Содержание

[Введение 4](#_Toc154050210)

[1 Постановка задачи 5](#_Toc154050211)

[1.1 Обзор аналогичных решений 5](#_Toc154050212)

[1.2 Требования к проекту 6](#_Toc154050213)

[2 Проектирование модели базы данных 7](#_Toc154050214)

[3 Разработка необходимых объектов 8](#_Toc154050215)

[3.1 Таблицы 8](#_Toc154050216)

[3.2 Пакеты 9](#_Toc154050217)

[3.3 Процедуры 10](#_Toc154050218)

[3.4 Функции 11](#_Toc154050219)

[3.5 Планировщик 12](#_Toc154050220)

[3.6 Синонимы 12](#_Toc154050221)

[3.7 Роли и пользователи 13](#_Toc154050222)

[3.8 Индексы 13](#_Toc154050223)

[4 Описание процедур импорта и экспорта данных 14](#_Toc154050224)

[4.1 Процедура экспорта данных из XML-файла 14](#_Toc154050225)

[4.2 Процедура импорта данных в формате XML 14](#_Toc154050226)

[5 Описание технологии и ее применения в базе данных 15](#_Toc154050227)

[6 Тестирование производительности 18](#_Toc154050228)

[7 Руководство пользователя 19](#_Toc154050229)

[7.1 Сторона администратора 19](#_Toc154050230)

[7.2 Сторона пользователя 19](#_Toc154050231)

[Заключение 20](#_Toc154050232)

[Список используемых источников 21](#_Toc154050233)

[ПРИЛОЖЕНИЕ A 22](#_Toc154050234)

[ПРИЛОЖЕНИЕ Б 23](#_Toc154050235)

[ПРИЛОЖЕНИЕ В 25](#_Toc154050236)

[ПРИЛОЖЕНИЕ Г 26](#_Toc154050237)

# Введение

В современном мире многие люди на регулярной основе посещают культурно-развлекательные мероприятия, такие как пьесы, кукольные театры и кинопоказы. Однако не каждый человек имеет возможность заблаговременно зарезервировать фиксированное количество билетов на то или иное мероприятие, тем самым обезопасить себя от ситуации, когда по приходу в заведение кинотеатра все билеты на желаемый фильм оказались распроданы. Более того, многие люди сталкиваются с проблемой сложности выбора той или иной кинокартины ввиду отсутствия цельного полного описания всех фильмах в текущем расписании кинотеатра.

База данных кинотеатра играет важную роль в организации и управлении информацией о фильмах, сеансах, зрителях, билетах и многих других аспектах, связанных с кинопоказом. Она является неотъемлемой частью современного кинобизнеса, обеспечивая эффективное планирование и управление кинопрограммой, а также учет прибыли и обслуживание клиентов.

В данном контексте, база данных предоставляет возможность хранить и оперативно извлекать информацию о фильмах, датах и времени сеансов, свободных местах в залах, а также о бронированиях и продажах билетов. Эта информация позволяет кинотеатрам оптимизировать свою деятельность, предоставлять лучший сервис клиентам и увеличивать прибыль.

Кроме того, базы данных кинотеатров могут быть интегрированы с онлайн-бронированием, мобильными приложениями и веб-сайтами, обеспечивая удобство и доступность информации для зрителей.

Целью данного курсового проекта является создание базы данных кинотеатра с применением мультимедийных типов данных в СУБД Oracle.

# Постановка задачи

* 1. **Обзор аналогичных решений**

Для более глубокого понимания и эффективной постановки требований к проекту разработки системы управления кинотеатром, представляется необходимым провести обзор аналогичных решений на рынке. Анализ существующих систем позволит выявить сильные и слабые стороны, успешные практики и возможные улучшения. При изучении аналогичных проектов следует обратить внимание на функциональность, пользовательский опыт, масштабируемость и безопасность. Такой обзор позволит извлечь ценные уроки из опыта других разработчиков и предоставит возможность определить ключевые характеристики, которые будут служить основой для успешной реализации и внедрения разрабатываемой системы в кинотеатральной среде.

Одним из таких решений является интернет-ресурс «КиноПоиск». Основные функции ресурса «КиноПоиск»:

* поиск фильмов и сериалов по различным критериям;
* подробная информация о фильмах и сериалах, включая рейтинги, оценки и описания;
* сведения о актерах и режиссерах, с биографиями и списками их работ;
* чтение рецензий и просмотр рейтингов фильмов от критиков и зрителей;
* возможность создания списков просмотренных и планируемых к просмотру фильмов.

Так же были проанализированы основные сущности базы данных данного ресурса: Movies, Actors, Users, Reviews, Ratings, Watchlists.

Из выявленных недостатков данного ресурса необходимо выделить невозможность забронировать фиксированное количество билетов на реальный кинопоказ в кинотеатре, удалить заказ из своего личного кабинета, удалить личный профиль. Также немаловажным недостатком «КиноПоиска» является достаточно неудобный пользовательский интерфейс, недостаточный учет индивидуальных интересов и пожеланий посетителя данного ресурса

Еще одним альтернативным решением задач, поставленных в данном курсовом проекте, является интернет-ресурс «Netflix». Функционал сайта включает потоковое воспроизведение широкой библиотеки контента, персонализированные рекомендации и алгоритмы подбора, создание нескольких профилей в рамках аккаунта, загрузка контента для просмотра оффлайн, поддержка множества устройств, создание списков просмотра и оценка контента, оригинальный контент, доступный только на Netflix, средства родительского контроля.

Одним из главных недостатков является платная подписка для возможно пользоваться функционалом и отсутствие возможности обсудить конкретный фильм с другими пользователями.

Также альтернативой решения поставленных задач является посещение реального кинотеатра города. Преимуществом данного выбора является возможность непосредственно купить билеты на определенный сеанс показа кинокартины.

Однако среди недостатков данной альтернативы можно выделить риск полной распродажи билетов к моменту прибытия в кинотеатр, отсутствие возможности найти фильм по вкусу, используя полнотекстовый поиск по описанию, сложность в ориентировании при выборе среди множества кинофильмов в виду отсутствия цельного списка кинокартин с непосредственным описанием каждой.

