Adding Business Intelligence to Paradata: The Blaise Audit Trail

Joel Devonshire and Gina-Qian Cheung - Survey Research Center, University of Michigan

1. Introduction

For the 2012 International Blaise Users Conference, we authored a paper that summarized an effort to make Blaise audit trail (ADT) data more readily available and easier to analyze for end users (Devonshire, Liu, & Cheung, 2012). This effort included parsing ADT files in an automated bulk process allowing the data to be loaded into relational database tables. This new centralized database represented a significant refinement in the University of Michigan's Survey Research Center, Survey Research Operations (SRO)'s approach to paradata processing, one that we hoped would facilitate responsive survey design and higher-quality data collection. While this effort was successful in meeting its initial goals, we also discovered a few shortcomings of this approach related to exploratory analysis capabilities for end users. This paper provides a summary of these shortcomings, and presents a strategy that was used to address them – an Online Analytical Processing cube that pre-aggregates raw ADT data and joins them to other sources of paradata.

1.1. Background

Ever since SRO began to use Blaise to support its operations, we have heavily utilized its paradata capabilities, and over the years have made improvements to how this paradata can be accessed and analyzed. For example, in Devonshire, Liu, & Cheung (2012), we describe the evolution of SRO's approach to ADT processing and analysis. Moving to a centralized SQL Server database to store "raw" ADT records – where every row in the table represents an entry into a Blaise field – was a significant step forward because it allowed the possibility of real-time data analysis on a secure and centralized server. That is, it removed the processing burden from the end user and helped to facilitate an on-demand analysis paradigm.[8]

The ultimate vision guiding that effort was one in which a survey director or analyst could log into a web site (or connect to the database directly with a tool such as SAS or Microsoft Access) and either manually write queries against the data or, ideally, point and click through an intuitive user interface to quickly answer specific questions. Such questions might include how long particular cases or groups of cases are taking, whether certain questions in the instrument might be problematic, how changes in the data model may be affecting data entry or interviewer behavior, or any number of issues. Thus, simply loading the ADT data into relational database tables was not sufficient in and of itself. We needed to consider what tools might be used to ultimately work with the data.

After some initial experimentation with a web-based user interface that read directly from the SQL Server database, we quickly began to realize that developing a tool that met all of our requirements would be challenging for several reasons. Among them were the following:

- As mentioned above, the main table in the ADT database stored one record per field entry in the Blaise instrument. This meant that very quickly, with only a few projects in the database, this table had millions of records. As of this writing, it contains roughly 37 million records over 15

[8] Throughout this paper, the term "end user" is used to generically refer to any potential user of the ADT data. End users could include survey directors, project managers, data managers, programmers, principal investigators, or anyone else granted access to the data.

projects. This translates into slow query performance, even for relatively simple queries with optimized tables (e.g., indexed columns, etc.).

- The point above was further complicated by the fact that most of the interesting questions that one might want to answer with ADT data involve some type of aggregation of the raw data rather than a simple subset. For example, one might want to compare average interview length between two or more groups of cases. Aggregation queries are generally more processing-intensive, and contribute to slower query performance.

- To be especially useful, ADT data need to be joined to other sources of data. For example, common queries might involve pulling supplemental sample management system paradata (e.g., result codes, sample types, interviewer attributes, etc.) as a basis for grouping the aggregated ADT data. A front-end tool that simply read the ADT database could not allow for this ready-made joining of disparate data sources.

- Finally, as large scale ADT data analysis is still a somewhat unexplored area of investigation, it is not uncommon to have analysts or project managers unsure of the questions they want to ask, or are possible to answer, with this kind of data. Consequently, one important requirement for an analysis tool is that it be very flexible and customizable, allowing for open data exploration as well as targeted analysis. Static pre-defined reports, or dynamic query tools that only provide limited options, would be useful up to a point but would not facilitate the kind of exploratory data analysis that we needed to allow.

1.2. Online Analytical Processing

Given these limitations and requirements, we settled on one solution that allows fast and flexible queries against the ADT data with minimal programming knowledge needed by the end user. Additionally, it allows for pre-joining of ADT data to other data sources so that a wider range of data is available for analysts. This tool is known as an Online Analytical Processing (OLAP) database, or "cube." OLAP cubes represent one aspect of what is commonly referred to as "Business Intelligence," or "BI." While there are various definitions of the term BI, the one attributed to Boris Evelson (2008) seems to capture BI in its broadest sense: A "set of theories, methodologies, processes, architectures, and technologies that transform raw data into meaningful and useful information for business purposes." BI can include elements such as dashboards and other data visualization tools, data mining tools, predictive analytics, and a host of other things, including OLAP cubes.

Online Analytical Processing stands in contrast to Online *Transaction* Processing (OLTP), which is what is traditionally used in relational database environments. The terms themselves highlight the relative strengths of each type of data storage paradigm. While transactional databases are optimized for data transactions and storage, analytical databases are optimized for analysis, particularly the kind of analysis that involves the aggregation of long data sets. While it is beyond the scope of this paper to fully describe all elements of an OLAP cube, the major concepts are presented below.

The word "cube" is employed for OLAP databases in order to point to the multi-dimensional nature of pre-aggregated data. While a cube has three dimensions, an OLAP cube theoretically has unlimited dimensions, which are simply groupings of the data. The data that is being grouped is referred to as "facts" or "measures," and what an OLAP cube typically boils down to is a series of dimension tables that are related to one or more fact tables. The dimension tables define how the fact data is to be grouped, and during cube processing, the fact data is aggregated in various ways according to the defined dimensions.

This is illustrated in Figure 1, an example cube image taken from the internet (Nicolas, G., n.d.). It represents a relatively common use of OLAP cubes: to summarize sales data. In this example, the fact/measure data are total sales (e.g., number of products or dollar amounts), and the groupings of that data, or dimensions, are Customer, Product, and Time. Each dimension has "members," which are the individual units of that grouping. The primary strength of the OLAP cube is that all of the sales totals are pre-aggregated across all dimensions, so that it becomes relatively easy to query the data and quickly find a total for a specific product for a specific customer on a specific date.

