The ttImportFromOracle utility

Contents

1. Overview

ttImportFromOracle analyzes a set of tables in an Oracle database and generates various SQL / script files to allow those tables and their data to be imported into TimesTen. Options are provided to allow for the import of associated objects, such as primary and foreign keys. A key feature is the ability to analyze the table schema and data content and then to recommend the optimal data type mapping and compression.

The main features provided by ttImportFromOracle are:

- Generates optimized scripts for importing objects from Oracle to TimesTen while minimizing the memory required to store those objects in TimesTen.
- Optional data type mapping to conserve space; mapping of Oracle NUMBER columns to TimesTen native types/NUMBER with smaller precision and mapping of [N]VARCHAR2 columns to smaller maximum sizes based on analysis of data in Oracle.
- Various options for mapping RAW/LONG/LONG RAW and LOBs to compatible TimesTen types.
- Optional compression evaluation and compression clause generation based on analysis of data in Oracle.
- Analyzes one or many Oracle tables in a single invocation
- Selectively include/omit imports of primary keys, foreign keys, unique indexes, non-unique indexes, bitmap indexes and materialized views
- Imports data directly from Oracle (using the ttLoadFromOracle() built-in). Generates an optimized load script that performs data loading in parallel to minimize elapsed time and maximize use of hardware.
- Handles tables containing columns with unsupported data types.

- Detailed sizing estimates for tables and indexes.
- Extensive suite of options provide flexibility for the user
- Supported on a wide range of platforms:

  Microsoft Windows 64 bit
  Linux x86 64 bit
  macOS 64 bit

- Supports Oracle database 11g, 12c, 18c, 19c and 21c. Support for 12c and upwards has some caveats (see the section on **Known Issues**).

This utility is not an official production utility and support will be provided on a 'best effort' basis. The most up to date version of this utility can always be found on the TimesTen downloads page here: https://www.oracle.com/downloads/samplecode/otsu-samplecode-downloads.html

Here is a summary of the command syntax.

```
Import table definitions and data from an Oracle database

Usage:

ttImportFromOracle [-h | -help | -? [full] ]

ttImportFromOracle [-V | -version]

ttImportFromOracle  -oraConn ouser[/opasswd][@oserver]
                   [-ttOwner ttuser]
                   [-tables tablespec [tablespec]...]...
                   [-tableFile filepath]...
                   [-importMviews { 0 | 1 }]
                   [-typeMap num[,var[,rawlong[,lob[,tstz[,padding[,stats]]]]]]]
                   [-compression level[,padding[,stats[,cfactor[,minrows]]]]]
                   [-inlineLimit n]
                   [-parallel dop]
                   [-dataLoadParallelism dlpspec]
                   [-indexes unique[,nonunique[,bitmap[,nodups]]]]
                   [-constraints pkey[,fkey]]
                   [-outputDir dirpath]
                   [-prefix fnprefix]
                   [-target { unix | windows }]
                   [-onBadType { 0 | 1 | 2 }]
                   [-verbosity { 0 | 1 }]
```

In sections 2 and 3 we will look in more detail at the meaning and usage of the **–typeMap** and **–compression** options. For full details on the other options please consult the detailed syntax and description in section 5.

In section 4 I present a simple worked end to end usage example. Section 5 is a full online help listing providing full details on all the command line options. Section 6 lists some caveats and limitations that you should be aware of and section 7 provides some tips on troubleshooting and how to get support.

2. Prerequisites

In order to use this utility, you must ensure that the following prerequisites are met:

- The Oracle Database version must be at least 11.2.2. Older versions are not supported.
- The system where you plan to run the utility must be using a supported operating system.
- The system where you plan to run the utility must have a suitable Oracle Client installed (the Instant Client is sufficient) and the client must be configured for connectivity to the Oracle database.
- When running the utility, you should ensure that your shell environment is properly set (PATH, LD_LIBRARY_PATH [or equivalent], etc.)

3. Data Type Mapping

When moving tables from an Oracle database to a TimesTen database some degree of storage 'inflation' is typically observed due to the differing internal storage architectures of the databases. Given that memory capacity is typically much more constrained than disk capacity it is very important to try and minimize memory usage especially when data volumes are very large.

ttImportFromOracle provides features that can help with this; one of these is the ability to map the data types used for columns in Oracle database to compatible data types in TimesTen that may consume less memory.

ttImportFromOracle provides options to perform type mapping of numeric types (NUMBER), variable length types (VARCHAR2 and NVARCHAR2), RAW/LONG/LONG RAW and LOBs (CLOB/NCLOB/BLOB).

Type mapping is controlled by the –typeMap command line option the syntax of which is as follows:

```
-typeMap num[,var[,rawlong[,lob[,tstz[,padding[,stats]]]]]]
```

These are the supported Oracle data types and the default corresponding TimesTen types.

Oracle Type	**TimesTen Type**
VARCHAR2	VARCHAR2
NVARCHAR2	NVARCHAR2
NUMBER	NUMBER
FLOAT	FLOAT
DATE	DATE
BINARY_FLOAT	BINARY_FLOAT

BINARY_DOUBLE	BINARY_DOUBLE
TIMESTAMP	TIMESTAMP
TIMESTAMP WITH TIMEZONE	TIMESTAMP
TIMESTAMP WITH LOCAL TIMEZONE	TIMESTAMP
CHAR	CHAR
NCHAR	NCHAR
CLOB	CLOB or VARCHAR2
NCLOB	NCLOB or NVARCHAR2
BLOB	BLOB or VARBINARY
RAW	VARBINARY
LONG	VARCHAR2
LONG RAW	VARBINARY

2.1 Numeric type mapping

Type mapping can be applied to numeric columns of type NUMBER and is controlled by the 'num' field in the **–typeMap** command line option. This field takes value of 0, 1 or 2 whose meaning is as follows:

0 No mapping; NUMBER columns in the TimesTen database will have the same type, precision and scale as in the Oracle database.

1 **This is the default.** Standard numeric mapping is performed; Oracle database columns of type NUMBER(p, 0) i.e. with explicitly specified precision and scale explicitly specified as 0, are mapped to a suitable TimesTen native integer type where possible. The mapping is performed as follows:.

p < 5	-	TT_SMALLINT
else		
p >= 5 && ***p*** < 10	-	TT_INTEGER
else		
p >= 10 && ***p*** < 19	-	TT_BIGINT
else		
		NUMBER(p, 0)

2 More aggressive type mapping is performed for NUMBER columns. The data in each NUMBER column in the Oracle database table is scanned (full table scan) to determine

metrics such as the largest value stored, the smallest value stored, the range of any non-integral values stored and whether there are any negative values stored. Based on these metrics, columns with only integral values are mapped as per standard type mapping except that the maximum stored precision is used rather than the maximum defined precision (but see also the padding option below). An additional mapping is used in this mode:

p < 3 and no negative values - TT_TINYINT

Columns containing non-integral values are mapped to the smallest possible NUMBER(p,s) type consistent with the range of stored values (where p <= 38, s <= 38 and p+s <= 38) or to NUMBER where values are outside this range.

2.2 Variable length type mapping

Type mapping can be applied to columns of type VARCHAR2 and NVARCHAR2 and is controlled by the 'var' field in the **–typeMap** command line option. This field takes value of 0 or 1 whose meaning is as follows:

0 **This is the default.** No mapping; the columns in the TimesTen database will have the same type and length as in the Oracle database

1 Aggressive variable length mapping. The data in each column in the Oracle database is analyzed to determine:

- The length semantics (char or byte)
- The maximum stored data length
- The average stored length.

Any column where the maximum stored data length (accounting for length semantics) is less than the declared maximum size is imported as a column with the same data type and length semantics but with a defined maximum length equal to the maximum stored data length (but see also the padding option below).

This mapping is of course potentially dangerous. The mapping is only valid for the data values that currently exist in the Oracle database tables. If the data in the Oracle database changes after the mapping is done then the new data may no longer meet the criteria used for the mapping. Nonetheless, the potential for saving significant amounts of memory, especially for very large data sets (millions of rows), makes this kind of mapping attractive.

2.3 RAW/LONG type mapping

TimesTen does not support the data types RAW, LONG and LONG RAW. However, the TimesTen types VARCHAR2 and VARBINARY are very similar and can often be used instead.

Type mapping can be applied to Oracle database columns of type RAW(n), lONG and LONG RAW and is controlled by the 'rawlong' field in the **–typeMap** command line option. This field takes value of 0 or 1 whose meaning is as follows:

0 No mapping; RAW, LONG and LONG RAW columns are treated as any other unsupported data type. The columns will not appear in the TimesTen tables.

1 **This is the default**. Columns will be mapped as follows:

- RAW(n) to VARBINARY(n)
- LONG to VARCHAR2(4194304)
- LONG RAW to VARBINARY(4194304)

In addition, for any RAW(n) columns mapped to VARBINARY(n), variable length type optimization will then be applied if var=1.

Columns mapped in this way can have their data loaded using the ttLoadFromOracle() builtin procedure (as long as the actual data length in Oracle does not exceed the 4 MB maximum for the TimesTen variable length types).

2.4 LOB type mapping

TimesTen supports the CLOB, NCLOB and BLOB data types. However, currently the ttLoadFromOracle() builtin cannot load data into these types in TimesTen. As a result, tables with these columns included will not have load commands generated in the LoadData.sql script. It may therefore be desirable to map these Oracle types to TimesTen VARCHAR2, NVARCHAR2 or VARBINARY types so that the data can be loaded.

Type mapping can be applied to Oracle database columns of type CLOB, NCLOB and BLOB and is controlled by the 'lob' field in the **–typeMap** command line option. This field takes value of 0 or 1 whose meaning is as follows:

0 No mapping; CLOB, NCLOB and BLOB columns are mapped to the same type in TimesTen. Any table containing columns of these types will not have its data loaded from Oracle using the generated LoadData.sql script.

1 **This is the default**. LOB columns will be mapped as follows:

- CLOB to VARCHAR2(4194304)
- NCLOB to NVARCHAR2(2097152)
- BLOB to VARBINARY(4194304)

In addition, if variable length type optimization was specified (var=1) then it will also be applied.