Обзор вышеперечисленных известных аналогов решений поставленных ранее задач позволяет проанализировать все преимущества и недостатки альтернативных возможностей реализации бронирования билетов на кинофильмы в Internet и позволяет сформулировать список требований, предъявляемых к программному средству, разрабатываемому в данном курсовом проекте.

* 1. **Требования к проекту**

Целью данного курсового проекта является проектирование базы данных кинотеатра. В качестве модели данных следует использовать реляционную модель. Проектирование необходимо произвести таким образом, чтобы конечные данные соответствовали общим требованиям к информации в базе данных.

Функционально должны быть выполнены следующие задачи:

* определение ролей (администратор, пользователь);
* управление фильмами (добавление, удаление, изменение);
* управление фильмами (добавление, удаление, изменение);
* просмотр трейлера фильма;
* просмотр расписания;
* просмотр афиши фильма;
* обеспечение бронирования билета.

Доступ к данным должен осуществляться исключительно через соответствующие процедуры, обеспечивая тем самым контролируемый и безопасный доступ. Реализация базы данных предусматривает проведение импорта данных из XML файлов и экспорта данных в формат XML. Важным этапом является тестирование производительности базы данных на таблице, содержащей не менее 100 000 строк. Технологический компонент задачи включает в себя применение технологий Oracle, а также демонстрацию применения выбранной технологии применения мультимедийных типов данных в контексте разработанной базы данных.

# Проектирование модели базы данных

Для реализации поставленной задачи разработана подключаемая база данных PLKINO в СУБД Oracle. Oracle Database – это объектно-реляционная система управления базами данных (СУБД) от компании Oracle. Она используется для создания структуры новой базы, ее наполнения, редактирования содержимого и отображения информации.

База данных состоит из пяти таблиц, взаимосвязанных внешними ключами, обеспечивающих эффективное хранение и структурирование данных. Диаграмма структуры полученной базы данных представлена на рисунке 2.1.

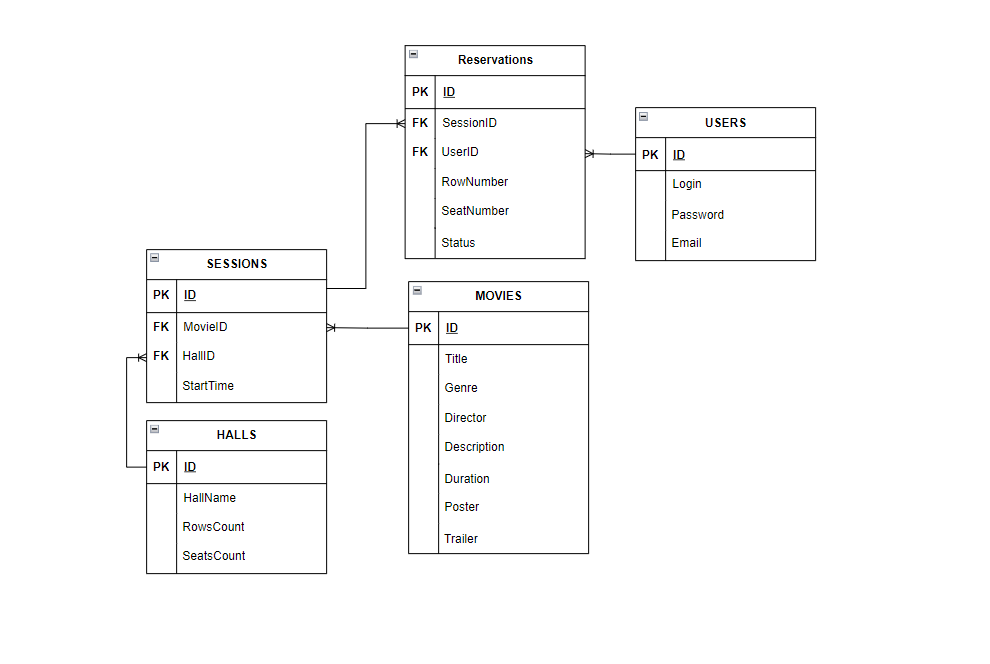


Рисунок 2.1 – Архитектура Oracle Multimedia

Таблица USERS содержит информацию о пользователях, включая их идентификатор, логин, пароль и электронную почту.

MOVIES представляет собой хранилище данных о фильмах, что облегчает поиск и доступ к соответствующей информации.

Информация о залах кинотеатра хранится в таблице HALLS, включая идентификатор, название зала, количество рядов и количество мест в зале. SESSIONS сохраняет данные о доступных сеансах, включая идентификатор, связанные идентификаторы фильма и зала, а также время начала сеанса.

RESERVATIONS, в свою очередь, служит для хранения информации о забронированных билетах пользователей, включая идентификатор брони, связанные идентификаторы сеанса и пользователя, номер ряда и места в зале.

Диаграмма вариантов использования для пользователя и администратора в приложении А.

Таблицы созданы с установлением внешних связей, что обеспечивает целостность данных и эффективность их использования в рамках задачи системы управления кинотеатром. Подробное описание структуры таблиц представлено в главе 3.

# Разработка необходимых объектов

## Таблицы

Таблицы являются неотъемлемой частью любой реляционной базы данных. Краткая характеристика каждой из таблиц была предоставлена в разделе 2, а код создания можно увидеть в приложении Б.

Таблица MOVIES состоит из восьми столбцов: ID, TITLE, GENRE, DIRECTOR, DESCRIPTION, DURATION. Структура представлена в таблице 3.1.

Таблица 3.1 – Структура таблицы MOVIES

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| ID | NUMBER | PRIMARY KEY |
| TITLE | VARCHAR2(255) | NOT NULL |
| GENRE | VARCHAR2(100) | NOT NULL |
| DIRECTOR | VARCHAR2(100) | NOT NULL |
| DESCRIPTION | CLOB | NOT NULL |
| DURATION | NUMBER | CHECK (Duration > 0)  NOT NULL |
| POSTER | BLOB | NOT NULL |
| TRAILER | BLOB | NOT NULL |

Таблица HALLS состоит из четырех столбцов: ID, HallName, RowsCount, SeatsCount. Структура представлена в таблице 3.2.

