IDEA: Interactive Data Exploration and Analysis

Peter G. Selfridge
AT&T Research
pgs@research.att.com

Divesh Srivastava
AT&T Research
divesh@research.att.com

Lynn O. Wilson
AT&T
low@ulysses.att.com

Abstract

The analysis of business data is often an ill-defined task characterized by large amounts of noisy data. Because of this, business data analysis must combine two kinds of intertwined tasks: *exploration* and *analysis*. Exploration is the process of finding the appropriate subset of data to analyze, and analysis is the process of measuring the data to provide the business answer. While there are many tools available both for exploration and for analysis, a single tool or set of tools may not provide full support for these intertwined tasks. We report here on a project that set out to understand a specific business data analysis problem and build an environment to support it. The results of this understanding are, first of all, a detailed list of requirements of this task; second, a set of capabilities that meet these requirements; and third, an implemented client-server solution that addresses many of these requirements and identifies others for future work. Our solution incorporates several novel perspectives on data analysis and combines a history mechanism with a graphical, re-usable representation of the analysis and exploration process. Our approach emphasizes using the database itself to represent as many of these functions as possible.

1 Introduction

Analyzing vast amounts of data to extract "knowledge" is now a business imperative. Typically, attention is focused on learning and data mining algorithms (see, e.g., [AIS93, PS91]) that provide the core capability of generalizing from large numbers of specific facts to useful high-level rules. However, real-world data analysis tasks can be extremely complex, and focusing attention on autonomous approaches (e.g., rule induction) tends to underemphasize the key role played by the human data analyst in all current day data analysis [BA94].

Current day data analysts (author LW) use a variety of statistical and other tools for analyzing business data. However, the task of the data analyst very often is complicated by the fact that she does not know what subset or view of the data is *relevant* to the task at hand. Determining this necessitates a prior step of *data exploration*, but the task of data exploration itself makes use of certain analysis techniques; data analysis and data exploration thus go hand in hand.

Let us examine a general example from the business domain. The AT&T Corporation markets a variety of telecommunications products and services. These activities include promotions, on-going advertisement, new service offerings, new equipment offerings, bundled offerings, etc. Of course, AT&T's competitors are engaged in the same kinds of activities. AT&T is vitally interested in understanding the general market reaction to these efforts; doing so is surprisingly difficult. While AT&T has many large databases containing billing and customer premise equipment information, it is still difficult to find the right data and interpret it in the right context to glean the appropriate business insight. It is the task of the business data analyst (BDA) to use this data to answer various business questions.

The form of the data has a great impact on the ease of analysis and the appropriateness of various tools. While relational databases are one obvious approach, it is important to understand the form and quantity of data involved. A data file, which combines data from many sources, might have 15 million records and take up 1/2 a gigabyte of storage. In the organization we are working with, there are hundreds of such data files. For this reason, these data files are mostly kept on 8mm tape until they are needed, at which point they are read into UNIX flat files for processing.

The tools used to explore and analyze this data are a small set of UNIX utilities, like "grep", "sort", "unique", programs written in programming languages like C or AWK, statistical packages like S, and tree induction routines. These tools are used under the X window system. The reason these tools are used, besides legacy reasons, again has to do with the quantity of data - it is typically much faster to process a flat file

SIGMOD '96 6/96 Montreal, Canada
© 1996 ACM 0-89791-794-4/96/0006...$3.50

Run custom AWK script to divide base file by credit history into 4 segment files.

Pick smallest segment file for initial exploration.

Visually scan data to get a feeling for number of nulls in the revenue field.

If it seems high, run a small script to actually count them. If still high, note down.

Decide to examine revenue by region - run a small script to translate data file into files that S can read.

Drop into S to do the graphing, potentially customize the graph using the S language.

Note that one region has an "interesting" value (perhaps much higher than expected).

Extract the records with that region (by running a small script) into a new file.

Examine some other attribute of that file, using S, and create a graph "really" worth saving.

Try to go back and "do the same thing" to all of the categories created, or some combinations of the categories (which, in this example, is credit history by region by revenue, with several other attributes).

Figure 1: Stream of Operations in a Sample Session

format if the calculations are well-known and essentially involve one pass through the data. The price paid is a lack of "meta-data" support: a flat file has no inherent structure, no information on the semantics or types of the data in the fields, and no integrity checking.

We present, in figure 1, a stream of operations from a small part of a typical 1 to 2 hour session of exploration and analysis by a BDA, for the reader to get some feel for the task of the BDA. The fragmentation of the tools used by the BDA necessitates (1) manual bookkeeping, and (2) data translation. The main problems faced by the BDA here are a lack of environmental support for keeping track of a *sequence* of operations, no support for *reuse* of work, and a lack of enforced *semantics* between operations and data (and thus between sequences of operations). Also unsupported are: translation of data between file formats, the capturing of relationships between files, recovery from errors earlier in a session, and window management.

To mitigate these problems, IDEA uses a database to store not only the data but also as much information about the *data exploration* and *analysis process* as possible. The use of a database provides support for meta-data, allowing the semantics between operations and data to be enforced. Maintaining information in the database about the process itself reduces the need for manual bookkeeping, allows easy tracking of the sequence of operations performed, and makes convenient the ability to "re-run" an analysis.

As we briefly mentioned earlier, most approaches to improving the extraction of information from data have focused on automatic data mining or machine learning techniques (also called "knowledge discovery"). There are some exceptions, however. The RECON system [SLK94] is a research tool turned product that integrates several data mining techniques in an interactive framework and is oriented towards the building of rule-based models, for example, of stock price behavior. RECON does not address the kind of *data exploration* we are considering. Holsheimer et al. present a prototype architecture for data mining [HK94], but its emphasis on data mining results in the neglect of the kind of human interaction we are seeking to support. The Knowledge Discovery Workbench [PSM92] has a similar emphasis on the automatic detection of dependencies in data, and does not support user-created segmentations. There are other similar approaches in the literature.

