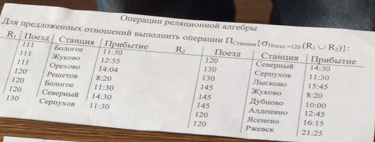
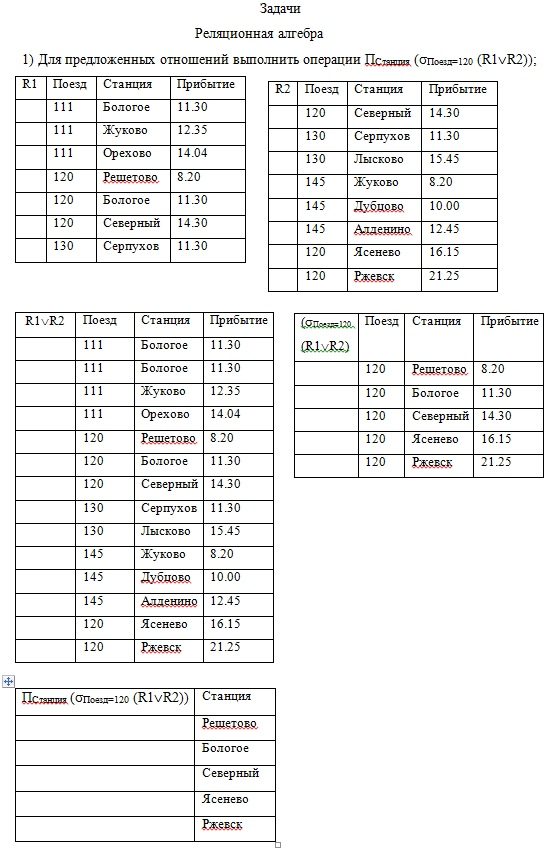
**Содержание**

1. Операции реляционной алгебры (1-5);
2. Триггеры (6-9);
3. Пользовательские функции (10-13);
4. Хранимые процедуры (14-17);
5. Курсоры (18-21);
6. Нормализация реляционных баз данных (22-25);
7. Построение ER-диаграмм (26-28);
8. Операторы создания таблиц (29-30);

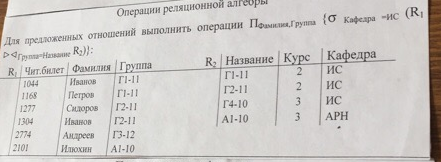
**1. Операции реляционной алгебры**



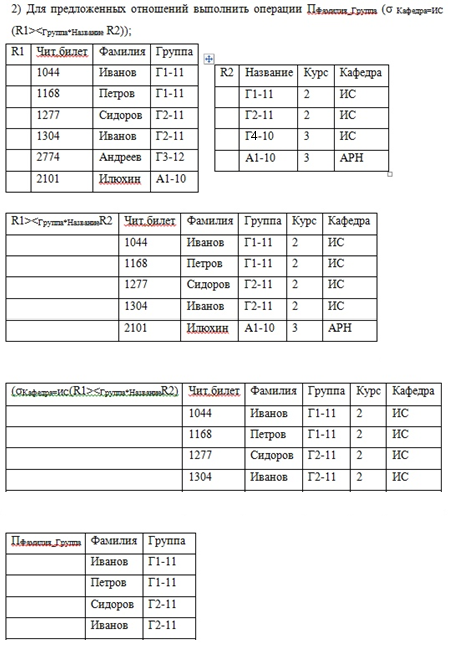
Решение



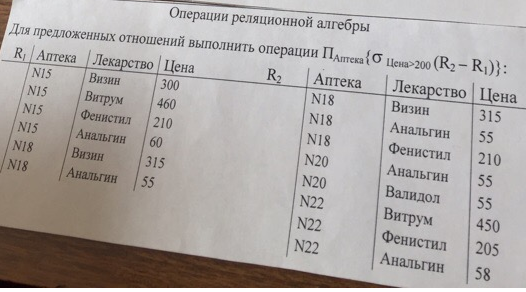
**2. Операции реляционной алгебры**

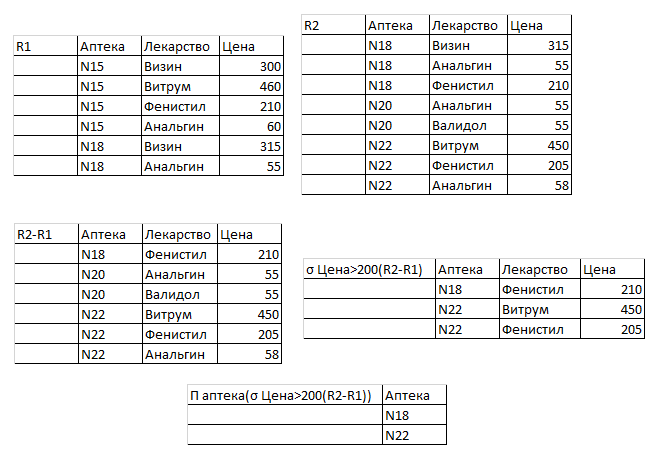


Решение

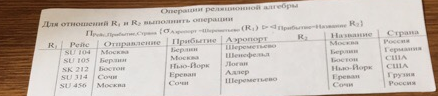


**3. Операции реляционной алгебры**

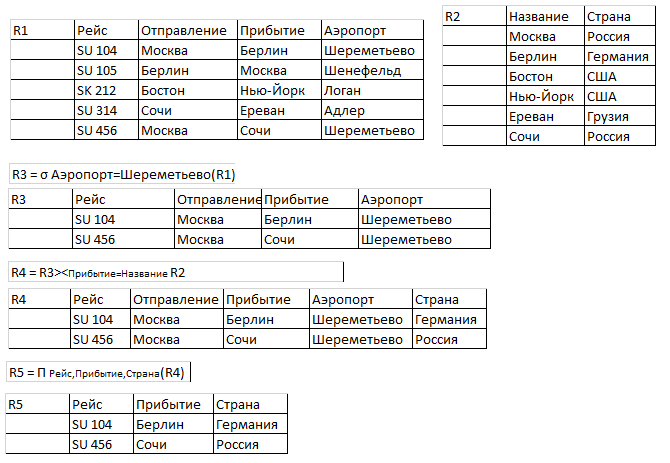




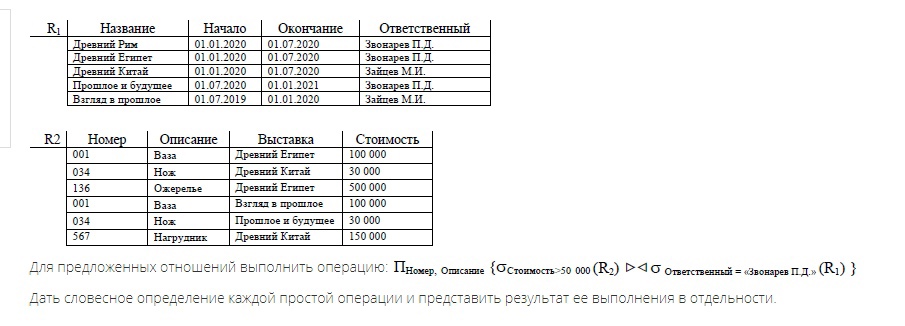
**4. Операции реляционной алгебры**

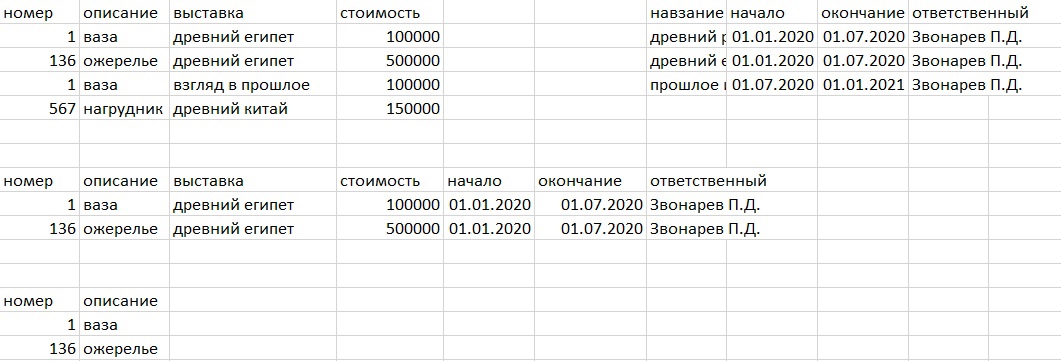


Решение

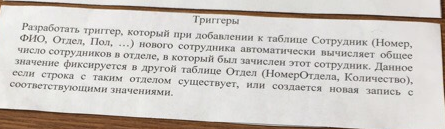


**5. Операции реляционной алгебры**





**6. Триггеры**



Решение

CREATE TRIGGER tr1

ON Сотрудник

AFTER INSERT

AS

BEGIN

DECLARE @dep INT, @kol INT

SELECT @dep = Отдел FROM inserted

SELECT @kol = COUNT(\*) FROM Сотрудник WHERE Отдел = @dep

IF NOT EXISTS (SELECT НомерОтдела FROM Отдел WHERE НомерОтдела = @dep)

