

Reading NIBRS into Microsoft ACCESS 2000

Justice Research and Statistics Association's Incident-Based Resource Center
www.jrsa.org/ibrrc

The National Incident-Based Reporting System (NIBRS) consists of predefined data elements and data values that describe each criminal incident and arrest reported by participating U.S. law enforcement agencies. The participating agencies (or their state UCR programs) submit NIBRS data to the Federal Bureau of Investigation (FBI). The FBI releases a consolidated NIBRS data file annually. This document provides instructions for importing data from the annual NIBRS data file into a Microsoft Access 2000 database.

NIBRS Data Segments and Segment Relationships

NIBRS incident and arrest reports are made up of more than 60 data elements grouped together in 7 segments. The first 6 segments record information on [*Group A offenses*](#), as defined by the FBI. The last segment records information on [*Group B offenses*](#). For more information on these segments, see [*Offense Group "A" Incident Report Administrative Segment Data Element Characteristics*](#).

Incident is the unit of count in NIBRS: it is the entity that is being reported. Each incident will involve one or many offenses (every incident involves at least one criminal offense), none, one, or many items of property (not all incidents involve loss of property), one or many victims (every incident has a victim, whether a person or an entity), one or many offenders (every incident involves at least one offender), and none, one, or many arrestees (an incident may not have an arrestee).

Similar relationships between data elements also exist within each segment. For example, one victim may have none, one, or many injuries reported. One arrestee may have been apprehended with none, one, or many weapons. The way in which we organize NIBRS data in tables in an MS Access database must reflect the inter- and intra-segment relationships between the NIBRS data elements.

Database Primer

A database can be defined generically as a collection of information. In this sense, the phone book or a box of recipes or the entire World Wide Web can be called databases.

A computer database is a collection of information organized in such a way that a computer program can quickly select desired pieces of data. The simplest form of computer database is the flat file database. A flat file is an unstructured data file that contains lines (or records) of data in no particular order. The annual NIBRS text file is a flat file. A flat file has no structure: it is just line after line of data. Simple structure is imposed on the file once we know what is contained in each record and where in the record we can find a particular piece of data. This map of the data in each record is called a record layout.

A sophisticated way of structuring data records is by placing the data in a table in a relational database. A table is a way of organizing data into columns, called fields, and rows, called records. In a relational database, distinct entities are stored in different database tables and relationships are used to connect the entities. The 7 data segments in the annual NIBRS data file can be thought of as entities.

Volumes have been written on the theory and practice of relational database design. In short, the basic rules for tables in a relational database are to design each table so that:

- there is a field or combination of fields in each table that makes each record unique;
- there are no repeating fields in a table;
- each table contains a common field or fields that are used to connect to other related tables in the database.

The connections between records in different data tables are provided by relationships. Relationships between tables are always made through keys. A key is a field or combination of fields. Database tables are related to each other by linking the key from one table to the key of another table. Key fields always have the same name in both tables. Each of these database table concepts is illustrated below.

Relational Database Table Example

We will use the NIBRS flat file record layout to place 8 data elements from the NIBRS Victim Segment into an example database table called Victim Table. After the table is loaded with fictitious data, we will verify that the table does not violate the rules of relational database table design described above. If any of the rules are violated, we will have to redesign the table.

Here is Victim Table:

Incident ID	Victim Number	Victim Offense 1	Victim Offense 2	Victim Age	Victim Injury 1	Victim Injury 2	Victim Injury 3
A	1	X		12	N		
A	2	X	Y	44	B		
B	2	W		20	L	I	O
C	1	W	Z	13	N		
C	2	X		35	U	T	
C	3	X	Z	76	L	O	

The first step in checking the integrity of the table is to identify our key field or fields (a key is a field or combination of fields that uniquely identifies every row in Victim Table).

Incident ID would seem a reasonable choice, however, a row-by-row glance at the data in the Incident ID field shows that the Incident ID repeats: Incident ID A has two victims and so appears in two rows and Incident ID C has three victims and so appears in three rows. The Incident ID field alone cannot be used as the key field for Victim Table. Can the Victim Number field be used to identify uniquely each row in the table? No. Like the Incident ID field, there are repeating instances of data in the Victim Number field.

The fields Incident ID and Victim Number can be used *together* as the key for Victim Table because there are no repeating row-by-row combinations of the two fields: the combinations are A 1, A 2, B 2, C 1, C 2 and C 3. From here on we will put the field names Incident ID and Victim Number in bold to indicate that they are key fields.

The next step in checking the integrity of the table is to verify that there are no repeating fields. We see quickly that there are multiple fields that identify the offense or offenses against the victim (Victim Offense 1 and Victim Offense 2) and multiple fields that identify the injury or injuries sustained by the victim (Victim Injury 1, Victim Injury 2, Victim Injury 3). These repeating columns violate one of the rules of relational table design, so we have to redesign the table. The table can be redesigned by creating *additional* tables that will be used to store the data in the repeating columns. The additional tables must also contain the key fields so that the tables can be linked to one another.

Victim Offense Table will be the first additional table created:

Incident ID	Victim Number	Victim Offense
A	1	X
A	2	X
A	2	Y
B	2	W
C	1	W
C	1	Z
C	2	X
C	3	X
C	3	Z

This new Victim Offense Table contains a *single* field called Victim Offense that contains the victim offense data from *two* fields in Victim Table. The key fields are present in Victim Offense Table and every row in the table is unique: there are no repeating row-by-row combinations of Incident ID, Victim Number, and Victim Offense. We delete the two Victim Offense fields from Victim Table now that the information is stored in Victim Offense Table so Victim Table now looks like this:

Incident ID	Victim Number	Victim Age	Victim Injury 1	Victim Injury 2	Victim Injury 3
A	1	12	N		
A	2	44	B		
B	2	20	L	I	O
C	1	13	N		
C	2	35	U	T	
C	3	76	L	O	

There is further redesign needed because there are multiple Victim Injury fields in Victim Table. We will create a new table called Victim Injury to store the data in the Victim Injury fields:

Victim Injury Table:

Incident ID	Victim Number	Victim Injury
A	1	N
A	2	B
B	2	L
B	2	I
B	2	O
C	1	N
C	2	U
C	2	T
C	3	L
C	3	O

This new Victim Injury Table contains a *single* field called Victim Injury that contains the victim injury data from *three* fields in Victim Table. The key fields are present in Victim Injury Table and every row in the table is unique: there are no repeating rows in the table. We delete the three Victim Injury fields from Victim Table now that the information is stored in Victim Injury Table so Victim Table now looks like this:

Incident ID	Victim Number	Victim Age
A	1	12
A	2	44
B	2	20
C	1	13
C	2	35
C	3	76

We check Victim Table again for multiple fields. There are none so there are no more actions to perform on the Victim Table.

