Proceedings of the 11th International Workshop on Quality in Databases (QDB'16)

Christoph Quix
Fraunhofer FIT
St. Augustin, Germany
christoph.quix@fit.fraunhofer.de

Rihan Hai
RWTH Aachen University
Aachen, Germany
hai@dbis.rwth-aachen.de

Hongzhi Wang
Harbin Institute of Technology
China
wangzh@hit.edu.cn

Verikat N. Gudivada
East Carolina University
Greenville, USA
gudivadav15@ecu.edu

Laure Berti
Hamad Bin Khalifa University
Qatar
lberti@qf.org.qa

Message from the Workshop Chairs

Data quality problems arise frequently when data is integrated from disparate sources. In the context of Big Data applications, data quality is becoming more important because of the unprecedented volume, large variety, and high velocity. The challenges caused by volume and velocity of Big Data have been addressed by many research projects and commercial solutions and can be partially solved by modern, scalable data management systems. However, variety remains to be a daunting challenge for Big Data Integration and requires also special methods for data quality management. Variety (or heterogeneity) exists at several levels: at the instance level, the same entity might be described with different attributes; at the schema level, the data is structured with various schemas; but also at the level of the modeling language, different data models can be used (e.g., relational, XML, or a document-oriented JSON representation). This might lead to data quality issues such as consistency, understandability, or completeness. The heterogeneity of data sources in the Big Data Era requires new integration approaches which can handle the large volume and speed of the generated data as well as the variety and quality of the data. Thus, heterogeneity and data quality are seen as challenges for many Big Data applications. While in some applications, a limited data quality for individual data items does not cause serious problems when a huge amount of data is aggregated, data quality problems in data sources are often revealed by the integration of these sources with other information. Data quality has been coined as 'fitness for use'; thus, if data is used in another context than originally planned, data quality might become an issue. Similar observations have been also made for data warehouses which lead to a separate research area about data warehouse quality.

The 11th International Workshop on Quality in DataBases (QDB) continued the successful workshop series on data quality management in database systems. This year's workshop had a special focus on problems related to Big Data Integration and Big Data Quality.

We received about 10 submissions from which four were be accepted as regular papers. In addition, the workshop had a very compelling keynote talk by Divesh Srivastava about data glitches which he characterized as constraint violations without empirical explanations. The workshop program was completed by four invited research presentations that especially focused on the practical aspects of data quality management and introduced systems that address the problems in Big Data Integration or Big Data Quality.

Workshop Chairs

Christoph Quix
Fraunhofer FIT, Germany

Rihan Hai
RWTH Aachen University, Germany

Hongzhi Wang
Harbin Institute of Technology, China

Verikat N. Gudivada
East Carolina University, USA

Laure Berti
Hamad Bin Khalifa University, Qatar

Program Committee

Helena Galhardas, Instituto Superior Tcnico, University of Lisbon and INESC-ID
Nan Tang, QCRi
Barbara Pernici, Politecnico di Milano
Cinzia Cappiello, Politecnico di Milano
Wenjie Zhang, UNSW
Felix Naumann, HPI
Melanie Herschel, Universitt Stuttgart
Christian Bizer, University of Mannheim
Divesh Srivastava, AT&T Labs Research
Eduard Dragut, Temple University
Andrea Maurino, universit degli studi di Milano - Bicocca
Mario Piattini, University of Castilla-La Mancha
Panos Vassiliadis, University of Ioannina
Weiyi Meng, Department of Computer Science, Binghamton University
Paolo Papotti, Arizona State University
Dmitri V. Kalashnikov, AT&T Labs Research
Michael Gertz, Heidelberg University
Srividya Bansal, Arizona State University
Hongwei Zhu, University of Massachusetts Lowell
Jiannan Wang, Simon Fraser University
Lukasz Golab, University of Waterloo
David Kensche, RWTH Aachen University
Erhard Rahm, University of Leipzig
Ulf Leser, Institut fr Informatik, Humboldt-Universität zu Berlin
Jolon Faichney, Griffith University
Kai-Uwe Sattler, TU Ilmenau
Tenindra Abeywickrama, Monash University
Michael Klaes, Fraunhofer Institute for Experimental Software Engineering

Table of Contents

Keynote Talk

Regular Papers

Abstracts of Invited Presentations

Data Glitches = Constraint Violations - Empirical Explanations

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

ABSTRACT

Data glitches are unusual observations that do not conform to data quality expectations, be they semantic or syntactic, logical or statistical. By naively applying integrity constraints, potentially large amounts of data could be flagged as being violations. Ignoring or repairing significant amounts of the data could fundamentally bias the results and conclusions drawn from analyses. In the context of Big Data where large volumes and varieties of data from disparate sources are integrated, it is likely that significant portions of these violations are actually legitimate usable data. We conjecture that empirical glitch explanations – concise characterizations of subsets of violating data – could be used to (a) identify legitimate data and release them back into the pool of clean data, thereby reduce cleaning-related statistical distortion of the data; and (b) refine existing integrity constraints and generate improved domain knowledge. We present a few real-world case studies in support of our conjecture, outline scalable techniques to address the challenges of discovering explanations, and demonstrate the utility of the explanations in reclaiming over 99% of the violating data.

DIRA: Data Integration to Return Ranked Alternatives

Reham I. Abdel Monem
Information Systems Department, Faculty of Computers and Information, Cairo University, Giza, Egypt
reham@fci-cu.edu.eg

Ali H. El-Bastawissy
Faculty of Computer Science, MSA University, Giza, Egypt
alibasta@fci-cu.edu.eg

Mohamed M. Elwakil
The Software Engineering Laboratory, Innopolis University, Innopolis, Russia
m.elwakil@innopolis.ru

ABSTRACT
Data integration (DI) is the process of collecting data needed for answering a query from distributed and heterogeneous data sources and providing users with a unified form of this data. Data integration is strictly tied with data quality due to two main data integration challenges first, providing user with high qualitative query results second, identifying and solving values conflicts on the same real-world objects efficiently and in the shortest time. In our work, we focus on providing user with high qualitative query results.

The quality of a query result can be enhanced by evaluating the quality of the data sources and retrieving results from the significant ones only. Data quality measures are used not only for determining the significant data sources but also in ranking data integration results according to user-required quality and presenting them in a reasonable time. In this paper, we perform an experiment that shows a mechanism to calculate and store a set of quality measures on different granularities through new data integration framework called data integration to return ranked alternatives (DIRA). These quality measures are used in selecting the most significant data sources and producing top-k query results according to query types that we proposed. DIRA validation using the transaction processing performance council (TPC) benchmark version called TPC-DI will show how our framework improves the returned query results.

1. INTRODUCTION
Data integration system (DIS) is a system where query results are combined from different and autonomous data sources. These query results may be found in one source or distributed among many sources [1].

