Using Blaise to apply edits to data held in an Input Data Warehouse

Fred Wensing (Australian Bureau of Statistics)

1. Introduction

Business survey and administrative data collected by the Australian Bureau of Statistics (ABS) are delivered to an Input Data Warehouse (IDW), which provides a single uniform database structure to support the various processes that need to be applied before final statistics are obtained. Through a system of status codes, the data in the IDW for all collections and all providers can be tracked from raw value through to final collected, imputed or estimated result.

An important process to be applied to all data is that of "editing" to detect inconsistencies and gaps in the collected data. Blaise has particular strengths in being able to check for anomalies and deliver the results of its checking to the operator. Blaise components can also be used to deliver the results of its checking to a system, thereby enabling edits to be effectively applied in a "batch" mode.

This paper describes the system that has been developed to apply edits to data held in an IDW to support both on-line and batch editing. The paper discusses the issues associated with transforming the data from the IDW into a file structure that can be acted on by Blaise and the way that edit results can be extracted and delivered to a management system for scrutiny and possible resolution.

2. Re-engineering of business statistics processes at ABS

The ABS has recently completed a three-year project, called the Business Statistics Innovation Program, which involved the re-engineering of operations, methodology and technology associated with business statistics. The main business drivers for this project were improved data quality, improvements in provider management, increased capacity, standardisation and operational efficiencies.

A significant aspect of the re-engineering was the adoption of technological and methodological initiatives to rationalise and reduce the disparate methods, processes and systems which were in current use.

All developments were defined and managed in the context of an End-to-End Business Framework which provided a high level view of the business requirements. A key component of the new business model is a central input data warehouse, which is described in more detail below. Another important feature is a shared provider environment to facilitate a total approach to the management of contact with providers.

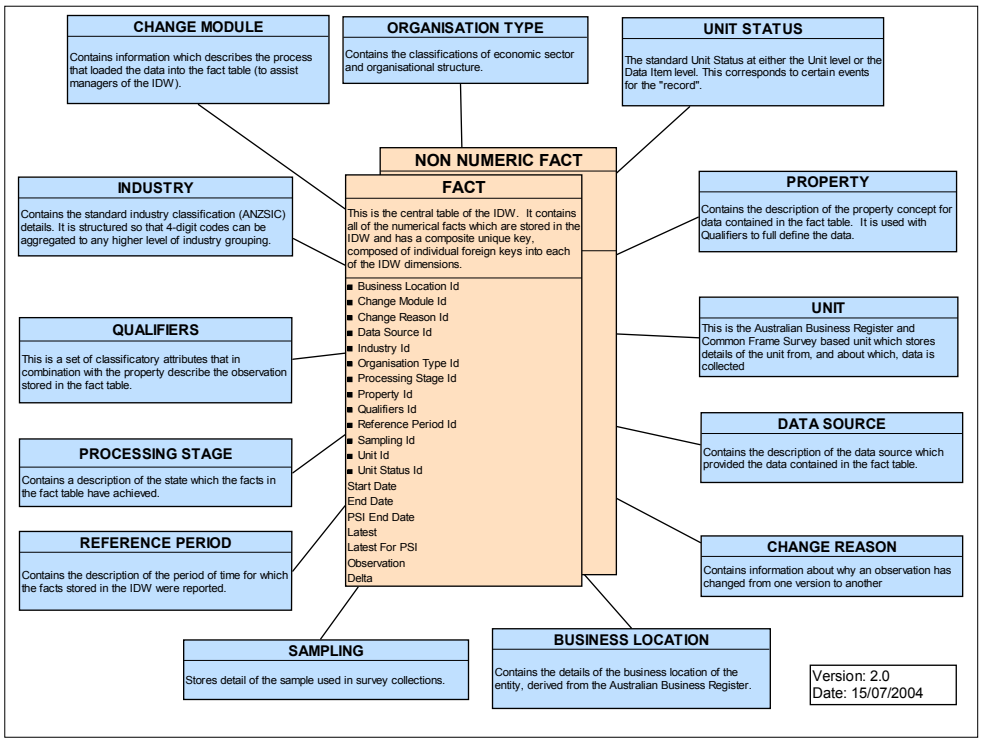
A key principle of the framework was the use of a modular approach to methodologies and systems, rather than a single all-inclusive solution. In this way processes are developed as needed, then updated and re-used where possible.

3. Input Data Warehouse (IDW)

The IDW is the central facility in the new business model and consists of a single managed unit record data store defined using a dimensional model (see Figure 1) and built in Oracle.

The centre of the dimensional model is a "fact" table which contains the numeric observations from the various data collections. Non-numeric observations are stored in a parallel "non numeric fact" table. The dimension tables define the various attributes of each fact, such as the data source, the business identity, industry etc. This structure, of a central fact table and a cluster of surrounding dimensional tables, is often called a "star schema".

