2013 IEEE International Conference on Healthcare Informatics

Automated Trauma Incident Cubes Analysis

Ankit Srivastava*, Lisa Ferrigno†‡, Stephen Kaminski‡, Xifeng Yan*, Jianwen Su*
* Department of Computer Science, UC Santa Barbara, CA, USA
{ankit, xifeng, su}@cs.ucsb.edu
† Translational Medicine Research Laboratory, UC Santa Barbara, CA, USA
‡ Trauma Services, Santa Barbara Cottage Hospital, Santa Barbara, CA, USA
{lferrigno, skaminski}@sbkh.org

***Abstract*—National Trauma Data Bank (NTDB) is the largest repository of statistically robust trauma data in the United States, assembled from trauma centers across the country. NTDB data has been commonly used in risk adjusted studies in the medical communities to describe patterns of injury, interventions and patient outcomes in order to better tailor trauma treatment. The studies have led to significant improvements in the standard of care delivered to trauma patients. A considerable amount of research efforts have been spent on development and maintenance of NTDB to continuously improve the quality and effectiveness of trauma patient records. Prior studies relied mostly on ad hoc and manual extraction processes of data from NTDB repository. Given the rapid growth of the NTDB datasets in an ever changing clinical environment, there is an urgent need to develop standard methodologies and software tools to support data analysis involving NTDB datasets. The goal of this research is to empower clinicians to be able to utilize collected content for such analysis by using standardized data collection and aggregation practices.**

Specifically, in this paper we generalize existing OLAP techniques to model NTDB data for capturing statistical and aggregated information. We present a system to automate the process of creating "incident cubes" for all permutations of attributes in NTDB data model, and a querying framework for extracting information from cubes. We also define a ranking function to discover new and surprising patterns from cubes, based on the information gain from each attribute. A case study is used to illustrate that we can take advantage of the system to support trauma data analysis effectively and efficiently.

I. Introduction

Trauma remains one of the leading causes of mortality in the United States, affecting all range of age groups. National Trauma Data Bank (NTDB)[1] is the largest trauma registry of trauma patients in the whole of US, Canada and Puerto Rico. Over the past several years, NTDB has been extensively used in trauma research, in the study of trauma epidemiology, quality of care, and patient safety. While NTDB provides an attractive option for obtaining a population reflective registry of patient records, it suffers from issues of selection bias, missing and erroneous data values. In order to improve the condition of trauma research and provide better health care facilities to trauma patients, National Trauma Data Standards (NTDS) was established in 2009 [3].

Haider et al [15] highlights the usefulness of NTDB, especially as practice standards with respect to data collection have evolved. NTDB has improved the standard of care of trauma patients. The data is accessible to clinician researchers, such that the number of publications based on NTDB went from 2 articles in 2003 to 29 in 2009 and 33 in 2010. However, the lack of standards with respect to data manipulation and analysis may cause spurious or misleading results. Given that these results may alter the care provided to patients, consistency and correctness in the methodology is important [13], [15].

One example of methodological inconsistency is the way missing data is handled by different analysis. The Journal of the American College of Surgeons, usually dedicated to the research and publication of clinically related surgical (including trauma) material, published a methodology for imputation to handle missing data in the NTDB [22]. Despite that, [15] pointed out that $>70\%$ of authors publishing on NTDB in 2009 and 2010 simply excluded missing data, frequently in a context where some other mechanism would have been more appropriate. Haider et al [13] reiterated some of the same concerns about data handling and analysis and implored for standardization of methodological practices.

The NTDB data model is inherently relational in nature. The data model consists of a set of relations containing patient incident records including relations for patient demographics, physiological parameters, type of injury, facilities information and event code lookup details. Due to its highly dimensional nature, Online Analytical Processing (OLAP) based methodologies to deal with multidimensional datasets seem to be a good fit for carrying out analytical studies. In this paper, we present an aggregation methodology for multidimensional data to carry out statistical and analytical studies and in correspondence with NTDB. We present a system to outline our approach by streamlining the processes of data extraction, aggregation and querying.

The shortcomings of modeling multidimensional data in relational databases to effectively consolidate, view and analyze data was first pointed out by Codd et al [7]. Their study characterized the analysis of data on multiple dimensions as one of the characteristics of an OLAP system. References [10], [17] also presented that relational databases have been designed for Online Transaction Processing (OLTP) and are inadequate for handling OLAP operations. Since then significant research efforts have been done in the development of dedicated multidimensional analysis systems. Reference [12] proposed an extension to SQL with a Data Cube operator that generalizes *group by* operator to produce multidimensional aggregations. There are two main approaches in storing and analyzing multidimensional data. One approach maintains the data as a multidimensional cube on a non-relational structure designed for storing multidimensional data. The other approach

[1] http://www.ntdbdatacenter.com

978-0-7695-5089-3/13 $26.00 © 2013 IEEE
DOI 10.1109/ICHI.2013.32

stores multidimensional data in a relational database. Multidimensional queries are translated into relational queries. This approach led to development of Relational OLAP systems (ROLAP) [6]. Agrawal et al [1] provided a hypercube based model to provide semantic foundations of multidimensional databases. They augmented the model with algebraic operators to translate cube operations to either a SQL-based backend or a special purpose multidimensional storage.

Gray et al [12] presented data cubes as a specialized OLAP methodology to effectively perform aggregation and summarization of data values. A data cube aggregates data along an N-dimensional space. Each cell in the cube is pre-calculated to store values across all N-dimensional values. Beyer and Ramakrishnan [5] described an efficient algorithm to compute "Iceberg" cubes, where not all cells are pre-computed during the creation of the cube. Instead, only those cells that are greater than a threshold aggregation amount are computed and stored in the data cube.

Many different software tools already exist that facilitate the process of creation of data cubes and multidimensional analysis. Microsoft Excel and SQL Server can be extended to support creation of data cubes and provide support for multi-dimensional (MDX) queries [8]. Oracle's Analytic Workspace Manager [24] is another commercial tool to design data model and create dimensions and cubes in an Oracle database. Pentaho's Mondrian Project [26] is an open source project that provides an OLAP server to enable the users with real time analysis of large quantities of data. Data Brewery provides Cubes [33], a lightweight Python framework for OLAP processing and multidimensional analysis.

