МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование кафедры)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки)

Технология программирования
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка BI-приложения для оценки страховых рисков

Студент А.А. Рябов (И.О. Фамилия) _____ (личная подпись)

Руководитель к.т.н, Т.Г. Султанов
(ученая степень, звание, И.О. Фамилия)

Консультант М.В. Дайнеко
(ученая степень, звание, И.О. Фамилия)

Тольятти 2020

Аннотация

Тема выпускной квалификационной работы – «Разработка BI-приложения для оценки страховых рисков».

Актуальность выпускной квалификационной работы обусловлена необходимостью повышения качества оценки страховых рисков.

Объектом исследования выпускной квалификационной работы является управление страховыми рисками.

Предметом исследования выпускной квалификационной работы является BI-приложение для оценки страховых рисков.

Цель выпускной квалификационной работы – разработка BI-приложения для оценки страховых рисков.

Методы исследования: математическое и программное обеспечение страховых BI-систем, методы и технологии разработки BI-приложений.

Разработана логическая архитектура BI-приложений и выполнена ее программная реализация на базе платформе 1С-Битрикс: Управление сайтом.

Тестирование BI-приложения подтвердило его работоспособность.

Результаты бакалаврской работы представляют практический интерес и могут быть рекомендованы для разработчиков BI-приложений для страховой деятельности.

Выпускная квалификационная работа состоит из 57 страниц текста с приложением, 22 рисунков, 4 таблиц и 29 источников.

Abstract

The topic of the given graduation work is Development of BI application for assessing insurance risks.

The relevance of the graduation work is due to the need to improve the quality of insurance risk assessment.

The object of study of the graduation work is insurance risk management.

The subject of study of the graduation work is BI application for assessing insurance risks.

The aim of the graduation work is to develop of BI application for impoving quality of insurance risk assessment.

Research methods: software for insurance BI-systems, methods and technologies for developing BI-applications.

The logical architecture and software of BI application on the base of 1C-Bitrix: Site Management platform were developed.

Testing of the developed BI application has confirmed its working capacity.

The results of the given graduation work are of practical interest and can be recommended for developers of BI-applications for insurance activities.

The graduation work consists of an explanatory note on 49 pages including 22 figures, 4 tables, the list of 29 references.

Оглавление

Введение 5
Глава 1 Анализ подходов к разработке BI-приложений для оценки страховых рисков 8
1.1 Методы оценки страховых рисков 8
1.2 Разработка требований к BI-приложению для оценки страховых рисков 11
1.3 Анализ подходов к архитектуре страховых BI-систем 12
1.3.1 Архитектура страховой BI-системы на основе OLAP-технологий 12
1.3.2 Архитектура страховой BI-системы на основе технологии Business Discovery 14
1.4 Анализ подходов к реализации BI-приложений для страхования 18
Глава 2 Моделирование BI-приложения для оценки страховых рисков 22
2.1 Методика построения BI-приложения 22
2.2 Разработка логической архитектуры BI-приложения 25
2.3 Разработка логической модели хранилища данных 26
2.4 Разработка модели ETL 29
Глава 3 Программная реализация BI-приложения для оценки страховых рисков 32
3.1 Выбор CMS для разработки BI-приложения 32
3.1.1 Система управления сайтом Drupal 32
3.1.2 Платформа «1С-Битрикс: Управление сайтом» 33
3.2 Архитектура реализации BI-приложения 35
3.3 Разработка программного обеспечения BI-приложения 38
3.4 Тестирование BI-приложения 43
3.5 Требование к аппаратно-программному обеспечению BI-приложения 46
Заключение 47
Список используемых источников и используемой литературы 48
Приложение А Фрагмент кода теста производительности BI-приложения 51

Введение

Одной из ключевых задач операционной деятельности страховой организации является оценка страхового риска.

Неправильная оценка страхового риска является причиной применения к договору страхования заниженного или завышенного тарифа, что в первом случае может привести к снижению показателей по страховой премии страховщика, а во втором – к недовольству клиента и отказу последнего от заключения договора с конкретным страховщиком.

Кроме того, неполная оценка страхового риска может привести к более серьезным последствиям, таким, как заключение договора с недобросовестным клиентом и, как следствие, к неоправданно большим страховым выплатам.

Для повышения качества оценки страхового риска и снижения влияния человеческого фактора на принятые решения используются приложения, разработанные на платформах бизнес-аналитики – BI (Business Intelligence).

Как показывает, практика наилучших результатов удается достичь при использовании BI-приложения, ориентированные на специфику операционной страховой деятельности у конкретного страховщика.

Таким образом, представляет **актуальность** разработка BI-приложения для оценки страховых рисков.

Объектом исследования бакалаврской работы является управление страховыми рисками.

Предмет исследования – BI-приложение для оценки страховых рисков.

Цель бакалаврской работы – разработка BI-приложения для оценки страховых рисков.

Для достижения данной цели необходимо выполнить следующие задачи:

– проанализировать подходы к разработке BI-приложений для оценки страховых рисков, а также функциональные и архитектурные особенности существующих ИТ-решений;

– разработать логическую модель BI-приложения для оценки страховых рисков;

– разработать программное обеспечение BI-приложения для оценки страховых рисков;

– выполнить тестирование разработанного BI-приложения оценки страховых рисков и подтвердить его соответствие установленным требованиям.

Методы исследования – математическое обеспечение страховых BI-систем, методы и технологии разработки BI-приложений.

Практическая значимость бакалаврской работы заключается в разработке комплекса автоматизированных тестов, обеспечивающих повышение эффективности тестирования веб-приложения.

Данная работа состоит из введения, трех глав, заключения, списка используемой литературы и используемых источников.

Первая глава посвящена анализу подходов к разработке BI-приложений для оценки рисков. Рассмотрены подходы к реализации BI-приложений для оценки страховых рисков. Произведен сравнительный анализ существующих ИТ-решений оценки страховых рисков.

Во второй главе описан процесс моделирования BI-приложения для оценки страховых рисков. Разработаны логическая архитектура и модель данных BI-приложения.

Третья глава посвящена реализации BI-приложения для оценки страховых рисков. Выбрана платформа для разработки программного обеспечения BI-приложения. Описаны процессы разработки и тестирования программного обеспечения BI-приложения.

В заключении описываются результаты выполнения выпускной квалификационной работы.

В приложении представлен фрагмент кода теста производительности BI-приложения.

Результаты бакалаврской работы представляют практический интерес и могут быть рекомендованы для разработчиков BI-приложений для страховой деятельности.

Бакалаврская работа состоит из 57 страниц текста, 22 рисунков, 4 таблиц и 29 источников.

Глава 1 Анализ подходов к разработке BI-приложений для оценки страховых рисков

1.1 Методы оценки страховых рисков

Согласно современному определению, платформы Business Intelligence (BI) позволяют предприятиям и компаниям создавать BI-приложения для решения задач в трех категориях: анализ, например, интерактивная аналитическая обработка; доставка информации, например, отчетов и информационных панелей; и интеграция платформ, например, управление метаданными BI и среда разработки [21].

В страховой деятельности BI-решения используются для предоставления отчетности, инструментальных панелей, технологий интеллектуального анализа данных и прогностического моделирования для анализа данных о страховых рисках и убытках, которые обеспечивают поддержку принятия управленческих решений, направленных на повышение эффективности страховой деятельности.

Иными словами, математическое обеспечение страховых BI-решений должно обеспечивать поддержку задач анализа убыточности и оценки страховых рисков на основе аналитической отчетности и прогнозных моделей.

Рассмотрим математическое обеспечение BI-систем анализа рисков КАСКО (добровольного автострахования).

Основным риском для данного вида страхования является риск возникновения дорожно-транспортного происшествия (ДТП) [13].

Для анализа страхового риска КАСКО страховые компании, как правило, используют собственные методики.

Как показывает практика, наиболее эффективной считаются методики, основанные на совместном анализе коэффициента убыточности по договорам

конкретного клиента и его коэффициента бонуса-малуса (КБМ) по ОСАГО [10, 20].

Коэффициент убыточности (Loss Ratio) по договорам страхования определяется по формуле:

$$K_{уб} = S_{уб}/S_{зп}, \quad (1)$$

где:

$S_{уб}$ – сумма понесенных убытков и расходов по регулированию убытков по договорам конкретного клиента за определенный период (3-5 лет);

$S_{зп}$ – заработанная премия по договорам о клиента за тот же период.

Рекомендации для принятии риска на страхования P вырабатываются на основе выражения [22]:

$$P = f(K_{уб}, B), \quad (2)$$

где:

B – количество страховых случаев по закончившимся договорам страхования клиента.

Функция P представлена в виде шкалы оценки риска (таблица 1)

Таблица 1 – Шкала оценки риска

$K_{уб}$	B	P
1.0 –2.0	< 3	Рекомендовано использовать при расчете страхового тарифа коэффициент 1,2-1,4
2.1 – 3.0	3	Рекомендовано использовать при расчете тарифа коэффициент 1,5-1,7
> 3 или >3		Рекомендовано отклонить заявление

Учетно-аналитическая информация, которая является входной для страхового BI-приложения, относится к категории финансовой и управленческой информации.

Можно выделить следующие свойства входной информации страховой BI-системы:

– генерируется различными источниками данных (рисунок 1) [14];

Рисунок 1 – Основные источники данных страховой BI-системы (пример)

– в основном структурирована;
– хронологически упорядочена;
– имеет относительно небольшие объемы (<10 Тб).

Следует учесть, что разнородность источников генерации страховой учетно-аналитической информации требует ее консолидации на входе BI-системы.

Для обеспечения полноты консолидированной информации необходимо обеспечить обогащение данных с помощью гибких алгоритмов.

Выходной информацией BI-приложения является аналитическая отчетность, используемая для поддержки принятия решения для управления страховыми рисками.

Для повышения эффективности процесса анализа используется метод «slice and dice», в котором некоторое количество информации делится на более мелкие части, особенно для более тщательного или разборчивого анализа [17].

1.2 Разработка требований к BI-приложению для оценки страховых рисков

Для разработки требований к BI-приложению используем модель FURPS+.

FURPS+ - это акроним, описывающий расширенную модель для классификации атрибутов качества программного обеспечения [19].

FURPS+ довольно широко используется в индустрии разработки программного обеспечения.

Опишем основные требования к BI-приложению для оценки страховых рисков, используя основные положения данной методологии [15].

1) Functionality (функциональность):
- работа в режиме онлайн;
- регистрация пользователей с разграничением прав доступа;
- поддержка задач анализа убыточности и оценки страховых рисков;
- формирование аналитической отчетности.

2) Usability (удобство использования):
- дружественный интерфейс;
- простота освоения;
- отсутствие функциональной избыточности.

3) Reliability (надежность):
- количество одновременно работающих пользователей – 3;
- допустимая частота/периодичность сбоев: 1 раз в 300 часов;
- среднее время сбоев: 1 раб. день;
- возможность восстановления системы после сбоев: 1 раб. день;
- режим работы 7/24/365.

4) Performance (производительность):
- допустимое количество одновременно работающих пользователей: 10;
- время реакции на возникновение аварийной ситуации – 10 с.

5) Supportability (поддержка):

− низкая стоимость владения приложением;

− простота интеграции в корпоративную информационную систему (КИС) страховой компании;

− простота настройки под специфику оценки риска конкретным страховщиком;

− время устранения критических проблем: в течение рабочего дня.

− Проектные ограничения:

− вычислительная архитектура «клиент-сервер»;

− разработка на основе современной CMS-платформы.

Проанализируем известные подходы к разработке страховых BI-систем на предмет реализации BI-приложения, соответствующего вышеперечисленным требованиям.

1.3 Анализ подходов к архитектуре страховых BI-систем

Рассмотрим известные подходы к архитектуре страховых BI-систем.

1.3.1 Архитектура страховой BI-системы на основе OLAP-технологий

OLAP (Online Analytical Processing) - технология, лежащая в основе многих BI-приложений. Это мощная технология обнаружения данных, включающая возможности неограниченного просмотра отчетов, сложные аналитические вычисления и прогнозное планирование сценариев «что-если» (бюджет, прогноз).

В отличие от реляционных баз данных инструменты OLAP не хранят отдельные записи транзакций в двумерном, построчном формате, например, в форме таблицы, а вместо этого используют многомерные структуры баз

данных, известные как многомерные кубы, для хранения массивов консолидированной информации.

Архитектура типовой страховой BI-системы на основе OLAP-технологий представлена на рисунке 2 [6].

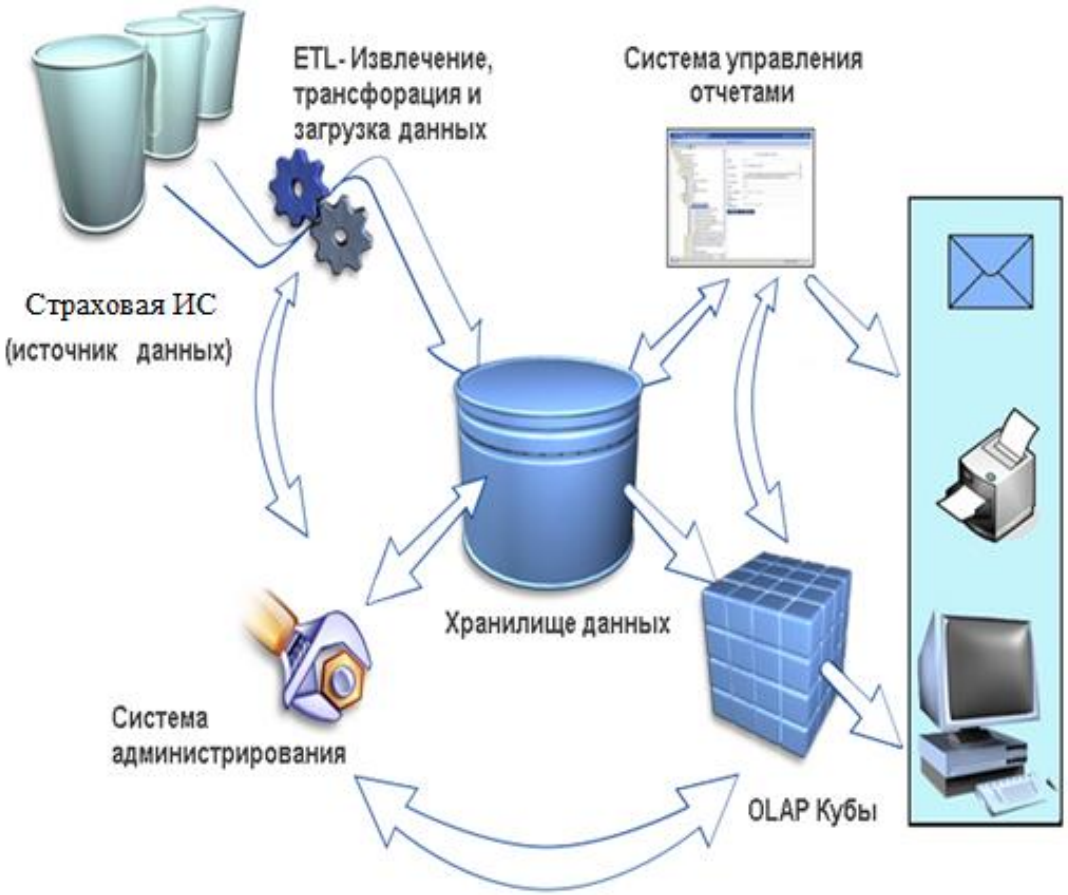

Рисунок 2 – Архитектура типовой BI-системы для страховой деятельности

Ключевыми компонентами BI-системы являются ETL (Extract-Transform-Load, Извлечение-Трансформация-Загрузка) и хранилище данных (Data Warehouse).

