

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления (ИУ)

КАФЕДРА «Информационная безопасность» (ИУ8)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОЙ РАБОТЕ
НА ТЕМУ:

***Разработка защищенной БД информационной
системы «Книжный магазин»***

Студент

ИУ8-81

(группа)

(подпись, дата)

М. Е. Шаповалов

(И.О. Фамилия)

Руководитель курсовой работы

(подпись, дата)

Е. В. Глинская

(И.О. Фамилия)

**Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой

ИУ8

(индекс)

М. А. Басараб

(И.О. Фамилия)

(подпись)

(дата)

**З А Д А Н И Е
на выполнение курсовой работы**

по дисциплине Безопасность систем баз данных

Студент группы ИУ8-81

Шаповалов Максим Евгеньевич

(Фамилия, имя, отчество)

Тема курсовой работы

Разработка защищенной БД информационной системы «Книжный магазин».

Направленность КР (учебная, исследовательская, практическая, производственная, др.)
учебно-практическая

Источник тематики (кафедра, предприятие,
НИР)

кафедра

Задание

Проанализировать существующую проблему информационной безопасности и защиты информации в базах данных информационной системы «Книжный магазин».

Представить решение проблемы информационной безопасности и защиты информации в базах данных информационной системы «Книжный магазин».

Оформление курсовой работы:

Расчетно-пояснительная записка (Отчет по КР) на 30-35 листах формата А4.

Презентация

Флешка

Дата выдачи задания « ____ » _____ 20__ г.

Руководитель курсовой работы

(подпись, дата)

Е. В. Глинская

(И.О. Фамилия)

Студент

(подпись, дата)

М. Е. Шаповалов

(И.О. Фамилия)

Аннотация

В курсовой работе выполняется разработка защищенной БД информационной БД «Книжный магазин».

Задачами курсовой работы являются:

1. анализ предметной области;
2. разработка базы данных;
3. защита базы данных;
4. разработка приложения для взаимодействия с базой данных;

Содержание

Введение.....	5
Анализ предметной области.....	6
Проектирование базы данных.....	7
1 Определение сущностей.....	7
2 Создание запросов.....	8
2.1 Запросы к таблице книг.....	8
2.2 Запросы к таблице клиентов.....	10
2.3 Запросы к таблице заказов.....	12
2.4 Запросы к таблице тегов.....	13
2.5 Запросы к таблице, связывающей книги и теги.....	13
2.6 Запросы к таблице отзывов.....	14
2.7 Запросы для работы с пользователями.....	15
Защита базы данных.....	16
1 Обеспечение конфиденциальности.....	16
2 Обеспечение целостности.....	22
3 Обеспечение доступности.....	24
Разработка приложения.....	26
Заключение.....	31
Список использованных источников.....	32

Введение

Одной из основных задач всех IT-проектов является оперативный доступ к данным, для чего используются базы данных. В настоящее время любая компания или предприятие используют электронные базы данных для более эффективного ведения своего бизнеса. Основным достоинством является серьезное увеличение скорости работы, из этого следует приобретение организацией принципиально новых качеств, дающих существенные конкурентные преимущества. Основным критерием конкурентоспособности является стабильная работа и бесперебойный доступ к любым ресурсам предприятия. Именно поэтому особое внимание уделяется безопасности, при обеспечении которой определяется уровень защиты системы, а также аппаратные и программные средства, служащие для обеспечения конфиденциальности целостности и доступности информации.

Целью данной курсовой работы является проектирование и разработка защищенной базы данных для информационной системы «Книжный магазин», а также разработка приложения для взаимодействия с базой данных.

Анализ предметной области

Книжный магазин — место, в котором *клиенты* магазина могут осуществлять поиск *книг* по названиям, авторам, дате выпуска, *тегам*; могут оформлять *заказы* на книги и оставлять на них *отзывы*.

В книжном магазине принята иерархия сотрудников:

1. Сотрудник ведет прямую работу с клиентами и выполняет основную работу магазина. Сотрудник имеет право добавлять и изменять список книг, клиентов и тегов, а также добавлять отзывы к книгам от имени клиента и оформлять заказы, но не способен ничего удалять;
2. Администратор контролирует работу обычных сотрудников, а также привлекается к решению неординарных ситуаций с клиентами. Администратор расширяет права сотрудника: он может изменять отзывы, а также удалять книги, клиентов, теги, отзывы;
3. Администратор безопасности расширяет права администратора: он может создавать новых сотрудников и удалять старых.

Из описания иерархии следует, что заказы нельзя изменять и удалять — это сделано для того, чтобы не терялась статистика продаж.

Проектирование базы данных

Разработку базы данных будем вести на MS SQL EXPRESS.

1 Определение сущностей

На основе анализа предметной области составим схему базы данных.

Определим сущности, их атрибуты и связи:

- Книга. Имеет уникальный идентификатор, заглавие, автора, дату выпуска, количество доступных для продажи книг и цену;
- Тег. Имеет уникальный идентификатор и название. Каждая книга может иметь множество тегов, также каждому тегу может соответствовать множество книг (many-to-many);
- Клиент. Имеет уникальный идентификатор, имя, номер телефона, пол;
- Отзыв. Имеет уникальный идентификатор, которым является связка уникальных идентификаторов книги и клиента, с которыми этот отзыв связан, оценку и текст отзыва;
- Заказ имеет уникальный идентификатор, дату создания, уникальный идентификатор клиента, который оформил заказ, и список книг, которые этому заказу соответствуют. Для списка книг определим отдельную сущность:
 - ЗаказаннаяКнига. Имеет уникальный идентификатор, которым является связка уникальных идентификаторов книги и заказа, с которыми эта заказанная книга связана, количество книг в заказе и цена книги на момент заказа (это необходимо, т. к. цена книги в целом может меняться со временем).

Также к сущностям книги и клиента добавим мета-поле, определяющее, удалена ли сущность или нет.

Таким образом имеем схему базы данных, представленную на рисунке

1.

