TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Школа Инженерная школа информационных технологий и робототехники
Направление подготовки 09.04.04 Программная инженерия
ООП/ОПОП Технологии больших данных
Отделение школы (НОЦ) Информационных технологий

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРАНТА

Тема работы
Разработка интеграционного функционала для автоматического тестирования баз данных Oracle

УДК 004.415.53:004.43

Обучающийся

Группа	ФИО	Подпись	Дата
8ПМ1И	Попова Мария Александровна		10.06.2023 г.

Руководитель ВКР

Должность	ФИО	Ученая степень, звание	Подпись	Дата
доцент ОИТ ИШИТР	Губин Евгений Иванович	к.ф.-м.н.		10.06.2023 г.

КОНСУЛЬТАНТЫ ПО РАЗДЕЛАМ:

По разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
доцент ОСГН ШБИП ТПУ	Спицына Любовь Юрьевна	к.э.н.		

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
доцент	Антоневич Ольга Алексеевна	к.б.		

ДОПУСТИТЬ К ЗАЩИТЕ:

Руководитель ООП, должность	ФИО	Ученая степень, звание	Подпись	Дата
доцент ОИТ ИШИТР	Губин Евгений Иванович	к.ф.-м.н.		

Томск – 2023 г.

ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОСВОЕНИЯ ООП

по направлению 09.04.04 «Программная инженерия»

Код компетенции	Наименование компетенции
Универсальные компетенции	
УК(У)-1	Способен осуществлять критический анализ проблемных ситуаций на основе системного подхода, вырабатывать стратегию действий
УК(У)-2	Способен управлять проектом на всех этапах его жизненного цикла
УК(У)-3	Способен организовывать и руководить работой команды, вырабатывая командную стратегию для достижения поставленной цели
УК(У)-4	Способен применять современные коммуникативные технологии, в том числе на иностранном (-ых) языке (-ах), для академического и профессионального взаимодействия
УК(У)-5	Способен анализировать и учитывать разнообразие культур в процессе межкультурного взаимодействия
УК(У)-6	Способен определять и реализовывать приоритеты собственной деятельности и способы ее совершенствования на основе самооценки
Общепрофессиональные компетенции	
ОПК(У)-1	Способен самостоятельно приобретать, развивать и применять математические, естественно-научные, социально-экономические и профессиональные знания для решения нестандартных задач, в том числе в новой или незнакомой среде и в междисциплинарном контексте
ОПК(У)-2	Способен разрабатывать оригинальные алгоритмы и программные средства, в том числе с использованием современных интеллектуальных технологий, для решения профессиональных задач
ОПК(У)-3	Способен анализировать профессиональную информацию, выделять в ней главное, структурировать, оформлять и представлять в виде аналитических обзоров с обоснованными выводами и рекомендациями
ОПК(У)-4	Способен применять на практике новые научные принципы и методы исследований
ОПК(У)-5	Способен разрабатывать и модернизировать программное и аппаратное обеспечение информационных и автоматизированных систем
ОПК(У)-6	Способен самостоятельно приобретать с помощью информационных технологий и использовать в практической деятельности новые знания и умения, в том числе в новых областях знаний, непосредственно не связанных со сферой деятельности
ОПК(У)-7	Способен применять при решении профессиональных задач методы и средства получения, хранения, переработки и трансляции информации посредством современных компьютерных технологий, в том числе, в глобальных компьютерных сетях
ОПК(У)-8	Способен осуществлять эффективное управление разработкой программных средств и проектов

Код компетенции	Наименование компетенции
Профессиональные компетенции	
ПК(У)-1	Способен к созданию вариантов архитектуры программного средства
ПК(У)-2	Способен разрабатывать и администрировать системы управления базам данных
ПК(У)-3	Способен управлять процессами и проектами по созданию (модификации) информационных ресурсов
ПК(У)-4	Способен проектировать и организовывать учебный процесс по образовательным программам с использованием современных образовательных технологий
ПК(У)-5	Способен осуществлять руководство разработкой комплексных проектов на всех стадиях и этапах выполнения работ

TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Школа Инженерная школа информационных технологий и робототехники
Направление подготовки 09.04.04 Программная инженерия
ООП/ОПОП Технологии больших данных
Отделение школы (НОЦ) Информационных технологий

УТВЕРЖДАЮ:
Руководитель ООП
_____ _____ Губин Е. И.
(подпись) (дата) (Ф.И.О.)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

Обучающийся:

Группа	ФИО
8ПМ1И	Попова Мария Александровна

Тема работы:

Разработка интеграционного функционала для автоматического тестирования баз данных Oracle	
Утверждена приказом директора (дата, номер)	№ 146-39/с от 26.05.2023 г.

Срок сдачи обучающимся выполненной работы:	10.06.2023 г.

ТЕХНИЧЕСКОЕ ЗАДАНИЕ:

Исходные данные к работе	1. База данных Oracle 2. Система интеграции TestIt 3. Техническое задание

Перечень разделов пояснительной записки, подлежащих исследованию, проектированию и разработке	1. Анализ существующих подходов к разработке автоматических тестов баз данных Oracle 2. Изучение интеграционных сервисов, позволяющие настроить запуск тестов и обеспечивающие обработку больших массивов данных в автоматическом режиме 3. Разработка единого стандарта создания автоматических тестов для любого количества входных данных и параметров запуска 4. Создание прикладного функционала на стороне базы данных Oracle с интеграцией с сервисом TestIt 5. Работа над разделом по финансовому менеджменту, ресурсоэффективности и ресурсосбережения. 6. Работа над разделом по социальной ответственности.
Перечень графического материала *(с точным указанием обязательных чертежей)*	1. ER – диаграмма созданных объектов хранения базы данных 2. Схема JSON – файла 3. Модель ввода входных параметров в интеграционной среде TestIt 4. Диаграмма Ганта

Консультанты по разделам выпускной квалификационной работы
(с указанием разделов)

Раздел	Консультант
Основная часть	доцент ОИТ ИШИТР, к.ф.-м.н., доцент Губин Е.И.
Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	доцент ОСГН ШБИП, к.э.н., доцент Спицына Л. Ю.
Социальная ответственность	доцент ООД ШБИП, к.б.н., доцент Антоневич О. А.
Раздел на английском языке	доцент, ОИЯ ШБИП к.ф.н, Уткина К. Н.

Названия разделов, которые должны быть написаны на иностранном языке:

Аналитический обзор научной, нормативной и технической документации

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику	1.03.2023 г.

Задание выдал руководитель ВКР:

Должность	ФИО	Ученая степень, звание	Подпись	Дата

доцент ОИТ ИШИТР	Губин Евгений Иванович	к.ф.-м.н., доцент		1.03.2023 г.

Задание принял к исполнению обучающийся:

Группа	ФИО	Подпись	Дата
8ПМ1И	Попова Мария Александровна		1.03.2023 г.

TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Инженерная школа Информационных технологий и робототехники

Направление подготовки (ООП / ОПОП) 09.04.04 Программная инженерия
Уровень образования магистратура
Отделение школы (НОЦ) Информационных технологий
Период выполнения весенний семестр 2022 /2023 учебного года

КАЛЕНДАРНЫЙ РЕЙТИНГ-ПЛАН
выполнения выпускной квалификационной работы

Обучающийся:

Группа	ФИО
8ПМ1И	Попова Мария Александровна

Тема работы:

Разработка интеграционного функционала для автоматического тестирования баз данных Oracle	
Срок сдачи обучающимся выполненной работы:	10.06.2023 г.

Дата контроля	Название раздела (модуля) / вид работы (исследования)	Максимальный балл раздела (модуля)
10.06.2023	Основная часть	70
10.06.2023	Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	10
10.06.2023	Социальная ответственность	10
10.06.2023	Раздел на английском языке	10

СОСТАВИЛ:
руководитель ВКР

Должность	ФИО	Ученая степень, звание	Подпись	Дата
доцент ОИТ ИШИТР	Губин Евгений Иванович	к.ф.-м.н.		

СОГЛАСОВАНО:
руководитель ООП

Должность	ФИО	Ученая степень, звание	Подпись	Дата
доцент ОИТ ИШИТР	Губин Евгений Иванович	к.ф.-м.н.		

Задание принял к исполнению обучающийся:

Группа	ФИО	Подпись	Дата
8ПМ1И	Попова Мария Александровна		1.03.2023 г.

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ»

Студенту:

Группа	ФИО		
8ПМ1И	Поповой Марии Александровне		
Школа	ИШИТР	**Отделение школы (НОЦ)**	09.04.04 Программная инженерия
Уровень образования	Магистратура	**Направление/специальность**	Технологии больших данных

Исходные данные к разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»:	
1. *Стоимость ресурсов научного исследования (НИ): материально-технических, энергетических, финансовых, информационных и человеческих*	*Бюджет проекта – 403 884 руб., в т.ч. затраты по оплате труда – 339 822 руб. за 6 месяцев для разработчика и 63 516 для научного руководителя, распечатка ВКР 450.*
2. *Нормы и нормативы расходования ресурсов*	*Накладные расходы - 9,92%* *Районный коэффициент 30%;*
3. *Используемая система налогообложения, ставки налогов, отчислений, дисконтирования и кредитования*	*Отчисления на уплату во внебюджетные фонды 30,2%*
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. *Оценка коммерческого и инновационного потенциала НТИ*	*Анализ улучшения производительности производства, оценка качества и перспективности проекта*
2. *Разработка устава научно-технического проекта*	*Определения заинтересованных сторон проекта, выявление целей и результата*
3. *Планирование процесса управления НТИ: структура и график проведения, бюджет, риски и организация закупок*	*Определение трудоемкости на разработку архитектуры проекта, выполнения работ разработки, расчет бюджета проекта*
4. *Определение ресурсной, финансовой, экономической эффективности*	*Определение эффективности разработки по отношению экономической составляющей компании*
Перечень графического материала *(с точным указанием обязательных чертежей):*	
1. *«Портрет» потребителя результатов НТИ* 2. *Сегментирование рынка* 3. *QuaD - анализ* 4. *Оценка конкурентоспособности технических решений* 5. *Матрица SWOT* 6. *Оценка ресурсной, финансовой и экономической эффективности НТИ* 7. *Потенциальные риски*	
Дата выдачи задания для раздела по линейному графику	01.03.2023

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
доцент ОСГН ШБИП ТПУ	Спицына Любовь Юрьевна	к.э.н.		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8ПМ1И	Попова Мария Александровна		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Студенту:

Группа		**ФИО**	
		Попова Мария Александровна	
Школа	**ИШИТР**	**Отделение (НОЦ)**	**ОИТ**
Уровень образования	магистратура	**Направление/ специальность**	*09.04.04 Программная инженерия*

Тема ВКР:

Разработка интеграционного функционала автоматического тестирования баз данных Oracle	
Исходные данные к разделу «Социальная ответственность»:	
Введение – Характеристика объекта исследования (вещество, материал, прибор, алгоритм, методика) и области его применения. – Описание рабочей зоны (рабочего места) при разработке проектного решения/при эксплуатации	*Объект исследования:* база данных Oracle *Область применения:* разработка ПО для работы с большими данными *Рабочая зона:* офис *Количество и наименование оборудования рабочей зоны*: персональный компьютер *Рабочие процессы, связанные с объектом исследования, осуществляющиеся в рабочей зоне:* разработка гибкой методологии автоматического тестирования баз данных Oracle с интеграцией с платформой для тестирования Test IT.
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Правовые и организационные вопросы обеспечения безопасности <u>при разработке проектного решения</u>: – специальные (характерные при эксплуатации объекта исследования, проектируемой рабочей зоны) правовые нормы трудового законодательства; – организационные мероприятия при компоновке рабочей зоны.	ГОСТ 12.2.032-78 Система стандартов безопасности труда. Рабочее место при выполнении работ сидя; Трудовой кодекс Российской Федерации от 30.12.2001 N 197-ФЗ; Федеральный закон от 28.12.2013 г. N 426-ФЗ "О специальной оценке условий труда"; ГОСТ-21889-76 Система «Человек-машина»

2. Производственная безопасность <u>при разработке проектного решения</u>*:* – Анализ выявленных вредных и опасных производственных факторов – Расчет уровня опасного или вредного производственного фактора	Вредные факторы: Отсутствие или недостаток необходимого естественного или искусственного освещения; Нервно – психические перегрузки, связанные с умственным перенапряжением; Физические статистическое перегрузки, связанные с рабочей позой; Отклонение параметров микроклимата; Повышенный уровень и другие неблагоприятные характеристики шума; Опасные факторы: Повышение напряжения в электрической цепи; Расчет: Методика расчета системы общего равномерного искусственного освещения
3. Экологическая безопасность <u>при разработке проектного решения</u>	Воздействие на литосферу: утилизация ПК и люминесцентных ламп
4. Безопасность в чрезвычайных ситуациях <u>при разработке проектного решения</u>	Возможные ЧС: Пожар; Неисправность систем водоснабжения; Увеличенная концентрация опасных веществ в воде; Наиболее вероятные: Пожар
Дата выдачи задания для раздела по линейному графику	**01.03.2023**

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
доцент	Антоневич Ольга Алексеевна	к.б.		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8ПМ1И	Попова Мария Александровна		

РЕФЕРАТ

Выпускная квалификационная работа 79 с., 11 рис., 17 табл., 22 источников, 2 прил.

Ключевые слова: база данных, фреймворк, Oracle, автоматизация тестирования, интеграция.