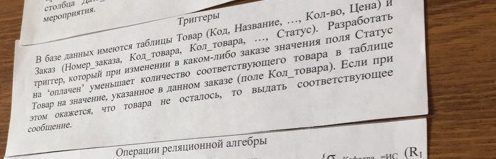
INSERT INTO Отдел VALUES (@dep, @kol)

ELSE

UPDATE Отдел SET Количество = @kol WHERE НомерОтдела = @dep

END

**7. Триггеры**



Решение

CREATE TRIGGER tr2

ON Заказ

AFTER UPDATE

AS

BEGIN

DECLARE [@st](https://vk.com/st) VARCHAR(20), [@kod](https://vk.com/kod) INT, @kskl INT, @kzak INT

SELECT [@st](https://vk.com/st) = Статус FROM inserted

IF [@st](https://vk.com/st) = 'Оплачен'

BEGIN

SELECT [@kod](https://vk.com/kod) = Код\_товара FROM inserted

SELECT @kskl = Кол-во FROM Товар WHERE Код = [@kod](https://vk.com/kod)

SELECT @kzak = Кол\_товара FROM Заказ WHERE Код\_товара = [@kod](https://vk.com/kod)

IF @kskl >= @kzak

UPDATE Товар SET Кол-во = @kskl - @kzak WHERE Код = [@kod](https://vk.com/kod)

ELSE

BEGIN

PRINT 'Недостаточно товара'

ROLLBACK TRANSACTION

RETURN

END

END

END

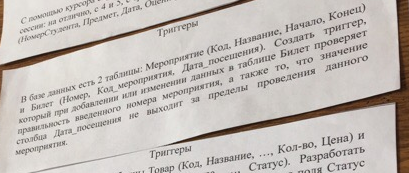
//@kod – код товара

//@kskl – количество товара на складе

//@kzak – количество товара в заказе

//@st – статус заказа

**8. Триггеры**



Решение

CREATE TRIGGER tr1

ON Билет

AFTER UPDATE, INSERT

AS

BEGIN

DECLARE @kod INT, @date DATE, @begin DATE, @end DATE

SELECT @kod = Код\_мероприятия FROM inserted

IF NOT EXISTS (SELECT \* FROM Мероприятие WHERE Код = @kod)

BEGIN

PRINT 'Неверный код мероприятия'

ROLLBACK TRAN

RETURN

END

SELECT @date = Дата\_посещения FROM inserted

IF NOT EXISTS (SELECT \* FROM Мероприятие WHERE @date BETWEEN Конец AND Начало AND Код = @kod)

BEGIN

PRINT 'Неверная дата'

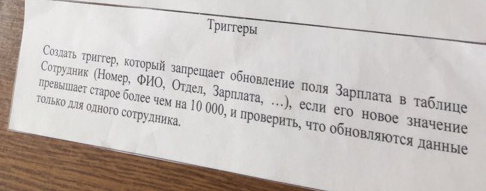
ROLLBACK TRAN

RETURN

END

END

**9. Триггеры**



Решение

CREATE TRIGGER tr1

ON Сотрудник

AFTER UPDATE

AS

BEGIN

DECLARE @salary INT, @count INT, @salarynew INT

SELECT @count = COUNT(\*) FROM inserted

IF @count>1

BEGIN

PRINT 'Нужно обновлять только одну строку'

ROLLBACK TRAN

RETURN

END

ELSE

BEGIN

SELECT @salary = Зарплата FROM deleted

SELECT @salarynew = Зарплата FROM inserted

IF ABS(@salarynew-@salary)>10000

BEGIN

PRINT 'Зарплата не должна быть увеличена больше, чем на 10000'

ROLLBACK TRAN

RETURN

END

END

END

**9.1. Триггеры**

В базе данных имеется следующий набор таблиц: Клиент (Номер, Фамилия), Товар (Код, Название, Кол-во, Цена), Заказ (Номерзаказа, Номерклиента, Кодтовара, Колтовара, Стоимость). Требуется разработать триггер типа INSTEAD OF, который при добавлении нового заказа проверяет:

добавление ТОЛЬКО ОДНОГО заказа;

СУЩЕСТВОВАНИЕ введенного товара в таблице Товар;

количество заказанного товара не должно превышать количества товара, имеющегося в наличии (данные в таблице Товар).

При невыполнении любого из условий выдать соответствующее сообщение.

CREATE TRIGGER TRIGGER1 ON ЗАКАЗ

INSTEAD OF INSERT

AS

BEGIN

DECLARE @ID ORDER INT, @ID TOVAR INT, @COUNT INT,

@COUNTTOVAR INT

SELECT @COUNT = COUNT(\*) FROM INSERTED

IF @COUNT>1

PRINT 'НЕЛЬЗЯ ДОБАВИТЬ БОЛЬШЕ ОДНОГО ЗАКАЗА ЗА РАЗ'

IF @COUNT = 1

BEGIN

IF EXISTS (SELECT \* FROM ТОВАР WHERE КОД = (SELECT КОД\_ТОВАРА FROM INSERTED))

BEGIN

IF (SELECT КОЛ\_ТОВАРА FROM INSERTED) <= (SELECT КОЛ-ВО FROM

ТОВАР WHERE КОД = (SELECT КОД\_ТОВАРА FROM INSERTED))

INSERT INTO ЗАКАЗ SELECT \* FROM INSERTED

ELSE

PRINT 'СЛИШКОМ МНОГО ТОВАРА'

END

ELSE

PRINT 'ТОВАР НЕ ЗАРЕГИСТРИРОВАН!'

END

**9.2. Триггеры**

В базе данных имеются таблицы Врач (ФИО, Год\_рождения, Специальность) и Вакансия (Специальность, Кол-во). Разработать триггер типа INSTEAD OF, который при добавлении нового врача:

проверяет добавление ТОЛЬКО ОДНОГО врача;

проверяет СУЩЕСТВОВАНИЕ вакансии по его специальности;

уменьшает на единицу число вакансий по данной специальности;

если при этом окажется, что число вакансий по данной специальности не осталось, то строка с данной специальностью должна быть удалена из таблицы Вакансия

При невыполнении любого из условий выдать соответствующее сообщение.

CREATE TRIGGER AddDoctor ON Врач

INSTEAD OF INSERT

AS

BEGIN

DECLARE @Count INT

SELECT @Count = Count(\*) FROM inserted

IF @Count > 1

PRINT 'Нельзя добавить больше одного врача за раз'

ELSE

BEGIN

DECLARE @Job VARCHAR(40), @fio VARCHAR(40), @god INT, @spec VARCHAR(40)

SELECT @Job = Специальность FROM inserted

IF NOT EXISTS (SELECT \*FROM Вакансия WHERE Специальность = @Job)

BEGIN

PRINT 'Не существует такой вакансии'

RETURN

END

ELSE

BEGIN

UPDATE Вакансия SET Кол-во = Кол-во - 1 WHERE Специальность = @Job

INSERT INTO Врач SELECT ФИО, Год\_рождения, Специальность FROM inserted

IF (SELECT Кол-во FROM Вакансия WHERE Специальность = @Job) = 0

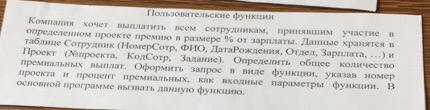
DELETE FROM Вакансия WHERE Специальность = @Job

END

END

END

**10. Пользовательские функции**



Решение

Создание функции:

CREATE FUNC f1

(@prj INT, @percent REAL)

RETURNS REAL

AS

BEGIN

DECLARE @prem\_sum REAL

SELECT @prem\_sum = SUM(Зарплата\*@percent) FROM Сотрудник

WHERE НомерСотр IN (SELECT КодСотр FROM Проект

WHERE №проекта = @prj)

RETURN @prem\_sum

END

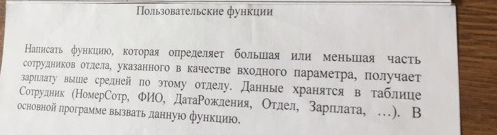
Вызов функции:

DECLARE @res REAL

SET @res = dbo.f1(1, 0.15)

PRINT 'премия составила: ' + CAST(@res AS VARCHAR(20))

**11. Пользовательские функции**



Решение

Создание функции:

CREATE FUNC f1

(@otd INT)

RETURNS INT

AS

BEGIN

DECLARE @sr INT, @low INT, @hig INT, @res INT

SELECT @sr = AVG(Зарплата) FROM Сотрудник

WHERE Отдел = @otd

SELECT @low = COUNT(\*) FROM Сотрудник

WHERE Отдел = @otd AND Зарплата <= @sr

SELECT @hig = COUNT(\*) FROM Сотрудник

WHERE Отдел = @otd AND Зарплата > @sr

IF @low > @hig

SET @res = 0 // большая часть получает зп ниже средней

IF @hig > @low

SET @res = 1 //большая часть получает зп выше средней

IF @low = @hig

SET @res = 2 //одинаково

RETURN @res

END

Вызов функции:

DECLARE @res INT

SET @res = dbo.f1(1)

IF @res = 0

PRINT 'меньшая часть получает зп выше средней'

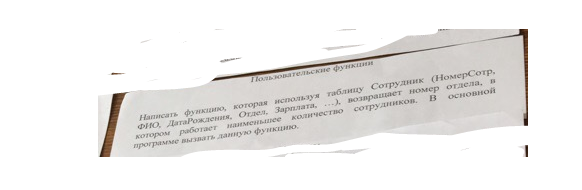
IF @res = 1

PRINT 'большая часть получает зп выше средней'

IF @res = 2

PRINT 'одинаково'

**12. Пользовательские функции**



Решение

Создание функции:

CREATE FUNC

f1()

RETURNS INT

AS

BEGIN

DECLARE @temp TABLE(otd INT, count\_sotr INT)

INSERT INTO @temp SELECT Отдел, COUNT(\*) FROM Сотрудник GROUP BY Отдел

DECLARE @min INT, @res INT

SELECT @min = MIN(count\_sotr) FROM @temp

SELECT @res = Отдел FROM @temp WHERE count\_sotr = @min

RETURN @res

END

Вызов функции:

DECLARE @res INT

SET @res = dbo.f1()

PRINT @res

**13. Пользовательские функции**

Данные о сотрудниках хранятся в таблице Сотрудник (Номер, Фамилия, Год\_рождения, Пол, Отдел, Зарплата). Написать функцию, возвращающую номер отдела, в котором работает наибольшее количество сотрудников, старше указанного возраста. В основной программе вызвать данную функцию.

