Under Construction: Surfin' Your Surfers With TDecisionCube

by Bob Swart

Last month we played with the `TClientDataSet` component, used it to connect to MIDAS and CORBA servers, and presented raw statistical web visitor data in a simple data-aware grid. This time, we'll take the analysis of the web visitor tracking data to the next level with the help of the Decision Cube Technology from Delphi's Client/Server Edition.

Problem Description

A website visitor tracking application logs each visit to a web page in a website. During each visit, not only the web page URL, but also the date, time, unique user IP address, the User Agent (web browser) and the referrer web page URL is logged. This information can be used for several purposes, for example to get a statistical overview of the web browsers and operating systems used to view my website, or to determine the busy hours at my website compared to quieter hours.

DecisionCube Theory

Delphi 3 Client/Server introduced the Decision Cube component collection, a set of components for crosstab analysis and presentation of data in a grid or chart. Crosstab means that we summarize data (such as the number of page requests or unique visitors on a daily basis), and present the data along a number of dimensions (such as 0..24 hour, web browser version and operating system).

In order to use the logfiles generated by last month's tracking application, we need to rewrite the conversion routine (from .TRK file to .DB table) and use the resulting table as the input for a Decision Query to feed the Decision Cube and other decision support components.

At the end of this article, we'll be able to draw some interesting conclusions, so let's get started...

Logfile Reconversion

We start with the tracking logfile 19981027.TRK generated on Tuesday 27th October (the day I was present at the Inprise Conference in London, good for 2,027,334 bytes or 4,759 page requests, 41% referred). Converting this logfile to a table that can be used by a Decision Cube means we have to consider the dimensions and summaries to be used by the Decision Cube. Some of the dimensions that we want to see are browser type (condensed), operating system, hours and maybe domain (derived from IP addresses). The summaries would include total page requests and unique visitors (or IP addresses).

This means a slightly modified `TrkTable` program from last time, with fields for `DateTime`, `Hours`, `IP`, `Browser`, `OSystem`, `ThisPage` and `Referrer` (see Listing 1).

Note that Listing 1 actually contains a number of hardcoded rules to convert the data from the tracking logfile to more condensed information (like Browser and OSystem) for the table.

DecisionCube Practice

With this new table, we can create a query and use the `Hours`, `Browser` and `OSystem` fields as dimensions, and the 'count' of `ThisPage` (the page requests) or unique IP numbers (unique visitors) as summary information.

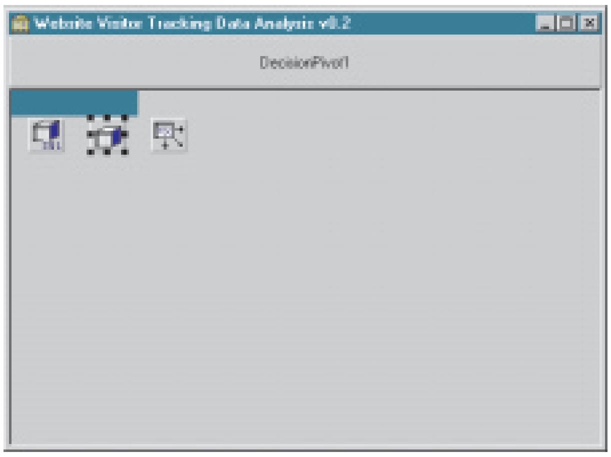
To create a Decision Cube project for the Website Visitor Tracking Data Analysis application, we can either use an ActiveForm and deploy it on the internet/intranet again, or a simple 'regular' new application. In this article, I have used the latter approach, but an ActiveForm demo version (with fixed data for 27th October) will be present on my website at www.drbob42.com/ActiveX/decision.htm by the time you read this.

For now, we'll start a regular new application, drop a `TDecisionPivot` component on it (set `Align` to `alTop`), and drop a `TDecisionQuery`, `TDecisionCube` and `TDecisionSource` right below the `DecisionPivot1`.

