At present it is very hard to find a company which does not use IT progress for administration and automation of production process. Different systems are able to simplify and automate a significant part of company's functions. One of the most important tasks of such systems is data systematization and creating reports on the basis of gathered information ("Reports generators").

Reports and different documents are integral part of any company, and only forms of these documents differ. But it is not always when data is "flat"; some tasks require hierarchy structure or tree structures. The majority of report generators work exactly with the flat data set or tables.

Below you will find several versions of how to solve the problem of hierarchy printing by means of "Fast Report" reports generator.

As a very simple example we can consider the example of hierarchy of company workers, that is, there are both directors and employees (in big companies such a hierarchy can be deep). Data can be presented as a flat table one of the fields of which refers to a higher employee (director).

An example of such a table can be seen in the figure (EmpNo – the unique number of an employee, EmpOwner shows who the employee is subordinate to).

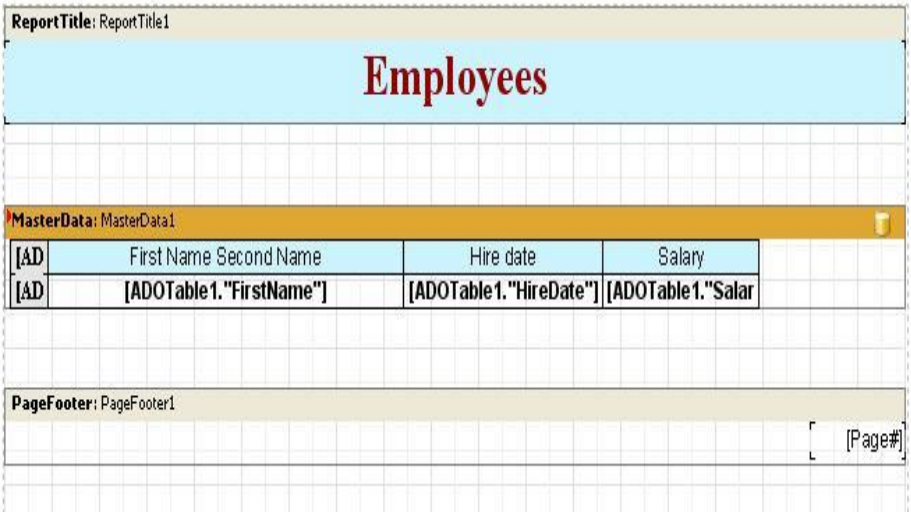
		EmpNo	LastName	FirstName	PhoneExt	HireDate	Salary	EmpOwner
	+	2	Nelson	Roberto	250	28.12.1988	40000	8
	+	4	Young	Bruce	233	28.12.1988	55500	
	+	5	Lambert	Kim	22	06.02.1989	25000	4
	+	8	Johnson	Leslie	410	05.04.1989	25050	4
	+	9	Forest	Phil	229	17.04.1989	25050	8
	+	11	Weston	K. J.	34	17.01.1990	33292,9375	8
	+	12	Lee	Terri	256	01.05.1990	45332	5
	+	14	Hall	Stewart	227	04.06.1990	34482,625	
	+	15	Young	Katherine	231	14.06.1990	24400	
	+	20	Papadopoulos	Chris	887	01.01.1990	25050	5
	+	24	Fisher	Pete	888	12.09.1990	23040	8
	+	28	Bennet	Ann	5	01.02.1991	34482,8	8
	+	29	De Souza	Roger	288	18.02.1991	25500	5
	+	34	Baldwin	Janet	2	21.03.1991	23300	5
▶	+	36	Reeves	Roger	6	25.04.1991	33620	
	+	37	Stansbury	Willie	7	25.04.1991	39224	

The report creating idea is very simple. It is necessary to sort out data in accordance with the employee's as well as director's number. It means that at first there will be employees without owners (blank field), then employees should follow each owner and like this one by one for each record. For data set sorting you can use the list where the names of the employees will be included and sorted accordingly. For transferring to the needed data set record let us use the Locate function.