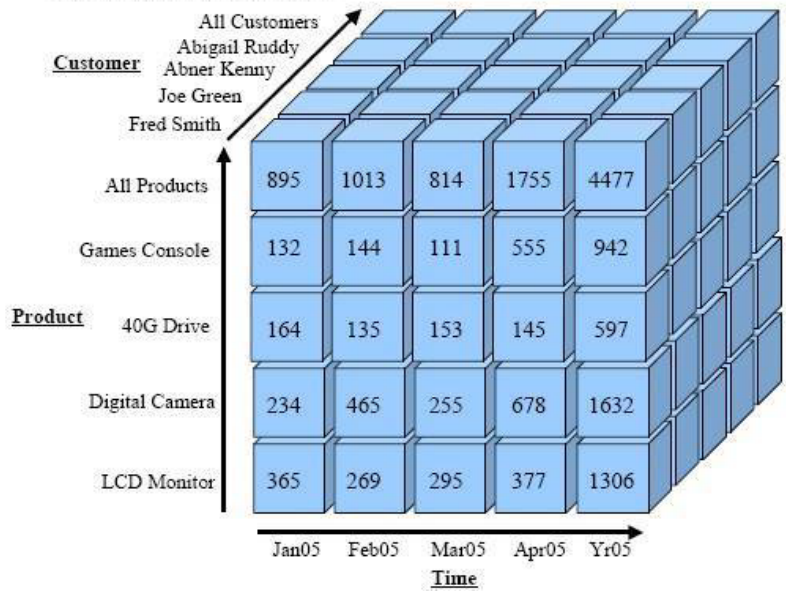

Figure 1. Example of an OLAP Cube (Nicolas, G., n.d.)

The same basic concept can be used to think about ADT data, where in this case the facts, or measures, are any quantifiable elements of the audit trail. Examples include things like elapsed time, keystroke counts, specific hot key counts, counts of "events" such as hard and soft errors, Help menu access, field backups, data-entry, and so forth. Each of these measures could be grouped by things such as Blaise data model fields and blocks, projects, data model versions, sample id, and any one of a number of sample characteristics gleaned from sample management system data that are joined to the ADT data (e.g., sample type, interviewer, result status, date of interview, etc.).

An additional strength of an OLAP cube is its ability to structure the data in such a way that hierarchical relationships are easy to define and navigate. For example, Blaise fields can be easily grouped into their respective blocks, interviewers can be grouped into teams defined by team leaders, days can be grouped into weeks, weeks into months, months into years, and so forth. This allows for a drill-down/roll-up capability when exploring the data.

Finally, OLAP cubes allow a choice between exploring the data through intuitive point-and-click user interfaces (of which more will be said later) or by manually writing queries in a language known as MultiDimensional eXpressions (MDX), which is somewhat similar to Structured Query Language (SQL) but allows one to refer to and query the multidimensional space that makes up a cube. The point to be made here is that OLAP databases can flexibly meet the knowledge base and skills of different user sets. While a set of common tools exist to navigate cube data, a more sophisticated data analyst could also write MDX code to generate a very precise data set to analyze.

2. Building a Data Warehouse

There are several software suites and basic approaches that one can use to create OLAP cubes. Again, it is beyond the scope of this paper to describe all of the considerations that went into deciding what specific approach we would take. It is worth mentioning, however, that we spent time considering two main options for software: SAS Enterprise Business Intelligence Server and Microsoft SQL Server Analysis Services. Both software suites offered very similar capabilities, but for a variety of reasons we decided to use MS SQL Server. This package actually involves three different, but related, services: SQL Server Integration Services (SSIS), SQL Server Analysis Services (SSAS), and SQL Server Reporting Services (SSRS). It is worth mentioning each one because they each serve a different function in the total BI implementation and will be described further in this paper.

One interesting aspect of the SQL Server package is its ability to allow different models to create an OLAP cube. While it is possible to create a cube directly on top of OLTP data sources – an option that carries the significant advantage of being much simpler to design and set up – we elected to go the more traditional route, which involves the design of an intermediate "staging ground" for the data. This staging ground is commonly referred to as a data warehouse, and it is designed in such a way so as to facilitate the cube creation by storing all of the needed data in one place, and having the data structure and relationships optimized for the cube aggregation processing. The process of preparing and loading the data into the data warehouse is known as the "Extract, Transform, and Load" (ETL) process, and will be described in more detail below.

2.1. The Star Schema

As mentioned above, an OLAP cube typically uses several dimension tables in the data warehouse that are related to one or more fact tables. Because all of the dimension tables need to relate back to the fact table(s), the resulting data warehouse database schema looks very much like a star when visualized. The "Star Schema" is one of the most common structures used to support OLAP cubes (though other schema designs are certainly possible). This is illustrated in Figure 2, which shows the structure of the ADT data warehouse as of this writing.

