

**Reading Local Incident-Based Data Into Microsoft ACCESS 2000**  
Justice Research and Statistics Association Incident-Based Reporting Resource Center  
www.jrsa.org/ibrc

Local Incident-Based Reporting System (IBRS) data consist of predefined data elements and data values that describe each local criminal incident and arrest reported by participating U.S. law enforcement agencies. The participating agencies submit their local-level IBRS data to the Federal Bureau of Investigation for inclusion in the National Incident-Based Reporting System (NIBRS). These data are reported to the FBI in a standardized format that is different from the format of the data the FBI releases through NIBRS. This document provides instructions for importing data from a local-level IBRS in the FBI-specified format into a Microsoft Access 2000 database.

### **IBRS Data Segments and Segment Relationships**

IBRS 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](#).

A detailed description of the IBRS data elements is beyond the scope of this document. Such detail can be found in *Uniform Crime Reporting, National Incident-Based Reporting System, Volume 2, Data Submission Specifications, May 1992*, available for download at [www.fbi.gov/ucr/ucr.htm](http://www.fbi.gov/ucr/ucr.htm).

Incident is the unit of count in IBRS: 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.

Similar relationships exist between data elements 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 IBRS data in tables in an MS Access database must reflect the inter- and intra-segment relationships between the IBRS 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 a database.

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. A flat file is an unstructured data file that contains lines (or records) of data in no particular order. An IBRS text file is a flat file: 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 structure 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 IBRS 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 an IBRS flat file record layout to place 8 data elements from the IBRS Victim Segment into an example database table called the 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 the 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 the 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 the Victim table. Can the Victim Number field be used to uniquely identify each row in the table? No. Like the Incident ID field, there are instances of repeating data in the Victim Number field.

The fields Incident ID and Victim Number can be used *together* as the key for the 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. In the tables that follow, 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.

The Victim Offense table will be the first additional table created:

<b>Incident ID</b>	<b>Victim Number</b>	<b>Victim Offense</b>
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 has the victim offense data from *two* fields in the Victim table. The key fields are present in the 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 the Victim table now that the information is stored in the Victim Offense table. The Victim table now looks like this:

<b>Incident ID</b>	<b>Victim Number</b>	<b>Victim Age</b>	<b>Victim Injury 1</b>	<b>Victim Injury 2</b>	<b>Victim Injury 3</b>
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 the Victim table. We will create a new table called Victim Injury to store the data in the Victim Injury fields.

## Victim Injury table

<b>Incident ID</b>	<b>Victim Number</b>	<b>Victim Injury</b>
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 the Victim table. The key fields are present in the 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 the Victim table now that the information is stored in Victim Injury table. The Victim table now looks like this:

<b>Incident ID</b>	<b>Victim Number</b>	<b>Victim Age</b>
A	1	12
A	2	44
B	2	20
C	1	13
C	2	13
C	3	76

We check the 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 the Victim table and must be associated with the

information contained in the 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	13
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 the Victim Offense table involves three victims: Victim Number 1, Victim Number 2 and Victim Number 3. The Victim table contains the ages of 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 IBRS Tables

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

The MS Access 2000 program application IMPORT\_IBR\_DATA00.mdb, available for download at [JRSA's Incident-Based Reporting Resource Center](http://www.jrsa.org/ibrcc), can be used to create normalized IBRS tables. This program application will: 1) create 7 table structures, one for each of the IBRS segments; 2) load the data from the IBRS flat file into the tables; 3) normalize the tables as described above; and 4) establish all table relationships. IMPORT\_IBR\_DATA00.mdb

performs these steps using embedded programs in the program application. It is easier to create and normalize tables programmatically than through the MS Access wizards because programs involve considerably fewer keystrokes. Programs are ideal for automating the repetitive task of creating, loading, and normalizing tables, and are 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 IBRS data into tables. IMPORT\_IBR\_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\_IBR\_DATA00.mdb program to import IBRS data into Microsoft Access 2000.

Note: Database application IMPORT\_IBR\_DATA00.mdb is designed for the import of a static IBRS data file. The application is not capable of performing updates to the records in the database tables.

### **Microsoft Access Database Terminology**

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

*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\_IBR\_DATA00.mdb**

The Microsoft Access application IMPORT\_IBR\_DATA00.mdb contains the 14 macros, 2 modules, 1 text link specification, and 103 data definition and data append queries needed to create 18 normalized IBRS tables and table relationships. First, the database application will create and load tables for the 7 IBRS segments:

- [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 IBRS file. When we use the IBRS 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.