Most closely related to IDEA is the IMACS system [BST+93], an intellectual precursor to this work. IMACS had many of the same ideas as IDEA, with a special emphasis on using a description logic as an "object-oriented front end" to relational data (this idea was also exploited in RECON). The data analysis problems we examined, however, did not need this capability. The IMACS approach was to load the relational data into a description logic-based knowledge base, and subsequently use the description logic to "explore" the data. While this architecture is adequate for exploring small amounts of data, it is unrealistic when the data exceeds a few thousand records; the overhead of using the description logic becomes substantial. To achieve scalability, IDEA uses a client-server architecture with a relational database as the server.

This paper first describes some user requirements extracted from experience with this kind of data exploration and analysis (primarily by LW). We then present some notation for formally describing the abstract database operations needed to address these requirements. The IDEA (Interactive Data Exploration and Analysis) framework is described, including its architecture, interface, and operation, as well as the current implementation and future directions. We then describe what we believe are the contributions of this work, and conclude with an evaluation of the approach.

2 User Requirements

From the results of our collaboration, we distilled out a set of five general user requirements for a support environment. We discuss these in turn now, with the following assumption. While we will use a relational database, we assume the data is available

in a single database table with a known schema (this may require that we *join* associated information from various structured text files). This single table consists of a set of *records* or *tuples*, each record consisting of a number of *fields* or *attributes*. These attributes have types, typically either numeric or string.

Querying: The process of querying is that of specifying *conditions* on one or more of the data attributes to extract "interesting" subsets of data. The result of querying is a subset of the table; that is, a subset of the original set of records.

Segmentation: To segment data is to divide the data into non-overlapping subsets based on the values of one or more attributes. Note that there are at least two kinds of segmentations: those based on attributes with a relatively small fixed set of possible values (for example, a *State* attribute restricted to state codes, i.e. {AL, ...}). We call these *natural segmentations*, because there is a natural way to divide the data up. On the other hand, quantitative attributes (like average revenue, or a person's age) require the user to specify a set of *segment boundaries* and an optional set of *segment names*. For example, a user may wish to segment the data on the *Age* attribute by specifying the following segments: for Age below 1, *baby*; for Age below 5 and above 1, *toddler*; for Age below 10 and above 5, *child*; for age below 13 and above 10, *preteen*; and so on.

Even for a natural segmentation, one can group the natural segments into larger groups and treat these groups as segments. For example, one might group the States into: *Eastern* = {MA, ME, NH, VT, RI, CN, ...}, *Western* = {CA, WA, OR, ...}, etc. These groupings must have no duplicates and use all of the original natural segments for them to be a true segmentation themselves.

Summary Information: Querying and segmentation divide and group the data; the user must be able to compute and present various kinds of summary information (e.g., COUNT, AVERAGE) over various data attributes. These are the actual computations that will make up part of an analysis. These computations must be presented to the user in various graphical forms, e.g., bar charts.

Being able to easily extract "interesting" subsets of data, being able to naturally divide the data into non-overlapping subsets, and computing and presenting summary information are the operations performed repeatedly by the BDA.

External Tools: While querying, segmentation, and computing summary information are the most commonly performed operations, the BDA often requires access to the capabilities of specialized systems — for example, statistical packages, like S, and other common analytical systems, like Excel — to further analyze and display the data.

As exploration and analysis proceed together, and a set of interesting results is derived, these results must eventually be compiled into a report, including graphics, text, and tables. This requires the ability to import the results of analysis into a separate report writing tool.

History Mechanism: One of the critical problems illustrated in the scenario in section 1 is the difficulty of keeping track of the operations performed. A comprehensive history mechanism would maintain a record of all tasks performed by the analyst, infer *semantic* relationships between the various tasks, and make convenient the ability to reuse work.

Existing tools used by the BDA are especially inadequate for the tasks of flexible data segmentation, maintaining semantic relationships between the tasks performed, and enabling reuse of work. The support environment provided by the IDEA system addresses these problems directly.

3 A Database View of the User Requirements

This section examines the user requirements and develops an abstract database notation for examining each more closely. We also describe how these requirements are met in the IDEA framework.

3.1 Querying

Consider a database table R whose schema has attributes $A_1, \ldots, A_n$. Let $\mathcal{D}_i$ be the domain of attribute $A_i, 1 \leq i \leq n$, i.e., the value of attribute A_i in each record in table R is drawn from $\mathcal{D}_i$. For example, the *BTN* (billing telephone number) attribute in the BDA's data is drawn from the domain of ten digit positive integers (actually, with the additional ill-formed and also changing constraint of being a "legitimate telephone number"), and the *total_minutes* attribute is drawn from the non-negative integers.

IDEA allows a user to *query* R by specifying independent conditions for each of the attributes of R. A *condition* for attribute A_i of R, denoted by C_i, can be one of the following.

Finite Collection : C_i can be a *finite collection* of values from $\mathcal{D}_i$; a single value is a special case. This is specified by explicitly enumerating the set of values from $\mathcal{D}_i$.

Range : C_i can be a *range* of values from $\mathcal{D}_i$; specifying this requires that $\mathcal{D}_i$ be a totally ordered domain. A

range is specified by its two end points, each of which is in $\mathcal{D}_i$.

Full Domain : C_i can be the full domain $\mathcal{D}_i$; this is the *default.*

The *result* of an IDEA query with conditions $C_1, \ldots, C_n$ on attributes $A_1, \ldots, A_n$ is a table R', with the same schema as R. A record r of R is in R' if and only if the value of attribute $A_i, 1 \leq i \leq n$, satisfies condition C_i.