Let us move from the description to report creation. Create a new template and set a connection to the data base, add a table or query for choosing the needed data. The example of such a table is attached to the article together with the example (HDemo.mdb). Add bands to the report such as "Report title", "First level data" and "Page footer". Place the "text" object on the report title with a free text (e.g. "Employees"), and the object "text" on the page footer with the [Page] variable. In our report the data bands are decorative for making a report look good. The

main processing will be connected with data bands. The report template should be just like in the figure below.

Add necessary fields from data set (drag from data tree) to data band but do not attach the band to the data. As a result a ready template will look as follows:



We will use a report script for processing, data select and shift. To do this, add the following events from listed objects (the events can be added through objects inspector):

- "First level data" (MasterData1) –OnAfterPrint and OnBeforePrint events.
- "Report"(Report can be chosen in the list of objects inspector), OnStopReport Event**.**

And so a template preparation is over and we can start sorting and derivation through the report script. For that, move to the "code" tab and start writing a script. On the very top of the script let us declare global variables:

```
var
    TreeList: TStringList;// sorting list
    ShiftList: TStringList;// sorting list
    FDBdataSet: TfrxDBDataSet;// data set
```

```
  FVal: variant;// temporary variable for values storage
  FPrevEmpNo: Integer;// a variable for the previous number storage
  oldCurX: Extended; // a variable for the original position of band output storage
```

Now let us move on to the main script procedure (a code between begin…end), variables will be initialized in it and completion of sorting list will be called.

```
begin
  {getting data set }
  FDBdataSet := TfrxDBDataSet(Report.GetDataset('ADOTable1'));
  { creating sorting list }
  TreeList := TStringList.Create;
  { creating shift  list }
  ShiftList := TStringList.Create;
  if FDBdataSet = nil then Exit;

  { opening data set and move on to the first record}
  FDBdataSet.Open;
  FDBdataSet.First;
  { cycle through all data set records }
  while not FDBdataSet.Eof do
  begin
    { getting current employee's number  }
    FPrevEmpNo := FDBdataSet.Value('EmpNo');
    AddEmp;
    {restore data set cursor after  AddEmp run  }
    FDBdataSet.Locate('EmpNo', FPrevEmpNo, 0);
    { moving on to the next record }
    FDBdataSet.Next;
  end;
  MasterData1.RowCount := TreeList.Count;
end.
```

The Report.GetDataset function returns data set object according to its name (If it is found). Creating sorting list is next, as well as shifting and data set opening. Then a cycle of all data set records is organized in which we can call a value add function into the AddEmp list (described below). FDBdataSet.Locate function locates a cursor in data set in a position where the specified field's value matches with the value sent as the second function parameter. MasterData1.RowCount sets a number of data band repetitions (rows number).

The sort function must be declared before the main procedure, its realization is written below:

```
{adding the employee number in the corresponding  hierarchical position }
function AddEmp: Integer;
 var
  FEmpNo: Integer;
  RNo: Integer;
  s: String;
 begin
  FVal := FDBdataSet.Value('EmpOwner');// director's number
  FEmpNo := FDBdataSet.Value('EmpNo');// current employee's number
  s := IntToStr(FEmpNo);
  Result := TreeList.IndexOf(s);// searching employee in the list,if found, then we won't add
  if Result <> -1 then exit;

  if FVal <> NULL then
  begin
   {employee has director }
   { move to the director }
   FDBdataSet.DataSet.Locate('EmpNo', FVal, 0);
   { add  employee after director }
   Result := AddEmp;
   { add employee after director }
   if Result + 1 >= TreeList.Count then
    TreeList.Add(s)
   else
    TreeList.Insert(Result + 1, s);
  end
  else
  begin
   {employee hasn't got director (how lucky he is !), simply add him to the list }
   TreeList.Add(s);
   Result := TreeList.Count;
  end;
 end;
```

The function adds employees following their directors, if the director is not in the list yet, a recursive call is made until the full chain fill. If the employee has been added already then, in the recursive call, exit from the function is done and moves to the second record (in the main procedure).

