

01.03.02 «Прикладная математика и информатика»

ВВЕДЕНИЕ.

Технологическая практика представляет собой реализацию практической работы по дисциплине базы данных. Состоит из трёх частей выполняемых последовательно. Отчёт пишется по завершении практики с описанием получившегося приложения. Первая часть посвящена серверной разработке - созданию базы данных, таблиц, связей, триггеров и хранимых процедур. Вторая часть - разработка клиентского приложения с отображением данных с сервера. Третья часть - реализация сложных запросов к базе данных с использованием теоретических основ баз данных.

В процессе прохождения практики осваивается базовый уровень следующих компетенций:

ОПК-3 Способность применять и модифицировать математические модели для решения задач в области профессиональной деятельности

ОПК-4 Способность решать задачи профессиональной деятельности с использованием существующих информационно-коммуникационных технологий и с учетом основных требований информационной безопасности

ПЕРВАЯ ЧАСТЬ.

1)*Созданы три таблицы

*Описание таблиц включает использование (хотя бы по одному разу): *NOT NULL, DEFAULT, PRIMARY KEY, CHECK и IDENTITY*.

Исходные данные:

Файл02. Справочник деталей

- ◆ код детали;
- ◆ тип детали (покупная или собственного производства);
- ◆ наименование;
- ◆ единица измерения;
- ◆ плановая цена за единицу.

Файл14. Учет отгрузки готовой продукции

- ◆ номер склада;
- ◆ номер документа об отгрузке;
- ◆ код покупателя;
- ◆ код готового изделия;
- ◆ единица измерения;
- ◆ количество;
- ◆ дата отгрузки;

Файл15. Покупатели

- ◆ код покупателя;
- ◆ имя покупателя;

- ◆ адрес покупателя;

15:Покупатели <->> 14:Учет отгрузки <<-> 02:Детали

```
create database kurs_work
create table details(
kod_det int not null primary key identity(201, 1),
type_det varchar(1) not null check(type_det = 'p' or type_det = 'o'),
name_det varchar(20),
ed_izm varchar(5) default('ps'),
plan_cost decimal(14, 2)
)
```

```
create table buyers(
kod_buyers int not null primary key identity(101,1),
name_buyers varchar(20) not null,
address_buyers varchar(50) default("")
)
```

```
create table storage(
id_storage int not null,
id_document int not null,
kod_done_product int not null,
kod_buyers int not null,
ed_izm varchar(5) not null default('ps'),
kolvo int default(0),
date_ship varchar(20) not null,
constraint pk_storage primary key(id_storage, id_document),
constraint fk1 foreign key(kod_done_product) references
details(kod_det) on delete cascade on update cascade,
constraint fk2 foreign key(kod_buyers)
references buyers(kod_buyers)
)
```

*Параметр **NOT NULL** вводится для контроля начального заполнения поля при его обработке

*Параметр **DEFAULT** задает значение соответствующего поля по умолчанию.

***PRIMARY KEY** – ограничение, указывающее, что в данной таблице данное поле представляет собой первичный ключ.

*Каждому полю со свойством **IDENTITY** SQL Server ставит в соответствие свой счетчик и будет следить за нумерацией вновь создаваемых записей.

2) Описаны две межтабличные связи:

- Одна без использования системного каскадного удаления и обновления. К этой связи определены два триггера, один из них на каскадное удаление в дочерней таблице.
- Другая с использованием системного каскадного удаления и обновления (*ON DELETE CASCADE ON UPDATE CASCADE*).
- Создана хранимая процедура с выходными параметрами.