3.2 Segmentation

Consider a database table R whose schema has attributes $A_1, \ldots, A_n$. First, we consider segmentation of table R on a single attribute A_i. Let $\mathcal{D}_i^1, \ldots, \mathcal{D}_i^{m_i}$ be a partition of the values in the domain $\mathcal{D}_i$ of attribute A_i.

Definition 3.1 (Segmentation) A *segmentation* of table R on attribute A_i using the partition $\mathcal{D}_i^1, \ldots, \mathcal{D}_i^{m_i}$ of domain $\mathcal{D}_i$ is a collection of m_i, possibly empty, tables $R^1, \ldots, R^{m_i}$, such that:

- The schema of each table R^j includes all the attributes of R, and one additional *segment description* attribute, D_s.
- For each record $\bar{t}$ in R, there exists exactly one $R^j, 1 \leq j \leq m_i$ such that $d^j \cdot \bar{t}$ is a record in R^j, where d^j is the segment description of table R^j, and "$\cdot$" is the record concatenation operator.
- The value of attribute A_i in each record in the table R^j is drawn from $\mathcal{D}_i^j, 1 \leq j \leq m_i$.

The tables $R^1, \ldots, R^{m_i}$ are referred to as *segments* of table R. □

IDEA allows a user to *segment* a table R using attribute A_i by specifying how the domain $\mathcal{D}_i$ should be partitioned. This can be specified in one of the following ways:

Simple Partition : A *simple partition* of domain $\mathcal{D}_i$ is a partition where each value in $\mathcal{D}_i$ is in a separate partition by itself; this is the default. When the table R is finite, the number of non-empty segments of a table is finite, even when the domain $\mathcal{D}_i$ is infinite.

Such a simple partition can be used, for example, to segment customer data on the basis of the *State* attribute: one partition for each state (e.g., MA, ME, AL).

Finite Collection Partition : A *finite collection partition* of domain $\mathcal{D}_i$ explicitly specifies the finite set of values in each partition of the domain; this can only be specified for finite domains.

Such a finite collection partition can be used, for example, to segment customer data on the *State* attribute by grouping states into regions, such as *NorthEast* = {MA, ME, CN, RI, VE, NH}, etc. This assumes that there is no pre-computed attribute called *Region*.

Range Partition : For a totally ordered domain $\mathcal{D}_i$, a *range partition* chooses $m_i - 1$ distinct elements $e^1 < \ldots < e^{m_i - 1}$ from $\mathcal{D}_i$ and partitions $\mathcal{D}_i$ such that all elements in the partition $\mathcal{D}_i^j, 1 \leq j \leq m_i - 1$ are $<= e^j$, and e^j is $<$ all elements in $\mathcal{D}_i^{j+1}, 1 \leq j \leq m_i - 1$.

Such a range partition of the domain can be used, for example, to segment customer data based on the *Average_revenue* attribute (say, by using the partition elements 10, 25, and 75). Typically, the BDA would associate a name with each segment, such as *low-revenue-customers*, *medium-revenue-customers*, etc.

Note that the user bounds the number of partitions when using finite collection partitions and range partitions, whereas the size of the domain and the size of the original table bounds the number of partitions when using simple partitions.

Segmentation of a table R using multiple attributes is a straightforward extension of segmentation using a single attribute. The number of segments created is the product of the number of partitions of each of the attribute domains. For example, the BDA might segment using both the above finite collection partition *Region* and the the above range partition on the *Average_revenue* partition. If there were 6 regions, this would result in 24 segments. Note that the BDA might again wish to name these segments, i.e., *NE-low-revenue-customer*, *NE-medium-revenue-customer*, etc.

3.3 Summary Information

Summary information, using aggregate functions such as COUNT and AVERAGE on an attribute of a table can be extremely useful in determining whether a given segmentation of a table R is suitable for subsequent analysis. IDEA allows the user to perform the following operations.

Summary Computation : Summary information can be computed using any of the SQL aggregate functions, on a given table, or a given segmentation of a table. For the MIN, MAX, SUM and AVERAGE aggregate functions, the user has to specify the attribute of the table (or of the segmentation of the table) on which the aggregate function needs to be computed. For the COUNT aggregate function, no additional attribute need be specified.

The result of a summary computation on a table is a unary table with a single record containing the aggregate value. On a segmentation of a table, the result is a binary table with m records, one record for each table in the segmentation; the value of the first attribute in a record is the segment description of the corresponding segment of the table; the value of the second attribute in each record is the corresponding aggregate value.

Summary Presentation : The computed summary information can be presented in any of a variety of ways, e.g., histograms, bar charts, and pie charts.

3.4 External Tools

The data exploration and analysis capabilities described so far are not enough. There are a variety of other functions performed by the business data analyst using other kinds of tools. For example, in the brief scenario in figure 1, the statistical package S was used to graph some data extracted earlier; S is also used to compute other kinds of statistics. Other tools of this kind include tree induction systems, business graphics packages, modeling tools, and other common business software like Word and Excel.

Typically, the final output of an exploration and analysis session is a report, documenting in words and graphics the important findings, open questions, interesting relationships, etc. found in the session, using a report writing tool.

It seems silly to try to duplicate some or all of this kind of functionality; there are several other approaches. One of the easiest is to provide a facility to dump data to an external file, perhaps in several formats. Then the user can independently run another tool which can read the file and manipulate the data. A more sophisticated approach would involve making the process more seamless by using, for example, the OLE protocol for embedding applications.