In this paper, we introduce the NTDB data model and highlight some of its issues, based on which we demonstrate the need of standardized OLAP technologies for streamlining the process of data extraction, aggregation and retrieval. We define a rule based language for identifying missing values and errors in NTDB, and develop a tool that leverages the language to mark all incidents in the system. To support the patient data analysis, we develop a prototype system [30] to automate the process of data extraction and creation of incident cubes by using the metadata associated with the logical definition of cube attributes. We further introduce an entropy based ranking function that accommodates missing values in classifying the cubes. The system is augmented with a querying frontend to allow the user to query aggregated data effectively and efficiently, across all combinations of incident cubes. Finally, we employ the system in conducting a gender based outcome analysis and our experiences show that our system not only improves analysis result through the use of entropy based ranking over earlier studies on NTDB data, it also greatly simplifies the process of patient data analysis by freeing the user from tedious tasks such as formatting and extracting data.

The remainder of this paper is structured as follows. Section II introduces NTDB and its data model, and discusses some shortcomings. In Section III, we briefly highlight the literature for OLAP methodologies and re-visit some existing techniques. Based on these concepts, we define Incident Cubes in Section IV, a prototype of a system to automate the process of creation of data cubes from NTDB model. We also present an entropy based ranking function for data cubes. Section V describes a case study conducted using the prototype of engine developed. We present the conclusions in Section VI.

TABLE I. DESCRIPTION OF RDS DATASET TABLES

Relation name	Information	Description
rds_aispcode	Incident	Abbreviated Injury Scale (AIS) code submitted by the health facility
rds_aisccode	Incident	AIS code submitted globally using ICDMAP-90
rds_ais98pcode	Incident	AIS Code globally mapped to AIS version 1998
rds_comorbid	Incident	Pre-existing comorbidity information
rds_complic	Incident	NTDS Complications
rds_demo	Incident	Patient demographics information
rds_dcode	Incident	Diagnosis codes for each incident
rds_discharge	Incident	Discharge details and outcome information
rds_ecode	Incident	ICD-9 external injury code for each incident
rds_ed	Incident	Emergency Department information
rds_pcode	Incident	Procedure and Monitor details
rds_protdev	Incident	Protective devices for incidents
rds_transport	Incident	Transportation Information
rds_vitals	Incident	Vital Parameters for each incident
rds_facility	Facility	Facilities Information
rds_ecodedes	Lookup	External Injury Codes
rds_pcodedes	Lookup	Procedures and Monitor Codes
rds_dcodedes	Lookup	Diagnosis Codes

II. TRAUMA PATIENT DATASETS

This section describes the National Trauma Data Bank's (NTDB's) data model and discusses the issues of missing data and data errors in NTDB, based on which we shall present an Online Analytical Processing (OLAP) based methodology of creation of data cubes in Sections III and IV.

NTDB is the largest aggregation of trauma registry data, collected and maintained by American College of Surgeons (ACS), from hundreds of individual health care facilities. As of 2011, NTDB contains more than 5 million incidents submitted voluntarily by more than 900 health care facilities across the nation [4], [9], including 219 Level I, 239 Level II, 192 Level III or IV trauma centers [9], [20]. Currently 95% of Level I trauma Centers and 80% of Level II trauma centers have contributed data to NTDB. NTDB provides datasets that can be used by researchers to study in multiple domains of trauma research. These datasets contains descriptive information about trauma patients including patient's demographics, physiological parameters recorded during the course of diagnosis, anatomical injury descriptions, diagnoses and interventions, patient outcomes and trauma center facilities information.

Data is submitted voluntarily to NTDB by the trauma centers, making NTDB data a convenience sample [4]. As a result, the NTDB datasets inherit individual deficiencies of contributing health care facilities and suffer from selection bias. Another major issue with NTDB, pointed out by [18], is that of missing data. The study observes that most of the incidents recorded in NTDB have a missing physiological parameter. Due to all of these limitations, NTDB cannot be considered as a representative sample of all hospitals. Despite its limitations, NTDB has been widely accepted as a major reference for risk adjusted studies for predicting patient outcomes [15].

A. NTDB Research Datasets

NTDB provides several Research Datasets (RDS) for scientific analysis and research studies in multiple domains of trauma. These datasets are prepared, organized and analyzed

by the individual(s) and have been used for publications in medical peer reviewed journals, for production of local hospital annual reports or quality assurance projects.

A typical RDS consists of a set of 18 relational data tables, provided for download as data files in different formats, that can be imported to various databases and statistical softwares [4]. Patient incident records are identified using unique incident identifier, "*inc_key*", whereas facilities are identified by a unique facilities identifier, "*fac_key*".

The tables in RDS consist of three types of information—incident descriptions, facilities description, and lookup information. Detailed description of each of the tables is listed in Table I. Three of the data tables (*rds_ecodedes*, *rds_dcodedes*, *rds_pcodedes*) serve as look-up tables with descriptions for ICD-9 [11] External Injury Codes, ICD-9 Diagnosis Codes, and ICD-9 Procedure Codes, respectively. The data table *rds_facility*, indexed by *fac_key*, provides facilities description for each of the 697 participating hospitals. The facilities information includes hospital characteristics such as bed size, trauma level as well as registry inclusion criteria for the participating hospital. The remaining 14 data tables contain patient incident records that are identified by *inc_key*. The table *rds_demo* contains patient demographic information including age, gender, race, and ethnicity, whereas *rds_dishcarge* contains outcome related information including total length of stay, number of ICU days and outcome. The table *rds_vitals* contains physiological parameters—motor, verbal, and eye components of the Glasgow Coma Score (GCS), Respiratory Rate (RR), Systolic Blood Pressure (SBP). Glasgow Coma Scale is a scoring system used to describe the level of consciousness in a patient following a traumatic brain injury and is calculated as a sum of scores observed for three components—eye opening, verbal response, and motor response in the patient. The physiological parameters are recorded twice for each patient, once by Emergency Medical Services (EMS) and the second time at the Emergency Department.

We have used NTDB RDS dataset from the Admission Year 2011 for our research. The dataset contains a total of $722,836$ incidents. These incidents are characterized by 1293 unique injury types and 6359 different diagnostic procedures. A total of 697 different facilities contributed data to the dataset. These facilities included 119 Level I, 147 Level II, and 92 Level III or IV trauma centers; while 339 facilities were not classified.

