Dynamic Workload Driven Data Integration in Tableau

Kristi Morton*
University of Washington
kmorton@cs.washington.edu

Ross Bunker, Jock Mackinlay, Robert Morton, and Chris Stolte
Tableau Software
{rbunker,jmackinlay,rmorton,cstolte}@tableausoftware.com

ABSTRACT

Tableau is a commercial business intelligence (BI) software tool that supports interactive, visual analysis of data. Armed with a visual interface to data and a focus on usability, Tableau enables a wide audience of end-users to gain insight into their datasets. The user experience is a fluid process of interaction in which exploring and visualizing data takes just a few simple drag-and-drop operations (no programming or DB experience necessary). In this context of exploratory, ad-hoc visual analysis, we describe a novel approach to integrating large, heterogeneous data sources. We present a new feature in Tableau called *data blending*, which gives users the ability to create data visualization mashups from structured, heterogeneous data sources dynamically without any upfront integration effort. Users can author visualizations that automatically integrate data from a variety of sources, including data warehouses, data marts, text files, spreadsheets, and data cubes. Because our data blending system is workload driven, we are able to bypass many of the pain-points and uncertainty in creating mediated schemas and schema-mappings in current pay-as-you-go integration systems.

Categories and Subject Descriptors: H.5.0 [**Information Systems**]: Information Interfaces and Presentation

General Terms: Design, Human Factors, Management

Keywords: Data Integration, Visualization

1. INTRODUCTION

Unlike databases, human brains have limited capacity for managing and making sense of large collections of data. In database terms, the feat of gaining insight in big data is often accomplished by issuing aggregation and filter queries. Yet, this approach is time-consuming as the user is forced to 1) figure out what queries to write, 2) write the queries, 3) wait for the results to be returned back in textual format, and finally 4) read through these textual summaries (often containing thousands of rows) to search for interesting patterns or anomalies. Tools like Tableau help bridge this gap by providing a visual interface to the data. This approach removes the burden of writing queries and has the user ask their questions through visual drag-and-drop operations (no queries or programming experience required). Additionally, answers are displayed visually, where patterns and outliers can quickly be identified.

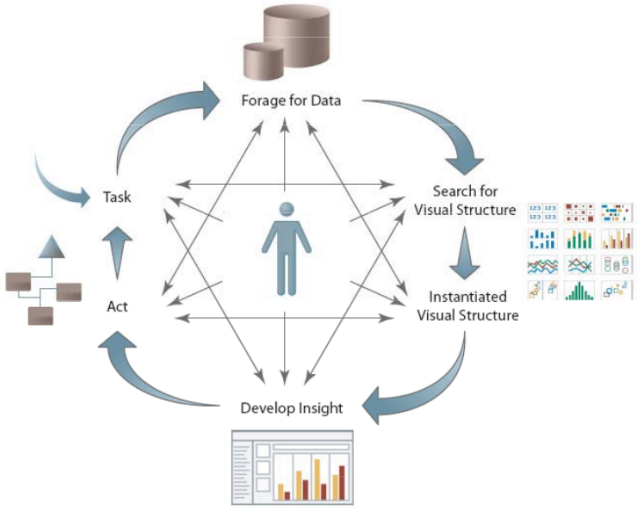


Figure 1: Cycle of Visual Analysis

Visualizations leverage the powerful human visual system to effectively digest large amounts of information. Figure 1 illustrates how visualization is a key component in the sense-making model [1], *i.e.* the theory of how people search for, organize, and generate new knowledge from data. The process starts with some task or question that a knowledge worker (shown at the center) seeks to gain understanding. In the first stage, the user forages for data that may contain relevant information for their analysis task. Next, they search for a visual structure that is appropriate for the data and instantiate that structure. At this point, the user interacts with the resulting visualization (*e.g.* drill down to details or roll up to summarize) to develop further insight. Once the necessary insight is obtained, the user can then make an informed decision and take action. This cycle is

*Work done as intern at Tableau Software, Seattle.

SIGMOD '12, May 20–24, 2012, Scottsdale, Arizona, USA.

centered around and driven by the user and requires that the visualization system be flexible enough to support user feedback and allow alternative paths based on the needs of the user's exploratory tasks. Most visualization tools [4, 5, 13], however, treat this cycle as a single, directed pipeline, and offer limited interaction with the user.

Moreover, users often want to ask their analytical questions over multiple data sources. However, the task of setting up data for integration is orthogonal to the analysis task at hand, requiring a context switch that interrupts the natural flow of the analysis cycle. We extend the visual analysis cycle with a new feature called *data blending* that allows the user to seamlessly combine and visualize data from multiple different data sources on-the-fly. Our blending system issues live queries to each data source to extract the minimum information necessary to accomplish the visual analysis task. Often, the visual level of detail is at a coarser level than the data sets. Aggregation queries, therefore, are issued to each data source before the results are copied over and joined in Tableau's local in-memory view. We refer to this type of join as a *post-aggregate join* and find it a natural fit for exploratory analysis, as less data is moved from the sources for each analytical task, resulting in a more responsive system. Finally, Tableau's data blending feature automatically infers how to integrate the datasets on-the-fly, involving the user only in resolving conflicts. This system also addresses a few other key data integration challenges, including combining datasets with mismatched domains or different levels of detail and dirty or missing data values. One interesting property of blending data in the context of a visualization is that the user can immediately observe any anomalies or problems through the resulting visualization.