Assign the `DataSet` property of `DecisionCube1` to the `DecisionQuery1` component (but in general we can use any dataset here). Next, assign the `DecisionCube` property of `DecisionSource1` to `DecisionCube1` (just like a datasource to a dataset), and assign the `DecisionSource` property of `DecisionPivot1` to `DecisionSource1` (again, similar to data-aware components connecting to a datasource).

Of the components on the form right now (see Figure 1), only the `DecisionPivot` is visual, the others are non-visual decision 'engine' components. The `DecisionPivot` will show buttons for each of the dimensions we're using in the `DecisionGrid`, and a drop-down

➤ *Figure 1*



selection list to pick one of the summary fields. The area under the DecisionPivot can be used to show the actual data from the DecisionCube in either a DecisionGrid (a sophisticated DBGrid) or a DecisionGraph (derived from TeeChart, so it has many charting possibilities).

We don't drop any DecisionGrids or DecisionGraphs just yet, but first concentrate on the input dataset to analyse here.

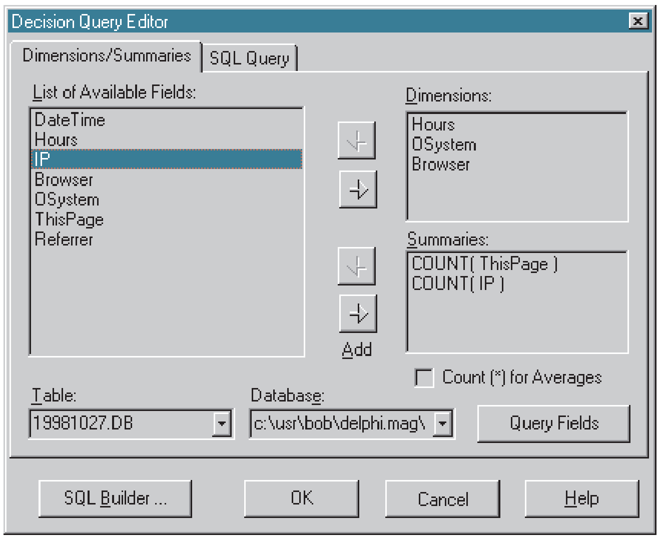
Decision Query

If we double click on the DecisionQuery1 component, the Decision Query Editor is started in which we can generate the SQL query to feed the DecisionCube. Note (from Figure 2) that we start with seven fields from the 19981027.DB table. Three of these fields (Hours, OSystem and Browser) can be used as dimensions, while ThisPage and IP can be used as summary information for the individual cells. When adding summary fields, Delphi offers an aggregate function (SUM, COUNT, etc) and we need the COUNT function to emunerate the number of page requests and visitors (IP addresses).

The Decision Query Editor (see Figure 2) offers a quick and easy way to turn fields into dimensions or summaries. However, we can always decide to use the SQL Builder Wizard to prebuild our SQL query. Additionally, we can switch to the SQL Query tab of the Decision Query Editor and see the source of the generated SQL Query itself. In this page, we can even modify the content of the SQL Query, for example to change COUNT(IP) into COUNT(DISTINCT IP) to make sure we only count unique visitors (that is, unique IP addresses). This option was not offered by the arrow-buttons in the first page, so we actually *had* to make this change in the SQL Query here (see Figure 3).