Таблица 3.2 – Структура таблицы Halls

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| ID | NUMBER | PRIMARY KEY |
| HallName | VARCHAR2(50) | NOT NULL |
| RowsCount | NUMBER | CHECK (RowsCount > 0)  NOT NULL |
| SeatsCount | NUMBER | CHECK (SeatsCount > 0)  NOT NULL |

Таблица SESSIONS состоит из четырех столбцов: ID, MovieID, HallID, StartTime. Структура представлена в таблице 3.3.

Таблица 3.3 – Структура таблицы Sessions

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| ID | NUMBER | PRIMARY KEY |
| MovieID | NUMBER | FOREIGN KEY |
| HallID | NUMBER | FOREIGN KEY |
| StartTime | DATE | NOT NULL |

Таблица RESERVATION состоит из пяти столбцов: ID, SessionID, UserID, RowNumber, SeatNumber. Структура представлена в таблице 3.4.

Таблица 3.4 – Структура таблицы Reservations

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| ID | NUMBER | PRIMARY KEY |
| SessionID | NUMBER | FOREIGN KEY |
| UserID | NUMBER | FOREIGN KEY |
| RowNumber | NUMBER | CHECK(RowNumber>=0)NOT NULL |
| SeatNumber | NUMBER | CHECK (SeatNumber>=0)NOT NULL |

Таблица USERS состоит из четырех столбцов: ID, MovieID, HallID, StartTime. Структура представлена в таблице 3.5.

Таблица 3.5 – Структура таблицы Users

|  |  |  |
| --- | --- | --- |
| Столбец | Тип данных | Ограничение целостности |
| ID | NUMBER | PRIMARY KEY |
| Login | VARCHAR2(50) | NOT NULL |
| Password | NUMBER | NOT NULL |
| Email | VARCHAR2(100) | NOT NULL |

Представленная структура базы данных обеспечивает эффективное и надежное хранение информации, необходимой для функционирования системы управления кинотеатром. Предусмотренные ограничения целостности и связи между таблицами гарантируют целостность данных и согласованность информации в системе.

## Пакеты

В базе данных были успешно созданы два пакета, а именно admin\_package и user\_package, предназначенные для эффективного управления и взаимодействия с базой данных. Пакет admin\_package включает в себя процедуры и функции, предназначенные для административных задач, таких как управление пользователями, добавление и обновление информации, выполнение привилегированных операций. Спецификация пакета для администратора представлена в приложении Б.

С другой стороны, user\_package содержит процедуры и функции, предназначенные для обеспечения функциональности для конечных пользователей, таких как запросы на чтение данных и другие операции, доступные обычным пользователям системы. Эти пакеты облегчают разделение обязанностей и управление доступом, что способствует более эффективной и безопасной работе с базой данных. Спецификация пакета для пользователя представлена в приложении В.

## Процедуры

В пакете admin\_package было успешно разработано 11 процедур для административных операций в контексте управления базой данных .

Процедура RegistrationUser осуществляет регистрацию новых пользователей, проверяя уникальность логина в таблице пользователей. После успешной регистрации процедура добавляет запись о пользователе в таблицу, предоставляя ему необходимые привилегии.

Для добавления нового фильма в базу данных предназначена процедура AddMovie. Перед внесением изменений процедура проверяет уникальность названия фильма. В случае обнаружения дубликата выводится сообщение об ошибке, иначе процедура добавляет новую запись о фильме.

Процедура AddHall отвечает за добавление нового зала в базу данных. Перед внесением изменений процедура проверяет уникальность названия зала. При успешной проверке она добавляет новую запись о зале, включая информацию о количестве рядов и мест.

Добавление нового сеанса в базу данных обеспечивает процедура AddSession. Перед добавлением процедура проверяет существование указанного фильма и зала по их идентификаторам, а также валидность даты и времени начала сеанса с использованием функции IsDateValid. При успешных проверках процедура добавляет новую запись о сеансе.

Процедура DeleteMovie удаляет фильм из базы данных по указанному идентификатору. Перед удалением процедура проверяет существование фильма в таблице фильмов.

Удаление зала из базы данных осуществляет процедура DeleteHall, проверяя его существование по идентификатору перед удалением.

Процедура DeleteSession удаляет сеанс из базы данных по указанному идентификатору. Перед удалением процедура проверяет существование сеанса в таблице сеансов.

Удаление брони из базы данных выполняет процедура DeleteReservation, предварительно проверяя ее существование по идентификатору.

Процедура DeleteUser удаляет пользователя из базы данных по указанному идентификатору. Перед удалением процедура проверяет существование пользователя в таблице пользователей.

Обновление информации о фильме в базе данных осуществляет процедура UpdateMovie, предварительно проверив его существование по идентификатору.

Процедура UpdateHall обновляет информацию о зале в базе данных. Перед внесением изменений процедура проверяет существование зала по идентификатору.

Обновление информации о сеансе в базе данных выполняет процедура UpdateSession. Перед внесением изменений процедура проверяет существование сеанса, а также указанных фильма и зала по их идентификаторам.

Пакет user\_package содержит 9 процедур, предназначенных для взаимодействия с базой данных на уровне пользователя.

Процедура GetFreeSeatsForSession предоставляет информацию о доступных местах для выбранного сеанса. Вначале проверяется существование сеанса, после чего определяется количество рядов и мест в зале для выявления свободных мест.

MakeReservation позволяет пользователю зарезервировать место на определенном сеансе, указав ряд и место. Процедура проверяет наличие сеанса и проводит валидацию введенных данных перед созданием записи о бронировании, предварительно удостоверившись в доступности выбранного места.

Процедура BuyTicket аналогична MakeReservation. Она позволяет пользователю купить билет на конкретный сеанс, выбрав ряд и место. После выполнения необходимых проверок создается запись о бронировании со статусом «Куплено».

UpdateReservationStatus обновляет статус бронирования на «Куплено», если текущий статус – «Забронировано».

Процедура CancelReservation предоставляет возможность пользователю отменить бронирование. Сначала проверяется наличие бронирования для пользователя, и в случае обнаружения, оно отменяется.

