BLAISE DATALINK

IBUC 2010 – Pre Conference workshop

Overview + 4.8.2 new features

Agenda

- Datalink overview
  - History, Concepts
- 4.8.2 Changes and new features
  - OLE DB Runtime Settings
    - Extensions made to Manipula and API
    - Commandline options
  - OLE DB Data storage
    - Changes in table structures and behavior
    - Record stream synchronization
      - Synchronization methods and scenarios
  - Data Centre
    - Overview functionality and new features

Datalink Overview

Basic concepts

Blaise Datalink

- Defines a link to data that is stored elsewhere
  - Oracle, SQL Server, MySQL, MS Access ...
- Based on Microsoft OLE DB technology
- Uses a Blaise OLE DB Interface file
  - Extension .BOI
  - Data file within Blaise
  - Can be used with all Blaise tools, like DEP, Blaise Internet, Manipula, Data Viewer

Blaise Datalink History

- Blaise 4.5 / BCP 1 (2001)
  - Existing table
- Blaise 4.6 / BCP 2 (2003)
  - Existing table / based on Blaise data model
- Blaise 4.7 Enterprise (2005)
  - Customizable BOI files, select statements, referencing other data files
- Blaise 4.8 Enterprise
  - 4.8.0 (2007): JOINKEY support, Data Server
  - 4.8.1 (2008): Generic storage, Versioning
  - 4.8.2 (2009): Stream synchronization, Runtime Settings

Blaise OLE DB Tools

- Blaise OLE DB Toolbox (since 4.6)
  - Blaise OLE DB Mapping Wizard
  - Blaise OLE DB Interface Create Wizard
  - Blaise OLE DB Tables Create Wizard
- Blaise OLE DB Command Builder (4.6)
- Blaise OLE DB Workshop (4.7)
  Blaise Data Centre (4.8.1)
- Record Stream Synchronization Tools (4.8.2)
  - Data Consistency Wizard in Data Centre
  - Stream Synchronization Service + Admin Tool

BOI files

Blaise OLE DB Interface

- Data file type available in Blaise
  - Extension .BOI
- Contains logical definitions; no data
  - Connection information
  - Database tables to be accessed
  - Field mappings to be used
  - Runtime settings and more
- Has an associated Blaise meta file (.bmi)
  - No data without meta data

BOI file types

- Simple (data only) BOI file
  - Based on existing database table/view or SQL query
  - Contains just one table definition
  - Can be used for lookups or exports
- Full BOI file
  - Based on existing Blaise data model
  - BDB equivalent; stores all data, including status, remarks, error information, etc
  - Can be used to store questionnaire data in external db
- BOI referencing another data file
  - Links to BDB, another BOI or an ASCII file
  - Used in combination with Blaise Internet and/or Data Server

Demo: Creation of a simple BOI file

Data Partition Types

Applies to full BOI files

Data Partition Types

- Determines how field data will be stored in the OLE DB data source
- Distinction between system and data tables
- Affects only the structure of the 'real' data tables
- Optimal type is determined by
  - Structure of data model
  - Data analysis objectives during collection process

Data Partition Types

- System tables
  - Tables that contain info needed by Blaise
  - Data partition type independent structure
  - DictionaryInfo, CaseInfo, FormInfo, KeyInfo, Open, Remark, IDInfo
- Data tables
  - Tables where the field data is stored
  - Structure is determined by data partition type
    - Flat, no blocks; all end data fields will have a column
    - Flat, blocks; table for each block type
    - In depth; field, status, value; each data type has its own column
    - In depth text; field; status; value as text
    - Stream; stream of data per record

Demo: Creation of a full boi file (MS Access)

Generic boi files

Applies to full boi files; available since 4.8.1

Generic BOI files

- Goals
  - To store questionnaire data of multiple surveys in a centralized input data store
  - To share database tables as much as possible between surveys
  - To reduce the number of database tables
  - To provide generic table structures and as a result uniform data access

Non-generic and Generic boi files Different, but not that different…

Non Generic

- Primary key Blaise Data Model
- Tailor made for data model

Generic

- Common primary key
- Two additional tables
  - Dictionary Info
  - Case Info
- Flat data tables are not generic
- Versioning