CREATE FUNCTION Tsk1 (@TrVosrast int)

RETURNS INT

AS

BEGIN

DECLARE @otdel INT, @max INT

DECLARE @nazvosrast INT

SET @nazvosrast = YEAR(GetDate()) - @TrVosrast

DECLARE @temp TABLE(otdel INT, count\_sotr INT)

INSERT INTO @temp SELECT Отдел, COUNT(\*) FROM Сотрудник

WHERE Год\_рождения < @nazvosrast GROUP BY Отдел))

SELECT @max = MAX(count\_sotr) FROM @temp

SELECT @otdel = otdel FROM @temp WHERE count\_sotr = @max

RETURN @otdel

END

DECLARE @res INT

SET @res = dbo.Tsk1(21)

PRINT 'Номер отдела, в котором работает наибольшее количество сотрудников, рожденных ранее указанного возраста: ' + CAST (@res AS VARCHAR(10))

**13.1. Пользовательские функции**

Данные о врачах хранятся в таблице Врач (ФИО, Год\_рождения, Пол, Специальность, Зарплата). Написать функцию, которая возвращает ту специальность, по которой работает наибольшее число врачей клиники. В основной программе вызвать данную функцию.

CREATE FUNCTION Vrach()

RETURNS VARCHAR(50)

AS

BEGIN

DECLARE @special VARCHAR(50), @max INT

DECLARE @temp TABLE(SPECIAL VARCHAR(50), kol INT)

INSERT INTO @temp SELECT Специальность, COUNT(\*) FROM Врач GROUP BY Специальность))

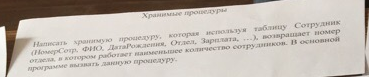
SELECT @max = MAX(kol) FROM @temp

SELECT @special = SPECIAL FROM @temp WHERE kol = @max

RETURN @special

END

**14. Хранимые процедуры**



Решение

Создание процедуры:

CREATE PROC pr

@res INT OUTPUT

AS

BEGIN

DECLARE @temp TABLE(otd INT, count\_sotr INT)

INSERT INTO @temp SELECT Отдел, COUNT(\*) FROM Сотрудник GROUP BY Отдел

DECLARE @min INT

SELECT @min = MIN(count\_sotr) FROM @temp

SELECT @res = Отдел FROM @temp WHERE count\_sotr = @min

END

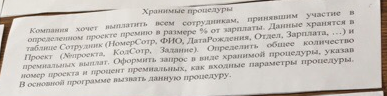
Вызов процедуры:

DECLARE @result INT

EXEC pr @result OUTPUT

PRINT @res

**15. Хранимые процедуры**



Решение

Создание процедуры:

CREATE PROC pr

@prj INT,

@percent REAL,

@prem\_sum REAL OUTPUT

AS

BEGIN

SELECT @prem\_sum = SUM(Зарплата\*@percent) FROM Сотрудник

WHERE НомерСотр IN (SELECT КодСотр FROM Проект

WHERE №проекта = @prj)

END

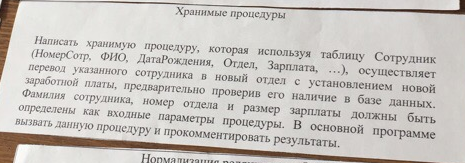
Вызов процедуры:

DECLARE @res REAL

EXEC pr 1, 0.15, @res OUTPUT

PRINT 'Премия составила: ' + CAST(@res AS VARCHAR(20))

**16. Хранимые процедуры**



Решение

Создание процедуры:

CREATE PROC pr

@fam VARCHAR(20),

@otd INT,

@salary INT

AS

BEGIN

IF EXISTS (SELECT \* FROM Сотрудник WHERE ФИО = @fam)

UPDATE Сотрудник SET Отдел = @otd, Зарплата = @salary

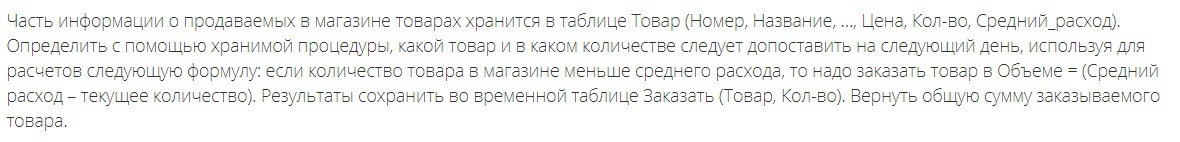
WHERE ФИО = @fam

END

Вызов процедуры:

EXEC pr ‘Степаненков И.И.’, 2, 5000

**17. Хранимые процедуры**



CREATE PROCEDURE Tsk1

@sum INT OUTPUT

AS

BEGIN

DECLARE @temp TABLE(Tovar VARCHAR(50), Kol INT)

INSERT INTO @temp SELECT Номер,

(Средний\_расход - Кол-во) FROM Товар WHERE Кол-во < Средний\_расход

SELECT @sum = SUM(Kol\*Цена) FROM @temp INNER JOIN Товар

ON @temp.Tovar = Товар.Номер

END

Вызов процедуры:

DECLARE @res INT

EXEC Tsk1 @res OUT

PRINT @res

**17.1**

Данные о проектах хранятся в таблицах Проект (Номерсотрудника, Кодпроекта, Задание) и ПроектДоп (Код, Название, Начало, Конец, …). Написать хранимую процедуру, которая вычисляет, сколько сотрудников участвовали в самом длительном проекте. В основной программе вызвать данную процедуру.

CREATE PROC pr

@chet INT OUTPUT

AS

BEGIN

DECLARE @chet1 INT

SELECT @chet = MAX(Конец - Начало) FROM ПроектДоп

SELECT @chet1 = Код FROM ПроектДоп WHERE (Конец - Начало) = @chet

SELECT @chet = COUNT(DISTINCT(Номерсотрудника)) FROM Проект WHERE Кодпроекта = @chet1

END

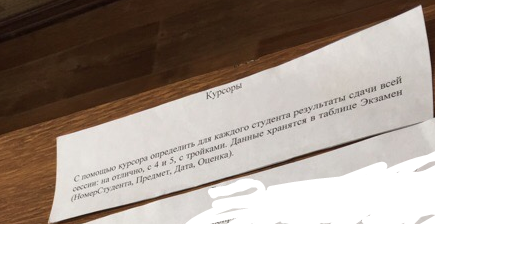
Вызов процедуры:

DECLARE @res INT

EXEC pr @res OUT

PRINT @res

**18. Курсоры**



Решение

DECLARE StudCursor CURSOR

LOCAL

FORWARD\_ONLY

STATIC

FOR SELECT НомерСтудента, MIN(Оценка) FROM Экзамен GROUP BY НомерСтудента

OPEN StudCursor

DECLARE @StudID int, @Mark int

FETCH NEXT FROM StudCursor INTO @StudID, @Mark

DECLARE @Cashe VARCHAR(20)

WHILE @@FETCH\_STATUS = 0

BEGIN

SET @Cashe = CAST(@StudID AS VARCHAR(10) ) + ' - '+

CASE

WHEN @Mark >= 45 THEN 'Отлично'

WHEN @Mark BETWEEN 35 AND 44 THEN 'Хорошо'

WHEN @Mark BETWEEN 25 AND 34 THEN 'Удовлетворительно'

ELSE 'Неудовлетворительно'

END

PRINT @Cashe

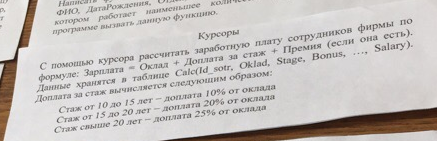
FETCH NEXT FROM ClientCursor INTO @name, @fio