The approach advocated by IDEA is more ambitious. Not only do we want to be able to *export* data into other tools but, if possible, we would like to be able to *import* the results of that processing into IDEA. Furthermore, we would like to capture *what* the processing steps were. Depending on how this was done, this would allow the system to re-run this processing. In the best of all worlds, the processing done by external tools could be captured in a meaningful representation and further manipulated by the system.

3.5 History Mechanism

Given the large number of tasks performed by the BDA during the course of a data exploration and analysis session, it is important to keep track of the various tasks performed and the connections between these tasks. The IDEA framework proposes to keep track of two kinds of connections between the various tasks.

Derivation History : The *derivation history* keeps track of all the actions performed by the data analyst, including querying and segmenting tables, computing and presenting summary information, and interaction with external tools.

For example, if the BDA first segmented table R using attribute *State*, and subsequently segmented table R using attribute *Average_revenue*, there would be a derivation-connection between R and the segmentation of R using the *State* attribute, and a similar derivation-connection between table R and the segmentation of R using the *Average_revenue* attribute. Similarly, computing summary information on the *State* segmentation would be derivation-connected to the *State* segmentation.

Note that maintaining a derivation history does not keep track of the temporal sequence of tasks performed, only the logical connections between the tasks.

Subsumption Connections : In performing a large number of tasks, it is inevitable that the various tables and segmentations of tables derived by the data analyst are interrelated. Maintaining a derivation history does not suffice to capture all the interesting connections between them.

For example, the BDA may have initially computed a segmentation S_{St} of table R using a simple partition on the domain of attribute *State*. At some later point in the analysis, she may resegment R using two attributes: a range partition on the domain of attribute *Average_revenue* and a simple partition on the domain of attribute *State*. Although the second segmentation $S_{Ar,St}$ was not derived from the first segmentation, there is a logical connection between the two: the second segmentation results in a *finer* partitioning of the original table than the first segmentation. Knowing about such relationships lets the computation be more efficient, as well as eases the task of the data analyst in preventing unnecessary repeated work.

IDEA identifies a variety of such relationships between the partial results of data exploration and analysis:

Query-query : If table $R2$ is a subset of table $R1$, the relationship from $R1$ to $R2$ is said to be a *query-query subsumption* relationship.

Segmentation-segmentation : If a segmentation $S2$ is a finer partition of the records in table R than segmentation $S1$, the relationship from $S1$

to $S2$ is said to be a *segmentation-segmentation subsumption* relationship.

The previous example of the relationship between the segmentation S_{St} and the segmentation $S_{Ar,St}$ illustrates a segmentation-segmentation subsumption relationship.

Segmentation-query : If table $R2$ is a subset of one of the segments of segmentation $S1$ (after projecting out the segment description attribute), the relationship from $S1$ to $R2$ is said to be a *segmentation-query subsumption* relationship.

For example, the relationship between the segmentation S_{St} and the query that requests for all tuples of R with state code = NJ and $Average\_revenue \geq 50$ is a segmentation-query subsumption relationship.

Query-segmentation : If segmentation $S2$ is a segmentation of a subset of table $R1$, the relationship from $R1$ to $S2$ is said to be a *query-segmentation subsumption* relationship.

Since a segmentation is always derived from a table, and summary information is always computed either on a table or on a segmentation, additional secondary subsumption relationships can be derived using the basic subsumption relationships described above.

IDEA proposes to maintain the *derivation history* itself in database tables to enable efficient management of the history and to allow the BDA to flexibly query the derivation history itself. The derivation history of the BDA's tasks is maintained in three tables:

- A binary table $Deriv\_nodes(Id, Type)$ that maintains node information about the type of each task performed by the BDA.
- A ternary table $Deriv\_edges(Parent\_Id, Child\_Id, Relative\_code)$ that maintains the derivation relationships, where $Relative\_code$ is the code to perform the task corresponding to $Child\_Id$, given the table corresponding to $Parent\_Id$; when $Child\_Id$ is a task that produces a table (e.g., querying, segmentation) this is SQL code; when $Child\_Id$ is a task that produces a presentation, this is the presentation code.
- A binary table $Absolute\_code(Id, Code)$ for tasks that produce tables, that maintains SQL code for generating the task corresponding to Id from the root table of the data exploration and analysis session. This is equivalent to the view definition obtained by merging the various view definitions for table producing tasks along the path from the root table in the derivation history.

When the BDA performs a new IDEA task, one record is added to each of the three tables. The records added to $Deriv\_nodes$ and $Deriv\_edges$ are straightforwardly obtained as described in previous sections. The record in the table $Absolute\_code$ is obtained by *merging* the absolute SQL code of the parent of the current task into the view definition corresponding to the relative code of the current task.

The *subsumption connections* between any two of the BDA's tasks are computed by the IDEA system and maintained in database tables by examining the SQL codes for the two tasks and their parent tasks, as follows.

Query-query : For tables $R1$ and $R2$, the pair $(R1, R2)$ is in the query-query subsumption relationship table $QQSR(Subsumer, Subsumed)$ if:

1. both $R1$ and $R2$ are tables generated by querying, and
2. the conditions in the WHERE clause of $R2$ are at least as strong as the conditions in the WHERE clause of $R1$.

Since all ancestors of a querying task are required to be querying tasks as well, both $R1$ and $R2$ will have only the root table in their FROM clauses; hence, no conditions on the FROM clauses of $R1$ and $R2$ are required.