Объектом исследования является процесс разработки функционала, единого стандарта, позволяющего разрабатывать автоматические тесты под любой существующий или разрабатываемый функционал, на любом количестве входных данных и параметров запуска на стороне базы данных Oracle.

Цель работы - разработка гибкой методологии автоматического тестирования, предназначенной для стандартизации подхода в покрытии автоматическими тестами максимального количества возможных функционалов на языке Oracle PL/SQL при работе с большими массивами данных в интеграционной средой TestIt.

В процессе исследования был проведен анализ существующих подходов к разработке автоматических тестов и изучены интеграционные среды, позволяющих настроить запуск теста.

В результате исследования была спроектирована архитектура системы интеграции базы данных Oracle с платформой TestIt, разработан пакет параметризации на языке Oracle PL/SQL.

Степень внедрения: данная работа проводилась в рамках компании «Совкомбанк Технологии», разработка внедрена в бизнес-процессы компании.

Экономическая эффективность/значимость работы: положительное влияние на качество разрабатываемых сервисов и поддержание стабильности работы системы.

Оглавление

Введение 14
1. Виды тестирования 16
1.1. Возможные проблемы при тестировании больших данных 17
1.2. Существующие подходы тестирования в базе данных Oracle 18
2. Использование внешней системы TestIt для процессов тестирования 19
3. Описание гибкости подхода разработанной системы 20
4. Архитектура взаимодействия между объектами базы 22
5. Структура базы данных и пакеты обработки 23
5.1. Таблица параметров метода 23
5.2. Таблица хранения автотестов и их процедур обработки 24
5.3. Таблица хранения результатов обработки 25
5.4. Таблица журналирования 26
5.5. Пакеты обработки данных 27
6. Интеграция с платформой TestIt 28
7. Исследование эффективности разработанной системы по сравнению с исходными методами тестирования 29
8. Обсуждение результатов 30
9. Финансовый менеджмент, ресурсоэффективность ресурсосбережение 32
9.1. Анализ потенциальных потребителей и конкурентов 32
9.2. Технология QuaD 33
9.3. Матрица SWOT 34
9.4. Цели и требования проекта 38
9.5. Организация структуры проекта 39
9.6. Ограничения 40
9.7. Планирование проекта 41
9.8. Финансирование проекта 46
9.8.1. Стоимость оборудования 46
9.8.2. Базовая заработная плата 47
9.8.3. Расходы научного руководителя 49
9.8.4. Отчисления на социальные нужды 50
9.9. Оценка готовности разработки к коммерциализации 50
10.10. Разработка экономической части 50
10.10.1. Первичный анализ проекта 50

10.10.2. Экономическая эффективность 51
11. Социальная ответственность 53
11.1. Правовые и организационные вопросы обеспечения безопасности 53
11.2. Организационные мероприятия при компоновке рабочей зоны 54
11.3. Производственная безопасность при разработке проектного решения 54
11.3.1. Отсутствие или недостаток необходимого естественного или искусственного освещения 55
11.3.2. Нервно – психические перегрузки, связанные с умственным перенапряжением .. 58
11.3.3. Физические статистическое перегрузки, связанные с рабочей позой 58
11.3.4. Отклонение параметров микроклимата 59
11.3.5. Повышенный уровень и другие неблагоприятные характеристики шума 60
11.3.6. Повышение напряжения в электрической цепи 60
11.4. Экологическая безопасность 60
11.5. Безопасность в чрезвычайных ситуациях 61
11.5.1. Анализ возможных ЧС, которые могут возникнуть при разработке в офисе 61
Заключение по разделу «Социальная ответственность» 63
Список используемых источников 64
Приложение А 66
Приложение Б 77

Введение

В силу роста популярности цифровых технологий, все процессы работы с данными перешли в цифровой формат, из-за чего их количество быстро и резко увеличилось за последние годы. В связи с этим объем поступающих исходных данных резко возрастает [1]. Когда речь идет о функционировании банковских систем, то это особенно заметно. Лавинообразный рост объемов данных, обрабатываемых на регулярной основе, предъявляет повышенные требования к скорости и качеству работы с ними. Необходимо отметить, что ежедневно банками обрабатываются данные миллионов клиентов. Вся информация хранится в базе данных и регулярно используется в обработке логики функционалов и продуктов компании.

Традиционные методы обработки данных становятся не актуальными, так как возрастает запрос по скорости и достоверности их обработки, а также в быстром и качественном тестировании для поддержания стабильности работы компании и бизнес процессов, требуя регулярного тестирования всех функциональных модулей на большом количестве разнообразных данных, что вызывает потребность из-за регулярных доработок и непрерывном развитии клиентских сервисов.

Допущенная ошибка во время разработки программного обеспечения, ведет к появлению, так называемого дефекта или бага. Внедрив некачественный продукт на клиентский сервер, банк может понести как финансовые, так и репутационные риски и потери.

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.

Для поддержания стабильности работы и непрерывности бизнес процессов компании, 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.

Обеспечить регулярное ручное тестирование на больших объемах данных силами команды тестирования не представляется возможным без разработки автоматических тестов и единого стандарта компании по их созданию.

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 массивах данных без создания интеграционного функционала с платформой TestIt под каждый тест-кейс, а с созданием только логики нового автоматического теста на стороне базы данных Oracle. Разработка предназначена для настройки и запуска автоматического кейса для команды тестирования в интерфейсе сервиса TestIt.

1. Виды тестирования

Автоматизация тестирования значительно позволяет снизить расходы команды разработчиков, сэкономить время и ресурсы, которые затрачиваются на разработку новых тест-кейсов, а также снизить выпуск на рынок некачественного продукта. В силу этого, технологии автоматизации тестирования набирают большую популярность среди компаний, связанных с разработкой программного обеспечения выпускаемого продукта.

В процессе тестирования исследуется программа на предмет поиска возможных ошибок, следствием которой становится несоответствие ожидаемых результатов от действительных в работе программного продукта.

Тестирование больших данных включает в себя использование различные инструменты, методы и структуры для обработки. Большие данные относятся к хранению, анализу данных и тестированию, которые отличаются по объему, разнообразию и скорости.

Тестирование приложения больших данных – это проверка как обработки данных, так и проверка результатов обработки отдельных функционалов программного продукта.

Тестирование подразделяется на несколько видов:

1. Модульное тестирование – заключается в тестировании отдельных методов и функций классов, компонентов или модулей. Как правильно модульные тесты не требуют больших расходов на автоматизацию и могут выполняться с интеграцией с сервером очень быстро.

2. Интеграционное тестирование – поверяется, хорошо ли работают вместе разные модули и сервисы. Можно протестировать взаимодействие с базой данных или убедиться, что между собой микросервисы работают как задумано.

3. Функциональное тестирование – проверяется только исходный результат некоторого действия без промежуточного состояния системы.

4. Массовое тестирование подразумевает под собой тестирование больших объемов данных, множества модулей и взаимодействующих между собой функционалов.

5. Сквозное тестирование представляет собой проверку поведение пользователя при работе с ПО в контексте всего приложения. Позволяет обеспечить контроль на любые действия пользователя в приложении.

1.1. Возможные проблемы при тестировании больших данных

На практике тестирование осуществляется путем выполнения определенного набора действий или скриптов, получения результатов выполнения и дальнейшей сверки с данными, которые определены, как эталонные. Выполнение данного алгоритма возможно как вручную, так и в автоматическом режиме с использованием различных программных решений.

В данном разделе будет рассмотрела специфика сложности тестирования больших данных, а также их тестирование в мануальном режиме и положительные и отрицательные стороны перехода на автоматизацию тестирования.

Большой объем данных из различных источников требует:

- больше времени на тестирование совместимости с разными платформами;
- больше времени для быстрой автоматизации;
- больше времени на проверку и подтверждение результатов обработки данных.

При тестировании продукта, необходимо учитывать возможные ошибки, связанные с человеческим фактором, так как алгоритм последовательности действий в таком случае строится самим человеком. При разработке нового тест-кейса необходимо привлекать эксперта для написания скриптов.

Команда тестирования проверяет бизнес логику продукта на каждом узле выполнения программы. Причем проверка должна быть выполнена на каждом узле перед запуском следующего шага в тестировании одного лишь функционала. В конечном итоге, необходимо оценить полученный результат с ожидаемым.

При работе с массовым тестированием большого объема данных, то есть получения для каждого клиента его ожидаемый и фактический результат, невозможно обеспечить ручным тестированием без разработки автоматических тестов, а также единого стандарта компании.

У автоматического тестирования большое количество преимуществ, но и есть ряд недостатков. К сильным сторонам автоматизированного тестирования относятся:

- скорость выполнения, в сотки тысяч раз превосходящая человеческие возможности;
- отсутствие влияния человеческого фактора, сюда входят ошибки по невнимательности и усталость;
- возможность многократно выполнять тесты, снижая тем самым затраты;

- выполнение тест-кейсов особой сложности;
- способность анализировать и сравнивать данные в удобном формате, особенно актуально при массовом тестировании, так как вручную не представляется возможности ежедневно проверять отдельно для миллионов клиентов ожидаемый результат их выполнения и фактический.

К недостаткам автоматизированного тестирования относятся:

- необходимость привлекать разработку для создания универсальной системы тестирования;
- финансовые затраты и риски на средства автоматизации и сложность их выбора;
- устаревание тестов в случае изменений требований.

1.2. Существующие подходы тестирования в базе данных Oracle

В данном разделе будут рассмотрены используемые утилиты для тестирования баз данных Oracle, в том числе та, что использовалась в данной разработке – *utPlSql* [2].

- *Oracle Real Application Testing*

Опция Oracle Real Application Testing в базе данных Oracle помогает надежно гарантировать целостность изменений в базе данных и управлять тестовыми данными. Она позволяет выполнять тестирование в базе данных Oracle в реальном времени [3].

Анализатор производительности SQL и воспроизведение базы данных являются ключевыми компоненты Oracle Real Application Testing. В зависимости от характера и влияния тестируемого системного изменения, а также от типа системы, которую будет выполнять тест, для выполнения тестирования можно использовать один или оба компонента.

Системные изменения, такие как обновление базы данных или добавление индекса, могут вызвать изменения в планах выполнения операторов SQL.

- *Code Tester for Oracle*

Программа тестирования с самым высоким уровнем автоматизации тестов. Тестовый код генерируется на основании поведения, определяемого в пользовательском интерфейсе, после чего программа выполняет тесты и выводит результаты. Успех/неудача каждого теста отмечаются на экране зеленым/красным цветом.

- *utPLSQL*

utPLSQL реализует принципы тестирования, принятые в методологии экстремального программирования. Автоматически выполняет написанный вручную тестовый код с проверкой результатов [4].

Плюсы утилиты:

- схожесть с классическими фреймворками в других языках программирования
- обширный функционал
- возможность сравнивать курсоры с данными для работы с большим количество результирующих данных фактических и ожидаемых

2. Использование внешней системы TestIt для процессов тестирования

Гибкость разработки подразумевает под собой настройку запуска тестов специалистами, не владеющими навыками разработки. Для создания настройки на стороне базы данных Oracle была оценена и выбрана интеграционная среда TestIt – российская система управления тестированием [5].

При ручном тестирование создается тест-кейс и по проектам происходит структурирование и хранение всех тест-сценарием с возможностью объединения повторяющихся действий в общие шаги. Есть возможность отслеживания вносимых изменений и возвращения к предыдущей версии изменений.

При автоматическом тестировании создание автоматического теста также происходит в реальном времени. Преимуществом платформы является возможность интеграции автоматических тестов на разных фреймворках, неважно на каком языке программирования написан автоматический тест – TestIt объединит любые тесты из репозитория и запустит их. Но для этого необходимо прописать интеграцию с базой данных, создав при этом свой фреймворк, что и есть цель данной работы.

Разработанный интеграционный функционал обеспечивает гибкую настройку запуска автоматических тестов, разрабатываемых по внедренному стандарту, посредством передачи произвольного набора параметров запуска в базу данных в формате JSON, вид которого будет предоставлен в следующем разделе. Это позволяет создавать автоматические тесты без доработок интеграционного сервиса, используя ресурсы команды разработки, которая имеет необходимую квалификацию в рамках тестируемого функционала.

3. Описание гибкости подхода разработанной системы

При работе с большими данными в крупных компаниях с огромной клиенткой базой, до того, как отправить разработку на боевую базу к клиентам, необходимо тщательно протестировать не только нововведение, но и также смежный функционал, который мог быть затронут. В силу того, что это миллионы операций с миллионами количества строк обработки данных, то это занимает большое время, особенно, если это делать мануальным методом.

При тестировании таким способом до релиза на боевую базу необходимо также проверить абсолютно каждый кейс, каждого клиента, на предмет информации, которая по нему возвращается после доработки.

Простроенная архитектура [6], продемонстрирована на рисунке 1, идея которой заключается в создании структуры таблиц с автоматическими тестами и их параметрами, а также журналом для контроля вносимых изменений. Так, на схеме видно, что, начиная с платформы поступает запрос к базе данных к таблице с названием «*proc_param*» (описание таблиц будет приведен ниже), получая ответ, формирует JSON файл, его вид приведен на рисунке 2. Тут видно, что ключом является номер тест-кейса из базы данных, а значением вложенный словарь, состоящий из названия входного параметра для данного теста и его значением.

Листинг кода по созданию объектов базы данных приведен в приложении Б.