Хранилище данных - это архитектура хранения, предназначенная для хранения данных, извлеченных из OLTP-систем, хранилищ оперативных данных и внешних источников. Затем хранилище объединяет эти данные в агрегированную сводную форму, подходящую для анализа данных на уровне

предприятия и составления отчетов по заранее определенным бизнес-потребностям.

Хранилище данных (ХД) содержит данные, сгруппированные в абстрагированные предметные области с изменяющимися во времени версиями одних и тех же записей, с соответствующим уровнем детализации или детализации данных, чтобы сделать его полезным для двух или более различных типов анализа.

Разновидность ХД - витрина данных (Data mart) содержит аналогично изменяющиеся во времени и предметно-ориентированные данные, но с отношениями, подразумевающими использование данных измерений, в которых факты четко отделены от данных измерений, что делает их более подходящими для отдельных категорий анализа.

ETL представляет собой интегрированный инструмент для извлечения данных из одной базы данных и помещения их в ХД BI-системы.

Извлечение - это процесс чтения данных из базы данных. На этом этапе данные собираются, часто из разных источников.

Преобразование - это процесс преобразования извлеченных данных из их предыдущей формы в форму, в которой они должны находиться, чтобы их можно было поместить в ХД. Преобразование происходит с использованием бизнес-правил и/или справочных таблиц путем консолидации данных из разных источников.

Загрузка - это процесс записи данных в целевое ХД.

Для анализа страховых рисков в основном используются BI-приложения, построенные на модели хранения данных ROLAP (реляционные OLAP).

1.3.2 Архитектура страховой BI-системы на основе технологии Business Discovery

Business Discovery – это технологии построения BI-систем, предложенная компанией QlikView [16].

QlikView - это аналитическая платформа, реализующая ассоциативную архитектуру с обработкой данных в оперативной памяти. Управление взаимосвязями между данными осуществляется не на прикладном уровне, а на уровне внутренних механизмов платформы.

QlikView хранит в оперативной памяти отдельные таблицы данных и ассоциативные связи между ними, построенные по принципы полей с одинаковыми названиями. Каждое значение каждого поля связано со всеми остальными значениями во всей базе данных. Наборы данных могут состоять из сотен таблиц с тысячами полей. Базы данных приводятся к третьей нормальной форме (рисунок 3).

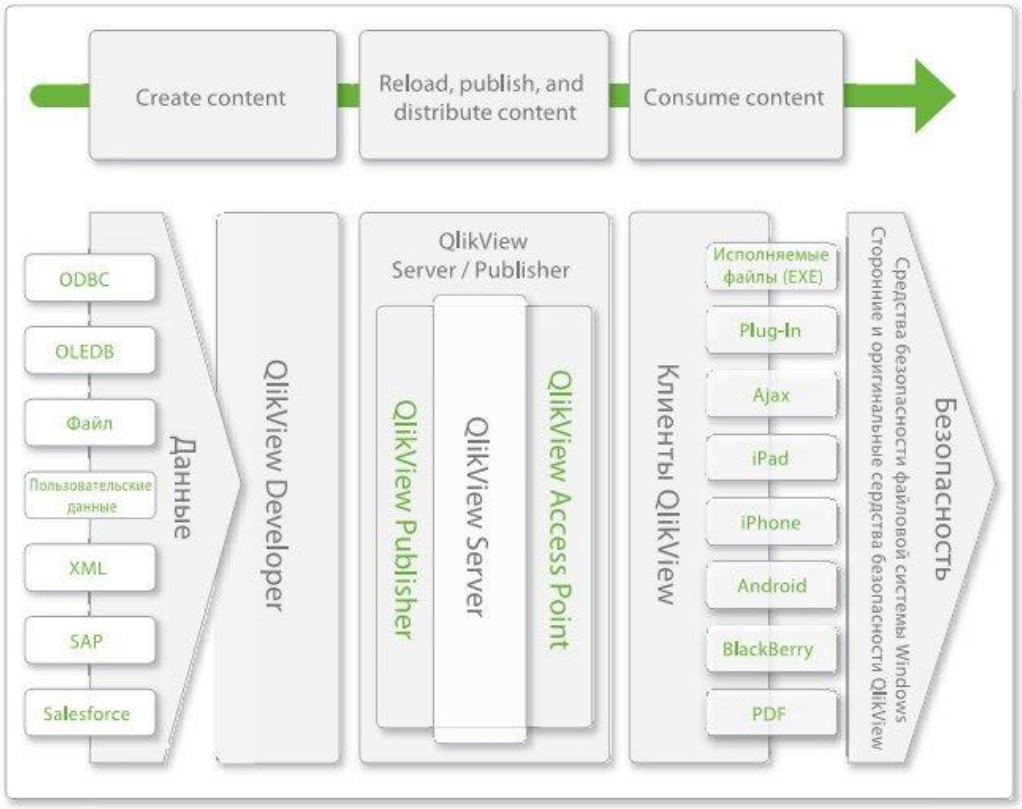

Рисунок 3 – Архитектура BI-решения QlikView

Иными словами, в технологии Business Discovery не используются OLAP-кубы и хранилища данных. В BI-системе используются

транзакционные данные, которые хранятся в OLTP-подсистеме. По мнению разработчиков, это снимает необходимость постоянно изменять модель/архитектуру данных.

При этом успешно решаются бизнес-задачи, а новая отчетность создается на уровне визуализации данных.

Платформа QlikView состоит из нескольких компонентов, отвечающих соответственно за создание контента QlikView (QlikView Developer), его доставку (QlikView Server/Publisher) и использование (клиентские приложения QlikView)

При большем объеме данных и увеличенном количестве пользователей целесообразно использование клиент-серверной архитектуры, предполагающей обработку данных на сервере и передачу результатов вычислений клиентам по сети.

Существуют два серверных продукта: QlikView Small Business Edition Server и QlikView Enterprise Edition Server.

QlikView Server обеспечивает:

– многопользовательскую работу и оптимизацию использования оперативной памяти;

– безопасное централизованное хранение данных;

– централизованное обновление данных во всех приложениях QlikView;

– единую точку доступа к QlikView-приложениям для всех пользователей;

– удаленную работу в сетях Интернет/интранет через различные типы QlikView Client;

– авторизацию и аутентификацию пользователей.

QlikView предлагает широчайший выбор клиентов: стандартный Windows.exe, plug-in для Microsoft Internet Explorer, тонкий AJAX-клиент и Java-клиент.

QlikView может работать на мобильных устройствах: версия QlikView Java Mobile Edition (мобильные телефоны с Java и коммуникаторы, включая BlackBerry) и версия QlikView for iPhone – первое BI-приложение для iPhone.

Расширение для серверного варианта под названием QlikView Publisher обеспечивает полный контроль за доставкой документов QlikView пользователям, автоматизируя процесс централизованного обновления данных. Данный модуль также обеспечивает персонализацию приложений для пользователей и своевременную рассылку пользователям отчетов в формате PDF.

Разработчики могут подключаться к любым источникам данных и объединять получаемую из них информацию, используя внутренние ETL-скрипты.

Скрипты QlikView поддерживает более 200 различных функций для обработки, связывания и очистки исходных данных для создания в QlikView резидентной модели данных. Пользователи могут легко загружать данные из любых ERP-систем, хранилищ данных, баз данных операционных систем и любых других источников данных, таких как текстовые файлы, документы Microsoft Excel, файлы XML или данные, получаемые из веб-сервисов.

Вместе с тем, по мнению некоторых аналитиков, решение QlikView имеет меньшую скорость обработки запросов и ограничения по обработки больших массивов данных.

На отечественном страховом ИТ-рынке представлен продукт QlikView for Insurance, реализованный на основе технологии Business Discovery.

Для сравнения рассмотренных подходов составлена таблица 2.

Таблица 2– Сравнение подходов к архитектуре страховых BI-систем

Характеристика	Архитектура на основе OLAP-технологий	Архитектура на основе технологии Business Discovery
поддержка задач анализа убыточности и оценки страховых рисков	+	+

Продолжение таблицы 2

низкая стоимость владения приложением	+	-
вычислительная архитектура «клиент-сервер»	+	+
простота интеграции с КИС СК	+	-
Итого	4	2

Таким образом, для BI-приложения для оценки страховых рисков более предпочтительной является архитектура на основе OLAP-технологий.

1.4 Анализ подходов к реализации BI-приложений для страхования

Как показал анализ, в настоящее время наиболее эффективным подходом к реализации страховых BI-приложений является применение промышленных BI-платформ.

Здесь можно выделить страховые аналитические системы, реализованные на платформе «1С: Предприятие 8» (рисунок 4) [12].

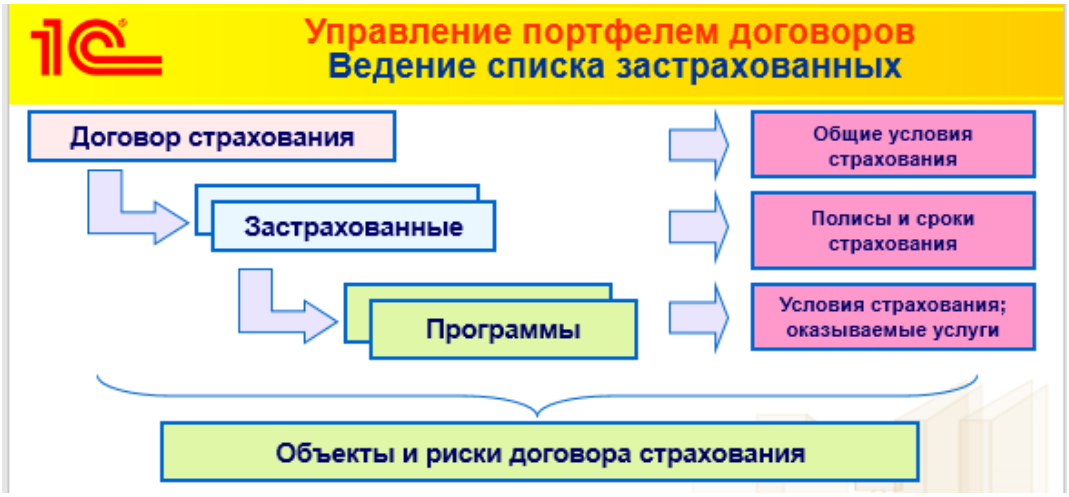

Рисунок 5 – Структурная схема блока «Управление портфелем договоров» ПП «1С:Управление страховой компанией 8»

Так, в программном продукте (ПП) «1С:Управление страховой компанией 8» имеется блок «Управление портфелем договоров», в котором

реализован механизм оценки страховых рисков при заключении договоров страхования.

Среди зарубежных BI-платформ помимо описанной выше платформы QlikView следует выделить платформу SAP BI.

Архитектура платформы SAP BI представлена на рисунке 4.

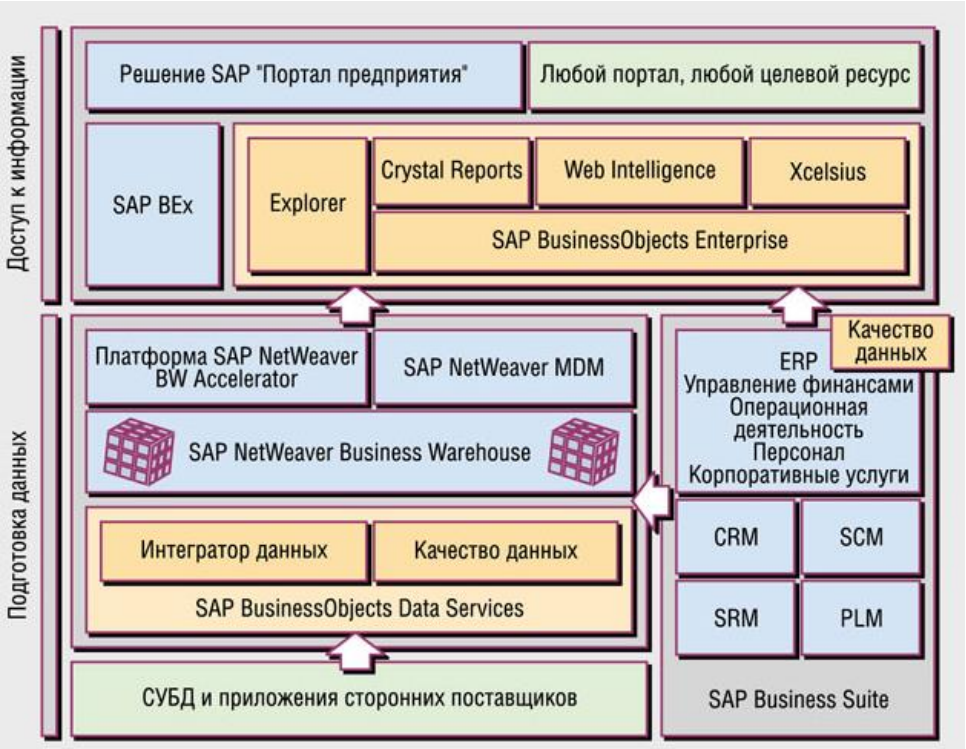

Рисунок 4 - Архитектура платформы SAP BI

Уровень платформы BI содержит службы BI для поддержки сложных задач и функций анализа [26].

