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

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-98 01 03 «Программное обеспечение информационной безопасности мобильных систем»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА**

по дисциплине «Базы данных»

Тема: «Реализация базы данных сети кинотеатров с применением технологии мультимедийных типов данных в БД»

**Исполнитель**

студент 2 курса 7 группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тышкевич Р.А.

подпись, дата

**Руководитель**

ассистент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.И. Комарова

(должность, уч. звание) (подпись, дата)

Допущен(а) к защите \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

дата, подпись

Курсовой проект защищен с оценкой

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.И. Комарова

подпись дата инициалы и фамилия

Минск 2023

**Содержание**

[Введение 3](#_Toc122118153)

[1 Постановка задачи 4](#_Toc122118154)

[2 Проектирование базы данных 6](#_Toc122118155)

[3 Разработка объектов базы данных 7](#_Toc122118156)

[4 Описание процедур импорта и экспорта 14](#_Toc122118170)

[5 Тестирование производительности 15](#_Toc122118171)

[6 Описание технологии и ее применения в базе данных 16](#_Toc122118172)

[7 Краткое описание приложения для демонстрации 18](#_Toc122118173)

[8 Руководство пользователя 20](#_Toc122118174)

[Заключение 23](#_Toc122118175)

[Список используемых источников 24](#_Toc122118176)

[Приложение 25](#_Toc122118177)

Введение

В современном мире, кинотеатры остаются одним из самых популярных мест, где люди проводят свое свободное время. Однако, с развитием технологий и появлением альтернативных способов просмотра фильмов, кинотеатры вынуждены постоянно совершенствоваться и предоставлять своим зрителям все новые возможности и услуги.

База данных — это организованная структура, предназначенная для хранения информации, систематизированная таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины. Реляционная база данных — база данных, основанная на реляционной модели данных. В качестве СУБД для базы данных была выбрана Oracle 12c, в связи с ее высокой производительностью и надежностью.

Так же было необходимо разработать приложение для демонстрации работы базы данных, взаимодействия с ней. Приложение было написано на языке программирования C# с графическим интерфейсом, выполненным с помощью фреймворка WinForms. Для взаимодействия с сервером базы данных Oracle использовался внутренний функционал WinForms.

Цель данного курсового проекта - разработать базу данных для сети кинотеатров, которая позволит автоматизировать и улучшить процессы бронирования билетов, управления афишами и анализа работы кинотеатров. В рамках проекта будут реализованы функции бронирования фильма, управления афишами, роли администратора и зрителя, дополнительные возможности, такие как покупка еды онлайн и заказ такси, а также анализ работы кинотеатра, который предоставит информацию о количестве гостей и проданных билетах в зависимости от фильма.

Одной из ключевых особенностей разработанной базы данных является применение технологии мультимедийных типов данных в БД. Это позволит улучшить процесс хранения и обработки данных о фильмах, а также обеспечить более удобный доступ к этим данным для зрителей и администраторов.

В итоге, создание такой базы данных сможет значительно повысить уровень обслуживания зрителей, улучшить управление афишами и работой кинотеатра в целом, а также обеспечить эффективный анализ работы кинотеатра.

1. Постановка задачи

Цель данного курсового проекта заключается в создании базы данных для сети кинотеатров с применением технологии мультимедийных типов данных. Это позволит создать удобную и функциональную систему, которая будет обеспечивать бронирование фильмов, управление афишами, роли администратора и зрителя, дополнительные возможности и анализ работы кинотеатра.

Для достижения поставленной цели, в процессе разработки базы данных будут решаться следующие задачи:

* Проектирование базы данных, учитывающей необходимый функционал. Для этого потребуется определить таблицы, их атрибуты, связи между таблицами и т.д.
* Создание объектов базы данных: таблиц, ключей, связей и т.д. В процессе разработки базы данных потребуется создать все необходимые объекты, установить связи между ними и настроить их работу.
* Разработка процедур импорта и экспорта данных. Для удобства работы с базой данных необходимо будет разработать процедуры импорта и экспорта данных, чтобы можно было удобно перемещать информацию между базами данных и программами.
* Тестирование производительности базы данных. Для того чтобы убедиться в эффективности работы базы данных, потребуется провести тестирование производительности и оптимизировать работу базы данных, если потребуется.
* Исследование и описание технологии мультимедийных типов данных и ее применения в базе данных. Для создания базы данных с применением мультимедийных типов данных потребуется изучить соответствующие технологии и определить наиболее эффективный способ их применения.
* Разработка приложения для демонстрации функционала базы данных. После создания базы данных потребуется разработать приложение, которое позволит демонстрировать ее функциональность и возможности.

В данном проекте функциональные требования могут быть разделены на три основные категории: требования к системе бронирования, требования к системе администрирования и требования к системе зрителей.

Первым и наиболее важным функциональным требованием является система бронирования. Она должна позволять зрителям выбирать дату, время и название фильма, а также выбирать ряд и место в зале. Кроме того, система бронирования должна предоставлять возможность отмены бронирования.

