Частное учреждение профессионального образования
«Высшая школа предпринимательства»
(ЧУПО «ВШП»)

КУРСОВОЙ ПРОЕКТ

«Разработка базы данных для библиотеки»

Выполнил:
студент 3-го курса специальности
09.02.07 «Информационные системы и программирование»
Ильина Ксения Игоревна
подпись: _____

Проверил:
преподаватель дисциплины,
преподаватель ЧУПО «ВШП»,
к.ф.н. Ткачев П.С.
оценка: _____
подпись: _____

Тверь, 2025 г.

Оглавление

Введение

Актуальность

В эпоху цифровой трансформации и информационного бума библиотеки остаются важными центрами знаний, культуры и образования. Однако далеко не все пользователи имеют постоянный доступ к обширным архивам печатных и электронных ресурсов, возможность их приобретения или требуемую техническую инфраструктуру для работы с цифровыми коллекциями. Поэтому библиотеки, предоставляя организованный доступ к этим ресурсам, продолжают играть незаменимую роль в обществе.

Разработка базы данных (БД) для библиотеки является актуальной и практически значимой задачей по следующим причинам:

1. Необходимость автоматизации рутинных процессов: Традиционные методы учета крайне трудоемки, подвержены человеческим ошибкам и неэффективны при работе с большими объемами данных. Автоматизация учета фондов, читателей, выдачи и возврата литературы – насущная потребность.
2. Обеспечение оперативного доступа к информации: Пользователи и сотрудники нуждаются в быстром и точном поиске информации о наличии книг, их местоположении, статусе, данных о читателях. Ручные системы не могут обеспечить необходимую скорость поиска.
3. Управление сложными и растущими архивами: Современные библиотечные архивы включают не только печатные книги, но и электронные издания, аудиовизуальные материалы. Эффективное управление, каталогизация, инвентаризация и списание такого разнородного контента требуют мощных инструментов.
4. Аналитика и отчетность: Администрации библиотеки необходимы данные для анализа востребованности архивов, планирования комплектования, оценки эффективности работы, формирования статистической отчетности. БД является источником этих данных.

Определение цели работы

Цель работы – разработать реляционную базу данных, моделирующую ключевые бизнес-процессы условной библиотеки и обеспечивающую эффективное управление ее ресурсами и операциями.

Краткая эволюция библиотечных систем (Контраст "Было/Стало")

Было:

- Поиск книги требовал ручного просмотра картотек, что занимало много времени.
- Учет выдачи велся в бумажных журналах или на книжных формулярах, контроль сроков возврата был затруднен.
- Инвентаризация фонда была громоздкой и редкой процедурой.
- Получение статистики требовало ручной обработки данных.
- Невозможность удаленного доступа пользователей к информации о фондах.

Стало:

- Мгновенный поиск: Электронный каталог (ЭК) позволяет находить издания по множеству критериев (автор, название, ключевое слово, ISBN, тема и т.д.) за секунды.
- Автоматизация операций: Автоматизированный учет выдачи/возврата, контроль сроков, расчет штрафов, печать напоминаний.
- Онлайн-сервисы: Удаленный доступ пользователей к каталогу (OPAC), бронирование, продление срока, просмотр своей истории.

Разница между "было" и "стало" принципиальна и демонстрирует критическую важность автоматизации на основе БД.

Постановка задач

1. Выбор СУБД.
2. Определить какие процессы происходят в библиотеке.
3. Сформулировать требования к разрабатываемой базе данных.
4. Построение ER-Диаграммы.

5. Разработать набор таблиц базы данных.
6. Заполнить таблицы данными для тестирования.
7. Настроить работу базы данных для выполнения бизнес-процессов с помощью СУБД.
8. Провести тестирование реализованных процессов.

Объект исследования

Объектом исследования является процесс проектирования и разработки реляционной базы данных, предназначенной для автоматизации основных бизнес-процессов условной библиотеки.

Метод исследования

Моделирование библиотечных процессов и поведения пользователей.

Первая глава

1.1 Выбор СУБД

Типы баз данных

1. Реляционные базы данных(SQL)

База, где данные хранятся в формате таблиц, они строго структурированы и связаны друг с другом. В таблице есть строки и столбцы, каждая строка представляет отдельную запись, а столбец — поле с назначенным ей типом данных. В каждой ячейке информация записана по шаблону.

Пример: базы данных MySQL, PostgreSQL и Oracle.

Лучше всего подходит для структурированных данных, таких как финансовые системы или управление запасами.

2. Нереляционные базы данных(NoSQL)

Хранит данные без четких связей друг с другом и четкой структуры. Вместо структурированных таблиц внутри базы находится множество разнородных документов, в том числе изображения, видео и даже публикации в социальных сетях. В отличие от реляционных БД, NoSQL базы данных не поддерживают запросы SQL.

Пример: MongoDB, Cassandra и DynamoDB.

Идеально подходит для приложений, требующих высокой масштабируемости и гибкости.(https://cloud.vk.com/blog/sravnenie-sql-i-nosql-kak-vybrat-sistemu-hraneniya-dannyh/)

Выбор подходящей базы данных зависит от потребностей приложения. Вот несколько ключевых факторов, которые следует учитывать при принятии этого решения:

Тип данных

Базы данных SQL идеально подходят для хранения и обработки структурированных данных, а базы данных NoSQL — это наилучшее решение для работы с неструктурированными и полуструктурированными данными. Если вы будете управлять и структурированными, и неструктурированными данными,

сделайте выбор в пользу объединения SQL и NoSQL. (https://habr.com/ru/companies/otus/articles/562852/)

Масштабируемость

По мере развития вашего веб-продукта должна расти и его база данных. На выбор базы данных может повлиять предпочитаемый вами тип масштабирования: горизонтальное или вертикальное. Нереляционные базы данных с их хранилищами типа «ключ-значение» оптимизированы для горизонтального масштабирования, а реляционные базы данных — для вертикального.