Segmentation-segmentation : For table segmentations $R1'$ and $R2'$, the pair $(R1', R2')$ is in the segmentation-segmentation subsumption relationship table $SSSR(Subsumer, Subsumed)$ if:

1. the parent tables $R1$ of $R1'$, and $R2$ of $R2'$ are such that either $R1 = R2$, or $(R1, R2)$ and $(R2, R1)$ are both in the table $QQSR$, i.e., the parent tables have identical extensions,
2. the attributes along which $R2$ is segmented includes all the attributes along which $R1$ is segmented, and
3. along each attribute A_i that $R1$ is segmented, the partitioning of the domain $\mathcal{D}_i$ in $R1'$ is identical to or is refined by the partitioning of the domain $\mathcal{D}_i$ in $R2'$.

Segmentation-query : For segmentation $R1'$ and table $R2$, the pair $(R1', R2)$ is in the segmentation-query subsumption relationship table $SQSR(Subsumer, Subsumed)$ if:

1. the parent table $R1$ of $R1'$ is such that the pair $(R1, R2)$ is in the table $QQSR$, and
2. along each attribute A_i that $R1$ is segmented, the condition on A_i in the WHERE clause of $R2$ is at least as strong as the condition on A_i in one of the segments of $R1$.

Query-segmentation : For table $R1$ and segmentation $R2'$, $(R1, R2')$ is in the query-segmentation subsumption relationship table $QSSR(Subsumer, Subsumed)$ if the parent table $R2$ of $R2'$ is such that either $R1 = R2$, or $(R1, R2)$ is in the table $QQSR$.

The graphical presentation of the derivation history is based on the database tables that maintain the derivation history.

3.6 Translation to SQL

A query on table R with conditions $C_1, \ldots, C_n$ on attributes $A_1, \ldots, A_n$ is translated into the following SQL code, where full domain conditions are dropped from the WHERE clause:

SELECT *
FROM R
WHERE C_1 AND ... AND C_n

The implementation of a segmentation of table R using attribute A_i depends on how the domain $\mathcal{D}_i$ is partitioned. For a simple partition of the domain, the following SQL code is generated, where all the segments are stored in a single view table. Recall that the additional segment description attribute is required by the definition of a segmentation.

CREATE VIEW R' AS
SELECT $A_i, R.*$
FROM R

For a finite collection partition of the domain $\mathcal{D}_i$ into $\mathcal{D}_i^1, \ldots, \mathcal{D}_i^{m_i}$, we first create an auxiliary binary table $CSD(Id, Val)$, such that for each value $e \in \mathcal{D}_i^j$, there is a record (d^j, e) in CSD, where d^j is the identifier for $\mathcal{D}_i^j$. The following SQL code is then generated to compute the segments of R. Note that the identifier for $\mathcal{D}_i^j$ is used as the segment description for R^j.

CREATE VIEW R' AS
SELECT $CSD.Id, R.*$
FROM R, CSD
WHERE $R.A_i = CSD.Val$

For a range partition of the domain $\mathcal{D}_i$ into $\mathcal{D}_i^1, \ldots, \mathcal{D}_i^{m_i}$, we first create an auxiliary ternary table $RSD(Id, Low, High)$, such that for each $\mathcal{D}_i^j, 1 \leq j \leq m_i$, there is a record (d^j, l^j, r^j) in RSD, where d^j is the identifier for $\mathcal{D}_i^j$, l^j is the left end point of the range for $\mathcal{D}_i^j$ and r^j is the right end point of the range for $\mathcal{D}_i^j$. The following SQL code is then generated to compute the segments of R.

CREATE VIEW R' AS
SELECT $RSD.Id, R.*$
FROM R, RSD
WHERE $R.A_i <= RSD.High$ AND
$RSD.Low < R.A_i$

The implementation of summary computation depends on whether it is computed on a table or on a segmentation of a table. Let **AGG** be the aggregate function that needs to be computed, on attribute A_j of table R. For summary computation on a single table R, the following SQL code is generated.

CREATE VIEW R_a AS
SELECT **AGG**(A_j)
FROM R

For summary computation on a segmentation R' of a table, the following SQL code is generated, where D_s is the segment description attribute of R'.

CREATE VIEW R_a AS
SELECT $R'.D_s$, **AGG**$(R'.A_j)$
FROM R'
GROUPBY $R'.D_s$

4 The IDEA Architecture

The IDEA architecture is based on three key ideas:

1. *All data*, including: (a) the SQL code for querying, segmentation, and summary computation, (b) tables corresponding to the results of the various tasks performed by the BDA, and (c) the derivation and semantic relationships between the BDA's various tasks, *are stored in database tables at the server.*

   This enables a clear separation of tasks between the client, with which the user interacts, and the server, where the data resides. The alternative, say of maintaining SQL code and the history information at the client side, would require considerable duplication of effort.

2. Whenever possible, tasks performed by the BDA *have a lazy evaluation.* For example, querying a table R generates only the relative and absolute SQL code needed to compute the resulting table R'. Similarly, a summary computation generates only the required SQL code.

   Only when the user explicitly requests that a table, a segmentation of a table or summary information be presented, that any actual computation is performed at the server. The advantage of lazy evaluation can be observed when, as is often the case, the user asks only for the presentation of summary information — in this case, intermediate tables and table segmentations need not be materialized.

   The alternative, of eagerly materializing each table and table segmentation, is extremely space inefficient.

3. The SQL code that is evaluated is generated *relative to the root table*, which is the starting point of the

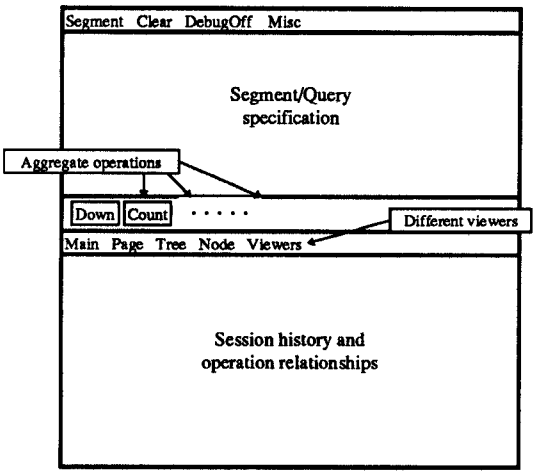

Figure 2: Main Screen of IDEA

BDA's exploration and analysis. The advantage of this technique is that the data exploration and analysis phase is parametrized by the data to be analyzed, which allows for complete *reuse* of the BDA's efforts.