Вторым важным функциональным требованием является система администрирования. Администратор должен иметь возможность добавлять и удалять афиши фильмов, назначать технические работы, а также управлять афишами, включая добавление фильма с датой, жанром и оценкой пользователей. Все эти функции должны быть доступны через интерфейс приложения для администраторов.

Третье функциональное требование к системе - это система зрителей. Она должна позволять зрителям покупать билеты, возвращать их, покупать дополнительные возможности, такие как заказ еды онлайн, заказ такси и выбор VIP мест в зале. Кроме того, зрители должны иметь возможность оставлять отзывы и ставить оценки к фильмам.

Дополнительные возможности, такие как покупка еды онлайн и заказ такси, должны быть интегрированы в систему, чтобы зрители могли получить полный сервис в одном месте. Также необходимо предусмотреть систему анализа работы кинотеатра, которая предоставляет отчеты о количестве гостей и количестве купленных билетов в зависимости от фильма.

1. Проектирование базы данных

Первым этапом проектирования базы данных сети кинотеатров является определение основных сущностей, которые необходимо хранить в базе данных. Они могут включать в себя информацию о фильмах, залы кинотеатров, расписание сеансов, билеты и т.д. Далее, необходимо определить связи между сущностями и описать атрибуты каждой таблицы.

Для реализации базы данных сети кинотеатров была использована реляционная модель данных, так как она является наиболее распространенной и эффективной для хранения и обработки больших объемов информации. Она позволяет хранить данные в виде таблиц (реляций), где каждая строка представляет отдельную запись, а каждый столбец - отдельный атрибут.

Одним из ключевых аспектов проектирования базы данных является определение первичных и внешних ключей. Первичный ключ - это уникальный идентификатор записи в таблице, который используется для связи с другими таблицами. Внешний ключ - это поле, которое связывает две таблицы и определяет зависимость между ними.

Следующим шагом является создание диаграммы базы данных, которая отображает все таблицы, их связи и атрибуты. Это позволяет легко визуализировать структуру базы данных и логические связи между таблицами.

Для хранения мультимедийных типов данных, таких как видео и аудио, в базе данных можно использовать BFILE (Binary File) тип данных. Он позволяет хранить бинарные объекты в виде двоичных данных внутри базы данных.

При проектировании базы данных для сети кинотеатров было принято решение использовать реляционную модель данных. В структуру базы данных входят таблицы, которые содержат информацию о фильмах, сеансах, зрителях и работниках кинотеатра. Название базы данных было выбрано "CinemaNet".

Все таблицы были разработаны с учетом требований к безопасности и соблюдения трех первых нормальных форм. Были созданы различные пользователи базы данных, каждому из которых предоставлены определенные права доступа. Важной частью базы данных являются представления и хранимые процедуры, которые обеспечивают эффективный доступ к данным.

Таблицы, которые входят в структуру базы данных "CinemaNet", включают в себя: FILMS, HALS, TICKETS, CLIENTS, ACTORS, DIRECTORS, REVIEWS. Каждая из таблиц содержит информацию о фильмах, их актерах и режиссерах, билетах, оплате, зрителях и работниках кинотеатра.

База данных "CinemaNet" обеспечивает высокую производительность приложения или сайта, использующего её, благодаря правильной структуре таблиц и эффективному использованию индексов. Кроме того, база данных позволяет удобно и безопасно хранить, и управлять данными, связанными со сетью кинотеатров.

1. Разработка объектов базы данных

Конечная база данных для сети кинотеатров включает в себя несколько таблиц, которые хранят информацию о фильмах, сеансах, зале, билетах и клиентах. Все таблицы в базе данных соблюдают требования трех первых нормальных форм и обеспечивают безопасность и производительность.

* Таблица HALLS:

Содержит информацию о залах кинотеатра:

HALL\_NUMBER (номер зала, тип NUMBER, первичный ключ)

SEATS\_COUNT (количество мест в зале, тип NUMBER)

* Таблица ACTORS:

Содержит информацию об актерах:

NAME (имя актера, тип VARCHAR2(50), первичный ключ)

BIRTH\_DATE (дата рождения актера, тип DATE)

* Таблица CLIENTS:

Содержит информацию о клиентах:

ID (идентификатор клиента, тип NUMBER, первичный ключ)

NAME (имя клиента, тип VARCHAR2(50))

EMAIL (адрес электронной почты клиента, тип VARCHAR2(50))

PHONE (телефон клиента, тип VARCHAR2(50))

* Таблица DIRECTORS:

Содержит информацию о режиссерах:

NAME (имя режиссера, тип VARCHAR2(50), первичный ключ)

BIRTH\_DATE (дата рождения режиссера, тип DATE)

* Таблица TICKETS:

Содержит информацию о билетах на фильмы:

ID (идентификатор билета, тип NUMBER, первичный ключ)

FILM\_NAME (название фильма, тип VARCHAR2(50))

HALL\_ID (идентификатор зала, тип NUMBER, внешний ключ, ссылается на HALLS(HALL\_NUMBER))

CLIENT\_ID (идентификатор клиента, тип NUMBER, внешний ключ, ссылается на CLIENTS(ID))