Безопасность

База данных должна быть хорошо защищена, поскольку в ней хранятся все данные пользователей. Реляционные базы данных, поддерживающие принципы ACID, более безопасны, чем нереляционные, которые делают упор на производительность и масштабируемость в ущерб согласованности и безопасности. Поэтому вам необходимо принять дополнительные меры для защиты своей базы данных NoSQL.

Интеграция

Важное замечание по выбору СУБД: убедитесь в том, что вашу систему управления базой данных можно интегрировать с другими инструментами и сервисами, входящими в ваш проект. Неспособность интегрироваться с другими решениями в большинстве случаев может привести к остановке проекта.

Для практической реализации базы данных условной библиотеки в рамках данной курсовой работы была выбрана система управления базами данных (СУБД) MySQL версии 8.0 в ее бесплатной Community Edition (CE).

MySQL Workbench CE — это инструмент для проектирования и управления базами данных, разработанный корпорацией Oracle, который позволяет пользователям визуально создавать, управлять и администрировать базы данных MySQL. Инструмент предлагает интуитивно понятный графический пользовательский интерфейс и поддерживает широкий спектр задач разработки баз данных, включая проектирование схем, разработку SQL и администрирование баз данных.

Это средство обеспечивает всестороннее моделирование баз данных с такими функциями, как реверс-инжиниринг для создания ER-диаграмм из существующих баз данных и прямой инжиниринг для создания сценариев SQL из моделей.

Пользователи могут выполнять SQL-запросы, получать доступ к объектам базы данных и редактировать записи данных с помощью встроенного в инструмент редактора SQL.(https://mysql-workbench-ce.updatestar.com/ru)

Функции администрирования MySQL Workbench CE позволяют пользователям управлять экземплярами сервера, настраивать пользователей и привилегии, а также отслеживать показатели производительности сервера.
Инструмент также поддерживает визуальные инструменты для переноса данных, синхронизации схем, а также операций резервного копирования и восстановления. MySQL Workbench CE доступен на различных платформах, включая Windows, Linux и macOS. Это программное обеспечение с открытым исходным кодом, распространяемое под лицензией GNU General Public License (GPL) версии 2.

В целом, MySQL Workbench CE предлагает набор мощных инструментов для разработки и администрирования баз данных, которые хорошо подходят как для начинающих, так и для опытных пользователей. Интуитивно понятный графический пользовательский интерфейс упрощает навигацию и помогает оптимизировать общие задачи, связанные с управлением базами данных MySQL.

Данный выбор обусловлен следующими ключевыми факторами:

1. Соответствие требованиям проекта: MySQL 8.0 является зрелой, высокопроизводительной реляционной СУБД, что полностью соответствует выбранной для библиотечной системы реляционной модели данных. Она эффективно решает задачи хранения структурированной информации о книгах, читателях, выдачах и обеспечивает необходимую целостность данных посредством поддержки транзакций (ACID-совместимость), внешних ключей (FOREIGN KEY) и других ограничений.
2. Открытость и доступность: Решение распространяется по лицензии GPL (GNU General Public License), что делает его абсолютно бесплатным для

использования в учебных и многих коммерческих целях. Это критически важно для студенческого проекта, исключая затраты на лицензирование.

3. Распространенность и поддержка: MySQL — одна из самых популярных СУБД в мире с огромным сообществом пользователей и разработчиков. Это гарантирует обилие учебных материалов, документации, форумов поддержки и примеров кода (SQL, хранимых процедур), что существенно упрощает процесс обучения и решения возникающих задач.
4. Функциональность MySQL 8.0 CE: Данная версия предоставляет все необходимые для реализации проекта функции.

Таким образом, связка MySQL Server 8.0 Community Edition и MySQL Workbench 8.0 CE представляет собой оптимальное, бесплатное, функционально полное и широко поддерживаемое решение, идеально подходящее для достижения цели данной курсовой работы – разработки базы данных для библиотеки. Этот выбор позволяет сосредоточиться на сути проектирования и реализации библиотечной БД, используя современные и отлаженные инструменты.

1.2 Процессы, происходящие в библиотеке

Актуальность данной темы заключается в том, что многие библиотеки до сих пор придерживаются традиционных библиотечных систем и процессов. Необходим переход от бумажной коммуникации к коммуникации безбумажной, чтоб сократить время поиска необходимой литературы, информации о читателе и т.д.

Процессы, которые требуется реализовать в данной работе:

1. Процесс записи новых читателей:

Проблема: Заполнение бумажных анкет и читательских билетов вручную отнимает много времени у сотрудника и посетителя. Хранение и поиск карточек читателей громоздки. Риск потери или повреждения карточек. Сложность ведения актуальной истории посещений или изменений данных читателя.

Решение БД: Электронная регистрация с мгновенным занесением данных в систему. Быстрый поиск читателя по ФИО, номеру билета. Легкое обновление информации. Формирование единой цифровой истории взаимодействия с читателем.

2. Процесс записи (каталогизации и учета) новых книг:

Проблема: Ручное заполнение инвентарных книг и карточек каталога для каждой книги крайне трудоемко и подвержено ошибкам (опечатки в названии, авторе, ISBN). Дублирование усилий (одна книга описывается для инвентарной книги, алфавитного каталога, систематического каталога). Сложность поиска места для новой книги на полке без электронного плана расстановки. Риск потери данных при повреждении бумажных носителей.

Решение БД: Единовременный ввод полного библиографического описания (с возможностью импорта из внешних источников по ISBN). Автоматическое присвоение id. Мгновенное отражение книги во всех разделах электронного каталога (ЭК). Надежное цифровое хранение информации. Поддержка стандартов (MARC, RDA).

3. Процесс выдачи книг:

Проблема: Ручное заполнение книги и читательского билета (дата выдачи, подпись). Необходимость отдельно проверять лимит выдачи и наличие задолженностей у читателя в бумажных журналах. Риск ошибок при записи даты возврата. Сложность контроля за актуальным статусом книги (выдана/в наличии) – книга может быть "потеряна" в каталоге, пока не вернется. Трудоемкий ручной расчет штрафов при просрочке.

Решение БД: Мгновенная проверка лимита читателя и задолженностей.

