

GPSQL – Colorado Springs Advanced Course

Note - This document is intended for individuals who currently have a solid working knowledge of GPSQL but want a more detailed look at the underlying technology, options and query building.

Brief review - GPSQL is a front end for a Microsoft Access Database (MDB). Microsoft own front end application called 'Access' is not required on the users system because the core 'data access objects' libraries aka, DAO) is installed along with GPSeismic. MDB databases have the following limitations:

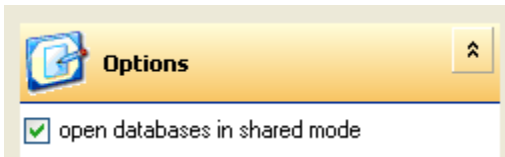
| Item | Maximum sizes/numbers |
|---|-----------------------|
| Database size | 1 Gb |
| Number of characters in an object name | 64 |
| Number of characters in a password | 14 |
| Number of characters in a user name or group name | 20 |
| Number of concurrent users | 255 |
| Number of characters in a table name | 64 |
| Number of characters in a field name | 64 |
| Number of fields in a table | 255 |
| Number of characters in a Text field | 255 |
| Number of characters in a Memo field | 65,535 / 1 Gb |

Strategies for large databases - While the maximum file size would seem to allow MDB databases to handle almost any conceivable project, large databases below this size still cause resource problems (read sluggishness). What you can do to avoid this? One key point is that things slow down when you have many records in a table, particularly the POSTPLOT table which has over 50 fields.

One thing you can do if you have quite a bit of culture is to keep this in a separate table. Do this by using one of the Append/Create Table routines in the Modifications menu. Its also possible to keep data such as culture in a separate database. Do this by selecting the 'Culture' database in QuikView and using masks to filter normal production or opening the 'Culture' database and using the 'MDB Table' item in the Import menu.

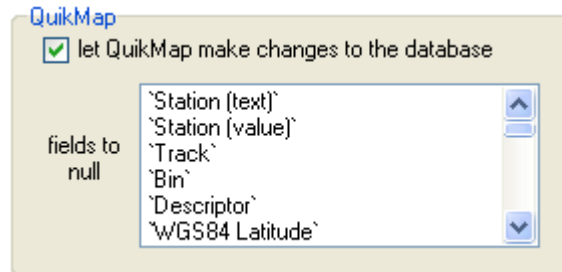
Another thing you can do is to use 'Compact/Repair' occasionally, an action analogous to defragging a hard drive. Two others involve liberal use of Aliases and Table Links. Aliases are discussed in detail later in this document. Linking to a table allows you to use a table in another database as if it were just another table in your open database. However, while all reporting routines are available to you, practically all modification routines are unavailable.

Future development – We are staying with the MDB database for the foreseeable future. The only possible change might be an eventual shift to what is called SQL Server. This type of database has better performance statistics then MDB databases. There is also a free version available. The downside is that the installation is very heavy, more users or greater size is not yet a factor in the debate, and many current clients actually have other MDB dependant software or are themselves Access users.



Shared mode notes – If you are going to want to access the database using a couple of applications simultaneously, make sure that you have selected 'shared mode' in Project Manager options. For example, if your intention is to launch QuikMap from

GPSQL and give QuikMap the ability to update the database, this option has to be selected and in GPSQL, when you get to the field selection dialog, you have a second checkbox to check. The 'fields to null' can be explained by an example. Lets say you move some points in QuikMap and update the database. The grid coordinates change but the latitude and longitude don't. By selecting the latitude and longitude fields here, any points that move result in erasure of the latitude and longitude field contents for those records.



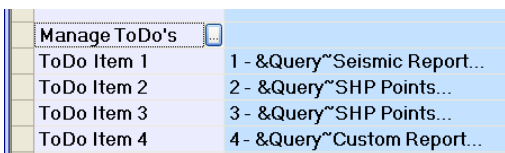
Useful Options - There are over a dozen options on the Miscellaneous dialog. Many were placed there to satisfy one or two users. In my opinion the most useful are these:

Create copy of database on startup – This does what it says. The name of the backup is the same name of the database but has a 'BU' extension. Just change the extension to MDB if you need to. Otherwise the backup is re-written each time you open the database. You are prompted for whether to delete when you close the application.

Remember last field selections for each query – If this is on, each query (0-99) remembers the last fields you used in the Field Selection dialog. The point is, this might or might not be right. If your query changed, you might be referencing a whole new table so the field selection would be wrong. If you always use the same queries, and IF you use non-default fields, this is for you..

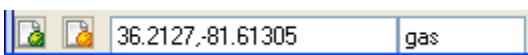
Attempt to restore geodetic parameters when opening a database – With this on, every time you open a database, GPSQL looks for the project PRJ file and automatically changes the coordinate system and geoid model to what's inside.

Confirm/Lock Transformations ... - Turn these on and when you do something a coordinate system (e. g., xy -> lat/lon conversion), if your coordinate system selection doesn't match what the current database coordinate system is, you get slapped on the wrist.



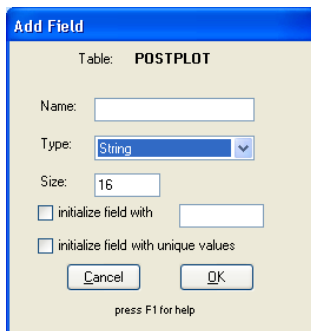
User Interface (UI) Subtleties – If you are training someone and want them to perform a number of routines in a particular order, you could write them down on a piece of paper, but a cooler way is to set this up in the ToDo list available in the

Current Settings panel of the UI. Essentially, it starts the user out on the road to performing each function.



There's two text boxes at the bottom of the UI. You can enter coordinates here and launch Google

Earth or your browser (and Google Maps) with the results. Note that its possible to copy coordinates in the spreadsheet to the clipboard.



Adding Fields To The Database – You can click on a table, right-click and select 'Add Field' to add a field to a table. Generally, a field is numeric or alphanumeric (string type). If you add a string, you must specify a length. If you specify a numeric field, you need to choose what type. If you don't know, make it a double which can be very small or large values.

If you need to change the length of a string field, it can be done without losing data.

Note that when you create a project, there is a new feature that allows you to create a number of user fields. I personally feel you should limit the number of user fields and an option to them are often something called aliases which are discussed later in this document.

A couple of options when adding a field is to initialize each record with a value or string or in the case of a 'long' type, a unique value.



Why Two Spreadsheets And Which Should I Use – There are two spreadsheets that allow viewing and editing. We call these the 'modern' and 'classic'. The spreadsheets are based on third party libraries and use two totally different database technologies. The classic uses something called DAO which allows it to directly access the database. When you edit something in this spreadsheet, you are directly editing the database. There is no 'Save' menu item like the modern spreadsheet. The modern spreadsheet uses a technology called ADO. When you display a query, this spreadsheet actually makes a copy of the query results in memory and displays it. When you edit something, the database is NOT modified. When you finish making the edits, the modern spreadsheet, armed with a list of what you did, modifies the parent database.

😊 What's the advantages of the modern? - There's a couple of reasons, one of which is that if we move to SQL Server in the future, it is based on ADO and the spreadsheet will fit in nicely. Second is that it is capable of many different looks including vertical and horizontal splits, inverted views, etc.

😞 What's the disadvantages of the modern? – Because it copies the results of the query into memory, it doesn't display a lot of data quickly. Also, ADO requires that the query have something called an index. This is added and removed each time you go into the modify mode. Finally, when you edit something, there will always be the additional step of saving.

😊 What's the advantage of the classic? – Speed with large queries and direct editing.

😞 What's the disadvantage of the classic? – Only capable of one view.

Which should I use – Get familiar with both. Because it can handle small queries (1-25,000 records) efficiently, we use the modern spreadsheet as the one and only view/edit mechanism in all other GPS seismic applications. The classic, however, would actually be my choice for very large queries. Both will be available for the foreseeable future.

😬 I can't make edits in the modern spreadsheet. What up? – There's a couple of possibilities:

1 – If you are editing data, remember that this is a copy of the data in the database. If someone edits the database while you are, when you go to save, you will get a 'concurrency' error. This means someone accessed the database concurrently.

2 – If you get an error trying to get into the view/edit mode that says something about not being able to add the primary index, chances are you already have an index. Look at the 'Type' of each field on the main interface. Do you see '(index)' or '(AI)' behind any? 'AI' stands for 'automatic incrementing' by the way and either type cannot be tolerated. There is a 'Remove Indexes' utility in the Table menu that you can use, but you may have to manually delete the field.

3 – Any error encountered trying to save is sometimes caused by a corrupt database. You should not hesitate to use the 'Compact/Repair' feature in the File menu.

Advanced Query Building - Its assumed that the reader can and has built a number of queries and requires no instruction about constructing a query with multiple criteria. Therefore, this document skips over this aspect of query building. We will cover three items: 1) Query building tools that you might not realize are there, 2) Joins, and 3) Aliases.