Figure 1. An outline of the dimensional model used in the Input Data Warehouse at ABS



The fact table contains all the numerical observations in the IDW, stored as one fact per row, and has a composite unique key which is composed of the individual foreign keys into each of the IDW dimensions. Other attributes in the fact table identify the status of the observation and whether it is the latest for a particular key. The operating principle which applies whenever the fact table is updated, is to add rows rather than to overwrite them. In that way a full track-record is maintained of every observation through all its possible states (eg. captured, edited, imputed, estimated and released).

This dimensional model incorporates two tables (Property and Qualifier) which reflect the ABS's move to align with international standards on data definition and management, more particularly ISO 11179 (Specification and Standardisation of Data Elements), as described in *Oakley, Hamilton and McGrath (2004)*. These two tables together describe the data item definition.

The dimensional data model targets OLAP (Online Analytical Processing) and analysis activities. It produces a user-friendly model for constructing analyses and it gives good query performance. The model and associated infrastructure supports both "real time" transaction processing and post processing analysis.

To obtain the most benefit from the IDW, data should be loaded as soon as possible after it has been captured and before any corrections or other processes have been applied.

4. Editing requirements

One key business function that the systems are required to support is that of "editing", that is, the examination of data for the purposes of identification of anomalies and the possible correction of errors. It is for this function that Blaise software was identified as providing suitable functionality.

While the scope of editing is broad, the main interest in editing in the context of the IDW was related to input editing, which included the following requirements:

- to check the completeness of reported data (ie. identify gaps);
- to check the validity of reported data (ie. whether the answers are within acceptable range);
- to check the internal consistency of reported data (ie. cross check between items);
- to check reported data against historical data for the same unit;
- to distinguish between serious anomalies (usually fatal) and others (usually non-fatal);
- to apply edits in batch mode, as well as on-line;
- to report on all edit failures;
- to examine and resolve edit failures on-line (ie. make corrections).

It was expected that input editing would be carried out at the record level, within a particular collection, and to be associated with a particular collection form (or instrument).

For editing to take place, a record from a defined collection and using a defined form would be extracted from the IDW and examined in the editing tool (Blaise in this case). Edit failures would be identified and possibly corrected with any changes returned to the IDW, along with an updated status code for those facts. In a batch editing mode it was expected that the edit failure information will be passed to a reporting facility for scrutiny later.

5. The decision for a Blaise editing solution

Blaise was identified as the most appropriate software to provide the desired editing solution. The following factors were relevant in coming to this conclusion:

- Blaise has the functionality to apply the kind of edits required;
- Blaise provides a data editing interface which will support on-line examination of data records;
- Blaise OLE DB links and the API should enable it to be integrated with the IDW systems;
- Blaise was available "off-the-shelf" and would therefore be more cost effective than building our own solution;
- Blaise software was already in use elsewhere in the ABS and was therefore a lower cost solution.

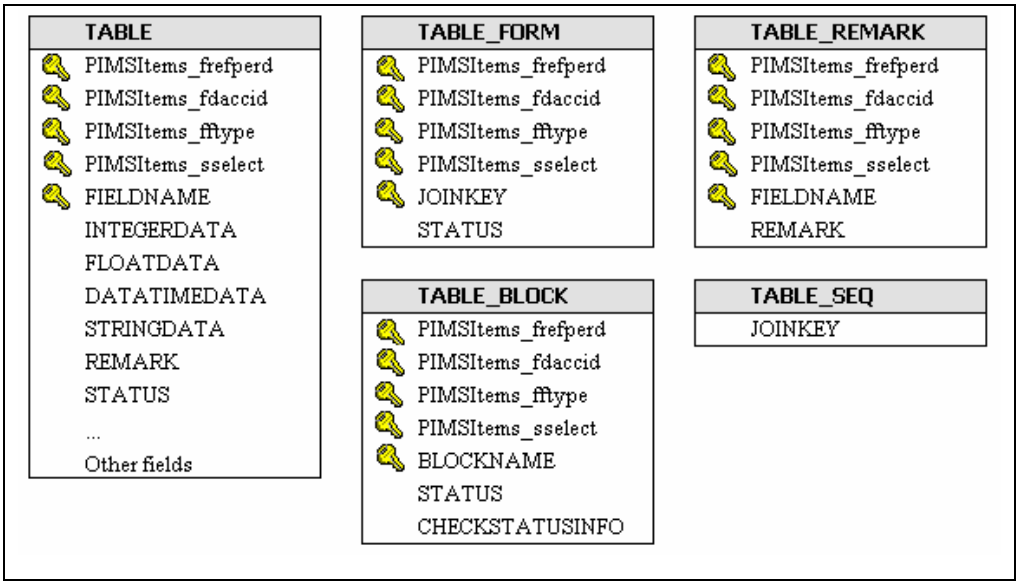
6. Using Blaise with data held in a dimensional model

While Blaise has the capability to access data in other data stores, through the Blaise datalink and OLE DB technology, none of the data partition types supported by that technology corresponds directly with the dimensional model described above.

Therefore, in order to operate on the IDW data using Blaise, it was necessary to develop processes which would extract the entries from the IDW (in Oracle), convert and load them to an interim store which was then readable by Blaise. Another process would also be required to transfer any changes made during editing back to the IDW.

