REPSI Tool

(Recording the Efficiency of Patterns / SQL Idioms)

Requirements Elicitation

Walter Weinmann

5th June, 2006

Contents

List of Figures

List of Abbreviations

API	Application Programming Interface
Java SE	Java Standard Edition
JDBC	Java Database Connectivity
RDBMS	Relational Database Management System
REPSI	Recording the Efficiency of Patterns / SQL Idioms

Chapter 1 Introduction

1 Purpose of the System

The REPSI (**R**ecording the **E**fficiency of **P**atterns / **S**QL **I**dioms) tool is an application designed to generate an appropriate schema and instance as a basis to run SQL statements both formulated with and without the application of a specific pattern or SQL idiom. The aim of the tool will be to measure and record the response times of the statements to allow an estimation of whether the related pattern or SQL idiom is applied by the database optimiser. Based on the results of running the tool with a specific database, an SQL user should have a guide to improve the efficiency of the specific database optimiser by the application of suitable patterns and SQL idioms. The tool will include a set of predefined patterns and SQL idioms together with the corresponding SQL queries.

2 Scope of the System

The REPSI tool will:

- Allow an authorised user to connect to a RDBMS (Relational Database Management System) called master RDBMS via the JDBC (Java Database Connectivity) API (Application Programming Interface) which is intended to:
  - Store data describing the supported test RDBMS including vendor, version, JDBC connection details etc,
  - Store a list of patterns and SQL idioms including data like name, context, problem, solution,
  - Store sets of parameters each to generate a specific relational database schema and instance called test RDBMS,
  - Store pairs of SQL statements
    - Each pair of SQL statements is related to a specific pattern or SQL idiom,
    - One SQL statement of the pair is formulated with application of the related pattern or SQL idiom and one without,
  - Store execution plans called test suites each defining the sequence and frequency of executing a set of pairs of SQL statements,

    - Store the characteristics, potential errors and response times of each of the SQL statements executed during the execution of a test suite.
- Allow the user to create and change connection parameter to a master RDBMS.
- Allow the user to generate a schema and instance of a master RDBMS including:
    - Example characteristics of the test RDBMS,
    - Patterns and SQL idioms with related SQL statements,
    - An example parameter set to generate a schema and an instance in the test RDBMS,
    - An example test suite incorporating all the SQL statements related to the patterns and SQL idioms.
- Allow the user to run a test suite including the detailed recording of the characteristics, potential errors and response times of the SQL statements (runtime results).
- Allow the user to view the characteristics, potential errors and response times of the SQL statements recorded during the run of an execution plan.
- Allow the user to use the same RDBMS as master RDBMS and as test RDBMS.

The REPSIT application does not need to:

- Check that a SQL statement related to a pattern or SQL idiom covers the intention represented by this pattern or SQL idiom.
- Create database users or roles.
- Connect to a RDBMS other than via JDBC.
- Consider specific requirements related to distributed database systems.
- Consider the influence of database remote access to the response times.
- Interpret the measured response times.

3 Objectives and Success Criteria

The REPSI tool will meet its objectives if:

- It is available as a prototype including the core functionality with a minimal user interface at the end of September 2006.
- It is available as a final version including the final user interface and the user documentation at the end of October 2006.
- It contains ready to run data including:
  - The characteristics of the test RDBMS,
  - A set of patterns and SQL idioms together with the related SQL statements,
  - At least one parameter set to generate a database schema and instance of the test RDBMS,
  - At least one test suite to run a set of SQL statements.
- It includes a detailed documentation of:
  - How to install the tool,
  - How to operate the tool, and
  - How to interpret the runtime results of a test suite.

Chapter 2 Proposed System

1 Functional requirements

System

The system is a passive actor in this tool which reacts or responds according to the actions of the user. The following is a list of the system's functionality:

- The system shall connect a sufficient authorised user to the master RDBMS.
- For the test RDBMS, the system shall allow the user to:
  - Maintain descriptive features like name, vendor, or version,
  - Maintain JDBC connection information like driver, URL, user name, or password,
  - Define the required variation of the SQL syntax,
  - Maintain suitable parameters to create a schema and an instance as an environment for running SQL statements.
- For patterns or SQL idioms, the system shall allow the user to:
  - Maintain descriptive features like name, context, problem, or solution,
  - Maintain pairs of SQL statements one formulated both with and without the application of the pattern or SQL idiom.
- For test suites, the system shall allow the user to specify the execution plans consisting of a predefined sequence of activities like
  - Creating a database table,
  - Generating the rows of database table,
  - Executing one of the both SQL statements of a certain pattern or SQL idiom in a defined frequency.
- The system shall execute on demand a selected test suite in the test RDBMS environment and thereby log in a very detailed manner the execution of the single SQL statements including:
  - The identification of the test suite,
  - The position of the SQL statement within the test suite,
  - The SQL statement itself,