This process results in three tables, Victim Table, Victim Offense Table and Victim Injury Table, that do not violate any of the rules for relational database table design. Tables created in this way are said to be normalized. The complete process is called database table normalization.

Next we will show how the tables are linked together through the key fields.

Looking at the Victim Offense Table, how do we determine the age of the three victims in Incident C? That information is contained in Victim Table and must be related back to the information contained in Victim Offense Table. The relationship between the tables is established via the key fields that appear in both tables--Incident ID and Victim Number:

Victim Table

Incident ID	Victim Number	Victim Age
A	1	12
A	2	44
B	2	20
C	1	13
C	2	35
C	3	76

Victim Offense Table

Incident ID	Victim Number	Victim Offense
A	1	X
A	2	X
A	2	Y
B	2	W
C	1	W
C	1	Z
C	2	X
C	3	X
C	3	Z

Incident C in Victim Offense Table involves three victims: Victim Number 1, Victim Number 2 and Victim Number 3. Victim Table contains the ages for the three victims in Incident C. The lines between the two tables show how the tables relate to one another via the key fields. The tables are not linked physically together; rather, they are linked logically by executing a short computer program. After the program executes, the tables "know" that they are linked to one another via the key fields.

Creating NIBRS Tables

The procedures described above illustrate the general procedure for creating and normalizing a relational database table. We will use this procedure to create normalized NIBRS tables in a Microsoft Access 2000 database.

The MS Access 2000 program application IMPORT_NIBRS_DATA00.mdb, available for download at [JRSA's Incident-Based Reporting Resource Center](#), can be used to create normalized NIBRS tables. This program application will: 1) create 7 table structures, one for each of the NIBRS segments; 2) load the data from the NIBRS flat file into the tables; 3) normalize the tables as described above; and 4) establish all table relationships. IMPORT_NIBRS_DATA00.mdb performs these steps using embedded programs in the application program. It is easier to create and normalize tables programmatically than through the MS Access wizards because programs involve considerably fewer keystrokes than using the wizards. Programs are ideal for automating the repetitive task of creating, loading, and normalizing tables. Programs are also easily shared between users. Once a program has been tested thoroughly and achieves the desired results, there is no need to "reinvent the wheel" every time a user wants to import NIBRS data into tables. IMPORT_NIBRS_DATA00.mdb also takes advantage of the interactive features in MS Access by prompting for user input and providing processing status information. The remainder of this document provides instructions on how to use the IMPORT_NIBRS_DATA00.mdb program to import NIBRS data into Microsoft Access 2000.

Microsoft Access Database Terminology

Since terminology often differs among software packages, the following are some of the terms that will be used throughout the rest of this document:

Query - a short program written in a language like Structured Query Language (SQL) that is used to extract information from a database. Queries can also be used to create or change database tables.

Index - a field or combination of fields used to order the records in a table. An index improves query performance and table sorting.

Macro - a set of one or more recorded actions that perform a particular operation in a Microsoft Access database. Macros can be used to automate many related or repetitive tasks.

Module - for our purposes, a module is a short program written in a programming language called Visual Basic for Applications. Modules often perform complex operations on databases and files.

Text Link Specification - instructions input by the user that tell Microsoft Access how to "connect" a flat file to a database.

Overview of IMPORT NIBRS DATA00.mdb

This Microsoft Access application contains the 13 macros, 1 module, 1 text link specification, and 91 data definition and data append queries needed to create 17 normalized NIBRS tables and table relationships in ACCESS 2000. First, the program application will create and load the 7 NIBRS segments tables. Please click on the appropriate table name to see the resulting table structure:

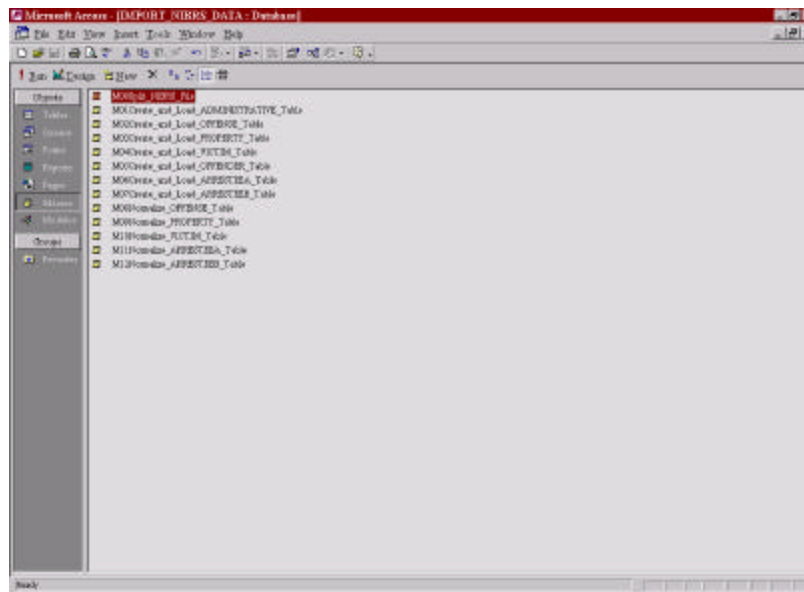
- [ADMINISTRATIVE](#)
- [OFFENSE](#)
- [PROPERTY](#)
- [VICTIM](#)
- [OFFENDER](#)
- [ARRESTEEA](#)
- [ARRESTEEB](#)

Once the ADMINISTRATIVE and OFFENDER tables are created, there are no more actions to perform on them. This is because of the way the Administrative and Offender information is organized in the NIBRS file. When we use the NIBRS record layout to place ADMINISTRATIVE and OFFENDER data in a database table, we see immediately that the rules of table design are met: each record in the table is already unique and there will be no repeating fields in the tables. These two tables are already normalized.