Платформа содержит механизм Analytic Engine, который обрабатывает данные, запрашиваемые с помощью навигационных систем BEx. Интерфейс позволяет вводить данные и манипулировать ими как часть интегрированного планирования BI.

Имеются специальные инструменты анализа, такие как Analysis Process Designer (APD) и Data Mining, которые предоставляют аналитикам страховой компании инструменты для консолидации, предварительной обработки, хранения и анализа данных.

На основе данной платформы разработано ИТ-решение SAP «Бизнес-аналитика для страхования» (SAP BI for Insurance).

SAP BI for Insurance – это полностью интегрированное решение, которое обеспечивает сбор, хранение и обработку бизнес-информации для создания точной картины деятельности страховой компании и своевременного принятия обоснованных управленческих решений.

Оно полностью соответствует современным требованиям, предъявляемым к аналитическим системам поддержки принятия решений - отличается открытой расширяемой архитектурой, одинаково хорошо работает как с приложениями SAP, так и с продуктами других поставщиков.

Мощные инструменты Business Objects для извлечения, трансформации и загрузки данных (ETL) и удобные средства построения отчетов и визуализации выходных данных в совокупности с хранилищем SAP BW, производительность которого масштабируемо для любых объемов бизнеса, делает аналитические приложения SAP идеальным решением для страховых компаний любого размера.

Следует отметить, общей проблемой платформенных решений является сложность интеграции с КИС, построенной на другой платформе.

Для сравнения рассмотренных подходов составлена таблица 3.

Таблица 3 – Сравнение подходов к реализации страховых BI-систем

Характеристика	1С:Управление страховой компанией 8	SAP BI
работа в режиме онлайн	+	+
низкая стоимость владения приложением	-	-
вычислительная архитектура «клиент-сервер»	+	+
применение веб-технологий	-	-
Итого	2	2

Как показал анализ, в представленных подходах не используются веб-технологии, поэтому принято решение разработать BI-приложение на промышленной CMS.

Выводы к главе 1

1. Математическое обеспечение страховых BI-решений должно обеспечивать поддержку задач анализа убыточности и оценки страховых рисков на основе аналитической отчетности и прогнозных моделей.

2. Входная информация страхового BI-приложения относится к категории финансовой и управленческой информации.

3. Выходная информация страхового BI-приложения представляет собой аналитическую отчетность.

4. Для разработки архитектуры страховых BI-систем используются два подхода: на основе OLAP-технологий и технологии Business Discovery компании QlikView. Как показал анализ, для BI-приложения для оценки страховых рисков использование архитектуры на основе OLAP-технологий является более предпочтительным.

5. Известные платформенные BI-приложения имеют высокую стоимость владения. Кроме того, в них не применяются веб-технологии, поэтому принято решение разработать новое BI-приложение на промышленной CMS.

Глава 2 Моделирование BI-приложения для оценки страховых рисков

2.1 Методика построения BI-приложения

Методика построения BI-приложения (модель жизненного цикла) состоит из следующих этапов (рисунок 5) [23, 27]:

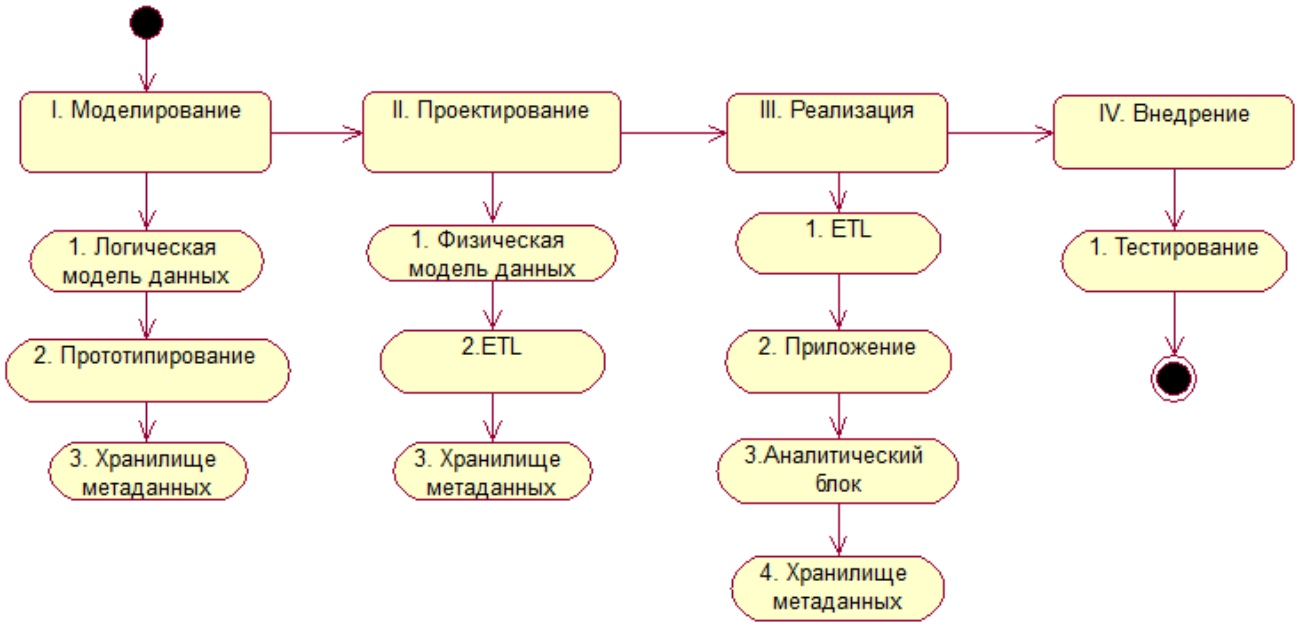

Рисунок 5 – Модель методики построения BI-приложения

Этап 1. Моделирование.

Шаг 1. Логическое моделирование данных.

Данный этап включает в себя идентификацию и проектирование источников данных, разработку детальных ER-диаграмм с атрибутами и связями между данными. Результатом является логическая модель данных.

Шаг 2 - Прототипирование приложения.

Создается и тестируется первоначальный прототип для проверки потребностей бизнеса. После тестирования результаты оцениваются с указанием положительных и отрицательных аспектов.

Шаг 3 - Модель хранилища метаданных.

Технические метаданные должны быть сопоставлены с бизнес-метаданными и затем перенесены в хранилище метаданных. Все метаданные, которые вводятся и хранятся в хранилище, должны быть документированы в модели логических метаданных. Кроме того, все требования к метаданным должны быть доступны пользователю через функцию интерактивной справки.

Этап 2. Проектирование приложения.

Шаг 1. Физическое моделирование данных.

На данном этапе логическая модель детализируется и уточняется, и на ее основе разрабатывается физическая модель под конкретную СУБД. Выбирается модель данных для обработки и хранения аналитической информации: реляционная, объектно-ориентированная или многомерная.

Шаг 2. Разработка процесса ETL.

Процесс ETL, также известный как процесс интеграции данных, является наиболее сложным элементом всего проекта. По его качеству можно судить об успешности реализации и деловой полезности всего проекта.

На уровень сложности процесса влияют следующие элементы:

- количество и качество источников учетно-аналитической информации;
- доступ к исходным данным;
- область информации, включенная в проектируемую систему;
- количество и степень сложности выходной отчетности;
- требования пользователей системы (количество и степень сложности специального отчета).

Рекомендуется, чтобы ETL-процесс был построен в одной среде, которая объединяет все модули организации, а не по отдельности, в каждом отделе. Правило должно быть: использовать один согласованный процесс ETL.

Шаг 3. Разработка хранилища метаданных.

Если используется готовое решение для хранилища метаданных, то на этом этапе оно настраивается в соответствии с требованиями проекта, в противном случае хранилище метаданных проектируется с точки зрения логической модели метаданных в зависимости от выбранной модели данных: реляционной, объектно-ориентированной или многомерной.

Этап 3. Реализация.

Шаг 1. Реализация ETL.

Для реализации процесса ETL используются инструменты фильтрации, процедуры и операторы. Фильтрация и преобразование данных зависит от качества источников данных. Эти источники различны, такие как: файлы, базы данных, электронная почта, Интернет и др.

Шаг 2. Реализация приложения.

Если прототип приложения удовлетворяет функциональным системным требованиям, можно начинать работу по разработке среды доступа и анализа. Разработанное приложение может быть простым преобразованием прототипа в полнофункциональную систему или гораздо более сложным новым проектом, основанным на различных ИТ-инструментах. Однако в обоих случаях этот этап выполняется одновременно с реализацией процесса ETL.

Шаг 3. Аналитический блок.

Этот этап включает в себя программную реализацию и тестирование алгоритмов и используемых методов анализа данных.

Шаг 4. Реализация хранилища метаданных.

На данном этапе разрабатываются словарь метаданных и интерфейсы доступа к данным.

Этап 4. Внедрение.

Внедрение - это процесс организации тренинга для менеджеров, подготовки окончательной документации, обеспечения технической поддержки, завершение процесса загрузки данных и настройки приложения.

Шаг 1. Тестирование.

Рекомендуется выполнить функциональное тестирование BI-приложения.

По результатам тестирования составляется окончательный отчет, в котором описываются характеристики приложения, а также некоторые его элементы, которые необходимо улучшить или перестроить.

Представленная методика принята в качестве базовой методики построения BI-приложения оценки страховых рисков.

2.2 Разработка логической архитектуры BI-приложения

Логическая модель информационной системы (ИС) - это статическое представление объектов и классов, которые составляют пространство проектирования и анализа ИС.

Другими словами, логическая модель, как модель классов является более строгой и ориентированной на дизайн моделью ИС [28].

Понятие логической модели ИС тесно связано с ее логической архитектурой.

Логическая архитектура - это структурная схема, которая позволяет разработчику показать как можно больше деталей ИС без привязки к конкретной технологии или среде, например, связи между компонентами программного обеспечения ИС.

Для логического моделирования BI-приложения используем технологии, основанные на применении языка визуального моделирования UML [8].

UML - это стандартный язык для определения, визуализации, конструирования и документирования артефактов программных систем.

UML не является языком программирования, но его инструменты могут использоваться для генерации кода на разных языках с использованием диаграмм UML.

В основу UML положена объектно-ориентированная концепция проектирования информационных систем, реализуемая с помощью набора диаграмм.

Диаграмма UML - это частичное графическое представление модели системы, находящейся в процессе разработки, реализации или уже существующей.

Для разработки логической архитектуры BI-приложения оценки страховых рисков используем диаграмму компонентов UML.

Диаграмма компонентов иллюстрирует части программного обеспечения, встроенные контроллеры и тому подобное, составляющие систему, а также их организацию и зависимости.

Диаграммы компонентов UML являются одним из двух видов диаграмм для моделирования физических аспектов объектно-ориентированных программных систем. Диаграмма компонентов показывает организацию и зависимости между набором программных компонентов ИС. Она используется для моделирования статического представления реализации программной системы.

На рисунке 6 представлена логическая архитектура BI-приложения для оценки рисков.

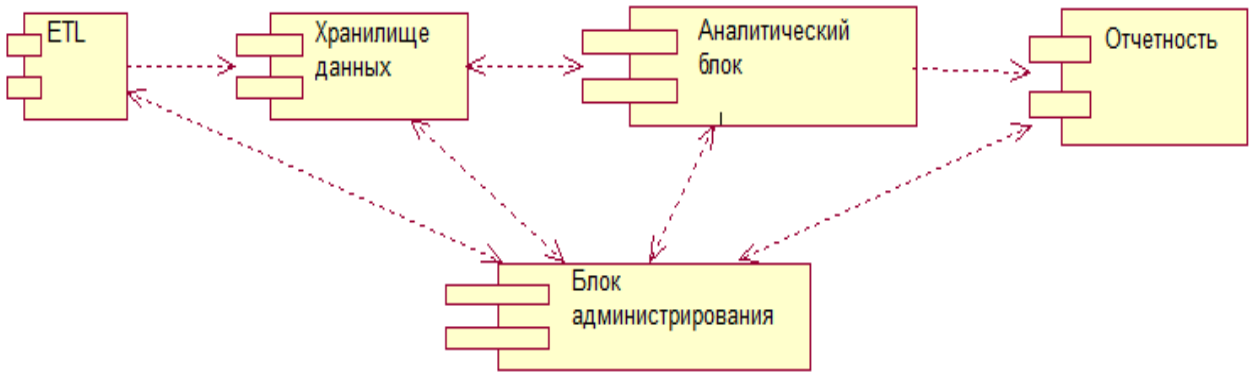

Рисунок 6 – Логическая архитектура BI-приложения для оценки рисков

2.3 Разработка логической модели хранилища данных

Для логического моделирования данных BI-приложения оценки страховых рисков используем методологию построения страховых систем на основе объектно-структурного подхода [9].

В данной методологии рекомендуется использовать в качестве элементов построения моделей страховых систем паттерны проектирования на UML, созданные как объектно-ориентированное представление реальных или виртуальных классов технологической онтологии страховой деятельности.

Предварительно необходимо выбрать технологию построения хранилища данных BI-приложения.

С учетом известных рекомендаций, особенностей ведения операционной страховой деятельности и построения страховых информационных систем выбираем технологию ROLAP (Relational OLAP) [26].

Реляционная онлайн-аналитическая обработка (ROLAP) – это форма онлайн-аналитической обработки (OLAP), которая выполняет динамический многомерный анализ данных, хранящихся в реляционной базе данных, а не в многомерной базе данных (которая обычно считается стандартом OLAP).

Принципиальным преимуществом ROLAP для страховой BI-системы является возможность формирования операционной аналитической отчетности на основе реляционной базы данных (БД), которая используется в операционной страховой ИС.

В качестве схемы данных используется схема «звезда», которая обеспечивает более высокую производительность обработки запросов.