Data integration may face three types of heterogeneity: technological heterogeneities because products are used by different vendors, applied in different categories of information and communication infrastructures, schema heterogeneities due to the use of different data models and different data representations and instance heterogeneities where the same object from different data sources represented by different data values. In our work, we focused on instance heterogeneities where data quality problems become very evident and as they have big effect on the query processing in data integration.

Such systems have different architectures but virtual integration and data warehousing architectures are the most commonly used ones. In this work, we use the virtual integration architecture where many local data sources are combined together to form a single virtual data source, data are stored in local data sources and are accessed through global schema which is the presentation where users send their queries to a data integration system.

Data sources quality changes frequently so it is important to store some data sources quality measures to use them during query planning.

In our previous work (DIRA) [2], we presented data integration framework called Data Integration to return Ranked Alternatives (DIRA) that uses data quality (DQ) in data integration systems to improve their performance and query results quality. This framework adds quality system components and data quality assessment module to any data integration system.

In this paper, we report on an experiment on the DIRA framework that uses the data integration benchmark TPC_DI [3].

In this paper data quality measures and the way of their assessment are introduced in section 2. Section 3 presents the DIRA framework. Experiments on DIRA framework using the data integration benchmark TPC_DI [3] are presented in section 4. Conclusion and future work are introduced in section 5.

2. DATA QUALITY MEASURES USED IN DATA INTEGRATION
Data quality is fitness for use or the ability to meet user's needs [4]. There are a lot of measures to assess data quality called data quality measures. They are classified according to many aspects [5] and table 1 presents one of their classifications.

Table 1. Information quality measures classification for data integration [6]

Data Integration Components	IQ criteria
Data Source	Reputation, Verifiability, Availability and Response Time.
Schema	Schema Completeness, Minimalism and Type Consistency.
Data	Data Completeness, Data Timeliness, Data Accuracy and Data Validity.

In DIRA, we focused on data quality measures important for user and related to data in the integration process.

Quality measures can assess the quality of information better if we use a combination of metrics, subjective ratings and qualitative description of issues. We use that in DIRA assessment module.

There are different levels of granularities that data quality measures can be calculated according to them and table 2 presents our selected data quality measures and the granularity level for each selected measure.

Table 2. Data quality measures selected by DIRA and the granularity for each measure

Data Quality Measures	Measures Granularities		
	Data Source Level	Relation Level	Attribute Level
Completeness			✓
Validity			✓
Accuracy			✓
Timeliness		✓	

Following sub-sections explain our selected data quality measures definitions and the equations for their assessment:

2.1 Data completeness

Data completeness categorized into two types: Null-Completeness and Population-Completeness. Null-Completeness represents the extent to which data set contains missing values. Population-Completeness represents the extent to which all needed data by the user is available [5].

In DIRA, Fact-Completeness was introduced and is defined as an inferred and accurate type of completeness which value is assessed using Null-Completeness and Population-Completeness.

Completeness types assessment at attribute level will be concluded as follows (Where a_m (r) is attribute number m in relation r) [7]:

- Null-Completeness assessment (C_{Null}): It is the percentage of existing values (non-null values) to the whole number of values in the universal relation.

$$C_{Null}(a_m\ (\mathrm{r})) \frac{\text{Number of non-null values } a_m\,(\mathrm{r})}{\text{Total number of values in the universal relation}} (1)$$

- Population-Completeness assessment ($C_{Population}$): It is the percentage of actually presented rows in a relation r to the number of rows in ref(r) where ref(r), is the relation of all rows that satisfy r relational schema.

$$C_{Population}(a_m\ (\mathrm{r})) = \frac{Cardinality\ of\ a_m\,(\mathrm{r})}{Cardinality\ of\ ref(r)} \quad (2)$$

- Fact-Completeness assessment (C_{fact}): It is subtraction of the number of missing values from the total number of existing values divided by the whole number of values in ref(r).

  If reference relation isn't available, fact-completeness will equal null-completeness.

$$InC_{Null}(a_m\,(\mathrm{r})) = \frac{\text{Number of null values for } a_m\,(r)}{Cardinality\ of\ ref(r)} \quad (3)$$

$$C_{fact}(a_m\,(\mathrm{r})) = C_{Population}\ (a_m\,(\mathrm{r})) - InC_{Null}\ (a_m\ (\mathrm{r})) \quad (4)$$

- Data completeness scaled aggregate value (C_{Type}) for attributes required by user in query

$$\text{Scaled Total } (C_{Type}) = \frac{\sum_{m=1}^{M} C_{Type}(a_m(r))}{M} \quad (5)$$

  Where M represents the number of attributes that required by the user in the query

2.2 Data validity

Data validity is the extent to which attribute value within specified domain [5].

Validity assessment at attribute level concluded as follows (Where a_m (r) is attribute number m in relation r) [7]:

- Data validity assessment (l): It is the percentage between of valid values and the whole number of values in the universal relation.

$$l(a_m\ (\mathrm{r})) = \frac{\text{Number of valid values } a_m\,(\mathrm{r})}{\text{Total number of values in the universal relation}} (6)$$

- Data validity scaled aggregate value (L) for attributes required by user in query

$$\text{Scaled Total } (L) = \frac{\sum_{m=1}^{M} l(a_m(r))}{M} \quad (7)$$

Where M represents the number of attributes that required by the user in the query.

2.3 Data accuracy

Data accuracy is divided into two types: semantic accuracy and (0 or 1) accuracy. Semantic accuracy represents the gap between recorded value v and correct value v'. (0 or 1) accuracy represents the ratio between data values considered accurate (they don't conflict with real-world values) and the total number of values in the universal relation. (0 or 1) accuracy was the type used in our work [5].

Since a reference relation is almost always missing, costly and time consuming, we compare the value of each attribute to its domain of allowed values (it will consider as validity) but if reference relation is available and user not care about cost or time; accuracy will be calculated and can be improved by complaints and domain experts' feedback that identify erred data and a correction for them.

Accuracy assessment at attribute level will be concluded as follows (Where a_m (r) is attribute number m in relation r) [7]:

Data accuracy assessment (a): It is the percentage of accurate values and the whole number of values in the universal relation.

$$a(a_m\ (\mathrm{r})) = \frac{\text{Number of accurate values } a_m\ (\mathrm{r})}{\text{Total number of values in the universal relation}} (8)$$

- Data Accuracy scaled aggregate value (A) for attributes required by user in query

$$\text{Scaled Total } (A) = \frac{\sum_{m=1}^{M} a(a_m(r))}{M} \quad (9)$$

Where M represents the number of attributes that required by user in query

2.4 Data timeliness

Data timeliness is the extent to which data is up-to-date [5].In DIRA, we judge how far the data is modern using insertion time and volatility that we store in DIRA metadata structure.

Timeliness assessment at relation level is concluded as follows [7]:

Data timeliness assessment (t): Timeliness for relation r can be calculated using currency and volatility variables that will be presented in the following equations 10 and 11.