Триггер (каскадное удаление):

```
CREATE or alter TRIGGER trigger_delete_1 ON buyers
INSTEAD OF DELETE
AS
BEGIN
    DELETE FROM storage
    WHERE id_document IN (SELECT kod_buyers FROM DELETED)
    DELETE FROM buyers
    WHERE kod_buyers IN (SELECT kod_buyers FROM DELETED)
END
```

Функция (вывод данных о детали по ее заданному коду):

```
create or alter function Get_name_det(@kod_det int)
returns table
    return select p.kod_det, p.name_det, p.plan_cost
    from details p
    where p.kod_det = @kod_det
```

Хранимая процедура (минимальная плановая стоимость заданного типа детали):

```
create or alter procedure Procedure_1
@type_det varchar(3) output, @plan_costs decimal(14,2) output
as
begin
    select @plan_costs = min(plan_cost) from details
    where type_det = @type_det
end
```

3)Подготовлен *SQL-script* для загрузки данных в таблицы и данные загружены (не менее пяти строк в каждой таблице). База данных корректна в смысле явно описанных ограничений целостности.

```
set identity_insert details on
insert into details(kod_det, type_det, name_det, ed_izm, plan_cost)
    values (201, 'p', 'provod', 'm', 0.5),
           (202, 'o', 'bolt', 'g', 3.2),
           (203, 'p', 'muka', 'kg', 10.5),
           (204, 'p', 'apple', 'ps', 10),
           (205, 'o', 'kvas', 'l', 3),
           (206, 'p', 'bread', 'ps', 10),
           (207, 'o', 'gaika', 'l', 3)
set identity_insert details off

set identity_insert buyers on
insert into buyers(kod_buyers, name_buyers, address_buyers)
    values (101, 'Darya', 'Kazan, Pushkina 32'),
           (102, 'Nikita', 'Kazan, Riharda Zorge 23'),
           (103, 'Regina', 'Kazan, Karla Marksa 11'),
           (104, 'Marina', 'Kazan, Prospekt Pobedy 128'),
           (105, 'Roman', 'Kazan, Kremlevskaya 35'),
           (107, 'Vladimir', 'Kazan, Pr Pob 35'),
```

```
(108, 'Regina', 'Moscow, Karla Marksa 11'),
(109, 'Marina', 'Moscow, Prospekt Pobedy 128'),
(106, 'Roman', 'Vyatskie Polyany, Kremlevskaya 35'),
(110, 'Vladimir', 'Kirov, Pr Pob 35')
```

```
set identity_insert buyers off
```

```
insert into storage(id_storage, id_document, kod_done_product, kod_buyers, ed_izm,
kolvo, date_ship)
```

```
values (201, 101, 201, 101, 'kg', 5, '21_05_2001'),
(202, 102, 202, 102, 'l', 1, '07_09_1999'),
(203, 103, 203, 103, 'm', 3, '04_08_2001'),
(204, 104, 204, 104, 'g', 17, '28_09_1999'),
(205, 105, 205, 105, 'ps', 9, '21_08_1999'),
(201, 103, 201, 101, 'kg', 7, '21_05_2001'),
(201, 171, 201, 107, 'kg', 5, '21_05_2001'),
(202, 105, 205, 108, 'ps', 9, '21_08_1999'),
(207, 103, 207, 107, 'kg', 7, '21_05_2001'),
(201, 131, 201, 107, 'kg', 5, '21_05_2001'),
(201, 143, 207, 109, 'kg', 7, '21_05_2001'),
(207, 114, 207, 109, 'kg', 5, '21_05_2001'),
(205, 194, 201, 102, 'kg', 5, '21_05_2001'),
(205, 123, 206, 101, 'kg', 5, '21_05_2001'),
(205, 174, 206, 104, 'kg', 5, '21_05_2001'),
(208, 1554, 206, 101, 'kg', 5, '21_05_2001')
```

```
select * from buyers
```

```
select * from details
```

```
select * from storage
```

ВТОРАЯ ЧАСТЬ.

1) Клиентская часть:

***Visual Studio-проект должен включать:** главную экранную форму, визуализирующую таблицы ранее созданной базы данных (**DataSet, TableAdapter, DataGridView, BindingSource**).

Моделью данных будет «Набор данных» - **kurs_work** – снимок части нашей базы данных, необходимой в клиентском приложении.

Он хранит коллекцию таблиц, каждая из них является объектом расширения класса **DataTable**.