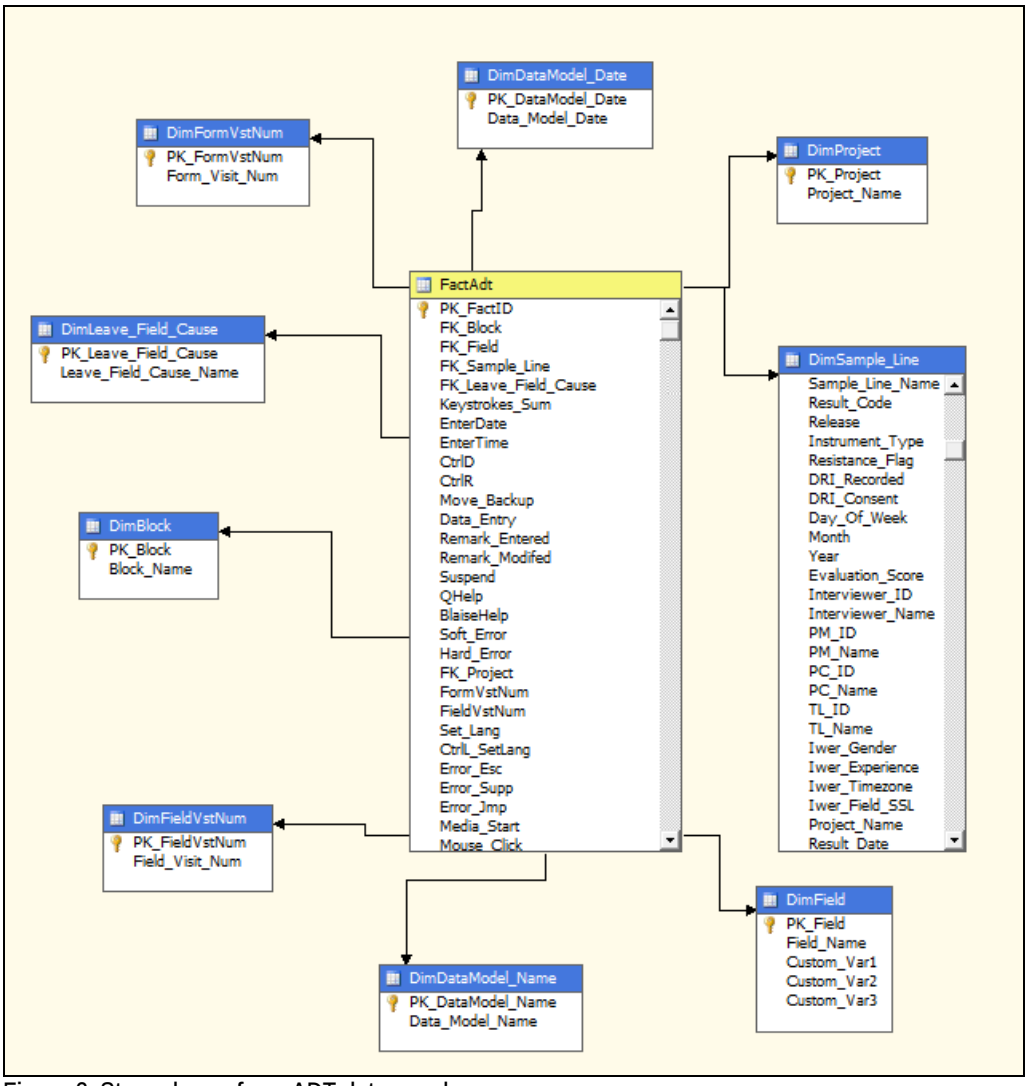

Figure 2. Star schema from ADT data warehouse

The ADT data warehouse currently contains nine dimension tables, and the primary keys in each one relate to foreign keys in the "FactAdt" table. A full list of all of the dimensions (including their attributes) and measures, as well as descriptions of each item, can be found in the Appendix.

2.2. Data Flow and Processing

In Devonshire, Liu, & Cheung (2012), we describe the process by which we parse individual ADT files and store them in an OLTP database. In the following discussion, this initial database will be referred to as the "raw ADT database." As we briefly mentioned above, the process of creating an OLAP cube on top of that raw data involves a few distinct steps, the most complicated probably being the ETL process. Figure 3 illustrates the broader picture of steps involved. For example, note how the ETL process is where the raw ADT data are combined with any other data sources that need to be available in the cube (e.g., our sample management system known as SurveyTrak, etc.). The ETL stage is complex because not only do different data sources need to be combined, but this is the stage where new data columns need to be calculated, existing columns transformed, and relationships created so that the end result is a set of data that can be loaded into the star schema pictured above.

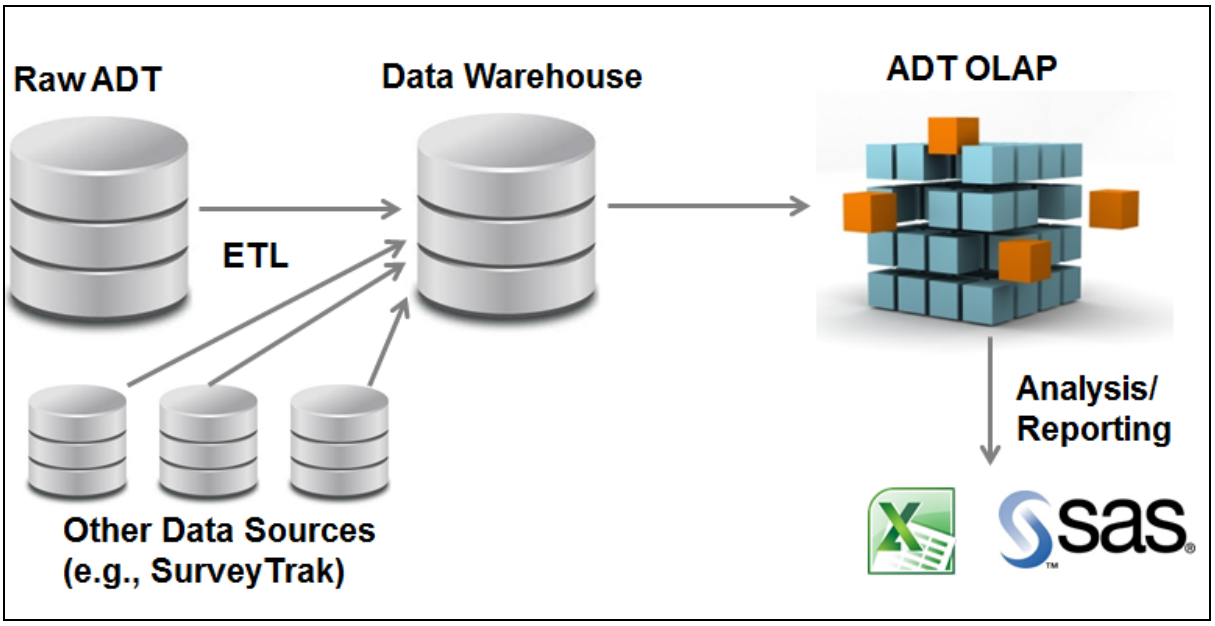

Figure 3. Steps to build an OLAP cube

The ETL stage is also where SQL Server Integration Services is used, which provides a user interface for developing a data flow procedure. An example of this is pictured in Figure 4, and shows each step of the process, including wiping out the data, loading each dimension table, and then loading the fact table. Figure 5 shows the steps involved in loading the fact table, and is only presented here to provide a sense of layered process behind preparing the raw data.