END

CLOSE StudCursor

DEALLOCATE StudCursor

**19. Курсоры**



Решение

DECLARE calc CURSOR

LOCAL

FORWARD\_ONLY

DYNAMIC

OPTIMISTIC

FOR SELECT Oklad, Stage, Bonus, Salary FROM dbo.Calc

OPEN calc

DECLARE @okl INT, @stage INT, @bonus INT, @salary INT

FETCH NEXT FROM calc INTO @okl, @stage, @bonus, @salary

WHILE @@FETCH\_STATUS = 0

BEGIN

SET @salary = @okl +

CASE

WHEN @stage BETWEEN 10 AND 15 THEN (@okl \* 0.1)

WHEN @stage BETWEEN 16 AND 20 THEN (@okl \*0.2)

WHEN @stage > 20 THEN (@okl \* 0.25)

ELSE 0

END

SET @salary = @salary +

CASE

WHEN @ bonus IS NOT NULL THEN @bonus

ELSE 0

END

UPDATE dbo.Calc SET Salary = @salary WHERE CURRENT OF calc

FETCH NEXT FROM calc INTO @okl, @stage, @bonus, @salary

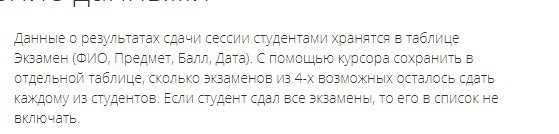
END

CLOSE calc

DEALLOCATE calc

SELECT \* FROM dbo.Calc

**20. Курсоры**



ВАРИАНТ 1 (через COUNT)

DECLARE task CURSOR

LOCAL

FORWARD\_ONLY

DYNAMIC

OPTIMISTIC

FOR SELECT FIO, COUNT(Predmet) FROM Ekzamen GROUP BY FIO

OPEN task

DECLARE @temp TABLE (FIO VARCHAR(50), SCHET INT)

DECLARE @fio VARCHAR(50), @count INT

FETCH NEXT FROM task INTO @fio, @count

WHILE @@FETCH\_STATUS = 0

BEGIN

IF (@count < 4) INSERT INTO @temp VALUES (@fio, @count)

FETCH NEXT FROM task INTO @fio, @count

END

CLOSE task

DEALLOCATE task

SELECT \* FROM @temp

ВАРИАНТ 2 (не через COUNT, лучше он, чем первый)

DECLARE task CURSOR

LOCAL

FORWARD\_ONLY

DYNAMIC

OPTIMISTIC

FOR SELECT FIO FROM Ekzamen

OPEN task

DECLARE @temp TABLE (FIO VARCHAR(50), ExamCount INT)

DECLARE @fio VARCHAR(50)

FETCH NEXT FROM task INTO @fio

WHILE @@FETCH\_STATUS = 0

BEGIN

INSERT INTO @temp VALUES (CASE

WHEN NOT EXISTS(SELECT FIO FROM @temp WHERE FIO = @fio) THEN @fio

ELSE 'Added'

END, 0)

UPDATE @temp SET ExamCount = ExamCount + 1 WHERE FIO = @fio

DELETE FROM @temp WHERE FIO = 'Added'

FETCH NEXT FROM task INTO @fio

END

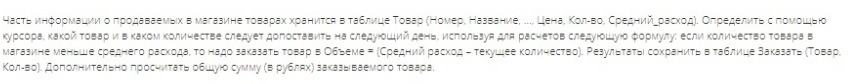
DELETE FROM @temp WHERE ExamCount = 4

CLOSE task

DEALLOCATE task

SELECT FIO, ExamCount FROM @temp

**21. Курсоры**



DECLARE task CURSOR

LOCAL

FORWARD\_ONLY

DYNAMIC

OPTIMISTIC

FOR SELECT Номер, Название, Цена, Количество, Средний\_расход FROM Товар

OPEN task

DECLARE @Zakaz TABLE (Товар VARCHAR(100), Количество INT)

DECLARE @tovar VARCHAR(100), @nazvanie VARCHAR(50), @kolichestvo INT, @nomer INT, @Цена INT, @sredniiRashod INT, @All\_Sum INT

SET @All\_Sum = 0

FETCH NEXT FROM task INTO @nomer, @nazvanie, @Цена, @kolichestvo, @sredniiRashod

WHILE @@FETCH\_STATUS = 0

BEGIN

SET @tovar = '№' + CAST(@nomer AS VARCHAR(50)) + ' - ' + @nazvanie

INSERT INTO @Zakaz VALUES (@tovar, CASE

WHEN @kolichestvo < @sredniiRashod THEN @sredniiRashod - @kolichestvo

ELSE 0

END)

SET @All\_Sum = @All\_Sum + (SELECT Количество FROM @Zakaz WHERE Товар = @tovar) \* @Цена

FETCH NEXT FROM task INTO @nomer, @nazvanie, @Цена, @kolichestvo, @sredniiRashod

END

DELETE FROM @Zakaz WHERE Количество = 0

CLOSE task

DEALLOCATE task

SELECT Товар, Количество FROM @Zakaz

PRINT 'Общая стоимость заказа = ' + CAST(@All\_Sum AS VARCHAR(50))

**21.1. Курсоры**

Данные о врачах клиники хранятся в таблице Врач (ФИО, Год\_рождения, Пол, Специальность). С помощью курсора составить список врачей-женщин терапевтов, которым до выхода на пенсию (по достижению 55 лет) отсталость менее 2-х лет.

DECLARE Doctor CURSOR

LOCAL

FORWARD\_ONLY

DYNAMIC

OPTIMISTIC

FOR SELECT ФИО, Год\_Рождения FROM Врач

OPEN Doctor

DECLARE @table TABLE (Fio varchar(50), Birth INT)

DECLARE @FIO VARCHAR(50), @Year\_Birth INT

FETCH NEXT FROM Doctor INTO @FIO, @Year\_Birth

WHILE @@FETCH\_STATUS = 0

BEGIN

IF (SELECT Пол FROM Врач WHERE ФИО = @FIO AND Год\_Рождения = @Year\_Birth) = 'ж' AND

(SELECT Специальаность FROM Врач WHERE ФИО = @FIO AND Год\_Рождения = @Year\_Birth) = 'Терапефт' AND

(YEAR(GetDate()) -(SELECT Год\_Рождения FROM Врач WHERE ФИО = @FIO AND Год\_Рождения = @Year\_Birth)) < 55 AND

(YEAR(GetDate()) -(SELECT Год\_Рождения FROM Врач WHERE ФИО = @FIO AND Год\_Рождения = @Year\_Birth)) > 53

BEGIN

INSERT INTO @table VALUES (@FIO, @Year\_Birth)

END

FETCH NEXT FROM Doctor INTO @FIO, @Year\_Birth

END

SELECT \* FROM @table

CLOSE Doctor

DEALLOCATE Doctor

**21.2 Курсоры**

Данные о результатах сдачи сессии студентами хранятся в таблице Экзамен (ФИО, Предмет, Балл, Дата). С помощью курсора составить список неуспевающих студентов, с указанием количества двоек, полученных им на экзаменах.

DECLARE bad\_student CURSOR LOCAL FORWARD\_ONLY STATIC

FOR SELECT ФИО, Балл FROM Экзамен

OPEN bad\_student

-- определить таблицу для хранения результатов

DECLARE @res\_table TABLE (fio VARCHAR(50), res INT)

-- объявление локальных переменных для FETCH

DECLARE @fio VARCHAR(50), @ball INT, @x INT

SET @x = 1

FETCH NEXT FROM bad\_student INTO @fio, @ball

WHILE @@FETCH\_STATUS = 0

BEGIN

IF (@ball = 2)

BEGIN

IF NOT EXISTS (SELECT fio FROM @res\_table WHERE fio = @fio)

INSERT INTO @res\_table VALUES (@fio, @x)

ELSE

UPDATE @res\_table SET res = res + 1 WHERE fio = @fio

END

FETCH NEXT FROM bad\_student INTO @fio, @ball

END

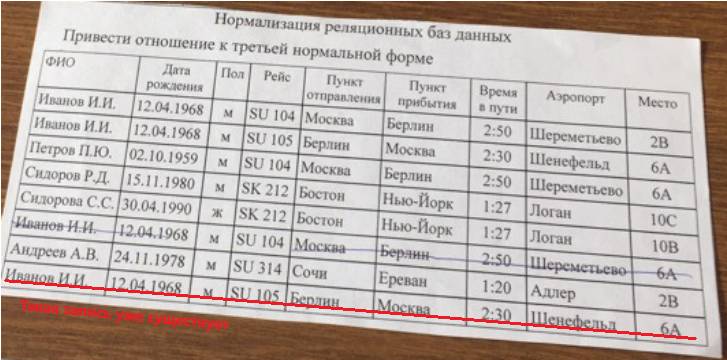
-- вывести результаты

SELECT \* FROM @res\_table

CLOSE bad\_student

DEALLOCATE bad\_student

**22. Нормализация реляционных баз данных**



Решение

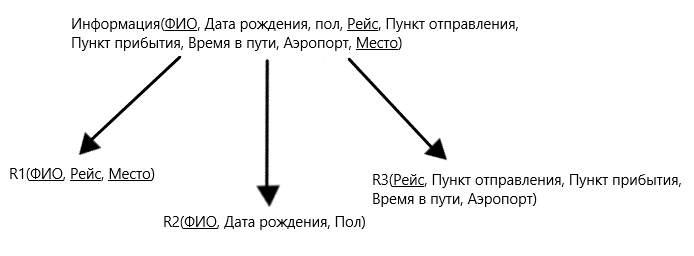
Первичный ключ(ФИО, Рейс, Место)