    - The timestamp directly before the appropriate JDBC interface call,
    - The timestamp directly after the appropriate JDBC interface call,
    - A potential error message from the JDBC interface,
    - Available RDBMS specific information, e.g. an execution plan with an Oracle RDBMS.
- The system shall present on demand the runtime results of the execution of a test suite in a drilldown fashion with different levels:
    - **Test Suite Overview**: a list of the executed test suites starting with the latest execution time
        - Selecting a certain test suite shows the SQL statement overview.
    - **SQL Statement Overview**: a list of the executed statements in the sequence as they were executed during the run of the test suite
        - This list may be reduced by selecting a certain pattern, failed or successful statements, or an interval of response times,
        - Selecting a certain SQL statement shows the SQL statement details.
    - **SQL Statement Details**: all details which were recorded during the execution of the statements.

User

A user is a person who can start the tool, is able to connect to the master RDBMS and test RDBMS, and access both RDBMSs with sufficient privileges to create, drop, insert, and select tables. A user shall be able to do:

- Maintain preferences, e.g. connection details to a master RDBMS,
- Login to a master RDBMS,
- Logout of the master RDBMS,
- Create a schema and an instance with a ready to run set of data in the master RDBMS,
- Maintain the following data of the generated instance of the master RDBMS:
    - Characteristics of the test RDBMS,
    - Patterns and SQL idioms including the related SQL statements,
    - Parameter sets to generate a schema and an instance of the test RDBMS,
    - Test suites defining the sequence and frequency in which defined SQL statements should be executed.
- View the results of the executed test suites,

2 Non-Functional requirements

Deployability

- The tool should require as few as possible prerequisites and dependencies to facilitate the deployment.

Maintainability

- The most important aspect of maintainability is the application of a test driven development approach.
- Additionally, appropriate documentation of the architecture, design, and implementation is indispensable to assure maintainability.
- The Java code should conform to Sun's Java code conventions.

Security

- The access to the RDBMSs must demand appropriate authentication from the user of the tool.

3 Process Constraints

Runtime Environment

- All database connections must be implemented by applying JDBC (Java Database Connectivity) API.
- The tool must support the SQL standard ISO/IEC 9075:2003 as a basis of the master RDBMS and test RDBMS. The level of support may be restricted by the limitations provided by the chosen master RDBMS and test RDBMS.
- The tool must support additionally by default the current version of RDBMS Oracle (10g Release 2) as a basis of the master RDBMS and the test RDBMS.

Development Environment

- The development environment must be Eclipse SDK Version 3.2.
- The tool must be implemented using the programming language Java SE version 6.0.

