inLen == 11) strcpy(maskString,"XXX-XX-");
    if (inLen >= 9 && inLen <= 11) {
        maskLen = inLen - 4;
        sprintf(outRSSN,"%*.*s%4.4s",maskLen,maskLen,maskString,&inSSN[maskLen]);
    } else
        sprintf(outRSSN,"BADFORMAT");

    return(1);
}
...

Ares> cc JCC$UTILITIES.C
Ares> link/nodeb/share/exe=jcc_tool_local:jcc$utilities.exe JCC$UTILITIES.opt/opt
Ares> shareable_install jcc_tool_local jcc$utilities.exe
```

MapResult In Action

- Rdb source database update:

```
SQL> update EMPLOYEES set SSN = '1234'||EMPLOYEE_ID;
26 rows updated
SQL> commit;
SQL> select EMPLOYEE_ID, SSN from EMPLOYEES
cont> order by EMPLOYEE_ID limit to 3 rows;
 EMPLOYEE_ID   SSN
 00164         123400164
 00165         123400165
 00166         123400166
3 rows selected
```

- Target database query

```
SQL> select EMPLOYEE_ID, SSN from EMPLOYEES
cont> order by EMPLOYEE_ID limit to 3 rows;
 EMPLOYEE_ID   SSN
 00164         XXXXX0164
 00165         XXXXX0165
 00166         XXXXX0166
3 rows selected
```

Example #7 – Sensitive Information

- DEPARTMENTS table needs to include the department credit card number
- New column DEPT_CREDIT_CARD is sensitive

- Target is not secure
  - Accessible via public website
  - Accessible to 3rd party partners

- Data breeches in the U.S.A.
- New regulations in the E.U. (e.g. http://www.eugdpr.org/)

Implementation Steps

- Shutdown Loader THMRDB
  `Ares> jcc_clml_shutdown thmrdb`

- Modify Source table(s) (as necessary)
- Generate new TABLE.INI (as necessary)
- Modify MAPTABLE.INI to add new column(s)
- Modify Target table(s) (as necessary)
- Modify FilterMap database (as necessary)

- Restart Loader
  `Ares> submit/noti/nopri/keep/log=[] CLML_THMRDB.COM`

- Update source data to populate new column(s)

Source Modifications

- In Source

```
alter table DEPARTMENTS
  add DEPT_CREDIT_CARD CHAR(16);
```

- Generate TABLE.INI

```
$ JCC_LML_CREATE_CONTROL_FILE mfp_db TABLE
```

- New Departments description

```
!
Table~DEPARTMENTS~2~~NoMapTable
Column~DEPARTMENTS~DEPARTMENT_CODE~1~4~0~14~0
Column~DEPARTMENTS~DEPARTMENT_NAME~2~30~0~14~0
Column~DEPARTMENTS~MANAGER_ID~3~5~0~14~0
Column~DEPARTMENTS~BUDGET_PROJECTED~4~4~0~8~0
Column~DEPARTMENTS~BUDGET_ACTUAL~5~4~0~8~0
Column~DEPARTMENTS~DEPT_CREDIT_CARD~6~16~0~14~0
!
```

Target Modifications

- In the MAPTABLE.INI

```
MapTable~DEPARTMENTS~DEPARTMENTS~Replicate
MapColumn~DEPARTMENTS~DEPARTMENT_CODE
MapColumn~DEPARTMENTS~DEPARTMENT_NAME
MapColumn~DEPARTMENTS~MANAGER_ID
MapColumn~DEPARTMENTS~BUDGET_PROJECTED
MapColumn~DEPARTMENTS~BUDGET_ACTUAL
MapKey~DEPARTMENTS~DEPARTMENT_CODE
!
MapResult~DEPARTMENTS~DEPT_CREDIT_CARD~ jcc$redactccn(DEPT_CREDIT_CARD)
```

- In Target

```
alter table DEPARTMENTS
  add DEPT_CREDIT_CARD CHAR(16);
```

FilterMap Database Modifications