PRICE (цена билета, тип NUMBER)

START\_TIME (время начала фильма, тип TIMESTAMP)

* Таблица REVIEWS:

Содержит информацию об отзывах о фильмах:

FILM\_NAME (название фильма, тип VARCHAR2(50))

CLIENT\_ID (идентификатор клиента, тип NUMBER, внешний ключ, ссылается на CLIENTS(ID))

RATING (оценка фильма, тип NUMBER)

REVIEW\_TEXT (текст отзыва, тип VARCHAR2(500), первичный ключ)

* Таблица FILMS:

Содержит информацию о фильмах:

TITLE (название фильма, тип VARCHAR2(50), первичный ключ)

DESCRIPTION (описание фильма, тип VARCHAR2(50))

RELEASE\_DATE (дата выпуска фильма, тип DATE)

DURATION (продолжительность фильма в минутах, тип NUMBER)

AGE\_RESTRICTION (ограничение по возрасту, тип VARCHAR2(50))

FILM\_ACTOR\_1 (имя актера 1, тип VARCHAR2(50), внешний ключ, ссылается на ACTORS(NAME))

FILM\_ACTOR\_2 (имя актера 2, тип VARCHAR2(50), внешний ключ, ссылается на ACTORS(NAME))

FILM\_ACTOR\_3 (имя актера 3, тип VARCHAR2(50), внешний ключ, ссылается на ACTORS(NAME))

FILM\_DIRECTORS (имя режиссера, тип VARCHAR2(50), внешний ключ, ссылается на DIRECTORS(NAME))

FILM\_REVIEW (текст отзыва, тип VARCHAR2(500), внешний ключ, ссылается на REVIEWS(REVIEW\_TEXT))

GENRE (жанр фильма, тип VARCHAR2(50))

POSTER (путь к файлу постера фильма, тип BFILE)

Диаграмма таблиц базы данных – это графическое представление структуры базы данных и связей между таблицами, хранящимися в этой базе данных. Диаграмма таблиц базы данных изображена на рисунке 3.1

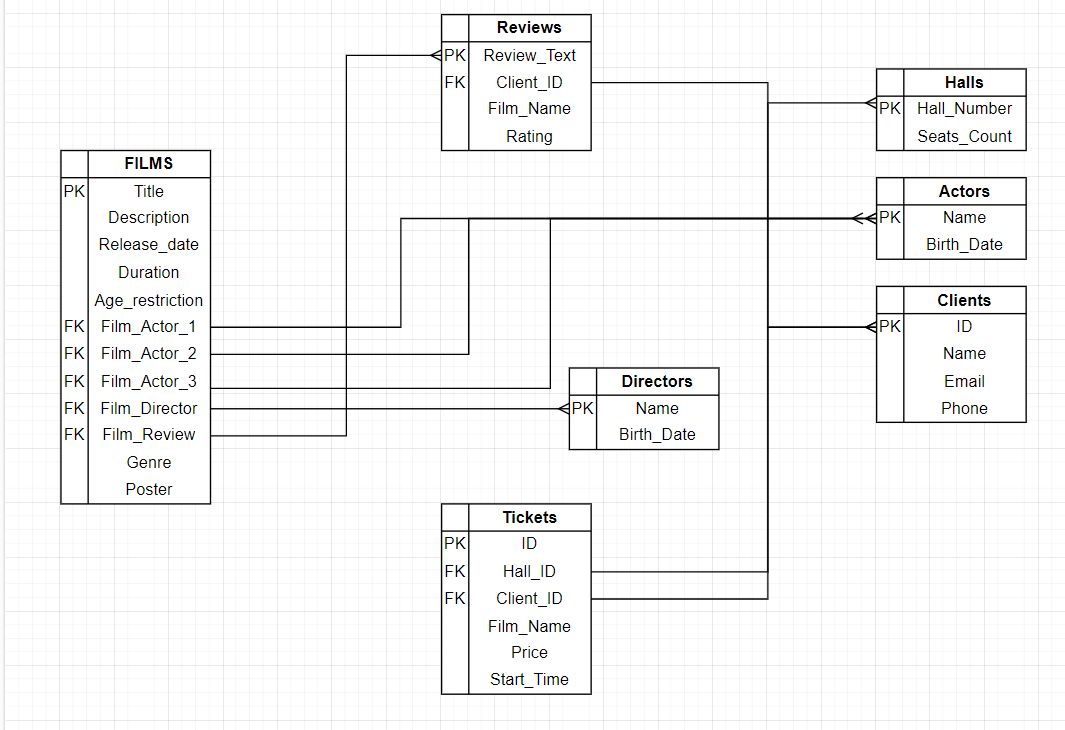


Рисунок 2.1 – Диаграмма базы данных кинотеатра

Для управления данными через приложение пользователи и администраторы использует хранимые процедуры. Хранимая процедура – объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере.