Для каждой таблицы кроме этого создается соответствующий класс **TableAdapter**.

Если мы хотим сами контролировать, откуда что появилось, добавим на форму элемент из «Панели элементов» **DataGridView**.

BindingSource связывает элементы на форме с источниками данных.

Form1

Формы Задача_1 Задача_2 Задача_3

1 для 10

Код покупателя	Имя покупателя	Адрес
101	Darya	Kazan
102	Nikita	Kazan
103	Regina	Kazan
104	Marina	Kazan
105	Roman	Kazan
106	Roman	Vyats

Код детали	Тип детали	Наимен. детали	Ед. измерения
201	p	provod	m
202	o	bolt	g
203	p	muka	kg
204	p	apple	ps
205	o	kvas	l
206	p	bread	ps

Код склада	Имя покупателя	Код готового продукта	Ед. измерения	Количество	Дата отп.
201	Darya	201	kg	5	21_05_201
201	Darya	201	kg	7	21_05_201
201	Vladimir	201	kg	5	21_05_201
201	Marina	207	kg	7	21_05_201
201	Vladimir	201	kg	5	21_05_201
202	Nikita	202	l	1	07_09_19

ИbTable

Предыдущ. Следующ. Первая Последняя

Сохранить Обновить

Процедура Результат процедуры

Для упрощения навигации по таблице добавим компонент **BindingNavigator** на форму приложения.

Реализовать альтернативную форму с динамическим отображением всех таблиц в одном DataGridView.

Form2

5 для 10

kod_buyers	name_buyers	address_buyers
101	Darya	Kazan, Pushkina ...
102	Nikita	Kazan, Riharda Z...
103	Regina	Kazan, Karla Mar...
104	Marina	Kazan, Prospekt ...
105	Roman	Kazan, Kremlevs...
106	Roman	Vyatskie Polyany....
107	Vladimir	Kazan, Pr Pob 35
108	Regina	Moscow, Karla M...
109	Marina	Moscow, Prospe...
110	Vladimir	Kirov, Pr Pob 35

Детали Склад Покупатели Сохранить Закрыть

Покупатели

Отобразить результат выполнения хранимой процедуры в метке (Label) на основной или альтернативной форме-1

1. **Connection** – подключение к БД
2. **CommandType** – тип команды: SQL-команда, хранимая процедура или таблица.

3. **CommandText** – собственно текст команды, имя хранимой процедуры или таблицы.
4. **Parameters** – коллекция параметров команды.

Код покупателя	Имя покупателя	Адрес
101	Darya	Kazan
102	Nikita	Kazan
103	Regina	Kazan
104	Marina	Kazan
105	Roman	Kazan
106	Roman	Vyats

Код детали	Тип детали	Наимен. детали	Ед. измерения
201	p	provod	m
202	o	bolt	g
203	p	muka	kg
204	p	apple	ps
205	o	kvas	l
206	p	bread	ps

Код склада	Имя покупателя	Код готового продукта	Ед. измерения	Количество	Дата отправки
205	Darya	206	kg	5	21_05_2001
205	Marina	206	kg	5	21_05_2001
208	Darya	206	kg	5	21_05_2001

Details

Предыдущ. Следующ. Первая Последняя

Сохранить Обновить

Процедура 0.5

Задача-1. Сведения об отгрузке деталей покупателям из Казани: номер склада; код детали; дата отгрузки; количество; наименование покупателя.

