A Strategy for Using BLOB Functions on Large Databases

Introduction

Alan Manifold's BLOB functions (GetBibBlob, GetFieldRaw, GetSubField, etc.) are priceless for people who write queries against a Voyager database, because they often provide the only way to get at data in certain MARC fields. But if you have a large database and if you don't use the BLOB functions wisely, you can easily create reports that run for hours. This document will describe a strategy for using the BLOB functions in a way that makes them practical for use with large databases.

Is my database "large", you ask? Well, if you have more than 800,000 bibs or so, your database is certainly large. If you are frustrated by how slow your queries that use the BLOB functions are, your database is large. Actually, the strategy described in this document works fine for databases of any size. But, if your database is not large, it may not be worth your while to take the extra steps that the strategy requires.

Libraries with databases of any size might want to look at the document, "Alternatives to the BLOB Queries," at http://office.ilcso.illinois.edu/secure/sql/resouces/alttoblob.pdf

A word about the conventions in this document: Each time a query is given as an example, it will be shown in 2 ways. The design view is provided because that is the easiest way to see what is going on. The SQL will also be provided so that you can copy the query into your own reports file and play with it. So breath easy! No one expects you to read the SQL!

The Problem with BLOB Functions in Large Databases

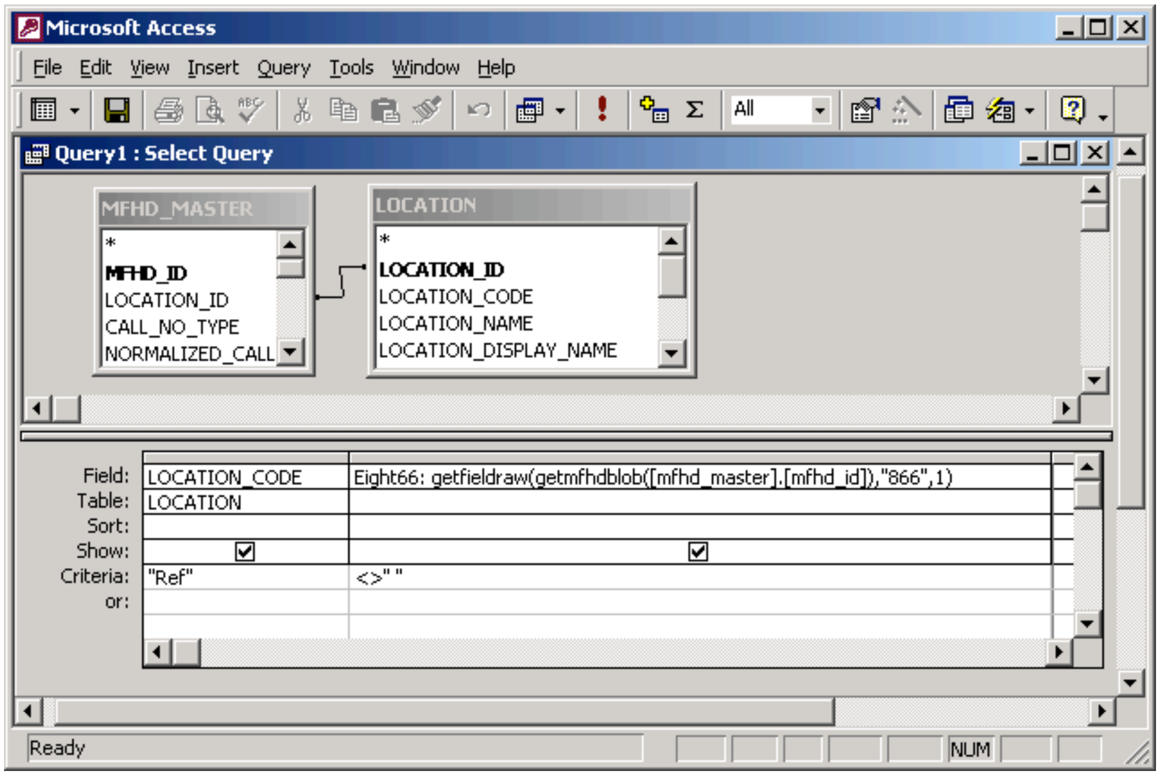
The reason the BLOB functions are slow is this: They are implemented in Access on your workstation. If you use a BLOB function on, say, all the bib records in your database, then all of the bib records have to be brought across the network to your workstation, and then parsed out by your workstation. Today's workstations are powerful computing machines, but parsing MARC records is an intensive task.

So there are 2 principles behind this strategy:

1) Use the BLOB functions on the smallest set of records that will still let your query accomplish what you need to do.

2) Don't put criteria on a column that is created by BLOB functions.

