REPSI Tool

(Recording the Efficiency of Patterns / SQL Idioms)

(Version 02.00)

User Manual

Walter Weinmann

20th February, 2007

Contents

List of Figures

List of Tables

List of Abbreviations

API	Application Programming Interface
DBMS	Database Management System
DDL	Data Definition Language
DML	Data Manipulation Language
IDE	Integrated Development Environment
J2SE	Java Standard Edition
JDBC	Java Database Connectivity
JRE	Java Runtime Environment
JVM	Java Virtual Machine
n/a	Not applicable / not available
RDBMS	Relational Database Management System
REPSI	Recording the Efficiency of Patterns / SQL Idioms
SDK	Software Development Kit
XML	Extensible Markup Language

Chapter 1 Introduction

1 Purpose and Functionality of the Tool

The REPSI (**R**ecording the **E**fficiency of **P**atterns / **S**QL **I**dioms) tool enables a response time comparison of two SQL queries, for example one without the application of a pattern (or SQL idiom) and one with. This allows the user to gain an understanding of the extent to which the applied pattern might improve the efficiency in the execution of the SQL query. To do this the tool is designed with the following functionality:

- Creation and maintenance of a master database containing:
  - Documentation of patterns and SQL idioms,
  - Pairs of SQL queries related to a certain pattern,
  - Test suites consisting of predefined sequences of actions such as:
    - Drop and / or create database tables in the test database,
    - Generate rows in a test database table, or
    - Execute pairs of SQL queries,
  - Description of runtime environments including information about processors, operating systems, and test database instances.
- Running a pair of SQL queries in a given frequency of occurrence (calibration mode) thereby recording in the master database, information about the runtime environment, the response times, and a variety of statistics, e.g. kurtosis, skewness, standard deviation, and variance.
- Running a test suite (trial mode) and recording the complete runtime information in the master database.
- Exporting the recorded information from the calibration runs or trial runs in Excel style files.
- Exporting the response time data from the calibration runs into flat files which can be imported into R (a language and environment for statistical computing and graphics) and then processed by R.

Based on the results of running the tool with a specific database, an SQL user should then have a guide to improve the efficiency of the specific database optimiser by the application of suitable patterns and SQL idioms when formulating SQL queries.

2 About this Document

This document serves as a user manual for the REPSI tool. It provides instructions to install, configure and to run the tool. The audiences of this tool are mainly database administrators and SQL developers, who either plan to test their own pattern and SQL idiom ideas, or want to test existing patterns or SQL idioms in their own database environment.

The REPSI user needs a profound knowledge of SQL and of the target database environment. For modifying or extending the tool the user also needs a good knowledge of Java and JDBC (Java Database Connectivity) (especially the meta data (data dictionary) related classes, e.g. `DatabaseMetaData` or `ResultSetMetaData`).

The rest of the document proceeds as follows:

- Chapter 2: Describes the software requirements and the installation of the binary and development versions. It also gives complete instructions on how to run the binary system by providing a worked example. Finally the limitations are explained.
- Chapter 3: Delineates the major concepts on which the tool is built.
- Chapter 4: Provides step by step instructions for the most important tasks that the tool can perform.
- Chapter 5: Presents a catalogue of patterns and SQL idioms which, when applied, should improve the efficiency of executed SQL statements.
- Chapter 6: Provides specific details around some key topics for reference.

References to literature and web sites and a bibliography can be found in the appendices.

3 Document Conventions

`Courier`	Courier font is used to represent commands, source code, or table names
`{…}`	Mandatory elements are enclosed by curly brackets
`(…)`	Optional elements are enclosed by brackets
`|`	Choices are separated by vertical lines
`...`	Repeating elements are expressed by three consecutive dots

4 Change Protocol

Changes since Version 01.30

- Changes in the data model:
  - Unique key in table `TMD_CALIBRATION` extended to enable several schemas in the same database instance
- Resolved bugs:
  - The comparison of the result sets treats the Oracle (SQL) data types CHAR and VARCHAR2 as equal.
  - The SQL trace functionality with Oracle® Databases was not enabled.
- Changes of functionality 'Calibration':
  - R function boxplot.stats added to R code.
  - If the column '`APPLIED_PATTERN_ORDER_BY`' or '`UNAPPLIED_PATTERN_ORDER_BY`' contains the value 'NOSORT' (written in any case), then the query result will not be sorted before the comparison.
- Changes of functionality 'Trial Run':
  - If the column '`APPLIED_PATTERN_ORDER_BY`' or 'UN`APPLIED_PATTERN_ORDER_BY`' contains the value '`NOSORT`' (case insensitive), then the query result will not be sorted before the comparison.
  - Was by accident not implemented in 01.30: The comparison of the result sets of both queries will only be performed if the test query pair contains a value in both columns '`APPLIED_PATTERN_ORDER_BY`' and '`UNAPPLIED_PATERN_ORDER_BY`' .

Changes since Version 01.20

- Changes in the data model:
  - Unique key in table `TMD_CALIBRATION` extended to enable several schemas in the same database instance
- Resolved bugs:
  - `NullPointerException` in `ResultSetComparator`, if the `ResultSet` returns `NULL`
- Changes of functionality 'Calibration':
  - Where the test database is an Oracle® Database, the SQL trace functionality is automatically enabled with the open connection and disabled with the close connection operation.

    - The comparison of the result sets of both queries will only be performed if the test query pair contains a value in both columns 'APPLIED_PATTERN_ORDER_BY' and 'UNAPPLIED_PATERN_ORDER_BY' .
- Changes of functionality 'Trial Run':
    - The comparison of the result sets of both queries will only be performed if the test query pair contains a value in both columns 'APPLIED_PATTERN_ORDER_BY' and 'UNAPPLIED_PATERN_ORDER_BY' .

Changes since Version 01.00

- Changes in the data model:
    - New tables TMD_CALIBRATION and TMD_CALIBRATION_STATISTICS have been added
    - New foreign keys TMD_TLR_TMD_DEI_FK and TMD_TLR_TMD_TTS_FK in table TMD_TRIAL_RUN have been added
- New input option via Excel file format:
    - The database instance can be maintained via standardised Excel files
- New functionality 'Calibrate':
    - The main purpose of the new calibration function is to allow the user to run a pair of SQL queries with a predefined number of cycles. If there is more than one cycle required, the REPSI tool calculates and stores a set of statistics.
- Changes of functionality 'Trial Run':
    - New options '**-exalt**' and '**-excon**' to control the sequence of the query execution.
    - New option '**-ign1**' to ignore the first reading.
    - New option '**-prec 999**' to define the required precision of the response time.
- The Javadoc documentation shall be deemed to be complete.

5 Acknowledgments

Many thanks go to the world wide community of open software developers whose very valuable contribution greatly facilitates the development of tools like REPSI. My special thanks go to the people behind Apache Ant, Commons CLI, and Commons Math, Checkstyle, Eclipse, FindBugs, JExcelAPI, PMD, R, Subclipse, and Subversion.

Trademarks:

6 Licence Agreement

The REPSI tool is licensed under the following BSD (Berkeley Software Distribution) style licence:

Chapter 2 Getting Started

1 Software Requirements

The REPSI tool was developed and tested in a software environment consisting of the following components:

- Operating system Microsoft Windows XP (including service pack 2),
- Relational database management system Oracle® Database 10g Release 2 Enterprise Edition (Oracle Technology Network, 2006),
- Java Development Kit (JDK) and Java Runtime Environment (JRE) from Sun Microsystems in version 5.0 (Sun Microsystems Inc., 2006),
- Jakarta Commons CLI version 1.0 (Apache Software Foundation, 2006a) an API (Application Programming Interface) for processing command line arguments,
- Jakarta Commons Math version 1.1 (Apache Software Foundation, 2006b) an API for solving mathematical and statistical problems, and
- JExcelApi version 2.6.2 (Khan, 2006) an API to read, write and modify Microsoft Excel spreadsheets.

The REPSI tool allows the user to create files containing R code out of the calibration run data. These files comprise the definition of vectors containing the response time values and the summary and boxplot functions applied to the vectors. The content of the files was tested using the following software version:

- R Project for Statistical Computing version 2.4.0 (R Foundation, 2006) a language and environment for statistical computing and graphics.

For pure development purposes the following tools and components were used:

- Eclipse SDK (Software Development Kit) version 3.3 M4 (The Eclipse Foundation, 2006) as a Java IDE (Integrated Development Environment), including the following plug-ins:
  - Static source code analyser Checkstyle version 4.1.1 (Schneider and Ködderitzsch, 2006),
  - Static source code analyser FindBugs version 1.1.3 (University of Maryland, 2006),

  - Static source code analyser PMD for Eclipse version 3.1.7 (InfoEther, 2006), and
  - Subversion integration with Subclipse version 1.0.4 (CollabNet, 2006).
- Oracle® Designer 10g (Oracle Technology Network, 2006) for database modelling purposes, and
- TOAD version 9.1 (Quest Software, 2006) as an Oracle® Database IDE.

To successfully run the REPSI tool the user of the REPSI tool should consider the following restrictions:

- The REPSI tool is written entirely in Java and should, therefore, be platform independent with respect to the operating system. However, due to the extensive use of generics (parameterised types) (Schildt, 2004) during development, at least version 5 of the Java SDK (development version) or Java JRE (binary version) is required.
- Conformity to SQL standards is defined at different levels (Melton and Simon, 2002) e.g. SQL-92 defines 'Entry SQL', 'Intermediate SQL', and 'Full SQL', whereas SQL:1999 defines 'Core SQL'. Database vendors usually strive for conformity only at the lowest level and furthermore even extend their implemented dialect with proprietary features. The REPSI tool tries to preserve independence of a specific RDBMS (Relational Database Management System) by including SQL rewriting capabilities. This requires the connection of deployed relational database systems and specific SQL statements by a so called SQL syntax code which characterises the SQL dialect in place. A yet very rudimentary rewriting functionality can then perform the translation between different syntactical versions (see details on page 16 "SQL Syntax Code and SQL Rewriting").

2 Installation – Binary Version

The binary version of the REPSI tool can be downloaded from the following URL: https://developer.berlios.de/project/showfiles.php?group_id=7172. The downloaded file named '`REPSI_Tool_vv.vv.zip`' (where 'vv.vv' is the current version number, e.g. 02.00) is of type zip and has therefore to be installed by unzipping this file with an appropriate tool into a new or existing directory. A detailed description of the structure of the resulting directory and file layout can be found on page 63 "File and Directory Layout – Binary Version".

3 First Steps – Binary Version

After the installation of the binary version, the master database and the test database (both schema and instance) can be set up by performing the following activities.

In addition this section also explains how to execute and evaluate a calibration run.

Most of the activities have to be performed using the command line tool (see details on page 51 "Command Line Interface"). In a Windows environment the command interpreter (cmd.exe) can be used. In a Unix environment a suitable shell is recommended. The examples in the blue boxes below and in the rest of this document make use of the Microsoft Windows command interpreter.

For the use of the command line interface it is important:

- to choose an appropriate JRE environment (e.g. by setting the environment variable %JAVA_HOME% under Windows),
- to extend the class path with the jar files necessary to run the REPSI tool, and
- to align the proposed commands to your specific environment, i.e. to add or modify the options as described in detail on page 51 "Command Line Interface".

Setting up the Master Database

To set up the master database the following steps are necessary:

1. Create an appropriate user in the master database (see details on page 31 "Create an Appropriate Database User").
2. Adapt the properties file `config\configuration.properties` to your specific environment. This file contains the relevant information (driver, password, user name etc.) to connect to the master database (see also details on page 65 "Master Database Property File Entries").
3. Execute an aligned version of the following commands with the command line tool of your operating system. The content of file `in\std_master_db_schema_create.sql` enables the creation of the master database schema. An already existing schema may be deleted by using the file `in\std_master_db_schema_drop.xml`.

```
set CLASSPATH=...
%JAVA_HOME%\bin\java -jar lib\repsi-tool-02.00.jar
                     -mode master
                     -fn0 in\std_master_db_schema_drop.xml
                     -fn0xml
%JAVA_HOME%\bin\java -jar lib\repsi-tool-02.00.jar
                     -mode master
                     -fn0 in\std_master_db_schema_create.sql
```

4. Execute an aligned version of the following commands with the command line tool of your operating system. The file

in\std_master_db_instance_mand.xls contains the necessary SQL statements to load the mandatory instance data of the master database. This concerns the database tables SQL syntax (TMD_SQL_SYNTAX), test suite operation (TMD_TEST_SUITE_OPERATION), and trial run status (TMD_TRIAL_RUN_STATUS).

```
set CLASSPATH=...
%JAVA_HOME%\bin\java -jar lib\repsi-tool-02.00.jar
                     -mode master
                     -fn0 in\std_master_db_instance_mand.xls
                     -fn0xls
```

At this point the master database instance is ready for use. The next section describes how to prepare the test database.

Setting up the Test Database

The REPSI tool offers different possibilities to set up the test database. In every case the test database must be described in the master database. In the given example, the schema is generated from SQL statements contained in a flat file and the instance data are generated by running a test suite.