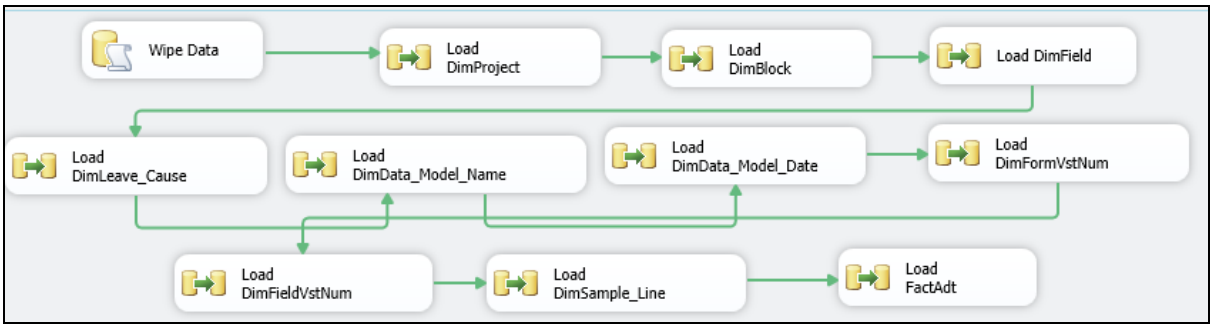

Figure 4. Data flow of ETL process

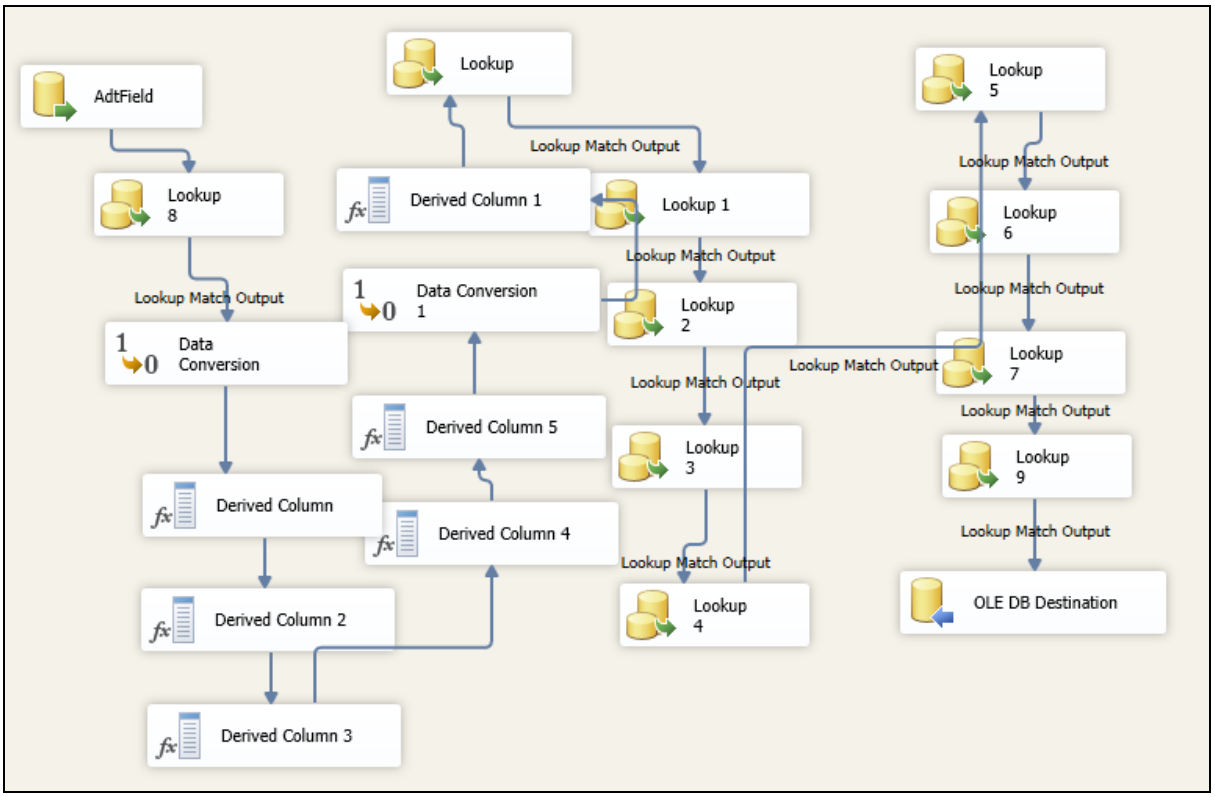

Figure 5. Breakdown of fact table loading process

3. Building the ADT OLAP Cube

Once the data warehouse is loaded with all the necessary data, SQL Server Analysis Services is used to construct the cube. This stage is also pictured in Figure 3. A different user interface is used to create a connection to the data warehouse, define all the dimensions and measures of the cube (which should correspond to the structure of the data warehouse), and define a variety of parameters of the cube, such as whether/how the data should be partitioned, what level of aggregation to define, and the various user roles that need to exist for end users. It is possible, for example, to define custom views of the cube, or allow access to only certain portions of the data to specific people.

3.1. Choosing Dimensions and Measures

Ideally, the dimensions and measures in the cube have been thought through and defined fairly early on in the process, long before the cube is created. This is necessary because, as is evident from Figure 3, the cube structure carries implications for how the ETL process and data warehouse are defined. In choosing the dimensions and measures for the ADT OLAP cube, we attempted to be broad but not overwhelming. For example, it is possible to include in the cube an average, minimum, maximum, standard deviation, and median values (among others) for any particular measure in the cube, such as elapsed time. Thus, a decision needs to be made up-front regarding what kinds of information might be most useful to see at the outset. Tools also exist for the end user to add their own custom calculated measures, which means that not everything needs to be included at the outset.

As can be seen in the appendix, we selected a wide range of measures and attempted to select enough dimensions to allow the same measures to be viewed at different levels of granularity (e.g., data model, block name, field name, field visit number, etc.). We also introduced an interviewer hierarchy to allow a

drill-down through various levels of field management structure. Some of the measures, such as “Data Entry” represent calculated measures defined in the ETL process (e.g., Data Entry = 1 when the value upon leaving the field does not equal the value upon entering the field). Another example of this is the measure “Move Backup,” which is calculated to equal 1 whenever the reason for leaving the field included pressing the back or up arrows, page up, etc. We also added fields to hold custom variables that can be defined for each project in the ETL process. These variables could contain different subsets of Blaise fields in order to create, for example, custom timing variables. Finally, we added a diverse set of variables from our sample management system, SurveyTrak. Some examples include indicators of whether the interview was digitally recorded, whether the case was associated with some respondent resistance, the experience level of the interviewer, the date of data collection, and so forth.

3.2. Automated Processing and Implementing Changes

Once the ETL process and OLAP cube have been created, they can both be deployed to a central server and put on a scheduler to run at an ideal time, which for us is overnight. This means that the data is refreshed and all cube processing happens in the background, before the end user connects and runs queries. The more complex the cube is, the longer processing will take, but the ADT OLAP cube typically takes roughly one and ½ hours, including the ETL process.

Once the cube is deployed in production, usage can be tracked in order to collect data about common queries run against the data. Requested changes to the cube structure can also be made fairly easily, and take effect the next time the cube is processed. Consequently, it is relatively easy to add in additional variables or calculations that analysts find useful.

4. Real-World Usage Examples

In many ways, the power and flexibility of an OLAP cube is difficult to grasp until one actually connects and starts to use it to explore the data. The structure of the data can be best likened to a large pivot table, in which different measures can be selected or combined, and the grouping changed at will by selecting various combinations of dimensions. In fact, the pivot table is exactly the mechanism that many applications use to display and work with cube data. In this section, we will discuss the ways to connect to and use cube data, and some potential real-world applications of the ADT data in particular.