Chapter 3 Use Cases

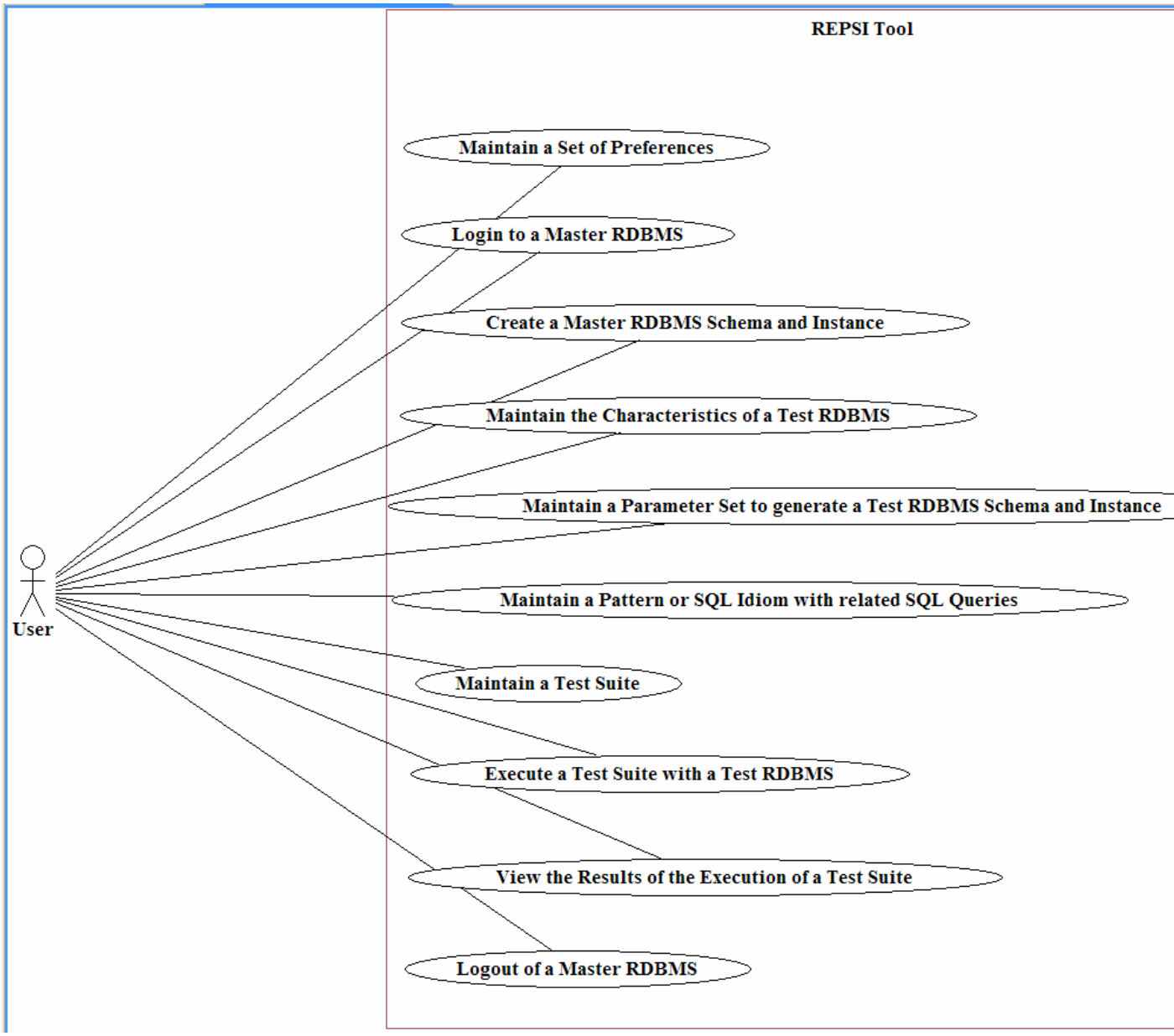

Figure 1: Use Case Overview

1 Maintain a Set of Preferences

Primary Actor:

User

Interests:

User wants to maintain the current set of preferences.

Preconditions:

System is installed and started.

Postconditions:

- The current set of preferences is modified if desired.
- The modified set of parameters is saved for later reuse if desired.

Basic Flow:

- User chooses to maintain the current set of preferences.
- System presents the current set of preferences.
- User modifies the current set of preferences.
- System validates the modified set of preferences.
- User requires storing the current set of preferences for later reuse.
- System tries to save the current set of parameters for later reuse and presents a failure or success message.

Alternative Flow:

User may select a previously saved set of preferences and load them as current set of preferences.

Frequency of Occurrence:

Occasionally

2 Login to Master RDBMS

Primary Actor:

User

Interests:

User wants to connect to the master RDBMS defined in the current preferences.

Preconditions:

The current set of preferences contains (eventually except the password) the connection data of an available master RDBMS.

Postconditions:

The user is connected to the master RDBMS as defined in the current set of preferences.

Basic Flow:

- User requires a connection to the master RDBMS.
- System presents the connection details as defined in the current set of preferences.
- User may enter or change the presented password.
- System tries to connect to the master RDBMS and presents a failure or a success message to the user.

Frequency of Occurrence:

At least once per session with the tool (except the user wants only to maintain preferences)

3 Create a Master RDBMS Schema and Instance

Primary Actor:

User

Interests:

User wants to create or recreate a schema and an instance in the master RDBMS.

Preconditions:

User is connected to the master RDBMS.

Postconditions:

- Already existing elements of the schema to be created are deleted in the master RDBMS.
- A new schema and an instance are created in the master RDBMS.

Basic Flow:

- User chooses to create a new schema in the master RDBMS.
- System checks for already existing elements of the schema to be created and asks the user for their deletion.
- If the user agrees or if there were no existing elements in the master RDBMS, the system:
  - Creates the necessary elements of the schema in the master RDBMS, and
  - Generates the instance with a basic set of data.

Frequency of Occurrence:

Normally once per master RDBMS

4 Maintain a Parameter Set to Generate the Test RDBMS Schema and Instance / Maintain a Pattern or SQL Idiom / Maintain a Test Suite / Maintain the Characteristics of the Test RDBMS

Primary Actor:

User

Interests:

User wants to maintain a data element of one of the following types:

- Parameter set to generate the test RDBMS schema and instance,
- Pattern / SQL idiom,
- Test suite, or
- Characteristics of the test RDBMS.

Preconditions:

- User is connected to the master RDBMS.
- User has previously created a schema and instance in the master RDBMS successfully.

