МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий
Кафедра информационных систем и технологий
Специальность 1-40 05 01 «Информационные системы и технологии»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА

по дисциплине «Базы данных»
Тема: «База данных туристического агенства»

Исполнитель
студент 2 курса 1 группы _____ Васильев В. В.
подпись, дата

Руководитель
Ассистент _____ Савельева М. Г.
должность, учен. степень, ученое звание подпись, дата

Допущен(а) к защите _____
_____ _____
дата, подпись

Курсовой проект защищен с оценкой _____

Руководитель _____ _____ Савельева М. Г.
подпись дата инициалы и фамилия

Минск 2025

Содержание

Введение

В современном мире туристическая индустрия развивается стремительными темпами. Люди активно пользуются туристическими услугами туристических агентств для организации путешествий, бронирования отелей, экскурсий и других мероприятий. Для эффективного управления такими процессами необходимо использовать современные информационные системы и базы данных.

Целью данного курсового проекта является разработка базы данных для туристического агентства, обеспечивающей хранение и обработку информации о клиентах, турах, бронированиях и платежах.

Основные задачи проекта:

- Разработка структуры базы данных, отвечающей требованиям агентства;
- Реализация механизмов бронирования и управления клиентами;
- Внедрение системы мониторинга и анализа данных;
- Обеспечение защиты и целостности данных;
- Применить технологию OLAP-хранилища.

Созданная база данных будет способствовать автоматизации работы туристического агентства, повышению скорости обработки заказов и улучшению качества обслуживания клиентов.

Для разработки базы данных была выбрана СУБД Oracle Database.

1 Постановка задачи

Туристическое агентство занимается организацией поездок, предоставлением туров, бронированием гостиниц, оформлением билетов и дополнительных услуг для клиентов. В процессе работы агентство взаимодействует с клиентами, отелями, авиакомпаниями и другими партнёрами.

В настоящее время управление этими процессами может осуществляться вручную или с использованием устаревших систем, что приводит к следующим проблемам:

– Неэффективное ведение базы клиентов и их заказов;
– Длительное время обработки бронирований;
– Отсутствие централизованного контроля за выполнением заказов;
– Сложность в анализе данных и формировании отчётности.

База данных должна стать основой для работы автоматизированной информационной системы туристического агентства, повысив ее производительность и удобство использования.

1.1 Обзор аналогичных решений

Туристический бизнес в Беларуси активно развивается, и всё больше людей предпочитают обращаться в проверенные агентства, чтобы организовать свой отдых с минимальными рисками и максимальным комфортом. Среди множества компаний, предоставляющих туристические услуги, особенно выделяются три агентства: Слетаем.by, ИнтерСити и География путешествий[1-3]. Они пользуются широкой популярностью благодаря удобным онлайн-сервисам, разнообразию предложений и профессиональному подходу к работе с клиентами. В данном обзоре мы детально рассмотрим каждое из агентств, их особенности, преимущества, а также возможные недостатки.

1.1.1 Аналог – Слетаем.by

Сервис Слетаем.by предоставляет пользователям возможность подбора и бронирования туров от различных туроператоров. Пользователь может указать страну, даты, количество человек, категорию отеля и другие параметры для поиска подходящих туров. Также сайт предлагает фильтрацию по цене, рейтингу и отзывам.

Предполагаемая структура базы данных:

- Users — данные о клиентах (ID, имя, контакты, история заказов).
- Tours — туры (ID, страна, город, отель, даты, стоимость, количество мест).
- Bookings — бронирования (ID, ID пользователя, ID тура, дата брони, статус).
- TourOperators — туроператоры (ID, название, рейтинг, контакты).
- Hotels — отели (ID, название, звёзды, расположение, услуги).
- Reviews — отзывы клиентов (ID, ID пользователя, ID тура, оценка, текст).

Предполагаемые связи:

- Один пользователь может сделать несколько бронирований — Users ↔ Bookings (один ко многим).

- Один туроператор предоставляет несколько туров — TourOperators ↔ Tours (один ко многим).
- Один отель может участвовать в нескольких турах — Hotels ↔ Tours (один ко многим).
- Каждый тур может иметь несколько отзывов — Tours ↔ Reviews (один ко многим).

Предполагаемая индексация и оптимизация:

- Индексы по полям: страна, даты вылета, категория отеля, цена.
- Использование материализованных представлений для популярных направлений.
- Разделение таблиц Bookings и Tours по дате или региону.

Импорт и экспорт данных:

- Импорт туров от туроператоров через XML/JSON.
- Экспорт истории бронирований и отзывов в Excel или PDF для отчётности.

1.1.2 Аналог – ИнтерСити

Сервис ИнтерСити специализируется на подборе автобусных и экскурсионных туров по Беларуси и ближнему зарубежью. Предлагаются авторские туры, информация о гидах и рейтинги.

Предполагаемая структура базы данных:

- Users — клиенты (ID, ФИО, email, телефон, история заказов).
- Tours — туры (ID, маршрут, даты, стоимость, количество мест, тип тура).
- Bookings — бронирования (ID, ID пользователя, ID тура, дата, количество людей).
- TourGuides — гиды (ID, ФИО, опыт, языки, рейтинг).
- TourTypes — типы туров (ID, название, описание).
- Reviews — отзывы о турах и гидах (ID, автор, оценка, комментарий).

Предполагаемые связи:

- Один пользователь → несколько бронирований.
- Один тур может быть привязан к одному гиду — Tours ↔ TourGuides (многие к одному).
- Один тип тура используется многими турами — TourTypes ↔ Tours (один ко многим).
- Каждый тур может иметь множество отзывов.

Индексация и оптимизация:

- Индексы по дате начала тура, маршруту, гиду.
- Оптимизация через партиционирование Bookings по дате.
- Кэширование расписания популярных туров.

Импорт/экспорт:

- Импорт расписаний туров в формате CSV.
- Экспорт клиентской базы и заказов для анализа в BI-системы.

1.1.3 Аналог – География путешествий

«География путешествий» — белорусское туристическое агентство, предлагающее отдых по всему миру. Есть детализированный подбор туров, персонализированные предложения, отзывы, акции и поддержка по телефону.

Предполагаемая структура базы данных:

- Users — клиенты (ID, имя, контакты, дата регистрации).
- Tours — туры (ID, направление, отель, даты, цена, тип питания).
- Bookings — заявки на бронирование (ID клиента, ID тура, количество человек, дата заявки).
- Promotions — акции и скидки (ID, описание, срок действия, связанные туры).
- Messages — переписка с менеджерами (ID, отправитель, получатель, текст, дата).
- Feedback — обратная связь после поездки (ID, ID клиента, оценка, отзыв).

Связи:

- Один пользователь может участвовать во многих переписках — Users ↔ Messages.
- Один тур может иметь несколько скидок — Tours ↔ Promotions (многие ко многим).
- Один пользователь может оставить один отзыв на тур — Feedback ↔ Users, Tours.

Индексация и оптимизация:

- Индексы по дате тура, стране, цене, названию акции.
- Оптимизация поиска по названию стран через полнотекстовый поиск.
- Архивация старых сообщений и заявок.

Импорт/экспорт:

- Импорт туров из Excel/CSV от поставщиков.
- Экспорт акций и отзывов на сайт или в соцсети.

1.2 Требования к проекту

На основе анализа аналогов можно выделить ключевые функциональные требования для разрабатываемой базы данных:

– Управление клиентами: регистрация, хранение данных, отслеживание заказов.

– Управление бронированием: создание, редактирование, отмена и подтверждение заказов.

– Поддержка платёжных операций: интеграция с платёжными системами.

– Аналитика и отчетность: генерация статистики по продажам, популярным направлениям, клиентам.

– Управление пользователями: разграничение доступа между администраторами, менеджерами и клиентами.