B. Missing Data

National Trauma Data Standard (NTDS) [3] was established in 2009 as a national standard for exchange of trauma registry data by ACS. NTDS has since standardized the process of inclusion and collection of information being added to NTDB. NTDS provides exact standards for submission of data to NTDB as part of its collection process [3].

Being the most comprehensive source for trauma research, NTDB also faces issues of missing and incomplete data. It was reported [18] that upto 30% of parameters required for predicting incident outcome are missing in NTDB (version 4.3) [2], parameters that have missing data include physiological and demographics parameters.

Many reasons have been attributed to missing values in NTDB, including absence of data values. Patients arriving at the Emergency Department in a critical situation or suffering from extra cranial trauma are more likely to face issues of missing data values. Motor, verbal, and eye components of GCS, RR, and SBP were considered missing if the data is absent, aberrant or the patient is sedated. Verbal component of GCS and RR were also considered missing when the patient is intubated or the data is aberrant.

In case of missing physiological data, many different methodologies have been adapted for trauma group comparisons. Traditional analytical techniques excluded all observations with missing data, leading to a biased group comparisons since patients with missing values can be systematically very different from patients with non-missing data [23]. Another methodology adopted by researchers is to omit the physiological parameters from the analysis, resulting in residual confounding due to incomplete adjustment for baseline risk [21], [31]. On the other hand, references [21]–[23], [25] explore the feasibility of advanced statistical and evidence based techniques such as Multiple Imputation to scientifically calculate the missing values for physiological parameters. The validity of Multiple Imputation relies on the hypothesis that physiological parameters are missing completely at random. The study by Roudsari et al [27] suggests that missing values are correlated to other parameters in the system. While these methodologies have been continuously debated, it still remains unclear which method provides the most statistically and physiologically correct way to deal with missing data.

C. Data Errors

Another limitation of the NTDB datasets is the high proportion of erroneous data. By erroneous data, we refer to all those incidents that violate the rules defined for collection of data values in NTDS manual for the year 2011 [3].

One example for erroneous data is the set of all those incidents that violate the way age is calculated by NTDS [3]. The rule to calculate a patient's age is defined as follows. Age is derived from the value of Date of Birth (DOB) of the patient. If DOB is not collected or not available, then attributes Age and Age Units are completed to the best approximation at the time of injury. If age is less than 24 hours, then complete the values for Age and Age Unit. Age and Age Units are completed only when DOB is not recorded or not available. And if Age is entered, then Age Unit must also be defined. On the other hand, NTDB Manual for the year 2011 defines a rule that Age Unit should be captured only when the patient's age is less than one year, leading to contradictions in the rules.

In order to test this rule, we consider the RDS dataset 2011 with $722,836$ incidents in total. Our tests show that there are a total of $21,887$ patients with valid date of birth and age attributes when age unit is not recorded. Whereas $34,134$ patients have a valid date of birth and age but their age unit was not applicable. As high as $658,329$ incidents have a valid value for all three attributes date of birth, age and age Unit (after excluding all those patients that have an age less than 24 hours). The preliminary results outline the fact that NTDS standards are not enforced strictly and this fact motivates the need of a tool to identify incident records that violate rules enforced by NTDS.

In Section IV-A, we define a language for specifying rules to mark such errors in the database and present a schema independent tool that uses this language to detect all errors. We also extend this tool to mark all incidents in the database that have missing values. These incidents can either be fixed by correcting the erroneous values or excluding/imputing the missing values, depending upon the analysis being conducted.

III. OLAP and Data Cubes

NTDB presents a typical example of a multidimensional dataset with patient records having 50 different types of attributes. These attributes are used for mining conceptual information based on patient demographics, vitals or type of injury in order to predict the value of the outcome variable. We leverage OLAP based data modeling methodologies for effective storage, retrieval and aggregation of data in NTDB. The objective of this section is to present the current state of OLAP methodologies and then introduce the concepts of data cubes. We also discuss the methodologies for data extraction from a normalized relational schema and creation of cubes. We will present a system to automate the process of creation of data cubes from NTDB dataset in Section IV.

A. OLAP Methodology

Multidimensional analysis are supported by OLAP methodologies [1], [6], [7], [10], [17]. OLAP models are designed to be analytical and to support complex querying scenarios. They typically use a multidimensional data model that is inherently simple to model real world business entities in complex scenarios [17]. One such OLAP technology, data cubes [12] provides the user with an efficient way of investigating a multidimensional dataset across many attributes. Data cubes provide an efficient way to summarize data at various level of details and on various combinations of attributes, making it suitable for multidimensional analysis. Data cubes precompute group-bys corresponding to all combinations of a list of attributes (multiple dimensions) [29].

B. Data Cubes

A data cube or hyper-cube represents the storage of factual data with multidimensional associations. A data cube stores all factual information with similar shape, i.e. similar set of dimensions. Each of the dimensions in the multidimensional analysis is treated as a dimension of some N-space. Each cell in the cube is an aggregation of particular set of attributes values. Querying for data is performed using queries that get translated using aggregation querying operations—rollup, drill down, slicing, and dicing [12].

- *Rollup* refers to the data operation of summarizing cube data along one specific dimension.
- *Drill down/up* allows navigation based on the detailing of the data. Going down increases the detailing in aggregation of data while drilling up leads to summarization.
- *Slicing* refers to the act of filtering a rectangular subset of a cube by choosing a single value for a dimension, resulting in a new cube with one less dimension.
- *Dicing* also creates a sub-cube by filtering data on multiple values for multiple dimensions. The total number of dimensions remains the same in this operation.

C. Star Schema

For the ease of implementation and design, three or more dimensional datasets are implemented using relational OLAP (ROLAP) model as backends. ROLAP model not only conforms to existing relational database engines, it also leverages SQL based querying framework for performing multidimensional analyses. In ROLAP, data is stored in a star schema and data attributes are modeled as either dimensional attributes or metrics. Star schema consists of two types of tables—dimensions and facts. Fact refers to numerical and quantitative transactional information and are huge in size. Dimensional tables are usually smaller and contain descriptive information used for performing detailing operation. A star schema is then used as an input to create a cube to increase the querying performance by pre-aggregating measures based on dimensional data points.

In Section IV, we present a system that leverages the star schema for creation of cubes and discuss in detail the process of construction of star schema from NTDB data model. The system automates the process of data extraction and creation of star schema by taking as input a JSON representation of data mapping information of the NTDB data model.