➤ *Listing 1: TrkTable.*

```
program TrkTable;
{$APPTYPE CONSOLE}
uses SysUtils, DB, DBTables;
var
  f: Text;
  i: Integer;
  Str,Agent: String;
  hits: Integer = 0;
  refer: Integer = 0;
function CopyStripDelete(var Str: String; From,Len:
  Integer): String;
begin
  Result := Copy(Str,From,Len); { copy }
  Delete(Str,1,Len); { delete }
  Len := Length(Result);
  while Result[Len] = #32 do
    Dec(Len);
  SetLength(Result,Len) { strip }
end {CopyStripDelete};
begin
  if ParamCount = 0 then begin
    writeln('Usage: TrkTable [datfile]');
    Halt
  end;
  with TTable.Create(nil) do
  try
    Active := False;
    TableType := ttParadox;
    TableName := ParamStr(1)+'.DB';
    with FieldDefs do begin
      Clear;
      Add('DateTime', ftString, 24, FALSE);
      Add('Hours', ftInteger, 0, FALSE);
      Add('IP', ftString, 16, FALSE);
      Add('Browser', ftString, 32, FALSE);
      Add('OSystem', ftString, 32, FALSE);
      Add('ThisPage', ftString, 128, FALSE);
      Add('Referrer', ftString, 128, FALSE);
    end;
    CreateTable;
    Open;
    System.Assign(f,ParamStr(1)+'.trk');
    System.Reset(f);
    while not System.Eof(f) do begin
      readln(f,Str);
      Append;
      Agent := CopyStripDelete(Str,1,24);
      FieldByName('DateTime').AsString := Agent;
      System.Delete(Agent,1,11);
      if Pos('AM',Agent) > 0 then begin
        System.Delete(Agent,Pos(':',Agent),255);
        if Agent = '12' then
          FieldByName('Hours').AsInteger := 0
        else
          FieldByName('Hours').AsInteger := StrToInt(Agent)
      end else begin
        { PM }
        System.Delete(Agent,Pos(':',Agent),255);
        FieldByName('Hours').AsInteger :=
          StrToInt(Agent) + 12
      end;
      FieldByName('IP').AsString :=
        CopyStripDelete(Str,1,16);
      Agent := CopyStripDelete(Str,1,128);
      if (Pos('Windows NT', Agent) > 0) or
        (Pos('WinNT', Agent) > 0) then
        FieldByName('OSystem').AsString := 'WinNT'
      else if (Pos('Windows 95', Agent) > 0) or
        (Pos('Win95', Agent) > 0) then
        FieldByName('OSystem').AsString := 'Win95'
      else if (Pos('Windows 98', Agent) > 0) or
        (Pos('Win98', Agent) > 0) then
        FieldByName('OSystem').AsString := 'Win98'
      else If Pos('Win16', Agent) > 0 then
        FieldByName('OSystem').AsString := 'Win16'
      else if Pos('Linux', Agent) > 0 then
        FieldByName('OSystem').AsString := 'Linux'
      else if Pos('Teleport', Agent) > 0 then
        FieldByName('OSystem').AsString := 'Teleport'
      else
        FieldByName('OSystem').AsString := 'other';
      if Pos('(compatible; ',Agent) > 0 then begin
        System.Delete(Agent,1,pos('(compatible;',Agent)+12);
        if Pos('MSIE',Agent) > 0 then
          System.Delete(Agent,1,Pos('MSIE',Agent)-1);
        if Pos(';',Agent) > 0 then
          System.Delete(Agent,Pos(';',Agent),255)
        else if Pos(')',Agent) > 0 then
          System.Delete(Agent,Pos(')',Agent),255)
      end else if Pos('MSIE',Agent) > 0 then
        System.Delete(Agent,1,Pos('MSIE',Agent)-1)
      else if Pos(' ',Agent) > 0 then
        System.Delete(Agent,Pos(' ',Agent),255);
      if Pos('Mozilla/',Agent) = 1 then begin
        System.Delete(Agent,1,8);
        Agent := 'Netscape ' + Agent
      end else if (Length(Agent) < 2) or
        (Agent[1] = '(') then
        Agent := 'other';
      i := Pos(' ',Agent);
      if i > 0 then begin
        repeat
          Inc(i)
        until not (Agent[i] in ['0'..'9','.']);
        System.Delete(Agent,i,255);
       {$IFDEF X}
        // change all .01 or .02 postfixes to .x
        i := Pos('.0',Agent);
        if (i > 0) and (Length(Agent) > i+1) then begin
          System.Delete(Agent,i+3,255);
          Agent[i+2] := 'x' { 4.0x }
        end
       {$ENDIF}
      end;
      if (FieldByName('OSystem').AsString = 'other') and
        (Pos('MSIE',Agent) > 0) then
        FieldByName('OSystem').AsString := 'Win16';
      FieldByName('Browser').AsString := Agent;
      FieldByName('ThisPage').AsString :=
        CopyStripDelete(Str,1,128);
      FieldByName('Referrer').AsString :=
        CopyStripDelete(Str,1,128);
      if FieldByName('Referrer').AsString <> '@' then
        Inc(refer); // actual referrer info
      Post;
      Inc(hits)
    end;
    writeln(hits,' page requests (', (refer*100) div hits,
      '% referred) ', 'in logfile ',ParamStr(1))
  finally
    System.Close(f);
    Close;
    Free
  end
end.
```