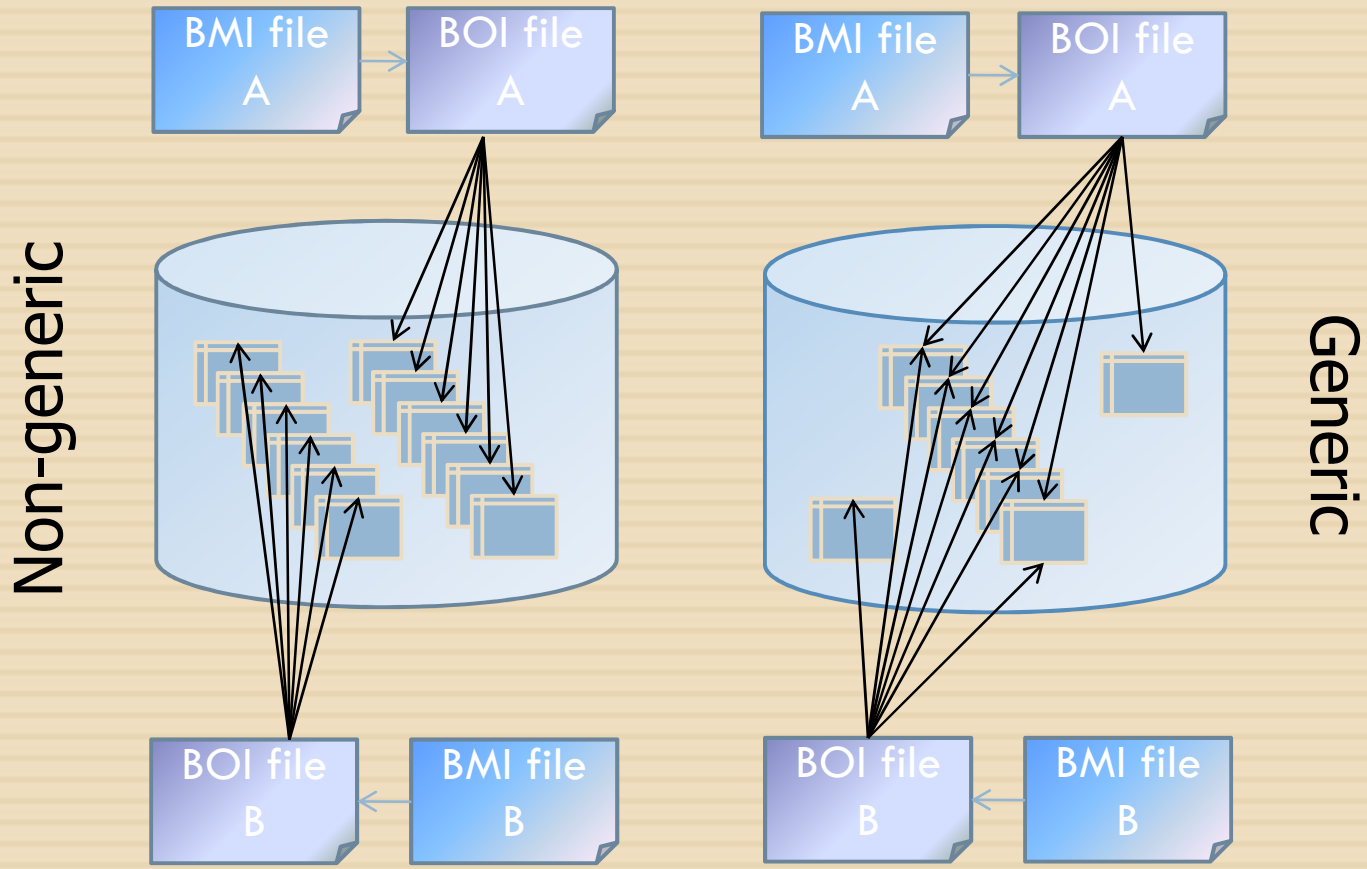

Table Access

Non-generic versus generic table access

Generic BOI files - Concepts

- Common primary key in all tables
  - JOINKEY
  - DMKEY
  - BEGINSTAMP
- Fixed predefined table structures
  - Column widths are set to maximum
  - Tables contain all available columns

Generic BOI files - Features

- Meta and data versioning
- Data versioning occurs on the record level
- Versioning columns
  - BEGINSTAMP
  - ENDSTAMP
    - Date time column type in database
    - Fixed string in Blaise
    - 'YYYYMMDD HH:MM:SS'