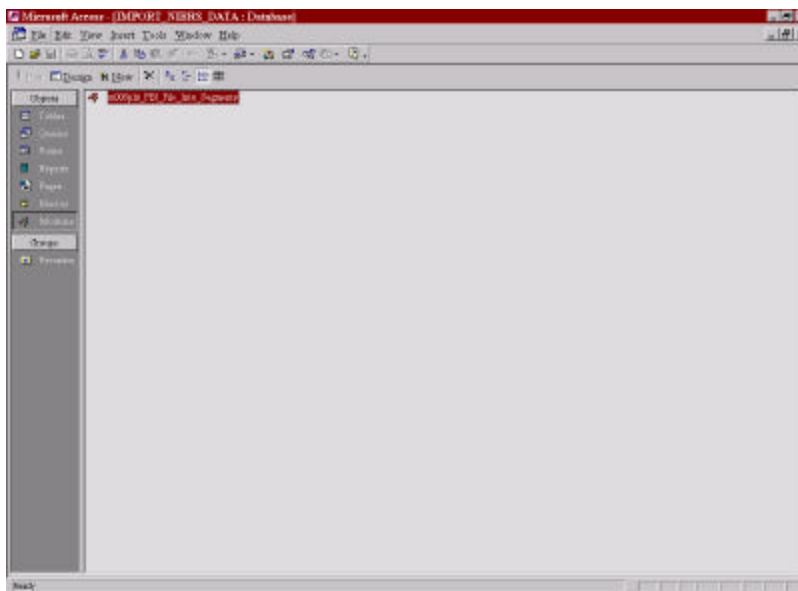
The OFFENSE, PROPERTY, VICTIM, ARRESTEEA and ARRESTEEB tables, however, will require further manipulation because the data for these segments are organized in the NIBRS flat file in a way that results in repeating fields in each table. Using the general procedure described above, these five tables will be normalized, resulting in 10 additional tables:

- [OFFENSE_ACTIVITY](#): from the three "criminal activity" fields originally in table OFFENSE
- [OFFENSE_USING](#): from the three "suspect using" fields originally in table OFFENSE
- [OFFENSE_WEAPON](#): from the 3 "weapon/force used" and 3 "automatic weapon indicator" fields originally in table OFFENSE
- [PROPERTY_DRUG](#): from the 3 "drug type," 3 "drug quantity" and 3 "drug measurement" fields originally in table OFFENSE
- [VICTIM_CIRCUMSTANCE](#): from the 3 victim circumstance fields originally in table VICTIM
- [VICTIM_INJURY](#): from the 5 victim injury fields originally in table VICTIM
- [VICTIM_OFFENSE](#): from the 10 victim offense fields originally in table VICTIM
- [VICTIM_TO_OFFENDER](#): from the 10 offender sequence number and 10 victim-to-offender relationship fields originally in table VICTIM
- [ARRESTEEA_WEAPON](#): from the 2 "arrestee armed with" and 2 "automatic weapon indicator" fields originally in table ARRESTEEA

The database macro view:



Each macro runs sets of related queries shown in the query view above.
Macro M01Split_NIBRS_File runs the program shown in the database module view below:



If you would also like to view the text link specification, [follow this link](#) for instructions.

Step-by-Step Instructions For Importing NIBRS Data Into MS Access 2000 **Using IMPORT_NIBRS_DATA.mdb**

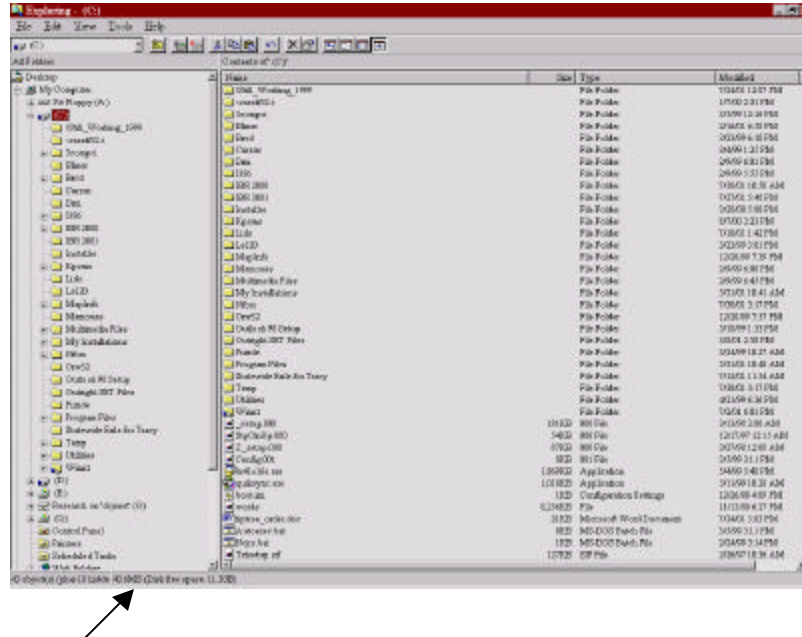
Now we will walk through the application and create NIBRS tables. In this example, we use the 1999 NIBRS flat file from the FBI as our input data. [If you need any assistance, please contact [JRSA](mailto:ibrrc@jrsa.org) (ibrrc@jrsa.org).] *Note: The actions performed by this program application are processor-intensive. After downloading the program, be sure to close all open applications on your PC before you begin.*