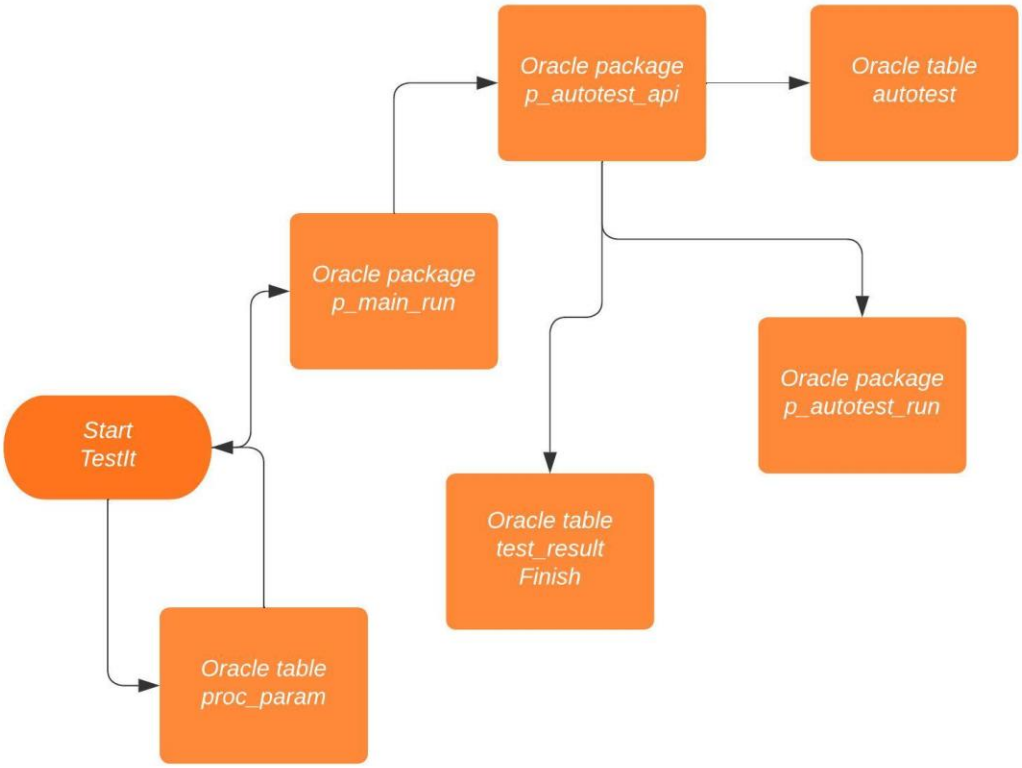


Рис. 1. Связь объектов автоматических тестов в базе данных Oracle

```
{id_test :
  {"param_1" : "value_1",
   "param_2" : "value_2"}
};
```

Рис. 2. Схема структуры JSON файла

После получения JSON файла на стороне базы данных Oracle [7] запускается основной пакет «*p_main_run*», в котором помимо процедур обработки ошибок находится процедура, принимающая на вход два параметра, а именно номер автоматического теста и есть входные параметры в JSON файла. Используя библиотеку JSON в Oracle идет перебор параметров и его значений в цикле. Соединяя в единый список, посредством динамического SQL происходит

вызов процедуры в связке с таблицей «*autotest*», где хранится номер теста и процедура, предназначенная для его запуска.

Динамический SQL это инструмент в области программирования приложений, позволяющий упростить взаимодействие приложения с базой данных путём создания SQL-запроса непосредственно из кода программы.

После отработки процедуры, связываясь с таблицей «*test_result*» происходит оценка рассчитанного значения с эталонным с расчетом на его погрешность, которая также определена в таблице. Сравнение данных построчно происходят с использованием утилиты utPlSql.

4. Архитектура взаимодействия между объектами базы

В данном разделе будет описана и представлена ER-диаграмма созданных таблиц хранения параметров, основной на информации об автоматических тестах и его результатами.

На рисунке 3 продемонстрирована ER-диаграмма из трех вышеупомянутых таблиц «*proc_param*», «*autotest*» и «*test_result*». К ним также добавлена таблица с журналированием, которая заполняется посредствам срабатывания триггера на таблицы «*autotest*» и «*proc_param*».

Таблица «*proc_param*» связана с таблицей «*autotest*» связью «один ко многим», так как для одного автоматического теста имеется неограниченное количество входных параметров.

Таблица «*autotest*» с «*test_result*» имеет также связь «один ко многим» в силу возможности выполнения одного теста несколько раз.

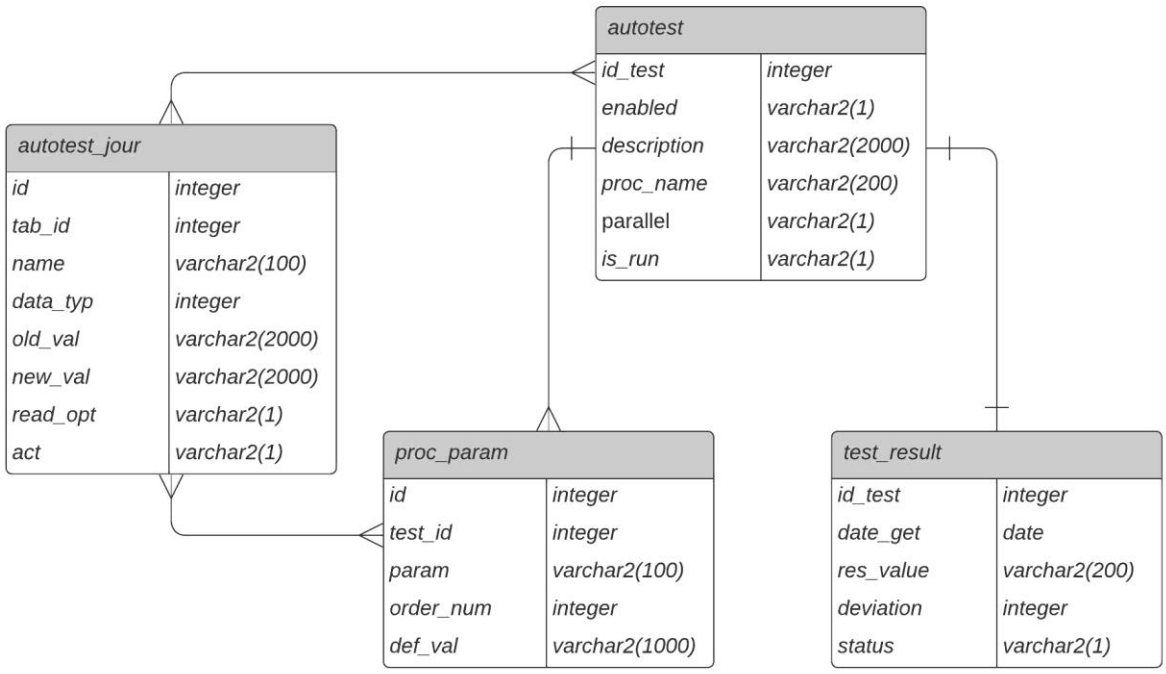


Рис. 3. ER – диаграмма объектов автотестирования

5. Структура базы данных и пакеты обработки

В данной части будет подробно рассмотрена структура таблиц, поля и их тип, а также пакеты с которыми взаимодействую таблицы.

5.1. Таблица параметров метода

Первая таблица, которая будет описана – таблица с которой начинается процесс с платформы TestIt – «*proc_param*» (рис. 4.). Данная таблица содержит в себе такие поля, как:

1. *id* – id записи, уникальное значение. Является первичным ключом для таблицы. Имеет целочисленный тип;

2. *test_id* – номер автоматического теста, а также процедуры для запуска тест-кейса с целочисленным типом;

3. *param* – название возможного параметра;

4. *order_num* – порядок заполнения данного параметра в процедуру обработки;

5. *def_val* – возможное дефолтное значение параметра, в случае, если оно не было передано от пользователя из TestIt.

proc_param	
id	*integer*
test_id	*integer*
param	*varchar2(100)*
order_num	*integer*
def_val	*varchar2(1000)*

Рис. 4. – Таблица параметров

5.2. Таблица хранения автотестов и их процедур обработки

В таблице «*autotest*» (рис. 5.) хранится основная информация об автотесте, а именно:

1. *id_test* – уникальное значение номера автотеста;
2. *enabled* – статус доступности теста для запуска;
3. *description* – краткое описание того, на что происходит проверка при отработке автотеста;
4. *proc_name* – название вызываемой процедуры с прописыванием пакета, в котором она находится;
5. *parallel* – возможность запуска теста параллельно с другими тестами;
6. *is_run* – статус работы теста в реальном времени.

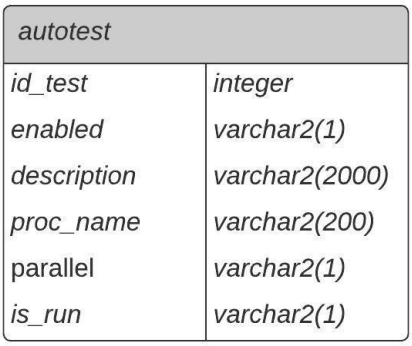


Рис. 5. Таблица автоматических тестов

5.3. Таблица хранения результатов обработки

Последней таблицей в списке хранения информации об автоматических тестах является таблица с названием «*test_result*» (рис. 6.), в которой хранится статус отработки автотеста в соответствии с полученным значением результата.

1. *id_test* – id автотеста;
2. *date_get* – дата его обработки;
3. *res_value* – значение полученного результата;
4. *deviation* – приемлемая погрешность результата в процентах;
5. *status* – статус корректности отработанного автотеста.

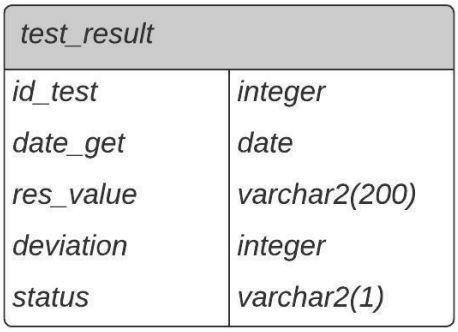


Рис. 6. Таблица результатов выполнения автоматических тестов

5.4. Таблица журналирования

Для контроля вносимых данных в таблицы с автоматическими тестами и вводимыми параметрами для их запуска, была создана таблица с журналированием (рис. 7), которая заполняется посредством созданного триггера с предикатами before insert or update or delete.

Структура таблицы выглядит следующим образом:

1. *id* – id записи в таблице;

2. tab_id - идентификатор таблицы, в которой были изменения;

3. *name* – имя колонки, в которой произошли изменения;

4. *data_typ* – тип данных столбца;

5. *old_val* – значение, которое стояло до внесения изменений;

6. *new_val* – новое значение, после внесения изменений;

7. *read_opt* – оптимизация для чтения;

8. *act* – статус активности/актуальности столбца.

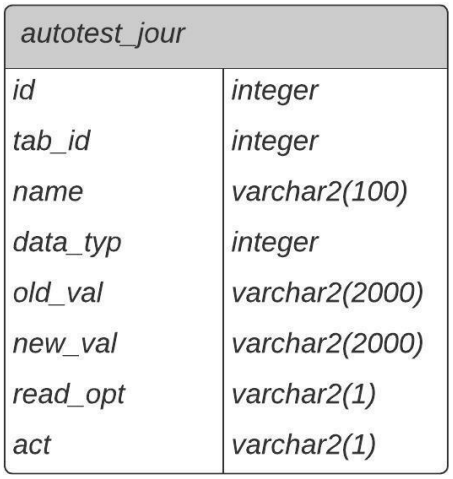


Рис. 7. Таблица журналирования

5.5. Пакеты обработки данных

Как уже было упомянуто выше, в разработке фигурирует три пакета, написанных на языке программирования Oracle PL/SQL, листинг которых приведен в приложении Б. Начиная с пакета «*p_main_run*» происходит извлечения из полученного JSON файла распределение параметров с использованием библиотеки JSON в Oracle.

Для передачи параметров и подстановки их в вызываемую процедуру, которая определена в таблице «*autotest*», используется динамический SQL.

Динамический SQL позволяет программе быть более гибкой, создавать и обрабатывать SQL предложения во время выполнения программ. Это немедленное выполнение динамического SQL – запроса или анонимного PL/SQL блока путем использования оператора *execute immediate*, основным аргументом которого является строка, содержащая SQL – запрос для выполнения. Большим преимуществом, которое используется в данном пакете, является возможность создания строки с использованием конкатенации.

Вспомогательными пакета для «*p_main_run*» являются «*p_autotest_api*» и «*p_autotest_run*». Во втором объявлено три вспомогательных функции и одна процедура.

1. *get_proc_name* – функция, возвращающая название процедуры из таблицы autotest по его id.

2. *get_parallel* вернет результат возможности запуска данного автотеста параллельно с другими автотестами.

3. *count_is_run* возвращает количество автотестов, которые находятся в статусе выполнения.

4. *set_status* – устанавливает статус запуска и остановки работы автотеста, Y и N соответственно.

Пакет «*p_autotest_run*», расположенный отдельно, предназначен для разработки каждого теста индивидуально. Именно этот пакет и нужно будет дорабатывать разработчику при создании своего автотеста, прописывая логику его работы.

Важным моментом является возможность запуска автоматического теста как для определенного кейса – например, по определенному договору, так и массовый запуск по всем договорам за определенную дату. Со стороны платформы TestIt это определяется путем

пропуска шага ввода параметров для автотеста или вводом только необходимой даты. А на стороне Oracle – это проверка и анализ поданного на вход в параметризованную процедуру JSON файла или вовсе его отсутствие.