ФИО -> Дата рождения, пол

Рейс -> Пункт отправления, пункт прибытия, время в пути, аэропорт

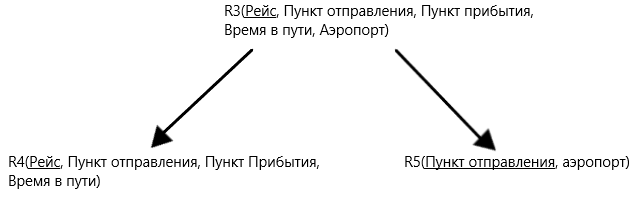
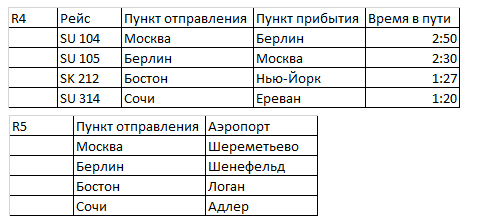
Аэропорт -> Пункт отправления

К 2НФ:



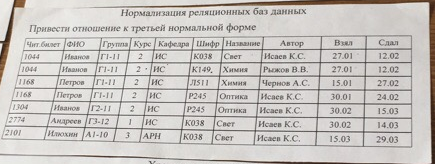


К 3НФ:

Итоговые таблицы: R1, R2, R4, R5

**23. Нормализация реляционных баз данных**



Решение

Первичный ключ(Чит. Билет, Шифр)

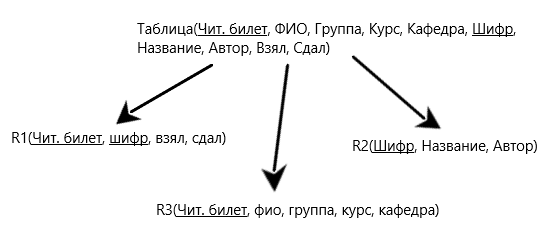
Чит. Билет, Шифр -> взял, сдал

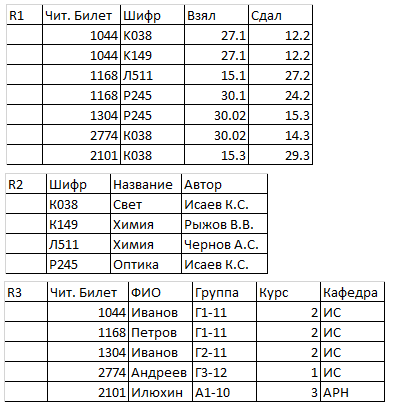
Чит. Билет -> фио, группа, курс, кафедра

Группа -> курс, кафедра

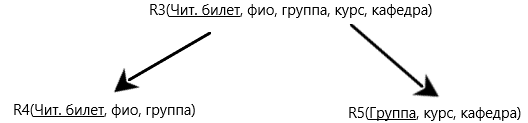
Шифр -> название, автор

К 2НФ:





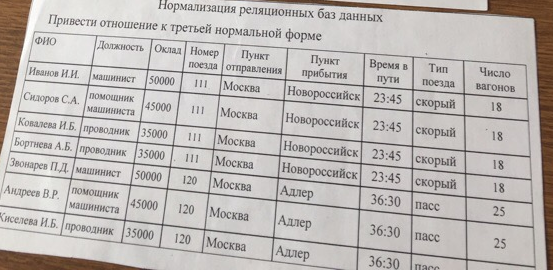
К 3НФ:





Итоговые таблицы: R1, R2, R4, R5

**24. Нормализация реляционных баз данных**



Решение

Первичный ключ(ФИО)

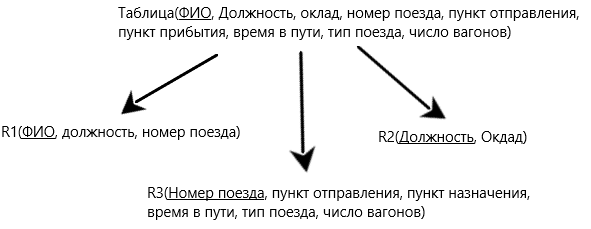
ФИО -> Должность, Оклад, номер поезда, пункт отправления, пункт прибытия, время в пути, тип поезда, число вагонов

Должность -> оклад

Номер поезда -> пункт отправления, пункт назначения, время в пути, тип поезда, число вагонов

БД уже находится во 2НФ, т.к. каждый не ключевой атрибут функционально полно зависит от первичного ключа.

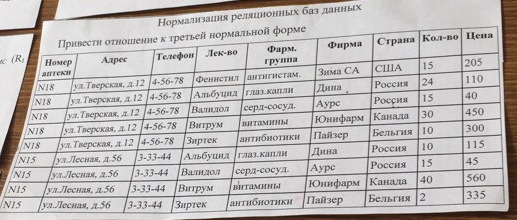
К 3НФ:





Итоговые таблицы: R1, R2, R3

**25. Нормализация реляционных баз данных**



Решение

Первичный ключ(Номер аптеки, лекарство)

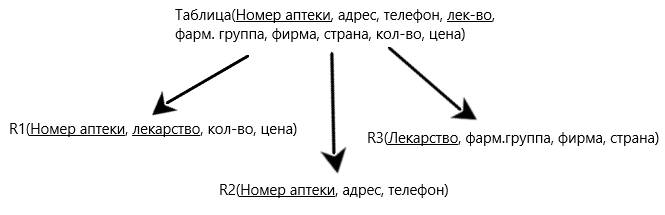
Номер аптеки, лекарство -> Адрес, телефон, лек-во, фарм. Группа, фирма, Страна, кол-во, цена

Номер аптеки -> адрес, телефон

Лекарство -> фарм. Группа, фирма, страна

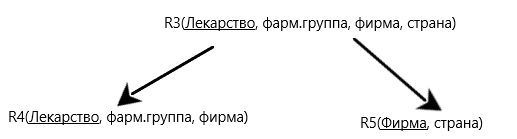
Фирма -> страна

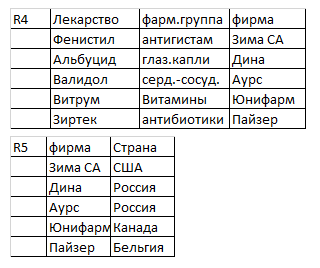
К 2 НФ:





К 3НФ:





**25.1.** **Нормализация реляционных баз данных**



1. 2НФ – ?

K = {Начало проведения, Место}

Начало проведения 🡪 Соревнование, Страна проведения, Место проведения, Год

Место проведения 🡪 Страна проведения

ФИО атлета 🡪 Страна, Дата рождения

Атрибуты: Соревнование, Страна проведения, Место проведения, Год частично зависят от первичного ключа, поэтому она не находится в 2НФ

R1 (Начало проведения, Место, ФИО атлета, Страна, Дата рождения)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Начало проведения | Место | ФИО атлета | Страна | Дата рождения |
| 08.03 2019, 18:15 | 4 | OEBERG Hanna | Швеция | 02.11.1995 |
| 08.03 2019, 18:15 | 1 | KUZMINA Anastasiya | Словакия | 28.08.1984 |
| 08.03 2019, 18:15 | 3 | DAHLMEIER Laura | Германия | 22.08.1993 |
| 08.03 2019, 18:15 | 8 | YURLOVA Ekaterina | Россия | 28.08.1984 |
| 17.03 2019, 15:15 | 4 | OEBERG Hanna | Швеция | 02.11.1995 |
| 17.03 2019, 15:15 | 28 | KUZMINA Anastasiya | Словакия | 28.08.1984 |
| 17.03 2019, 15:15 | 6 | DAHLMEIER Laura | Германия | 22.08.1993 |
| 17.03 2019, 15:15 | 2 | YURLOVA Ekaterina | Россия | 28.08.1984 |
| 10.02 2017, 16:45 | 2 | DAHLMEIER Laura | Германия | 22.08.1993 |
| 10.02 2017, 16:45 | 8 | KUZMINA Anastasiya | Словакия | 28.08.1984 |

R2 (Год, Место проведения, Страна проведения, Соревнование, Начало проведения)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Год | Место проведения | Страна проведения | Соревнование | Начало проведения |
| 2019 | Эстерсунд | Швеция | Спринт (женщины) | 08.03 2019, 18:15 |
| 2019 | Эстерсунд | Швеция | Масс-старт (женщины) | 17.03 2019, 15:15 |
| 2017 | Хохфильцен | Австрия | Спринт (женщины) | 10.02 2017, 16:45 |

1. 3НФ - ?

