create database Гостиница;

go

use Гостиница;

go

create table Гости (

id\_гостя int primary key identity(1,1),

ФИО varchar(50),

Телефон\_гостя varchar(20),

Город varchar(20),

Номер\_паспорта varchar(30)

);

create table График\_дежурства (

id\_дежурства int primary key identity(1,1),

Время\_начала\_работы time,

Время\_окончания\_работы time

);

create table Сотрудники (

id\_сотрудника int primary key identity(1,1),

ФИО varchar(50),

Должность varchar(20),

Телефон\_сотрудника varchar(20),

id\_дежурства int,

foreign key(id\_дежурства) references График\_дежурства(id\_дежурства)

);

create table Номера (

id\_номера int primary key identity(1,1),

Этаж varchar(15),

Класс varchar(15),

Стоимость money,

id\_сотрудника int,

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

);

create table Услуги\_в\_номере (

id\_услуги int primary key identity(1,1),

Вид\_услуги varchar(30),

Стоимость\_услуги money

);

create table Номер\_услуга (

id\_номера int,

id\_услуги int,

foreign key(id\_номера) references Номера(id\_номера),

foreign key(id\_услуги) references Услуги\_в\_номере(id\_услуги)

);

create table Регистрация (

id\_регистрации int primary key identity(1,1),

id\_гостя int,

id\_сотрудника int,

id\_номера int,

foreign key(id\_гостя) references Гости(id\_гостя),

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

foreign key(id\_номера) references Номера(id\_номера)

);

Use Гостиница

go

-- Заполнение таблицы "Гости"

INSERT INTO Гости (ФИО, Телефон\_гостя, Город, Номер\_паспорта)

VALUES

('Иванов Иван Иванович', '123456789', 'Москва', 'AB1234567'),

('Петров Петр Петрович', '987654321', 'Санкт-Петербург', 'CD9876543'),

('Сидорова Елена Александровна', '555111222', 'Новосибирск', 'EF5551112');

-- Заполнение таблицы "График\_дежурства"

INSERT INTO График\_дежурства (Время\_начала\_работы, Время\_окончания\_работы)

VALUES

('08:00', '16:00'),

('16:00', '00:00'),

('00:00', '08:00');

-- Заполнение таблицы "Сотрудники"

INSERT INTO Сотрудники (ФИО, Должность, Телефон\_сотрудника, id\_дежурства)

VALUES

('Иванова Анна Сергеевна', 'Администратор', '111222333', 1),

('Петров Алексей Иванович', 'Консьерж', '444555666', 2),

('Сидорова Ольга Петровна', 'Уборщица', '777888999', 3);

-- Заполнение таблицы "Номера"

INSERT INTO Номера (Этаж, Класс, Стоимость, id\_сотрудника)

VALUES

('2', 'Стандарт', 1500, 1),

('4', 'Люкс', 3000, 2),

('1', 'Эконом', 1000, 3);

-- Заполнение таблицы "Услуги\_в\_номере"

INSERT INTO Услуги\_в\_номере (Вид\_услуги, Стоимость\_услуги)

VALUES

('Завтрак', 200),

('Уборка номера', 100),

('Wi-Fi', 50);

-- Заполнение таблицы "Номер\_услуга"

INSERT INTO Номер\_услуга (id\_номера, id\_услуги)

VALUES

(1, 1),

(1, 2),

(2, 1),

(3, 3);

-- Заполнение таблицы "Регистрация"

INSERT INTO Регистрация (id\_гостя, id\_сотрудника, id\_номера)

VALUES

(1, 1, 1),

(2, 2, 2),

(3, 3, 3);

use Гостиница

go

CREATE FUNCTION ПолучитьИнформациюОГосте(@id\_гостя int)

RETURNS TABLE

AS

RETURN

(

SELECT Гости.ФИО, Гости.Телефон\_гостя, Гости.Город, Гости.Номер\_паспорта, Регистрация.id\_номера

FROM Гости

INNER JOIN Регистрация ON Гости.id\_гостя = Регистрация.id\_гостя

WHERE Гости.id\_гостя = @id\_гостя

);

go

CREATE FUNCTION ПолучитьСписокДежурныхСотрудников(@время time)

RETURNS TABLE

AS

RETURN

(

SELECT Сотрудники.ФИО, Сотрудники.Должность, График\_дежурства.Время\_начала\_работы, График\_дежурства.Время\_окончания\_работы

FROM Сотрудники

INNER JOIN График\_дежурства ON Сотрудники.id\_дежурства = График\_дежурства.id\_дежурства

WHERE @время BETWEEN График\_дежурства.Время\_начала\_работы AND График\_дежурства.Время\_окончания\_работы

);

go

CREATE FUNCTION ПолучитьСписокУслугВНомере(@id\_номера int)

RETURNS TABLE

AS

RETURN

(

SELECT Услуги\_в\_номере.Вид\_услуги, Услуги\_в\_номере.Стоимость\_услуги

FROM Услуги\_в\_номере

INNER JOIN Номер\_услуга ON Услуги\_в\_номере.id\_услуги = Номер\_услуга.id\_услуги

WHERE Номер\_услуга.id\_номера = @id\_номера

);

Go

use Гостиница

go

CREATE TRIGGER Проверка\_стоимости\_номера

ON Номера

AFTER INSERT, UPDATE

AS

BEGIN

IF EXISTS (SELECT \* FROM inserted WHERE Стоимость < 0)

BEGIN

RAISERROR('Стоимость номера не может быть отрицательной.', 16, 1)

ROLLBACK TRANSACTION

END

END;

go

CREATE TRIGGER Удаление\_связанных\_записей\_при\_удалении\_гостя

ON Гости

AFTER DELETE

AS

BEGIN

DELETE FROM Регистрация

WHERE id\_гостя IN (SELECT id\_гостя FROM deleted)

