Initial Fast Loading Standard Dictionaries in an integrated TMS and Oracle Clinical Environment

DBMS Consulting, Inc
Sunil G. Singh
Introduction

- Sunil G. Singh of DBMS Consulting, Inc.
- Specialize in large Oracle Clinical and Oracle Application implementations and long-term support.
Acknowledgements

• Thanks to the OCUG and TMS Focus Group for the opportunity to present this paper
Overview

- Discussing Initial TMS Dictionary Loading only
- Review Documented Dictionary loading process
- Why attempt different loading techniques?
- Accelerating Documented Dictionary loading process
- Using export/import for Fast Initial Standard Dictionary Loading
- Using Transportable Tablespaces in 8i
TMS Dictionary loading process

- Base install of TMS required
- Each dictionary has to be manually created via the TMS interface very carefully
- Loading scripts for staging tables need to be developed for SQL*Loader
Loading Scripts for populating TMS_PREDICT tables need to be developed

Activation of Dictionaries with periodic Database statistic generation needs to occur, can be a very long process

Dictionary has to be tested and checked for any loading errors

Repeat for other TMS environments, such as testing, validation, training, development, etc…
Why consider alternate TMS dictionary loading methods?

- Very time consuming and very expensive to develop loading scripts for TMS
- Scripts need to be tested and validated
- Load process itself can be extremely long and error-prone
- Necessary to consistently rebuild dictionary environments for testing and validation
Accelerating Documented TMS dictionary loading process

- Use `direct=true` and `buffer=10000000` for `sqlload` commands to load staging tables.
- Generate statistics continuously during activation if there is enough I/O and CPU power. Documentation says to generate statistics “half-way” through the activation process, but this can be difficult to determine.
- Separate staging tables into a different tablesapce and I/O path.
To monitor activation from SQL*Plus:

You can monitor the activation process and analyze the tables to optimize execution speed.

1. From SQL*Plus, type:
   ```sql
   select count(*) from tms_dict_contents
   ```

2. **When the job is half done**, compute statistics to speed the job. For each of the production tables, `tms_dict_contents` and `tms_dict_relations` type:
   ```sql
   analyze table <tablename> compute statistics
   ```

3. When the job is completed and the dictionary data active, re-analyze the tables.

**About Activation**

During activation, TMS processes terms and relations in one activation group at a time, enforcing the integrity of their relations against the level relations defined for the dictionary. TMS gathers *threads* of data — terms related directly and indirectly to each other — and checks all the links in the thread.

**Note:** During activation, TMS checks for cardinality violations. If you have defined relations to more than one term in a level with only a single cardinality relation defined, TMS activates the first relation defined and rejects subsequent relations.
Goal: Perform the Initial TMS dictionary Load once

- Test and validate dictionary structure and integrity one time
- Use this TMS instance as a SOURCE database
- Consistently rebuild standard TMS dictionaries quickly from this source.
Export/Import loading of Initial TMS Dictionaries (1)

- Possible from the export command file given for installing TMS with Symmetric Replication across instances.
- Only way to move loaded dictionaries across different OS, e.g., UNIX to NT.
- Performs very well. Orders of magnitude faster than documented loading techniques.
Export/Import loading of Initial TMS Dictionaries (2)

- Start with an export from of the full TMS schema in the SOURCE database
- Perform an initial install of TMS in the target environment
- Truncate tables and drop constraints and sequences from all TMS schema tables
- Re-import full TMS schema into TARGET instance listed in the export command
- Update the TMS_DEF_INSTANCES table
Transportable Tablespaces: Advantages

- Fastest method of loading dictionaries on the same server
- Uses some Oracle 8i functionality
- Exact, dead-consistent match of TMS data between instances
Transportable Tablespaces: Drawbacks

- Source and target databases must:
  - be on same OS platform
  - have same character set
  - have same database block size

- TMS uses domain indexes when Oracle Intermedia is installed. This is NOT supported
  - create index tms_dict_contents_ci1 on tms_dict_contents(term) indextype is ctxsys.context;