4.1. Tools To Connect To the Cube

Just as there are many tools one could use to build an OLAP cube, there are many ways to view the resulting data. One of these tools is familiar among many people at SRC/SRO, Microsoft Excel, which can natively support connecting to a SSAS OLAP cube once it is deployed to a central server. When using Excel, navigation of cubes takes place with a pivot table that reads all of the dimensions and measures directly from the cube. This is illustrated in Figures 6 and 7.

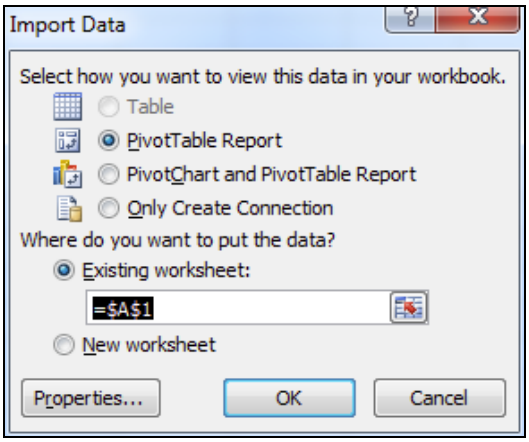

Figure 6. Option to create pivot table or pivot chart in Microsoft Excel

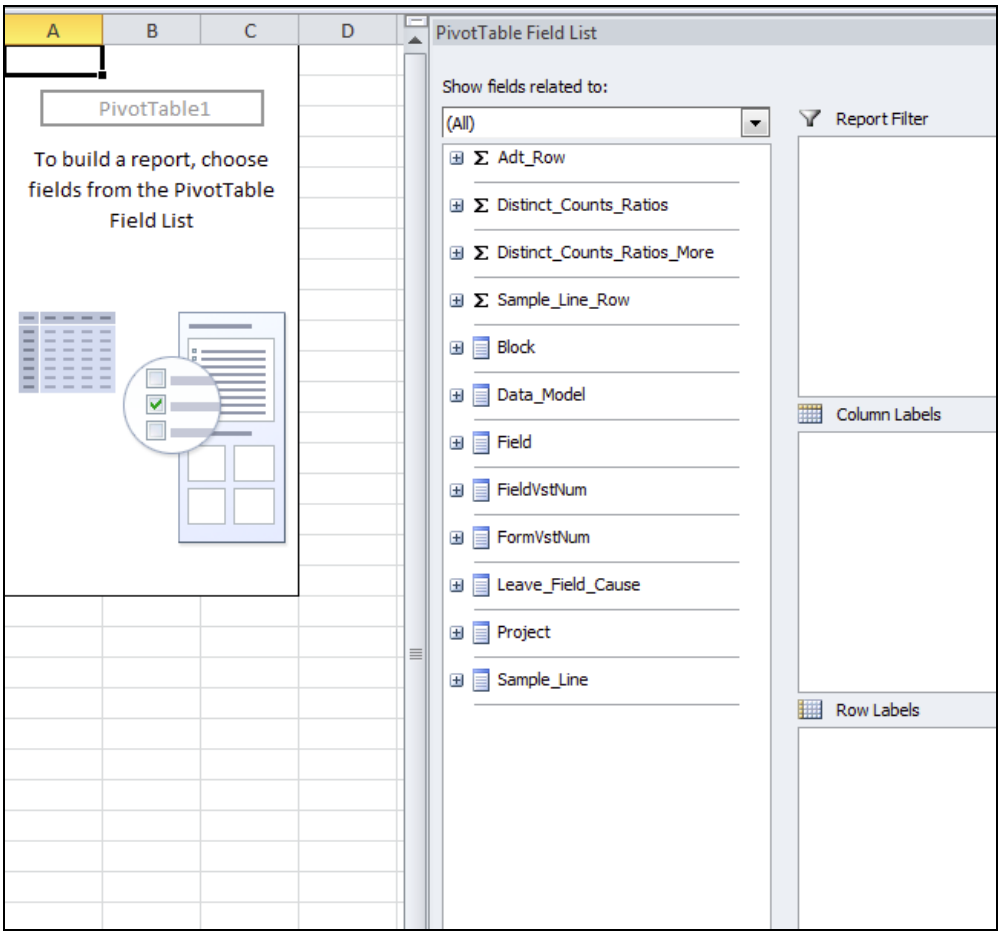

Figure 7. Pivot table in Excel once connected to the cube

4.2. Creating data sets

A full primer on the features and usage of pivot tables is beyond the scope of this paper. Briefly, pivot tables allow a wide range of features, including sorting, filtering, calculating, and even writing data back to the cube (if allowed by the administrator). Pivot charts can be used to add some visualization as well.

In Figure 7, the measures and dimensions are displayed in a collapsed state, which means that the various dimension attributes and specific measures are not visible. However, in practice, all of them would be fully expanded, and the strength of the pivot table design becomes apparent as one simply clicks on checkboxes to add, remove, or combine elements to the table. To give just a sense of the different combinations possible, we will present a few examples of data sets that could be created.

Let us imagine that questionnaire designers would like to assess the various items in the questionnaire. They would like to have some sense of average times spent within each question, the number of hard or soft errors that are encountered at each question, the number of times interviewers back up to the previous question, the number of times data is entered, and the number of times each question tends to be presented in a typical interview. With the OLAP cube, building such a report is a matter of just a few clicks of the mouse. For example, the first step would be to filter the cube to just the project of interest (assuming one did not want to compare across multiple projects). An example of the project list is pictured in Figure 8.

	A	B	C
1	**Row Labels**		
2	NSFGCYCLE7Female		
3	NSFGCYCLE7Male		
4	Robust		
5	SRC.DST.CAS.APR13		
6	SRC.DST.CAS.FEB13		
7	SRC.DST.CAS.JAN13		
8	SRC.DST.CAS.JUN13		
9	SRC.DST.CAS.MAR13		
10	SRC.DST.CAS.MAY13		
11	SRC.SRO.DUST13.PRETEST.PROD		
12	SRC.SRO.DUST13.PROD		
13	SRC.SRO.HRS11.PROD.MAIN		
14	SRC.SRO.HRS2012.PROD		
15	SRC.SRO.MRRS2010.PROD		
16	SRC.SRO.MRRS3.PROD		
17	SRC.SRO.PSID11.PROD		
18	SRC.SRO.PSID13.PROD		
19	SRC.SRO.PTMS.TRACK.PROD		
20	SRC.SRO.PTMSNR2011.PROD		
21	SRC.SRO.TA2011.PROD		
22	**Grand Total**		

Figure 8. Project list from OLAP cube

Once the project was selected, then the user could click on the Field Name dimension to see a list of all Blaise fields in that project's data model (Figure 9).