Two data structures were considered suitable for the interim data store: a set of Oracle tables, formatted to suit Blaise; or a Blaise database. Oracle tables were preferable because the IDW was also held in Oracle, and the conversion and loading was considered to be simpler as a result. The OLE DB partition type found to be most suitable for the interim store was the "in-depth" partition type which has as its main table one which holds the data in a Field/Value format. The other tables for this partition type are used to store the Form information, Block information and Remarks (see Figure 2).

Figure 2. Tables used in the Blaise OLE DB in-depth partition



Before Blaise can use the interim store it would be necessary to transform the observations from the IDW into the set of tables needed for the in-depth partition type. The transformation which takes place involves:

- matching the observations with the business identity to create the primary keys for the in-depth tables;
- converting the IDW identities for the extracted observations to their corresponding full Blaise name;
- populating the main data table of the in-depth partition with the observations by inserting the full Blaise name and its correspnding value;
- adding an entry to the Form table for each record that is extracted;
- adding entries to the Block table for each block in the instrument (for each record).

Population of the Block table is important because it is used to track the status of the edits. One small oversight in the original implementation of this part of the transformation, however, was to forget to place an entry in the Block table for the datamodel itself. This omission led to a problem with some instruments because fields assigned in the top level of the instrument were not recognised as being on the path until the rules in the instrument were applied a second time. This was corrected by adding the internal datamodel name to the Block table (for each record).

Vital in the loading of the observations from the IDW, was the ability to map the identity (found in the Property dimension) of the facts extracted from the IDW to the field names in the Blaise instrument. This issue is discussed later.

Figure 3 shows what the main table and the block table look like after they have been loaded with one record. You will notice that the main data table has a field/value appearance. The Block table also contains a block called "main" which is the name used on the DATAMODEL statement in the instrument.