These aforementioned design decisions were grounded in the needs of Tableau's typical BI user base. Thanks to the availability of a wide-variety of rich public datasets from sites like data.gov, many of Tableau's users integrate data from external sources such as the Web or corporate data such as internally-curated Excel spreadsheets into their enterprise data warehouses to do predictive, what-if analysis. However, the task of integrating external data sources into their enterprise systems is complicated. First, such repositories are under strict management by IT departments, and often IT does not have the bandwidth to incorporate and maintain each additional data source. Second, users often have restricted permissions and cannot add external data sources themselves. Such users cannot integrate their external and enterprise sources without having them collocated. An alternative approach is to move the data sets to a data repository that the user has access to, but moving large data is expensive and often untenable. We therefore architected data blending with the following principles in mind: 1) move as little data as possible, 2) push the computations to the data, and 3) automate the integration challenges as much as possible, involving the user only in resolving conflicts.

The rest of this paper is organized as follows. First in Section 2 we present a brief overview of Tableau and data blending. Then in Section 3 we discuss the use-case scenarios and underlying principles that drove the design of our data blending architecture. Section 4 describes our overall approach to data blending. Section 5 covers related work. Finally, in Section 6, we discuss interesting research directions, and in Section 7 we conclude.

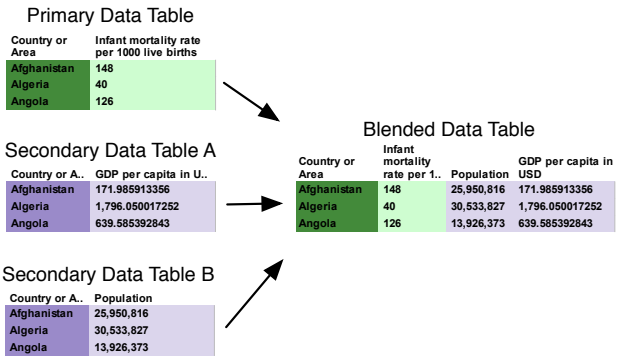


Figure 2: Small Sample Tables of Infant Mortality Rates, GDP, and Population in 2000

2. BACKGROUND

In this section we describe Tableau and data blending at a high-level. To make the concepts more concrete, we discuss a simple example that blends data from three data sources to produce a compelling visualization.

2.1 Tableau Overview

Tableau [11] is a data visualization tool that sits between the end-user and the database and allows the user to create visualizations by dragging and dropping fields from their datasets onto a visual canvas. In response to these actions, Tableau generates formal VizQL (Visual Query Language) [10] statements to build the requested visualization. VizQL is a structured query language with support for rendering graphics. Each VizQL statement is compiled into the SQL or MDX queries necessary to generate the data for the visualization.

2.2 Blending Data Overview

The *data blending* feature, released in Tableau 6.0, allows an end-user to dynamically combine and visualize data from multiple heterogeneous sources without any upfront integration effort. A user authors a visualization starting with a single data source – known as the primary – which establishes the context for subsequent blending operations in that visualization. Data blending begins when the user drags in fields from a different data source, known as a secondary data source. Blending happens automatically, and only requires user intervention to resolve conflicts. Thus the user can continue modifying the visualization, including bringing in additional secondary data sources, drilling down to finer-grained details, etc., without disrupting their analytical flow. The novelty of this approach is that the entire architecture supporting the task of integration is created at runtime and adapts to the evolving queries in typical analytical workflows.

2.3 Simple Illustrative Example

In this section we discuss a scenario in which three unique data sources (see left half of Figure 2 for sample tables) are blended together to create the visualization shown in Figure 3[1]. This is a simple, yet compelling mashup of three

[1] http://public.tableausoftware.com/views/InfantMortalityRatein2000/IMRvsGDP

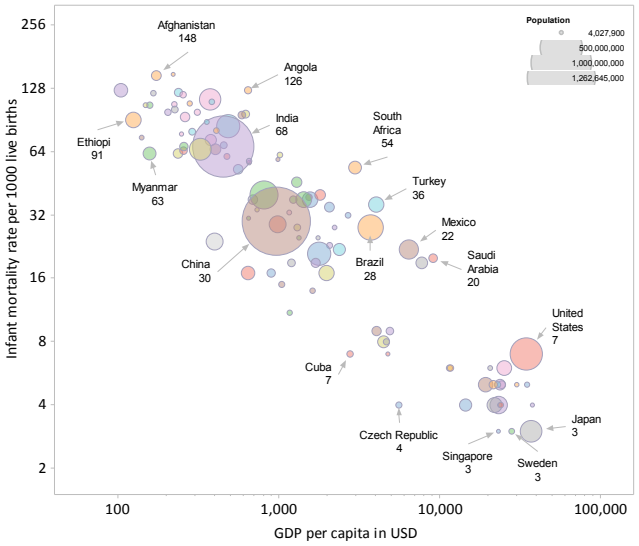


Figure 3: Blended View of Infant Mortality Rates, GDP, and Population in 2000

unique measures that tells an interesting story about the complexities of global infant mortality rates in the year 2000.

