**ВВЕДЕНИЕ**

Онлайн-сервисы сериалов – это доступные по подписке веб-сервисы, которые позволяют смотреть сериалы на подключенном к интернету устройстве. По мере развития технологий появилась возможность не посещать реальные кинотеатры или дожидаться долгожданной премьеры на каком-либо канале телевидения, а предоставлять пользователю уже готовые сервисы, позволяющие комфортно смотреть нужные ему сериалы. Рост популярности таких технологий приводит к постоянному увеличению объёма информации, которая нуждается в хранении и администрировании. Рационально использовать для этого базы данных База данных – совокупность структурированных взаимосвязанных данных, относящихся к определённой предметной области и организованных таким образом, что эти данные могут быть использованы для решения многих задач многими пользователями. Цель базы данных – помочь работникам и самим компаниям вести учет актуальных данных и осуществлять информационную поддержку своих приложений.

Практически любая современная организация нуждается в базе данных, удовлетворяющей тем или иным потребностям по хранению, управлению и администрированию данных. Онлайн-сервисы сериалов работают с большими объёмами данных, включающих в себя информацию как о пользователях, так и о самих сериалах. Поэтому, безусловно, использование базы данных является актуальной задачей в сегодняшнее время.

Целью данного курсового проекта является разработка веб-приложения, проектирование эффективной и безопасной базы данных для поддержки работы онлайн-сервиса сериалов. Предметная область – сериалы. Перед разработкой были поставлены следующие задачи: работа с сериалами (удаление, добавление, редактирование и просмотр детальной информации, фильтрация по различным параметрам), работа с пользователями (добавление, редактирование информации и просмотр детальной информации), работа с подписками (просмотр, добавление, удаление, и редактирование информации, фильтрация), операции с покупками (добавление, удаление, изменение, просмотр), работа с рейтингами (просмотр по различным критериям, удаление, изменение, добавление). Основное назначение спроектированной базы данных – представление и изменение информации о сериалах, пользователях, рейтингах, подписках и покупках. Курсовой проект проверен в системе «Атиплагиат». Процент оригинальности составляет …

**1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ЕЕ ФОРМАЛИЗАЦИЯ ДЛЯ ПРОЕКТИРОВАНИЯ БАЗЫ ДАННЫХ**

**1.1 Описание предметной области**

Предметная область – часть реального мира, подлежащая изучению с целью организации управления и, в конечном счете, автоматизации.

Предметная область, изучаемая в данном курсовом проектировании, – онлайн-сервис сериалов. Она содержит в себе информацию, которую необходимо формализовать для внесения в базу данных. Результатом формализации будут являться данные в виде сущностей и атрибутов.

Однако, стоит понимать, что для формализации данных необходимо, в первую очередь, выявить полезную информацию, ключевые закономерности и правила, которые распространяются в выбранной предметной области.

Процесс взаимодействия онлайн-сервиса сериалов с пользователем предполагает определенные закономерности. При обращении к онлайн-сервису, пользователю необходимо ввести свои персональные данные, после чего он получает доступ непосредственно к основному функционалу. Пользователь может просматривать краткую информацию о сериалах, которые доступны на сайте (название, жанр). При желании просмотреть сериал, пользователь может получить доступ к полной информации (рейтинг, описание, дату премьеры, количество серий), и, непосредственно, к самому просмотру. Также, для формирования рейтинговой системы, сериалы могут быть оценены по десятибалльной шкале и сопровождены опциональной аннотацией.

В целях извлечения прибыли сервисом, реализована система подписок. Подписки дают право просматривать сериалы, недоступные в стандартном режиме. Пользователь приобретает подписку на срок от одного месяца и на это время ему доступны сериалы, предполагающие просмотр с данным типом подписки на сервис. При покупке рассчитывается дата покупки, срок действия (в месяцах) и суммарная стоимость.