Figure 3. Sample of in-depth tables containing data for one record

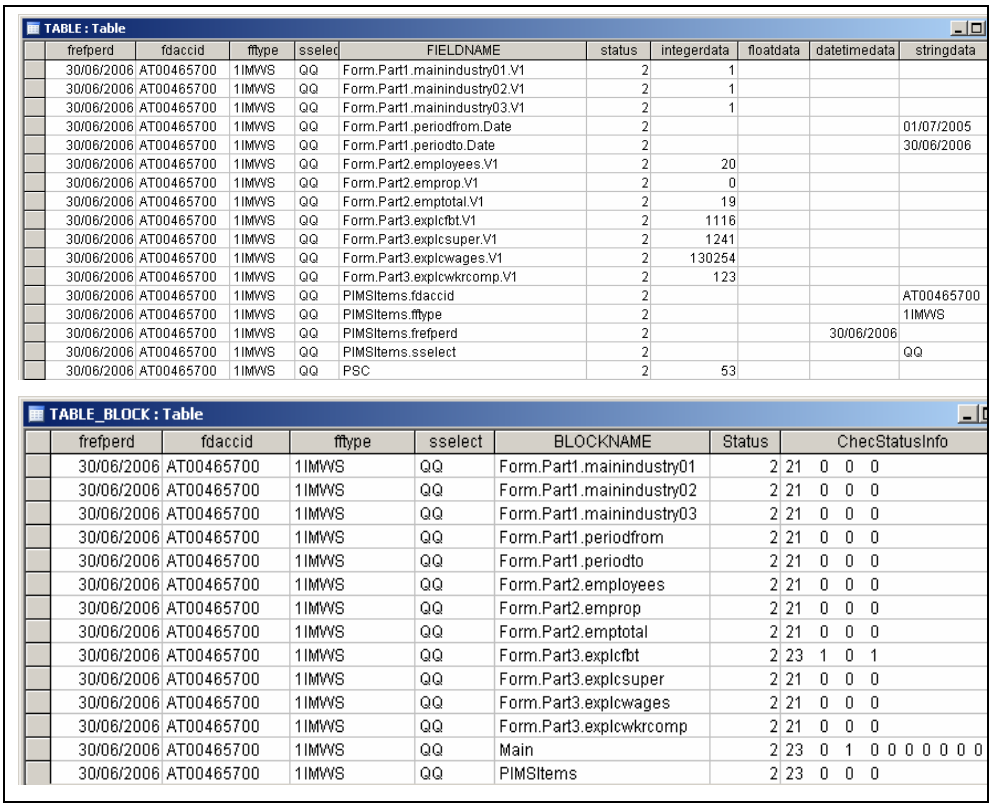


TABLE : Table

frefperd	fdaccid	fftype	sselec	FIELDNAME	status	integerdata	floatdata	datetimedata	stringdata
30/06/2006	AT00465700	1IMWS	QQ	Form.Part1.mainindustry01.V1	2	1			
30/06/2006	AT00465700	1IMWS	QQ	Form.Part1.mainindustry02.V1	2	1			
30/06/2006	AT00465700	1IMWS	QQ	Form.Part1.mainindustry03.V1	2	1			
30/06/2006	AT00465700	1IMWS	QQ	Form.Part1.periodfrom.Date	2				01/07/2005
30/06/2006	AT00465700	1IMWS	QQ	Form.Part1.periodto.Date	2				30/06/2006
30/06/2006	AT00465700	1IMWS	QQ	Form.Part2.employees.V1	2	20			
30/06/2006	AT00465700	1IMWS	QQ	Form.Part2.emprop.V1	2	0			
30/06/2006	AT00465700	1IMWS	QQ	Form.Part2.emptotal.V1	2	19			
30/06/2006	AT00465700	1IMWS	QQ	Form.Part3.explcfbt.V1	2	1116			
30/06/2006	AT00465700	1IMWS	QQ	Form.Part3.explcsuper.V1	2	1241			
30/06/2006	AT00465700	1IMWS	QQ	Form.Part3.explcwages.V1	2	130254			
30/06/2006	AT00465700	1IMWS	QQ	Form.Part3.explcwkrcomp.V1	2	123			
30/06/2006	AT00465700	1IMWS	QQ	PIMSItems.fdaccid	2				AT00465700
30/06/2006	AT00465700	1IMWS	QQ	PIMSItems.fftype	2				1IMWS
30/06/2006	AT00465700	1IMWS	QQ	PIMSItems.frefperd	2			30/06/2006	
30/06/2006	AT00465700	1IMWS	QQ	PIMSItems.sselect	2				QQ
30/06/2006	AT00465700	1IMWS	QQ	PSC	2	53			

TABLE_BLOCK : Table

frefperd	fdaccid	fftype	sselect	BLOCKNAME	Status	ChecStatusInfo
30/06/2006	AT00465700	1IMWS	QQ	Form.Part1.mainindustry01	2	21 0 0 0
30/06/2006	AT00465700	1IMWS	QQ	Form.Part1.mainindustry02	2	21 0 0 0
30/06/2006	AT00465700	1IMWS	QQ	Form.Part1.mainindustry03	2	21 0 0 0
30/06/2006	AT00465700	1IMWS	QQ	Form.Part1.periodfrom	2	21 0 0 0
30/06/2006	AT00465700	1IMWS	QQ	Form.Part1.periodto	2	21 0 0 0
30/06/2006	AT00465700	1IMWS	QQ	Form.Part2.employees	2	21 0 0 0
30/06/2006	AT00465700	1IMWS	QQ	Form.Part2.emprop	2	21 0 0 0
30/06/2006	AT00465700	1IMWS	QQ	Form.Part2.emptotal	2	21 0 0 0
30/06/2006	AT00465700	1IMWS	QQ	Form.Part3.explcfbt	2	23 1 0 1
30/06/2006	AT00465700	1IMWS	QQ	Form.Part3.explcsuper	2	21 0 0 0
30/06/2006	AT00465700	1IMWS	QQ	Form.Part3.explcwages	2	21 0 0 0
30/06/2006	AT00465700	1IMWS	QQ	Form.Part3.explcwkrcomp	2	21 0 0 0
30/06/2006	AT00465700	1IMWS	QQ	Main	2	23 0 1 0 0 0 0 0 0
30/06/2006	AT00465700	1IMWS	QQ	PIMSItems	2	23 0 0 0

7. Operating Blaise editing in a batch mode

There are two ways in which edits can be applied using Blaise. The edit conditions can be incorporated into a program which is run against the data (eg. in a Manipula program) or they can be incorporated into the instrument logic itself. While the former solution is one that fits a traditional view of editing, the latter is much better because the edits are activated whenever the instrument is opened, and re-activated whenever any of the data is changed.

With the edit rules incorporated into the instrument logic, batch editing amounts to simply opening each record, applying the rules, extracting details of the edit

failures, then closing the record and moving on to the next one. All these process steps can be carried out using any standard scripting language and the Blaise API.

Furthermore, the Blaise API provides access to the error objects held in the Blaise Object Model where it can extract the error messages, along with the fields and values involved. All that remains is for the error messages to be processed for reporting.

8. How to report the results of the application of edits

The error messages produced by the Blaise instrument can be conveniently transported to other systems by converting them to XML. In that format, messages concerning the editing outcome(s) can be readily sent to other systems for processing and display.

The processing script which extracted the editing results was enhanced to produce one XML report for each unit edited. Figure 4 shows typical XML entries for a unit in which some errors have been detected.