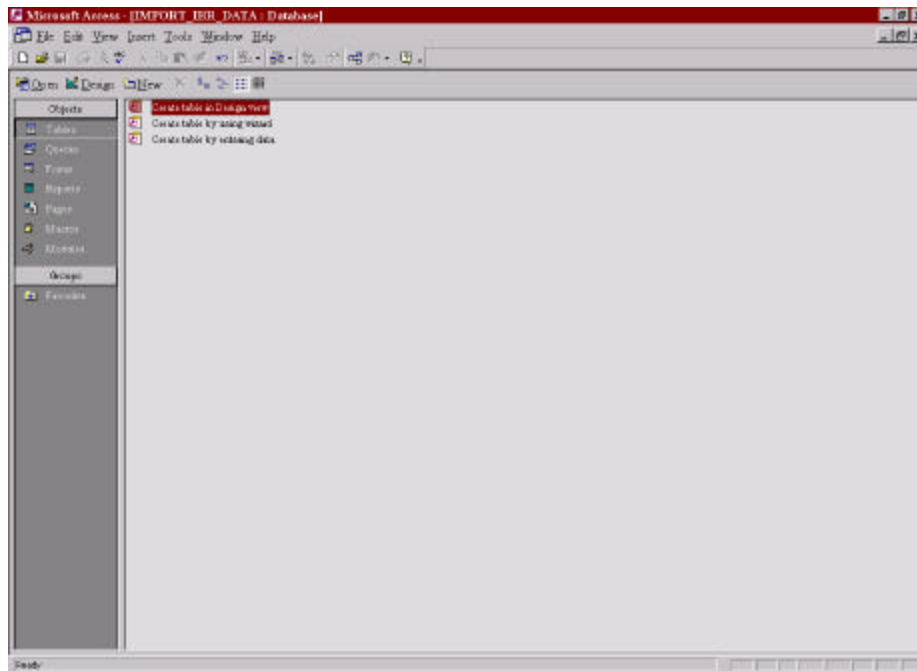
The Offense, Property, Victim, ArresteeA and ArresteeB tables, however, will require further manipulation because the data for these segments are organized in the IBRS flat file in a way that results in repeating fields in each table. Using the general procedure described above, these five tables will be normalized, resulting in 11 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
- [PROPERTY\\_DESCRIPTION](#): from the 50 "property description," "property value" and "property recovery date" fields originally in table PROPERTY
- [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
- [ARRESTEEB\\_WEAPON](#): from the 2 "arrestee armed with" and 2 "automatic weapon indicator" fields originally in table ARRESTEEA

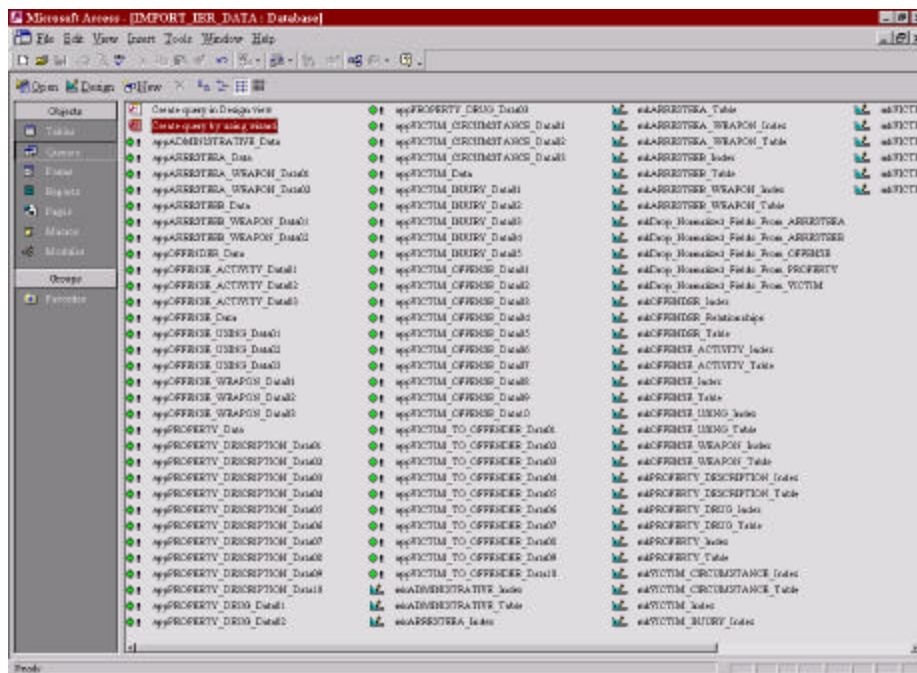
The fields that are stored in these 11 additional tables are then deleted from original tables Offense, Property, Victim, ArresteeA and ArresteeB.

Now we will take a brief tour of IMPORT\_IBR\_DATA00.mdb.

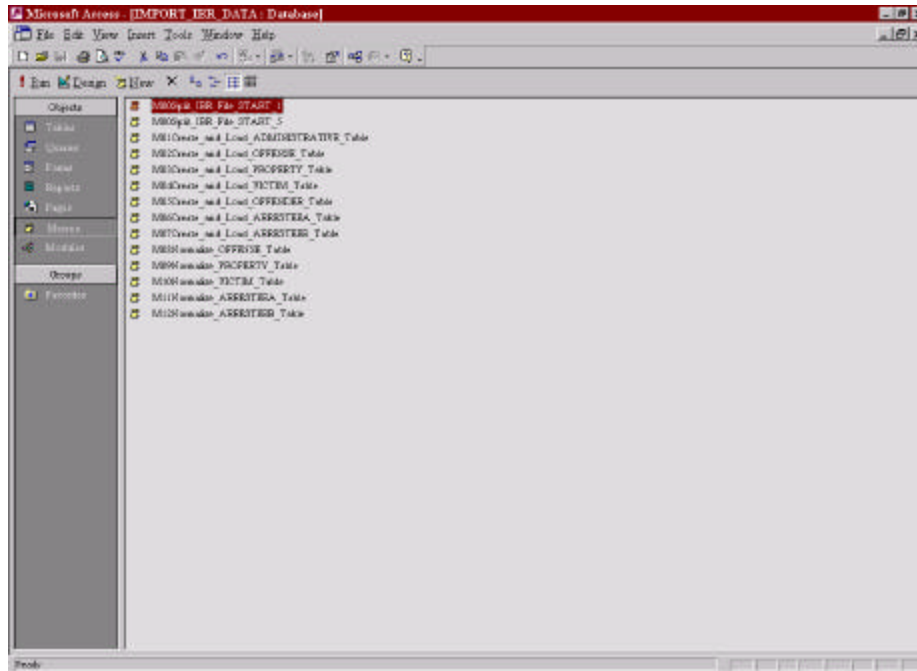
Here is the database table window before the IBRS data is imported into tables:



Here is the database query view showing the names of the 103 queries:



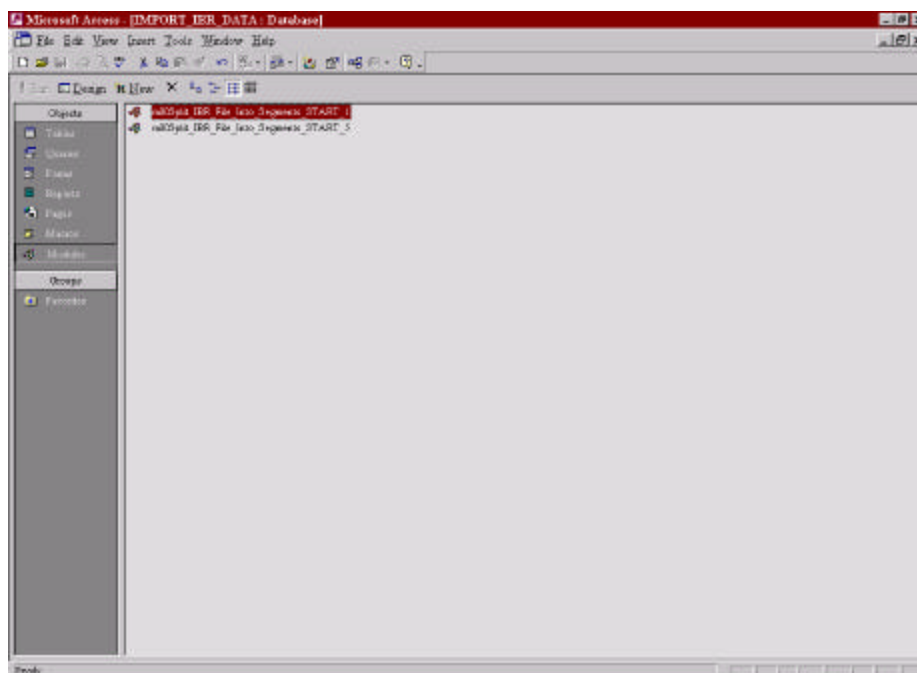
The database macro view:



Each macro calls sets of related queries shown in the query view above.

Macros M00Split\_IBR\_File\_START\_1 and M00Split\_IBR\_File\_START\_5 call the respective programs shown in the database module view below.

Database module view:

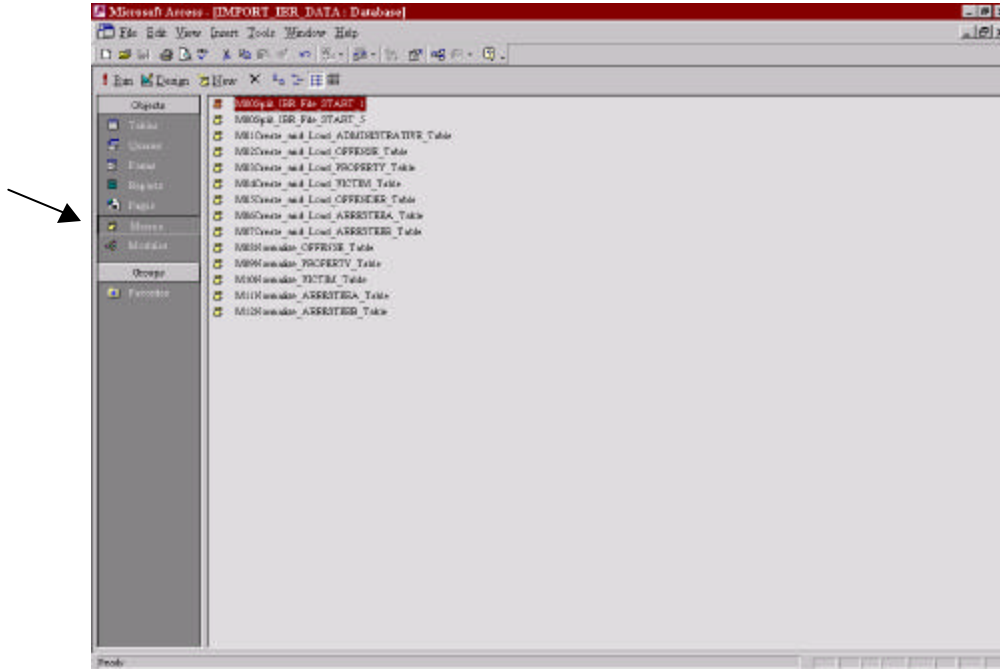


If you would also like to view the text link specification, [follow this link](#) for instructions.