1. To describe the test database in the master database the following database tables of the master database have to be adapted: vendor (TMD_VENDOR), database (TMD_DATABASE), operating system (TMD_OPERATING_SYSTEM), processor (TMD_PROCESSOR), and database instance (TMD_DATABASE_INSTANCE) (see details on page 35 "Define a Test Database Instance"). The file in\std_master_db_instance_opt.xls is only an example and needs to be adapted to your environment.
2. After adapting the above files execute an aligned version of the following commands with the command line tool of your operating system:

```
set CLASSPATH=...
%JAVA_HOME%\bin\java -jar lib\repsi-tool-02.00.jar
                     -mode master
                     -fn0 in\std_master_db_instance_opt.xls
                     -fn0xls
```

3. Applying the file in\halevy_test_db_schema_create.sql creates the schema of the test database as shown in Appendix A. To apply this file execute an aligned version of the following commands with the command line tool of your operating system (example is based on test database instance (note: this example is based on test database instance 1):

```
set CLASSPATH=...
%JAVA_HOME%\bin\java -jar lib\repsi-tool-02.00.jar
                     -mode test
                     -di 1
                     -fn0 in\halevy_test_db_schema_create.sql
```

4. The file `in\halevy_master_db_instance.sql` contains the necessary instance data to generate the test database instance based on the master database tables test suite (`TMD_TEST_SUITE`), test table (`TMD_TEST_TABLE`), and test suite action (`TMD_TEST_SUITE_ACTION`). To apply this file execute an aligned version of the following commands with the command line tool of your operating system:

```
set CLASSPATH=...
%JAVA_HOME%\bin\java -jar lib\repsi-tool-02.00.jar
                     -mode master
                     -fn0 in\halevy_master_db_instance.xls
                     -fn0xls
```

5. Finally execute an aligned version of the following commands with the command line tool of your operating system to generate the instance data of the test database:

```
set CLASSPATH=...
%JAVA_HOME%\bin\java -jar lib\repsi-tool-02.00.jar
                     -mode trial
                     -di 1
                     -ts 901
```

Running a Calibration

To run and evaluate a calibration run the following steps are necessary:

1. A calibration run is based on a test query pair (`TMD_TEST_QUERY_PAIR`) which is related to a pattern or SQL idiom (`TMD_PATTERN_SQL_IDIOM`). The file `in\brass_master_db_instance.xls` contains appropriate example data. To apply this file execute an aligned version of the following commands with the command line tool of your operating system:

```
set CLASSPATH=...
%JAVA_HOME%\bin\java -jar lib\repsi-tool-02.00.jar
                     -mode master
                     -fn0 in\brass_master_db_instance.xls
                     -fn0xls
```

2. To run a calibration execute an aligned version of the following commands with the command line tool of your operating system (note: this example is based on test database instance 1 and test query pair 90201):

```
set CLASSPATH=...
%JAVA_HOME%\bin\java -jar lib\repsi-tool-02.00.jar
                     -mode calibration
                     -di 1
                     -exalt
                     -obj query
                     -tqp 90201
                     -verb
```

3. To load the data produced by the calibration run into an Excel file named `out\CalibrationRunData.xls` execute an aligned version of the following commands with the command line tool of your operating system (see also details on page 47 "Export Calibration results"):

```
set CLASSPATH=...
%JAVA_HOME%\bin\java -jar lib\repsi-tool-02.00.jar
                     -mode result_cn
                     -efn out\CalibrationRunData.xls
```

4 Installation – Development Version

The REPSI development (source) version can be downloaded from the same URL as the binary version: https://developer.berlios.de/project/showfiles.php?group_id=7172. The downloaded file named '`REPSI_Tool_vv.vv_Development.zip`' (where 'vv.vv' is the current version number, e.g. 02.00) is of type zip and has therefore to be unzipped with an appropriate tool into a new or existing directory. A detailed description of the structure of the resulting directory and file layout can be found on page 64 "File and Directory Layout – Development Version".

The development version is a complete export file of an Eclipse SDK. Provided that the same Eclipse version is available, only the directory of the unzipped file has to be targeted in the import functionality of Eclipse (File → Import → General → Existing Projects into Workspace → Select root directory …). (See also Figure 1: Eclipse - Import Projects)

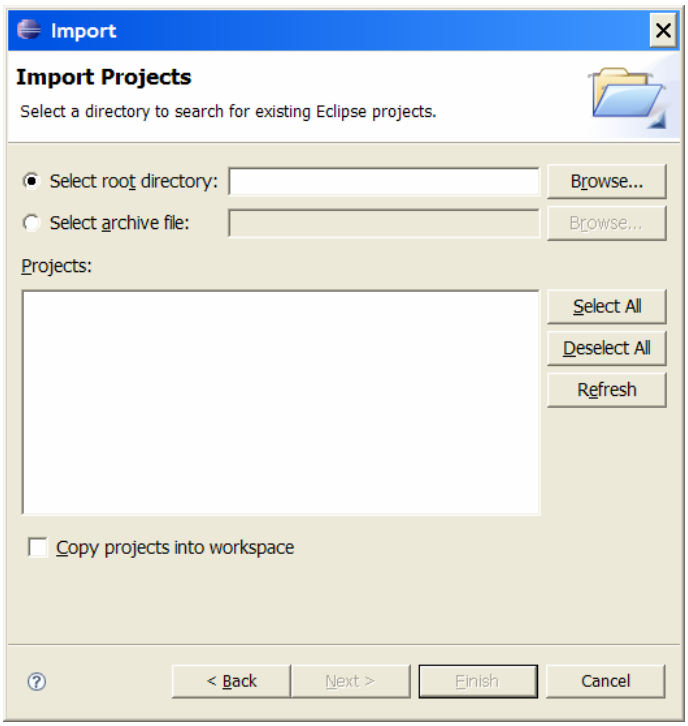

Figure 1: Eclipse - Import Projects

Additionally the original Eclipse preferences from the development may be imported; the appropriate file is in the `preferences` directory. For quality purposes the use of one or more static source code analysers is highly recommended. Checkstyle, FindBugs or PMD are among other suitable choices. For Checkstyle and PMD, configuration files are included in the development version.

5 Known Limitations

- **SQL rewriting**: The rewriting capabilities are currently very limited (see also details on page 16 "SQL Syntax Code and SQL Rewriting").
- **Supported database versions**: The tool currently supports only Oracle database systems out of the box. For database systems with a different SQL syntax the delivered files (see directory `in`) have to be adapted manually.
- **User interface**: The user interface is rather primitive and restricted purely to the command line. Input may be made easier by applying batch files (also called batch programs or scripts) in combination with Excel files or XML (Extensible Markup Language) documents. The output possibilities are limited to logging and Excel files.

Chapter 3 Concepts

1 SQL Syntax Code and SQL Rewriting

The SQL syntax code (see details Table 1: Valid SQL Syntax Codes) determines the outline of the appropriate rewriting functionality. An SQL syntax code has to be defined in:

- The configuration file (`configuration.properties`) to determine the SQL syntax of the master database,
- The following database tables to define the necessary SQL syntax:
  - `TMD_DATABASE_SYSTEM` (test database),
  - `TMD_TEST_QUERY_PAIR` (test SQL queries),
  - `TMD_TEST_TABLE` (test database tables, default '`SQL:1999`'), and
  - `TMD_TEST_TABLE_DDL` (SQL DDL (Data Definition Language) statements to create the test database tables).

SQL_SYNTAX_CODE	NAME	VERSION
ORACLE-10G	Oracle 10g	10
SQL:1999	Standard SQL:1999 (SQL3)	SQL:1999
SQL:2003	Standard SQL:2003	SQL:2003
SQL-86	Standard SQL-86 (SQL1)	SQL-86
SQL-89	Standard SQL-86 (SQL1) + Integrity Enhancement Feature	SQL-89
SQL-92	Standard SQL-92 (SQL2)	SQL-92

Table 1: Valid SQL Syntax Codes

Depending on the SQL syntax of the SQL statement to be executed (source) and the SQL syntax of the target database system, the following SQL rewriting mechanisms will be applied:

- In general, empty statements and simple comments (starting with '`--`') are ignored.
- If the SQL syntax codes of the SQL statement and of the target database system are identical, no further rewriting will be performed.

- From SQL syntax '`SQL:1999`' to '`ORACLE-10G`':
  - The leading and trailing whitespaces in the statement are removed.
  - Lower case characters are replaced by upper case characters (except in character string literals).
  - `CREATE DOMAIN` statements: Oracle doesn't support this feature of the SQL standard and therefore the rewriting mechanism stores the domain definitions and replaces them in subsequent `CREATE TABLE` statements.
  - `CREATE TABLE` statements: Domain references are replaced by the appropriate data types.
  - `DROP TABLE` and `DROP VIEW` statements: '`CASCADE`' is replaced by '`CASCADE CONSTRAINTS`'.
- Other combinations of SQL syntax codes are currently reflected in an appropriate error message, but the statements are not processed.

2 Master Database

The master database instance contains all the necessary information to perform calibration or trial runs and stores all subsequent results. A detailed data model can be found in on page 55 "Detailed Data Model". The most important tables of the master database are:

- `TMD_CALIBRATION` / `TMD_CALIBRATION_STATISTIC`: These tables record all information required to document the environmental conditions and the results of concrete calibration runs.
- `TMD_DATABASE_INSTANCE`: Describes the properties of a concrete test database instance to run a trial.
- `TMD_TEST_SUITE` / `TMD_TEST_SUITE_ACTION`: These tables define a set of actions (or operations) which build the elements of a test suite. A trial run executes these actions inside a specified test database environment.
- `TMD_TRIAL_RUN` / `TMD_TRIAL_RUN_ACTION` / `TMD_TRIAL_RUN_PROTOCOL`: These tables record all the information required to document the environmental conditions and the results of concrete trial runs.

Master database tables and test database tables may reside inside the same database instance but care must be taken to ensure that table names are not conflicting.

3 Test Database

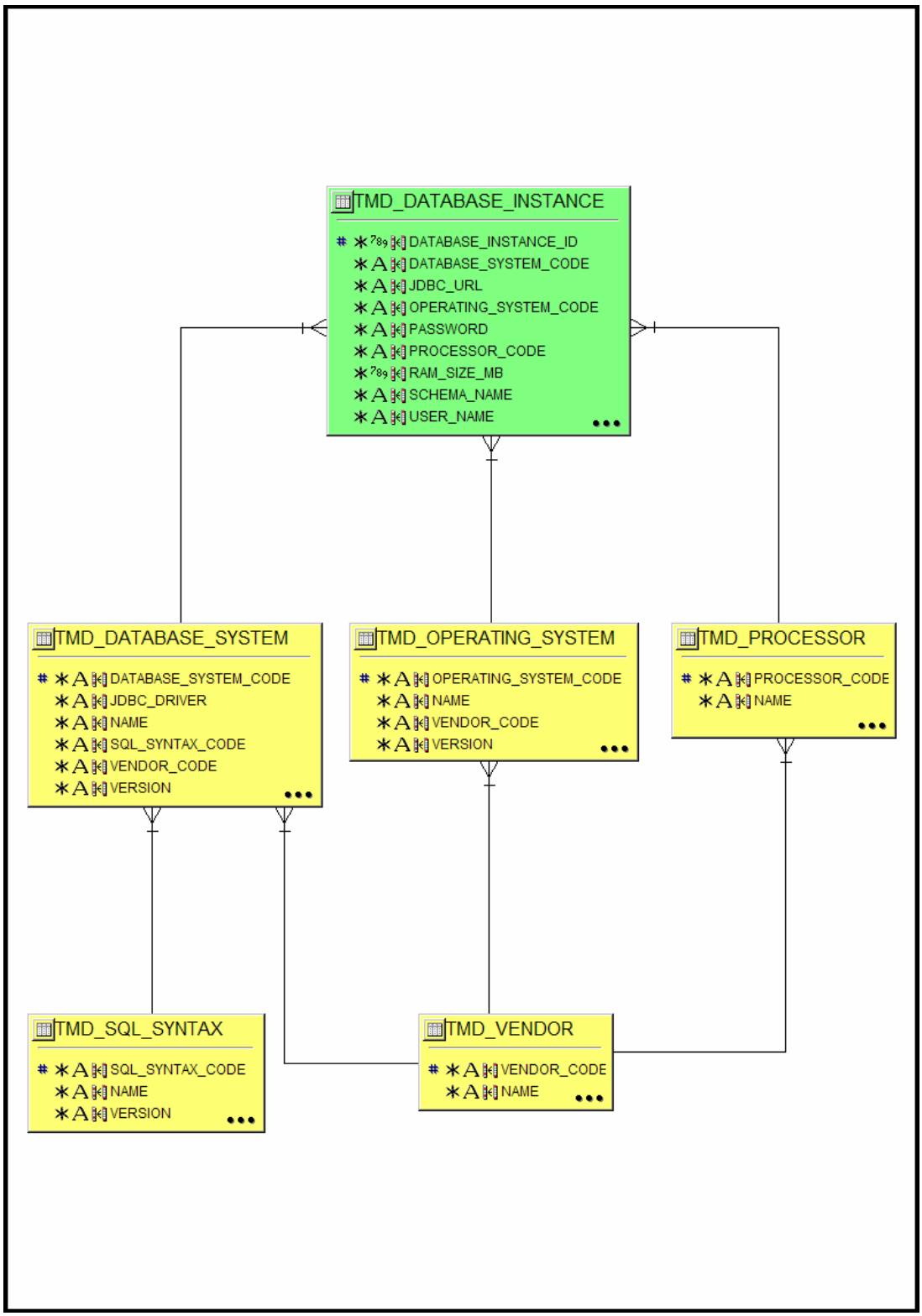

Figure 2: ERD Diagram Test Database Instance

The green database table in Figure 2 describes the properties of a concrete test database instance to run a trial. A database instance requires the prior definition of:

- A database system (requires SQL syntax and vendor),
- An operating system (requires vendor), and
- A processor (requires vendor).