1.3 Вывод по разделу

Анализ существующих решений показал, что глобальные системы бронирования (GDS) обладают широким функционалом, но требуют значительных финансовых затрат и сложной интеграции. Специализированные CRM-системы для туристических агентств обеспечивают удобное управление клиентами и бронированиями, но могут быть ограничены в функционале.

Собственная база данных позволит туристическому агентству гибко управлять процессами бронирования, учитывать потребности клиентов и предоставлять удобный интерфейс для сотрудников. Это обеспечит автономность в управлении заказами, снизит зависимость от сторонних сервисов и позволит адаптировать систему под внутренние бизнес-процессы.

2 Разработка архитектуры проекта

2.1 Определение вариантов использования

На этапе проектирования базы данных важно определить основные сценарии взаимодействия пользователя с системой. Эти сценарии описывают, какие действия могут выполнять различные категории пользователей в рамках функционала базы данных туристического агентства.

Система предусматривает наличие трех основных ролей:

– Администратор – управляет системой в целом, следит за целостностью данных, занимается созданием и удалением учетных записей пользователей, контролирует аналитические отчеты, а также имеет доступ к просмотру и редактированию всех записей.

– Менеджер – сотрудник агентства, который работает с клиентами, оформляет бронирования, регистрирует оплаты, отслеживает маршруты и взаимодействует с отзывами.

– Клиент – конечный пользователь, осуществляющий поиск туров, просмотр предложений, оформление бронирований, а также оплату и оставление отзывов.

Общие сценарии для всех ролей:

– Просмотр туров с возможностью фильтрации и сортировки;

– Просмотр информации об отелях, маршрутах, рейтингах и скидках;

– Доступ к истории действий (в рамках своих прав);

Сценарии для клиента:

– Регистрация и авторизация в системе;

– Поиск туров по дате, цене, стране, категории;

– Оформление бронирования на выбранный тур;

– Просмотр своих заказов и платежей;

– Отмена или редактирование бронирования (если не подтверждено);

– Оплата тура;

– Оставление отзыва на тур после завершения поездки;

– Получение и отправка сообщений менеджерам.

Сценарии для менеджера:

– Создание и редактирование туров, указание маршрута, страны, даты, стоимости, скидок;

– Привязка туров к отелям и загрузка изображений;

– Управление бронированиями клиентов: подтверждение, отмена, изменения статуса;

– Добавление платежей вручную при поступлении средств;

– Ответы на отзывы, коммуникация с клиентами;

– Просмотр и анализ загрузки туров.

Сценарии для администратора:

– Управление всеми пользователями: создание, изменение ролей, блокировка;

– Удаление или корректировка данных в случае ошибок;

– Просмотр отчетов по продажам, туроператорам, отзывам;

– Контроль за действиями менеджеров;
– Модерация отзывов (при необходимости);
– Полный доступ ко всем функциям системы.

Таким образом, структура вариантов использования охватывает все основные бизнес-процессы туристического агентства и служит основой для построения логики базы данных, разграничения прав доступа и построения интерфейса будущих компонентов системы.

2.2 Диаграммы UML, взаимодействие всех компонентов

Для визуализации структуры базы данных и взаимодействия её компонентов были разработаны диаграммы UML. Данный подход позволяет формализовать логику работы информационной системы, а также облегчить процесс проектирования и понимания архитектуры базы данных.

В рамках проекта были построены следующие диаграммы:

– Диаграмма вариантов использования (Use Case Diagram) (Приложение А) – отражает действия, доступные пользователям различных ролей: администратора, менеджера и клиента. Она показывает, какие функции доступны каждому типу пользователя в системе бронирования туров.

– Диаграмма классов (Class Diagram) (Приложение А) – отображает основные сущности базы данных, их атрибуты и связи между ними. В данном проекте в качестве классов выступают таблицы базы данных, такие как AppUsers, Tours, Bookings, Payments, TourReviews и другие. Связи между классами соответствуют внешним ключам, реализованным в структуре базы данных.

– Диаграммы компонентов (Component Diagram) (Рисунок 2.1) – показывает структуру системы на уровне модулей и их взаимодействие. Включает логические блоки: работа с пользователями, управление турами, бронирование, платежи, маршруты, отзывы и сообщения. Каждому модулю соответствует один или несколько объектов базы данных, обеспечивающих реализацию конкретной подсистемы.

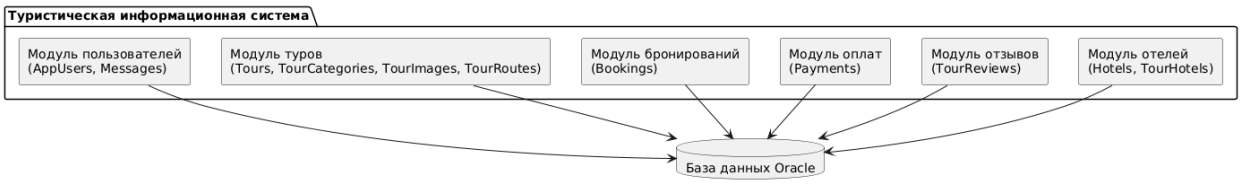

Рисунок 2.1 – Диаграмма компонентов

– Диаграмма последовательности (Sequence Diagram) (Рисунок 2.2) – используется для демонстрации хода выполнения одного из ключевых сценариев, таких как оформление бронирования. На ней отображены участники процесса (пользователь, система, таблицы базы данных) и последовательность сообщений, передаваемых между ними.

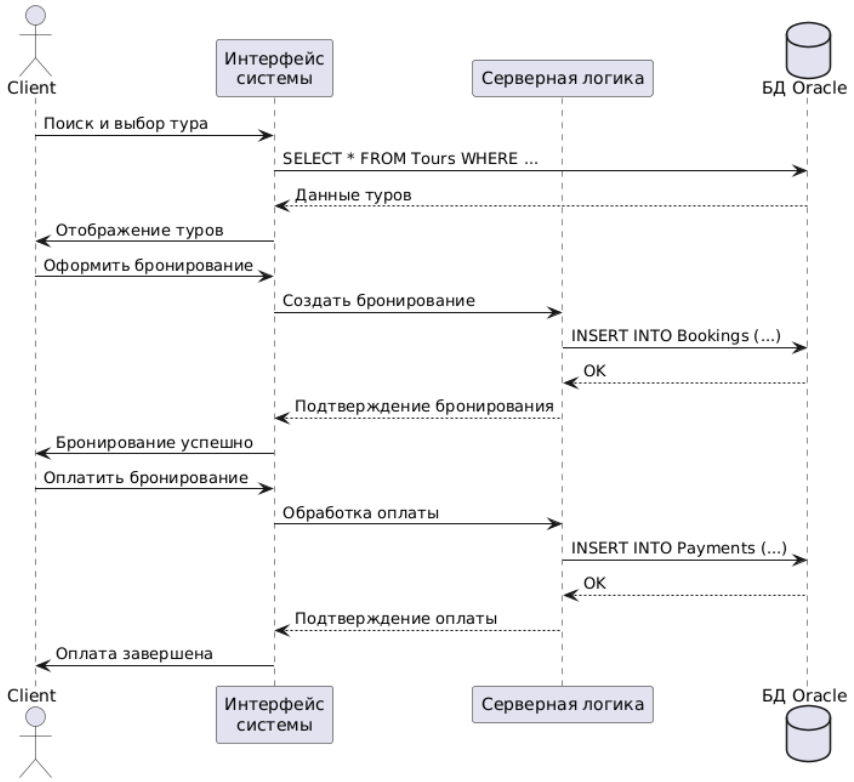

Рисунок 2.2 – Диаграмма последовательности

Разработка диаграмм была выполнена с использованием языка моделирования UML и графических средств, что позволяет быстро и наглядно визуализировать все аспекты архитектуры базы данных. Созданные диаграммы служат основой для реализации логической и физической моделей базы, а также обеспечивают единое понимание проекта всеми участниками разработки.