For example, the BDA currently performs data exploration and analysis on a sample of the complete data; when satisfied, she repeats the entire sequence of analysis on the complete data set. Making the SQL code relative to the root table enables the IDEA system to *automatically* perform this second phase of the BDA's analysis.

The IDEA system has been implemented on a PC, using a client-server architecture. The relational tables containing the data that need to be analyzed, as well as the intermediate tables and the history information generated during the BDA's data exploration and analysis session are stored on a PC database server, WATCOM SQL [WAT]. The user interacts with a graphical user interface (GUI), implemented using ToolBook [Asy] which acts as a client of the PC database server. Communication between the client and the server is straightforward, where the client issues SQL commands to the database server, which returns tables to be presented to the user. The actual network connection uses the ODBC protocol on top of TCP/IP. This will allow IDEA to use a variety of databases in the future.

Figure 2 shows a "map" of the main screen of the current IDEA implementation. The top section is where the user specifies either a query or a segmentation (the menu item that says "SEGMENT" toggles between these two operations). The user uses the pull-down menus beside each attribute to either restrict the attribute (in querying) or define a set of segment boundaries (in segmenting). As described in section 3.2, there are several ways of doing this, depending on the kind of attribute. Alternatively, the user can directly type in the restriction or segment boundaries and also add them to their respective menus.

When the user has formulated a query or a segmentation to her satisfaction, the middle section of buttons provides a mechanism for adding the operation to the current session (the "Down" button). Other buttons represent aggregation operations, like COUNT or AVERAGE, that may require the user to select the aggregation attribute. Other operations, in particular, various viewers for the data, are currently available under the "viewers" menu bar.

The bottom section of the IDEA implementation is a combined session process flow diagram (history mechanism) and a place where the relationships between different session operations can be represented. As the user generates a set of operations using the top query/segmentation part of of the interface, IDEA generates a set of node icons representing each operation, and places it in the appropriate place in a session graph. Each node has associated with it a condition or set of conditions that represent the operation it stands for.

Figure 3 shows an early part of a session. The node labeled "BASE" represents the entire data set. This data set is a sample from a much larger data set of telecommunications data. This data contains information about a set of customers who make many international long-distance calls; the attributes of the data include the most common country called, the region of the world that country is in ("Pacific Rim", "Western Europe", "Africa", etc), and various aggregate revenue and usage information ("av_i_rev" means "average international revenue", for example).

The user has specified two segmentations. The bottom segmentation (represented by the hatched node) is a segmentation by "REGION". The user has followed this segmentation with the "COUNT" operation and the "HISTOGRAM" viewer. The top segmentation is by "av_i_rev", or average international revenue. To specify this segmentation, the user had to indicate segment boundaries, in this case, $10, 25, 75$. After these sequences of operations were indicated, the user indicated that she wanted to actually view the results. IDEA starts from the right-hand "leaves" of the tree, and goes backwards towards the root (the entire data set). It collects the conditions at each node and combines them into SQL, which is then executed. Figure 3 shows the results in the form of two histograms: one, a "region profile" of all the data, and two, a "revenue profile".

Figure 4 illustrates a later part of this same session. The viewed results in figure 3 (the two histograms) led the BDA into two directions: the first was to isolate the "high-end customers", those whose average in-