Distinct Key Word – Let's make this brief. This isn't a very useful item. If you use it, and there are exact duplicates based on the field selection of your query, then only one record is returned. For example, consider this query:

```
Select [POSTPLOT].`Track` From [POSTPLOT]
```

You will get every record in the database. However, add the DISTINCT keyword, e. g.:

```
Select Distinct [POSTPLOT].`Track` From [POSTPLOT]
```

And you will get a list of unique Track numbers. You could possibly use it to find exact record duplicates and indicate that you need to purge the database. For example, consider the following:

```
Select DISTINCT [POSTPLOT].`WGS84 Latitude`,`[POSTPLOT].`WGS84 Longitude`,`[POSTPLOT].`Survey Time (GMT)` From [POSTPLOT]
```

If the number of records without the 'DISTINCT' key word is greater than with, you probably have to use the purge routine.

Miscellaneous Query Helpers – The query builder has a few helpful items to assist you in entering criteria using some seldom used keywords. These include 'Between' and 'In'. They are also useful in helping you identify fields without contents.

Wildcard Characters – One thing you might not know is that there are a couple of different flavors of SQL floating around. One flavor uses the asterisk (*) as a wildcard and a question mark (?) as a single character wildcard. Another flavor uses the percentage sign (%) and the underscore (_) for the same wildcards. GPSeismic uses the former, however, when you write a query and display it in the modern spreadsheet (which uses the second flavor), our software analyzes the query and replaces any asterisks with percentage signs and any question marks with underscores before executing the query. At present, we don't do anything with underscores and percentage signs you might use in queries displayed by the modern viewer because it's difficult to figure out what context they are being used. Keep this in mind because you might get some unexpected results.

So how do you write a query to find all records where a string field has a wildcard? You would bracket it. So instead of...

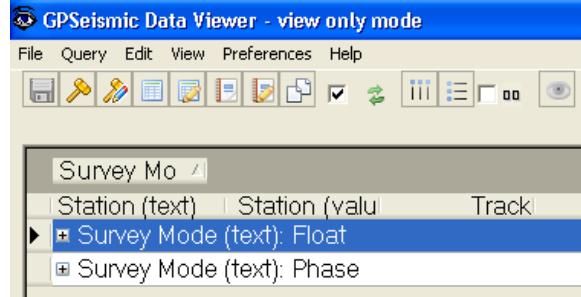
```
Select [POSTPLOT].* From [POSTPLOT] Where [POSTPLOT].`Station (text)` Like '*?*'
```

...you would write...

```
Select [POSTPLOT].* From [POSTPLOT] Where [POSTPLOT].`Station (text)` Like '*[?]*'
```

The same holds true for any wildcard.

Group By queries – Group By queries are often used to do what their name implies, that being to create groups of records based on some field. For example, you could group records by Survey Mode. The modern viewer has a Group By viewing option and produced the graphic here.



This is cool but not useful if you are looking for a quick way to get the total float and total phase. There is a query that will actually display the totals but you can't build it from the query building dialog. There is a Table menu item that will build the Group By query for you. Lets say your pack operators are assigned specific GPS receivers so you can determine the operator by the receiver serial number. Let us further assume you want to see a total of all points grouped by receiver serial number (pack op). This is what you do:

- 1- select an unused query
- 2- click on the field 'Receiver SN' on the main interface
- 3- Right click to display the Table menu and select 'Insert Group By ...' item
- 4- Answer Yes to the prompt and a query will be pasted in the query text box that looks like this:

```
SELECT [POSTPLOT].` Receiver SN` , COUNT([POSTPLOT].` Receiver SN` ) AS Total FROM [POSTPLOT] GROUP BY [POSTPLOT].` Receiver SN`
```

| Receiver SN | Total |
|-------------|-------|
| | 0 |
| 453072 | 225 |
| 452892 | 226 |
| 453047 | 261 |
| 452594 | 354 |
| 452827 | 823 |
| 457638 | 1024 |

Displaying the query will give you something like this, where you get all distinct occurrences of receivers and how many there were. Unfortunately, GPSQL will balk at creating a custom or special report, but you can always copy and paste or export a CSV file. Importantly, this is useful data that gives you how many points were shot by each pack op. You can modify the query, but you have to do it manually. It

turns out that if required, you could copy and paste a 'Where' clause in order to restrict the information to some criteria. For example, I've modified the query above to be for a specific day.

```
SELECT [POSTPLOT].` Receiver SN` , COUNT([POSTPLOT].` Receiver SN` ) AS Total FROM [POSTPLOT] Where ([POSTPLOT].` Julian Date (Local)` =2007206) GROUP BY [POSTPLOT].` Receiver SN`
```

Note that the 'Where' clause goes right before the 'GROUP BY'.

Another useful Group By query is this one:

```
SELECT [POSTPLOT].` Track` , COUNT([POSTPLOT].` Bin` ) AS Total FROM [POSTPLOT] GROUP BY [POSTPLOT].` Track`
```

This gives you a list of all tracks and how many bins for each. You might want to modify it with a 'Where' clause to ensure culture and other records don't get included:

```
SELECT [POSTPLOT].`Track`, COUNT([POSTPLOT].`Bin`) AS Total FROM [POSTPLOT] Where ([POSTPLOT].`Track` > 0) GROUP BY [POSTPLOT].`Track`
```

The following query is a tricky way to produce a list that allows you to spot duplicates:

```
SELECT [POSTPLOT].`Station (value)`, COUNT([POSTPLOT].`Bin`) AS Total FROM [POSTPLOT] GROUP BY [POSTPLOT].`Station (value)`
```

If there were no duplicates based on 'Station (value)' there would be '1' as the count for each record. Click on the spreadsheet column head twice to order in a descending manner and the duplicates will rise to the top.

'COUNT' is something called an aggregate function. There are several others including AVG, SUM, MAX, MIN. Sometimes they can be used effectively to isolate information. For example, supposed we wanted a list of each receiver serial number and what the highest PDOP was recorded by each. This query would do it:


```
SELECT [POSTPLOT].`Receiver SN`, MAX([POSTPLOT].`PDOP`) AS MaxPdop FROM [POSTPLOT] Group By [POSTPLOT].`Receiver SN`
```

Change the 'MAX' to 'AVG' and we will get a concise list of the average geometry each pack operator is utilizing for his overall work.

Table Joins – Queries that employ table joins are useful for a number of reasons. For example if you wanted to generate a report in which each record contained both the survey coordinates and the preplot coordinates, then a table join will do the trick:

```
Select [POSTPLOT].*, [PREPLOT].* From [POSTPLOT], [PREPLOT] Where [POSTPLOT].`Station (value)` = [PREPLOT].`Station (value)`
```

Notice the criteria in which we are only generating records where there is a match in the field, 'Station (value)'. This join is an effective way to not only make the fields of both tables available to you, but to also insure that you only have points that have been surveyed.

 *Table join downsides* - There are only a couple of cautions. One is that if you try to execute a join without a field specified to join them on (Station (value) in the query above), the query will return a number of records that is the number of records in one table times the other. This means that if you have 50,000 records in both tables, the query will return 2.5 billion records!

The other downside is that table joins are not actually editable, so practically all modification routines will fail. One notable exception is updating a field using the Update dialog.

Left and Right Joins – The default join is called an 'Inner' join. The 'Inner' keyword is not required. There are other type of joins called Left' and 'Right' that look exactly like the query above but do contain the 'Left' and 'Right' keywords.

Here is a 'Left' join:

```
Select [POSTPLOT].*,[PREPLOT].* From [POSTPLOT] Left Join [PREPLOT] On  
[POSTPLOT].`Station (value)`=[PREPLOT].`Station (value)`
```

And here is a right join:

```
Select [POSTPLOT].*,[PREPLOT].* From [POSTPLOT] Right Join [PREPLOT] On  
[POSTPLOT].`Station (value)`=[PREPLOT].`Station (value)`
```

As you can see, they are exactly the same except the single keyword. However, what they do is totally different. The 'Left' join will produce all records of the table listed immediately after the 'Select' keyword along with matching records of the other. If, for a particular record, there is no match, you still get the fields of the other but they have no content. For example, lets say we had 4 records in the Postplot table and 2 in the Preplot table. A Left (Postplot) table join query would produce 4 records:

| | |
|-------------------|---------------------------|
| Postplot record 1 | Preplot matching record 1 |
| Postplot record 2 | Preplot matching record 2 |
| Postplot record 3 | |
| Postplot record 4 | |

A right join for the same tables would produce 2 records. In this case both Preplot and Postplot fields would be filled because each existing Preplot record has a match.