- TMS can not be in use.
Initial Load of TMS Dictionaries: Transportable Tablespaces (1)

- Drop domain index tms_dict_contents_ci1
- Check if tablespaces are self-contained:
  - Exec
    dbms_tts.transport_set_check('tms_data,tms_id x',false);
- select * from transport_set_violations; This table should contain no records.
OCUG Orlando 2000: Fast Loading Standard Dictionaries in TMS and OC Environment

```
SQL> exec dbms_tts.transport_set_check('tms_data,tms_idx',false);

PL/SQL procedure successfully completed.

SQL> select * from transport_set_violations;

VIOLATIONS
-----------------------------------------------
Secondary Object TMS.DR$TMS_DICT_CONTENTS_CI1$I in tablespace TMS_DATA not allowed in transportable set

Secondary Object TMS.DR$TMS_DICT_CONTENTS_CI1$R in tablespace TMS_DATA not allowed in transportable set
```
Initial Load of TMS Dictionaries: Transportable Tablespaces (2)

- Change status of tablespaces to read only:
  - alter tablespace TMS_DATA read only;
  - alter tablespace TMS_IDX read only;
- Get path information for datafiles that belong to the TMS tablespaces:
  - select file_name from dba_data_files where tablespace_name in ('TMS_DATA','TMS_IDX');
Initial Load of TMS Dictionaries: Transportable Tablespaces (3)

- Transportable tablespaces require a metadata export only, not actual data.
- Run an export with the following parameters:
  - USERID='sys/change_on_install as SYSDBA'
  - LOG=tms_trans_exp.log FILE=tms_data.dmp
  - TRANSPORT_TABLESPACE=Y
  - TABLESPACESES=(tms_data,tms_idx)
  - CONSTRAINTS=Y GRANTS=Y TRIGGERS=Y
  - DIRECT=Y
Initial Load of TMS Dictionaries: Transportable Tablespaces (4)

- Copy datafiles to target server or location
- Put **SOURCE** tablespaces back in write mode:
  - alter tablespace TMS_DATA read write;
  - alter tablespace TMS_IDX read write;
- Drop TMS tablespaces at the **TARGET** database
  - drop tablespace tms_data including contents cascade constraints;
  - drop tablespace tms_idx including contents cascade constraints;
Initial Load of TMS Dictionaries: Transportable Tablespaces (5)

- Run Import on the **TARGET** database with the following parameters:
  - USERID='sys/change_on_install@octms as SYSDBA'
  - LOG=tms_trans_imp.log FILE=tms_data.dmp
  - TRANSPORT_TABLESPACE=y
  - DATAFILES=('C:\ORADATA\OCTMS\TMS_DATA1.DBF', 'D:\ORADATA\OCTMS\TMS_IDX1.DBF')

- DATAFILES= is the new location of the datafiles copied from the SOURCE Database
OCUG Orlando 2000: Fast Loading Standard Dictionaries in TMS and OC Environment

Initial Load of TMS Dictionaries: Transportable Tablespaces (6)

- Put new tablespaces into read-write mode
  - alter tablespace TMS_DATA read write;
  - alter tablespace TMS_IDX read write;
- Recompile all packages in the entire instance
- Re-Run `%OPA_HOME%\tmscontext.sql` to rebuild Domain indexes
  - Can take several minutes
- Update TMS_DEF_INSTANCES table
- Generate a template for controlfile rebuild
  - alter database backup controlfile to trace;

DBMS Consulting, Inc.
Sunil G. Singh and Jose A. Garcia
Conclusions:

- Both export/import and transporatable tablespace methods reduce initial dictionary load time remarkably and provide very consistent data.
- Good Installation Qualification and QA tests can verify dictionary integrity after these loads.
- Excellent way to rebuild test, development and training environment, especially for testing dictionary updates or changes in Domain structure.

DBMS Consulting, Inc.
Sunil G. Singh and Jose A. Garcia