Currency = Age + (DeliveryTime – InputTime) (10)

Where:

Currency: It reflects how far the data is modern.[8]

Age: It reflects how old the data is when it is delivered.

DeliveryTime: Data delivery time to user.

InputTime: The data obtaining time.

$$\text{Timeliness (t (r))} = \max\left\{0{,}1 - \frac{Currency}{Volatility}\right\} \quad (11)$$

Where:

Volatility: The data validity lifetime.

In our work, we suppose that DeliveryTime = InputTime (no time between obtaining data and delivering it to the user) so Currency = Age

- Data timeliness aggregate value (T) for attributes required by user in query

$$Total(T) = \text{Maximum (t (r))} \quad (12)$$

3. DATA INTEGRATION TO RETURN RANKED ALTERNATIVES (DIRA) FRAMEWORK

DIRA framework consists of data quality assessment module and data quality system components.

3.1 DIRA data quality assessment module

There are many components to assess data quality in DIRA assessment module; these components are [9]:

- Assessment metrics. They are procedures for assessing data quality measures assessment scores using a scoring function.
- Aggregation metrics**.** They are procedures for assessing aggregated score. This aggregate score is calculated from individual assessment scores using aggregation functions like sum, count, and average functions.
- Data quality measures. They are metadata for describing how data is suitable for a specific task.
- Scoring functions. They are the way for assessing data quality measures. They may be simple comparisons, set function, aggregation function and complex statistical function.

3.2 DIRA quality system components

Quality system components consist of data quality acquisition and user input, these components are added to integration systems to enhance query results.

3.2.1 Data quality acquisition

Data quality acquisition stores data sources columns, tables and data quality assessment module results in the metadata store.

Hierarchical quality framework [10] that is introduced in DIRA is used in building DIRA metadata store entities presented in figure 1.

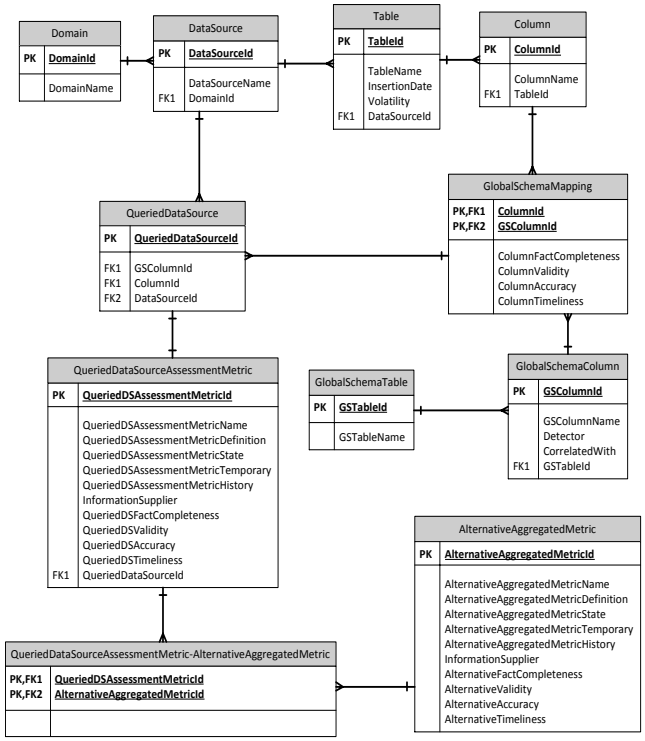


Figure 1. DIRA metadata structure [2]

3.2.2 User input

Qualified results can be returned to the user using some quality measures in a user query. SQL can include some quality measures as we present in the following query [1]:

```
Select A1… Ak
From G
Where < selection condition >
With < data quality goal >

Where A1, A2, Ai are global attributes of G
```

3.3 DIRA workflow components

DIRA consists of some basic components that can clarify how DIRA works. These components are

- Data quality assessment metrics of data sources attributes (columns). They are responsible for calculating the chosen data quality measures scores for all data sources attributes (columns) with its associated columns in the global schema. These scores are stored in global schema mapping entity.
- Data quality assessment metrics for queried data sources. Queried data sources are data sources that participate in query answering by attributes. In these assessment metrics, the aggregate data quality score for all attributes that the data source will participate with will be assessed for each measure.
- Alternatives formation. Alternative is one or more queried data source. There are two types of alternatives: qualified alternatives and not qualified alternatives. Alternative

considers qualified if it satisfies the specified quality level in the user query and not qualified otherwise.

In our framework, query type specifies the way of alternatives formation. Queries devoid of any quality constraint, combinations of all queried data sources are formed to build alternatives and all are qualified alternatives. Queries have one or more quality constraint, qualified alternatives of single queried data source that satisfies the specified quality level are built and other data sources are pruned from forming alternatives (First Pruning) then alternatives from more than one queried data sources are built from combinations of all queried data sources, alternatives that don't satisfy the quality constraint are pruned (Second Pruning) and the rest are qualified alternatives.

- Alternatives aggregated metrics. They are responsible for assessing aggregated scores of alternatives containing more than one queried data source.

  Alternative aggregated score is single representative value for alternative queried data sources assessment scores that assessed using aggregate function.

  There are different aggregate functions to represent single value for collection of values like average (mean), mode and median. In DIRA, we use simple average (arithmetic mean) aggregate function.

  Simple average is aggregate function that provides accurate description of entire data and uses every value in data so it considers good representative for data. Following in table 3 we present what the best aggregate function is with respect to the different types of variables.

Table 3. The best aggregate function with respect to the different types of variables[11][12]

Type of variable	The best aggregate function
Nominal	Mode
Ordinal	Median
Interval/Ratio	Mean or median

These aggregated scores are built from queried data sources assessment metrics according to the following equations [13].

- Alternative data fact completeness

$$C_{DSs} = \sum_{q=1}^{Q} C_{fact}(DS_q)/Q \quad (13)$$

- Alternative data validity

$$L_{DSs} = \sum_{q=1}^{Q} L(DS_q)/Q \quad (14)$$

- Alternative data accuracy

$$A_{DSs} = \sum_{q=1}^{Q} A(DS_q)/Q \quad (15)$$

- Alternative data timeliness

$$T_{DSs} = \text{Maximum}\big(T(DS_q)\big) \quad (16)$$

- Alternatives ranking according to proposed queries types. In this component, alternatives ranking based on different types of queries; No-measure top-k selection query, quantitative measure-feature top-K selection query, qualitative single-measure top-K selection query, quantitative multi-measure top-K selection query and qualitative multi-measure top-K selection query; These types vary in number of quality measures in query condition (from one to four) and the kind of quality measures value (quantitative or qualitative).

  In no-measure top-k selection query, alternatives returned to the user are ranked according to all scope measures and user chooses the most suitable ranking for him.

  In quantitative single-measure top-K selection query, alternatives returned to user are ranked according to specified data quality measure in user query, in qualitative single-measure top-K selection query, DIS receives message as input from user with quality measure score that represents the qualitative value for quality measure in user query and DIS returns alternatives ranked according to required data quality measure in user query.