IV. Incident Cubes

In this section we present a system to automate the process of creation of multiple data cubes, named "Incident Cubes", in NTDB dataset. The system consists of three main components: an error detection tool for preparing the dataset, a module to create a relational star schema and an engine to create data cubes from a star schema. We also outline an entropy based ranking function to determine relevance between two cubes. Fig. 1 presents the model architecture of our engine.

A. Detection of Missing Values and Errors in NTDB dataset

We discussed the issues of missing data and error values in NTDB data in Section II-B. In this sub-section, we present a tool to detect errors and missing values and mark the incidents thus affected. The tool leverages a rule based language to describe different types of errors, as described below. The tool is schema independent and can be plugged into any relational database with its set of rules.

The language uses rule based constructs. Rules are written to specify data conditions and to mark incidents containing data quality errors in the system. Any incident violating the condition in a rule is marked as an error. Rules can span all possible attributes in the database schema and are provided in an input file. The error detection engine parses the input file to instantiate a running instance of the error detection process.

A rule in the system consists of two components: a text description of the rule and a logic condition stating what the rule should do. The *text description* of a rule contains metadata about the rule: a unique name of the rule, a human readable text explaining the intent of the rule, and an optional database

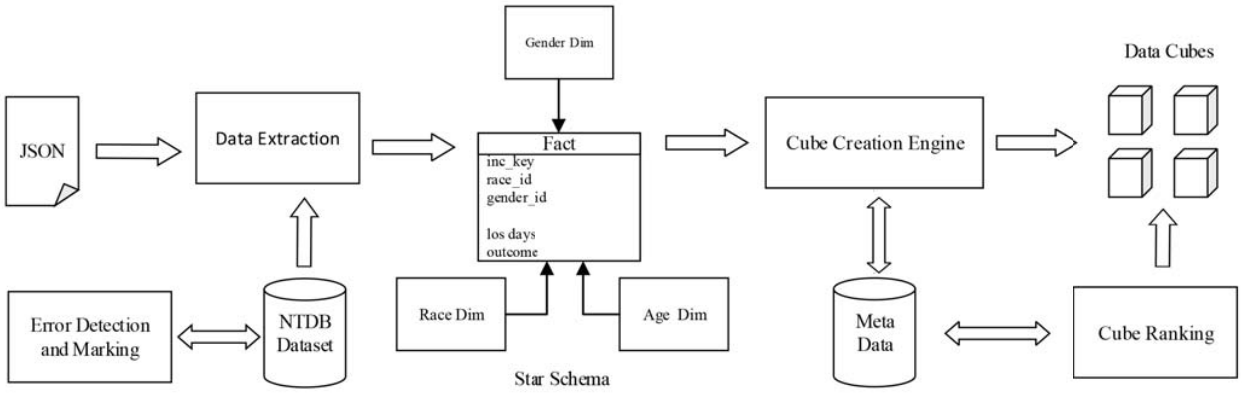

Fig. 1. Architecture of the Cube Creation Engine

relation name on which the rule is to be applied. The relation name is then used for mapping database relation columns defined in rule's *logic condition* to actual relational entities in the database during detection. A rule description has one of the following forms:

`@rule` ⟨*ruleName*⟩ ⟨*text description*⟩
`@rule` ⟨*ruleName*⟩ ⟨*text description*⟩ ⟨*tableName*⟩

where ⟨*ruleName*⟩ is a string repsenting a unique name given to the rule, ⟨*text description*⟩ is the informal description about the rule (in quotes), and ⟨*tableName*⟩ is a name of a table in the database.

Each rule description must be accompanied by a logic condition. The logic condition contains an `if-then` construct to establish the rationale for detecting data quality errors in the system. Rule logic contains an explicit or implicit mapping of relational columns, provided that the rule definition contains the relation on which rule is applied, and the logic to detect data quality errors. Currently, rule logic can be constructed using `if-then` constructs but the language can be easily extended to use more complex structures as well. For example, nested `if-then` constructs are not supported but can be easily added. An example of a rule logic condition is given below.

```
InvColVal if table1.column1 == 0
          then raise invalidColumnError
```

Note that the relation name "`table1.`" is optional and can be omitted if the name is specified in the rule description, and `invalidColumnError` is an error name explained below.

Errors in the system are defined in a similar way as rule descriptions. Each error definition contains a unique error name and a text description. Error definitions do not have any relational table definitions associated with them. The error in the above example is defined as following.

`@error invalidColumnError` ⟨*logical description*⟩

where ⟨*logical description*⟩ is a quoted text explaining the meaning of this error.

As discussed in Section II-C, NTDB datasets have large number of data quality errors that restrict the potential to which this data can be used in studies. Our system uses the language presented above to model rules for detecting errors and missing data in NTDB. In the following, we use NTDB datasets us illustrate the rules in this languages. One example of a logical description for a rule is to identify all non-valid values of a patient's age in NTDB dataset, this is defined as the following.

```
@rule ageRangeRule
      "Rule to describe valid age range"
```

The rule has a name `ageRangeRule` and a description. The logical definition for this rule is given below:

```
ageRangeRule if rds_demo.age < 0 or
                rds_demo.age > 89
             then raise invalidAgeRange
```

Consider another example of a rule logic to detect missing values in the dataset. The rule detects all patients with a valid age but an invalid age unit attribute. This rule logic condition is expressed as:

```
validAgeIncidents
         if rds_demo.age != -2 or
            rds_demo.ageUnit is missing
         then raise invalidAgeIncidents
```

where "`-2`" denotes an invalid value in NDTB and the string "`is missing`" indicates a missing value for the referenced attribute "rds_demo.ageUnit".

Error rules are processed by an error detection engine. The error detection engine performs two main operations: (1) execute the rules to mark all erroneous rows in the schema and (2) maintain the schema metadata containing error information. The execution engine takes as input a set of files containing the rules (along with their logic conditions) and information about different errors that are valid in the database instance. The output after execution is a metadata table consisting of all records from the database that have raised one or more errors. Rows in this new relation are indexed by the primary key column of the row in the original table, the original table

name and error identifier. One row can raise multiple errors. It is left to the user carrying out the analysis to decide how to deal with these errors. They can either reject erroneous data columns or correct them using the methodologies discussed earlier.

The rule-based error detection engine supports detection of missing values automatically. Rules can be designed to mark each incident that has a missing value for relational columns specified in the rule. The result set can be provided as an input to any of the multiple imputation algorithms or researchers can choose to ignore them depending upon the type of analysis.