Figure 4. Extract of XML containing information about edit failures

```
- <pin:DetectedAnomalies xmlns:pin="urn:abs.gov.au/StatisticsProcessInputs">
  - <pin:UnitAttributes>
      <pin:UnitAttribute name="PIMSItems.FDACCID">AT00465700</pin:UnitAttribute>
      <pin:UnitAttribute name="PIMSItems.FREFPERD">30-JUN-2005</pin:UnitAttribute>
      <pin:UnitAttribute name="PIMSItems.FFTYPE">1IMWS</pin:UnitAttribute>
      <pin:UnitAttribute name="PIMSItems.SSELECT">QQ</pin:UnitAttribute>
      <pin:UnitAttribute name="PIMSItems.UANZSIC">XXXX</pin:UnitAttribute>
      <pin:ProcessingCode>53</pin:ProcessingCode>
      <pin:ProcessingCodeDescription>Fatal Anomalies</pin:ProcessingCodeDescription>
    </pin:UnitAttributes>
  - <pin:Anomalies>
    - <pin:Anomaly>
        <pin:Type>Soft</pin:Type>
      - <pin:AssociatedCells>
        - <pin:Cell>
            <pin:Name>Form.Part2.EMPTOTAL.V1</pin:Name>
            <pin:Value>19</pin:Value>
          </pin:Cell>
        </pin:AssociatedCells>
        <pin:Message>4.03 (Q) Total employment differs from BM by more than 20% , check that the
          reporting is correct.</pin:Message>
      </pin:Anomaly>
    - <pin:Anomaly>
        <pin:Type>Hard</pin:Type>
      - <pin:AssociatedCells>
        - <pin:Cell>
            <pin:Name>Form.Part2.EMPPROP.V1</pin:Name>
            <pin:Value>0</pin:Value>
          </pin:Cell>
        </pin:AssociatedCells>
        <pin:Message>4.05 (F) Working Proprietors reported and frame TOLO is not equal to 6 or 7 or
          8.</pin:Message>
      </pin:Anomaly>
    - <pin:Anomaly>
```

9. Managing the editing process - The Editor's Toolkit

To complete the editing facilities, it was necessary to develop a user interface that could be used to manage the editing process. This is particularly important given the number of collections which would eventually be using the editing systems.

The Editor's Toolkit (ETK) is a facility, built in Lotus Notes, which provides the required operational functionality to manage the editing process. Through this

facility it is possible for operators to see the status of records that have been edited. From within this facility they can launch Blaise editing in both on-line and batch modes. The manager can see summary information about the editing that has been done and can assign work to the operators.

The ETK consists of a database containing documents (or forms) for some or all units involved in a collection. For some surveys, only units which have failed at least one edit are recorded in the ETK. For other surveys, a document is created for every unit, even those which are "clean".

The ETK is mail-enabled and receives the XML messages that have been produced by the editing processes. These XML messages are then transformed into documents that end up containing the texts of the error messages along with the fieldname and value of the first involved field. The views in the database provide summary information about the edit failures and the status of records.

Figure 5 shows a typical view of records in the Editors Toolkit showing form type (code number) and error message with counts against each. The navigator window shows links to other views and actions. Figure 6 shows a sample document from the ETK with edit failures listed.

To provide a better understanding of the editing processes and how Blaise, the ETK and the IDW interact, a process diagram has been included at Attachment A.

Figure 5. Typical view of documents in the Editors Toolkit showing them grouped by error messages

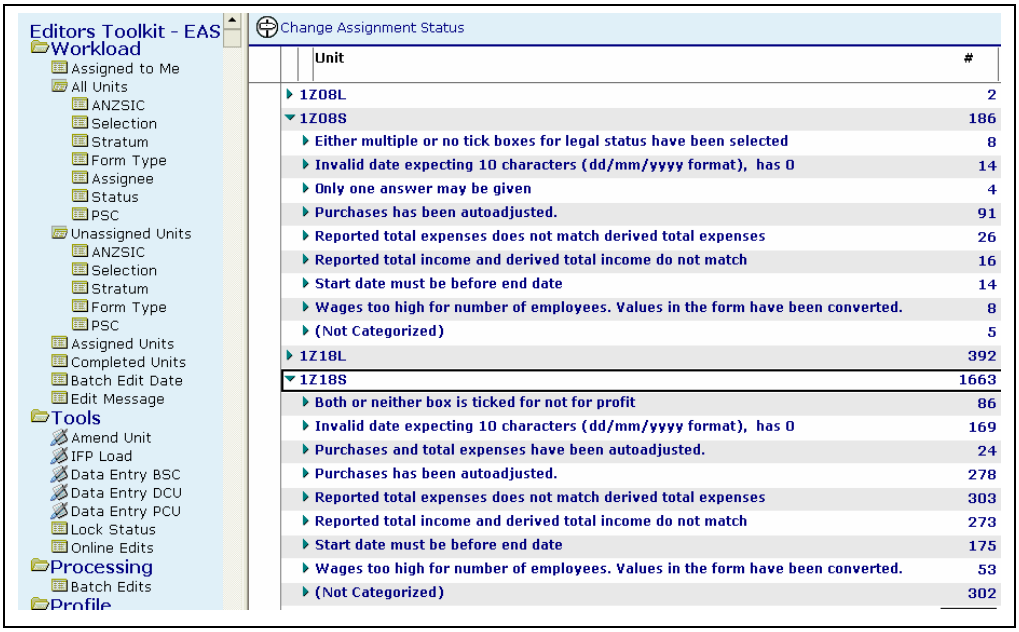
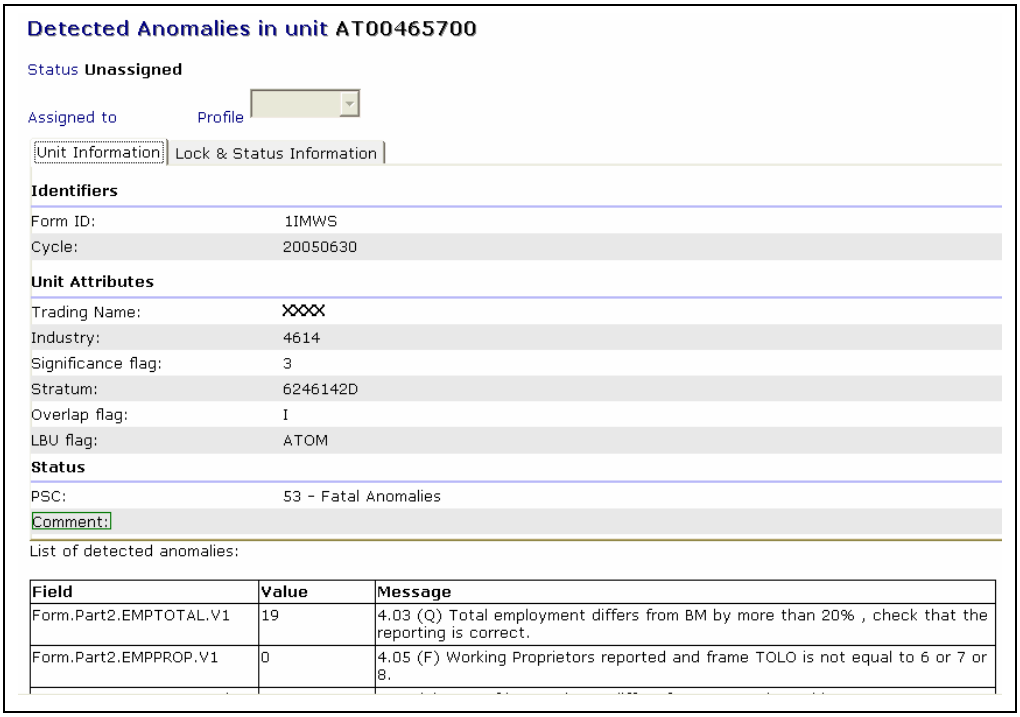