### **Step-by-Step Instructions For Importing IBRS Data Into MS Access 2000 Using IMPORT\_IBR\_DATA.mdb**

1. Verify that you have the segment-level record layouts for your local IBR flat file.
2. Download the document "Uniform Crime Reporting, National Incident-Based Reporting System, Volume 2, Data Submission Specifications, May 1992" from the Web at [www.fbi.gov/ucr/ucr.htm](http://www.fbi.gov/ucr/ucr.htm). This volume contains the record layouts for local IBRS data submissions to the NIBRS. The record layouts there show how each IBR data element should follow the other in a specific order. The start position of the first IBR data element in the record, however, may vary between local IBRS files. Some local agencies may produce an IBRS flat file with the data beginning at position 1 in the data record. Other agencies may produce an IBRS flat file with the data beginning at position 5 in the record.
3. Compare your local IBRS flat file segment-level record layouts with the record layouts on pages 87-99 of the Data Submission Specifications document that you downloaded in step 2 above. Note whether the data elements in your local IBRS file record layout are ordered exactly as the data elements in the record layouts on pages 87-99. Also note in your local IBRS record layout whether or not the data element called "Level" or "Segment" or "Record ID" falls at position 1 or position 5 in the record. This shift of information in the data record will be taken into account when importing the data into database tables. *If the record layout of your local IBRS data file varies in any other way from the record layouts on pages 87-9 - for example, your local file contains additional data elements within the data record or the start position of the "Level" data element in the local file is other than position 1 or position 5 - then you cannot use IMPORT\_IBR\_DATA.mdb to import the data.*
4. Verify that you have Microsoft Access 2000 installed on your PC.
5. Create a folder on your hard drive for the IBR file (in this example, we will create the folder on the "c" drive--C:/IBR 2001/). Copy the local IBR flat file into this folder. If the file has the extension .DAT, open the file in a text editor such as WordPad or Word and save the file as a .TXT file. If there are blank spaces in the file name, replace the blanks with underscores or remove the blanks altogether. For example, an IBR file called IBR Jan 2001.DAT would be renamed IBR\_Jan\_2001.txt or IBRJan2001.txt. Renaming the file and removing spaces in the file name is very important.
6. Download database file IMPORT\_IBR\_DATA.mdb from the JRSA site and save it in the folder you created in Step 5. Download the database file a second time and save it in the same folder. Rename the second file IMPORT\_IBR\_DATA\_BACKUP.mdb. *Work with one .mdb file and save the other as a backup.*
7. The first task is to split the IBR flat file. This database application includes a short program that will split the IBRS file into 7 IBR segment-level files.

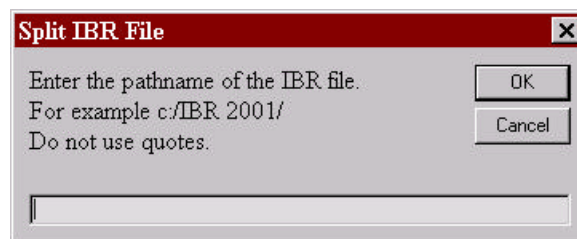
8. Open database IMPORT\_IBR\_DATA.mdb and select object button "Macros."



9. *If your local IBR file record layout shows data start at position 1 in the record, double-click macro M00Split\_IBR\_File\_START\_1.*

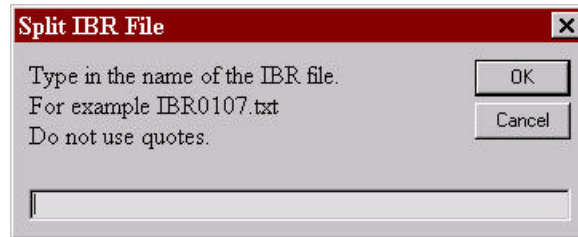
*If your local IBR file record layout shows data start at position 5 in the record, double-click macro M00Split\_IBR\_File\_START\_5.*

10. The first of two prompts asks for the pathname of the IBRS text file:



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

11. A second asks for the name of the IBR text file.



Enter the file name without using quotes. For example, IBR\_2000.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 IBR text file into 7 segment-level text files begins. This should take anywhere from 5 minutes to 15 minutes to complete depending on the speed and configuration of your computer and the size of the IBR file. An hourglass will appear in the middle of the current window indicating that the program is processing the IBR file. A message box will appear when the program is done:

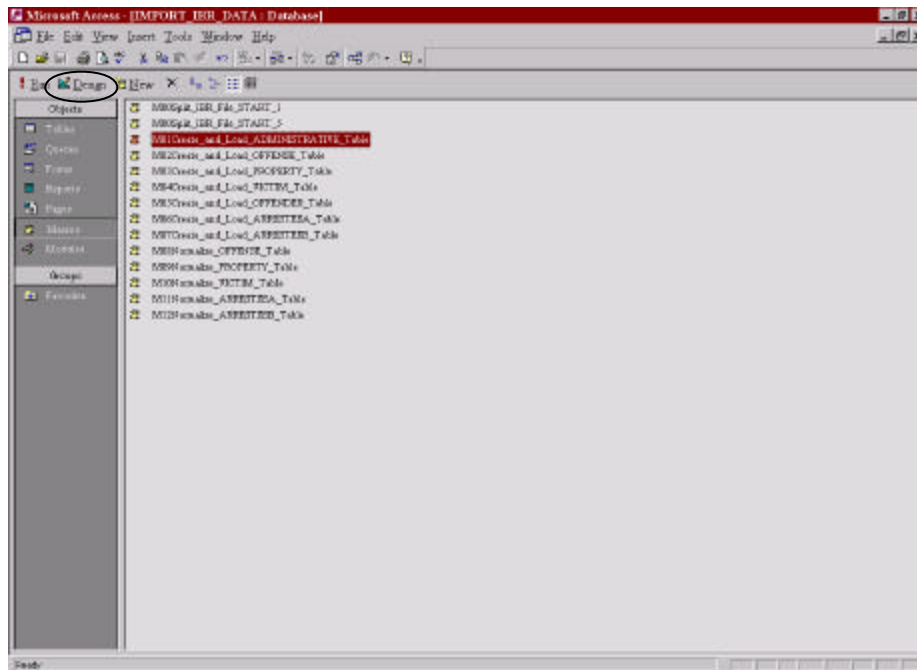


This box lists the name and location of the newly created segment-level files. The files are named according to the convention "SEGXX.txt", where "XX" corresponds to the IBRS segment level (01 = Administrative, 02 = Offense, 03 = Property, 04 = Victim, 05 = Offender, 06 = Arrestee Group A, 07 = Arrestee Group B). Make a note of these seven file names, then click "OK" or press "Enter."

