МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ МОДУЛЬНОЙ ANTIFRAUD СИСТЕМЫ СРЕДСТВАМИ PL/SQL

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 5 курса 551 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Гонтарь Антона Владимировича

Научный руководитель
зав. каф. техн. пр., к. ф.-м. н., доцент _____ И. А. Батраева

Заведующий кафедрой
к. ф.-м. н., доцент _____ А. С. Иванов

Саратов 2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сегодня мир невозможно представить без финансовых институтов. В свою очередь современный финансовый институт невозможно представить без развитой IT-инфраструктуры. Будь то удобное мобильное приложение для клиентов либо высоко-функциональное программное обеспечение для сотрудников.

С каждым годом финансовые учреждения наращивают IT потенциал и постепенно превращаются в IT-компании, предоставляющие широкий спектр услуг свои клиентам. Сегодня банковская – это вторая отрасль после IT по количеству рабочих мест для разработчиков, системных аналитиков, администраторов баз данных и т.д.

Одной из важнейших IT систем банка является база данных (далее БД). В ней хранятся: персональные данные клиентов и сотрудников, данные по всем финансовым операциям, данные по продуктам и услугам и т.д. Наличие такого реестра данных является обязательным для всех без исключения банков и регламентируется законом о банках и их деятельности.

Тема данной работы связана с существующим проектом и описывает этапы его реализации. Данный проект разработан в рамках минимизации финансовых и операционных банковских рисков. Заказчиком проекта выступает коммерческий банк основной функцией которого является потребительское кредитование физических лиц.

Целью работы является применение методов Business intelligence (далее BI), таких, как агрегация и денормализация данных, а также методов оптимизации запросов к БД. Реализуемый в работе проект призван систематизировать данные в более удобный для анализа и чтения вид.

1 Методы проектирования системы

1.1 Выбор технологии

1.1.1 СУБД

Проект выполнен в объектно-реляционной СУБД Oracle Database 11g. Как среда разработки используется PL/SQL Developer.

Существует огромное количество определений «баз данных». Для бакалаврской работы было выбрано определение за авторством Коннолли Томас и Бегг Каролин. База данных — совместно используемый набор логически связанных данных (и описание этих данных), предназначенный для удовлетворения информационных потребностей организации [1].

Система управления базами данных (DBMS) - комплекс программ, позволяющих создать базу данных и манипулировать данными (вставлять, обновлять, удалять и выбирать). Система обеспечивает безопасность, надежность хранения и целостность данных, а также предоставляет средства для администрирования БД.

Базы данных можно классифицировать по модели данных. Модель данных — это определение операторов и объектов, составляющих в купе абстрактную машину доступа к данным, с которой взаимодействует пользователь. Эти операторы позволяют моделировать поведение данных, а объекты — структуру данных.

1.1.2 Хранилище данных

Корпоративное хранилище данных (DWH) – предметно-ориентированная информационная база данных, специально разработанная и предназначенная для подготовки отчетов и бизнес-анализа с целью поддержки принятия решений в организации [2].

Основные источники данных для проекта:

- Корпоративное хранилище данных (далее КХД);
- Денормализованные витрины данных Antifraud.

Хранилище данных, с которым предстоит работать при разработке проекта является нормализованным, данные находятся в предметно ориентированных таблицах третьей нормальной формы.

Нормальная форма — свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, потенциально приводящей к логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение. Конечной целью нормализации является уменьшение потенциальной противоречивости хранимой в базе данных информации [3].

Для того, чтобы дойти до конечного пользователя, данные несколько раз преобразовываются:

- Шаг 1. Автоматизированная банковская система формирует данные в транзакционной базе данных (OLTP);
- Шаг 2. При помощи процессов ETL данные извлекаются, преобразовываются и записываются в корпоративное хранилище данных;
- Шаг 3. Данные денормализуются и записываются в витрины Antifraud.

OLTP (Online Transaction Processing), транзакционная система — обработка транзакций в реальном времени. Способ организации БД, при котором система работает с небольшими по размерам транзакциями, но идущими большим потоком. При этом клиенту требуется от системы минимальное время отклика.