  In quantitative multi-measure top-K selection query, alternatives returned to user are ranked through ranking algorithm called threshold algorithm [14] according to three case; Case1: user's query contains data quality measures, they are separated with (AND) and all are fulfilled, in this case, threshold algorithm returns alternatives ranked by total score. Case2: user's query contains data quality measures, they are separated with (AND) and the quality level for one or more of data quality measures isn't compatible with quality level for other data quality measures or isn't achieved, in this case, user receives a message to notify him that the quality level for query answering can't be achieved. Case3: user's query contains data quality measures, they are separated with (OR) and the quality level for all data measures fulfilled or the quality level for one or more of data quality measures isn't compatible with the quality level for other data quality measures or can't be achieved, in this case, threshold algorithm ranks alternatives with total score.

  In qualitative multi-measure top-K selection query, DIS returns alternatives ranked by threshold algorithm according to previous three cases but after receiving the message as input from the user with quality measures scores that represent the qualitative values for quality measures in user's query.

Following in figure 2, we present DIRA workflow components that we explained above

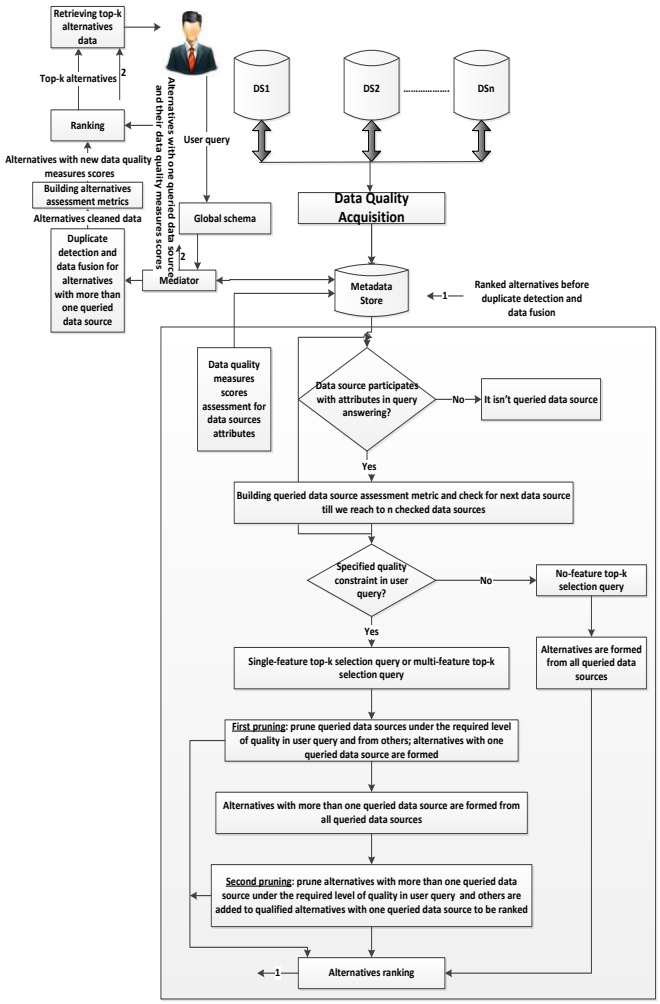


Figure 2. DIRA workflow components [2]

4. EXPERMENTS AND RESULTS

In this section, we clarify how to implement and validate our data integration framework and data quality system components. The experiments aim to calculate the response time and data sources number used to return query results. The following execution paths are the principles of our experiments:

- No-measure top-k selection query. This means, query doesn't contain quality constraints. Top-k alternatives are returned ranked by every scope data quality measure.
- Single-measure top-k selection query. This means, user specifies one data quality measure as a constraint, the data integration system (DIS) retrieves top-k alternatives ranked according to specified data quality measure.
- Multi-measure top-k selection query. This means, user specifies more than one data quality measure as a constraint, the data integration system (DIS) retrieves top-k alternatives ranked according to specified data quality measures together.

We execute the experiments on a laptop with an Intel(R) Core(TM) I5-5200U CPU @ 2.20GHz and 8 GB RAM. The laptop works with Windows 8.1 Enterprise edition. Microsoft SQL Server 2014 and Microsoft Visual Studio Express 2012 C# are tools that we use in our experiments.

4.1 TPC-DI benchmark

Transaction processing performance council (TPC) is an organization that defines benchmarks related to transaction processing and database. TPC benchmarks are TPC-C, TPC-DI, TPC-DS, TPC-E, TPC-H, TPC-VMS, TPCx-HS and TPCx-V, evaluating computer systems performance is their goal [15].

Our scope benchmark is TPC-DI; TPC released the first version of its data integration benchmark in January 2014.TPC-DI uses some tools to estimate the performance of data integration systems. Figure 3 illustrates a conceptual view of TPC-DI benchmark.

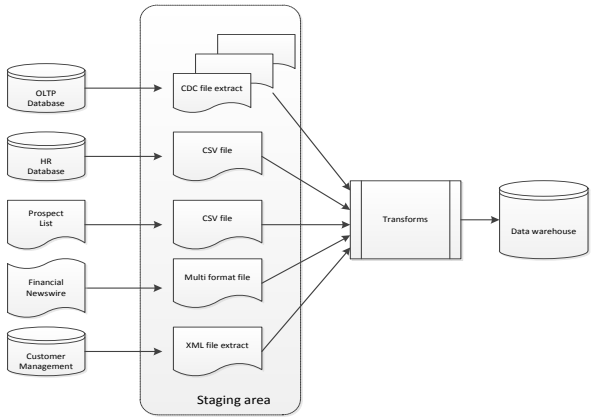


Figure 3. A conceptual view of TPC-DI benchmark [16]

The benchmark defines:

- Many schemas for data sources and file formats.
- The way to generate source data and how to store them.
- The schema for destination source (data warehouse).
- The way to transform and move data from data sources to the data warehouse.

In our experiment, we use only TPC-DI data sources to test our framework but with some modifications in data as TPC-DI benchmark depends on 100% accurate data sources but in our framework we depend on data sources with different levels of data quality. Data modification was done by adding errors in data; we replaced some values with nulls and others with not valid and not accurate values. We will consider the original TPC-DI data as reference data.

TPC-DI data sources that we use in our experiments are an online transaction processing database (OLTP DB) a human resource system (HR) and a customer relationship management system (CRM).

TPC data sources files are created using data integration generator called (DIGen) [16]. DIGen uses parallel data generation framework (PDGF) in data generation. PDGF is a common data generation framework that provides a set of data generation capabilities to generate specific TPC-DI data with specific prosperities [16].

We downloaded TPC-DI tool that contains DIGen file and PDGF folder from TPC website [15]. We downloaded Java SE 8 as Java

Virtual Machine (JVM) with a minimum of Java SE 7 must be used with DIGen to create the source data.