Figure 6. Typical view of a document from the Editors Toolkit showing listing of error messages



10. Instrument design and operation for interactive editing

Once the data has been edited (and reports produced) the editing system needs to enable the operator to examine particular failed records for possible correction.

A special Blaise instrument design and configuration has been developed to provide useful features for interactive editing. These include:

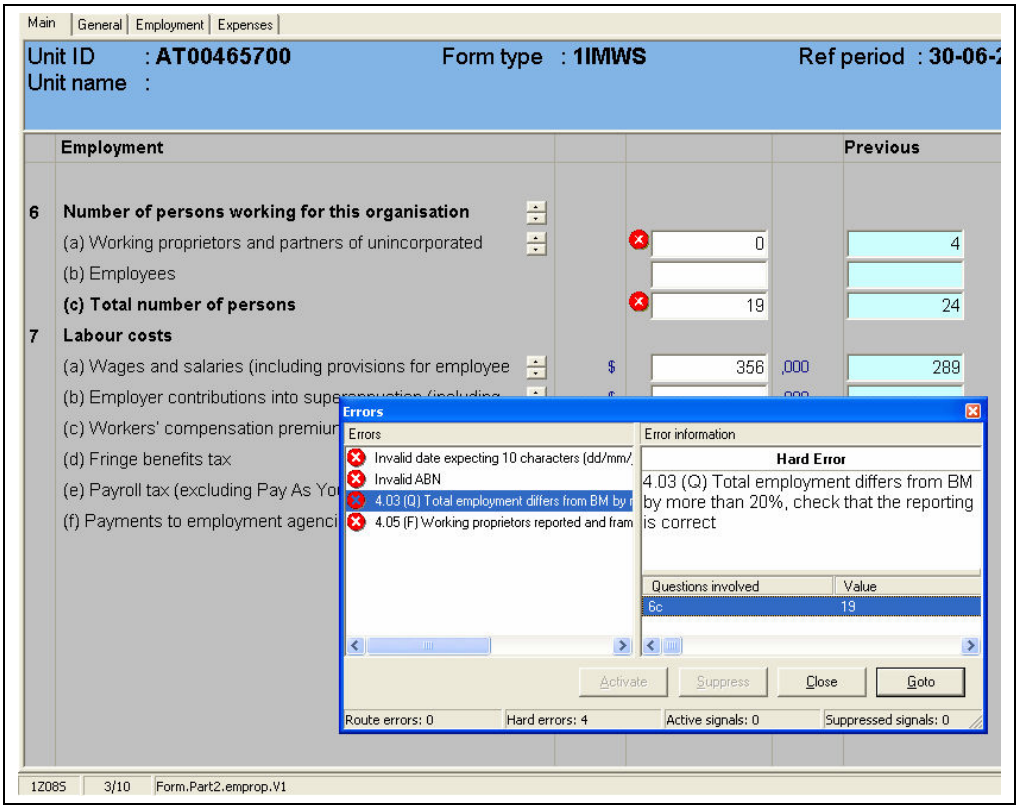
- screen appearance which is similar to the paper forms with which the data was collected;
- use of the error and warning symbols to signify fields with edit failure(s);
- free navigation through all fields by using the editing mode rather than interviewing mode.