ETL (Extract, Transform, Load) — один из основных процессов в управлении хранилищами данных, который включает в себя:

- извлечение данных из внешних источников;
- их трансформация и очистка, чтобы они соответствовали потребностям бизнес-модели;
- и загрузка их в хранилище данных.

Business intelligence (сокращенно BI) — обозначение компьютерных методов и инструментов, обеспечивающих перевод транзакционной деловой информации в человекочитаемую форму, пригодную для бизнес-анализа.

Для извлечения данных воспользуемся базовыми принципами BI:

- Многомерная агрегация и размещение данных в витринах;
- Денормализация таблиц баз данных.

1.2 Язык PL/SQL

PL/SQL - это процедурный блочно-структурированный язык. Он представляет собой расширение языка SQL и предназначен для работы с СУБД Oracle.

Основной программной единицей PL/SQL является блок, который может содержать вложенные блоки, называемые иногда подблоками. Блок позволяет объединять объявления и операторы, связанные общей логикой; может быть анонимным и именованным. Блок состоит из трех основных частей:

- Секция объявлений (необязательная часть);
- Тело блока;
- Обработчики исключений (необязательная часть).

Язык PL/SQL позволяет определять следующие типы именованных блоков, которые могут быть скомпилированы и сохранены как объекты базы данных в некоторой её схеме:

- Процедуры;
- Функции;
- Объекты;
- Пакеты.

Язык PL/SQL поддерживает следующие операторы управления в блоках:

- Операторы выбора (IF, CASE);
- Операторы цикла (LOOP, WHILE, FOR);
- Операторы безусловного перехода (GOTO, NULL).

1.3 Оптимизация

1.3.1 Индексы

Индекс — объект базы данных, создаваемый с целью повышения производительности поиска данных. Ускорение работы с использованием индексов достигается в первую очередь за счёт того, что индекс имеет структуру, оптимизированную под поиск — например, сбалансированного дерева [4].

Простейшим примером применения индексов в реальной жизни является оглавление книги. Вместо перебора всех страниц книги читателю достаточно найти конкретную главу на странице оглавления.

Индексы имеют две базовые функции:

- увеличение скорости доступа к данным;
- поддержка уникальности данных.

1.3.2 Секционирование

Сегодня секционирование — является основным инструмент для создания, больших по объему данных, таблиц имеющие жесткие требования к производительности. При помощи секционирования архитекторы и администраторы БД могут решать ряд сложных проблем [5].

Объекты БД (таблицы и индексы) секционируются при помощи ключа секционирования – это столбец или набор столбцов, который определяет, в какую секцию будет попадать запись.

С помощью конструкции ADD PARTITION, в рамках DDL-операции ALTER TABLE, возможно добавить секции к верхнему пределу существующей секционированной таблицы. Если авто-секционирование в таблице не предусмотрено, то применение данной конструкции можно автоматизировать в рамках блока PL/SQL.

1.3.3 Автоматизация

Планировщик заданий – это пакет, который позволяет запускать блоки PL/SQL по-определенному расписанию исходя из условий заданных пользователем.

Помимо автоматизации планировщик задач позволяет разработчику оптимизировать ресурсоемкую задачу по времени за счет распараллеливания выполнения расчетов. При этом не стоит забывать, что такой подход к выполнению задачи будет использовать дополнительные сессии пользователя.

1.3.4 Подсказки оптимизатору

Важным фактором при оптимизации процесса является скорость работы SQL запроса. Причины тому могут быть следующие:

- Плохая статистика по таблицам и индексам запроса;
- Проблемы с индексами в запросе;
- Проблемы с хинтами (подсказками) в запросе;
- Неэффективно построенный запрос.

При построении SQL-запроса разработчику важно уметь пользоваться планом запроса. Зачастую оптимизатор строит план выполнения SQL-запроса не оптимально, основываясь на собственной статистике. Избежать «плохого» сценария выполнения позволяют подсказки. **Подсказка (hint)** — средство, позволяющее явным образом влиять на план запроса.