The first of these principles makes sense on the face of it. The second is more subtle and best illustrated by an example. Suppose you want to find all of the MFHDs in your reference collection that have a note in the 866 field. You might build this query:



Field:	LOCATION_CODE	Eight66: getfieldraw(getmfhdblob([mfhd_master].[mfhd_id]),"866",1)
Table:	LOCATION	
Sort:		
Show:	✓	✓
Criteria:	"Ref"	<>" "
or:		

```
SELECT LOCATION.LOCATION_CODE,
getfieldraw(getmfhdblob([mfhd_master].[mfhd_id]),"866",1) AS Eight66
FROM MFHD_MASTER INNER JOIN LOCATION
ON MFHD_MASTER.LOCATION_ID = LOCATION.LOCATION_ID
WHERE (((LOCATION.LOCATION_CODE)="Ref") AND
((getfieldraw(getmfhdblob([mfhd_master].[mfhd_id]),"866",1))<>" "));
```

You might think that Access will realize that it only has to parse the MFHDs from your reference collection. Not so! Access is going to get all of the MFHDs in your database, bring them to your workstation, and parse them all there to see if they meet the criterion on the 866 field.

The Strategy

The way to get around this problem is to break your query into 3 separate queries, thus:

Query 1 will create the shortest possible list of BIB_IDs, MFHD_IDs, or AUTH_IDs on which the BLOB functions will be used. It will be a Make Table query, so the list of ID numbers will be written into a table in the reports.mdb file on your workstation. It is a good idea to set the Unique Values property for this query.

Query 2 will use the list of ID numbers generated by Query 1 and do the BLOB functions on just those records. There should be no criteria in Query 2. This will be a Make Table query, so the parsed MARC fields or subfields will be written into a table in the reports.mdb file on your workstation.

Query 3 will pull together the information from Query 2 and whatever other fields you need. You can include criteria on any field, including the fields that came from Query 2. You can link to other tables to get any additional fields that you need.

A quick refresher: To change a query into a Make Table query, click the drop down arrow just to the left of the red exclamation mark on the tool bar, select Make Table, and Access will ask you what you want to name the table. To set the Unique Values property on Query 1, right-click on the gray area surrounding the tables in the design pane and set Unique Values (not Unique Records, that's different) to Yes. If you need more clues than this the paragraph provides, try looking up Make Table Query or Unique Values in the online Access help or in your favorite Access book.

In the examples below, the table created by Query 1 will be called Table 1, and the table created by Query 2 will be called Table 2. This is just to make the examples easier to read. You'll probably choose more meaningful names.

When your Query 3 is done, you might want to delete Table 1 and Table 2 from your reports.mdb so it doesn't take up room on your hard drive.

Some Thoughts on Query 1

There's some art to figuring out how to write Query 1 so you get a nice, short list of ID numbers (BIB_IDs, MFH_IDs, or AUTH_IDs). Sometimes you can do it by writing a query against the table that you already know and love. Example A below is a good example of this. Sometimes you need to search for a string in an unparsed MARC record. Examples B and C below show this. If you're having difficulty figuring out what to do with Query 1, take a cataloger to lunch. You'll be able to figure it out together. Or send a note to CARLI On Call asking for ideas.

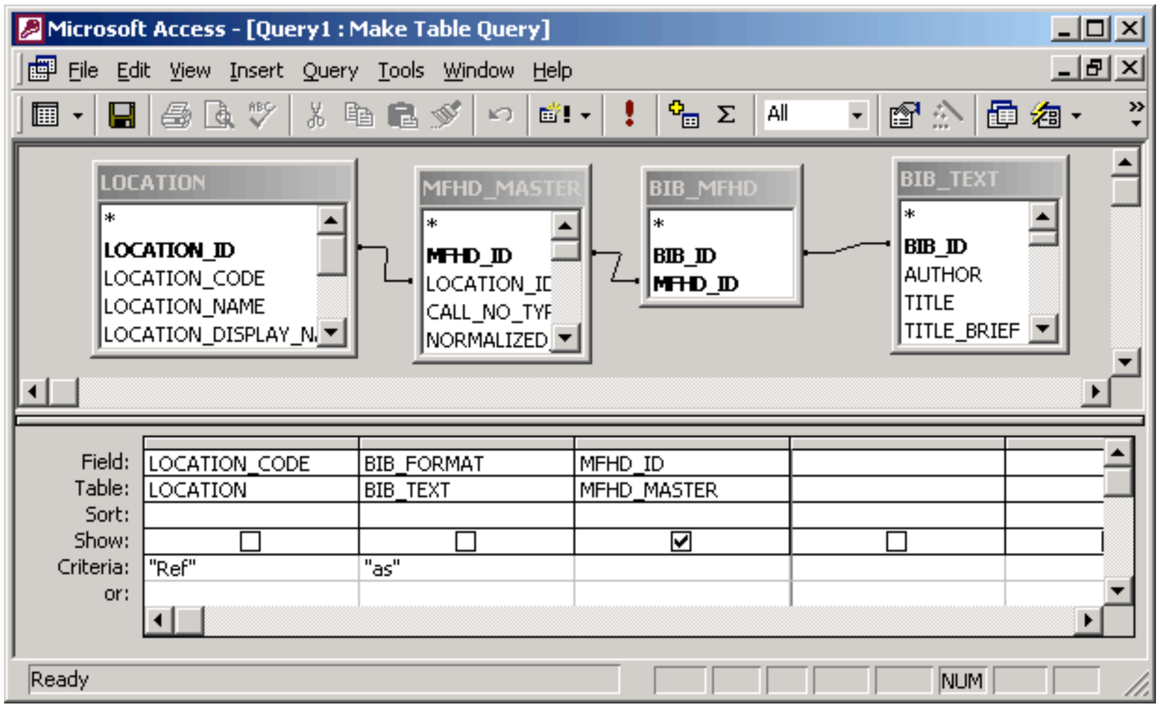
The rest of this document is three examples of BLOB queries that use the technique we've been discussing.