2.5 TIMESTAMP WITH TIMEZONE type mapping

TimesTen does not support the TIMESTAMP WITH TIMEZONE data type. As a result, it is not possible to import columns of this type into TimesTen while preserving all aspects of the data. Columns of this type will be mapped to a regular TIMESTAMP in TimesTen but this means that when the data is loaded the associated timezone information will be lost.

Type mapping can be applied to Oracle database columns of type TIMESTAMP WITH TIMEZONE and is controlled by the 'tstz' field in the **–typeMap** command line option. This field takes value of 0 or 1 whose meaning is as follows:

0 No mapping; TIMESTAMP WITH TIMEZONE columns are treated as any other unsupported data type. The columns will not appear in the TimesTen tables.

1 **This is the default**. TIMESTAMP WITH TIMEZONE columns will be mapped to TIMESTAMP in TimesTen. When data is loaded into these columns the timezone information is discarded.

2.6 Padding

When using the aggressive type mappings (num=2, var=1) it may be useful to allow some 'padding', or headroom, to cater for a possible increase in the maximum size of the values stored in the Oracle database tables and thus avoid having to re-analyze the data and regenerate the TimesTen schema if the data changes

A padding factor can be applied to aggressive type mapping and this is controlled by the 'padding' field in the **–typeMap** command line option. This field takes an integer value in the range 0 (the default) to 1000. This value represents the percentage increase to be applied to the maximum precision / length of stored values as determined by the data analysis.

For example, if a NUMBER column is to be mapped and data analysis determines that the maximum precision of any value stored in this column at present is 8 and the maximum scale value is 2 then with a padding factor of 50 (i.e. 50%) these would be increased to 12 and 3 respectively before type mapping is performed. Similarly if the maximum stored value length for a VARCHAR2 column were 100 then with a padding factor of 25 (i.e. 25%) this would be increased to 125.

2.7 Type mapping statistics

During data analysis various statistics are gathered about the data. If desired these statistics can be output, as comments, in the generated CreateTables.sql file. This is controlled by the 'stats' field of the **–typeMap** command line option. This field takes a value of 0 or 1 whose meaning is as follows:

0 **This is the default.** The statistics are not output.

1 The statistics will be output.

The space (memory) savings from type mapping can be quite considerable but are of course highly schema and data dependent. Use of type mapping also introduces some 'risk'.

Standard numeric type mapping (num=1) is completely safe for Oracle -> TimesTen data movement but the native integer types in TimesTen are capable of storing values that may be larger than can fit in the corresponding Oracle NUMBER columns (depending on the exact definition) and so if it is planned to insert/update data in the TimesTen database and then move that data back to the Oracle database then this must be taken into account.

Aggressive type mapping (num=2 and/or var=1) is more risky, since subsequent changes (insert/update) to data in the Oracle database may mean that the sizes used for columns in the TimesTen database are now too small to hold the maximum stored values from the Oracle database. Careful use of the padding capability, together with a good understanding of the application data and how it is used, can help to mitigate this risk.

4. Compression Evaluation

Caveat: The TimesTen compression feature is only supported in TimesTen Classic and is not supported for tables cached from an Oracle database.

As well as data type mapping another good way to save memory, and potentially improve query performance, is to use the TimesTen in-memory compression feature. This enables you to compress individual columns, or groups of columns, within a table so that they occupy (much) less memory. The compression scheme used is a dictionary based mechanism. A separate dictionary of unique values is maintained for each compressed column, or column group, and the column in each table row contains a pointer (index) to the actual value in this dictionary. Depending on the maximum number of unique values (including NULL) that must be stored the size of this pointer may be 1, 2 or 4 bytes. Clearly if the compressed column is, say, a CHAR(20) and this can be replaced in every row by, say, a 2 byte pointer then the space savings are very large. For a full description of this feature please see the **TimesTen SQL Reference**.

The ttImportFromOracle utility can analyze data in Oracle database tables and determine if specific columns and tables will benefit from being compressed when they are moved to a TimesTen database. This feature is controlled by the **–compression** command line option. The syntax for this option is:

```
-compression level[,padding[,stats[,cfactor[,minrows]]]]
```

3.1 Compression level

There are three different compression levels supported by ttImportFromOracle and these are controlled by the 'level' field of the **–compression** command line option. This takes a value of 0, 1 or 2 whose meaning is as follows:

0 This is the default. Oracle database tables are not analyzed for compression and no compression clauses are generated for TimesTen tables.

1 Oracle database tables are analyzed for compression using the maximum dictionary pointer size (4 bytes) for every column. For tables/columns that meet the criteria to qualify for being compressed (see below) appropriate compression clauses are output in the generated **CreateTables.sql** DDL file.

2 As for 1, but the dictionary pointer size is adjusted based on the actual number of distinct values in the data. This may result in additional space savings but if data is subsequently changed (insert/update) in the Oracle database tables such that the number of distinct values increases for a compressed column the new data may exceed that which can be accommodated by the pointer size for the compressed TimesTen table. See the 'padding' option below for one possible solution to this.

3.2 Padding

When using adaptive compression (level=2) it may be useful to allow some 'padding' or headroom to cater for a possible increase in the maximum number of distinct values stored in the Oracle database tables and thus avoid having to re-analyze the data and regenerate the TimesTen schema if the data changes.

A padding factor can be applied to adaptive compression and this is controlled by the 'padding' field in the **–compression** command line option. This field takes an integer value in the range 0 (the default) to 1000. This value represents the percentage increase to be applied to the number of distinct values for a column as determined by the data analysis.

For example, if a column is to be considered for adaptive compression and data analysis determines that the column currently contains 237 distinct values (including NULL) then with a padding factor of 50 (i.e. 50%) this would be increased to 356 before compression evaluation is performed.

3.3 Compression statistics

During compression evaluation various statistics are gathered about the data. If desired these statistics can be output, as comments, in the generated CreateTables.sql file. This is controlled by the 'stats' field of the **–compression** command line option. This field takes a value of 0 or 1 whose meaning is as follows:

0 **This is the default.** The statistics are not output.

1 The statistics will be output.

3.4 Minimum required compression factor

While compression can result in significant space savings and also may improve the performance of queries that have to scan large amounts of data it also represents an overhead for DML operations. As a result it is generally considered best to only compress a table when the resulting space savings are deemed worthwhile. One way to control this is to specify a minimum required compression factor; if compressing a table will not result in it being compressed by at least this amount then the table will not be eligible for compression.

You can specify a minimum required compression factor using the 'cfactor' field of the **–compression** command line option. This field takes an integer value in the range 1 to 100. It represents the percentage compression computed as:

((storage used by compressed table) / (storage used by uncompressed table)) * 100

Thus a compression factor of 30 (i.e. 30%) means that a compressed table will require approximately 1/3 the storage of the uncompressed form.

The default value for cfactor is 70 (i.e. 70%).

See the description below to understand how this is used in the compression evaluation process.

3.5 Minimum number of rows

The compressed columns/column groups are stored in TimesTen as separate tables, one per column group. There is some storage overhead associated with each table and so it is generally not beneficial to compress tables with only a small number of rows.

You can specify a minimum rowcount below which a table will not be considered for compression. This is controlled by the 'minrows' field in the **–compression** command line option. This field takes a (64-bit) integer value; any table with less than this number of rows will not be considered for compression. The minimum value is 1 (always consider compression) and the default is 1024. There should normally be little reason to change this value.

When evaluating Oracle database columns and tables for compression the algorithm used is as follows:

- Compression is only evaluated if a compression level > 0 is specified.
- For each table being imported:
    - If the table's rowcount is < minrows then no further evaluation is done and the table will not be compressed in the TimesTen database.
    - If the table qualifies based on rowcount then the table is scanned and column level metrics (total number of non-NULL values, total number of distinct values) are gathered.
    - For each column in the table:
        - If padding > 0 was specified, the number of distinct values determined is increased by padding%.
        - If compression level=2 is being used then the pointer size (1, 2 or 4 bytes) is determined based on the number of distinct values otherwise it is set to 4 (compression level=1).
        - The total size of the data for this column (all rows) in the TimesTen database when not compressed is estimated. Similarly for the size when compressed. If compressed size / uncompressed size is <= the target minimum compression factor then the column is marked as a candidate for compression.
    - A detailed sizing calculation is performed at the table level to estimate the total size of the uncompressed table in TimesTen and the total size of the compressed table (where all candidate columns are compressed) in TimesTen. This calculation goes into a lot of detail and takes account of the various internal storage structures and overheads not just the raw data. If the projected compressed size / the projected uncompressed size is <= the target minimum compression factor then the table is marked for compression.

When the CREATE TABLE DDL statements are generated in the **CreateTables.sql** file the required compression clauses are output for any table marked for compression. All columns marked for compression will be compressed.

In general the space savings from compression will be larger than those from type mapping. Using both options together will provide maximum space savings.

NOTES:

1. The compression evaluation only considers individual columns not column groups. This is because (a) column groupings are really determined by the kinds of queries to be run and the utility has no information on that and (b) evaluating all possible groupings is a >> N! problem space and the compute time required is enormous.

2. The degree of compression that will actually be achieved depends primarily on the data; size of data types, actual values stored and number of distinct values. The TimesTen ttSize utility is therefore not able to make a sensible prediction for memory usage for empty tables which use compression. The estimates from ttImportFromOracle are based on analysis of the actual data and so in general are much more accurate than those obtained by running ttSize against empty tables.

3. If an Oracle database table has to be scanned for any reason (rowcount determination, type mapping, compression evaluation) then all necessary analysis will be performed in a single scan. No table will be scanned more than once. Parallel scan will be used where possible to expedite the data analysis.

5. Parallel Data Loading

ttImportFromOracle provides two ways to load data. It generates a SQL script, **LoadData.sql**, consisting of a sequence of ttLoadFromOracle() calls. This script can be run in ttIsql to load table data from Oracle into TimesTen. A disadvantage of this script is that all the loading is done serially; there is one ttLoadFromOracle call per table and each one is executed one after the other. As a result this script is fine for loading small volumes of data into a small number of tables but it is rather inefficient for large numbers of tables and/or large volumes of data. This script can be used on both Windows and Unix/Linux platforms.