Ввиду того, что аналитическая отчетность по страховым рискам ориентирована конкретно на поддержку задач управления договорами страхования, представляется более эффективным в качестве хранилища данных использовать витрину данных отдела продаж страховых полисов.

Для разработки объектной модели витрины данных используем диаграмму классов UML.

Диаграмма классов изображает статическое представление приложения. Он представляет типы объектов, находящихся в системе, и отношения между ними.

Класс состоит из своих объектов, а также он может наследовать от других классов.

Диаграмма классов используется для визуализации, описания, документирования различных аспектов системы, а также для построения исполняемого программного кода.

Она показывает атрибуты, классы, функции и отношения, чтобы дать общее представление о системе программного обеспечения.

На рисунке 7 представлена диаграмма классов витрины данных BI-приложения для страховой деятельности на основе ROLAP, построенного по схеме «звезда».

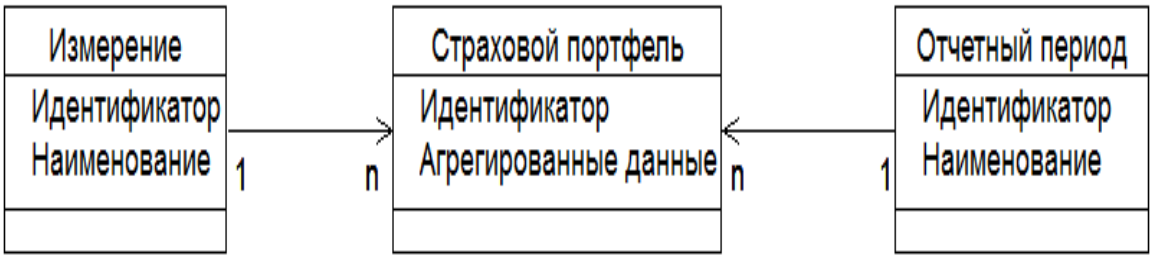

Рисунок 7 – Диаграмма классов витрины данных BI-приложения для страховой деятельности

Таблицы фактов и измерений реализуется на основе паттернов проектирования «Страховой портфель» и «Измерение», соответственно.

С учетом (1) в качестве таблицы фактов используется таблица «Страховой портфель клиента».

В качестве таблицы измерения используется таблица «Виды страхования».

С учетом вышеизложенного разработана логическая модель витрины данных BI-приложения.

Для разработки логической модели данных использована среда MySQL Workbench [24].

MySQL Workbench — это унифицированный визуальный инструмент для архитекторов и разработчиков БД, использующих в своих проектах реляционную СУБД MySQL.

MySQL Workbench предоставляет возможность моделирование данных, разработку SQL и комплексные инструменты администрирования для конфигурации сервера, администрирования пользователей, резервного копирования и др.

На рисунке 8 изображена логическая модель витрины данных оценки страховых рисков по договору страхования клиента (ROLAP, «звезда»).

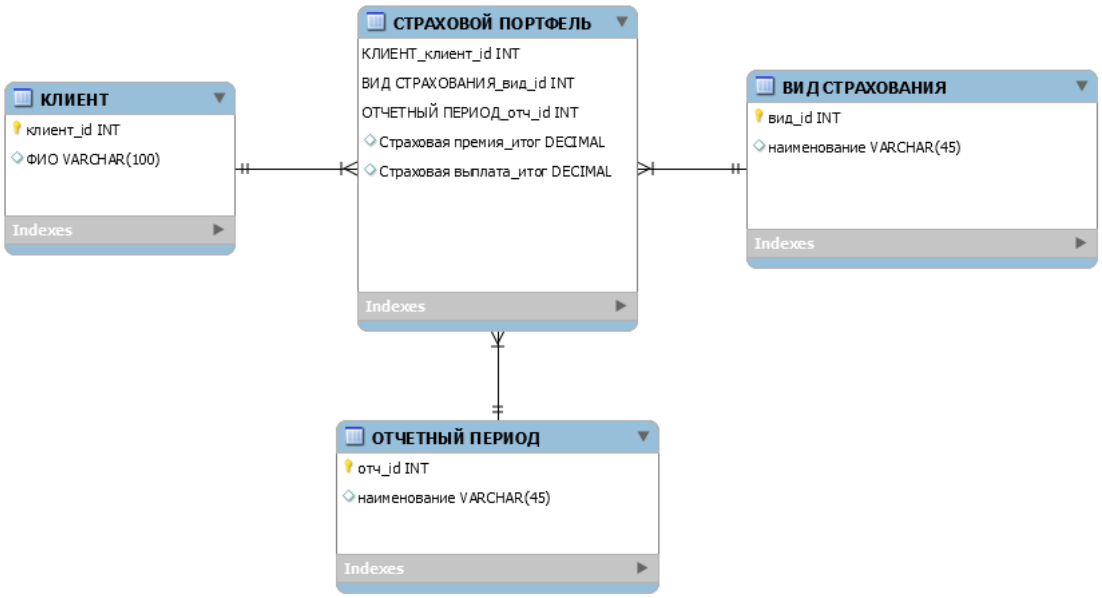

Рисунок 8 - Логическая модель витрины данных

В соответствии с правилами построения такой модели все связи идентифицирующие, «один-ко-многим».

Разработанная логическая модель является основой для реализации хранилища данных BI-приложения оценки страховых рисков.

2.4 Разработка модели ETL

ETL (extract, transform, load) является аббревиатурой от извлечения, преобразования, загрузки - трех функций БД, которые объединены в один инструмент для извлечения данных из базы данных страховой системы и помещения их в хранилище данных [5].

Извлечение - это процесс чтения данных из базы данных. На этом этапе данные собираются, часто из разных источников.

Преобразование - это процесс преобразования извлеченных данных из их предыдущей формы в форму, в которой они должны находиться, чтобы их можно было поместить в другую базу данных. Преобразование происходит с использованием правил или справочных таблиц, а также путем объединения данных с другими данными. Загрузка - это процесс записи данных в целевое хранилище данных.

Для отражения функционального аспекта разрабатываемой ETL используем диаграмму вариантов использования. Диаграмма вариантов использования является одной из базовых диаграмм языка UML. Она инкапсулирует функциональность системы, включая варианты использования, участников (акторов) и их отношения. Она моделирует задачи, сервисы и функции, требуемые системой / подсистемой приложения, а также показывает, как пользователь обращается с системой.

Диаграмма вариантов использования процесса ETL изображена на рисунке 9.

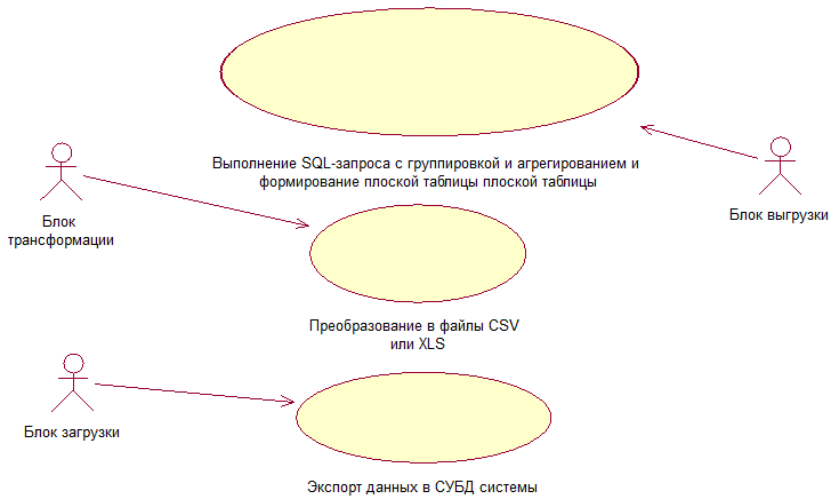

Рисунок 9 – Диаграмма вариантов использования процесса ETL

Акторами на данной диаграмме являются блок выгрузки, блок трансформации и блок загрузки.

Источником данных ETL является реляционная БД.

Представленное решение ETL упрощает процесс интеграции BI-приложения со страховой ИС.

Выводы к главе 2

1. Для логического моделирования данных BI-приложения оценки страховых рисков применяется методология построения страховых систем на основе объектно-структурного подхода.

2. Наиболее эффективной для построения хранилища данных BI-приложения страховых рисков является технология ROLAP.

3. Так как аналитическая отчетность по страховым рискам ориентирована на поддержку задач управления договорами страхования, представляется более эффективным в качестве хранилища данных использовать витрину данных отдела продаж страховых полисов.

Глава 3 Программная реализация BI-приложения для оценки страховых рисков

3.1 Выбор CMS для разработки BI-приложения

CMS (Content Management System) представляет собой программное обеспечение (ПО), предназначенное для помощи пользователям в создании и редактировании веб-сайта.

CMS позволяют разработать внешний вид веб-сайтов, отслеживать сеансы пользователей, обрабатывать запросы, собирать комментарии посетителей, организовывать форумы и многое другое.

Для выбора CMS сравним возможности двух популярных среди отечественных разработчиков CMS: Drupal и «1С-Битрикс: Управление сайтом».

3.1.1 Система управления сайтом Drupal

Drupal – это CMS, используемая для создания многих веб-сайтов и приложений [18].

Проект Drupal - это ПО с открытым исходным кодом. Он основан на таких принципах, как сотрудничество, глобализм и инновации и распространяется в соответствии с условиями GNU General Public License (GPL).

Drupal обладает отличными стандартными функциями, такими как простое создание контента, надежная производительность и отличная безопасность.

По мнению разработчиков CMS, главным ее достоинства является гибкость.

Модульность является одним из основных принципов Drupal. Встроенный инструментарий помогает создавать универсальный, структурированный контент, который нужен динамическому веб-интерфейсу.

Drupal применяется для разработки BI-приложений.

Так, среди ключевых потребностей Drupal-сообщества отмечается важность создания модуля BI, который можно использовать для определения запросов, анализирующих БД Drupal (рисунок 10).

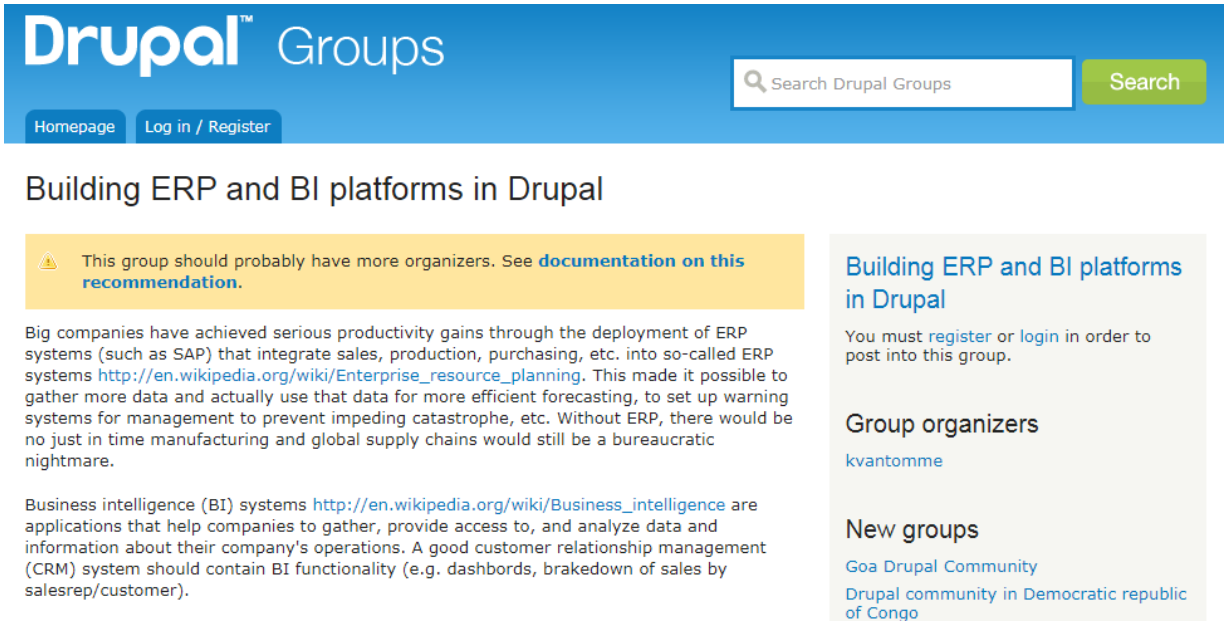

Рисунок 10 – Страница Drupal для группы BI- разработчиков

CMS Drupal ориентирована на платформу LAMP (Linux, Appache, MySQL, PHP), но возможны варианты использования другого системного ПО.

В настоящее время разработчикам предлагается версия Drupal 8.x.

3.1.2 Платформа «1С-Битрикс: Управление сайтом»

Платформа «1С-Битрикс: Управление сайтом» (далее - 1С-Битрикс) идеально подходит для разработки сайтов СМИ, тематических сайтов, блогов, информационных порталов, сайтов сообществ.

Она позволяет общаться с посетителями на форумах, проводить опросы, создавать фотогалереи, отправлять рассылки подписчикам.

По мнению вендора, 1С-Битрикс — самая популярная коммерческая CMS по реальным установкам на сайтах по рейтингу iTrack (на март 2020 года) [2].

Это объясняется тем, что сайты на платформе «1С-Битрикс» отличаются удобством, надежностью и высокой посещаемостью (рисунок 11).