Example A

This is a lot like the example that already looked at on pages 1 and 2 of this document. I want to find the serials in the reference collection that have a note in the MFHD 866 field.

Example A – Query 1

I can identify serials because the BIB_FORMAT field in the BIB_TEXT table is set to "as". In Query 2, I'm going to have to use the BLOB functions on the MFHDs to get the 866 field. So here's a query that will list the MFHD_IDs of the serials in the reference location. It's hard to see in design view, but I remembered to set the Unique Values property and to make this a Make Table query. The table I'm making is called Table 1.



Field:	LOCATION_CODE	BIB_FORMAT	MFHD_ID	
Table:	LOCATION	BIB_TEXT	MFHD_MASTER	
Sort:				
Show:	☐	☐	☑	☐
Criteria:	"Ref"	"as"		
or:				

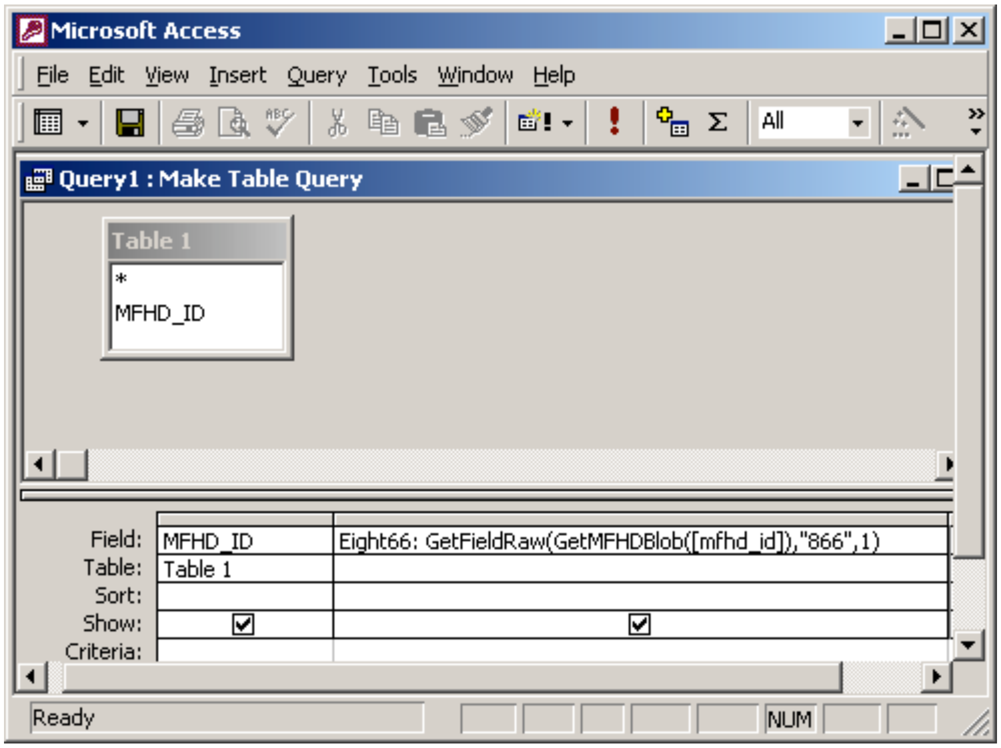
```
SELECT DISTINCT MFHD_MASTER.MFHD_ID INTO [Table 1]
FROM ((LOCATION INNER JOIN MFHD_MASTER
ON LOCATION.LOCATION_ID = MFHD_MASTER.LOCATION_ID)
INNER JOIN BIB_MFHD ON MFHD_MASTER.MFHD_ID = BIB_MFHD.MFHD_ID)
INNER JOIN BIB_TEXT ON BIB_MFHD.BIB_ID = BIB_TEXT.BIB_ID
WHERE (((LOCATION.LOCATION_CODE)="Ref")
AND ((BIB_TEXT.BIB_FORMAT)="as"));
```

When I run this query, if I've already got a Table 1, Access will ask if it's OK to delete the old Table 1 so a new one can be made.