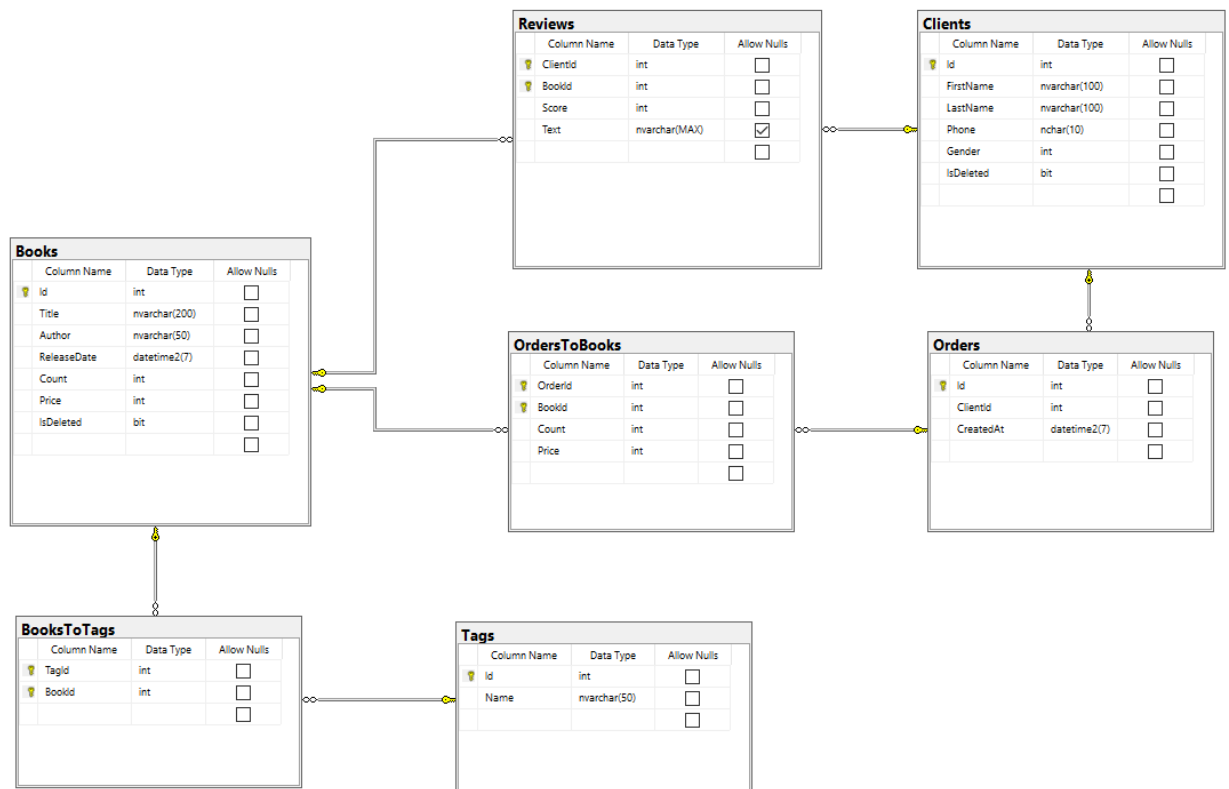


Рисунок 1. Схема базы данных

2 Создание запросов

Создадим запросы к таблицам. Часть запросов является «шаблонами», так как предполагают вставку какого-то значения на место шаблона. Такие места обрамлены фигурными скобками, например *{id}*.

2.1 Запросы к таблице книг

- Поиск книги по уникальному идентификатору

```
select b.Id, b.Title, b.Author, b.ReleaseDate, b.IsDeleted, b.Count, b.Price
from Books b
where b.IsDeleted = 0 and b.Id = {id}
```

- Поиск книг по названию

```
select b.Id, b.Title, b.Author, b.ReleaseDate, b.IsDeleted, b.Count, b.Price
from Books b
where b.IsDeleted = 0 and b.Title = {title}
order by b.Id
```

- Поиск книг по автору

```
select b.Id, b.Title, b.Author, b.ReleaseDate, b.IsDeleted, b.Count, b.Price
from Books b
where b.IsDeleted = 0 and b.Author = {author}
order by b.Id
```


- Поиск книг, количество которых меньше заданного числа

```
select b.Id, b.Title, b.Author, b.ReleaseDate, b.IsDeleted, b.Count, b.Price
from Books b
where b.IsDeleted = 0 and b.Count < {count}
order by b.Id
```

- Поиск книг, которые имеют данные теги

```
select b.Id, b.Title, b.Author, b.ReleaseDate, b.IsDeleted, b.Count, b.Price
from Books b
where b.IsDeleted = 0 and {condition}
order by b.Id
```

Condition имеет формат:

```
{tagName1} in (select t.Name from BooksToTags btt join Tags t on btt.TagId =
t.Id where b.Id = btt.BookId)

and

{tagName2} in (select t.Name from BooksToTags btt join Tags t on btt.TagId =
t.Id where b.Id = btt.BookId)

and

...
```

- Создание книги

```
insert into Books (Title, Author, ReleaseDate, Count, Price)
output inserted.*
values ({title}, {author}, {releaseDate}, {price}, {count})
```

- Изменение книги

```
update Books
set Title = {title}, Author = {author}, ReleaseDate = {releaseDate},
    Count = {count}, Price = {price}
where Id = {id}
```

- Удаление книги

```
delete from Books where Id = {id}
```

- Количество продаж у книги

```
select coalesce(sum(o.Count), 0) as Value
from OrdersToBooks o
join Books b on o.BookId = b.Id
where o.BookId = {id} and b.IsDeleted = 0
```

- Средняя оценка книги

```
select avg(cast(r.Score as float)) as Value
from Reviews r
```

```
join Books b on r.BookId = b.Id  
where r.BookId = {id} and b.IsDeleted = 0
```

- Принесенная книгой выручка

```
select coalesce(sum(Count * Price), 0) as Value  
from OrdersToBooks  
where BookId = {id}
```

- Топ книг по продажам

```
select top({topCount}) b.Id, b.Title, b.Author, b.ReleaseDate, b.IsDeleted,  
b.Count, b.Price  
from Books b  
where b.IsDeleted = 0  
order by  
(  
    coalesce((select coalesce(sum(o.Count), 0) from OrdersToBooks o where b.Id  
= o.BookId), 0)  
) desc, b.Id
```

- Топ книг по оценке

```
select top({topCount}) b.Id, b.Title, b.Author, b.ReleaseDate, b.IsDeleted,  
b.Count, b.Price  
from Books b  
where b.IsDeleted = 0  
order by  
(  
    coalesce((select avg(cast(r.Score as float)) from Reviews r where b.Id =  
r.BookId), 0)  
) desc, b.Id
```

- Топ книг по принесенной выручке

```
select top({topCount}) b.Id, b.Title, b.Author, b.ReleaseDate, b.IsDeleted,  
b.Count, b.Price  
from Books b  
where b.IsDeleted = 0  
order by  
(  
    coalesce((select coalesce(sum(o.Count * o.Price), 0) from OrdersToBooks o  
where b.Id = o.BookId), 0)  
) desc, b.Id
```