2.3 Выводы по разделу

В рамках проектирования базы данных туристического агентства были определены ключевые сценарии использования системы для различных категорий пользователей: клиента, менеджера и администратора. Это позволило выделить функциональные зоны, каждая из которых отвечает за выполнение определенного набора задач, связанных с бронированием туров, оплатой, управлением отзывами и взаимодействием между пользователями.

Разработанные диаграммы UML обеспечили формализацию архитектуры системы и наглядно отразили связи между сущностями, логическими модулями и действиями пользователей. Диаграмма классов позволила визуализировать структуру базы данных, в то время как диаграмма последовательностей отразила алгоритм оформления и оплаты бронирования. Диаграмма компонентов показала общую организацию функциональных блоков системы, обеспечивающих модульность для реализации ее объектов

Таким образом, этап проектирования позволил сформировать полное представление о логике функционирования базы данных и подготовить прочную основу для реализации ее объектов.

3 Разработка объектов базы данных

3.1 Разработка таблиц базы данных

На этапе реализации базы данных были созданы основные таблицы, соответствующие ранее разработанной логической модели. Каждая таблица предназначена для хранения информации, необходимой для функционирования туристического агентства, с учетом требований к целостности, уникальности и связности данных. Детальное описание всех таблиц представлено в приложении Б.

Структура таблиц сформирована в соответствии с нормализованной моделью. В таблицах применяются первичные и внешние ключи, ограничения по типам и значениям данных, а также механизмы автоматической генерации идентификаторов с использованием *GENERATE BY DEFAULT AS IDENTITY*

SQL-запрос создания таблицы AppUsers предоставлен в листинге 3.1.

```
CREATE TABLE AppUsers (
    UserId NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    FullName VARCHAR2(100),
    Email VARCHAR2(100) UNIQUE,
    PasswordHash VARCHAR2(200),
    Role VARCHAR2(50)
);
```

Листинг 3.1 – Запрос создания таблицы AppUsers

SQL-запросы для создания всех таблиц предоставлены в приложении В.

Каждая таблица была создана с учетом следующих требований:

– Использование уникальных идентификаторов в качестве первичных ключей.

– Обязательное определение внешних ключей для обеспечения ссылочной целостности.

– Применение ограничений CHECK (например, допустимый диапазон рейтинга или звезд отеля).

– Установка значений по умолчанию для повышения удобства работы с данными.

Таким образом, на данном этапе была построена основа базы данных, обеспечивающая хранение, связность и структурность всей информации, необходимой для автоматизации процессов туристического агентства.

3.2 Разработка схем базы данных

Проектирование и реализация схемы базы данных осуществлялись на основе предварительного анализа предметной области, сформулированных требований и описанной структуры системы.

База данных создавалась и настраивалась в пользовательской контейнерной базе данных (PDB) под названием TRAAGPDB, развёрнутой в среде Oracle Database. Это решение позволило обеспечить изолированную среду для хранения и управления данными, а также упростить администрирование и масштабирование.

Для проектирования схемы были определены основные сущности, связи между ними, а также реализованы ограничения целостности данных. Модель была

реализована с соблюдением нормальных форм и ориентирована на дальнейшее развитие системы.

Таким образом, выбранная организация схемы базы данных обеспечивает простоту, целостность и возможность масштабирования в будущем.

3.3 Разработка процедур базы данных

Для автоматизации типовых операций в базе данных туристического агентства были разработаны хранимые процедуры. Они позволяют централизовать бизнес-логику, снизить дублирование кода и повысить безопасность выполнения операций, особенно при работе с бронированиями и оплатами.

Хранимые процедуры реализованы на языке PL/SQL, встроенном в Oracle Database. Ниже приведены ключевые процедуры, реализованные в рамках проекта.

1 MakeBooking – оформление бронирования. Позволяет создать новую запись в таблице Booking, указав идентификаторы пользователя и тура.

```
CREATE OR REPLACE PROCEDURE MakeBooking (
    p_UserId IN NUMBER,
    p_TourId IN NUMBER
) AS
BEGIN
    INSERT INTO Bookings (UserId, TourId, BookingDate, Status)
    VALUES (p_UserId, p_TourId, SYSDATE, 'Pending');
END;
```

Листинг 3.2 – Процедура MakeBooking

2 ConfirmBookingPayment — подтверждение оплаты бронирования. Обновляет статус бронирования и добавляет запись об оплате.

3 CancelBooking — отмена бронирования.Изменяет статус бронирования на «Cancelled».

4 LeaveReview — оставление отзыва о туре. Позволяет добавить отзыв с текстом и рейтингом.

5 AddTour — добавление нового тура менеджером.

6 AddRouteToTour — добавление маршрута к туру.

7 AddHotel — добавление отеля.

8 LinkHotelToTour — привязка отеля к туру.

9 SendMessage — отправка сообщения.

Sql-запросы создания всех процедур предоставлены в приложении Г.

Разработка хранимых процедур позволила упростить выполнение часто используемых операций и обеспечить соблюдение логики бизнес-процессов на уровне СУБД.

3.4 Разработка функций базы данных

Функции в Oracle PL/SQL используются для выполнения вычислений и возврата значений, которые могут быть использованы в SQL-запросах, процедурах и представлениях. В рамках проекта были разработаны пользовательские функции, реализующие полезные аналитические операции и агрегатные расчёты на уровне базы данных.

1 GetAverageTourRating — получение среднего рейтинга тура**.** Функция возвращает среднюю оценку по всем отзывам для заданного тура.

2 GetUserBookingCount — количество бронирований клиента. Функция возвращает общее количество бронирований, оформленных конкретным пользователем.

4 GetTourRevenue — суммарная выручка по туру**.** Функция возвращает общую сумму оплат, полученных за конкретный тур.

Sql-запросы создания всех процедур предоставлены в приложении Д.

Разработка пользовательских функций позволяет упростить реализацию аналитических операций, повысить читаемость SQL-запросов и обеспечить повторное использование кода.

3.5 Разработка представлений базы данных

Представления (views) в Oracle позволяют создавать виртуальные таблицы, формируемые на основе одного или нескольких SQL-запросов. Они используются для упрощения доступа к связанным данным, улучшения читаемости сложных запросов и предоставления пользователям только необходимой информации без доступа к исходным таблицам.

В рамках проекта были разработаны следующие представления, отражающие ключевые бизнес-сценарии туристического агентства.

1 View_BookingDetails — детализированное представление по бронированиям. Объединяет данные о клиентах, турах и статусе бронирования.

```
CREATE OR REPLACE VIEW View_BookingDetails AS
SELECT
    B.BookingId,
    B.BookingDate,
    B.Status AS BookingStatus,
    U.FullName AS ClientName,
    T.ShortTitle AS TourName,
    T.Price,
    T.StartDate,
    T.EndDate
FROM Bookings B
JOIN AppUsers U ON B.UserId = U.UserId
JOIN Tours T ON B.TourId = T.TourId;
```

Листинг 3.3 – Процедура View_BookingDetails

2 View_PaymentSummary — представление по оплатам**.** Позволяет быстро получить информацию об оплатах, включая сумму и клиента.

```
CREATE OR REPLACE VIEW View_PaymentSummary AS
SELECT
    P.PaymentId,
    P.PaymentDate,
    P.Amount,
    P.PaymentMethod,
    B.BookingId,
    U.FullName AS ClientName
FROM Payments P
JOIN Bookings B ON P.BookingId = B.BookingId
```

```
JOIN AppUsers U ON B.UserId = U.UserId;
```

Листинг 3.4 – Процедура View_PaymentSummary

3 View_TourReviewsExtended — отзывы с деталями**.** Показывает отзывы клиентов вместе с названием тура и именем пользователя.

```
CREATE OR REPLACE VIEW View_TourReviewsExtended AS
SELECT
    R.ReviewId,
    R.Rating,
    R.ReviewComment,
    R.ReviewDate,
    U.FullName AS Reviewer,
    T.ShortTitle AS Tour
FROM TourReviews R
JOIN AppUsers U ON R.UserId = U.UserId
JOIN Tours T ON R.TourId = T.TourId;
```

Листинг 3.5 – Процедура View_TourReviewsExtended

Создание представлений позволило сократить повторение SQL-запросов в отчётах и упростить получение агрегированной информации. Представления также могут использоваться как основа для экспорта данных или подтверждения внешних аналитических систем.

