Conversion of Blaise Databases to Relational Databases*

Yogeeta Purohit & Latha Srinivasamohan, U.S. Census Bureau

1. Introduction

This paper discusses the conversion process from Blaise to relational databases using the OLEDB workshop. It is possible to convert Blaise databases to relational databases and vice-versa using the Blaise Component Package (BCP). This functionality was introduced in Blaise 4.5. The OLEDB (Object Linking and Embedding Database) workshop tool is an enhancement to this concept. It is used to create the BOI (Blaise OLEDB Interface) files that are used as a "Datalink" utility. The OLEDB tool is a graphical and user-friendly system. This tool was introduced in Blaise 4.7 and its latest builds have greatly enhanced it.

The authors of this paper researched the BOI file technology to determine whether they could simplify the back-end processing of the survey data. The paper will compare two methods of a coding system. The first is the currently implemented method that uses only Blaise software. The second method is a proposed process that converts the Blaise database to a relational database. Using the second method, the clients would be provided the output in the resulting relational database.

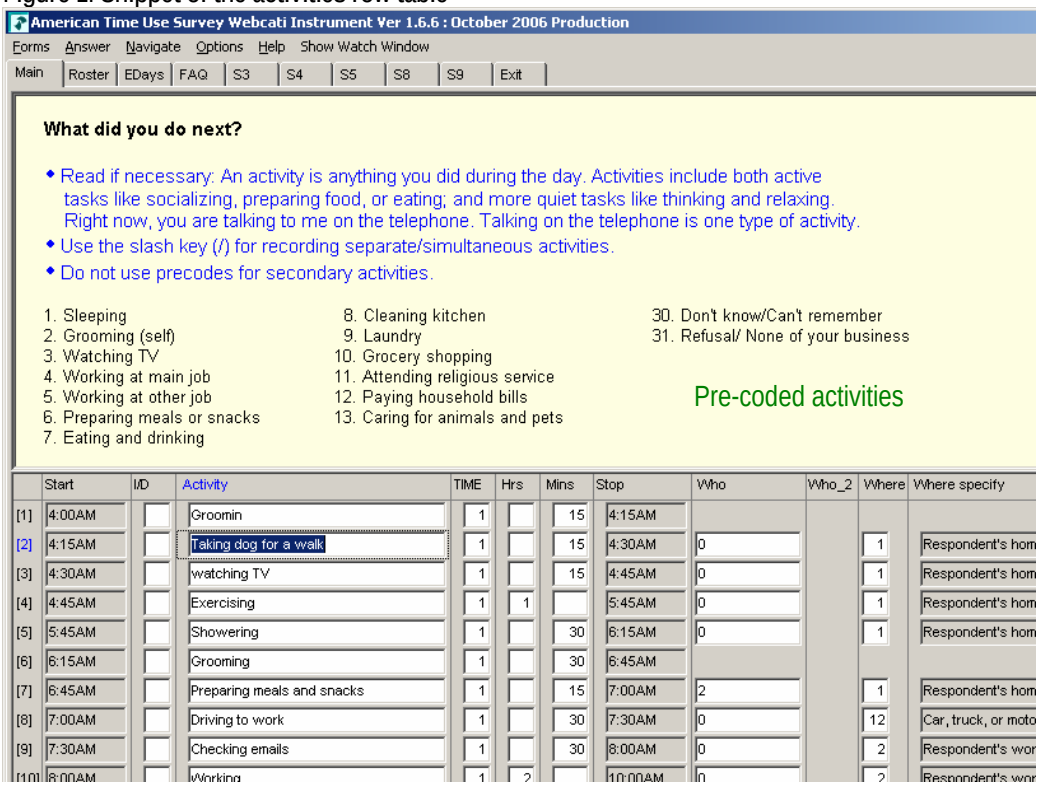
2. Survey Background

The American Time Use Survey (ATUS) was used as the sample survey for this research. This survey collects the time that a person spent on various activities in a 24-hour period. There is a limit of 90 activities reported per day. The interviewer can either pick the activity from the pre-coded list or record a new text entry, as shown in Figure 1. The interviewer has to record every activity during the day, so that every minute of a 24-hour period has been recorded for.

---

* ***Disclaimer***: *This document is to inform interested parties of ongoing research and to encourage discussion of Blaise instrument design issues. The views expressed are those of the authors and not necessarily those of the U.S. Census Bureau.*

Figure 1. Snippet of the activities row table



3. Current Process (Blaise to Blaise)

Once the interviewer collects the Blaise data, the next step is to code the activities for analysis and drawing statistics. This second stage process of assigning generic codes is known as the coding system, which is described here. The current process uses only Blaise DEP and Manipula scripts to code the activities.

After the survey interviewing is completed, data from the activities table is extracted and a subset of the main instrument data is created. Once the subset is created, various scripts are run to extract and load the activity data into another instrument, the coding instrument. Here the activities are coded using generic codes provided by the sponsors. Figure 2 shows the flow of the process currently implemented.

The sponsors provide the coding rules and specifications for the coding system. Each activity code is subdivided in a three-tier layout. See Figure 3 for a sample view of the generic code split-up. During the coding process, the pre-coded activities are assigned their corresponding codes automatically, and the non-precoded activities (i.e., text entries) are coded using the coding system.