Написанные мной в ходе разработки курсового проекта процедуры можно разбить на несколько категорий:

* выборка данных из таблиц,
* выборка данных со статистикой,
* выборка данных по поисковому запросу с пагинацией,
* заполнение таблиц 100 000 строк,
* добавление данных в таблицы,
* удаление данных из таблиц,
* изменение данных в таблицах,

Для получения информации из таблиц базы данных были созданы две процедуры.

Процедура get\_ticket\_count позволяет получить количество купленных билетов в заданном диапазоне дат. Она принимает два параметра:

start\_date: начальная дата диапазона (тип данных: DATE).

end\_date: конечная дата диапазона (тип данных: DATE).

Результат, количество купленных билетов, возвращается через выходной параметр ticket\_count (тип данных: NUMBER).

Процедура get\_ticket\_count\_by\_film позволяет получить количество купленных билетов для определенного фильма. Она принимает следующие параметры:

movie\_name: название фильма (тип данных: VARCHAR2).

Результат, количество купленных билетов, возвращается через выходной параметр ticket\_count (тип данных: NUMBER).

Эти процедуры позволяют осуществлять выборку данных из основных таблиц базы данных, связанных с информацией о билетах. Для заполнения таблиц воспользуйтесь следующими процедурами: FILL\_HALLS, FILL\_ACTORS, FILL\_CLIENTS, FILL\_DIRECTORS, FILL\_TICKETS, FILL\_REVIEWS и FILL\_FILMS.

Процедура FILL\_HALLS используется для заполнения таблицы с информацией о залах. Вы можете передать необходимые параметры, такие как номер зала, вместимость, тип зала и другие данные, чтобы заполнить таблицу соответствующим образом.

Аналогично, процедура FILL\_ACTORS предназначена для заполнения таблицы с информацией об актерах. Вы можете передать данные об актере, такие как имя, фамилия, дата рождения и другие, чтобы добавить соответствующую информацию в таблицу.

Процедура FILL\_CLIENTS используется для заполнения таблицы с информацией о клиентах. Вы можете передать данные о клиенте, такие как имя, фамилия, контактная информация и другие, чтобы добавить соответствующую запись в таблицу.

Аналогично, процедура FILL\_DIRECTORS предназначена для заполнения таблицы с информацией о режиссерах. Вы можете передать данные о режиссере, такие как имя, фамилия, дата рождения и другие, чтобы добавить соответствующую информацию в таблицу.

Для заполнения таблицы с информацией о билетах используйте процедуру FILL\_TICKETS. Вы можете передать данные о билете, такие как идентификатор клиента, идентификатор сеанса, дата покупки и другие, чтобы добавить соответствующую запись в таблицу.

Аналогично, процедура FILL\_REVIEWS предназначена для заполнения таблицы с информацией о рецензиях на фильмы. Вы можете передать данные о рецензии, такие как идентификатор фильма, текст рецензии, оценка и другие, чтобы добавить соответствующую информацию в таблицу.

Наконец, процедура FILL\_FILMS используется для заполнения таблицы с информацией о фильмах. Вы можете передать данные о фильме, такие как название, жанр, рейтинг, год выпуска и другие, чтобы добавить соответствующую запись в таблицу. Пример процедуры FILL\_ACTORS представлен на листинге 3.1.

CREATE OR REPLACE PROCEDURE FILL\_ACTORS IS

BEGIN

INSERT INTO ACTORS (NAME, BIRTH\_DATE) VALUES ('John Doe', TO\_DATE('1990-01-01', 'YYYY-MM-DD'));

INSERT INTO ACTORS (NAME, BIRTH\_DATE) VALUES ('Jane Smith', TO\_DATE('1985-03-15', 'YYYY-MM-DD'));

INSERT INTO ACTORS (NAME, BIRTH\_DATE) VALUES ('Michael Johnson', TO\_DATE('1982-07-20', 'YYYY-MM-DD'));

INSERT INTO ACTORS (NAME, BIRTH\_DATE) VALUES ('Emily Davis', TO\_DATE('1993-09-10', 'YYYY-MM-DD'));

INSERT INTO ACTORS (NAME, BIRTH\_DATE) VALUES ('David Wilson', TO\_DATE('1988-05-05', 'YYYY-MM-DD'));

INSERT INTO ACTORS (NAME, BIRTH\_DATE) VALUES ('Sarah Thompson', TO\_DATE('1992-11-30', 'YYYY-MM-DD'));

INSERT INTO ACTORS (NAME, BIRTH\_DATE) VALUES ('Robert Lee', TO\_DATE('1980-12-25', 'YYYY-MM-DD'));

INSERT INTO ACTORS (NAME, BIRTH\_DATE) VALUES ('Jennifer Brown', TO\_DATE('1987-04-12', 'YYYY-MM-DD'));

INSERT INTO ACTORS (NAME, BIRTH\_DATE) VALUES ('Daniel Miller', TO\_DATE('1991-08-18', 'YYYY-MM-DD'));

INSERT INTO ACTORS (NAME, BIRTH\_DATE) VALUES ('Olivia Wilson', TO\_DATE('1995-06-22', 'YYYY-MM-DD'));

INSERT INTO ACTORS (NAME, BIRTH\_DATE) VALUES ('Matthew Taylor', TO\_DATE('1984-02-08', 'YYYY-MM-DD'));

