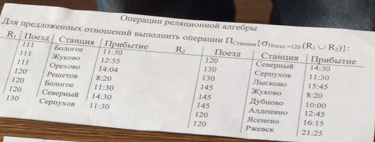
**Содержание**

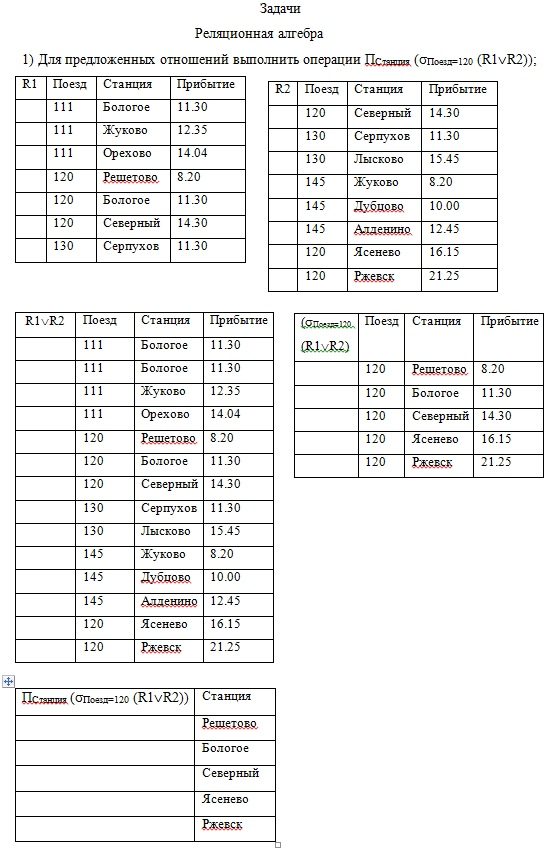
1. Операции реляционной алгебры (1-4)
2. Триггеры (5-8)
3. Пользовательские функции (9-11)
4. Хранимые процедуры(12-14)
5. Курсоры (15-16)
6. Нормализация реляционных баз данных (17-20)
7. Построение ER-диаграмм (21-23)
8. Операторы создания таблиц (24-25)