When the target platform is Unix/Linux, an additional shell script, **ttPDL.sh**, is also generated (unless parallel data loading has been explicitly disabled). This script will load individual tables in parallel (multiple ttLoadFromOracle calls per table) and also multiple tables concurrently. There are various options available to control the degree of parallelism both globally and on a per table basis (see the utility's online help for details; the –dataLoadParallelism option and the various table specification attributes) but the defaults have been chosen to give good results on an Exalytics system so if you are using one of those then you likely will not need to worry about any fine tuning. Parallel data loading via this script will more efficiently utilize the hardware capability of both the machine hosting TimesTen and the backend Oracle database server to greatly accelerate data loading thereby reducing overall load times.

6. Usage Example

In this section I present a step by step usage example. It is a very simple one but hopefully it serves to illustrate how the utility is used. The typical sequence of steps is as follows:

1. Determine access information (user, password, hostname, port, service name etc.) for the Oracle database

2. Make sure that the system where you intent to run ttImportFromOracle has a functioning Oracle client installation (the InstantClient is sufficient). Configure the client installation so that it can connect to the target Oracle database and verify this using SQL*Plus.

3. Obtain the correct ttImportFromOracle binary for your platform by downloading it from OTN (see section 1). Place the binary somewhere where you can easily run it from and be sure to set the permissions to make it executable (Unix or Linux platforms only).

4. Identify the tables you wish to analyze and import

5. Run the utility to generate the necessary DDL files and other scripts

6. Edit (if required) and then run the CreateUsers.sql file to create the required user(s).

7. Run the CreateTables.sql file to create the tables.

8. Import the data into TimesTen by running the ttPDL.sh script.

9. If desired, run the CreateIndexes.sql file to create the imported indexes and constraints.

10. Update optimizer statistics by running the UpdateStats.sql file.

In the rest of this section I take a more detailed look at each of these steps and provide examples.

The Oracle database server is accessed by TNS name 'MYDB'. The TNS entry looks like this:

```
MYDB =
  (DESCRIPTION =
    (ADDRESS =
      (PROTOCOL = TCP)(HOST = oel5-odb1.oracle.net)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = orcl)
    )
  )
```

As for any Oracle application, the TNS_ADMIN environment variable should be set to point to the directory containing the TNSNAMES.ORA file before running the utility if TNS name resolution is the desired method for accessing the Oracle DB server. ttImportFromOracle also supports the use of Oracle 'Easy Connect' syntax (serverhostname:portno/dbservicename) for accessing the Oracle database.

In this example, the credentials to access the Oracle database are username 'scott' and password 'tiger'. User scott has the following tables, indexes and data:

```
CREATE TABLE PARENTTAB
(
    P_PK          NUMBER(10,0) NOT NULL,
    P_C1          NUMBER,
    P_C2          DATE NOT NULL,
    P_C3          VARCHAR2(200),
    P_C4          NVARCHAR2(1000),
    CONSTRAINT CP_PK PRIMARY KEY ( P_PK )
);

CREATE UNIQUE INDEX P_IX1 ON PARENTTAB( P_C1 );

CREATE INDEX P_IX2 ON PARENTTAB( P_C2 );
```

```
INSERT INTO PARENTTAB VALUES(1, 10, '2012-12-01', 'ABCDE', 'ABCDE');
INSERT INTO PARENTTAB VALUES(2, 20, '2012-12-02', NULL, 'A');
INSERT INTO PARENTTAB VALUES(3, 30, '2012-12-03', 'ABCDEFGHIJKLMNOPQRSTUVWXYZ', NULL);
INSERT INTO PARENTTAB VALUES(4, 40, '2012-12-04', 'ABC', 'ABCDEFGHIJKLMNOPQRSTUVWXYZ');
INSERT INTO PARENTTAB VALUES(5, 50, '2012-12-05', 'ABCDE', NULL);

CREATE TABLE CHILDTAB
(
    C_PK          NUMBER(6,0) NOT NULL,
    C_FK          NUMBER(10,0),
    C_C1          VARCHAR2(255),
    CONSTRAINT CC_PK PRIMARY KEY ( C_PK ),
    CONSTRAINT CC_FK FOREIGN KEY ( C_FK)
        REFERENCES PARENTTAB( P_PK )
);

INSERT INTO CHILDTAB VALUES ( 1, 3, 'ZZZ');
INSERT INTO CHILDTAB VALUES ( 2, 2, NULL );
INSERT INTO CHILDTAB VALUES ( 3, 5, 'ABCDEFGHIJKLMNOPQRST' );
```

We have a TimesTen installation where the instance administrator user is 'oracle' and where there is a DSN, MYTTDB, configured as follows:

```
[MYTTDB]
Driver=/home/oracle/TimesTen/tt1122/lib/libtten.so
DataStore=/home/oracle/TimesTen/tt1122/info/DemoDataStore/myttdb
PermSize=128
TempSize=32
PLSQL=1
DatabaseCharacterSet=AL32UTF8
ConnectionCharacterSet=AL32UTF8
OracleNetServiceName=MYDB
```

The necessary network and TNS configuration is in place to allow this TimesTen database to communicate with the Oracle database previously described.

First let us run ttImportFromOracle with default options to analyze these simple tables and generate some DDL files and scripts. On the TimesTen system we would run this command:

```
$ ttImportFromOracle  –oraConn  scott/tiger@mydb –tables parenttab childtab
```

this will result in the following output as the utility executes the various processing steps

```
Beginning processing
Resolving any tablename wildcards
Eliminating any duplicate tables
Getting metadata from source
Generating database user list
Assigning TimesTen datatypes
Analyzing source tables
Analyzing table 'SCOTT.PARENTTAB' ...
Analyzing table 'SCOTT.CHILDTAB' ...
Estimating table sizes
Estimating index sizes
Evaluating parallel data load
Generating output files
Finished processing
```

The following output files will have been generated

```
$ ls
ttPDL.sh
CreateIndexes.sql
CreateTables.sql
CreateUsers.sql
DropIndexes.sql
DropTables.sql
LoadData.sql
TableList.txt
UpdateStats.sql
```

Let us examine some of them …

CreateUsers.sql

```
------------------------------------------------------------------------
-- Generated by ttImportFromOracle 3.4.0 - Mon Feb  3 16:30:33 GMT 2014
------------------------------------------------------------------------

CREATE USER SCOTT IDENTIFIED BY 'scott';

GRANT CREATE SESSION, CREATE TABLE TO SCOTT;

-- GRANT CREATE VIEW, CREATE SEQUENCE, CREATE SYNONYM
--       TO SCOTT;
```

This can be used to create the user 'scott' in the TimesTen database. The password used defaults to the same as the username; you should edit this file to use a more secure password or change the user password after the user is created.

Some of the permission grants are commented out as it may not be desirable to grant this user all these permissions. If you desire to grant any of these additional permissions then edit the file accordingly before running it.

CreateTables.sql

```
------------------------------------------------------------------------
-- Generated by ttImportFromOracle 3.4.0 - Mon Feb  3 16:30:33 GMT 2014
------------------------------------------------------------------------
--
-- See end of file for table size totals
--
-- Source is Oracle Database 11.2.0.2.0
--
-- Character set is AL32UTF8
--
------------------------------------------------------------------------
--
-- Type mapping modes:
--             Numeric: 1 - STANDARD
--              Varlen: 0 - NONE
--  RAW/LONG/LONG RAW: 1 - YES
--               LOBs: 1 - YES
--  TIMESTAMP WITH TZ: 1 - YES
--
```

```
-- Compression mode: 0 - NONE
--
------------------------------------------------------------------------------

-- *************************************************************
-- Rows: 5
-- Bytes per row: Oracle   - max 2251
--                TimesTen - inline 53, not inline max 2232, total 2285
-- Estimated table size = 29265 bytes
CREATE TABLE SCOTT.PARENTTAB
(
    P_PK                          TT_BIGINT NOT NULL,
-- Oracle type:                     NUMBER(10,0)

    P_C1                          NUMBER,
-- Oracle type:                     NUMBER

    P_C2                          DATE NOT NULL,
-- Oracle type:                     DATE

    P_C3                          VARCHAR2(200 BYTE) NOT INLINE,
-- Oracle type:                     VARCHAR2(200 BYTE)

    P_C4                          NVARCHAR2(1000) NOT INLINE
-- Oracle type:                     NVARCHAR2(1000)
);

-- *************************************************************
-- Rows: 3
-- Bytes per row: Oracle   - max 299
--                TimesTen - inline 20, not inline max 271, total 291
-- Estimated table size = 15205 bytes
CREATE TABLE SCOTT.CHILDTAB
(
    C_PK                          TT_INTEGER NOT NULL,
-- Oracle type:                     NUMBER(6,0)

    C_FK                          TT_BIGINT,
-- Oracle type:                     NUMBER(10,0)

    C_C1                          VARCHAR2(255 BYTE) NOT INLINE
-- Oracle type:                     VARCHAR2(255 BYTE)
);

------------------------------------------------------------------------------
-- Estimated total size for all tables: 44470 bytes
------------------------------------------------------------------------------
```

Here we can see that the default type mapping (standard) has been applied and so some of the numeric columns have been mapped to TimesTen native integer types. You can also see that the character length semantic settings and INLINE/NOT INLINE storage settings have been explicitly specified. As all the variable length columns have a declared maximum size > 128 bytes they have been stored NOT INLINE.