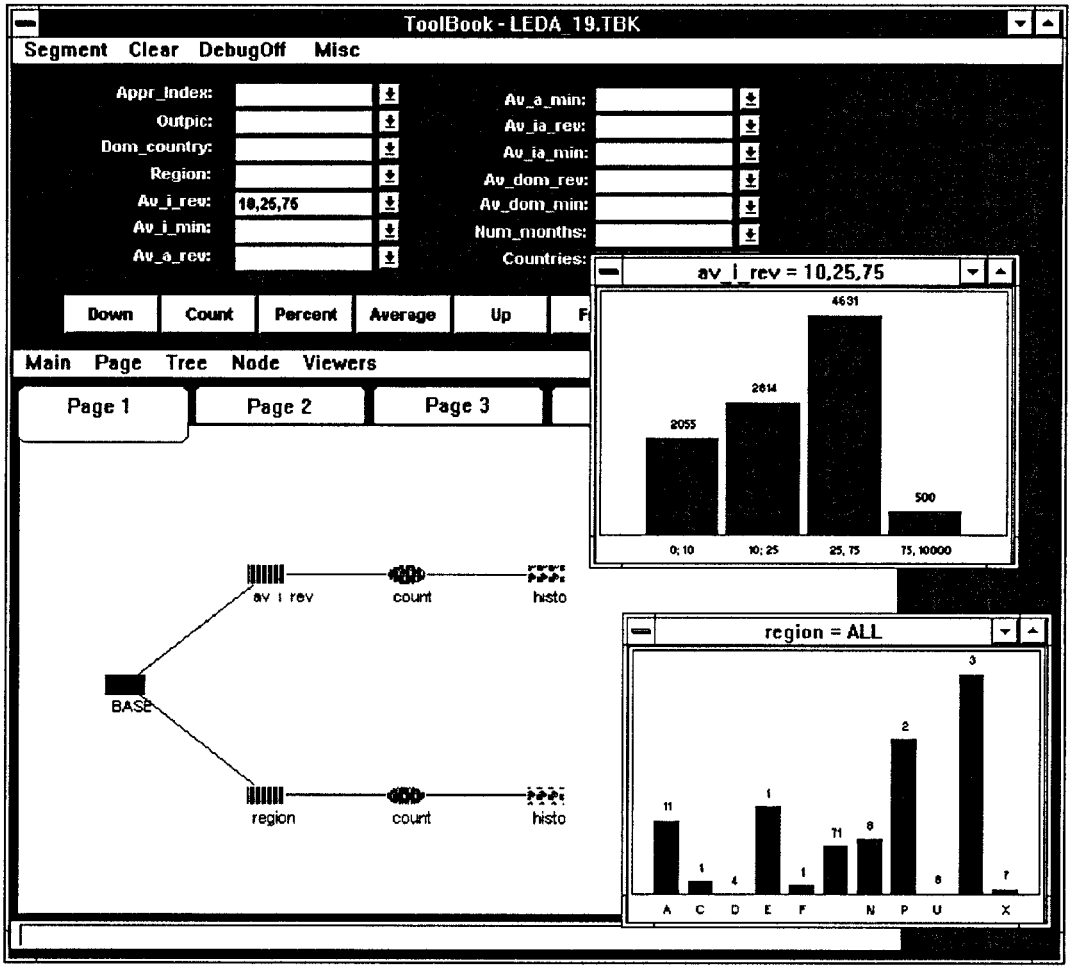

Figure 3: Early Part of IDEA Session

ternational revenue was > 75, and examine their "region profile". An interesting difference is noted: while the population as a whole calls countries in the W region (Western Europe) most often, followed by the P region (Pacific Rim), this relationship is reversed for the "high-end customers". This led the BDA to pull out those customers (the "P-calling customers" and the "W-calling customers"), and examine *their* revenue profile. The BDA decides to do this at a finer grain than before, choosing the segment boundaries $10, 20, 30, 40, 50, 60, 70, 80, 90, 100$. Again, an interesting difference is noted in the histograms.

A report writing tool can now be used to create a report containing the last two interesting histograms, and associating text with the histograms.

5 Discussion

5.1 Summary

The above examples illustrate just a few of the implemented parts of the IDEA framework. A few additional comments are in order. First of all, the creation of a set of data operations is easy and intuitive, and the history mechanism represents the current state of analysis. Second, while a user creates a picture of the analysis steps she desired, no actual database querying takes place until the user requests it (lazy evaluation). Third, there are other operations available for copying and re-using whole "branches" of analysis steps. Fourth, the IDEA framework and current implementation supports the idea of doing an analysis on a sampled subset of the data. Once the analysis is satisfactory, it can be "re-run" on the original dataset.

5.2 What is New

The issues of database exploration and data analysis are not new. There are dozens of tools available under the overlapping categories of "decision support systems", "executive information systems", analysis environments, and OLAP (on-line analytic processing) tools [Byt95]. Other tools include ad-hoc query tools and report writers. New tools appear every week from large database vendors to small start-up companies. As

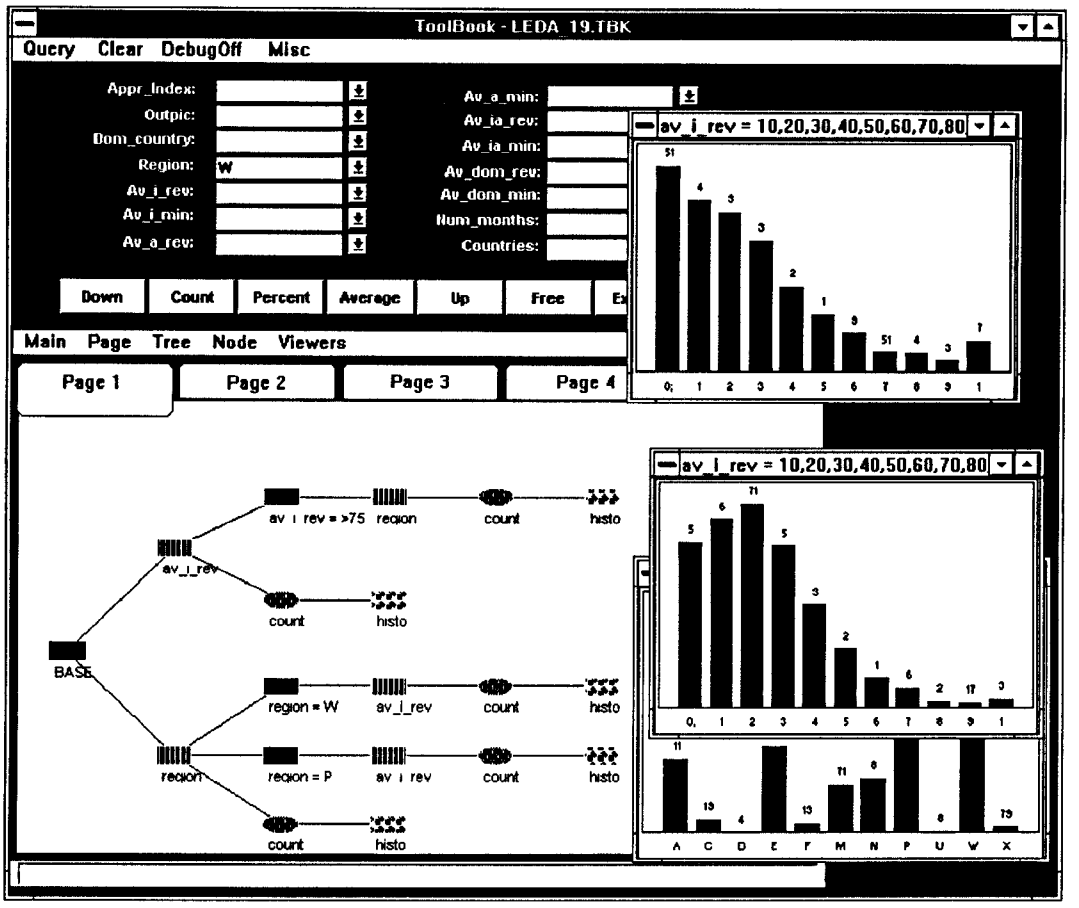

Figure 4: Later Part of IDEA Session

with programming languages, many tools can support many tasks, and it is difficult to understand when a tool supports a task in a novel and comprehensive fashion.