Figure 2. Standard process flow

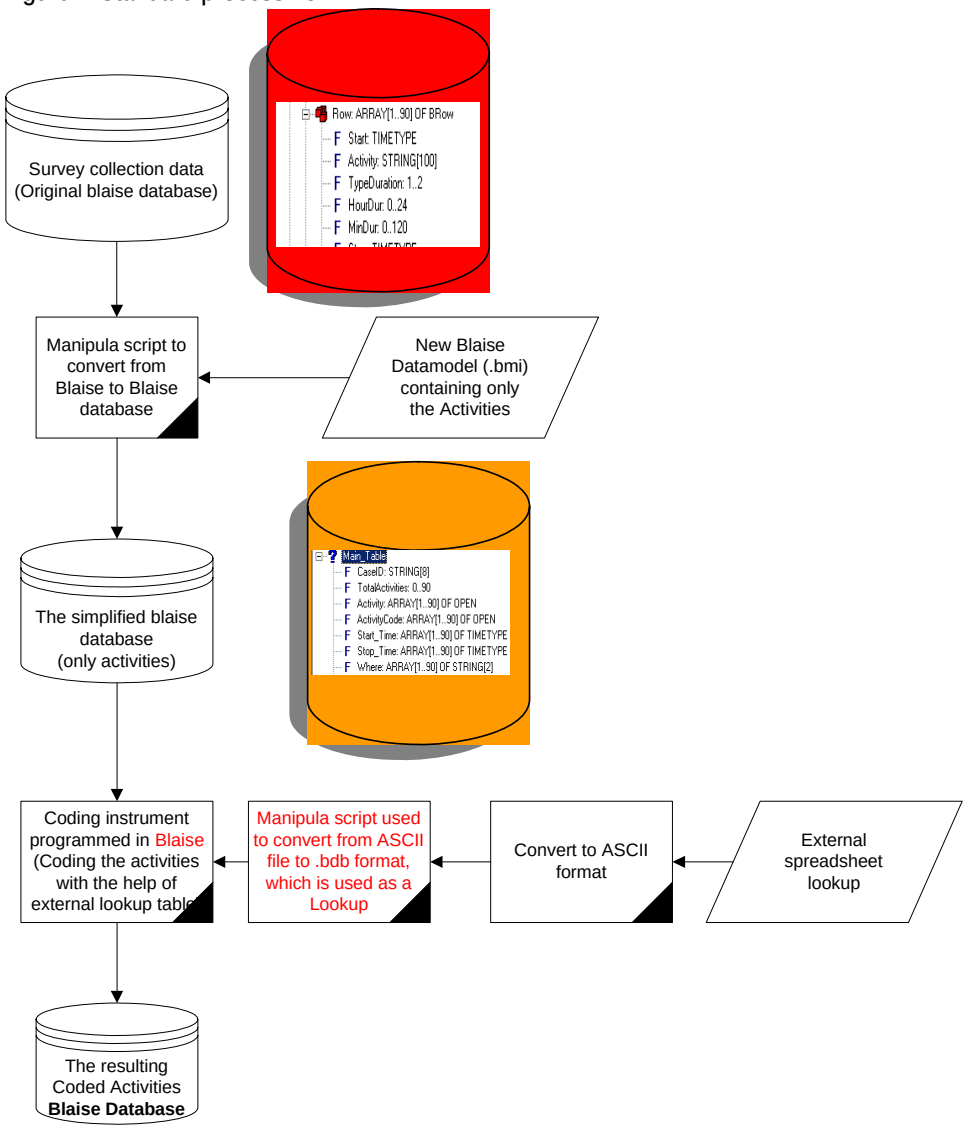
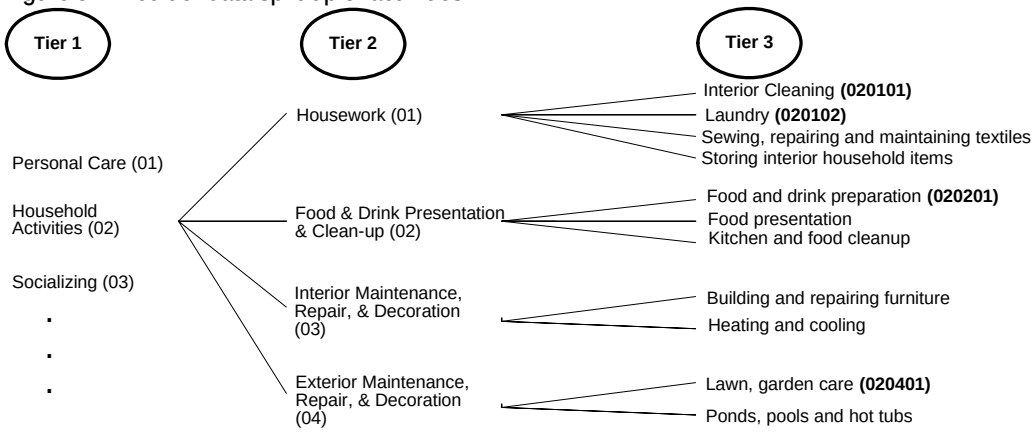