3.6 Разработка триггеров базы данных

Триггеры в Oracle используются для автоматического выполнения заданных действий при наступлении определённых событий в базе данных, таких как вставка, обновление или удаление строк. Их применение позволяет реализовать бизнес-логику на уровне СУБД, обеспечивая реакцию системы без необходимости внешнего вмешательства.

В рамках проекта туристического агентства были реализованы следующие триггеры:

1 trg_UpdateTourRating — автоматический пересчёт среднего рейтинга тура при новом отзыве**.** После добавления нового отзыва рейтинг в таблице Tours обновляется на основе среднего значения всех оценок по данному туру.

```
CREATE OR REPLACE TRIGGER trg_UpdateTourRating
AFTER INSERT ON TourReviews
FOR EACH ROW
BEGIN
    UPDATE Tours
    SET Rating = (
        SELECT ROUND(AVG(Rating))
        FROM TourReviews
        WHERE TourId = :NEW.TourId
    )
    WHERE TourId = :NEW.TourId;
END;
```

Листинг 3.6 – Триггер trg_UpdateTourRating

2 trg_LogBookingStatusChange — логирование изменений статуса бронирования. Создаётся журнал, фиксирующий дату и пользователя, изменившего статус брони. Для этого создаём сначала таблицу логов:

```
CREATE TABLE BookingStatusLog (
    LogId NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    BookingId NUMBER,
    OldStatus VARCHAR2(50),
    NewStatus VARCHAR2(50),
    ChangedAt DATE DEFAULT SYSDATE
);
```

Листинг 3.7 – Таблица BookingStatusLog

И сам триггер:

```
CREATE OR REPLACE TRIGGER trg_LogBookingStatusChange
BEFORE UPDATE OF Status ON Bookings
FOR EACH ROW
WHEN (OLD.Status != NEW.Status)
BEGIN
    INSERT INTO BookingStatusLog (BookingId, OldStatus, NewStatus)
    VALUES (:OLD.BookingId, :OLD.Status, :NEW.Status);
END;
```

Листинг 3.8 – Триггер trg_LogBookingStatusChange

3 trg_DefaultReviewRating — установка значения по умолчанию, если рейтинг не указан.

```
CREATE OR REPLACE TRIGGER trg_DefaultReviewRating
BEFORE INSERT ON TourReviews
FOR EACH ROW
BEGIN
    IF :NEW.Rating IS NULL THEN
        :NEW.Rating := 3;
    END IF;
END;
```

Листинг 3.9 – Триггер trg_DefaultReviewRating

Триггеры позволяют автоматизировать служебные процессы, гарантировать актуальность агрегированных данных (например, рейтингов) и обеспечивают аудит значимых событий в базе.

3.7 Роли и пользователи

Для обеспечения безопасного доступа к данным и реализации разграничения прав в базе данных туристического агентства была разработана ролевая модель. В Oracle Database роли используются для группировки прав доступа и последующего назначения этих прав конкретным пользователям.

В системе предусмотрены следующие роли:

1 client_role – роль клиента

Назначается пользователям, которые бронируют туры и взаимодействуют с интерфейсом как конечные потребители услуг.

Права:

– Чтение данных о турах, маршрутах, отелях и изображениях;
– Просмотр своих бронирований;
– Добавление отзывов;
– Отправка сообщений.

```
CREATE ROLE client_role;

GRANT SELECT ON Tours TO client_role;
GRANT SELECT ON TourRoutes TO client_role;
GRANT SELECT ON Hotels TO client_role;
GRANT SELECT ON TourImages TO client_role;
GRANT SELECT, INSERT ON Bookings TO client_role;
GRANT SELECT, INSERT ON Payments TO client_role;
GRANT SELECT, INSERT ON TourReviews TO client_role;
GRANT SELECT, INSERT ON Messages TO client_role;
```

Листинг 3.10 – Роль client_role

2 manager_role – роль менеджера

Предназначена для сотрудников агентства, работающих с клиентами и управляющих бронированиями.

Права:

– Полный доступ к бронированиям и оплатам;
– Управление турами, маршрутами, отелями и изображениями;
– Ответы на сообщения и модерация отзывов.

```
CREATE ROLE manager_role;

GRANT SELECT, INSERT, UPDATE, DELETE ON Bookings TO manager_role;
GRANT SELECT, INSERT, UPDATE, DELETE ON Payments TO manager_role;
GRANT SELECT, INSERT, UPDATE, DELETE ON Tours TO manager_role;
GRANT SELECT, INSERT, UPDATE, DELETE ON TourRoutes TO
manager_role;
GRANT SELECT, INSERT, UPDATE, DELETE ON TourHotels TO
manager_role;
GRANT SELECT, INSERT, UPDATE, DELETE ON TourImages TO
manager_role;
GRANT SELECT, INSERT, UPDATE ON TourReviews TO manager_role;
GRANT SELECT, INSERT, UPDATE ON Messages TO manager_role;
```

Листинг 3.11 – Роль manager_role

3 admin_role – роль администратора

Администратор имеет полный контроль над всеми объектами базы данных.

Права:

– Управление всеми таблицами и ролями;
– Мониторинг данных, отчётность, аудит;
– Управление пользователями.

```
CREATE ROLE admin_role;
GRANT ALL PRIVILEGES TO admin_role;
```

Листинг 3.12 – Роль admin_role

Разработка и назначение ролей позволяют четко разграничить права доступа к данным, обеспечивая безопасность, защиту персональной информации и контроль над бизнес-операциями в системе.

3.8 Вывод по разделу

На этапе разработки объектов базы данных была построена полноценная структура, охватывающая все аспекты работы туристического агентства. В результате:

– Спроектированы и созданы все необходимые таблицы, обеспечивающие хранение информации о пользователях, турах, маршрутах, отелях, бронированиях, платежах, отзывах и сообщениях.

– Определена схема хранения данных, обеспечивающая логическую группировку по модулям и поддерживающая целостность информации с помощью внешних ключей и ограничений.

– Разработаны хранимые процедуры для автоматизации базовых операций: бронирование тура, регистрация платежей, отмена заказов и добавления отзывов.

– Созданы пользовательские функции для вычисления агрегатов и аналитики, таких как средний рейтинг туров, количество бронирований и выручка.

– Реализованы представления, упрощающие доступ к связанной информации и обеспечивающие основу для отчетности.

– Настроены триггеры для автоматического пересчета рейтинга и аудита изменений статуса бронирований.

– Создана ролевая модель безопасности, разделяющая права доступа между администраторами, менеджерами и клиентами.

– Выполнено создание пользователей и назначение им соответствующих ролей в СУБД Oracle.

Таким образом, база данных полностью готова к эксплуатации и может использоваться как самостоятельная информационная система или как основа для дальнейшего подключения клиентских интерфейсов и аналитических модулей.

4 Описание процедур экспорта и импорта данных

Современные информационные системы требуют поддержки обмена данных с внешними источниками. Это позволяет облегчить загрузку справочной информации, обмен между системами, а также резервное копирование и восстановление. В рамках проекта была реализована возможность импорта данных их Excel-файлов и экспорта в формате XML, что позволяет интегрировать базу данных туристического агентства с внешними сервисами и упростить ввод/вывод информации.

4.1 Процедура импорта данных из Excel-файлов

Для импорта данных из Excel в Oracle DataBase использовался следующий подход:

– Excel-файл предварительно сохраняется в формате CSV/

– Сторонними средствами данные загружаются во временную таблицу.