4. Процесс приема (возврата) книг:

Проблема: Ручная отметка о возврате в формуляре книги и читательском билете. Необходимость вручную проверять сроки возврата по журналу выдачи для расчета штрафов. Риск ошибки или пропуска штрафа. Сложность оперативного обновления статуса книги в каталоге (она снова доступна только после ручной отметки). Проблема возврата книг не тем сотрудником или в отсутствие ответственного.

Решение БД: Сканирование штрих-кода возвращаемой книги. Автоматическое снятие записи о выдаче. Фиксация факта возврата и сотрудника, его оформившего.

1.3 Формулировка требований к разрабатываемой базе данных.

Краткое описание требований:

- многократное использование данных, различным образом – для анализа, формирования отчетов, переноса в другие БД;
- простота пользования. Оператор должен, открыв БД, мгновенно понимать, какие данные доступны, как их структурировать, менять, обновлять;
- гибкость использования. Требование обязывает реализовывать различные механизмы доступа;
- быстрая обработка запросов. Использование защитных и аналитических механизмов работы с данными не должно перегружать ресурсы системы. При этом используемый язык программирования должен создавать возможность генерации нетривиальных запросов и отчетов;
- быстрая установка обновлений;
- возможность распределенной обработки данных;
- адаптивность и масштабируемость;
- механизм контроля за целостностью данных;
- возможность полноценного восстановления данных после сбоев;
- автоматическая реорганизация или перемещение данных.

(https://searchinform.ru/services/outsource-ib/zaschita-informatsii/bezopasnost-baz-dannykh/zaschita-baz-dannykh/tekhnologiya-razrabotki-i-zaschity-baz-dannykh/)

Требования к разработке баз данных (БД) предъявляют:

1. Функциональные требования
2. Технические требования

Эти требования определяют, какие функции должна выполнять БД, какие технические особенности она должна поддерживать и какие меры по обеспечению безопасности необходимо принять. Разберемся подробнее что из себя представляет каждый вид требований.

– Функциональные требования

Поддержка обработки данных. Например, использование подходящего языка запросов (SQL или другого), наличие дополнительных возможностей (агрегатные функции, подзапросы, индексирование).

Масштабируемость. Возможность увеличивать объём данных и количество пользователей без существенного снижения производительности и доступности системы.

Гибкость. Возможность адаптации под изменяющиеся бизнес-требования, поддержка различных типов данных и сценариев использования.

– Технические требования

Выбор модели данных. Например, реляционные базы данных подходят для структурированных данных, а нереляционные (NoSQL) — для неструктурированных.

Оптимизация структуры. Использование индексов для ускорения поиска данных, нормализация таблиц для устранения избыточности.

Настройка инфраструктуры. Установка и настройка серверов, настройка параметров конфигурации (размер кэша, управление подключениями).

В следствии чего мы можем выдвинуть требования к базе данной которую разрабатываем:

1. Реляционная модель данных:

База данных должна быть реализована на основе реляционной модели данных. Данные должны быть организованы в виде набора связанных таблиц (отношений), где строки представляют экземпляры сущностей (книги, читатели, выдачи и т.д.), а столбцы – их атрибуты (название, автор, ФИО, дата выдачи и т.д.).

Обоснование: Реляционная модель обеспечивает четкую структуризацию данных, гарантирует целостность информации через механизмы ключей и ограничений, предоставляет мощный и стандартизированный язык запросов (SQL), что критически важно для эффективного управления библиотечными фондами,

читателями и операциями. Она доказала свою надежность и является оптимальным выбором для данной предметной области.

2. Автоматизация ключевых библиотечных процессов:

База данных должна содержать необходимые таблицы и структуры для полной автоматизации основных бизнес-процессов библиотеки

Обоснование: Цель БД – заменить неэффективные бумажные системы, обеспечив цифровую основу для всех основных операций библиотеки, повысив их скорость, точность и управляемость.

3. Целостность данных и связи между сущностями:

Таблицы базы данных должны быть логически связаны между собой через механизмы первичных (PK) и внешних ключей (FK). Это необходимо для:

Обеспечения ссылочной целостности данных (например, нельзя удалить читателя, у которого есть активные выдачи).

Корректного отражения бизнес-логики библиотеки (книга принадлежит изданию, выдача связывает читателя и экземпляр, возврат закрывает выдачу).

Эффективного выполнения сложных запросов, затрагивающих данные из нескольких сущностей (например, поиск всех книг определенного автора, выданных на руки конкретному читателю).

Обоснование: Связи между таблицами являются фундаментом для корректного функционирования БД и точного отражения реальных библиотечных процессов и отношений между объектами (книга-читатель-выдача).

4. Производительность и масштабируемость:

База данных должна обеспечивать стабильную и отзывчивую работу при одновременном обращении множества пользователей (как сотрудников, так и читателей через OPAC). Архитектура и выбранная СУБД должны поддерживать обработку асинхронных запросов без значительного снижения производительности.

База данных должна быть спроектирована с учетом возможности роста: увеличения объема фондов, числа читателей и транзакций. Структура БД должна допускать относительно легкую модификацию и расширение.

Обоснование: Современная библиотека требует постоянного доступа к данным. Онлайн-каталог (OPAC) доступен 24/7, сотрудники работают параллельно. БД должна справляться с пиковыми нагрузками (начало семестра, акции) и расти вместе с библиотекой. Использование стандартных технологий (SQL, реляционная модель) способствует универсальности и упрощает потенциальную миграцию или интеграцию.

После установки требований и выбора СУБД можно приступать к разработке базы данных.

Глава вторая

2.1 Построение ER-Диаграммы

ER-модель (от англ. Entity-Relationship model, модель «сущность — связь») — модель данных, позволяющая описывать концептуальные схемы предметной области.

Во время проектирования баз данных происходит преобразование схемы, созданной на основе ER-модели, в конкретную схему базы данных на основе выбранной модели данных (реляционной, объектной, сетевой или др.).

ER-модель представляет собой формальную конструкцию, которая сама по себе не предписывает никаких графических средств её визуализации. В качестве стандартной графической нотации, с помощью которой можно визуализировать ER-модель, была предложена диаграмма «сущность-связь» (англ. Entity-Relationship diagram, ERD, ER-диаграмма).