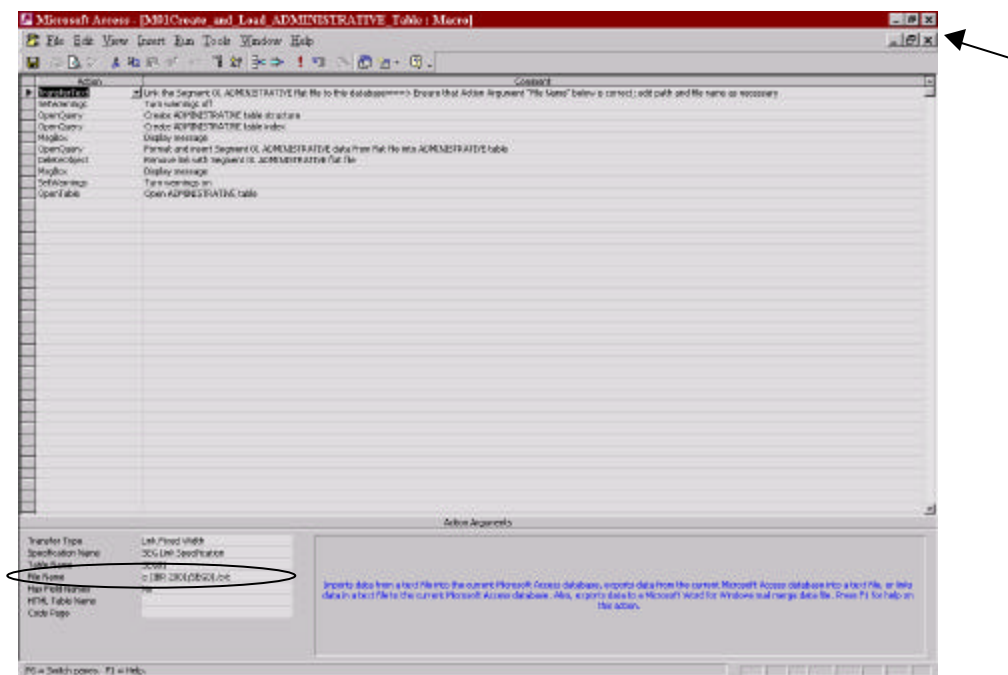
Now that the file has been split, we are ready to create each table structure, index, and relationship. We'll detail the process for the Administrative table.

## Create ADMINISTRATIVE Table

a) Highlight 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:



One item in this macro must be changed before the macro is run: File Name (circled). The path and filename of the split-out Segment 01 file should be entered here. C:/IBR 2001/SEG01.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."

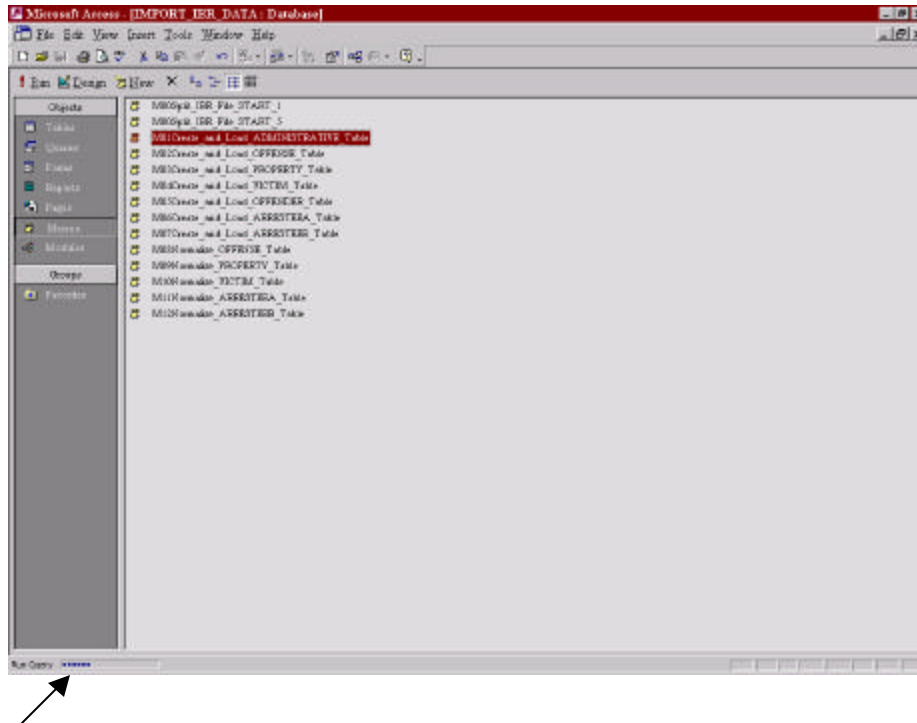
d) After the "Yes" button is clicked, the Macro objects window will appear (like the window in Step b above). Run the macro by double-clicking on macro M01Create\_and\_Load\_ADMINISTRATIVE\_Table.

e) The macro first links the Administrative (Segment 01) file c:/IBR 2001/SEG01.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:



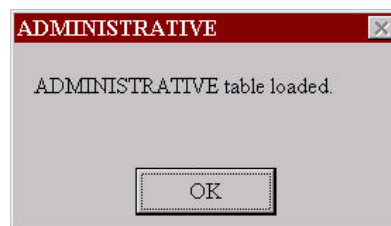
Click "OK" or press "Enter."

