create database Библиотека

go

use Библиотека

go

--Создадим таблицу Читатель:

create table Читатель (

id\_читателя int primary key identity(1,1) NOT NULL,

ФИО\_читателя varchar(50),

Телефон\_читателя varchar(15),

Адрес varchar(50)

);

--Создадим таблицу Книжный отдел:

create table Книжный\_отдел (

id\_отдела int primary key identity(1,1) NOT NULL,

Название\_отдела varchar(50) NOT NULL,

Количество\_книг int NOT NULL,

Телефон\_отдела varchar(15)

);

--Создадим таблицу Сотрудники:

create table Сотрудники (

id\_сотрудника int primary key identity(1,1) NOT NULL,

ФИО\_сотрудника varchar(50),

Телефон\_сотрудника varchar(15)

);

--Создадим таблицу Книги:

create table Книги (

id\_книги int primary key identity(1,1) NOT NULL,

Название\_книги varchar(50),

Автор varchar(50),

Количество\_страниц int,

);

--Создадим таблицу Выдача Книг:

create table Выдача\_книг (

id\_выдачи int primary key identity(1,1) NOT NULL,

id\_читателя int,

id\_отдела int,

id\_сотрудника int,

id\_книги int,

Инвентарный\_номер varchar(50),

Дата\_выдачи\_книги date,

Дата\_возврата\_книги date,

foreign key (id\_читателя) references Читатель(id\_читателя), -- создание связей между таблицами

foreign key (id\_сотрудника) references Сотрудники(id\_сотрудника),

foreign key (id\_отдела) references Книжный\_отдел(id\_отдела),

foreign key (id\_книги) references Книги(id\_книги),

);

use Библиотека

go

--добавление читателей

insert into Читатель(ФИО\_читателя, Телефон\_читателя, Адрес)

values ('Соловьев Виктор Иванович', '89286898776', 'пер.Гагарина, 06'),

('Игнатьева Олеся Владимировна', '89997777668', 'бульвар Чехова, 56'),

('Полупан Юлия Викторовна', '80986787665', 'проезд Гоголя, 23'),

('Дегтярева Лариса Николаевна', '88997654553', 'въезд Будапештсткая, 79'),

('Белозеров Евгений Владимирович', '89990988776', 'спуск Славы, 11')

--Заполним таблицу «Книги»:

insert into Книги(Название\_книги, Автор, Количество\_страниц)

values ('Мастер и Маргарита', 'Михаил Булгаков', 512),

('Преступление и наказание', 'Фёдор Достоевский', 331),

('Война и мир', 'Лев Толстой', 478),

('Собачье сердце', 'Михаил Булгаков', 288),

('Идиот', 'Фёдор Достоевский', 672)

--Заполним таблицу «Отдел книг»:

insert into Книжный\_отдел(Название\_отдела, Количество\_книг, Телефон\_отдела)

values ('Отдел комплектования и научной обработки', 31974, '87899876556'),

('Отдел хранения фондов', 41987, '89766543212'),

('Отдел обслуживания', 11688, '89995678997'),

('Отдел компьютеризации библиотечных процессов', 13839, '89565674334'),

('Научно-библиографический отдел', 17536, '87654563443')

--Заполним таблицу «Сотрудники»:

insert into Сотрудники(ФИО\_сотрудника, Телефон\_сотрудника)

values ('Лебедев Андрей Маратович', '80988765667'),

('Устинова Амалия Константиновна', '89887898778'),

('Тимофеева Анастасия Данииловна', '87776776556'),

('Яшина Полина Кирилловна', '86899873223'),

('Лаврова Александра Григорьевна', '88887659889')

--Заполним таблицу «Выдача книг»:

insert into Выдача\_книг(id\_читателя, id\_отдела, id\_сотрудника, id\_книги, Инвентарный\_номер, Дата\_выдачи\_книги, Дата\_возврата\_книги)

values (1, 1, 1, 1, 167, '2022-04-10', '2022-05-10'),

(2, 2, 2, 2, 254, '2022-01-12', '2022-02-12'),

(3, 3, 3, 3, 367, '2022-08-25', '2022-09-25'),

(4, 4, 4, 4, 453, '2022-11-14', '2022-11-28'),

(5, 5, 5, 5, 586, '2022-07-11', '2022-08-11')

use Библиотека

go

--триггер увеличивает значение количества книг в отделе при добавлении новой книги

CREATE TRIGGER Обновить\_Общее\_Количество\_Книг

ON Книги

AFTER INSERT

AS

BEGIN

UPDATE Книжный\_отдел

SET Количество\_книг = Количество\_книг + 1

WHERE id\_отдела = (SELECT id\_отдела FROM inserted)

END

go

--триггер изменяет дату в поле Дата\_возврата\_книги после обновления поля Дата\_выдачи\_книги

CREATE TRIGGER Обновить\_Дата\_Возврата\_Книги

ON Выдача\_книг

AFTER UPDATE

AS

BEGIN

UPDATE Выдача\_книг

SET Дата\_возврата\_книги = DATEADD(day, 14, inserted.Дата\_выдачи\_книги)

FROM inserted

WHERE Выдача\_книг.id\_выдачи = inserted.id\_выдачи

END

go

--триггер обновляет поле "Дата\_выдачи\_книги" в таблице "Выдача\_книг" на текущую дату и время.