Понятия «ER-модель» и «ER-диаграмма» часто не различают, хотя для визуализации ER-моделей могут быть использованы и другие графические нотации, либо визуализация может вообще не применяться (например, использоваться текстовое описание).

ER-диаграммы используются для:

Проектирование структуры базы данных: Это основная цель. ERD служит "чертежом" для создания физической реляционной (или другой) базы данных.

Помогает определить таблицы (сущности), их столбцы (атрибуты), первичные и внешние ключи (связи) и ограничения (степень связи, обязательность).

Визуализация и Понимание данных: Позволяет быстро и наглядно понять, какие данные хранятся в системе, как они связаны между собой и какие существуют бизнес-правила (кардинальность, обязательность). Незаменима для новых членов команды.

Коммуникация и Согласование: Обеспечивает общий язык между различными заинтересованными сторонами: бизнес-аналитиками, разработчиками, архитекторами, конечными пользователями. Помогает выявить недопонимание требований на ранней стадии.

Документирование системы: ERD является ключевой частью технической документации системы. Описывает логическую модель данных независимо от конкретной СУБД.

Анализ и Оптимизация существующих баз данных: Помогает понять структуру унаследованной или плохо документированной БД. Выявить избыточность данных, проблемы нормализации, неэффективные связи, потенциальные точки отказа.

Устранение неполадок (Troubleshooting): При возникновении проблем с данными (дублирование, потеря целостности, некорректные отчеты) ERD помогает быстро понять, где в структуре могла возникнуть ошибка или где нарушено бизнес-правило.

Перепроектирование (Refactoring) и Модернизация систем: При изменении бизнес-требований ERD позволяет оценить влияние изменений на структуру данных, спланировать миграцию данных и минимизировать риски.

Проверка целостности данных: Четкое определение первичных/внешних ключей, обязательности и кардинальности позволяет СУБД автоматически обеспечивать ссылочную целостность (Referential Integrity), предотвращая появление "висячих" ссылок и некорректных данных.

Нормализация базы данных: Процесс приведения структуры БД к формам нормализации (1NF, 2NF, 3NF, BCNF и т.д.) для минимизации избыточности и аномалий обновления часто выполняется на основе ER-модели.

Основываясь на перечне необходимых возможностей из предыдущего раздела, подготовимся к построению ER-диаграммы, сформулировав состав из 5 основных таблиц будущей базы данных.

Компоненты ER модели

Сущности (Entities) - это центральные элементы ER модели. Каждая сущность представляет объект или понятие, значимое для системы. Например, в системе управления университетом основными сущностями могут быть студенты, курсы и преподаватели. Эти объекты обладают определёнными свойствами и взаимодействуют между собой.

Атрибуты (Attributes) - характеристики, описывающие сущности. Атрибуты детализируют сущности, добавляя им контекста и уникальности. Например, атрибуты сущности "Студент" могут включать имя, номер зачетки и дату рождения. Такой подход позволяет более точно моделировать данные и их особенности.

Связи (Relationships) определяют, как сущности взаимодействуют друг с другом. Проектирование связей помогает установить логические связи между объектами и создать более реалистичную модель данных. Например, студент записывается на курс, а преподаватель ведёт этот курс - эти взаимодействия отражаются через связи в ER модели.

Типы отношений - важный аспект связей. Они могут быть одного из трёх основных видов: один к одному (1:1), один ко многим (1:N) и многие ко многим (M:N). Понимание типов отношений способствует правильному моделированию данных, предотвращая дублирование и обеспечивая целостность базы.

Создавая ER модель, важно учитывать каждую из этих составляющих. Применение данных компонентов на практике помогает построить надежную и эффективную структуру базы данных, обеспечивая её стабильное

функционирование и удобство использования.(https://skyeng.ru/magazine/wiki/it-industriya/chto-takoe-er-diagramma/)

Таблица “Users”- данная таблица отражает сущность пользователей. Таблица должна содержать данные для ввода email и выбора роли.

Таблица “Borrowings”- данная таблица отражает сущность выдачи. Таблица должна содержать данные для ввода даты выдачи/возварата, id книги, id книги.

Таблица “Books”- данная таблица отражает сущность книг. Таблица должна содержать данные для названия и года публикации.

Таблица “Authors”- данная таблица отражает сущность авторов книг. Таблица должна содержать данные для ФИО.

Таблица “Categories”- данная таблица отражает сущность категорий книг. Таблица должна содержать данные для ввода названий категорий.

Исходя из этих данных создадим ER-диаграмму

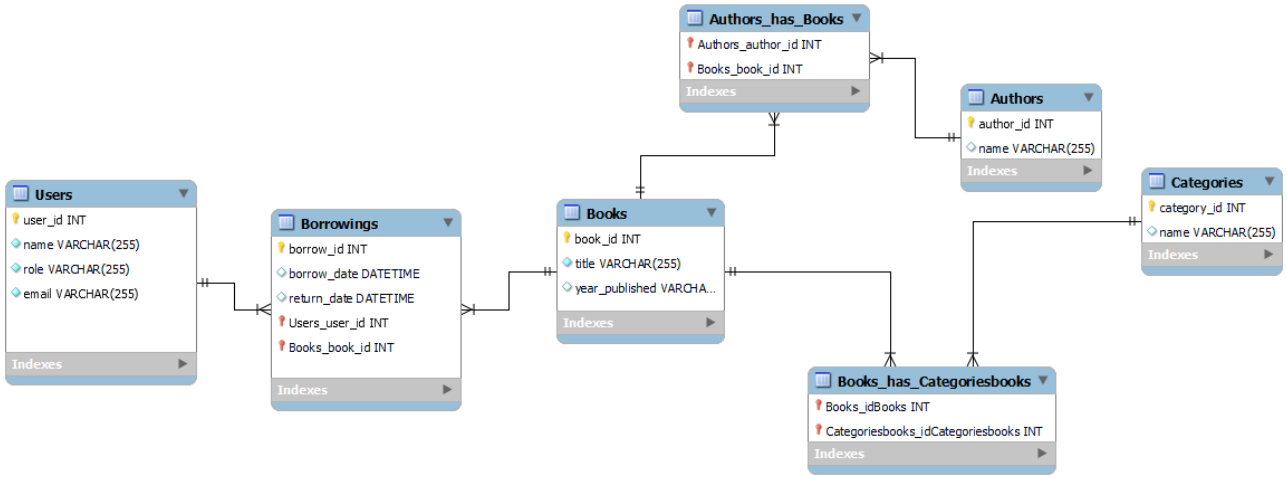

Рис.1

2.2 Разработка набора связанных таблиц базы данных

Описание таблиц

Таблица ‘users’