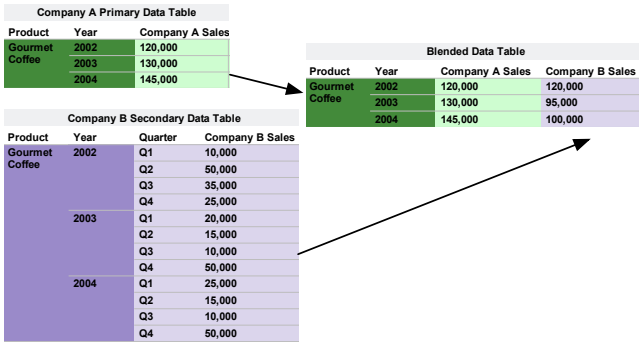
In this example, the Tableau user wants to understand if there is a connection between infant mortality rates, GDP, and population. She has three distinct spreadsheets with the following characteristics: the first data source contains information about the infant mortality rates per 1000 live births for each country, the second contains information about each country's total population, and the third source contains country-level GDP. For this analysis task, the user drags the fields, `"Country or Area"` and `"Infant mortality rate per 1000 live births"`, from her first data source onto the blank visual canvas. Since these fields were the first ones selected by the user, then the data source associated with these fields becomes the primary data source. This action produces a visualization showing the relative infant mortality rates for each country. But the user wants to understand if there is a correlation between GDP and infant mortality, so she then drags the `"GDP per capita in US dollars"` field onto the current visual canvas from Data Table A. The step to join the GDP measure from this separate data source happens automatically: the blending system detects the common join key (i.e. `"Country or Area"`) and combines the GDP data with the infant mortality data for each country. Finally, to complete her analysis task, she adds the `"Population"` measure from Data Table B, to the visual canvas, which produces the visualization in Figure 3 associated with the blended data table in Figure 2.

3. USAGE SCENARIOS AND DESIGN PRINCIPLES

In this section, we discuss when it is advantageous to leverage the data blending feature for integrating datasets. Additionally, we discuss some common use-case scenarios that blending makes more accessible to the end-user.

3.1 Why blend data?

Tableau is equipped with two ways of integrating data. First, in the case where the data sets are collocated (or



Company A Primary Data Table

Product	Year	Company A Sales
Gourmet Coffee	2002	120,000
	2003	130,000
	2004	145,000

Company B Secondary Data Table

Product	Year	Quarter	Company B Sales
Gourmet Coffee	2002	Q1	10,000
		Q2	50,000
		Q3	35,000
		Q4	25,000
	2003	Q1	20,000
		Q2	15,000
		Q3	10,000
		Q4	50,000
	2004	Q1	25,000
		Q2	15,000
		Q3	10,000
		Q4	50,000

Blended Data Table

Product	Year	Company A Sales	Company B Sales
Gourmet Coffee	2002	120,000	120,000
	2003	130,000	95,000
	2004	145,000	100,000

Figure 4: Company A and B Data Tables (left) and Blended Result (right)

can be collocated), Tableau formulates a query that joins them to produce a visualization. However, in the case where the data sets are not collocated (or cannot be collocated), Tableau federates queries to each data source, and creates a dynamic, *blended* view that consists of the joined result sets of the queries. For the purpose of exploratory visual analytics, we find that data blending is a complementary technology to the standard collocated approach with the following benefits:

- Resolves many data granularity problems
- Resolves collocation problems
- Adapts to needs of exploratory visual analytics

Resolves data granularity problems

Oftentimes a user wants to combine data that may not be at the same granularity (*i.e.* they have different primary keys). For example let's say that an employee at company A wants to compare the yearly growth of sales to a competitor company B. The dataset for company B (see Figure 4) contains a detailed quarterly growth of sales for B (quarter, year is the primary key), while company A's dataset only includes the yearly sales (year is the primary key). If the employee simply joins these two datasets on yearly earnings, then each row from A will be duplicated for each quarter in B for a given year resulting in an inaccurate overestimate of A's yearly earnings. This duplication problem can be avoided if for example, company B's sales dataset were first aggregated to the level of year, then joined with company A's dataset. In this case, data blending detects that the data sets are at different granularities by examining their primary keys and notes that in order to join them, the common field is year. In order to join them on year, an aggregation query is issued to company B's dataset, which returns the sales aggregated up to the yearly level as shown in Figure 4.This result is blended with company A's dataset to produce the desired visualization of yearly sales for companies A and B. The blending feature does all of this on-the-fly without user-intervention.

Resolves collocation problems

As mentioned in Section 1, many of Tableau's BI users are faced with challenges in integrating their external data sources into their IT-managed data repositories. Some of them simply cannot collocate their datasets for integration

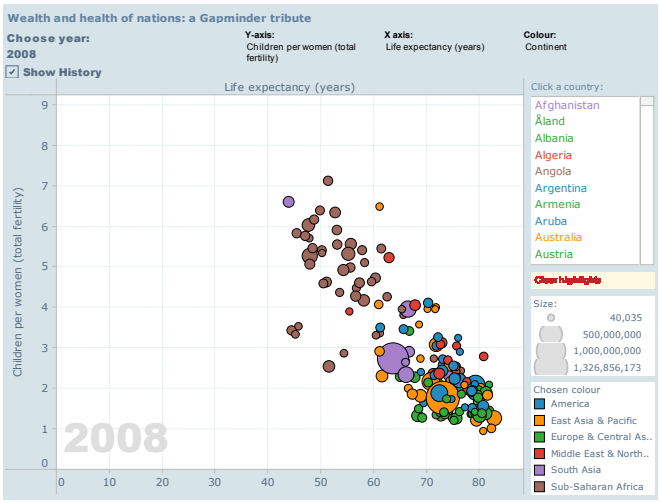


Figure 5: Gapminder tribute demonstrating on-the-fly creation of blended views

either because they lack the appropriate permissions or because moving their large data to their own locally-managed repository is expensive and untenable. In other cases, the data repository may have rigid structure, as with cubes, to ensure performance, support security or protect data quality. Furthermore, it is often unclear if it is worth the effort of integrating an external data set that has uncertain value. The user may not know until she has started exploring the data if it has enough value to justify spending the time to integrate and load it into her repository. Thus, one of the paramount benefits of data blending is that it allows the user to quickly start exploring their data, and as they explore the integration happens automatically as a natural part of the analysis cycle. An interesting final benefit of the blending approach is that it enables users to seamlessly integrate across different types of data (which usually exist in separate repositories) such as relational, cubes, text files, spreadsheets, etc.