ViewUserReservations выводит информацию о бронированиях пользователя, включая название фильма, дату сеанса, номер ряда, номер места и статус.

Процедура ViewSessionsByDate отображает сеансы, запланированные на указанную дату, предварительно проверив, что эта дата находится в будущем.

GetScheduleForNextWeek предоставляет расписание на следующую неделю, перечисляя фильмы и соответствующие даты сеансов.

GetMovieByID получает информацию о фильме по его идентификатору, выполняя запрос к базе данных и выводя соответствующую информацию, такую как название, жанр, режиссер, описание и продолжительность.

## Функции

Были разработаны функции IsMovieExists, IsHallExists, IsSessionExists, IsReservationExists и IsUserExists, которые используются в пакетах admin\_package и user\_package. Данные функции осуществляют поиск записей в соответствующих таблицах Movies, Halls, Sessions, Reservations и Users по указанным идентификаторам. Обработка исключений позволяет избежать ошибок при выполнении запросов и возвратить FALSE в случае любых непредвиденных ситуаций. Такие функции могут быть полезны, например, при предварительной проверке существования объектов перед выполнением операций добавления, обновления или удаления в базе данных.

Для проверки даты была реализована функция IsDateValid проверяющая доступность заданной даты для проведения нового сеанса в указанном кинозале. Используя входные параметры функция извлекает уникальные даты сеансов в указанном зале. Затем она проверяет, не пересекается ли заданная дата с уже запланированными сеансами.

С целью расширения функционала пользователя в user\_package присутствуют функции OpenPoster и OpenTrailer. Данные функции в пакете предназначены для получения бинарных данных изображения постера и трейлера соответственно для заданного фильма.

## Планировщик

Планировщик используется для автоматизации ряда задач, таких как ежедневные обслуживание, регулярные проверки данных, сбор статистики и т.д. Используя функционал планировщика Oracle Database (DBMS\_SCHEDULER), создается задача, которая регулярно запускает процедуру для удаления записей, у которых дата начала предшествует текущей дате. Пример создания задачи представлен в листинге 3.1.

|  |
| --- |
| BEGIN  DBMS\_SCHEDULER.create\_job (  job\_name => 'DELETE\_OLD\_SESSIONS\_JOB',  job\_type => 'PLSQL\_BLOCK',  job\_action => 'BEGIN DELETE\_OLD\_SESSIONS; END;',  start\_date => SYSTIMESTAMP,  repeat\_interval => 'FREQ=DAILY; BYHOUR=0; BYMINUTE=0; BYSECOND=0',  enabled => TRUE  );  END;  CREATE OR REPLACE PROCEDURE DELETE\_OLD\_SESSIONS IS  BEGIN  DELETE FROM Sessions WHERE StartTime < SYSDATE;  COMMIT;  END DELETE\_OLD\_SESSIONS; |

Листинг 3.1 – Создание планировщика

Этот механизм предназначен для обеспечения автоматической очистки базы данных от устаревших данных и поддержания их актуальности.

## Синонимы

Создание синонима в базе данных Oracle выполняется с целью предоставления альтернативного и удобного имени для объекта базы данных. Синоним может использоваться для скрытия сложных или длинных имен объектов, что повышает читаемость кода и облегчает его поддержку. В базе данных создан публичный синоним user\_action для ADMIN\_DATABASE. Создание синонимов представлено на листинге 3.2.

|  |
| --- |
| create public synonym user\_action for ADMIN\_DATABASE.user\_package; |

Листинг 3.2 – Синоним

В целом, использование синонимов способствует более гибкому взаимодействию с объектами базы данных, а также улучшает общую структуру базы данных.

## Роли и пользователи

Для определения функциональных возможностей пользователей были создана роль user\_role. Данная роль назначается каждому зарегистрированному пользователю и содержит права на создание сессии и выполнения user\_package.

Для определения функциональных возможностей администратора был создан пользователь базы данных admin\_database и роль admin role. При назначении администратору роли, он получает расширенные привилегии, такие как возможность создания процедур, пользователей, сессий и таблиц, а также высокие административные права, включая DBA. Роль admin\_role группирует эти привилегии, обеспечивая централизованное управление доступом к базе данных.

Такая структура ролей и их назначение пользователю спроектированы для обеспечения эффективного и безопасного управления базой данных, предоставляя администратору только необходимые привилегии для выполнения своих обязанностей без предоставления излишних прав, которые могли бы представлять потенциальные угрозы безопасности.

## Индексы

Индекс – объект базы данных, создаваемый с целью повышения производительности поиска данных.

Таблицы в базе данных могут иметь большое количество строк, которые хранятся в произвольном порядке, и их поиск по заданному критерию путём последовательного просмотра таблицы строка за строкой может занимать много времени. Индексы в базе данных используются для ускорения выполнения запросов, особенно при поиске, сортировке и объединении данных. Индексы обеспечивают эффективный доступ к данным, уменьшая время выполнения запросов за счет организации структуры данных в определенном порядке.

Для таблицы SESSIONS были созданы индексы на столбце HallID, а также на столбце MovieID, так как эти столбцы используются в условиях WHERE для процедур связанных с поиском.

# Описание процедур импорта и экспорта данных

## Процедура экспорта данных из XML-файла

Экспорт в контексте баз данных обозначает процесс сохранения данных из базы данных во внешний файл или другой хранилище данных. Это может быть использовано для резервного копирования, обмена данными между системами, создания отчетов или анализа данных в сторонних инструментах. В результате экспорта, данные из базы данных становятся доступными в структурированном формате, который можно легко передавать или анализировать вне контекста базы данных.

Одним из методов экспорта таблиц, использующим тип данных CLOB, является создание файла XML, содержащего данные из указанной таблицы. Процедура EXPORT\_TO\_FILE, представленная в приложении Г, принимает два параметра: p\_query и p\_filename.