List of Available Fields:

DateTime
Hours
IP
Browser
OSystem
ThisPage
Referrer

Dimensions:

Hours
OSystem
Browser

Summaries:

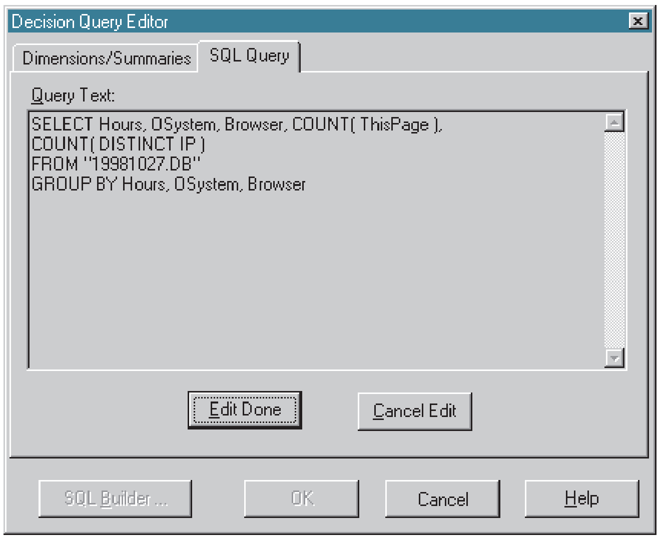
COUNT(ThisPage)
COUNT(IP)

☐ Count (*) for Averages

Table: 19981027.DB

Database: c:\usr\bob\delphi.mag\

➤ *Figure 2*



Query Text:

```
SELECT Hours, OSystem, Browser, COUNT( ThisPage ),
COUNT( DISTINCT IP )
FROM "19981027.DB"
GROUP BY Hours, OSystem, Browser
```

➤ *Figure 3*

We see that dimensions are implemented by the SQL GROUP statement, while the summaries are all other (aggregated) fields, holding the value of COUNT(ThisPage) and COUNT(DISTINCT IP). If we decide to use a regular TQuery component instead of a TDecisionQuery component, then we need to ensure that we also include a GROUP BY clause in the SQL Query, and that the fields in this GROUP BY match the order of the fields in the SELECT statement (for non-aggregated fields).

After we click on the Edit Done button, the SQL Query is parsed and checked against the SQL ANSI-92 standard.

Decision Cube

Now that we have defined the DecisionQuery, it's time to see the effect in the Decision Cube, the central processing component that is fed by the DecisionQuery's data.

If we double click on the Decision Cube component, we enter the Decision Cube Editor where we can customise the Dimension and Summary fields. For example, the OSystem field gets a display name of OS, while the COUNT OF ThisPage field is named Webpage Requests and COUNT OF IP is named Unique Visitors. Note that here, again, DISTINCT is not mentioned.

