Лабораторная работа №5

РАБОТА С SQL

Цель работы:
- изучение компонента TADOQuery;
- создание структурированных запросов SQL;
- разработка формы с реализацией различных запросов к БД кафе.

Краткие теоретические сведения

1 Компонент TADOQuery

Компонент TADOQuery предназначен для:

- формирования набора данных (НД), источником данных для которого могут служить записи как одной, так и нескольких таблиц БД;
- выполнения запросов к БД, не возвращающих наборов данных (добавление, изменение, удаление записей в таблицах БД и др.).

Результирующий НД компонента TADOQuery формируется путем выполнения запроса к БД на языке SQL (Structured Query Language, язык структурированных запросов).

Запросы, выполняемые компонентом TADOQuery - независимо от того, возвращают они набор данных или просто производят какие-либо действия в БД, - могут быть статическими и динамическими.

Статический запрос характерен тем, что описывающий его SQL-оператор не изменяется в процессе выполнения приложения.

SQL-оператор **динамического запроса** может частично изменяться в процессе выполнения приложения. В этом случае изменяемые части SQL-запроса оформляют в качестве параметров, значения которых могут многократно изменяться в процессе выполнения приложения. Таким образом, можно использовать один компонент TADOQuery для выполнения множества разнесенных во времени запросов к БД, различающихся по значению параметров. Заметим, что состав параметров также может меняться во время выполнения. Это более характерно для формируемых запросов - разновидности динамических запросов.

Формируемые запросы - такие запросы, текст SQL-оператора которых формируется программно в процессе выполнения приложения. Действия по формированию такого запроса состоят в очистке предыдущего содержимого свойства SQL и программного занесения в это свойство нового текста SQL-запроса (вид которого зависит от текущей ситуации и определяется рядом условий), а также в последующем его выполнении.

2 SQL оператор запросов SELECT

Оператор Select – наиболее часто используемый оператор SQL. Позволяет производить выборки данных из ТБД и преобразовывать к нужному виду полученные результаты. С его помощью можно реализовать весьма сложные условия выбора данных из различных таблиц.

В самом общем виде он имеет формат:

Что берём:

SELECT [DISTINCT | ALL] {* | <значение1> [, <значение2>...]}

Откуда:

FROM <таблица1> [, <таблица2>…]

Каким условиям отвечает:

[WHERE <условие отбора>]

Группировка по колонкам:

[GROUP BY столбец [, столбец1 …]

Условия на группы

[HAVING <условия поиска>]

Порядок вывода (сортировка):

[ORDER BY <список_столбцов>]

Всё что находится в [] скобках не является обязательным, т.е. для создания простейшего запроса выводящего всю таблицу, или выбранные из неё поля достаточно первых двух строк.

После ключевого слова **Select** приводится список значений, каждое из которых определяет столбец возвращаемого оператором результирующего набора данных. Звёздочка «*» указывает, что в результат запроса надо включить все столбцы той или иной таблицы.

После **From** указывается список ТБД, из которых будет происходить выборка данных.

Использование предложения WHERE.

В набор данных, возвращаемый оператором Select, будут включаться только те записи, которые удовлетворяют условию поиска, указанному после Where.

Варианты формирования условий поиска:

1. Сравнение значения столбца с константой или столбцом другой таблицы.

<имя столбца таблицы1> < оператор> < константа или имя столбца таблицы2 >

= равно	!< не меньше (больше или равно)
<меньше	!> не больше (меньше или равно)
> больше	<> не равно
<= меньше или равно	!= не равно

>= больше или равно |

WHERE Last_Name = ‘Петров’

2. Использование логических выражений.
   Строятся при помощи логических выражений AND, OR и NOT. Важно: операции отношения в них имеют меньший приоритет, чем логические операции, что избавляет от необходимости расстановки многочисленных скобок.

WHERE Last_Name = ‘Петров’ AND First_Name = ‘Иван’

3. Сравнение столбца с результатом вычисления.

<выражение> <оператор> <столбец> или <столбец> <оператор> <выражение>

Однако этот способ чаще применяется при использовании механизма вложенных подзапросов (вложенных операторов Select), речь о которых пойдёт ниже.

4. Использование BETWEEN.

В условии поиска можно указать, что некоторое значение (столбец или вычисленное значение) должно находиться в интервале между *значением 1* и *значением 2*

<значение> [NOT] BETWEEN < значение 1> AND < значение 2>

WHERE Oklad BETWEEN 2000 AND 3000

5. Использование IN.

Если нужно, чтобы значение какого либо столбца (или результат вычисление некоторого выражения) совпадало с одним из дискретных значений, в условии поиска указывается предложение

< значение > [NOT] IN (<значение 1> [, < значение 2>…])

WHERE Last_Name IN (‘Петров’, ‘Иванов’, ‘Сидоров’)

6. Использование функции UPPER.
   UPPER (<значение>)

Преобразует все буквы аргумента <значение> (содержимого столбца, результата вычисления выражения) к заглавным. Обычно эта функция используется в условиях поиска, когда необходимо игнорировать возможную разницу в высоте букв. Функция UPPER может фигурировать как в списке столбцов результирующего набора данных (после SELECT), так и в условии поиска в предложении WHERE.

WHERE UPPER(Last_Name) = ‘петров’

7. Использование LIKE.