Kazan

id_storage	kod_det	date_ship	kolvo	kod_buyers
201	201	21_05_2001	5	101
201	201	21_05_2001	7	101
201	201	21_05_2001	5	107
201	201	21_05_2001	5	107
202	202	07_09_1999	1	102
203	203	04_08_2001	3	103
204	204	28_09_1999	17	104

Показать

1) В первом варианте решения поставленной задачи воспользуемся Set-ориентированными средствами SQL для работы с таблицами и объектом типа **TableAdapter** для связи с SQL-сервером.


```

private void btnShow_Click(object sender, EventArgs e)
{
    String city = Convert.ToString(txtCity.Text);
    this.zadacha1_1TableAdapter.Fill(kurs_workDataSet.zadacha1_1, city);
}

private void zadacha1_1_Load(object sender, EventArgs e)
{
    dataGridView1.AutoGenerateColumns = true;
    zadacha1_1TableAdapter.Fill(kurs_workDataSet.zadacha1_1, "0dgfdggfsf");
}

```

2) Во **втором варианте** решения поставленной задачи воспользуемся *Record-*ориентированными средствами *Visual Studio* для работы с таблицами, наиболее близкими к традиционным процедурным средствам обработки файлов.

```

public void QueryZapr1()
{
    kurs_workDataSet.zadacha1_1.Clear();
    foreach (kurs_workDataSet.storageRow sRow in
kurs_workDataSet.storage.Rows)
    {
        foreach (kurs_workDataSet.detailsRow dRow in
kurs_workDataSet.details.Rows)
        {
            foreach (kurs_workDataSet.buyersRow bRow in
kurs_workDataSet.buyers.Rows)
            {
                string[] subs = bRow.address_buyers.Split(',');
                if (subs[0] == Convert.ToString(txtCity.Text) && (dRow.kod_det ==
sRow.kod_done_product) && (bRow.kod_buyers == sRow.kod_buyers))
                {
                    kurs_workDataSet.zadacha1_1Row zRow =
kurs_workDataSet.zadacha1_1.Newzadacha1_1Row();
                    zRow.id_storage = sRow.id_storage;
                    zRow.kod_det = dRow.kod_det;
                    zRow.date_ship = sRow.date_ship;
                    zRow.kolvo = sRow.kolvo;
                    zRow.kod_buyers = bRow.kod_buyers;
                    zRow.name_buyers = bRow.name_buyers;
                    kurs_workDataSet.zadacha1_1.Addzadacha1_1Row(zRow);
                }
            }
        }
    }
    dataGridView1.Refresh();
}

```

```
private void btnShow_Click(object sender, EventArgs e)
{
    QueryZapr1();
}
```

3) В **третьем варианте** решения поставленной задачи, воспользуемся первичными ключами для поиска в таблице с интересующим нас кодом товара. Поиск строки в таблице по её ключу можно выполнить с помощью метода **Find** объекта **DataRowCollection**. У этого метода две перегрузки для одинарного и для составного первичного ключа.

```
public void QueryZapr2()
{
    kurs_workDataSet.zadacha1_1.Clear();

    foreach (kurs_workDataSet.buyersRow bRow in
kurs_workDataSet.buyers.Rows)
    {

        foreach (kurs_workDataSet.storageRow sRow in
kurs_workDataSet.storage.Rows)
        {
            string[] subs = bRow.address_buyers.Split(',');
            if (subs[0] == Convert.ToString(txtCity.Text) && (bRow.kod_buyers ==
sRow.kod_buyers))
            {
                kurs_workDataSet.detailsRow dRow =
kurs_workDataSet.details.Rows.Find(sRow.kod_done_product) as
kurs_workDataSet.detailsRow;
                kurs_workDataSet.buyersRow gRow =
kurs_workDataSet.buyers.Rows.Find(sRow.kod_buyers) as
kurs_workDataSet.buyersRow;
                kurs_workDataSet.zadacha1_1Row zRow =
kurs_workDataSet.zadacha1_1.Newzadacha1_1Row();
                zRow.id_storage = sRow.id_storage;
                zRow.kod_det = dRow.kod_det;
                zRow.date_ship = sRow.date_ship;
                zRow.kolvo = sRow.kolvo;
                zRow.kod_buyers = gRow.kod_buyers;
                zRow.name_buyers = gRow.name_buyers;
                kurs_workDataSet.zadacha1_1.Addzadacha1_1Row(zRow);
            }
        }
    }
    dataGridView1.Refresh();
    this.Text = "Запрос через поиск по ключу";
}

private void btnShow_Click(object sender, EventArgs e)
{
}
```

```

        QueryZapr2();
    }