We used some commands to generate source data by DIGen like "java –jar DIGen.jar". The generated source data was in some batches in the form of file.txt, file.xml and file.csv. DIGen also generate statistics file named "digen_report.txt". The statistics file has some information about the way to generate data and number of rows in each batch. The schemas created in a SQL server database called "TPC-DI" and loaded the data into it.

As we mentioned above in section 3.2.1, the process of storing data sources attributes and relations in the metadata store is the responsibility of data quality acquisition component (N/P: In our experiment we will work in relation with population – completeness = 1). It executes also data quality queries on the data sources and stores their results in the metadata store. So, metadata store described in figure 1 was created to include eleven tables in the same database "TPC-DI". We also build a mapping tool to match the global schema columns with the local schema columns.

Table 4 presents the used global schema tables and global schema columns:

Table 4. Global schema tables and global schema columns that will be used in our experiment

Global Schema Tables	Global Schema Columns			
Customer	CustomerId	CLastName	CFirstName	CGender
	CAddressLine	CCity	CState	CPhone
	CCountry	CAge	c_m_name	c_maritalstatus
	c_postalcode	c_income	c_networth	c_numbercards
Account	CA_ID	CA_NAME	CA_STATE	

We use stored procedures to implement data quality queries that run by the data quality acquisition component. Those stored procedures have the equations used to assess the completeness, validity, accuracy and timeliness of the data sources attributes, execute according to schedule job created by SQL server and re-execute as soon as data sources update.

User input is the second component of data quality system components. User input lets the user specifies as optional the quality constraints. He can select between completeness or validity or accuracy or timeliness or any combinations. The user also can specify the quality levels for his chosen quality constraints, in addition to the number of alternatives that he wants to retrieve.

Following tables and response time of data sources used in our experiment are presented in table 5

Table 5. Tables and response time of data sources

Data Sources	Tables	Response Time
OLTP (DS1)	Customer and Account	500 sec
HR (DS2)	Employee	500 sec
CRM (DS3)	Customer and Account	500 sec

Following queried data sources assessment metrics are presented in table 6 (N/P: Required attributes in user query are CFirstName, CLastName, CAddressLine, CPhone and Cage and we assume that user chooses top-1 alternative from the list of ranked alternatives).

Table 6. Queried data sources assessment metric

Queried Data Sources	Retrieved Attributes	Aggregate Completeness for Retrieved Attributes	Aggregate Validity for Retrieved Attributes	Aggregate Accuracy for Retrieved Attributes	Aggregate Timeliness for Retrieved Attributes
DS1	CFirstName CLastName CAddressLine CPhone CAge	0.998	0.982	0.975	0.773
DS2	CFirstName CLastName CAddressLine CPhone CAge	0.987	0.964	0.956	0.628
DS3	CFirstName CLastName CAddressLine CPhone CAge	0.987	0.956	0.948	0.299

Table 7 presents simple form for alternatives aggregated metrics[2] according to scope data quality measures and queried data sources presented in table 6.

Table 7. Alternatives aggregated metrics

Alternative Name	Alternative Queried Data Sources	Alternative Completeness	Alternative Validity	Alternative Accuracy	Alternative Timeliness
Alternative1	DS1	0.998	0.982	0.975	0.773
Alternative2	DS2	0.987	0.964	0.956	0.628
Alternative3	DS3	0.987	0.956	0.948	0.299
Alternative4	DS1,DS2	0.992	0.973	0.966	0.773
Alternative5	DS1,DS3	0.992	0.969	0.962	0.773
Alternative6	DS2,DS3	0.987	0.960	0.952	0.628
Alternative7	DS1,DS2, DS3	0.990	0.967	0.960	0.773

4.1.1 Number of ranked alternatives and the number of accessed data sources in user selected alternative (Example 1)

In example 1, we choose completeness and timeliness from scope data quality measures as quality constraints and table 8 presents the number of returned ranked alternatives according to our framework and the number of accessed data sources in user selected alternative that used to execute query according to different execution paths.

Table 8. Number of ranked alternatives and the number of accessed data sources in user selected alternative

Execution Paths	Retrieved Attributes	Quality Constraint (Optional)	Number of Accessed Data Sources	Number Of Ranked Alternatives
No-measure top-k selection query	CFirstName CLastName CAddressLine CPhone CAge	-	3	7
Single-measure top-k selection query	CFirstName CLastName CAddressLine CPhone Cage	Completeness > 0.992	1	1

Multi-measure top-k selection query	CFirstName CLastName CAddressLine CPhone CAge	Completeness > 0.992 and Timeliness > 0.7	1	1

The results in table 8 and figure 4 show that if no determined quality measures; whole data sources need to be queried by DIS and return all combinations of data sources as alternatives. While adding quality measures reduce the number of accessed data sources to 1 instead of 3 and the number of returned alternatives that satisfy user requirements to 1 instead of 7.

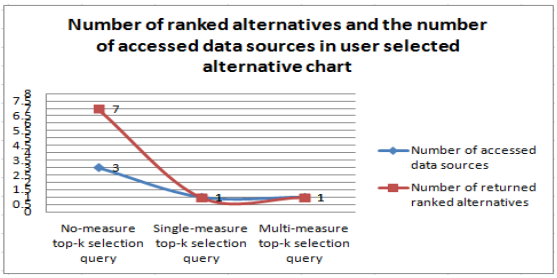


Figure 4. Number of ranked alternatives and the number of accessed data sources in user selected alternative chart

4.1.2 Response time (Example 1)

It is the time between the mediator query submission and receiving complete query answers from data sources.

In our work, we use calibration techniques[17],[18] to measure response time. The standard unit to measure the time interval is seconds. We assume that all data sources have the same capabilities for answering queries, network traffic, the servers' workload, database management system and hardware.

Table 9 shows the response time of our framework according to example 1 in different execution paths.

Table 9. Response time

Execution Paths	Retrieved Attributes	Quality Constraint (Optional)	Response Time (sec)
No-measure top-k selection query	CFirstName CLastName CAddressLine CPhone CAge	-	3.225 sec
Single-measure top-k selection query	CFirstName CLastName CAddressLine CPhone CAge	Completeness > 0.992	2.595 sec
Multi-measure top-k selection query	CFirstName CLastName CAddressLine CPhone CAge	Completeness > 0.992 and Timeliness > 0.7	2.595 sec

Table 9 and figure 5 shows that response time reduced by adding data quality measures.