Процедура экспорта данных предоставляет эффективный способ сохранения содержимого таблицы в формате XML. За счет параметра запроса (p\_query) возможно выборочное извлечение данных, обеспечивая гибкость в определении объема экспортируемой информации. Создаваемый файл (p\_filename) сохраняет результаты экспорта, обеспечивая легкость в последующем анализе и обработке данных. Этот метод экспорта предоставляет удобный инструмент для работы с большим объемом информации из базы данных в удобном формате

## Процедура импорта данных в формате XML

Импорт – это процесс внесения данных из внешнего источника, такого как файл или другая база данных, в текущую систему или приложение. В контексте баз данных, импорт часто используется для загрузки данных из файлов различных форматов или из других баз данных в существующую базу данных. Этот процесс позволяет обновлять, дополнять или восстанавливать данные в базе данных с учетом изменений внешних источников.

Процедура FILE\_TO\_CLOB выполняет преобразование содержимого XML-файла в формат CLOB. Данный процесс осуществляется путем последовательного чтения файла построчно, а затем объединения прочитанных строк для формирования CLOB.

При этом, процедура IMPORT\_DATA\_XML использует FILE\_TO\_CLOB для преобразования XML-файла в CLOB, а затем в XMLTYPE. Далее происходит извлечение необходимых данных из XML, которые могут быть выведены в серверный вывод или вставлены в таблицу, в зависимости от требований. Важно отметить, что данная процедура обеспечивает правильное чтение и обработку данных, а также обрабатывает возможные ошибки, которые могут возникнуть в процессе выполнения. Данные процедуры представлены в приложении Г.

# Описание технологии и ее применения в базе данных

Мультимедийные типы данных – это типы данных позволяющие хранить данные формата мультимедиа. В Oracle данная возможность реализована через Oracle Multimedia.

Oracle Multimedia представляет собой компонент Oracle Database, который специализируется на обработке мультимедийных данных, таких как изображения, звук и видео. Он обеспечивает возможности для хранения, управления и анализа мультимедийных данных прямо в базе данных Oracle. На рисунке 5.1 представлена архитектура Oracle Multimedia.

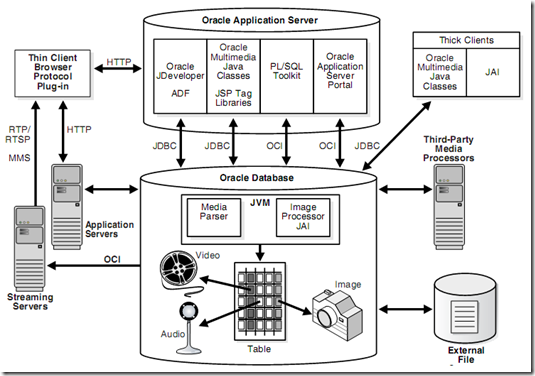


Рисунок 5.1 – Архитектура Oracle Multimedia

Основные компоненты включают ORDImage для работы с изображениями, ORDVideo для видеофайлов, а также поддержку DICOM для обработки медицинских изображений. Он также предоставляет функциональность анализа изображений и видео напрямую в базе данных.

Oracle Multimedia интегрирует возможности пространственных данных с мультимедийными, что позволяет эффективно управлять пространственной и мультимедийной информацией в рамках единой базы данных. С его помощью можно выполнять SQL-операции с мультимедийными данными, что упрощает их интеграцию в приложения, использующие Oracle Database.

В курсовом проекте технология использовалась для хранения трейлера и постера фильмов в таблице Movies.

Мультимедийные типы данных могут храниться в виде BLOB, для этого в таблице Movies были созданы два столбца Poster и Trailer типа BLOB. Для хранения данных используется директории POSTER\_DIR и ТRAILER\_DIR. Объект Directory является логической ссылкой в базе данных на каталог файловой системы сервера, где установлена БД Oracle. Код создания директорий представлен в листинге 5.1.

|  |
| --- |
| create directory POSTER\_DIR as 'C:\bd\poster';  create directory TRAILER\_DIR as 'C:\bd\trailer' |

Листинг 5.1 – Создание директорий

Вставка мультимедийных данных происходит в функции AddMovie. В параметры подается название файла, затем с помощью функции BFILENAME возвращается локатор BFILE, соответствующий имени физического файла операционной системы. Это позволяет хранить только путь к файлу в базе данных, сохраняя сам мультимедийный файл в операционной системе. Это может быть эффективным способом управления большими объемами мультимедийных данных, не загружая их напрямую в базу данных. Добавление фото и видео в базу данных представлено в листинге 5.2.

|  |
| --- |
| INSERT INTO Movies (Title, Genre, Director, Description, Duration, Poster, Trailer)  VALUES (p\_Title, p\_Genre, p\_Director, p\_Description, p\_Duration,BFILENAME('POSTER\_DIR', p\_Poster), BFILENAME('TRAILER\_DIR', p\_Trailer)); |

Листинг 5.2 – Добавление фото и видео

Для просмотра видео и изображения были разработаны функции OpenTrailer и OpenPoster, которые представлены в листинге 5.3.

|  |
| --- |
| FUNCTION OpenPoster(  p\_movie\_id IN NUMBER  ) RETURN BLOB IS  v\_poster BLOB;  BEGIN  if(IsMovieExists(p\_movie\_id)) then  SELECT Poster INTO v\_poster  FROM Movies  WHERE ID = p\_movie\_id;  RETURN v\_poster;  else  DBMS\_OUTPUT.PUT\_LINE('Ошибка OpenPoster: Фильм с заданным ID не найден');  end if;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Ошибка OpenPoster: ' || SQLERRM);  RETURN NULL;  END OpenPoster; |

Листинг 5.3 – Функция OpenPoster

Для просмотра видео нужно зайти в редактор и загрузить файл в формате mp4. Для открытия изображения нужно зайти в редактор blob и выбрать птичку image, что показано на рисунке 5.4