Таким образом, в соответствии с предметной областью система строится с учетом следующих правил:

– необходимым условием использования сервиса является ввод персональных данных;

– каждый пользователь имеет право ставить оценку доступным ему сериалам;

– определенные сериалы могут быть доступны по определенному виду подписки;

– минимальный срок покупки подписки составляет месяц;

Выделим базовые сущности данной предметной области, которые образуют структуру проектируемой ИС, и их атрибуты. ﻿Сущность –это выделенный на концептуальном уровне объект для базы данных. ﻿Атрибуты сущностей содержат данные, внесенные пользователем или администратором.

Сущность пользователя обладает такими атрибутами, как идентификатор, имя, возраст, адрес электронной почты и идентификатор подписки в качестве внешнего ключа.

Сущность подписки содержит следующие атрибуты: идентификатор, наименование, ежемесячная стоимость.

Сущность сериала включает в себя идентификатор, наименование, количество серий, описание, год премьеры, жанр, идентификатор подписки в качестве внешнего ключа.

Сущность рейтинга включает такие атрибуты, как наименование, значение, аннотацию идентификаторы пользователя и сериала в качестве внешних ключей.

Сущность покупки содержит атрибуты: идентификатор, дата, количество месяцев, общая стоимость, идентификаторы пользователя и подписки в качестве внешних ключей.

Для каждой сущности идентификатор выступает в роли первичного ключа.

**1.2 Анализ информационных потребностей пользователей и предварительное описание запросов**

Для базы данных онлайн-сервиса сериалов могут существовать следующие группы пользователей: администратор и посетитель сайта. Для каждой группы пользователей будут существовать свои ограничения на использование данных, за исключением администратора базы данных, который имеет большие привилегии над данными.

Посетитель сайта имеет доступ на просмотр информации на сервисе, совершение покупок подписок, выставление оценок, их удаление, изменение. Он будет иметь ограничение на изменение, обновление и удаление какой-либо информации о сериалах, подписках и других пользователях.

Помимо обычного пользователя, имеется администратор, который занимается обработкой отображаемых данных на сервисе. Обработка отображаемых данных включает в себя просмотр, добавление, удаление, редактирования информации по сериалам, подпискам, рейтингам и пользователям.

Для удобного просмотра контента, который ищет пользователей, сервис поддерживает отображение информации по определенным фильтрам. Например, пользователь сможет осуществить вывод сериалов по дате премьеры (от наиболее старых к наиболее новым или наоборот), по определенному названию, с наибольшим, наименьшим числом серий, доступные по его подписке, просмотреть количество сериалов, сгруппированных по жару, просмотреть все жанры.

Администратору необходим вывод по всем сериалам, что есть на сайте, возможность удалять, обновлять, добавлять новые сериалы.

Пользователь может просматривать данные о себе по собственному идентификатору, но не о других пользователях. Администратор же может добавлять, изменять удалять пользователей, а также просматривать информацию как по идентификатору, так и всех пользователей в базе данных. Администратор имеет доступ к фильтру просмотра всех пользователей, совершивших покупку с возможным указанием минимальной стоимости.

Пользователь может получить информацию по идентификатору конкретной подписки, по всем доступным подпискам, выбрать наиболее дорогую, наиболее дешевую, выбрать среднюю стоимость. Администратору, помимо этих запросов, необходимы удаление, добавление, изменение подписок.

Пользователю понадобится вывод информации по всем его покупкам, по конкретной с помощью идентификатора, а также запрос на добавление новой покупки в базу данных. Администратору понадобится просмотреть самые дорогие покупки, все покупки на сервисе, удалить покупку по идентификатору, изменить данные о покупке.

Как пользователь, так и администратор, захотят сформировать запрос на получение среднего рейтинга сериала, установить рейтинг сериалу. Администратор может удалить рейтинги конкретного сериала, пользователя по их идентификаторам, изменять их или же добавлять.

**1.3 Определение требований и ограничений к базе данных с точки зрения предметной области**

База данных представляет собой совокупность структурированных взаимосвязанных данных, относящихся к определенной предметной области и организованных для решения определенных задач разными пользователями.