In GPSQL, tables are always listed alphabetically, therefore a Left join is always as described above, namely, all records of the Postplot table. We could use this to our advantage to find all survey data that does not have a match in the Preplot table. This could be useful information because it essentially represents everything we surveyed without a preplot (no match). The key is to take the Left table join query and add some criteria that eliminates the matches. The query below will do that:

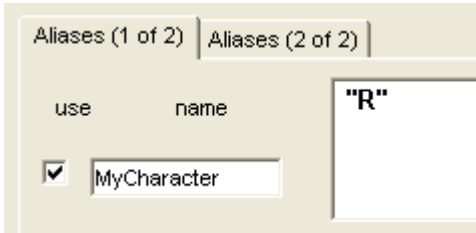
```
Select [POSTPLOT].*,[PREPLOT].* From [POSTPLOT] Left Join [PREPLOT] On  
[POSTPLOT].`Station (value)`=[PREPLOT].`Station (value)` Where ([PREPLOT].`Station (text)`  
Not Like '*')
```

Aliases – An alias is an expression in a query that appears to add a field to the table! However, its really not a field and this is a good thing because it allows you to enhance you reporting and other capabilities without increasing the size of the database. As our first example, lets say you have to produce a SEG file where each record has the format below:

```
R5121      51211189 02491672N074265872W25987965 3290073  0
```

Here is a SEG file that has an 'R' and the track number in the first few columns of the record. What do you do? You could make the file and go at it with a good text editor, but that's fairly labor intensive.

When you build a query, got to the Alias tab page and make it look like this:



Now when you use that query in any way, you will notice an extra field called 'MyCharacter' that has an 'R' in it:

| MyCharacter | Station (text) | Station |
|-------------|----------------|---------|
| R | 51211189 | 5 |
| R | 51211190 | 5 |

This 'alias' field is also available in the custom and seismic report builder like any other field. The actual query looks like this:

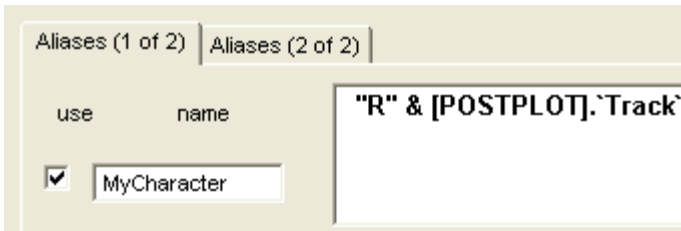
Select "R" AS `MyCharacter`, [POSTPLOT].* From [POSTPLOT]

You should note the syntax of an alias is

Expression AS Name

So in the above query, our expression was simply one character (surrounded by quotes) and we gave it the name of 'MyCharacter'

However, this is still not exactly what we want because we want to see 'R5121', not just 'R'. here is where we step it up a notch and make the alias a bit more complex:



Notice we added an ampersand which is used to concatenate items, and the Track field.

This displays as:

| MyCharacter | Station (text) | Station |
|-------------|----------------|---------|
| R1287 | 12875228 | 12 |
| R1287 | 12875229 | 12 |
| R1287 | 12875230 | 12 |
| R1293 | 12935211 | 12 |
| R1293 | 12935212 | 12 |

And the query looks like this:

```
Select "R" & [POSTPLOT].`Track` AS `MyCharacter`, [POSTPLOT].* From [POSTPLOT]
```

Now you have everything you need to make the file. In the field selection dialog, swap 'MyCharacter' with Descriptor. Here are a number of aliases and what they do:

Simple String Manipulation

| | |
|--|---|
| Creates a field that is all uppercase . Use 'LCASE' for lower case. | Select UCASE([POSTPLOT].`Descriptor`) AS `MyAlias`, [POSTPLOT].* From [POSTPLOT] |
| Creates a field that is the leftmost 4 characters of the Station (text) field. Use 'Right' to get the rightmost characters. | Select Left([POSTPLOT].`Station (text)`, 4) AS `MyAlias`, [POSTPLOT].* From [POSTPLOT] |
| Creates a field which is the three characters of the Station (text) field starting from the second character. | Select MID ([POSTPLOT].`Station (text)`, 2, 3) AS `MyAlias`, [POSTPLOT].* From [POSTPLOT] |
| Removes the spaces from either side of a string field. There is also 'LTRIM' and 'RTRIM' to remove left and right spaces only. | Select TRIM ([POSTPLOT].`Station (text)`) AS `MyAlias`, [POSTPLOT].* From [POSTPLOT] |

Date Related Aliases

| | |
|---|---|
| Creates a field that is the number of days ago the point was shot | Select DateDiff("d", [POSTPLOT].`Survey Time (Local)`, Now) AS `MyAlias`, [POSTPLOT].* From [POSTPLOT] |
| Creates a field that is the Julian Day the point was shot | Select DateDiff("d", "01/01/07", [POSTPLOT].`Survey Time (Local)`) AS `MyAlias`, [POSTPLOT].* From [POSTPLOT] |

Graph-like Aliases

| | |
|--|---|
| Creates a field that graphs DOP | Select String([POSTPLOT].`PDOP` * 10, " ") AS `MyAlias`, [POSTPLOT].* From [POSTPLOT] |
| As above but slightly different look | Select " " & String([POSTPLOT].`PDOP` * 10, "-") & " " AS `MyAlias`, [POSTPLOT].* From [POSTPLOT] |
| Creates a field of point/reference baselines | Select INT([POSTPLOT].`GPS Baseline`) & " m " & String([POSTPLOT].`GPS Baseline` / 120, ".") & CHR(206) AS `MyAlias`, [POSTPLOT].* From [POSTPLOT] |
| Creates a height profile | Select " " & String([POSTPLOT].`Local Height`, ".") & INT([POSTPLOT].`Local Height`) AS `MyAlias`, [POSTPLOT].* From [POSTPLOT] Order By [POSTPLOT].`Station (text)` [POSTPLOT] |

Note: You should display the range of a value before you create the aliases above because its often necessary to scale the number to provide a reasonable number of characters.

Coordinate/Height Manipulation Aliases

| | |
|--|---|
| Creates a field that is the difference between local height and DEM height (assuming the latter exists). | Select [POSTPLOT].` Local Height` - [POSTPLOT].` DEM Height` AS `MyAlias` ,[POSTPLOT].* From [POSTPLOT] |
| There are two aliases here that use the offset fields and survey points to produce the original preplot. | Select [POSTPLOT].` Local Easting` - [POSTPLOT].` Offset (East)` AS `Preplot Easting` ,[POSTPLOT].` Local Northing` - [POSTPLOT].` Offset (North)` AS `Preplot Northing` ,[POSTPLOT].* From [POSTPLOT] |
| Yes, you can use aliases with table joins. There are two aliases here that produce the difference between the preplot and postplot coordinates. | Select [POSTPLOT].` Local Easting` - [PREPLOT].` Local Easting` AS `DX` ,[POSTPLOT].` Local Northing` - [PREPLOT].` Local Northing` AS `DY` ,[POSTPLOT].*,[PREPLOT].* From [POSTPLOT],[PREPLOT] Where [POSTPLOT].` Station (value)` =[PREPLOT].` Station (value)` |
| If you need an instant stub line, try this tactic. Create a query that isolates a desired line and then create an alias that manipulates the coordinates appropriately. Send to QuikMap, replacing the default coordinate fields with the alias. | Select [PREPLOT].` Local Northing` - 500 AS `MyAlias` ,[PREPLOT].* From [PREPLOT] Where ([PREPLOT].` Track` =1203) |

The following are example of aliases using the IIF keyword. The syntax is like this:

IIF ('Some Expression' , 'Do this if true', 'Do this if false')

So after the IIF keyword, there are three comma delimited items in parentheses. If the expression is true, the alias will display the second argument and if its false, the alias displays the last argument.

| | |
|--|--|
| Creates a field that says 'good dop' or 'bad dop' | Select IIF([POSTPLOT].` PDOP` >2,"bad dop","good dop") AS `GoodDopBadDop` ,[POSTPLOT].* From [POSTPLOT] |
| Creates a field that is whatever is in the survey mode (text) field unless the survey mode (value) field is 3. | Select IIF([POSTPLOT].` Survey Mode (value)` =3,"SuperFine",[POSTPLOT].` Survey Mode (text)`) AS `ReplaceField` ,[POSTPLOT].* From [POSTPLOT] |
| A very complex example. There are two aliases here which displays arrows indicating the cross line offset. | Select IIF ([POSTPLOT].` Offset (Crossline)` >.2,"->","") AS `OffsetRight` ,IIF ([POSTPLOT].` Offset (Crossline)` <-.2,"<-","") AS `OffsetLeft` ,[POSTPLOT].* From [POSTPLOT] Where (abs([POSTPLOT].` Offset (Crossline)`)>=.2) And ([POSTPLOT].` Track` >3000) |