Adapts to needs of exploratory visual analytics
A key benefit of data blending is its flexibility; it gives the user the freedom to view their blended data at different granularities and control how data is integrated on-the-fly. The blended views are dynamically created as the user is visually exploring the datasets. For example, the user can drill-down, roll-up, pivot, or filter any blended view as needed during her exploratory analysis. This feature is useful for data exploration and what-if analysis.

3.2 Real User Application Scenarios

In this section we present a couple application scenarios that highlight interesting uses of data blending by Tableau's customers. These examples (along with the illustrative example in Section 2.3) were taken from Tableau Public [12], which is an online repository of published visualizations.

Wealth and health of nations: a Gapminder tribute[2]
A Tableau user created an interactive visualization inspired by Gapminder [8] (see Figure 5), which allows users to

[2]http://public.tableausoftware.com/views/Gapminder/Gapminder

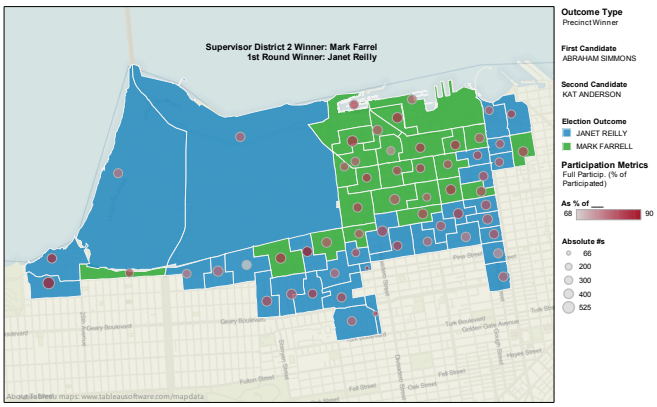


Figure 6: San Francisco Election Outcome Modeling

select and plot various demographic data in a scatterplot, with each data point representing a country. This is a nice example that demonstrates the flexibility of data blending. A user can select from three different data sources to blend on the x-axis (including life expectancy, CO_2 emissions, and population) and three sources for the y-axis (including number of children per woman, energy use, and income per person). This demonstrates the flexibility of the data blending feature, namely that users can dynamically change their blended views by pivoting on different data sources and measures to blend in their visualizations.

Modeling Election Outcomes[3]
Figure 6 illustrates the possible outcomes of an election for District 2 Supervisor of San Francisco. With this type of visualization, the user can select different election styles and see how their choice affects the outcome of the election. What's interesting from a blending standpoint is that this is an example of a many-to-one relationship between the primary and secondary datasets. This means that the fields being left-joined in by the secondary data sources match multiple rows from the primary dataset and results in these values being duplicated. Thus any subsequent aggregation operations would reflect this duplicate data, resulting in overestimates. The blending feature, however, prevents this scenario from occurring by performing all aggregation prior to duplicating data during the left-join.

3.3 Guiding Design Principles

We describe the primary design principles upon which Tableau's data blending feature was based. These principles were influenced by the application needs of Tableau's end-user. In particular, we designed the blending system to be able to integrate datasets on-the-fly, be responsive to change, and driven by the visualization. Additionally, we assumed that the user may not know exactly what she is looking for initially, and needs a flexible, interactive system that can handle exploratory visual analysis.

Push Computation to Data and Minimize Data Movement

Tableau's approach to data visualization allows users to leverage the power of a fast database system. Tableau

[3]http://public.tableausoftware.com/views/SFElections3_1/SD2Map

compiles the VizQL [10] declarative formalism representing a visual specification into SQL or MDX and pushes this computation close to the data, where the fast database system handles computationally intensive aggregation and filtering operations. In response, the database provides a relatively small result set for Tableau to render. This is an important factor in Tableau's choice of post-aggregate data integration across disparate data sources – since the integrated result sets must represent a cognitively manageable amount of information, the data integration process operates on small amounts of aggregated, filtered data from each data source. This approach avoids the costly migration effort to collocate massive data sets in a single warehouse, and continues to leverage fast databases for performing expensive queries close to the data.

Automate as Much as Possible, but Keep User in Loop

Tableau's primary focus has been on ease of use since most of Tableau's end-users are not database experts, but range from a variety of domains and disciplines: business analysts, journalists, scientists, students, etc. We therefore, took a simple, pay-as-you-go [3] integration approach in which the user invests minimal upfront effort or time to receive the benefits of the system. For example, the data blending system does not require the user to specify schemas for their data sets, rather the system tries to infer this information as well as how to apply schema matching techniques to blend them for a given visualization. Furthermore the system provides a simple drag-and-drop interface for the user to specify the fields for a visualization, and if there are fields from multiple data sources in play at the same time, the blending system infers how to join them to satisfy the needs of the visualization.