In the OnStopReport event, we the deleted the created list.

```
procedure ReportOnStopReport(Sender: TfrxComponent);
begin
  { deleteing the list }
  TreeList.Free;
  ShiftList.Free;
end;
```

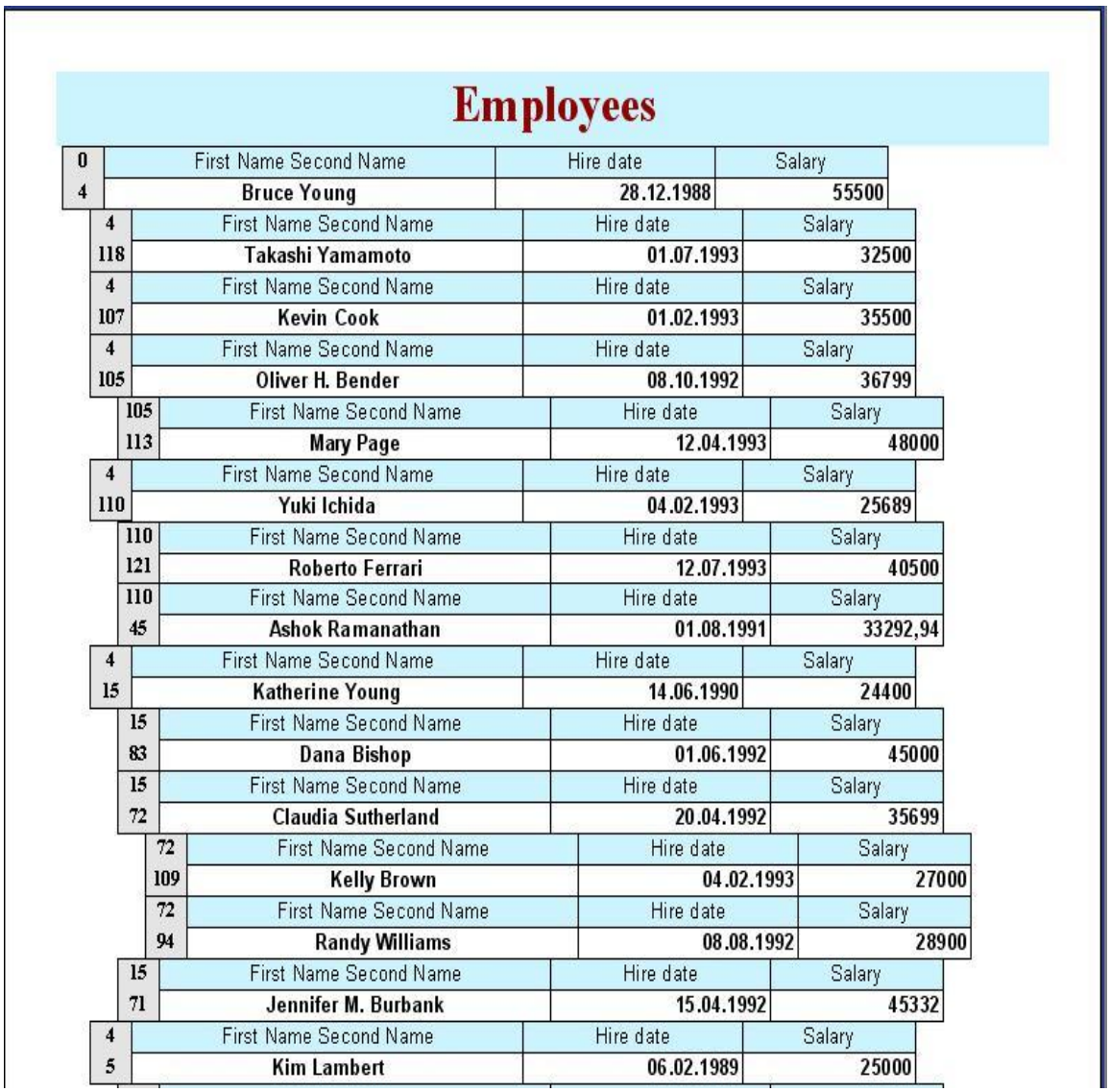
Selecting data before derivation corresponding to band sorting and shifting has been realized in the MasterData1OnBeforePrint event. Moving to a needed row can be done by using the function called Locate, the value from the list is used as a key.

```
procedure MasterData1OnBeforePrint(Sender: TfrxComponent);
var
  eOwner: String;
  i: Integer;
begin
  { pause the addition to the page so that CurX is not made zero }
  if MasterData1.Height > Engine.FreeSpace then Engine.NewPage;
  FPrevEmpNo := <ADOTable1."EmpOwner">;
 {restore the position in the data set }
  FDBdataSet.DataSet.Locate('EmpNo', TreeList[<Line> - 1], 0);
  { get the current director }
  eOwner := IntToStr(<ADOTable1."EmpOwner">);
  { if the director is not found we add him }
  { Else we get the shifting index for the employee }
  i := ShiftList.IndexOf(eOwner);
  if i = -1 then
  begin
    { for shifting the nodes correctly , we test the previous index }
    i := ShiftList.IndexOf(IntToStr(FPrevEmpNo)) + 1;
    {if the previous key is the last in the list, then we add a new shift}
}
    if i >= ShiftList.Count then
    begin
      ShiftList.Add(eOwner);
      i := ShiftList.Count - 1;
    end
    { else we change the search key }
    else
       ShiftList[i] := eOwner;
  end;
  {Remember the current band position }
  oldCurX := EnginE.CurX;
  { смещаем бэнд }
  EnginE.CurX := EnginE.CurX + 20 * i;
end;
```

