УДК 004.4

А. Г. Майоров

СОЗДАНИЕ МНОГОПОТОЧНОГО ПРИЛОЖЕНИЯ ДЛЯ АНАЛИЗА И ТЕСТИРОВАНИЯ КОНСИСТЕНТНОСТИ ДАННЫХ, ХРАНИМЫХ В РЕЛЯЦИОННЫХ БАЗАХ ДАННЫХ

В статье описан процесс разработки и реализации библиотеки на языке Java для проверки консистентности записей в реляционных базах данных. Разработанное приложение является библиотекой, подключаемой при помощи одного из существующих пакетных менеджеров, используемых в языке Java. API, предоставляемое библиотекой, позволяет создавать и автоматизировать запуск тестов, проверяющих консистентности данных, а в случае обнаружения несоответствия приложение позволяет получить детальный отчет о несоответствии хранимых данных.

Любое крупное промышленное приложение использует несколько источников данных, таких как облачные хранилища, кэши и базы данных, но основным источником данных, зачастую является база данных. Нередко она состоит из десятка или даже сотни таблиц, каждая из которых может содержать в себе огромное количество данных [1].

В процессе разработки любое приложение проходит несколько стадий тестирования, каждая из которых требует своего собственного независимого рабочего окружения. Каждое окружение должно иметь свое собственное хранилище данных (рисунок 1), которое будет использовано только тем приложением, которое запущено в этом окружении. Структура таблиц и связей всех источников данных должна быть идентичной для правильного функционирования приложения. Также таблицы каждого из источников данных должны быть заполнены валидными данными для полноценного тестирования.

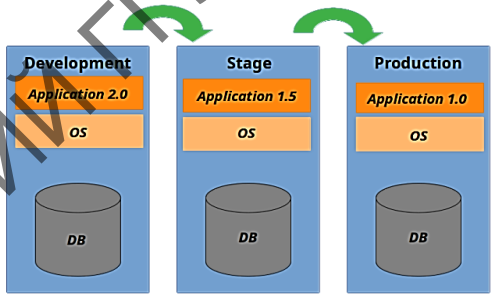

Рисунок 1 – Представление тестовых окружений при разработке приложений

Зачастую каждое хранилище данных имеет несколько ключевых таблиц, некорректные данные в которых могут сильно повлиять на функционирование всего приложения, а иногда и вовсе не дать ему быть запущенным. В процессе тестирования приложения в любую из таких таблиц могут попасть некорректные данные, по разным причинам: сбой системы, использование заведомо невалидных данных с целью тестирования, мануальное вмешательство в структуру таблицы или устаревшие данные требующие удаления. Во избежание ситуаций, когда невалидные данные препятствуют полноценному функционированию или запуску приложения, было разработано приложение, которое позволяет в автоматическом режиме выполнять запуск заранее подготовленного набора проверок для ключевых таблиц.

Каждая проверка представляет из себя запрос к определенной таблице, который проверяет соответствие хранимых в ней данных заданным критериям.

Во избежание проблем с производительностью при использовании данной библиотеки для таблиц с большими объемами данных было решено воспользоваться возможностями многопоточной среды исполнения языка Java. Многопоточность позволяет разделить таблицу на равные части, которые будут обрабатываться параллельно каждым потоком (рисунок 2), что позволит в разы ускорить процесс проверки. При обнаружении одним из потоков несоответствия в проверяемом участке данных информация будет сохранена в локальный отчет, и проверка продолжится. По окончании тестирования всех наборов данных информация об ошибках из локальных отчетов будет консолидирована в один детализированный отчет, который будет предоставлен пользователю.

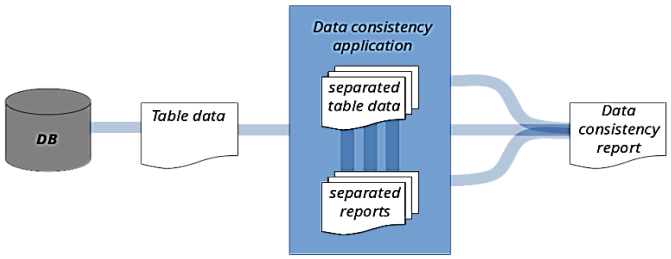

Рисунок 2 – Процесс обработки данных в приложении

При разработке системы также требовалось решить проблему использования библиотеки с разными видами СУБД на стороне пользователя. В современных промышленных приложениях принято использовать одну из нескольких общепринятых СУБД, таких как PostgreSQL, MySQL, MariaDB, Oracle. Каждая из них поддерживает основной синтаксис SQL, поэтому большинство сценариев, написанных для одной СУБД, могут быть без труда запущены для другой, но на уровне работы приложения, способ и алгоритм взаимодействия с каждой из них требует своего собственного уникального подхода.