There are three *metadata tables* that get populated by error detection engine—*meta_rules*, *meta_errors*, and *meta_errorInstances*. The tables *meta_rules* and *meta_errors* contain definition of all the rules and errors specified in the input file. The table *meta_errorInstances* gets populated after execution of the error detection engine is completed. It consists of all rows in the database that raised one or more errors. Any table in the schema can join to this table in order to identify the records that contain one or more errors.

The rule detection engine can also be adapted in solving problems related to other medical scenarios and domains. The engine provides support for rules with proper relational table and columns mapping, so any researcher with knowledge about the relational structure of their dataset can write rules and provide them as input to the engine. The detection engine can also be extended with a web based user interface to provide support for other users.

B. Data Extraction Module

The input to the system described in Fig. 1 is in form of a JSON file. The input file contains logical representation of a star schema in form of mappings between physical relations and logical star schema relations, that are created by the module.

1) Data Extraction: The data extraction module retrieves the data from the underlying physical database based on the mapping information present in the input file. Dimensional tables are populated first in order to populate the surrogate key for each dimensional value before facts are processed. Fact extraction process follows dimensional tables, in which raw transactional data is first populated in a staging table with transformations on dimensional attributes and aggregations on factual measures as defined in the input file. Fact tables are populated from the staging table after resolving for each dimensional values' surrogate key using a lookup on the dimension table. The output of the data extraction module is a relational star schema created from the NTDB dataset.

2) Star Schema modeling in NTDB: The process of creation of a star schema is a step driven by human activity. It requires significant understanding of the NTDB data model and incident population as it requires in modeling of patient attributes as dimensional and factual entities. There are various modeling difficulties that need to be solved in different ways depending as analysis being conducted. One such problem is that of mapping individual physiological parameters—GCS Score, Respiratory Rate (RR), and Systolic Blood Pressure (SBP). Physiological parameters are captured at two instances in NTDB, one during EMS and another during the Emergency Department (ED); unless when data values are missing because of reasons discussed above. A designer can choose to model this scenario in multiple ways. One way would be to create logical dimensions for each set of physiological parameters. Logical dimensions to map each physiological parameter to either its EMS value or the ED value. Another way to solve this modeling problem would be to create a new dimension in the system to identify the type of physiological parameter, either EMS or ED. In our system, we chose to go with the former approach.

3) Metadata Management: Metadata Manager is responsible for creation of all metadata objects in the system. Each entity, whether fact or dimension, has a metadata object associated with it. This is used in components described later to create logical model for input into the cubes creation engine. The Metadata Manager maintains separate tables for maintaining data associated with dimensional and fact attributes.

C. Cube Creation Engine

Many analytical softwares exist that facilitate the process of cube creation from a star schema (see discussions in the Introduction). In order to create cubes through managed code, we leverage Cubes [33], a lightweight Python framework for performing multidimensional analysis and browsing of aggregated data.

1) Python Cubes Framework: Cubes is a part of the data analysis framework provided by Data Brewery [32]. In order to automate the creation of cubes from a star schema generated in the previous step, we provide a wrapper around the Cubes library in Python to programmatically create physical data cubes stored as materialized views. The cubes framework consists of four modules and a command line tool:

- Model: One of the main features of Cubes is its logical model that provides an abstraction over the physical data to provide an end user layer.
- Browser: The framework provides an aggregation browser to perform cube based operations, namely slicing, dicing, roll up, and drill down.
- Backend: Implementation of data aggregation and browsing functionality is provided. Cubes comes with built-in support for ROLAP backends that leverage SQL through SQLAlchemy.
- Server: The Cubes framework provides an easy-to-install web service WSGI server with API that covers most of the Cubes logical model metadata and aggregation browsing functionality.

2) Creation of a Cube: We leverage the framework provided by Cubes to create OLAP Cubes. The process of creation of data cubes is automated using the Cubes browser module that takes as input a logical fact table describing a list of metrics and a set of dimensions. The list of metrics and a set of dimensions are obtained from the data collected by the Metadata Manager in the previous step of data extraction. We provide a wrapper class to the Cubes browser module that automates the retrieval of metadata associated to a star schema—list of fact metrics and set of dimensions, and creates

a logical cube using a subset of dimensions. Data is aggregated against this subset of dimensions and stored in form of materialized views in the physical database.

3) Multiple Cubes: One way to deal with a large number of dimensional attributes is to create one large cube with all attributes. Large cubes cause decreased querying performance since data needs to be aggregated/consolidated with each query operation. The other option is to create multiple smaller cubes. Smaller cubes have more pre-calculated data than larger cubes, making them more efficient. We automate the creation of multiple cubes by creating cubes for all possible permutation of dimensional attributes.

The system automatically repeats the process of creation of a single data cube, as described above, for all permutations of dimensions obtained from the Metadata Manager. After completion of cube creation process, each cube is ranked by the Ranking Module based on the ranking function (discussed below). In order to implement a model system, we assume that all dimensions are independent of each other, although this might not be completely accurate in an application scenario. Ranking of cubes help in providing precedence among cubes during an exploratory data analysis.

D. Ranking of Cubes

Mutual Information has been accepted as a relatively good measure to calculate the relevance of an attribute to an outcome variable in many machine learning algorithms [19], [28]. Decision tree algorithms use mutual information to find the set of attributes that are most relevant in predicting the value of an outcome variable. Mutual Information is expressed information gain in an entropy model and is defined as:

$$I(Y;X) = H(Y) - H(Y|X) \tag{1}$$

where X and Y are two random variables, $H(Y)$ is the entropy [16] of variable Y, $H(Y|X)$ represents the conditional entropy [16] of Y given the state of X, and $I(Y;X)$ represents the information gain about Y obtained from an observation from the state of X. Information gain denotes the reduction of entropy of Y achieved by learning the state of random variable X.

Similarly, for a given a set of random attributes, $X_1, X_2, ..., X_N$, selected during the construction of the cube and each attribute assumed independent of all other attributes, we calculate the total reduction in entropy of the outcome variable Y, given the individual mutual information of each attribute with the outcome variable, as $H(Y|X_1, X_2, ..., X_N)$. We use the total reduction in entropy of the outcome variable to give a score to each cube created, as defined previously. The formula for the reduction in entropy of Y from a given set of independent random attributes is defined as:

$$H(Y|X_1, X_2, ..., X_N) \\ = H(Y) - I(Y;X_1) - I(Y;X_2) - I(Y;X_3) - ... - I(Y;X_N)$$

where $H(Y|X_1, X_2, ..., X_N)$ denotes the conditional entropy of Y given the state of random variables $X_1, X_2, ..., X_N$ and $I(Y;X_i)$ refers to individual information gain in outcome variable from a random attribute X_i and is defined in equation 1. Fig. 2 illustrates the entropy of the system consisting of an outcome variable and a set of independent random attributes.