In the case that something goes wrong, for example, if the schema matching could not succeed, the blending system provides a simple interface for specifying data source relationships and how blending should proceed. Additionally, the system provides several techniques for managing the impact of dirty data on blending, which we discuss in Section 4.8

4. INTEGRATING DATA IN TABLEAU

In this section we discuss in greater detail how data blending works. Then we discuss how a user builds visualizations using data blending using several large datasets involving airline statistics.

4.1 Data Blending Architecture

The data blending system, shown in Figure 7 takes as input the VizQL query workload generated by the user's GUI actions and data source schemas, and automatically infers how to query the data sources remotely and combine their results on-the-fly. The system features a two-tier mediator-based architecture in which the VizQL query workload is analyzed and partitioned at runtime based on the corresponding data source fields being used. The primary mediator initiates this process by removing the visual encodings from the VizQL query workload to yield an *abstract query*. The abstract query is partitioned for further processing by the primary mediator and one or more secondary mediators. The primary mediator creates the mediated schema for the given query workload. It then federates the abstract queries to the

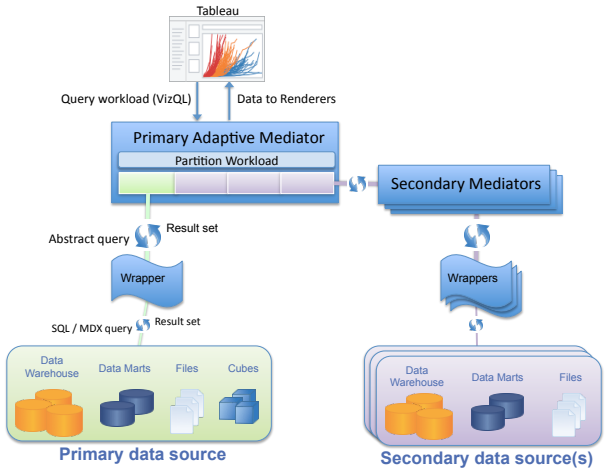


Figure 7: Tableau's Data Blending Architecture

primary data source as well as the secondary mediators and their respective data sources. The wrappers compile the abstract queries into concrete SQL or MDX queries and instantiate the semantic mappings between the data sources and the mediated schema for each query. The primary mediator joins all the result sets returned from all data sources to produce the mediated result set used by the rendering system.

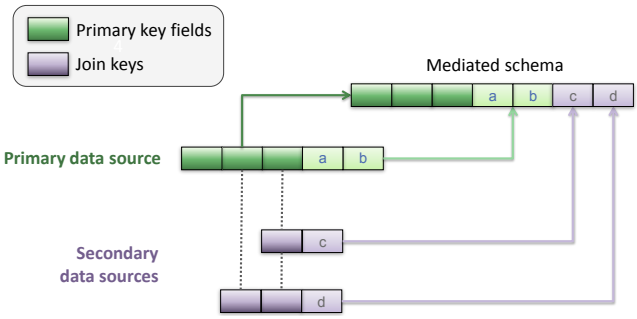


Figure 8: Components of the Mediated Schema

4.2 Post-aggregate Join

A visualization is organized by its discrete fields into pages, partitions, colors, etc., and like a GROUP BY clause in SQL, these *grouping fields* comprise the primary key of the visualization. In a blended visualization, the grouping fields from the primary data source become the primary key of the mediated schema. In Figure 8 these are shown as the dark-green fields in the primary data source, and the light-green fields represent the aggregated data. Each secondary data source must contain at least one field that matches a visualization grouping field in order to blend into the mediated schema. The matching fields in a secondary data source comprise its join key, and fields appear in the GROUP BY clause issued by the secondary mediator wrappers. The aggregated data from the secondary data source, shown in light-purple, is then left-joined along its join key into the mediated result set. We refer to this left-join of aggregated result sets as a *post-aggregate join*.