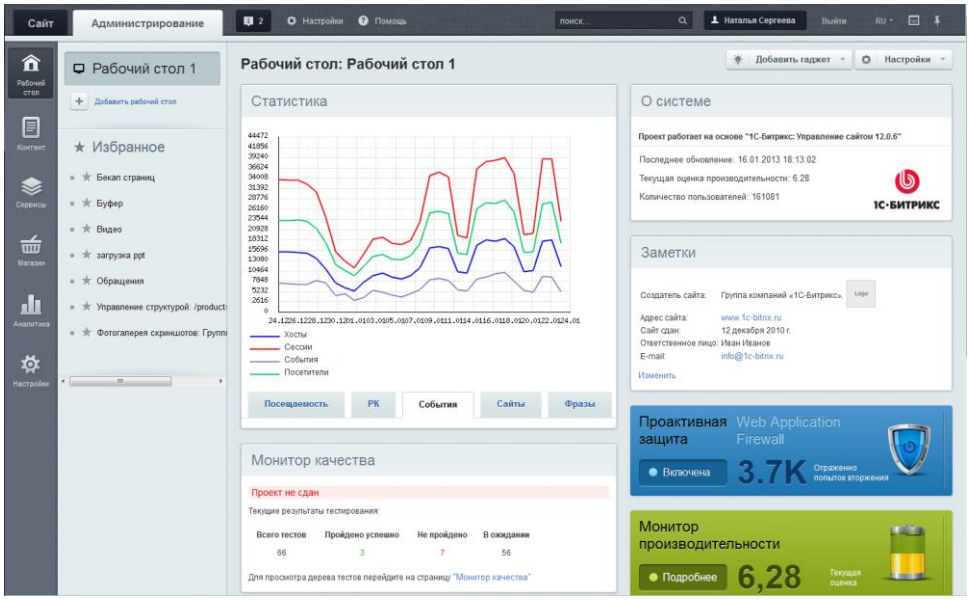

Рисунок 11 – Главный модуль платформы «1С-Битрикс: Управление сайтом»

Функциональные возможности платформы:

– профессиональное управление;
– адаптивный дизайн;
– продающие лендинги;
– готовые интеграции;
– e-mail-маркетинг;
– конструктор отчетов и др.

Стоимость годовой лицензии «Стандарт» платформы составляет 15900 руб.

Продукт поддерживает технологию LAMP и полностью совместим с платформой «1С: Предприятие 8.х».

Следует отметить, что данная платформа широко применяется для разработки веб-аналитики и BI-систем [29].

Для выбора CMS по выработанным критериям используем таблицу сравнительного анализа характеристик описанных платформ (таблица 4).

Таблица 4 – Сравнительный анализ характеристик CMS Drupal и 1С-Битрикс: Управление сайтом

Характеристика	Drupal	1С-Битрикс: Управление сайтом
Поддержка русского языка	+	+
Применение для разработки BI-приложений	+	+
Простота интеграции с КИС СК	-	+
Простота настройки	-	+
Низкая стоимость владения	+	-
Предпочтение разработчика	-	+
Итого:	3	5

Таким образом, на основании сравнительного анализа в качестве платформы для реализации BI-приложения для оценки страховых рисков выбираем CMS 1С-Битрикс: Управление сайтом.

3.2 Архитектура реализации BI-приложения

Программная архитектура CMS 1С-Битрикс: Управление сайтом представлена на рисунке 12.

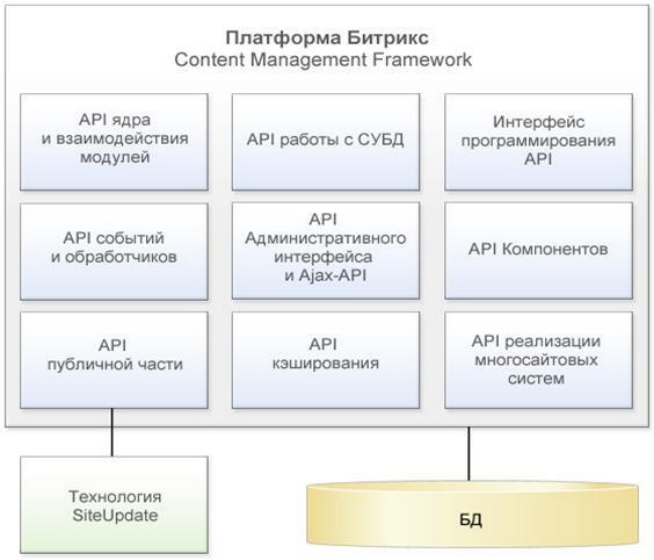

Рисунок 12 – Программная архитектура CMS 1С-Битрикс: Управление сайтом

Для разработки программной архитектуры BI-приложения используем диаграмму пакетов UML.

Диаграмма пакетов - структурная схема, которая показывает расположение и организацию элементов модели в среднем и крупномасштабном проекте.

С помощью диаграмма пакетов можно представить зависимости между подсистемами или модулями, показывая различные виды системы, например, как многоуровневое (многоуровневое) приложение - модель многоуровневого приложения.

На рисунке 13 представлена программная архитектура BI-приложения.

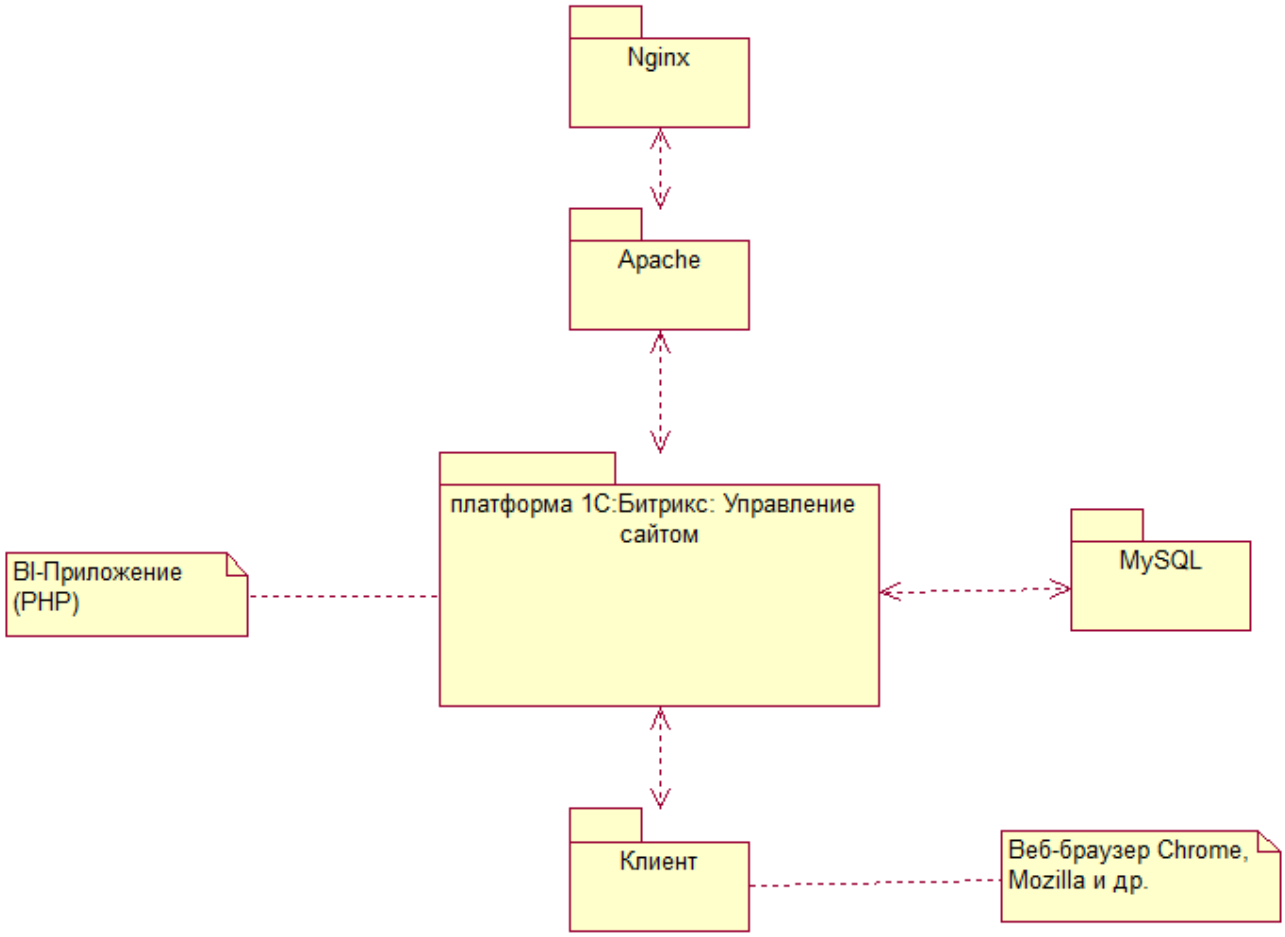

Рисунок 13 – Программная архитектура BI-приложения

В качестве вычислительной архитектуры BI-приложения используется трехзвенная архитектура «клиент-сервер» [1].

Для представления вычислительной архитектуры BI-приложения используем диаграмму развертывания UML.

Диаграммы развертывания используются для визуализации топологии физических компонентов системы, в которой развернуты программные компоненты.

Диаграммы развертывания используются для описания статического представления развертывания системы. Диаграммы развертывания состоят из узлов и их взаимосвязей.

На рисунке 14 представлена диаграмма развертывания BI-приложения.

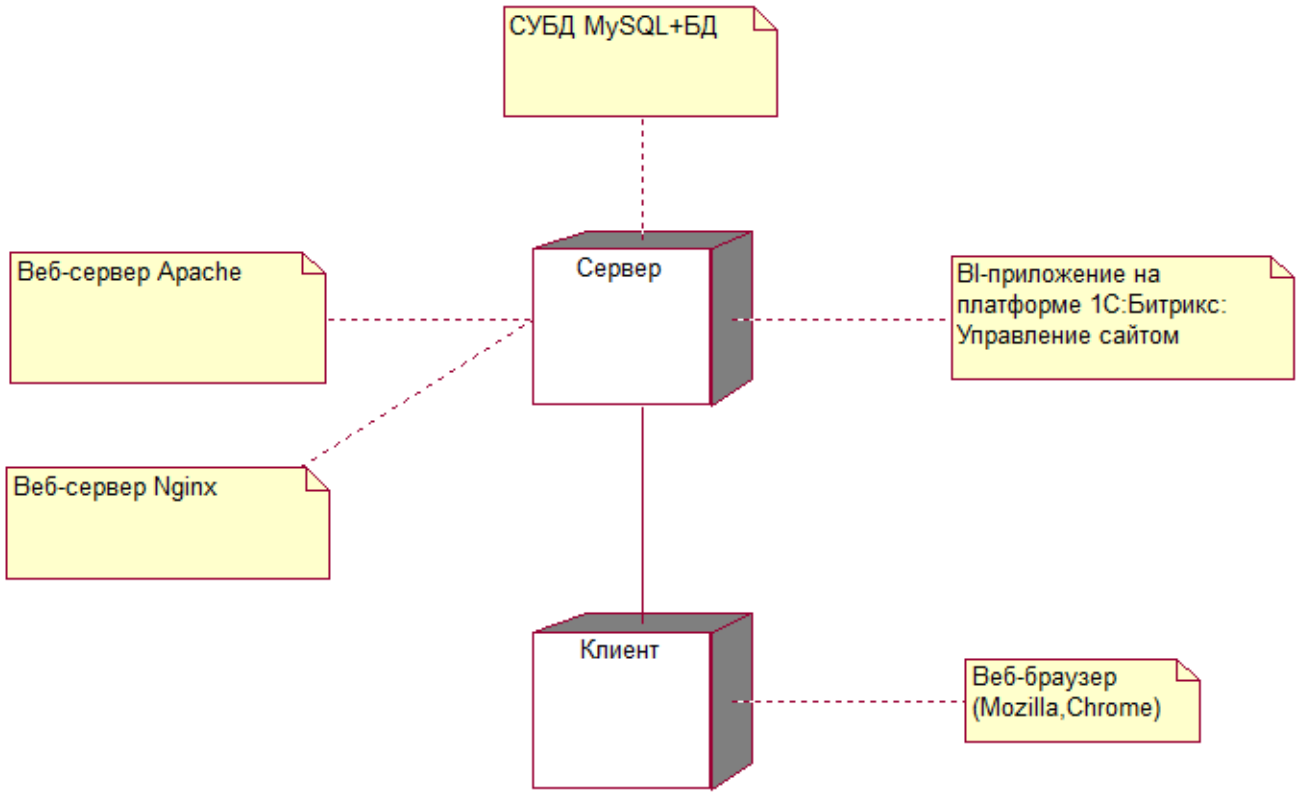

Рисунок 14 - Диаграмма развертывания BI-приложения

Как следует из представленных диаграмм, для развертывания BI-приложения используются два веб-сервера – Apache и Nginx, которые входят в состав программно-аппаратного обеспечения платформы 1С-Битрикс: Управления сайтом.

Веб-сервер Apache используется для развертывания BI-приложения на PHP.

Веб-сервер Nginx позволяет снизить нагрузку на компьютер сервера, обусловленную применением такого медленного и тяжеловесного бэкенд-сервера, как Apache. Кроме того, он используется для отдачи статического контента, а также в качестве обратного прокси-сервера перед веб-приложением.

3.3 Разработка программного обеспечения BI-приложения

Как было отмечено выше, для реализации ПО BI-приложения используется платформа 1С-Битрикс: Управления сайтом.

Диаграмма деятельности BI-приложения представлена на рисунке 15.

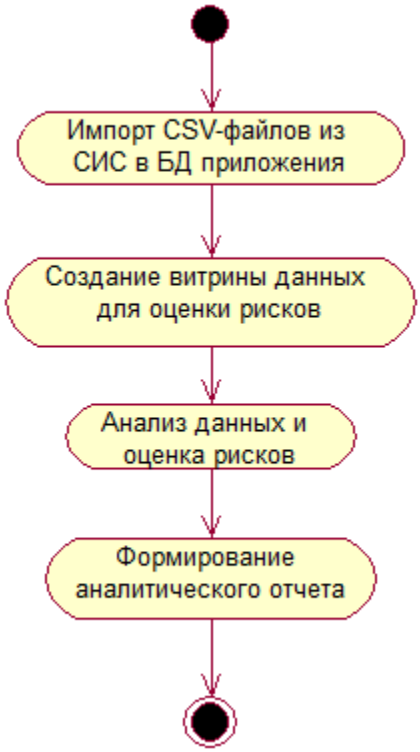

Рисунок 15 – Диаграмма деятельности BI-приложения

Для загрузки данных из страховой ИС (СИС), построенной на платформе 1С: Предприятие 8, используется встроенный механизм импорта данных из CSV-файлов (рисунок 16).

Файл CSV (Comma Separated Values File) – это текстовый файл, с разделенными строками, предназначенный для представления таблиц БД.