The second tab of the Decision Cube Editor contains Memory Control options. These are important, and we must realise that the Decision Cube is maintaining the information in memory. So, each dimension potentially results in a combinatoric explosion, and the three dimensions we use (Hour, OS and Browser) already result in about 9,000 records. Note that this is about twice as much as the number of lines in the original logfile, and things could have been worse (in other words bigger) if we used the real time, included domain information or more detailed information about the OS and Browser types.

Decision Grid

Now it's time to add another visual part to our form. Just drop a TDecisionGrid under the DecisionPivot1, set Align to Client, and connect the DecisionSource property to the DecisionSource1 component. If we set the Active property of the DecisionQuery1 component to True, we can see the live analysis in the Decision Grid at design-time.

Note that every column of the Decision Grid has the same width, while I would prefer to use a greater width for the first few columns (which have the dimension values), and a smaller width for the remaining columns. This unfortunately cannot be done, since the TDecisionGrid only publishes the DefaultColWidth property (which works for all columns) and not the ColWidths index property (which can be used to set the individual cell widths).

TDecisionGrid is derived from TCustomDecisionGrid, which is derived from TCustomGrid where we can find the ColWidths as protected property. So, all we need to do is unprotect this property, or make it public. The easiest way to do that is by deriving a new component THackDecisionGrid that only re-defines the ColWidths property as a public property:

```
THackDecisionGrid =
  class(TDecisionGrid)
public
  property ColWidths;
end;
```

We can now either install the THackDecisionGrid and use it instead of the original TDecision Grid, or we can just typecast the TDecisionGrid component to a THackDecisionGrid, since they both have the same layout (the only difference is the fact that ColWidths is declared public in the THackDecisionCube). Casting the DecisionCube1 component to a THackDecisionCube means we can now assign 56 to the ColWidhts[0] (the operating system) and 110 to ColWidths[1] (the browser type). See Listing 2 for details.

Decision Action!

We are now ready to compile and run the Website Visitor Tracking Data Analysis application.

The first thing we can analyse is the distribution of operating systems among the unique visitors for each hour on that particular 27th October. It turns out that Win95 is still the most used OS by my visitors (almost 50%), followed by WinNT. Win98 is a good third, and we can effectively ignore the others (less than 3% combined). It is strange to see Linux in the list, although we must remember that a web browser machine is not necessarily the development machine. The Teleport entry is in fact a WebRobot 'crawling' (downloading) my entire site, which happened twice that day (and a few times more each month, actually).

Apart from Operating Systems, most web masters are of course interested in the distribution of web browsers and their versions. This can be seen if we click on the OS button of the DecisionPivot (to turn this dimension off) and click on the Browser button instead (to turn that dimension on). The effects can be seen in Figure 6.

As we saw last month already, the version 4.x browsers of both Netscape and Internet Explorer are used for well over 75% of the total page requests of my website that day (over 90% if you discard the 750 'page requests' made by the Teleport WebRobot). This is again good news, since it means we can use JDK 1.1 applets on the JBuilder section, and other HTML advanced features in the entire website.

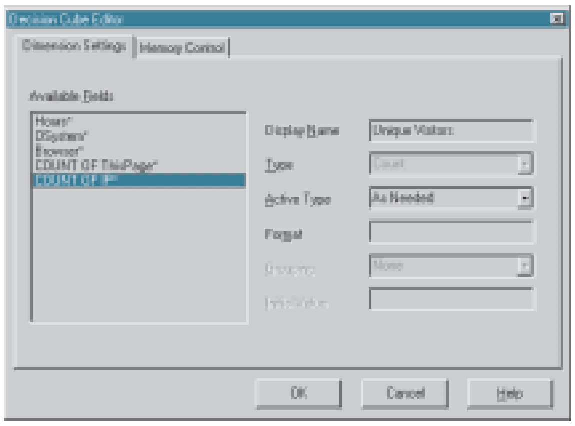
Internet Explorer is used for just a little less than half of the total page requests. This means it may not be a good idea to use too many ActiveX controls or ActiveForms, since more than half of the pages are not serviced by a browser which can, by default, handle them. Especially since the ActiveX plugin for Netscape Communicator 4.x is not free, we need to keep an eye on this figure before starting to utilise ActiveX solutions more, even thin ActiveForms like those you can find at www.drbob42.com/ActiveX/.