	A
1	Row Labels
2	SRC.SRO.PSID13.PROD
3	AddrPayment.Dust.DUSTINTRO[1]
4	AddrPayment.ProxyAddr.Addr1
5	AddrPayment.ProxyAddr.Addr2
6	AddrPayment.ProxyAddr.AptSte
7	AddrPayment.ProxyAddr.City
8	AddrPayment.ProxyAddr.Country
9	AddrPayment.ProxyAddr.InCO
10	AddrPayment.ProxyAddr.NamF
11	AddrPayment.ProxyAddr.NamL
12	AddrPayment.ProxyAddr.NamM
13	AddrPayment.ProxyAddr.State
14	AddrPayment.ProxyAddr.Suffix
15	AddrPayment.ProxyAddr.Title
16	AddrPayment.ProxyAddr.Zip
17	AddrPayment.RMailAddr.Addr1
18	AddrPayment.RMailAddr.Addr2
19	AddrPayment.RMailAddr.AptSte
20	AddrPayment.RMailAddr.City
21	AddrPayment.RMailAddr.Country
22	AddrPayment.RMailAddr.InCO
23	AddrPayment.RMailAddr.NamF
24	AddrPayment.RMailAddr.NamL
25	AddrPayment.RMailAddr.NamM
26	AddrPayment.RMailAddr.State
27	AddrPayment.RMailAddr.StateAbbrev
28	AddrPayment.RMailAddr.Suffix

Figure 9. Field list from OLAP cube

From there, the user would simply click on the various measures they would like to add to the table, and within literally a few seconds, the data set would be created (Figure 10).

	A	B	C	D	E	F	G
1	Row Labels	AdtRow_Count	Error Hard	Error Soft	Timing_FieldAvg	Move Backup	Data Entry
2	SRC.SRO.PSID13.PROD	4435538	8211	1552	0.1605	280221	3333232
3	AddrPayment.Dust.DUSTINTRO[1]	2102	15	0	0.3391	107	1832
4	AddrPayment.ProxyAddr.Addr1	92	0	0	0.1886	10	75
5	AddrPayment.ProxyAddr.Addr2	102	0	0	0.0381	15	28
6	AddrPayment.ProxyAddr.AptSte	97	0	0	0.0248	13	8
7	AddrPayment.ProxyAddr.City	96	0	0	0.0695	15	70
8	AddrPayment.ProxyAddr.Country	1	0	0	0.0658	0	1
9	AddrPayment.ProxyAddr.InCO	81	0	0	0.0235	8	4
10	AddrPayment.ProxyAddr.NamF	86	0	0	0.0598	9	69
11	AddrPayment.ProxyAddr.NamL	88	0	0	0.0533	16	65
12	AddrPayment.ProxyAddr.NamM	94	0	0	0.0239	11	22
13	AddrPayment.ProxyAddr.State	84	0	0	0.0474	9	65
14	AddrPayment.ProxyAddr.Suffix	78	0	0	0.0188	8	2
15	AddrPayment.ProxyAddr.Title	80	0	0	0.1149	3	41
16	AddrPayment.ProxyAddr.Zip	83	0	0	0.1022	8	67
17	AddrPayment.RMailAddr.Addr1	8536	0	0	0.1096	296	1836
18	AddrPayment.RMailAddr.Addr2	8965	0	0	0.0282	848	98

Figure 10. Field-level summary report

At that point, the user could simply uncheck the “Field Name” dimension and select some other dimension, such as the Interviewer Name or Sample Id, to almost instantaneously change the grouping of the same measures. (Note in Figure 11 that the first column has changed from field name to sample Id). In this way, users can point and click various combinations of dimensions and measures to very quickly create data sets that could then be analyzed in other programs.

	A	B	C	D	E	F	G
1	Row Labels	AdtRow_Count	Error Hard	Error Soft	Timing_FieldAvg	Move Backup	Data Entry
2	SRC.SRO.PSID13.PROD	4435538	8211	1552	0.1605	280221	3333232
3	0004151	672	3	0	0.1766	34	547
4	0004192	475	0	0	0.2286	16	392
5	0004312	432	0	0	0.2092	13	342
6	0004712	693	1	2	0.2748	36	581
7	0005001	685	0	0	0.1878	18	574
8	0005002	477	1	0	0.1814	24	374
9	0005003	678	1	0	0.1869	38	532
10	0005311	688	0	0	0.1384	88	479
11	0005511	592	1	0	0.2119	24	492
12	0006001	680	0	0	0.2121	25	550
13	0006002	645	3	0	0.1889	21	528
14	0006003	407	1	0	0.2389	17	283
15	0006141	610	0	0	0.1793	17	506

Figure 11. Sample ID-level report

As mentioned above, visualization can also be added to assist analysis. Figure 12 shows a chart that was created in less than a minute, showing a comparison of average interview length over time between two groups of sample.

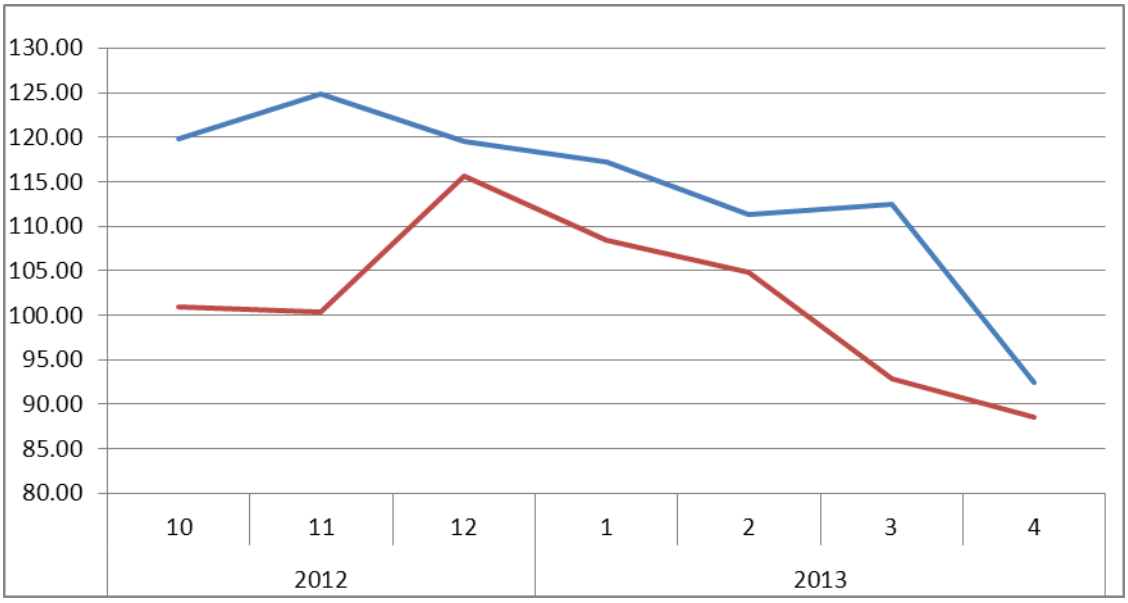

Figure 12. Pivot chart of average interview length over time between two sample groups.