When the query is done, Access will ask if it's OK to add some number of rows to Table 1. Say OK.

Example A – Query 2

All I need to do in this query is to use the BLOB functions to get the 866 field (if there is one) from the MFHDs whose MFHD_IDs are now in Table 1. There might be more than one 866 field in a MFHD, but I think I'll just ask for the first one. And I think I want to see the indicators and the subfield coding, so I'm going to use GetFieldRaw. I've remembered not to use any criteria in this query and to Make a Table called Table 2. Notice that I've kept the MFHD_ID in a column. I'll need it in a moment.


Microsoft Access
File Edit View Insert Query Tools Window Help
All
Query1 : Make Table Query
Table 1
*
MFHD_ID
Ready
NUM

Field:	MFHD_ID	Eight66: GetFieldRaw(GetMFHDBlob([mfhd_id]),"866",1)
Table:	Table 1	
Sort:		
Show:	✓	✓
Criteria:		

```
SELECT [Table 1].MFHD_ID,
GetFieldRaw(GetMFHDBlob([mfhd_id]),"866",1) AS Eight66
INTO [Table 2]
FROM [Table 1];
```

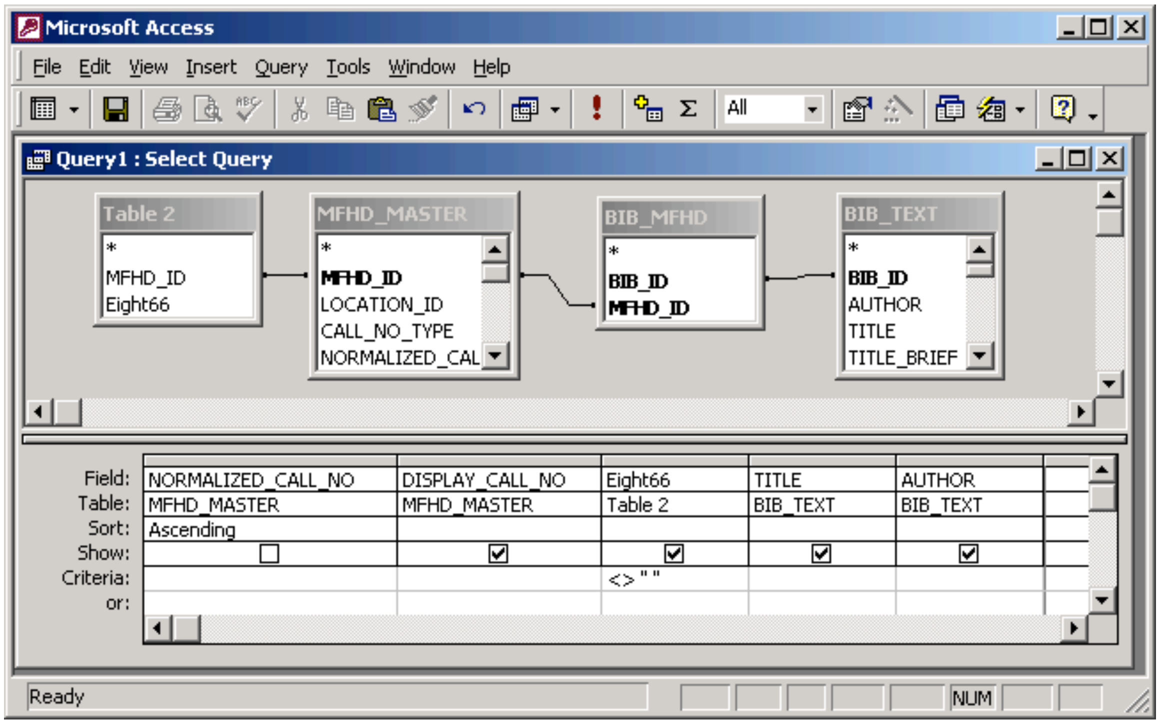
When I run this query, if I've already got a Table 2, Access will ask if it's OK to delete the old Table 2 so a new one can be made.

When the query is done, Access will ask if it's OK to add some number of rows to Table 2. Say OK.

Example A – Query 3

Now I can start from Table 2, add any criteria that I wish, and link up the other tables that I need. I think I'm going to sort the results by call number and include the title and author.