1. Verify that you have the 1999 NIBRS flat file from the FBI and Microsoft Access 2000 installed on your PC.
2. Make note of the code(s) of the state(s) for which you want to create tables. The 1999 FBI NIBRS flat file contains full or partial data from these 18 states:

State Name	State Code
Arkansas	03
Colorado	05
Connecticut	06
Idaho	11
Iowa	14
Kentucky	16
Massachusetts	20
Michigan	21
Nebraska	26
North Dakota	33
Ohio	34
South Carolina	39
Tennessee	41
Texas	42
Utah	43
Vermont	44
Virginia	45
West Virginia	47

3. Verify that you have at least 1.0GB of free space on your hard drive for the 985,142KB 1999 NIBRS text file plus an addition 500MB for each state selected. A separate set of tables will be created for each state. For example, if you want to create tables for one state, you should have at least 1.5GB free disk space. If you want to create tables for three states you should have at least 2.5GB free disk space. To check the amount of free disk space available on your hard drive, open Windows Explorer, and

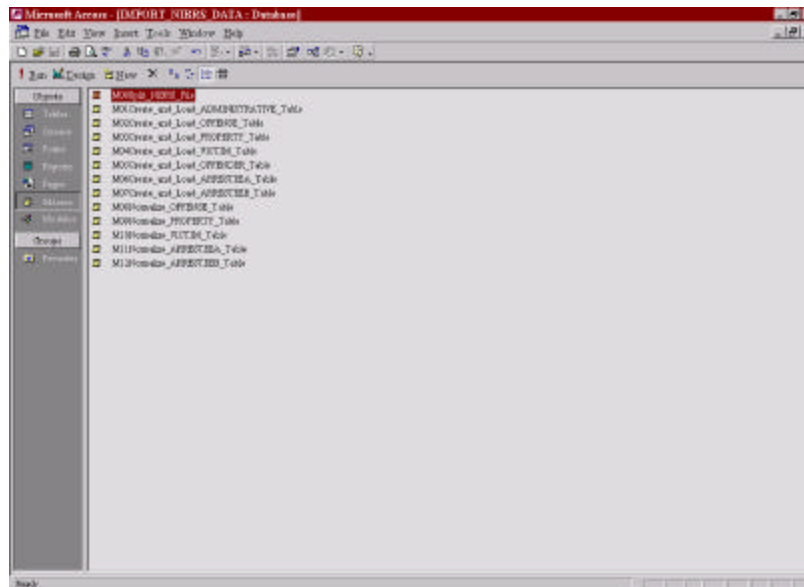
double-click the drive on which you want to copy the NIBRS text file. Drive C is chosen in this example:



The amount of disk free space is displayed in the status bar in the lower left-hand corner of the screen (indicated by the arrow above).

4. Create a folder called NIBRS on your hard drive. In this example we will create the folder on the C drive -- c:/NIBRS/.
5. Move the 1999 NIBRS flat file into this folder. If the file has the extension .DAT, rename the file with the extension .TXT. If there are blank spaces in the file name replace the blanks with underscores or remove the blanks all together. For example, a NIBRS file called NIBRS 1999 Flat File.DAT would be renamed NIBRS_1999_Flat_File.txt or NIBRS1999FlatFile.txt. Since ACCESS will not recognize .DAT files or file names with spaces, renaming the file and removing spaces is very important.
6. Download the program file IMPORT_NIBRS_DATA00.mdb from the [JRSA IBR Resource Center Web site](#) and save it in the NIBRS folder. This file has been compressed, so make sure you have a program such as WinZip or PKZip on your computer. These shareware programs can be downloaded from the Internet. Once the program has been unzipped, make a copy of IMPORT_NIBRS_DATA00.mdb for each set of state tables to be created. For example, to make tables for the states of Virginia and West Virginia (see step 2 above) make two copies of the program. *Be sure to give each program copy a meaningful name, for example, IMPORT_NIBRS_DATA_VIRGINIA.mdb or IMPORT_NIBRS_DATA_VA.mdb. Since changes have to be made to modify the original file, work with a copy of the .mdb file, not the original.* We will use IMPORT_NIBRS_DATA_VIRGINIA.mdb in this example.

7. The first task is to split the large NIBRS flat file. This program application includes a short program that will split the NIBRS file into the 7 NIBRS segment-level files for the state specified by the user. Open program IMPORT_NIBRS_DATA_VIRGINIA.mdb and select object button "Macros."



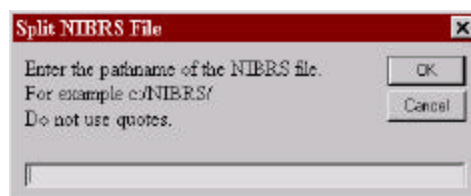
8. Double click on the macro M00Split_NIBRS_File.

9. The first of three prompts will appear:



Enter the two-digit state code and click "OK" or press "Enter." *If you enter anything other than a two-digit code from the list or if you press "Cancel", the program will end.*

10. The next prompt that appears asks for the pathname of the large NIBRS text file:



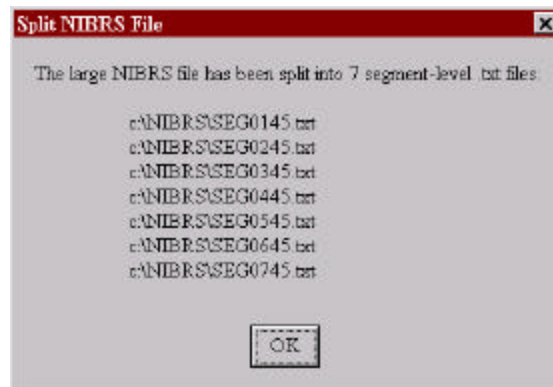
Without using quotes, enter the pathname then click "OK." For example, if you created the NIBRS folder on your C drive, you would enter c:/NIBRS/ (don't forget to include the closing slash). *If you press "Cancel", the program will end.*