Figure 3. Three-tier data split-up of activities



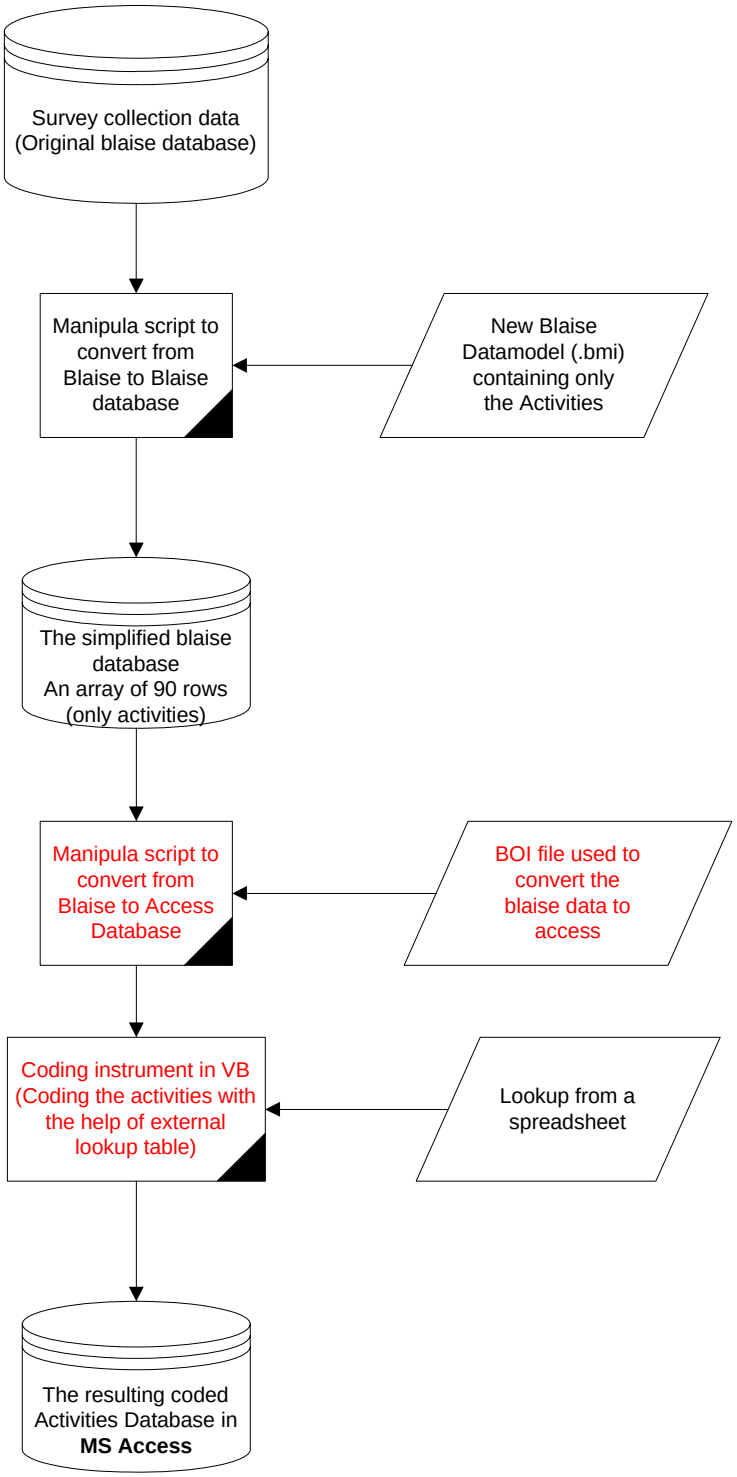
4. Alternative Process (Blaise to RDBMS)

Converting Blaise data into a relational database has been possible using earlier versions of Blaise, but it has not been simple. We will now look at the conversion of Blaise to a relational database management system (RDBMS) using the Datalink tool provided by Blaise 4.7.

The process copies the data from Blaise to a relational database using the BOI files. For our example, MS Access is the resulting relational database. The first step in the data conversion process is to create a subset of Blaise data. The reason for creating a subset is that in the original Blaise database the activities are embedded several layers deep within the blocks and tables[1]. The new Blaise database consists of a block with 90-row arrayed activities, as shown in Figure 4. Once the data subset is created, the next step is to convert it to MS Access using the BOI file. This resulting Access database is used as the data storage for the coding system. A Visual Basic application is used for the coding system. Figure 4 explains the flow of the process:

[1] The intermediate Blaise db is not required. It was used to simplify the conversion task. The new developments to the BOI file allow complicated data extractions.

Figure 4. Flow of the alternative process



5. Relational Database Extraction Process

Blaise is an effective front-end survey application. However, the client may prefer data in relational database format for the back-end data processing. In this section, the paper addresses the steps to create a BOI file and customize it using the OLEDB package provided by Blaise 4.7. This helps convert data in relational databases such as MS Access, Oracle, MS SQL Server, and so on to Blaise and vice-versa. In our example, we convert data from Blaise to a MS Access database.

The first step is to create a BOI file. The OLEDB Toolbox (shown in Figure 5) is used to setup a BOI file based on the user requirements for the resulting relational database structure. The basic components needed to build the file are a .bdb (Blaise database) and its corresponding .bmi (Blaise data model).