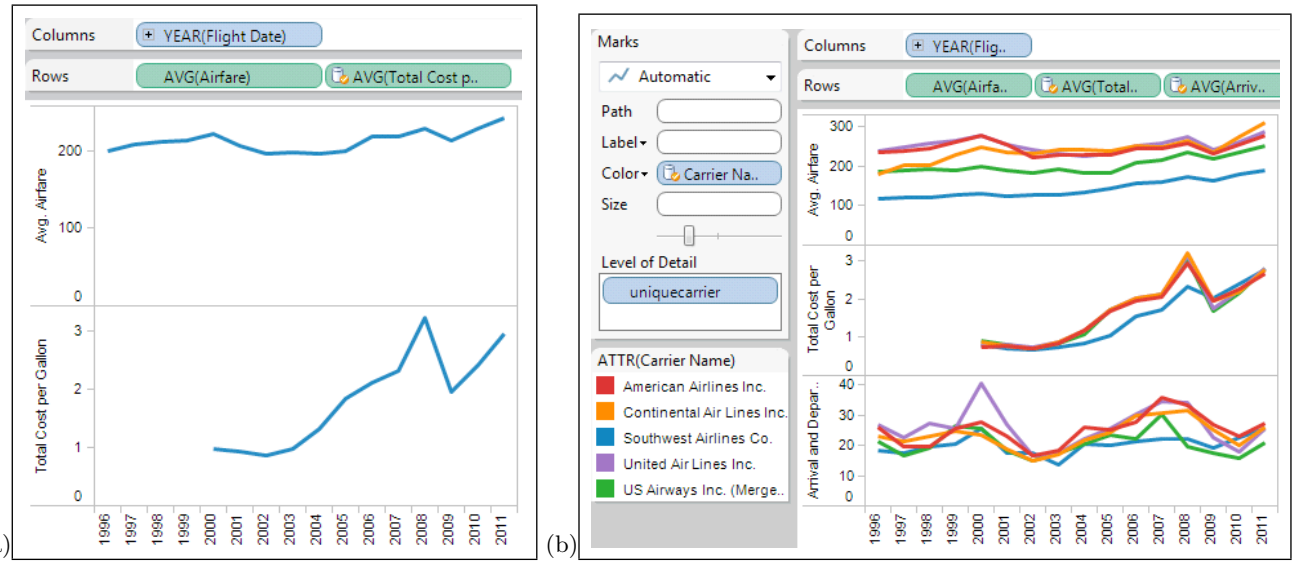


Figure 9: Screenshots of Blended Airline Data from Bureau of Transportation Statistics: (a) Airfare Price (324M rows) and Fuel Costs (8K rows) by year (b) Both datasets from (a), On-Time Performance (140M rows), and Carrier Names (1.5K rows).

4.3 Primary Key Cardinality

As a user blends additional data into her visualization, she is guaranteed that the blended data will not affect the cardinality of the result set. The left-join ensures that no rows from the primary data source result set are lost due to missing data from a secondary data source. Additionally there cannot be a one-to-many mapping between the domain values of the primary key and those of the secondary join key, because the secondary join key is a subset of the primary key and contains only unique values in the aggregated secondary result set. We find that this approach is the most natural for augmenting a visualization with secondary data sources of uncertain value or quality, which is a common scenario for Tableau users.

Data blending supports many-to-one relationships between the primary and each secondary. This can occur when the secondary data source contains coarser-grained data than the mediated result set, as discussed in Section 3.1. Since the join key in a secondary result set may match a subset of the blended result set primary key, portions of the secondary result set may be duplicated across repeated values in the mediated result set. This does not pose risk of double-counting measure values, becaused all aggregation is performed prior to the join. When a blended visualization uses multiple secondary data sources, each secondary join key may match any subset of the primary key. The primary mediator handles duplicating each secondary result set as needed to join with the mediated result set.

Finally, a secondary dimension which is not part of the join key (and thus not a grouping field in the secondary query) can still be used in the visualization. If it is functionally dependent on the join key, a secondary dimension can be used without affecting the result set cardinality. Tableau references this kind of non-grouping dimension using both MIN and MAX aggregations in the query issued to the secondary data source, which allows Tableau to determine if the dimension is functionally dependent on the join key. For each row in the secondary result set, if the two aggregated values are the same then the value is used as-is, reflecting the functional dependence on the grouping fields. If the aggregated values differ, Tableau represents the value using a special form of NULL called *Many Values*. This is represented in the visualization as a '*', but retains the behavior of NULL when used in calculated fields or other computations. The visual feedback allows a user to distinguish this lack of data from the NULLs which occur due to missing or mismatched data.

4.4 Inferring Join Keys

Tableau uses very simple rules for automatically detecting candidate join keys: 1) the secondary data source field name must match a field with the same name in the primary data source, 2) the data types must match 3) if they are date/time fields, they must represent the same granularity date bin in the date/time hierarchy, *e.g.* both are MONTH. A user can intervene to force a match either by providing field captions to rename fields within the Tableau data model, or by explicitly defining a link between fields using a simple user interface.

4.5 Simple Blending Example

A Tableau data blending scenario is shown in Figure 9, which includes multiple views that were composed in minutes by uniquely mashing up four different airline datasets, the largest of which include a 324 million row ticket pricing database and a 140 million row on-time performance database. A user starts by dragging fields from any dataset on to a blank visual canvas, iteratively building a VizQL statement which ultimately produces a visualization. In this example, the user first drags the VizQL fields, `YEAR(Flight Date)` and `AVG(Airfare)`, from the pricing dataset onto the visual canvas.

Data blending occurs when the user adds fields from a separate dataset to an existing VizQL statement in order to augment their analysis. Tableau assigns the existing

dataset to the primary mediator and uses secondary mediators to manage each subsequent dataset added to the VizQL. The mediated schema has a primary key composed of the grouping VizQL fields from the primary dataset (*e.g.* `YEAR(Flight Date)`); the remaining fields in the mediated schema are the aggregated VizQL fields from the primary dataset along with the VizQL fields from each secondary dataset. Continuing our example, the user wishes to drag `AVG(Total Cost per Gallon)` from the fuel cost dataset to the visualization. The schema matching algorithm examines the secondary dataset for one or more fields whose name exactly matches a field in the primary key of the mediated schema. While the proposed matches are often sufficient and acceptable, the user can specify an override. Since the fuel cost dataset has a field named `Date`, the user provides a caption of `Flight Date` to resolve the schema discrepancy.

At this point the mediated schema is created and the VizQL workload is then federated to the wrappers for each dataset. Each wrapper compiles VizQL to SQL or MDX for the given workload, executes the query, and maps the result set into the intermediate form expected by the primary mediator. The mapping is performed dynamically, since both the VizQL and the data model evolve during a user's iterative analytical workflow. Finally, the primary mediator performs a left-join of each secondary result set along the primary key of the mediated schema. In this example, the mediated result set is rendered to produce the visualization shown in Figure 9(a).