11. The third and final prompt that appears asks for the name of the large NIBRS text file.



Enter the file name without using quotes; for example, NIBRS_FILE_1999.txt. Click "OK" or press "Enter." *If the path and/or file name you entered are incorrect or do not exist, or if you press "Cancel", the program will end.*

12. The process of splitting the large NIBRS text file into 7 segment-level text files for the state specified begins. This could take anywhere from 5 - 30 minutes, depending on the speed and configuration of your computer. An hourglass will appear in the middle of the current window indicating that the program is processing the NIBRS file. A message box will appear when the program is done:

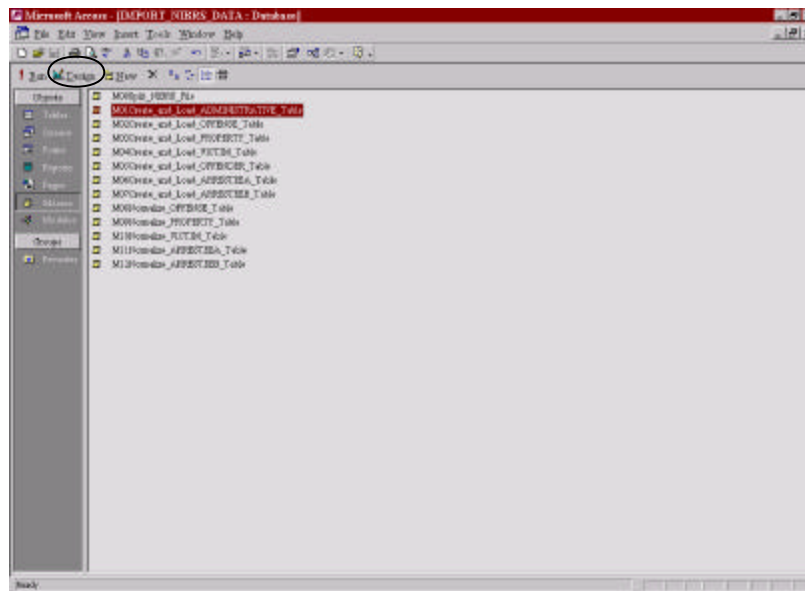


This box lists the name and location of the newly created segment-level files. The files are named according to the convention "SEGXXYY.txt", where "XX" corresponds to the NIBRS segment level (01 = Administrative, 02 = Offense, 03 = Property, 04 = Victim, 05 = Offender, 06 = Arrestee, 07 = Group B Arrest) and where "YY" corresponds to the two-digit state code. The files are saved on the same drive and folder entered in steps 9 and 10 above. Make a note of these seven file names, then click "OK" or press "Enter."

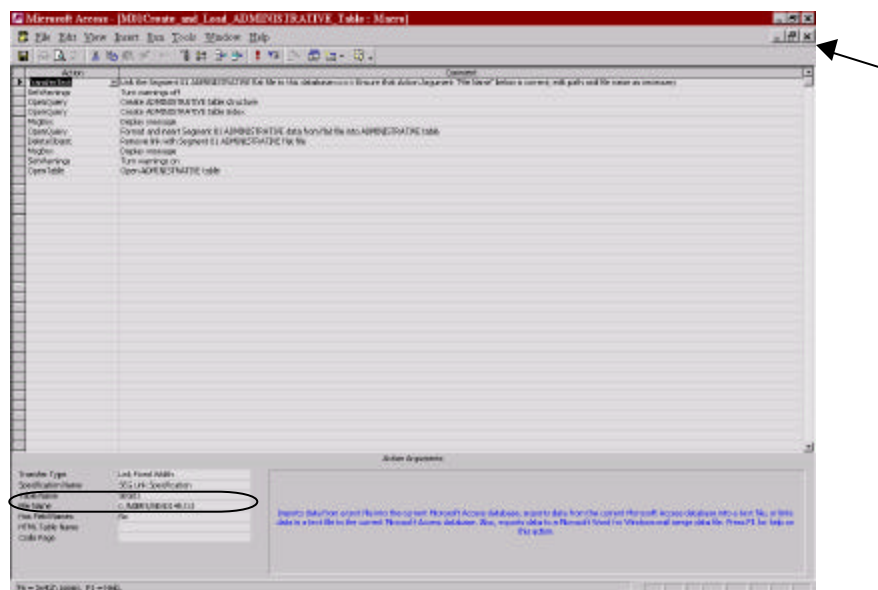
Now that the file has been split, we are ready to create each table structure, index, and relationship. We'll detail the process for the ADMINISTRATIVE table.

Create ADMINISTRATIVE Table

a) Highlight the macro M01Create_and_Load_ADMINISTRATIVE_Table by clicking on it *once*:

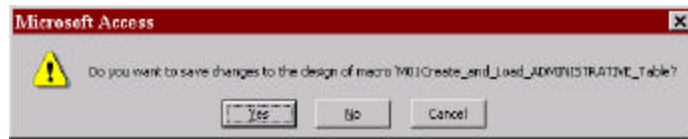


b) Click the "Design" button (circled above). This will open the macro in design view:



There is one item in this macro that must be changed before the macro is run: File Name (circled). The path and filename of the Segment 01 file to be created should be entered here. C:/NIBRS/SEG0145.txt is used for this example. Do not put quotes around the entry.

c) Save the macro by clicking the Close Window button (the "x" indicated by the arrow above). The Save Macro warning box will appear:



Click "Yes" or press "Enter."