CREATE TRIGGER Обновить\_Дата\_Выдачи\_Книги

ON Выдача\_книг

AFTER INSERT

AS

BEGIN

UPDATE Выдача\_книг

SET Дата\_выдачи\_книги = GETDATE()

FROM inserted

WHERE Выдача\_книг.id\_выдачи = inserted.id\_выдачи

END

Go

use Библиотека

go

--процедура выводит информацию о книге по инвентарному номеру

CREATE PROCEDURE get\_invent\_post @post int

AS

BEGIN

SELECT Название\_книги, Автор, Дата\_возврата\_книги

FROM Книги INNER JOIN Выдача\_книг ON Книги.id\_книги = Выдача\_книг.id\_книги

WHERE Инвентарный\_номер = @post

END

go

--Хранимая процедура для поиска книги по названию:

CREATE PROCEDURE Найти\_Книгу\_По\_Названию

@Название\_книги VARCHAR(50)

AS

BEGIN

SELECT \*

FROM Книги

WHERE Название\_книги LIKE '%' + @Название\_книги + '%'

END

go

--Хранимая процедура для добавления новой книги:

CREATE PROCEDURE Добавить\_Книгу

@Название\_книги VARCHAR(50),

@Автор VARCHAR(50),

@Количество\_страниц INT

AS

BEGIN

INSERT INTO Книги (Название\_книги, Автор, Количество\_страниц)

VALUES (@Название\_книги, @Автор, @Количество\_страниц)

END

Go

use Библиотека

go

CREATE FUNCTION ПолучитьСписокКнигПоАвтору(@автор varchar(50))

RETURNS TABLE

AS

RETURN

(

SELECT Книги.id\_книги, Книги.Название\_книги, Книги.Количество\_страниц

FROM Книги

WHERE Книги.Автор = @автор

);

go

CREATE FUNCTION ПолучитьСписокВыданныхКнигПоЧитателю(@id\_читателя int)

RETURNS TABLE

AS

RETURN

(

SELECT Книги.Название\_книги, Выдача\_книг.Дата\_выдачи\_книги, Выдача\_книг.Дата\_возврата\_книги

FROM Выдача\_книг

INNER JOIN Книги ON Выдача\_книг.id\_книги = Книги.id\_книги

WHERE Выдача\_книг.id\_читателя = @id\_читателя

);

go

CREATE FUNCTION ПолучитьСписокЧитателейПоОтделу(@id\_отдела int)

RETURNS TABLE

AS

RETURN

(

SELECT Читатель.ФИО\_читателя, Читатель.Телефон\_читателя

FROM Выдача\_книг

INNER JOIN Читатель ON Выдача\_книг.id\_читателя = Читатель.id\_читателя

WHERE Выдача\_книг.id\_отдела = @id\_отдела

);

Go

use Библиотека

go

--отображает информацию о книгах, которые есть в каждом книжном отделе на основе данных из таблицы "Выдача\_книг"

CREATE VIEW Книги\_в\_Книжном\_Отделе AS

SELECT Книжный\_отдел.Название\_отдела, Книги.Название\_книги, Книги.Автор, Книги.Количество\_страниц

FROM Книжный\_отдел

JOIN Выдача\_книг ON Книжный\_отдел.id\_отдела = Выдача\_книг.id\_отдела

JOIN Книги ON Выдача\_книг.id\_книги = Книги.id\_книги;

go

-- представление выводит все книги, которые выданы после 25-08-2022

CREATE VIEW Представление

AS SELECT Название\_книги, Автор, Количество\_страниц

FROM Книги join Выдача\_книг on Выдача\_книг.id\_книги = Книги.id\_книги

WHERE Дата\_выдачи\_книги > '25-08-2022'

WITH CHECK OPTION

go

-- представление выводит данные о сотруднике, который работает в отделе хранения фондов

CREATE VIEW Представление\_2

AS SELECT Сотрудники.Телефон\_сотрудника, Сотрудники.ФИО\_сотрудника, Книжный\_отдел.Телефон\_отдела

FROM Книжный\_отдел INNER JOIN Выдача\_книг ON Книжный\_отдел.id\_отдела = Выдача\_книг.id\_отдела

INNER JOIN Сотрудники ON Выдача\_книг.id\_сотрудника = Сотрудники.id\_сотрудника

WHERE Название\_отдела = 'Отдел хранения фондов'

WITH CHECK OPTION

go

1. **Опишите основные принципы структуризации и нормализации базы данных.**

Основные принципы структуризации и нормализации базы данных включают следующее:

1. Единообразие: База данных должна быть организована по единым стандартам и соглашениям. Это включает использование одинаковых имен для таблиц, полей и других элементов базы данных, а также согласованное применение типов данных и форматов.

2. Минимизация избыточности: Избыточность данных может привести к проблемам согласованности и производительности. При проектировании базы данных следует стремиться к минимизации повторяющихся данных, чтобы каждая информация хранилась только в одном месте.

3. Уникальность идентификаторов: Каждая запись в базе данных должна иметь уникальный идентификатор, который позволяет однозначно идентифицировать эту запись. Обычно это реализуется с помощью первичных ключей, которые гарантируют уникальность значений и обеспечивают быстрый доступ к данным.

4. Нормализация: Нормализация является процессом разделения таблиц на более мелкие и логически связанные единицы для устранения аномалий и избыточности данных. Нормализация позволяет улучшить структуру базы данных, упростить операции вставки, обновления и удаления данных, а также повысить целостность данных.

5. Связи и отношения: База данных должна использовать связи и отношения для установления связей между таблицами и представления связанных данных. Это обеспечивает целостность данных, позволяет эффективно использовать данные из разных таблиц и упрощает выполнение запросов.

6. Независимость данных: Структура базы данных должна быть независима от приложений, которые используют эти данные. Это позволяет изменять и модифицировать приложения, не затрагивая структуру базы данных, и обеспечивает гибкость и масштабируемость системы.

7. Согласованность и целостность данных: База данных должна поддерживать согласованность и целостность данных путем применения ограничений, проверок и правил, которые гарантируют правильность и корректность данных.