```
SQL> create domain jcc$ccn char(19);
SQL> ...
SQL> create procedure redactccn (
cont>     in      :ccn   jcc$ccn by reference,
cont>     inout   :rccn  jcc$ccn by reference)
cont>     language SQL;
cont>     external
cont>         name redactccn
cont>         location 'jcc_tool_local:jcc$utilities.exe'
cont>             with ALL logical_name translation
cont>         language C
cont>         GENERAL parameter style
cont>     comment is 'Return a redacted CCN';
SQL> ...
SQL> create module JCC$UTILITIES
cont>     language SQL
cont>     ...
cont>     function jcc$redactccn (in :inccn jcc$ccn)
cont>     returns jcc$ccn
cont>     variant
cont>     comment is 'This is a functional wrapper which returns'
cont>     /           'the redacted CCN';
cont>     begin
cont>     declare :res jcc$ccn;
cont>
cont>         call redactccn(:inccn,:res);
cont>         return(:res);
cont>   ...
cont> end module;
SQL> grant execute on module JCC$UTILITIES to public;
SQL> commit;
```

Custom Function redactccn

```
Ares> type JCC$UTILITIES.C
...
int redactccn(char *,char *);
...
int redactccn(inCCN, outRCCN)
char *inCCN;
char *outRCCN;
{
int inLen;
int maskLen;

    for (inLen=0; isdigit(inCCN[inLen]) || inCCN[inLen] == '-'; inLen++) ;
    if (inLen >= 12 && inLen <= 19) {
        maskLen = inLen - 4;
        sprintf(outRCCN,"%*.*s%4.4s",maskLen,maskLen,"XXXXXXXXXXXXXXXXXX",&inCCN[maskLen]);
    } else
        strcpy(outRCCN,"INVALID");

    return(1);
}
...

Ares> cc JCC$UTILITIES.C
Ares> link/nodeb/share/exe=jcc_tool_local:jcc$utilities.exe JCC$UTILITIES.opt/opt
Ares> shareable_install jcc_tool_local jcc$utilities.exe
```

MapResult In Action

- Rdb source database update:

```
SQL> update DEPARTMENTS set DEPT_CREDIT_CARD = '12345678901'||MANAGER_ID;
26 rows updated
SQL> commit;
SQL> select DEPARTMENT_CODE, DEPT_CREDIT_CARD from DEPARTMENTS
cont> order by DEPARTMENT_CODE limit to 3 rows;
DEPARTMENT_CODE   DEPT_CREDIT_CARD
 ADMN              1234567890100225
 ELEL              1234567890100188
 ELGS              1234567890100369
3 rows selected
```

- Target database query

```
SQL> select DEPARTMENT_CODE, DEPT_CREDIT_CARD from DEPARTMENTS
cont> order by DEPARTMENT_CODE limit to 3 rows;
 DEPARTMENT_CODE   DEPT_CREDIT_CARD
 ADMN              XXXXXXXXXXXX0225
 ELEL              XXXXXXXXXXXX0188
 ELGS              XXXXXXXXXXXX0369
3 rows selected
```

Example #8 – Sensitive Information

- CANDIDATES table needs to include login credentials for a website
- New columns
  - USERNAME
  - PASSWORD (sensitive)
- Target is not secure
  - Accessible via public website
  - Accessible to 3rd party partners
- Data breeches in the U.S.A.
- New regulations in the E.U. (e.g. http://www.eugdpr.org/)

Implementation Steps

- Shutdown Loader THMRDB
  `Ares> jcc_clml_shutdown thmrdb`

- Modify Source table(s) (as necessary)
- Generate new TABLE.INI (as necessary)
- Modify MAPTABLE.INI to add new column(s)
- Modify Target table(s) (as necessary)
- Modify FilterMap database (as necessary)

- Restart Loader
  `Ares> submit/noti/nopri/keep/log=[] CLML_THMRDB.COM`

- Update source data to populate new column(s)

Source Modifications

- In Source