The important information in the database instance table consists of the connection details like user name, password, database URL and driver (from table database system). The rest of the data has a purely documentary character.

If the database schema or database instance does not yet exist, the schema and / or the instance must be created:

- To create the test database schema the REPSI tool offers two possibilities (see details on page 38 "Create or Modify the Test Database Schema" and on page 41 "Define a Test Suite"):
  - Running a file containing appropriate SQL DDL statements, or
  - Modifying the tables `TMD_TEST_TABLE` and `TMD_TEST_TABLE_DDL` in the master database with the desired test tables, and creating and running a test suite containing appropriate schema operations.
- The instance of the test database can be created by defining and running a test suite containing appropriate instance operations to generate data load in the test database (see also details on page 41 "Define a Test Suite").

4 Generating Test Data

The data generator functionality generates an arbitrary number of rows for a table in the test database. The generation functionality considers already existing rows in the database table. The content of the columns are determined by applying randomised methods:

- **Columns of type `DATE`**: The actual value is drawn at random from an interval between 0 and 36501 and multiplied by the number of seconds per day (24*60*60 = 86400). Based on the Java class `Date` the resulting date value lies between 1.1.1970 and approximately (ignoring the leap year for simplification) 31.12.2069.
- **Columns of type `NUMERIC`**: The actual value is drawn at random from a double value and rounded to the given precision.
- **Columns of type `CHAR` and `VARCHAR`**: If the type is `VARCHAR` then in a first step the actual value for the length is drawn at random from an interval between 0 and the maximum length. Then each character is drawn in at random from a vector of characters consisting of the lower case letters 'a' to 'z', the capital letters 'A' to 'Z', and the digits '0' to '9'.

Columns which are part of a primary key or of the foreign key are subject to special treatment:

- **Primary key columns**: The values of the independent primary key columns are determined depending on the data type:
  - **Columns of type `DATE`**: Consecutive values starting with date 1.1.1970 and an interval of 1 day.
  - **Columns of type `NUMERIC`**: Consecutive values starting with 1 and interval of 1.
  - **Columns of type `CHAR` and `VARCHAR`**: Consecutive values starting with string like 'a…a' and varying
    - firstly the last position of the string from 'b' to 'z', from 'A' to 'Z', and from '0' to '9',
    - then changing the second last character to 'b' and again varying the last character from 'a' to 'z', from 'A' to 'Z', and from '0' to '9',
    - and so on.

  Note: Primary key columns which are also contained in a foreign key are treated as foreign key columns (this may lead in rare cases to duplicate primary keys, which are then logged and ignored).
- **Foreign key columns**: The actual value is drawn at random from a vector containing the current values of the referenced table.

5 Test Suite

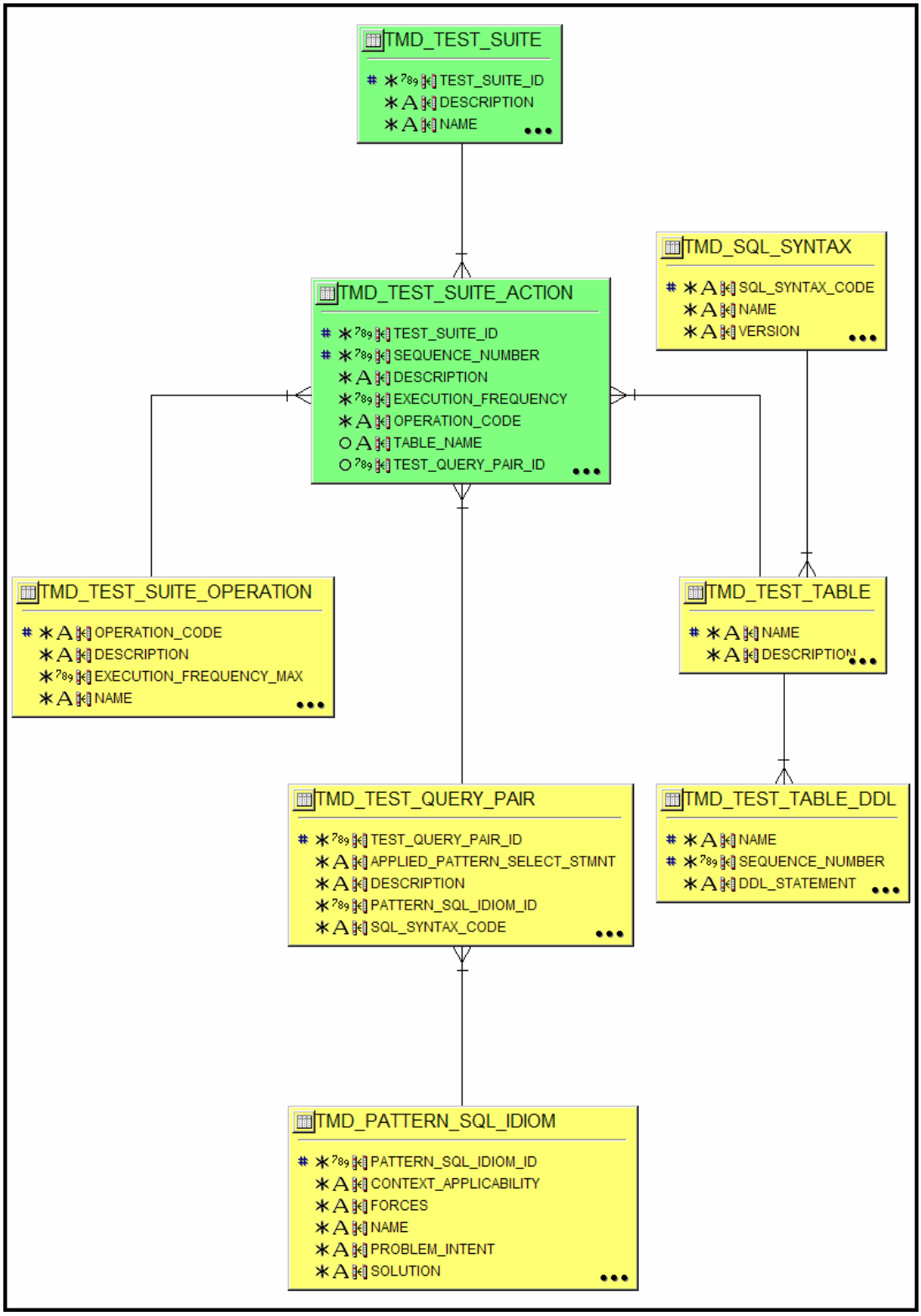

Figure 3: ERD Diagram Test Suite

The test suite (the green database tables in Figure 3) defines a set of operations (also called actions) which have to be executed in a given sequence inside a test database environment during a trial run (see details in Table 2: Valid Test Suite Operations). REPSI supports currently the following operation types and operations:

- **Schema**: Operations related to the test database schema:
  - **CT - Create database table**: This operation requires a reference to a test table (`TMD_TEST_TABLE`) and a set of SQL DDL statements (`TMD_TEST_TABLE_DDL`). REPSI executes these SQL DDL statements in the test database in the specified sequence.
  - **DT - Drop database table**: This operation requires a reference to a test table. REPSI generates and executes internally an appropriate '`DROP TABLE xxx CASCADE`' statement in the test database.
  - **DC - Drop and create database table**: This operation is the combination of the previous two operations and allows the re-creation of a database table in the test database during a trial run.
- **Instance**: The operation related to the test database instance:
  - **IR - Insert row into database table**: This operation inserts a specified number of rows into a given database table of the test database.
- **Query**: Operations related to the execution of a database query:
  - **EQ - Execute trial query pair**: This operation executes a query with and a query without the application of a pattern or SQL statement.
  - **EA - Execute trial query pair – applied version**: This operation executes only the query with the application of a pattern or SQL statement.
  - **EU - Execute trial query pair – unapplied version**: This operation executes only the query without the application of a pattern or SQL statement.

D..	DESCRIPTION	E.	NAME	OPERA...
CT	Create database table	1	Create table	Schema
DC	Drop and create database table	1	Drop and create table	Schema
DT	Drop database table	1	Drop table	Schema
EA	Execute trial query pair - applied version	0	Execute query applied	Query
EQ	Execute trial query pair	0	Execute query	Query
EU	Execute trial query pair - unapplied version	0	Execute query unapplied	Query
IR	Insert row into database table	0	Insert row	Instance

Table 2: Valid Test Suite Operations

6 Trial Run and Results

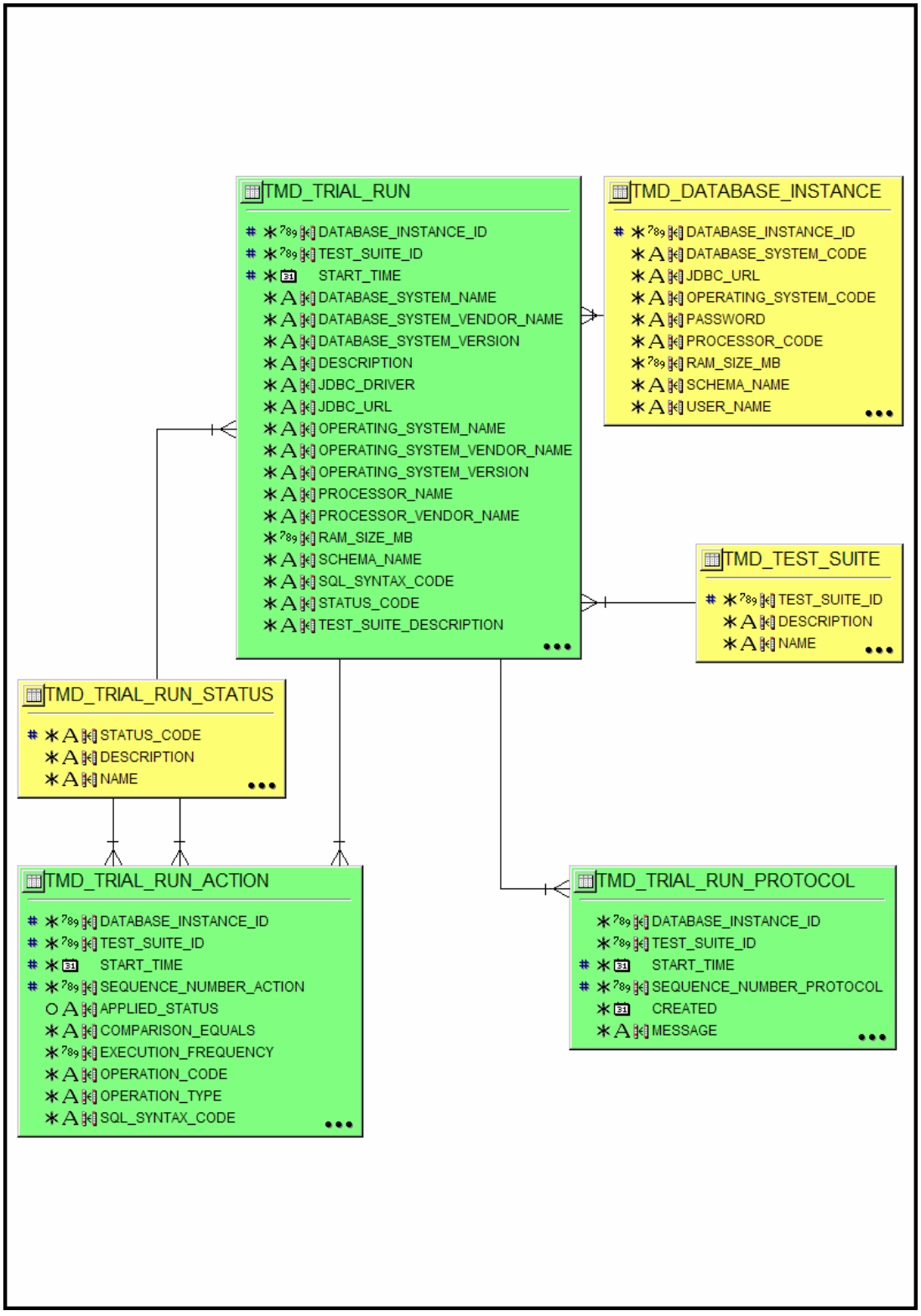

Figure 4: ERD Diagram Trial Run

The green database tables in Figure 4 record all the information relevant to document the environmental conditions and the results of a concrete trial run. These database tables are designed in a deliberately redundant way because the data should give a genuine historic view which future changes in the database should not be able to corrupt.