Notice that I'm using the MFHD_MASTER, BIB_MFHD, and BIB_TEXT tables, which I used in Query 1 as well. I could have kept the call number, title, and author back in Query 1 and copied them into Table 2. I didn't do that because these fields are kind of long. I'd have copies of them in Table 1 and Table 2, so my reports.mdb would be larger. I decided to be frugal of my hard drive.



Field:	NORMALIZED_CALL_NO	DISPLAY_CALL_NO	Eight66	TITLE	AUTHOR
Table:	MFHD_MASTER	MFHD_MASTER	Table 2	BIB_TEXT	BIB_TEXT
Sort:	Ascending				
Show:	☐	☑	☑	☑	☑
Criteria:			<> " "		
or:					

```
SELECT MFHD_MASTER.DISPLAY_CALL_NO,
[Table 2].Eight66, BIB_TEXT.TITLE, BIB_TEXT.AUTHOR
FROM (([Table 2] INNER JOIN MFHD_MASTER
ON [Table 2].MFHD_ID = MFHD_MASTER.MFHD_ID) INNER JOIN BIB_MFHD
ON MFHD_MASTER.MFHD_ID = BIB_MFHD.MFHD_ID) INNER JOIN BIB_TEXT
ON BIB_MFHD.BIB_ID = BIB_TEXT.BIB_ID
WHERE ((([Table 2].Eight66)<>" "))
ORDER BY MFHD_MASTER.NORMALIZED_CALL_NO;
```

Example B

I want to find any bib records that have the obsolete MARC 582 field so I can move the data to the proper MARC field. The 582 is not indexed, so I can't use the BIB_INDEX table to find it.

Example B – Query 1

For the first query, I'm going to try to locate the 582 tag in the record directory of the MARC bib. To do this, I need to recall the structure of a MARC record. The record directory appears near the beginning of each MARC record. It has an entry for each variable field in the record. The entries consist of the 3-digit MARC tag, a 4-digit length of the variable field, and a 5-digit offset that locates the variable field in the record.

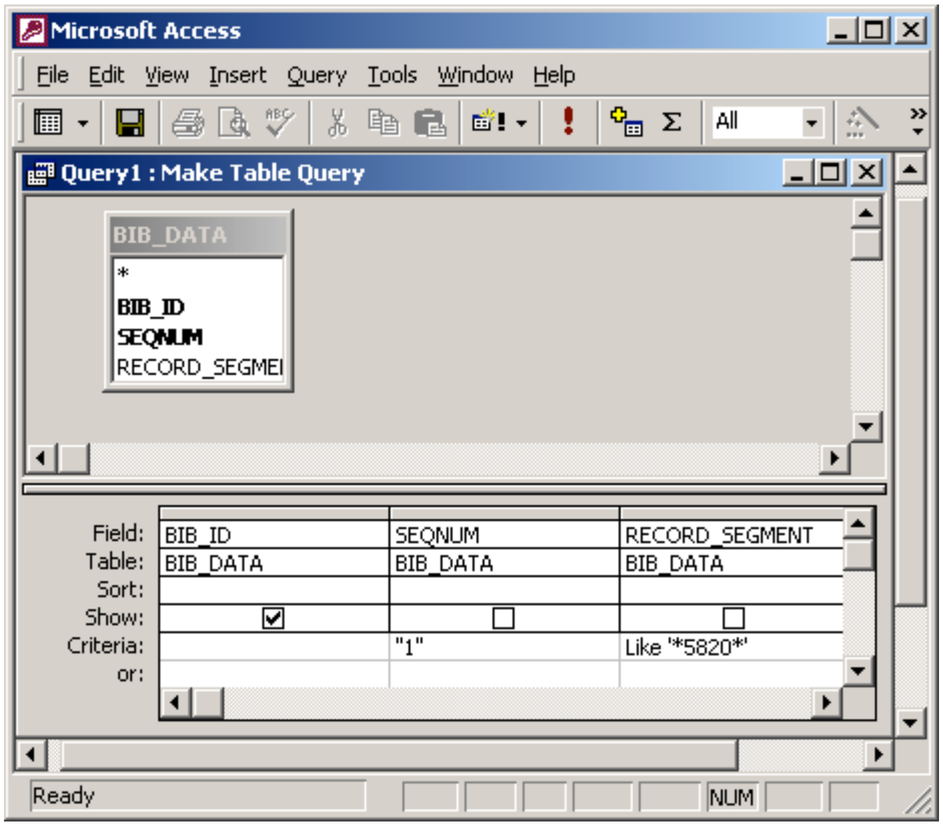
In the Voyager BIB_DATA table, unparsed MARC bib records are split up into segments that are up to 990 characters in length. The first segment has seqnum=1, the second has seqnum=2, etc. A bib record that is less than 990 characters long will have a single segment with seqnum=1. Longer MARC records will have several segments with appropriate values in seqnum. So, in my query, a way to search near the beginning of a record is to search records with seqnum=1.