-

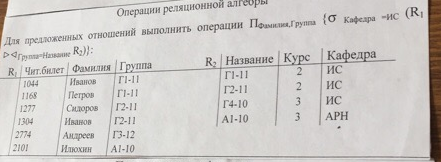
**1. Операции реляционной алгебры**



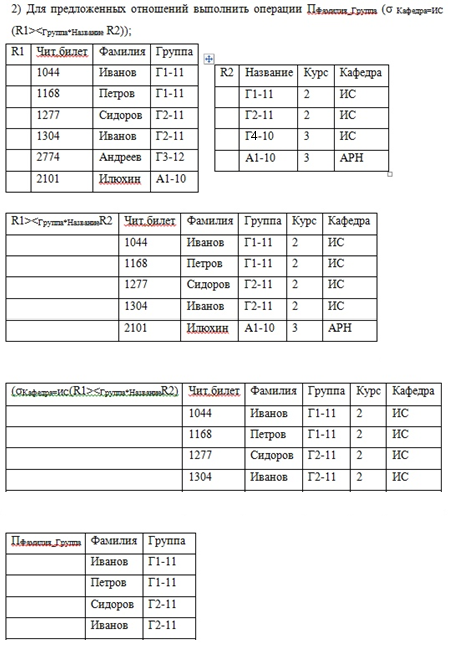
Решение



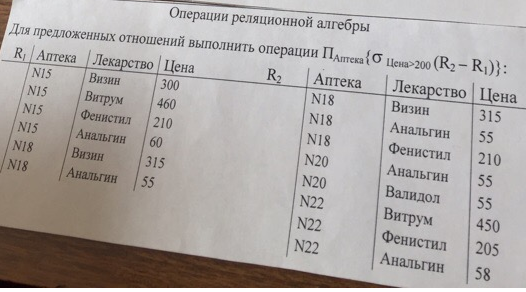
**2. Операции реляционной алгебры**

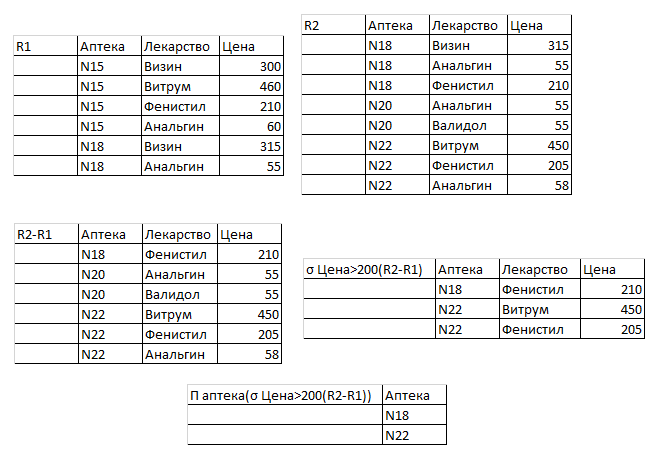


Решение

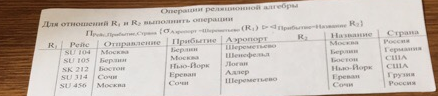


**3. Операции реляционной алгебры**

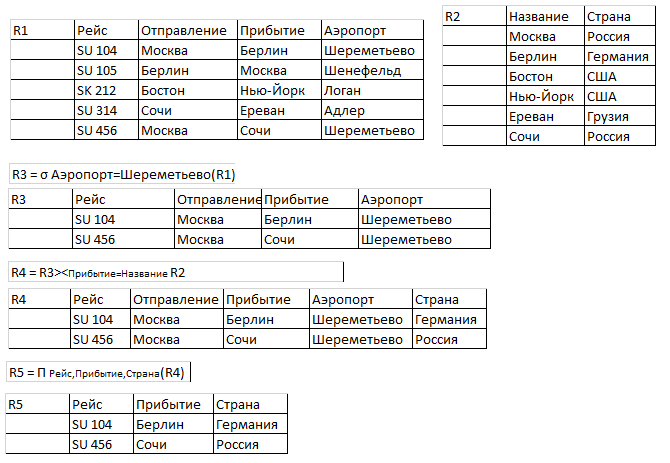




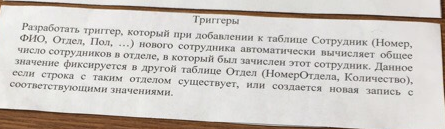
**4. Операции реляционной алгебры**



Решение



**5. 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)

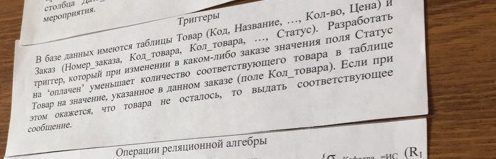
INSERT INTO Отдел VALUES (@dep, @kol)

ELSE

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

END

**6. 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 = Кол-во - @kzak WHERE Код = [@kod](https://vk.com/kod)

ELSE

BEGIN

PRINT 'Товара не осталось'

ROLLBACK TRANSACTION

END

END

END

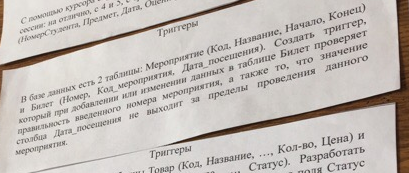
//@kod – код товара

//@kskl – количество товара на складе

//@kzak – количество товара в заказе

//@st – статус заказа

**7. 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

END

SELECT @date = Дата\_посещения FROM inserted

SELECT @begin = Начало, @end=Конец FROM Мероприятие

WHERE Код = @kod

IF NOT (@date BETWEEN @begin AND @end)