Note that there are many Netscape version 4.0x browsers, and it would be a good thing if we could summarize them all under one 'Netscape 4.x' entry instead of identifying all these sub-versions. We have to go back to the TrkTable program again, and modify the conversion process. This is done by the {$IFDEF X} compiler directive as can be seen back in Listing 1.

Advanced Decisions

Apart from showing either Operating Systems or Browsers (or both) against Hour, we can also show Browsers against Operating Systems, which would not tell much about my website, but more

➤ Figure 4



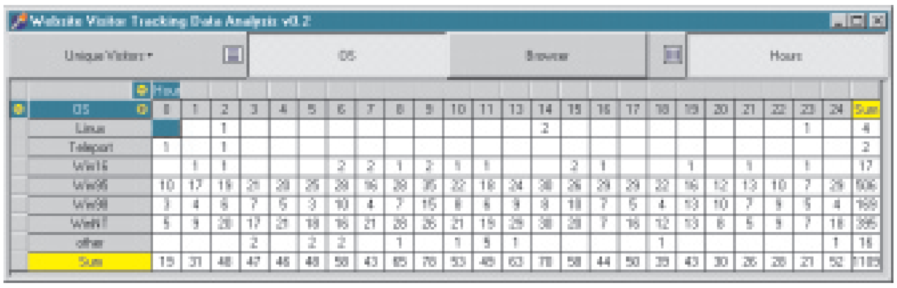
Decision Cube Editor

Dimension Settings | Memory Control

Available Fields:
- Hours*
- OSystem*
- Browser*
- COUNT OF ThisPage*
- COUNT OF IP*

Display Name	Unique Visitors
Type	Count
Active Type	As Needed
Format	
Grouping	None
Initial Value	

➤ Figure 5



Website Visitor Tracking Data Analysis v0.2

OS	0	1	2	3	4	5	6	7	8	9	10	11	13	14	15	16	17	18	19	20	21	22	23	24	Sum
Linux			1											2									1		4
Teleport	1		1																						2
Win16		1	1				2	2	1	2	1	1			2	1			1		1		1		17
Win95	10	17	18	21	28	25	28	16	38	35	22	18	24	38	26	29	29	22	16	12	13	10	7	28	506
Win98	3	4	6	7	5	3	10	4	7	15	8	6	9	8	18	7	5	4	13	10	7	9	5	4	169
WinNT	5	9	20	17	21	18	16	21	28	26	21	15	29	38	28	7	16	12	13	6	5	9	7	18	395
other				2		2	2		1		1	5	1					1						1	16
Sum	15	31	48	47	46	48	58	43	85	78	53	48	63	78	58	44	50	39	43	30	26	28	21	52	1185