4.6 Evolved Blending Example

Figure 9(b) shows further evolution of the analysis of airline datasets, and demonstrates several key points of data blending. First, the user adds a unique ID field named `uniquecarrier` from the primary dataset to the VizQL to visualize results for each airline ID over time. The mediated schema adapts by adding this field to its primary key, and the secondary mediator automatically queries the fuel cost dataset at this finer granularity since it too has a field named `uniquecarrier`. Next, the user decorates the visualization with descriptive airline names for each airline ID by dragging a field named `Carrier Name` from a lookup table. This dataset is at a coarser granularity than the existing mediated schema, since it does not represent changes to the carrier name over time. Our system automatically handles this challenge by allowing the left-join to use a subset of the mediated result set primary key, and replicating the carrier name across the mediated result set. Figure 10 demonstrates this effect using a tabular view of a portion of the mediated result set, along with portions of the primary and secondary result sets. The figure also demonstrates how the left-join preserves data for years which have no fuel cost records. Last, the user adds average airline delays from a 140 million row dataset which matches on `Flight Date` and `uniquecarrier`. This is a fast operation, since the wrapper performs mapping operations on the relatively small, aggregated result set produced by the remote database. Note that none of these additional analytical tasks required the user to intervene in data integration tasks, allowing their focus to remain on finding insight in the data.

4.7 Filtering

Tableau provides several options for filtering data. Data may be filtered based on aggregate conditions, such as excluding airlines having a low total count of flights. A user can filter aggregate data from the primary and secondary data sources in this fashion, which results in rows being removed from the mediated result set. In contrast, row-level filters are only allowed for the primary data source. To improve performance of queries sent to the secondary data sources, Tableau will filter the join keys to exclude values which are not present in the domain of the primary data source result set, since these values would be discarded by the left-join.

4.8 Data Cleaning Capabilities

As mentioned in Section 4.4, Tableau supports user intervention in resolving field names when schema matching fails. And once the schemas match and data is blended, the visualization can help provide feedback regarding the validity of the underlying data values and domains. If there are any data inconsistencies, users can provide aliases for a field's data values which will override the original values in any query results involving that field. The primary mediator performs a left-join using the aliases of the data values, allowing users to blend data despite discrepancies from data entry errors and spelling variations. Tableau provides a simple user interface for editing field aliases.

Calculated fields are another aspect of Tableau's data model which support data cleaning. Calculated fields support arbitrary transformations of original data values into new data values, such as trimming whitespace from a string or constructing a date from an epoch-based integer timestamp. As with database fields, calculated fields can be used as primary keys or join keys.

Last, Tableau allows users to organize a field's related data values into groups. These *ad-hoc groups* can be used for entity resolution, such as binding multiple variations of business names to a canonical form. Ad-hoc groups also allow constructing coarser-grained structures, such as grouping states into regions. Data blending supports joins between two ad-hoc groups, as well as joins between an ad-hoc group and a string field.

5. RELATED WORK

For integrating and querying heterogeneous data, we discuss two primary differentiating axes: 1) how data is fetched and 2) the level of detail at which the datasets are joined. For the first axis, the data integration system either moves the data into a local data store (referred to as a *materialized approach*) or leaves the data at the sources, where data is brought in on-demand from each source and combined on-the-fly using a distributed query processing engine (referred to as a *virtual approach*). For the second axis, data that is joined at the row level and the result is aggregated is called a *pre-aggregation join* and data that is first aggregated for each data source and then joined is a *post-aggregation join*.

Unlike prior work in the context of integrating heterogeneous data for visual analysis [4, 5, 6, 9], Tableau employs a virtual, post-aggregation approach. The virtual approach is well-suited to interactivity, as it allows the user to quickly start asking questions of their data, and leverages the power of the source databases by pushing the computations to them. With the materialized approach, however, the user pays the cost up front of moving the data to a local repository – this approach is not as amenable to exploratory