CreateIndexes.sql

```
------------------------------------------------------------------------------
-- Generated by ttImportFromOracle 3.4.0 - Mon Feb  3 16:30:33 GMT 2014
------------------------------------------------------------------------------
```

```
--
-- See end of file for index size totals. Sizing estimates for alternate
-- index types are shown, enclosed in parentheses, for information only and
-- are not included in the total. Index types should be chosen based on
-- workload not on size.
--
------------------------------------------------------------------------------

-- timing 1;

-- ************************************************************
-- Index: SCOTT.P_IX1
-- Table: SCOTT.PARENTTAB
-- Rows:  5
-- Ttree estimated size = 4074 bytes
-- (Btree estimated size = 6194 bytes)
-- (Hash estimated size = 3114 bytes)
CREATE UNIQUE INDEX SCOTT.P_IX1
          ON SCOTT.PARENTTAB
             (
                 P_C1
             );

-- ************************************************************
-- Index: SCOTT.P_IX2
-- Table: SCOTT.PARENTTAB
-- Rows:  5
-- Ttree estimated size = 4074 bytes
-- (Btree estimated size = 5170 bytes)
-- (Hash estimated size = 2922 bytes)
CREATE INDEX SCOTT.P_IX2
          ON SCOTT.PARENTTAB
             (
                 P_C2
             );

-- ************************************************************
-- Index: SCOTT.CP_PK
-- Table: SCOTT.PARENTTAB
-- Rows:  5
-- Ttree estimated size = 4074 bytes
-- (Btree estimated size = 5170 bytes)
-- (Hash estimated size = 2874 bytes)
ALTER TABLE SCOTT.PARENTTAB
    ADD CONSTRAINT CP_PK
    PRIMARY KEY
    (
        P_PK
    );

-- ************************************************************
-- Index: SCOTT.CC_PK
-- Table: SCOTT.CHILDTAB
-- Rows:  3
-- Ttree estimated size = 4074 bytes
-- (Btree estimated size = 5170 bytes)
-- (Hash estimated size = 2874 bytes)
ALTER TABLE SCOTT.CHILDTAB
    ADD CONSTRAINT CC_PK
    PRIMARY KEY
    (
        C_PK
```

```
    );

-- ************************************************************
-- Index: SCOTT.CC_FK
-- Table: SCOTT.CHILDTAB
-- Rows:  3
-- Ttree estimated size = 4074 bytes
-- (Btree estimated size = 5170 bytes)
-- (Hash estimated size = 2922 bytes)
ALTER TABLE SCOTT.CHILDTAB
    ADD CONSTRAINT CC_FK FOREIGN KEY
    (
        C_FK
    )
    REFERENCES SCOTT.PARENTTAB
    (
        P_PK
    );

--------------------------------------------------------------------------------
-- Estimated total index size for all indexes = 20370 bytes
```

We can see that all of the unique index, non-unique index, primary and foreign key definitions have been imported. This is the default behavior.

TableList.txt

```
################################################################################
## Generated by ttImportFromOracle 3.4.0 - Mon Feb  3 16:30:33 GMT 2014
################################################################################

SCOTT.PARENTTAB:rows=5:ttwriters=3:oradop=1:orapar=none
SCOTT.CHILDTAB:rows=3:ttwriters=3:oradop=1:orapar=none
```

This is the generated tablefile suitable for use with the –tableFile option.

Now let us re-run the utility this time asking for aggressive type mapping for both numeric and variable length types (-typeMap 2,1) and compression evaluation (-compression 2). In addition we will suppress the output of foreign keys and primary keys (-constraints 0,0). In this example I use Easy Connect syntax by way of illustration.

```
$ ttImportFromOracle -oraConn scott/tiger@oel5-odb1.oracle.net:1521/orcl
     -tableFile tablelist.txt -typeMap 2,1 -compression 2 -constraints 0,0
Beginning processing
Resolving any tablename wildcards
Eliminating any duplicate tables
Getting metadata from source
Generating database user list
Assigning TimesTen datatypes
Analyzing source tables (this may take some time)
Analyzing table 'SCOTT.PARENTTAB' ...
Analyzing table 'SCOTT.CHILDTAB' ...
Optimizing TimesTen datatypes
Estimating compression ratios
Estimating index sizes
Evaluating parallel data load
Generating output files
Finished processing
```

Let's see how this affects the output files…

CreateTables.sql

```
--------------------------------------------------------------------------------
-- Generated by ttImportFromOracle 3.4.0 - Mon Feb  3 16:30:33 GMT 2014
--------------------------------------------------------------------------------
--
-- See end of file for table size totals
--
-- Source is Oracle Database 11.2.0.2.0
--
-- Character set is AL32UTF8
--
--------------------------------------------------------------------------------
--
-- Type mapping modes:
--            Numeric: 2 - AGGRESSIVE
--             Varlen: 1 - AGGRESSIVE
--  RAW/LONG/LONG RAW: 1 - YES
--               LOBs: 1 - YES
--  TIMESTAMP WITH TZ: 1 - YES
--
-- Compression mode: 2 - VARIABLE
--                   Target = 70%, minrows = 1024
--
--------------------------------------------------------------------------------

-- ************************************************************
-- Rows: 5
-- Bytes per row: Oracle   - max 2251
--                TimesTen - inline 209, not inline max 0, total 209
-- Estimated table size = 56363 bytes
-- Table does not qualify for compression
CREATE TABLE SCOTT.PARENTTAB
(
    P_PK                          TT_TINYINT NOT NULL,
-- Oracle type:                     NUMBER(10,0)

    P_C1                          TT_TINYINT,
-- Oracle type:                     NUMBER

    P_C2                          DATE NOT NULL,
-- Oracle type:                     DATE

    P_C3                          VARCHAR2(104 BYTE) INLINE,
-- Oracle type:                     VARCHAR2(200 BYTE)

    P_C4                          NVARCHAR2(26) INLINE
-- Oracle type:                     NVARCHAR2(1000)
);

-- ************************************************************
-- Rows: 3
-- Bytes per row: Oracle   - max 299
--                TimesTen - inline 100, not inline max 0, total 100
-- Estimated table size = 33205 bytes
-- Table does not qualify for compression
CREATE TABLE SCOTT.CHILDTAB
```

```
(
    C_PK                           TT_TINYINT NOT NULL,
-- Oracle type:                      NUMBER(6,0)

    C_FK                           TT_TINYINT,
-- Oracle type:                      NUMBER(10,0)

    C_C1                           VARCHAR2(80 BYTE) INLINE
-- Oracle type:                      VARCHAR2(255 BYTE)
);

------------------------------------------------------------------------
-- Estimated total uncompressed size for all tables: 89568 bytes
-- Estimated total compressed size for all tables:   89568 bytes
-- Estimated total actual size for all tables:       89568 bytes
------------------------------------------------------------------------
```

As can be seen, there are many differences compared to standard type mapping; the numeric columns have all been mapped to native integer types and because of the actual range of values in the tables they have been mapped to TT_TINYINT. The variable length columns have all been mapped with much smaller maximum lengths and as a side effect of that are now all INLINE. One side effect of this is that the tables actually now occupy more space due to the change of the [N]VARCHAR2 columns from NOT INLINE to INLINE!

A compression evaluation has been performed for each table and the results output as comments. For these very small tables there is of course no benefit to be expected from compression but for tables with many more rows and a significant percentage of duplicate values the story would be different.

CreateIndexes.sql

```
------------------------------------------------------------------------
-- Generated by ttImportFromOracle 3.4.0 - Mon Feb  3 16:30:33 GMT 2014
------------------------------------------------------------------------
--
-- See end of file for index size totals. Sizing estimates for alternate
-- index types are shown, enclosed in parentheses, for information only and
-- are not included in the total. Index types should be chosen based on
-- workload not on size.
--
------------------------------------------------------------------------

-- timing 1;

-- *************************************************************
-- Index: SCOTT.P_IX1
-- Table: SCOTT.PARENTTAB
-- Rows:  5
-- Ttree estimated size = 4074 bytes
-- (Btree estimated size = 6194 bytes)
-- (Hash estimated size = 3114 bytes)
CREATE UNIQUE INDEX SCOTT.P_IX1
          ON SCOTT.PARENTTAB
             (
                 P_C1
             );

-- *************************************************************
-- Index: SCOTT.P_IX2
```

```
-- Table: SCOTT.PARENTTAB
-- Rows:  5
-- Ttree estimated size = 4074 bytes
-- (Btree estimated size = 5170 bytes)
-- (Hash estimated size = 2922 bytes)
CREATE INDEX SCOTT.P_IX2
          ON SCOTT.PARENTTAB
             (
                 P_C2
             );

--------------------------------------------------------------------------------
-- Estimated total index size for all tables = 8148 bytes
--------------------------------------------------------------------------------
```

Here we see that the primary and foreign key definitions have not been included.

We will now proceed to run some of the generated files to create the user, tables and indexes in TimesTen and then to import the data. First we will use the CreateUsers.sql file to create the user 'scott' in TimesTen.

```
$ ttIsql -f CreateUsers.sql myttdb
Copyright (c) 1996-2011, Oracle.  All rights reserved.
Type ? or "help" for help, type "exit" to quit ttIsql.

…

run "CreateUsers.sql";

--------------------------------------------------------------------------------
-- Generated by ttImportFromOracle 3.4.0 - Mon Feb  3 16:30:33 GMT 2014
--------------------------------------------------------------------------------

CREATE USER SCOTT IDENTIFIED BY 'scott';
User created.
GRANT CREATE SESSION, CREATE TABLE TO SCOTT;
-- GRANT CREATE VIEW, CREATE SEQUENCE, CREATE SYNONYM
--       TO SCOTT;
exit;
Disconnecting...
Done.
```

Now we will create the tables

```
$ ttIsql -f CreateTables.sql myttdb
Copyright (c) 1996-2011, Oracle.  All rights reserved.
Type ? or "help" for help, type "exit" to quit ttIsql.

…

run "CreateTables.sql";

--------------------------------------------------------------------------------
-- Generated by ttImportFromOracle 3.4.0 - Mon Feb  3 16:30:33 GMT 2014
--------------------------------------------------------------------------------
--
-- See end of file for table size totals
--
-- Source is Oracle Database 11.2.0.2.0
```

```
--
-- Character set is AL32UTF8
--
------------------------------------------------------------------------
--
-- Type mapping modes:
--            Numeric: 2 - AGGRESSIVE
--             Varlen: 1 - AGGRESSIVE
--  RAW/LONG/LONG RAW: 1 - YES
--               LOBs: 1 - YES
--  TIMESTAMP WITH TZ: 1 - YES
--
-- Compression mode: 2 - VARIABLE
--                   Target = 70%, minrows = 1024
--
------------------------------------------------------------------------

-- ************************************************************
-- Rows: 5
-- Bytes per row: Oracle   - max 2251
--                TimesTen - inline 209, not inline max 0, total 209
-- Estimated table size = 56363 bytes
-- Table does not qualify for compression
CREATE TABLE SCOTT.PARENTTAB
(
    P_PK                          TT_TINYINT NOT NULL,
-- Oracle type:                     NUMBER(10,0)

    P_C1                          TT_TINYINT,
-- Oracle type:                     NUMBER

    P_C2                          DATE NOT NULL,
-- Oracle type:                     DATE

    P_C3                          VARCHAR2(104 BYTE) INLINE,
-- Oracle type:                     VARCHAR2(200 BYTE)

    P_C4                          NVARCHAR2(26) INLINE
-- Oracle type:                     NVARCHAR2(1000)
);

-- ************************************************************
-- Rows: 3
-- Bytes per row: Oracle   - max 299
--                TimesTen - inline 100, not inline max 0, total 100
-- Estimated table size = 33205 bytes
-- Table does not qualify for compression
CREATE TABLE SCOTT.CHILDTAB
(
    C_PK                          TT_TINYINT NOT NULL,
-- Oracle type:                     NUMBER(6,0)

    C_FK                          TT_TINYINT,
-- Oracle type:                     NUMBER(10,0)

    C_C1                          VARCHAR2(80 BYTE) INLINE
-- Oracle type:                     VARCHAR2(255 BYTE)
);

------------------------------------------------------------------------
-- Estimated total uncompressed size for all tables: 89568 bytes
-- Estimated total compressed size for all tables:   89568 bytes
```

```
-- Estimated total actual size for all tables:       89568 bytes
--------------------------------------------------------------------------------

exit;
Disconnecting...
Done.
```