2.2 Запросы к таблице клиентов

- Поиск клиента по уникальному идентификатору

```
select c.Id, c.FirstName, c.LastName, c.Phone, c.Gender, c.IsDeleted  
from Clients c  
where c.IsDeleted = 0 and c.Id = {id}
```

- Поиск клиентов по имени

```
select c.Id, c.FirstName, c.LastName, c.Phone, c.Gender, c.IsDeleted
```

```
from Clients c
where c.IsDeleted = 0 and c.FirstName = {firstName} and c.LastName =
{lastName}
```

- Поиск клиента по телефону

```
select c.Id, c.FirstName, c.LastName, c.Phone, c.Gender, c.IsDeleted
from Clients c
where c.IsDeleted = 0 and c.Phone = {phone}
```

- Поиск клиентов по полу

```
select c.Id, c.FirstName, c.LastName, c.Phone, c.Gender, c.IsDeleted
from Clients c
where c.IsDeleted = 0 and c.Gender = {gender}
order by c.Id
```

- Создание клиента

```
insert into Clients (FirstName, LastName, Phone, Gender)
output inserted.*
values ({firstName}, {lastName}, {phone}, {gender})
```

- Изменение клиента

```
update Clients
set FirstName = {FirstName}, LastName = {LastName},
    Phone = {Phone}, Gender = {Gender}
where Id = {Id}
```

- Удаление клиента

```
delete from Clients where Id = {Id}
```

- Принесенная клиентом выручка

```
select coalesce(sum(otb.Count * otb.Price), 0) as Value
from Orders o
join Clients c on o.ClientId = c.Id
join OrdersToBooks otb on o.Id = otb.OrderId
where c.IsDeleted = 0 and o.ClientId = {client.Id}
```

- Топ клиентов по принесенной выручке

```
select top({topCount}) c.Id, c.FirstName, c.LastName, c.Phone, c.Gender,
c.IsDeleted
from Clients c
where c.IsDeleted = 0
order by (coalesce((
    select coalesce(sum(otb.Count * otb.Price), 0)
    from OrdersToBooks otb
    join Orders o on otb.OrderId = o.Id
    join Clients cc on cc.Id = o.ClientId
    where cc.Id = c.Id), 0)
) desc, c.Id
```

2.3 Запросы к таблице заказов

- Поиск заказа по уникальному идентификатору

```
select o.Id as OrderId, o.ClientId, o.CreatedAt, c.FirstName, c.LastName,
c.Phone, c.Gender, c.IsDeleted as IsClientDeleted
from Orders o
join Clients c on c.Id = o.ClientId
where o.Id = {id}
```

- Поиск заказов по клиенту

```
select o.Id as OrderId, o.ClientId, o.CreatedAt, c.FirstName, c.LastName,
c.Phone, c.Gender, c.IsDeleted as IsClientDeleted
from Orders o
join Clients c on c.Id = o.ClientId
where c.Id = {clientId}
```

- Поиск заказов по книге

```
select o.Id as OrderId, o.ClientId, o.CreatedAt, c.FirstName, c.LastName,
c.Phone, c.Gender, c.IsDeleted as IsClientDeleted
from Orders o
join Clients c on c.Id = o.ClientId
where exists(select 1 from OrdersToBooks otb
             where o.Id = otb.OrderId and otb.BookId = {bookId})
```

- Создание заказа

```
declare @orders table(Id int)

insert into Orders (ClientId)
output inserted.Id into @orders
values ({clientId})

declare @orderId int
set @orderId = (select top(1) Id from @orders)

insert into OrdersToBooks (OrderId, BookId, Count)
values {booksStr}

select o.Id as OrderId, o.ClientId, o.CreatedAt,
c.FirstName, c.LastName, c.Phone, c.Gender, c.IsDeleted as IsClientDeleted
from Orders o
join Clients c on c.Id = o.ClientId
where o.Id = @orderId
```

bookStr имеет формат:

```
(@orderId, {book1Id}, {book1Count}), (@orderId, {book2Id}, {book2Count}), ...
```

- Получение итоговой стоимости заказа

```
select COALESCE(SUM(otb.Count * b.Price), 0) as Value
from OrdersToBooks otb
join Books b on b.Id = otb.BookId
```

```
where otb.OrderId = {orderId}
```

- Получение списка книг из заказа

```
select otb.BookId, otb.OrderId, otb.Count as OrderedCount, otb.Price as  
OrderedPrice, b.Title as BookTitle, b.Author as BookAuthor, b.ReleaseDate as  
BookReleaseDate, b.IsDeleted as IsBookDeleted, b.Count as TotalCount, b.Price  
as BookPrice  
from OrdersToBooks otb  
join Books b on otb.BookId = b.Id  
where OrderId = {orderId}
```

2.4 Запросы к таблице тегов

- Поиск тега по уникальному идентификатору

```
select t.Id, t.Name  
from Tags t  
where t.Id = {id}
```

- Поиск тега по имени

```
select t.Id, t.Name  
from Tags t  
where t.Name = {name}
```

- Поиск тегов по книге

```
select t.Id, t.Name  
from Books b  
join BooksToTags btt on btt.BookId = b.Id  
join Tags t on btt.TagId = t.Id  
where b.IsDeleted = 0 and b.Id = {book.Id}
```

- Создание тега

```
insert into Tags (Name)  
output inserted.*  
values ({name})
```

- Изменение тега

```
update Tags  
set Name = {name}  
where Id = {id}
```

- Удаление тега

```
delete from Tags where Id = {id}
```

2.5 Запросы к таблице, связывающей книги и теги

- Создать связь книги и тега

```
insert into BooksToTags (BookId, TagId)  
values ({bookId}, {tagId})
```

- Удалить связь книги и тега

```
delete from BookToTags where BookId = {bookId} and TagId = {tagId}
```

2.6 Запросы к таблице отзывов

- Поиск отзыва по уникальному идентификатору

```
select r.BookId, r.ClientId, r.Score, r.Text, b.Title, b.Author,
b.ReleaseDate, b.Count, b.Price, b.IsDeleted as IsBookDeleted, c.FirstName,
c.LastName, c.Phone, c.Gender, c.IsDeleted as IsClientDeleted
from Reviews r
join Clients c on c.Id = r.ClientId
join Books b on b.Id = r.BookId
where c.Id = {clientId} and b.Id = {bookId}
```