Сам SQL-запрос содержит указание, какую информацию необходимо получить из базы данных, но не содержит указаний, каким образом это делать. В общем случае, реляционные СУБД по собственным правилам определяют план запроса и, соответственно, его выполняют. Однако на практике может возникнуть случай, что такой план запроса, в силу неучтённых средствами СУБД факторов, несовершенства логики или особых требований может оказаться неоптимальным. Подсказка позволяет явно вмешаться в формирование плана запроса, не полагаясь полностью на автоматику [4].

2 Разработка системы

2.1 Системный анализ

Поскольку задача на реализацию пришла в виде бизнес-требования, было сформировано техническое задание. **Техническое задание (ТЗ)** – документ, где зафиксированы требования к решениям, которые должны быть реализованы в ходе создания программного обеспечения (ПО).

Требование к системе были не полные. В частности правила могут меняться, удалятся и добавятся с течением времени (перечень правил в ТЗ был опушен). В связи с этим была выбрана гибкая методология разработки (Agile). Agile (гибкая методология разработки) - это философия призванная увеличить эффективность модели разработки.

Данная методология позволит работать с нечеткими бизнес требованиями, показать работающий продукт как можно раньше и внедрять модули (привила) постепенно.

Архитектура системы, исходя из технического задания, состоит из пакета содержащего модули (правила) и реестра, куда будут записываться данные по итогу работы модулей системы.

При анализе кредитного портфеля было решено секционировать основную таблицу `CPM_TRIGGERS_CREDIT_BLANKS` по полю `CPM_TRIGGER`. Портфель кредитных анкет составляет около 16 млн. шт. с начала 2018 года. Количество правил согласно ТЗ равно 64 шт. На сегодняшний день количество записей в основной таблице будет составлять $16000000 \times 64 = 1024000000$ шт. Каждое новое правило, в дальнейшем, будет добавлять к таблице, как минимум, 16 млн. записей.

2.2 Объекты БД

2.2.1 Создание таблиц

Таблица - это объект базы данных состоящий из совокупности связанных данных, хранящихся в структурированном виде. Она состоит из столбцов и строк [6].

Реестр системы состоит из четырех таблиц: основная таблица, справочник правил, сводная таблица и журнал работы правил.

Основная таблица `CPM_TRIGGERS_CREDIT_BLANKS` содержит правила, анкеты и признаки по ним в нормальной форме.

Таблица справочник CPM_TRIGGERS_DICT помимо расшифровки привил, будет выполнять функцию управляющей таблицы. Каждое правило в системе обращается к справочнику и проверяет признак активности в столбце IS_ACTIVE.

Сводная таблица CPM_TCB_PIVOT необходима, для агрегации данных из основной таблицы и ускорения поиска сработавших правил на анкету для пользователей.

Журнал CPM_TRIGGERS_LOGS собирает данные по итогам работы правил. Таблица поможет осуществлять мониторинг работы системы.

2.2.2 Создание индексов

Для основной таблицы CPM_TRIGGERS_CREDIT_BLANKS создаем локальный секционированный индекс по полю IDBLANK. Уникальным ключом таблицы является совокупность полей IDBLANK и CPM_TRIGGER.

Индекс для словаря CPM_TRIGGERS_DICT, на данный момент, не требуется, поскольку количество записей в таблице относительно мало.

Для сводной таблицы CPM_TCB_PIVOT создаем два глобальных индекса по полям IDBLANK и IDCLIENT. Поле IDBLANK является уникальным ключом таблицы.

2.2.3 Создание пакета

Пакет — это объект базы данных, который группирует логически связанные типы, программные объекты и подпрограммы PL/SQL [7].

Пакет состоит из двух частей: спецификация и тело. Спецификация пакета - это интерфейс с объявленными приложениями из тела. Тело пакета определяет подпрограммы и содержит скрытые детали реализации.