Now we will load the data. For simple tables with few rows such as these we could use the basic LoadData.sql script but for illustrative purposes will run the parallel data load script, ttPDL.sh.

```
$ ttPDL.sh "DSN=myttdb;UID=scott"
TimesTen Password for scott:
Oracle password for scott:

ttImportFromOracle parallel data load started at Wed Feb  5 16:30:32 GMT 2014

Work directory is '/tmp/ttPDL/wrk18193'
Data will be loaded using flashback query as of timestamp 2014-02-05 16:30:32

----------------------------------------
Executing Parallel Load Group: 1
Group started at: Wed Feb  5 16:30:32 GMT 2014
Total rows loaded: 8
Group finished at: Wed Feb  5 16:30:40 GMT 2014

----------------------------------------
Load completed: Log in ttPDLlog.txt
ttImportFromOracle parallel data load finished at Wed Feb  5 16:30:41 GMT 2014
```

Next we will create the indexes:

```
$ ttIsql -f CreateIndexes.sql myttdb
Copyright (c) 1996-2011, Oracle.  All rights reserved.
Type ? or "help" for help, type "exit" to quit ttIsql.

…
run "CreateIndexes.sql";

--------------------------------------------------------------------------------
-- Generated by ttImportFromOracle 3.4.0 - Mon Feb  3 16:30:33 GMT 2014
--------------------------------------------------------------------------------
--
-- See end of file for index size totals. Sizing estimates for alternate
-- index types are shown, enclosed in parentheses, for information only and
-- are not included in the total. Index types should be chosen based on
-- workload not on size.
--
--------------------------------------------------------------------------------

-- timing 1;

-- ************************************************************
-- Index: SCOTT.P_IX1
-- Table: SCOTT.PARENTTAB
-- Rows:  5
-- Ttree estimated size = 4074 bytes
-- (Btree estimated size = 6194 bytes)
-- (Hash estimated size = 3114 bytes)
```

```
CREATE UNIQUE INDEX SCOTT.P_IX1
          ON SCOTT.PARENTTAB
             (
                 P_C1
             );

-- *************************************************************
-- Index: SCOTT.P_IX2
-- Table: SCOTT.PARENTTAB
-- Rows:  5
-- Ttree estimated size = 4074 bytes
-- (Btree estimated size = 5170 bytes)
-- (Hash estimated size = 2922 bytes)
CREATE INDEX SCOTT.P_IX2
          ON SCOTT.PARENTTAB
             (
                 P_C2
             );

------------------------------------------------------------------------------
-- Estimated total index size for all tables = 8148 bytes
------------------------------------------------------------------------------

exit;
Disconnecting...
Done.
```

And finally we update statistics:

```
$ ttIsql –f UpdateStats.sql myttdb

…

-- timing 1;

call ttOptUpdateStats('SCOTT.PARENTTAB',1);

call ttOptUpdateStats('SCOTT.CHILDTAB',1);

quit;
Disconnecting...
Done.
```

And we are done!