BEGIN

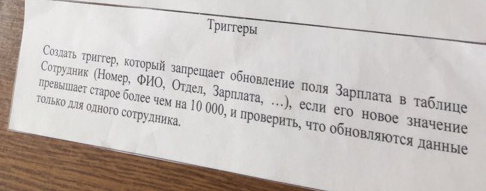
PRINT 'Проверьте дату'

ROLLBACK TRAN

END

END

**8. 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

END

SELECT @salary = Зарплата FROM deleted

SELECT @salarynew = Зарплата FROM inserted

IF (@salarynew-@salary)>10000

BEGIN

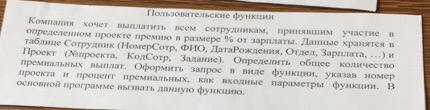
PRINT 'Зарплата не должна быть увеличена больше, чем на 10000'

ROLLBACK TRAN

END

END

**9. 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

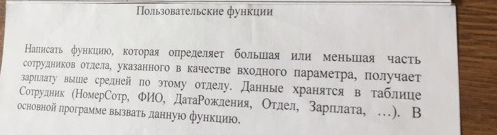
Вызов функции:

DECLARE @res REAL

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

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

**10. 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'

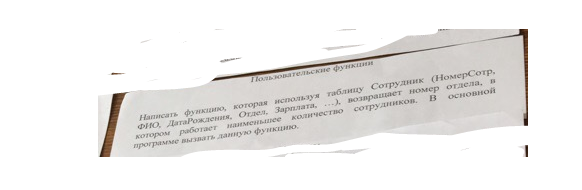
IF @res = 1

PRINT 'большая часть получает зп выше средней'

IF @res = 2

PRINT 'одинаково'

**11. 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

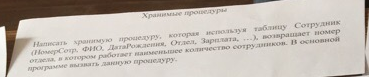
Вызов функции:

DECLARE @res INT

SET @res = dbo.f1()

PRINT @res

**12. Хранимые процедуры**



Решение

Создание процедуры:

CREATE PROC pr @res INT OUT

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

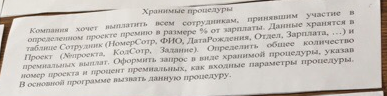
Вызов процедуры:

DECLARE @result INT

EXEC pr @result OUT

PRINT @res

**13. Хранимые процедуры**



Решение

Создание процедуры:

CREATE PROC pr

@prj INT,

@percent REAL,

@prem\_sum REAL OUT

AS

BEGIN

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

WHERE НомерСотр IN (SELECT КодСотр FROM Проект

WHERE №проекта = @prj)

END

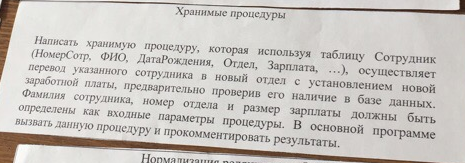
Вызов процедуры:

DECLARE @res REAL

EXEC pr 1, 0.15, @res OUT

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

**14. Хранимые процедуры**



Решение

Создание процедуры:

CREATE PROC pr

@fam VARCHAR(20),

@otd INT,

@salary INT

AS

BEGIN

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

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

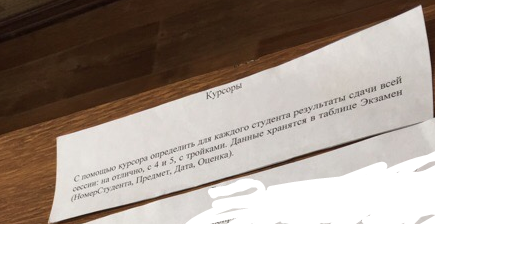
WHERE ФИО = @fam

END

Вызов процедуры:

EXEC pr ‘Степаненков И.И.’, 2, 5000

**15. Курсоры**



Решение

DECALRE cur1 CURSOR

LOCAL

FORWARD\_ONLY

STATIC

READ\_ONLY

FOR SELECT НомерСтудента, MIN(Оценка) FROM Экзамен

GROUP BY НомерСтудента

OPEN cur1

DECLARE @id INT, @oc INT

DECLARE @otv TABLE (id INT, res VARCHAR(20))

FETCH NEXT FROM cur1 INTO @id, @oc

WHILE @@fetch\_status = 0

BEGIN

INSERT INTO @otv VALUES (@id, CASE @oc

WHEN 5 THEN 'отлично'

WHEN 4 THEN 'с четверками'

ELSE 'с тройками'

END)

FETCH NEXT FROM cur1 INTO @id, @oc

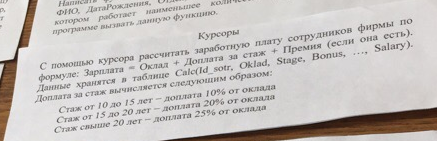
END

CLOSE cur1

DEALLOCATE cur1

SELECT \* FROM @otv //Вывод результата

**16. Курсоры**



Решение

DECLARE cur1 CURSOR

FORWARD\_ONLY

DYNAMIC

OPTIMISTIC

FOR SELECT oklad, stage, bonus, salary FROM Calc

DECLARE @okl INT, @st INT, @bon INT, @sal INT

OPEN cur1

FETCH NEXT cur1 INTO @okl, @st, @bon, @sal

WHILE @@fetch\_status = 0

BEGIN

SET @sal=@okl

SET @sal= @sal + CASE

WHEN @st BETWEEN 10 AND 15 THEN @okl\*0,1

WHEN @st BETWEEN 16 AND 20 THEN @okl\*0,2

WHEN @st>20 THEN @okl\*0,25

ELSE 0

END

IF @bon IS NOT NULL

SET @sal=@sal+ @Bon

UPDATE Calc SET salary = @sal WHERE current OF cur1

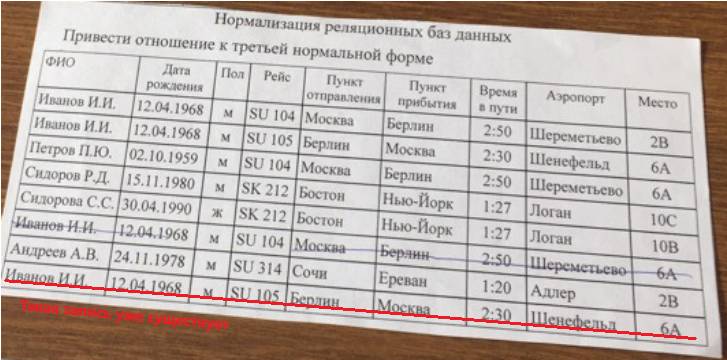
FETCH NEXT cur1 INTO @okl, @st, @bon, @sal

END

CLOSE cur1

DEALLOCATE cur1

**17. Нормализация реляционных баз данных**



Решение

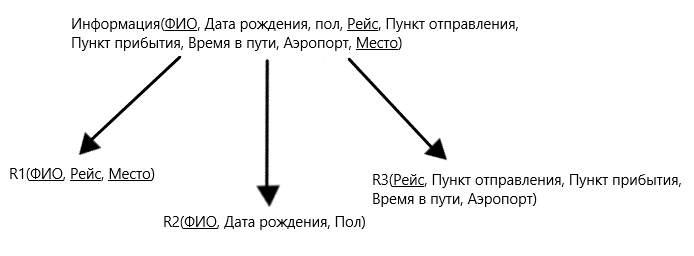
Первичный ключ(ФИО, Рейс, Место)