A trial run processes the following options (see also details on page 51 "Command Line Interface"):

- Mandatory:
  - Option **`di`** (identification of the test database instance): the database instance describes the properties of a concrete test database instance to run a trial (see also details on page 35 "Define a Test database Instance").
  - Option **`ts`** (identification of the test suite): the test suite comprises a set of actions or operations to be executed in a predefined sequence (see also details on page 41 "Define a Test Suite").
- Optionally:
  - Option **`cyc`**: determines how often the trial run should be executed (default value is 1).
  - Option **`des`**: provides an explanatory description characterising the nature of the trial run (default value is 'n/a').
  - Option **`fs`**: determines the number of rows physically retrieved from the database at one time by the JDBC driver when scrolling through a query `ResultSet` with `next()` (default value is 10).
  - Option **`prec`**: determines the exponent (base 10) of the response time precision (default value is 0 – results in nanoseconds).

The results of a trial run are stored in the following three tables:

1. `TMD_TRIAL_RUN`: consists of one row per trial run containing the following data:
   - Timestamp of the start and end time of the run, and the final status (see details Table 3: Valid Trial Run Status) of the total run,
   - Detailed information about the test database instance, including the characteristics of the used processor, operating system, and database system, as well as database driver, URL, and user,
   - Detailed information about the test suite.

STATUS_CODE	DESCRIPTION	NAME
EA	n/a	End Action
EF	n/a	End Finalisation
EI	n/a	End Initialisation
EP	n/a	End Program
SA	n/a	Start Action
SF	n/a	Start Finalisation
SI	n/a	Start Initialisation
SP	n/a	Start Program

Table 3: Valid Trial Run Status

2. `TMD_TRIAL_RUN_ACTION`: consists of one row per action defined in the underlying test suite , each containing the following data:
   - Operation type, operation code, and SQL syntax code,
   - In the case of a schema operation
     - Table name
   - In the case of an instance operation
     - Table name and frequency,
   - In the case of a query operation
     - The Pattern or SQL idiom,
     - For both applied and unapplied versions of the query
       - the SQL query itself,
       - the timestamp of the start and end time of the operation,
       - the duration in nanoseconds,
       - a potential error message;
       - and the status,
     - A flag showing the equality of the result sets and a reason in case of inequality.

3. `TMD_TRIAL_RUN_PROTOCOL`: contains a detailed error and information protocol showing specific events of the trial run.

7 Calibration Run and Results

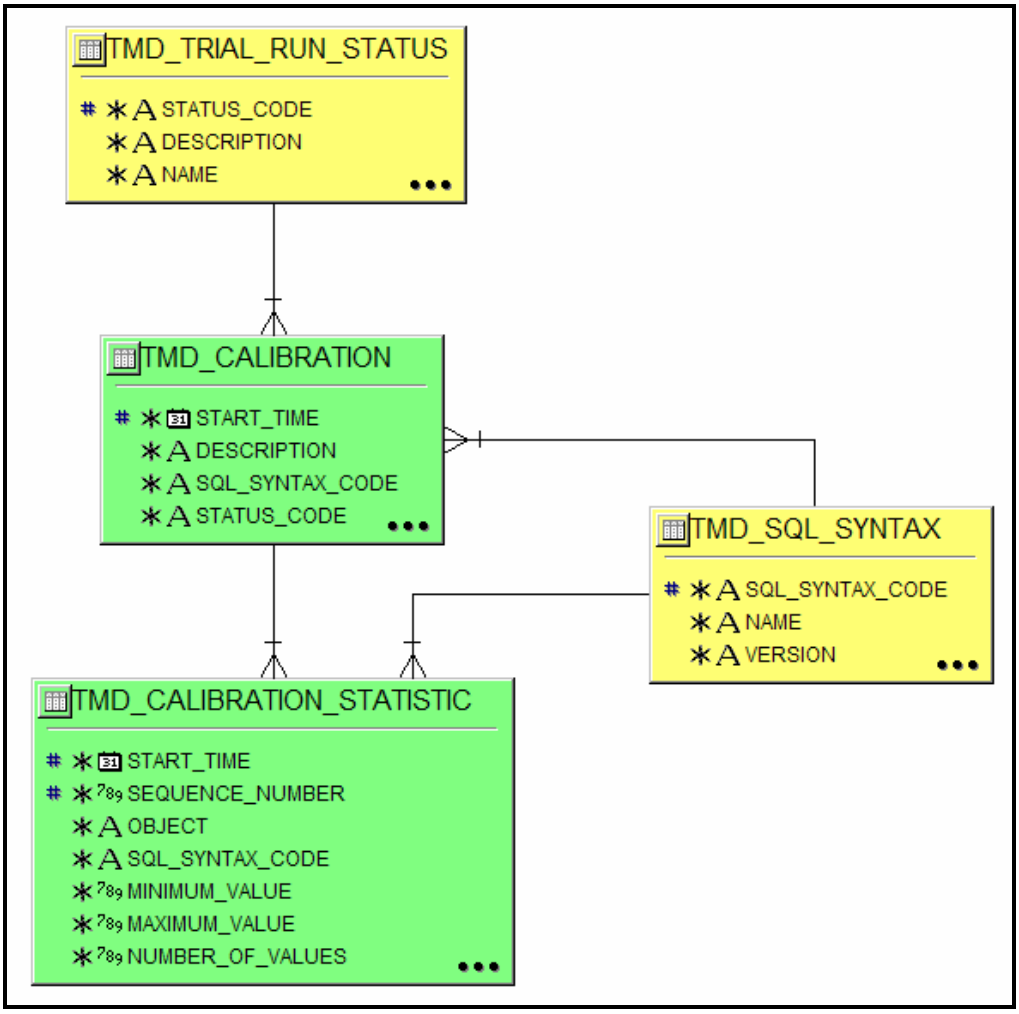

Figure 5: ERD Diagram Calibration Run

The green database tables in Figure 5 record all the information relevant to document the environmental conditions and the results of a concrete calibration run. These database tables are designed in a deliberately redundant way because the data should give a genuine historic view which future changes in the database should not be able to corrupt.

A calibration run processes the following options (see also details on page 51 "Command Line Interface"):

- Mandatory:
  - Option `di` (identification of the test database instance): the database instance describes the properties of a concrete test database instance to run a trial (see also details on page 35 "Define a Test Database Instance").

- Option **opt**: defines the calibration object:
  - **nanotime**: calibrates the Java method `System.nanoTime()`,
  - **query**: calibrates a test query pair.

- Optionally:
  - Option **cyc**: determines how often the trial run should be executed (default value is 50).
  - Option **des**: provides an explanatory description characterising the nature of the trial run (default value is 'n/a').
  - Option **exalt**: executes the queries alternating the one without application of the pattern and the one with application of the pattern.
  - Option **excon**: executes first the query without application of the pattern and then the query with application of the pattern.
  - Option **fs**: determines the number of rows physically retrieved from the database at one time by the JDBC driver when scrolling through a query `ResultSet` with `next()` (default value is 10).
  - Option **ign1**: ignores the first reading.
  - Option **prec**: determines the exponent (base 10) of the response time precision (default value is 0 – results in nanoseconds).
  - Option **tqp** determines the identification of the test query pair, mandatory if the calibration object is of type query.

The results of a calibration run are stored in the following two tables:

1. `TMD_CALIBRATION`: consists of one row per trial run containing the following data:
   - Timestamp of the start and end time of the run, and the final status (Table 3: Valid Trial Run Status) of the total run,
   - Detailed information about the test database instance, including the characteristics of the used processor, operating system, and database system, as well as database driver, URL, and user,

2. `TMD_CALIBRATION_STATISTIC`: contains in the case of the calibration object '`nanotime`' one row. In the case of the calibration object '`query`' one row per query (with and without application of the pattern) and execution type (alternating / consecutive). Each of the rows contains the following data:
   - Calibration object,
   - In the case of the calibration object '`query`'
     - Name of the underlying pattern or SQL idiom,
     - SQL query statement, and
     - SQL syntax code,

- The measured response time per execution of the calibration object (column READINGS),
- The number of executions (column NUMBER_OF_VALUES),
- The extreme values minimum and maximum response time,
- A set of statistics calculated based on the readings:
  - Kurtosis,
  - Median (column PERCENTILE_50),
  - 1st and 3rd quartile (columns PERCENTILE_25 and PECENTILE_75),
  - Skewness,
  - Standard deviation, and
  - Variance.

8 Measuring the Response Time

The response time is measured using Java's `nanoTime` method from the `System` class (Sun Microsystems Inc., 2004). This method delivers the current value of the system timer in nanosecond precision, but, depending on the hardware platform not necessarily in nanosecond accuracy. This time value is not in any relationship to the UTC (Coordinated Universal Time), but is the representation of the number of processor clock cycles converted into nanoseconds (e.g. on a 2 GHz processor: 1 nanosecond ~ 2 clock cycles) (Green, 2006). Under Windows, the JVM (Java Virtual Machine) uses the Win32 methods `QueryPerformanceCounter` and `QueryPerformanceFrequency` to determine and calculate the nanoseconds (Green, 2006, Microsoft Corporation, 2005). This is, in the given environment, the best solution because the time measurement is not corrupted by external interferences, e.g. triggered by the operating system's execution of an automated time correction. The precision of the response time readings can be determined by the option '**`prec`**' which acts as an exponent to base 10. If the response time is required in microseconds the value to be chosen is 3.

Chapter 4 Tasks

In general, running the REPSI tool requires the adherence to the rules described in detail on page 51 "Command Line Interface".

1 Create an Appropriate Database User

The creation of the database users is dependent on the chosen DBMS (Database Management System).

RDBMS Oracle® Database 10g

In an Oracle RDBMS the terms user and schema are interchangeable, because there is a 1 to 1 relationship between a user (who can connect to a database) and a schema (which comprises all objects which a user owns). The roles and privileges assigned to a user restrict the functionality of a user in relation to database objects and operations. Quotas are a means to limit the space a user can use on a tablespace. For details please refer to the Oracle Database Administrator's Guide (Fogel and Lane, 2006).

For the REPSI tool the necessary minimum rights for the user of the **master database** are as follows:

Roles:

- `CONNECT`
- `RESOURCE`

System Privileges:

- `CREATE ANY SYNONYM`
- `CREATE ANY VIEW`
- `UNLIMITED TABLESPACE` (alternatively grant Quota)

Quota:

- Choice of an appropriate value (alternatively system privilege)

For the REPSI tool the necessary minimum rights for the user of the **test database** are as follows:

Roles:

- `CONNECT`
- `RESOURCE`

System Privileges:

- `ALTER SESSSION`
- `CREATE ANY VIEW`
- `UNLIMITED TABLESPACE` (alternatively grant Quota)

Quota:

- Choice of an appropriate value (alternatively system privilege)

Note: These are the minimum rights necessary to work with the supplied scripts. Additional roles, object and system privileges may be necessary depending on the database features employed.

2 Set up the Master Database Schema and Default Instance

Prerequisites:

- The binary version of the REPSI tool is downloaded and installed.
- An appropriate user for the master database has been created in the Oracle® Database (see also details on page 31 "Create an Appropriate Database User").
- The property file for the master database has been adapted (see also details on page 65 "Master Database Property File Entries").

Execution:

- To delete existing elements of a master database schema apply the following command:

```
java -jar repsi-tool-vv.vv.jar
     -mode master
     -fn0 in\std_master_db_schema_drop.xml -fn0xml
    [-pf x1 [-pfxml]]
```

- To create a new master database schema apply the following command:

```
java -jar repsi-tool-vv.vv.jar
     -mode master
     -fn0 in\std_master_db_schema_create.sql
    [-pf x1 [-pfxml]]
```

- To create the mandatory elements of a master database instance apply the following commands:

```
java -jar repsi-tool-vv.vv.jar
     -mode master
     -fn0 in\std_master_db_instance_mand.xls -fn0xls
    [-pf x1 [-pfxml]]
```

```
java -jar repsi-tool-vv.vv.jar
     -mode master
     -fn0 in\std_master_db_instance_opt.xls -fn0xls
    [-pf x1 [-pfxml]]
```

whereby: x1 – name of the master database property file
vv.vv – version of the REPSI tool

Result:

Every run of the REPSI tool should finish without any error message. The last message of a successful run reads as follows:

```
INFO: Task completed successfully
```

3 Define a Test Database Instance

Prerequisites:

- The schema and default instance of the master database have been set up (see also details on page 33 "Set up the Master Database Schema and Default Instance").