В теле пакета правила будут реализованы как отдельные хранимые процедуры. Хранимая процедура — объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере [8].

В основной процедуре RUN_JOBS будем запускать расчет каждой группы правил в параллельных сессиях за счет использования самоуничтожающихся заданий. Каждая итерация запуска заданий нумерует их с помощью последовательности (SEQUENCE) SEQ_CPM_TRIGGERS. Последовательность (SEQUENCE) – это объект базы данных, который генерирует целые

числа в соответствии с правилами, установленными во время его создания [9].

2.2.4 Создание заданий (JOBS)

Расчет правил в системе будет запускаться ежедневно, автоматически по расписанию. Расчет основного реестра будет начинаться в 20:00 по МСК в задании `J_CPM_TRIGGERS`.

Расчет сводной таблицы будет начинаться в 6:00 по МСК в задании `J_CPM_TRIGGERS_PIVOT`.

2.3 Формирование запросов к БД

2.3.1 Оператор MERGE

Сердцем каждого правила будет конструкция `MERGE`. MERGE - DML-оператор языка SQL, который позволяет слить данные одной таблицы с данными другой таблицы или подзапроса. При слиянии проверяется условие, и если оно истинно, то выполняется `UPDATE`, а если нет - `INSERT`.

2.3.2 Конструкция CASE

Для преобразования данных из источника внутри `MERGE` будет использоваться подзапрос в предложении USING. Основой данного запроса является конструкция `CASE`. CASE - это инструкция, которая проверяет список условий и возвращает соответствующий результат. Если говорить в целом о программировании, то `CASE` - это что-то вроде многократного использования конструкции `IF-ELSE` [10].

`CASE` возвращает результат первого выражения `THEN`, условие которого выполнилось, т.е. `WHEN` возвращает TRUE. Таким образом, если `CASE` содержит несколько эквивалентных условий `WHEN`, которые будут возвращать TRUE, вернется результат (указанный в `THEN`) первого выражения [11].

2.3.3 Групповые функции

Для получения данных один к одному будем использовать групповую функцию. Групповые функции работают на наборах строк, чтобы выдать один результат на группу.

Как правило групповые функции используют с оператором `GROUP BY`, он позволяет группировать результат функции по полям (столбцам) для получения результирующих значений в каждой группе.

2.3.4 Аналитические функции

Основной целью аналитических функций является увеличение скорости выполнения запросов выявляющих внутренние отношения и зависимости в данных. Дополнительно они позволяют дать лаконичную формулировку «аналитическим запросам».

Основное преимущество использования аналитических (оконных) функций над агрегационными функциями заключается в следующем: оконные функции не приводят к группированию строк в одну строку вывода, строки сохраняют свои отдельные идентификаторы, а агрегированное значение добавляется к каждой строке. При этом групповые функции могут быть использованы как оконные.

Предложение *PARTITION BY* определяет столбец, по которому будет производиться группировка, и он является ключевым в разбиении набора строк на окна. Вместе с *PARTITION BY* может применяться предложение *ORDER BY*, которое определяет порядок сортировки внутри окна. Порядок сортировки очень важен, ведь оконная функция будет обрабатывать данные согласно этому порядку [12].

2.4 Журналирование

Функцию журналирования работы правил выполняет отдельная процедура внутри соответствующего пакета.

Вызов процедуры осуществляется на последнем этапе работы каждого правила. На вход она принимает три параметра: начало работы правила, номер правила и сумму модифицированных строк. Внутри себя процедура вычисляет количество минут работы правила и проверяет целостность реестра. Результат записывает в журнал `CPM_TRIGGERS_LOGS`.

Следующий этап - реализовать ежедневное информирование по результатам работы системы с помощью электронной почты. Формировать отчетность будем в отдельном пакете предназначенным для отправки подобных писем. Для наглядности отчет сформируем в формате HTML.

После того как отчет будет сформирован, он отправляется электронным письмом на ответственных сотрудников. Тем самым происходит ежедневный мониторинг работы системы.