ФИО -> Дата рождения, пол

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

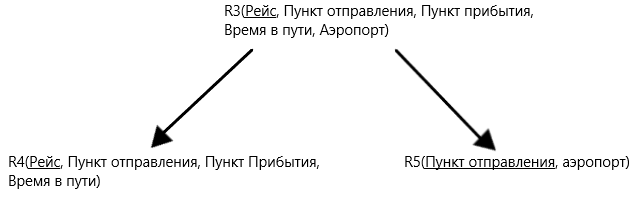
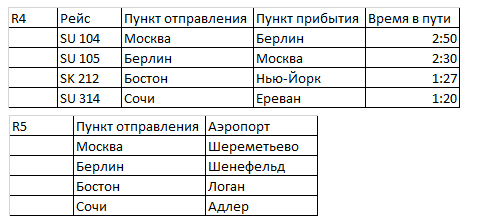
Пункт отправления -> Аэропорт

К 2НФ:



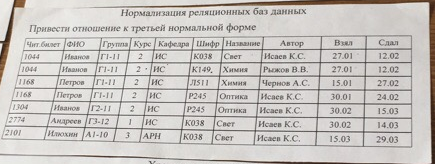


К 3НФ:

Итоговые таблицы: R1, R2, R4, R5

**18. Нормализация реляционных баз данных**



Решение

Первичный ключ(Чит. Билет, Шифр)

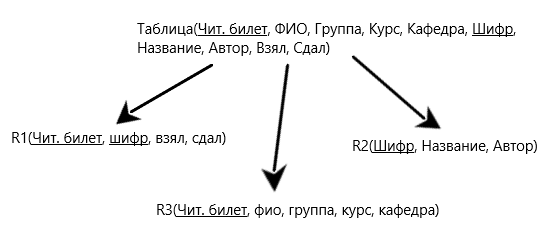
Чит. Билет, Шифр -> фио, группа, курс, кафедра, название, автор, взял, сдал

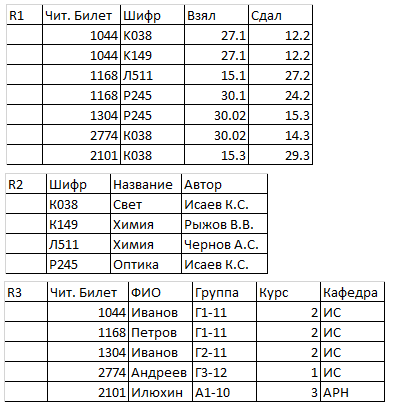
Чит. Билет -> фио, группа, курс, кафедра

Группа -> курс, кафедра

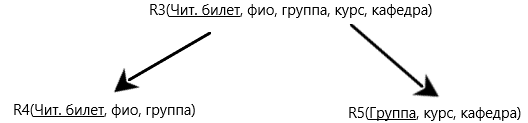
Шифр -> название, автор

К 2НФ:





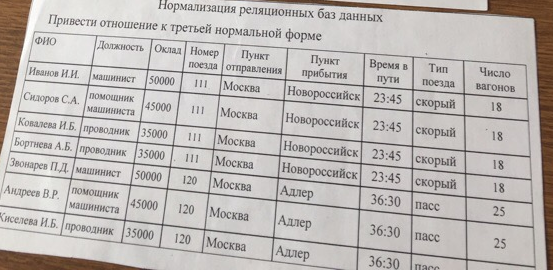
К 3 НФ:





Итоговые таблицы: R1, R2, R4, R5

**19. Нормализация реляционных баз данных**



Решение

Первичный ключ(ФИО)

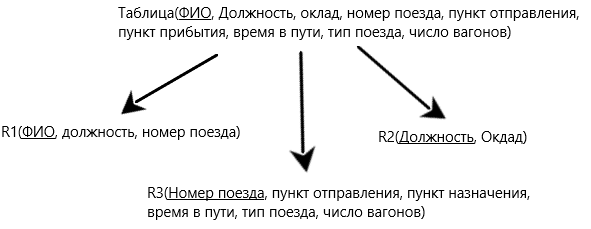
ФИО -> Должность, Оклад, номер поезда, пункт отправления, пункт прибытия, время в пути, тип поезда, число вагонов

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.

