Тема 14. Состояния и режимы наборов данных

Наборы данных могут быть в открытом или закрытом состояниях. Текущее состояние набора данных указывается в свойстве Active: Active=True - набор открыт, иначе – закрыт. Открытый компонент Table содержит набор данных, соответствующий таблице TableName. Набор данных Query соответствует результату выполнения SQL-запроса, содержащегося в свойстве SQL компонента.

Пример.

Открытие набора данных Query.

```
procedure TForm1.Button1Click(  );
begin
Query1.Active:=False;
Query1.SQL.Clear;
Query1.SQL.Add( ‘SELECT * FROM  имя_файла.dbf’ );
Query1.Active:=True;
end;
```

Открывать и закрывать наборы данных можно также методами Open и Close. При вызове этих методов генерируются события BeforeOpen, AfterOpen, BeforeClose, AfterClose.

При закрытии набора данных текущая запись не сохраняется; для ее сохранения нужно использовать метод Post.

Режим – это уточненное состояние набора данных. С каждым режимом связано множество допустимых операций над набором данных. Текущий режим определяется свойством State. Для перевода набора в требуемый режим используются специальные методы, которые могут вызываться как явно, так и неявно.

Режимы:

- dsInactive – неактивность,
- dsBrowse – просмотр данных,
- dsEdit – редактирование текущей записи,

- dsInsert – вставка новой записи,
- dsSetKey – поиск записи по ключу,
- dsCalcFields – расчет вычисляемых полей,
- dsFilter – фильтрация записей,
- …

Набор данных Table

Компонент Table – набор данных, связанный только с одной таблицей БД. Формируется на базе навигационных способов доступа. Рекомендуется использовать для локальных БД (dBase, Paradox). Связь между компонентом и таблицей задается свойством TableName.

Пример.

Открытие набора данных Table.

```
procedure TForm1.Button1Click(  );
begin
if OpenDialog1.Execute then
  begin
    Table1.Active:= False;
    Table1.TableName:=OpenDialog1.FileName;
    Table1.Active:=True;
  end;
end;
```

Здесь компонент OpenDialog используется для ввода имени таблицы (свойство FileName). Метод Execute (выполнить) возвращает логическое значение True, если диалог был успешным и False, если имя таблицы не было определено.

Свойство TableType определяет тип таблицы: ttDefault, ttParadox, ttDBase, ttFoxPro, ttASCII, … .

Другие свойства:

- ReadOnly – возможность редактирования данных,
- Exclusive – монопольный доступ,

- IndexName – текущий индекс.

Обычно индекс определяется при разработке таблицы и в дальнейшем не изменяется. Для динамического изменения индекса можно использовать методы AddIndex и DeleteIndex.

Набор данных Query

Компонент Query - набор данных, формируемый в результате выполнения SQL-запроса и использующий реляционный способ доступа. Компонент Query может включать записи более чем одной таблицы БД. Запрос, на основании которого отбираются записи, содержится в свойстве SQL типа TStrings. Запрос состоит из операторов SQL и выполняется при открытии набора данных. Результат выполнения запроса можно визуализировать с помощью компонента DBGrid (сетка).

Обычно набор данных Query доступен только для чтения. Для получения редактируемого набора данных нужно установить RequestLive=True. Дополнительно должны выполняться следующие условия:

- данные отбираются только из одной таблицы или представления, допускающего модификацию;
- в запросе не используются DISTINCT и агрегатные функции;
- не применяется соединение таблиц;
- нет вложенных запросов;
- не используется группирование;
- сортировка только по индексированным полям.

Объекты поля

Объект поле Field типа TField представляет поле (столбец) набора данных. Абстрактный класс TField непосредственно не используется. Применяются производные от TField классы TBinaryField, TStringField, TNumericField и другие. Объекты поля невизуальные, служат для доступа к данным полей записи.

Состав полей набора данных можно задать двумя способами:

- по умолчанию (динамические поля),
- с помощью редактора полей (статические поля).

Динамические поля автоматически создаются для каждого столбца набора данных при его открытии. Статические поля создаются при конструировании приложения. Они позволяют задавать вычисляемые поля, ограничивать их состав, изменять порядок полей и т.д.

Операции с полями

Доступ к значению полей

Набор данных имеет свойства Fields (массив полей) и FieldCount (количество полей), которые позволяют обратиться к значению поля по его номеру. Индекс поля лежит в диапазоне 0 .. FieldCount. Свойство FieldNo – порядковый номер поля в наборе данных.

Пример.

```
procedure TForm1.Button1Click(  );
var n:integer;
begin
for n:=0 to Table1.FieldCount-1 do
ListBox1.Items.Add(Table1.Fields[n].AsString);
end;
```

В список строк последовательно включаются значения полей текущей записи, преобразованные в строковую форму.

Объект поле имеет имя (свойство Name). Имя динамического поля совпадает с именем физического поля таблицы (свойство FieldName). Имя статического поля образуется слиянием имени набора данных и имени физического поля. Имя статического поля можно изменить.

Для доступа к полю по имени используется метод FieldByName: Table1.FieldByName(‘Number’).

Другие способы обращения к полю: Table1[Number] или Table1Number. Для доступа к значению поля используются свойства Value и AsXXX. Свойство Value представляет фактические данные в объекте Field. Оно служит и для

чтения, и для записи в поле. Необходимые преобразования типов возлагаются на программиста.

Пример.

```
var a,b: integer;
a:=Table1.FieldByName('Number').Value;
b:=Table1Number.Value;
```

При использовании вариантов свойства AsXXX происходит автоматическое преобразование типа данных поля к типу названия свойства.

Пример.

a:=Table1.FieldByName('Number').AsString; - преобразование в строку.

Проверка типа и значения поля

Тип данных поля определяет свойство DataType, принимающее значения ftUnknown, ftString, ftInteger и т.д.

Для проверки допустимости вводимых символов в качестве значения поля можно использовать метод IsValidChar:

```
if Table1.Fields[3].IsValidChar(Edit1.Text[1]) then ….
```

Для числовых полей определены свойства MinValue и MaxValue, ограничивающие диапазоны вводимых значений.

Для контроля ввода можно также использовать свойство CustomConstraint и обработчики событий OnSetText, OnValidate, OnChange, возникающих при изменении значения поля.