Особенности каждой предметной области могут накладывать разнообразные требования и ограничения на хранимые в базе данные.

В проектируемой базе данных будут следующие ограничения:

– все идентификаторы, используемые в качестве первичных и внешних ключей не могут быть пустыми;

– при добавлении новых данных в таблицу значение идентификатора увеличивается на единицу;

– имя и стоимость подписки не могут быть пустыми;

– имя пользователя не может быть пустым;

– адрес электронной почты пользователя не может быть пустым и имеет уникальное значение в таблице;

– ни один атрибут сущности покупки не может быть пустым, так как каждое из них важно как для отображения информации, так и для логики приложения;  
 – имя сериала и количество серий не могут быть пустыми, так как эта информация важна для отображения пользователю;

– атрибут значения сущности рейтинга не может быть пустым, так как иначе данная сущность не имеет смысла.

* 1. **Постановка решаемой задачи**

В процессе постановки решаемой задачи в ходе курсового проектирования задачи необходимо:

Получить практические ﻿навыки по проектированию, разработке и введению в эксплуатацию базы данных с учетом выбора оптимальной системы управления базой данных, спроектировать основные запросы и выбрать клиентское программное обеспечение для работы с базой данных (в дальнейшем, БД). Для достижения поставленной цели необходимо выполнить следующие задачи:

– выполнить изучение предметной области с последующей формализацией,

– выполнить анализ информационных потребностей клиентского программного обеспечения;

– выполнить проектирование, программирование и начальное наполнение базы данных, а также спроектировать основные запросы к БД от нескольких групп потенциальных пользователей.

В результате выполнения поставленных задач в качестве результата планируется получить документацию, дающую ответ о содержании проектного решения, а также само программное обеспечение с веб-интерфейсом, посредством которого различные категории пользователей смогут взаимодействовать с данными в спроектированной БД.

**2 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ ДЛЯ ОСНОВНОГО ВИДА ДЕЯТЕЛЬНОСТИ РАССМАТРИВАЕМОЙ ПРЕДМЕТНОЙ ОБЛАСТИ**

**2.1 Разработка инфологической модели предметной области базы данных**

Проектирование баз данных – одна из наиболее сложных и ответственных задач, связанных с созданием информационной системы. В результате решения этой ﻿задачи должны быть определены содержание базы данных онлайн-сервиса сериалов, эффективный для всех ее будущих пользователей способ организации данных и инструментальные сред﻿ства управления данными.

Инфологическая модель данных – обобщенное неформальное описание создаваемой базы данных, выполненное с использованием естественного средств, понятных всем людям, работающим над проектированием базы данных. Она является человеко-ориентированной моделью, которая полностью независима от физических параметров среды хранения данных. Основная цель инфологического моделирования – обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить в создаваемой базе данных.

. Основными конструктивными элементами инфологических моделей являются сущности, атрибуты и связи между ними. В данном курсовом проекте представлены пять основных сущностей: сериал, пользователь, подписка, рейтинг, покупка. Каждая сущность в свою очередь имеет список атрибутов, по которым будут осуществляться связи. Тип связи будет определять отношения между атрибутами сущности. Сущность (объектное множество, таблица) – абстракция реального или виртуального объекта, процесса, явления, о котором необходимо собирать и хранить информацию. В ходе выполнения данной курсовой работы были спроектированы следующие таблицы:

− сущность «user» (информация о пользователях, их возраст и электронная почта);

− сущность «serial» (информация о сериале, подписке по которому он доступен);

− сущность «rating» (рейтинг, выставленный конкретным пользователем конкретному сериалу, аннотация);

− Сущность «subscription» (информация о видах подписок);

− Сущность «purchase» (информация о совершенных покупках, полной стоимости, какой пользователь какую подписку приобрёл и на какой срок).