INSERT INTO ACTORS (NAME, BIRTH\_DATE) VALUES ('Emma Martinez', TO\_DATE('1994-10-04', 'YYYY-MM-DD'));

INSERT INTO ACTORS (NAME, BIRTH\_DATE) VALUES ('William Anderson', TO\_DATE('1983-11-14', 'YYYY-MM-DD'));

INSERT INTO ACTORS (NAME, BIRTH\_DATE) VALUES ('Sophia Hernandez', TO\_DATE('1996-03-07', 'YYYY-MM-DD'));

INSERT INTO ACTORS (NAME, BIRTH\_DATE) VALUES ('James Garcia', TO\_DATE('1989-09-25', 'YYYY-MM-DD'));

COMMIT;

END;

Листинг 3.1 – Процедура FILL\_ACTORS

Для заполнения таблиц было разработана процедура INSERT\_100000\_CLIENTS, заполняющая таблицу CLIENTS 100000 клиентами. Процедура представлена на листинге 3.2

CREATE OR REPLACE PROCEDURE INSERT\_100000\_CLIENTS AS

v\_phone\_prefix VARCHAR2(4) := '+375'; -- Префикс телефонного номера

v\_phone\_length NUMBER := 9; -- Длина номера телефона без префикса

v\_phone\_number VARCHAR2(20);

BEGIN

FOR i IN 16..100000 LOOP

v\_phone\_number := v\_phone\_prefix || LPAD(TRUNC(DBMS\_RANDOM.VALUE(0, POWER(10, v\_phone\_length))), v\_phone\_length, '0');

INSERT INTO CLIENTS (ID, NAME, EMAIL, PHONE)

VALUES (i, 'Client ' || i, 'client' || i || '@example.com', v\_phone\_number);

END LOOP;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('100000 clients successfully inserted into CLIENTS table.');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error occurred: ' || SQLERRM);

ROLLBACK;

END;

Листинг 3.2 – Процедура INSERT\_100000\_CLIENTS

Процедуры значительно упрощают управление данными в системе и обеспечивают удобство для клиентов и администраторов кинотеатра. Они являются важным функциональным компонентом базы данных, поддерживая ее эффективность и надежность. Поэтому были разработаны следующие процедуры:

* Процедура покупки билета (BUY\_TICKET), позволяющая клиентам приобретать билеты на сеансы в кинотеатре. Она принимает в качестве входных параметров идентификатор клиента, идентификатор сеанса, номер места и другие дополнительные параметры (например, тип оплаты), а также проверяет наличие свободных мест на сеансе. После успешной покупки процедура обновляет соответствующие данные в базе данных. Процедура представлена на листинге 3.3

CREATE OR REPLACE PROCEDURE buy\_ticket(

p\_film\_name VARCHAR2,

p\_hall\_id NUMBER,

p\_client\_id NUMBER,

p\_price NUMBER,

p\_start\_time TIMESTAMP

) AS

last\_ticket\_id NUMBER;

new\_ticket\_id NUMBER;

BEGIN

-- Получение последнего ID билета

SELECT MAX(ID) INTO last\_ticket\_id FROM TICKETS;

-- Покупка нового билета

new\_ticket\_id := last\_ticket\_id + 1;

INSERT INTO TICKETS (ID, FILM\_NAME, HALL\_ID, CLIENT\_ID, PRICE, START\_TIME)

VALUES (new\_ticket\_id, p\_film\_name, p\_hall\_id, p\_client\_id, p\_price, p\_start\_time);

-- Коммит транзакции

COMMIT;

-- Вывод информации о купленном билете

DBMS\_OUTPUT.PUT\_LINE('Билет успешно куплен. ID билета: ' || new\_ticket\_id);

EXCEPTION

WHEN OTHERS THEN

-- Обработка ошибок

DBMS\_OUTPUT.PUT\_LINE('Ошибка при покупке билета: ' || SQLERRM);

ROLLBACK;

END;

Листинг 3.3 – Процедура BUY\_TICKET

* Процедура оставки отзыва (ADD\_REVIEW), позволяющая клиентам оставлять свои отзывы и оценки на фильмы, которые они посмотрели в кинотеатре. Она принимает в качестве входных параметров идентификатор клиента, идентификатор фильма, текст отзыва и оценку. После успешной оставки отзыва процедура добавляет его в базу данных. Процедура представлена на листинге 3.4

CREATE OR REPLACE PROCEDURE add\_review(

p\_film\_name VARCHAR2,

p\_client\_id NUMBER,

p\_rating NUMBER,

p\_comment VARCHAR2

) AS

film\_count NUMBER;

BEGIN

-- Проверка существования фильма

SELECT COUNT(\*) INTO film\_count FROM TICKETS WHERE FILM\_NAME = p\_film\_name;

IF film\_count = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Фильм с названием ' || p\_film\_name || ' не найден.');

RETURN;

END IF;

-- Добавление отзыва

INSERT INTO REVIEWS (FILM\_NAME, CLIENT\_ID, RATING, REVIEW\_TEXT)