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 to 15 minutes to complete 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:

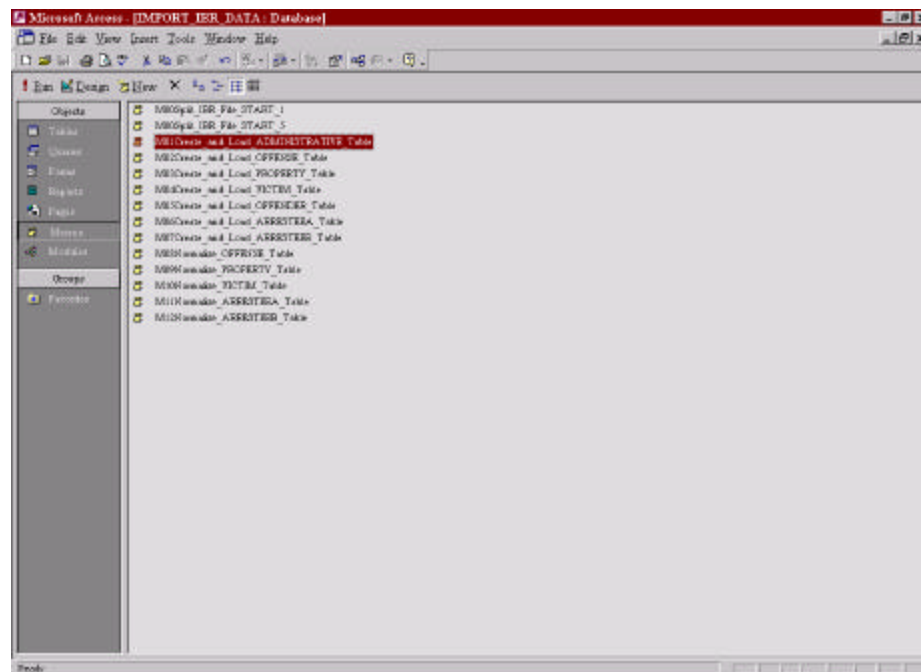


h) Click "OK" or press "Enter". The Administrative table will then open:

SEQ	UID	INC_YEAR	INC_MM	INC_DD	RPTDT_INDIC	INC_HR	CLR_EX	CLR_EX_YEAR	CLR_EX_MM	CLR_EX_DD
1	VA00100002000-006555	2000	5	1		12	B			
1	VA00100002000-010000	2000	11	29			N	2001	7	31
1	VA00100002001-001260	2000	12	5			N			
1	VA00100002001-005481	2001	6	18			N			
1	VA00100002001-005967	2001	6	21			N			
1	VA00100002001-006002	2001	7	1			N			
1	VA00100002001-006024	2001	7	1			N			
1	VA00100002001-006044	2001	7	2			N			
1	VA00100002001-006050	2001	6	28			N			
1	VA00100002001-006057	2001	7	2			N			
1	VA00100002001-006076	2001	7	3			N			
1	VA00100002001-006080	2001	7	2			N			
1	VA00100002001-006081	2001	7	2			N			
1	VA00100002001-006084	2001	7	2			N			
1	VA00100002001-006093	2001	7	3			N			
1	VA00100002001-006125	2001	7	3			N			
1	VA00100002001-006127	2001	7	4			N			
1	VA00100002001-006134	2001	7	4			N			
1	VA00100002001-006135	2001	7	4			N			
1	VA00100002001-006140	2001	7	4			N			
1	VA00100002001-006148	2001	7	4			N			
1	VA00100002001-006151	2001	7	4			N			
1	VA00100002001-006179	2001	7	5			N			
1	VA00100002001-006181	2001	7	4			N			
1	VA00100002001-006182	2001	7	4			N			
1	VA00100002001-006183	2001	7	5			N			
1	VA00100002001-006184	2001	7	5			N			
1	VA00100002001-006187	2001	7	5			N			
1	VA00100002001-006190	2001	7	5			N			
1	VA00100002001-006194	2001	7	5			N			
1	VA00100002001-006199	2001	7	4			B	2001	7	31
1	VA00100002001-006200	2001	7	5			N			
1	VA00100002001-006202	2001	7	3			N			

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

i) Close the Administrative table. You are automatically returned to the database "Macros" view:



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

M02Create\_and\_Load\_OFFENSE\_Table  
(Change File Name to c:/IBR 2001/SEG02.txt)

M03Create\_and\_Load\_PROPERTY\_Table  
(Change File Name to c:/IBR 2001/SEG03.txt)

M04Create\_and\_Load\_VICTIM\_Table  
(Change File Name to c:/IBR 2001/SEG04.txt)

M05Create\_and\_Load\_OFFENDER\_Table  
(Change File Name to c:/IBR 2001/SEG05.txt)

M06Create\_and\_Load\_ARRESTEEA\_Table  
(Change File Name to c:/IBR 2001/SEG06.txt)