Versioning – Record states

- Actual record
  - ENDSTAMP is filled with '99991231 00:00:00'
  - Updateable
- Historical record
  - ENDSTAMP is filled with 'real' timestamp
  - Not updateable

Example versioning

Blaise DataLink 4.8.2

New features

Overruling OLE DB Runtime settings

OLE DB Runtime settings

- Can be used to influence the behavior at runtime
  - ADO settings (affect OLE DB Provider / ODBC driver)
    - LockType
    - CursorType
    - CursorLocation
    - CommandType
  - Blaise settings (affect the way Datalink works)
    - KeepConnectionOpen
    - QueryMode
    - WriteMode
    - UseVersioning
    - Fast reading (obsolete in 4.8.2)
  - Other settings
    - ConnectionString
    - RecordFilter

OLE DB Runtime settings

- Weaknesses until now
  - Rigid; stored in BOI; not setable elsewhere
  - Everyone in every situation uses the same settings
    - Applying the correct setting can make a hugh difference in performance
    - No way to set them for a certain execution context
- 4.8.2 makes some of these settings setable from Manipula, command line, etc

OLE DB Runtime settings

- Settings that can be overruled in 4.8.2
  - ConnectionString
  - QueryMode
  - KeepConnectionOpen
  - RecordFilter

OLE DB Runtime Settings

- Overruling the ConnectionString
  - Can be done completely or partly
  - Methods
    - Command Line (DEP, DataViewer, Data Centre)
    - Manipula
    - API
  - Supported only for BOI files that target an OLE DB database; not for BOI files that reference a BDB file
  - Works with and without Blaise services

ConnectionString overrule mechanism

- Specify the Key Value pairs to overrule
  - ConnectionString as specified in BOI

```
"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=boitest2.mdb;Persist Security Info=False"
```

  - Specified in overrule method

```
"Data Source=ProductionDatabase.mdb"
```

  - Resulting ConnectionString

```
"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=ProductionDatabase.mdb;Persist Security Info=False"
```

Overruling the ConnectionString

- In Manipula
  - By using a file setting

```
updatefile b:boitest ('Mydata.boi', oledb)
settings
  connectionstring = "Data Source=boitest2.mdb;"
```

  - By using a file Method

```
updatefile b:MyData('boitest2.boi',oledb)
settings
  open=no

manipulate
  b.setconnectionstring('Data Source=boitest2.mdb;')
  b.open
```

Overruling the ConnectionString

- Via CommandLine
  - DEP
  - Database Browser
  - Data Centre

Demo: Logon to database

KeepConnectionOpen

- Affects internal working Datalink component (DEP, Manipula, Blaise API)
- Open and Close occurs at two levels
  - Datalink Open – Close (corresponds with DEP and Manipula open/close methods)
  - OLE DB connection Open and Close
- If true; OLE DB connection stays open between Datalink's Open and Close calls
  - Better performance
  - Each client app has its own open connection; no pooling of connections
- If false; underlying OLE DB connection will be opened and closed for each read and write action
  - Enables connection pooling
  - Better scalability
- Manipula File setting: KeepConnectionOpen

OLE DB Runtime settings – QueryMode

- Affects internal working of Datalink component (DEP, Manipula, Blaise API) when records must be read
- Can be set to
  - Record
    - Results in a query to the database server for each record
  - Set
    - Retrieves all records at once in a set
    - Reading occurs on the retrieved set
    - Can be used to quickly export data from BOI to another data file
- Can be set by using Manipula file setting QueryMode
  - Available only in the non-service situation

Example QueryMode

Record filters

- BOI file supports record filters
  - Can be used in Manipula, Blaise API and BDV files
  - Can be stored in the BOI file
  - Only records that comply to the filter will be retrieved
- No design time check upon correctness
- Execution at runtime
  - Leads to the generation of an SQL where clause
  - Blaise field names are translated to their table and column equivalents
  - All fields that have a flat and in depth data column can be used in a record filter

Example Record Filter

Blaise OLE DB Data Storage

Knowing the storage internals

Blaise OLE DB Data Storage

- BOI 4.8.1 and later: data is stored redundantly
  - In the regular data tables
  - As a binary stream of data in Form Info table
- Advantages
  - Optimal integration with Blaise system
  - Record streams can be delivered directly to the Blaise system without any translation