VALUES (p\_film\_name, p\_client\_id, p\_rating, p\_comment);

-- Коммит транзакции

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Отзыв успешно добавлен.');

EXCEPTION

WHEN OTHERS THEN

-- Обработка ошибок

DBMS\_OUTPUT.PUT\_LINE('Ошибка при добавлении отзыва: ' || SQLERRM);

ROLLBACK;

END;

Листинг 3.4 – Процедура ADD\_REVIEW

* Процедура удаления отзыва (DELETE\_REVIEW), позволяющая клиентам удалять свои ранее оставленные отзывы на фильмы. Она принимает в качестве входных параметров идентификатор клиента и идентификатор отзыва. После успешного удаления процедура обновляет базу данных. Процедура представлена на листинге 3.5

CREATE OR REPLACE PROCEDURE delete\_review(

p\_review\_text VARCHAR2

) AS

v\_review\_count NUMBER;

BEGIN

-- Проверка существования отзыва

SELECT COUNT(\*) INTO v\_review\_count FROM REVIEWS WHERE REVIEW\_TEXT = p\_review\_text;

IF v\_review\_count = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Отзыв с текстом "' || p\_review\_text || '" не найден.');

RETURN;

END IF;

-- Удаление отзыва

DELETE FROM REVIEWS WHERE REVIEW\_TEXT = p\_review\_text;

-- Коммит транзакции

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Отзыв успешно удален.');

EXCEPTION

WHEN OTHERS THEN

-- Обработка ошибок

DBMS\_OUTPUT.PUT\_LINE('Ошибка при удалении отзыва: ' || SQLERRM);

ROLLBACK;

END;

Листинг 3.5 – Процедура DELETE\_REVIEW

* Процедура удаления фильма (DELETE\_FILM), позволяющая администраторам удалять фильмы из базы данных. Она принимает в качестве входных параметров идентификатор фильма. После успешного удаления процедура обновляет базу данных. Процедура представлена на листинге 3.6

CREATE OR REPLACE PROCEDURE DELETE\_FILM (

p\_film\_name IN VARCHAR2

)

IS

BEGIN

UPDATE FILMS SET FILM\_REVIEW = NULL WHERE TITLE = p\_film\_name;

-- Удаляем отзыв связанный с фильмом из таблицы REVIEWS

DELETE FROM REVIEWS WHERE FILM\_NAME = p\_film\_name;

-- Удаляем фильм из таблицы FILMS

DELETE FROM FILMS WHERE TITLE = p\_film\_name;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

-- Обрабатываем ошибку, если произошла

ROLLBACK;

RAISE;

END;

Листинг 3.6 – Процедура DELETE\_FILM

* Процедура покупки билета с дополнительными возможностями (BUY\_TICKET\_WITH\_EXTRAS), позволяющая клиентам приобретать билеты с дополнительными услугами, такими как выбор места, заказ еды и напитков и т.д. Она принимает в качестве входных параметров идентификатор клиента, идентификатор сеанса, тип билета и другие дополнительные параметры. После успешной покупки процедура обновляет базу данных. Процедура представлена на листинге 3.7

CREATE OR REPLACE PROCEDURE buy\_ticket\_with\_extras(

p\_film\_name VARCHAR2,

p\_hall\_id NUMBER,

p\_client\_id NUMBER,

p\_price NUMBER,

p\_start\_time TIMESTAMP

) AS

last\_ticket\_id NUMBER;

new\_ticket\_id NUMBER;

calculated\_price NUMBER;

BEGIN

-- Получение последнего ID билета

SELECT MAX(ID) INTO last\_ticket\_id FROM TICKETS;

-- Покупка нового билета

new\_ticket\_id := last\_ticket\_id + 1;

-- Расчет цены с дополнительными возможностями (удвоение цены)

calculated\_price := p\_price \* 2;

INSERT INTO TICKETS (ID, FILM\_NAME, HALL\_ID, CLIENT\_ID, PRICE, START\_TIME)

VALUES (new\_ticket\_id, p\_film\_name, p\_hall\_id, p\_client\_id, calculated\_price, p\_start\_time);

-- Коммит транзакции

COMMIT;

-- Проверка и предоставление дополнительных возможностей

IF calculated\_price > 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Вы купили билет с дополнительными возможностями.');

DBMS\_OUTPUT.PUT\_LINE('Вам оплачен такси, а также еда внутри кинотеатра стала бесплатной.');

END IF;

-- Вывод информации о купленном билете

DBMS\_OUTPUT.PUT\_LINE('Билет успешно куплен. ID билета: ' || new\_ticket\_id);

EXCEPTION

WHEN OTHERS THEN

-- Обработка ошибок

DBMS\_OUTPUT.PUT\_LINE('Ошибка при покупке билета: ' || SQLERRM);

ROLLBACK;

END;

Листинг 3.7 – Процедура BUY\_TICKET\_WITH\_EXTRAS

* Процедура возврата билета (RETURN\_TICKET), позволяющая клиентам возвращать приобретенные билеты и получать возмещение. Она принимает в качестве входных параметров идентификатор клиента и идентификатор билета. После успешного возврата процедура обновляет базу данных. Процедура представлена на листинге 3.8

