

Build Gantt Charts with MS Graph

Doug Den Hoed



The MS Graph engine that powers charts in Access and Excel is extremely flexible, but it often requires some tinkering. Doug Den Hoed shares his techniques for building Gantt charts, and he reveals some tricks to help you make your charts look more professional.

A picture is worth a thousand words. Fortunately, you can deliver one for considerably less than that, thanks to the MS Graph library that's behind Access and Excel. Recently, my most demanding client (me) posed a new challenge: Build a Gantt chart using MS Graph. I opened a form in design view and fearlessly clicked the Insert | Chart menu choice. The Chart Wizard popped up and asked me for a datasource. I paused as the truth sunk in. I didn't even know what my Gantt chart data should *look* like! So, for once, I abandoned Access.

In case of emergency, break rules

Instead, I opened Excel. When I clicked Insert | Chart in Excel, a much friendlier, four-step wizard appeared.

The first panel offered me a smorgasbord of charts on two tabs: Standard Types and Custom Types. The latter had an entry called "Floating Bars" that sounded promising. Its description read, "Requires two series—first specifies beginning of bars, second specifies length of bars." I reread it a few times and... Aha! I realized that this was the solution to my problem: I'd create an invisible bar in front of each real bar that would push the real bars into the right position!

Now I knew what my data had to look like. I canceled the wizard and entered the sample data shown in [Figure 1](#). I called the column for the invisible bars "Offset" to indicate how far (for example, in days) from the left edge of the chart each bar should be pushed. I knew that I wanted to show certain tasks being extended, so I also added both "OriginalBudget" and "RevisedBudget" columns. And to keep my bars straight, I created a "Task" column and numbered each task in order of execution.

I selected all of my sample data, clicked Insert | Chart again, and returned to the Chart Wizard. When I previewed the Floating Bars type, it looked a bit odd—and too 3D-ish, as well. I chose "Bar" from the Standard Types tab because it had a horizontal orientation (I wanted the bars to stretch over time). Six chart subtypes

appeared for Bar charts: Clustered, Stacked, and 100% Stacked, in either 2D or 3D. I chose "Stacked, 2D" and previewed. It wasn't great, but the orientation was what I wanted, so I clicked Next.

The second form in the wizard offered two tabs to choose my Data Range and Series. Since I'd already highlighted my data, the Data Range was filled in. When I switched from "Series in Rows" to "Series in Columns," my graph preview improved dramatically. The bars had the progression that I expected, although the bars went from longest to shortest as I looked down the chart. This made the chart seem upside down. On the Series tab, the chart had plotted the task numbers as values, which wasn't what I'd intended, either, so I removed the Task series. Everything else looked fine.

The third form in the wizard covered formatting: Titles, Axes, Gridlines, Legend, Data Labels, and Data Tables. Since I knew I could adjust my real chart in Access later, I skipped them. The fourth form simply asked where I wanted to put the chart. I placed it underneath my sample data, and the wizard closed.

It took me a while to find the last few tricks I needed. I clicked an Offset bar, which selected the whole series, then right-clicked and chose Format Series. On the Patterns tab, I set the Border property and Area to None. That made the Offset bars invisible. However, the "Offset" label was still in the legend. I considered Nulling the column name out, but I knew that wouldn't be allowed in Access: Every column in a SQL SELECT statement has to

| | A | B | C | D |
|---|------|--------|----------------|---------------|
| 1 | Task | Offset | OriginalBudget | RevisedBudget |
| 2 | 1 | 0 | 15 | 5 |
| 3 | 2 | 20 | 10 | |
| 4 | 3 | 30 | 20 | |
| 5 | 4 | 50 | 5 | |
| 6 | | | | |

Figure 1. Gantt chart sample data.

have a name. So I cheated. I renamed the Offset column header to “Start Date” and changed its border to a dashed line. This had the pleasant effect of guiding my eye to each bar, and it made Start Date look like it belonged in the legend.

To fix my upside-down issue, I selected the vertical axis by clicking the left edge of the chart, right-clicked, and then chose Format Axis. On the Scale tab, I enabled “Categories in reverse order.” At last, my graph looked like a Gantt chart (see Figure 2). I’ve included GANTT.XLS in this month’s Source Code file at www.smartaccessnewsletter.com.