В R1 и R2 есть транзитивная зависимость

Начало проведения 🡪 Место проведения 🡪 Страна проведения

Начало проведение, Место 🡪 ФИО атлета 🡪 Страна, Дата рождения

R3 (Год, Место проведения, Соревнование, Начало проведения)

|  |  |  |  |
| --- | --- | --- | --- |
| Год | Место проведения | Соревнование | Начало проведения |
| 2019 | Эстерсунд | Спринт (женщины) | 08.03 2019, 18:15 |
| 2019 | Эстерсунд | Масс-старт (женщины) | 17.03 2019, 15:15 |
| 2017 | Хохфильцен | Спринт (женщины) | 10.02 2017, 16:45 |

R4 (Место проведения, Страна проведения)

|  |  |
| --- | --- |
| Место проведения | Страна проведения |
| Эстерсунд | Швеция |
| Хохфильцен | Австрия |

R5 (Начало проведение, Место, ФИО атлета)

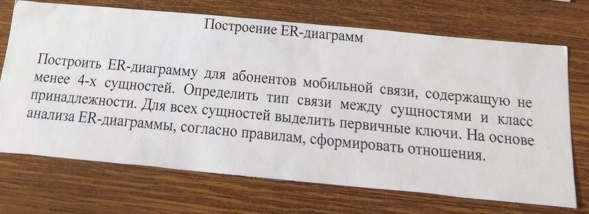
|  |  |  |
| --- | --- | --- |
| Начало проведения | Место | ФИО атлета |
| 08.03 2019, 18:15 | 4 | OEBERG Hanna |
| 08.03 2019, 18:15 | 1 | KUZMINA Anastasiya |
| 08.03 2019, 18:15 | 3 | DAHLMEIER Laura |
| 08.03 2019, 18:15 | 8 | YURLOVA Ekaterina |
| 17.03 2019, 15:15 | 4 | OEBERG Hanna |
| 17.03 2019, 15:15 | 28 | KUZMINA Anastasiya |
| 17.03 2019, 15:15 | 6 | DAHLMEIER Laura |
| 17.03 2019, 15:15 | 2 | YURLOVA Ekaterina |
| 10.02 2017, 16:45 | 2 | DAHLMEIER Laura |
| 10.02 2017, 16:45 | 8 | KUZMINA Anastasiya |

R6 (ФИО атлета, Страна, Дата рождения)

|  |  |  |
| --- | --- | --- |
| ФИО атлета | Страна | Дата рождения |
| OEBERG Hanna | Швеция | 02.11.1995 |
| KUZMINA Anastasiya | Словакия | 28.08.1984 |
| DAHLMEIER Laura | Германия | 22.08.1993 |
| YURLOVA Ekaterina | Россия | 28.08.1984 |

Ответ: R3 (Год, Место проведения, Соревнование, Начало проведения), R4 (Место проведения, Страна проведения), R5 (Начало проведение, Место, ФИО атлета), R6 (ФИО атлета, Страна, Дата рождения)

**26. Построение ER-диаграмм**



Решение



Связь заключает имеет тип 1:М, так как каждый абонент может заключить один или несколько договоров, а конкретный договор может быть заключен только с конкретным абонентом. Сущность абонент имеет обязательный класс принадлежности, так как предполагаем, что он обязательно обратился к оператору для заключения договора. Сущность договор имеет обязательный класс принадлежности, так как он обязательно должен быть заключен с каким-либо абонентом.



Связь заключает имеет тип 1:М, т.к. сотрудник может заключить множество договоров, а конкретный договор может быть заключен конкретным сотрудником. Сущность сотрудник имеет необязательный класс принадлежности, т.к. он мог не участвовать в заключении договоров (например, будучи уборщиком). Сущность договор имеет обязательный класс принадлежности, т.к. обязательно должен быть кем-то заключен.



Связь содержит имеет тип М:1, т.к. в договоре может быть указан только один тариф, а какой-либо тариф может содержаться во множестве договоров. Сущность договор имеет обязательный класс принадлежности, т.к. в нем обязательно должен содержаться тариф. Сущность тариф имеет необязательный класс принадлежности, т.к. может пока не содержаться ни в одном договоре.

Связь 1 удовлетворяет условиям правила 4, в соответствии с которым получаем 2 отношения:

1. Абонент (КодАбонента, …)

2. Договор (КодДоговора, КодАбонента, …) – добавился неключевой атрибут КодАбонента.

Связь 2 удовлетворяет условиям правила 4, в соответствии с которым получаем 2 отношения:

1. Сотрудник (КодСотрудника, …)

2. Договор (КодДоговора, КодСотрудника, …) – добавился неключевой атрибут КодСотрудника.

Связь 3 удовлетворяет условиям правила 4, в соответствии с которым получаем 2 отношения:

1. Тариф (КодТарифа, …)

2. Договор (КодДоговора, КодТарифа, …) – добавился неключевой атрибут КодТарифа.

Добавим неключевые атрибуты в каждое из предварительных отношений:

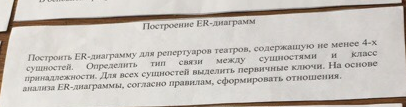
1. Абонент (КодАбонента, ФИО, ПаспортныеДанные, ДатаРождения)

2. Сотрудник (КодСотрудника, ФИО, Должность, Телефон)

3. Тариф (КодТарифа, НазваниеТарифа, СистемаОплаты, СтоимостьПодключения, АбонентскаяПлата)

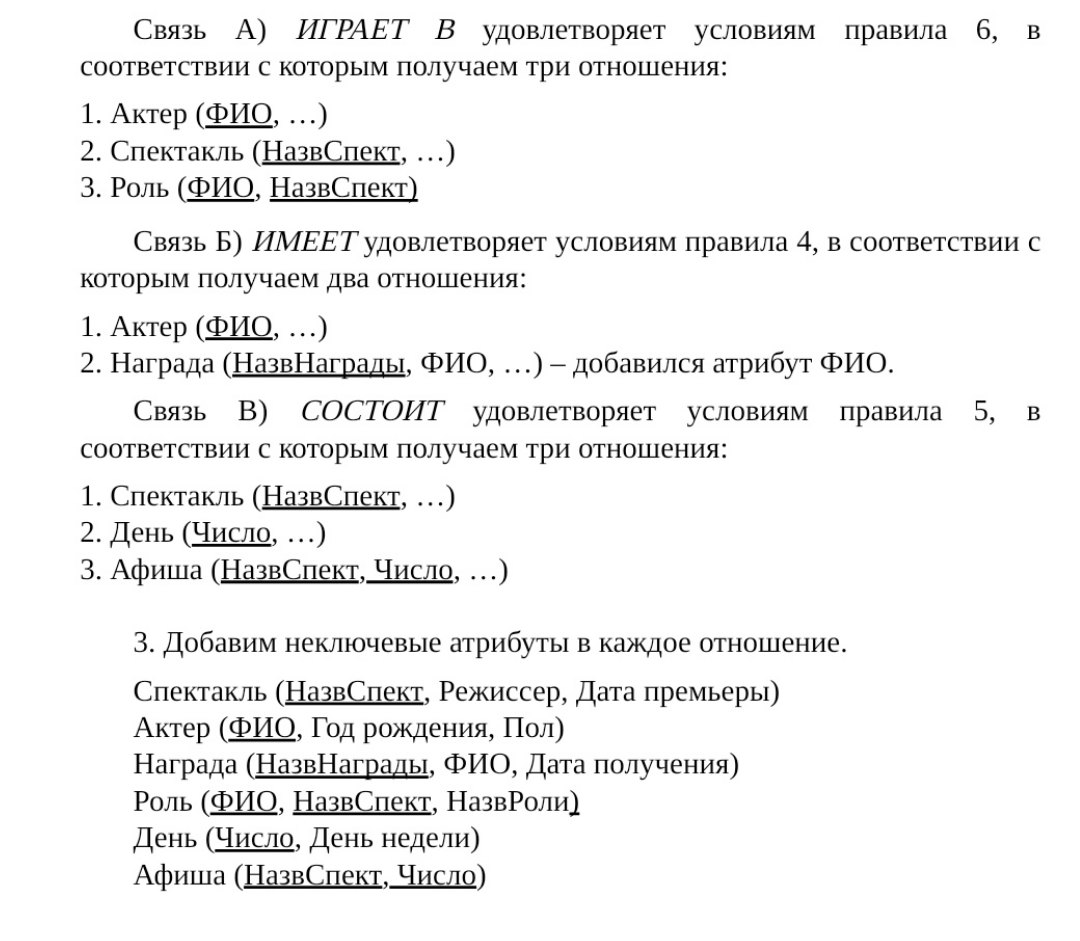
4. Договор (КодДоговора, КодАбонента, КодСотрудника, КодТарифа, НомерТелефона, ДатаЗаключения, СрокДействия)