– Затем с помощью PL/SQL данные переносятся в основные таблицы с валидацией.

Пример: Импорт стран из Excel

1 Excel-файл с данными о турах сохраняется в формате CSV (например, tours.csv).

2 Создается временная таблица для загрузки данных

```
CREATE TABLE temp_tours (
  SHORTTITLE   VARCHAR2(150),
  FULLTITLE    VARCHAR2(255),
  DESCRIPTION  CLOB,
  PRICE        NUMBER(10,2),
  STARTDATE    DATE,
  ENDDATE      DATE,
  RATING       NUMBER(3,0),
  ISAVAILABLE  NUMBER(1,0),
  DISCOUNT     NUMBER(5,2),
  COUNTRY      VARCHAR2(100),
  CATEGORYID   NUMBER
);
```

Листинг 4.1 – Временная таблица temp_tours

3 Загружается CSV-файл через SQL Developer.

4 После загрузки вызывается процедура импорта:

```
CREATE OR REPLACE PROCEDURE ImportToursFromTemp IS
BEGIN
  INSERT INTO Tours (
    SHORTTITLE, FULLTITLE, DESCRIPTION, PRICE, STARTDATE, ENDDATE,
    RATING, ISAVAILABLE, DISCOUNT, COUNTRY, CATEGORYID
  )
  SELECT
    SHORTTITLE, FULLTITLE, DESCRIPTION, PRICE, STARTDATE, ENDDATE,
    RATING, ISAVAILABLE, DISCOUNT, COUNTRY, CATEGORYID
  FROM temp_tours;
```

```
  COMMIT;
END;
```

Листинг 4.2 – Процедура ImportToursFromTemp

4.2 Процедура экспорта данных в XML-формат

Для экспорта информации о турах в XML-формат применяется пакет DBMS_XMLGEN. Данные могут сохраняться, например, в таблицу логов экспорта

```
CREATE TABLE export_xml_log (
  id         NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  exportdate DATE,
  xmldata    CLOB
);
```

Листинг 4.3 – Таблица export_xml_log

Процедура экспорта

```
CREATE OR REPLACE PROCEDURE ExportToursToXML IS
  ctx  DBMS_XMLGEN.ctxhandle;
  xml  CLOB;
BEGIN
  ctx := DBMS_XMLGEN.newcontext('SELECT * FROM Tours');
  DBMS_XMLGEN.setrowsettag(ctx, 'Tours');
  DBMS_XMLGEN.setrowtag(ctx, 'Tour');
  xml := DBMS_XMLGEN.getxml(ctx);

  INSERT INTO export_xml_log (exportdate, xmldata)
  VALUES (SYSDATE, xml);

  DBMS_XMLGEN.closecontext(ctx);
  COMMIT;
END;
```

Листинг 4.4 – Процедура ExportToursToXml

4.3 Вывод по разделу

Таким образом, в системе реализован импорт данных из Excel-файлов (CSV) через временную таблицу и автоматическую загрузку в основную таблицу Tours. Экспорт данных реализуется в XML-формате с помощью встроенного пакета DBMS_XMLGEN. эти процедуры позволяют упростить администрирование и расширить возможности системы по взаимодействию с внешними источниками данных

5 Тестирование производительности

5.1 Заполнение таблицы

Проверка производительности базы данных является важным этапом оценки её эффективности и готовности к работе в условиях реальной нагрузки. В рамках данного проекта было проведено тестирование операций фильтрации данных в таблице Tours с целью определения влияния индексации на скорость выполнения запросов.

На первом этапе в таблицу Tours было добавлено 100 000 тестовых записей. Для этого была написана PL/SQL-процедура, автоматизирующая генерацию данных с различными значениями полей, включая даты начала и окончания тура, цены, скидки, страны и категории. В листинге 5.1 представлен SQL-запрос для генерации тестовых данных и выполнения выборки.

```
BEGIN
  FOR i IN 1..100000 LOOP
    INSERT INTO Tours (
      TOURID, SHORTTITLE, FULLTITLE, DESCRIPTION, PRICE,
      STARTDATE, ENDDATE, RATING, ISAVAILABLE, DISCOUNT,
      COUNTRY, CATEGORYID
    ) VALUES (
      tour_seq.NEXTVAL,
      'ShortTitle ' || i,
      'FullTitle of Tour ' || i,
      'Description for Tour ' || i,
      ROUND(DBMS_RANDOM.VALUE(500, 5000), 2),
      SYSDATE + DBMS_RANDOM.VALUE(1, 30),
      SYSDATE + DBMS_RANDOM.VALUE(31, 60),
      MOD(i, 10),
      MOD(i, 2),
      ROUND(DBMS_RANDOM.VALUE(0, 30), 2),
      'Country ' || MOD(i, 5),
      MOD(i, 3) + 1
    );
  END LOOP;
  COMMIT;
END;
```

Листинг 5.1 – SQL-запрос для генерации тестовых данных

После заполнения таблицы была выполнена операция фильтрации записей, у которых значение поля Country равно ‘Country 3’. На рисунке 5.1 представлен SQL-запрос и план выполнения до создания индекса.

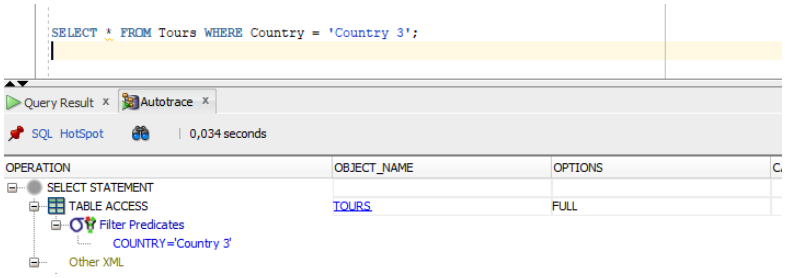

Рисунок 5.1 – План выполнения запроса без использования индекса

План запроса показал, что выполнение происходило через полное сканирование таблицы (Full Table Scan), что заняло 34 миллисекунд.

Для повышения производительности на поле Country был создан индекс, который представлен в листинге 5.2.

```
CREATE INDEX idx_tours_country ON Tours(Country);
```

Листинг 5.2 – Индекс на поле Country

После создания индекса план выполнения изменился: теперь используется поиск по индексу (Index Range Scan). Запрос стал выполняться за 8 миллисекунд, что существенно сократило время обработки.

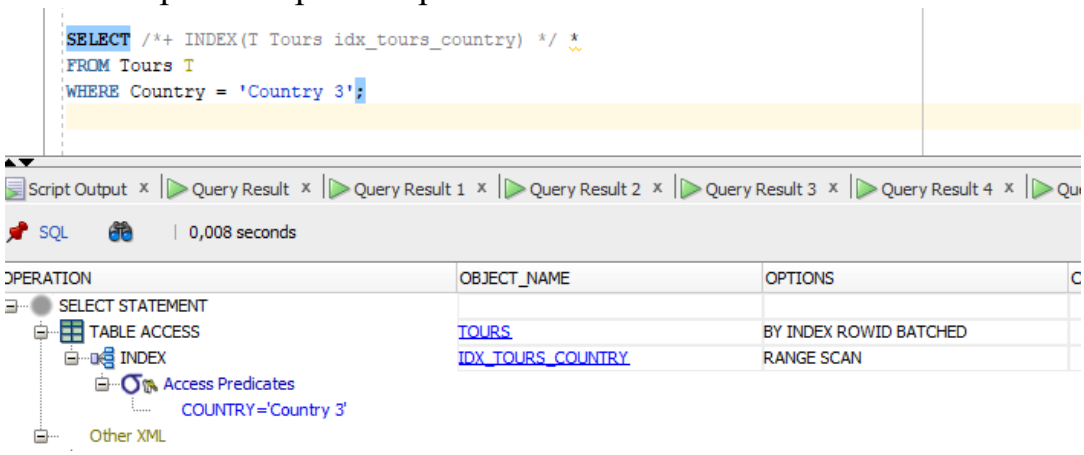

Рисунок 5.2 – План выполнения запроса после создания индекса

Таким образом, тестирование подтвердило эффективность индексирования при работе с большими объемами данных

5.2 Вывод по разделу

Результаты тестирования показали, что наличие индексов значительно ускоряет выполнение поисковых запросов, особенно при фильтрации по часто используемым полям. В ходе эксперимента время выполнения запроса уменьшилось с 34 до 8 миллисекунд после добавления индекса на поле Country. Это демонстрирует высокую эффективность индексирования в оптимизации операций выборки.