Resume next

Bolstered by my success in Excel, I confidently returned to Access and created the table tblTask to hold the data for my Gantt chart. I kept it similar to my sample data in Excel, except for two changes: I added a StartDate column with dates to coincide with the Offset integers, and I added a Task column for short descriptions of each task. I excluded both of these columns from my chart’s Rowsourc query (called “qselGantt, Order By TaskID”), but I knew the data would be a helpful reference. With the table loaded with data, I created rptGantt, bound it to tblTask, and painted the Task ID, Start Date, Task, Original Budget, and Revised Budget in the detail section of the report. There was only one thing left: the chart.

Armed with my datasource, I tried the Access Chart Wizard again. In the wizard, I clicked Queries, selected qselGantt, and clicked the Next button. On the next form, I moved all four fields in qselGantt from the “Available Fields” list box on the left to the “Fields for Chart” list box on the right. I clicked Next and was pleased to see a selection of chart types, just as Excel had offered. I chose Bar Chart and hoped that the next panel would offer me a chance to narrow it to a specific subtype. It didn’t.

Instead, the wizard presented a form where I could drag and drop my fields and preview my report. Although I’d used this form on other occasions, I still found it cumbersome. After some experimentation, I settled on Axis set to Task ID and Data set to Original

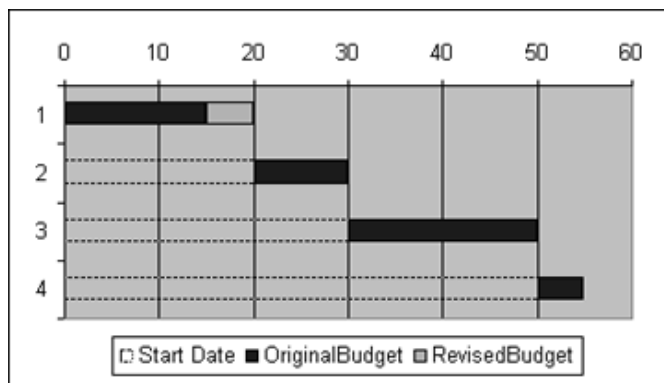


Figure 2. Gantt chart prototype in Excel.

Budget and Revised Budget. I knew I wanted one series (stacked), but, paradoxically, it looked most accurate when I left Series set to nothing.

The next form in the wizard surprised me again. Because I’d already set a Rowsource for rptGantt for the fields I’d painted in the detail section, the wizard offered me a way of linking report fields to chart fields. This would have the report data “drive” the chart. I guessed that this would make sense if I’d painted the chart in the detail section of the report, and made the chart smaller. However, since I wanted to show the chart once, right at the start of the report, in its own report group, I didn’t need to link any fields to the chart. I blanked the fields out on the form (later, I confirmed that this panel only appears for bound reports).

The last panel invited me to name my chart’s title, show or hide the legend, and get help with customizing the chart (which I declined). I clicked the Finish button, which closed the wizard and brought me back to rptGantt and my new, rather plain chart, which I christened objChartGantt and saved.

In Figure 3, you can see my result (rptGantt), which loads when you open the GANTT.MDB database, also included in this month’s Source Code file.

Professional touches

I spent a lot of time formatting what I now consider “my look” for charts. I suspect you’ll also choose to create your own look (aesthetics are always subjective). However, to get you started, I’ve listed most of the features that I used to achieve the look of the sample application. Note, though, that these are MS Graph tips rather than Access tips. I set these options after I’d double-clicked the objChartGantt (or right-clicked and selected Chart Object | Edit), which invokes MS Graph. I

Gantt Demo, and “My Look”

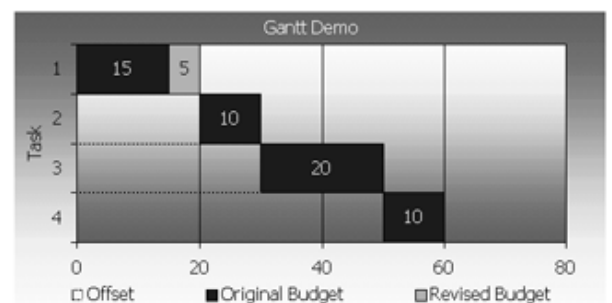


Figure 3. Finished Gantt chart in Access.

used the shortcut menus to manipulate the features, so I've prefaced each tip with the object that you must select to find the feature. You can also use MS Graph's menus and toolbars to achieve the same results.