The special screen design for editing makes use of rows and columns in the Form Pane to display all the important text information associated with a data field in a way that approaches the appearance used on the paper form. This was achieved by developing standard Blocks for certain field types (number, string, percentage etc) and desired layout, then using parameters to pass the various texts into the block for presentation on screen. The implementation of this block approach was based, in part, on the method described by *De Bolster (2004)*. Figure 7 shows a typical screen for an instrument which has been set up for editing.

Some flexibility was added to the editing instrument to enable matching data from previous cycles of the survey to be displayed on the same "row" of the screen, if so desired.

To support the application of different edits in batch mode, compared to on-line, a special field called OperationMode was incorporated into each instrument. Edits which only apply in a particular mode are then written to involve this field. The field is set to the particular mode by the system whenever the edits are applied, thereby causing selected edits to be activated or not-activated as the case may be.

Figure 7. Typical screen layout for a business survey instrument used in editing, showing also the edit message dialog box



With this instrument all the usual Blaise navigation techniques, such as "Navigate/Show all errors", are available to make it easy for operators to locate anomalies for investigation. A typical error dialog is also shown in Figure 7.

11. Aligning field definitions from the dimensional model with fields in Blaise

As mentioned previously, one important issue to be handled in the editing system was to be able to map the identities of the facts extracted from the IDW to the field names used in the Blaise instrument. That way data could be readily transformed from the IDW store to a Blaise-readable form.

Data elements in the IDW dimensional model are identified through a system name, which is a character string of up to 16 characters in length. The system names are recorded in the Property dimension in the IDW. These system names are assigned to each data element by staff from a central standards section and are unique across all business collections.

While the early Blaise instruments used for editing made direct use of the system name, this limited the design possibilities for those instruments. When instruments contained special elements, such as arrays, the direct matching on name became difficult and additional logic had to be added to the extraction/transformation programs.

With the adoption of the instrument design described above, the alignment of Blaise field names and the IDW fact identities was simplified so that a match occurred when the system name could be found "within" the full Blaise name (for matching purposes, the dot character "." found in full Blaise names was treated as an underscore character "_") (see Figure 8). This meant that there was more scope

for variations in the placement of fields in the instrument while making the matching of names more generic.

Figure 8. Illustration of matching various IDW system names to Blaise field names

IDW system name		Full Blaise name
EMPTOTAL	=	Form.Part2.EMPTOTAL.V1
INCSERVOTR1_V	=	Form.Part3.INCSERVOTR1.V1
INCSERVOTR1_S	=	Form.Part3.INCSERVOTR1.Specify
PIMSItems_FDACCID	=	PIMSItems.FDACCID
INCSERVOTR1_OLD_V1	=	Form.Part3.INCSERVOTR1.Old.V1

The alignment method described here does place some constraints on the development of Blaise instruments because it needs to make use of the pre-defined system names. However, it is expected that this method will be superseded after changes are made to make the system more metadata-driven. In particular, it is hoped that instruments used for editing of business data will be "generated" from the same metadata that will be used to define the data elements in the IDW.

12. Performance issues for interactive editing

The system described in this paper is largely focused on the batch editing process. Although the same architecture could apply for interactive (on-line) editing, it was found that there were some performance issues to contend with.

Performance of the Blaise OLE DB Interface (BOI) file in Oracle suffered somewhat when accessed on-line over the WAN. This was rectified by adding a further step to the extraction and loading of the in-depth partition. The additional step, used for interactive editing only, would transfer the data in the main data table into a native Blaise file located on the operator's computer. Interaction of the operator with this local data was then carried out through the data entry program (DEP.exe) rather than the API.

Dispatching the XML messages containing the edit results to the ETK produced an unnecessary round-trip and consequential delay for interactive editing. Given that the ETK was still connected, and open, it was more effective to transfer the edit results directly to the relevant document in the ETK, by-passing the XML steps.

The configuration of the editing process involves a chain of events in which various connections are made between Lotus Notes, Oracle and Blaise objects, not to mention additional links that these components may have with other parts of the environment. It appeared that each time a unit was selected for interactive editing, all these connections are opened up before the record is presented on the screen. When the editing of that unit is over, all the connections are closed again. A number of avenues of investigation are planned to deal with this. One solution would be to reconfigure the systems to keep the various connections "open" so that performance will improve with the second and subsequent record. Another solution would be for the system to extract more than one record from the IDW and deliver them locally to the user for editing.

13. Conclusion

The system described by this paper illustrates how it is possible for Blaise to perform an editing role in the context of an input data warehouse, by exploiting the OLE DB datalink and Blaise API technology.

While the Blaise instruments which have been developed have been designed for data editing they can readily be modified to perform data entry functions, or with some additional work, for computer assisted interviewing.