- Поиск отзывов по клиенту

```
select r.BookId, r.ClientId, r.Score, r.Text, b.Title, b.Author,
b.ReleaseDate, b.Count, b.Price, b.IsDeleted as IsBookDeleted, c.FirstName,
c.LastName, c.Phone, c.Gender, c.IsDeleted as IsClientDeleted
from Reviews r
join Clients c on c.Id = r.ClientId
join Books b on b.Id = r.BookId
where c.Id = {clientId}
```

- Поиск отзывов по книге

```
select r.BookId, r.ClientId, r.Score, r.Text, b.Title, b.Author,
b.ReleaseDate, b.Count, b.Price, b.IsDeleted as IsBookDeleted, c.FirstName,
c.LastName, c.Phone, c.Gender, c.IsDeleted as IsClientDeleted
from Reviews r
join Clients c on c.Id = r.ClientId
join Books b on b.Id = r.BookId
where b.Id = {bookId}
```

- Создание отзыва

```
insert into Reviews (ClientId, BookId, Score, Text)
values ({clientId}, {bookId}, {score}, {text})

select r.BookId, r.ClientId, r.Score, r.Text, b.Title, b.Author,
b.ReleaseDate, b.Count, b.Price, b.IsDeleted as IsBookDeleted, c.FirstName,
c.LastName, c.Phone, c.Gender, c.IsDeleted as IsClientDeleted
from Reviews r
join Clients c on c.Id = r.ClientId
join Books b on b.Id = r.BookId
where c.Id = {clientId} and b.Id = {bookId}
```

- Обновление отзыва

```
update Reviews
set Score = {score}, Text = {text}
where ClientId = {clientId} and BookId = {bookId}
```

- Удаление отзыва

```
delete from Reviews where BookId = {bookId} and ClientId = {clientId}
```

2.7 Запросы для работы с пользователями

В данных запросах используются процедуры, исходный код которых приведен далее по тексту в блоке про обеспечение конфиденциальности.

- Поиск пользователя по уникальному идентификатору

```
select Id, Name, Role as RoleString  
from bsbd_principals  
where Id = {id}
```

- Поиск пользователя по имени

```
select Id, Name, Role as RoleString  
from bsbd_principals  
where Name = {name}
```

- Создание пользователя

```
exec bsbd_create_user {name}, {password}, {role}
```

- Удаление пользователя

```
exec bsbd_delete_user {name}
```

- Изменение пароля пользователя

```
exec bsbd_change_user_password {name}, {newPassword}, {oldPassword}
```

Защита базы данных

1 Обеспечение конфиденциальности

Для начала обеспечим защищенный канал связи для работы с базой данных. Для этого в настройках конфигурации включим принудительное шифрование (см. рис. 2), а также создадим самоподписанный сертификат (см. рис. 3) с помощью скрипта (см. листинг 1) и выдадим доступ к ключам для MS SQL (см. рис. 4) [1].

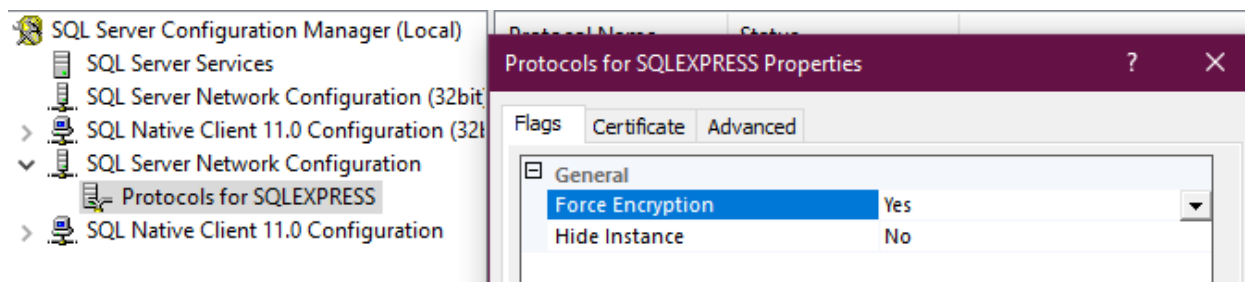


Рисунок 2. Включение шифрование канала связи с базой данных

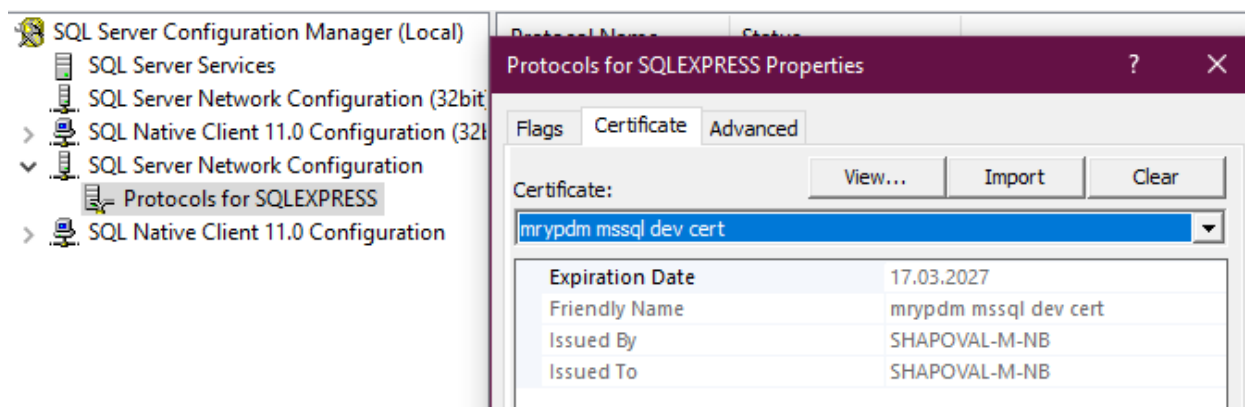


Рисунок 3. Выбор сертификата

Листинг 1. Скрипт создания самоподписанного сертификата

```
New-SelfSignedCertificate `
-Type SSLServerAuthentication `
-Subject "CN=$env:COMPUTERNAME" `
-DnsName "$env:COMPUTERNAME",'localhost.' `
-KeyAlgorithm "RSA" `
-KeyLength 2048 `
-Hash "SHA256" `
-TextExtension "2.5.29.37={text}1.3.6.1.5.5.7.3.1" `
-NotAfter (Get-Date).AddMonths(36) `
-KeySpec KeyExchange `
-Provider "Microsoft RSA SChannel Cryptographic Provider"
```