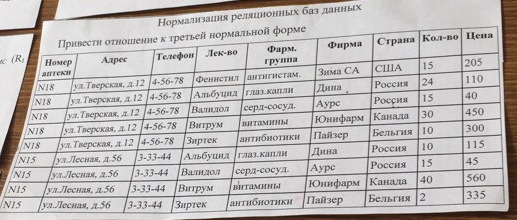
К 3НФ:





Итоговые таблицы: R1, R2, R3

**20. Нормализация реляционных баз данных**



Решение

Первичный ключ(Номер аптеки, лекарство)

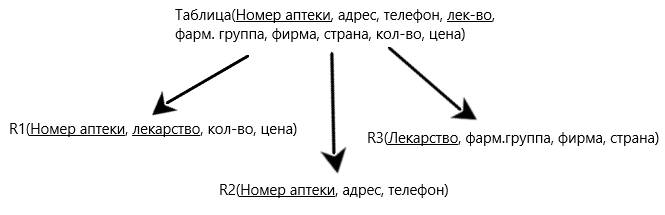
Номер аптеки, лекарство -> Адрес, телефон, лек-во, фарм. Группа, фирма, Страна, кол-во, цена

Номер аптеки -> адрес, телефон

Лекарство -> фарм. Группа, фирма, страна

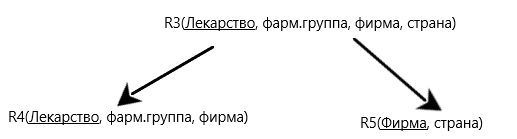
Фирма -> страна

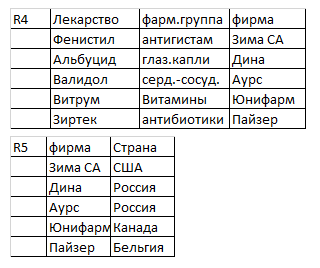
К 2 НФ:



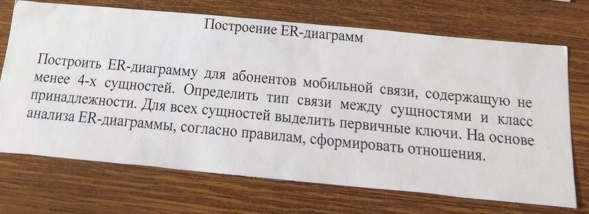


К 3НФ:





**21. 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:

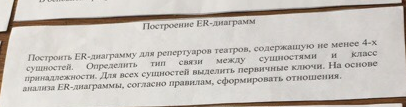
1. Абонент (КодАбонента, ФИО, ПаспортныеДанные, ДатаРождения)