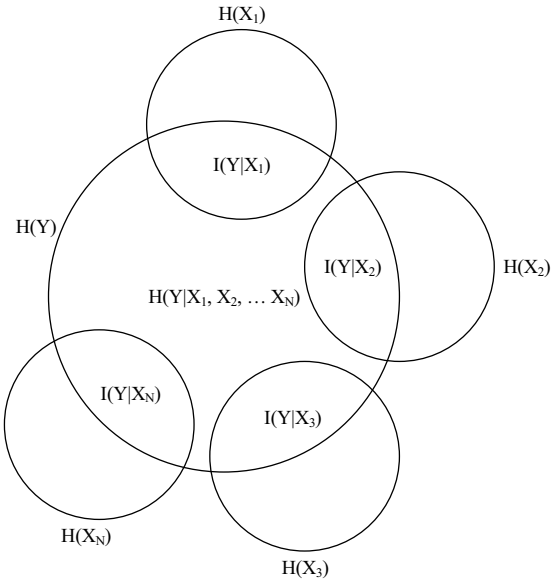

Fig. 2. Conditional Entropy of an Outcome Variable (Y) Given Values for Random Variables $X_1, X_2, ..., X_n$

We denote the score for each cube as the conditional entropy of the outcome variable that is calculated as above. The cube with the lowest score denotes the system with lowest entropy. Lower entropy signifies that the variable has a more non-uniform distribution making it more predictable when the values for the attributes used in the construction of the cube are provided. Cubes are ranked from highest to lowest, based on which cube has a lower score for the entropy function.

E. Querying Framework

User defined multidimensional queries are translated by the querying framework module. The queries are translated by a two step process: resolving the query to the targeted cube(s) and performing cube operations (rollup, drill down, slicing and dicing) to get the correct information from the cube. The first step of translation involves extracting the dimensional attributes from the user query. These attributes are then used by the Metadata Manager to get the corresponding cube(s). In the second step, the user query is then executed against the cube(s) obtained and the data obtained in the result is provided to the user in form of a JSON object(s).

In Section V, we present a case study to deal with the need of controlling variables in a risk adjusted analysis conducted with survival as an outcome. We leverage the querying framework to solve this problem. By default, a list of variables can be defined in the querying framework with baseline filter conditions, in order to define the controlled population properties. And the corresponding attributes can be appended to all user queries to ensure that the user queries always adhere to the controlled group population.

V. Case Study: Gender Based Outcome Analysis

Like the population at large, the trauma population is changing. It used to be a younger population of risk takers.

Improvements in automobile safety, an overall decrease in gun violence in many locales and the increase in the ageing population has led to a population at risk from a different sort of adverse outcome: the increasing need of a place other than home, post discharge, with skilled nursing facilities. The ramifications are huge on many levels; aside from the emotional and familial burdens, the potential costs to society are high as well. It is not unusual to have a number of patients being maintained in the hospital setting awaiting a suitable placement as the patient is unable to return home. The ability to look at attributes or variables which may have a higher likelihood of needing placement would enable hospital, rehabilitation centers and nursing facilities to anticipate and perhaps begin to allocate resources appropriately within communities. In this case study, we validate the need of developing highly skilled facilities for patient care post discharge by providing analytical insights from the NTDB dataset.

In a risk adjusted analysis, there are certain variables that are known to affect the outcome of survival: patient's age, sex, measure of both anatomic and physiologic injury severity, and mechanism or type of injury and these variables that must be controlled for, as also observed by Haider et al [15]. Frequently, not all of these variables are controlled for in studies investigating survival as an outcome in NTDB. The risk of ignoring this adjustment of variables can be expressed by an example.

Let us say that we want to evaluate the risk of death in NTDB population based on whether someone has long or short fingernails. We decide to control for age because we were aware of significant effect that age has on survival by stratification or utilization of a regression model, but arbitrarily do not control for any other variable. We conclude based on our analysis that having long fingernail at the time of trauma has a protective effect. Although this is an unrealistic example and since NTDB does not collect fingernail information, if one was to know that gender consistently makes a difference with respect to survival and that women tend to fare better [14], the bias in our conclusion, for those with long fingernail is clear: we did not control for gender and women tend to have longer fingernail, and hence have better outcomes. Had we controlled for gender, we must not have reached the same conclusion. Usually, of course, the variable being investigated and one of the known significant variables is not so obvious.

In our case study, we perform a gender based outcome analysis to determine conclusive facts about the need of improving patient care in skilled nursing facilities, using cube operations described in Section III-B. NTDB RDS 2011 dataset is used for this analysis. The analysis considers mortality, discharge to a skilled nursing facility and hospice care as *negative* patient outcomes, and discharge event with no or organized home services and short term or long term rehabilitation program are considered *positive* outcome. We control the gender attribute for this analysis, and compare its distribution against different outcomes. Other attributes that are considered for this study are based on the individual information gain to the outcome variable, we select the top two attributes thus obtained, namely "age", and "GCS". We consider the cube with age, gender, GCS and outcome as its attributes.

Rollup by gender dimension informs us that there are a total of $877,101$ males and $510,321$ females who conform with

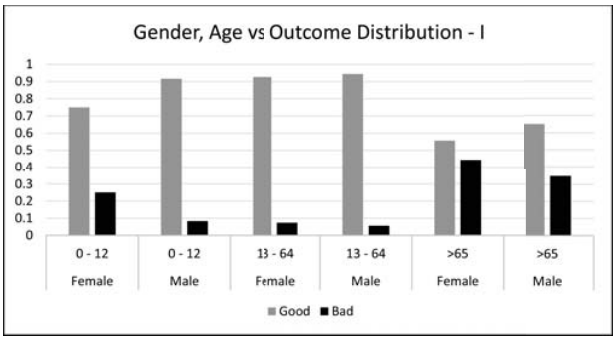

Fig. 3. Gender and Age based Distribution of Positive and Negative Outcomes for the Age Groups 0-13, 13-64, and >64.