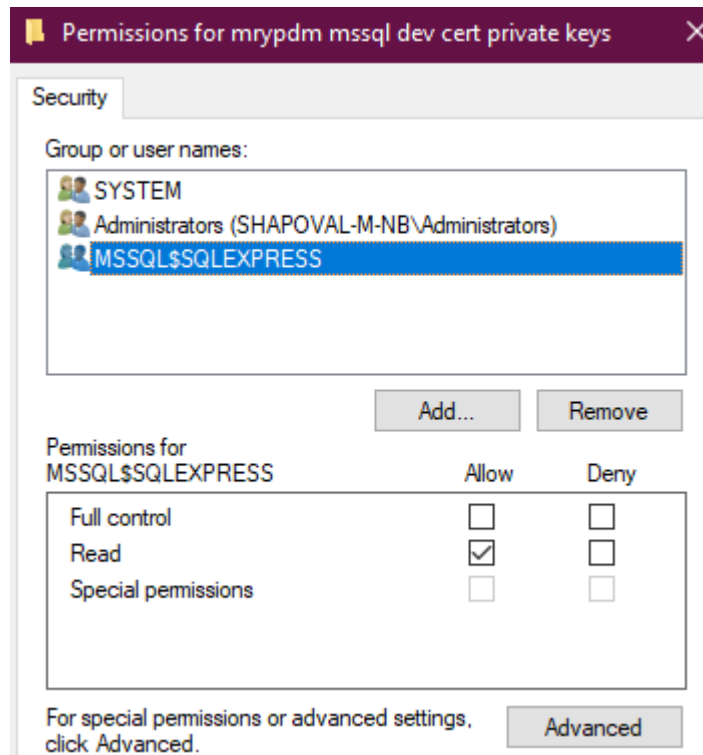



Рисунок 4. Выдача прав на чтение секретных
ключей для MS SQL

Далее настроим разграничение доступа. Для этого будем использовать средства, встроенные в MS SQL. Но для того, чтобы мы могли создавать полноценных пользователей с логином и паролем, предварительно нужно сделать базу данных автономной (см. рис. 5) [2].

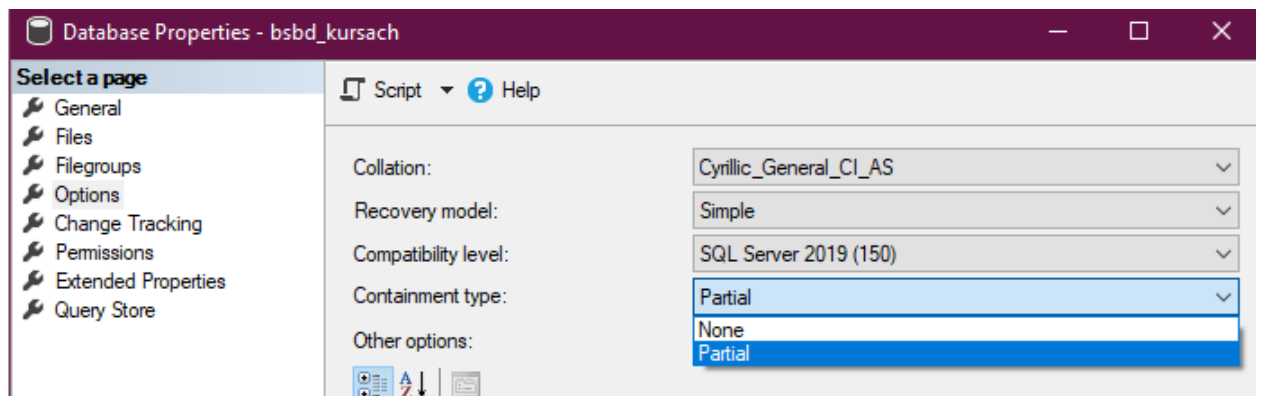


Рисунок 5. Автономная база данных

Далее определим представление всех пользователей:

```
create view bsbd_principals (Id, Name, Role) as
select m.principal_id, m.name, dbo.bsbd_role_to_int(r.name)
from sys.database_role_members rm
    join sys.database_principals r on rm.role_principal_id = r.principal_id
    join sys.database_principals m on rm.member_principal_id = m.principal_id
```

```
WHERE m.type_desc = 'SQL_USER' AND m.name ≠ 'dbo'
```

Определим роли и их права:

- readonly – пользователи этой роли могут читать все таблицы (select) и изменять свой пароль:

```
create role bsbd_readonly_role

grant select on Books to bsbd_readonly_role
grant select on Tags to bsbd_readonly_role
grant select on Clients to bsbd_readonly_role
grant select on Orders to bsbd_readonly_role
grant select on Reviews to bsbd_readonly_role
grant select on OrdersToBooks to bsbd_readonly_role
grant select on BooksToTags to bsbd_readonly_role
grant select on bsbd_principals to bsbd_readonly_role
grant execute on bsbd_change_user_password to bsbd_readonly_role
```

- worker – пользователи этой роли могут создавать и обновлять книги, клиентов и теги, создавать отзывы и заказы, удалять связь книги и тега:

```
create role bsbd_worker_role

grant insert on Books to bsbd_worker_role
grant update on Books (Title, Author, ReleaseDate, Count) to bsbd_worker_role
grant insert, delete on BooksToTags to bsbd_worker_role

grant insert on Clients to bsbd_worker_role
grant update on Clients (FirstName, LastName, Phone, Gender) to
bsbd_worker_role

grant insert, update on Tags to bsbd_worker_role

grant insert on Reviews to bsbd_worker_role

grant insert on Orders to bsbd_worker_role
grant insert on OrdersToBooks to bsbd_worker_role

alter role bsbd_readonly_role add member bsbd_worker_role
```

- admin – пользователи этой роли могут то же, что и работник, а также изменять и удалять отзывы, удалять книги, клиентов, теги:

```
create role bsbd_admin_role

grant insert, update, delete on Books to bsbd_admin_role
grant insert, delete on BooksToTags to bsbd_admin_role

grant insert, update, delete on Clients to bsbd_admin_role

grant insert, update, delete on Tags to bsbd_admin_role
```

```
grant insert, update, delete on Reviews to bsbd_admin_role

grant insert on Orders to bsbd_admin_role
grant insert on OrdersToBooks to bsbd_admin_role

alter role bsbd_worker_role add member bsbd_admin_role
```

