ОПИСАНИЕ АРХИТЕКТУРЫ РАЗРАБОТКИ ИНТЕГРАЦИОННОГО ФУНКЦИОНАЛА АВТОМАТИЧЕСКОГО ТЕСТИРОВАНИЯ БАЗ ДАННЫХ ORACLE

Попова М.А.
НИ ТПУ, ИШИТР, 8ПМ1И, email: map37@tpu.ru

Введение

Допущенная ошибка во время разработки программного обеспечения, ведет к появлению так называемого дефекта или бага. Когда речь идет о крупномасштабной разработке проекта с разнообразным функционалом, во избежание грубых ошибок перед релизом разного рода проектов на основную базу данных, необходимо провести массовое и модульное тестирование.

Крупным компаниям необходимо большую часть своего функционала покрывать автоматическим тестированием, чтобы повысить эффективность разработки программного обеспечения, избавиться от тривиальных кейсов, а также исключить возможность ошибки, связанной с человеческим фактором.

Целью работы является разработка гибкой методологии автоматического тестирования, предназначенной для стандартизации подхода в покрытии автоматическими тестами максимального количества возможных функционалов на языке PL/SQL на больших объемах данных без переписывания функционала, а лишь с созданием нового автотеста в интегрированной среде, которая будет основана на едином корпоративном стандарте.

Описание алгоритма

Фреймворк utPLSQL используется для модульного тестирования для Oracle PL/SQL. Утилита позволяет автоматически тестировать такие объекты, как пакеты, функции, процедуры, триггеры, представления и все, что можно выполнить и наблюдать из PL/SQL.

Для конфигурации теста используется определенная аннотация, которая размещается непосредственно перед процедурой. Например, для инициализации пакета, как набора тестов, используется обязательная аннотация: *--%suite(<description>)*. А для инициализации теста *--%test(<test name>)*.

После создания тестов, они вызываются в анонимном блоке типа:

```
begin
  ut.run('test_package_name');
end;
```

Внутри пакета *test_package_name* также должны прописаны процедуры проверки фактических результатов с ожидаемыми, которые осуществляются посредством вызова метода *expect*:

```
begin
    ut.expect(procedure_name('param1, param2)).to_equal('expected_result');
end;
```

На данный момент автотестирование на предприятии представляет собой больше модульное тестирование руками, которое зачастую не удовлетворяет требованиям. Связи с этим, встает необходимость разработки нового функционала автоматического тестирования с интеграцией на платформу TestIt и возможностью гибкого использования других функционалов компании с наименьшими трудозатратами на разработку новых тестов.

Простроенная архитектура продемонстрирована на рисунке 1, идея которой заключается в корректировке структуры таблиц с автотестами и их параметрами в более удобный формат для дальнейшей интеграции их в динамику, а также в разработке пакета на языке PL/SQL для вызова процедур – автотестов при помощи утилиты utPLSQL в динамике, что позволит сделать систему гибкой и использовать не только для определенного функционала.

На рисунке видна интеграция с платформой TestIt по средствам запросов XML/JSON к базе Oracle к справочнику, в котором будут описаны автотесты. По представленным параметрам для теста, совершается вызов процедуры расчета определенного функционала, связанного с данным автотестом

из справочника в пакете для запуска, тем самым обеспечивая универсальную обработку данных с одной точкой входа и по одному стандарту для разного функционала.

Для более ясной картины стоит привести конкретный пример возможного использования данного функционала. Представим есть некая подписка, за которую ежемесячно взимается некоторая сумма, которая зависит от определенных условий и параметров. Подписку можно соответственно подключить, отключить, а также пролонгировать. Есть база данных, на которой мы проводим массовое тестирование количества договором для пролонгации подписок сегодняшним днем, а также суммы для списания. Сравнить полученное количество пролонгирующих договоров до наката доработки и после – недостаточно. Нам необходимо рассматривать все договора из выборки по отдельности и сравнивать с ожидаемым результатом: прошла пролонгация или нет, а также какая сумма представлена для списания у клиента. Вручную такое сделать нереально, поэтому мы покрываем это автоматическим тестированием, проверяя каждый договор в отдельности, используя методы утилиты utPLSQL.