Blaise OLE DB Data Storage

- Disadvantages
  - Changes directly made to the data tables, outside Blaise, are not visible in Blaise, because Blaise uses the record streams instead of the data in the data tables
  - As a result data in record stream and table data may become inconsistent
- Recommendation
  - Don't apply changes to the data tables directly; always use Blaise applications to store your data

Blaise OLE DB Data Storage

- Sometimes it is desired to update the data tables from outside Blaise
- Examples:
  - Your Case Management System wants to update status or contact information fields of your data model directly
  - Integration with other systems; these systems ideally want to modify the data of certain fields directly and real-time
- If this is the case then the record streams have to be synchronized with the updated data in the data tables in order to see the changes in Blaise

Stream Sychronization

Differences between 4.8.1 and 4.8.2

Blaise 4.8.1 Stream Synchronization

- Methods
  - Manipula Synchronize file method
  - Data Centre Data Consistency Wizard
  - By running a Manipula setup with a boi updatefile
    - Set Fast-Reading to true if you want to make the table data consistent with the streams
    - Set Fast-Reading to false if you want to make the streams consistent with the current table data
- Problems
  - FastReading setting works only for record oriented clients like DEP, Manipula (and not for Data Viewer)
  - Available Synchronize methods work on all records

Blaise 4.8.2 Stream Synchronization

- Stream synchronization features
  - Introduction of a special StreamStatus column
  - Manipula's Synchronize file method
  - Data Centre's Data Consistency Wizard
  - Creation of Database Triggers
    - To detect out-of-sync table data
  - OLE DB Stream Synchronisation Service

StreamStatus Column

- New column in Form Info table (BLAISE_FORM)
- Value can be set to an 'O' (out of sync)
- All 4.8.2 synchronize methods will check the value of this column
  - Manipula Synchronize file method
  - Data Consistency Wizard
  - Blaise API SynchronizeOLEDBStreams method
  - OLE DB Stream Synchronization service
- Change in behavior: only records which have an 'O' in the StreamStatus column will be treated by the 4.8.2 synchronization features

Stream Synchronization Scenarios

- Manual synchronization
- Semi-Automatic synchronization
- Automatic synchronization

Manual synchronization

- Update the required data tables outside Blaise
- Set the StreamStatus column of the corresponding records to an 'O'
- Run one of the 4.8.2 Synchronization methods
  - Manipula's Synchronize method
  - Data Consistency Wizard
  - API SynchronizeOLEDBStreams method

Semi-Automatic Synchronization

- By using database triggers
  - Available for SQL server, Oracle and MySQL
- Create an update trigger on each data table
  - Triggers can be created with OLE DB Workshop
- A trigger automatically updates the Stream Status Column (value becomes 'O') if updates are being made to a data table outside Blaise
- Run one of the 4.8.2 Synchronization methods
  - Manipula's Synchronize method
  - Data Consistency Wizard
  - API SynchronizeOLEDBStreams method

Automatic Synchronization

- Database triggers automatically update the Stream Status Column (value becomes 'O') if updates are being made to the data tables outside Blaise
- OLE DB Synchronization service looks (according a certain time interval) whether there are 'Out-Of-Sync' records and, if there are, will synchronize their streams automatically
- Synchronization Service Admin Tool
  - Can be used to
    - Set the Time Interval to look for out-of-sync records
    - Register BOI files that must be automatically synchronized

Automatic Synchronization

- Installing the OLE DB Stream Synchronization service
  - Service binaries will be stored in Blaise installation folder during Blaise installation
  - Service is not installed by default
    - Use .NET utility installutil.exe to install the service
      - Can be found in the .NET Framework version
      - For example C:\Windows\Microsoft.NET\Framework\v2.0.50727
      - Use the 32-bit version of installutil.exe
    - After installation you can start the service in the Services Microsoft Management Console
      - Type 'services.msc' in the search programs and files search box

Blaise Data Centre

Blaise Data Centre

- Provides access to Blaise data files
  - Supports both bdb and boi files
  - View, filter, extract and analyze data
- Case Management tasks
  - Simple administration of cases
  - Create views on your data
  - Deployment of cases to data files
  - Synchronization of data files with centralized data store
  - Record stream synchronization

Blaise Data Centre – New features

- Create views on your data
  - BDV files
  - BOI files
- Save and Open view settings
- Command Line options
  - Use Data Centre in single file mode
  - Disable connection management
  - Use DEP configuration files
- Data Consistency Wizard
  - Synchronize record streams with table data
  - Synchronize table data with record streams