> (Note: If I were using this technique on MFHDs instead of bibs, I would not use seqnum=1 to limit my search to the first segment. MFHD segments are only 300 characters long, so it's not unusual for the record directory to spill over into segments 2, 3, and beyond.)

Now what should I search for? I want to search for '582' at least. I think I can do a little better than that. I bet the length of my 582s will never exceed 999 characters, so the first digit of the 4-digit length will be a zero. So I'm going to search for the string '5820' anywhere in the first segment of a MARC bib record.

Now the string '5820' will certainly be present in any bib record that has a 582 tag. But it might also be present in other places, for example, as part of an ISBN. That's OK. All I'm trying to do in Query 1 is to reduce the number of bib records that I have to parse using the BLOB functions. Looking for '5820' is a reasonable way to accomplish this. When I do Query 3, I'll drop the cases where there is no 582 tag.

It's hard to see in design view, but I remembered to set the Unique Values property and to make this a Make Table query. The table I'm making is called Table 1.

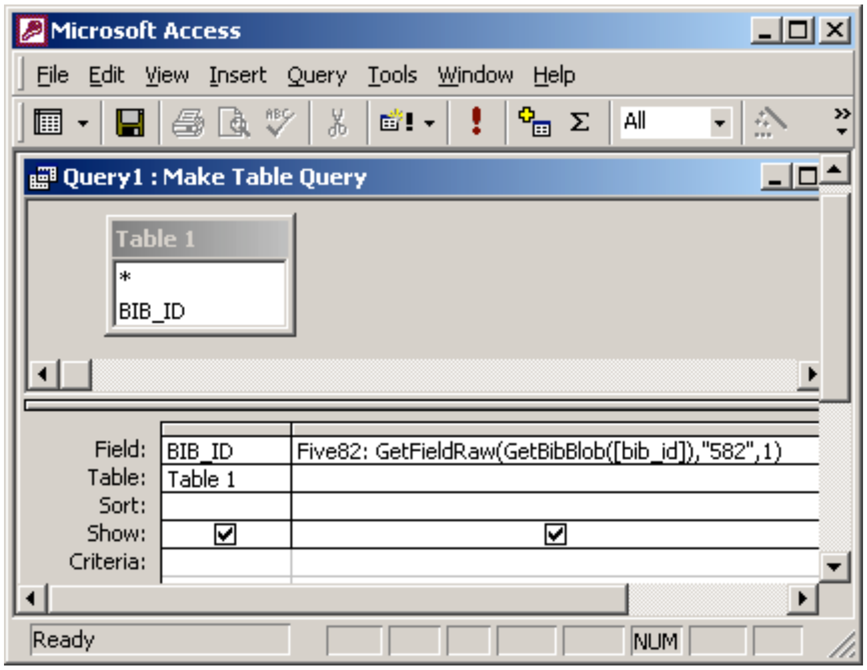

Microsoft Access
File Edit View Insert Query Tools Window Help
All
Query1 : Make Table Query
BIB_DATA
*
BIB_ID
SEQNUM
RECORD_SEGMEI
Ready
NUM

Field:	BIB_ID	SEQNUM	RECORD_SEGMENT
Table:	BIB_DATA	BIB_DATA	BIB_DATA
Sort:			
Show:	☑	☐	☐
Criteria:		"1"	Like '*5820*'
or:			

```
SELECT DISTINCT BIB_DATA.BIB_ID INTO [Table 1]
FROM BIB_DATA
WHERE (((BIB_DATA.SEQNUM)="1") AND ((BIB_DATA.RECORD_SEGMENT) Like '*5820*'));
```

Example B – Query 2

Now I need to use the BLOB functions to get the 582 field. As with Example A, I'm satisfied if I just get the first 582 field because I'll see any others that may be present when I look at the first one.