7. Detailed Syntax and Description

```
Import table definitions and data from an Oracle database

Usage:

ttImportFromOracle [-h | -help | -? [full] ]

ttImportFromOracle [-V | -version]

ttImportFromOracle  -oraConn ouser[/opasswd][@oserver]
                    [-ttOwner ttuser]
                    [-tables tablespec [tablespec]...]...
                    [-tableFile filepath]...
                    [-importMviews { 0 | 1 }]
                    [-typeMap num[,var[,rawlong[,lob[,tstz[,padding[,stats]]]]]]]
                    [-compression level[,padding[,stats[,cfactor[,minrows]]]]]
                    [-inlineLimit n]
                    [-parallel dop]
                    [-dataLoadParallelism dlpspec]
                    [-indexes unique[,nonunique[,bitmap[,nodups]]]]
                    [-constraints pkey[,fkey]]
                    [-outputDir dirpath]
                    [-prefix fnprefix]
                    [-target { unix | windows }]
                    [-onBadType { 0 | 1 | 2 }]
                    [-verbosity { 0 | 1 }]

Options:

  -h or -help or -? [full]

     Display this help information on stdout. If 'full' is specified then
     this will include the 'Description' section otherwise it just displays
     the 'Options' section.

  -V or -version

    Display program version information on stdout.

  -oraConn ouser[/opasswd][@oserver]

     Specifies connection credentials for the source Oracle database.
       ouser      - The Oracle username.
       opasswd    - The password for 'ouser'. If this is omitted
                    the user will be prompted for the password.
       oserver    - Identifies the source Oracle database server. This
                    value may be a TNS name or an Easy Connect string
                    in the format 'hostname:portno/dbservice'.
                    'oserver' may be omitted if you are connecting
                    locally to the default database.

  -ttOwner ttuser

     Controls object ownership in TimesTen.
       ttuser     - The name of the TimesTen user who will own all the
                    imported objects. If this option is not used then
                    objects will have the same owners in TimesTen as in
                    the source database.

  -tables tablespec [tablespec]...
```

```
   Specifies tables to be imported.
     tablespec  - A table specification in the format:
                    [owner.]name[:attribute]...
   This option can be specified multiple times to build up a list
   of tables to be imported. It can also be combined with one or more
   occurrences of the -tableFile option. See the full help for details
   on the exact syntax of a table specification.

-tableFile filepath

   Specify a file containing a list of table specifications.
     filepath   - Path of a text file containing table specifications
                  of tables to be imported. These should be specified
                  one per line in the same format as for the '-tables'
                  option.
   This option can be specified multiple times and it can also be
   combined with one or more occurrences if the '-tables' option.

-importMviews { 0 | 1 }

   Controls whether materialized views are imported.
     0 (default) - Materialized views are not imported.
     1           - Materialized views are imported as regular tables.

-typeMap num[,var[,rawlong[,lob[,tstz[,padding[,stats]]]]]]

   Controls how source column data types are mapped to TimesTen types.

     num        - Controls type mapping for numeric columns.
                  Values are:
       0           - Type mapping is not performed.
       1 (default) - Standard numeric type mapping is performed (NUMBER
                     to TimesTen native integer where possible/safe).
       2           - Aggressive type mapping is performed for
                     NUMBER columns.

     var        - Controls type mapping for VARCHAR2 and NVARCHAR2 columns.
                  Values are:
       0 (default) - Type mapping is not performed.
       1           - Aggressive type mapping is performed for
                     VARCHAR2 and NVARCHAR2 columns.

     rawlong    - Controls type mapping for RAW, LONG and LONG RAW columns.
                  Values are:
       0           - Type mapping is not performed. RAW, LONG and LONG RAW
                     will be treated as unsupported data types.
       1 (default) - RAW(n) will be mapped to VARBINARY(n), LONG to
                     VARCHAR2(4194304) and LONG RAW to VARBINARY(4194304).
                     In addition, for RAW(n), 'var' mapping will then be
                     applied.

     lob        - Controls type mapping for LOB columns.
                  Values are:
       0           - Type mapping is not performed.
       1 (default) - CLOB will be mapped to VARCHAR2(4194304), NCLOB to
                     NVARCHAR2(2097152) and BLOB to VARBINARY(4194304).
                     In addition, 'var' mapping will then be applied.

     tstz       - Controls type mapping for TIMESTAMP WITH TIMEZONE columns.
                  Values are:
       0           - Type mapping is not performed. TIMESTAMP WITH TIMEZONE
```

```
                    will be treated as an unsupported data type.
      1 (default) - TIMESTAMP WITH TIMEZONE will be mapped to TIMESTAMP.
                    When the data is imported the timezone component will
                    be lost.

    padding     - This is an integer value representing a percentage
                  (i.e. 50 means 50%). When using num=2 or var=1,
                  increase the maximum precision/length and scale, as
                  determined by data analysis, by padding% in order to
                  allow for future changes in the source data. The minimum
                  (and default) value is 0 (no padding) and the maximum is
                  1000 (10x).

    stats     -   Controls whether additional data analysis statistics
                  relating to type mapping will be output, as comments,
                  in the generated DDL file.
                  Values are:
      0 (default) - Additional statistics are suppressed.
      1           - Additional statistics are included.

Examples:
  -typeMap 2,1            (num=2, var=1, rest are default)
  -typeMap 0,1,0,0,0,50   (num=0, var=1, rawlong=0, lob=0, tstz=0,
                           padding=50%, stats=default)

-compression level[,padding[,stats[,cfactor[,minrows]]]]

   Controls compression evaluation and clause generation.

    level       - Controls the compression type and level.
                  Values are:
      0 (default) - Source tables are not analyzed for compression.
      1           - Source tables are analyzed for compression using
                    the maximum dictionary pointer size for every column.
                    For qualifying tables, the resulting compression
                    clauses are output in the generated DDL files.
      2           - As for 1 but the dictionary pointer size is adjusted
                    based on the actual number of distinct values in the
                    data.

    padding     - This is an integer value representing a percentage
                  (i.e. 50 means 50%). When using level 1 or 2, increase
                  the value determined for the number of unique values by
                  padding% in order to allow for future changes in the
                  source data. The minimum (and default) value is 0 (no
                  padding) and the maximum is 1000 (10x).

    stats       - Controls whether additional data analysis statistics
                  relating to compression will be output, as comments,
                  in the generated DDL file.
                  Values are:
      0 (default) - Additional statistics are suppressed.
      1           - Additional statistics are included.

    cfactor     - The minimum compression factor that must be achieved
                  for a column/table in order for it to be considered
                  for compression. The value is specified in terms
                  of compressed size / uncompressed size expressed as
                  a percentage and so it is always > 0 and <= 100.
                  Smaller values represent higher compression. The
                  default is 70.
```

```
     minrows    - The minimum number of rows that a source table must
                  contain in order to be considered for compression.
                  The minimum allowed value is 1 and the default is 1024.

Example:
  -compression 1,0,1,60  (level=1, padding=0, stats=1, cfactor=60%,
                          minrows=default)

-inlineLimit n

   Control INLINE / NOT INLINE storage for variable length types.
     n          - VARCHAR2 and NVARCHAR2 columns whose maximum
                  defined size exceeds 'n' bytes will be stored
                  NOT INLINE otherwise they will be stored INLINE.
                  The minimum allowed value is 0 (always store NOT
                  INLINE), the maximum value is 32768 and the default
                  is 128.

-parallel dop

   Specifies the degree of parallelism used when querying the
   source database in the analysis phase.
     dop        - The degree of parallelism used for source table
                  analysis scans. Values can be in the range 0 to 128
                  (0 or 1 means do not use parallel scan). The default
                  is to use the default degree of parallelism defined
                  in the source database.
   This option has no effect on the parallel data load function.

-dataLoadParallelism dlpspec

   Specifies overall values and defaults for parallel data loading into
   TimesTen. This is an advanced usage option; in most cases the defaults
   will provide good performance. You should only use this option if you
   have a good understanding of the way in which parallel data loading is
   implemented. See the full help (-help full) or the documentation for
   more details on parallel data loading and the usage of this option.
   Parallel data loading is not supported on Windows.

-indexes unique[,nonunique[,bitmap[,nodups]]]

   Controls how indexes on source tables are imported into TimesTen.

     unique     - Controls the import of unique indexes.
                  Values are:
       0          - Unique indexes are not imported.
       1 (default) - Unique indexes are imported.

     nonunique  - Controls the import of non-unique indexes.
                  Values are:
       0          - Non-unique indexes are not imported.
       1 (default) - Non-unique indexes are imported.

     bitmap     - Controls the import of bitmap indexes.
                  Values are:
       0          - Bitmap indexes are not imported.
       1 (default) - Bitmap indexes are imported as range indexes.
       2          - Bitmap indexes are imported as bitmap indexes.

     nodups     - Controls whether duplicate indexes on the same table
                  are suppressed or imported. Values are:
       0          - Duplicate indexes are imported.
```

```
      1 (default) - Duplicate indexes are suppressed.
    TimesTen will give an error if you try and create multiple identical
    indexes (i.e. same set of columns, same type of index) on a table.
    This option prevents that situation.

 Example:
   -indexes 0,1,1,1   (unique=0, nonunique=1, bitmap=1, nodups=1)

-constraints pkey[,fkey]

   Controls how constraints on source tables are imported into TimesTen.

     pkey       - Controls the import of primary keys.
                  Values are:
       0           - Primary keys are not imported.
       1           - Primary keys are imported as unique indexes.
       2 (default) - Primary keys are imported as primary keys.

     fkey       - Controls the import of foreign keys.
                  Values are:
       0           - Foreign keys are not imported.
       1           - Foreign keys are imported as non-unique indexes.
       2 (default) - Foreign keys are imported as foreign keys.

   If 2 is specified for 'fkey' then either 1 or 2 must be specified
   for 'pkey'.

 Example:
   -constraints 1,1       (pkey=1, fkey=1)

 -outputDir dirpath

    Specifies the location for the output files.
      dirpath    - The path of the directory where the output files will
                   be written. The default is the current directory.

 -prefix fnprefix

    Controls the naming of the output files.
      fnprefix   - A prefix to be used for the output filenames.
                   May only contain alphanumeric characters, '_',
                   '-' and '.'.

 -target { unix | windows }

    Specifies the target platform for the output files and scripts.
      unix       -  Generate Unix/Linux compatible files and scripts.
      windows    -  Generate Windows compatible files and scripts.
    The default is the platform where the utility is being run.

 -onBadType { 0 | 1 | 2 }

    Specifies the action to take when an unsupported data type
    is encountered for a source table column.
     0           - Processing stops with an error message.
     1           - The current table is skipped but processing
                   continue. A warning message is issued.
     2 (default) - The current column is skipped but processing
                   continues. A warning message is issued.

 -verbosity  { 0 | 1 }
```

```
    Controls the volume and type of output messages.
     0           -    Only error and warning messages are output
     1 (default) -    Information and progress messages are output as
                      well as errors and warnings.

Description:

This utility will connect to the Oracle database identified by 'oserver'
using the supplied credentials. It will then examine a set of tables.
These tables can be specified in one of three ways:

1.    By using the '-tables' option

2.    By using the '-tableFile' option

3.    If neither '-tables' or '-tableFile' are specified then all tables
      owned by 'ouser' will be examined

Options (1) and (2) can be used separately or in combination and may occur
multiple times on the command line.

A table specification (tablespec) has the format:

  [owner.]name[:attribute]...

Owner and/or attributes are optional.  A table name of '*' (which may need
to be quoted if used on the command line) will be expanded to represent all
tables belonging to the owner. Any database identifier (e.g. schema owner,
table name, username etc.) must be properly quoted if it does not conform
to the standard rules for database identifiers.

The attributes are a set of colon (:) separated key/value pairs most of which
control aspects of parallel data loading (see later for more details on
parallel data loading). The supported attributes are:

  rows=n
      Specifies the rowcount for the table. 'n' must be an integer between 0
      and and 9223372036854775807. Normally the rowcount for a table is
      determined when the table is analyzed. Specifying a rowcount can avoid
      a table scan in cases where the only reason to analyze the table is to
      determine the rowcount. Knowing the rowcount for a table is essential
      for the compression evaluation, table size estimation and parallel
      data load functions. There is no default value.

  ttwriters=w
      Specifies the maximum number of TimesTen threads, per reader, to be
      used when performing intra-table parallel load for this table. 'w'
      is an integer value between 1 and 16. The default value is determined
      by the 'ttthreads' value (minus 1) specified in the -dataLoad-
      Parallelism option.

  oradop=n
      Specifies the Degree of Parallelism to be used for each Oracle reader
      in the Oracle database. 'n' is an integer value between 0 and 16. 0
      or 1 mean 'no parallel query' in Oracle. The default value is determined
      by the 'oradop' value specified in the -dataLoadParallelism option.
      Note that whether the value specifed is honoured or not depends on the
      configuration of the Oracle database.

  orapar={none|auto[(n)]|hash[(n[,colexpr])],partition[(n)]}
      Specifies how intra-table parallel data load will be executed for
      tables matching this specification. The meaning of the different
```

values is as follows:

```
  none                - Table will not be loaded in parallel.

  hash[(n[,colexpr])] - Table will be loaded using 'n' readers.
                        Rows will be distributed across the readers
                        based on ORA_HASH(colexpr,n-1). If 'colexpr'
                        is not specified then ROWID will be used.
                        If 'colexpr' is specified as '=PK=' then a
                        concatenation of the table's primary key
                        columns will be used - in this case it is an
                        error if the table does not have a defined
                        primary key. If 'n' is not specified then the
                        value used is the default value specified in
                        the -dataLoadParallelism option or 8 otherwise.

  partition[(n)] -      Table will be loaded using 'n' readers, one
                        per table partition. If the table is not a
                        partitioned table then an error is issued.
                        If the table has more than 'n' partitions
                        then the load of the table will be split
                        into multiple separate parallel load jobs
                        each with (at most) 'n' readers. If 'n'
                        is not specified then the value used is
                        the default value specified in the -dataLoad-
                        Parallelism option or 8 otherwise.

  auto[(n)]      -      If the table is partitioned then it will be
                        loaded as for 'partition[(n)]' otherwise it
                        will be loaded as for 'hash[(n)]'
```

The default is 'auto'. Tables are always considered for inter-table parallel load unless parallel data load is completely disabled. If the 'orapar' attribute is used it must be the last attribute in the list.

The -dataLoadParallelism option allows you to override various built-in limits and defaults that control parallel data loading. It should rarely be necessary to change any of these values but if you feel you need to then here is the syntax. The 'dlpspec' parameter is a string in one of the following formats:

disabled
Parallel data load is disabled. Tables are loaded sequentially with no parallelism of any kind. Any parallel data load attributes specified in table specifications are silently ignored.