```

4) В **четвертом варианте** решения поставленной задачи воспользуемся операционной связью **DataRelation** для таблиц в наборе данных.

```

public void QueryZapr3()
{
    kurs_workDataSet.zadacha1_1.Clear();
    foreach (kurs_workDataSet.buyersRow bRow in
kurs_workDataSet.buyers.Rows)
    {
        foreach (kurs_workDataSet.storageRow sRow in
kurs_workDataSet.storage.Rows)
        {
            string[] subs = bRow.address_buyers.Split(',');
            if (subs[0] == Convert.ToString(txtCity.Text) && (bRow.kod_buyers ==
sRow.kod_buyers))
            {
                kurs_workDataSet.detailsRow dRow = sRow.GetParentRow("fk1") as
kurs_workDataSet.detailsRow;
                kurs_workDataSet.buyersRow gRow = sRow.GetParentRow("fk2") as
kurs_workDataSet.buyersRow;
                kurs_workDataSet.zadacha1_1Row zRow =
kurs_workDataSet.zadacha1_1.Newzadacha1_1Row();
                zRow.id_storage = sRow.id_storage;
                zRow.kod_det = dRow.kod_det;
                zRow.date_ship = sRow.date_ship;
                zRow.kolvo = sRow.kolvo;
                zRow.kod_buyers = gRow.kod_buyers;
                zRow.name_buyers = gRow.name_buyers;
                kurs_workDataSet.zadacha1_1.Addzadacha1_1Row(zRow);
            }
        }
    }
    dataGridView1.Refresh();
    this.Text = "Запрос через DataRelation";
}

private void btnShow_Click(object sender, EventArgs e)
{
    QueryZapr3();
}

```

5) В **пятом варианте** решения поставленной задачи будем использовать присоединенный режим работы с базой данных. То есть сами открываем соединение с БД, выполняем команду SQL и закрываем соединение. Основной объект получения данных **SqlDataReader**.

```

public void FillGrid()

```

```

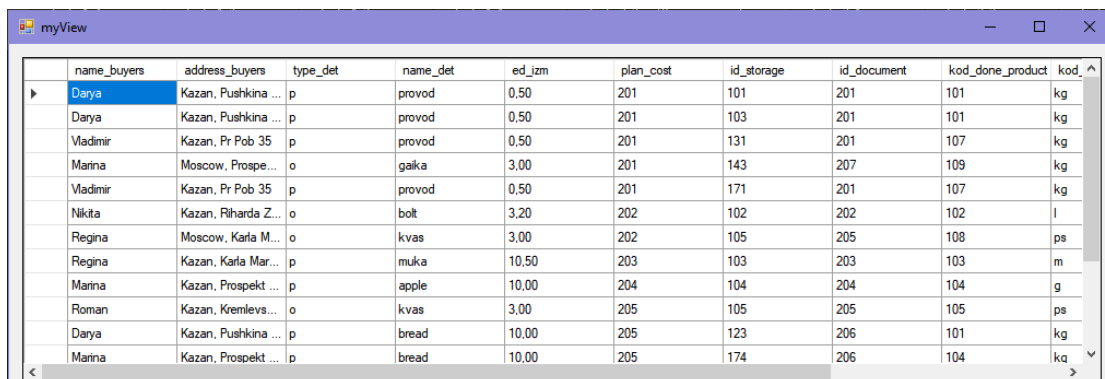
{
    kurs_workDataSet.zadacha1_1.Clear();
    SqlConnection con = new
SqlConnection(Properties.Settings.Default.kurs_workConnectionString);
    con.Open();
    string str = " SELECT buyers.kod_buyers, storage.id_storage, details.kod_det,
storage.date_ship, storage.kolvo, buyers.name_buyers FROM buyers INNER JOIN "
+
    " storage ON buyers.kod_buyers = storage.kod_buyers INNER JOIN
details ON storage.kod_done_product = details.kod_det" +
    " WHERE (buyers.address_buyers LIKE '%' + @City + '%)";
    SqlCommand cmd = new SqlCommand(str, con);
    cmd.Parameters.Add("@City", SqlDbType.VarChar).Value = txtCity.Text;
    SqlDataReader rdr = cmd.ExecuteReader();
    DataTable dt = new DataTable();
    dt.Load(rdr);
    con.Close();
    BindingSource bs = new BindingSource();
    bs.DataSource = dt;
    dataGridView1.DataSource = bs;
    dataGridView1.Refresh();
}

private void btnShow_Click(object sender, EventArgs e)
{
    FillGrid();
}