6. Интеграция с платформой TestIt

В данном разделе будет продемонстрирована визуальная часть интеграции с платформой TestIt. В качестве примера была создана модель на не существующих данных в силу их конфиденциальности. На рисунке 8 приведен пример интеграции системы с базой данных Oracle. Параметры для ввода подтягиваются исходя из вышеупомянутой таблицы «*proc_param*». На рисунке видно название автоматического теста, которое можно задать локально на платформе, также добавить его описание. А сами же параметры предоставляются ниже для их заполнения.

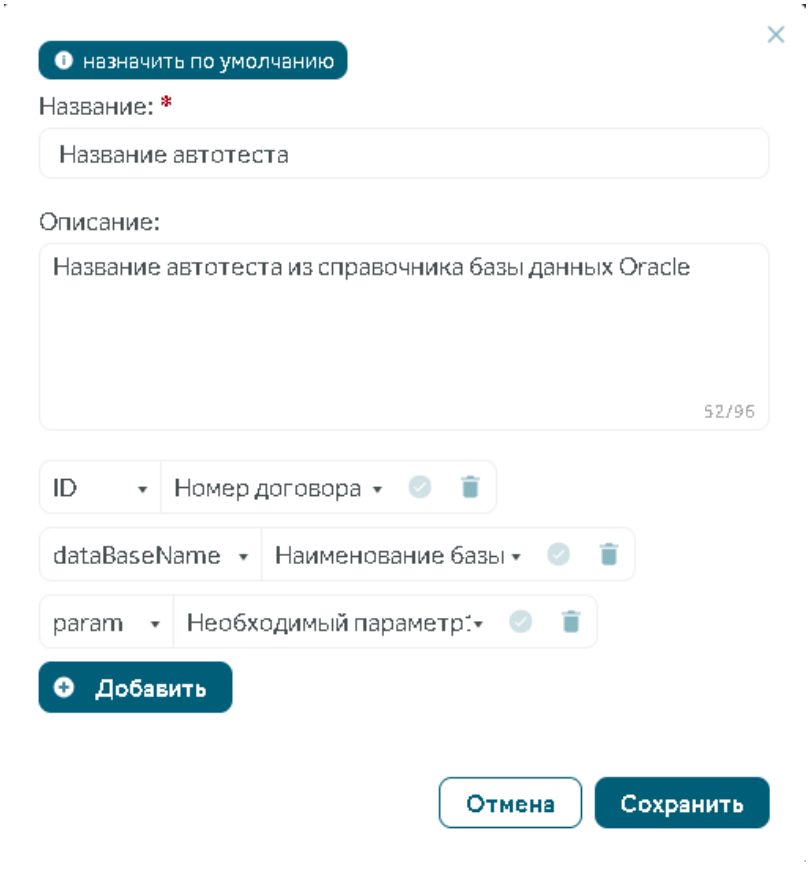


Рис. 8. Модель заполнение параметров для автоматического теста на TestIt

7. Исследование эффективности разработанной системы по сравнению с исходными методами тестирования

Уникальность данной разработки заключается в разработке для банка стандарт и методологию автоматизации тестирования больших объёмов данных, которые к тому же продолжают стремительно расти.

Поскольку банк – является отличным примером масштаба данных для хранения, необходимо качественно и эффективно производить тестирование функционала, имея возможность быстро обрабатывать большое количество данных в режиме онлайн. Сюда относится регулярное тестирование всех функциональных модулей, состоящих из больших массивов данных, требующих непрерывный контроль для обеспечения текущих бизнес требований компании.

Таким образом мы вырабатываем общий подход, который сможет использовать данную разработку и пакет параметризации, чтобы автоматические тесты разрабатывать было удобно, быстро и на любых количествах данных в качестве входных параметров.

Результат диссертационной работы внедрен в бизнес процессы компании, что позволяет значительно ускорить разработку специализированных тестов. Имеется положительное влияние на качество разрабатываемых сервисов и поддержание стабильности работы системы.

Количественная статистика по временным затратам обработки тест-кейсов в режимах маниуальном и автоматическом приведена в таблице 1.

Анализ на неэффективность и нестабильность процессов при ручном тестировании основывался на таких показателях как:

- время обработки;
- экономические затраты;
- надежность системы.

Оценка надежности разработанной интеграционной системы установлена экспериментально на уровне 98%. Среднее время обработки тест-кейса для одного функционала у сотрудника, осуществляющего тестирование, занимает 15 минут.

В результате проведенного анализа было выявлено, что разработанная методология по всем показателям значительно превышает старые способы тестирования данных для приведенных примеров.

Таблица 1 – Анализ качественных показателей разработанной интеграционной системы по сравнению со старыми способами тестирования данных

Бизнес процесс	Способ реализации	Время обработки (входит время, затраченное на работу базы данных)	Надежность системы	Вид тестирования	Эффективность
Проведение тестирования одного тест-кейса	Ручное написание скриптов и сравнивание полученных результатов	15 мин	Небольшая вероятность ошибки сотрудника. 90%	Ручной	92%
Проведение тестирования одного тест-кейса	Автоматическая обработка	0.58 мин	99%	Автоматический	99%
Проведение тестирования массового тестирования (10 000 договоров)	Ручное написание скриптов и сравнивание полученных результатов	≈ 150 000 мин ≈ 2500 ч ≈ 104 д ≈ 3.47 мес.	Экстремально высокая вероятность ошибки сотрудника. 0. 009%	Ручной	Не представляется возможным
Проведение тестирования массового тестирования (10 000 договоров)	Автоматическая обработка	218 мин	97% Ошибка, связанная с прерыванием работы базы данных	Автоматический	98%

8. Обсуждение результатов

В рамках проделанной работе по внедрению нового метода автоматического тестирования функционала, были проделаны несколько этапов:

- Произведен анализ существующего метода тестирования.

Проанализированы возможные методы разработки автоматических тестов в базе данных Oracle

- Разработан метод внедрения интеграционного функционала автоматического тестирования баз данных Oracle, являющаяся гибким методом в покрытии тестами разного функционала лишь с использованием одного шаблона на входе в программу
- Построены соответствующие ER-диаграммы и схема связи объектов в базе данных
- Созданы новые объекты хранения данных об автоматических тестах в базе данных
- Написаны пакеты параметризации и API процедур на языке программирования PL/SQL, где прописано получение параметров из JSON файла и универсальный вызов процедуры обработки.
- Проведен анализ эффективности разработки по сравнению с предыдущим методом тестировании в компании

9. Финансовый менеджмент, ресурсоэффективность ресурсосбережение

Проект инициализации определяет цели и содержание, фиксируя финансовые ресурсы. Определены внутренние заинтересованные стороны проекта, а также общее влияние на результат проекта.

9.1. Анализ потенциальных потребителей и конкурентов

Заказчиком данного проекта является один из крупных банков, в котором я занимаю позицию backend разработчика карточного продукта банка, то есть работа идет с логической частью работы продукта. Было решено со стороны компании не упоминать название банка. Проект является внутренним в функционале компании, а не частью доработки продукта для клиентов. Для более быстрого и удобного формата тестирования функционала банка перед накатом на боевую базу, поступил заказ по разработке и усовершенствованию процессов тестирования в команде. Необходимо разработать фреймворк для всего функционала карточного продукта с одним входом и без создания каждый раз нового автоматического теста. Разработка велась на языке программирования oracle PL/SQL.

Так как проект является внутренним и к тому же собственностью банка, у меня нет возможности оценить процесс автоматического тестирования в других компаниях. В открытом доступе этой информации нет. В данном случае конкурентами могут выступать лишь другие отделы компании, которые внедрят свой процесс автоматического тестирования, пока же преобладает ручное тестирование сотрудниками тестирования.

Информация о заинтересованных сторонах проекта, а также ожидаемом результате и критериях достижения целей представлена в таблице ниже.

Таблица 2 – Информация актуальности проекта

Заинтересованные стороны проекта	**Ожидание заинтересованных сторон от проекта**
Разработчики и тестировщики проекта. Бизнес аналитики компании.	Использование проекта удобно в организации тестирования продуктов компании. Затрата меньшего количества

	ресурсов разработки в дальнейшем. Высокая производительность. Низкая стоимость.
Томский Политехнический университет	Магистерская диссертация студента ТПУ

9.2. Технология QuaD

Технология QuaD позволяет оценить перспективность разработки на рынке и целесообразность вложения средств в научно-исследовательский проект. Результаты оценки, проведенной в табличной форме, представлены в таблице 3.

Таблица 3 – QuaD-анализ разработки

Критерии оценки	Вес критерия	Средний балл	Максимальный балл	Относительное значение (3/4)	Средневзвешенное значение (5х2)
1	2	3	4	5	6
Производительность	0,22	80	100	0,8	0,176
Отказоустойчивость	0,17	90	100	0,9	1,07
Унифицированность	0,1	70	100	0,7	0,07
Безопасность	0,05	80	100	0,8	0,04
Потребность в ресурсах памяти	0,13	95	100	0,95	0,1235
Функциональная мощность	0,1	75	100	0,75	0,075
Простота эксплуатации	0,01	40	100	0,4	0,004
Масштабируемость	0,06	75	100	0,75	0,045
Конкурентоспособнос	0,03	50	100	0,5	0,015

ть продукта					
Перспективность рынка	0,09	85	100	0,85	0,0765
Цена	0,2	40	100	0,4	0,08
Финансовая эффективность научной разработки	0,3	80	100	0,8	0,24
Итого	1				2,015

Анализ альтернатив:

$$K = \sum B_i + Б_i, \quad (1)$$

гдеK-конкурентоспособность разработки;

Вi-вес показателя;

Бi-балл i-го показателя.

По результатам оценки качества и перспективности можно утверждать, что перспективность текущей разработки немного ниже среднего в силу низкого критерия перспективности рынка из-за локальной разработки для компании.

Как говорилось выше, данный проект является внутренним заказом компании и возможность оценить внутренние возможности тестирования в других компания не представляется возможности из-за конфиденциальности разработки. Таким образом, можно сказать, что аналогов разработки не представляется.

9.3. Матрица SWOT

SWOT-анализ используется для определения сильных и слабых сторон проекта.

Таблица 4 – SWOT анализ

Сильные стороны: С1. Повышение эффективности тестирования продукта. С2. Уменьшение трудозатрат тестировщиков и разработчиков С3. Исключение тривиальных кейсов	Слабые стороны: Сл1. Зависимость от поддержки программного обеспечения Сл2. Дополнительное обучение тестировщиков или аналитиков использования нового продукта Сл3. Нет фронт части
Возможности: В1. Покрытие всего функционала компании автоматическим тестированием в автономном режиме В2. Гибкая разработка для последующих проектов тестирования любого функционала В3. Возможность обработки большого объема данных в несколько миллионов строк	Угрозы: У1. Прекращение поддержания программного обеспечения У2. Большие затраты на новую разработку

Для выявления соответствия сильных и слабых сторон проекта строится интерактивная матрица проекта. Ее использование помогает разобраться с различными комбинациями взаимосвязей области матрицы SWOT. Итоговая матрица SWOT представлена ниже.

Таблица 5 - Матрица сильных сторон и возможностей проекта

	С1	С2	С3
В1	+	+	+
В2	-	+	-
В3	+	-	-

Анализ этой интерактивной электронной таблицы показал корреляцию сильных сторон и возможностей: В1С1С2С3, В2С2, С3С1.

Таблица 6 - Матрица возможностей и слабых сторон проекта

	Сл1	Сл2	Сл3
В1	-	-	-
В2	-	+	-
В3	-	-	-

Соотношения слабых сторон и возможностей следующие: В2Сл2. Следующим шагом в анализе проекта является выявление соотношения сильных сторон и угроз.

Таблица 7 – Матрица сильных сторон и угроз

	С1	С2	С3
У1	-	-	-
У2	-	+	-

Соотношения угроз и сильных сторон следующие: У2С2. Следующим шагом в анализе проекта является выявление соотношения слабых сторон и угроз.

Таблица 8 – Матрица слабых сторон и угроз

	Сл1	Сл2	Сл3
У1	+	+	-
У2	+	+	-

Соотношение недостатков и угроз следующие: У1Сл1Сл2, У2Сл1Сл2.

Данная разработка обладает рядом возможностей в условиях низкой вероятности возникновения угроз. Разработка спроектирована таких образом, что сильные стороны предусматривают изменение требований к разработки любых автотестов без построения архитектуры, имеет универсальный подход с одним входом.

Итоговая матрица SWOT – анализа приведена в таблице 9.

Таблица 9 - Итоговая матрица SWOT – анализа

	Сильные стороны: С1. Повышение эффективности тестирования продукта. С2. Уменьшение трудозатрат тестировщиков и разработчиков С3. Исключение тривиальных кейсов	Слабые стороны: Сл1. Зависимость от поддержки программного обеспечения Сл2. Дополнительное обучение тестировщиков или аналитиков использования нового продукта Сл3. Нет фронт части
Возможности: В1. Покрытие всего функционала компании автоматическим тестированием в автономном режиме В2. Гибкая разработка для последующих проектов тестирования любого функционала	В1С1С2С3, В2С2, С3С1	В2Сл2

В3. Возможность обработки большого объема данных в несколько миллионов строк		
Угрозы: У1. Прекращение поддержания программного обеспечения У2. Большие затраты на новую разработку	У2С2	У1Сл1Сл2, У2Сл1Сл2

9.4. Цели и требования проекта

Целью проекта является урегулирование и перенесение части тестевых кейсов, в частности трудоемких и сложных, на автоматический формат их выполнения. Для этого необходима разработка нового внутреннего фреймворка, который позволит упростить написание автотестов для всего функционала банка. Разработчику не придется заново простраивать уникальную архитектуру выполнения для каждого теста, нужно лишь сделать insert в таблицу, в которой хранится наименование автотеста, процедура запуска его и входные параметры (их количество может варьироваться). А также прописать функцию с вызовом нужного функционала для тестирования, которая включает в себя использование разработанных в данном проекте фреймворке.