More IIF Aliases

| | |
|--|---|
| As above but also indicates cross line distance | <pre>Select IIF ([POSTPLOT].` Offset (Crossline)` >0,INT([POSTPLOT].` Offset (Crossline)`) & "->", "") AS `OffsetRight`, IIF ([POSTPLOT].` Offset (Crossline)` <0,"<- " & abs(INT([POSTPLOT].` Offset (Crossline)`)), "") AS `OffsetLeft`, [POSTPLOT].* From [POSTPLOT] Where abs(INT([POSTPLOT].` Offset (Crossline)`)) > 10</pre> |
| As above but includes in-line information and is more verbose. | <pre>Select IIF ([POSTPLOT].` Offset (Crossline)` > 0, IIF([POSTPLOT].` Offset (Inline)` >0, "Offset to right and ahead", "Offset to right and back"), IIF([POSTPLOT].` Offset (Inline)` >0, "Offset to left and ahead", "Offset to left and back")) AS `Offset`, [POSTPLOT].* From [POSTPLOT] Where ([POSTPLOT].` Station (value)` >0)</pre> |

A Couple More Using the IsNumeric and IsNull Keywords

| | |
|--|--|
| This IIF alias does something based on if the contents of a field is numeric | <pre>Select IIF(IsNumeric([POSTPLOT].` Descriptor`), "OilWell", [POSTPLOT].` Descriptor`) AS `IsANumber`, [POSTPLOT].* From [POSTPLOT]</pre> |
| This IIF alias does something based on if there is something in a field. | <pre>Select IIF(IsNull([POSTPLOT].` Offset (Range)`), "undefined", Int([POSTPLOT].` Offset (Range)`)) AS `IsNothing`, [POSTPLOT].* From [POSTPLOT]</pre> |

Here's a list of all keywords that I know work:

- VAL (exp) - the value of the expression
- CSTR (exp) - the string value of the expression
- SIN (exp) - the trigonometric sine of the expression
- COS (exp) - trigonometric cosine of the expression
- TAN (exp) - trigonometric tangent of the expression
- INT (exp) - integer portion but converts -8.4 to -9 for example
- FIX (exp) - integer portion but converts -8.4 to -8 for example
- SQR (exp) - square root of the expression
- ABS (exp) - absolute value of the expression
- LOG (exp) - natural logarithm of the expression
- TRIM (exp) - removes any spaces left or right of the expression
- RTRIM (exp) - removes any spaces to right of the expression
- LTRIM (exp) - removes any spaces to left of the expression
- MID (exp,i,j) - returns the number of characters specified by j starting at i
- IIF (exp,a,b) - returns a if expression is true otherwise return b

DateDiff (format, date1, date2) - returns difference in two times where format is as shown below:

| | |
|---------|------|
| Year | yyyy |
| quarter | q |
| Month | m |
| Day | d |
| Week | ww |

String (exp1, exp2) - returns expression 2 a certain number of times dictated by expression 1
IsNumeric (exp) - returns -1 if expression is a number (true) and 0 if not (false)
IsNull (exp) - returns -1 if expression has no contents (true) and 0 if it does (false)

And finally, a slightly special case:

Exp MOD n - takes the expression and divides by 'n' and returns the remainder

A Few Final Facts About Aliases

Expressions can be used elsewhere - One important thing to keep in mind is that what you have learned here about aliases translates well to the topic of updating a field. If you actually want to modify a field, you can do that by selecting your query, selecting the 'Update Field(s)...' item from the Modifications menu, choosing the field you want to modify and finally picking an option. One of those options is to make the selected field equivalent to an expression. The expression can involve any of the keywords we have seen here. So for example, if you had actually added a string field to your Postplot table and wanted to make it equivalent to the Track and a preceding character, the expression might look like this:

`"R" & [POSTPLOT].`Track``

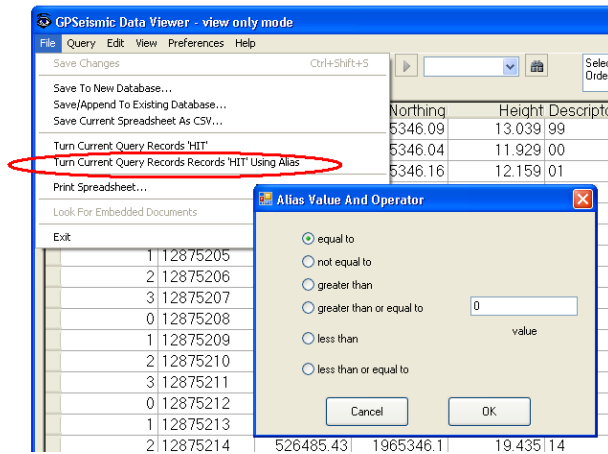
Using an Alias to flag points in QuikMap - One thing you can't do is put the cart before the horse. Another is to use the result of an alias to do something else. For example, if you tried to write a query where the criteria involved an alias it won't work. For example this won't work:

```
Select [POSTPLOT].* From [POSTPLOT] Where [POSTPLOT].`Local Height` -100 AS `MyHt` = 0
```

However, we did put a special function in QuikMap to deal with the result of Aliases. Lets say you have an initial layer in QuikMap and you want to turn all points divisible by 4 to hit status. What you could do is display the points in the modern spreadsheet (using the Outputs dialog), and then build a query with an alias as shown here:

`[PRIMARY].`Station` MOD 4`

This produces a field where only the stations divisible by 4 are zero. The rest have values greater than zero.



From the File menu select 'Turn Current Query Records 'Hit' Using Alias'. This displays a dialog that allows you to specify which of the alias values to use for turning the points hit.

There are some more complicated aliases you can create that allows you to pick out specific points. Recently, a client had interpolated half stations in QuikMap and wanted to isolate only 104.5, 106.5, 108.5, etc. The following alias created a field where only the

desired records had values of zero:

`([PRIMARY].`Station` - 104.5) MOD 2 = 0`

Essentially, the alias takes each station, subtracts 104.5 from it, divides it by 2 and returns the remainder.

Aliases can be used for all reporting purpose – Try to remember that aliases add flexibility to almost all reporting routines. Case in point is the production statistics, also known as the mileage calculator. Recently, a client requested that that entire routine be modified so as to include the BOL and EOL grid coordinates. This wasn't necessary because when the field selection dialog is presented you could simply replace the default Descriptor field with an alias, that alias being something like:

`INT ([POSTPLOT].` Local Easting `) & "/" & INT ([POSTPLOT].` Local Northing `)`

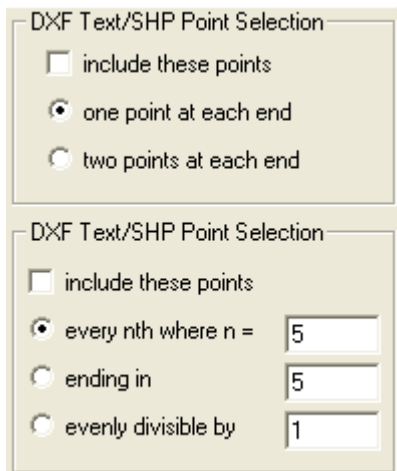
Using GPSQL To Assist You In Creating Maps

GPSQL has a number of tools that allow you to create layers for maps. This essentially come down to DXF and SHP files. If you are a GPArc or ArcView user, you probably want to stay with SHP files because these mapping applications render these types of files efficiently.

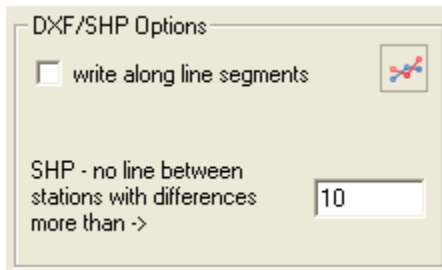
Multiple Map Utility – This utility is available in both GPSQL and GPArc. Note that in GPSQL, there is a 'classic' and 'modern' utility. The 'classic' will disappear by the end of the year so you want to make sure that if you start using the utility, do so with the 'modern'.

In order to use the Multiple Map Utility, you select a database, a folder to create files and construct up to twenty map layer configurations. The parameters for a map layer must include the selection of one of twenty queries (that you build), whether the map layer is SHP or DXF type, and several settings that are applicable to that type of file.

Here are the types of files you can produce:



SHP points – options exist for whether to create selected points (line ends, every nth, etc.)



SHP lines along tracks – The user must enter a value that dictates where to break the lines. For example, if you enter 10, then when the difference between two consecutive stations (ordered by their station numbers) differs by more than 10, this represent a break in the line. Note that because of the numbering associated with some oblique grids and the 'gap' nature of brick patterns, you probably don't want to use this for those types of preplots.

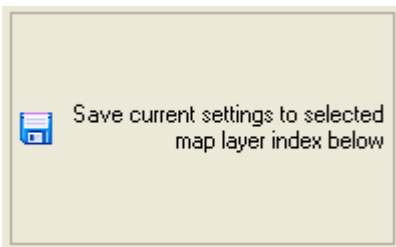
SHP lines connecting postplot and preplot – If this is selected, its assumed you selected a Postplot query that represents the points you want to connect to their Preplot counterpart. It also assumes that this is a standard GPSeismic database that contains the Preplot table and it is populated. If it doesn't detect all of these conditions, it does nothing.