➤ Listing 2

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, Db, DBTables, mxtables, Grids, mxgrid,
  mxDB, mxstore, ExtCtrls, mxpivsrc, TeeProcs, TeEngine,
  Chart, mxgraph, Series;
type
  TForm1 = class(TForm)
    DecisionPivot1: TDecisionPivot;
    DecisionCube1: TDecisionCube;
    DecisionSource1: TDecisionSource;
    DecisionGrid1: TDecisionGrid;
    DecisionQuery1: TDecisionQuery;
    DecisionGraph1: TDecisionGraph;
    Series2: TPieSeries;
    Series3: TPieSeries;
    Series1: TLineSeries;
    Series4: TLineSeries;
    Series6: TLineSeries;
    Series7: TLineSeries;
    Series8: TLineSeries;
    Series9: TLineSeries;
    Series10: TLineSeries;
    Series5: TBarSeries;
    procedure DecisionGrid1DecisionDrawCell(Sender:
      TObject; Col,Row: Integer; var Value: String;
      var aFont: TFont; var aColor: TColor;
      AState: TGridDrawState; aDrawState:
      TDecisionDrawState);
    procedure DecisionPivot1Click(Sender: TObject);
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
type
  THackDecisionGrid = class(TDecisionGrid)
  public
    property ColWidths;
  end;
procedure TForm1.DecisionGrid1DecisionDrawCell(
  Sender: TObject; Col, Row: Integer; var Value: String;
  var aFont: TFont; var aColor: TColor;
  AState: TGridDrawState; aDrawState: TDecisionDrawState);
begin
  if (Col <= 0) and (Row <= 0) then begin
    if DecisionGrid1.FixedCols > 1 then
      if DecisionGrid1.Cells[-2,-1] = 'OS' then
        THackDecisionGrid(DecisionGrid1).ColWidths[1] := 56
      else
        THackDecisionGrid(DecisionGrid1).ColWidths[1] :=
          110;
    if DecisionGrid1.FixedCols > 2 then
      THackDecisionGrid(DecisionGrid1).ColWidths[2] := 110
  end
end;
procedure TForm1.DecisionPivot1Click(Sender: TObject);
begin
  DecisionGraph1.Visible := not DecisionGraph1.Visible;
end;
end.
```

about the types of browsers and their relative availability. The `DecisionPivot` enables us to 'open and close' dimensions by clicking on the buttons, but we can also drag dimensions from one side to another. Just right click on the button and move it to the other axis. Doing so, we can set `OS` against `Browsers`, as can be seen in Figure 7.

The browser 'Thai' turned out to be the Thai version of Internet Explorer, so I changed the `TrkTable` parser to eliminate Thai and add the numbers to the correct versions of Internet Explorer instead.