1. Select the option to create the BOI file as shown in Figure 5. The OLEDB Toolbox wizard appears as shown in Figure 6.

Figure 5. OLEDB Toolbox

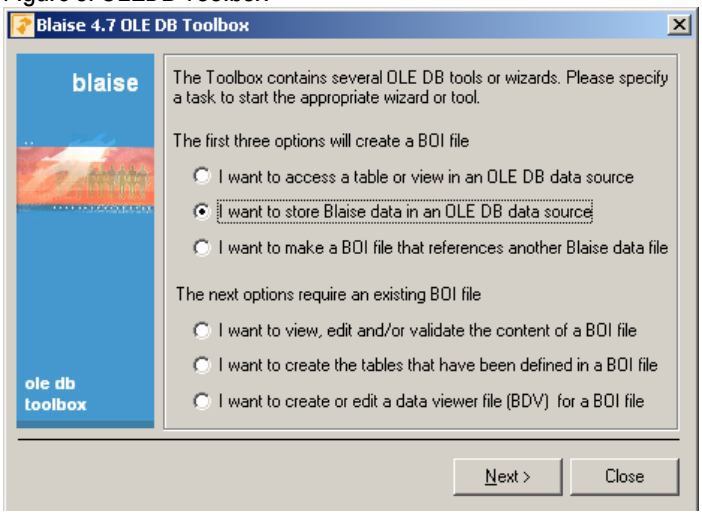
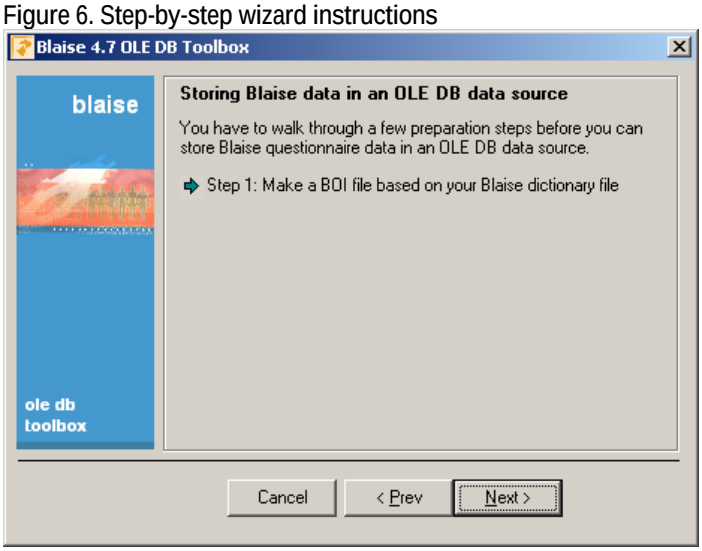
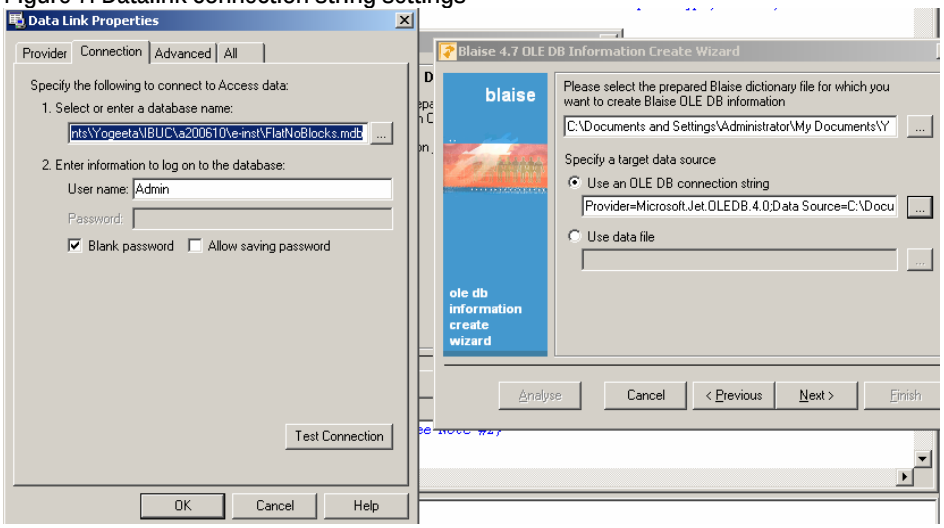


Figure 6. Step-by-step wizard instructions



2. The next step is to pick the OLEDB connection string for the database to be converted. Here the user selects the OLEDB provider needed for the final database. For instance, we use the 'Microsoft Jet OLEDB 4.0' provider to connect to MS Access database. The properties for the relational database are set at this point, such as password protection, read/write mode, etc.

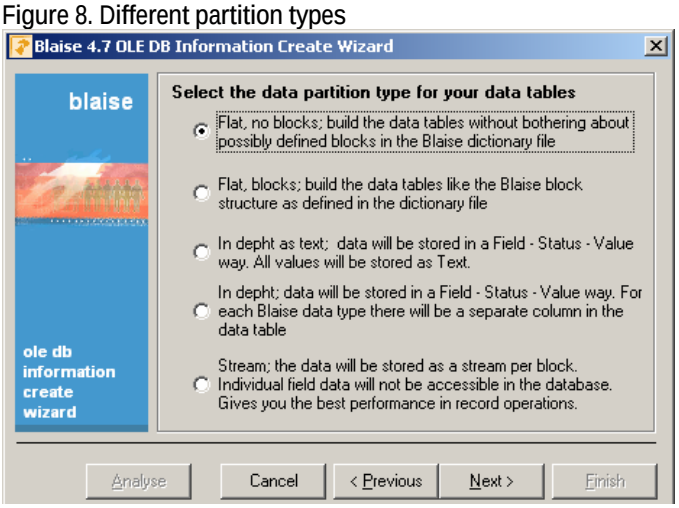
Figure 7. Datalink connection string settings



The OLEDB transfer provides several methods of dividing the data in the resulting database. The provider along with the BOI file helps setup the tables and the data connections. We selected the 'Flat, no blocks' as shown in Figure 8. This option transfers the data from a Blaise **field** to relational database **column** (Figure 10). MS Access 2000 has a limit of 255 columns per table. Therefore, once the limit is reached, a new table is automatically created and the remaining columns roll over to the new table. This process continues until all the fields from the Blaise database are transferred.