Предложение LIKE определяет шаблоны сравнения строковых значений. Если необходимо, чтобы сравниваемое значение (значение столбца или результат вычисления строкового выражения) удовлетворяло шаблону, в условии поиска необходимо указать

<значение> [NOT] LIKE <шаблон> [ESCAPE <подшаблон>]

В шаблоне используются специальные символы – «%» (« *») и «_» (“?”). Символ «%» означает, что на его месте может быть строка любой длины, а символ «_» используется для указания любого единичного символа.

```
Select * from people where Last_Name like "И%"
```

Использование псевдонимов таблиц

Каждой таблице используемой в запросе можно присвоить краткое имя (например, просто букву), а затем обращаться по псевдониму.

```
SELECT ....
FROM <таблица1 псевдоним1> [, <таблица2 псевдоним2>…]
WHERE ....
```

Внутреннее соединение таблиц

Необходимо когда вы используете в запросе более чем одну связанную таблицу, т.е. для правильной работы запроса необходимо связать таблицы. Эта процедура похожа на создание реляционных отношений.

<Имя столбца 1 таблицы> = <Имя столбца 2 таблицы>

Это условие указывается в предложении WHERE и далее через логический оператор AND указываются условия запроса.

Предложение ORDER BY – определение сортировки

Результирующий НД можно отсортировать с помощью предложения:

ORDER BY <список_столбцов> [asc или desc]

Список столбцов содержит имена столбцов, по которым будет производиться сортировка. Если указаны два или более столбцов, первый столбец будет использован для глобальной сортировки, второй столбец – для сортировки внутри группы, определяемой единым значением первого столбца, и т.д. При указании asc (принято по умолчанию) сортировка происходит по возрастанию, desc – сортировка производится по убыванию.

Расчёт значений вычисляемых столбцов

Для расчёта значений вычисляемых столбцов результирующего НД используются арифметические значения. При этом в списке возвращаемых

столбцов после SELECT вместо имени вычисляемого столбца указывается выражение:

SELECT [DISTINCT | ALL] {* | <столбец1> [, <выражение1>...]}
FROM <таблица1> [, <таблица2>…]

Ели вы хотите задать новое временное имя столбца

Это особенно актуально для вычисляемых столбцов и столбцов при слиянии со строкой, после имени столбца или выражения нужно указать ключевое слово AS и новое имя столбца:

SELECT ... {* | <значение1> [, <выражение1 [AS <имя столбца>]>...]}
SELECT Last_Name AS Фамилия

Агрегирующие функции

Агрегирующие функции предназначены для вычисления итоговых значений операций над всеми записями НД.

К агрегирующим относятся следующие функции:

- COUNT (<выражение>) –Подсчитывает число вхождений значения выражения во все записи результирующего НД;
- SUM (<выражение>) - Суммирует значения выражения;
- AVG (<выражение>) – Находит среднее значение;
- MAX (<выражение>) – Определяет максимальное значение;
- MIN (<выражение>) – Определяет минимальное значение.

Если из группы одинаковых записей нужно учитывать только одну, перед выражением в скобках включают слово DISTINCT:

COUNT (DISTINCT <выражение>)

Чаще всего в качестве выражения выступают имена столбцов, при этом выражение может вычисляться и по значениям нескольких таблиц.

Группировка записей

Иногда требуется получить агрегированные значения (минимум, максимум, среднее) не по всему результирующему НД, а по каждой из входящих в него групп записей, характеризующихся одинаковым значением какого-либо столбца. Например, выдать общее количество сотрудников в каждом отделе. В этом случае в оператор SELECT вводится предложение

GROUP BY столбец [,столбец1 …]

При этом необходимо, чтобы один из столбцов результирующего НД был представлен агрегатной функцией.

Предложение HAVING – наложение ограничений на группировку записей

Если нужно в результирующем НД выдавать агрегацию не по всем группам, а только по тем из них, которые отвечают некоторому условию, после предложения GROUP BY указывают предложение: HAVING <агрегатная функция> <отношение> <значение>

Где

- Агрегирующая функция – одна из функций MIN, MAX, AVG и SUM;
- Отношение – одна из операций отношения =, <>, <, <=, >, >=;
- Значение – константа, результат вычисления выражения или единичное значение, возвращаемое вложенным оператором SELECT.

Таким образом, после HAVING указываются условия, которые отличаются от условий, определяемых в предложении WHERE, одним важным обстоятельством: в HAVING обязательно должна быть указана одна из агрегатных функций, в то время как в предложении WHERE такие функции указывать нельзя.

Использование подзапросов

Часто невозможно решить поставленную задачу путём использования единственного запроса. Например, в тех случаях, когда при использовании условия поиска <сравниваемое значение> <оператор> <значение, с которым сравнивать> в предложении WHERE параметр <значение, с которым сравнивать> заранее не определён и должен вычисляться в момент выполнения оператора SELECT или представляет собой не одно, а несколько значений. В такого рода случаях используется подзапросы (вложенные запросы). Оператор SELECT с подзапросом имеет такой вид:

SELECT …
FROM …
WHERE <сравниваемое значение> <оператор> (SELECT …)

Синтаксис вложенного запроса ни чем не отличается от синтаксиса основного запроса и, следовательно, вложенный запрос может в свою очередь содержать подзапрос. Заметим, что SQL – синтаксис требует заключать подзапрос в круглые скобки.

Замечание: распространённой ошибкой является использование вложенного оператора SELECT, который вместо единичного значения способен возвращать список значений.

Использование IS NULL

Если требуется выдать все записи, в которых некоторый столбец (или результат вычисления выражения) имеет значение NULL (т.е. не имеет никакого значения), достаточно в условии поиска указать предложение:

<значение> IS [NOT] NULL

Использование операции сцепления строк

Операция + соединяет два строковых значения, которые могут быть представлены выражениями:

<строковое выражение1> + <строковое выражение2>

Эту операцию можно использовать как после слова SELECT для указания возвращаемых значений, так и в предложении WHERE.

```
SELECT Last_Name + ‘ ‘+ First_Name +‘ ‘ + Second_Name as ФИО
FROM People
WHERE Last_Name = ‘Макаров’
```

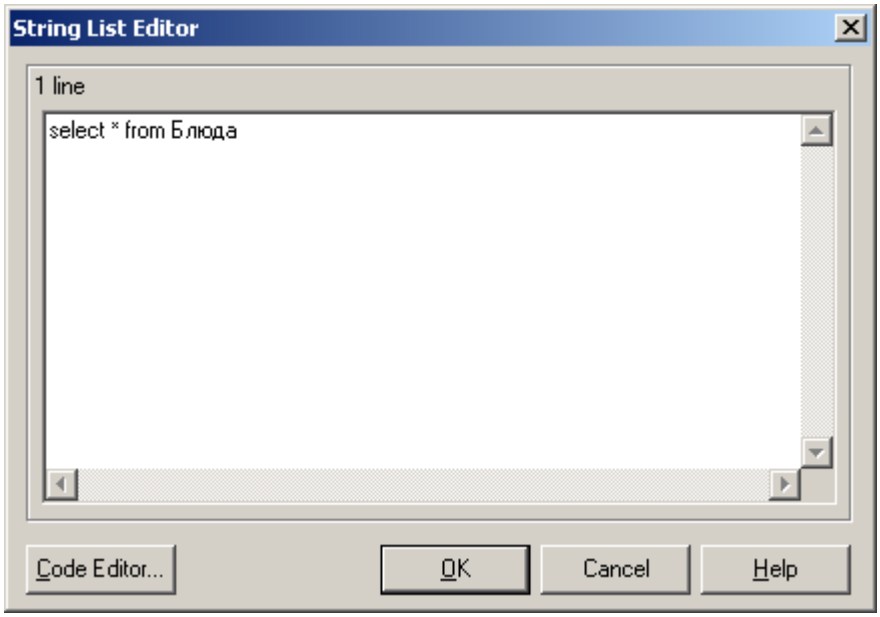
3 Соединение компонента TADOQuery и визуальных компонентов для работы с данными

Соединение компонента TADOQuery с визуальными компонентами происходит через промежуточный компонент TDataSource. Для этого в компоненте TDataSource в свойстве DataSet необходимо указать имя компонента TADOQuery. В визуальных компонентах (TDBGrid, TDBEdit и т.д.) в свойстве DataSource указывается имя компонента TDataSource и, если необходимо, в свойстве DataField выбирается имя интересующего поля.

4 Выполнение статических запросов

Для формирования статического запроса необходимо:

1. Выбрать для существующего компонента TADOQuery в инспекторе объектов свойство SQL и нажать кнопку в правой части строки;
2. В появившемся окне текстового редактора набрать текст SQL-запроса:



3. Установить свойство Active компонента TADOQuery в True, если НД должен быть открыт в момент начала работы приложения или оставить свойство Active в состоянии False, если открытие НД будет производиться в программе в некоторый момент работы приложения.

Формирование текста SQL-оператора SELECT может осуществляться и программно.

5 Методы открытия и закрытия компонента TADOQuery

Компонент TADOQuery может возвращать НД (если компонент использует оператор SELECT, то есть осуществляет выборку из одной или более таблиц БД) и выполнять действие над одной или более таблицей БД (SQL-операторы INSERT, UPDATE, DELETE).

В случае использования оператора SELECT после открытия компонента TADOQuery возвращается НД, в котором указатель текущей записи всегда установлен на первую запись (если она имеется). Такой компонент TADOQuery следует открывать:

- установкой свойства Active в значение true,
- или выполнением метода

procedure Open();

В случае использования операторов INSERT, UPDATE, DELETE набор данных не возвращается. Такой компонент TADOQuery следует открывать, выполняя метод

procedure ExecSQL();

Метод ExecSQL посылает серверу для выполнения SQL-оператор и свойства SQL данного компонента TADOQuery.

Закрытие компонента TADOQuery осуществляется методом

procedure Close();

или установкой в False свойства Active, например:

При этом следует помнить, что для компонента TADOQuery, не возвращающего набор данных, выполнение метода Close не имеет последствий, поскольку с данным компонентом не связан открытый НД. Для динамических запросов, особенно для отсылаемых к удаленной БД, полезно использовать методы, осуществляющие "связывание" параметров с их фактическими значениями (Prepare) и отменяющие такое "связывание" (UnPrepare).

6 Выполнение динамических запросов

Динамическим (параметрическим) является запрос, в SQL-операторе которого в процессе выполнения приложения могут изменяться отдельные

его составляющие. В этом случае изменяемая часть оператора оформляется как Параметры.

Например, пусть в процессе выполнения приложения может быть выдан запрос: выдать все записи из таблицы Contract, составленные 01 марта 2004 г:

```
SELECT * FROM CONTRACT
WHERE CreateDay = "01.03.08"
```

И запрос: выдать все записи из таблицы Contract, составленные 10 марта 2004 г:

```
SELECT * FROM CONTRACT
WHERE CreateDay = "10.03.08"
```

Также аналогичный запрос по договорам за другие даты.

SQL оператор, в котором изменяющиеся части заменены на параметры:

```
SELECT * FROM CONTRACT
WHERE CreateDay = :DayCreate
```

Под параметром понимается имя, предваренное кавычками “:”.

В динамических запросах параметры всегда заменяют значения, которые могут изменяться в процессе выполнения. Имена параметров произвольны и могут не совпадать со значениями полей таблицы, которым они обычно ставятся в соответствие.

Заметим, что такой подход не годится для случая удаления записей при помощи SQL-оператора DELETE, поскольку после удаления в НД не существует записи с запомненным значением уникального поля (полей).

7 Установка значений параметров динамического запроса во время выполнения

Самым распространенным способом указания текущих значений параметров является их ввод пользователем в поля ввода (компоненты TEdit и другие) и последующее программное назначение параметров.

Параметры компонента TADOQuery доступны через его свойство

ADOQuery->Parameters

Обратиться к конкретному параметру можно используя:

function ParamByName(const AnsiString Value);

где Value определяет имя параметра.

Для установки значения конкретного параметра используется одно из свойств компонента TParam AsNNN (AsString, Aslnteger и т.д.) или более общее свойство

property Value – значение в параметре;

8 Формируемые запросы

Часто один компонент TADOQuery используют для выполнения различных отстоящих друг от друга во времени запросов. Такой подход уменьшает число используемых компонентов, но может привести к возрастанию программного кода.

Свойство SQL компонента TADOQuery имеет тип TStrings:

и потому содержимое свойства SQL может формироваться программно методами Add (добавить элемент), Delete (удалить элемент), Clear (очистить список) и прочими для TStrings.

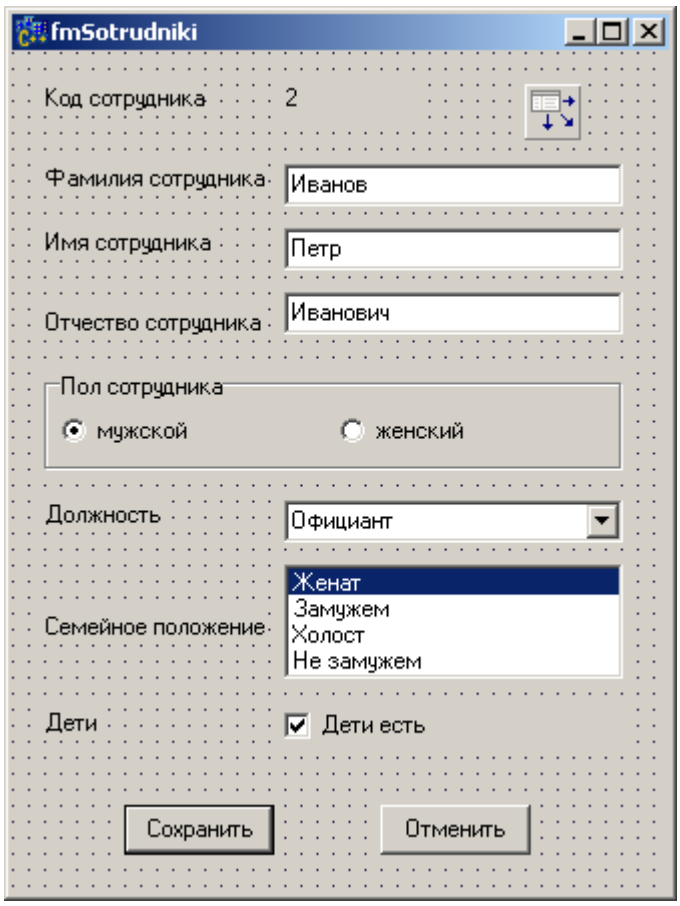
Например:

```
ADOQuery1->Close();
ADOQuery1->SQL->Clear();
ADOQuery1->SQL->Add("SELECT EmpNo, LastName, FirstName,
HireDate");
ADOQuery1->SQL->Add("FROM Employee");
ADOQuery1->Open();
```

Свойство TQuery->SQL->Text позволяет одной строкой кода задать весь текст запроса, затирая предыдущее значение свойства SQL.

Задание №1. Статические запросы.

- *Перейдите на форму fmSotrudniki:*



Будем должность в DBComboBox1 формировать из значений должностей в таблице Сотрудники.

Для этого необходимо написать запрос:
SELECT Должность FROM Сотрудники
GROUP BY Должность

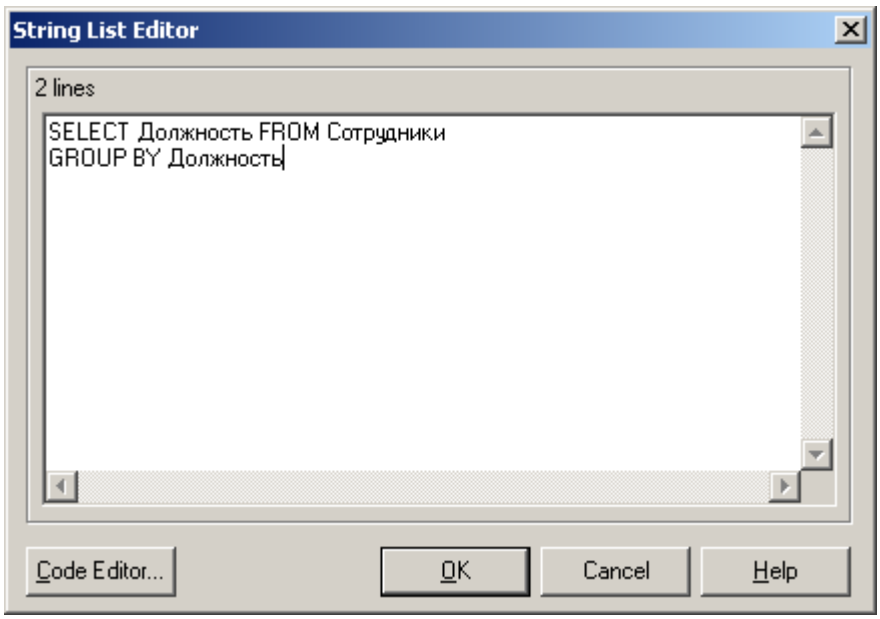
Данный запрос меняться в программе не будет, следовательно, можно реализовать его как статический запрос.

Для этого с закладки ADO разместите на форму fmSotrudniki компонент ADOQuery1. Для него укажите через Object Inspector:

- свойство Name в ADOQueryJob;

- свойство Connection в DMMain->ADOConnection1;

-свойство SQL в:



- свойство Active в true.

- На момент показа формы fmSotrudnik будем формировать в DBComboBox1 пункты по записям из запроса. Для этого в событии OnShow у fmSotrudniki пропишите:

```
DBComboBox1->Items->Clear();
ADOQueryJob->Open();
ADOQueryJob->First();
while (!ADOQueryJob->Eof)
{
    DBComboBox1->Items->Add(ADOQueryJob->FieldByName("Должность")->AsString);
    ADOQueryJob->Next();
}
ADOQueryJob->Close();
```

- Нажмите F9.

- Посмотрите, что в DBComboBox1 список состоит из значений в поле Должность таблицы Сотрудники.

Задание №2. Формируемые запросы

Будем отражать работу с запросами через отдельную форму.

- Нажмите в меню File/New/Form.

- Свойство Name у новой формы напишите в fmSQL.

- Сохраните форму в проект. Для этого нажмите File/Save all и в диалоговом окне сохраните новую форму в той же папке, что и другие модули проекта. При сохранении укажите новому модулю имя SQL.cpp.

- Перейдите на форму fmMain.

- Подключите на форме fmMain форму fmSQL.

Для этого, находясь на форме fmMain, выберете в меню File/Include Unit Hdr…(или нажмите Alt+F11). В диалоговом окне Use Unit выберете модуль SQL.

После этого в модуле Main.cpp появится запись #include "SQL.h".

На форме fmMain.

- Для акции actSQL в ActionList1 пропишите в событии OnExecute:

```
fmSQL->Caption="Запросы";
fmSQL->ShowModal();
```

- Наша акция actSQL уже выбрана для пунктов меню и панели инструментов.

- Перейдите на форму fmSQL.

- Подключите на форме fmSQL модуль данных. Для этого, находясь на форме fmSQL, выберете в меню File/Include Unit Hdr…(или нажмите Alt+F11). В диалоговом окне Use Unit выберете модуль DM.

После этого в модуле SQL.cpp появится запись #include "DM.h".

- Разместите с закладки Win32 компонент PageControl1.

- Установите для PageControl1 свойство Align (выравнивание) в alClient.

- Нажмите правую кнопку на PageControl1 и выберете команду New Page. Появится новая закладка TabSheet1. Для нее установите свойство Name в TabSheetPrimers, свойство Caption в Формируемые запросы.

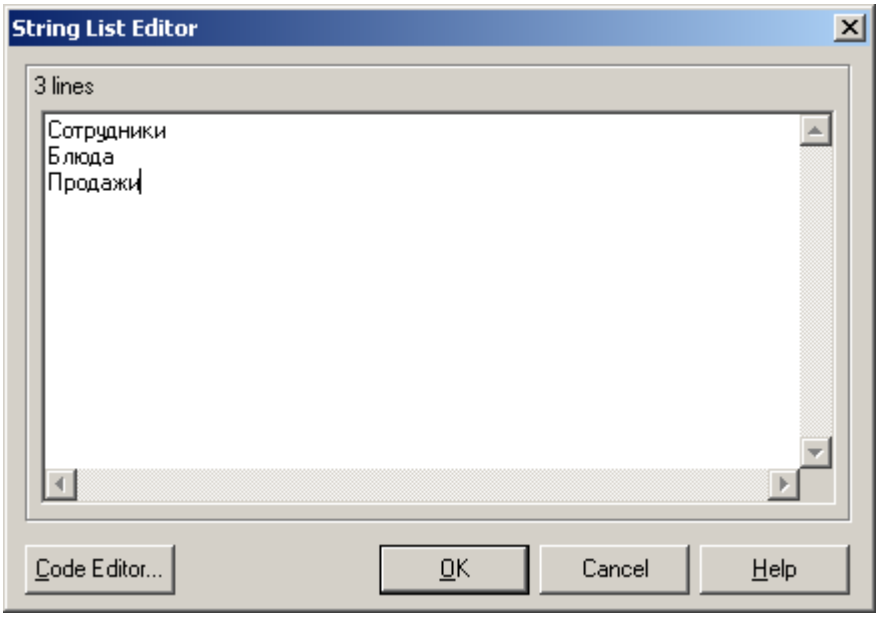
-Разместите на TabSheetPrimers с закладки ADO компонент ADOQuery1. Свойство Name укажите для него ADOQueryPrimers. Свойство Connection укажите в DMMain->ADOConnection1.

Будем для ADOQueryPrimers свойство SQL формировать программно (формируемы запросы) на примере показа данных из разных таблиц БД и отображать в визуальных компонентах.

- Разместите на TabSheetPrimers с закладки Data Access компонент DataSource1. Свойство Name для него укажите в dsPrimers. Свойство DataSet для dsPrimers укажите в ADOQueryPrimers.

- Разместите с закладки Standart компонент RadioGroup1 на TabSheetPrimers. Свойство Name укажите для него в RadioGroupPrimers. Укажите для него свойство Caption в «Выбрать данные», свойство Align в alTop.

- В свойстве Items у RadioGroupPrimers напишите:



Примечание:

Свойство ItemIndex у TRadioGroup хранит, какая радиокнопка нажата: -1 – не нажата ни одна, 0 – нажата первая кнопка и далее.

- Разместите на TabSheetPrimers компонент DBGrid1 с закладки Data Controls. Укажите для него свойство Name в DBGridPrimers, свойство DataSource в dsPrimers, свойство Align в alClient.

- Нажмите на RadioGroupPrimers два раза и пропишите в событии OnClick:

```
ADOQueryPrimers->Close();
switch (RadioGroupPrimers->ItemIndex)
{
    case 0:
        ADOQueryPrimers->SQL->Text="SELECT * FROM
Сотрудники";
        break;
    case 1:
        ADOQueryPrimers->SQL->Text="SELECT Название, Тип, Выход
FROM Блюда";
        break;
    case 2:
        ADOQueryPrimers->SQL->Clear();
        ADOQueryPrimers->SQL->Add("SELECT Номер_продажи, Дата,
Фамилия, Имя, Отчество");
        ADOQueryPrimers->SQL->Add("FROM Продажи, Сотрудники");
        ADOQueryPrimers->SQL->Add("WHERE
Продажи.Код_сотрудника=Сотрудники.Код_сотрудника");
        break;
    }
```

ADOQueryPrimers->Open();

- Нажмите F9. Посмотрите результаты выполнения запросов в приложении.

Задание №3. Динамические (параметрические) запросы

- Нажмите правую кнопку на PageControl1 и выберете команду New Page. Появится новая закладка TabSheet1. Для нее установите свойство Name в TabSheetParams, свойство Caption в «Параметрические запросы».

-Разместите на TabSheetParams:

- с закладки Standart компонент GroupBox1. Укажите для него свойство Caption в «Прибыль за период».

- с закладки Standart компонент GroupBox2. Укажите для него свойство Caption в «Продажи сотрудников».

- Разместите на GroupBox1 с закладки Standart компонент Label1, для которого свойствоName укажите в LabelDate1, свойство Caption установите в Дата начала периода.

- Разместите на GroupBox1 рядом с LabelDate1 с закладки Win32 компонент DateTimePicker1, для которого свойствоName укажите в DateTimePickerDate1.

- Разместите на GroupBox1 с закладки Standart компонент Label1, для которого свойствоName укажите в LabelDate2, свойство Caption установите в Дата окончания периода.

- Разместите на GroupBox1 рядом с LabelDate2 с закладки Win32 компонент DateTimePicker1, для которого свойствоName укажите в DateTimePickerDate2.

- Разместите на GroupBox1 с закладки Standart компонент Button1, для которого свойствоName укажите в ButtonResult, свойство Caption установите в Прибыль за период.

-Разместите на GroupBox1 с закладки Standart компонент Memo1, для которого свойство Name укажите в MemoResult, свойство Lines очистите.

Будем выводить результат вычисления прибыли за период в MemoResult.

- С закладки ADO разместите компонент ADOQuery1. Свойство Name укажите для него ADOQueryResult. Свойство Connection в DMMain->ADOConnection1.

- В свойстве SQL у ADOQueryResult пропишите запрос:

```
SELECT Sum(CCur([Количество]*[Цена]*(1-[Скидка]))) AS Прибыль
FROM Продажи, Продано
```

WHERE (Продажи.Дата Between :d1 And :d2) AND Продажи.Номер_продажи=Продано.Номер_продажи

Где CCur – функция Access для перевода в денежный формат; d1 и d2 – параметры.

Чтобы определить данные параметры нажмите свойство Parameters у ADOQueryResult. В диалоговом окне определите параметры d1 и d2 через свойство Name для параметров в d1 и d2 соответственно, свойство DataType в ftDateTime для обоих параметров.

Даты периода для вычисления прибыли могут меняться пользователем. В этом случае запрос на вычисление прибыли должен выполняться несколько раз с параметрами, задаваемыми через DateTimePickerDate1 и DateTimePickerDate2.

- На событии OnClick у ButtonResult пропишите:

```
ADOQueryResult->Close();
ADOQueryResult->Parameters->ParamByName("d1")->Value=DateTimePickerDate1->Date;
ADOQueryResult->Parameters->ParamByName("d2")->Value=DateTimePickerDate2->Date;
ADOQueryResult->Open();
MemoResult->Text="Прибыль за период \r\nc" + DateToStr(DateTimePickerDate1->Date)+
        "по "+DateToStr(DateTimePickerDate2->Date)+"\r\nссоставляет: "+ADOQueryResult->FieldByName("Прибыль")->AsString+" руб.";
}
```

Где \r\nc – переход на новую строку.

- Разместите на компоненте GroupBox2 с закладки Standart компонент Label1, для которого свойствоName укажите в LabelFam, свойство Caption установите в Фамилия сотрудника.

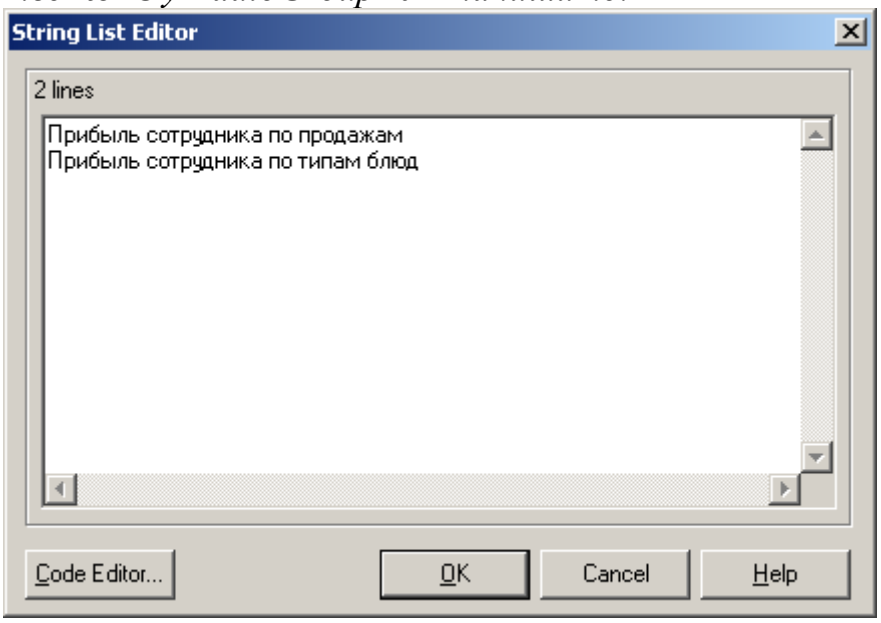
- Разместите на GroupBox2 рядом с LabelFam с закладки Standart компонент Edit1, для которого свойствоName укажите в EditFam, свойство Text очистите.

- Разместите на GroupBox2 с закладки Standart компонент Button1, для которого свойствоName укажите в ButtonFam, свойство Caption установите в Прибыль сотрудников.

Для просмотра детализации прибыли сотрудников разместите на GroupBox2 с закладки Standart компонент RadioGroup1. Свойство Name

укажите для него в RadioGroupFam. Укажите для него свойство Caption в Просмотреть детализацию по:

- В свойстве Items у RadioGroupFam напишите:



-С закладки ADO разместите компонент ADOQuery1. Свойство Name укажите для него ADOQueryFam. Свойство Connection в DMMain->ADOConnection1.

-В свойстве SQL для ADOQueryFam напишите:

```
SELECT Сотрудники.Код_сотрудника, Фамилия, Имя, Отчество,
Sum(CCur(Количество*Цена*(1-скидка))) AS Прибыль
FROM Сотрудники, Продажи, Блюда, Продано
WHERE Фамилия LIKE :p  AND Блюда.Код_блюда = Продано.Код_блюда
AND Продажи.Номер_продажи = Продано.Номер_продажи AND
Сотрудники.Код_сотрудника = Продажи.Код_сотрудника
GROUP BY Сотрудники.Код_сотрудника, Фамилия, Имя, Отчество
```

Где p – параметр.
Чтобы определить данный параметр нажмите свойство Parameters у ADOQueryFam. В диалоговом окне определите параметр p через свойство Name.

- Разместите на GroupBox2 с закладки Data Access компонент DataSource1. Свойство Name для него укажите в dsFam. Свойство DataSet для dsFam укажите в ADOQueryFam.

- Разместите на GroupBox2 компонент DBGrid1 с закладки Data Controls. Укажите для него свойство Name в DBGridFam, свойство DataSource в dsFam.

-На событии OnClick у ButtonFam запишите:

```
ADOQueryFam->Close();
```

```
ADOQueryFam->Parameters->ParamByName("p")->Value=EditFam->Text+"%";
ADOQueryFam->Open();
if (ADOQueryFam->RecordCount==0)
Application->MessageBox("Таких сотрудников нет", "Внимание", MB_ICONINFORMATION);
```

Где символ % - замена нескольких символов. Таким образом, отбираем сотрудников по первым буквам фамилии.

RecordCount – определяет количество записей в НД.

- На событии OnClick у RadioGroupFam пропишите:

```
ADOQueryFam->Close();
    switch (RadioGroupFam->ItemIndex)
{
    case 0: //нажата первая кнопка
        ADOQueryFam->Parameters->ParamByName("p")->Value=EditFam->Text+"%";
        ADOQueryFam->SQL->Strings[0]="SELECT Сотрудники.Код_сотрудника, Фамилия, Имя, Отчество, Продажи.Номер_продажи, Sum(CCur([Количество]*[Цена]*(1-[скидка]))) AS Прибыль";
        ADOQueryFam->SQL->Strings[3]="GROUP BY Сотрудники.Код_сотрудника, Сотрудники.Фамилия, Сотрудники.Имя, Сотрудники.Отчество, Продажи.Номер_продажи";
        break;
    case 1: //нажата вторая кнопка

        ADOQueryFam->Parameters->ParamByName("p")->Value=EditFam->Text+"%";
        ADOQueryFam->SQL->Strings[0]="SELECT Сотрудники.Код_сотрудника, Фамилия, Имя, Отчество, Тип, Sum([Количество]*[Цена]*(1-[скидка])) AS Прибыль";
        ADOQueryFam->SQL->Strings[3]="GROUP BY Сотрудники.Код_сотрудника, Сотрудники.Фамилия, Сотрудники.Имя, Сотрудники.Отчество, Тип";
        break;
    }
    ADOQueryFam->Open();
```

- Нажмите F9. Посмотрите результаты выполнения запросов в приложении.

Задание №4. Запросы манипулирования данными

- Нажмите правую кнопку на PageControl1 и выберете команду New Page. Появится новая закладка TabSheet1. Для нее установите свойство Name в TabSheetModify, свойство Caption в Запросы изменения.

-Разместите на TabSheetModify:

- с закладки ADO компонент ADOQuery1. Свойство Name укажите для него ADOQueryModify. Свойство Connection в DMMain->ADOConnection1.

Будем для него свойство SQL формировать программно для запросов на добавление, редактирование и удаление данных на примере таблицы БД Сотрудники.

Чтобы видеть отработку результатов на изменение данных разместите на TabSheetModify:

- компонент DBGrid1 с закладки Data Controls. Укажите для него свойство Name в DBGridModify, свойство DataSource в fmList->dsDishes (предварительно подключив модуль List), свойство DataSet которого установлено на форме fmList в DMMain->ADOT_Dishes .

- С закладки Standart разместите на TabSheetModify компонент Label1, для которого свойствоName укажите в LabelName, свойство Caption установите в Название блюда.

- Рядом с LabelName с закладки Standart разместите на TabSheetModify компонент Edit1. Для него свойство Name установите в EditName. Свойство Text очистите.

- Под LabelName разместите с закладки Standart на TabSheetModify компонент Label1, для которого свойство Name укажите в *LabelType , свойство Caption установите в Тип блюда.*

- Рядом с LabelType с закладки Standart разместите на TabSheetModify компонент Edit1. Для него свойство Name установите в EditType. Свойство Text очистите.

- С закладки Standart разместите на TabSheetModify компонент Button1, для которого свойство Name укажите в ButtonInsert, свойство Caption установите в Добавить.

- С закладки Standart разместите на TabSheetModify компонент Button1, для которого свойство Name укажите в ButtonUpdate, свойство Caption установите в Изменить.

- С закладки Standart разместите на TabSheetModify компонент Button1, для которого свойство Name укажите в ButtonDelete1, свойство Caption установите в «Удалить по (Названию) или (Типу) или (Названию и Типу)».

- С закладки Standart разместите на TabSheetModify компонент Button1, для которого свойство Name укажите в ButtonDelete2, свойство Caption установите в «Удалить по Названию и Типу совместно».

Кнопка Добавить:

Будем добавлять в таблицу Блюда, блюда, для которых указаны Название и Тип в строках ввода EditName и EditType (через их свойство Text).

В данном примере проводится добавление данных в таблицу Блюда по двум полям (Название и Тип). Остальным полям будет присвоено пустое значение. В нашей БД такая ситуация возможна, так как в нашей БД Название и Тип должны быть обязательно указаны, для остальных полей таких требований не было.

В нашем случае пользователь может много раз задавать название и тип для нового блюда. В этом случае запрос на добавление должен выполняться несколько раз с параметрами, задаваемыми через строки ввода EditName и EditType.

- На событии OnClick у ButtonInsert пропишите:

```
LabelName->Caption="Новое название блюда";
LabelType->Caption="Новый тип блюда";
 ADOQueryModify->Close();
 ADOQueryModify->SQL->Clear();
 ADOQueryModify->SQL->Add("INSERT INTO Блюда (Название, Тип)");
 ADOQueryModify->SQL->Add("VALUES (:n, :t)");
 ADOQueryModify->Parameters->ParamByName("n")->Value=EditName->Text;
ADOQueryModify->Parameters->ParamByName("t")->Value=EditType->Text;
ADOQueryModify->ExecSQL();
Application->MessageBox("Будет добавлена запись", "Внимание", MB_ICONINFORMATION);
// Чтобы обновить данные в НД для Блюд после добавления
DMMain->ADOT_Dishes->Close();
DMMain->ADOT_Dishes->Open();
```

Кнопка Изменить:

Будем изменять название блюда в значение введенное в строку ввода EditName при условии, что тип блюда соответствует типу блюда, введенному в EditType.

В нашем случае пользователь может много раз задавать название и тип для изменяемого блюда. В этом случае запрос на обновление должен выполняться несколько раз с параметрами, задаваемыми через строки ввода EditName и EditType.

- На событии OnClick у ButtonUpdate пропишите:

```
LabelName->Caption="Смена названия блюда";
LabelType->Caption="Для типа";
ADOQueryModify->Close();
```

```
ADOQueryModify->SQL->Clear();
 ADOQueryModify->SQL->Add("UPDATE Блюда  SET Название=:n");
 ADOQueryModify->SQL->Add("WHERE Тип =:t");
ADOQueryModify->Parameters->ParamByName("n")->Value=EditName->Text;
ADOQueryModify->Parameters->ParamByName("t")->Value=EditType->Text;
 ADOQueryModify->ExecSQL();
//Application->MessageBox("Будет изменена запись", "Внимание", MB_ICONINFORMATION);
DMMain->ADOT_Dishes->Close();
DMMain->ADOT_Dishes->Open();
```

Будем удалять блюдо по названию и типу по разным правилам:

1) по логике OR:

можно удалять только по типу, можно удалять только по названию, можно удалять по типу и названию.

2) по логике AND:

можно удалять только по типу и названию.

a	b	a AND b	a OR b
1	1	1	1
1	0	0	1
1	–	–	1
0	1	0	1
0	0	0	0
0	–	0	–
–	1	–	1
–	0	0	–
–	–	–	–

В нашем случае пользователь может много раз задавать название и тип для удаляемого блюда. В этом случае запрос на удаление должен выполняться несколько раз с параметрами, задаваемыми через строки ввода EditName и EditType.

Кнопка Удалить по (Названию) или (Типу) или (Названию и Типу):

- В событии OnClick у ButtonDelete1 пропишите:

```
LabelName->Caption="Удаление блюда с названием";
LabelType->Caption="Удаление блюда с типом";
ADOQueryModify->Close();
ADOQueryModify->SQL->Clear();
```

```
ADOQueryModify->SQL->Add("DELETE FROM Блюда");
ADOQueryModify->SQL->Add("WHERE Название =:n OR Тип=:t");
ADOQueryModify->Parameters->ParamByName("n")->Value=EditName->Text;
ADOQueryModify->Parameters->ParamByName("t")->Value=EditType->Text;
ADOQueryModify->ExecSQL();
DMMain->ADOT_Dishes->Close();
DMMain->ADOT_Dishes->Open();
```

Кнопка Удалить по Названию и Типу совместно:

- В событии OnClick у ButtonDelete2 пропишите:

```
LabelName->Caption="Удаление блюда с названием";
LabelType->Caption="Удаление блюда с типом";
ADOQueryModify->Close();
ADOQueryModify->SQL->Clear();
ADOQueryModify->SQL->Add("DELETE FROM Блюда");
ADOQueryModify->SQL->Add("WHERE Название =:n AND Тип=:t");
ADOQueryModify->Parameters->ParamByName("n")->Value=EditName->Text;
ADOQueryModify->Parameters->ParamByName("t")->Value=EditType->Text;
ADOQueryModify->ExecSQL();
DMMain->ADOT_Dishes->Close();
DMMain->ADOT_Dishes->Open();
```

- *Нажмите F9.*
- *Посмотрите отработку запросов в приложении.*

Задание 5. Оформление отчета к лабораторной работе

Оформить отчёт со следующим содержанием:

1. Титульный лист.
2. Цель работы.
3. Постановка задачи.
4. Краткая теория и ход выполнения заданий.
5. Описание результатов.

Приведите скриншоты полученной программы и листинг программы.
Покажите отработку запросов в приложении.

6. Заключение (выводы).