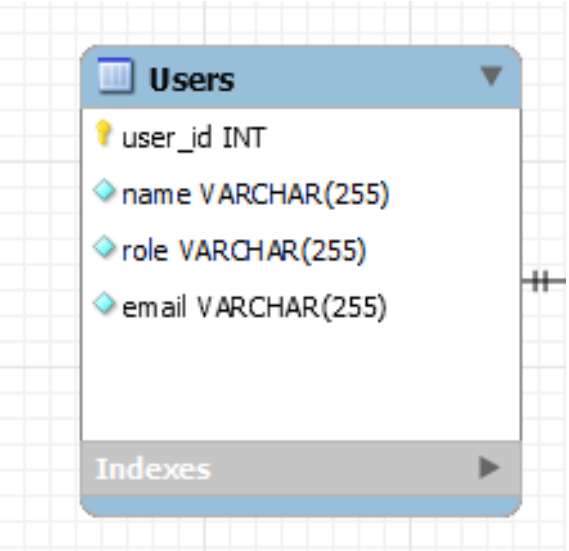

Рис.2

Таблица содержит поля: user_id, name, role, email.

1. Для поля user_id выбран тип данных INT , поскольку он является первичным ключом.
2. Для поля name выбран тип данных VARCHAR(255).
3. Для поля role выбран тип данных VARCHAR(255).
4. Для поля email выбран тип данных VARCHAR(255).

Так же все поля в данной таблице имеют значение NOT NULL что является обязательным полем для заполнения.

Часто встречающееся ограничение VARCHAR(255) для хранения текстовых данных в базах данных объясняется несколькими факторами. Во-первых, это исторически сложившаяся практика, уходящая корнями в ограничения старых систем управления базами данных (СУБД). Во-вторых, такой размер поля часто оказывался достаточным для многих распространённых применений, например, для хранения имён, адресов или коротких описаний. Таким образом, VARCHAR(255) стал своего рода негласным стандартом, облегчающим разработку и миграции данных. (https://telegra.ph/Pochemu-varchar-255-Pochemu-VARCHAR255-Glubokoe-pogruzhenie-v-mir-hraneniya-strok-v-bazah-dannyh-12-02)

Таблица ‘Borrowings’

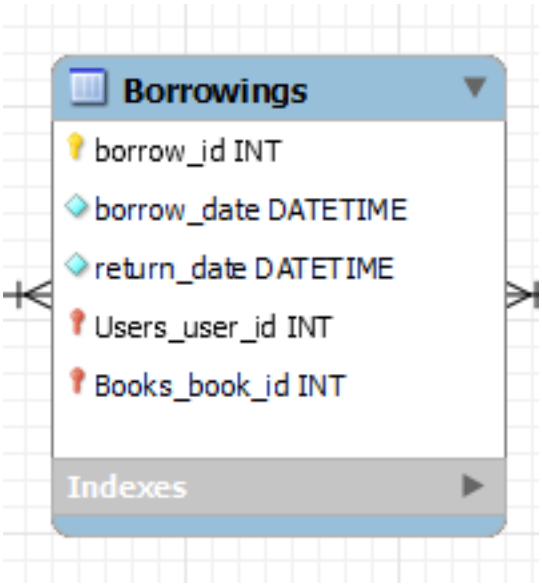

Рис.3

Таблица содержит поля: borrow_id, borrow_date, return_date.

1. borrow_id – по такой же логике как в таблице ‘users’.
2. Для поля borrow_date (дата выдачи) выбран тип данных DATETIME.
3. Для поля return_date (дата возврата) так же выбран тип данных DATETIME.

Тип данных DATETIME

используется для хранения информации о дате и времени в формате YYYY-MM-DD HH: MM: SS. Он позволяет представлять даты и время с высокой степенью точности и не зависит от часового пояса при изменениях.

Характеристики

1. Формат: ГГГГ-ММ-ДД ЧЧ:ММ:СС

2. Диапазон: с 1 января 1000 года по 31 декабря 9999 года.

3. Размер хранилища: 8 байт.

4. Часовой пояс: не преобразует и не сохраняет информацию о часовом поясе. Сохраняет дату и время без учета настроек часового пояса сервера.

Преимущества

Фиксированное представление: поскольку DATETIME не зависит от часовых поясов, оно обеспечивает согласованное представление значений даты и времени независимо от того, где и когда осуществляется доступ к данным.

Длинный диапазон: поддерживает более широкий диапазон дат по сравнению с TIMESTAMP, что делает его подходящим для исторических или будущих дат.

Тип INT в MySQL

Это как "цифровая коробка" для хранения целых чисел (чисел без дробной части). Вот где и зачем его используют:

1. Для уникальных номеров записей

Чаще всего INT используют для колонки id — уникального номера каждой записи в таблице. Например, у каждого пользователя есть свой user_id (1, 2, 3...). MySQL может автоматически увеличивать этот номер при добавлении новых записей (AUTO_INCREMENT), чтобы номера никогда не повторялись.

2. Для подсчёта количества

Когда нужно хранить числа для расчётов:

Количество товаров на складе (quantity INT)

Число лайков под постом (likes INT)

Возраст человека (age INT)

3. Для ссылок между таблицами

Если у вас есть две связанные таблицы (например, Пользователи и Заказы), INT хранит ID из одной таблицы в другой. Например, в заказе будет поле user_id со значением 15 — это значит "этот заказ принадлежит пользователю с id=15".