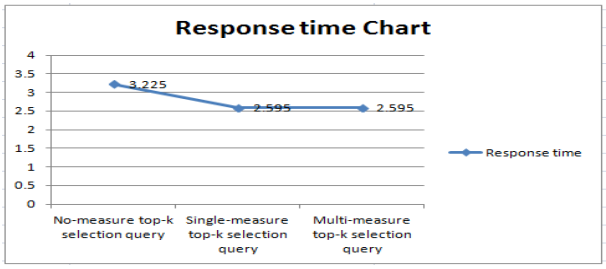


Figure 5. Response time chart

4.1.3 Number of ranked alternatives and the number of accessed data sources in user selected alternative (Example 2)

In example 2, user query become more complex by using completeness, validity, accuracy and timeliness together as quality constraints, the required quality level for them is satisfied by more than one alternative and one of satisfied alternatives consists of more than one queried data source that they integrate to achieve the required quality levels.

Table 10 presents the number of returned ranked alternatives according to our framework and the number of accessed data sources in user selected alternative that used to execute query according to different execution paths.

Table 10. Number of ranked alternatives and the number of accessed data sources in user selected alternative

Execution Paths	Retrieved Attributes	Quality Constraint (Optional)	Number of Accessed Data Sources	Number Of Ranked Alternatives
No-measure top-k selection query	CFirstName CLastName CAddressLine CPhone CAge	-	3	7
Single-measure top-k selection query	CFirstName CLastName CAddressLine CPhone CAge	Completeness > 0.990	1	3
Multi-measure top-k selection query	CFirstName CLastName CAddressLine CPhone CAge	Completeness > 0.990 and validity > 0.960 and accuracy > 0.962 and Timeliness > 0.7	1	2

The results in table 10 and figure 6 show that if no determined quality measures; whole data sources need to be queried by DIS and return all combinations of data sources as alternatives. While adding quality measures reduce the number of accessed data sources to 1 instead of 3 (the number of accessed data source may not reduce in another queries and it is a worth case) and the number of returned alternatives that satisfy user requirements to 2 or 3 instead of 7.

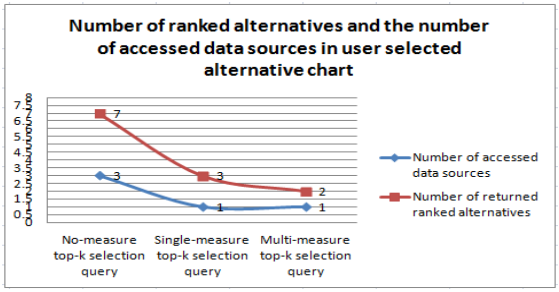


Figure 6. Number of ranked alternatives and the number of accessed data sources in user selected alternative chart

4.1.4 Response time (Example 2)

Table 11 shows the response time of our framework according to example 2 in different execution paths.

Table 11. Response time

Execution Paths	Retrieved Attributes	Quality Constraint (Optional)	Response Time (sec)
No-measure top-k selection query	CFirstName CLastName CAddressLine CPhone	-	3.225 sec

	CAge		
Single-measure top-k selection query	CFirstName CLastName CAddressLine CPhone CAge	Completeness > 0.990	2.595 sec
Multi-measure top-k selection query	CFirstName CLastName CAddressLine CPhone CAge	Completeness > 0.990 and validity > 0.960 and accuracy > 0.962 and Timeliness > 0.7	2.595 sec

Table 11 and figure 7 shows that response time reduced by adding data quality measures.

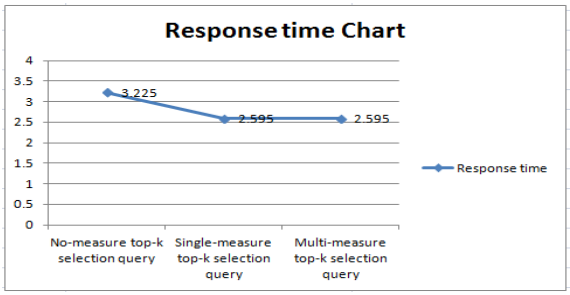


Figure 7. Response time chart

5. CONCULSION AND FUTURE WORK

Query results obtained from data integration system have some problems; they are all returned to the user from all queried data sources without any specified quality level and hence they are not ranked and they take a long time.

In this paper, we present data integration framework called DIRA, this framework improves query results obtained from DIS and generates them in reasonable time by adding quality system components and data quality assessment module to any DIS to retrieve results from only convenient data sources and return these results ranked according to their quality in both cases if quality measures are specified in user query or not.

Our experiments illustrate that our framework can retrieve results with number of data sources less than the original DIS, hence less number of ranked alternatives in a reasonable time.

We can extend our work to include different types of databases like semi-structured and unstructured data sources, use additional data quality measures, use different ranking algorithms, use both as view (BaV) as data integration system mapping technique and use integrity constraints in global schema to make query answers more consistent.

6. REFERENCES

[1] M. S. Abdel-moneim and A. H. El-bastawissy, "Data Quality Based Data Integration Approach," World of Computer Science and Information Technology Journal (WCSIT), vol. 5, no. 10, pp. 155–164, 2015.

[2] Reham I. Abdel Monem, A. H. El-bastawissy, and M. M. Elwakil, "DIRA : A Framework Of Data Integration Using Data Quality," International Journal of Data Mining & Knowledge Management Process (IJDKP), vol. 6, no. 2, pp. 37–58, 2016.

[3] M. Poess, T. Rabl, B. Caufield, and I. Datastage, "TPC-DI : The First Industry Benchmark for Data Integration," the 40th International Conference on Very Large Data Bases, vol. 7, no. 13, pp. 1367–1378, 2014.

[4] F. Sidi, P. Hassany, S. Panahy, L. S. Affendey, M. A. Jabar, H. Ibrahim, and A. Mustapha, "Data Quality : A Survey of Data Quality Dimensions," IEEE, pp. 300–304, 2012.

[5] M. Kaiser, "A Conceptual Approach to Unify Completeness , Consistency , and Accuracy as Quality Dimensions of Data Values," European and Mediterranean Conference on Information Systems, vol. 2010, pp. 1–17, 2010.

[6] C. Moraes and A. C. Salgado, "Information Quality Measurement in Data Integration Schemas," ACM, 2007.

[7] C. Batini and M. Scannapieco, Data Quality concepts,Methodologies and techniques. 2006.

[8] W. Fan, F. Geerts, N. Tang, and W. Yu, "Inferring Data Currency and Consistency for Conflict Resolution," ICDE, 2013.

[9] P. N. Mendes, H. Mühleisen, and C. Bizer, "Sieve: Linked Data Quality Assessment and Fusion," Proceedings of the 2012 Joint EDBT/ICDT Workshops, pp. 116–123, 2012.

[10] I. N. R. Etrieval, "A Flexible Quality Framework For Use Within Information Retrieval," Proceedings of the Eighth International Conference on Information Quality (ICIQ-03), pp. 297–313.

[11] "Measures of central tendency." [Online]. Available: https://statistics.laerd.com/statistical-guides/measures-central-tendency-mean-mode-median.php.

[12] "Fundamentals of statics." [Online]. Available: http://www.usablestats.com/lessons/noir.

[13] P. Angeles and F. García-ugalde, "A Data Quality Practical Approach," International Journal on Advances in Software, vol. 2, no. 2, pp. 259–274, 2009.