В результате был составлен следующий список атрибутов:

1. Сущность «user»:

­– user\_id;

– age;

– username;

– email;

– subscription\_id;

1. Сущность «serial»:

– serial\_id;

– name;

– amount\_of\_series;

– description;

– genre;

– release\_year;

– subscription\_id;

1. Сущность «rating»:

­– rating\_id;

– value;

– annotation;

– serial\_id;

– user\_id;

1. Сущность «subscription»:

– subscription\_id;

– name;

– price\_per\_month;

1. Сущность «purchase»:

­– purchase\_id;

– date;

– amount\_of\_months;

– total\_price;

– user\_id;

– subscription\_id;

Между данными сущностями можно осуществить 6 связей типа «Один-ко-многим».

В таблице 2.1 представлены связи в спроектированной базе данных:

Таблица 2.1 ­– Классификация связей в базе данных

|  |  |  |  |
| --- | --- | --- | --- |
| Номер связи | Родительская таблица | Дочерняя таблица | Тип связи |
| 1 | users | purchases | 1: M |
| 2 | users | ratings | 1: M |
| 3 | subscription | users | 1: M |
| 4 | subscription | purchases | 1: M |
| 5 | subscription | serials | 1: M |
| 6 | serials | ratings | 1: M |

Например, в таблице подписок есть первичный ключ. Тогда создадим в таблице сериалов внешний ключ subscription\_id. Таким образом, одной подписке может соответствовать множество сериалов. По такому принципу построены все остальные связи типа «Один-ко-многим».

Первичными ключами в спроектированной базе даных будут являться следующие атрибуты:

– user\_id (сущность «user»);

– subscription\_id (сущность « subscription»);

– serial\_id (сущность « serial» );

– purchase\_id (сущность « purchase»);

– rating\_id (сущность « rating»).

**2.2 Выбор и обоснование используемых типов, данных и ограничений (доменов)**

Важным этапом проектирования базы данных является выбор и обоснование типов данных. Любые данные, помещаемые в столбец, должны отвечать этому типу данных. В некоторых случаях следует указывать и допустимую длину данных в столбце. MySQL поддерживает несколько типов столбцов, которые можно разделить на три категории: числовые типы данных, типы данных для хранения даты и времени и символьные (строковые) типы данных. При проектировании базы данных были использованы такие типы данных как:

− DECIMAL;

− INT;

− VARCHAR;

− DATE.

Рассмотрим каждый тип данных в подробном применении к данным, находящимся в таблицах. INT – представляет целые числа, занимает 4 байта. Тип данных VARCHAR имеет переменное количество символов, что позволит сократить память, выделенную для этой строки, если в ней указано символов меньше, чем задано. Тип данных DATE используется для величин с информацией только о дате, без части, содержащей время. MySQL извлекает и выводит величины DATE в формате «YYYY-MM-DD». Поддерживается диапазон величин от «1000-01-01» до «9999-12-31». В MySQL тип данных DECIMAL хранит точные значения с фиксированной точкой. Этот тип данных используется, когда важно сохранить точную точность, например, хранение денежных данных.

Приведем таблицы спроектированной базы данных, охарактеризованные приведёнными типами данных и соответствующими им размерами полей (количество символов). В таблице 2.2 представлен состав таблицы «users»:

Таблица 2.2 ­– Состав таблицы «users»

|  |  |  |
| --- | --- | --- |
| Наименование атрибута | Тип полей | Ключи таблицы |
| user\_id | int | PK |
| age | int | - |
| username | varchar (50) | - |
| email | varchar (60) | - |
| subscription\_id | int | FK |

В таблице 2.3 представлен состав таблицы «ratings»:

Таблица 2.3 ­– Состав таблицы «ratings»

|  |  |  |
| --- | --- | --- |
| Наименование атрибута | Тип полей | Ключи таблицы |
| rating\_id | int | PK |
| value | int | - |
| annotation | varchar (1000) | - |
| serial\_id | int | FK |
| user\_id | int | FK |

В таблице 2.4 представлен состав таблицы «subscriptions»:

Таблица 2.4 ­– Состав таблицы «subscriptions»

|  |  |  |
| --- | --- | --- |
| Наименование атрибута | Тип полей | Ключи таблицы |
| rating\_id | int | PK |
| name | varchar (60) | - |
| price\_per\_month | decimal (19,5) | - |