Таким образом мы вырабатываем общий подход, который сможет использовать данную утилиту utPLSQL и пакет параметризации, чтобы тесты разрабатывать было удобно, быстро и на любых количествах данных.

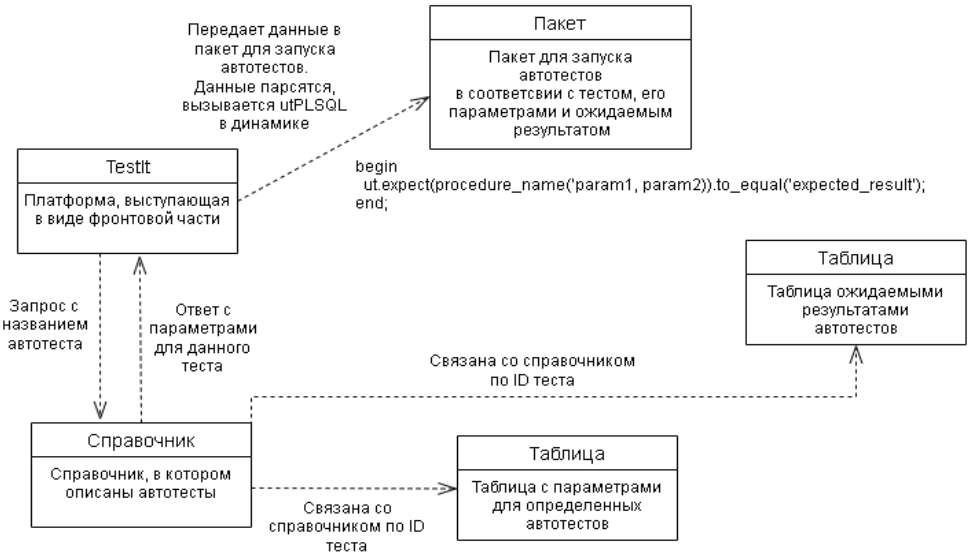

Рис. 1. Схема взаимодействия объектов автотестов в БД

Пример заполнения параметрами со стороны платформы TestIt представлен на рисунке 2. В демонстрации показано как выглядит создание автотеста в соответствии с возможными тестами из справочника. Далее прописав параметры, в данном случае, как пример приведен «Номер договора», обязательно наименование базы, на которой будет запущен автотест, а также «Необходимый параметр».

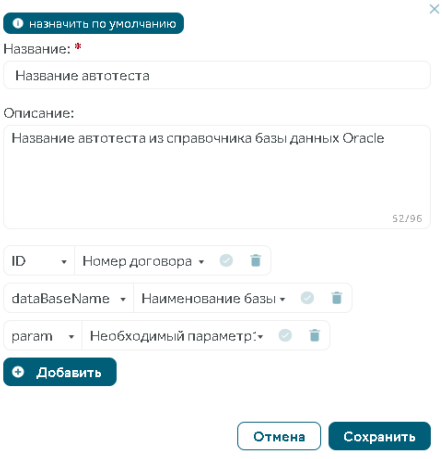

Рис. 2. Заполнение параметров для автотеста на TestIt

Заключение

На данном этапе разработки интеграционного функционала автоматического тестирования баз данных Oracle, построена архитектура, которая отличается от имеющегося метода, являющаяся гибким методом в покрытии тестами разного функционала лишь с использованием одного шаблона на входе в программу. Начата разработка пакета PL/SQL, в котором динамически будут обрабатываться процедуры с тестами, а также интеграция с TestIt для удобства использования.

Список использованных источников

1. Фейерштейн С. Oracle PL-SQL для профессионалов, 2015.
2. Документация utPLSQL [Электронный ресурс]. – URL: https://www.utplsql.org/index.html (дата обращения 22.02.2023).
3. Swagger API для платформы TestIt [Электронный ресурс]. – URL: https://assertible.com/blog/testing-an-api-using-swagger (дата обращения 22.02.2023).