Execution:

- Create a file containing SQL DML (Data Manipulation Language) statements for the following database tables (the sequence is determined by the referential integrity constraints):
  - Vendor table (`TMD_VENDOR`): to create a vendor (for processor, operating system, or database system) entry in the master database apply an aligned version of the following SQL DML statement:

```
INSERT INTO TMD_VENDOR
            (vendor_code,
             created_by,
             date_created,
             NAME
            )
     VALUES ('MICROSOFT',
             'REPSI Tool',
             SYSDATE,
             'Microsoft Corporation'
            );
```

  - Processor table (`TMD_PROCESSOR`): to create the description of a processor entry in the master database apply an aligned version of the following SQL DML statement (note: for example Intel offers a processor identification utility to determine the exact type of an Intel processor):

```
INSERT INTO TMD_PROCESSOR
            (processor_code,
             created_by,
             date_created,
             NAME,
             vendor_code
            )
     VALUES ('PENTIUM-M-1700',
             'REPSI Tool',
             SYSDATE,
             'Intel(R) Pentium(R) M processor 1700MHz',
             'INTEL'
            );
```

- Operating system table (TMD_OPERATING_SYSTEM): to create the description of an operating system entry in the master database apply an aligned version of the following SQL DML statement:

```
INSERT INTO TMD_OPERATING_SYSTEM
            (operating_system_code,
             created_by,
             date_created,
             NAME,
             vendor_code,
             VERSION
            )
     VALUES ('MS-WINDOWS-XP-PROF',
             'REPSI Tool',
             SYSDATE,
             'Microsoft Windows XP Professional',
             'MICROSOFT',
             '2002 Servicepack 2'
            );
```

- Database system table (TMD_DATABASE_SYSTEM): to create the description of a database system entry in the master database apply an aligned version of the following SQL DML statement:

```
INSERT INTO TMD_DATABASE_SYSTEM
            (database_system_code,
             created_by,
             date_created,
             jdbc_driver,
             NAME,
             sql_syntax_code,
             vendor_code,
             VERSION
            )
     VALUES ('ORACLE-10G-R2',
             'REPSI Tool',
             SYSDATE,
             'oracle.jdbc.driver.OracleDriver',
             'Oracle 10g Release 2',
             'ORACLE-10G',
             'ORACLE',
             '10.2.0.1.0'
            );
```

- Database instance table (TMD_DATABASE_INSTANCE): to create a database instance entry in the master database apply an aligned version of the following SQL DML statement:

```
INSERT INTO TMD_DATABASE_INSTANCE
            (database_instance_id,
             created_by,
```

```
            date_created,
            database_system_code,
            jdbc_url,
            operating_system_code,
            PASSWORD,
            processor_code,
            ram_size_mb,
            schema_name,
            user_name
          )
    VALUES (1, '
            REPSI Tool',
                 SYSDATE,
                'ORACLE-10G-R2',
                'jdbc:oracle:thin:@weinmann:1521:bcag10g2',
                'MS-WINDOWS-XP-PROF',
                'repsi_test',
                'PENTIUM-M-1700',
                2048,
                'REPSI_TEST',
                'REPSI_TEST'
          );
```

- To execute the SQL DML statements described above apply an aligned version of the following command:

```
java -jar repsi-tool-vv.vv.jar
              -mode master
              -fn0 x1 [-fn0xml]
             [-pf x2 [-pfxml]]
```

whereby:
- x1 – name of the data file containing the SQL DML statements
- x2 – name of the master database property file
- vv.vv – version of the REPSI tool

Results:

The REPSI tool should finish without any error message. The last message of a successful run reads as follows:

```
INFO: Task completed successfully
```

4 Create or Modify the Test Database Schema

The REPSI tool offers two options to create or modify the schema of the test database:

- **Option 1**: Requires storing the SQL DML statements in the master database related to the test tables, then defining a test suite with appropriate schema actions and finally running the REPSI tool in the trial mode.
- **Option 2**: Similar to the task on page 33 "Set up the Master Database Schema and Default Instance" option 2 requires creating a file containing the SQL DDL statements and then running the REPSI tool in the test mode.

Prerequisites:

- **Option 1**: The schema and default instance of the master database have been set up (see also details on page 33 "Set up the Master Database Schema and Default Instance").

Option 1 - Execution:

- Create a file containing SQL DML statements for the following database tables (the sequence is determined by the referential integrity constraints):
  - Test suite table (`TMD_TEST_SUITE`): to create a test suite master entry in the master database apply an aligned version of the following SQL DML statement:

```
INSERT INTO TMD_TEST_SUITE
            (test_suite_id,
             created_by,
             date_created,
             NAME
            )
     VALUES (1,
             'REPSI Tool',
             SYSDATE,
             'Example test suite'
            );
```

  - Test table (`TMD_TEST_TABLE`): to create a test table entry in the master database apply an aligned version of the following SQL DML statement:

```
INSERT INTO TMD_TEST_TABLE
            (NAME,
             created_by,
             date_created
            )
     VALUES ('TTD_TABLE_01',
             'REPSI Tool',
              SYSDATE
            );
```

- Test table DDL Statement table (`TMD_TEST_TABLE_DDL`): to create a test table DDL entry in the master database apply an aligned version of the following SQL DML statement:

```
INSERT INTO TMD_TEST_TABLE_DDL
            (NAME,
             sequence_number,
             ddl_statement,
             sql_syntax_code
            )
     VALUES ('TTD_TABLE_01',
             1,
             'CREATE TABLE TTD_TABLE_01 (COL_01_VARCHAR
VARCHAR(10) NOT NULL,COL_02_DATE DATE,COL_03_VARCHAR
VARCHAR(30),COL_04_DATE DATE,COL_05_VARCHAR
VARCHAR(30),COL_06_VARCHAR VARCHAR(50) NOT NULL);',
        'SQL:1999'
        );
```

- Test suite action table (`TMD_TEST_SUITE_ACTION`): to create a test suite action (test suite detail) entry in the master database apply an aligned version of the following SQL DML statement:

```
INSERT INTO TMD_TEST_SUITE_ACTION
            (test_suite_id,
             sequence_number,
             operation_code,
             table_name
            )
     VALUES (1,
             1,
            'DC',
            'TTD_TABLE_01'
            );
```

- To execute the SQL DML statements described above apply an aligned version of the following command:

```
java -jar repsi-tool-vv.vv.jar
               -mode trial
               -di x1
              [-pf x2 [-pfxml]]
               -ts x3
```

whereby:
- x1 – the identification of the test database instance
- x2 – name of the master database property file
- x3 – the identification of the test suite
- vv.vv – version of the REPSI tool

Option 2 - Execution:

Alternatively the DDL statements to change the instance of the test database may be stored in flat file. This file can then be executed by running the REPSI tool in the test mode.

- Create a file containing the necessary SQL DDL statements for the required test database tables, e.g.:

```
CREATE TABLE TTD_TABLE_01 (
          COL_01_VARCHAR VARCHAR(10) NOT NULL,
          COL_02_DATE    DATE,
          COL_03_VARCHAR VARCHAR(30),
          COL_04_DATE    DATE,
          COL_05_VARCHAR VARCHAR(30),
          COL_06_VARCHAR VARCHAR(50) NOT NULL
                          );
```

- To execute the SQL DDL statements described above apply an aligned version of the following command:

```
java –jar repsi–tool–vv.vv.jar
               –mode test
               –di x1
               –fn0 x2 [–fn0xml]
              [–pf x3 [–pfxml]]
```

whereby:
- x1 – the identification of the test database instance
- x2 – name of the file containing the SQL DDL statements
- x3 – name of the master database property file
- vv.vv – version of the REPSI tool

Results:

The REPSI tool should finish without any error message. The last message of a successful run reads as follows:

```
INFO: Task completed successfully
```

5 Define a Test Suite

Prerequisites:

- The schema and default instance of the master database have been set up (see also details on page 33 "Set up the Master Database Schema and Default Instance").

Execution:

- Create a file containing SQL DML statements for the following database tables (the sequence is determined by the referential integrity constraints):
  - Test suite table (`TMD_TEST_SUITE`): to create a test suite master entry in the master database apply an aligned version of the following SQL DML statement:

```
INSERT INTO TMD_TEST_SUITE
            (test_suite_id,
             created_by,
             date_created,
             NAME
            )
     VALUES (1,
             'REPSI Tool',
             SYSDATE,
             'Example test suite'
            );
```

- To define a test table to modify the schema of the test database please follow the instructions on page 38 "Create or Modify the Test Database Schema" – option 1.
- To generate data load in the instance of the test database the following database entries are necessary:
  - Test suite action table (`TMD_TEST_SUITE_ACTION`): to create a test suite action (test suite detail) entry in the master database apply an aligned version of the following SQL DML statement:

```
INSERT INTO TMD_TEST_SUITE_ACTION
            (test_suite_id,
             sequence_number,
             execution_frequency,
             operation_code,
             table_name
            )
     VALUES (1,
             2,
             1000,
            'IR',
            'TTD_TABLE_01'
            );
```

- To test the run time behaviour of a pair of queries (one with application of a pattern or SQL idiom and one without) running the following database entries in the test database is necessary:
  - Pattern or SQL idiom table (`TMD_PATTERN_SQL_IDIOM`): to create a pattern or SQL idiom entry in the master database apply an aligned version of the following SQL DML statement:

```
INSERT INTO TMD_PATTERN_SQL_IDIOM
            (pattern_sql_idiom_id,
             created_by,
             date_created,
             NAME
            )
     VALUES (01,
             'REPSI Tool',
             SYSDATE,
             'Flatten Views'
            );
```

  - Test query pair table (`TMD_TEST_QUERY_PAIR`) : to create a test query pair entry in the master database apply an aligned version of the following SQL DML statement::

```
INSERT INTO TMD_TEST_QUERY_PAIR
            (test_query_pair_id,
             created_by,
             date_created,
             applied_pattern_select_stmnt,
             description,
             pattern_sql_idiom_id,
             sql_syntax_code,
             unapplied_pattern_select_stmnt
            )
     VALUES (101,
             'REPSI Tool',
             SYSDATE,
             'SELECT rd.student, ce.title FROM thy_teaches ts,
thy_prof pf, thy_registered rd, thy_course ce WHERE pf.NAME =
ts.prof AND ts.c_number = rd.c_number AND ts.quarter =
rd.quarter AND rd.c_number = ce.c_number AND ce.c_number >=
500 AND pf.area = 10 ORDER BY 1, 2',
             'n/a',
             01,
             'SQL:1999',
             'SELECT ge.student, ge.title FROM thy_teaches ts,
thy_prof pf, vhy_graduate ge WHERE pf.NAME = ts.prof AND
ts.c_number = ge.c_number AND ts.quarter = ge.quarter AND
ge.c_number >= 500 AND pf.area = 10 ORDER BY 1, 2'
            );;
```

- Test suite action table (`TMD_TEST_SUITE_ACTION`): to create a test suite action (test suite detail) entry in the master database apply an aligned version of the following SQL DML statement:

```
INSERT INTO TMD_TEST_SUITE_ACTION
            (test_suite_id,
             sequence_number,
             created_by,
             date_created,
             operation_code,
             test_query_pair_id
            )
     VALUES (1,
             3,
             'REPSI Tool',
             SYSDATE,
             'EQ',
             101
            );
```

- To execute the SQL DML statements described above apply an aligned version of the following command:

```
java -jar repsi-tool-vv.vv.jar
               -mode master
               -fn0 x1 [-fn0xml]
              [-pf x2 [-pfxml]]
```

whereby: x1 – name of the file containing the SQL DML statements
x2 – name of the master database property file
vv.vv – version of the REPSI tool

Results:

The REPSI tool should finish without any error message. The last message of a successful run reads as follows:

```
INFO: Task completed successfully
```

6 Perform a Trial Run

Prerequisites:

- A test database instance is defined in the master database (see also details on page 35 "Define a Test Database Instance").
- A test suite is defined in the master database (see also details on page 41 "Define a Test Suite").
- The schema of the test database has been set up (see also details on page 38 "Create or Modify the Test Database Schema").

Execution:

To perform a trial run apply an aligned version of the following command:

```
java –jar repsi–tool–vv.vv.jar
               –mode trial
              [–cyc x1]
              [–des x2]
               –di x3
              [–fs x4]
              [–pf x5 [–pfxml]]
              [–prec x6]
               –ts x7
```

whereby:

x1	–	number of cycles to run the test suite
x2	–	a descriptive text to characterise the trial run
x3	–	identification of the test database instance
x4	–	fetch size
x5	–	name of the master database property file
x6	–	the exponent (base 10) of the time precision
x7	–	identification of the test suite
vv.vv	–	version of the REPSI tool

Result:

The REPSI tool should finish without any error message. The last message of a successful run reads as follows:

```
INFO: Task completed successfully
```

7 Export Trial Results into Excel Compatible Files

Prerequisites:

At least one trial run has been executed (see also details on page 44 "Perform a Trial Run").

Execution:

To export trial results into an Excel file apply an aligned version of the following command:

```
java -jar repsi-tool-vv.vv.jar
              -mode result_tl
             [-efn x1]
             [-efnall]
             [-pf x2 [-pfxml]]
```

whereby: x1 – name of the output Excel file
x2 – name of the master database property file
vv.vv – version of the REPSI tool

Result:

The REPSI tool should finish without any error message. The last message of a successful run reads as follows:

```
INFO: Task completed successfully
```

Note: The Excel file specified with the '`-efn`' option contains the exported data from database table `TMD_TRIAL_RUN` in the first worksheet, from `TMD_TRIAL_RUN_ACTION` in the second, and from `TMD_TRIAL_RUN_PROTOCOL` In the third (see details on page 24 "Trial Run and Results").

8 Perform a Calibration Run

Prerequisites:

- A test database instance is defined in the master database (see also details on page 35 "Define a Test Database Instance").
- The schema of the test database has been set up (see also details on page 38 "Create or Modify the Test Database Schema").