Datasheet Window: Sample Data

MS Graph offered some default data for me in its datasheet, but it wasn't a good fit for a Gantt chart. To help me visualize my final product, I replaced that default data with my own, as shown in Figure 1.

Chart Area: ChartType

For my Gantt chart to work, I needed to stack the bars. I clicked the chart once, near the top left corner. Handles appeared, and the ToolTip text said "Chart Area." I right-clicked, chose "Chart Type..." and was happy to see the first form from the Excel wizard, with the subtype options. There are nearly 100 different charts available. I chose Stacked Bar.

Chart Area: Chart Options

I right-clicked the Chart Area again and chose "Chart Options..." The third form from the Excel wizard appeared. I set my Titles, Axes, Gridlines, Legend, Data Labels, and Data Tables preferences there.

Chart Area: Format Chart Area

I right-clicked the Chart Area one final time and chose "Format Chart Area..." On the Patterns tab, I set Border to None and then clicked the Fill Effects button to choose a shaded gradient—a fast way to make the chart more appealing. The Font tab on this dialog box applies to all text on the chart. Normally, I use the Onyx font for my look: It's skinny, but legible. For the demo, I chose the Tahoma font, which is bigger. Later, I overrode some of these default font settings for certain fields (for example, Title color, Legend border) by double-clicking and formatting them individually.

Plot Area: Format Chart Area

I clicked the chart near the bars (but not on them) until handles appeared and the ToolTip text said "Plot Area." I right-clicked and chose "Format Plot Area..." For my look, I set the same gradient as on the Chart Area, but in the reverse direction. Gradients subtly guide the user's eyes to bars and make comparisons easier. I used to choose more colorful gradients, but they turned out too dark on black and white printers.

Category Axis: Format Axis

I clicked the left edge of the chart Plot Area until handles appeared and the ToolTip text said "Category Axis." I right-clicked and chose "Format Axis..." This exposed tabs to set the Axis' Pattern, Scale, Font, Number, and Alignment formats. As in my Excel prototype, I checked

"Categories in reverse order" to give the bars a top-left-to-bottom-right progression. Unfortunately, when I reversed the categories, the horizontal Value Axis moved to the top of the Plot Area, just as you see in Figure 2. I wanted that at the bottom, so I checked the "Value (Y) axis crosses at maximum value" option, which moved them there.

Series: Format Series

I found it hard to select a series. I clicked around inside one of the Offset bars, but the handles were around a single bar rather than all of the bars. I clicked closer to the center of the bar, and two handles appeared around each data label in each bar in the Offset series, but that was just the series' text. I clicked just right of center, and, finally, *one* handle appeared on each of the other Offset bars, and the ToolTip text said "Series 'Offset' Value..."

Another trick I learned is to temporarily inflate a value: It makes it obvious which series you're hitting.

Either way, I right-clicked, then chose "Format Axis..." and set the Patterns (in the case of the Offset series, that was Border to Dashed and Area to None). I skipped the Axis tab, since my only series was already plotted on the Primary Axis by default. I also skipped the Y-Error Bars, since they weren't applicable. On the Data Labels tab, I chose "None" for the Offset series and "Show values" for the other two series. On the Options tab, I set the Overlap to 100 percent and the Gap width to 0, giving the aligned, touching bars that I wanted for my look.

Value Axis: Format Axis

I chose to leave my Value Axis in days, showing the cumulative effect of the tasks. I'll leave it to you to try to show them as dates.