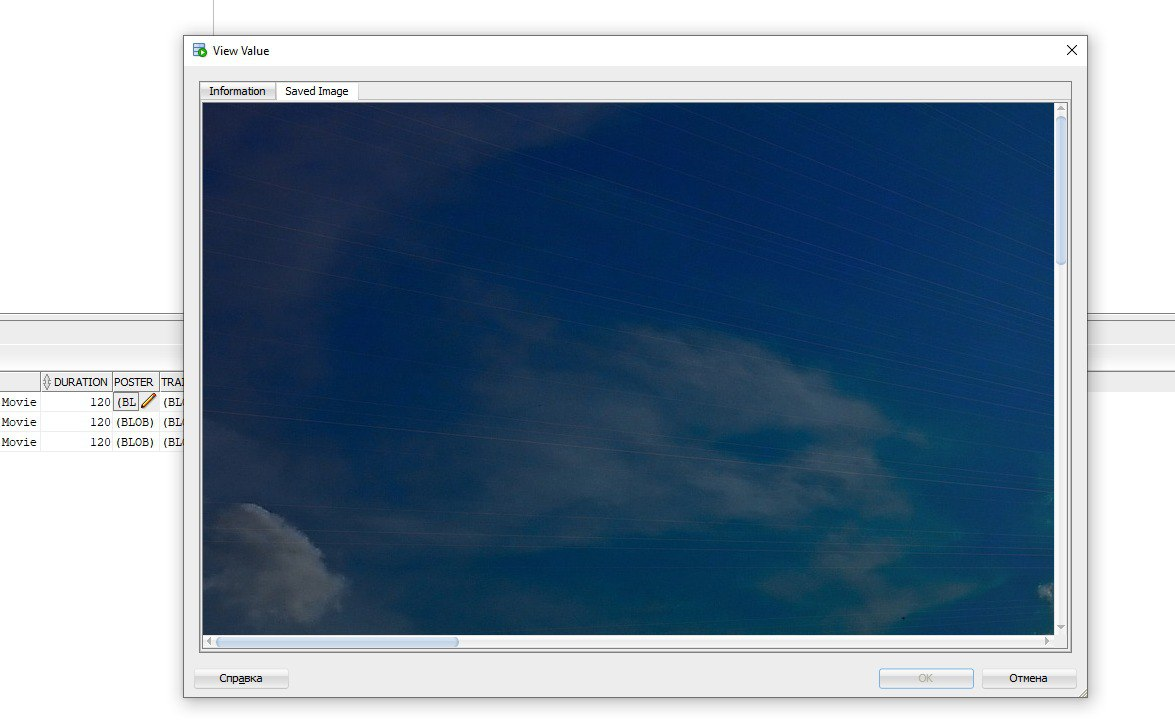


Рисунок 5.4 – Просмотр изображения в sql-developer

В рамках курсового проекта была успешно реализована функциональность хранения мультимедийных типов данных, таких как фотографии (изображения) и видео. Это включает в себя эффективное управление данными о фильмах, ассоциированными с их постерами и трейлерами. Реализованный функционал предоставляет возможность сохранения и извлечения мультимедийных контентов, обеспечивая полноценное взаимодействие с базой данных в контексте хранения и представления разнообразной медийной информации.

# Тестирование производительности

Тестирование производительности является важным этапом разработки, поскольку позволяет определить, насколько хорошо база данных может обрабатывать запросы и как быстро она может возвращать результаты.

Для тестирования производительности базы данных была выбрана таблица SESSIONS, а именно процедура пользователя GetFreeSeatsForSessions.

Результаты выполнения запроса к таблице указывают на значительные затраты времени и ресурсов, особенно при сканировании всей таблицы и применении фильтра. Существует потенциал для оптимизации с целью улучшения производительности. Результаты запроса будут представлены на рисунке 6.1.

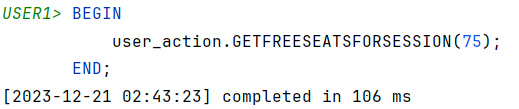


Рисунок 6.1 – Результат выполнения запроса

Для ускорения данного процесса можно создан табличных индекс для столбца HallID. Табличные индексы предназначены для индексирования столбцов с высокой селективностью. Результат после индексирования представлен на рисунке 6.2.

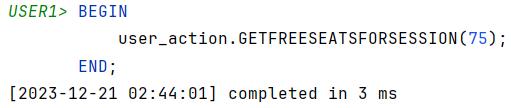


Рисунок 6.2 – Результат выполнения запроса

После создания индекса видно, что процедура стала выполняться намного быстрее. Время выполнения стало практически в 100 раз быстрее.

Создание индексов на соответствующих столбцах таблицы существенно улучшило время выполнения запроса. Индексы позволяют эффективнее организовывать доступ к данным, снижая стоимость выполнения операций. Таким образом, база данных оказывается более подготовленной к обработке больших объемов данных, что является важным аспектом в условиях активной работы системы с множеством запросов и операций.

# Руководство пользователя

Так как база данных, поддерживает 2 типа пользователей, то есть и 2 сценария использования. Первый – при входе от администратора, второй – от пользователя.

## Сторона администратора

Для начала работы в роли администратора необходимо авторизоваться под учетной записью admin\_database. После успешной авторизации станут доступны функциональности администрирования базы данных кинотеатра.

Администратор обладает возможностью добавления данных в таблицы Movies, Halls и Sessions, используя соответствующие процедуры: AddMovie, AddHall, AddSessions. Для обновления данных в указанных таблицах он может воспользоваться функциями UpdateMovie, UpdateHalls, UpdateSessions. В случае, если данные устаревают или становятся неактуальными, администратор может удалить их, используя процедуры DeleteMovie, DeleteHalls, DeleteSessions.

Важной задачей администратора является регистрация пользователей для обеспечения доступа и взаимодействия с базой данных. Для этого предусмотрена функция RegistrationUser. В случае нарушения правил использования продукта, администратор может удалить информацию и данные о пользователе.

## Сторона пользователя

Пользователь, начиная работу в системе, должен сначала осуществить вход под своей учетной записью. После успешной авторизации становятся доступными функции, предназначенные для конечного пользователя.

Для бронирования места на сеансе и последующей покупки билета предназначены процедуры MakeReservation и BuyTicket.

Одним из важных функционалов для пользователя является возможность просмотра своих собственных бронирований, что выполняется с использованием процедуры ViewUserReservations. В случае необходимости отмены бронирования, пользователь может воспользоваться процедурой CancelReservation.

Для более удобного планирования посещения кинотеатра предоставлены функции ViewSessionsByDate, позволяющая просматривать сеансы на указанную дату, GetFreeSeatsForSession для демонстрации свободных мест сеанса и GetScheduleForNextWeek, выводящая расписание на следующую неделю.