Кроме того, использование индексов способствует снижению нагрузки на сервер базы данных и сокращает объём обрабатываемых данных за счёт исключения необходимости полного сканирования таблицы. Таким образом, внедрение индексной структуры является одним из ключевых факторов в обеспечении высокой производительности информационной системы.

6 Описание технологии и её применение в базе данных

6.1 Технология OLAP-хранилища данных

Для расширения функциональности базы данных туристического агентства была выбрана технология OLAP (Online Analytical Processing) – многомерный анализ данных, позволяющий эффективно обрабатывать большие объемы информации и формировать отчётность в удобном виде. В отличие от транзакционных операций OLTP, OLAP ориентирован на анализ и принятие решений, особенно полезен для бизнес-аналитиков, менеджеров и администраторов.

OLAP-хранилище позволяет выполнять агрегированные запросы по продажам, бронированиям, популярности туров и направлениям. Оно строится на основе фактов и измерений:

Факт (основной объект анализа):

– Платеж (таблица Payments)

Измерения:

– Дата (дата платежа)
– Тур (таблица Tours)
– Клиент (таблица AppUsers)
– Категория тура (таблица TourCategories)
– Страна направления (Tours.Country)
– Статус бронирования (таблица Bookings)

```
SELECT
  T.Country,
  TO_CHAR(P.PaymentDate, 'YYYY-MM') AS Month,
  SUM(P.Amount) AS TotalRevenue
FROM Payments P
JOIN Bookings B ON P.BookingId = B.BookingId
JOIN Tours T ON B.TourId = T.TourId
GROUP BY T.Country, TO_CHAR(P.PaymentDate, 'YYYY-MM')
ORDER BY T.Country, Month;
```

Листинг 6.1 – Пример OLAP-запроса

Возможности аналитики:

– Подсчет суммарной выручки по месяцам, странам, категориям туров;
– Определение самых продаваемых туров;
– Выявление сезонности спроса;
– Оценка средней стоимости туров по регионам;
– Анализ отзывов клиентов по направлениям и категориям.

Инструменты:

Oracle поддерживает реализацию OLAP-решений как средствами SQL (GROUP BY, ROLLUP, CUBE), так и с использованием OLAP Cubes и представлений Materialized Views. В рамках проекта реализована облегчённая модель OLAP-аналитики средствами SQL-запросов с возможностью последующего подключения внешних BI-инструментов (например, Power BI или Oracle Analytics).

6.2 Вывод по разделу

Применение технологии OLAP в разработанной базе данных позволило реализовать основу для гибкой и масштабируемой системы аналитики. Аналитические запросы позволяют формировать управленческие отчеты, выявлять закономерность в поведении клиентов, отслеживать эффективность туров и принимать обоснованные бизнес-решения. Это делает систему не просто хранилищем данных, а полноценным инструментом поддержки управленческих процессов.

Заключение

В ходе выполнения курсового проекта была спроектирована и реализована база данных для автоматизации работы туристического агентства. Разработка охватывала все ключевые аспекты функционирования системы: хранение информации о пользователях, турах, маршрутах, отелях, бронированиях, оплатах и отзывах.

Проект выполнен в среде Oracle Database, с использованием языка SQL и PL/SQL, что обеспечило высокую гибкость и надёжность реализации. Были разработаны хранимые процедуры и функции, автоматизирующие основные бизнес-процессы. Реализована ролевая модель безопасности, разграничивающая права доступа между клиентами, менеджерами и администраторами.

Дополнительно в систему была интегрирована технология OLAP, что позволило реализовать аналитические запросы и формировать отчётность по продажам, популярности туров и активности клиентов. Также предусмотрены механизмы импорта и экспорта данных, обеспечивающие гибкое взаимодействие с внешними источниками.

Созданная база данных полностью соответствует поставленным требованиям, является масштабируемой, логически обоснованной и готовой к дальнейшему расширению. Она может быть использована как основа для построения полноценной информационной системы туристического агентства.

Список использованных источников

1. Аналог «Слетаем.by» [Электронный ресурс]. – Режим доступа: https://sletaem.by/ – Дата доступа: 09.03.2025.
2. Аналог «ИнтерСити» [Электронный ресурс]. – Режим доступа: https://intercity.by/ – Дата доступа: 09.03.2025.
3. Аналог «География путешествий» [Электронный ресурс]. – Режим доступа: https://geograf.by/ – Дата доступа: 09.03.2025.