DXF points – as with SHP, options exist for whether to create selected points. The user also specifies symbol and all associated parameters (size, layer and associated text).

DXF lines along tracks – This is similar to SHP but you have a number of line labeling options.

What the 'Token'? – To my knowledge, this feature is only used by a couple of people but just so you know how it works, here's the gist. If a query is constructed that has the string '<Token>' in it, and in configuring the map layer, you indicate that the token string is 'OilWell', then when the query is executed, the word 'OilWell' replaces the word '<Token>'. Why do this? Because if you want to configure a separate map layer and use a second string, say 'GasWell', you can. So in summary, it's a way you can use one query to produce several map layers. Confused? Don't be. I doubt you'll ever use it.

Creating queries – This is the standard query building dialog so hopefully you don't have much problem here. Note that one query might actually be used for more than one map layer. For example, suppose you isolated the source points with a query. This one query could be used for the point layer, the lines along tracks layer and the postplot/preplot connecting line query.



IMPORTANT – Press this after you have made any changes to a map layer configuration.



Are you feeling lucky punk? Well, are you? If you are new to this utility and want to jumpstart yourself, after selecting your database and folder, press this button. It will instantly configure the utility for eight map layers consisting of postplot and preplot points, along track lines, connecting lines and culture (as defined by alpha station names).

Daily use of the utility – Here's the deal. The first time you use this, you should create the preplot files, or perhaps you want to create the preplots using one of GSQL's query routines. However, on a daily basis, you re-create all of the postplot data, re-writing the file you created the day before. This is a 'one click' procedure once the utility is configured. Hopefully, your mapping application is like GPArC in the sense that you create your map by rendering particular layers in a certain manner and then save this to a file which contains the rendering settings. After you re-create all your layers with the utility, you simply re-display your map using your rendering settings. GPArC is blissfully unaware that there are now more points in the files.

Creating SHP and DXF Files From Selected Queries

SHP Point – When you select a query and create a SHP point files, your options on the field selection dialog include point selection (BOL, EOL,...) and whether you want all table fields to go into the underlying database. Selected points might be used if you wanted a map layer of points that were used for labeling exclusively. GPMAP would require this but GPArC has conditional labeling so you wouldn't need this.

SHP Line – There are three options for how to connect the points. One will create lines between each pair of points and allows you to enter a value which is used to break the line between two points whose station values differ by more than this amount. One option creates one line per track (requiring the correct number of bin digits be entered), and one option connects all the points in the query.

SHP Polygon – You have two choices here. Create one polygon with all the points in the query or one polygon per track. For the latter choice, your field procedures for surveying a number of lakes or archaeological sites for example would involve coming up with a numbering scheme like 1001, 1002, 1003,... and so on for one feature, then 2001, 2002, 2003,... for the next, and so on. You could then use 3 for the number of bin digits and GPSQL would create the individual polygons.

SHP Contours – This is a relatively new feature. You are first prompted with the name of the SHP file to create. On the field selection dialog, you need to specify easting, northing and z-field. The latter is typically the height, but remember that you can use any field to represent 'z'. Some interesting plots can be made using precision, dop, etc. On the field selection dialog, you also enter the contour interval. You can leave this at 0 for automatic.

The second item entered is the number of grid nodes. This is an important value and requires some explanation. Behind the scenes, the routine must first create a digital elevation model (DEM) from the points from the query. The evenly spaced DEM is then used to create the contours. The best case scenario is that the points in your query exhibit good spatial separation, that is, the entire area to be contoured is neither narrow (in any direction) or bunched up (e.g., many points in one area and devoid of points in another). The grid node value is used to evenly divide up the spatial extents of your data. For example with a value of 100, the 'behind the scenes' DEM consists of 10,000 grid nodes (100 x 100). From experience, the number of grid nodes should be fairly high (say 100 or more). The maximum is 2000 and will provide the highest resolution set of contours (at the expense of time to create).

DXF files – You can create DXF point or line files. For line files, you have a number of selections for how to label the lines.

DBF Field Name Compatible MDBs – SHP files are actually three files in one. There is the 'SHP' file which contains the actual coordinates of the features, the 'SHX' file which contains information that allows speedy rendering of the features, and the 'DBF' file which is actually a database of the DBASE format that contains information for each point. What information is pretty much up to the application creating the file. Regardless of what information is contained in the DBF, there are certain restrictions pertaining to the DBF field names. There's a limitation in the size of the name and what characters are contained in the name.

You will notice whenever you make a SHP file in GPSQL, the DBF field names are truncated and modified to meet the specifications of the DBF format. Not too long ago, a client who was using a mapping application that used the MDB database directly complained that the application could not use the MDB because of the field names. That was why a utility was added to create an MDB database from the selected query, but one in which the field names were compatible with any DBF created from it.

Selected point Annotations – This obscure routine will use the points selection options on the field selection dialog to place specific strings in a specified field. It will place the string 'BOL' at the first one (or two) points of a track, 'EOL' at the last one (or two), and the word, 'BOX' for all of the others. Why do this? Because some mapping applications (like GPArC) can conditionally label points based on the contents of certain fields of the underlying database. Therefore, this gives you a good way of labeling a point selection that would otherwise be impossible to label.

Vibe Data Processing Tools – GSQL and QuikMap have several tools that allow you to deal with vibe data.

Import – There are import routines for the most common vibe files including Pelton, Sercel and TigerNav. The tables resulting from the import are very similar to the Postplot table through the first dozen fields or so. You will also notice that if the file type supports it, there will be a vibe ID field and a sweeps field. There is provision for transforming coordinates on input and applying a geoid height correction in order to obtain local heights.

Caution – I personally think there is a conspiracy to include header records in some of these files that are difficult if not impossible to recognize. So try removing header records before importing.

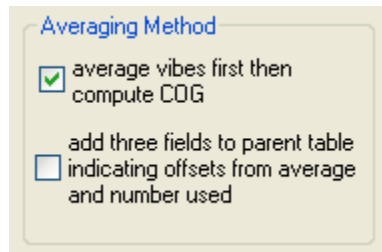
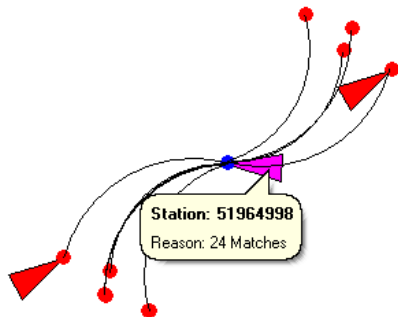


Table averaging – Once data is imported and you have a query to select your vibe data, you can select 'Append/Create Table Of Averaged Data'. This averages all points with the same station name. However, there are two ways to come up with the average. One is lump all coordinates together to come up with the average, and the other is to come up with averages for each individual vibe, and then average those averages. This latter method is the most logical method.

On the field selection dialog, there is also an option for whether to add some fields to the parent table indicating the distance of each position to the resulting average. This information is good for spotting problems since a large value would normally indicate one or more positions were outliers. Regardless of what you choose to do here, you will still get similar information for each record in the table of averaged data.



QuikMap and the Multiple Comparison Layer – After the parent table and the averaged table exist, one of the most graphic things you can do is send to queries to QuikMap (in this case our parent table and our averaged table), and select 'File/Compare Offsets' from the File menu in QuikMap. Once you do, it will be fairly obvious where there are problems. Also, there will be a File/Compare sub-menu item called 'Multiple' that will allow you to display some diagnostic pointers to make it clear where certain groups have missing or additional positions and high COG statistics.

Duplicate manager – The Duplicate Manager is a useful tool for identifying trouble spots in the parent table vibe data. Its set up to use sweeps information and can summarize the time duration of the sweeps. Its possible to edit data in the Duplicate Manager including deleting records.

Seismic file (Vibe Positions) – This is a relatively new feature that does not necessarily analyze imported vibe data, but rather creates SEG and other type of ASCII coordinate files. Specifically, you specify a query of positions and a template (TPL) file. The TPL file is a small file which defines points relative to a center point and includes a reference azimuth. What the routine does is uses the offset information of the TPL file to create a group of points for each point in the query. Possible uses include generating preplots for all individual vibe or creating a file to compare to the vibe positions as recorded.

Query Actions You Should Know About

Production utilities – These are handy to determine production both in a tabular and graphic manner. The 'Production Statistics' utility has been around for a while and was formerly called the 'Mileage Calculator'. It gives you a BOL/EOL summary for the selected query.