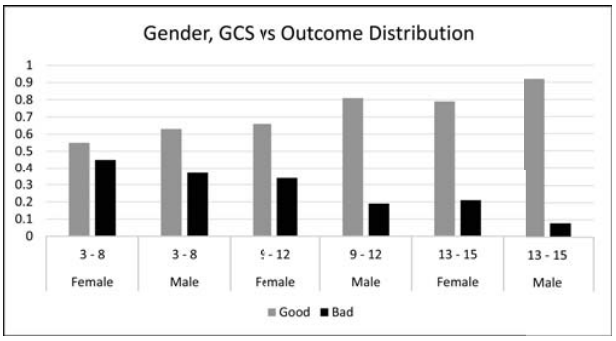

Fig. 4. Gender and GCS based Distribution of Positive and Negative Outcomes for the GCS Groups 3-8, 9-12 and 13-15

the outcome codes provided above. Gender based outcome distribution shows that 23.2% females had negative outcomes as compared to 10.3% males. As against the general notion [14] and the observation we made above, males fared well as compared to females. Figs. 3 and 4 show the individual distribution of age and GCS, after controlling for gender, with the outcome variable. When, age and gender are controlled for, we notice that age group >65 years provides interesting insights on female vs. male population when it comes to negative outcomes. In order to validate that, we performed the drill down operation to a detailed age group and considered the population across 65-75, 75-85 and >85 separately, shown in Fig. 5. Fig. 4 compares the effect of GCS on determining the outcome. In the GCS group 13-15, which reflects the population with least severe trauma conditions, females tend to have more negative outcomes. Fig. 6 correlates these two observations based on age and GCS and provides a conclusive fact that females tend to have a worse outcomes than males when age and GCS are controlled as age is in the groups (65-75, 75-85) and GCS is in the group 13-15. After drilling down (along with the splicing operation) to compare different kinds of bad outcomes for females, while still controlling age and GCS, we observe that there is a high proportion of females that go to skilled nursing facilities after discharge.

In our study, we did not consider the population group obtained by controlling GCS in 3-8 and 9-12 groups. The reason for that is discussed here. Fig. 4 shows that females tend to have more negative outcomes in all three GCS groups 3-8,

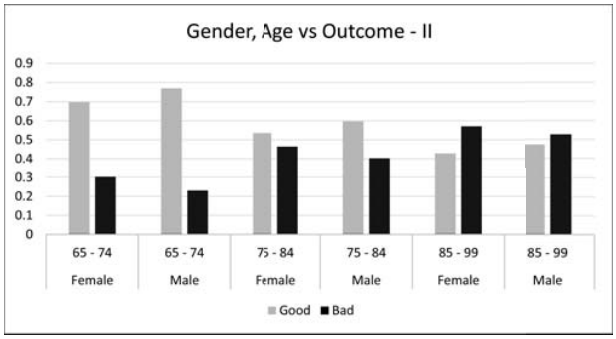

Fig. 5. Gender and Age based Distribution of Positive and Negative Outcomes for Age Groups 65-74, 75-84, and 85-99

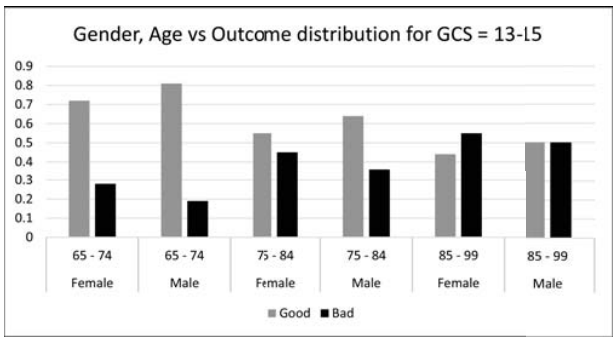

Fig. 6. Gender and Age based Distribution of Positive and Negative Outcomes for Age Groups 65-74, 75-84, and 85-99 and GCS Group 13-15

9-12, and 13-15. The GCS group 3-8 refers to the population of very severely injured patients. Our intuition suggests that the distribution of this group over age group >65 years would be quite uniform, since older patients tend to do worse in case of an severe trauma injury. Although the reason for selection of GCS group 13-15 over 9-12, as the controlling variable value is derived from the entropy model. The ranking algorithm suggests that the incident cube with attributes age, GCS, gender and outcome, with age controlled as >65 years and GCS controlled as 9-12 has a higher entropy as compared to the cube with age controlled as >65 years and GCS controlled as in the 13-15 range.

The above case strengthens our claim for the system we described in Section IV. We conducted the same analysis twice, once during the data exploration and understanding phase of NTDB, and once after we had the system in place. The earlier attempt was done manually and it was prone to human errors. The responsibility for fixing control variable values was based on intuition and required significant insights into the datasets. The incident cubes system simplifies the data retrieval methodology, the querying framework makes it easier to retrieve results based on multidimensional attributes. We also leveraged the ranking function in order to take decisions for selecting control variable values. The system also improved the experience of choosing control variable values, an issue raised by Haider et al [13], [15], based on statistical relevance.

VI. Conclusion

This paper reports the preliminary work on developing a OLAP cubes based methodology for data extraction, aggregation and retrieval. A case for a tool, based on a language with rules based constructs, is highlighted for detection of missing values and errors in NTDB datasets. A methodology for creation of incident cubes from NTDB data model is specified and the process is automated for all possible combinations of data attributes to create a set of incident cubes. We establish a ranking function to classify incident cubes based on the entropy of an outcome variable. The prototype that implements the proposed approach for analyzing NTDB datasets is developed. For the future work, we will carry out extensive investigation of standardization issues faced in trauma research to mitigate the risk associated with the analysis and provide more statistically relevant analytical insights.

Acknowledgment

The authors are grateful to Dr. Avery B. Nathens (ACS Trauma Quality Improvement Program, TQIP), Dr. Scott D. Hammond (Translation Medicine Research Laboratory at UC Santa Barbara), and Rohit Sharma (MD, Santa Barbara Cottage Hospital) for their support and and fruitful discussions, and to Yi Xiao, Bo Yang, and Xiaoxi Yu who helped in the design and implemenation of the error detection language and engine.