```

ТРЕТЬЯ ЧАСТЬ.

Создать представление (*VIEW*), которое показывает в виде одной таблицы данные из трех таблиц, соединенных в соответствии с межтабличными связями между ними.



	name_buyers	address_buyers	type_det	name_det	ed_izm	plan_cost	id_storage	id_document	kod_done_product	kod_
►	Darya	Kazan, Pushkina ...	p	provod	0,50	201	101	201	101	kg
	Darya	Kazan, Pushkina ...	p	provod	0,50	201	103	201	101	kg
	Vladimir	Kazan, Pr Pob 35	p	provod	0,50	201	131	201	107	kg
	Marina	Moscow, Prospe...	o	gaika	3,00	201	143	207	109	kg
	Vladimir	Kazan, Pr Pob 35	p	provod	0,50	201	171	201	107	kg
	Nikita	Kazan, Riharda Z...	o	bolt	3,20	202	102	202	102	l
	Regina	Moscow, Karla M...	o	kvas	3,00	202	105	205	108	ps
	Regina	Kazan, Karla Mar...	p	muka	10,50	203	103	203	103	m
	Marina	Kazan, Prospekt ...	p	apple	10,00	204	104	204	104	g
	Roman	Kazan, Kremlevs...	o	kvas	3,00	205	105	205	105	ps
	Darya	Kazan, Pushkina ...	p	bread	10,00	205	123	206	101	kg
	Marina	Kazan, Prospekt ...	p	bread	10,00	205	174	206	104	kg

К задаче-2 задания создать *SQL*-запрос с группировкой и решение с помощью record-ориентированных средств.

zadacha2_1

Kazan

	id_storage	kod_det	all_kolvo
▶	201	201	22
	205	201	5
	202	202	1
	203	203	3
	204	204	17
	205	205	9
	205	206	10
	208	206	5
	207	207	7
*			

Показать

Задача-2. Сведения об отгрузке деталей покупателям из Казани: номер склада; код детали; общее отгруженное количество.

Запрос:

```
SELECT s.id_storage, d.kod_det, SUM(s.kolvo) as all_kolvo
FROM storage s
INNER JOIN buyers b ON b.kod_buyers = s.kod_buyers
INNER JOIN details d ON s.kod_done_product = d.kod_det
WHERE (b.address_buyers LIKE '%' + 'Kazan' + '%')
GROUP BY s.id_storage, d.kod_det
```

Record-ориентированный подход:

```
public int Allkolvo(int idstorage, int koddoneprod, string city)
{
    int sum = 0;
    foreach (kurs_workDataSet.storageRow sRow in
kurs_workDataSet.storage.Rows)
    {
        if (sRow.id_storage==idstorage &&
koddoneprod==sRow.kod_done_product)
        {
            kurs_workDataSet.buyersRow dRow =
kurs_workDataSet.buyers.FindBykod_buyers(sRow.kod_buyers);
            string[] subs = dRow.address_buyers.Split(';');
            if (city == subs[0])
                sum += sRow.kolvo;
        }
    }
    return sum;
}
public void QueryZapr2()
```