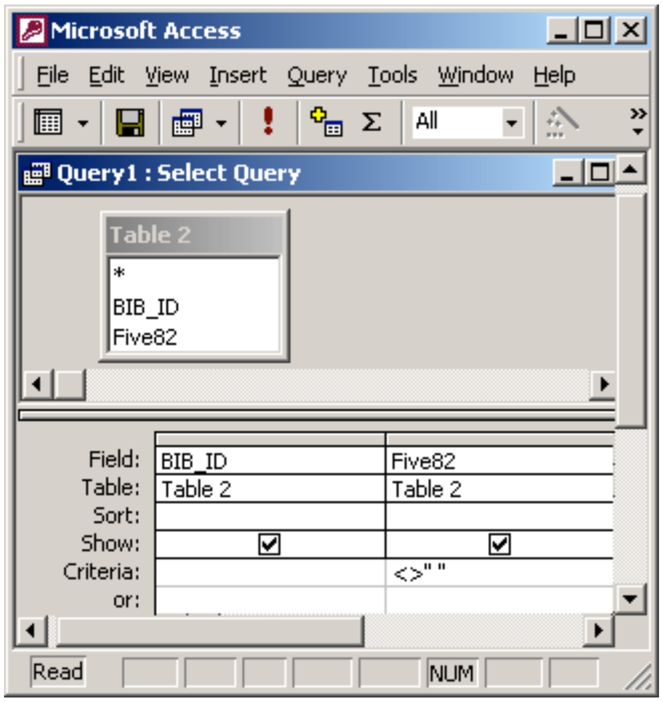

Microsoft Access
File Edit View Insert Query Tools Window Help
All
Query1 : Make Table Query
Table 1
*
BIB_ID
Ready
NUM

Field:	BIB_ID	Five82: GetFieldRaw(GetBibBlob([bib_id]),"582",1)
Table:	Table 1	
Sort:		
Show:	☑	☑
Criteria:		

```
SELECT [Table 1].BIB_ID, GetFieldRaw(GetBibBlob([bib_id]),"582",1) AS Five82
INTO [Table 2]
FROM [Table 1];
```

Example B – Query 3

Now I just need to weed out the records that truly have no 582 field.



Field:	BIB_ID	Five82
Table:	Table 2	Table 2
Sort:		
Show:	✓	✓
Criteria:		<>" "
or:		

```
SELECT [Table 2].BIB_ID, [Table 2].Five82
FROM [Table 2]
WHERE ((([Table 2].Five82)<>" "));
```

Example C

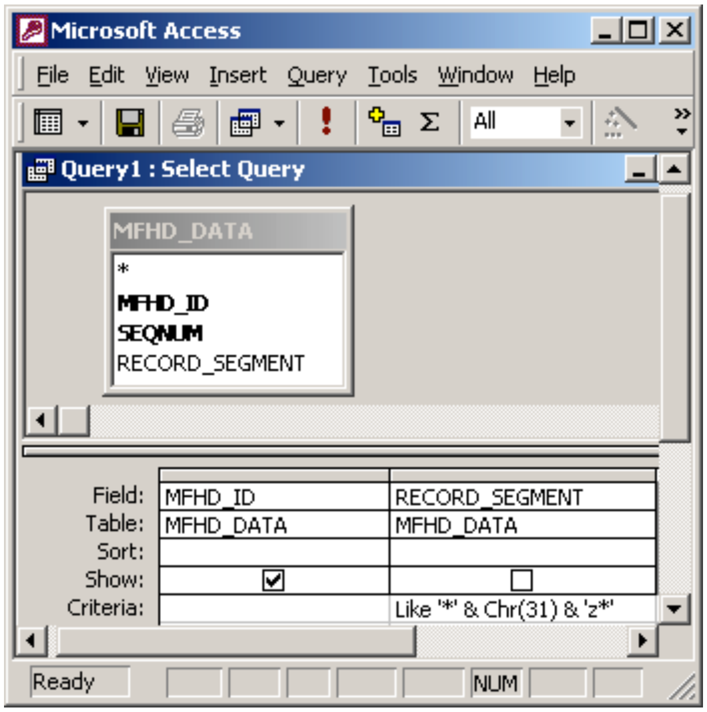
I want a list of all the public notes in the 852 field. Public notes are in 852 $z.

Example C – Query 1

Since all MFHDs have an 852 field, I can't use the strategy that I used in Example B. Subfield z is not very common in a MFHD, though. It occurs in 852 and also in the 86x fields. But I think this is good enough. So I want to search the unparsed MFHDs for a subfield delimiter followed by a 'z'.

The subfield delimiter is an unprintable character in Access, so I'm going to have to use CHR(31) to express it. It makes the criterion look complicated, but it works. (BTW, CHR(30) is the MARC end-of-field character and CHR(29) is the MARC end-of-record character, should you ever need them.)

It's hard to see in design view, but I remembered to set the Unique Values property and to make this a Make Table query. The table I'm making is called Table 1.