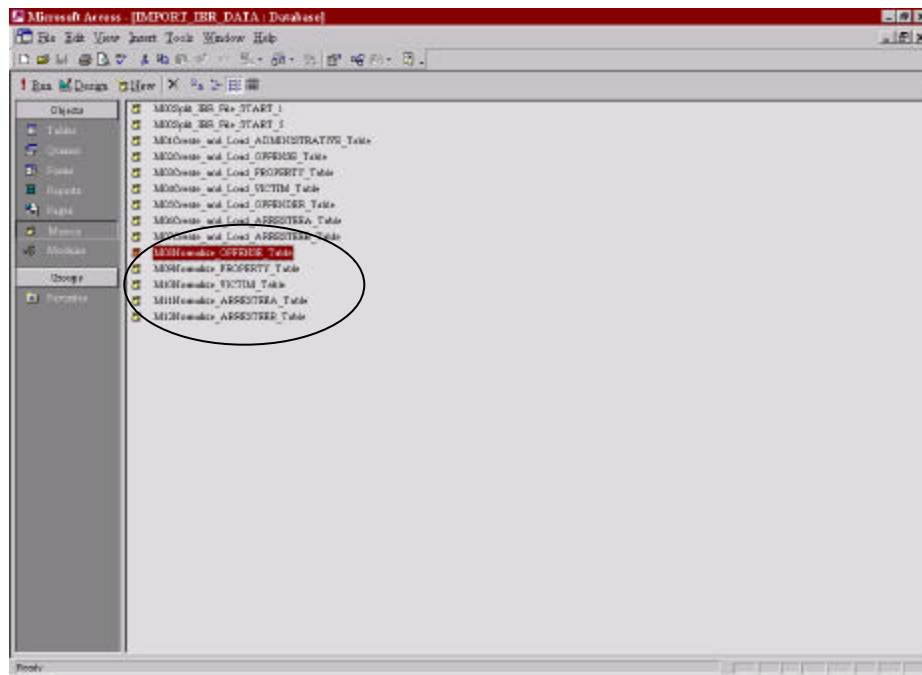
M07Create\_and\_Load\_ARRESTEEB\_Table  
(Change File Name to c:/IBR 2001/SEG07.txt)

*Run the macros in order M01 to M07. Running out of order could result in an error.*

The next task is to normalize the Offense, Property, Victim, ArresteeA and ArresteeB tables and create relationships between the tables.

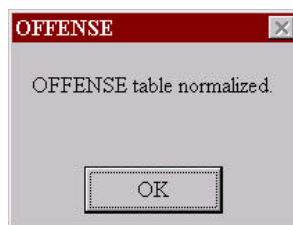
### **Normalizing The Tables and Creating Table Relationships**

The Offense, Property, Victim, ArresteeA and ArresteeB tables are normalized and relationships created by running 5 macros (circled):



M08Normalize\_OFFENSE\_Table  
M09Normalize\_PROPERTY\_Table  
M10Normalize\_VICTIM\_Table  
M11Normalize\_ARRESTEEA\_Table  
M12Normalize\_ARRESTEEB\_Table

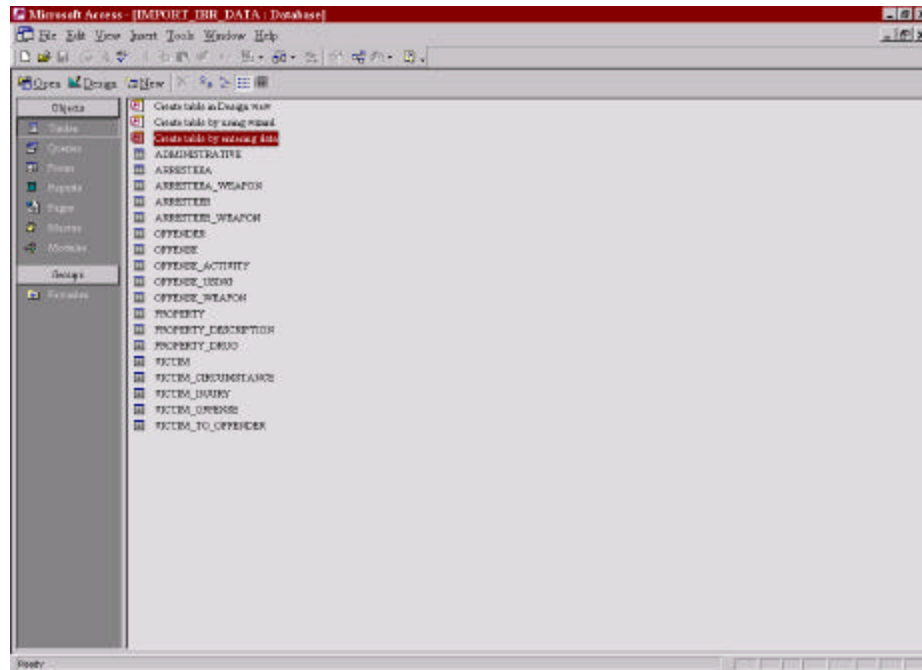
No changes to the normalizing macros are required. To run, simply double-click on the macro name. After each macro completes, a message box appears. The message box for M08 is:



A similar box appears after each macro M09 through M12 completes. Each normalizing macro should complete in anywhere from 1 to 10 minutes. It is important to *run the normalizing macros only after macros M01 through M07 have completed and only in this order--M08, M09, M10, M11, M12. Otherwise errors will occur.*