При выполнении джоба функционал умеет парсить входные параметры, подставляя их в выполняемую процедуру в динамическом SQL с использованием языка программирования PL/SQL.

Информация об иерархии целей проекта и критериях достижения целей представлена в таблице ниже.

Таблица 10 – Цели и результаты проекта

Цели проекта	Обеспечить гибкую разработку тестирования продуктов компании с большим объемом данных, повысить время обработки продукты и эффективность
Ожидаемый результат от проекта	На основании проведенного исследования найти недостатки аналога разработки и устранить их путем построения архитектуры разработки универсального подхода и улучшения показателей гибкости использования
Критерии результата проекта	Экономия ресурсов разработки на выполнение новый кейсов тестирования, улучшение качества и быстродействия продукта
Требования к результатам проекта	Завершение проекта в срок
	Упрощение процессов тестирования
	Удобство в использовании разработанного фреймворка
	Повышение скорости обработки массовых процессов с большим объемом данных

9.5. Организация структуры проекта

Организация структуры проекта продемонстрирована в таблице ниже.

Таблица 11 – Рабочая группа проекта

№	**ФИО**	**Позиция**	**Функции**	**Время чч**
1	Попова М.А.	Разработчик	Построение архитектуры и разработка проекта	800

2	Консультант от компании	Консультант	Консультирование в части построения архитектуры разработки, создание технического задания	112
3	Губин Е.И.	Научный руководитель	Консультация по оформлению дипломной работы. Проверка ВКР	128
Всего:				816

9.6. Ограничения

Ограничения и допущения приведены в таблице ниже.

Таблица 12 – Ограничения

Фактор	**Ограничение**
Бюджет проекта	403 884 руб
Источник финансирования	Бюджет организации заказчика
Срок исполнения проекта	20 декабря 2022 – 1 июня 2023
Дата утверждения проекта	20 декабря 2022
Дата завершения проекта	1 июня 2023
Распечатка ВКР	450 руб
Расходы научного руководителя	63 516 руб

В результате инициализации проекта была поставлена задача проекта, описаны ожидаемые результаты, определены требования, выставлены сроки и утвержден график работ.

9.7. Планирование проекта

Для определения графика реализации проекта используется диаграмма Ганта, которая представляет собой горизонтальный график работы за длительные периоды, описываемые датами начала и завершения назначенной работы.

Так как таблица 13 получится слишком большая, то трудозатраты научного руководителя будут отмечены в днях со специальным символом «*».

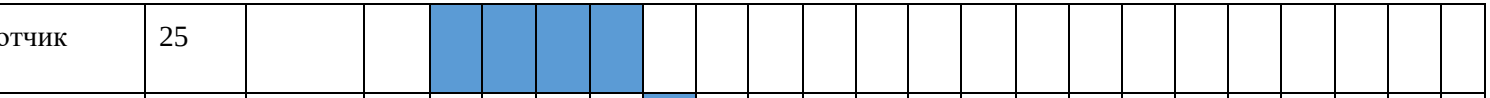
Таблица 13 – Сроки проектирования и исследований

Задание	Трудоемкость						Длительность выполнения задачи в днях T_{pi}		Продолжительность задачи в календарных днях T_{ki}	
	$t_{мин}$, человеко-дней		$t_{макс}$, человеко-дней		$t_{сред}$, человеко-дней					
	Разработчик	Консультант	Разработчик	Консультант	Разработчик	Консультант	Разработчик	Консультант	Разработчик	Консультант
Постановка задачи разработки	2	5	5	6	3	5	3	4	3	4
Построение архитектуры разработки	5	2	6	3	5	2	5	2	5	2
Создание инструмента гибкого использования	20	0	25	0	22	0	25	0	25	0
Построение ER-диаграмм	3	0	4	0	3	0	3	0	3	0
Построение плана разработки на основании проработанной архитектуры	2	5	5	8	3	6	4	5	4	5
Анализ недостатка существующего продукта	3	0	5	0	4	0	4	0	4	0
Оценка исправления загруженности обработки	1	1	1	1	1	1	1	1	1	1
Оценка трудозатрат разработчика	1	1	1	1	1	1	1	1	1	1

Разработка объектов PL/SQL	30	0	50	0	30	0	68	0	68	0
Подбор кейсов	1	0	3	0	2	0	1	0	1	0
Оценка произоводительности	1	2	2	3	1	2	1	1	1	1
Тестирование продукта	1	0	1	0	1	0	1	0	1	0
Составление итогового отчета	3	0	4	0	3	0	2	0	4	0
Проверка ВКР и отчета *16	1	0	1	0	2	0	2	0	2	0

Таблица 14 – График разработки проекта

№	Задание	Иполнитель	$T_{кi}$дней	Продолжительность задачи																					
				Декабрь	Январь				Февраль				Март				Апрель					Май			
				1	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	5	1	2	3	4
1	Постановка задачи разработки	Консультант	3																						
		Разработчик	4																						
2	Построение архитектуры разработки	Консультант	5																						
		Разработчик	2																						

3	Создание инструмента гибкого использования	Разработчик	25	
4	Построение ER-диаграмм	Разработчик	3	
5	Построение плана разработки на основании проработанной архитектуры	Консультант	4	
		Разработчик	5	
6	Анализ недостатка существующего продукта	Разработчик	4	
7	Оценка исправления загруженности обработки	Консультант	1	
		Разработчик	1	
8	Оценка трудозатрат разработчика	Консультант	1	
		Разработчик	1	
9	Разработка объектов PL/SQL	Разработчик	48	
10	Подбор кейсов	Разработчик	1	
11	Оценка производительности	Консультант	1	
		Разработчик	1	
12	Тестирование продукта	Разработчик	1	
13	Составление итогового отчета	Разработчик	2	
14	Проверка ВКР и отчета	Научный руководитель	16	

		Разработчик	2																						

	Разработчик	100
	Консультант	14
	Руководитель	16

Таким образом продолжительность выполнения задачи разработчиком составила 100 дней, консультантов – 14 дней, научным руководителем 16 дней.

9.8. Финансирование проекта

Бюджет проекта отображает значения всех видов трудозатрат, связанных с его реализацией. В стоимость данного проекта входит:

• Затраты на приобретения оборудования;

• Затраты на аренду программного обеспечения;

• Расходы на основной и дополнительный заработок сотрудника;

• Расходы научного руководителя;

• Распечатка ВКР.

9.8.1. Стоимость оборудования

Таблица 15 – Расчет стоимости основного оборудования

Название оборудования	**Количество**	**Стоимость, тысяч рублей**
Компьютер HP Z240 S core i7	1	92
Монитор Philips	2	42
Мышка	1	1.5
Клавиатура	1	2.5
Программное обеспечение Oracle	6	84
Windows 10	1	21
Всего		243

Так как оборудование не покупалось отдельно для данного проекта, то стоит сделать переоценку на срок его использования. Срок эксплуатации компьютера — 7 лет, лицензии

Microsoft Windows 10 — 4 года, остального ПО — год. Исходя из того, что проект выполнялся 6 месяцев, затраты составили следующие:

Компьютер + мониторы + мышка +клавиатура = (92 +42+1,5+2,5) * 1000 : 7 : 2 = 9 857 рублей

ПО Oracle = 84 000 за 6 месяцев

Windows 10 = 21 000 :4 :2 = 2625 рублей

Таблица 16 – Расчет стоимости основного оборудования с расчетом на 6 месяцев

Название оборудования	**Количество**	**Стоимость, тысяч рублей**
Компьютер HP Z240 S core i7	1	6,571
Монитор Philips	2	3
Мышка	1	0,107
Клавиатура	1	0,179
Программное обеспечение Oracle	6	84
Windows 10	1	2,625
Всего		96,482

9.8.2. Базовая заработная плата

Размер расходов на оплату труда работников определяется исходя из трудоемкости выполняемых работ и действующей системы окладов и тарифных ставок.

Расчет базового оклада руководителя научного проекта осуществляется на основе отраслевой системы оплаты труда.

1. Заработная плата - определяется предприятием в соответствии с уровнем разработчика.
2. Поощрительные выплаты - устанавливаются начальником отдела за эффективную работу, выполнение дополнительных обязанностей и т.д.

3. Другие платежи: районный коэффициент.

Поскольку поощрительные премии, иные выплаты и поощрения зависят именно от деятельности руководителя, коэффициент поощрительных премий примем равным 30%, а коэффициент поощрения руководителя за добросовестную трудовую деятельность - 25%.

Базовая заработная плата руководителя определяется по формуле:

$$S_{\text{b}} = S_r \cdot T_w, \quad (2)$$

где S_r – обычная заработная плата работника;

$T_{\text{р}}$- продолжительность работы, рабочие дни.

Дополнительная заработная плата:

$$S_{\text{add}} = 0{,}15\, S_{\text{b}} \quad (3)$$

Средняя дневная заработная плата при 5-дневной рабочей неделе:

$$S_{\text{d}} = \frac{S_{\text{M}} \cdot M}{F_d}, \quad (4)$$

где S_{M} – месячная заработная плата рабочего, руб.;

F_d – количество рабочих дней в месяце, дней,

M – количество месяцев работы без отпуска в течение года

Полная заработная плата может быть определена как:

$$S_F = S_b + S_{add}, \quad (5)$$

Средняя заработная плата разработчика уровня junior составляет 42 000 рублей оклад, районный коэффициент по томской области составляет 30%, премия за хорошие достижения составляет 25%. Налог составляет 13%

Ежемесячная заработная плата:

- Для разработчика:

$$S_b = S_r \cdot \left(1 + k_{\text{pr}} + k_d\right) \cdot k_r = 42000 \cdot (1 + 0{,}3 + 0{,}25) \cdot 0{,}87 = 56\,637 \text{ RUB}$$

Итого для разработки проекта с учетом количества месяцев разработки необходимо на зарплату сотрудника 339 822 рублей.

Средняя дневная заработная плата:

$$S_{\text{D.sup.}} = \frac{S_{\text{b.sup.}}}{F_d} = 2\,752{,}04 \text{ RUB} \quad (6)$$

где среднее количество рабочих дней в месяце определялось как:

$$F_d = \frac{\text{T}_w}{12} = \frac{247}{12} = 20{,}58. \quad (7)$$

9.8.3. Расходы научного руководителя

Месячный должностной оклад работника вычисляется по следующей формуле:

$$З_{\text{дн}} = З_{\text{б}} \cdot \left(к_{\text{пр}} + к_{\text{д}}\right) \cdot к_{\text{р}}, \quad (8)$$

где $к_{\text{пр}}$ - премиальный коэффициент оплаты труда;

$З_{\text{б}}$ - базовый должностной оклад;

$к_{\text{д}}$ - коэффициент доплат и надбавок (15-20%);

$к_{\text{р}}$ - районный коэффициент (1,3 для Томска).

Для расчета основной заработной платы руководителя возьмем оклад равный 36 000 рублей. Тогда размер основной заработной платы с учетом вычета налога в размере 13% составит 63 516 рублей.

9.8.4. Отчисления на социальные нужды

Включает в себя отчисления во внебюджетные фонды и определяется по формуле (9).

$$С_{внеб} = k_{внеб} (З_{осн} + З_{доп}), \quad (9)$$

где $k_{внеб}$ – коэффициент отчисления на уплату во внебюджетные фонды.

Всего расходов за 1 месяцев работы разработчика и 1 месяц работы научного руководителя составят 56 637 и 63 516 рублей, соответственно.

Таким образом, отчисления во внебюджетные фонды, исходя из всех вышеперечисленных взносов, составляют:

$$С_{внеб} = 0{,}3 \cdot (56\,637 + 63\,516) = 36\,045 \quad (10)$$

9.9. Оценка готовности разработки к коммерциализации

Одной из важных задач в ходе выполнения данного раздела является оценка готовности разработки к коммерциализации. Поскольку данная разработка является прямым заказом компании, то данный проект является коммерческим и уже внедряется в работу компании. За счет данной разработки универсального тестирования продуктов компании, которые включают в себя множество этапов предобработки большого объема данных, компания сможет экономить финансы в силу уменьшения инцидентов на боевой базе и своевременного их выявления.

10.10. Разработка экономической части

10.10.1. Первичный анализ проекта

Текущее исследование экономической оценке эффективности использования, разработанного фреймворка для тестирования продуктов компании. Основная цель – найти экономически эффективный способ разработки программного обеспечения и его тестирования.

Для оценки окупаемости продукта потребуется месяца, чтобы оценить дельта финансовых потерь, связанных с более качественным и быстрым тестированием

разработанного продукта и внедрения его на основную базу данных, где происходит реальная работа с клиентами.

Разработка позволяет уменьшить тем самым риски связанные с допущенными ошибками во время разработки программного обеспечения, за что компания несет финансовые убытки в больших объемах.