5. Further Development and Conclusion

5.1. Strengths and Weaknesses of the OLAP approach

Using the OLAP cube approach for analyzing and reporting on ADT data offers some very compelling advantages. First and foremost, OLAP cubes are extremely flexible, inviting data exploration and new query ideas through an interface that does not require advanced programming skills. It additionally takes the burden of processing times and complex data joins off of the end user, so that they can concentrate on quickly answering the questions of interest. Cubes are also very customizable, offering the ability to define limited views, hierarchical relationships, and options to pull together data from a variety of sources. They are also accessible with familiar tools, such as Excel. Cube data can very easily be used as the basis of "dashboards" or other data visualization tools with key metrics to help keep the pulse of data collection. Finally, since the data are stored in aggregated form, it is easy to strip out all personally identifying information (PII) so that access to the ADT data can be granted more widely than it otherwise might.

However, OLAP cubes are not without their own set of constraints and limitations. While it seemed very well suited for this particular application, building and maintaining an OLAP cube is also a complicated endeavor, and can easily be overkill for data sets that do not necessarily need to be heavily restructured, combined, or aggregated. While they remove a lot of burden from the analyst, that burden is generally shifted to the IT support staff or database administrators (i.e., whoever designed and maintains the cube). This can lead to a situation where certain aspects of the data structure are necessarily under centralized control, and changes to the cube, while usually relatively easy after the initial design phase, could still be subject to potential bottlenecks.

Additionally, cubes can be deceptively easy to use, especially when working with tools such as Excel pivot tables. It is easy, for example, to build a data set or comparison from the underlying cube. What is more difficult is to carefully think through the goals of the analysis, or what the various calculations mean from a practical perspective. While the tool is flexible, it can also be overwhelming to users, at least until some initial familiarity is developed.

Finally, while it is a significant advantage to be able to join ADT data to other sources and store it in one central location, this can also present challenges related to harmonization among different studies that may have each included a slightly different set of variables, or stored the same variables in different locations or named them differently, and so forth. Attempting to create a one-size-fits-all cube to hold all projects can consequently make the ETL process extremely complicated, as project-specific variables are recoded into standardized variables.

5.2. Potential Enhancements

The ADT OLAP cube is still a relatively young phenomenon at SRO, and it will be interesting to see how it grows and evolves over time, or indeed if other cubes are constructed to summarize other kinds of paradata altogether. What seems most likely in terms of short-term future enhancements are refinements and/or additions to the list of measures and dimensions, as well as the addition of some other sources of data, such as interviewer timekeeping and interview evaluation or verification data. Additional projects will continue to be loaded into the cube, and further thought will need to be given about data harmonization among studies. In the long term, perhaps the ADT cube could grow into just one aspect of a much larger cube that captures most, if not all, of the paradata we collect.

Finally, it is worth mentioning that many off-the-shelf products are available to connect to and work with OLAP cubes once they are deployed. Many of these products are web-based, and once installed on a web server, can provide a similar point-and-click user interface, but perhaps more easily accessible. Depending on the needs for this level of access, we may consider such web-based tools in the future.

6. References

Devonshire, J., Liu, Y, & Cheung, G. (2012). Blaise Audit Trail Data in a Relational Database. *Complete Volume of 14th IBUC 2012.* (38-46) Paper presented at the 14th International Blaise Users Conference, London, UK.. NatCen

Evelson, B. (2008). Topic Overview: Business Intelligence. Retrieved from Forrester Research database.

Nicolas, G. (n.d.). Oracle Olap. GerardNicocom Weblog RSS. Retrieved Aug. 2013. From http://gerardnico.com/wiki/database/oracle/oracle_olap

7. Appendix

Table 1. Structure of table ADT OLAP cube (Measures, Dimensions, and Attributes)

TYPE	MEASUREGROUP	NAME	MEASURE_DESC
Measure	Adt_Row	Last Processed Date	Date of last ETL data refresh
Measure	Adt_Row	AdtRow_Count	Count of all rows of ADT data (i.e., field entries).
Measure	Adt_Row	Blaise Help	Count of Blaise Help instances.
Measure	Adt_Row	Ctrl L Set Lang	Count of Ctrl-L (set/change language) instances.
Measure	Adt_Row	Ctrl_D	Count of Ctrl-D (“Don't Know” hot key) instances.
Measure	Adt_Row	Ctrl_R	Count of Ctrl-R (“Refuse” hot key) instances.
Measure	Adt_Row	Data Entry	Count where field LEAVE_VALUE <> field ENTER_VALUE
Measure	Adt_Row	Data Prepopulated	Count where field ENTER_VALUE <> Empty
Measure	Adt_Row	Enter_Date_Max	Max of Enter Date.
Measure	Adt_Row	Enter_Date_Min	Min of Enter Date.
Measure	Adt_Row	Enter_Time_Min	Min of enter Time.
Measure	Adt_Row	Error Esc	Count of “Escape” responses to Blaise error.
Measure	Adt_Row	Error Hard	Count of Hard Blaise error instances.
Measure	Adt_Row	Error Jmp	Count of Jump responses to Blaise error (i.e., jumps to another question).
Measure	Adt_Row	Error Soft	Count of Soft Blaise error instances.
Measure	Adt_Row	Error Supp	Count of “Suppress” responses to Blaise error.
Measure	Adt_Row	Field Recorded	Distinct Count of recorded fields/questions.
Measure	Adt_Row	FieldVstNum_Max	Max Blaise field visit number. (i.e., number of times the field was visited)