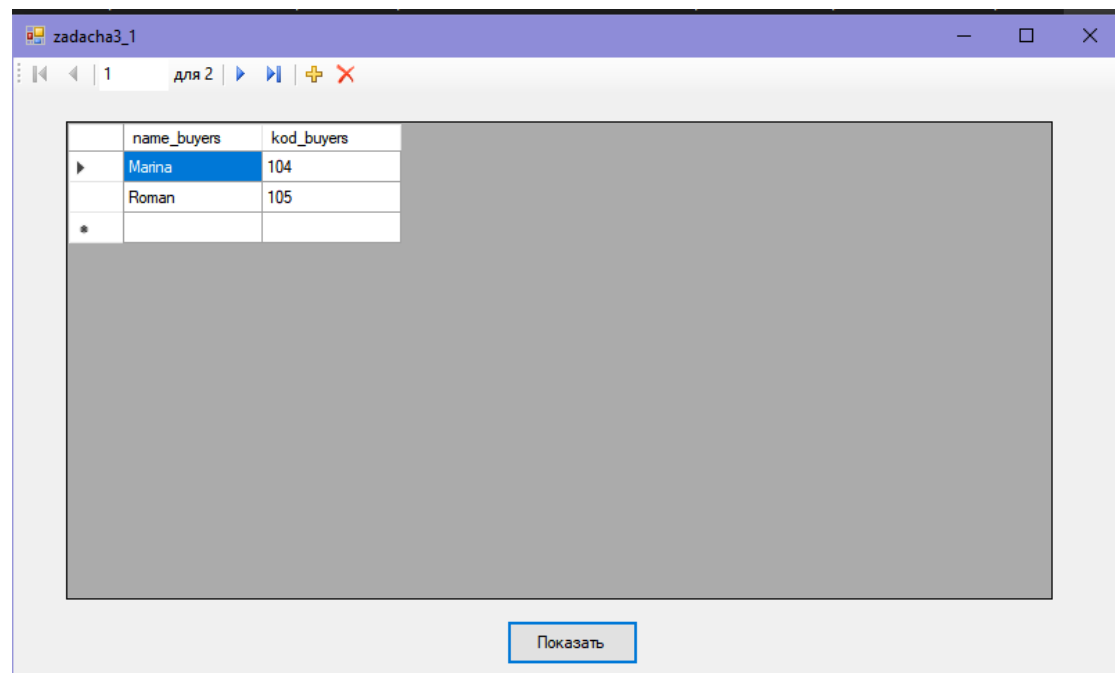
```

        {
            kurs_workDataSet.zadacha2_1.Clear();
            foreach (kurs_workDataSet.storageRow sRow in
kurs_workDataSet.storage.Rows)
            {
                foreach (kurs_workDataSet.detailsRow dRow in
kurs_workDataSet.details.Rows)
                {
                    foreach (kurs_workDataSet.buyersRow bRow in
kurs_workDataSet.buyers.Rows)
                    {
                        string[] subs = bRow.address_buyers.Split(',');
                        if (subs[0] == Convert.ToString(txtCity.Text) && (bRow.kod_buyers
== sRow.kod_buyers) && (dRow.kod_det == sRow.kod_done_product))
                        {
                            int sum = Allkolvo(sRow.id_storage, sRow.kod_done_product,
Convert.ToString(txtCity.Text));
                            bool was = false;
                            foreach (kurs_workDataSet.zadacha2_1Row zzRow in
kurs_workDataSet.zadacha2_1.Rows)
                            {
                                if (zzRow.kod_det == sRow.kod_done_product &&
zzRow.id_storage == sRow.id_storage)
                                {
                                    was = true;
                                    break;
                                }
                            }
                            if (!was)
                            {
                                kurs_workDataSet.zadacha2_1Row zRow =
kurs_workDataSet.zadacha2_1.Newzadacha2_1Row();
                                zRow.id_storage = sRow.id_storage;
                                zRow.kod_det = dRow.kod_det;
                                zRow.all_kolvo = sum;
                                kurs_workDataSet.zadacha2_1.Addzadacha2_1Row(zRow);
                            }
                        }
                    }
                }
            }
            dataGridView1.Refresh();
        }

private void btnShow_Click(object sender, EventArgs e)
{
    QueryZapr2();
}

```

К задаче-3 задания написать две программы: решение с помощью кванторного *SQL*-запроса с подзапросами и решение с помощью *record*-ориентированных средств, а также написать выражения на языках РА и РИК решающих данную задачу.