Оценить количественную эффективность разработки можно по формуле:

$$NPV = \sum_{t=1}^{T} \frac{CF_t}{(1+r)^t} - INV, \tag{11}$$

где CF_t – чистый денежный поток за t-период;

г – ставка дисконта;

t – номер периода времени;

T – время жизни проекта;

INV – первоначальные инвестиции.

10.10.2. Экономическая эффективность

Показатели экономической эффективности проекта учитывают финансовые последствия для предприятия или компании, которая реализует данный проект.

Определение эффективности происходит на основе расчета интегрального финансового показателя, который рассчитывается следующим образом:

$$I = Ф(p)/(Ф\_max) \tag{12}$$

где Ф(p) - стоимость исполнения работ; Ф_max - максимально допустимая стоимость исполнения проекта.

Общий бюджет проекта составил 403 884 рублей. Исходя из того, что разработка является достаточно сложной для разработчика уровня junior, если бы ее выполнял

разработчик уровня выше middle, затраты бы на заработную плату увеличились и составили:

$$S_b = S_r \cdot \left(1 + k_{\text{pr}} + k_d\right) \cdot k_r = 68000 \cdot (1 + 0{,}3 + 0{,}25) \cdot 0{,}87 = 96\,135 \text{ RUB} \quad (13)$$

Таким образом показатель эффективности составляет:

$$I = \frac{56\,637}{96\,135} = 0{,}59 \quad (14)$$

Значение финансового показателя составляет 0,59, что свидетельствует об эффективном использовании финансовых ресурсов.

$$\text{I} = \Phi(\text{p})/(\Phi\_\text{max}\ ) \quad (15)$$

11. Социальная ответственность

Объектом исследования является разработка гибкой методологии автоматического тестирования баз данных Oracle с большим объемом данных. Созданная система позволяет эффективнее и быстрее выполнять массовое тестирование больших данных по сравнению с мануальным тестированием. В соответствии с созданной интеграцией с платформой TestIt, пропадает необходимость разработки шагов, связанных с запуском автотеста и проверки его результата.

Выполнение данного проекта происходит в офисе компании «Совкомбанк Технологии», которая является заказчиком разработки для внедрения ее в корпоративный стандарт тестирования функционала компании. Помещение является закрытым, отапливаемым и вентилируемым, на рабочем месте предоставлен персональный компьютер.

Целью данного раздела является выявление специальных правовых норм трудового законодательства, микроклимата помещения, освещенности, а также возможных опасных факторов на окружающую среду и человека.

11.1. Правовые и организационные вопросы обеспечения безопасности

Регулирование государственных гарантий трудовых правил и свобод граждан, создание благоприятных условий труда и защита прав и интересов работника и работодателя освещено Трудовым кодексом Российской Федерации от 30.12.2001 N 197-ФЗ [1]. Запрещается принудительный труд и дискриминация в сфере труда, обеспечивая права каждого гражданина на справедливые условия труда, включая ограничение рабочего времени, полную и своевременную выплату труда и равенство возможности работников.

В соответствии со ст. 111 ТК РФ [8] рабочая неделя не должна превышать 40 часов в неделю. При пятидневной рабочей неделе работникам предоставляются два выходных дня в неделю, при шестидневной рабочей неделе - один выходной день.

В соответствии со ст. 212 [8] работодатель обязан обеспечить безопасные условия и охраны труда, а также социальные страхование от несчастного случая.

Данные о работнике обрабатываются только с согласия и охраняются Федеральным Законом от 27.07.2006 N 152-ФЗ «О персональных данных» [9].

11.2. Организационные мероприятия при компоновке рабочей зоны

Согласно ГОСТ 12.12.032-78 [10] по системе стандартом безопасности рабочего места при выполнении работ сидя устанавливаются следующие требования:

- Конструкция рабочего места должны обеспечивать выполнение трудовых операций в пределах зоны досягаемости.
- Положение работающего должно быть обеспечено оптимально производственного оборудования и рабочего места, которое достигается регулированием высоты рабочей поверхности, сиденья и пространства для ног. Конструкция регулируемого кресла должна соответствовать требованиям ГОСТ-21889-76 [11].
- Форма рабочей поверхности может быть прямоугольной формы, иметь вырез для корпуса работающего или углубление.
- Ширина пространства для ног должна быть не менее 500 мм.
- Средства отображения информации, которые требуют менее точного и быстрого считывания показателей, следует располагать в вертикальной поверхности под углом ±30º от нормальной линии взгляд и в горизонтальной плоскости под углом ±30º от сагиттальной плоскости.
- Экран видеомонитора должен находиться на расстоянии 600-700 мм от глаз пользователя.

11.3. Производственная безопасность при разработке проектного решения

В данном разделе анализируется вредные и опасные факторы, которые возникают при разработке в офисе проектируемого решения. В таблице 17 представлены возможные вредные и опасные факторы с учетом деления работы на три этапа: проектирование, разработка и эксплуатация.

Таблица 17 – Возможные опасные и вредные факторы на рабочем месте при выполнении НИР

Факторы (ГОСТ 12.0.003-2015)	Нормативные документы
1. Отсутствие или недостаток необходимого естественного или искусственного освещения	СНиП 23-05-95* «Естественное и искусственное освещение»

	СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания»
2. Нервно – психические перегрузки, связанные с умственным перенапряжением	ТОИ Р-45-084-01 «Типовая инструкция по охране труда при работе на персональном компьютере»
3. Физические статистическое перегрузки, связанные с рабочей позой	ГОСТ 12.2.032-78 ССБТ «Рабочее место при выполнении работ сидя»
4. Отклонение параметров микроклимата	СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания»
5. Повышенный уровень и другие неблагоприятные характеристики шума	СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания»
6. Повышение напряжения в электрической цепи	СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания» ГОСТ 12.1.019-2017 «Система стандартов безопасности труда. Электробезопасность»

11.3.1. Отсутствие или недостаток необходимого естественного или искусственного освещения

Для рассмотрения данного фактора необходимо учитывать световой день в городе Томск в зависимости от времени года. Световой день в зимнее время года длится в среднем 6 часов 50 минут, а летом более 17 часов. Поэтому в летнее время чаще преобладает естественное освещение в кабинете в силу погодных условий, а зимой искусственное.

Недостаточно освещенности рабочей зоны может сказаться на здоровье работника, в частности воздействуя на зрение человека, а также производительность труда. Согласно СанПиН 1.2.3685-21 [12] освещенность поверхности рабочей зоны должна быть 300-500 лк.

Схема кабинета показана на рисунке 9. Проведем расчет общего равномерного искусственного освещения методом коэффициента светового потока, учитывающего световой поток, отраженный от потолка и стен.

Исходя из схемы на рисунке 1, площадь общего помещения имеет размер 43.2+8.7 = 51.9 метра. Высота потолков 4.5 м. Высота рабочей поверхности $h_{рп}$ = 0.8м. Потолок представляет собой чистый бетон, а стены свежепобеленные с окнами без штор, тогда значение коэффициентов отражения составляет 50% и 30%, соответственно. Помещение с малым выделением пыли имеет коэффициент запаса K = 1.5, а коэффициент неравномерности для люминесцентных ламп Z = 1.1. Рассчитываем систему общего люминесцентного освещения с λ = 1.4 для светильников типа ОД. Приняв h_c = 0,5 м, получаем:

h = 4,5 - 0,5 - 0,8 = 3,2 м;

L =λ · h = 1,4 · 3,2 =4,5 м;

L/3 = 1,5 м.

Светильники размещены в 4 ряда по 3 лампы с возможностью включения/отключение половины из них. Мощность светильника типа ОД с длиной 1.5 м 65 Вт при расстоянии между светильниками в 50 см. В каждом светильнике установлено 2 лампы, общее их количество составляет $N_л$ = 24.

Находим индекс помещения:

$$i = \frac{S}{h(A+B)} = \frac{51.9}{3.2(8+6.5)} = 1.12 \quad (16)$$

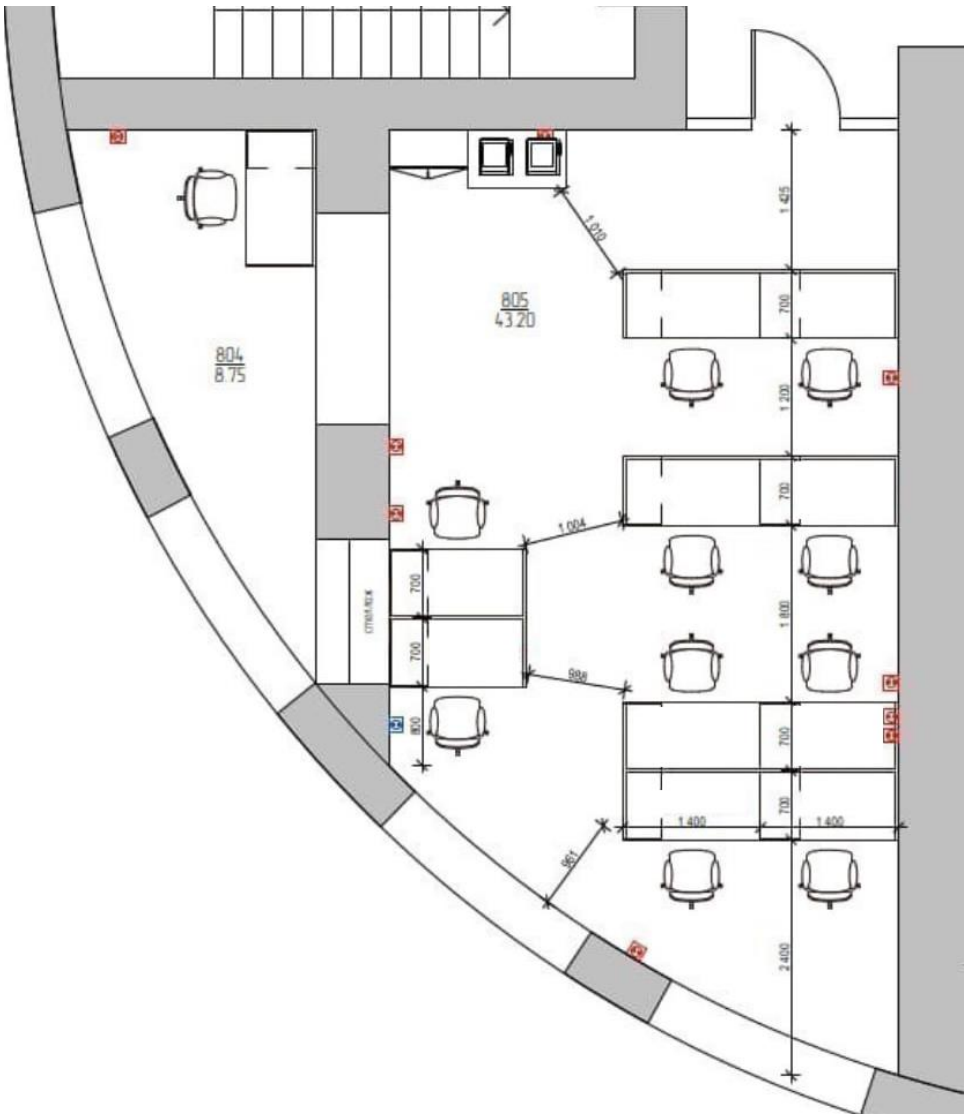


Рис.9 Схема кабинета

Исходя из табличных значений показателя коэффициента использования светового потока светильников с люминесцентными лампами с учетом коэффициентов отражения потолка и стен и индекса помещение η = 49%.

Определяем потребный световой поток лам в каждом из рядов:

$$\Phi = \frac{E_н \cdot S \cdot K_з \cdot Z}{N_л \cdot \eta} = 4460 \text{ лм}; \quad (17)$$

Выбираем ближайшую стандартную лампу - ЛХБ 80 Вт с потоком 5000 лм. Проводя проверка полученных значений:

$$-10\% \leq \frac{\Phi_{л.станд} - \Phi_{л.расч}}{\Phi_{л.станд}} \cdot 100\% \leq +20\%; \quad (18)$$

Получаем:

$$-10\% \leq 10\% \leq +20\%; \quad (19)$$

Электрическая мощность всей осветительной системы:

$$\mathrm{P} = N_{л} \cdot \mathrm{p}_{л} = 1920 \text{ Вт}. \quad (20)$$

11.3.2. Нервно – психические перегрузки, связанные с умственным перенапряжением

Согласно ГОСТ 12.0.003-2015 [13], работник при работе с информационной нагрузкой также подвергается еще одному вредному фактору – умственному перенапряжению.

Чрезмерный поток информации оказывает угнетающее влияние на деятельность работающего. Ухудшается восприятие, а также функции памяти, как кратковременной, так и долговременной.

При длительной нагрузки во избежание ухудшения зрения и головных болей, необходимо качественные сенсорные и зрительные система. Согласно ТОИ Р-45-084-01 [14], при 8-часовой рабочей смене и работе на компьютере при виде трудовой деятельности в режиме – творческая работа диалога с компьютеров (группа В), необходимо проводить перерывы через 1,5 - 2,0 часа от начала рабочей смены и через 1,5 - 2,0 часа после обеденного перерыва продолжительностью 20 минут каждый или продолжительностью 15 минут через каждый час работы.

Для исключения данного фактора в офисе имеется зона отдыха, в которой размещены как спортивный инвентарь для занятия физическими нагрузками, так и диван с кресло-мешком для спокойного отдыха от рабочего процесса.