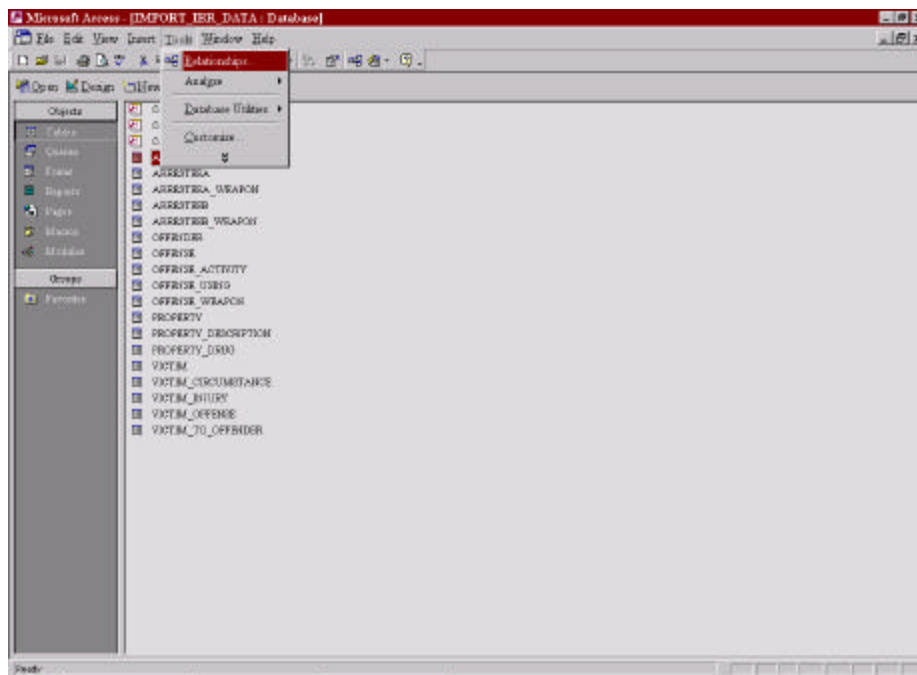
### **Final IBRS Tables**

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

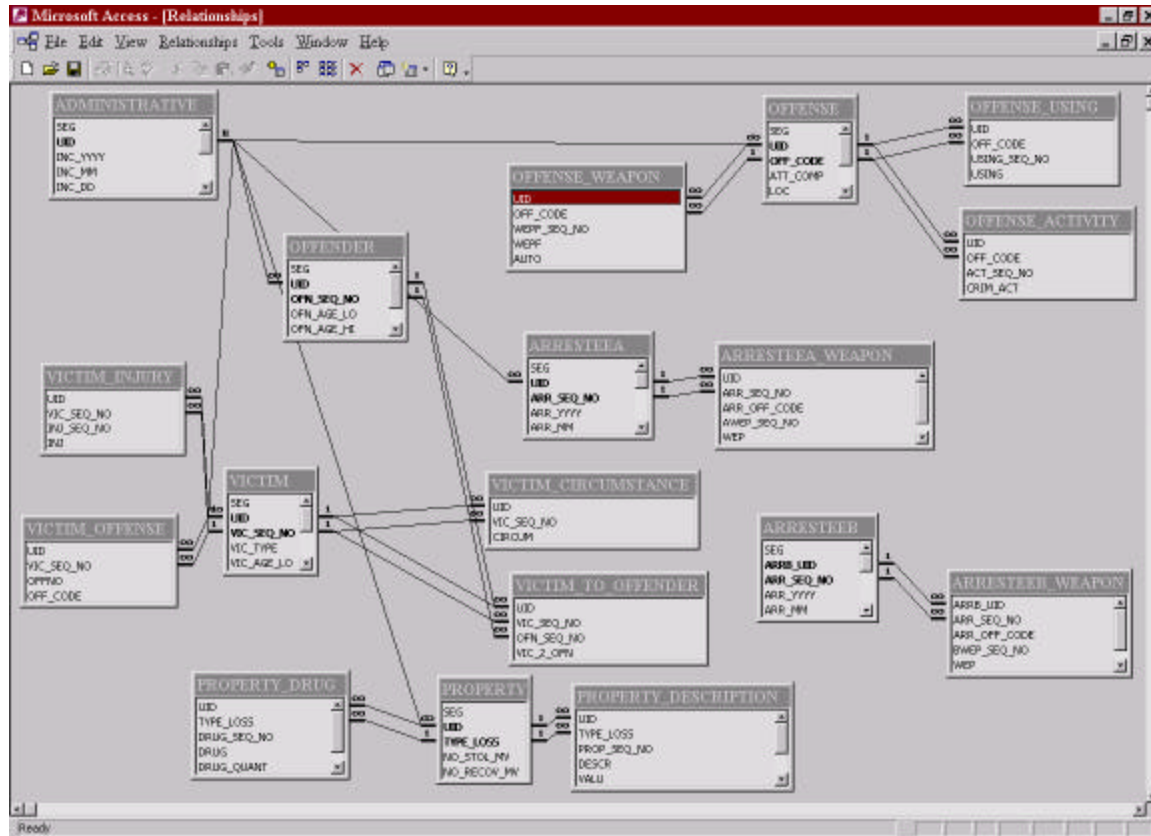


## View the Table Relationships

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 IBRS 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 IBRS 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 IBRS 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 represented 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 and Property\_Description.
- 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](http://www.jrsa.org/ibrc) 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](mailto:ibrc@jrsa.org) at [ibrc@jrsa.org](mailto:ibrc@jrsa.org).