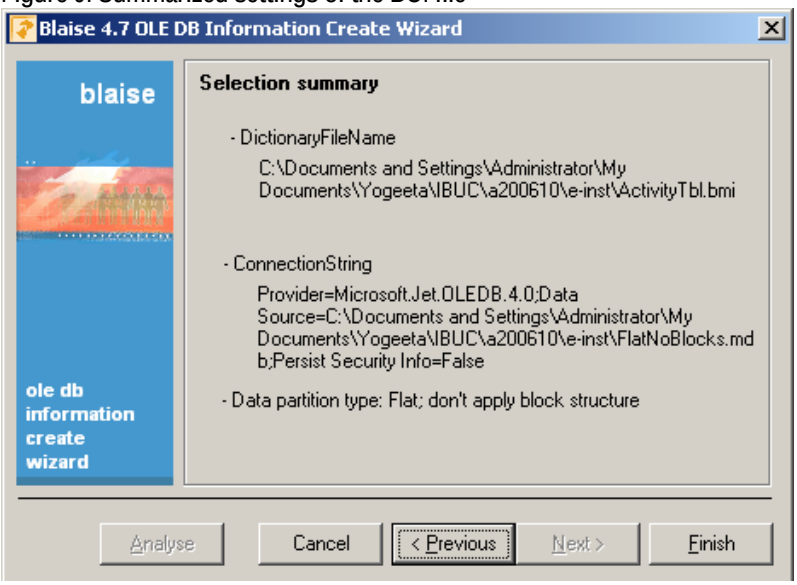
Since we did not have the need to set the secondary keys in the resulting database, we used the 'Flat, no blocks' option. The 'In depth' option is better suited for conversions that require secondary keys in the resulting database. As mentioned in Mr. Arno Rouschen's abstract, "Generic Data Storage with Help from Blaise 4.8 Datalink", primary and/or secondary keys are used to normalize the data, which leads to low database administration.

Figure 8. Different partition types



We have now created the BOI file, and a summary of its setting is shown in Figure 9.

Figure 9. Summarized settings of the BOI file



3. Once the BOI file is created, proceed to customize it using the 'Creating the table' wizard.

Note: The settings of the BOI tables can be either modified using the wizard or by using the OLEDB File Manager.

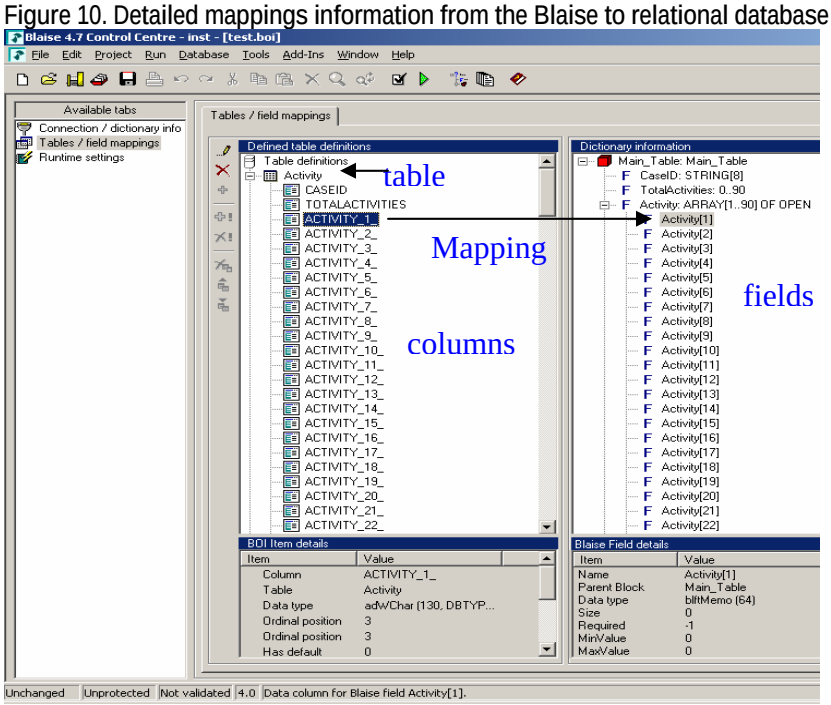
A few common examples in which a BOI file can be customized are given below:

a. The connection string to the relational database can be reset.

b. The connections between the Blaise fields and Access columns can be established. Figure 10 gives a good graphical view of this process. These are known as mappings. For setting up a better relational database

structure, the columns are moved around based on their relevancy to the tables. This improves the data accessibility by the back end application. Mappings can be added and deleted.

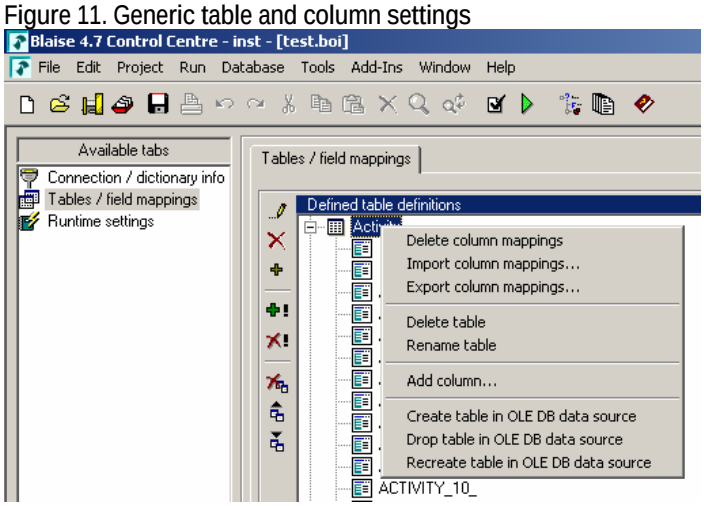
Figure 10. Detailed mappings information from the Blaise to relational database



c. The settings of the tables can be updated in the BOI file. The list of different table functions is shown in Figure 11. Similarly, the column settings can be updated.