Measure	Adt_Row	FormVstNum_Max	Max Blaise form (instrument) visit number. (i.e., number of suspends)
Measure	Adt_Row	Keystrokes_FieldAvg	Average number of keystrokes for the Blaise field.
Measure	Adt_Row	Keystrokes_FieldStDev	Standard deviation of keystrokes for the Blaise field.
Measure	Adt_Row	Keystrokes_Sum	Count of keystrokes.
Measure	Adt_Row	Media Start	Count of times media was started from within Blaise.
Measure	Adt_Row	Mouse Click	Count of times the mouse was clicked from within Blaise.
Measure	Adt_Row	Move Backup	Count of times iwer backed up to previous field.
Measure	Adt_Row	Q Help	Count of QXQ Help instances.
Measure	Adt_Row	Remark Entered	Count of times a new remark was entered.
Measure	Adt_Row	Remark Modifed	Count of times an existing remark was modified.
Measure	Adt_Row	Set Lang	Count of times the language was set on this row.
Measure	Adt_Row	Suspend	Count of times a suspend was initiated on this row.
Measure	Adt_Row	Timing_Btw_Field_Avg	Average time (sec.) spent between leaving one field and entering the next.
Measure	Adt_Row	Timing_FieldAvg	Average time (min.) spent in the Blaise field.
Measure	Adt_Row	Timing_FieldMax	Max time (min.) spent in the Blaise field.
Measure	Adt_Row	Timing_FieldMedian	Median time (min.) spent in the Blaise field.
Measure	Adt_Row	Timing_FieldMin	Min time (min.) spent in the Blaise field.
Measure	Adt_Row	Timing_FieldStDev	Standard deviation of time (min.) spent in the Blaise field.
Measure	Adt_Row	Timing_Sum	Sum of time (min.) spent in the Blaise field.
Measure	Distinct_Counts_ Ratios	Distinct_Field_Count	Distinct Count of Blaise fields.
Measure	Distinct_Counts_ Ratios	Distinct_Field_Count_IwAvg	Average number of distinct fields within an interview.
Measure	Distinct_Counts_ Ratios	TotalRows_DistinctField_Ratio	AdtRow_Count/ Distinct_Field_Count
Measure	Distinct_Counts_ Ratios_More	Distinct_DataEntry_Count	Count of distinct fields for which data was entered.
Measure	Sample_Line_Row	Enter_Date_Range	Number of days between first and last ENTER_DATE
Measure	Sample_Line_Row	Keystrokes_IwAvg	Average number of keystrokes for the Interview
Measure	Sample_Line_Row	Keystrokes_IwStDev	Standard deviation of keystrokes for the Interview
Measure	Sample_Line_Row	PSID_PostIw_Tasks_Time_IwAvg	PSID-specific timing variable.
Measure	Sample_Line_Row	PSID_R_Burden_Time_IwAvg	PSID-specific timing variable.
Measure	Sample_Line_Row	PSID_Total_Iw_Time_Avg	PSID-specific timing variable.
Measure	Sample_Line_Row	Sample_Line_Count	Distinct count of sample lines.

Measure	Sample_Line_Row	Timing_IwAvg	Average time (min.) spent in the Interview
Measure	Sample_Line_Row	Timing_IwMax	Max time (min.) spent in the Interview
Measure	Sample_Line_Row	Timing_IwMedian	Median time (min.) spent in the Interview
Measure	Sample_Line_Row	Timing_IwMin	Min time (min.) spent in the Interview
Measure	Sample_Line_Row	Timing_IwStDev	Standard deviation of time (min.) spent in the Interview
Dimension	N/A	Block	Blaise Block name
Dimension	N/A	Data_Model_Name	Name of the data model
Dimension	N/A	Data_Model_Date	Release date of data model version
Dimension	N/A	Field	Blaise Field name
Dimension	N/A	FieldVstNum	The visit number of the Blaise field.
Dimension	N/A	FormVstNum	The visit number of the Blaise form (instrument).
Dimension	N/A	Leave_Field_Cause	The cause of leaving the current field (e.g., ENTER, move back, suspend, etc.)
Dimension	N/A	Project	Project name
Dimension	N/A	Sample_Line	Sample line ID
Attribute	Dim: Field	Custom_Var1	Custom-defined variable representing any combination of Blaise fields.
Attribute	Dim: Field	Custom_Var2	Custom-defined variable representing any combination of Blaise fields.
Attribute	Dim: Field	Custom_Var3	Custom-defined variable representing any combination of Blaise fields.
Attribute	Dim: Sample_Line	Interviewer_Hierarchy	Drilldown from Project, PM, PC, TL, and interviewer levels
Attribute	Dim: Sample_Line	Day of Week	Integer that corresponds to the day of the week (from Iw result date); 1=Mon
Attribute	Dim: Sample_Line	DRI Consent	Whether consent was provided to DRI recording (1=yes)
Attribute	Dim: Sample_Line	DRI Recorded	Whether the sample line was recorded (1=yes)
Attribute	Dim: Sample_Line	Evaluation Score	From OLIVE database; evaluation score.
Attribute	Dim: Sample_Line	HRS Cohort	Text label of HRS cohorts
Attribute	Dim: Sample_Line	HRS Language	Text label of HRS language
Attribute	Dim: Sample_Line	HRS Pref Mode	Text label of HRS preferred mode
Attribute	Dim: Sample_Line	Instrument Type	SELF, EXIT, P-EXIT
Attribute	Dim: Sample_Line	Interviewer Name	Interviewer Name (without hierarchy)
Attribute	Dim: Sample_Line	Iwer Experience	Iwer experience level (should be standardized across projects)
Attribute	Dim: Sample_Line	Iwer Field SSL	Interviewer designation, field vs. lab (SSL)
Attribute	Dim: Sample_Line	Iwer Gender	Text label of interviewer's gender.
Attribute	Dim: Sample_Line	Iwer Timezone	Timezone of iwer's home location.
Attribute	Dim: Sample_Line	Month	Integer that corresponds to the month of iw (from Iw result date).
Attribute	Dim: Sample_Line	Project Name	Name of project (only for SurveyTrak projects).

Attribute	Dim: Sample_Line	PSID cell complete	Indicator of whether PSID interview was completed by cell phone.
Attribute	Dim: Sample_Line	PSID Sample Type	Text label of PSID sample type.
Attribute	Dim: Sample_Line	Release	From SurveyTrak, tSample_Line.sRelease
Attribute	Dim: Sample_Line	Resistance Flag	From SurveyTrak, tSample_Line.blRCIndFlag
Attribute	Dim: Sample_Line	Result Code	From SurveyTrak, tSample_Line.sResultCodeID
Attribute	Dim: Sample_Line	Result Date	From SurveyTrak, tSample_Line.dResultDate
Attribute	Dim: Sample_Line	Sample Line	From SurveyTrak, tSample_Line.vSample_LineID
Attribute	Dim: Sample_Line	Year	Year of interview (from lw result date)