Приложение А

Рисунок А.1 – Диаграмма вариантов использования

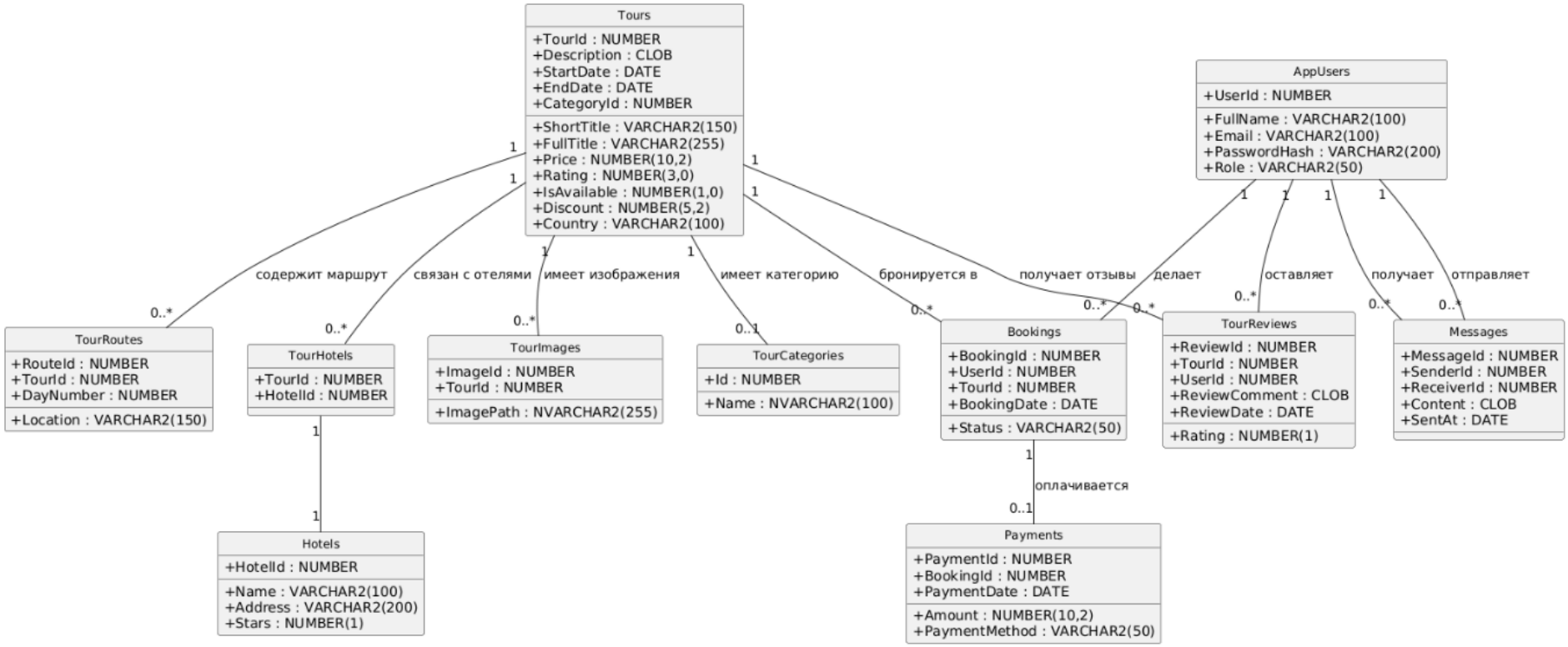

Рисунок А.2 – Диаграмма классов

Приложение Б

Таблица Б.1 – Сведения о всех таблицах

Имя таблицы	**Назначение таблицы**
AppUsers	Хранит информацию о зарегистрированных пользователях, включая ФИО, email, роль и пароль.
Tours	Содержит данные о турах: названия, описание, цены, даты, страна, категория и пр.
Bookings	Хранит информацию о бронированиях туров пользователями.
Payments	Хранит информацию об оплате бронирований.
TourReviews	Содержит отзывы и оценки, оставленные пользователями о турах.
TourRoutes	Хранит информацию о маршрутах туров: локации и дни.
Hotels	Информация об отелях, включая название, адрес и рейтинг (звёзды).
TourHotels	Связь между турами и отелями, участвующими в маршруте.
Messages	Переписка между пользователями.
TourCategories	Категории туров (пляжный, экскурсионный и т. д.).
TourImages	Хранит изображения туров.
BookingStatusLog	История изменения статусов бронирований.
Export_XML_Log	Хранит XML-результаты экспорта данных.

Таблица Б.2 – Описание таблицы AppUsers

Поле таблицы	**Тип данных**	**Назначение поля**
UserId	NUMBER	Уникальный идентификатор пользователя.
FullName	VARCHAR2(100)	ФИО пользователя.
Email	VARCHAR2(100)	Электронная почта.
PasswordHash	VARCHAR2(200)	Хеш пароля.
Role	VARCHAR2(50)	Роль пользователя (Client, Manager, Admin).

Таблица Б.3 – Описание таблицы Tours

Поле таблицы	**Тип данных**	**Назначение поля**
TourId	NUMBER	Уникальный идентификатор тура.
ShortTitle	VARCHAR2(150)	Краткое название тура.
FullTitle	VARCHAR2(255)	Полное название тура.
Description	CLOB	Подробное описание тура.
Price	NUMBER(10,2)	Стоимость тура.
StartDate	DATE	Дата начала тура.
EndDate	DATE	Дата окончания тура.
Rating	NUMBER(3,0)	Средний рейтинг тура.
IsAvailable	NUMBER(1,0)	Флаг доступности тура.
Discount	NUMBER(5,2)	Размер скидки.

Поле таблицы	Тип данных	Назначение поля
Country	VARCHAR2(100)	Страна назначения.
CategoryId	NUMBER	Идентификатор категории тура.

Таблица Б.4 – Описание таблицы Bookings

Поле таблицы	Тип данных	Назначение поля
BookingId	NUMBER	Уникальный идентификатор бронирования.
UserId	NUMBER	Идентификатор пользователя, сделавшего бронирование.
TourId	NUMBER	Идентификатор выбранного тура.
BookingDate	DATE	Дата бронирования.
Status	VARCHAR2(50)	Текущий статус бронирования.

Таблица Б.5 – Описание таблицы Payments

Поле таблицы	Тип данных	Назначение поля
PaymentId	NUMBER	Уникальный идентификатор платежа.
BookingId	NUMBER	Идентификатор бронирования.
Amount	NUMBER(10,2)	Сумма оплаты.
PaymentDate	DATE	Дата проведения оплаты.
PaymentMethod	VARCHAR2(50)	Метод оплаты (картой, онлайн и т.д.).

Таблица Б.6 – Описание таблицы TourReviews

Поле таблицы	Тип данных	Назначение поля
ReviewId	NUMBER	Уникальный идентификатор отзыва.
TourId	NUMBER	Идентификатор тура, к которому относится отзыв.
UserId	NUMBER	Идентификатор пользователя, оставившего отзыв.
Rating	NUMBER(1)	Оценка (1–5).
ReviewComment	CLOB	Комментарий пользователя.
ReviewDate	DATE	Дата публикации отзыва.

Таблица Б.7 – Описание таблицы TourRoutes

Поле таблицы	Тип данных	Назначение поля
RouteId	NUMBER	Уникальный идентификатор маршрута.
TourId	NUMBER	Идентификатор тура.
Location	VARCHAR2(150)	Локация в рамках маршрута.
DayNumber	NUMBER	Номер дня, соответствующий маршруту.

Таблица Б.8 – Описание таблицы Hotels

Поле таблицы	Тип данных	Назначение поля
HotelId	NUMBER	Уникальный идентификатор отеля.

Поле таблицы	Тип данных	Назначение поля
Name	VARCHAR2(100)	Название отеля.
Address	VARCHAR2(200)	Адрес отеля.
Stars	NUMBER(1)	Количество звёзд (1–5).

Таблица Б.9 – Описание таблицы TourHotels

Поле таблицы	Тип данных	Назначение поля
TourId	NUMBER	Идентификатор тура.
HotelId	NUMBER	Идентификатор отеля.

Таблица Б.10 – Описание таблицы Messages

Поле таблицы	Тип данных	Назначение поля
MessageId	NUMBER	Уникальный идентификатор сообщения.
SenderId	NUMBER	ID отправителя.
ReceiverId	NUMBER	ID получателя.
Content	CLOB	Содержание сообщения.
SentAt	DATE	Дата отправки.

Таблица Б.11 – Описание таблицы TourCategories

Поле таблицы	Тип данных	Назначение поля
ID	NUMBER(38,0)	Уникальный идентификатор.
Name	NVARCHAR2(100)	Название категории тура.

Таблица Б.12 – Описание таблицы TourImages

Поле таблицы	Тип данных	Назначение поля
ImageId	NUMBER	Уникальный ID изображения.
ImagePath	NVARCHAR2(255)	Путь к изображению.
TourId	NUMBER	Идентификатор тура.

Таблица Б.13 – Описание таблицы BookingStatusLog

Поле таблицы	Тип данных	Назначение поля
LogId	NUMBER	Уникальный ID записи лога.
BookingId	NUMBER	Идентификатор бронирования.
OldStatus	VARCHAR2(50)	Старый статус.
NewStatus	VARCHAR2(50)	Новый статус.
ChangedAt	DATE	Дата изменения.

Таблица Б.14 – Описание таблицы Export_XML_Log

Поле таблицы	Тип данных	Назначение поля
ID	NUMBER	Уникальный ID записи.
ExportDate	DATE	Дата экспорта.
XmlData	CLOB	Содержимое XML-данных.