11.3.3. Физические статистическое перегрузки, связанные с рабочей позой

Согласно ГОСТ 12.0.003-2015 [13], выделяются ряд физических перегрузок, которые подразделяются также на статические, связанные с рабочей позой.

При большой концентрации информационного потока, организм находится в напряжении, сковывая мышцы шеи и спины. Находясь долго в таком положении сидя может привести к проблемам со здоровьем тела работника.

Во избежание проблем корпуса человека, согласно ГОСТ 12.2.032-78 ССБТ [9] конструкция рабочего места и расположение всех элементов должна соответствовать антропометрическим, физиологическим и психологическим требованиям, а также характеру работы.

Рабочая поверхность может содержать дополнительное углубление для периферийных устройств (клавиатуры). При легкой работе сидящего, рабочее место организуется так, чтобы оно не требовало свободного передвижения.

При работе с ПЭВМ при отсутствии регулирующих механизмов рабочей поверхности, высота его должна составлять 630 мм для женщин, высота сиденья 400 мм.

11.3.4. Отклонение параметров микроклимата

Согласно СанПиН 1.2.3685-21 [12] устанавливаются санитарные правила, нормы и гигиенические нормативы микроклимата рабочего места с учетом интенсивности энергозатрат работающего в зависимости от периода года.

При 8-часовой рабочей смене при минимальной физической активности (сидячий вид деятельности, умственный труд) вводятся оптимальные показатели микроклимата, которые приведены в таблице 17.

Таблица 17 – Оптимальные величины микроклимата для работающих в помещении при минимальной физической активности

Период года	Температура воздуха, ºС	Температура поверхностей, ºС	Относительная влажность воздуха, %	Скорость движения воздуха, м/с
Теплый	22-24	21-25	40-60	0,1
Холодный	23-25	22-26	40-60	0,1

При определении температуры воздуха и поверхностей использовался термометр. Показания температуры воздуха составили 23 ºС, а поверхностей 22 ºС.

11.3.5. Повышенный уровень и другие неблагоприятные характеристики шума

Превышение уровня шума является также вредным фактором на рабочем место. Присутствие постоянных шумов воздействует на слуховой аппарат, нервную систему и влияет на общее самочувствие и концентрацию работника.

В летнее время в помещении активно работает кондиционер, который издает шум, поэтому согласно СанПиН 1.2.3685-21 [12] допустимые уровни шума в рабочем офисном помещении не должен превышать 50дБА.

11.3.6. Повышение напряжения в электрической цепи

К опасным факторам на рабочем месте относится поражение электрическим током.

Согласно ГОСТ 12.1.019-2017 [15] по системе стандартов безопасности труда в части электробезопасности, выделяются базовые принципы защиты от поражения электрическим током:

- Проводящие части, находящиеся под опасным рабочим, наведенным, остаточным напряжением, не должны быть доступными, а доступные проводящие части не должны находиться под опасным напряжением при нормальных условиях (при отсутствии повреждения), а также в случае единичного повреждения.
- Защиту при нормальных условиях (защиту от прямого прикосновения) обеспечивают посредством основной защиты, а защиту при условиях единичного повреждения (защиту при косвенном прикосновении) обеспечивают посредством защиты при повреждении.

11.4. Экологическая безопасность

Целью данного раздела является выявление потенциальных опасных объектов для окружающей среды, а также применение мер безопасности при разработке проекта.

Имеются два отходных оборудования при разработке данного проекта, которые могут повлечь за собой вред окружающей среде, а именно литосфере, при их утилизации.

Согласно ГОСТ 12.3.031-83 «Работы со ртутью» [16] ртутные отходы, в том числе люминесцентные лампы, содержащие ртуть, подлежат утилизации в соответствии с

требованиями технологической документации. Неутилизированные ртутные отходы должны быть захоронены в соответствии с санитарными правилами проектирования, строительства и эксплуатации полигонов захоронения не утилизируемых промышленных отходов.

Согласно требованиям, ГОСТ Р 70146-2022 по утилизации отходов электроники и электробытовой техники организация должна [17]:

- обеспечить контроль воздействия на окружающую среду опасных и вредных факторов, соблюдения техники безопасности и охраны труда, когда работы по сбору, транспортированию, обработке и утилизации отходов выполняются силами сторонних исполнителей;
- подтвердить наличие технических возможностей сбора транспортирования, обработки и утилизации отходов электронного и электробытового оборудования, которые она принимает, чтобы обеспечить соблюдение нормативных правовых требований и обеспечение охраны окружающей среды;
- в случае эксплуатации объектов, используемых для обезвреживания отходов I—V классов опасности, применения новой техники, технологии, использование которых может оказать воздействие на окружающую среду, организация должна получить на них положительное заключение государственной экологической экспертизы в порядке, установленном законодательством Российской Федерации.

11.5. Безопасность в чрезвычайных ситуациях

11.5.1. Анализ возможных ЧС, которые могут возникнуть при разработке в офисе

К вероятным ЧС на рабочем месте можно отнести: пожар, неисправность систем водоснабжения.

В следствии несоблюдения правил технической безопасности наиболее вероятным ЧС на рабочем месте можно отнести пожар в здании, связанный с коротким замыканием электропроводки.

Согласно Федеральному закону "Технический регламент о требованиях пожарной безопасности" от 22.07.2008 N 123-ФЗ (ред. от 30.04.2021) [18] каждое офисное помещение должно иметь порошковый и углекислотный огнетушители и систему пожарного

оповещения. В кабинетах, на этажах должна быть оборудована система оповещения и контроля эвакуации.

При возникновении пожара необходимо обратиться в службу спасения. Если не сработала сигнализация, необходимо оповестить персонал о возникновении пожара нажатием на кнопку сигнализации. В случае тушения электрооборудования применяются углекислотные огнетушители. Если имеется опасность поражения электротоком, помещение необходимо обесточить перед тем как приступить к тушению пожара.

Заключение по разделу «Социальная ответственность»

По продемонстрированным данным в разделе, было установлено, что значения всех производственных фактором на рабочем месте соответствуют нормам. В результате анализа и проведения расчета общего равномерного искусственного освещения методом коэффициента светового потока, было установлено, что помещение, в котором выполняются работы, удовлетворяет требованиям, установленными нормативными документами.

Категория помещения по электробезопасности согласно ПУЭ соответствует классу – «помещения без повышенной опасности, в которых отсутствуют условия, создающие повышенную или особую опасность» [19].

Согласно правилам по охране труда при эксплуатации электроустановок [20], персоналу следует присвоить группу I, которая относится к неэлектротехническому персоналу. Присваивание группы I происходит с оформление в журнале и проведением инструктажа, завершающимся проверкой знаний в форме устного опроса и проверкой приобретенных навыком безопасных способов работы и оказания первой медицинской помощи при поражении электрическим током.

Категория тяжести труда по СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания» [12] относится к lб – «Работы, производимые сидя, стоя или связанные с ходьбой и сопровождающиеся физическим напряжением категории помещений по взрывопожарной и пожарной опасности.

Согласно СП 12.13130.2009 «Определение категорий помещений, зданий и наружных установок по взрывопожарной и пожарной опасности»», категория помещения по взрывопожарной и пожарной безопасности относится к классу Д – «Негорючие вещества и материалы в холодном состоянии» [21].

Рассмотренный объект оказывает незначительно воздействие на окружающую среду и относится к классу III отходов опасности [22].

Список используемых источников

1. How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read. – [Электронный ресурс] URL: https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-createevery-day-the-mind-blowing-stats-everyone-should-read. Дата обращения: 02.04.2023.

2. Running tests utPLSQL. [Электронный ресурс] URL: https://www.utplsql.org/utPLSQL/v3.0.0/userguide/running-unit-tests.html. Дата обращения: 27.04.2023.

3. Introduction to Oracle Database Testing. [Электронный ресурс] URL: https://docs.oracle.com/en/database/oracle/oracle-database/21/ratug/introduction-to-oracle-database-testing.html#GUID-C6C2FA1F-3474-4B5A-B814-C41415B4C248. Дата обращения: 02.04.2023.

4. Testing best practices. [Электронный ресурс] URL: https://www.utplsql.org/utPLSQL/latest/userguide/best-practices.html. Дата обращения: 02.04.2023.

5. Система управления тестированием TetsIt. [Электронный ресурс] URL: https://testit.software/product. Дата обращения: 12.03.2023

6. Мартин Р., Чистая архитектура. Искусство разработки программного обеспечения, 2018

7. Фейерштейн С., Oracle PL/SQL. Программирование в Oracle для профессионалов, 2003.

8. Трудовым кодексом Российской Федерации от 30.12.2001 N 197-ФЗ (ред. от 19.12.2022, с изм. от 11.04.2023) (с изм. и доп., вступ. в силу с 01.03.2023)

9. Федеральный Закон от 27.07.2006 N 152-ФЗ «О персональных данных»

10. ГОСТ 12.12.032-78 ССБТ. Рабочее место при выполнении работ сидя

11. ГОСТ-21889-76. Система "Человек-машина"

12. СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания»

13. ГОСТ 12.0.003-2015 ССБТ. Опасные и вредные производственные факторы. Классификация

14. ТОИ Р-45-084-01 Типовая инструкция по охране труда при работе на персональном компьютере

15. ГОСТ 12.1.019-2017 Система стандартов безопасности труда. Электробезопасность

16. ГОСТ 12.3.031-83 ССБТ. Работы со ртутью

17. ГОСТ Р 70146-2022 Ресурсосбережение. Отходы электроники и электробытовой техники

18. Федеральному закону "Технический регламент о требованиях пожарной безопасности" от 22.07.2008 N 123-ФЗ (ред. от 30.04.2021)

19. Правило устройства электроустановок (ПУЭ)

20. Правил по охране труда при эксплуатации электроустановок (с изм. от 29.04.2022)

21. СП 12.13130.2009 Определение категорий помещений, зданий и наружных установок по взрывопожарной и пожарной опасности

22. Критерии отнесения объектов, оказывающих негативное воздействие на окружающую среду, к объектам I, II, III и IV категорий (с изм. от 7.10.2022)

Приложение А

The technical overview

Студент

Группа	ФИО	Подпись	Дата
8ПМ1И	Попова Мария Александровна		

Руководитель ВКР

Должность	ФИО	Ученая степень, звание	Подпись	Дата
доцент	Губин Евгений Иванович	к.ф.-м.н.		

Консультант-лингвист отделения иностранных языков ШБИП

Должность	ФИО	Ученая степень, звание	Подпись	Дата
доцент	Уткина Анна Николаевна	к.ф.н		

1. The general issue of big data testing

Today, the amount of data is growing dramatically. Due to the growing popularity of digital technologies, all data processes have gone digital, which is why their number has increased rapidly and dramatically in recent years [1].

When it comes to the functioning of banking systems, there is no doubt that the growth in the volume of data processed on a regular basis. Every day, banks process the data of millions of customers. All information is stored in a database and is regular used in processing the logic of the company's functionalities and products.

Traditional data processing methods are becoming irrelevant, as the demand for speed and reliability of their processing increases, as well as for fast and high-quality testing to support the stability of the company and business processes, which needs regular testing of all functional modules on a large amount of various data, which causes the need due to regular improvements and continuous development of customer services.

A mistake that made during software development leads to the appearance of defect or bug. When it comes to large-scale development of a project with a full of functionality, in order to avoid mistakes before releasing different kinds of projects to the main database, it is necessary to conduct mass and unit testing.

Huge companies need to cover most of their functionality with automated testing in order to increase the efficiency of software development, get avoid of trivial cases, and also eliminate the possibility of an error associated with the human factor.

It is impossible to provide regular manual testing on large amount of data by testing team without the development of automatic tests and a single company standard for its creation

The aim of the work is to develop a flexible methodology for automatic testing, designed to standardize the approach in coverage automatic tests of the maximum number of possible functionals in the language Oracle PL/SQL on big data arrays without creating integration functionality with platform TestIt for each test case, but only with the creation of a new automatic test on the Oracle database side. Development is designed to be configured and runned automatic case for the testing team in the interface of the TestIt service.

2. Types of testing

Test automation significantly reduces team costs developers, save time and resources that spent on developing new test cases, as well as reduce the release of a low-quality product to the market. In case of this, test automation technologies are becoming popularity among companies involved in the development of software for the product being released.

During the testing process, the program is examined in order to find possible errors, the result of which is a variance between the expected results from valid in the operation of the software product.

Big data testing involves the use of various tools, methods and frameworks for processing. Big data refers to data storage, analysis and testing, which differ in volume, variety and speed.

1. Testing a big data application is a test of both data processing and a test of the results of processing individual functionality of a software product.

2. Testing a big data application is a test of how data is processed and also check the results of processing individual functionals of the software product.

3. Testing process is divided into several types:

4. Unit testing – testing process individual methods and functions of classes, components or modules. Generally, unit tests do not require a lot of automation and can be performed very quickly with integration with the server.

5. Integration Testing - checking whether different modules and services work well together. You can test the interaction with the database or make sure that the microservices work with each other as intended.

6. Functional testing - only the initial result of some action is checked without an intermediate state of the system.

7. Mass testing means testing large amounts of data, many modules and interacting functionalities.

End-to-end testing is a test of user behavior when working with software in the context of the entire application. Allows you to provide control over any user actions in the application.

3. Possible problems during testing big data

In practice, testing is performed by executing a certain set of actions or scripts, obtaining the results of execution and further checking with the data that are defined as reference. The execution of this algorithm is possible both manually and automatically using various software solutions.