2.5 Принцип работы системы

После завершения разработки, но перед введением системы в эксплуатацию, требуется внести в реестр исторические данные. Это обязательный этап, поскольку правила захватывают анкеты только за последние 30 дней. Такое ограничение глубины выборки требуется в целях быстродействия работы системы.

Следующий этап - тестирование, т.е. проверка записанных в реестр данных на корректность. Дополнительно требуется сверить данные основной и сводной таблиц.

Перед началом эксплуатации системы, пользователи должны ознакомиться с принципами её работы. Для этого была создана техническая документация. Данный документ описывает: технологию работы системы, инструкцию по её эксплуатации и применяемую бизнес-логику.

Для ответственных сотрудников, чья задача обслуживать систему, была создана техническая блок-схема, которая показывает взаимосвязь объектов системы. Наличие такого описания поможет новым сотрудникам, которые не знакомы с архитектурой системы вникнуть в принцип её работы.

ЗАКЛЮЧЕНИЕ

В настоящей работе был рассмотрен процесс разработки программного обеспечения на примере автоматизированной модульной anifraud системы. Для реализации поставленной задачи, потребовалось пройти несколько этапов. Началом проекта, послужило формирование бизнес-требования заказчиком. На первом этапе потребовалось провести анализ предметной области. Вторым этапом стало написание технического задания. На заключительном этапе, перед началом разработки, потребовалось сконструировать архитектуру будущей системы.

В ходе работы над проектом, помимо улучшения навыков применения языка SQL, был получен опыт:

- Проектирование модульных систем;
- Сбор и формулирование требований;
- Формирование объектов базы данных;
- Применение принципов business intelligence;
- Автоматизация процесса;
- Оптимизация запросов к базе данных;
- Написание технической документации;
- Руководство реальным IT-проектом.

Подводя итоги, необходимо подчеркнуть, что данный проект реализует лишь часть мер по снижению финансовых рисков организации. Для более эффективного подхода стоит применять комплекс различных мер по минимизации операционных рисков.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Коннолли, Т.* Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Т. Коннолли, К. Бегг. — М.: Вильямс, 2017. — 1440 с.

2 *Грофф, Д. Р.* SQL. Полное руководство / Д. Р. Грофф, П. Н. Вайнберг, Э. Д. Оппель. — М.: Диалектика, 2019. — 960 с.

3 *Дейт, К. Д.* Введение в системы баз данных / К. Д. Дейт. — М.: Вильямс, 2005. — 1328 с.

4 *Тарасов, С. В.* СУБД для программиста. Базы данных изнутри / С. В. Тарасов. — М.: Солон-Пресс, 2015. — 320 с.

5 *Фейерштейн, С.* Oracle PLSQL. Программирование в Oracle для профессионалов / С. Фейерштейн, Б. Прибыл. — СПб.: Питер, 2015. — 1024 с.

6 *Форта, Б.* Oracle PLSQL за 10 минут / Б. Форта. — М.: Диалектика-Вильямс, 2019. — 336 с.

7 *Мак-Локлин, М.* Oracle Database 11g. Программирование на языке PLSQL / М. Мак-Локлин. — М.: Лори, 2017. — 880 с.

8 *Кайт, Т.* Oracle для профессионалов. Архитектура, методики программирования и основные особенности версий 9i, 10g, 11g и 12c / Т. Кайт, Д. Кун. — М.: Вильямс, 2019. — 960 с.

9 *Бьюли, А.* Секреты Oracle SQL / А. Бьюли, С. Мишра. — М.: Символ-Плюс, 2006. — 368 с.

10 *Форта, Б.* Язык T-SQL для Microsoft SQL Server за 10 минут / Б. Форта. — М.: Вильямс, 2017. — 384 с.

11 *Кайт, Т.* Эффективное проектирование приложений Oracle / Т. Кайт. — М.: Лори, 2008. — 656 с.

12 *Гринвальд, Р.* Oracle 11g. Основы / Р. Гринвальд, Р. Стаковьяк, Д. Стерн. — М.: Символ-Плюс, 2009. — 464 с.