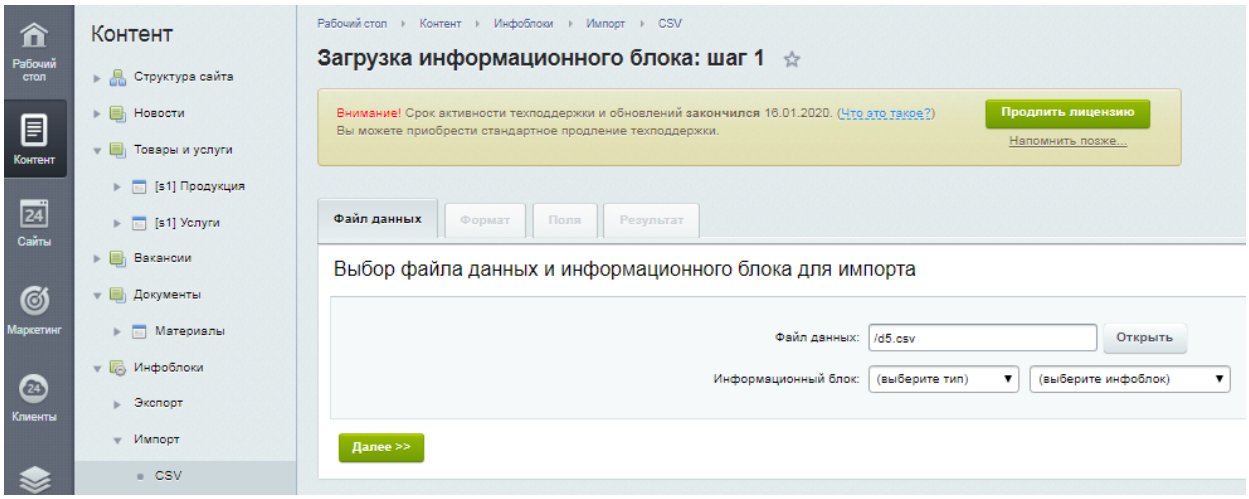

Рисунок 16 – Окно импорта данных для оценки риска

Далее средствами СУБД MySQL на основе загруженных данных создается витрина данных (ROLAP) для оценки страховых рисков по модели, представленной на рисунке 8.

Расчет показателей и оценка рисков после ввода всех необходимых параметров производятся по методике, описанной в п.1.1.

На первом этапе производится выборка данных (рисунок 17) и формирование страховой истории клиента по анализируемому риску (рисунок 18)

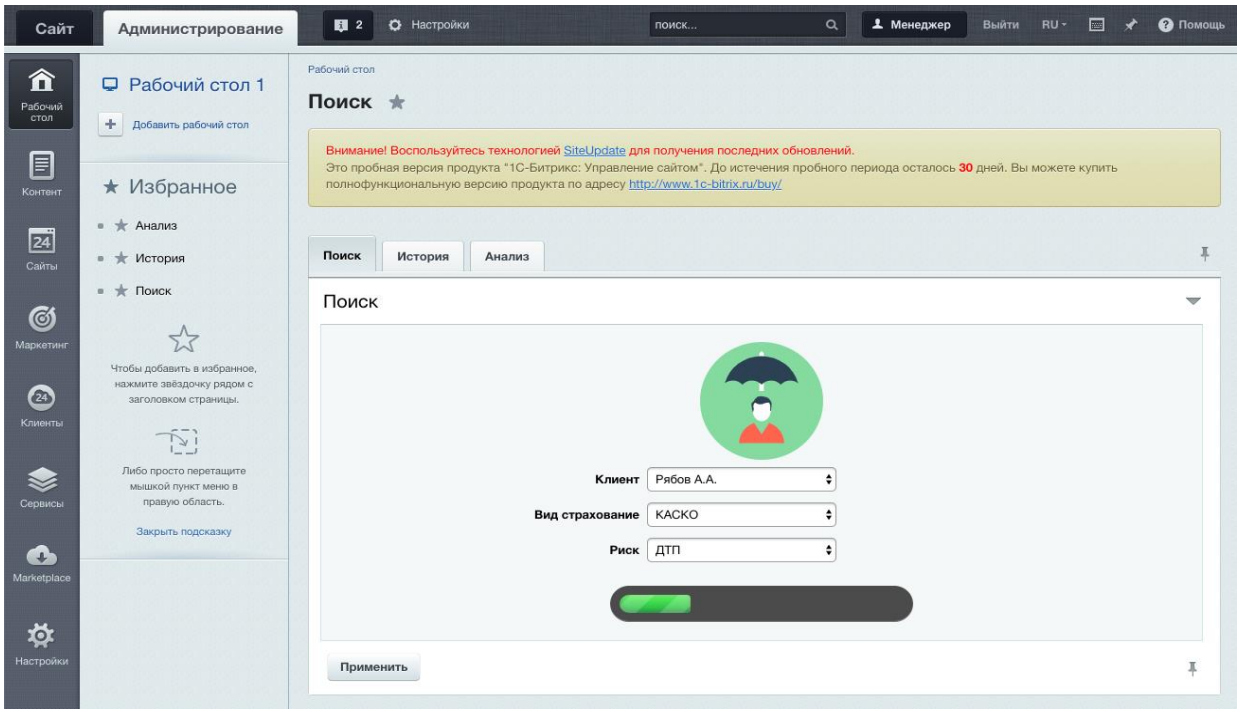

Рисунок 17 – Окно выборки данных клиента

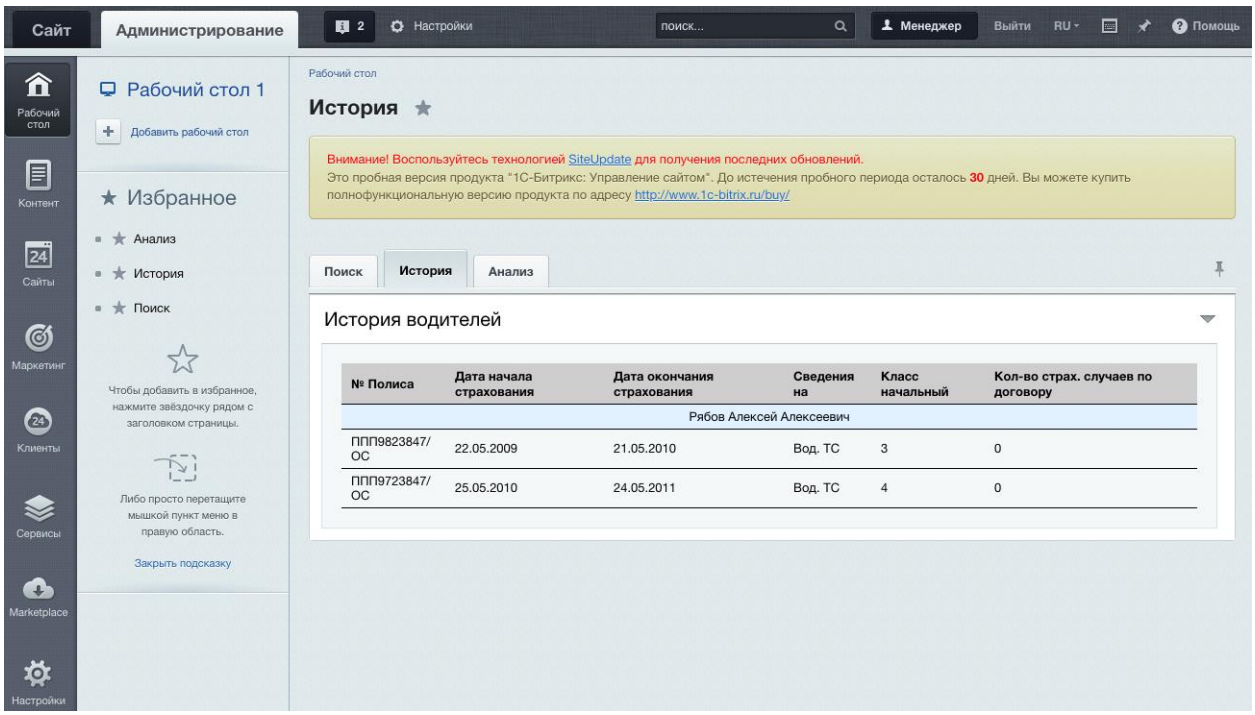

Рисунок 18 – Страховая история клиента

Далее формируется отчет анализа выбранного риска на предмет принятия на страхование (рисунок 19)

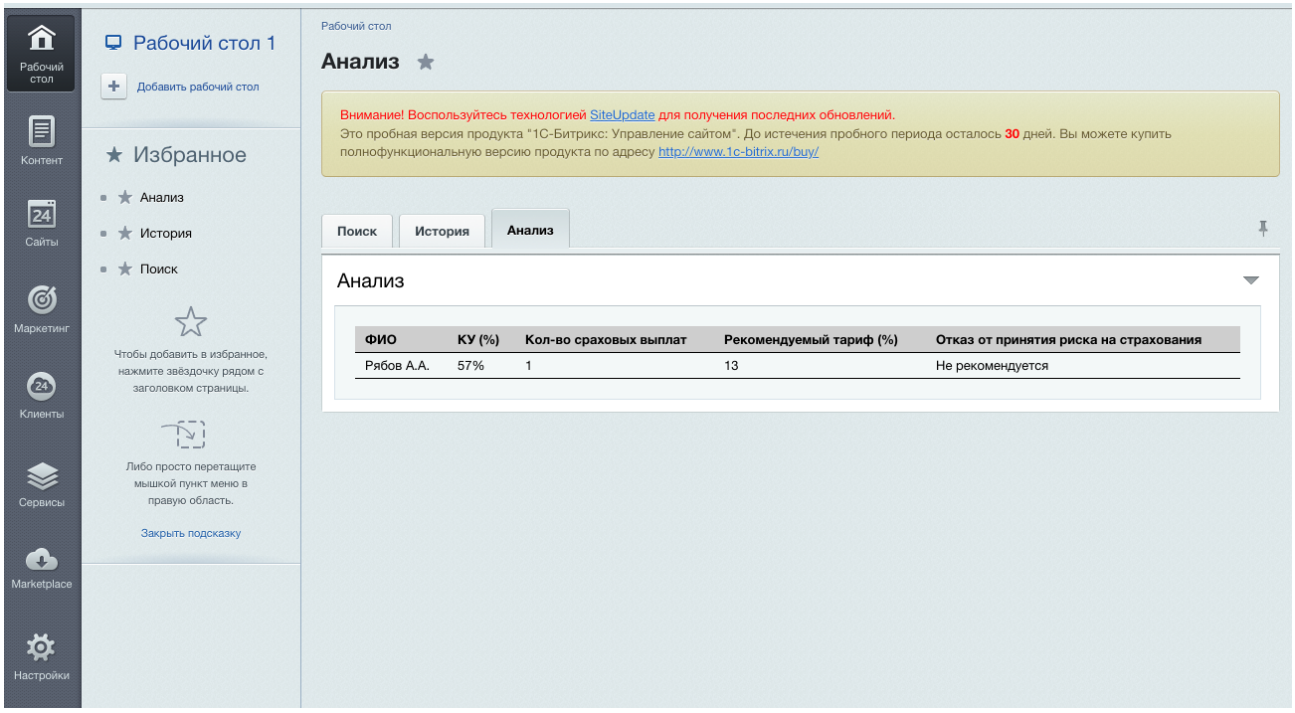

Рисунок 19 – Отчет оценки страхового риска

Блок анализа и формирования отчетности не входят в базовую конфигурацию платформы.

Данные блоки разработаны на языке PHP.

Фрагмент программного кода аналитического блока приведен ниже.

```
<?php
require_once __DIR__ . '/vendor/autoload.php';
# Импортирование модулей
use Phpml\Classification\LinearRegression;
use Phpml\Metrics\MeanSquaredError;
use Phpml\ModelSelection\TrainTestSplit;
use AnalisisConverter;
$X = require_once('_load_data.php');
$y = array_keys($X);
foreach($X as $k=>$v) {
   $X[$k] = AnalisisConverter::process($v['class'), $v);
}
$X = array_values($X);
// Разделение данных
list($X_train, $X_test, $y_train, $y_test) = (new TrainTestSplit($X, $y, $test_size=0.3, $random_state=42))->split();
$reg_all = new LinearRegression();
# Установка данных
$reg_all->train($X_train, $y_train);
# Получение прогнозируемого результата
$y_pred = $reg_all->predict($X_test);
# Вычисление квадратичного отклонения
$score = $reg_all->score($X_test, $y_test);
print("R^2: " . $score);
$error = new MeanSquaredError($y_test, $y_pred);
$rmse = sqrt($error->calc());
```

print("Root Mean Squared Error: " . $rmse);

Применение в качестве среды разработки платформы 1С-Битрикс: Управления сайтом позволило упростить разработку ПО BI-приложения и его интеграцию в КИС страховой компании.

3.4 Тестирование BI-приложения

Для проверки качества разработанного BI-приложения выполнено его тестирование.

В качестве вида тестирования выбрано тестирование производительности сайта BI-приложения [Котляров].

Для автоматизации тестирования использовано встроенное средство «Монитор производительности», которое позволяет в абсолютных величинах протестировать конфигурацию приложения и общую производительность проекта.

На рисунке 20 представлен результат мониторинга производительности конфигурации приложения.

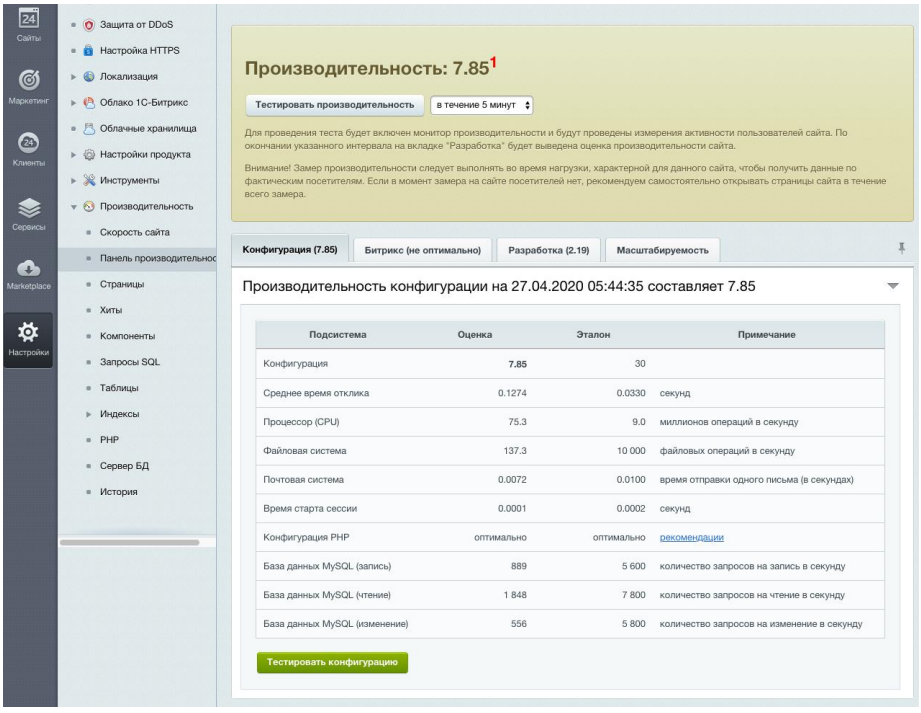

Рисунок 20 – Результаты оценки производительности конфигурации

Производительность конфигурации – это показатель количества пустых страниц, генерируемых сервером в секунду.