References

[1] R. Agrawal, A. Gupta, and S. Sarawagi. Modeling multidimensional databases. In *Proc. of 13th IEEE International Conference on Data Engineering (ICDE)*, pages 232–243, 1997.

[2] American College of Surgeons. National Trauma Databank Pediatric Report 2004. http://www.facs.org/trauma/ntdbpediatric2004.pdf, 2004.

[3] American College of Surgeons. National Trauma Data Standard Data Dictionary. http://www.ntdsdictionary.org/dataElements/datasetDictionary.html, 2011.

[4] American College of Surgeons. NTDB Research Data Set Admission Year 2011 – User Manual. http://www.facs.org/trauma/ntdb/pdf/ntdbmanual2011.pdf, 2011.

[5] K. Beyer and R. Ramakrishnan. Bottom-up computation of sparse and Iceberg CUBE. *ACM SIGMOD Record*, 28(2):359–370, June 1999.

[6] S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. *ACM SIGMOD Record*, 26(1):65–74, 1997.

[7] E. F. Codd, S. B. Codd, and C. T. Salley. Providing OLAP (On-line Analytical Processing) to User-analysts: An IT Mandate. *Codd and Date*, 32:3–5, 1993.

[8] Microsoft Corporation. SQL Server 2012 Analysis Services. http://www.microsoft.com/en-us/sqlserver/solutions-technologies/business-intelligence/analysis.aspx, 2012.

[9] R. J. Fantus and M. L. Nance. Annual Report 2011: Eightfold over eight years. http://www.facs.org/trauma/ntdb/fantus/0112.pdf, 2011.

[10] R. Finkelstein. Database reaches the next dimension. *Database Programming and Design*, 8:26–26, 1995.

[11] Centers for Disease Control and Prevention. Classification of Diseases, Functioning, and Disability : International Classification of Diseases, Ninth Revision (ICD-9). http://www.cdc.gov/nchs/icd/icd9.htm.

[12] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Mining and Knowledge Discovery*, 1(1):29–53, 1997.

[13] A. H. Haider. Improving the quality of science arising from the NTDB: We can do this! *Journal of Trauma and Acute Care Surgery*, 74(2):352–353, 2013.

[14] A. H. Haider, J. G. Crompton, D. C. Chang, D. T. Efron, E. R. Haut, N. Handly, and E. E. Cornwell III. Evidence of hormonal basis for improved survival among females with trauma-associated shock: an analysis of the National Trauma Data Bank. *The Journal of Trauma and Acute Care Surgery*, 69(3):537–540, 2010.

[15] A. H. Haider, T. Saleem, J. J. Leow, C. V. Villegas, M. Kisat, E. B. Schneider, E. R. Haut, K. A. Stevens, E. E. Cornwell III, E. J. MacKenzie, and D. T. Efron. Influence of the National Trauma Data Bank on the Study of Trauma Outcomes: Is It Time to Set Research Best Practices to Further Enhance its Impact? *Journal of the American College of Surgeons*, 214(5):756–768, 2012.

[16] S. Ihara. *Information theory for continuous systems*, volume 2. World Scientific Publishing Company Incorporated, 1993.

[17] R. Kimball and K. Strehlo. What's wrong with SQL. *Datamation*, 1994.

[18] K. J. Koval, C. W. Tingey, and K. F. Spratt. Are Patients Being Transferred to Level-I Trauma Centers for Reasons Other than Medical Necessity? *The Journal of Bone & Joint Surgery*, 88(10):2124–2132, 2006.

[19] W. Li. Mutual information functions versus correlation functions. *Journal of Statistical Physics*, 60(5):823–837, 1990.

[20] E. J. MacKenzie, F. P. Rivara, G. J. Jurkovich, A. B. Nathens, K. P. Frey, B. L. Egleston, D. S. Salkever, and D. O. Scharfstein. A National Evaluation of the Effect of Trauma-Center Care on Mortality. *New England Journal of Medicine*, 354(4):366–378, 2006. PMID: 16436768.

[21] L. Moore, J. A. Hanley, A. Lavoie, and A. Turgeon. Evaluating the validity of multiple imputation for missing physiological data in the National Trauma Data Bank. *Journal of Emergencies, Trauma & Shock*, 2(2):73–79, 2009.

[22] L. Moore, J. A. Hanley, A. F. Turgeon, A. Lavoie, and M. Emond. A Multiple Imputation Model for Imputing Missing Physiologic Data in the National Trauma Data Bank. *Journal of the American College of Surgeons*, 209(5):572–279, 2009.

[23] L. Moore, A. Lavoie, N. LeSage, M. Liberman, J. S. Sampalis, E. Bergeron, and B. Abdous. Multiple Imputation of the Glasgow Coma Score. *The Journal of Trauma: Injury, Infection, and Critical Care*, 59(3):698–704, 2005.

[24] Oracle. Oracle Analytic Workspace Manager. http://www.oracle.com/technetwork/database/options/olap/olap-downloads-098860.html.

[25] T. A. Oyetunji, J. G. Crompton, I. D. Ehanire, K. A. Stevens, D. T. Efron, E. R. Haut, D. C. Chang, E. E. Cornwell III, M. L. Crandall, and A. H. Haider. Multiple Imputation in Trauma Disparity Research. *Journal of Surgical Research*, 165(1):e37 – e41, 2011.

[26] Pentaho. Pentaho Analysis Services. http://mondrian.pentaho.com/.

[27] B. Roudsari, C. Field, and R. Caetano. Clustered and missing data in the US National Trauma Data Bank: implications for analysis. *Injury Prevention*, 14(2):96–100, 2008.

[28] G. Salton and M. J. McGill. Introduction to modern information retrieval. 1986.

[29] S. Sarawagi, R. Agrawal, and A. Gupta. *On computing the data cube*. IBM Research Division, 1996.

[30] A. Srivastava, L. Ferrigno, S. Kaminski, X. Yan, and J. Su. Incident Cubes : A framework for data aggregation and exploration in NTDB. http://cs.ucsb.edu/~isel/IncidentCubes/, 2013.

[31] J. William Thomas. Risk Adjustment for Measuring Health Care Outcomes, 3rd edition. *International Journal for Quality in Health Care*, 16(2):181–182, 2004.

[32] S. Urbanek. Data Brewery: A set of frameworks with tools for Data Analysis. http://databrewery.org/.

[33] S. Urbanek. Cubes: Light-weight Online Analytical Processing. http://pythonhosted.org/cubes/, 2011.