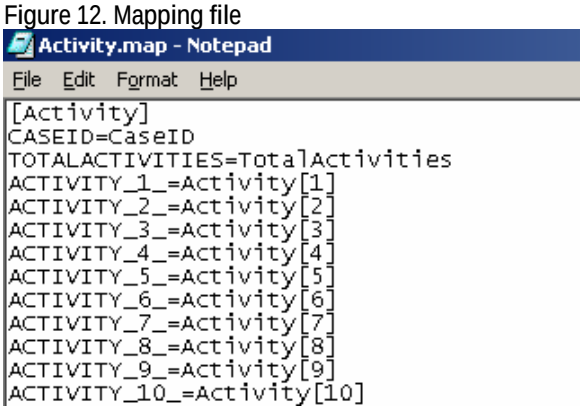
Note: The tables and the columns listed in the BOI need to be (re)created in the OLEDB. The BOI settings just provide an overview of the resulting database. The settings do not apply to the relational database until the tables are recreated. (Caution: Recreating the table wipes out the data in the corresponding current relational database table.)

Figure 11. Generic table and column settings



d. The table and the column mappings shown in Figure 10 can be imported or exported to a file. A snippet of this file is shown in Figure 12. It is helpful to create a mapping file in case the table settings need to be reused or in case the BOI file gets corrupted.

Figure 12. Mapping file




```
[Activity]
CASEID=CaseID
TOTALACTIVITIES=TotalActivities
ACTIVITY_1_=Activity[1]
ACTIVITY_2_=Activity[2]
ACTIVITY_3_=Activity[3]
ACTIVITY_4_=Activity[4]
ACTIVITY_5_=Activity[5]
ACTIVITY_6_=Activity[6]
ACTIVITY_7_=Activity[7]
ACTIVITY_8_=Activity[8]
ACTIVITY_9_=Activity[9]
ACTIVITY_10_=Activity[10]
```

4. After creating the BOI file, the next step is to process it. A Manipula script is used to transfer the Blaise data to the Access database. The conversion in this case is record-oriented, which means it will convert the data record by record. This can be a time consuming conversion.

   Note: In Blaise 4.7.1122 the BOI file can be customized for a set-oriented conversion. A drawback of this conversion method is that it converts databases consisting of only one data table. This approach is faster and efficient for copying data after the completion of data collection process. On the other hand, the record-oriented copy is helpful in a client-server environment since it locks only one record at a time. This issue needs further research.

Figure 13. Manipula conversion script

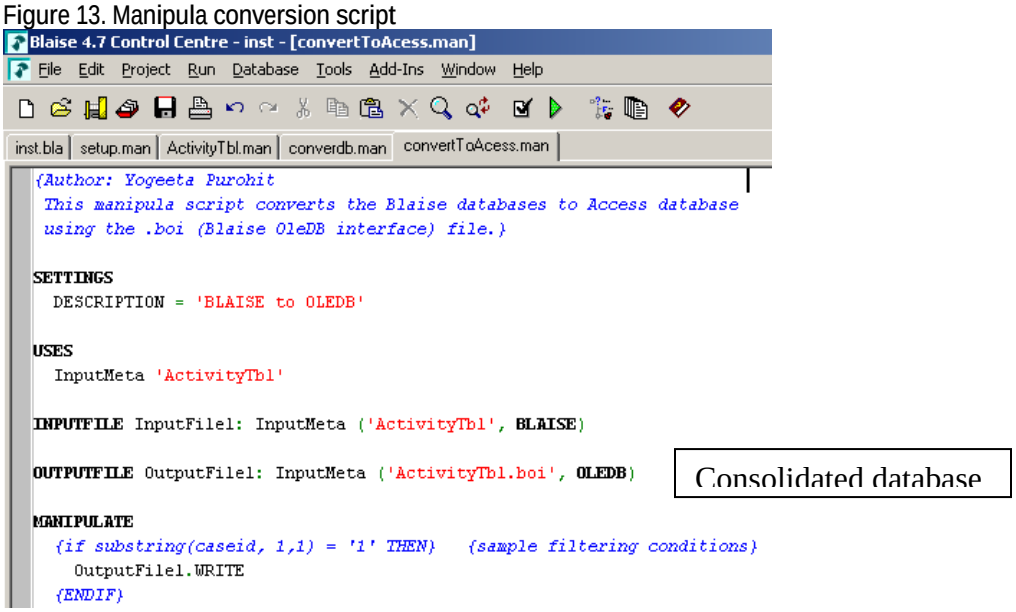
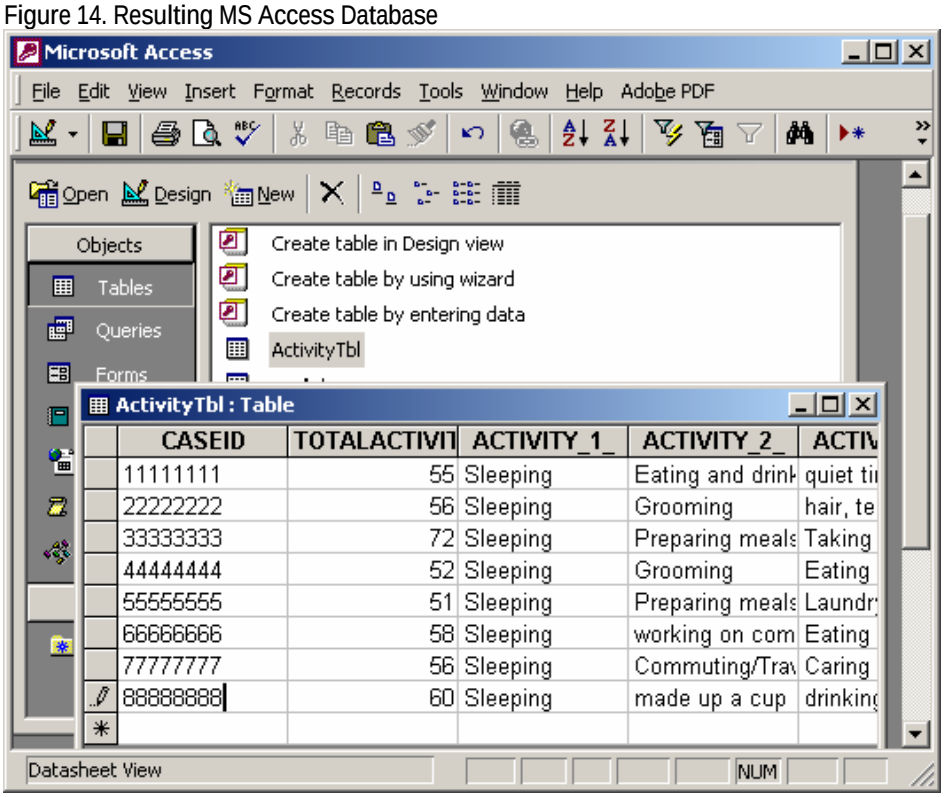