In the MasterData1OnAfterPrint event, we restore the position after shifting.

```
procedure MasterData1OnAfterPrint(Sender: TfrxComponent);
begin
  { restoring the position of the band }
  EnginE.CurX := oldCurX;
end;
```

Report is ready:



Employees

	First Name Second Name	Hire date	Salary
0			
4	**Bruce Young**	**28.12.1988**	**55500**
4			
118	**Takashi Yamamoto**	**01.07.1993**	**32500**
4			
107	**Kevin Cook**	**01.02.1993**	**35500**
4			
105	**Oliver H. Bender**	**08.10.1992**	**36799**
105			
113	**Mary Page**	**12.04.1993**	**48000**
4			
110	**Yuki Ichida**	**04.02.1993**	**25689**
110			
121	**Roberto Ferrari**	**12.07.1993**	**40500**
110			
45	**Ashok Ramanathan**	**01.08.1991**	**33292,94**
4			
15	**Katherine Young**	**14.06.1990**	**24400**
15			
83	**Dana Bishop**	**01.06.1992**	**45000**
15			
72	**Claudia Sutherland**	**20.04.1992**	**35699**
72			
109	**Kelly Brown**	**04.02.1993**	**27000**
72			
94	**Randy Williams**	**08.08.1992**	**28900**
15			
71	**Jennifer M. Burbank**	**15.04.1992**	**45332**
4			
5	**Kim Lambert**	**06.02.1989**	**25000**

To conclude, need to point out that, using Locate on a big data set can reduce the productivity of report creation, in this case, all the needed data can be saved in a either in a list, or in a tree structure. Without using Locate when selecting data. The general creation principal remains the same.

In the presence of the tree structure, derivation is very simpler, since there is no need to sort the data, and it's possible to immediately move on the tree in the OnManualBuild event derivates nodes via the Engine.ShowBand band, but this is a topic for a different article.

The report examples and data applications can be run in both «Fast Report 4 VCL», and in «Fast Report Studio».