4. Для простых чисел без дробей

INT подходит только для целых чисел. Если нужны дроби (например, цена 199.99), берут DECIMAL или FLOAT. Диапазон INT огромен: от -2 миллиардов до +2 миллиардов. Если нужно только положительные числа — используют INT UNSIGNED (от 0 до 4 миллиардов).

Таблица ‘Books’

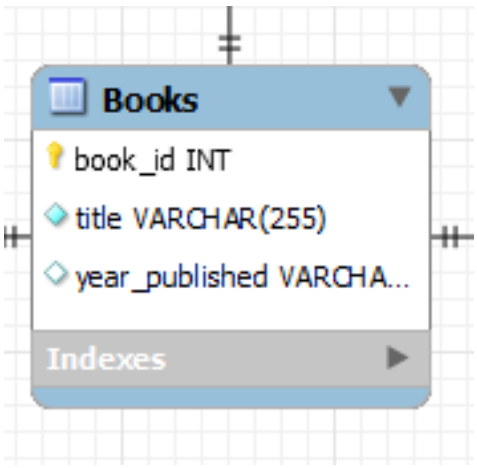

Рис.3

Таблица содержит поля: book_id, title, year_published.

1. book_id – так же имеет тип данных INT как в таблице ‘users’.
2. Для поля title выбран тип данных VARCHAR(255).
3. Для поля year_published так же выбран тип данных VARCHAR(255), но отсутствует значение NOT NULL.

Таблица ‘Authors’

Рис.4

Таблица содержит поля: auhor_id, name.

1. book_id – так же имеет тип данных INT как в таблице ‘users’.
2. Для поля name выбран тип данных VARCHAR(255).

Таблица ‘Categories’

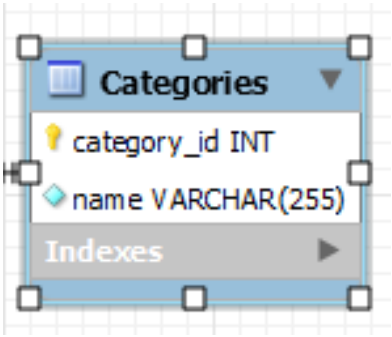

Рис.5

Таблица содержит поля: category_id, name.

1. category_id – так же имеет тип данных INT как в таблице ‘users’.
2. Для поля name выбран тип данных VARCHAR(255).

Создав ER-диаграмму можно конвертировать её, вот код для создания некоторых вышеуказанные таблицы.

Таблица ‘users’

```
CREATE TABLE IF NOT EXISTS `mydb`.`Users` (
  `user_id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NOT NULL,
  `role` VARCHAR(255) NOT NULL,
  `email` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`user_id`))
ENGINE = InnoDB;
```

Таблица ‘Books’

```
CREATE TABLE IF NOT EXISTS `mydb`.`Books` (
  `book_id` INT NOT NULL AUTO_INCREMENT,
  `title` VARCHAR(255) NOT NULL,
  `year_published` VARCHAR(255) NULL,
  PRIMARY KEY (`book_id`))
ENGINE = InnoDB;
```

Создание полной базы данных смотреть (Приложение 1)

Организация связей между таблицами в реляционных БД

Одно из ключевых преимуществ реляционной модели — возможность устанавливать связи между таблицами. Эти связи отражают логические отношения между разными типами сущностей и позволяют моделировать сложные предметные области.

В реляционных базах данных существует три основных типа связей:

Один к одному (1:1) — каждой записи в первой таблице соответствует максимум одна запись во второй таблице, и наоборот

Один ко многим (1:N) — каждой записи в первой таблице соответствует произвольное количество записей во второй таблице, но каждая запись во второй таблице связана только с одной записью в первой

Многие ко многим (M:N) — каждой записи в первой таблице соответствует произвольное количество записей во второй таблице, и наоборот

Связи реализуются через механизм внешних ключей. Внешний ключ — это атрибут или набор атрибутов одной таблицы, который ссылается на первичный ключ другой таблицы. Он обеспечивает ссылочную целостность данных — гарантию того, что ссылки между таблицами всегда корректны.

2.3 Заполнение таблицы данными.
Для таблицы ‘users’:

```
INSERT INTO Users (name, role, email) VALUES
('Иван Иванов', 'Читатель', 'ivanov@mail.com'),
('Мария Смирнова', 'Библиотекарь', 'smirnova@mail.com'),
('Алексей Козлов', 'Читатель', 'kozlov@mail.com'),
('Ольга Петрова', 'Читатель', 'petrova@mail.com'),
('Дмитрий Сидоров', 'Читатель', 'sidorov@mail.com'),
('Наталья Орлова', 'Читатель', 'orlova@mail.com'),
('Павел Михайлов', 'Читатель', 'mihailov@mail.com');
```

Полный код запроса смотреть (Приложение 2)

2.4 Бизнес-процессы

Типовые запросы

Запрос 1:

Данный запрос помогает понять какие книги относятся к каким категориям.

```
SELECT
    c.name AS category_name,
```

```
    b.title AS book_title
FROM Books_has_Categoriesbooks bhc
JOIN Categories c ON bhc.Categoriesbooks_idCategoriesbooks =
c.category_id
JOIN Books b ON bhc.Books_idBooks = b.book_id
ORDER BY c.name, b.title;
```

Запрос 2:

Данный запрос выведет список категорий и количество книг в каждой.

```
SELECT c.name AS category_name,
COUNT(bhc.Books_idBooks) AS book_count
FROM Categories c
LEFT JOIN Books_has_Categoriesbooks bhc ON c.category_id =
bhc.Categoriesbooks_idCategoriesbooks
GROUP BY c.category_id, c.name
ORDER BY book_count DESC;
```

Запрос 3:

Данный запрос выведет книги, которые сейчас находятся в пользовании.

```
SELECT b.title, u.name AS borrowed_by, br.borrow_date
FROM Borrowings br
JOIN Books b ON br.Books_book_id = b.book_id
JOIN Users u ON br.Users_user_id = u.user_id
WHERE br.return_date IS NULL;
```

Запрос 4:

Данный запрос выведет пользователей, бравших книги.

```
SELECT u.user_id, u.name, COUNT(br.borrow_id) AS borrow_count
FROM Users u
JOIN Borrowings br ON u.user_id = br.Users_user_id
GROUP BY u.user_id;
```

Запрос 5:

Данный запрос выведет список авторов и их книг.

```
SELECT a.name AS author_name, b.title AS book_title
FROM Authors_has_Books ab
JOIN Authors a ON ab.Authors_author_id = a.author_id
JOIN Books b ON ab.Books_book_id = b.book_id;
```

Представление

```
CREATE VIEW View_BookAuthors AS
SELECT b.title, a.name AS author
FROM Books b
JOIN Authors_has_Books ab ON b.book_id = ab.Books_book_id
```

```
JOIN Authors a ON ab.Authors_author_id = a.author_id;
```
Данное представление отображает имя автора и название его книги.