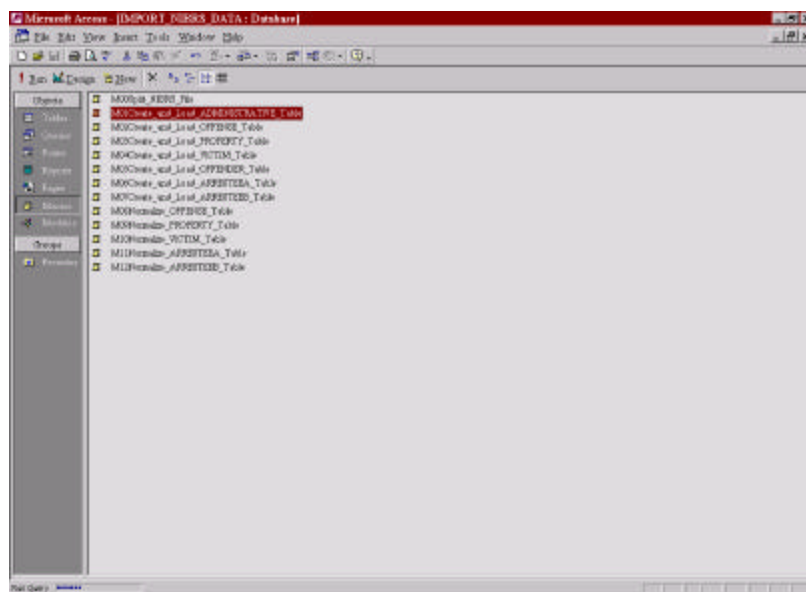
d) After clicking the "Yes" button, the Macro objects window will resemble the window in Step b above. Run the macro by double clicking on the macro name M01Create_and_Load_ADMINISTRATIVE_Table.

e) The macro will first link the Administrative (Segment 01) file c:/NIBRS/SEG0145.txt to the database. Then it creates the structure for the ADMINISTRATIVE table and the table index. These actions are performed quickly. When complete, a message box appears:



Click "OK" or press "Enter."

f) The macro continues by formatting the data from the text file, inserting the formatted data into the ADMINISTRATIVE table, and removing the link with the Segment 01 flat file. This is accomplished by a number of queries being executed. These queries could take anywhere from 5 - 30 minutes depending on the speed and configuration of your PC and the size of the segment-level file that is being formatted:

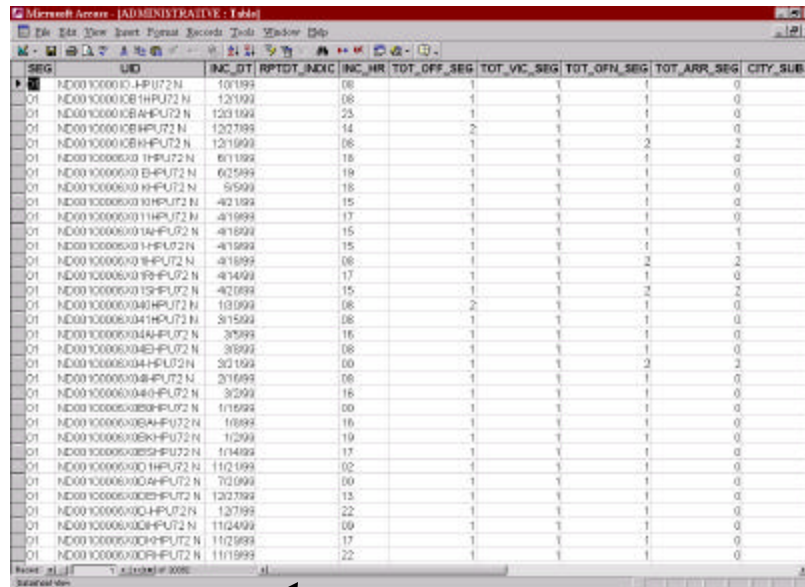


Watch the query progress bar (indicated by the arrow above) for macro status.

g) When the macro is done, another message box appears:



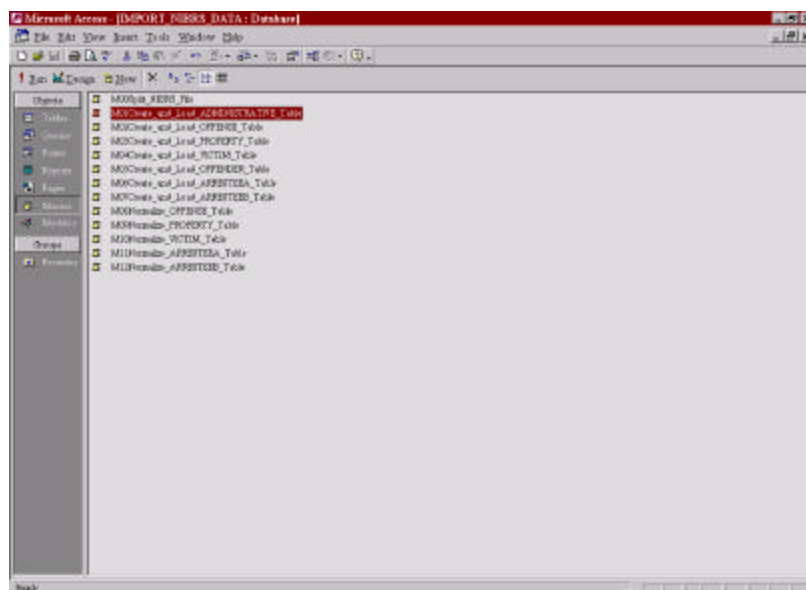
After clicking "OK" or pressing "Enter", the ADMINISTRATIVE table opens:

A screenshot of the Microsoft Access 'ADMINISTRATIVE' table view. The table is displayed in a grid format with the following columns: SEG, UID, INC_DT, RPTDT_INXC, INC_HR, TOT_OFF_SEG, TOT_VC_SEG, TOT_OFN_SEG, TOT_ARR_SEG, and CITY_SUB. The data rows show various values for these fields, such as '01', 'ND00 Y0000 I01 HPU72 N', '120199', '08', '1', '1', '1', '1', '0'. A scroll bar is visible at the bottom of the table grid, with an arrow pointing to it from the text below.

This is a partial view of the table. There are more fields to the right of the screen. Use the scroll bar (indicated by the arrow above) to scroll over to view the remaining fields in the table.