```
alter table CANDIDATES
  add USERNAME CHAR(64)
  add PASSWORD CHAR(64);
```

- Generate TABLE.INI

```
$ JCC_LML_CREATE_CONTROL_FILE mfp_db TABLE
```

- New CANDIDATES description

```
!
Table~CANDIDATES~2~~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
Column~CANDIDATES~USERNAME~5~64~0~14~0
Column~CANDIDATES~PASSWORD~6~64~0~14~0
!
```

MAPTABLE Modifications

- In the MAPTABLE.INI

```
MapTable~CANDIDATES~CANDIDATES~Replicate
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
!
MapColumn~CANDIDATES~USERNAME
MapResult~CANDIDATES~PASSWORD~ jcc$encrypt(PASSWORD)
```

- In Target

```
alter table CANDIDATES
  add USERNAME CHAR(64)
  add PASSWORD CHAR(96);
```

FilterMap Database Modifications

```
SQL> create domain jcc$char300 varchar(300);
SQL> create domain jcc$char200 varchar(200);
SQL> ...
SQL> create procedure encrypt (
cont>     in      jcc$char200 by reference,
cont>     inout   jcc$char300 by reference)
cont>     language SQL;
cont>     external
cont>         name jcc$encrypt
cont>         location 'jcc_tool_local:jcc$crypt.exe'
cont>             with ALL logical_name translation
cont>         language C
cont>         parameter style GENERAL
cont>     comment is 'Encrypts and base64 encodes a character string';
SQL> create procedure decrypt (
SQL> ...
SQL> create module JCC$UTILITIES
cont>     language SQL
cont>     ...
cont>     function jcc$encrypt (in :instr jcc$char200)
cont>     returns jcc$char300
cont>     variant
cont>     comment is 'This is a functional wrapper which returns'
cont>     /           'the encrypted, base64 encoded string';
cont>     begin
cont>     declare :res jcc$char300;
cont>         call encrypt(:instr,:res);
cont>         return(:res);
cont>   ...
cont> end module;
SQL> grant execute on module JCC$UTILITIES to public;
SQL> commit;
```

Functions jcc$encrypt, jcc$decrypt

```
Ares> type JCC$UTILITIES.C
...
int jcc$encrypt(char *, char *);
int jcc$decrypt(char *, char *);
...
int jcc$encrypt(char *inString, char *outCryptB64)
{
...
    inlen = strlen(inString);
    if (inlen >= _MAX_UNENC_SIZE) {
        strcpy(outmime,"<TOOBIG>");
        return(0);
    }
...
    strcpy(outCryptB64,outmime);
    return(1);
}
...
int jcc$decrypt(char *inCryptB64, char *outString)
{
...

Ares> cc JCC$UTILITIES.C
Ares> link/nodeb/share/exe=jcc_tool_local:jcc$utilities.exe JCC$UTILITIES.opt/opt
Ares> shareable_install jcc_tool_local jcc$utilities.exe
```

MapResult In Action

- Rdb source database update:

```
SQL> update CANDIDATES set USERNAME=trim(FIRST_NAME)||'.'||trim(LAST_NAME),
cont> PASSWORD='ChangeMe!' where LAST_NAME='Wilson';
1 row updated
SQL> commit;
SQL> select USERNAME, PASSWORD from CANDIDATES
cont> where LAST_NAME='Wilson';
USERNAME                                                        PASSWORD
 Oscar.Wilson                                                    ChangeMe!
1 row selected
```

- Target database query

```
SQL> select USERNAME, PASSWORD from CANDIDATES
cont> where LAST_NAME='Wilson';
 USERNAME
   PASSWORD
 Oscar.Wilson
   GkKI2SOJ0mEP+uI1VXWlNg==

1 row selected
```

Summary

- Brief framework of the JCC LogMiner Loader and data transforms
- Detailed examples of configuring a replication to take advantage of MapResult feature

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

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.