In this section, we will consider the specifics of the complexity of testing big data, as well as testing them in manual mode and the positive and negative aspects of switching to test automation.

A large amount of data from various sources requires more time:

1. to test compatibility with different platforms

2. for fast automation

3. to check and confirm the results of data processing

During testing a product, it is necessary to take into account possible mistakes associated with the human factor, since the person himself in this case generates the algorithm for the sequence of actions. When developing a new test case, it is necessary to involve an expert to write scripts.

Testers check the business logic of the product on each program execution node. Moreover, the check must be performed on each node before starting the next step in testing only one functional. Ultimately, it is necessary to evaluate the result obtained with the expected.

When working with mass testing of a large amount of data, that is, obtaining its expected and actual result for each client, it is impossible to ensure river testing without the development of automatic tests, as well as a single company standard.

When working with mass testing of a large amount of data, that is, obtaining for each client its expected and actual results, it is impossible to ensure manual testing without the development of automatic tests, as well as a single company standard.

A single standard that was developed that allows you to create automated tests for any existing or developed functionality, on any number of input data and launch parameters. Application functionality has been created on the side of the Oracle database, which provides integration interaction of automated tests with the TestIt service.

Automated testing has many advantages and has limitations. The strengths of automated testing include:

- execution speed, hundreds of thousands of times higher than human possibilities
- lack of influence of the human factor, this includes errors due to inattention and tiredness
- the ability to repeatedly run tests, thus reducing costs
- execution of test cases of special complexity
- the ability to analyze and compare data in a convenient format, especially important for mass testing, since it is not possible to manually check daily separately for millions of customers the expected result of their implementation and the actual

Disadvantages of automated testing include:

- the need to involve development to create universal testing systems
- financial costs and risks for automation tools and the complexity of their choice
- irrelevance of tests in case of changes in requirements

4. Existing testing approaches in the Oracle database.

Before developing a universal program in a flexible format that allows developing automatic tests on the side of the Oracle database with integration from TestIt, it was necessary to define a utility that will used in as an evaluation of the results of processing automatic tests.

1. *Oracle Real Application Testing*

The Oracle Real Application Testing option in Oracle Database helps you securely guarantee the integrity of database changes and manage test data. It allows you to perform real-time testing against an Oracle database. [2]

SQL Performance Analyzer and Database Replay are key components of Oracle Real Application Testing. Depending on the nature and impact of the system change being tested, and the type of system the test will run, one or both of the components can be used to perform the test.

System changes, such as updating a database or adding an index, can cause changes to the execution plans of SQL statements.

2.*utPLSQL*

utPLSQL implements testing principles adopted in Extreme Programming methodology. Automatically executes manually written test code and checks the results [3].

Benefits of the utility:

1. similarity with classical frameworks in other programming languages;

2. extensive functionality;

3. the ability to compare cursors with data to work with a large amount of resulting data, actual and expected.

Amongst many benefits, it provides ability to see the progress of test execution for long-running tests - real-time reporting use many reporting formats simultaneously and save reports to files map your project source files and test files into database objects

Test Isolation and Dependency [4]:

1. Tests should not depend on a specific order to run;

2. Tests should not depend on other tests to execute;

3. Tests should not depend on specific database state, they should setup the expected state before being run;

4. Tests should keep the environment unchanged post execution.

Important rules to keep in mind when developing an agile testing methodology:

1. tests should not imitate/duplicate the logic of the code under test;
2. tests should contain zero logic (or as close to zero as possible);
3. organize (set up inputs/data/environment for code under test);
4. action (execution of code under test);
5. approve (confirm execution results);
6. each tested procedure/function/trigger (code block) must have more than one test;
7. each test should check only one behavior (one requirement) of the code block under test;
8. tests should be maintained as carefully as production code;
9. every test needs to be built to fail, tests that do not fail when they need to are useless.

3. *Code Tester for Oracle*

A testing program with the highest level of test automation. The generated test code based on the behavior defined in the user interface, after which the program runs the tests and displays the results. The success/failure of each test is marked on the screen in green/red.

Code Tester creates a PL/SQL package that implements user-defined tests and includes any customization logic provided. The user can then move forward while the automated tests are running.

5. Using an external TestIt system for testing processes

Flexibility in development means customizing the launch of tests by specialists who do not have development skills. To create a customization on the Oracle database side, the TestIt integration environment was evaluated and selected – Russian test management system.

It is possible to create a test case during manual testing at the system. According to projects, all test scenarios are structured and stored with the ability to combine repetitive actions into common steps. It is possible to track the changes made and return to the previous version of the changes.

With automated testing, the creation of an autotest also occurs in real time. The advantage of the platform is the ability to integrate autotests on different frameworks, no matter what programming language the autotest is written in - TestIt will combine any tests from the repository and run it. However, for this it is necessary to register integration with the database, while creating your own framework, which is the goal of this work.

The developed integration functionality provides flexible configuration for launching automated tests developed according to the implemented standard by passing an arbitrary set of launch parameters to the database in JSON format.

It allows you to create automatic tests without modifications to the integration service, using the resources of a development team that has the necessary qualifications within the framework of the tested functionality.

When working with the integration platform, it is possible to enter a local description of the automatic test and its name. The parameters for substitution will be display in accordance with the possible parameters from the table in the database by passing them through the request as a JSON file.

6. Scope of the methodology

When working with big data in large companies with a huge client base, before sending the development to a combat base to customers, it is necessary thoroughly to test not only the innovation, but also related functionality that could be affected. Because these are millions of operations with millions of rows of data processing, it takes a lot of time, if, moreover, it is done manually.

When testing in this way, before release to the online base, it is also necessary to check every case, every client, for information that is returned after it is finalized.

Thus, we are developing a common approach that can use this development and the parameterization package so that it is convenient, fast and on any amount of data to develop tests.

The built-in architecture [5] is demonstrated at Figure 1, the idea of which is to create a table structure with automatic tests and their parameters, as well as a log to control the changes made.

Auxiliary packages *p_main_run*, *p_autotest_api* and *p_autotest_run*. The second one declares three helper functions and one procedure:

1. *get_proc_name* - a function that returns the name of the procedure from the table autotest by its id;

2. *get_parallel* will return the result of the ability to run this autotest in parallel with other autotests;

3. *count_is_run* returns the number of autotests that are in status execution;

4. *set_status* - sets the start and stop status of the autotest, Y and N respectively.

The *p_autotest_run* package, located separately and created to develop each test individually. The only you need is just to finalize developing this package when creating his own autotest by creating your own logic of test case, without recoding the logic of its work.

The diagram shows that, starting from the platform, a database request is made to a table called *proc_param*, receiving a response, it forms a JSON file, and its appearance is shown in Figure 2. You can see that the key is the test case id from the database, and the value is a nested dictionary consisting of the name of the input parameter for this test and its value.

An important point is the ability to run an automatic test both for a specific case - for example, for a specific bank account, and a mass launch for all accounts for a certain date. On the part of the TestIt platform, this is determined by skipping the step of entering parameters for the autotest, or entering only the required date. Moreover, on the Oracle side, this is the verification and analysis of the JSON file submitted for input to the parameterized procedure, or its absence at all.

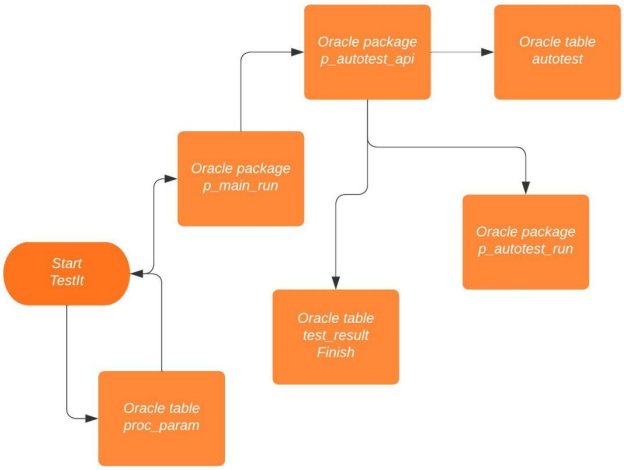


Figure 1 - Linking Automated Test Objects in Oracle Database

```
{id_test :
  {"param_1" : "value_1",
   "param_2" : "value_2"}
};
```

Figure 2 - JSON file structure

After receiving the JSON file on the side of the Oracle database the main package *p_main_run* is launching, in which, in addition to error handling procedures, there is a procedure that takes two parameters as input: the id of the automatic test and there are input parameters in the JSON file. Using the JSON library in Oracle, iterates over the parameters and its values in a «for loop». Combining into a single list and using dynamic SQL, a procedure is calling in connection with the «*autotest*» table, where the test id and the procedure designed to run it are stored.

Dynamic SQL is a tool in the field of application programming allowing simplifying the interaction of the application with the database by creating an SQL query directly from the program code.

Dynamic SQL allows the program to be more flexible, create and process SQL statements at runtime. This is the immediate execution of a dynamic SQL query or an anonymous PL/SQL block by using the execute immediate statement whose main argument is a string containing the SQL query to be executed. The big advantage that is used in this package is the ability to create a string using concatenation.

After working out the procedure, by contacting the table *test_result*, the calculated value is evaluating with the reference one, taking into account its error, which is also defined in the table. Data comparison row by row is performing using the utPlSql utility.

Conclusion

To sum up, among the huge variety of different testing applications only a one suits the research task. After studying the existing approaches to the development of automated tests, integration services were studied, and a single standard was developed that allows you to create automated tests for any existing and developed functionality in the Oracle database.

The aim of this overview is to help to describe the significance of development by comparing all types of testing with a description of their positive and negative sides. Citing big data testing as an example, we can conclude that a careful approach to the development of the corresponding functionality is necessary.

The following requirements that should be satisfied:

- the ability to integrate with an external platform for the possibility of simplifying the launch of automated tests and their visualization
- fast processing of large data sets
- flexible approach that excludes repetitive actions when developing new automatic tests

References

1. How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read. – URL: https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-createevery-day-the-mind-blowing-stats-everyone-should-read.

2. Introduction to Oracle Database Testing. URL: https://docs.oracle.com/en/database/oracle/oracle-database/21/ratug/introduction-to-oracle-database-testing.html#GUID-C6C2FA1F-3474-4B5A-B814-C41415B4C248.

3. Running tests utPLSQL. URL: https://www.utplsql.org/utPLSQL/v3.0.0/userguide/running-unit-tests.html.

4. Testing best practices https://www.utplsql.org/utPLSQL/latest/userguide/best-practices.html.

5. Feuerstein S., Oracle PL/SQL. For Professionals, 2014.

Приложение Б

package p_autotest_api

```
 function get_proc_name(p_id number) return varchar2
 is
  l_proc_name varchar2(200);
 begin
  select a.proc_name into l_proc_name from autotest a where a.id_test = p_id;
  return l_proc_name;
 exception
  when no_data_found then
    log(p_ls.c_log_error, 'Не найдена процедура выполнения с id = ' || p_id);
 end;

 function get_parallel(p_id number) return varchar2
 is
  l_status varchar2(1);
 begin
  select a.parallel into l_status from autotest a where a.id_test = p_id;
  return l_status;
 exception
  when no_data_found then
    log(p_ls.c_log_error, 'Не найдена процедура выполнения с id = ' || p_id);
 end;

 function count_is_run return number
 is
  l_run_now number;
 begin
  select count(*) into l_run_now from autotest a where a.is_run = 'Y';
  return l_run_now;
```

```
  end;
  procedure set_status(p_id number, p_status varchar2)
  is
  begin
    update autotest a
       set a.is_run = p_status
     where a.id_test = p_id;
  end;
```

package p_autotest_run

```
  procedure autotest1(p_param_1 varchar2, p_limit varchar2)
  is
    l_param varchar2(15);
  begin
    --- логика обработки конкретного тест-кейса для определенного функционала
/* скрыто */
  end;
```

package p_main_run

```
  procedure run_parse(p_id_test_run in number, p_json_clob in clob default null) is
    l_json json;
    l_keys brk.json_list;
    l_count number;
    l_proc_name varchar2(2000);
  begin
    if p_autotest_api.get_parallel(p_id_test_run) = 'Y' or p_autotest_api.count_is_run = 0 then --
тест можно запускать одновременно с другими
      p_autotest_api.set_status(p_id_test_run, 'Y'); --ставим статус, что начали работу с тестом
      commit;
      --получаем имя выполняемой процедуры автотеста
      l_proc_name := p_autotest_api.get_proc_name(p_id_test_run) ||'(';
```

```
    if p_json_clob is not null then
      l_json := json(p_json_clob);
      l_keys:= l_json.get_keys;
      for i in 1 .. l_keys.count loop
        l_proc_name := l_proc_name ||l_keys.get(i).str || ' => ' || l_json.get(i).str ||',';
      end loop;
      l_proc_name := trim(trailing ',' from l_proc_name) || ')';
    end if;
    execute immediate 'begin ut.run('''||l_proc_name ||'''); end;';
    p_ls.set_param(c_prf, 'last_run', to_char(sysdate, 'dd.mm.yyyy hh24:mi:ss'));
    p_autotest_api.set_status(p_id_test_run, 'N'); --закончили выполнение автотеста
  end if;
  commit;
exception
  when others then
    rollback;
    log(p_ls.c_log_error, 'Ошибка парсинга для теста id =  '||p_id_test_run
       ||chr(13)||chr(10)||sqlerrm||chr(13)||chr(10)||dbms_utility.format_error_backtrace);
end;
```