В таблице 2.5 представлен состав таблицы «purchases»:

Таблица 2.5 ­– Состав таблицы «purchases»

|  |  |  |
| --- | --- | --- |
| Наименование атрибута | Тип полей | Ключи таблицы |
| purchase\_id | int | PK |
| date | date | - |
| amount\_of\_months | int | - |
| total\_price | decimal (19,5) | - |
| user\_id | int | FK |
| subscription\_id | int | FK |

В таблице 2.6 представлен состав таблицы «serials»:

Таблица 2.6 ­– Состав таблицы «serials»

|  |  |  |
| --- | --- | --- |
| Наименование атрибута | Тип полей | Ключи таблицы |
| serial\_id | int | PK |
| name | varchar (60) | - |
| amount\_of\_series | int | - |
| description | varchar (1000) | - |
| genre | varchar (50) | - |
| release\_year | int | - |
| subscription\_id | int | FK |

**2.3 Проектирование запросов к базе данных**

В рамках данного курсового проекта при помощи структурированного языка запросов SQL была организована выборка информации из разработанной ранее базы данных. Были сформулированы запросы различных типов, реализуемых средствами выбранного программного средства.

Для взаимодействия с базой данных посредством программных возможностей языка C# была использована технология Dapper, представляющая собой технологию сопоставления результатов SQL-запросов с классами С#. Для того, чтобы передавать в запросы необходимые параметры, были задействованы возможности Dynamic SQL, позволяющие упростить взаимодействие приложения с базой данных путём создания SQL-запроса непосредственно из кода программы. Таким образом, используя символ “@”, с параметром, во время своего выполнения, программа будет подставлять вместо него конкретное значение.

Простой запрос с выборкой по идентификатору имеет следующую формулировку: выбрать все атрибуты из таблицы «purchases». Он имеет вид:

*SELECT \* FROM purchases WHERE purchase\_id = @Id*

Запрос на обновление данных будет иметь вид:

*UPDATE purchases SET amount\_of\_months = @AmountOfMonths, date = @Date, total\_price = @TotalPrice, user\_id = @UserId, subscription\_id = @SubscriptionId WHERE purchase\_id = @Id*

Здесь подразумевается следующая формулировка: обновить столбцы “amount\_of\_months”, “total\_price”, “user\_id”, “subscription\_id” у сущности покупки с идентификатором, равным какому-то Id.

Запрос на удаление имеет вид:

*DELETE FROM purchases WHERE purchase\_id = @Id*

Он говорит следующее: удалить строку из таблицы “purchases” с идентификатором Id.

Оператор вставки формулируется так: вставить в поля “name”, “price\_per\_month” таблицы “subscriptions” значения Name, PricePerMonth. Сам запрос выглядит следующим образом:

*INSERT INTO subscriptions (name, price\_per\_month) values (@Name, @PricePerMonth)*

Запрос нахождения средней цены выглядит так:

*SELECT AVG(price\_per\_month) AS AveragePrice FROM subscriptions*

Он формулируется так: выбрать среднее значение по столбцу price\_per\_month из таблицы “subscriptions” и результат представить в столбце с названием “AveragePrice”.

Пример запроса с использованием оператора JOIN:

*SELECT ratings.value as Value, ratings.annotation as Annotation, users.username as UserName FROM ratings JOIN users ON users.user\_id = ratings.user\_id WHERE serial\_id = @Id*

Результат выполнения данного запроса вернёт все рейтинги с информацией о значении оценки, имени пользователя и аннотации.

Пример несвязанного подзапроса с оператором ANY:

*SELECT users.user\_id AS Id, users.username AS UserName, users.email AS Email, users.age as Age, purchases.total\_price AS TotalPrice FROM users JOIN purchases on users.subscription\_id = purchases.subscription\_id WHERE purchases.purchase\_id = ANY (SELECT purchase\_id from purchases WHERE total\_price >= @Price)*

Формулируется следующим образом: выбрать значения user\_id, username, email, age из таблицы users, total\_price из purchases путём объединения таблиц users и purchases по условию равенства идентификаторов subscription\_id в этих таблицах, где purchase\_id равно всем идентификаторам сущностей purchase где total\_price больше либо равно значению Price. Таким образом, данный запрос выберет данные пользователей, которые совершили покупку на стоимость не менее, чем значение Price.