- security – пользователи этой роли могут то же, что и администраторы, а также удалять и создавать пользователей:

```
create role bsbd_security_role

grant execute, alter on bsbd_create_user to bsbd_security_role
grant execute, alter on bsbd_delete_user to bsbd_security_role
grant alter on bsbd_change_user_password to bsbd_security_role

alter role bsbd_admin_role add member bsbd_security_role
alter role db_securityadmin add member bsbd_security_role
alter role db_accessadmin add member bsbd_security_role
```

Определим вспомогательные процедуры:

- Создание пользователя:

```
create proc bsbd_create_user @userName nvarchar(50), @password nvarchar(50),
@role int as
    if dbo.bsbd_validate_injection(@userName) = 1
        or dbo.bsbd_validate_injection(@password) = 1
    begin
        raiserror ('Credential cannot contains symbols: [];"', 15, 1)
        return
    end

    declare @roleString nvarchar(max)
    set @roleString = dbo.bsbd_int_to_role(@role)
    if @roleString IS NULL
    begin
        raiserror ('Role must be in range [0; 2]', 15, 1)
        return
    end

    declare @createUserSql nvarchar(max)
    set @createUserSql =
        'create user [' + @userName + '] ' +
        ' with password = N''' + @password + ''''
    exec (@createUserSql)

    declare @addToRoleSql nvarchar(max)
    set @addToRoleSql =
        'alter role [' + @roleString + '] ' +
        'add member [' + @userName + ']'
    exec (@addToRoleSql)
```

- Удаление пользователя:

```

create procedure bsbd_delete_user @userName nvarchar(50) as
    if dbo.bsbd_validate_injection (@userName) = 1
    begin
        raiserror ('Credential cannot contains symbols: [];"', 15, 1)
        return
    end

    declare @deleteUserSql nvarchar(max)
    set @deleteUserSql = 'drop user [' + @userName + ']'
    exec (@deleteUserSql)

```

- Изменение пароля пользователя. В случае, если oldPassword равен NULL, происходит принудительная смена пароля. Доступ на такое действие имеет только роль security:

```

create proc bsbd_change_user_password @userName nvarchar(50),
                                     @newPassword nvarchar(50),
                                     @oldPassword nvarchar(50) = null as
    if dbo.bsbd_validate_injection(@userName) = 1
    or dbo.bsbd_validate_injection(@newPassword) = 1
    or dbo.bsbd_validate_injection (@oldPassword) = 1
    begin
        raiserror ('Credential cannot contains symbols: [];"', 15, 1)
        return
    end

    declare @changePasswordSql nvarchar(max)

    if @oldPassword is null
    begin
        set @changePasswordSql =
            'alter user [' + @userName + '] ' +
            'with password=N''' + @newPassword + ''''
    end
    else
    begin
        set @changePasswordSql =
            'alter user [' + @userName + '] ' +
            'with password=N''' + @newPassword + ''' ' +
            'old_password=N''' + @oldPassword + ''''
    end

    exec (@changePasswordSql)

```

- Проверка аргумента на SQL инъекцию:

```

create function bsbd_validate_injection(@str nvarchar(100))
returns int as
begin
    if @str is null
    begin
        return 2
    end

```

```

    if charindex('[', @str) > 0 or charindex(']', @str) > 0
        or charindex('"', @str) > 0 or charindex(';', @str) > 0
        or charindex(''''', @str) > 0
    begin
        return 1
    end

    return 0
end

```

- Перевод роли в число:

```

create function bsbd_role_to_int(@roleName nvarchar(max))
    returns int as
begin
    if @roleName = 'bsbd_security_role'
    begin
        return 0
    end

    if @roleName = 'bsbd_admin_role'
    begin
        return 1
    end

    if @roleName = 'bsbd_worker_role'
    begin
        return 2
    end

    return 3
end

```

- Перевод числа в роль

```

create function bsbd_int_to_role(@roleNumber int)
    returns nvarchar(max) as
begin
    if @roleNumber = 0
    begin
        return 'bsbd_security_role'
    end

    if @roleNumber = 1
    begin
        return 'bsbd_admin_role'
    end

    if @roleNumber = 2
    begin
        return 'bsbd_worker_role'
    end

    return null
end

```

```
end
```

2 Обеспечение целостности

Определим ограничения на атрибуты:

- Пол клиента имеет всего 2 значения: мужской и женский – это достигается типом данных BIT;
- Текст отзыва может быть пустым – это достигается разращением NULL для столбца при создании таблицы;
- Номер телефона клиента является уникальной строкой из 10 цифр:

```
alter table Clients
with check add constraint CK_Clients
check (Phone like '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')

create unique nonclustered index IX_Clients_Phone on Clients (Phone asc)
where (IsDeleted = 0)

WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
on [PRIMARY]
```

- Имя тега должно быть уникальным:

```
create unique nonclustered index IX_Tags_Name on Tags (Name asc)
with (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)
on [PRIMARY]
```

- Дата создания заказа имеет значение по умолчанию, равное текущим дате и времени:

```
alter table Orders
add constraint DF_Orders_CreatedAt
default (getdate()) for CreatedAt
```

Определим триггеры:

- Вместо удаления книги помечать ее удаленной:

```
create trigger bsbd_mark_book_as_deleted on Books instead of delete as
begin
    update Books
    set
        IsDeleted = 1,
        Count = 0
    from Books c join deleted d on c.Id = d.Id
```

```
delete bt from BooksToTags bt join deleted d on bt.BookId = d.Id
end
```

- Вместо удаления клиента помечать его удаленным:

```
create trigger bsbd_mark_client_as_deleted on Clients instead of delete as
begin
    update Clients
    set
        FirstName = '',
        LastName = '',
        Phone = '0000000000',
        IsDeleted = 1
    from Clients c join deleted d on c.Id = d.Id
end
```

- Запрет удаления и обновления заказов и заказанных книг:

```
create trigger bsbd_prevent_orders_change on Orders after update, delete as
begin
    rollback transaction
end

create trigger bsbd_prevent_books_in_orders_change on OrdersToBooks after
update, delete as
begin
    rollback transaction
end
```

- Проверка, что в магазине достаточно книг для нового заказа:

```
create trigger bsbd_verify_order on OrdersToBooks instead of insert as
begin
    if exists((select OrderId from (select * from OrdersToBooks except select
* from inserted) as c)
        intersect
        select OrderId from inserted)
    begin
        raiserror('Cannot add books to already created order', 15, 1)
        rollback transaction
    end

    if exists(select b.Id from inserted i join Books b on i.BookId = b.Id
where i.Count > b.Count)
    begin
        raiserror('Not enough books to order', 15, 1)
        rollback transaction
    end

    update Books
    set Count = b.Count - i.Count
    from Books b join inserted i on b.Id = i.BookId

    insert into OrdersToBooks (OrderId, BookId, Count, Price)
```

```
select i.OrderId, i.BookId, i.Count, b.Price  
from inserted i join Books b on i.BookId = b.Id  
end
```