Для решения проблемы было принято использовать стандартный инструмент JDBC [2], встроенный в язык программирования Java. Это позволит добиться легковесности библиотеки, а также применить унифицированный подход в работе систочниками данных на уровне кода.

Преимущества использования инструмента JDBC состоят в следующем. Легковесность, которая позволит избежать проблемы дополнительных зависимостей, поставляемых с библиотекой, а также любых проблем с конфликтом версий уже существующих библиотек в проекте, к которому подключили разработанную систему. Унификация доступа обеспечивается тем, что встроенный механизм JDBC предоставляет единый интерфейс взаимодействия с базами данных. Для работы с определенной базой требуется предоставить драйвер базы данных. Поэтому код библиотеки может быть написан лишь раз, а JDBC драйвер позволит запускать его на любой базе данных. Унифицированный подход, достигнутый при помощи использования JDBC драйвера, при работе с базами данных также позволяет упростить процесс первоначальной настройки и уменьшить время взаимодействия пользователя с библиотекой. Использование разработанного инструмента подразумевает, что у конечного пользователя уже есть нужный драйвер для работы с конкретной базой данных, заранее настроенный для работы основного приложения. Исходя из этого, было решено при старте тестирования сканировать рабочую директорию проекта для поиска уже настроенного драйвера и дальнейшего его использования. Использование уже существующего драйвера в проекте, также позволит избежать переконфигурации библиотеки при смене типа источника данных в основном приложении.

Для сборки и запуска современных промышленных приложений используются системы сборки. В языке Java существуют несколько распространенных систем [3], таких как Maven, Gradle, Ant. Каждая из них является менеджером зависимостей, в котором указываются версии используемых библиотек. Любая из систем сборки предоставляет возможность настройки процессов компиляции и запуска приложения, а также подключения модульных библиотек, которые позволяют кастомизировать существующие этапы сборки или добавлять абсолютные новые этапы (рисунок 3). Каждая из систем сборки уже имеет такой этап в процессе сборки, как тестирование. На этом этапе система сканирует приложение в поиске тестов и запускает их. Было решено использовать похожую концепцию при разработке системы тестирования консистентности данных.

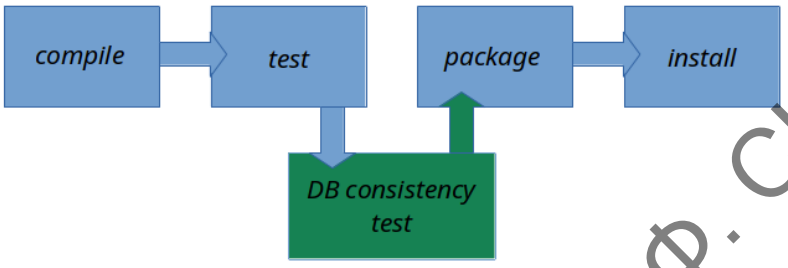

Рисунок 3 – Описание этапов сборки проекта

Разработанное приложение является подключаемым модулем к любой из систем сборки. При подключении к проекту появляется новый этап в процессе сборки. Данный этап сканирует определенную директорию в поиске заранее написанных тестов консистентности, которые являются файлами типа SQL, и запускает их в параллельном режиме, используя уже существующий в проекте драйвер для работы с базой данных. По завершении процесса тестирования, генерируется специальный HTML файл, содержащий в себе статистику выполненных и неудавшихся тестов в текстовом и графическом представлении.

Литература

1 Васильев, А. Н. Java: объектно-ориентированное программирование / А. Н. Васильев. – Санкт-Петербург : Питер, 2010. – 400 с.

2 Хорстман, К. Java: тонкости программирования: Том 2 / К. Хорстман. – Москва : Изд. дом «Вильямс», 2010. – 530 с.

3 Шилдт, Г. Полный справочник по Java / Г. Шилдт. – 6-е изд. – Санкт-Петербург : Вильямс, 2007. – 1040 с.

УДК 004.774.6:339

И. С. Мамичев

РАЗРАБОТКА МАРКЕТ-ПЛОЩАДКИ ДЛЯ ОРГАНИЗАЦИИ РАБОТЫ С ТОВАРАМИ

Статья посвящена созданию web-приложения по работе с товарами. Оно позволяет осуществлять покупку, продажу товаров и обмениваться ими. Разработанный программный продукт повышает эффективность работы торговой компании и качество оказания услуг, сокращает затраты на бумажные работы, предполагает