Выполнение группировки числа сериалов по их жанрам достигается следующим запросом:

*SELECT genre AS Genre, COUNT(\*) AS Amount FROM serials GROUP BY genre*

Результатом выполнения данного запроса будет являться таблица со столбцами Genre с названием жанра и Amount с соответствующим этому жанру количеством сериалов.

Пример скалярного запроса, выбирающего значения всех жанров из таблицы “serials”:

*SELECT genre FROM serials*

Пример запроса с получением определенного числа значений, отсортированных по максимальной стоимости:

*SELECT \* FROM purchases ORDER BY total\_price DESC LIMIT @AmountOfPurchases*

В данном случае запрос вернет число строк, равное AmountOfPurchases, с наибольшей стоимостью.

**2.4 Программная реализация и документирование базы данных**

В качестве логической модели базы данных была выбрана реляционная модель. Стандартный язык запросов SQL позволяет из таблиц строить производные таблицы, отвечающие на простые и сложные запросы.

В качестве языка программирования был выбран C# в связке с кросс-платформенным фреймворком ASP.NET Core, который очень широко используется для построения современных веб-приложений. Данный стек инструментов обеспечивает гибкое и простое взаимодействие с базой данных.

Для написания клиентского приложения будут использованы HTML, CSS, JavaScript и широко используемой библиотекой React. Использование данного приложения обычным пользователем должно соответствовать таким требованиям, как безопасность, отказоустойчивость, надежность, функциональность, кроссплатформенность. Данное приложение необходимо пользователю для удобного поиска, просмотра сериалов с использованием понятного для него интерфейса.

Для разработки приложения с использованием данных технологий отлично подойдет IDE Visual Studio 2022.

Как упоминалось ранее, база данных содержит 5 таблиц:

− рейтинги;

− сериалы;

− пользователи;

− подписки;

− покупки.

**2.5 Проектирование разрабатываемого приложения**

Серверная часть приложения имеет трёхуровневую архитектуру и состоит из следующих уровней:

* доступа к данным (Data Access Layer);
* бизнес-логики (Business Logic Layer);
* представления (Presentation Layer).

Отдельное физическое расположение этих уровней – это то, что делает приложение более безопасным, гибким, масштабируемым и простым в управлении. Каждый слой ничего не знает и выше лежащих слоях, а выше лежащие слои имеют доступ лишь к предыдщуему.

На первом уровне данные таблицы представлены в виде классов, содержащие поля с теми же типами данных, что и в самой базе данных. Такие классы обычно называют Entity. Также, созданы классы, которые отражают возвращаемые данные из различных запросов, поскольку не всегда необходимо получать данные, которые обладают тем же набором полей, что и хранимая сущность. В коде данного приложения подобные классы носят название EntityView.

На втором уровне выполняется бизнес-логика приложения, например, рассчитывается суммарная стоимость покупки в зависимости от типа подписки и количества месяцев, на которое она была приобретена. Также, если пользователь захочет удалить строку, которая не существует в базе данных, на этом уровне осуществится проверка и будет возвращена ошибка. На данном уровне созданы классы, которые называются моделями, являющиеся результатом сопоставления классов между верхним и нижним слоем.

Финальный уровень представляет собой endpoints, с помощью которых по протоколу HTTP можно получить необходимые данные в формате JSON. Также здесь созданы классы, отражающие поля, которые увидит пользователь или которые будет отправлять пользователь, чтобы сделать какой-либо запрос.

Такие классы называются моделями представления.

Таким образом, серверная часть приложения представляет собой API, посредством которого через клиентское приложение можно делать HTTP-запросы для взаимодействия с данными.