CREATE OR REPLACE FUNCTION return\_ticket(

p\_ticket\_id NUMBER

) RETURN BOOLEAN AS

BEGIN

-- Удаление билета

DELETE FROM TICKETS WHERE ID = p\_ticket\_id;

COMMIT;

-- Возвращение значения TRUE, если удаление прошло успешно

RETURN TRUE;

EXCEPTION

WHEN OTHERS THEN

-- Обработка ошибок

DBMS\_OUTPUT.PUT\_LINE('Ошибка при возврате билета: ' || SQLERRM);

ROLLBACK;

RETURN FALSE;

END;

Листинг 3.8 – Процедура RETURN\_TICKET

1. Описание процедур импорта и экспорта

Для обеспечения возможности экспорта и импорта данных для таблицы "ACTORS" были разработаны две процедуры: "ACTORS\_TO\_XML" и "XML\_TO\_ACTORS".

Процедура "ACTORS\_TO\_XML" предназначена для экспорта данных из таблицы "ACTORS" в формате XML. В ходе выполнения процедуры, используется метод "utl\_file" для работы с файловой системой, а объект "XMLType" применяется для обработки XML-данных. Процедура осуществляет чтение данных из таблицы "ACTORS" и создание соответствующего XML-документа, который затем сохраняется в файловой си

Процедура "XML\_TO\_ACTORS" выполняет обратную операцию, позволяя импортировать данные актеров из файла формата XML в таблицу "ACTORS". Процедура считывает XML-документ из файла с помощью метода "utl\_file", а затем преобразует его в объект "XMLType". В дальнейшем, процедура извлекает необходимые данные из XML-объекта и загружает их в таблицу "ACTORS".

Такие процедуры позволяют удобно управлять данными актеров, осуществлять их экспорт и импорт в удобном формате XML, а также обеспечивают гибкость при обмене информацией между различными системами и приложениями. Пример процедуры ACTORS\_TO\_XML представлен на листинге 4.1, а XML\_TO\_ACTORS на листинге 4.2

CREATE OR REPLACE PROCEDURE ACTORS\_TO\_XML

IS

xml\_data CLOB;

file\_handle UTL\_FILE.FILE\_TYPE;

file\_path VARCHAR2(100) := 'D:\Photos\Actors\_info.xml';

BEGIN

SELECT DBMS\_XMLGEN.GETXML('SELECT \* FROM ACTORS') INTO xml\_data FROM DUAL;

-- Открываем файл для записи

file\_handle := UTL\_FILE.FOPEN('MY\_DIR', 'Actors\_info.xml', 'W');

-- Записываем XML-данные в файл

UTL\_FILE.PUT\_RAW(file\_handle, UTL\_RAW.CAST\_TO\_RAW(xml\_data));

-- Закрываем файл

UTL\_FILE.FCLOSE(file\_handle);

DBMS\_OUTPUT.PUT\_LINE('XML файл успешно создан: ' || file\_path);

EXCEPTION

WHEN OTHERS THEN

-- Обрабатываем ошибку, если произошла

DBMS\_OUTPUT.PUT\_LINE('Ошибка при создании XML файла: ' || SQLERRM);

RAISE;

END;

Листинг 4.1 – процедура ACTORS\_TO\_XML

CREATE OR REPLACE PROCEDURE XML\_TO\_ACTORS AS

-- Путь к XML файлу

xml\_path VARCHAR2(100) := 'Actors\_info.xml';

-- XMLType переменная для чтения XML файла

xml\_data XMLType;

BEGIN

-- Чтение XML файла в переменную xml\_data

xml\_data := XMLType(BFILENAME('MY\_DIR', xml\_path), NLS\_CHARSET\_ID('UTF8'));

-- Заполнение таблицы ACTORS из XML данных

INSERT INTO ACTORS (NAME, BIRTH\_DATE)

SELECT

extractvalue(value(p), '/ROW/NAME') AS NAME,

to\_date(extractvalue(value(p), '/ROW/BIRTH\_DATE'), 'DD-MM-YYYY') AS BIRTH\_DATE

FROM

TABLE(xmlsequence(xml\_data.extract('/ROWSET/ROW'))) p;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('XML data successfully loaded into ACTORS table.');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error occurred: ' || SQLERRM);

ROLLBACK;

END;

Листинг 4.2 – процедура XML\_TO\_ACTORS

1. Тестирование производительности

Производительность базы данных в сети кинотеатров является критическим фактором для обеспечения эффективной работы системы и удовлетворения потребностей пользователей. Для достижения высокой производительности базы данных важно оптимизировать запросы, эффективно использовать индексы и правильно организовать структуру базы данных.

Для оценки производительности была выбрана процедура "get\_ticket\_count\_by\_film", которая получает количество купленных билетов в зависимости от фильма. В рамках тестирования процедура была вызвана с определенными параметрами, например, для получения количества купленных билетов для фильмов с названием "Fight Club".