Execution:

To perform a calibration run apply an aligned version of the following command:

```
java -jar repsi-tool-vv.vv.jar
               -mode calibration
               -pf x1 [-pfxml]
               -di x2
               -obj {nanotime|query}
              [-cyc x3]
              [-des x4]
              [-exalt]
              [-excon]
              [-fs x5]
              [-ign1]
              [-prec x6]
              [-tqp x7]
```

whereby:
- x1 – name of the master database property file
- x2 – identification of the test database instance
- x3 – number of cycles to run the test query pair
- x4 – a descriptive text to characterise the calibration run
- x5 – fetch size
- x6 – the exponent (base 10) of the time precision
- x7 – identification of the test query pair
- vv.vv – version of the REPSI tool

Result:

The REPSI tool should finish without any error message. The last message of a successful run reads as follows:

```
INFO: Task completed successfully
```

9 Export Calibration Results into Excel Compatible Files

Prerequisites:

At least one calibration run has been executed (see also details on page 46 "Perform a Calibration Run").

Execution:

To export calibration results into an Excel file apply an aligned version of the following command:

```
java –jar repsi–tool–vv.vv.jar
               –mode result_cn
              [–efn x1}
              [–pf x2 [–pfxml]]
```

whereby: x1 – name of the output Excel file
x2 – name of the master database property file
vv.vv – version of the REPSI tool

Result:

The REPSI tool should finish without any error message. The last message of a successful run reads as follows:

```
INFO: Task completed successfully
```

Note: The Excel file specified with the '`–efn`' option contains the exported data from database table `TMD_CALIBRATION` in the first worksheet and from `TMD_CALIBRATION_STATISTIC` in the second (see details on page 27 "Calibration Run and Results").

10 Export Calibration Data into R Code Files

Prerequisites:

At least one calibration run has been executed (see also details on page 46 "Perform a Calibration Run").

Execution:

To create files containing R code based on calibration results apply an aligned version of the following command:

```
java -jar repsi-tool-vv.vv.jar
               -mode result_r
              [-idn x1]
              [-pf x2 [-pfxml]]
```

whereby: x1 – name of the directory to place the R compatible files

x2 – name of the master database property file

vv.vv – version of the REPSI tool

Result:

The REPSI tool should finish without any error message. The last message of a successful run reads as follows:

```
INFO: Task completed successfully
```

Note: The content of the files containing R code depends on the calibrated object and, with calibrated queries, on the execution type of the queries. Usually one to four vectors containing the measured response times are created first. Then a summary function per vector follows. Finally box plots presenting one vector (obj = nanotime) or contrasting two vectors (obj = query) are created. A typical example for a query object:

```
if (exists("A_A")) remove("A_A")
if (exists("A_U")) remove("A_U")
if (exists("C_A")) remove("C_A")
if (exists("C_U")) remove("C_U")
A_U<-c(6656,1456,1601,1333,2087,1819,1326,1822,2281,1311,1394,
1604,1299,1300,1605,1359,1339,1293,1270,1269,3571,1309,2558,21
38,1230,2125,1335,1196,1671,1291)
A_A<-c(23163,1431,1695,1203,1210,1424,1245,1983,1593,1335,
1440,1185,1352,1308,1979,1604,1187,1162,1144,1138,1666,1590,11
67,6568,1171,1109,1628,1102,1104,1395)
if (exists("A_U")) summary(A_U)
if (exists("A_A")) summary(A_A)
if (exists("C_U")) summary(C_U)
if (exists("C_A")) summary(C_A)
```

```
if (exists("A_U"))
boxplot(A_U,A_A,col="lightblue",horizontal=TRUE,log="x",match=
TRUE,names=c("Not applied","Applied"),notch=TRUE)
if (exists("C_U"))
boxplot(C_U,C_A,col="lightblue",horizontal=TRUE,log="x",match=
TRUE,names=c("Not applied","Applied"),notch=TRUE)
```

Chapter 5 Catalogue of Patterns and SQL Idioms

A set of patterns will be published after the final submission of my dissertation.

Chapter 6 Reference

1 Command Line Interface

The command line interface facilitates the completion of the following tasks:

- Execution of a calibration run, i.e. running a selected method or an SQL query pair multiple times (see option 'cyc') in a specific database environment,
- Execution of a trial run, i.e. running a test suite in a specific database environment,
- Extracting the result data of the calibration runs.
- Extracting the result data of the trial runs.
- Modifying the schema or instance of a master database or a test database,

The command line interface is based on Java (as is the whole application) and therefore to run a task the following two steps have to be considered:

- Adding the following jar files (provided in the lib directory) to the class path:
  - commons-cli-1.0.jar (from The Apache Software Foundation)
  - commons-math-1.1.jar (from The Apache Software Foundation)
  - jxl.jar (from Andy Khan's Java Excel API)
  - ojdbc14.jar (from Oracle Corporation)

```
set CLASSPATH=.;lib\repsi-tool-02.00.jar;lib\commmons-cli-
1.0.jar;lib\commmons-math-1.1.jar;lib\jxl.jar;lib\ojdbc14.jar
```

- Extending the Windows path variable to include the JRE5.0 or combine the invocation of the JVM with the appropriate directory.

An equivalent logging level of the JVM (e.g. `.level=FINER`) provides very detailed information of the program flow (see the file `logging.properties` in directory `…\jre\lib` of your current JVM).

The selection and execution of a task is guided by options. The options (see Table 4) may be supplemented by arguments. Options are case sensitive but the order of the options is arbitrary.

Option	Argument	Description
cyc	999	Number of cycles to run the calibration or the trial. The default value is 50 (calibration) / 1 (trial) and with mode calibration the maximum value is 999.
des	text	A description of the specific calibration or trial run (enclosed in quotes). The default value is "n/a".
di	999	Identification of the database instance.
efn	file name	Output file (optionally included a directory) containing the required trial in an Excel-like file. The default value is "`CalibrationData.xls`" with mode "result_cn" and "`TrialRunData.xls`" with mode "result_tl", both in directory "`out`".
efnall	n/a	Indicates that the result data of all available trial runs should be included in the Excel file (default: only the latest trial run is included).
exalt	n/a	Executes the queries alternating without and with the application of the pattern (e.g. q1, q2, q1, q2, …).
excon	n/a	Executes first the query without application of the pattern and then the query with application of the pattern (e.g. q1, q1, …, q1, q2,q2, …,q2).
fn0xls … **fn9xls**	n/a	Indicates that the input file contains an Excel-like file.
fn0xml … **fn9xml**	n/a	Indicates that the input file contains an XML document.
fs	fetch size	Determines the number of rows physically retrieved from the database at one time by the JDBC driver when scrolling through a query `ResultSet` with `next()`. The default value is 10.
idn	directory name	Name of the directory in which the R compatible files should be stored. The default value is '`in/R`'.
ign1	n/a	Ignore first reading.
mode	{calibration | master | result_cn | result_tl | test | trial}	Defines the task to be executed.
obj	{nanotime |	Defines the object to be calibrated.

Option	Argument	Description
	query}	
pf	file name	File name (optionally including a directory) of the properties file which contains the connection details of the master database. The default value is file '`configuration.properties`' in directory '`config`'.
pfxml	n/a	Indicates that the properties file contains an XML document.
prec	999	Exponent (base 10) of the time precision. The default value is 0.
tqp	999	Identification of a test query pair.
ts	999	Identification of the test suite.
verb	n/a	Print an statistical overview

Table 4: All Available Command Line Options

The options 'mode' determines the task to be executed and is always mandatory:

- **`calibration`** Perform a calibration run,
- **`master`** Modify the schema or instance of the master database,
- **`result_cn`** Export the results of a calibration run into an Excel file,
- **`result_r`** Export the results of a calibration run into a R compatible file,
- **`result_tl`** Export the results of a trial run into an Excel file,
- **`test`** Modify the schema or instance of a test database,
- **`trial`** Perform a trial run.

Depending on the specific task additional options are either mandatory or optional (see Table 5):

Option	Mandatory with mode	Optional with mode
cyc		calibration, trial
des		calibration, trial
di	calibration, test, trial	
efn		result_cn, result_tl
efnall		result_tl
exalt		calibration
excon		calibration
fn0 … fn9	at least one with master or test	

Option	Mandatory with mode	Optional with mode
fn0xls … fn9xls	not along with fn.xml	master, test
fn0xml … fn9xml	not along with fn.xls	master, test
fs		calibration, trial
idn		calibration
ign1		calibration
mode	all modes	
obj	calibration	
pf		all modes
pfxml		all modes
prec		calibration, trial
tqp	calibration with object 'query'	
ts	trial	
verb		calibration

Table 5: Available Options Depending on Mode

The following example modifies the schema or instance of the test database, based on the content of the XML file `'test_drop_schema.xml'`, with the database instance identifier 1. The file `'configuration.properties'` in directory "`config`" contains the connection details of the master database.

```
> set CLASSPATH=.;lib\repsi-tool-02.00.jar;lib\commmons-cli-
1.0.jar;lib\commmons-math-1.1.jar;lib\jxl.jar;lib\ojdbc14.jar
> set REPSI_JAVA_HOME=%JAVA_HOME%
> %REPSI_JAVA_HOME%\bin\java -jar lib\repsi-tool-02.00.jar -
mode test -di 1 -fn0 in\test_drop_schema.xml -fn0xml
```

2 Detailed Data Model

Table TMD_CALIBRATION

Column Name	ID	Pk	Null?	Data Type	Default	C
START_TIME	1	1	N	TIMESTAMP(6)		
CREATED_BY	2		Y	VARCHAR2 (30 Byte)		
DATE_CREATED	3		Y	DATE		
COMPARISON_EQUALS	4		Y	VARCHAR2 (1 Byte)		
COMPARISON_MESSAGE	5		Y	VARCHAR2 (2000 Byte)		
DATABASE_SYSTEM_NAME	6		Y	VARCHAR2 (30 Byte)		
DATABASE_SYSTEM_VENDOR_NAME	7		Y	VARCHAR2 (50 Byte)		
DATABASE_SYSTEM_VERSION	8		Y	VARCHAR2 (20 Byte)		
DESCRIPTION	9		N	VARCHAR2 (2000 Byte)	'n/a'	
END_TIME	10		Y	TIMESTAMP(6)		
ERROR_MESSAGE	11		Y	VARCHAR2 (2000 Byte)		
JDBC_DRIVER	12		Y	VARCHAR2 (255 Byte)		
JDBC_URL	13		Y	VARCHAR2 (255 Byte)		
OPERATING_SYSTEM_NAME	14		Y	VARCHAR2 (50 Byte)		
OPERATING_SYSTEM_VENDOR_NAME	15		Y	VARCHAR2 (50 Byte)		
OPERATING_SYSTEM_VERSION	16		Y	VARCHAR2 (20 Byte)		
PATTERN_SQL_IDIOM_NAME	17		Y	VARCHAR2 (255 Byte)		
PROCESSOR_NAME	18		Y	VARCHAR2 (50 Byte)		
PROCESSOR_VENDOR_NAME	19		Y	VARCHAR2 (50 Byte)		
RAM_SIZE_MB	20		Y	NUMBER (6)		
SCHEMA_NAME	21		Y	VARCHAR2 (30 Byte)		
SQL_SYNTAX_CODE_DEI	22		Y	VARCHAR2 (10 Byte)		
SQL_SYNTAX_CODE_TTQP	23		Y	VARCHAR2 (10 Byte)		
STATUS_CODE	24		N	CHAR (2 Byte)		
TEST_QUERY_PAIR_ID	25		Y	NUMBER (8)		
USER_NAME	26		Y	VARCHAR2 (30 Byte)		

Table TMD_CALIBRATION_STATISTIC

Column Name	ID	Pk	Null?	Data Type	Default
START_TIME	1	1	N	TIMESTAMP(6)	
SEQUENCE_NUMBER	2	2	N	NUMBER (8)	
OBJECT	3		N	VARCHAR2 (255 Byte)	
ORDER_BY	4		Y	VARCHAR2 (2000 Byte)	
SQL_STATEMENT	5		Y	VARCHAR2 (2000 Byte)	
ARITHMETIC_MEAN	6		Y	NUMBER (38)	
GEOMETRIC_MEAN	7		Y	NUMBER (38)	
KURTOSIS	8		Y	NUMBER (38)	
SKEWNESS	9		Y	NUMBER (38)	
STANDARD_DEVIATION	10		Y	NUMBER (38)	
VARIANCE	11		Y	NUMBER (38)	
MINIMUM_VALUE	12		N	NUMBER (20)	
PERCENTILE_25	13		Y	NUMBER (38)	
PERCENTILE_50	14		Y	NUMBER (38)	
PERCENTILE_75	15		Y	NUMBER (38)	
MAXIMUM_VALUE	16		N	NUMBER (20)	
NUMBER_OF_VALUES	17		N	NUMBER (10)	
READINGS	18		Y	CT_MD_READINGS	

Type CT_MD_READINGS

```
CREATE OR REPLACE TYPE ct_md_readings
       AS VARRAY (999) OF NUMERIC (20)
```