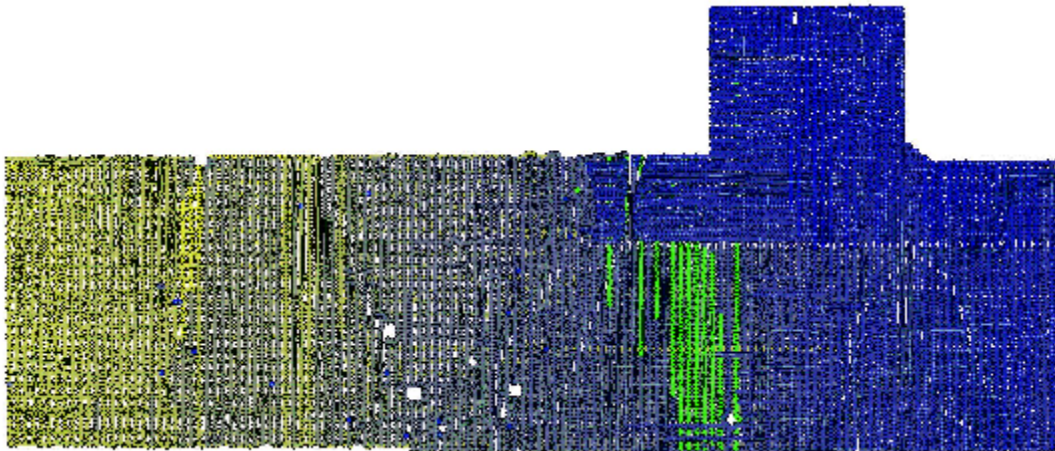
| Wednesday | Thursday |
|--------------------|------------------|
| 7 | 8 |
| 156/72Pts/4.3km | 152/36Pts/2.2km |
| Total - 72Pts/4.3k | 156/16Pts/1.0km |
| | 160/38Pts/2.3km |
| | 164/132Pts/7.9km |

The 'Production Calendar' requires that you specify the start and stop dates for the selected query and a station interval (and of course the bin digits). There is a auto-select button that analyzes the query to determine the start and stop dates for you. The resulting calendar can have total linear distance or line by line totals. If you have too many distinct lines, I'd opt for the totals only. Right click on the resulting calendar

and you can display daily, weekly or monthly views.

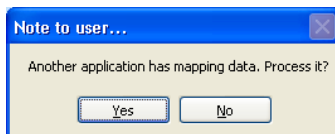
| December 2005 | | | | | | | January 2006 | | | | | | |
|---------------|----|----|----|----|----|----|--------------|----|----|----|----|----|----|
| S | M | T | W | T | F | S | S | M | T | W | T | F | S |
| 27 | 28 | 29 | 30 | 1 | 2 | 3 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 4 | 5 | 6 | 7 | 8 | 9 | 10 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |
| 25 | 26 | 27 | 28 | 29 | 30 | 31 | 29 | 30 | 31 | | | | |

The 'Production QuikMap/SHP' utility will require that you specify the colors you want particular days displayed in. Then you can create a QuikMap plot of your prospect colorized to match your selection. This is a great way to tell what was shot on what day(s). In the example here, we have used ramp colors but specified that one particular day was to be plotted in green. Below is the resulting QuikMap plot:



So the next time the client asks where the pack ops were working on a particular day, you can show them. This utility can also create a SHP file of polygons where the underlying database contains day of year, day of month, and day of week. In this manner the polygons can be rendered to the users tastes.

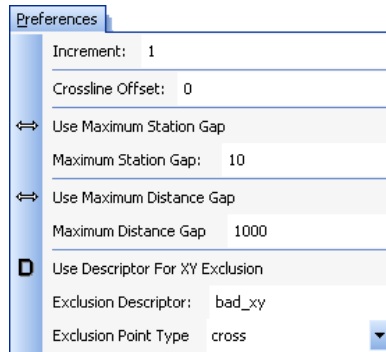
GPSQL/QuikMap And The Mapping Table – If you need to isolate records in the database that fall in distinctly different geographical areas, it can be hard to do (or impossible) using a single table query. The Mapping table is a special table that can be created by first sending a query to QuikMap.



Once in QuikMap, you would lasso a desired subset of points and when you display the popup menu, select the 'Graphical Query' item. GPSQL will report that there is data to process. Say YES and a table called 'Mapping' is created. The table contains two fields (Station (value) and Station (text) which will initially contain all the

stations you lassoed. Repeat the procedure and you will be allowed to append to the table. The significance of this is that you now have a means of performing a table join to isolate only the records you want. Also note that there is helper function in the Define menu to assist you in automatically creating this table join (not that you need it).

Interpolation utilities –



Interpolate – The standard interpolation routine is not new. You select a query, configure the field selection dialog and then are presented a dialog which shows a plan view and height profile of the data. Your query is normally one that isolates one specific track or line. The key items for interpolation are in the Preferences menu and the most important item is the increment. Other items are seldom used. Note that all interpolated points have a descriptor with the word ‘interpolated’ in it.

When you create interpolated points using the Postplot table, very few of the fields actually get populated. More often than not only the station, grid coordinates and height fields are filled. However there is utility for filling those fields described below.

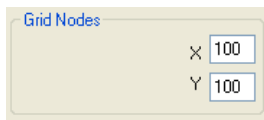


Interpolate All – This bad boy interpolates all points within a query based on a user supplied increment. The ‘station gap’ dictates where not to interpolate points. For example, with a value of 100, if the difference between two consecutive station differ by more than this amount, no points are interpolated between the two. This utility might be preferable to the one above. Its depends if you need the sophistication of the interpolation dialog and all it brings to the table. Note that all interpolated points have a descriptor with the word ‘interpolated’ in it.

Interpolate Heights – As the name implies, this utility allows you to interpolate heights. Its done graphically. You want to note that on the field selection dialog, the required fields are station, height and descriptor and the latter field is used to write the string ‘interpolated’ into.

Fill Postplot Table – This utility can be used to fill most of the blank fields in interpolated records. You really need to make sure your current coordinate system, geoid model and number of bin digits are specified correctly.

DEM Utilities – There's two basic things you can do: 1) Create a DEM and 2) use a DEM to create an additional field with what the DEM says the height is.



Create A DEM – You select a query and based on the easting, northing and height, a DEM is created. You do need to enter the number of grid nodes the DEM is to contain. The DEM could be pretty good if the points have good spatial extent. If they don't the DEM is still created but DEM heights in areas where there were no points are basically interpolated (if points exist to either side of the void). Note that there is no extrapolation.

Compute DEM Heights – Assuming you had a DEM, you can send the grid coordinates of a query to it and have a field (DEM Heights) be populated with what the DEM says the height should be for each point. If the DEM is in a different coordinate system from the database system, you need to specify the coordinate system of the DEM on the Miscellaneous dialog. Otherwise, make sure the check box on the Miscellaneous dialog that says 'systems are identical' is checked.

The 'DEM Height' field will be populated with either the DEM height or a value of -9999 depending on whether the point fell inside or outside the DEM. This mechanism can be used to your advantage if you have a number of DEMs that cover your prospect. Send all the points to the first DEM and then write a query that isolates all records where the 'DEM Height = -9999'. Using this query, keep running the same routine with different DEMs until there are no records in the query (which means every point was assigned a DEM height).



Great Googly Moogly – We already saw where you can enter some geographic coordinates into a textbox located in the status bar and launch Google Earth (if installed) or your browser with Google Maps. You can also select a query and create points or lines in Google Earth.



Tip - If you have many points, use lines rather than points.

The behind the scenes stuff going on involves creating a file with an extension of 'KML' and starting Google Earth with this file as an argument. The same thing could be done by finding the KML file and double clicking on it. Because Windows associates this extension type with Google Earth, it's launched and immediately displays what it finds in the file. The contents of the file should be latitudes and longitudes and be in the WGS84 datum because this is the reference system for all of Google Earth's (and Google Map's) coordinates.



Tip - If you want to find the KML file that GPSQL creates, look in the Google Earth folder immediately after running the routine. You will find a file called 'GPSQL.KML' there. If you need to, you could email this to your boss or client to show where the prospect is. Another option is to simply use the new Utility item to create a KML file from the query.



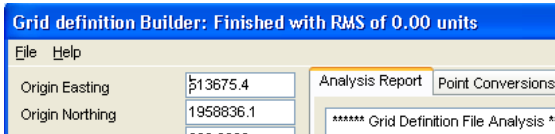
Tip – As I said above, Google Maps is actually based on latitude and longitude. If you get your map centered at a place of interest, you can find the latitude and longitude for that point by typing in the following in the URL textbox:

```
javascript:void(prompt("",gApplication.getMap().getCenter()));
```

And Now The Rest Of The Story – As I look over the query actions and modifications GPSQL is capable of, I can't help but list a number of items that we should cover, if only briefly.

Creation Of Grid Definitions Files – Grid definition files can be used for many operations including re-binning and preplot design. Both GPSQL and QuikMap have an automatic grid definition builder but its not always going to give you the right answer. However, there are tests for whether the grid definition file is right and with a few tweaks, you can make the grid definition file perform flawlessly.