[14] I. F. Ilyas, G. Beskales, and M. a. Soliman, "Query Processing Techniques in Relational Database Systems," ACM Computing Surveys, vol. 40, no. 4, pp. 1–58, 2008.

[15] "TPC." [Online]. Available: http://www.tpc.org/information/benchmarks.asp.

[16] S. Specification and A. R. Reserved, "TPC BENCHMARK ™ DI Transaction Processing Performance Council (TPC)," no. November, 2014.

[17] R. M. G. and M. A. L. Jeff C.Gust, "Stopwatch and Timer Calibrations (2009 edition)," 2009.

[18] M. Spiliopoulou, I. Wirtschaftsinformatik, and H. Berlin, "A Calibration Mechanism Identifying the Optimization Technique of a Multidatabase Participant 3 Optimizer Calibration Methodology," Conference on Parallel and Distributed Computing Systems (PDCS), Dijon, France, 1996.

Communicating Data Quality in On-Demand Curation

Poonam Kumari
University at Buffalo
poonamku@buffalo.edu

Said Achmiz
achmizs@gmail.com

Oliver Kennedy
University at Buffalo
okennedy@buffalo.edu

ABSTRACT

On-demand curation (ODC) tools like Paygo, KATARA, and Mimir allow users to defer expensive curation effort until it is necessary. In contrast to classical databases that do not respond to queries over potentially erroneous data, ODC systems instead answer with guesses or approximations. The quality and scope of these guesses may vary and it is critical that an ODC system be able to communicate this information to an end-user. The central contribution of this paper is a preliminary user study evaluating the cognitive burden and expressiveness of four representations of "attribute-level" uncertainty. The study shows (1) insignificant differences in time taken for users to interpret the four types of uncertainty tested, and (2) that different presentations of uncertainty change the way people interpret and react to data. Ultimately, we show that a set of UI design guidelines and best practices for conveying uncertainty will be necessary for ODC tools to be effective. This paper represents the first step towards establishing such guidelines.

1. INTRODUCTION

Historically, the quality of a dataset would be ensured before it was analyzed, often through complex, carefully developed curation processes designed to completely shield analysts from any and all uncertainty. This curation establishes trust in the data, which in turn helps to establish trust in the results of analyses. However, as typical data sizes and rates grow, this type of brute-force *upfront* curation process is becoming increasingly impractical. As a result, analysts have started turning to new, "on-demand" or "pay-as-you-go" approaches [1, 2, 7, 10, 12, 14] to data curation. On-demand curation (ODC) systems minimize the amount of upfront time and effort required to load, curate, and integrate data. Data stored in an ODC is, initially at least, of low quality and queries are liable to produce incomplete or incorrect results. To mitigate the unreliability of these results, ODC systems typically provide a form of provenance or lineage, tracking the effects of uncertainty through queries and tagging results with relevant quality metrics (e.g., confidence bounds, standard deviations, or probabilities). If the analyst finds the result quality insufficient, the ODC can help her to prioritize her data curation efforts.

Most ODC efforts are specialized forms of *probabilistic databases* [13] that allow for queries over uncertain, probabilistically defined data. Classical probabilistic databases produce outputs either in the form of "certain" answers (that provide only limited practical utility), or in the form of probability distributions. Representing a query output as a distribution alleviates the monotonous (and error-prone) task of handling probabilities, error conditions, and outliers in the middle of a query. Nevertheless, error-handling logic is still necessary, even if it is never expressly declared; A human interpreting the results must decide whether and how to act on the results given. Just having a probability distribution for query results is insufficient: *the uncertainty must be communicated to the users who will ultimately act on the results*. Complicating matters further is the fact that many database users lack the extensive background in statistics necessary to interpret complex probability distributions.

In this paper, we present our initial efforts to explore how probabilistic databases can communicate uncertainty about query results to their users. Fundamentally, we are interested in how the database should represent potential errors in tabular data being presented to the user. A representation that communicates too much information can create an unnecessary cognitive burden for users. Conversely, if a representation communicates too little, the user may not realize that data values are compromised and act on invalid information.

To explore this tradeoff between imposed cognitive burden and efficacy, we conducted a preliminary user study with 14 participants drawn from the Department of Computer Science and Engineering at the University at Buffalo. We explored four different representations of one specific form of data uncertainty called attribute-level uncertainty. Our results show that the choice of how to communicate low-quality data has a substantial impact on how users react to that information. Responses to different representations ranged from a desire for more information, an efficient use of presented contextual details, and even included mild fear responses to the data being presented. Thus, we argue that the design of interface elements for representing uncertainty is a critical part of probabilistic databases, ODCs, and data quality research in general. Concretely, this paper makes the following contributions: (1) We outline a user study that

Product	Rating Source			Note
	Buybeast	Amazeo	Targe	
Samesung	4.5	3.0	3.5	
Magnetbox	2.5		3.0	
Mapple	5.0	3.5		Not a TV?

Figure 1: **Examples of uncertainty.**

explores four different presentations of attribute-level uncertainty. (2) We quantitatively analyze the tradeoff between cognitive burden and decision-making based on results from our study. (3) We qualitatively analyze the different representations' effects on study participants' thought processes.

2. BACKGROUND

A probabilistic database [13] $\langle \mathbb{D}, P \rangle$ is typically defined as a set of deterministic database instances $D \in \mathbb{D}$ that share a common schema, and a probability measure $P : \mathbb{D} \mapsto [0, 1]$ over this set. Under *possible worlds semantics*, a deterministic query Q may be evaluated on a probabilistic database by (conceptually) evaluating it simultaneously on all instances in $\mathbb{D}$, producing a set of relation instances:

$$Q(\mathbb{D}) = \{\ Q(D) \mid D \in \mathbb{D}\ \}$$

Note that these semantics also induce a probability measure over the set of possible query results as a marginal of P computed over the result set.

Numerous semi-automated tools for curating low-quality data [1, 2, 14, 11] emit probabilistic database relations. These relations model the ambiguity that arises during automated data curation, most frequently appearing in one of three forms: (1) Row-level uncertainty, (2) Attribute-level uncertainty, and (3) Open-world uncertainty. Row-level uncertainty arises when a specific tuple's membership in a relation is unknown. Attribute-level uncertainty arises when specific values in the database are not known precisely. Finally, open-world uncertainty arises when a relation can not be bounded to a finite set of possible tuples.