Table TMD_DATABASE_INSTANCE

Column Name	ID	Pk	Null?	Data Type	Default
DATABASE_INSTANCE_ID	1	1	N	NUMBER (8)	
DATE_CREATED	2		Y	DATE	
CREATED_BY	3		Y	VARCHAR2 (30 Byte)	
DATE_MODIFIED	4		Y	DATE	
MODIFIED_BY	5		Y	VARCHAR2 (30 Byte)	
DATABASE_SYSTEM_CODE	6		N	VARCHAR2 (20 Byte)	
JDBC_URL	7		N	VARCHAR2 (255 Byte)	
OPERATING_SYSTEM_CODE	8		N	VARCHAR2 (20 Byte)	
PASSWORD	9		N	VARCHAR2 (30 Byte)	
PROCESSOR_CODE	10		N	VARCHAR2 (20 Byte)	
RAM_SIZE_MB	11		N	NUMBER (6)	
SCHEMA_NAME	12		N	VARCHAR2 (30 Byte)	
USER_NAME	13		N	VARCHAR2 (30 Byte)	

Table TMD_DATABASE_SYSTEM

Column Name	ID	Pk	Null?	Data Type	Default
DATABASE_SYSTEM_CODE	1	1	N	VARCHAR2 (20 Byte)	
DATE_CREATED	2		Y	DATE	
CREATED_BY	3		Y	VARCHAR2 (30 Byte)	
DATE_MODIFIED	4		Y	DATE	
MODIFIED_BY	5		Y	VARCHAR2 (30 Byte)	
JDBC_DRIVER	6		N	VARCHAR2 (255 Byte)	
NAME	7		N	VARCHAR2 (30 Byte)	
SQL_SYNTAX_CODE	8		N	VARCHAR2 (10 Byte)	
VENDOR_CODE	9		N	VARCHAR2 (10 Byte)	
VERSION	10		N	VARCHAR2 (20 Byte)	

Table TMD_OPERATING_SYSTEM

Column Name	ID	Pk	Null?	Data Type	Default
OPERATING_SYSTEM_CODE	1	1	N	VARCHAR2 (20 Byte)	
DATE_CREATED	2		Y	DATE	
CREATED_BY	3		Y	VARCHAR2 (30 Byte)	
DATE_MODIFIED	4		Y	DATE	
MODIFIED_BY	5		Y	VARCHAR2 (30 Byte)	
NAME	6		N	VARCHAR2 (50 Byte)	
VENDOR_CODE	7		N	VARCHAR2 (10 Byte)	
VERSION	8		N	VARCHAR2 (20 Byte)	

Table TMD_PATTERN_SQL_IDIOM

Column Name	ID	Pk	Null?	Data Type	Default
PATTERN_SQL_IDIOM_ID	1	1	N	NUMBER (8)	
DATE_CREATED	2		Y	DATE	
CREATED_BY	3		Y	VARCHAR2 (30 Byte)	
DATE_MODIFIED	4		Y	DATE	
MODIFIED_BY	5		Y	VARCHAR2 (30 Byte)	
CONTEXT_APPLICABILITY	6		N	VARCHAR2 (2000 Byte)	'n/a'
FORCES	7		N	VARCHAR2 (2000 Byte)	'n/a'
NAME	8		N	VARCHAR2 (255 Byte)	
PROBLEM_INTENT	9		N	VARCHAR2 (2000 Byte)	'n/a'
SOLUTION	10		N	VARCHAR2 (2000 Byte)	'n/a'

Table TMD_PROCESSOR

Column Name	ID	Pk	Null?	Data Type	Default
PROCESSOR_CODE	1	1	N	VARCHAR2 (20 Byte)	
DATE_CREATED	2		Y	DATE	
CREATED_BY	3		Y	VARCHAR2 (30 Byte)	
DATE_MODIFIED	4		Y	DATE	
MODIFIED_BY	5		Y	VARCHAR2 (30 Byte)	
NAME	6		N	VARCHAR2 (50 Byte)	
VENDOR_CODE	7		N	VARCHAR2 (10 Byte)	

Table TMD_SQL_SYNTAX

Column Name	ID	Pk	Null?	Data Type	Default
SQL_SYNTAX_CODE	1	1	N	VARCHAR2 (10 Byte)	
DATE_CREATED	2		Y	DATE	
CREATED_BY	3		Y	VARCHAR2 (30 Byte)	
DATE_MODIFIED	4		Y	DATE	
MODIFIED_BY	5		Y	VARCHAR2 (30 Byte)	
NAME	6		N	VARCHAR2 (255 Byte)	
VERSION	7		N	VARCHAR2 (20 Byte)	

Table TMD_TEST_QUERY_PAIR

Column Name	ID	Pk	Null?	Data Type	Default
TEST_QUERY_PAIR_ID	1	1	N	NUMBER (8)	
DATE_CREATED	2		Y	DATE	
CREATED_BY	3		Y	VARCHAR2 (30 Byte)	
DATE_MODIFIED	4		Y	DATE	
MODIFIED_BY	5		Y	VARCHAR2 (30 Byte)	
APPLIED_PATTERN_ORDER_BY	6		Y	VARCHAR2 (2000 Byte)	
APPLIED_PATTERN_SELECT_STMNT	7		N	VARCHAR2 (2000 Byte)	
DESCRIPTION	8		N	VARCHAR2 (2000 Byte)	'n/a'
PATTERN_SQL_IDIOM_ID	9		N	NUMBER (8)	
SQL_SYNTAX_CODE	10		N	VARCHAR2 (10 Byte)	
UNAPPLIED_PATTERN_ORDER_BY	11		Y	VARCHAR2 (2000 Byte)	
UNAPPLIED_PATTERN_SELECT_STMNT	12		N	VARCHAR2 (2000 Byte)	

Table TMD_TEST_SUITE

Column Name	ID	Pk	Null?	Data Type	Default
TEST_SUITE_ID	1	1	N	NUMBER (8)	
DATE_CREATED	2		Y	DATE	
CREATED_BY	3		Y	VARCHAR2 (30 Byte)	
DATE_MODIFIED	4		Y	DATE	
MODIFIED_BY	5		Y	VARCHAR2 (30 Byte)	
DESCRIPTION	6		N	VARCHAR2 (2000 Byte)	'n/a'
NAME	7		N	VARCHAR2 (50 Byte)	

Table TMD_TEST_SUITE_ACTION

Column Name	ID	Pk	Null?	Data Type	Default
TEST_SUITE_ID	1	1	N	NUMBER (8)	
SEQUENCE_NUMBER	2	2	N	NUMBER (8)	
DESCRIPTION	3		N	VARCHAR2 (2000 Byte)	'n/a'
EXECUTION_FREQUENCY	4		N	NUMBER (8)	1
OPERATION_CODE	5		N	CHAR (2 Byte)	
TABLE_NAME	6		Y	VARCHAR2 (30 Byte)	
TEST_QUERY_PAIR_ID	7		Y	NUMBER (8)	

Table TMD_TEST_SUITE_OPERATION

Column Name	ID	Pk	Null?	Data Type	Default
OPERATION_CODE	1	1	N	CHAR (2 Byte)	
DATE_CREATED	2		Y	DATE	
CREATED_BY	3		Y	VARCHAR2 (30 Byte)	
DATE_MODIFIED	4		Y	DATE	
MODIFIED_BY	5		Y	VARCHAR2 (30 Byte)	
DESCRIPTION	6		N	VARCHAR2 (2000 Byte)	'n/a'
EXECUTION_FREQUENCY_MAX	7		N	NUMBER (8)	1
NAME	8		N	VARCHAR2 (50 Byte)	
OPERATION_TYPE	9		N	VARCHAR2 (10 Byte)	'Query'

Table TMD_TEST_TABLE

Column Name	ID	Pk	Null?	Data Type	Default
NAME	1	1	N	VARCHAR2 (30 Byte)	
DATE_CREATED	2		Y	DATE	
CREATED_BY	3		Y	VARCHAR2 (30 Byte)	
DATE_MODIFIED	4		Y	DATE	
MODIFIED_BY	5		Y	VARCHAR2 (30 Byte)	
DESCRIPTION	6		N	VARCHAR2 (255 Byte)	'n/a'
SQL_SYNTAX_CODE	7		N	VARCHAR2 (10 Byte)	'SQL:1999'

Table TMD_TEST_TABLE_DDL

Column Name	ID	Pk	Null?	Data Type	Default
NAME	1	1	N	VARCHAR2 (30 Byte)	
SEQUENCE_NUMBER	2	2	N	NUMBER (8)	
DDL_STATEMENT	3		N	VARCHAR2 (2000 Byte)	
SQL_SYNTAX_CODE	4		N	VARCHAR2 (10 Byte)	

Table TMD_TRIAL_RUN

Column Name	ID	Pk	Null?	Data Type	Default
DATABASE_INSTANCE_ID	1	1	N	NUMBER (8)	
TEST_SUITE_ID	2	2	N	NUMBER (8)	
START_TIME	3	3	N	TIMESTAMP(6)	
CREATED_BY	4		Y	VARCHAR2 (30 Byte)	
DATE_CREATED	5		Y	DATE	
DATABASE_SYSTEM_NAME	6		N	VARCHAR2 (30 Byte)	
DATABASE_SYSTEM_VENDOR_NAME	7		N	VARCHAR2 (50 Byte)	
DATABASE_SYSTEM_VERSION	8		N	VARCHAR2 (20 Byte)	
DESCRIPTION	9		N	VARCHAR2 (2000 Byte)	'n/a'
END_TIME	10		Y	TIMESTAMP(6)	
ERROR_MESSAGE	11		Y	VARCHAR2 (2000 Byte)	
JDBC_DRIVER	12		N	VARCHAR2 (255 Byte)	
JDBC_URL	13		N	VARCHAR2 (255 Byte)	
OPERATING_SYSTEM_NAME	14		N	VARCHAR2 (50 Byte)	
OPERATING_SYSTEM_VENDOR_NAME	15		N	VARCHAR2 (50 Byte)	
OPERATING_SYSTEM_VERSION	16		N	VARCHAR2 (20 Byte)	
PROCESSOR_NAME	17		N	VARCHAR2 (50 Byte)	
PROCESSOR_VENDOR_NAME	18		N	VARCHAR2 (50 Byte)	
RAM_SIZE_MB	19		N	NUMBER (6)	
SCHEMA_NAME	20		N	VARCHAR2 (30 Byte)	
SQL_SYNTAX_CODE	21		N	VARCHAR2 (10 Byte)	
STATUS_CODE	22		N	CHAR (2 Byte)	
TEST_SUITE_DESCRIPTION	23		N	VARCHAR2 (2000 Byte)	
TEST_SUITE_NAME	24		N	VARCHAR2 (50 Byte)	
USER_NAME	25		N	VARCHAR2 (30 Byte)	

Table TMD_TRIAL_RUN_ACTION

Column Name	ID	Pk	Null?	Data Type	Default
DATABASE_INSTANCE_ID	1	1	N	NUMBER (8)	
TEST_SUITE_ID	2	2	N	NUMBER (8)	
START_TIME	3	3	N	TIMESTAMP(6)	
SEQUENCE_NUMBER_ACTION	4	4	N	NUMBER (8)	
APPLIED_DURATION	5		Y	NUMBER (38)	
APPLIED_END_TIME	6		Y	TIMESTAMP(6)	
APPLIED_ERROR_MESSAGE	7		Y	VARCHAR2 (2000 Byte)	
APPLIED_PATTERN_ORDER_BY	8		Y	VARCHAR2 (2000 Byte)	
APPLIED_PATTERN_SELECT_STMNT	9		Y	VARCHAR2 (2000 Byte)	
APPLIED_STATUS	10		Y	CHAR (2 Byte)	
APPLIED_START_TIME	11		Y	TIMESTAMP(6)	
COMPARISON_EQUALS	12		N	VARCHAR2 (1 Byte)	'U'
COMPARISON_MESSAGE	13		Y	VARCHAR2 (2000 Byte)	
EXECUTION_FREQUENCY	14		N	NUMBER (8)	
OPERATION_CODE	15		N	CHAR (2 Byte)	
OPERATION_TYPE	16		N	VARCHAR2 (10 Byte)	
PATTERN_SQL_IDIOM_NAME	17		Y	VARCHAR2 (255 Byte)	
SQL_SYNTAX_CODE	18		N	VARCHAR2 (10 Byte)	
TABLE_NAME	19		Y	VARCHAR2 (30 Byte)	
TEST_QUERY_PAIR_DESCRIPTION	20		Y	VARCHAR2 (2000 Byte)	
TEST_SUITE_ACTION_DESCRIPTION	21		N	VARCHAR2 (2000 Byte)	
TEST_SUITE_OPERATION_NAME	22		N	VARCHAR2 (50 Byte)	
TEST_TABLE_DESCRIPTION	23		Y	VARCHAR2 (255 Byte)	
UNAPPLIED_DURATION	24		Y	NUMBER (38)	
UNAPPLIED_END_TIME	25		Y	TIMESTAMP(6)	
UNAPPLIED_ERROR_MESSAGE	26		Y	VARCHAR2 (2000 Byte)	
UNAPPLIED_PATTERN_ORDER_BY	27		Y	VARCHAR2 (2000 Byte)	
UNAPPLIED_PATTERN_SELECT_STMNT	28		Y	VARCHAR2 (2000 Byte)	
UNAPPLIED_START_TIME	29		Y	TIMESTAMP(6)	
UNAPPLIED_STATUS	30		Y	CHAR (2 Byte)	