Задача-3. Все покупатели, такие что:

для некоторой детали с ценой > 100

все документы об отгрузке этой детали этому покупателю были только со склада 5.

Запрос:

```
SELECT b.name_buyers, b.kod_buyers
FROM buyers b
WHERE EXISTS (
```

```
    SELECT d.kod_det FROM details d, storage s --если деталь такая, то для
такой детали отгрузка была для нашего покупателя только с 205 склада
```

```
    WHERE d.plan_cost > 0.5 and d.kod_det = s.kod_done_product and
s.kod_buyers=b.kod_buyers
    and not EXISTS(
    SELECT * FROM storage s
    WHERE
    (s.kod_done_product = d.kod_det and
    s.kod_buyers = b.kod_buyers) and
    s.id_storage != 205
    ));
```

Решение с помощью *record*-ориентированных средств:

```
private void FillGridZ2()
{
    kurs_workDataSet.zadacha3_1.Clear();
```

```

        foreach (kurs_workDataSet.detailsRow dRow in
kurs_workDataSet.details.Rows)
        {

            if ((Convert.ToDouble(dRow.plan_cost) > 0.5))
                foreach (kurs_workDataSet.buyersRow bRow in
kurs_workDataSet.buyers.Rows)
                {
                    bool a = true;
                    bool b = false;
                    foreach (kurs_workDataSet.storageRow sRow in
kurs_workDataSet.storage.Rows)
                    {
                        if (sRow.kod_done_product == dRow.kod_det &&
sRow.kod_buyers == bRow.kod_buyers && sRow.id_storage == 205)
                        {
                            b = true;
                        }
                        else
                        {
                            if (sRow.kod_done_product != dRow.kod_det ||
sRow.kod_buyers != bRow.kod_buyers)
                                continue;
                            b = false;
                            break;
                        }
                    }

                    if (a == b)
                    {
                        bool was = true;
                        kurs_workDataSet.zadacha3_1Row zRow =
kurs_workDataSet.zadacha3_1.Newzadacha3_1Row();
                        zRow.name_buyers = bRow.name_buyers;
                        zRow.kod_buyers = bRow.kod_buyers;
                        foreach (kurs_workDataSet.zadacha3_1Row zzRow in
kurs_workDataSet.zadacha3_1.Rows)
                        {
                            if(zRow.kod_buyers==zzRow.kod_buyers)
                            {
                                was = false;
                                break;
                            }
                        }
                        if (was)
                            kurs_workDataSet.zadacha3_1.Addzadacha3_1Row(zRow);
                        else
                            zRow.Delete();
                    }
                }
        }

```



```

    }
    dataGridView1.Refresh();
}

```

Решение задачи 3: выражение на языке Реляционной Алгебры:

[name_buyers,kod_buyers]((([kod_det][plan_cost>0.5]((details)*(storage))-
[id_document][id_storage!=205](storage)*(buyers)*(details))*(buyers))

Решение задачи 3: запрос на языке Реляционного Исчисления Кортежей:

НАЙТИ{(b.name_buyers,b.kod_buyers)|b in buyers}
 EXISTS(d.kod_det | d in details) ((d.plan_cost>0.5 & d.kod_det =
 s.kod_done_product & s.kod_buyers=b.kod_buyers)& FORALL(s.id_document | s in
 storage) (s.kod_done_product = d.kod_det & s.kod_buyers = b.kod_buyers
 and s.id_storage=205))

Заключение.

В результате прохождения практики были освоены следующие компетенции:

ОПК-3 Способность применять и модифицировать математические модели для решения задач в области профессиональной деятельности

ОПК-4 Способность решать задачи профессиональной деятельности с использованием существующих информационно-коммуникационных технологий и с учетом основных требований информационной безопасности