Заключение

В данной работе были достигнуты все цели и задачи:

MySQL был выбран как надежное, производительное и широко распространенное решение, идеально подходящее для задач управления библиотечными данными.

Были изучены и формализованы основные бизнес-процессы, происходящие в библиотеке (учет книг, читателей, выдача/прием, бронирование и т.д.).

Четко определены функциональные требования к базе данных.

Построена ER-диаграмма, наглядно отобразившая сущности предметной области (Книги, Читатели, Авторы, Жанры, Выдачи и др.) и связи между ними, что стало основой для дальнейшей разработки.

На основе ER-диаграммы успешно разработан и реализован в MySQL набор взаимосвязанных таблиц, обеспечивающих целостное хранение всей необходимой информации.

Реализованы необходимые SQL-запросы (поиск книг, регистрация выдачи/возврата, управление читательскими билетами, формирование отчетов), обеспечена целостность данных (ключи, ограничения) и базовые аспекты безопасности.

Таким образом, применение MySQL Workbench 8.0 CE было выбрано как оптимальное решение для реализации поставленной в курсовой работе цели. Данная бесплатная, функционально завершенная и широко распространенная платформа идеально подходит для разработки базы данных библиотеки, позволяя сфокусироваться на основных задачах проектирования и внедрения, используя актуальные и проверенные инструменты.

Список литературы

1. [Электронный ресурс] // MySQL Workbench CE : [сайт]. — URL: https://mysql-workbench-ce.updatestar.com/ru (дата обращения: 07.06.2025).
2. Курсовая работа / [Электронный ресурс] // : [сайт]. — URL: https://portal.bgsha.ru/upload/iblock/815/Mongush_PIS.pdf (дата обращения: 07.06.2025).
3. Технология разработки и защиты баз данных / [Электронный ресурс] // searchinform.ru : [сайт]. — URL: https://searchinform.ru/services/outsource-ib/zaschita-informatsii/bezopasnost-baz-dannykh/zaschita-baz-dannykh/tekhnologiya-razrabotki-i-zaschity-baz-dannykh/(дата обращения: 07.06.2025).
4. Основные требования, предъявляемые к базам данных / [Электронный ресурс] // studfile.net : [сайт]. — URL: https://studfile.net/preview/10714766/page:2/ (дата обращения: 07.06.2025).
5. Сравнение SQL и NoSQL: как выбрать систему хранения данных / [Электронный ресурс] // cloud.vk.com : [сайт]. — URL: https://cloud.vk.com/blog/sravnenie-sql-i-nosql-kak-vybrat-sistemu-hraneniya-dannyh/ (дата обращения: 07.06.2025).
6. ER-модель / [Электронный ресурс] // Википедия : [сайт]. — URL: https://ru.wikipedia.org/wiki/ER-модель (дата обращения: 07.06.2025).

7. / [Электронный ресурс] // : [сайт]. — URL: https://sky.pro/wiki/analytics/dannye-v-relyacionnoj-bd-predstavleny-v-vide-struktura-i-organizaciya/ (дата обращения: 07.06.2025).
8. / [Электронный ресурс] // : [сайт]. — URL: https://skyeng.ru/magazine/wiki/it-industriya/chto-takoe-er-diagramma/ (дата обращения: 07.06.2025).
9. Английский источник / [Электронный ресурс] // : [сайт]. — URL: https://www.geeksforgeeks.org (дата обращения: 07.06.2025).

Приложение 1

```
CREATE TABLE IF NOT EXISTS `mydb`.`Users` (
  `user_id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NOT NULL,
  `role` VARCHAR(255) NOT NULL,
  `email` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`user_id`))
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `mydb`.`Books` (
  `book_id` INT NOT NULL AUTO_INCREMENT,
  `title` VARCHAR(255) NOT NULL,
  `year_published` VARCHAR(255) NULL,
  PRIMARY KEY (`book_id`))
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `mydb`.`Authors` (
  `author_id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NULL,
  PRIMARY KEY (`author_id`))
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `mydb`.`Borrowings` (
  `borrow_id` INT NOT NULL AUTO_INCREMENT,
  `borrow_date` DATETIME NOT NULL,
  `return_date` DATETIME NOT NULL,
  `Users_user_id` INT NOT NULL,
  `Books_book_id` INT NOT NULL,
  PRIMARY KEY (`borrow_id`, `Users_user_id`, `Books_book_id`),
  INDEX `fk_Borrowings_Users1_idx` (`Users_user_id` ASC) VISIBLE,
  INDEX `fk_Borrowings_Books1_idx` (`Books_book_id` ASC) VISIBLE,
  CONSTRAINT `fk_Borrowings_Users1`
    FOREIGN KEY (`Users_user_id`)
    REFERENCES `mydb`.`Users` (`user_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Borrowings_Books1`
    FOREIGN KEY (`Books_book_id`)
    REFERENCES `mydb`.`Books` (`book_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `mydb`.`Categories` (
  `category_id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NULL,
  PRIMARY KEY (`category_id`))
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `mydb`.`Books_has_Categoriesbooks` (
  `Books_idBooks` INT NOT NULL,
  `Categoriesbooks_idCategoriesbooks` INT NOT NULL,
```

```
  PRIMARY KEY (`Books_idBooks`,
`Categoriesbooks_idCategoriesbooks`),
  INDEX `fk_Books_has_Categoriesbooks_Categoriesbooks1_idx`
(`Categoriesbooks_idCategoriesbooks` ASC) VISIBLE,
  INDEX `fk_Books_has_Categoriesbooks_Books1_idx` (`Books_idBooks`
ASC) VISIBLE,
  CONSTRAINT `fk_Books_has_Categoriesbooks_Books1`
    FOREIGN KEY (`Books_idBooks`)
    REFERENCES `mydb`.`Books` (`book_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Books_has_Categoriesbooks_Categoriesbooks1`
    FOREIGN KEY (`Categoriesbooks_idCategoriesbooks`)
    REFERENCES `mydb`.`` (`category_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `mydb`.`Authors_has_Books` (
  `Authors_author_id` INT NOT NULL,
  `Books_book_id` INT NOT NULL,
  PRIMARY KEY (`Authors_author_id`, `Books_book_id`),
  INDEX `fk_Authors_has_Books_Books1_idx` (`Books_book_id` ASC)
VISIBLE,
  INDEX `fk_Authors_has_Books_Authors1_idx` (`Authors_author_id`
ASC) VISIBLE,
  CONSTRAINT `fk_Authors_has_Books_Authors1`
    FOREIGN KEY (`Authors_author_id`)
    REFERENCES `mydb`.`Authors` (`author_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Authors_has_Books_Books1`
    FOREIGN KEY (`Books_book_id`)
    REFERENCES `mydb`.`Books` (`book_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Приложение 2