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

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

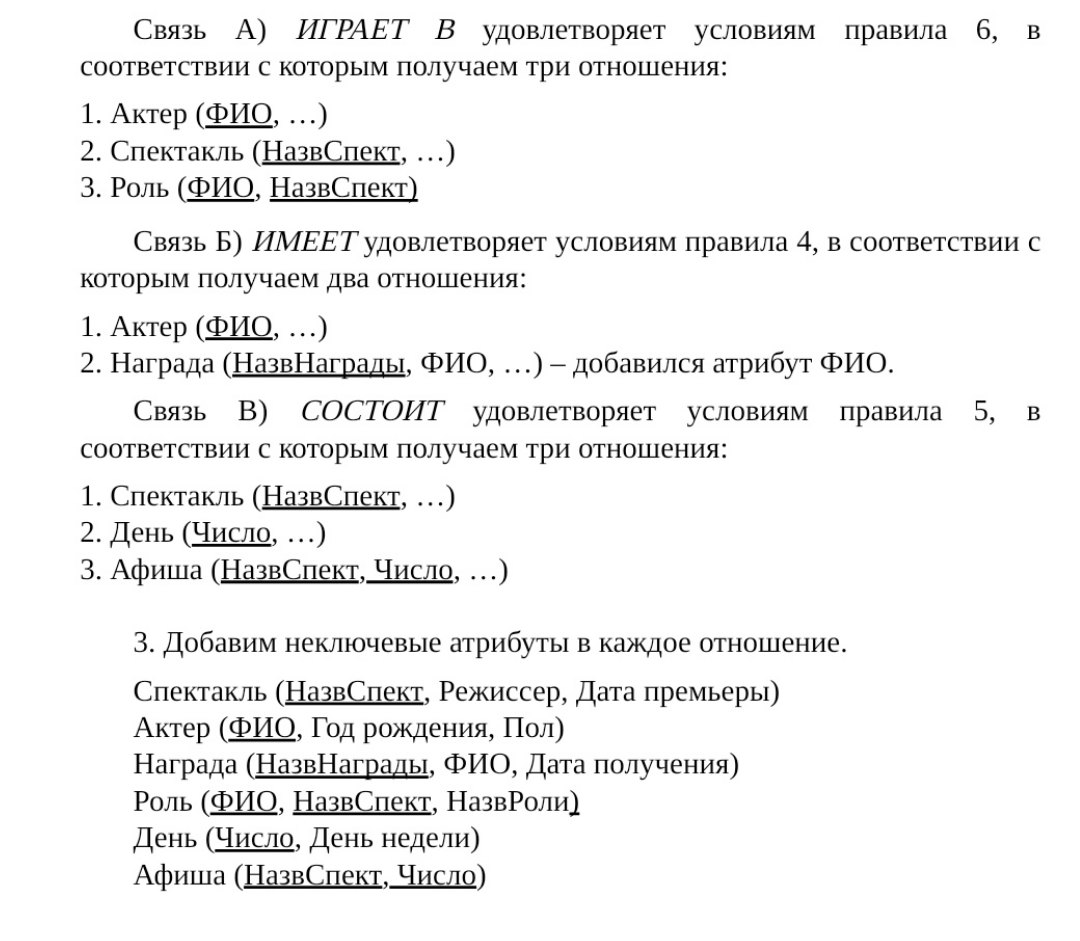
4. Договор (КодДоговора, КодАбонента, КодСотрудника, КодТарифа, НомерТелефона, ДатаЗаключения, СрокДействия)