Chart Window: Sizes

I wanted the Chart's overall size to fit the space I'd reserved in objChartGantt. I learned to watch the bottom left corner of the chart in Access in the background while I resized the edges of the Chart Window, the Chart Area, and the Plot Area in MS Graph. I also learned to think twice before reopening a chart in MS Graph once I'd sized it properly. MS Graph has a maddening habit of shrinking the chart slightly whenever you open it, whether you change anything or not!

objChartGantt: Other Properties

Back in Access, I also set the objChartGantt's RowsSource to qselGantt, since the wizard transformed it, and I made the borders transparent. I also set the SizeMode to Clip, although I usually use Zoom. I previewed rptGantt and created Figure 3 for my article.

Uncharted territory

The other reason that I created my Gantt demo on a report

was to warn you that it's harder to work with a chart on a report than on a form. I say that because (are you ready?) the chart's behavior *changes*. Surprised? So was I! So to spare you some frustration, I've summarized three of my most disconcerting discoveries in [Table 1](#).

You can use my techniques to add professional-looking Gantt charts to your Access (and Excel) applications that will impress your users. I find that getting a chart to look just right is fun and satisfying, and it speaks volumes to my users (at least a thousand words!). The tips and workarounds I've shared here

should inspire you to experiment with the MS Graph library's Chart object in your own applications. ▲

 [GANTT.ZIP at www.smartaccessnewsletter.com](http://www.smartaccessnewsletter.com)

Doug Den Hoed is a founder of Lumina Systems Delivery in Calgary, Canada, which specializes in customized software solutions using Access, Visual Basic, InterDev, SQL Server, and Oracle. Doug drew this article from The KB™ (<http://www.thekb.com>), his Access-based commercial package for managing software development projects. doug.denhoed@home.com.

Table 1. Chart properties differ on forms and reports.

| Gotcha | Details, advice, and workarounds |
|----------------------------|---|
| Property imparity | After several hours of denial, I finally confirmed the awful truth: A chart painted on a form has more properties (61) than the exact same chart painted on a report (42). My emphasis is on the differences. If you're counting on changing some properties at runtime on a report, I recommend that you enumerate the properties in design mode (for example, for each prp in objChart.Properties debug.print prp.name...), note what's available, and test with Access's /runtime startup option. |
| Can't see/change Rowsource | One of the main features in my real application lets users change the chart's criteria on the fly. On a form's chart, it was trivial: I changed the chart's Rowsource, and the chart obligingly repainted itself with the new data. On a report's chart, however, I couldn't even see the Rowsource property, let alone set it! I spent several hours superstitiously repainting in different ways to see whether I could get the chart to "keep" its properties. No luck. I did find a workaround: Bind the chart to a working query (for example, qtmpChart), and manipulate the query's SQL property at runtime. It's a fine line between a cool technique and cheating. |
| Can't change ChartType | The ChartType is another powerful property that disappears when you paint the chart on a report. One workaround I can offer is to paint one chart for each type you might need, and use a Select Case to set the Visible property of the chart that you want to display. This solution has its own problems, as the invisible charts take up resources even if you use separate temporary queries and try to minimize their impact by setting them to impossible conditions (for example, giving the query a WHERE clause of 1=2). Ultimately, I abandoned this approach and created a separate report for each chart type. |

Accessing Records...

Continued from page 17

The other fields provided by the Internet Publishing provider include the name of the parent directory (RESOURCE_PARENTNAME) and flags indicating whether the file is hidden or read-only, among other information.

You can add new fields to the collection just by referring to them. For instance, this code adds a new field called Handled to a Record's Fields collection and sets it to True:

```
rc.Handled = True
```

The field that's created will have the Variant data type, which Microsoft's documentation says isn't yet supported by ADO. The fields that are added are only temporary unless you use the Fields collection's Update

method, at which point, presumably, the underlying physical entity will be updated with the new field and its content. The Internet Publishing provider won't update a file or directory, so it will only let you add fields temporarily. Calling the Update method of the Fields collection for the Internet Publishing provider will generate the message "Current Provider does not support adding and deleting columns on the Record object."

Working with files

In addition to browsing the structure of the Web site and retrieving files, you can use the Record object to change the files on the site by using the Record object's CopyRecord, DeleteRecord, and MoveRecord methods.

The DeleteRecord method is the simplest. This method, used without a parameter, deletes the resource specified by the Record object. This code deletes the physical entity pointed to by the Record object:

```
rc.DeleteRecord
```