Perform a quick "eyeball" check of the table to ensure that the data were imported properly.

h) Close the ADMINISTRATIVE table. You are automatically returned to the database "Macros" view:



i) Repeat steps b) through h) for each of the following macros:

M02Create_and_Load_OFFENSE_Table
(Change File Name to c:/NIBRS/SEG0245.txt)

M03Create_and_Load_PROPERTY_Table
(Change File Name to c:/NIBRS/SEG0345.txt)

M04Create_and_Load_VICTIM_Table
(Change File Name to c:/NIBRS/SEG0445.txt)

M05Create_and_Load_OFFENDER_Table
(Change File Name to c:/NIBRS/SEG0545.txt)

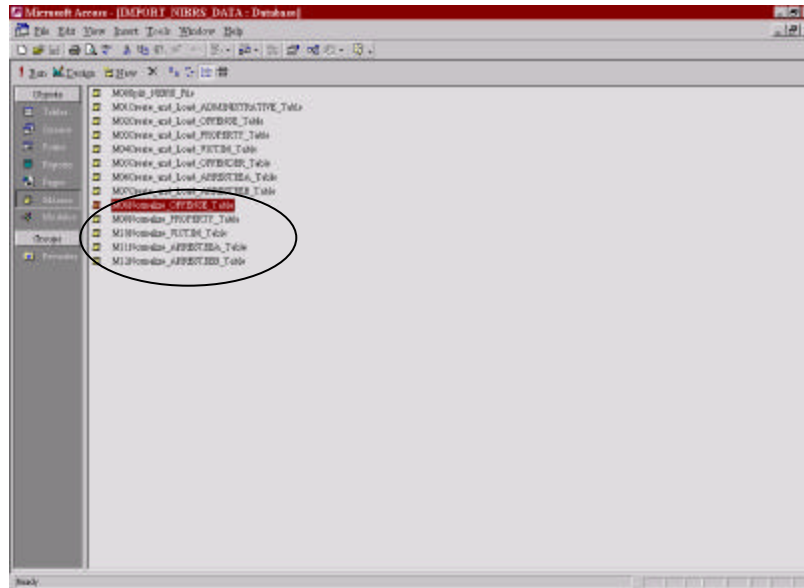
M06Create_and_Load_ARRESTEEA_Table
(Change File Name to c:/NIBRS/SEG0645.txt)

M07Create_and_Load_ARRESTEEB_Table
(Change File Name to c:/NIBRS/SEG0745.txt)

Run the macros in the order presented. Running out of order could result in an error.

Normalizing The Tables and Creating Table Relationships

The next task is to normalize the OFFENSE, PROPERTY, VICTIM, ARRESTEEA and ARRESTEEB tables and create relationships among the tables. This is done by running 5 macros: M08Normalize_OFFENSE_Table, M09Normalize_PROPERTY_Table, M10Normalize_VICTIM_Table, M11Normalize_ARRESTEEA_Table, and M12Normalize_ARRESTEEB_Table (circled):



No changes need to be made to the normalizing macros. To run, simply double-click on the macro name. After each macro is complete, a message box appears. The message box for M08 is:

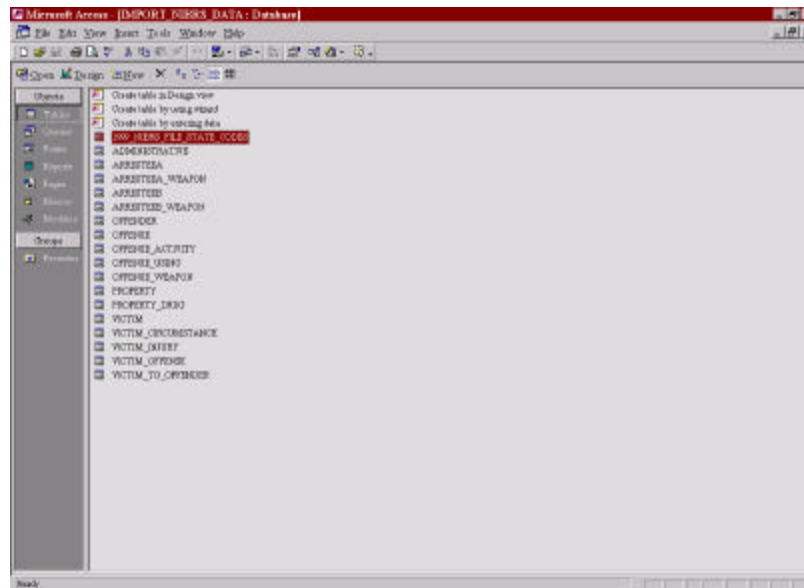


A similar box appears after macros M09 through M12 are complete. It is important to *run the normalizing macros only after macros M01 through M07 have been completed and they must be run in order--M08, M09, M10, M11, M12. Otherwise, errors will occur.*

To create another set of tables for an additional state selected in step 2 above, simply open the corresponding program (copied and renamed in step 5 above) and repeat all instructions.

Final NIBRS Tables

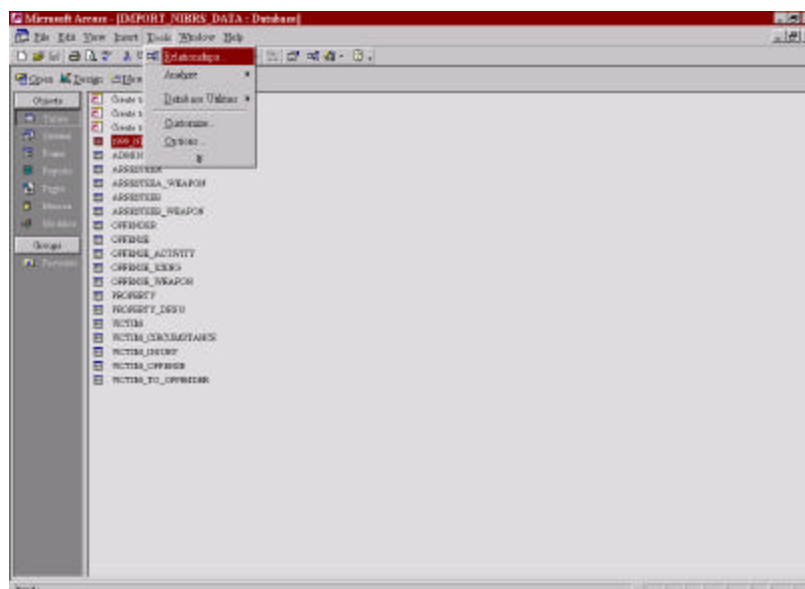
When all table-creating and table-normalizing macros are complete, the database table window will appear as follows:



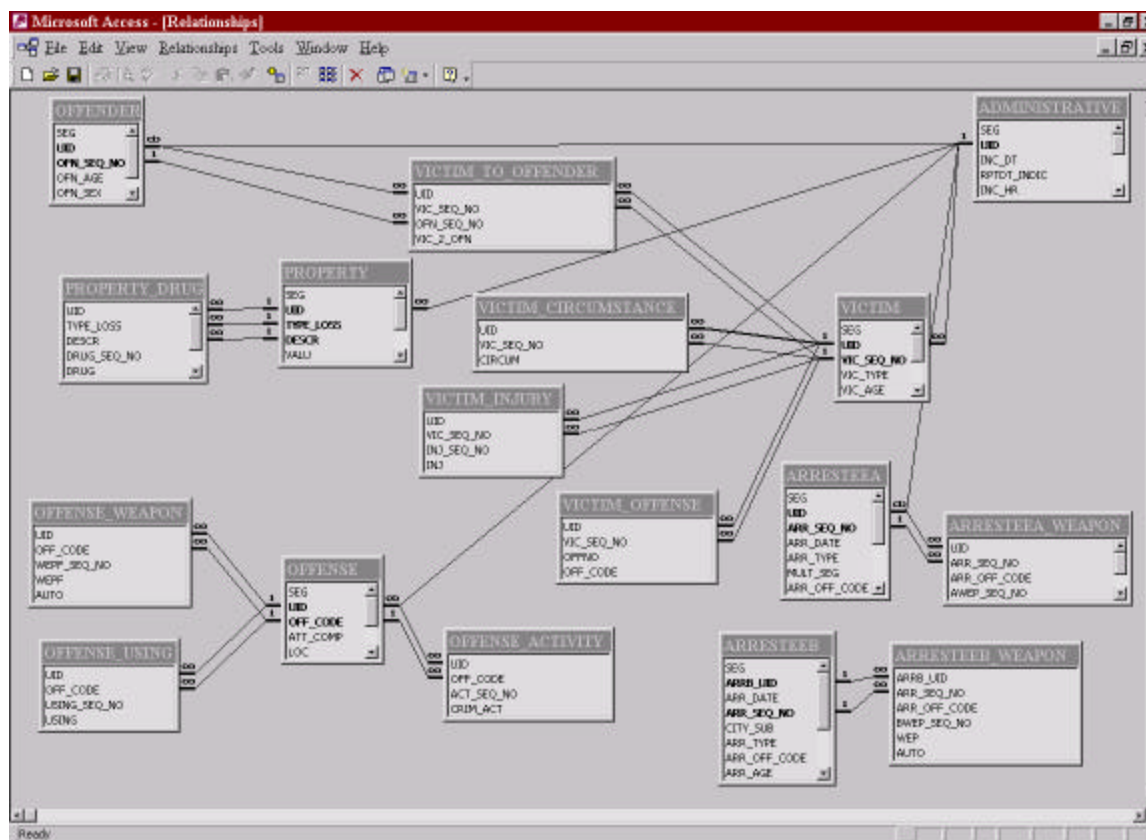
As you can see, the newly created NIBRS tables are listed.

View the Table Relationships

To view the relationships between the tables, click "Tools" then "Relationships" on the top menu bar:



This daunting-looking window will appear:



Notice that every NIBRS table appears in this view, with linking lines between related tables. The "infinity" symbols to the sides of the tables represents a "many" relationship. For instance, for every offense record in the OFFENSE table, there could be many weapons in the OFFENSE_WEAPON table associated with the offense. In fact, a maximum of three weapons may be recorded in the NIBRS data for each offense record. A "many" relationship here means there could be none, one, two, or three records in the OFFENSE_WEAPON table that correspond to one record in the OFFENSE table. The bold names in each table are the fields that comprise that table's key. The key from one table is used to link to the key of a related table.

The links between the tables are outlined below:

1. The ADMINISTRATIVE table is linked to the OFFENSE, PROPERTY, VICTIM, OFFENDER and ARRESTEEA tables through the key called UID. UID, a combination of the NIBRS data elements "ORI Number" and "Incident Number," is the field that uniquely identifies each record in the ADMINISTRATIVE table. These are the inter-segment relationships discussed in the beginning section of this document.
2. The intra-segment relationships are depicted as well:

- Each unique record in OFFENSE is linked to the records in tables OFFENSE_WEAPON, OFFENSE_USING and OFFENSE_ACTIVITY.
- Each unique record in PROPERTY is linked to the records in table PROPERTY_DRUG.
- The unique records in table VICTIM are related to the records in tables VICTIM_CIRCUMSTANCES, VICTIM_INJURY, VICTIM_OFFENSE and VICTIM_TO_OFFENDER. Notice that there is a link between segment-level table OFFENDER and intra-segment-level table VICTIM_TO_OFFENDER.
- Each unique record in ARRESTEEA is related to the records in ARRESTEEA_WEAPON.

There is no link between the ARRESTEEB table and the ADMINISTRATIVE table because ARRESTEEB records are not tied to the incidents reported in table ADMINISTRATIVE.

The queries used in this program can be viewed from the [JRSA IBR Resource Center](#) Web page. These queries are written in SQL and can be modified to fit your needs.

Now that the tables have been created and related to each other, you are now ready to analyze the data. Enjoy!

If you have any problems, comments, or suggestions about using incident-based data in ACCESS, please [contact JRSA](#) at ibrrc@jrsa.org.