Apart from showing all the browser types against all the operating systems, it's also possible to 'instantiate' one or more of the dimensions to a fixed value. For example, if we'd like to see the browser distribution on Windows 98 during the day, we can right click on the `OS` button and select the `Win98` value (Figure 8).

Despite Mr Gates' assurances that Windows 98 is *open* and people can run any browser on this operating system, it would appear that in practice over 90% of the people are using Internet Explorer 4.0 or higher anyway.

Visual Decisions

A final exercise involves using a `DecisionChart` instead of a `DecisionGrid`, thereby presenting the data in a more visual manner. `DecisionCharts` are derived from `TeeChart` (in fact, they are just `TeeCharts` obtaining their information from the `DecisionCube`), and can be used as plain `TeeCharts` accordingly. Figure 9 shows the browser versions among all unique visitors for that day.

Summary

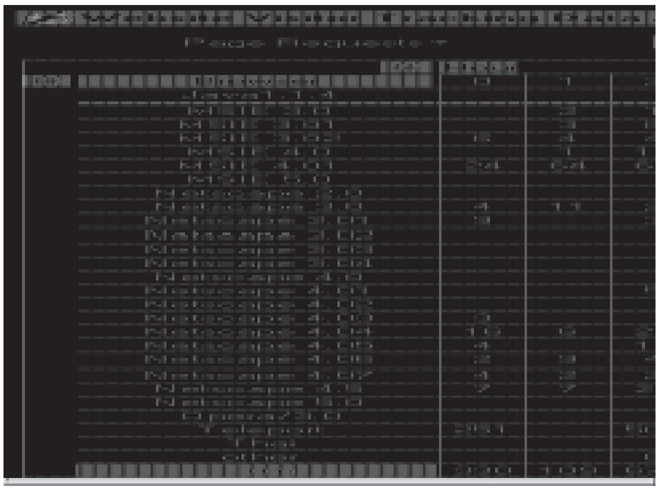
We have explored `TDecisionCubes`, learned how to connect them to input queries (`TDecisionQuery` or `TQuery`), connect via a `TDecisionPivot` to visual components like `TDecisionGrid` and `TDecisionChart`. We also learned to limit the number of dimensions to avoid memory problems, and we have seen ways to tweak the analysis views at runtime. All in all, the Decision Cube component set provides a powerful way to analyse your data.

Next Time

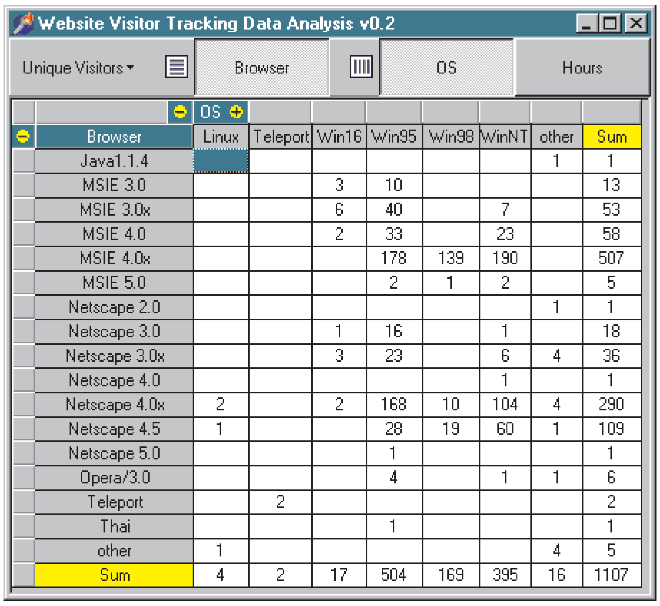
After all this multi-tier violence, it's time to have a little break by taking a closer look at implementing NNTP (Network News Transfer Protocol) techniques for reading and writing newsgroup messages. *Stay tuned!*

Bob Swart (aka Dr.Bob, visit www.drbob42.com) is a professional knowledge engineer technical consultant using Delphi, JBuilder and C++Builder for Bolesian (www.bolesian.com) and freelance technical author.

➤ Below: Figure 6

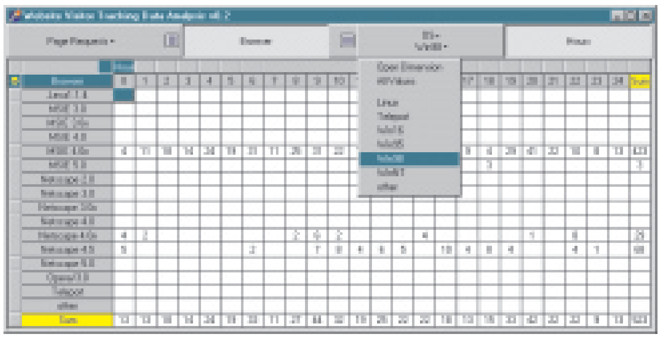


➤ Right: Figure 7



Browser	Linux	Teleport	Win16	Win95	Win98	WinNT	other	Sum
Java1.1.4							1	1
MSIE 3.0			3	10				13
MSIE 3.0x			6	40		7		53
MSIE 4.0			2	33		23		58
MSIE 4.0x				178	139	190		507
MSIE 5.0				2	1	2		5
Netscape 2.0							1	1
Netscape 3.0			1	16		1		18
Netscape 3.0x			3	23		6	4	36
Netscape 4.0						1		1
Netscape 4.0x	2		2	168	10	104	4	290
Netscape 4.5	1			28	19	60	1	109
Netscape 5.0				1				1
Opera/3.0				4		1	1	6
Teleport		2						2
Thai				1				1
other	1						4	5
Sum	4	2	17	504	169	395	16	1107

➤ Figure 8



➤ Figure 9