# Заключение

В рамках предоставленной курсовой работы была реализована база данных для кинотеатра, используя систему управления базами данных Oracle. Процесс начался с тщательного проектирования структуры базы данных, включая определение сущностей, их атрибутов и взаимосвязей. Модель базы данных была разработана с учетом основных потребностей кинотеатра, включая информацию о фильмах, залах, сеансах, бронированиях и пользователях.

Создание объектов базы данных включало в себя создание ключевых таблиц, таких как Movies, Halls, Sessions, Reservations и Users. Каждая из этих таблиц была разработана для оптимального хранения и управления соответствующей информацией. Были также созданы индексы и ограничения для обеспечения эффективности и целостности базы данных.

Особое внимание уделялось безопасности системы. С использованием ролей и профилей безопасности был создан механизм управления доступом к данным, а соответствующие процедуры обеспечивают создание пользователей и присвоение им необходимых привилегий.

Работа также включала этап тестирования производительности базы данных. Проведенные тесты подтвердили стабильность и эффективность разработанной системы при работе с большим объемом данных.

Было разработано руководство пользователя, которое содержит инструкции по использованию базы данных кинотеатра. В нем описаны основные функции и возможности системы.

Использование технологии применением мультимедийных типов данных с Oracle Multimedia оказалось весьма полезным для расширения функциональности и улучшения производительности базы данных.

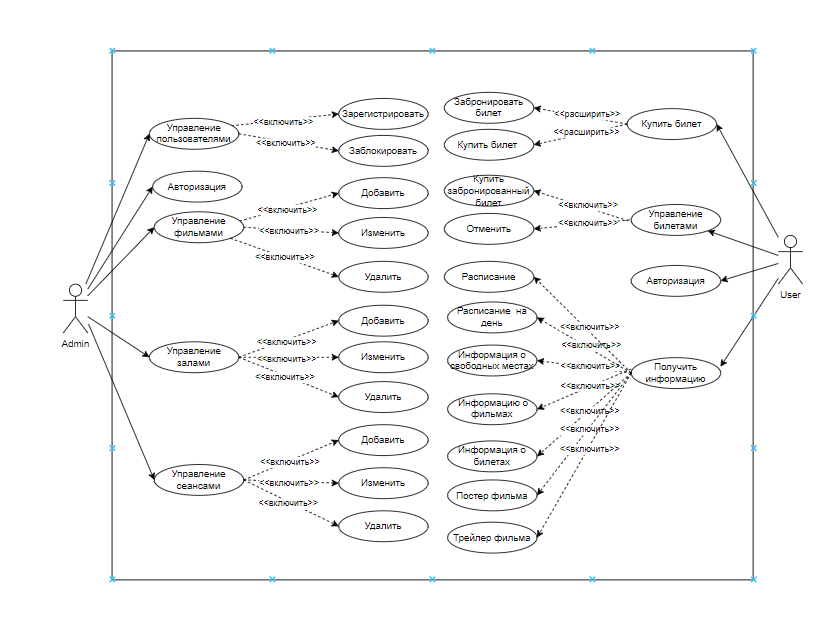
Список используемых источников

1. Официальный сайт Oracle [Электронный ресурс] / Режим доступа – URL:

https://www.oracle.com/ – Дата доступа: 05.10.2023.

1. Официальная документация Oracle [Электронный ресурс] / Режим доступа – URL: https://docs.oracle.com/en/ – Дата доступа: 07.10.2023.
2. Информационный портал Oracle-patches [Электронный ресурс] / Режим доступа – URL: https://oracle-patches.com/ – Дата доступа: 10.10.2023.
3. Работа с файлами в Oracle, [Электронный ресурс] / Режим доступа: https://docs.oracle.com/cd/F49540\_01/DOC/server.815/a68001/utl\_file.htm. – Дата доступа: 22.10.2023.

# ПРИЛОЖЕНИЕ A



# ПРИЛОЖЕНИЕ Б

|  |
| --- |
| CREATE OR REPLACE PACKAGE admin\_package AS  FUNCTION IsMovieExists(p\_ID IN NUMBER) RETURN BOOLEAN;  FUNCTION IsHallExists(p\_ID IN NUMBER) RETURN BOOLEAN;  FUNCTION IsSessionExists(p\_ID IN NUMBER) RETURN BOOLEAN;  FUNCTION IsReservationExists(p\_ID IN NUMBER) RETURN BOOLEAN;  FUNCTION IsUserExists(p\_ID IN NUMBER) RETURN BOOLEAN;  PROCEDURE RegistrationUser(  p\_Login IN VARCHAR2,  p\_Password IN VARCHAR2,  p\_Email IN VARCHAR2);  PROCEDURE AddMovie(  p\_Title IN VARCHAR2,  p\_Genre IN VARCHAR2,  p\_Director IN VARCHAR2,  p\_Description IN VARCHAR2,  p\_Duration IN NUMBER,  p\_Poster IN VARCHAR2,  p\_Trailer IN VARCHAR2  );  PROCEDURE AddHall(  p\_HallName IN VARCHAR2,  p\_RowsCount IN NUMBER,  p\_SeatsCount IN NUMBER  );  PROCEDURE AddSession(  p\_MovieID IN NUMBER,  p\_HallID IN NUMBER,  p\_StartTime IN VARCHAR2  );  PROCEDURE DeleteMovie(  p\_MovieID IN NUMBER  );  PROCEDURE DeleteHall(  p\_HallID IN NUMBER  );  PROCEDURE DeleteSession(  p\_SessionID IN NUMBER  );  PROCEDURE DeleteReservation(  p\_ReservationID IN NUMBER  );  PROCEDURE DeleteUser(  p\_UserID IN NUMBER  );  PROCEDURE UpdateMovie(  p\_MovieID IN NUMBER,  p\_Title IN VARCHAR2,  p\_Genre IN VARCHAR2,  p\_Director IN VARCHAR2,  p\_Description IN CLOB,  p\_Duration IN NUMBER,  p\_Poster IN VARCHAR2,  p\_Trailer IN VARCHAR2  );  PROCEDURE UpdateHall(  p\_HallID IN NUMBER,  p\_HallName IN VARCHAR2,  p\_RowsCount IN NUMBER,  p\_SeatsCount IN NUMBER  );  PROCEDURE UpdateSession(  p\_SessionID IN NUMBER,  p\_MovieID IN NUMBER,  p\_HallID IN NUMBER,  p\_StartTime IN varchar2  );  FUNCTION IsDateValid(  p\_InputDate IN DATE,  p\_MovieID IN Number,  p\_HallID in Number) return boolean;  END admin\_package; |