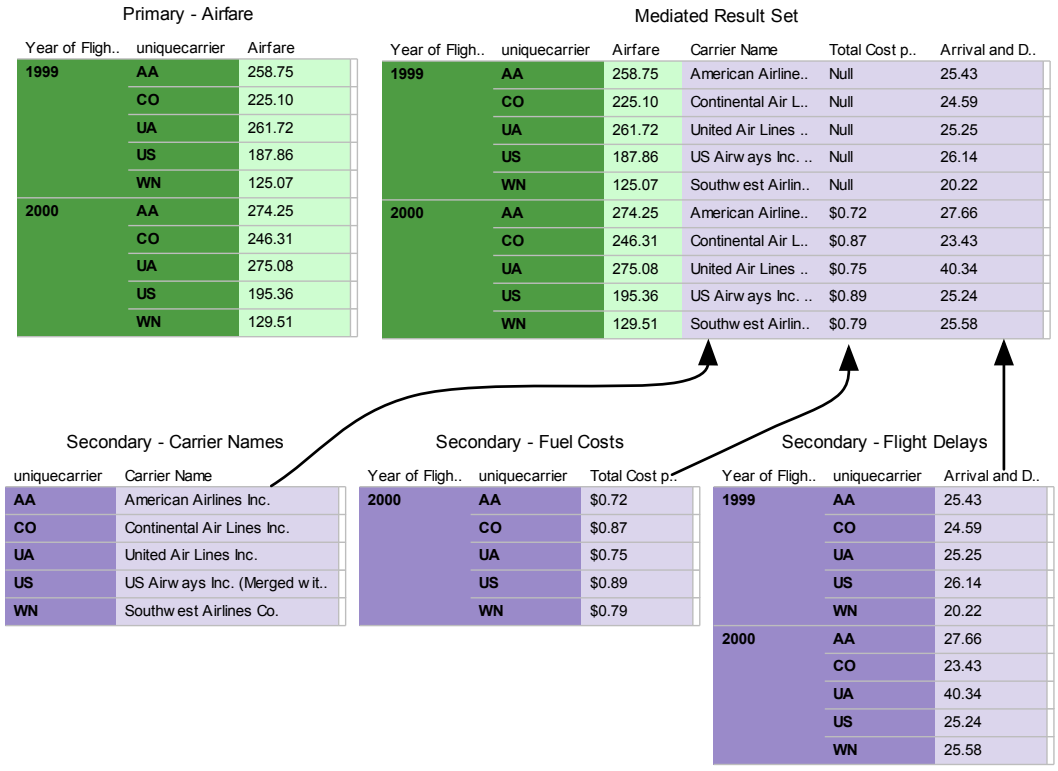


Figure 10: Sample Data Tables for Airline Blending Example

analysis where the user may not know in advance what data will be of interest initially.

While our architecture resembles a typical mediator-based, pay-as-you-go integration system [3, 4, 5, 7, 2], the primary difference is that the mediated schema and mappings are created on-the-fly for each VizQL query workload. Additionally, current systems either 1) place the burden on the user to orchestrate some or all data integration tasks in advance or 2) try to automate the integration, but create the mediated schemas and semantic mappings before the end-user issues her queries. In either case, the system does not leverage the context of the workload queries to guide the integration effort or constrain the space of possible matches.

6. FUTURE WORK

The simplicity of the blending feature creates some artificial restrictions on how data is combined. For example, the system requires that the join key be an exact match on fields from the primary and secondary datasets. As a result, the visualization is forced to proceed at the granularity dictated by the join key. As a simple example, let's say a user has a primary dataset containing per employee salary information and a secondary dataset containing organization information about employees and the team they belong to for a given company. Furthermore, the users wants to visualize the average salary per team, which means that the visualization needs to be at the granularity of "team". Currently, the blending system would detect that the common field to blend on is "employee" and not "team". One solution is to give the user control over how to join their datasets by explicitly specifying the join key. Another solution is to automatically infer the appropriate join key from the desired visualization granularity by leveraging functional dependencies. In the case that there are multiple join key candidates that are functionally dependent, one performance optimization strategy could be to pick the candidate that is at the coarsest granularity.

Additionally, the blending system currently only supports left-join operations. A few extensions that our customers have suggested include supporting other types of operators such as inner joins and unions. Furthermore, customers have expressed interest in being able to filter their primary data sources based on a filter condition on the secondary data source. This feature would result in less data being moved. Finally, we would like to improve our schema matching heuristic by additionally considering the domains of the attributes, since the current matching scheme is quite restrictive (*i.e.* only matching on attribute names and types).

7. CONCLUSION

Data blending is a new feature in Tableau that simplifies the process of integrating data from a variety of heterogeneous sources. We presented several compelling customer applications that leverage the feature and highlight its usefulness for interactive visual analysis. Additionally, we are engaging with our customers to improve the flexibility and capabilities of the feature. Ultimately, our goal is to make it easier for users to discover and combine data that will help them in their exploratory analysis.

8. ACKNOWLEDGEMENTS

We thank Magdalena Balazinska, AnHai Doan, Dan Grossman, Alon Halevy, and Richard Wesley for their helpful feedback on earlier versions of this paper.

9. REFERENCES

[1] S.K. Card,J. Mackinlay,B. Shneiderman. *Readings in Information Visualization: Using Vision To Think*. Morgan-Kaufmann: San Francisco, CA, 1999.

[2] A. Das Sarma, X. Dong, and A. Halevy. Bootstrapping Pay-As-You-Go Data Integration Systems In *SIGMOD*, 2008.

[3] M. Franklin, A. Halevy, and D. Maier. From databases to dataspaces: a new abstraction for information management. In *SIGMOD Rec.*, 2005.

[4] H. Gonzalez, A. Halevy, C. Jensen, A. Langen, J. Madhavan, R. Shapley, and W. Shen. Google Fusion Tables: Data Management, Integration and Collaboration in the Cloud. In *SoCC*, 2010.

[5] H. Gonzalez, A. Halevy, C. Jensen, A. Langen, J. Madhavan, R. Shapley, W. Shen, and J. Goldberg-Kidon. Google Fusion Tables: Web-Centered Data Management and Collaboration. In *SIGMOD Conference*, 2010.

[6] QlikView, 2012. `http://www.qlikview.com`.

[7] M. Hentschel, L. Haas, R. Miller. Just-in-time Data Integration in Action In *VLDB*, 2010.

[8] H. Rosling. GapMinder Foundation, 2012. `http://www.gapminder.org/`

[9] Spotfire, 2012. `http://spotfire.tibco.com`.

[10] C. Stolte, D. Tang, and P. Hanrahan. Polaris: a system for query, analysis, and visualization of multidimensional databases. In *IEEE Transaction on Visualization and Computer Graphics*, 2002.

[11] Tableau Software, 2012. `http://www.tableausoftware.com`.

[12] Tableau Software, 2012. `http://www.tableaupublic.com`.

[13] F. B. Viegas, M. Wattenberg, F. van Ham, J. Kriss, M. McKeon. Many Eyes: A Site for Visualization at Internet Scale. In *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1121-1128, Nov/Dec, 2007.