3 Обеспечение доступности

Определим индексы для повышения быстродействия:

- Поиск книги по автору:

```
create nonclustered index IX_Books_Author on Books (Author asc)  
where (IsDeleted = 0)  
with (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,  
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS =  
ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)  
on [PRIMARY]
```

- Поиск книги по названию:

```
create nonclustered index IX_Books_Title on Books (Title asc)  
where (IsDeleted = 0)  
with (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,  
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS =  
ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF)  
on [PRIMARY]
```

Далее настроим резервное копирование базы данных [3]. Для этого создадим процедуру `sp_BackupDatabases` [4] и добавим ее исполнение в планировщик задач (см. рис. 6). Как видно из рисунка 7, созданная задача работает верно.

Разработка приложения

Для взаимодействия с базой данных разработаем приложение с графическим интерфейсом. Для этого используем язык C# и программную платформу .NET 8 [5] с использованием WPF [6] для графического интерфейса и Entity Framework Core [7] для работы с базой данных – его преимущества над встроенным в C# пространством имен SqlClient в том, что он обеспечивает защиту от SQL инъекций.

Исходный код приложения представлен в репозитории https://github.com/mrypdm/bsbd_kursach

На рисунке 8 представлен вид главного окна при запуске приложения. На рисунке 9 то же окно, но после аутентификации.

На рисунке 10 представлено окно аутентификации, а на рисунке 11 окно смены пароля.

На рисунке 12 представлено окно сущности – оно отображает список всех сущностей, которые подходят под некоторое условие (эти условия представлены кнопками на главном окне). Из этого окна можно переходить в другие окна сущностей, если такая связь возможна, например, по книге можно получить список заказов с этой книгой (см. рисунок 13).

Для проверки разграничения доступа проверим создание и удаление отзыва. Изначально список отзывов для книги с ID=1 пуст (см. рис. 14). Для проверки создадим от имени bsbd_owner с ролью security пользователя с ролью работника (см. рис. 15-16) и от имени работника попробуем создать (см. рис. 17) и удалить отзыв. Как видно из рисунка 18, кнопка удаления недоступна, потому работник не сможет удалить отзыв. Но для большей уверенности попробуем удалить отзыв через SQL запрос (см. рис. 19) – на видно, это тоже невозможно.



Рисунок 8. Начальный вид главного окна

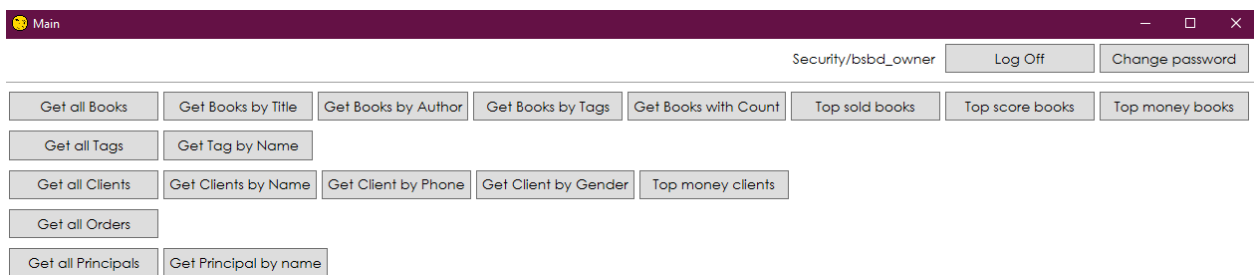


Рисунок 9. Главное окно

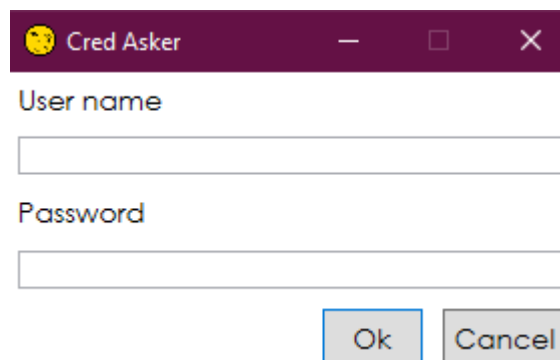
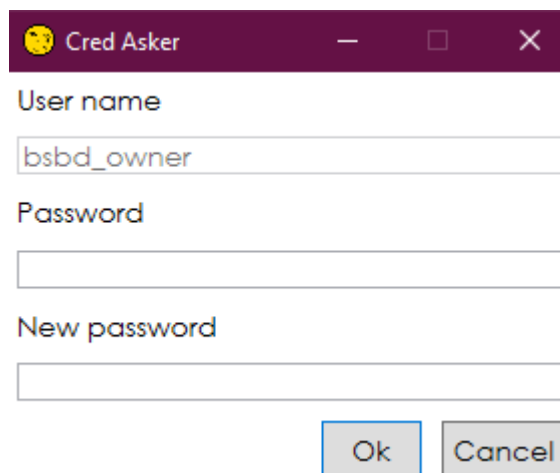