**22. Построение ER-диаграмм**

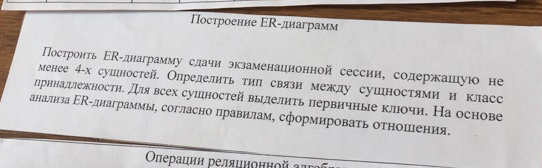


Решение

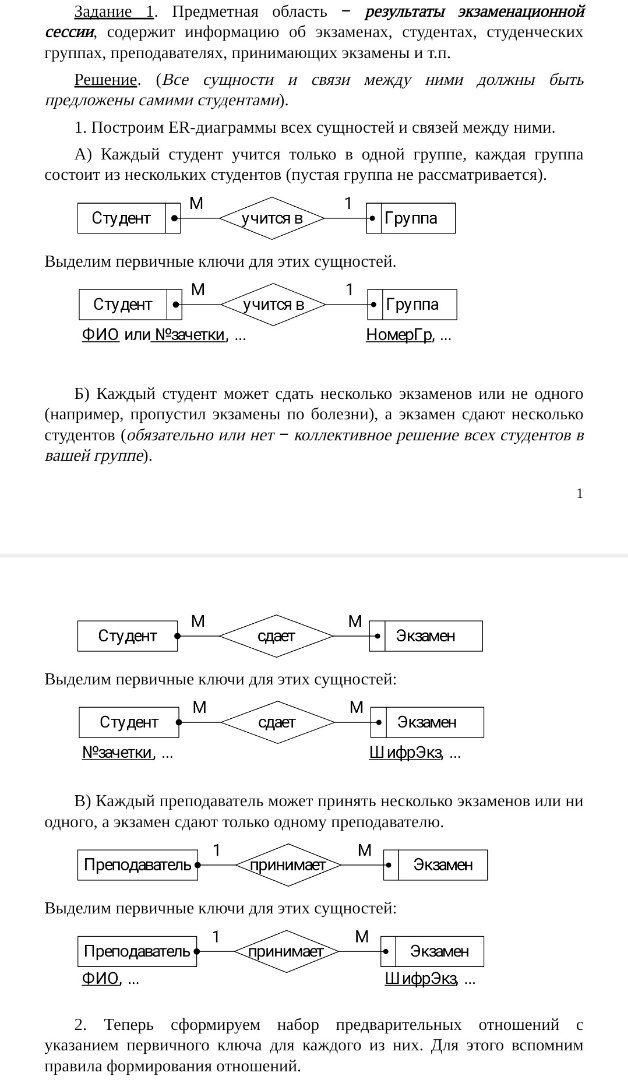




**23. Построение ER-диаграмм**

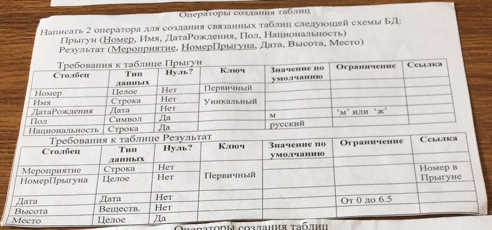


Решение

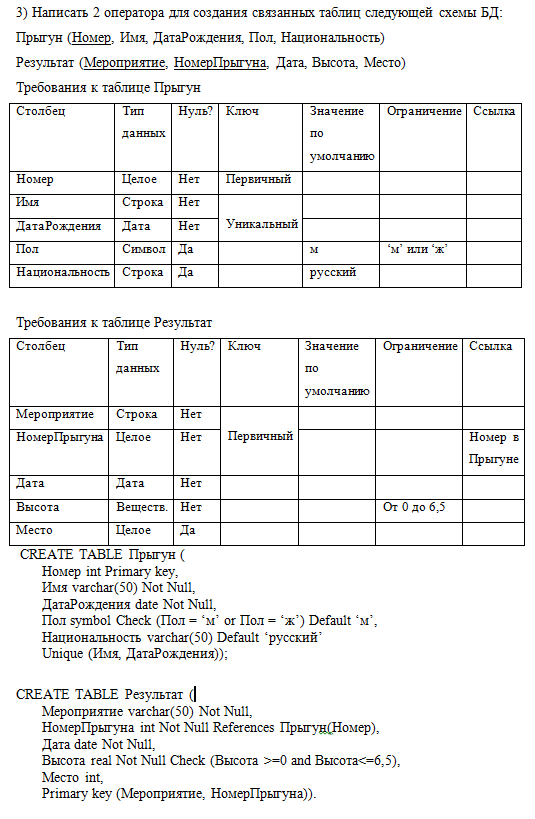




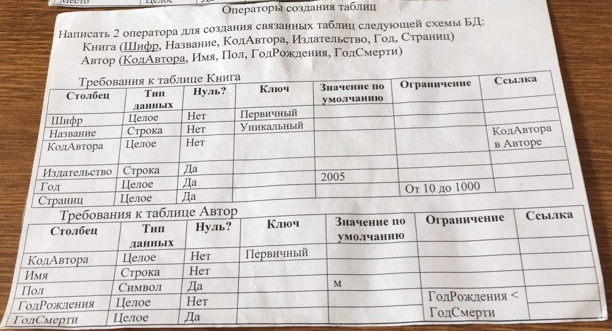
**24. Операторы создания таблиц**



Решение



**25. Операторы создания таблиц**



Решение