**27. Построение ER-диаграмм**

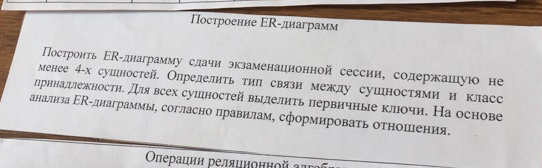


Решение

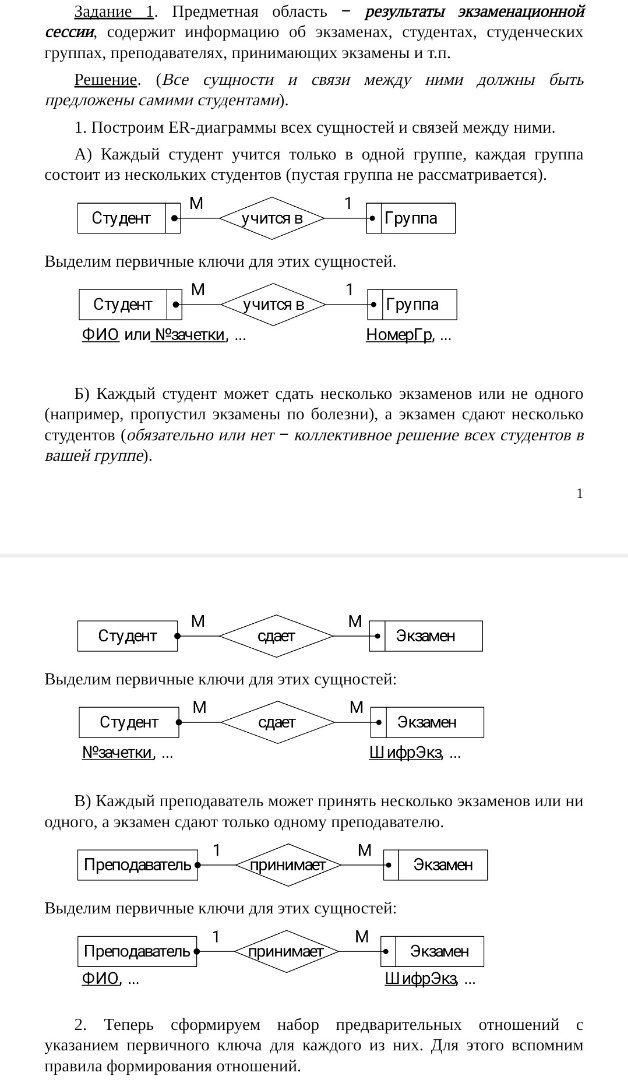




**28. Построение ER-диаграмм**

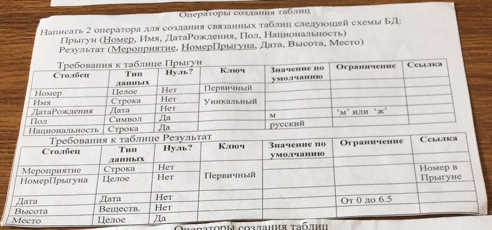


Решение

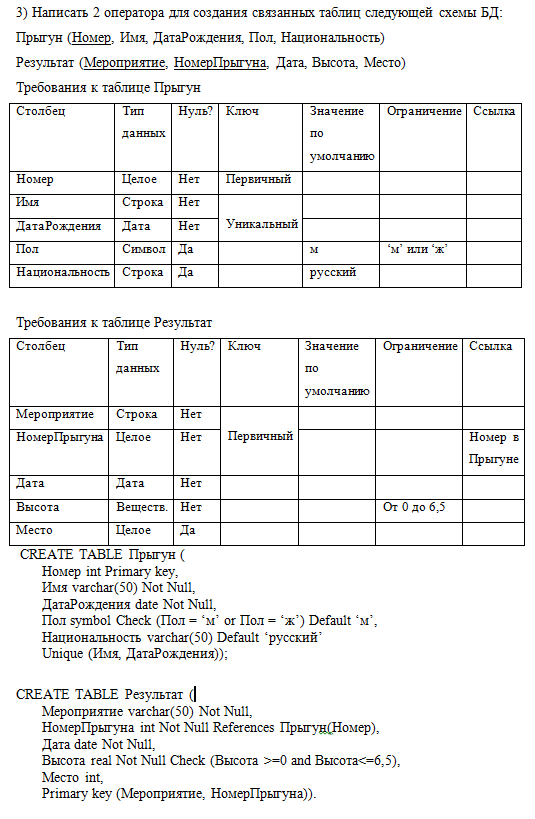




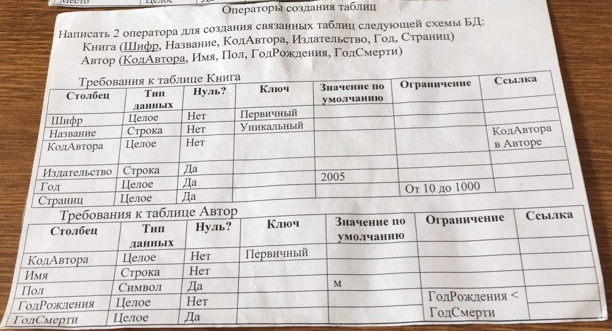
**29. Операторы создания таблиц**



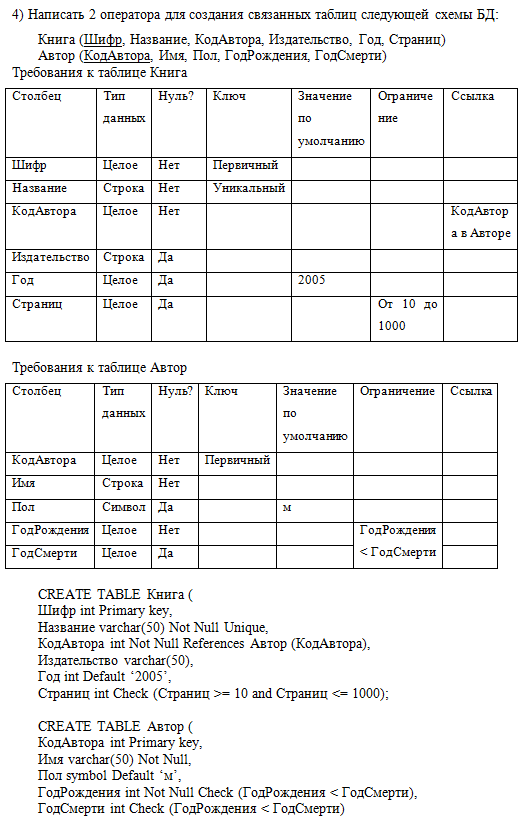
Решение



**30. Операторы создания таблиц**



Решение



КУРСОР

Используя курсор, определите студентов указанной группы, имеющих высокий средний балл по результатам экзаменов

USE Zalivalov

DECLARE average\_cursor CURSOR

LOCAL FORWARD\_ONLY STATIC

FOR SELECT Id\_Student, FIO FROM Student

DECLARE @Id\_Student INT, @FIO VARCHAR(50)

DECLARE @res INT, @n INT

SET @n = 10

-- определить таблицу для хранения результатов

DECLARE @TOP\_STUDENTS TABLE (Id\_Student INT, FIO VARCHAR(50), res INT)

OPEN average\_cursor

FETCH NEXT FROM average\_cursor INTO @Id\_Student, @FIO

IF @n = 0

PRINT 'Вы задали значение нуль. Задайте значение >0'

WHILE @@FETCH\_STATUS = 0

BEGIN

SET @res = (SELECT AVG(Mark) FROM Exam WHERE Id\_Student = @Id\_Student GROUP BY Id\_Student)

INSERT INTO @TOP\_STUDENTS VALUES (@Id\_Student, @FIO, @res)

FETCH NEXT FROM average\_cursor INTO @Id\_Student, @FIO

END

SELECT FIO, res FROM @TOP\_STUDENTS ORDER BY res DESC OFFSET 0 ROWS FETCH NEXT @n ROWS ONLY

CLOSE average\_cursor

DEALLOCATE average\_cursor

КУРСОР

Руководство института хочет выплатить всем преподавателя, принимавшим экзамены, выплатить премию, которая зависит от стажа следующим образом: стаж от 5 до 10 лет − премия составляет 4000 руб. стаж от 11 до 20 лет − премия составляет 5500 руб. стаж свыше 20 лет − премия составляет 8000 руб. С помощью курсора определить общее количество премиальных выплат.

USE Zalivalov

DECLARE premia\_cursor CURSOR

LOCAL FORWARD\_ONLY STATIC

FOR SELECT Id\_Lect, FIO, Stage FROM Lecturer

DECLARE @Id\_Lect INT, @FIO VARCHAR(50), @Stage INT

DECLARE @sum\_premia INT, @res INT

-- определить таблицу для хранения результатов

DECLARE @res\_table TABLE (Id\_Lect INT, FIO VARCHAR(50), res INT)

SET @sum\_premia = 0

OPEN premia\_cursor

FETCH NEXT FROM premia\_cursor INTO @Id\_Lect, @FIO, @Stage

WHILE @@FETCH\_STATUS = 0

BEGIN

IF EXISTS (SELECT \* FROM Exam WHERE Id\_Lect = @Id\_Lect)

BEGIN

SET @res = CASE

WHEN @Stage Between 5 AND 10 THEN 4000

WHEN @Stage Between 11 AND 20 THEN 5500

WHEN @Stage > 20 THEN 8000

ELSE 0

END

SET @sum\_premia = @sum\_premia + @res

IF @Stage Between 5 AND 10 INSERT INTO @res\_table VALUES (@Id\_Lect, @FIO, @res)

IF @Stage Between 11 AND 20 INSERT INTO @res\_table VALUES (@Id\_Lect, @FIO, @res)

IF @Stage > 20 INSERT INTO @res\_table VALUES (@Id\_Lect, @FIO, @res)

END

FETCH NEXT FROM premia\_cursor INTO @Id\_Lect, @FIO, @Stage

END

PRINT 'Сумма премиальный выплат: ' + CAST(@sum\_premia AS VARCHAR (6))

SELECT FIO, res FROM @res\_table

CLOSE premia\_cursor

DEALLOCATE premia\_cursor

ПРОЦЕДУРЫ

С помощью хранимой процедуры выполнить перевод студента Иванова Р.А. в группу А-12-02, предварительно проверив его наличие в базе данных и правильность задания новой группы. В основной программе вызвать данную процедуру и прокомментировать результаты

CREATE PROCEDURE [dbo].[Change\_group2]

-- Add the parameters for the stored procedure here

@Fio VARCHAR(50), @Group VARCHAR(20), @Transact VARCHAR(50) OUTPUT

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

-- Insert statements for procedure here

IF EXISTS (SELECT \* FROM Student WHERE Fio = @Fio)

BEGIN

IF EXISTS (SELECT \* FROM Student WHERE NameGroup = @Group)

BEGIN

UPDATE Student SET NameGroup = @Group

WHERE Fio = @Fio

SET @Transact = 'Перевод студента в группу - ' + CAST(@Group AS VARCHAR(50)) + ' совершён!'