Postconditions:

The chosen data element is maintained as intended by the user.

Basic Flow:

- User chooses to maintain a data element in the master RDBMS.
- System presents a form with appropriate selection parameters.
- User enters the selection parameters optionally.
- System presents a list containing the data elements according to the selection parameters.
- User selects one data element from the list with intent to maintain this data element.
- System presents all details of the selected data element in a data entry form.
- User may change the details of data element.
- System validates the modified data element.
- User requires storing the validated data element in the master RDBMS.
- System tries to store the validated data element and presents a failure message or a success message to the user.

Alternative Flow:

- User chooses to create a new data element in the master RDBMS.

- System provides an empty form.
- User enters the data.
- System validates the new data element.
- User requires storing the validated data element in the master RDBMS.
- System tries to store the validated data element and presents a failure message or a success message to the user.

5 Execute a Test Suite with the Test RDBMS

Primary Actor:

User

Interests:

User wants to execute a test suite with the test RDBMS.

Preconditions:

- User is connected to the master RDBMS.
- User has previously created a schema and instance in the master RDBMS successfully.

Postconditions:

The test suite is executed using the test RDBMS and the results are recorded in the master RDBMS.

Basic Flow:

- User chooses to execute a test suite.
- System presents a list of the available test suites.
- User chooses a test suite.
- REPSI connects to the test RDBMS and presents a failure or success message.
- User starts the execution of the test suite.
- System executes the SQL statements contained in the test suite in the defined sequence and frequency. System records the characteristics, error messages and response time of each executed SQL statement in the master RDBMS. System presents a message summarising the number of executed SQL statements both with and without errors.

Frequency of Occurrence:

Main purpose of the tool

6 View the Results of the Execution of a Test Suite

Primary Actor:

User

Interests:

User wants to view the recorded results of executed test suites.

Preconditions:

- User is connected to the master RDBMS.
- User has previously created a schema and instance in the master RDBMS successfully.
- User has previously executed at least one test suite.

Preconditions:

Not applicable

Basic Flow:

- User chooses to view the results of the execution of a test suite.
- System presents the list of previously executed test suites with the newest first.
- User selects a test suite.
- System presents a list of the executed SQL statements (SQL Statement Overview) in the sequence as they were executed during the run of the test suite.
- User may reduce the list of SQL statements by restricting it to a pattern or SQL idiom, failed or successful statements, or an interval of response times.
- System presents a reduced list of executed SQL statements.
- User selects an SQL statement.
- System presents all the details of the chosen SQL statement (SQL Statement Details):
  - The identification of the test suite,
  - The position of the SQL statement within the test suite,
  - The SQL statement itself,
  - The timestamp directly before the appropriate JDBC interface call,
  - The timestamp directly after the appropriate JDBC interface call,
  - An execution time as the difference of the two timestamps above,
  - A potential error message from the JDBC interface,

- Available RDBMS specific information, e.g. an execution plan with an Oracle RDBMS.

Alternative Flow:

- User may switch from 'SQL Statement Overview' back to 'Test Suite Overview'.
- User may switch from 'SQL Statement Details' back to 'Test Suite Overview'.
- User may switch from ' SQL Statement Details ' back to 'SQL statement Overview'.

Frequency of Occurrence:

Checking the results of the main purpose of the tool

7 Logout of Master RDBMS

Primary Actor:

User

Interests:

User wants to disconnect from the master RDBMS.

Preconditions:

User is connected to the master RDBMS.

Postconditions:

The user is disconnected of the master RDBMS.

Basic Flow:

- User requires the disconnection of the master RDBMS.
- System tries to disconnect of the master RDBMS and presents a failure or success message to the user.

Frequency of Occurrence:

- Required action before the user can connect to another master RDBMS.
- Recommended action before the user terminates working with the system.

Appendix

Appendix A – Bibliography

Larman, C. (2002) *Applying UML and patterns : an introduction to object-oriented analysis and design and the unified process,* (2nd Edn), Prentice Hall, Upper Saddle River, NJ, USA.

Open University M880/Unit RS (2002) *M880 Software Engineering : Requirements Specification,* (3rd Edn), Open University, Milton Keynes, England.

Pressman, R. S. (2005) *Software engineering : a practitioner's approach,* (6th Edn), McGraw-Hill, Boston, MA, USA.

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

Tidwell, J. (2006) *Designing interfaces,* O'Reilly, Sebastopol, CA, USA.

Yu, C. (2004) *Collaborative Tool : Requirements Elicitation and Analysis*. Available from: http://storm.prohosting.com/charyiu/Prototype/4191.htm [Accessed 06.05.2006].