Figure 14 shows a snippet of the resulting MS Access database.

Figure 14. Resulting MS Access Database



5. The resulting relational database is used by the second stage application for coding.

   In our example, the Access database is used in a Visual Basic application. From this point onwards, the application requires no connection to the original Blaise

database. The Access database is updated with the user's entries in the application. This completes our coding system and the resulting output is stored in MS Access database.

6. Overview of Benefits and Drawbacks

This section identifies some benefits and drawbacks of the BOI files. These should be considered based on the client requirements.

6.1 Benefits

- The use of the BOI files allows the user to store data in a more normalized relational database.
- Better data administration may be provided using a true database.
- A subset of Blaise data could be shipped as well to protect sensitive data.
- BOI files are suited for backward compatibility. That is, the BOI functionality can be implemented on surveys built in Blaise versions 4.7, 4.6 and 4.5.
- BOI files do not require an additional programming language for the conversion.
- BOI files can be password protected, better in the client-server environment.
- The connectivity in the latest version of Blaise 4.7 has been improved.

6.2 Drawbacks

- Relational databases require more maintenance.
- All relational databases have a set limit on their number of columns and total record length. Therefore, it is essential to take into consideration these aspects while dividing the data in the BOI file.
- The BOI file adds an extra layer to the transformation as compared to the BCP, and therefore adds a delay to the data copy from Blaise to a relational database. (Refer to 'Listing Part 2 – Using Blaise 4.7 for Coding Listing Instruments' paper IBUC conference, 2006)

7. Technical Details

Blaise databases can be converted to relational databases provided there exists an OLEDB (Object Linking and Embedding Database) provider or ODBC (Open Database Connectivity) driver. This information is stored in the file called oledb.dbi.

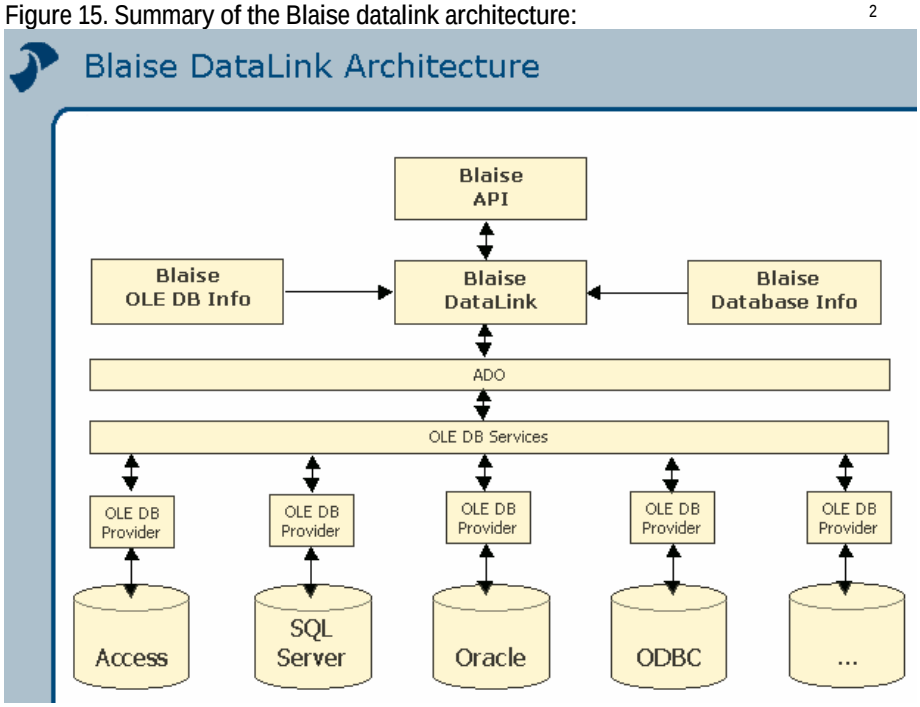
An OLEDB provider is a software component that provides an interface for converting data to or from an OLEDB. Some of major OLEDB providers are Microsoft, Oracle and Sybase. Here is a list of the providers that are currently supported by Blaise:

- Microsoft Jet 4.0 OLE DB Provider
- Microsoft OLE DB Provider for DB2
- Microsoft OLE DB Provider for ODBC
- Microsoft OLE DB Provider for Oracle
- Microsoft OLE DB Provider for SQL Server
- Oracle Provider for OLE DB
- SAS IOM Data Provider
- SAS/Share Data Provider
- Sybase Adaptive Server Anywhere Provider