minrows,ttthreads,tttabwriters,orareaders,oradop,consistency,orapar
Sets parameters to control parallel data loading defaults:

```
      minrows       - An integer specifying the minimum number of rows
                      a table must contain before it will be considered
                      for intra-table parallel load. A value of 0 means
                      all tables will be considered. The default is
                      500,000.

      ttthreads     - The maximum number of concurrent threads allowed
                      in TimesTen during parallel data loading. Each
                      reader is a thread as is each writer. Must be
                      >= 0.  0 means no limit. The default is 36.

      tttabwriters  - The default maximum number of writers per reader
```

```
                    when performing intra-table parallel load on a
                    table in TimesTen. Must be >= 1 and <= 16. The
                    default is 3.

          orareaders    - The maximum number of concurrent readers allowed
                    in the Oracle database during parallel data
                    loading. Must be >= 0. 0 means no limit and is
                    the default.
                    It is not permitted for both 'ttthreads' and
                    'orareaders' to be 0.

          consistency   - Specifies what data consistency is required when
                    loading data. Allowed values are:
                      0           - No consistency enforcement. If
                                    the table data in Oracle is
                                    modified in any way while the
                                    load is running then the loaded
                                    data may be inconsistent.

                      1 (default) - All data loads are performed using
                                    Oracle flashback queries based
                                    on the time when the load script
                                    starts running.

                      2           - All data loads are performed using
                                    Oracle flashback queries based
                                    on the current SCN when the load
                                    script starts running. In order
                                    to use this option your Oracle
                                    database user must have been
                                    granted EXECUTE privilege on the
                                    DBMS_FLASHBACK package.

          oradop        - The default degree of parallelism used for each
                    reader in the Oracle database during parallel
                    data loading. Must be >=0 and <= 16. 0 and 1 both
                    mean no parallel query. The default is 1.

          orapar        - Specifies the default mechanism for intra-table
                    parallel load. Allowed values are 'none',
                    'auto[(n)]', 'partition[(n)]' or 'hash[(n)]'.
                    See the description of 'orapar' in the table
                    specification section above for the meaning of
                    each of these. The default is 'auto(8)'.

Parallel data loading is not supported when the target platform is Windows.
When running the utility on Windows you can generate a parallel data load
script for execution on Unix/Linux by specifying '-target unix'. When the
target platform is Windows, the utility behaves as if the option
'-dataLoadParallelism disabled' had been specified.

Parallel data load is executed as follows. For each table for which intra-
table parallel load is enabled a number of Oracle database readers are
assigned based on explicit configuration and/or the relevant defaults.
Each reader will process a subset of the total table rows based on one of
the mechanisms (partition or hash) described above. Each reader, when
querying the Oracle database to retrieve the rows, will use parallel
query with a DOP value as specified (or non-parallel query if DOP=0 or 1).
For each reader, there will be a set of writers inserting the data into
the table in TimesTen. The number of writers per reader will be the value
specified by the 'ttwriters' attribute (or the 'tttabwriters' default).
When loading a table the limits imposed by total Oracle readers and total
```

```
TimesTen threads values, as specified in the -dataLoadParallelism option will
always be respected.

For inter table parallelism, tables are grouped into sets to be loaded
together based on their intra-table parallelism and the overall limits
defined for Oracle database readers, and TimesTen threads as specified
in the -dataLoadParallelism option.

Some examples of valid table specifications are:

  sometable                 - table 'ouser.sometable'
  scott.tab2                - table 'scott.tab2'
  customer:rows=125000      - table 'ouser.customer' with rowcount 125,000
  *                         - all tables owned by 'ouser'
  scott.*                   - all tables owned by 'scott'
  scott.*:rows=1000         - all tables owned by 'scott' applying a
                              rowcount of 1000 to all of them

For the purposes of examination and analysis materialized views are
treated as tables. This can be controlled by the -importMviews option.
Regular views are not imported.

For each table examined, a TimesTen compatible definition is generated
which may include some or all of primary key definitions, foreign key
definitions, unique indexes, non-unique indexes, bitmap indexes,
recommended data type mappings and compression clauses. Many options are
available to control what is included in the generated DDL.

The program will perform data type mapping from source to TimesTen in order
to optimize space usage within the TimesTen database. There are various sorts
of type mapping that can be applied:

    Numeric:            Columns of type NUMBER with scale of zero are
                        mapped to a suitable TimesTen native integer type.
                        This is called 'standard' mapping and it is safe
                        (no danger of data truncation). This mapping does
                        NOT require analysis of the source data.
                        A more aggressive mapping is also available; an
                        analysis is performed on the source tables looking
                        for columns of type NUMBER where the maximum precision
                        or scale of any value stored is smaller than the value
                        defined for the column. For these columns, a smaller
                        precision and/or scale is used for the TimesTen
                        definition. This mapping is only safe for the data
                        present in the source tables at the time the analysis
                        is performed; if new data is added to, or data is
                        updated in, the source tables then that data may
                        no longer meet the criteria used for the mapping (if
                        the new/modified data increases the maximum value
                        stored in a column).

    Varlen:             An analysis is performed on the source tables looking
                        for columns of type VARCHAR2 and NVARCHAR2 where the
                        maximum length of data stored is less than the defined
                        size of the column such that space savings can be
                        realized by defining the column to be smaller in
                        TimesTen. This mapping is only safe for the data
                        present in the source tables at the time the analysis
                        is performed; if new data is added to, or data is
                        updated in, the source tables then the data may no
                        longer meet the criteria used for the mapping (if
                        the added/updated data now exceeds the maximum
```

```
                    column size).

    RAW/LONG/LONG RAW: TimesTen does not support the data types RAW, LONG or
                    LONG RAW. With this mapping, source columns of those
                    types are mapped to compatible types in TimesTen.
                        RAW(n)    -> VARBINARY(n)
                        LONG      -> VARCHAR2(4194304)
                        LONG RAW -> VARBINARY(4194304)
                    The maximum size for LONG and LONG RAW items in Oracle
                    database is 2 GB while the maximum size for VARCHAR2
                    and VARBINARY items in TimesTen is 4 MB so not all
                    possible values will fit in TimesTen. After this
                    mapping has been applied any VARBINARY columns
                    resulting from mapping RAW(n) columns will also have
                    'varlen' mapping applied if it was specified in the
                    type mapping options'

    LOB:            TimesTen supports CLOB, NCLOB and BLOB types. However,
                    currently the ttLoadFromOracle() builtin cannot load
                    data into LOB columns. So, it may be useful to map LOBs
                    to compatible types in TimesTen so that the data may
                    be loaded.
                        CLOB  -> VARCHAR2(4194304)
                        NCLOB -> NVARCHAR2(2097152)
                        BLOB  -> VARBINARY(4194304)
                    Oracle database LOBs may be very large while TimesTen
                    variable types have much smaller maximum sizes so again
                    there may be issues with some values being too large.
                    When LOB columns are mapped to VARCHAR2/VARBINARY
                    'varlen' mapping will also be applied if it was
                    specified in the type mapping options.

The aggressive type mappings (num=2, var=1) and compression evaluation require
a FULL SCAN of the source tables in order to analyze the data therein.

When evaluating a table for compression three factors are considered,
controlled by parameters to the '-compression' option:

- The number of rows in the table (minrows). Tables with less than this
  number of rows will never be compressed. The minimum value that can be
  specified is 1 (always consider compressing the table) and the maximum
  value is  9223372036854775807. The default is 1024.

- The padding factor (padding). Once the number of distinct values in a column
  has been determined (by data analysis) it is increased by 'padding' percent
  to allow for future changes in the source data.

- The minimum required compression factor (cfactor). First, an estimation
  is made of how much storage will be saved by compressing each table
  column. If the estimated compression factor (estimated compressed size
  divided by the estimated uncompressed size) is lower than 'cfactor'
  (lower values mean higher compression) then the column is a candidate
  for compression. Once all columns have been evaluated for compression,
  an overall estimation is performed at the table level to see how much
  the entire table would be compressed if all the candidate columns were
  compressed. If this estimation yields a compression factor less than or
  equal to 'cfactor' then the table will be compressed. The smallest
  value that can be specified for 'cfactor' is 1% (100x compression)
  and the highest is 100% (no compression). The default is 70%.

NOTES:
```

1. If a table needs to be scanned for multiple purposes (rowcounts, type, mapping, compression evaluation) then this will all be accomplished in a single scan. The same table will not be scanned multiple times.

2. Any tables for which rowcounts are not specified will be scanned to determine their rowcounts. These rowcounts will be output in the generated tableList.txt file.

3. By default, when analysing data, the program will utilize parallel query in the source database using the system default level of parallelism. Parallel query can be disabled using the '-parallel 0' option or an explicit degree of parallelism can be specified using the '-parallel' option with a suitable value in the range 2 through 128. Whether the requested degree of parallelism can actually be used or not depends on the configuration of the source database. The '-parallel' option only affects the use of parallel query for data analysis. Use (or not) of parallel query during data loading operation is controlled separately; see the sections on the '-dataLoadParallelism' option and table specification attributes for details.

4. Scanning and analyzing source data can be an expensive and time consuming operation. The actual time required depends on many factors such as the number of rows in the table, the number and types of the columns being analyzed, the performance of the source database etc. In general determining rowcounts and optimizing VARCHAR2/NVARCHAR2 storage (-typeMap x,1) are significantly faster/cheaper than evaluating compression (-compression 1 or 2) and optimizing NUMBER types (-typeMap 2).

Only certain source data types are supported by TimesTen. If the utility encounters a source table where a column has an unsupported or unrecognized data type then by default it will ignore that column (after reporting the problem) and continue processing. You can use the '-onBadType' option to modify this behavior. The default is 2 (skip the column). Other possible values are 0 (stop processing) and 1 (skip the table).

The following files will be created in the output directory (the default is your current directory):

[fnprefix]TableList.txt	-	A list of all tables evaluated in this run in a format suitable for use with the '-tableFile' option
[fnprefix]CreateUsers.sql	-	SQL script to create the TimesTen database users
[fnprefix]DropTables.sql	-	SQL script to drop all the tables
[fnprefix]CreateTables.sql	-	SQL script to create all the tables
[fnprefix]DropIndexes.sql	-	SQL script to drop all the indexes and foreign keys
[fnprefix]CreateIndexes.sql	-	SQL script to create all the indexes, primary keys and foreign keys
[fnprefix]UpdateStats.sql	-	SQL script to update TimesTen optimizer statistics for each table
[fnprefix]LoadData.sql	-	SQL script to load data into TimesTen tables from the source tables; tables with LOB columns will not be included in this script

```
[fnprefix]ttPDL.sh            -   Shell script to load table data using
                                  parallel data load. Use this instead of
                                  the LoadData.sql SQL script if you wish
                                  take advantage of the parallel data load
                                  capability. This script is not generated
                                  when the target is Windows.
```

The SQL scripts can be run from within ttIsql after connecting to the TimesTen database as a suitable user. The scripts can be run from the ttIsql command prompt, after connecting to the datastore, as follows:

```
Command> @CreateUsers;
```

Alternatively, the scripts can be run in 'batch mode' from the O/S command line as follows:

```
$ ttIsql -f CreateUsers.sql MyDSN
```

In general, the SQL scripts must be run as the TimesTen instance admin user (the user who installed TimesTen) or as a user who has suitable permissions on all the tables.

In order to create tables, the user[s] who own them must have already been created in the TimesTen database (you can use the 'CreateUsers.sql' script to do this). Only the instance administrator can create users.

The UpdateStats.sql script can be run as the instance administrator user or as the user which owns the affected tables. If run as any other user then that user must have the ALTER ANY TABLE system privilege.