END;

go

CREATE TRIGGER Авто\_добавление\_услуги\_при\_добавлении\_номера

ON Номера

AFTER INSERT

AS

BEGIN

DECLARE @id\_номера INT

-- Получаем идентификатор добавленного номера

SET @id\_номера = (SELECT id\_номера FROM inserted)

-- Добавляем услугу по умолчанию для нового номера

INSERT INTO Номер\_услуга (id\_номера, id\_услуги)

VALUES (@id\_номера, (SELECT TOP 1 id\_услуги FROM Услуги\_в\_номере))

END;

Go

use Гостиница

go

CREATE PROCEDURE Добавить\_гостя

@ФИО varchar(50),

@Телефон\_гостя varchar(20),

@Город varchar(20),

@Номер\_паспорта varchar(30)

AS

BEGIN

INSERT INTO Гости (ФИО, Телефон\_гостя, Город, Номер\_паспорта)

VALUES (@ФИО, @Телефон\_гостя, @Город, @Номер\_паспорта)

END;

go

CREATE PROCEDURE Зарегистрировать\_гостя

@id\_гостя int,

@id\_сотрудника int,

@id\_номера int

AS

BEGIN

INSERT INTO Регистрация (id\_гостя, id\_сотрудника, id\_номера)

VALUES (@id\_гостя, @id\_сотрудника, @id\_номера)

END;

go

CREATE PROCEDURE Обновить\_информацию\_о\_номере

@id\_номера int,

@Этаж varchar(15),

@Класс varchar(15),

@Стоимость money

AS

BEGIN

UPDATE Номера

SET Этаж = @Этаж, Класс = @Класс, Стоимость = @Стоимость

WHERE id\_номера = @id\_номера

END;

Go

use Гостиница

go

CREATE VIEW Информация\_о\_гостях\_и\_номерах AS

SELECT

Гости.ФИО AS Гость,

Гости.Телефон\_гостя AS Телефон,

Номера.Этаж AS Этаж,

Номера.Класс AS Класс

FROM

Гости

JOIN

Регистрация ON Гости.id\_гостя = Регистрация.id\_гостя

JOIN

Номера ON Регистрация.id\_номера = Номера.id\_номера;

go

CREATE VIEW Список\_сотрудников\_и\_должностей AS

SELECT

Сотрудники.ФИО AS Сотрудник,

Сотрудники.Должность AS Должность

FROM

Сотрудники;

go

CREATE VIEW Информация\_о\_номерах\_и\_услугах AS

SELECT

Номера.id\_номера AS Номер,

Номера.Этаж AS Этаж,

Номера.Класс AS Класс,

Услуги\_в\_номере.Вид\_услуги AS Услуга,

Услуги\_в\_номере.Стоимость\_услуги AS Стоимость\_услуги

FROM

Номера

JOIN

Номер\_услуга ON Номера.id\_номера = Номер\_услуга.id\_номера

JOIN

Услуги\_в\_номере ON Номер\_услуга.id\_услуги = Услуги\_в\_номере.id\_услуги;

go

1. **Дайте определение структуры данных СУБД, общий подход к организации представлений, таблиц, индексов и кластеров.**

Структура данных в СУБД (системе управления базами данных) определяет организацию и хранение данных в базе данных. Она определяет формат данных, способ их хранения, связи между данными и доступ к ним. Структура данных включает в себя следующие элементы:

1. Таблицы: Таблица представляет собой структуру данных, состоящую из строк и столбцов.

2. Поля: Поля представляют собой отдельные элементы данных в таблице. Они определяют тип данных (например, целые числа, строки, даты), ограничения, форматирование и другие атрибуты данных.

3. Ключи: Ключи используются для уникальной идентификации записей в таблице. Основным ключом (Primary Key) является уникальный идентификатор каждой записи, который обеспечивает идентификацию и целостность данных. Вторичные ключи (Secondary Key) могут использоваться для быстрого доступа к данным по определенным критериям.

4. Связи: Связи определяют отношения и связи между таблицами. Они позволяют объединять данные из разных таблиц, основываясь на общих полях или ключах.

5. Индексы: Индексы создаются для оптимизации производительности поиска и сортировки данных. Они создаются на одном или нескольких полях таблицы и позволяют быстрый доступ к данным по этим полям.

Общий подход к организации представлений, таблиц, индексов и кластеров в базе данных включает следующие принципы:

1. Правильное проектирование схемы базы данных: перед созданием таблиц и других структур данных необходимо провести анализ требований и разработать соответствующую схему базы данных. Это включает определение сущностей, их атрибутов, связей и правил целостности данных.

2. Нормализация таблиц: Нормализация является процессом разделения таблиц на более мелкие и связанные сущности, чтобы устранить избыточность данных и обеспечить целостность. Нормализация помогает предотвратить дублирование данных и обеспечить эффективность операций обновления и запросов.

3. Создание индексов: Индексы используются для ускорения выполнения операций поиска и сортировки. Они создаются на ключевых полях таблицы, которые часто используются в запросах. Оптимальный подход к созданию индексов включает анализ запросов и выбор ключевых полей, на которых индексы будут наиболее эффективны.

4. Кластеризация данных: Кластеризация определяет физическую организацию данных на диске. Целью кластеризации является логическое и физическое размещение связанных данных рядом друг с другом. Кластеризация может быть основана на ключевых полях или других критериях. Оптимальное размещение данных на диске может существенно повысить производительность операций чтения и записи.

5. Эффективное использование представлений: Представления представляют собой виртуальные таблицы, которые основаны на данных из одной или нескольких таблиц. Они используются для упрощения доступа к данным и скрытия сложности физической структуры базы данных.