END

ELSE SET @Transact = 'Перевод студента в группу - ' + CAST(@Group AS VARCHAR(50)) + ' не совершён! Проверьте правильность введённой группы!'

END

ELSE SET @Transact = 'Перевод студента - ' + CAST(@Fio AS VARCHAR(50)) + ' не совершён! Проверьте правильность введённого имени!'

ПРОЦЕДУРА

Написать функцию или хранимую процедуру, возвращающую количество студентов, не имеющих ни одной оценки

CREATE PROCEDURE [dbo].[count\_students\_no\_mark6]

-- Add the parameters for the stored procedure here

@Transact INT OUTPUT

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

-- Insert statements for procedure here

SELECT @Transact = COUNT(DISTINCT Student.Id\_Student) FROM Exam JOIN Student ON Exam.Id\_Student = Student.Id\_Student

GROUP BY Student.Id\_Student

--@Transact = COUNT(Student.Id\_Student)

SET @Transact = (SELECT COUNT(Id\_Student) FROM Student) - @Transact

END

ПРОЦЕДУРА

Данные о сотрудниках хранятся в таблице Сотрудник (Номер, Фамилия, Год\_рождения, Пол, Отдел, Зарплата). Написать хранимую процедуру, возвращающую номер отдела, в котором работает наибольшее количество сотрудников, старше указанного возраста. В основной программе вызвать данную процедуру.

CREATE PROCEDURE dep\_get

-- Add the parameters for the stored procedure here

@age INT

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @dep INT, @count1 INT

DECLARE @vivod VARCHAR(50)

DECLARE @table TABLE (dep2 INT, count2 INT)

INSERT INTO @table SELECT Отдел, COUNT(Фамилия) FROM Сотрудник WHERE YEAR(GETDATE()) - YEAR(Год\_рождения) > @age GROUP BY Отдел

--SELECT \* FROM @table

SELECT @vivod = MAX(dep2) FROM @table

PRINT @vivod

END

GO

-- вызов процедуры с переменной количество лет

EXECUTE dep\_get 28

ФУНКЦИЯ

Данные о результатах сдачи сессии студентами хранятся в таблицах: Студент (ФИО, Группа, ДатаРождения, Стипендия) и Экзамен (ФИО, Предмет, Балл, Дата). Написать функцию, возвращающую количество студентов указанной группы, не имеющих ни одной оценки.

В основной программе вызвать данную функцию. Если таких студентов нет, то прокомментировать это.

CREATE FUNCTION count\_student\_nomark\_

(

-- Add the parameters for the function here

@group VARCHAR(50)

)

RETURNS INT

AS

BEGIN

-- Declare the return variable here

DECLARE @count INT, @count2 INT

SET @count2 = (SELECT COUNT(DISTINCT Экзамен.ФИО) FROM Экзамен

JOIN Студент ON Студент.ФИО = Экзамен.ФИО WHERE Студент.Группа = @group)

SELECT @count = COUNT(DISTINCT ФИО) FROM Студент WHERE Группа = @group

SET @count = @count - @count2

RETURN @count

END

GO

PRINT 'Количество студентов не имеющих оценок: '

-- вызов функции с переменной, задаваемой пользователем

PRINT dbo.count\_student\_nomark\_('ИДБ')

КУРСОР

Решено выплатить всем хирургам, имеющим стаж от 5 до 10 лет, премию в размере 20% от зарплаты, а свыше 10 лет − в размере 25% от зарплаты. С помощью курсора определить общее количество премиальных выплат. Данные хранятся в таблице Врач (ФИО, Год\_рождения, Пол, Специальность, Стаж, Зарплата).

DECLARE sum\_premia CURSOR LOCAL FORWARD\_ONLY STATIC

FOR SELECT ФИО, Специальность, Стаж, Зарплата FROM Врач

OPEN sum\_premia

-- определить таблицу для хранения результатов

-- объявление локальных переменных для FETCH

DECLARE @fio VARCHAR(50), @spec VARCHAR(50), @stage INT, @salary INT, @x INT

SET @x = 0

FETCH NEXT FROM sum\_premia INTO @fio, @spec, @stage, @salary

WHILE @@FETCH\_STATUS = 0

BEGIN

IF (@spec = 'Хирург' AND @stage BETWEEN 5 AND 10)

BEGIN

SET @x = @x + ( @salary \* 0.2 )

END

IF (@spec = 'Хирург' AND @stage > 10)

BEGIN

SET @x = @x + ( @salary \* 0.25 )

END

FETCH NEXT FROM sum\_premia INTO @fio, @spec, @stage, @salary

END

-- вывести результаты

PRINT @x

CLOSE sum\_premia

DEALLOCATE sum\_premia

ПРОЦЕДУРА

Данные о врачах хранятся в таблице Врач (ФИО, Год\_рождения, Специальность, № клиники, Зарплата). Написать хранимую процедуру, которая возвращает самую высококооплачиваемую специальность врачей (по наибольшему среднему заработку) в указанной клинике. В основной программе вызвать данную процедуру.

CREATE PROCEDURE best\_spec\_for\_salary

-- Add the parameters for the stored procedure here

@klinika INT

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

-- Insert statements for procedure here

DECLARE @speci INT, @salary\_avg INT

DECLARE @vivod VARCHAR(50)

DECLARE @table TABLE (spec VARCHAR(50), average\_salary INT)

INSERT INTO @table SELECT Специальность, AVG(Зарплата) FROM Врач WHERE [№ клиники] = @klinika GROUP BY Специальность

SELECT @vivod = spec FROM @table WHERE average\_salary = (SELECT MAX(average\_salary) FROM @table)

PRINT @vivod

END

GO

-- вызов процедуры с переменной номера клиники

EXECUTE best\_spec\_for\_salary 1

ФУНКЦИИ

Данные о результатах сдачи сессии студентами хранятся в таблицах: Студент (ФИО, Группа, ДатаРождения, Стипендия) и Экзамен (ФИО, Предмет, Балл, Дата). Кто из указанной группы сдал все экзамены на отлично?

В основной программе вызвать данную функцию. Если таких студентов нет, то прокомментировать это.

CREATE FUNCTION Student\_with\_5\_marks

(

-- Add the parameters for the function here

@group VARCHAR(20)

)

RETURNS TABLE

AS

RETURN

(

-- Add the SELECT statement with parameter references here

-- Add the SELECT statement with parameter references here

SELECT Экзамен.ФИО

FROM Экзамен JOIN Студент ON Экзамен.ФИО = Студент.ФИО

WHERE Группа = @group GROUP BY Экзамен.ФИО HAVING Min(Балл) > 44

)

GO

IF EXISTS (SELECT \* FROM dbo.Student\_with\_5\_marks('ИДБ'))

SELECT \* FROM dbo.Student\_with\_5\_marks('ИДБ')

ELSE PRINT 'Таких студентов нет'

ПОЛЬЗОВАТЕЛЬСКИЕ ФУНКЦИИ

Данные о результатах сдачи сессии студентами хранятся в таблицах: Студент (ФИО, Группа, ДатаРождения, Стипендия), СтудГруппа (Номер, Курс, Кафедра) и Экзамен (ФИО, Предмет, Балл, Дата). Кто среди студентов 3-го курса сдал указанный экзамен на отлично? Вывести их фамилии и группы.

В основной программе вызвать данную функцию. Если таких студентов нет, то прокомментировать это.

CREATE FUNCTION Exam (@Subject VARCHAR(30))

RETURNS TABLE

AS

RETURN

(

SELECT ФИО, Группа FROM Студент

JOIN СтудГруппа ON Студент.Группа = СтудГруппа.Номер

JOIN Экзамен ON Студент.ФИО = Экзамен.ФИО

WHERE Экзамен.Предмет = @Subject

AND СтудГруппа.Курс = 3 AND Экзамен.Балл >= 45

)

Вызов функции:

if (SELECT COUNT(\*) FROM Exam('Управление данными')) = 0

PRINT 'Таких студентов нет'

ELSE

SELECT \* FROM Exam('Управление данными')