Результаты выполнения запроса показывают высокую скорость выполнения процедуры. Например, процедура "get\_ticket\_count\_by\_film" может выполняться совсем недолго за 182 мс. Время выполнения запроса составляет 162 мс, что уходит на получение необходимых данных из базы данных, и всего 20 мс требуется на исполнение самой процедуры.

Такие результаты свидетельствуют о высокой производительности базы данных и эффективности процедуры "get\_ticket\_count\_by\_film". Это позволяет обеспечить быстрый доступ к данным и оперативное получение информации о количестве купленных билетов для конкретного фильма, что является важным для удобства и удовлетворения потребностей пользователей системы кинотеатров. Исходя из полученных данных можно сделать вывод, что тест производительности пройден.

1. Описание технологии и ее применения в базе данных

В базе данных реализовано хранение мультимедийных данных. В таблице FILMS в столбце POSTER хранится информация о посторе к фильму в формате BFILE. Фрагмент данных таблицы можно увидеть на рисунке 6.1



Рисунок 6.1 - Фрагмент данных таблицы FILMS

Таким образом реализуется хранение файлов c картинками в базе данных. Стоит отметить, что в ходе разработки базы данных использование этой технологии являлась экспериментом, так как хорошо известно из опыта разработчиков баз данных, что хранить мультимедийные файлы лучше в файловой системе, так как при обращении, например, с файлами картинок, резко повышается кол-во данных, передаваемых в запросе и скорость этих самых запросов. Также обычно имеют ограничения на размер хранимых данных. Это может ограничивать количество мультимедиа, которое может быть храниться в базе данных. Даже использование бинарных форматов данных не сильно меняют эту ситуацию. Бинарные файлы, такие как изображения и видео, обычно имеют большой размер. Это может привести к тому, что база данных будет занимать большое количество места на жестком диске и потреблять больше ресурсов при бэкапах. Хоть для маленького приложения, разработанного в рамках курсового проекта такое решение выглядит неплохо, в случае расширения использование базы данных от такого подхода придётся отказаться. Возможно, в будущем в случае изобретение более эффективных по памяти способов хранения данных вопросы скорости и размеров отойдут на второй план, а на первый выйдет логическая структура хранения и тогда хранение мультимедийных файлов в базе данных станет неотъемлемой частью любого приложения.

Заключение

Базы данных играют ключевую роль в надежном хранении и доступе к информации, обеспечивая организациям эффективное управление и администрирование данных. Они способны хранить различные типы данных, включая структурированные данные, текстовые, изображения, аудио и видеофайлы. Базы данных позволяют обрабатывать и обновлять данные эффективно, что является важным для различных бизнес-процессов, таких как обработка заказов, учет рабочего времени и управление складами. Также базы данных могут быть использованы для сбора и анализа статистических данных, что помогает в принятии решений в бизнесе.

В рамках разработки курсового проекта на тему "Реализация базы данных сети кинотеатров с применением технологии мультимедийных типов данных в БД" с использованием Oracle 12c и WinForms, была достигнута поставленная цель. При разработке проекта использовались различные объекты базы данных, включая таблицы, хранимые процедуры, триггеры, представления и индексы.

Важными аспектами разработанной базы данных являются надежность и быстродействие. Она обеспечивает эффективную работу организации благодаря удобному интерфейсу и гибкой системе управления. Также в базе данных уделено внимание безопасности данных для защиты конфиденциальной информации. Различные пользователи базы данных были созданы для разграничения доступа, а запросы к информации выполняются с помощью процедур.

База данных прошла успешное тестирование с использованием большого объема данных, а также были реализованы процедуры для импорта и экспорта данных в формате XML. Технология хранения мультимедийных данных была успешно внедрена, позволяя хранить и обрабатывать файлы различных типов.

Для работы с базой данных было создано клиентское приложение на библиотеке WinForms. Такое решение обеспечивает удобный интерфейс для пользователей и позволяет эффективно взаимодействовать с базой данных.

Полученные результаты работы свидетельствуют о корректной функциональности разработанной базы данных, а все требования, указанные в техническом задании, были выполнены в полном объеме. Это гарантирует надежное хранение и своевременный доступ к информации, обеспечивая эффективное функционирование сети кинотеатров.

Список литературы

1. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: https://metanit.com.

2. Oracle-dba.ru [Электронный ресурс] / Режим доступа: https://oracle-dba.ru.

3. Developing and Using Stored Procedures [Электронный ресурс] /Режим доступа: <https://docs.oracle.com/cd/B2835901/appdev.111/b28843/tdddgprocedures.htm>

4. Stackoverflow.com [Электронный ресурс] / Режим доступа: https://stackoverflow.com – Дата доступа: 08.05.2023

5.Официальный сайт Oracle [Электронный ресурс] / Foundation, Inc. https://www.oracle.com/database/database-vault/index.html– Дата доступа: 10.05.2023.

6.Продукты Oracle [Электронный ресурс] [Электронный ресурс] / Foundation, Inc. http://www.interface.ru/home.asp?artId=24678 – Дата доступа: 22.05.2023.