# ПРИЛОЖЕНИЕ В

|  |
| --- |
| CREATE OR REPLACE PACKAGE user\_package AS  FUNCTION IsMovieExists(p\_ID IN NUMBER) RETURN BOOLEAN;  FUNCTION IsSessionExists(p\_ID IN NUMBER) RETURN BOOLEAN;  FUNCTION IsReservationExists(p\_ID IN NUMBER) RETURN BOOLEAN;  PROCEDURE GetFreeSeatsForSession(  p\_SessionID IN NUMBER  );  procedure GetScheduleForNextWeek;  PROCEDURE MakeReservation(  p\_SessionID IN NUMBER,  p\_RowNumber IN NUMBER,  p\_SeatNumber IN NUMBER  );  PROCEDURE BuyTicket(  p\_SessionID IN NUMBER,  p\_RowNumber IN NUMBER,  p\_SeatNumber IN NUMBER  );  PROCEDURE UpdateReservationStatus(  p\_ReservationID IN NUMBER);  PROCEDURE GetMovieByID(  p\_ID IN NUMBER);  PROCEDURE ViewUserReservations;  PROCEDURE CancelReservation(  p\_ReservationID IN NUMBER  );  FUNCTION OpenPoster(  p\_movie\_id IN NUMBER  ) RETURN BLOB;  FUNCTION OpenTrailer(  p\_movie\_id IN NUMBER  ) RETURN BLOB;  CURSOR c\_SessionsByDate(p\_Date VARCHAR2) IS  SELECT  M.Title AS MovieTitle,  S.StartTime AS SessionDate,  H.HallName  FROM  Sessions S  JOIN Movies M ON S.MovieID = M.ID  JOIN Halls H ON S.HallID = H.ID  WHERE  TRUNC(S.StartTime) = TRUNC(TO\_DATE(p\_Date, 'YYYY-MM-DD'));  PROCEDURE ViewSessionsByDate(  p\_Date IN VARCHAR2  );  END user\_package; |

# ПРИЛОЖЕНИЕ Г

|  |
| --- |
| create directory XML\_DIR as 'C:\bd\xml';  create or replace PROCEDURE EXPORT\_TO\_FILE(  p\_query IN NVARCHAR2,  p\_filename IN NVARCHAR2  )AS  v\_clob NCLOB;  v\_file UTL\_FILE.FILE\_TYPE;  BEGIN  SELECT DBMS\_XMLGEN.GETXML(p\_query) INTO v\_clob FROM DUAL;  v\_file := UTL\_FILE.FOPEN('XML\_DIR', p\_filename || '.xml', 'w');  BEGIN  UTL\_FILE.PUT(v\_file, v\_clob);  EXCEPTION  WHEN UTL\_FILE.WRITE\_ERROR THEN  DBMS\_OUTPUT.PUT\_LINE('Error writing to file');  END;  UTL\_FILE.FCLOSE(v\_file);  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('Query returned no data');  WHEN UTL\_FILE.INVALID\_PATH THEN  DBMS\_OUTPUT.PUT\_LINE('Invalid path for directory');  WHEN UTL\_FILE.INVALID\_MODE THEN  DBMS\_OUTPUT.PUT\_LINE('Invalid file mode');  WHEN UTL\_FILE.INVALID\_FILEHANDLE THEN  DBMS\_OUTPUT.PUT\_LINE('Invalid file handle');  WHEN UTL\_FILE.INVALID\_OPERATION THEN  DBMS\_OUTPUT.PUT\_LINE('Invalid file operation');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Unknown error: ' || SQLERRM);  END;  begin  EXPORT\_TO\_FILE('select \* from sessions', 'exportfile');  end;  CREATE OR REPLACE PROCEDURE import\_data\_xml(p\_file IN NVARCHAR2) AS  v\_xml XMLTYPE := XMLTYPE(p\_file);  BEGIN  FOR item IN (  SELECT extractvalue(value(r), '/ROW/ID') AS ID,  extractvalue(value(r), '/ROW/MovieID') AS MovieID,  extractvalue(value(r), '/ROW/HallID') AS HallID,  extractvalue(value(r), '/ROW/StartTime') AS StartTime  FROM TABLE (XMLSEQUENCE(EXTRACT(v\_xml, '/ROWSET/ROW'))) r  )  LOOP  DBMS\_OUTPUT.PUT\_LINE(item.id ||' '|| item.MovieID|| item.HallID||item.StartTime);  END LOOP;  COMMIT;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Error occurred: ' || SQLERRM);  END;  begin  import\_data\_xml('exportfile');  end;  create PROCEDURE FILE\_TO\_CLOB  (  p\_file\_name IN NVARCHAR2,  p\_clob OUT CLOB  )  AS  v\_file UTL\_FILE.FILE\_TYPE;  v\_filename NVARCHAR2(100);  v\_buffer NVARCHAR2(32767);  BEGIN  v\_filename := p\_file\_name || '.xml';  v\_file := UTL\_FILE.FOPEN('XML\_DIR', v\_filename, 'r');  LOOP  UTL\_FILE.GET\_LINE(v\_file, v\_buffer);  IF v\_buffer = '</ROWSET>' THEN  p\_clob := p\_clob || v\_buffer;  EXIT;  ELSE  p\_clob := p\_clob || v\_buffer;  END IF;  END LOOP;  UTL\_FILE.FCLOSE(v\_file);  END;  create or replace PROCEDURE import\_data  IS  v\_clob CLOB;  BEGIN  FILE\_TO\_CLOB('exportfile', v\_clob);  import\_data\_xml(v\_clob);  commit;  end;  begin  import\_data;  end; |