The LoadData.sql script must be run as a TimesTen database user which also exists in the backend Oracle DB. You must specify the TimesTen and Oracle user credentials as part of the connection string. For example:

```
  ttIsql -f LoadData.sql "DSN=MyDSN;UID=scott;PWD=tiger;OraclePWD=tiger"
```

In addition, the TimesTen DSN that is used must either be configured for connectivity to the source Oracle database. i.e. the DSN definition must include the OracleNetServiceName attribute with a suitable value, or the OracleNetServiceName attribute, with a suitable value, must be passed as part of the connection string.

The ttPDL.sh script requires a TimesTen connection string that includes a DSN and a UID. You may optionally include PWD and/or OraclePWD but if you do not then the script will prompt you for their values (which is more secure than specifying them on the command line). For example:

```
  ttPDL.sh "DSN=MyDSN;UID=scott"
```

8. Caveats and Limitations

There are many significant differences between the Oracle database and the TimesTen database, thus there are many DDL related features available in the Oracle database that cannot be properly mapped to a TimesTen database. ttImportFromOracle focuses on specific areas of functionality so as a result there are naturally some limitations. The list below highlights some of the more important caveats and limitations, but it is not guaranteed to be a complete or definitive list.

1. The Oracle Database Extended Types feature (12c and later) is not supported. The utility will refuse to run against an Oracle database configured with MAX_STRING_SIZE = 'EXTENDED'.

2. The only database objects that can be imported are users, tables, materialized views, unique constraints, primary key constraints, foreign key constraints and indexes. No other object types are supported.

3. When importing users only the username is imported. No other attributes of the user are imported from Oracle. A default password, equal to the lower case username, is assigned in the TimesTen script and basic privileges (CREATE SESSION and CREATE TABLE) are granted. You should edit this script before running it if a different password or additional privileges are required.

4. Materialized views can only be imported as tables, not as actual materialized views (i.e. the view and its data will end up as a normal table in TimesTen). To import materialized views as tables you must specify the option '–importMviews 1'. The default is to not import materialized views.

5. Oracle database supports circular and self referencing foreign keys while TimesTen does not. ttImportFromOracle will detect circular and self referencing foreign keys and will, if foreign key import has been specified, import them as non-unique indexes (a warning is issued should this occur).

6. Any foreign key constraint that does not reference a table that is being imported will not be imported as a foreign key constraint. If the option to import foreign keys as non-unique indexes (-constraints x,1) is used it will be imported as a non-unique index instead, otherwise it will be ignored.

7. Any constraint that maps to an index that is not a candidate for import will be ignored (with a suitable warning message).

8. The parallel data load function (**ttPDL.sh**) does not take account of any foreign key constraints and may fail if executed while foreign keys are in place. For this reason it is recommended that data loading be performed prior to creating indexes and constraints, or that indexes and constraints be dropped prior to data loading and re-created afterwards. This is easily accomplished using the **DropIndexes.sql** and **CreateIndexes.sql** scripts that are generated by the utility.

9. Columns in Oracle database tables with data types not supported by TimesTen (user defined types and XML types for example) will not be imported. They will be handled based on the setting of the '–onBadType' option.

10. TimesTen does not support the data type TIMESTAMP WITH TIMEZONE. When this type of column is imported it is not possible to import the timezone information which is therefore lost. An option is provided (-typeMap x,x,x,x,0|1) to allow the user to choose between the default behavior (import as TIMESTAMP losing the timezone information – a warning is issued) or to treat this as an unsupported data type.

11. Column 'DEFAULT' value clauses are not imported.

12. Only regular and bitmap indexes are imported. All other types of index are ignored. An option is provided to convert bitmap indexes to range indexes (this is the default).

13. Only constraints in an ENABLED, VALIDATED state will be imported.

14. Only indexes in VALID, VISIBLE state will be imported.

15. Oracle database allows 'duplicate' indexes (same set of columns in the same order) to be created on a table while TimesTen does not. By default ttImportFromOracle will suppress all except one of any duplicate indexes. This behavior can be disabled by setting the 'nodups' parameter of the '-indexes' option to 0 (-indexes x,x,x,0) but in this case if any duplicate indexes are present then some errors are to be expected when running the **CreateIndexes.sql** script.

9. Troubleshooting

In the event of any problems, the utility includes a debugging and tracing facility. This is controlled by an environment variable called TTIMPORTDEBUG.

To utilize this facility set the variable as follows from O/S shell prompt before running the program.

Unix/Linux sh/ksh/bash:

```
export TTIMPORTDEBUG=5:./debug.txt
```

Unix/Linux csh:

```
setenv TTIMPORTDEBUG 5:./debug.txt
```

Windows:

```
set TTIMPORTDEBUG=5:.\debug.txt
```

This will cause debug information to be written to the file ‘debug.txt’ in your current directory (a different pathname can be used if desired).

When requesting support, please submit the debug file (compressed, as it may be quite large), along with the generated output files, a description of the problem and any error messages output by the utility.

To turn off debugging (to avoid the overhead and the creation of the debug file) you can clear the environment variable as follows.

Unix/Linux sh/ksh/bash:

```
unset TTIMPORTDEBUG
```

Unix/Linux csh:

```
unsetenv TTIMPORTDEBUG
```

Windows:

```
set TTIMPORTDEBUG=
```

10. Changes in this release

Changes for release 3.8.0 from release 3.7.1

- Updated build OS and compiler versions to address compatibility issues with newer OS versions.
- Reduced the set of supported platforms.

Changes for release 3.7.1 from release 3.7.0

- When running against an Oracle 10.2 database, automatically fall back to using the non-PLSQL analysis code.
- When running against an Oracle 12c database, check if the database is configured for extended types (MAX_STRING_SIZE = 'EXTENDED') and if so refuse to continue.
- Reduced the list of supported platforms (dropped all 32-bit platforms).

Changes for release 3.7.0 from release 3.6.0

- Fixed a problem where very long, but valid, lines in a table file would be reported as invalid.

Changes for release 3.6.0 from release 3.5.0

- Fixed a problem that caused a crash (SEGV) if the **rows=nnnn** option is used on a tablespec.
- Added support for Mac OS X 64-bit. This requires the use of the TimesTen 11.2.2.7.8 (or later) 64-bit client. Mac OS X 32-bit is still supported using the TimesTen 11.2.1 32-bit client.

Changes for release 3.5.0 from release 3.4.0

- Switched to using PL/SQL based type mapping and compression evaluation by default. Users can select the previous C/OCI/SQL method via the undocumented command line switch '-plsql 0'.
- Previously, if ttImportFromOracle was run with an invalid connection string such that it could not connect to the Oracle database it would still create (empty) SQL and script files. This is no longer the case.

Changes for release 3.4.0 from release 11.2.2.6.0 / 3.3.0

- The version of ttImportFromOracle that is shipped with TimesTen has been frozen at the version (3.3.0) that was shipped as part of the TimesTen 11.2.2.6.0 release. Future enhancements, and any

bug fixes, will be delivered in newer versions posted on the OTN TimesTen download page: http://www.oracle.com/technetwork/indexes/samplecode/utilities-2180928.html

- Detailed sizing information is now computed for indexes and is output as comments in the generated CreateIndexes.sql file.

- The ttSizing.sh / ttSizing.cmd scripts are no longer generated since all necessary sizing estimates are output directly by ttImportFromOracle in the CreateTables.sql and CreateIndexes.sql files.

- The ttPDL.sh script no longer requires you to specify the PWD and OraclePWD attributes, and their values, on the command line. If either is not specified then the script will securely prompt you for the password(s).

- The utility is now supported on the following platforms:

  Microsoft Windows 32 and 64 bit
  Linux x86 32 and 64 bit
  Solaris SPARC 32 and 64 bit
  Solaris x86 64-bit
  AIX 32 and 64 bit
  Mac OS X 32-bit (using the TimesTen 11.2.1 32-bit client)

Changes for release 11.2.2.6.0 from release 11.2.2.5.0

- Commented out the 'timing 1;' command in the generated scripts to make the scripts SQL Developer compatible. Also changed from using 'statsupdate' to 'call ttOptUpdateStats()' for the same reason. All generated SQL scripts are now compatible with SQL Developer.

- Improved type optimization and padding behavior for NUMBER columns that have non-integral values.

- Added some 'sum of table sizes' comments to end of CreateTables.sql file and a comment at the start of the file directing the user to the end of the file for the information.

- Added the capability to detect and suppress duplicate indexes (same set of columns in same order) defined on a table. Suppressing duplicates is the default but an additional option on the '-indexes' option allows for disabling this behavior. TimesTen does not allow multiple identical indexes to be created on the same table.

- A problem where some command line options incorrectly accepted non-numeric values has been fixed.

- A problem where ttImportFromOracle was not properly quoting table owner/name in the generated ttLoadFromOracle() calls in the ttPDL.sh script has been fixed.

- A problem causing partitioned indexes to not be imported has been fixed.

- The names of the log and error files produced by the ttPDL.sh script now include a date/time stamp to prevent script script runs from overwriting the logs from a previous run.

- The generated ttPDL.sh script now supports a –noflashback command line option to force the script to run without using flashback query regardless of what was specified in the –dataLoadParallelism option when ttImportFromOracle was run.

- A problem where the generated parallel data load script, ttPDL.sh, did not correctly implement the –ttOwner value specified when ttImportFromOracle was run has been fixed.

- A problem has been fixed where excessive resource usage might occur on the Oracle database server that ttImportFromOracle was running against.

11. Known Issues

- If an Oracle owner name (user name, schema name) or table name needs to be quoted then it will cause the UpdateStats.sql script to fail. This is due to an issue in TimesTen which may be addressed in a future TimesTen release. There is no workaround.

- If an Oracle object name is the same as an Oracle or TimesTen reserved word then it will cause various scripts to fail. This is primarily due to an issue in TimesTen which may be addressed in a future TimesTen release. In some cases, the problem can be worked around by editing the generated scripts to add double quotes around the object name. However, this will not resolve the issue in all cases (UpdateStats.sql script).

- If the set of tables being processed contains an owner name (user name, schema name) or table name that differs only in the case of letters (for example "ABC" and "abc") then it will not be possible to create some of the objects in TimesTen since currently TimesTen does not treat object names in a case sensitive manner even when they are quoted. This issue may be addressed in a future TimesTen release.