В нашем случае она равна примерно 8 страницам, что является приемлемым результатом для BI-приложений.

Для оценки производительности SQL запросов используется соответствующий мониторинг (рисунок 21).

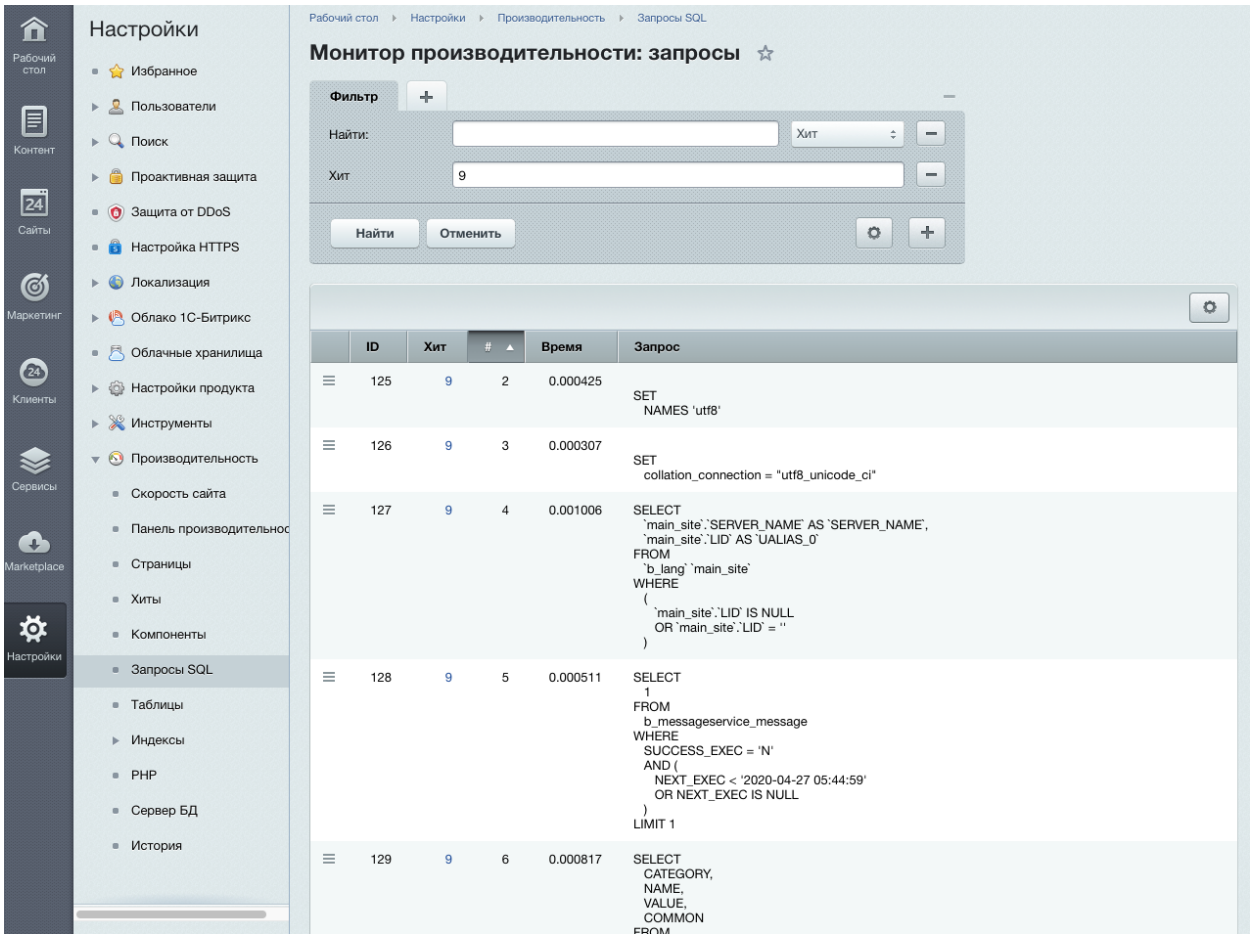

Рисунок 21 – Мониторинг производительности SQL запросов

Следует отметить, что данный показатель не является критичным для BI-приложения, так как в нем используется запрос для создания OLAP-куба.

Для оценки масштабируемости приложения используется тест производительности многопоточных и веб-кластерных систем (рисунок 22).

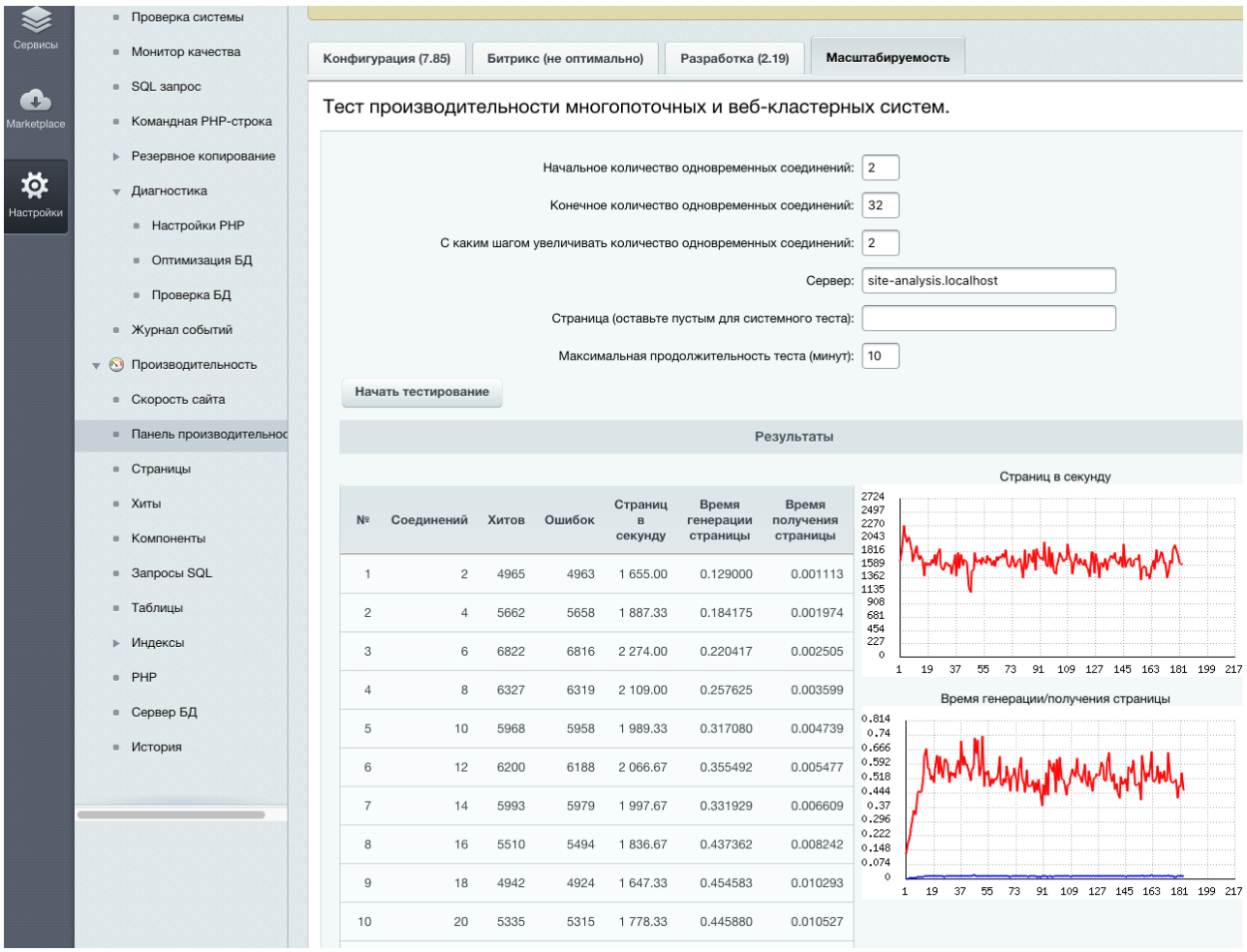

Рисунок 22 – Результаты теста производительности многопоточных и веб-кластерных систем

Целью данного нагрузочного теста является определение, насколько характеристики используемого хостинга соответствуют установленным требованиям.

Результаты теста показали, что ухудшение производительности работы BI-приложения наблюдается при количестве пользователей, превышающем 8, что соответствует требованиям.

Таким образом, тестирование BI-приложения подтвердило его работоспособность и соответствие требованиям, предъявляемым к приложениям такого класса.

Фрагмент кода теста производительности BI-приложения представлен в Приложении А.

3.5 Требование к аппаратно-программному обеспечению BI-приложения

Программное обеспечение серверной части должно соответствовать требованиям:

– операционные системы FreeBSD, Windows или Linux;

– СУБД MySQL 7.1 или выше;

– Веб-сервер Apache версии 2.4 или выше;

– Веб-сервер Nginx версии 1.10 или выше;

– Язык программирования PHP версии 5.6 или выше.

Аппаратное обеспечение серверной части должно обеспечивать поддержку указанного программного обеспечения.

Для обеспечение указанных требований рекомендуется воспользоваться услугами коммерческого хостинга.

Программное обеспечение клиентской части должно включать

– Веб-браузеры Mozilla Firefox, Google Chrome и др.;

– пакет MS Office.

Аппаратное обеспечение клиентской части должно обеспечивать поддержку указанного программного обеспечения.

Выводы к главе 3

1. Результаты анализа показали, что наиболее полно требованиям по реализации BI-приложения для оценки страховых рисков соответвествует CMS 1С-Битрикс: Управление сайтом.

2. Применение в качестве среды разработки платформы 1С-Битрикс: Управления сайтом позволило упростить разработку ПО BI-приложения и его интеграцию в КИС страховой компании.

3. Тестирование BI-приложения подтвердило его работоспособность и соответствие требованиям, предъявляемым к приложениям такого класса.

Заключение

Бакалаврская работа посвящена актуальной проблеме разработки BI-приложения для оценки страховых рисков.

В ходе выполнения бакалаврской работы достигнуты следующие результаты:

1. Проанализированы подходы к разработке BI-приложений для оценки страховых рисков. Как показал анализ, для BI-приложения для оценки страховых рисков является более предпочтительным использование архитектуры на основе OLAP-технологий.

2. Проанализированы функциональные и архитектурные особенности известных BI-приложений для страховой деятельности. Как показал анализ, помимл сложности интеграции в КИС страховой компании в известных BI-приложениях не используются веб-технологии, поэтому принято решение разработать BI-приложение на промышленной CMS.

3. На основе объектно-структурного подхода разработана логическая модель BI-приложения. В качестве хранилища данных использована витрина данных отдела продаж страховых полисов, построенная на основе технологии ROLAP.

4. Разработано программное обеспечение BI-приложения. В качестве платформы для разработки выбрана промышленная CMS 1С-Битрикс: Управление сайтом.

5. Выполнено тестирование разработанного BI-приложения, которое подтвердило его работоспособность и соответствие требованиям, предъявляемым к приложениям такого класса.

Результаты бакалаврской работы представляют практический интерес и могут быть рекомендованы для разработчиков BI-приложений для страховой деятельности.

Список используемых источников и используемой литературы

1. Белугина С.В. Разработка программных модулей программного обеспечения для компьютерных систем. Прикладное программирование: учебное пособие. – СПб.: Лань, 2020. 312 с.

2. Битрикс CMF (Content Management Framework) [Электронный ресурс]. URL: https://dev.1c-bitrix.ru/community/blogs/rsv/152.php (дата обращения: 12.02.2020).

3. ГОСТ 19.402–78. Единая система программной документации. Описание программы, 1980. – Москва, 2010.

4. ГОСТ 28806-90. Качество программных средств. Термины и определения.

5. Информационные аналитические системы [Электронный ресурс]: учебник/ Т.В. Алексеева [и др.]. Москва: Московский финансово-промышленный университет «Синергия», 2013. 384 с. URL: http://www.iprbookshop.ru/17015.html.

6. Корпоративное хранилище данных страховой компании (Instras-report). [Электронный ресурс]. URL: https://www.vtsft.ru/projects/insurance/korporativnoe-xranilishhe-dannyix-straxovoj-kompanii-(instras-report) (дата обращения: 12.02.2020).

7. Котляров В.П. Основы тестирования программного обеспечения [Электронный ресурс]. - Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. 334 с. URL: http://www.iprbookshop.ru/62820.html. ЭБС «IPRbooks» (дата обращения: 12.02.2020).

8. Леоненков А.В. Объектно-ориентированный анализ и проектирование с использованием UML и IBM Rational Rose. [Электронный ресурс]: учебное пособие. Москва, Саратов: ИНТУИТ, Вузовское образование, 2017. 318 с. URL: http://www.iprbookshop.ru/67388.html (дата обращения: 12.02.2020).

9. Мкртычев С.В. Объектно-структурное моделирование страховых информационных систем // Вектор науки ТГУ. № 2(23). 2013. С. 210-212.

10. Назарова В.В., Матвеева Е.А. Концепция риск-менеджмента в страховых организациях // Научный журнал НИУ ИТМО. Серия «Экономика и экологический менеджмент». 2014. №4. С. 275.