Table TMD_TRIAL_RUN_PROTOCOL

Column Name	ID	Pk	Null?	Data Type	Default
DATABASE_INSTANCE_ID	1		N	NUMBER (8)	
TEST_SUITE_ID	2		N	NUMBER (8)	
START_TIME	3	1	N	TIMESTAMP(6)	
SEQUENCE_NUMBER_PROTOCOL	4	2	N	NUMBER (8)	
CREATED	5		N	TIMESTAMP(6)	
MESSAGE	6		N	VARCHAR2 (2000 Byte)	
SEQUENCE_NUMBER_ACTION	7		Y	NUMBER (8)	

Table TMD_RUN_STATUS

Column Name	ID	Pk	Null?	Data Type	Default
STATUS_CODE	1	1	N	CHAR (2 Byte)	
DATE_CREATED	2		Y	DATE	
CREATED_BY	3		Y	VARCHAR2 (30 Byte)	
DATE_MODIFIED	4		Y	DATE	
MODIFIED_BY	5		Y	VARCHAR2 (30 Byte)	
DESCRIPTION	6		N	VARCHAR2 (2000 Byte)	'n/a'
NAME	7		N	VARCHAR2 (50 Byte)	

Table TMD_VENDOR

Column Name	ID	Pk	Null?	Data Type	Default
VENDOR_CODE	1	1	N	VARCHAR2 (10 Byte)	
DATE_CREATED	2		Y	DATE	
CREATED_BY	3		Y	VARCHAR2 (30 Byte)	
DATE_MODIFIED	4		Y	DATE	
MODIFIED_BY	5		Y	VARCHAR2 (30 Byte)	
NAME	6		N	VARCHAR2 (50 Byte)	

3 Directory and File Layout – Binary Version

- **config**\
  - **configuration.properties** – A Java properties file containing the connection parameters for the master database.
- **doc**\ - Containing all the documentation.
  - **api**\ - Javadoc documentation files.
  - **REPSI_Tool_Requirements_Elicitation.pdf** – Requirements elicitation document.
  - **REPSI_Tool_User_Manual.pdf** – This document – the User Manual.
- **in**\ - Containing all example input files for the REPSI tool.
  - **dtd**\
    - **script_file.dtd** – The document type definition file containing the conformance rules of the input files.
  - **x…x.sql** – Files containing SQL DDL or DML statements.
  - **x…x.xls** – Excel file containing database table content.
  - **x…x.xml** – Simple XML documents containing SQL DDL or DML statements.
- **lib**\ - Containing all the necessary JAR files to run the tool.
- **licences**\ -Licence documentation of the tool and all the used third-party software.
- **out**\ - Empty auxiliary directory – assumed to be used for any tool output files.
- **xxx.{bat|sh}** – Several template batch files for the Windows environment and bash shell scripts for the Linux environment.

4 Directory and File Layout – Development Version

In addition to the directories and files of the binary version, the development version contains the following relevant directories and files. File not mentioned here are mostly configuration files of third-party software (e.g. Checkstyle, Eclipse, FindBugs, PMD, and so on).

- **bin**\ - Contains the templates of the batch files for the binary version.
- **classes**\ - Contains the class files.
- **doc**\ - Contains, in addition to the binary version, the Microsoft Word versions of the requirements elicitation and the user manual document.
- **launch**\ - Contains Eclipse specific backup files of the used launch configurations to run Java applications or to create the distribution files.
- **preferences**\ - Contains backup files of third party software configuration parameters, e.g. Checkstyle, Eclipse, or PMD.
- **sql**\ - Contains files comprising SQL commands which may be useful for maintenance purposes.
- **src**\ - Contains the Java source files.
- **build.xml** – The Ant build file to create the deployment files.

5 Master Database Property File Entries

The connection details of the master database have to be defined in a Java property file format (see Table 6) which may be a simple line or XML format according to the Java specifications.

See also http://java.sun.com/j2se/1.5.0/docs/api/java/util/Properties.html.

A template file is available in the directory `config`.

Key	Value
database.master.driver	The name of the database driver e.g. oracle.jdbc.driver.OracleDriver.
database.master.password	The password to authenticate against the database instance.
database.master.sql.syntax.code	
database.master.url	The URL of the database instance, e.g. jdbc:oracle:thin:@hostx:1521:dby.
database.master.username	The user name to authenticate against the database instance.

Table 6: Master Database Property File Entries

6 Tested Environments

The following system components were successfully used to run the REPSI tool:

Processors

AMD Duron™ 1800 MHz
Intel® Pentium® D CPU 930 3.00 GHz
Intel® Pentium® M processor 1700 MHz

Operating Systems

Microsoft Windows XP Professional (Servicepack 2)
SUSE Linux 10.1
openSUSE Linux 10.2

Java Environment

J2SE™ Development Kit 5.0 Update 10
Java™ SE (Standard Edition) Development Kit 6

Relational DBMS (Master and Test Database)

Oracle® Database 10g Express Edition
Oracle® Database 10g Release 2 (10.2.0.1.0) Enterprise/Standard Edition

Appendix A – Halevy's Data Model

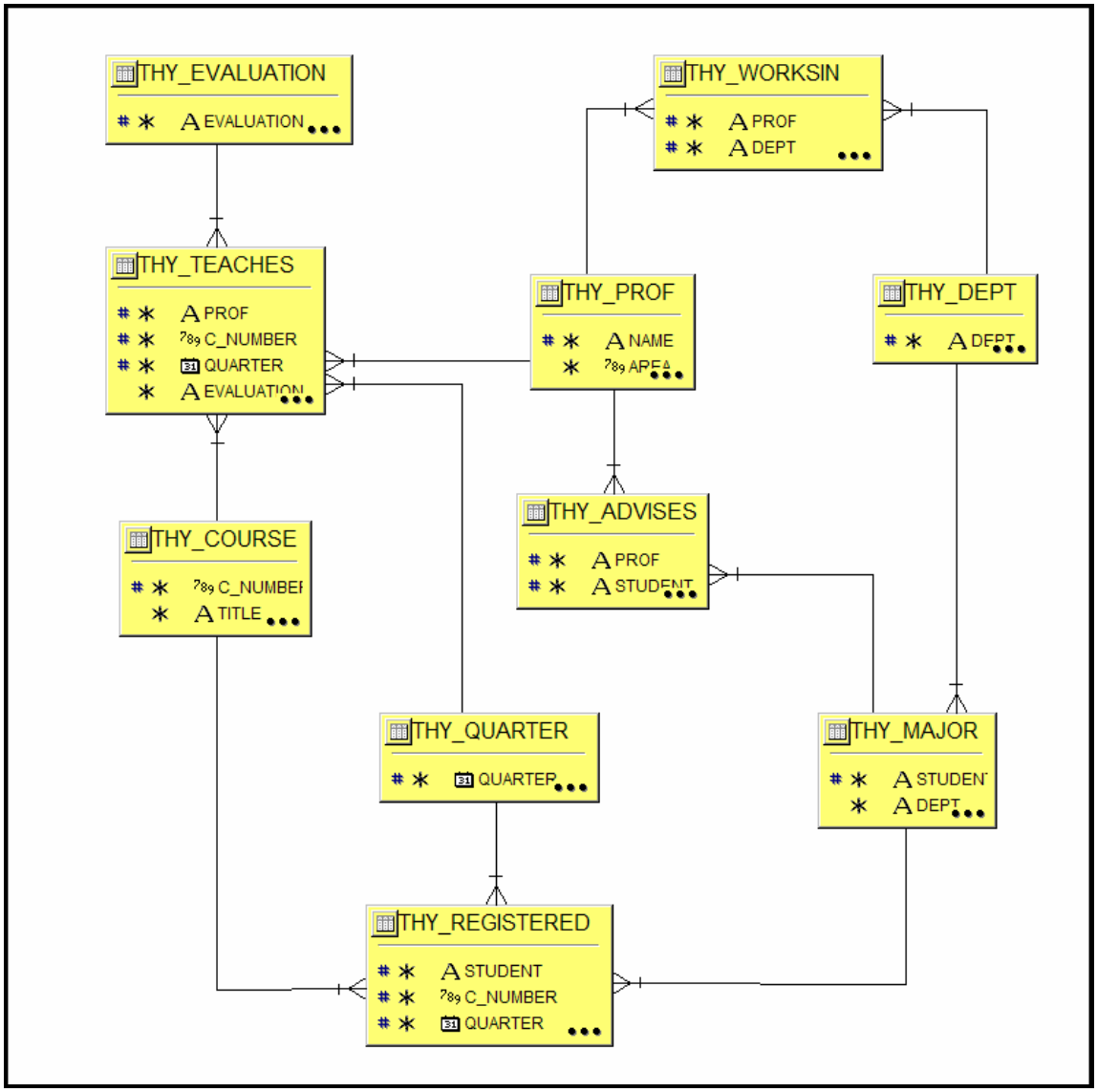

Figure 6: Data model based on Halevy (Halevy, 2001)

Appendix B – References

Apache Software Foundation (2006a) *Commons CLI*. Available from: http://jakarta.apache.org/commons/cli/ [Accessed 26.08.2006].

Apache Software Foundation (2006b) *Commons Math*. Available from: http://jakarta.apache.org/commons/math/ [Accessed 04.10.2006].

CollabNet (2006) *Projects > subclipse* Available from: http://subclipse.tigris.org/ [Accessed 27.08.2006].

Fogel, S. and Lane, P. (2006) *Oracle Database Administrator's Guide, 10g Release 2 (10.2)*, Oracle Corporation. Available from: http://download-uk.oracle.com/docs/cd/B19306_01/server.102/b14231.pdf [Accessed 27.08.2006].

Green, R. (2006) *time: Java glossary*, Canadian Mind Products. Available from: http://mindprod.com/jgloss/time.html [Accessed 05.09.2006].

Halevy, A. Y. (2001) 'Answering queries using views: A survey', *VLDB Journal,* 10 (4), pp. 270-294.

InfoEther (2006) *PMD*. Available from: http://pmd.sourceforge.net/ [Accessed 27.08.2006].

Khan, A. (2006) *Java Excel API - A Java API to read, write, and modify Excel spreadsheets*. Available from: http://jexcelapi.sourceforge.net/ [Accessed 26.08.2006].

Melton, J. and Simon, A. R. (2002) *SQL:1999 : Understanding Relational Language Components,* Morgan Kaufmann, San Francisco, CA, USA.

Microsoft Corporation (2005) *How to use QueryPerformanceCounter to time code*. Available from: http://support.microsoft.com/kb/172338/en-us [Accessed 05.09.2006].

Oracle Technology Network (2006) *Software downloads*. Available from: http://www.oracle.com/technology/software/index.html [Accessed 26.08.2006].

Quest Software (2006) *Toad for Oracle*. Available from: http://www.toadsoft.com/toad_oracle.htm [Accessed 27.08.2006].

R Foundation (2006) *The Comprehensive R Archive Network*. Available from: http://cran.at.r-project.org/ [Accessed 30.10.2006].

Schildt, H. (2004) *Java 2, v5.0 (Tiger) : new features,* McGraw-Hill, New York, USA.

Schneider, D. and Ködderitzsch, L. (2006) *Eclipse Checkstyle Plug-In*. Available from: http://eclipse-cs.sourceforge.net/ [Accessed 27.08.2006].

Sun Microsystems Inc. (2004) *Class System*. Available from: http://java.sun.com/j2se/1.5.0/docs/api/java/lang/System.html#nanoTime() [Accessed 05.09.2006].

Sun Microsystems Inc. (2006) *Java SE downloads*. Available from: http://java.sun.com/javase/downloads/index.jsp [Accessed 26.08.2006].

The Eclipse Foundation (2006) *Eclipse downloads home - get Eclipse*. Available from: http://www.eclipse.org/downloads/ [Accessed 26.08.2006].

University of Maryland (2006) *FindBugs™ - Find Bugs in Java Programs*. Available from: http://findbugs.sourceforge.net/ [Accessed 27.06.2006].

Appendix C – Bibliography

Burke, E. M. and Coyner, B. M. (2003) *Java Extreme Programming cookbook,* O'Reilly, Sebastopol, CA, USA.

Clayberg, E. and Rubel, D. (2006) *Eclipse : building commercial-quality plug-ins,* (2nd Edn), Addison-Wesley, Upper Saddle River, NJ, USA.

Eckel, B. (2006) *Thinking in Java,* (4th Edn), Prentice Hall, Upper Saddle River, NJ, USA.

Flanagan, D. (2005) *Java in a nutshell,* (5th Edn), O'Reilly, Sebastopol, CA, USA.

Holzner, S. and Tilly, J. (2005) *Ant : the definitive guide,* (2nd Edn), O'Reilly, Sebastopol, CA, USA.

McLaughlin, B. (2002) *Java & XML,* (2nd Edn), O'Reilly, Köln, Germany.

Nock, C. (2004) *Data access patterns : database interactions in object-oriented applications,* Addison-Wesley, Boston, MA, USA.

Reese, G. (2000) *Database programming with JDBC and Java,* (2nd Edn), O'Reilly, Cambridge, MA, USA.

Reese, G. (2003) *Java database best practices,* (1st Edn), O'Reilly, Sebastopol, CA, USA.

Sun Microsystems Inc. (2006) *Code Conventions for the Java Programming Language*. Available from: http://java.sun.com/docs/codeconv/ [Accessed 12.05.2006].