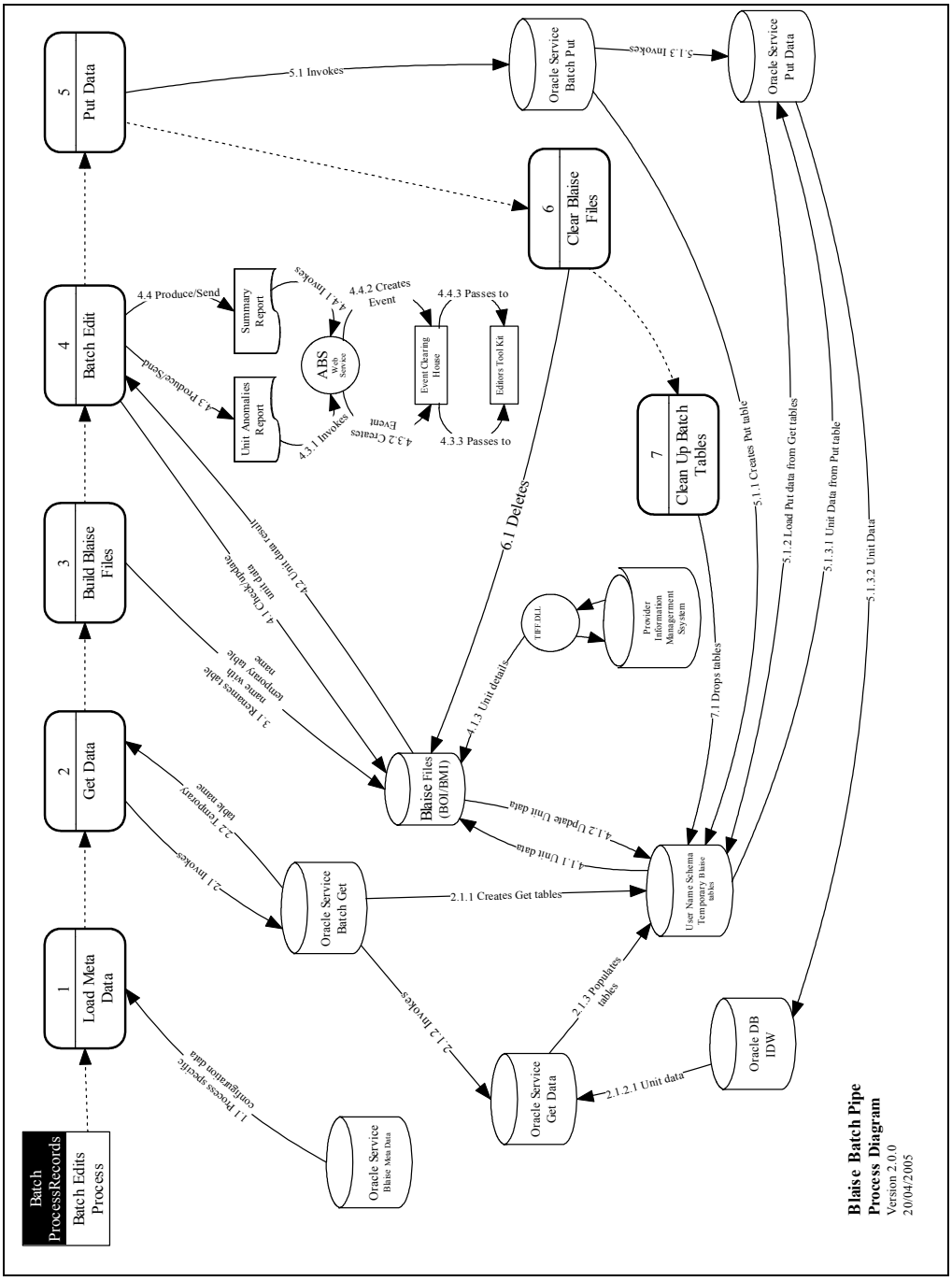
The Blaise instruments used for editing have been developed by hand but a set of appropriate instrument standards and considerable common source code has now been developed. The next logical step is to look at ways that the instruments can be generated from the same metadata that is used to support the IDW.

14. References

Oakley G., Hamilton, A., McGrath M.(2004): Using ISO/IEC 11179 to help with metadata management problems, Presented to the Joint UNECE/Eurostat/OECD Work Session on Statistical Metadata (METIS) , Geneva, 9-11 February 2004.

De Bolster, G. (2004): Generating Blaise with Blaise, Proceedings of the 9th International Blaise Users Conference, Gatineau, Québec, Canada, September 2004.

Attachment A. Process diagram for Blaise batch editing of data in the Input Data Warehouse at the ABS.



Session 4.2: Alongside the Data (Paradata)

Blaise PlayBack and Recovery System
Youhong Liu & Gina-Qian Cheung
(University of Michigan, USA)

Active Management: A New Strategy for Managing Data Collection
Luc Tremblay
(Statistics Canada)

Work Flow for the Weighting of the German Microcensus Data Using Blaise Bascula
Kirsten Iversen
(Federal Statistical Office, Germany)