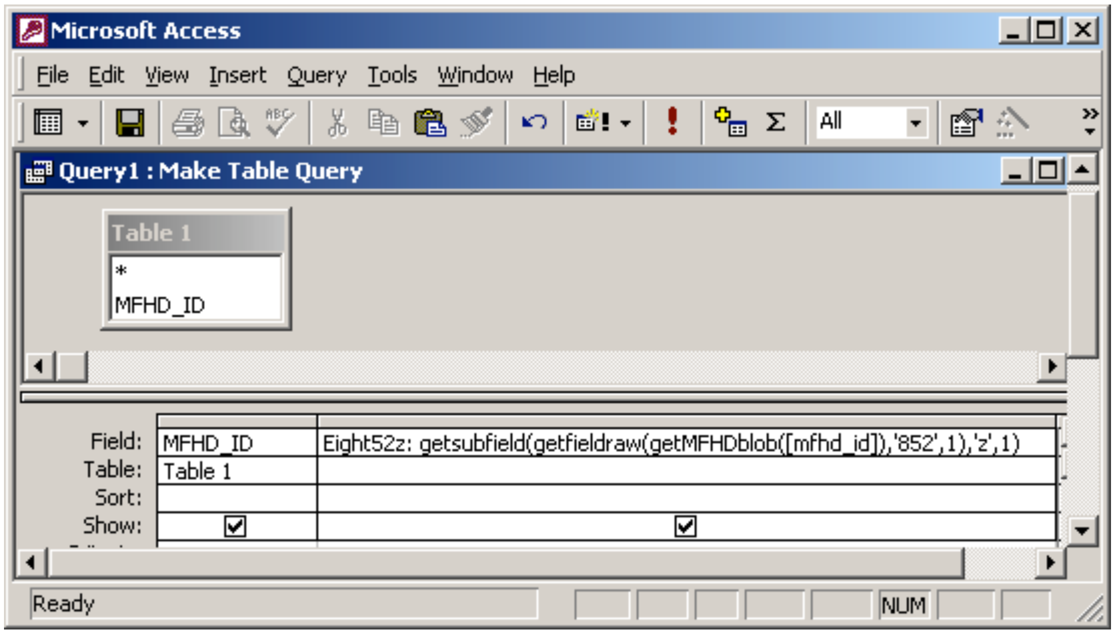


Field:	MFHD_ID	RECORD_SEGMENT
Table:	MFHD_DATA	MFHD_DATA
Sort:		
Show:	☑	☐
Criteria:		Like '*' & Chr(31) & 'z*'

SELECT DISTINCT MFHD_DATA.MFHD_ID INTO [Table 1]
FROM MFHD_DATA
WHERE (((MFHD_DATA.RECORD_SEGMENT) Like '*' & Chr(31) & 'z*'));

Example C – Query 2

Now I want to parse the MFHDs and list the 852$z.

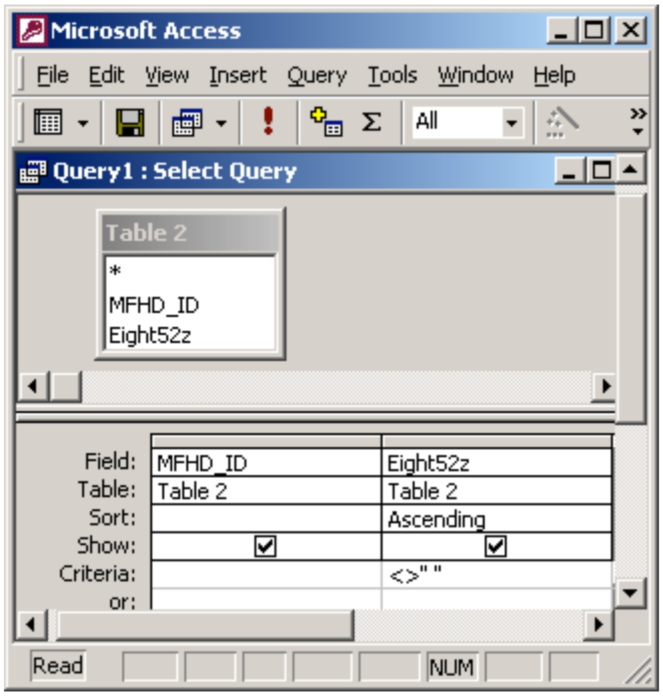

Microsoft Access
File Edit View Insert Query Tools Window Help
All
Query1 : Make Table Query
Table 1
*
MFHD_ID
Ready
NUM

Field:	MFHD_ID	Eight52z: getsubfield(getfieldraw(getMFHDblob([mfhd_id]),'852',1),'z',1)
Table:	Table 1	
Sort:		
Show:	✓	✓

```
SELECT [Table 1].MFHD_ID, getsubfield(getfieldraw(getMFHDblob([mfhd_id]),'852',1),'z',1) AS
Eight52z INTO [Table 2]
FROM [Table 1];
```

Example C – Query 3

And now I just need to drop the entries that have nothing in 852$z. I think I'll sort by the note too, so I can see if I've been entering my notes in a consistent format.



Field:	MFHD_ID	Eight52z
Table:	Table 2	Table 2
Sort:		Ascending
Show:	✓	✓
Criteria:		<>" "
or:		

```
SELECT [Table 2].MFHD_ID, [Table 2].Eight52z
FROM [Table 2]
WHERE ((([Table 2].Eight52z)<>" "))
ORDER BY [Table 2].Eight52z;
```