Приложение В

Листинг создания таблиц базы данных предоставлен ниже.

```
CREATE TABLE AppUsers (
    UserId NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    FullName VARCHAR2(100),
    Email VARCHAR2(100) UNIQUE,
    PasswordHash VARCHAR2(200),
    Role VARCHAR2(50)
);
CREATE TABLE Bookings (
    BookingId NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    UserId NUMBER REFERENCES AppUsers(UserId),
    TourId NUMBER REFERENCES Tours(TourId),
    BookingDate DATE DEFAULT SYSDATE,
    Status VARCHAR2(50) DEFAULT 'Pending'
);
CREATE TABLE TourRoutes (
    RouteId NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    TourId NUMBER REFERENCES Tours(TourId),
    Location VARCHAR2(150),
    DayNumber NUMBER
);
CREATE TABLE Hotels (
    HotelId NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    Name VARCHAR2(100),
    Address VARCHAR2(200),
    Stars NUMBER(1) CHECK (Stars BETWEEN 1 AND 5)
);
CREATE TABLE TourReviews (
    ReviewId NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    TourId NUMBER REFERENCES Tours(TourId),
    UserId NUMBER REFERENCES AppUsers(UserId),
    Rating NUMBER(1) CHECK (Rating BETWEEN 1 AND 5),
    ReviewComment CLOB,
    ReviewDate DATE DEFAULT SYSDATE
);
CREATE TABLE Payments (
    PaymentId NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    BookingId NUMBER REFERENCES Bookings(BookingId),
    Amount NUMBER(10, 2) NOT NULL,
    PaymentDate DATE DEFAULT SYSDATE,
    PaymentMethod VARCHAR2(50)
);
CREATE TABLE TourHotels (
    TourId NUMBER REFERENCES Tours(TourId),
    HotelId NUMBER REFERENCES Hotels(HotelId),
    PRIMARY KEY (TourId, HotelId)
);
CREATE TABLE Messages (
    MessageId NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    SenderId NUMBER REFERENCES AppUsers(UserId),
    ReceiverId NUMBER REFERENCES AppUsers(UserId),
    Content CLOB,
```

```
        SentAt DATE DEFAULT SYSDATE
    );
    CREATE TABLE Tours (
        TourId NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
        ShortTitle VARCHAR2(100) NOT NULL,
        FullTitle VARCHAR2(255),
        Description CLOB,
        Price NUMBER(10, 2) NOT NULL,
        StartDate DATE,
        EndDate DATE,
        Rating NUMBER(3,2),
        IsAvailable NUMBER(1) DEFAULT 1,
        Discount NUMBER(5,2),
        Country VARCHAR2(100),
        CategoryId NUMBER,
        CONSTRAINT   FK_Tour_Category   FOREIGN   KEY   (CategoryId)
REFERENCES TourCategories(Id)
    );
    CREATE TABLE BookingStatusLog (
        LogId NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
        BookingId NUMBER,
        OldStatus VARCHAR2(50),
        NewStatus VARCHAR2(50),
        ChangedAt DATE DEFAULT SYSDATE
    );
    CREATE TABLE export_xml_log (
      id         NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
      exportdate DATE,
      xmldata    CLOB
    );
    CREATE TABLE TourImages (
        Id INT PRIMARY KEY IDENTITY,
        ImagePath NVARCHAR(255) NOT NULL,
        TourId INT NOT NULL,
        FOREIGN KEY (TourId) REFERENCES Tours(Id)
    );
    CREATE TABLE TourCategories (
        Id INT PRIMARY KEY IDENTITY,
        Name NVARCHAR(100) NOT NULL
    );
```

Листинг В.1 – Запросы создания таблиц базы данных

Приложение Г

Листинг SQL-запросов для создания процедур базы данных предоставлен ниже.

```
CREATE OR REPLACE PROCEDURE MakeBooking (
    p_UserId IN NUMBER,
    p_TourId IN NUMBER
) AS
BEGIN
    INSERT INTO Bookings (UserId, TourId, BookingDate, Status)
    VALUES (p_UserId, p_TourId, SYSDATE, 'Pending');
END;
CREATE OR REPLACE PROCEDURE ConfirmBookingPayment (
    p_BookingId IN NUMBER,
    p_Amount IN NUMBER,
    p_Method IN VARCHAR2
) AS
BEGIN
    UPDATE Bookings
    SET Status = 'Confirmed'
    WHERE BookingId = p_BookingId;

    INSERT INTO Payments (BookingId, Amount, PaymentMethod,
PaymentDate)
    VALUES (p_BookingId, p_Amount, p_Method, SYSDATE);
END;
CREATE OR REPLACE PROCEDURE CancelBooking (
    p_BookingId IN NUMBER
) AS
BEGIN
    UPDATE Bookings
    SET Status = 'Cancelled'
    WHERE BookingId = p_BookingId;
END;
CREATE OR REPLACE PROCEDURE LeaveReview (
    p_UserId IN NUMBER,
    p_TourId IN NUMBER,
    p_Rating IN NUMBER,
    p_Comment IN CLOB
) AS
BEGIN
    INSERT INTO TourReviews (TourId, UserId, Rating,
ReviewComment, ReviewDate)
    VALUES (p_TourId, p_UserId, p_Rating, p_Comment, SYSDATE);
END;
CREATE OR REPLACE PROCEDURE AddRouteToTour (
    p_TourId IN NUMBER,
    p_Location IN VARCHAR2,
    p_DayNumber IN NUMBER
) AS
BEGIN
    INSERT INTO TourRoutes (TourId, Location, DayNumber)
    VALUES (p_TourId, p_Location, p_DayNumber);
```

```
END;
CREATE OR REPLACE PROCEDURE AddHotel (
    p_Name IN VARCHAR2,
    p_Address IN VARCHAR2,
    p_Stars IN NUMBER
) AS
BEGIN
    INSERT INTO Hotels (Name, Address, Stars)
    VALUES (p_Name, p_Address, p_Stars);
END;
CREATE OR REPLACE PROCEDURE LinkHotelToTour (
    p_TourId IN NUMBER,
    p_HotelId IN NUMBER
) AS
BEGIN
    INSERT INTO TourHotels (TourId, HotelId)
    VALUES (p_TourId, p_HotelId);
END;
CREATE OR REPLACE PROCEDURE SendMessage (
    p_SenderId IN NUMBER,
    p_ReceiverId IN NUMBER,
    p_Content IN CLOB
) AS
BEGIN
    INSERT INTO Messages (SenderId, ReceiverId, Content, SentAt)
    VALUES (p_SenderId, p_ReceiverId, p_Content, SYSDATE);
END;
CREATE OR REPLACE PROCEDURE AddTour (
    p_ShortTitle   IN VARCHAR2,
    p_FullTitle    IN VARCHAR2,
    p_Description  IN CLOB,
    p_Price        IN NUMBER,
    p_StartDate    IN DATE,
    p_EndDate      IN DATE,
    p_IsAvailable  IN NUMBER,
    p_Discount     IN NUMBER,
    p_Country      IN VARCHAR2,
    p_CategoryId   IN NUMBER
)
IS
BEGIN
    INSERT INTO Tours (
        ShortTitle, FullTitle, Description,
        Price, StartDate, EndDate,
        Rating, IsAvailable, Discount,
        Country, CategoryId
    ) VALUES (
        p_ShortTitle, p_FullTitle, p_Description, p_Price,
        p_StartDate, p_EndDate, NULL,       p_IsAvailable,
        p_Discount, p_Country,p_CategoryId
    );
END;
```

Листинг Г.1 – Запросы создания функций базы данных

Приложение Д

Листинг SQL-запросов для создания функций базы данных предоставлен ниже.

```
CREATE OR REPLACE FUNCTION GetAverageTourRating (
    p_TourId IN NUMBER
) RETURN NUMBER IS
    v_AvgRating NUMBER;
BEGIN
    SELECT AVG(Rating)
    INTO v_AvgRating
    FROM TourReviews
    WHERE TourId = p_TourId;

    RETURN v_AvgRating;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN NULL;
END;
CREATE OR REPLACE FUNCTION GetUserBookingCount (
    p_UserId IN NUMBER
) RETURN NUMBER IS
    v_Count NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO v_Count
    FROM Bookings
    WHERE UserId = p_UserId;

    RETURN v_Count;
END;
CREATE OR REPLACE FUNCTION GetTourRevenue (
    p_TourId IN NUMBER
) RETURN NUMBER IS
    v_Revenue NUMBER;
BEGIN
    SELECT SUM(P.Amount)
    INTO v_Revenue
    FROM Payments P
    JOIN Bookings B ON P.BookingId = B.BookingId
    WHERE B.TourId = p_TourId;

    RETURN NVL(v_Revenue, 0);
END;
```

Листинг Д.1 – Запросы создания функций базы данных