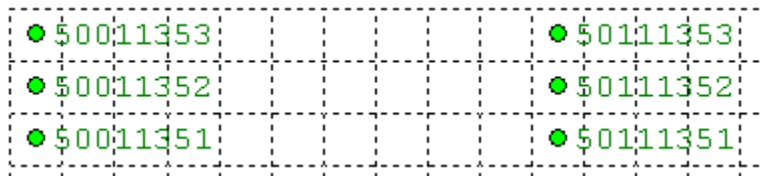
One key point if you select a query for use with the automatic grid definition builder is that you better be selecting preplots and just the source or receiver. Also, if any part of the preplots have been offset for culture, you want to avoid them since they will cause the parameters to be bogus (or not computed at all).



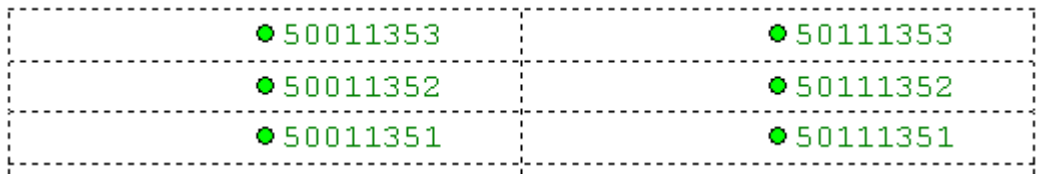
One key point when the utility is finished is to look at the RMS value. The last thing the utility does after it has computed the grid definitions is to take a dozen random stations, and based on the station value, computes the station's

coordinates. It compares this with the actual coordinates and then provides you with an average of the differences. A small number means the grid definition parameters appear to be working.

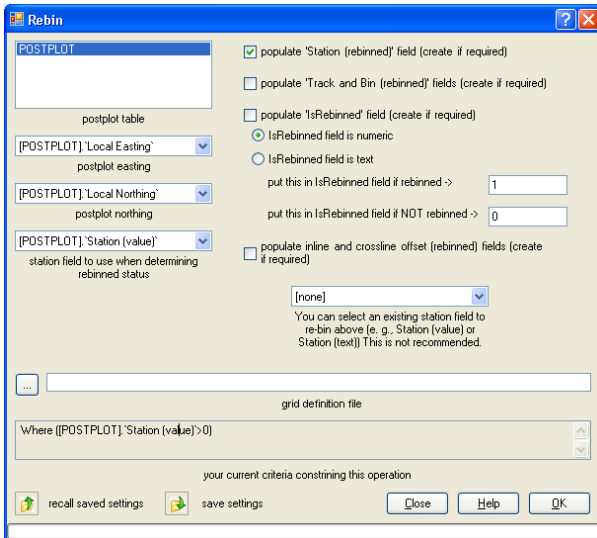
The words 'appear to be working' isn't a mistake. It is possible that the grid definition file created correct parameters for a grid unlike the one you wanted. What I mean by this is that the utility determined the number of bin digits incorrectly. Consider the grid definition below:



This looks fine if we are looking at lines 5001 and 5011 (stations 1351 to 1353). However, the is another grid that entirely valid:



If we are looking at lines 500 and 501 (stations 11351 to 11353), then the above grid is fine. So bottom line, if you want to get a valid grid definition file, create it, then look closely at the parameters and make sure the bin digits are what they are supposed to be. Also make sure you look at it visually in QuikMap. Its easy to see if its correct or not.



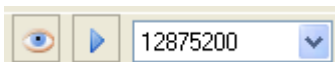
Use Of Grid Definition Files In Re-Binning – First, 're-binning' in GPSQL means coming up with the name of the station based on where it is located in the theoretical grid. GPSQL will minimally add a field called 'Station (rebinned)' and place the station value there based on the station's coordinates.



Tip – Never, and I mean never, instruct your pack ops to re-number stations that they offset. Why? Well, they might get the station number wrong. Also, since there is no preplot for a re-numbered station, your offset information goes bye-bye. It is very confusing to have a number of manufactured surveyed stations to deal with. Trust me. I've been

contracted out to go to projects for the sole reason of trying to resolve problems that are caused by this.

Ok, where were we? Right...re-binning. You want to note the available options on the dialog. You can add other fields including a indicator for whether a station was re-binned and yet others that are offsets relative to the actual position. Don't get too uptight about doing this either. Remember that you can always delete the fields you create. Also, note carefully that with the original station number and the re-binned one, you are in a perfect position to create a SEG or other file with whatever station the client asks for.



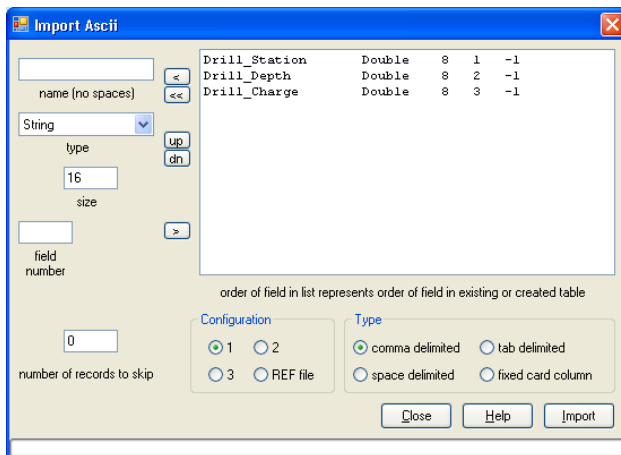
Handling Duplicates – There's been quite a few requests for this so lets take a quick look. The first way of spotting duplicates is through the spreadsheets. Once you click on a column head of any field, the query is sorted by that field and you can press on an 'eye' tool button which will scan that field for duplicates and fill a list box with the duplicates it finds. Press the arrow next to a station you select and the spreadsheet will be scrolled to the point. What you do at this point is up to you. My preference would be not to delete one of the records, but rather, change it in some way so I don't consider it to be a valid record. Perhaps, I would change the 'Station (value)' to zero and place 'BAD' in front of the 'Station (text)' name.

The 'Duplicates Manager' can be used. We have seen that this utility lists all duplicates and does allow for deletion of points. Again, however, my preference would be to change the record, not delete it.

The 'Purging Duplicates' utility is a specialized utility that is really more suited for exact duplicates, i. e., ones that might be the result of processing the same data collector file twice. The utility relies on three user fields for determining what duplicates are. The default are latitude, longitude and time, the theory being that you can't be in the same place at the same time without it being an exact duplicate.

Importing Data – Its important to look at this topic because one real powerful tool when it comes to managing data is the table join. Consider getting a file of points that were drilled. Import that and do a table join and you now have the ability to give a client a coordinate file of everything drilled or producing a map of the same. The question is, how do I get the file in. You first have to ask yourself what format the file is in. Chances are its an ASCII text file but it could be an Excel XLS file. In either case, you are in good shape. For example purposes, let's assume we have information that includes the station drilled, the depth at which it was drilled and the charge amount. Let's first assume its ASCII and looks like this:

50911226,34,6
 50911227,33,6
 50911228,34,6
 50911229,34,7



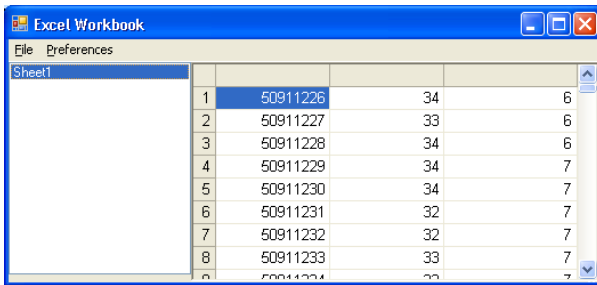
Here we would use the 'ASCII File' item in the Import menu. When all is said and done, the dialog would look like it does here. You might use different names for the fields but chances are that the rest would be the same. Notice that I made all fields doubles. For the station, its not necessary but if you do make it a string field and hope to join it to the Postplot or Preplot tables on this field, the field size has to match. Also, if there is as much as a space in one station that is not in the other, you won't get a match. So its much better and more reliable to stay with numbers, and in this case, doubles, since

the Postplot and Preplot 'Station (value)' field is a double.

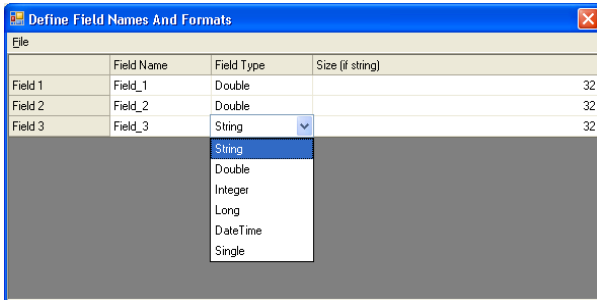
Of note for the ASCII Import is that the order you place the fields in the list box is the order that the fields will be created in the new table. Also, each delimited field has to have its delimited place in the record specified and delimited fields don't have to be sequential. In other words, even if the station was the 100th delimited field in a record, it wouldn't matter. We simply would indicate this when we add a field to the list.