11. Платформа «1С-Битрикс: Управление сайтом» [Электронный ресурс]. URL: https://www.1c-bitrix.ru/products/cms/ (дата обращения: 12.02.2020).

12. Программный продукт «1С: Управление страховой компанией 8» [Электронный ресурс]. – Режим доступа:https://www.orticongroup.ru/solutions/industry/resheniya-dlya-strahovyh-kompaniy/1c-upravlenie-strahovoy-kompaniey-8/#prise (дата обращения: 12.02.2020).

13. Трегубова А.А., Скрипкина Н.В. Оценка рисков в автостраховании: возможности применения поправочных коэффициентов // Учет и статистика. 2015. №1 (37). С. 101-109.

14. Языков А. Практический опыт использования BI в ВТБ Страхование [Электронный ресурс]. URL: https://docplayer.ru/29178413-Prakticheskiy-opyt-ispolzovaniya-bi-v-vtb-strahovanie-aleksandr-yazykov-otdel-razrabotki-otchetnosti-dsiit-ooo-sk-vtb-strahovanie.html (дата обращения: 12.02.2020).

15. Bara A. and others. A model for Business Intelligence Systems' Development, Informatica Economică vol. 13, no. 4/2009, pp. 99-107.

16. BI consult [Электронный ресурс]. URL: http://biconsult.ru/ (дата обращения: 12.02.2020).

17. Business Intelligence – путь к решению проблем страхования [Электронный ресурс]. URL: http://citforum.ru/consulting/BI/biz_int_strah/ (дата обращения: 12.02.2020).

18. CMS Drupal – официальный сайт [Электронный ресурс]. URL: https://www.drupal.org/ (дата обращения: 12.02.2020).

19. Define the Problem Requirements FURPS Requirement Model [Электронный ресурс]. URL: http://www.cs.unh.edu/~cs619/slides/usecase.pdf (дата обращения: 12.02.2020).

20. Denuit M. et al. Actuarial Modelling of Claim Counts, London: Wiley, 2007. -356 p.

21. Gartner glossary [Электронный ресурс]. URL: https://www.gartner.com/en/glossary (дата обращения: 12.02.2020).

22. Mkrtychev S., Enik O. Automated Underwriting Control in a Regional Insurance Company, Advances in Economics, 2019 Business and Management Research., №47, pp. 258-260.

23. Moss L.T., Atre S. - Business Intelligence Roadmap: The Complete Project Lifecycle for Decision-Support Applications. Addison Wesley, Boston 2003.

24. MySQL Workbench [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/MySQL_Workbench (дата обращения: 12.02.2020).

25. Overview of SAP BI Architecture [Электронный ресурс]. URL: http://www.sapspot.com/overview-sap-bi-architecture/ (дата обращения: 12.02.2020).

26. Relational online analytical processing (ROLAP) [Электронный ресурс]. URL: https://searchoracle.techtarget.com/definition/relational-online-analytical-processing (дата обращения: 12.02.2020).

27. Rostek K. Business Intelligence for Insurance Companies, Foundations of Management, 2009, Vol. 1, No. 1, pp. 65-82.

28. The Logical Model [Электронный ресурс]. URL: https://sparxsystems.com/resources/tutorials/uml/logical-model.html (дата обращения: 12.02.2020).

29. Web и BI аналитика [Электронный ресурс]. URL: https://int-man.ru/services/web_analitika_i_bi_sistemy/ (дата обращения: 12.02.2020).

Приложение А

Фрагмент кода теста производительности BI-приложения

```
<?
@ini_set("track_errors", "1");
@ini_set('display_errors', 1);
error_reporting(E_ALL & ~E_NOTICE & ~E_STRICT &
~E_DEPRECATED);
$message = null;

define('DEBUG_FLAG',
str_replace('\\','/',$_SERVER['DOCUMENT_ROOT'] .
'/bitrix/site_checker_debug'));
require($_SERVER['DOCUMENT_ROOT'].'/bitrix/modules/main/classes/ge
neral/site_checker.php');

// NO AUTH TESTS
if ($_REQUEST['unique_id'])
{
  if (!file_exists(DEBUG_FLAG) && $_REQUEST['unique_id'] !=
checker_get_unique_id())
        die('Permission denied: UNIQUE ID ERROR');

  switch ($_GET['test_type'])
  {
        case 'socket_test':
              echo "SUCCESS";
        break;
        case 'webdav_test':
              if ($_SERVER['REQUEST_METHOD'] == $_GET['method'])
                    echo "SUCCESS";
              else
                    echo 'Incorrect $_SERVER[REQUEST_METHOD]:
'.$_SERVER['REQUEST_METHOD'].', expected: '.preg_replace('#[^A-Z]#', '',
$_GET['method']);
        break;
        case 'compression':
              echo str_repeat('SUCCESS', 8*1024);
        break;
        case 'perf':
              define("NOT_CHECK_PERMISSIONS", true);
              define("LDAP_NO_PORT_REDIRECTION", true);
```

```
	require_once($_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/main/include/prolog_before.php");

	require_once($_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/main/include/epilog_after.php");
			echo $main_exec_time;
		break;
		case 'fast_download':
			header('X-Accel-Redirect: /bitrix/tmp/success.txt');
		break;
		case 'dbconn_test':
			ob_start();
			define('NOT_CHECK_PERMISSIONS', true);

	require($_SERVER['DOCUMENT_ROOT'].'/bitrix/modules/main/include/prolog_before.php');
			$buff = '';
			while(ob_get_level())
			{
				$buff .= ob_get_contents();
				ob_end_clean();
			}
			ob_end_clean();
			if (function_exists('mb_internal_encoding'))
				mb_internal_encoding('ISO-8859-1');
			echo $buff === '' ? 'SUCCESS' : 'Length: '.strlen($buff).' ('.$buff . ')';
		break;
		case 'pcre_recursion_test':
			$a = str_repeat('a',4096);
			if (preg_match('/(a)+/',$a)) // Segmentation fault (core dumped)
				echo 'SUCCESS';
			else
				echo 'CLEAN';
		break;
		case 'method_exists':
			$arRes= Array
			(
				"CLASS" => "",
				"CALC_METHOD" => ""
			);
			method_exists($arRes['CLASS'], $arRes['CALC_METHOD']);
			echo 'SUCCESS';
```

```
		break;
		case 'upload_test':
			if (function_exists('mb_internal_encoding'))
				mb_internal_encoding('ISO-8859-1');

			$dir = $_SERVER['DOCUMENT_ROOT'].'/bitrix/tmp';
			if (!file_exists($dir))
				mkdir($dir);

			$binaryData = '';
			for($i=40;$i<240;$i++)
				$binaryData .= chr($i);
			if ($_REQUEST['big'])
				$binaryData = str_repeat($binaryData, 21000);

			if ($_REQUEST['raw'])
				$binaryData_received = file_get_contents('php://input');
			elseif (move_uploaded_file($tmp_name =
$_FILES['test_file']['tmp_name'], $image = $dir.'/site_checker.bin'))
			{
				$binaryData_received = file_get_contents($image);
				unlink($image);
			}
			else
			{
				echo
'move_uploaded_file('.$tmp_name.','.$image.')=false'."\n";
				echo '$_FILES='."\n";
				print_r($_FILES);
				die();
			}

			if ($binaryData === $binaryData_received)
				echo "SUCCESS";
			else
				echo 'strlen($binaryData)='.strlen($binaryData).',
strlen($binaryData_received)='.strlen($binaryData_received);
		break;
		case 'post_test':
			$ok = true;
			for ($i=0;$i<201;$i++)
				$ok = $ok && ($_POST['i'.$i] == md5($i));

			echo $ok ? 'SUCCESS' : 'FAIL';
```

```
                break;
        case 'memory_test':
                @ini_set("memory_limit", "512M");
                $max = intval($_GET['max']);
                if ($max)
                {
                        for($i=1;$i<=$max;$i++)
                                $a[] = str_repeat(chr($i),1024*1024); // 1 Mb

                        echo "SUCCESS";
                }
        break;
        case 'auth_test':
                $remote_user = $_SERVER["REMOTE_USER"] ?
$_SERVER["REMOTE_USER"] :
$_SERVER["REDIRECT_REMOTE_USER"];
                $strTmp = base64_decode(substr($remote_user,6));
                if ($strTmp)
                        list($_SERVER['PHP_AUTH_USER'],
$_SERVER['PHP_AUTH_PW']) = explode(':', $strTmp);
                if ($_SERVER['PHP_AUTH_USER']=='test_user' &&
$_SERVER['PHP_AUTH_PW']=='test_password')
                        echo('SUCCESS');
        break;
        case 'session_test':
                session_start();
                echo $_SESSION['CHECKER_CHECK_SESSION'];
                $_SESSION['CHECKER_CHECK_SESSION'] = 'SUCCESS';
        break;
        case 'redirect_test':

      foreach(array('SERVER_PORT','HTTPS','FCGI_ROLE','SERVER_PROTO
COL','SERVER_PORT','HTTP_HOST') as $key)
                        $GLOBALS['_SERVER'][$key] =
$GLOBALS['_REQUEST'][$key];
                function IsHTTPS()
                {
                        return ($_SERVER["SERVER_PORT"]==443 ||
strtolower($_SERVER["HTTPS"])=="on");
                }

                function SetStatus($status)
                {
                        $bCgi = (stristr(php_sapi_name(), "cgi") !== false);
```

```
                        $bFastCgi = ($bCgi &&
(array_key_exists('FCGI_ROLE', $_SERVER) || array_key_exists('FCGI_ROLE',
$_ENV)));
                        if($bCgi && !$bFastCgi)
                                header("Status: ".$status);
                        else
                                header($_SERVER["SERVER_PROTOCOL"]."
".$status);
                }

                if ($_REQUEST['done'])
                        echo 'SUCCESS';
                else
                {
                        SetStatus("302 Found");
                        $protocol = (IsHTTPS() ? "https" : "http");
                        $host = $_SERVER['HTTP_HOST'];
                        if($_SERVER['SERVER_PORT'] <> 80 &&
$_SERVER['SERVER_PORT'] <> 443 && $_SERVER['SERVER_PORT'] > 0
&& strpos($_SERVER['HTTP_HOST'], ":") === false)
                                $host .= ":".$_SERVER['SERVER_PORT'];
                        $url =
"?redirect_test=Y&done=Y&unique_id=".checker_get_unique_id();
                        header("Request-URI: ".$protocol."://".$host.$url);
                        header("Content-Location: ".$protocol."://".$host.$url);
                        header("Location: ".$protocol."://".$host.$url);
                        exit;
                }
                break;
                default:
                break;
        }

        if ($fix_mode = intval($_GET['fix_mode']))
        {
                if ($_REQUEST['charset'])
                {
                        define('LANG_CHARSET', preg_replace('#[^a-z0-9\-]#i', ",
$_REQUEST['charset']));
                        header('Content-type: text/plain; charset='.LANG_CHARSET);
                }
                define('LANGUAGE_ID', preg_match('#[a-
z]{2}#',$_REQUEST['lang'],$regs) ? $regs[0] : 'en');
```

```
		if (file_exists($file =
$_SERVER['DOCUMENT_ROOT'].'/bitrix/modules/main/lang/'.LANGUAGE_ID
.'/admin/site_checker.php'))
			include_once($file);
		else

	include_once($_SERVER['DOCUMENT_ROOT'].'/bitrix/modules/main/lan
g/en/admin/site_checker.php');
		InitPureDB();
		if (!function_exists('AddMessage2Log'))
		{
			function AddMessage2Log($sText, $sModule = "", $traceDepth
= 6, $bShowArgs = false)
			{
				echo $sText;
			}
		}

		if (!function_exists('htmlspecialcharsbx'))
		{
			function htmlspecialcharsbx($string, $flags=ENT_COMPAT)
			{
				//shitty function for php 5.4 where default encoding is
UTF-8
				return htmlspecialchars($string, $flags,
(defined("BX_UTF")? "UTF-8" : "ISO-8859-1"));
			}
		}

		if (!function_exists('GetMessage'))
		{
			function GetMessage($code, $arReplace = array())
			{
				global $MESS;
				$strResult = $MESS[$code];
				foreach($arReplace as $k => $v)
					$strResult = str_replace($k, $v, $strResult);
				return $strResult;
			}
		}

		if (!function_exists('JSEscape'))
		{
			function JSEscape($s)
```

```
                {
                        static $aSearch = array("\xe2\x80\xa9", "\\", "'", "\"",
"\r\n", "\r", "\n", "\xe2\x80\xa8");
                        static $aReplace = array(" ", "\\\\", "\\'", '\\"', "\n", "\n",
"\\n'+\n'", "\\n'+\n'");
                        $val =  str_replace($aSearch, $aReplace, $s);
                        return preg_replace("'</script'i", "</s'+'cript", $val);
                }
        }

        $oTest = new CSiteCheckerTest($_REQUEST['step'], 0, $fix_mode);
        if (file_exists(DEBUG_FLAG))
                $oTest->timeout = 30;

        if ($_REQUEST['global_test_vars'] && ($d =
base64_decode($_REQUEST['global_test_vars'])))
                $oTest->arTestVars = unserialize($d);
        else
                $oTest->arTestVars = array();

        $oTest->Start();
        if ($oTest->percent < 100)
        {
                $strNextRequest = '&step='.$oTest-
>step.'&global_test_vars='.base64_encode(serialize($oTest->arTestVars));
                $strFinalStatus = '';
        }
        else
        {
                $strNextRequest = '';
                $strFinalStatus = '100%';
        }
        // fix mode
        echo '
                iPercent = '.$oTest->percent.';
                test_percent = '.$oTest->test_percent.';
                strCurrentTestFunc = "'.$oTest->last_function.'";
                strCurrentTestName = "'.JSEscape($oTest-
>strCurrentTestName).'";
                strNextTestName = "'.JSEscape($oTest->strNextTestName).'";
                strNextRequest = "'.JSEscape($strNextRequest).'";
                strResult = "'.JSEscape(str_replace(array("\r","\n"),"",$oTest-
>strResult)).'";
                strFinalStatus = "'.JSEscape($strFinalStatus).'";
```

```
                    test_result = '.($oTest->result === true ? 1 : ($oTest->result
=== false ? -1 : 0)).'; // 0 = note
                ';
          }
          die();
        }
        // END NO AUTH TESTS
```