```
INSERT INTO Users (name, role, email) VALUES
('Иван Иванов', 'Читатель', 'ivanov@mail.com'),
('Мария Смирнова', 'Библиотекарь', 'smirnova@mail.com'),
('Алексей Козлов', 'Читатель', 'kozlov@mail.com'),
('Ольга Петрова', 'Читатель', 'petrova@mail.com'),
('Дмитрий Сидоров', 'Читатель', 'sidorov@mail.com'),
('Наталья Орлова', 'Читатель', 'orlova@mail.com'),
('Павел Михайлов', 'Читатель', 'mihailov@mail.com');

-- Данные для Authors
INSERT INTO Authors (name) VALUES
('Лев Толстой'),
('Фёдор Достоевский'),
('Александр Пушкин'),
('Михаил Булгаков'),
('Илья Ильф'),
('Евгений Петров'),
('Антон Чехов'),
('Николай Гоголь'),
('Иван Тургенев');

-- Данные для Books
INSERT INTO Books (title, year_published) VALUES
('Война и мир', '1869'),
('Преступление и наказание', '1866'),
('Евгений Онегин', '1833'),
('Мастер и Маргарита', '1967'),
('Двенадцать стульев', '1928'),
('Чайка', '1896'),
('Мёртвые души', '1842'),
('Отцы и дети', '1862'),
('Анна Каренина', '1877'),
('Идиот', '1869'),
('Бесы', '1872'),
('Палата №6', '1892');

-- Связи Authors_has_Books
INSERT INTO Authors_has_Books (Authors_author_id, Books_book_id)
VALUES
(1, 1), -- Толстой — Война и мир
(1, 9), -- Толстой — Анна Каренина
(2, 2), -- Достоевский — Преступление и наказание
(2, 10), -- Достоевский — Идиот
(2, 11), -- Достоевский — Бесы
(3, 3), -- Пушкин — Евгений Онегин
(4, 4), -- Булгаков — Мастер и Маргарита
(5, 5), -- Ильф — Двенадцать стульев
(6, 5), -- Петров — Двенадцать стульев
(7, 6), -- Чехов — Чайка
(7, 12), -- Чехов — Палата №6
(8, 7), -- Гоголь — Мёртвые души
```

```
(9, 8); -- Тургенев — Отцы и дети

-- Категории
INSERT INTO Categories (name) VALUES
('Классика'),
('Философский роман'),
('Юмор'),
('Психология'),
('Советская литература');

-- Связи Books_has_Categoriesbooks
INSERT INTO Books_has_Categoriesbooks (Books_idBooks,
Categoriesbooks_idCategoriesbooks) VALUES
(1, 1), (2, 1), (3, 1), (4, 2), (5, 3),
(6, 1), (7, 1), (8, 1), (9, 2), (10, 2),
(11, 2), (12, 4), (5, 5);
-- Данные для Borrowings
INSERT INTO Borrowings (borrow_date, return_date, Users_user_id,
Books_book_id) VALUES
('2025-05-01 10:00:00', '2025-05-05 14:00:00', 1, 1),
('2025-05-02 12:30:00', NULL, 1, 2),
('2025-05-03 15:00:00', NULL, 3, 3),
('2025-05-04 09:00:00', NULL, 4, 4),
('2025-05-05 11:15:00', NULL, 5, 5),
('2025-05-06 13:45:00', NULL, 6, 6),
('2025-05-07 08:10:00', NULL, 7, 7),
('2025-05-08 09:30:00', NULL, 1, 8),
('2025-05-09 14:00:00', NULL, 2, 9),
('2025-05-10 16:20:00', NULL, 2, 10),
('2025-05-11 17:40:00', NULL, 3, 11),
('2025-05-12 18:00:00', NULL, 4, 12);
```

Приложение 3
QR код на репозиторий

Отчет о проверке № 9480650

Дата выгрузки: 2025-06-08 15:49:24
Пользователь: hioka.333@gmail.com, ID: 9480650

Отчет предоставлен сервисом «Антиплагиат»
на сайте antiplagius.ru/

Информация о документе

№ документа: 9480650
Имя исходного файла: Курсовая по бд (5).docx
Размер файла: 0.14 МБ
Размер текста: 36879
Слов в тексте: 5171
Число предложений: 501

Информация об отчете

Дата: 2025-06-08 15:49:24 - Последний готовый отчет
Оценка оригинальности: 100%
Заимствования: 0%

100.00% 0%

Источники:

Доля в тексте	Ссылка

Информация о документе: