Тема 17. Способ доступа к данным: навигационный и реляционный

Навигация в наборе данных – это перемещение тем или иным способом от одной записи (текущей) к другой записи. В этом случае записи обрабатываются по одной (позаписно). Такой способ доступа может применяться для наборов данных Table (преимущественно) и Query локальных баз данных. Навигационный доступ основан на использовании указателя текущей записи, управляемом командами навигации.

Навигационный способ доступа позволяет:

- сортировать записи,
- производить навигацию,
- редактировать записи,
- вставлять и удалять записи,
- фильтровать записи.

Кроме этого, можно выполнять действия над таблицами в целом: создавать, удалять, переименовывать таблицы, задавать режимы доступа к ним. Эти операции лучше выполнять на этапе проектирования с помощью Database Desktop.

Сортировка

Сортировка – упорядочение записей по возрастанию или убыванию значения некоторого поля или набора полей. Сортировка набора данных Table выполняется автоматически по текущему индексу. При смене индекса производится автоматическое переупорядочение записей. Текущий индекс задается свойством IndexName или IndexFieldNames.

Пример.

```
Table1.IndexFieldNames := ’Name1;Name2’;
```

Навигация

Для перемещения в наборе данных используются методы:

- First – к первой записи,
- Next – к следующей,

- Last – к последней,
- Prior – к предыдущей,
- MoveBy – переместиться на n записей от текущей.

Пример.

```
m:=0;
Table1.First;
for n:= 1 to Table1.RecordCount do
begin
m:=m+ Table1.FieldByName(‘Number).AsInteger;
Table1.Next;
End;
```

Свойство RecNo - номер текущей записи, BOF/EOF типа Boolean - начало/конец набора данных.

В случае длительного сканирования наборов данных, для предотвращения монополизации приложением процессорного времени рекомендуется использовать метод Application.ProcessMessages.

Фильтрация

Фильтрация – задание ограничений для отбираемых в набор данных записей. Допускается фильтрация по выражению и по диапазону.

Фильтрация по выражению применяется к любым полям, в том числе и неиндексированным. Фильтр задается свойством Filter типа TString.

Примеры фильтров.

Оклад<100, Должность=’должн1’ OR Должность=’должн2’.

Дополнительно к фильтру можно использовать обработчик события OnFilterRecord. Для активизации фильтра используется свойство Filtered типа Boolean. Действие фильтра можно уточнить с помощью свойства FilterOptions.

Фильтрация по диапазону

В набор данных включаются записи, значения полей которых попадают в заданный диапазон. Применяется к набору данных Table и только для

индексированных полей, ведется индексно-последовательным методом. Соответствующий индекс должен быть текущим.

Для включения и выключения фильтрации используются методы ApplyRange и CancelRange. Методы SetRangeStart и SetRangeEnd устанавливают нижнюю и верхнюю границу диапазона. Границы можно изменить методами EditRangeStart и EditRangeEnd.

Пример.

```
with Table1 do
begin
IndexName:=’indname’;
SetRangeStart;                                  // начало диапазонов
FieldByName(‘fname1’).AsString:=’sstring1’;     // начало для fname1
FieldByName(‘fname2’).AsString:=Edit1.Text;     // начало для fname2
SetRangeEnd;                                    // конец диапазона
FieldByName(‘fname1’).AsString:=’estring1’;     // конец для fname1
ApplyRange;                                     // включение фильтрации
end;
```

Метод SetRange объединяет возможности методов SetRangeStart, SetRangeEnd и ApplyRange.

Поиск записей

Для поиска записей по заданным значениям полей используются методы Locate и Lookup. Поля могут быть неиндексированными. Метод Locate ищет запись с заданными значениями полей. Если таких записей несколько, то указатель текущей записи устанавливается на первую из них. Если запись найдена, метод возвращает true, иначе false.

Пример.

```
Table1.Locate(‘Field1;Field2;’,VarArrayOf([‘val1’,’val2’]),[]);
```

Функция Lookup возвращает значения требуемых полей найденной записи.

Поиск по индексным полям

Все поисковые поля содержатся в текущем индексе. Используются методы: FindKey, SetKey, EditKey, GotoKey.

Метод FindKey ищет в наборе данных Table первую запись с указанными значениями полей. Найденная запись становится текущей.

```
if not Table1.FindKey([Edit1.Text,Edit2.Text]) then
ShowMessage(…);
```

Методы SetKey, EditKey и GotoKey используются совместно. Они заменяют метод FindKey.

Модификация набора данных

Модификация – редактирование, удаление и добавление записей. Для управления возможностью модификации используется свойство ReadOnly для набора данных Table и свойство RequestLive для набора данных Query. Для проверки возможности изменения набора данных используется свойство CanModify.

Редактирование

Редактировать можно только текущую запись. Необходимы следующие действия:

- Перевести набор данных в режим редактирования,
- Изменить значения полей записи,
- Подтвердить сделанные изменения или отказаться от них.

Пример.

```
if Table1.CanModify then
begin
Table1.Edit;
Table1.FieldByName('name').AsString:=Edit1.Text;
Table1.Post;
end;
```

Добавление записей

Последовательность действий:

- Перевести набор данных в режим вставки,

- Задать значения полей новой записи,
- Подтвердить сделанные изменения или отказаться от них (Cancel).

```
Table1.Insert;
Table1.FieldByName('name').AsString:=Edit1.Text;
…
Table1.Post;
```

Метод SetFields позволяет устанавливать значения сразу нескольких полей.

Метод InsertRecord объединяет методы Insert и SetFields.

Удаление записей

Метод Delete удаляет текущую запись набора данных.

```
Table1.Delete;
```

Реляционный способ доступа к данным

Реляционный доступ основан на операциях с группами записей и языке SQL. Применяется к набору данных Query, рекомендуется для удаленных баз данных. Текст запроса находится в свойстве SQL набора данных. Запрос можно конструировать как на этапе проектирования приложения, так и на этапе выполнения. Запрос выполняется при открытии набора данных.

Пример.

```
Query1.Close;
Query1.SQL.Clear;
Query1.SQL.Add('SELECT * FROM R.db WHERE A>B');
Query1.Open;
```