Рисунок 10. Окно аутентификации



Cred Asker

User name

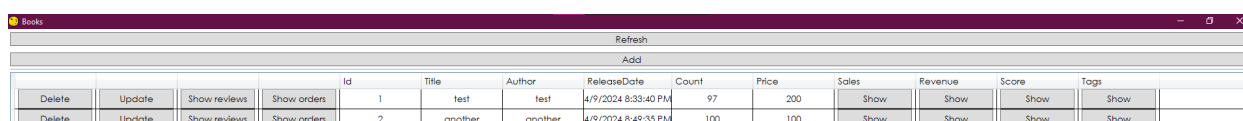
bsbd_owner

Password

New password

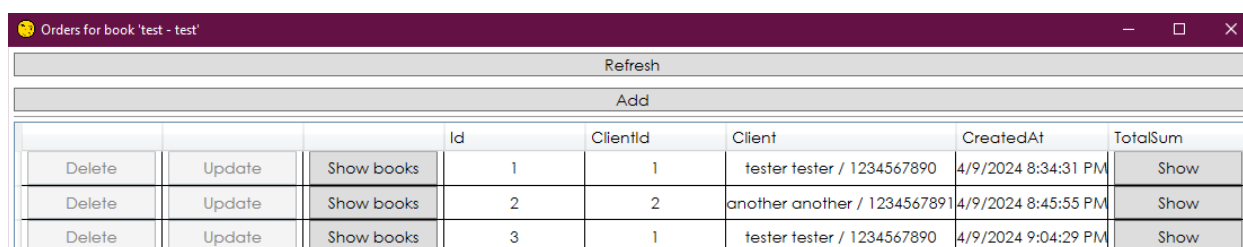
Ok Cancel

Рисунок 11. Окно смены пароля



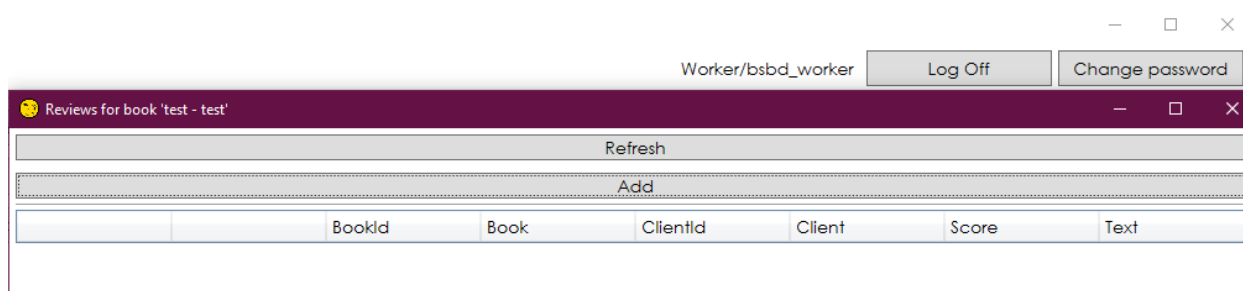
Books													
Refresh													
Add													
Delete	Update	Show reviews	Show orders	Id	Title	Author	ReleaseDate	Count	Price	Sales	Revenue	Score	Tags
				1	test	test	4/9/2024 8:33:40 PM	97	200	Show	Show	Show	Show
				2	another	another	4/9/2024 8:49:35 PM	100	100	Show	Show	Show	Show

Рисунок 12. Список всех книг



Orders for book 'test - test'							
Refresh							
Add							
Delete	Update	Show books	Id	ClientId	Client	CreatedAt	TotalSum
			1	1	tester tester / 1234567890	4/9/2024 8:34:31 PM	Show
			2	2	another another / 1234567891	4/9/2024 8:45:55 PM	Show
			3	1	tester tester / 1234567890	4/9/2024 9:04:29 PM	Show

Рисунок 13. Список заказов с книгой с ID=1



Worker/bsbd_worker Log Off Change password

Reviews for book 'test - test'						
Refresh						
Add						
	BookId	Book	ClientId	Client	Score	Text

Рисунок 14. Список отзывов книги с ID=1 до создания отзыва

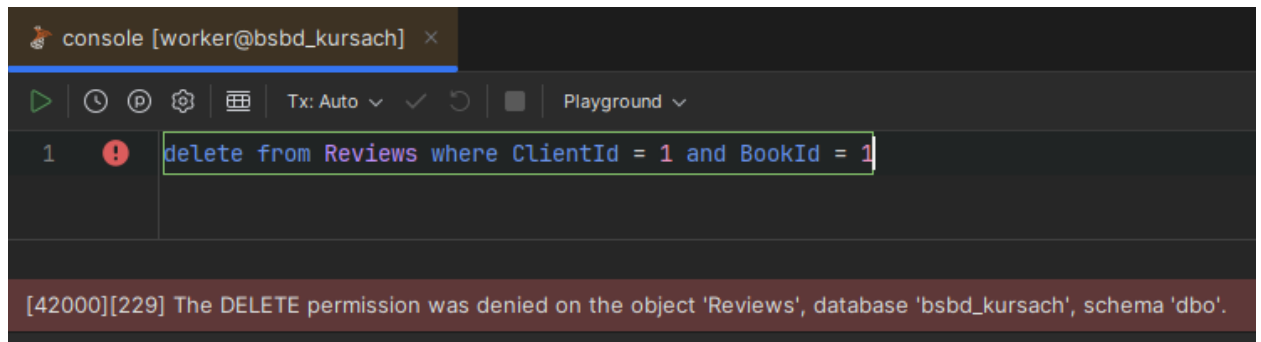


Рисунок 19. Попытка удаления через SQL запрос

Заключение

В ходе выполнения курсовой работы была разработана защищенная база данных информационной системы «Книжный магазин».

Были настроены роли и права для обеспечения конфиденциальности.

Были разработаны ограничения, индексы, триггеры, обеспечивающие целостность данных.

Для обеспечения доступности были разработаны индексы (которые повышают производительность) и настроено резервное копирование базы данных.

Для работы с базой данных было разработано приложение с графическим интерфейсом.

Список использованных источников

- 1 How to generate a self-signed SSL certificate for MS SQL server 2008 R2 using OpenSSL? // StackExchange – URL: <https://dba.stackexchange.com/a/284769/288326> (дата обращения: 14.04.2024)
- 2 Автономные базы данных [Электронный ресурс] // Microsoft Learn – URL: <https://learn.microsoft.com/ru-ru/sql/relational-databases/databases/contained-databases> (дата обращения 14.04.2024)
- 3 Планирование и автоматизация резервного копирования баз данных SQL Server в SQL Server Express [Электронный ресурс] // Microsoft Learn – URL: <https://learn.microsoft.com/ru-ru/troubleshoot/sql/database-engine/backup-restore/schedule-automate-backup-database> (дата обращения: 14.04.2024)
- 4 Скрипт создания резервной копии базы данных [Электронный ресурс] // Github – URL: https://raw.githubusercontent.com/microsoft/mssql-support/master/sample-scripts/backup_restore/SQL_Express_Backups.sql (дата обращения: 14.04.2024)
- 5 Документация по .NET [Электронный ресурс] // Microsoft Learn – URL: <https://learn.microsoft.com/ru-ru/dotnet/> (дата обращения: 14.04.2024)
- 6 Документация по WPF [Электронный ресурс] // Microsoft Learn – URL: <https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf> (дата обращения: 14.04.2024)
- 7 Обзор Entity Framework Core [Электронный ресурс] // Microsoft Learn – URL: <https://learn.microsoft.com/ru-ru/ef/core/> (дата обращения: 14.04.2024)