Every Access database developer knows that tables must be indexed properly to optimize query performance. Recently I was working on a database that used lots of queries, with one query in particular was that very slow to run. It took about one or two minutes to complete, and I attributed the poor performance to its being a UNION query of several other queries. Then I remembered JETSHOWPLAN, and decided to see if I could improve its performance.

If you are unfamiliar with JETSHOWPLAN, there is a description at http://builder.com.com/5102-6388-5064388.html. It's a good introduction, but it doesn't show you how a bad design appears in the output file. That's where I come in! Snippets of the output file from three runs of the query are shown below. Steps 01 thru 09 are identical for each run, so I snipped them for brevity.

I had used JETSHOWPLAN previously, but never with much success -- perhaps my queries were already optimal (at least, from Jet's perspective). But this query's output included items that were new to me, and I used them as clues to the table design optimization.

"Scanning" can be a performance-killer because it means that Jet must examine every record in the table, rather than using an index to restrict the number of records that it examines. But the scanning in steps 01 and 07 is on small tables in a field that has only two values, and Jet's optimizer realizes there is not much performance to be gained by indexing a binary field.

Notice the use of "temporary index" in steps 04 and 06. So I added indexes to the tables according to the fieldnames shown in 04 and 06. No change to query performance.

Next I looked at step 12 -- notice it's using an existing primarykey index, and that the join uses a field named AVIRawData.LandUnitIDfk. AVIRawData.LandUnitIDfk is the foreign key in a large table, so it should be indexed. With that change, the query ran instantaneously. Running the query for a second and third time confirmed the speed improvement -- what used to take one to two minutes was now taking about half a second. WOW!

Examining the third run's output file showed that Jet had changed the query order. A different table was scanned in step 10, different joins were performed in steps 11 and 12, and the new index was used in step 11.

What did I learn from this exercise?
- JETSHOWPLAN can help find suboptimal queries, but don't expect them to leap off the page at you. You must examine each step carefully and understand its implications. You still need to know whether the join fields are indexed.

- JETSHOWPLAN is a PITA to make work properly. You need to modify the Registry to turn it on and off (requiring Administrative permissions on the computer to do so!). It also requires an Access restart to come into effect.

- PITA II -- JETSHOWPLAN It requires the query to be re-compiled before it will write its output, therefore, existing queries must be opened, modified, saved, and run before their results will appear in the output file.

- PITA III -- you should save the output files into "version sets" for comparison. JETSHOWPLAN seems to write its output file (showplan.out) into the My Documents folder, but you may need to search your computer to find it.

- temporary indexes are still a mystery to me, but they don't appear to be performance-killers.

I hope this information can benefit somebody else...

```
' ORIGINAL VERSION - slow

01) Restrict rows of table tblSpecies
      by scanning
      testing expression "tblSpecies.PotentialBiomass=-1"
02) Inner Join result of '01)' to table 'AVIRawData'
      using index 'AVIRawData!tblSpeciesAVIRawData1'
      join expression "tblSpecies.SpeciesID=mqryPotentialBiomassSitesRAW.bio_sp"
      then test expression "AVIRawData.SubjectiveDeletion=0"
03) Sort result of '02)'
04) Inner Join table 'tblBiomassVolumeClass' to result of '03)'
      using temporary index
      join expression "tblBiomassVolumeClass.ForestCoverGroupClassIDfk=mqryPotentialBiomassSites.F_COVGRP And
tblBiomassVolumeClass.DensityClassIDfk=mqryPotentialBiomassSites.DENSITY And
tblBiomassVolumeClass.SpeciesIDfk=mqryPotentialBiomassSites.bio_sp"
05) Sort table 'tblHeightClass'
06) Inner Join result of '04)' to result of '05)'
      using temporary index
      join expression "tblBiomassVolumeClass.HeightClassIDfk=mqryPotentialBiomassSites.HeightClassID"
      then test expression "mqryPotentialBiomassSitesRAW.HEIGHT<[Lessthanboundary] And
mqryPotentialBiomassSitesRAW.HEIGHT>=[Greaterthanequalboundary]"
07) Restrict rows of table tblLandUnit
```

```
     by scanning
     testing expression "tblLandUnit.IncludeInBiomassCalculation=-1"
08) Sort result of '07)'
09) Inner Join result of '06)' to result of '08)'
     using temporary index
     join expression "AVIRawData.LandUnitIDfk=tblLandUnit.LandUnitID"
10) Restrict rows of table tblSpecies
     by scanning
     testing expression "tblSpecies.PotentialBiomass=-1"
11) Inner Join result of '10)' to table 'AVIRawData'
     using index 'AVIRawData!tblSpeciesAVIRawData1'
     join expression "tblSpecies.SpeciesID=mqryPotentialBiomassSitesRAW.bio_sp"
     then test expression "AVIRawData.SubjectiveDeletion=0"
12) Inner Join result of '11)' to table 'tblLandUnit'
     using index 'tblLandUnit!PrimaryKey'
     join expression "AVIRawData.LandUnitIDfk=tblLandUnit.LandUnitID"
     then test expression "tblLandUnit.IncludeInBiomassCalculation=-1"

' FIRST EDIT - still slow
<snip>
10) Restrict rows of table tblSpecies
     by scanning
     testing expression "tblSpecies.PotentialBiomass=-1"
11) Inner Join result of '10)' to table 'AVIRawData'
     using index 'AVIRawData!tblSpeciesAVIRawData1'
     join expression "tblSpecies.SpeciesID=mqryPotentialBiomassSitesRAW.bio_sp"
     then test expression "AVIRawData.SubjectiveDeletion=0"
12) Inner Join result of '11)' to table 'tblLandUnit'
     using index 'tblLandUnit!PrimaryKey'
     join expression "AVIRawData.LandUnitIDfk=tblLandUnit.LandUnitID"
     then test expression "tblLandUnit.IncludeInBiomassCalculation=-1"

' FINAL VERSION - fast
<snip>
10) Restrict rows of table tblLandUnit
     by scanning
     testing expression "tblLandUnit.IncludeInBiomassCalculation=-1"
11) Inner Join result of '10)' to table 'AVIRawData'
     using index 'AVIRawData!LandUnitIDfk'
     join expression "tblLandUnit.LandUnitID=AVIRawData.LandUnitIDfk"
     then test expression "AVIRawData.SubjectiveDeletion=0"
12) Inner Join result of '11)' to table 'tblSpecies'
     using index 'tblSpecies!PrimaryKey'
     join expression "mqryPotentialBiomassSitesRAW.bio_sp=tblSpecies.SpeciesID"
     then test expression "tblSpecies.PotentialBiomass=-1"
```