Finally note that if we get a new file from the driller, if it is to be appended, just specify the table to append to. If the data is a superset, then its best just to delete the previous table and re-create it.

Now, you might get your data in the form of an Excel XLS file. This is usually no problem because there is an Excel import mechanism. You can also opt to make a comma delimited file from Excel and import it as above. If you do import directly from Excel, here's a brief explanation of how its done.



After being asked for the table to append to (or create), you will get a special dialog from which you can open an XLS file. You would then click on the desired sheet and the dialog will now look like what you see here.



At this point, you would choose 'Define Field Names And Format' from the Preferences menu and do two things: 1) Give each field a name and 2) select what the data types are. Once you do this, from the main dialog you would elect to 'Place records in the import cache'. The reason for this is that there may be more than one sheet in the XLS file and you will have the opportunity to

place the records from the various sheets into the cache. Once done, select 'Finish Importing' from the File menu and the import will be completed.

Importing And Using Swath Definition Tables – This is a specialized import feature and its purpose is to allow the user a way to isolate swaths when swaths are defined by low and high track values and low and high bin values. You start by developing a comma delimited file of:

swath number, low track, high track, low bin, high bin

Below could be a typical file which would be imported:

```
1,0,5000,1000,2000
2,0,5000,2000,3000
3,0,5000,3000,4000
4,0,5000,4000,5000
5,0,5000,5000,6000
6,0,5000,6000,7000
7,0,5000,7000,8000
8,0,5000,8000,9000
9,0,5000,9000,10000
```

In the above example, there are 9 swath as identified by the numbers 1-9. The first swath is defined by a low track of zero, a high track of 5000, a low bin of 1000 and a high bin of 2000. The remaining swaths are identified similarly.

Once imported, a table is created. For the example below, make sure you call the table 'Swaths'. If the above data was imported, you would have five fields and nine records. The field names are called:

Swath
LoTrack1
HiTrack1
LoBin1
HiBin1

The query builder is designed so that it is possible to drag and drop table fields into the specified value textbox. This allows 'BETWEEN' statements to be constructed based on multiple table fields. For example, if two tables are selected and called SWATHS and POSTPLOT, you could build a criteria like:

```
Select [POSTPLOT].*,[Swaths].* From [POSTPLOT],[Swaths] Where [POSTPLOT].`Track` Between [Swaths].`LoTrack1` And [Swaths].`HiTrack1` And [POSTPLOT].`Bin` Between [Swaths].`LoBin1` And [Swaths].`HiBin1`
```

When you build such a query, you are cautioned to add a join to the table, but you don't have to. Trust me!

And here is the same query for the Preplot table:

```
Select [PREPLOT].*,[Swaths].* From [PREPLOT],[Swaths] Where [PREPLOT].`Track` Between [Swaths].`LoTrack1` And [Swaths].`HiTrack1` And [PREPLOT].`Bin` Between [Swaths].`LoBin1` And [Swaths].`HiBin1`
```

And finally, here is how you can isolate only one swath:

```
Select [PREPLOT].*,[Swaths].* From [PREPLOT],[Swaths] Where [PREPLOT].`Track` Between [Swaths].`LoTrack1` And [Swaths].`HiTrack1` And [PREPLOT].`Bin` Between [Swaths].`LoBin1` And [Swaths].`HiBin1` And [Swaths].`Swath` = 4
```

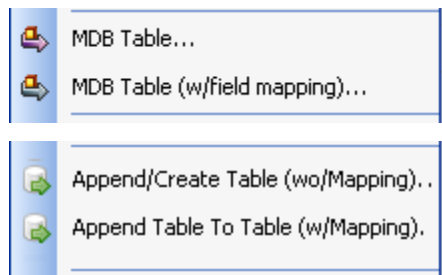
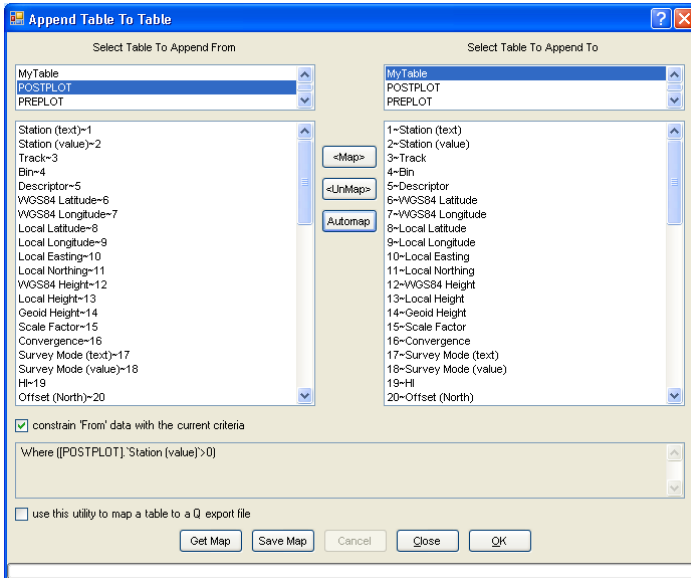


Table To Table Appends – Here is a topic that pertains to importing of data from other databases and moving data from one table to another table in the same database. If you are importing data and choose 'MDB Table...' you will be prompted for the database to import from, the table in that database to append from, and the table in the current table to append to. The field structures between the two tables must be an exact match. It doesn't matter what the names are, but the field types have to agree in

type (an in the case of string fields, length) and order. The same holds true for table to table appends within the same database. In this case, you would choose 'Append/Create Table (wo/Mapping)' from the Modifications menu and again, the field types and order must match exactly between the two.

Now, what do you do when tables don't match. This is probably more common than when they do. Fortunately, there is a mechanism for both importing tables and appending table data within the same database, even if they don't match. We will look at appending from one table to another within the same database although the technique can be used in an almost identical fashion when importing.



At left is the dialog displayed when you append data with mapping. You select a table on the left which is the 'from' table and select a table on the right which is the 'to' table. The next step is to select a field on the left and right and press the 'Map' button. This places a number at the end of the 'from' field. This number indicates what field in the list on the right that the data will be placed on.

The 'auto-map' button will simply look at the field names and automatically append the field number based on a name match.

This might be right or wrong. The user should always check this. An 'UnMap' button removes the field assignment number.



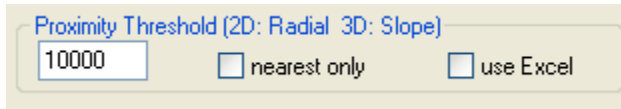
The table-to-table append with mapping has been enhanced in Version 2007.2 to allow the user to append from a table join query.

- calculate offsets based on selected grid definition file and selected station field
- calculate offsets based on selected preplot fields and entered azimuth
- calculate offsets based on selected preplot fields and azimuth from grid definition file

Re-Calculating Offsets – Only a quick mention about re-calculating offsets. First why would you have to do this? I can think of two reasons. The first is that when you processed a data collector file in QuikView, there were no preplots and thus no offsets. The other is that you had the wrong reference azimuth in QuikView so the offsets might be present but are incorrect.

There are three methods to compute offsets. If you base the new offsets on a grid definition file, you select a grid definition file and from this GPSQL computes the theoretical preplot location for each station based on the station value. This theoretical preplot and the actual survey coordinates are used to come up with the offsets. The other two methods require that you specify the preplot table from where a match to the postplot can be found. The two methods require a reference azimuth. One option requires you enter this and the other will obtain it from a grid definition file you specify.

Proximity Utilities – These utilities have been around for a while. In general, they are used to specify either one or two queries so that you can find points that are within some user entered distance of another.



Proximity Threshold (2D: Radial 3D: Slope)
10000 nearest only use Excel

I'm only mentioning these utilities because in the case of the two-query proximity tests, an option was recently added to find the nearest single point. This could be

useful in a situation where you have a query that defines hazard points and you need to find the one nearest survey point to each. In this example, what you would do is to run the 'two query' proximity routine (2D or 3D) and select the hazard points for the first query and survey points for the second. On the small dialog that comes up first that allows you to select the two queries, the first is on the left, the second is on the right.

When you get to the field selection dialog, you want to enter a large threshold value and check the 'nearest only' check box which is a new option on this dialog. Note that if there is no point within the threshold then no record is written. This means that you want a fairly large threshold if this option is utilized.

One final note. When the first field selection dialog comes up, you might want to replace 'Station (value)' with Station (text) if your hazard points have a distinctive alphanumeric name but values of zero. It will aid in the identification of the hazard.