An ODBC driver is an interface between an application and the DBMS (Database Management System), which translates the data to graphical format. The user easily understands this format. Some of the commonly used ODBC drivers that are currently supported by Blaise are:

- Microsoft Access Driver (*.mdb)
- Microsoft dBase Driver (*.dbf)
- Microsoft Excel Driver (*.xls)
- Microsoft ODBC Driver for Oracle
- INTERSOLV 3.10 32-BIT SequeLink
- Microsoft ODBC for Oracle
- MySQL
- MySQL ODBC 3.51 Driver
- Oracle ODBC Driver
- Oracle73
- SAS
- SQL Server
- SPSS 11.0.1 32-BIT Data Driver (*.sav)

Figure 15. Summary of the Blaise datalink architecture: [2]



In order to use the BOI files, we need to have Blaise 4.7 work with the BCP 2. For that, the following files need to be registered:

- **BlAPI4A2.dll** - Blaise API Objects Library
- **BlBOI4A2.dll** - Blaise OLE DB Interface Component
- **BlDBI4A2.dll** - Blaise Database Information Component
- **BlDL4A2.dll** - The Blaise DataLink Component (used for reading and writing from the OLEDB data sources)
- **SnReg1A.dll**
- **SiStrClA.dll**
- **OLEDB.dbi** (Can not be registered, but must be in the same folder as BlDBI4A2.dll) – This file contains the OLEDB provider and ODBC driver information that are used with the Datalink. New relational database information can be added to this file in order for them to work with the BOI.

[2] Figure taken from the Blaise Reference Manual.

8. Some Useful Observations

1. The database software does not need to be installed as long as the OLEDB provider is available for data transfer. In our example, while creating the .mdb file, only the MDAC component package was installed, the MS Office software was not required. This is helpful in scenarios where the user does not need to look at the data in its original form, and can just have access through applications such as Visual Basic, C++, Java, etc.

2. The Datalink package can even work with non-relational data-files such as spreadsheets. A classic example of this is the conversion of an .xls file into a .bdb. The .bdb can be used for lookups and trigram searches in the Blaise instrument.

3. The "**create table wizard**" allows the user to create SQL scripts, which are used to set up the different relational databases. An example of this code is shown in Figure 16.

Figure 16. Sample SQL code:

ActivityTbl.sql - Notepad

File Edit Format View Help

```
`CASEID` VARCHAR(8) NOT NULL,
`BLOCKNAME` VARCHAR(255) NOT NULL,
`STATUS` INTEGER,
`CHECKSTATUSINFO` VARCHAR(255))

ALTER TABLE `ACTIVITYTBL_BLOCK` ADD PRIMARY KEY (`CASEID`,`BLOCKNAME`)

CREATE TABLE `ACTIVITYTBL_FORM` (
`CASEID` VARCHAR(8) NOT NULL,
`JOINKEY` INTEGER,
`STATUS` INTEGER)

ALTER TABLE `ACTIVITYTBL_FORM` ADD PRIMARY KEY (`CASEID`)

CREATE INDEX `ACTIVITYTBL_FORMSTATUS` ON `ACTIVITYTBL_FORM` (`STATUS`,`CASEID`)

CREATE TABLE `ACTIVITYTBL_MAIN_TABLE1` (
`CASEID` VARCHAR(8) NOT NULL,
`BLOCKNAME` VARCHAR(255) NOT NULL,
`INSTANCE` INTEGER NOT NULL,
`TOTALACTIVITIES` INTEGER,
`START_TIME` DATETIME,
`CO0001` DATETIME,
`CO0002` DATETIME,
`CO0003` DATETIME,
`CO0004` DATETIME,
`CO0005` DATETIME,
`CO0006` DATETIME,
`CO0007` DATETIME,
`CO0008` DATETIME,
`CO0009` DATETIME,
`CO0010` DATETIME,
`CO0011` DATETIME,
`CO0012` DATETIME,
```

9. Conclusion

We have reviewed two methods of creating a coding system. Both of these methods are efficient and can help different clients meet their requirements. The difference in the processes lies in the final output of the coding instrument. The alternative process provides data in a format that is easily accessible to the clients. The end product can be immediately put to use. Blaise Datalink has significantly simplified the task of transferring data from Blaise to RDBMS and vice-versa, which may help better serve the needs of clients.

10. Acknowledgments

We would like to acknowledge some people from the authoring staff, who provided valuable feedback on this paper. Our special thanks go to Karen Bagwell, Tom Spaulding and Rob Wallace for their valuable recommendations on the paper. Also, thanks to Michael Mangiapane, Roberto Picha, Ed Dyer and Leslie Fleet for their feedback.

11. References

Blaise 4.7 Reference Manual

Rouschen, A; Statistics Netherlands (2007): Generic Data Storage with Help from Blaise 4.8 Datalink, Abstract for the 11th International Blaise Users Conference 2007, Annapolis, MD, 2007.

Wallace, R., Picha, R., Mangiapane, M.; US Census Bureau, USA (2006): Listing Part 2 - Using Blaise 4.7 for Coding Listing Instruments, Statistics Netherlands, Presented at the 10th International Blaise Users Conference 2006, Netherlands, 2006.

Wikipedia.org. OLE DB and OLE DB Providers.
16 Jul. 2007 < http://en.wikipedia.org/wiki/OLE_DB >.

Webopedia.com. ODBC.
16 Jul. 2007 <http://www.webopedia.com/TERM/O/ODBC.html>.