The academic database research community has also addressed the issue of data exploration and analysis in two main areas. First, in understanding difficult technical issues of query optimization, database structure, and advancing the underlying relational data model to handle new types of data, and presenting improved algorithms addressing these issues. Second, "knowledge discovery in databases" has arisen as a hot new area similar to, but not identical with, the more commercial phrase of "data mining". This area is exploding with machine learning and statistical approaches to deriving new knowledge from large corporate and scientific databases.

We have taken a different approach to those of choosing and evaluating a commercial tool (although we are familiar with many of them) or taking a more academic approach. Instead, we took advantage of a local business data analyst and attempted to understand her task and the tools she had at her disposal. This "bottom-up" approach resulted in a novel and interesting tool as well as insight into the support of a real-life data analysis and exploration task. We list here what we believe are some of the contributions of this work:

- A novel graphical history mechanism for representing an exploration and analysis session.
- The incorporation of higher-level integrity constraints on the analysis itself, ensuring that the semantics of the operations match the data.
- The idea of a *reusable* analysis session that facilitates both exploration on a sample database and the generation of periodic reports from data.
- Segmentation as a fundamental and first-class operation for the analyst and several supporting mechanisms in the environment for creating, re-using, and efficiently computing segmentations.

5.3 Future Work

There are many interesting directions of future research, some of which are described below.

- The IDEA system allows for a limited form of segmentation. It is worth investigating how to incorporate segmentations where the specifications of the segments relate multiple attributes of a table. For example, a table can be segmented into two tables as follows: the records in the first segment satisfy the condition *Average_revenue* < *Average_international_revenue*, and the records in the second segment satisfy *Average_revenue* $\geq$ *Average_international_revenue*. Implementing this in SQL may require more sophisticated techniques than using auxiliary tables of the form described.

- The IDEA system currently supports a limited interface to external tools; in particular, re-running an analysis does not automatically invoke the use of the external tools. Developing a tighter integration with external tools is very important.

- When some table segments are materialized, it may be more efficient to compute subsumed table segments or summary information from the subsumer table segments than using the absolute SQL code that uses the root table. Determining how to do this efficiently is an interesting direction of research.

- Determining what kinds of operations (e.g., querying) of the derivation history tables are useful to the BDA, and how to efficiently support these operations.

6 Conclusion

This work is very much in progress; however, we will make a few remarks about our general approach. We are trying to build a comprehensive framework for addressing: (1) theoretical issues (such as optimization of multiple analysis operations), (2) general practical issues (such as reuse of analysis and easy connection to multiple databases), and (3) a set of problems encountered by author LW in the context of a specific data analysis task. We are trying to validate a number of general principles. First, represent as much of the data and the work context itself in the database. Second, add features only when the specific task requires it. Third, make the framework open to other tools, both in terms of being able to *export* data, but also try to capture and represent the processing and returned data of the other tools. This approach, as reported here, has resulted in an evolving, usable tool, as well as a number of novel ideas, some only partly implemented. We anticipate continued synergy between the theoretical ideas and the tasks derived from the actual work of business data analysis.

Acknowledgements

We would like to thank Loren Terveen for his many suggestions that helped improve the presentation of the paper, and Shaul Dar for discussions on table segmentations.

References

[AIS93] R. Agrawal, T. Imielinski, and A. Swami. Database mining: A performance perspective. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):914–925, December 1993.

[Asy] Asymetrix Corporation, 110-110th Ave N.E., Suite 700, Bellevue, WA 98004. *ToolBook Manual.*

[BA94] R. J. Brachman and T. Anand. The knowledge discovery process. In *Working Notes of the AAAI-94 Workshop on Knowledge Discovery in Databases*, 1994.

[BST+93] R. J. Brachman, P. G. Selfridge, L. G. Terveen, B. Altman, F. Halper, T. Kirk, and A. Lazar. Integrated support for data archeology. *International Journal of Intelligent and Cooperative Information Systems*, 2(2):159–185, June 1993.

[Byt95] A data miner's tools. *Byte Magazine*, 2(10):91, October 1995.

[HK94] M. Holsheimer and M. L. Kersten. Architectural support for data mining. In *Working Notes of the AAAI-94 Workshop on Knowledge Discovery in Databases*, 1994.

[PS91] G. Piatetsky-Shapiro, editor. *Knowledge Discovery in Databases.* AAAI/MIT Press, 1991.

[PSM92] G. Piatetsky-Shapiro and C. J. Matheus. Knowledge discovery workbench for exploring business databases. *International Journal of Intelligent Systems*, 7(7):675–686, 1992.

[SLK94] E. Simoudis, B. Livezey, and R. Kerber. Integration inductive and deductive reasoning for database mining. In *Working Notes of the AAAI-94 Workshop on Knowledge Discovery in Databases*, 1994.

[WAT] WATCOM International Corporation, 415 Phillip Street, Waterloo, Ontario, CA NXL 3X2. *WATCOM SQL 4.0.*