Example 1. *The example spreadsheet given in Figure 1 shows reviews for 3 fictional television products from 3 fictional sources. Each of the three types of uncertainty are illustrated: It is unclear whether the Mapple is actually a television (row-level uncertainty). There are ratings missing for both the Magnetbox and the Mapple (attribute-level uncertainty). Finally, there is the possibility that the spreadsheet is incomplete and there are television products missing (open-world uncertainty).*

Several mechanisms for presenting probability distributions to end-users have been proposed. A common approach is to present only so-called "certain" answers [3] — the subset of the output relation with no row- or attribute-level uncertainty. Although computing certain answers presents a computationally interesting challenge, completely excluding low-quality results significantly decreases the utility of the entire result set. Another common approach is to compute statistical metrics like expectations or variances for attribute-level uncertainty, and per-row probabilities (confidences) for row-level uncertainty. Presenting this information to users in a way that can be clearly distinguished from deterministic data is challenging. Thus, systems like MayBMS [5] and MCDB [6] typically require users to explicitly request specific statistical metrics as part of queries. The mental overhead of manually tracking which attributes of a dataset are uncertain is an unnecessary burden on users; In multiple efforts where we have attempted to deploy probabilistic databases in practice [9, 14, 11], manual management of uncertain data has proven to be a non-starter.

Uncertainty also arises in other contexts. For example, Online Aggregation [4] uses sampling to approximate and incrementally refine results for aggregate queries. The user interface explicitly gives an expectation, confidence bounds, and % completion, clearly communicating that the result is an approximation, and the level of quality a user can expect from it. A second example, Jigsaw [9] simulates what-if scenarios, producing graphs that illustrate possible outcomes over time. Uncertainty is presented visually, with error bars and secondary lines used to show standard-deviations. Wrangler [8] helps users to visualize errors in data: A "data quality" bar communicates the fraction of data in each column that conforms to the column's type and the number of blank records. Finally, the Mimir system [14, 11] uses automatic data curation operators that tag curated records with markers that persist through queries. These markers manifest as highlights that communicate the presence of attribute and row-level uncertainty. Users click on fields or rows to learn more about why the value/row is uncertain.

3. EXPERIMENTAL DESIGN

The experiment consisted of a ranking task where participants were presented with a web form that had a 3x3 matrix showing three ratings each for three products. Participants were told that the ratings came from three different sources and were normalized to a scale of 1 to 5, with 5 being best and 1 being worst. Each participant was asked to evaluate the products for purchase by ranking the products in the order of their preference. A total of 14 participants, predominantly students in the Department of Computer Science and Engineering at the University at Buffalo, participated in the experiment.

To ensure a roughly predictable ordering from participants, ratings for each product were generated uniformly at random with the following constraints: Ratings for one of the three products (henceforth termed 'A') relative to a second product (termed 'B') had to include one extremely favorable comparison for A (one source gave A a rating at least 1 higher than B), one somewhat favorable comparison (one source gave A a rating at least as high as B but no more than 1 higher), and one disfavorable comparison (the final source gave A a rating worse than or equal to B but no more than 1 worse). Similar comparisons also had to hold between B and the final product (C). These constraints were designed to elicit a ranking of 'A', 'B', and 'C' from participants deciding based on either majority vote or based on the average of the three ratings.

Participants were asked to complete either one or five rounds of survey, with each round consisting of four trials. A single trial consisted of a single ranking task. The first **Certain** trial in each round served as a control: The matrix shown was generated exactly as described above. The remaining trials in each round each evaluated a single representation of uncertainty. In these trials, base data generation followed an identical process. However, in each trial, one of the following representations of uncertainty was used to annotate a small number (2-4) of product rating values. (1) **Asterisk**: Some ratings were marked with an asterisk

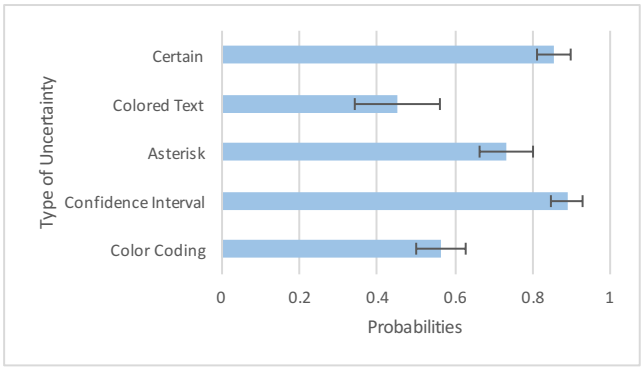


Figure 2: **Probability of the user's selection agreeing with the BestOf3 ranking.**

(e.g., `4.5*`) and participants were informed that these values were uncertain. (2) **Colored text**: The text of some ratings was colored red (e.g., 4.5) and participants were informed that these fields were uncertain. (3) **Confidence interval**: Some ratings were annotated with $\pm X$ where $X \in [0.5, 1.5]$ (e.g., `4.5 +/- 0.5`) and participants were informed that the value for those fields could range over the indicated interval. (4) **Color coding**: The cells containing some ratings were given a red background (e.g., `4.5`) and participants were informed that these fields were uncertain.

Interactions with the web-form — such as product selection, re-ordering the product list, and submitting the participant's final order — were logged along with timestamps. In addition to interactions with the web form, the experiment also used a think-aloud protocol: Participants were asked to verbalize their thought process while performing the task. Audio logs were transcribed and the anonymized transcriptions were tagged and coded for analysis.

4. EFFICIENCY AND EFFECTIVENESS

The two primary questions that we sought to answer for each of the four representations of uncertainty were (1) Is the representation *effective* at communicating uncertainty, and (2) What is the *cognitive burden* of interpreting the representation? Concretely, we identified at least three distinct behavioral responses to uncertainty in the data presented, suggesting differences in the efficacy of each representation. We also noted that all four representations of uncertainty required a similar amount of decision time, suggesting that all four representations impose similar cognitive burdens.

Effectiveness. The data presented to users was carefully selected to have two properties: First, each dataset was selected to elicit a specific ordering, regardless of whether participants made their choice based on the best two ratings or based on the average of all three ratings. We term this ranking order **BestOf3**. Second, uncertainty annotations were applied to specific cells of the table specifically to create ambiguity. As a consequence, we would expect users who chose to disregard uncertain data entirely to pick orderings effectively at random relative to **BestOf3**.

In short, if a representation of uncertainty is effective at communicating uncertainty, we would expect to see a more random product ranking. In the confidence interval representation — where bounds were not wide enough to prompt a significant level of ambiguity — we would expect to see ranking close to **BestOf3**.

Figure 2 summarizes our results, showing the probability of agreement between the participant-selected ordering and the **BestOf3** ordering. Standard deviations are computed under the assumption that agreement with **BestOf3** follows a Beta-Bernoulli distribution. A 16.7% agreement would indicate a purely random ordering. The 'certain', deterministic baseline shows a consistent, roughly 85% agreement with **BestOf3**, and as predicted, so does the confidence interval presentation (89%). Both colored text and color coding significantly altered participant behavior (45% and 56% agreement with **BestOf3**). Asterisks were not as effective at altering participant behavior (73% agreement). This is consistent with colored text and color coding signaling significant errors, while asterisks signal caveats or minor considerations on the values presented.

Efficiency. We measure time taken for each form of uncertainty as a proxy for cognitive burden. Figure 3 illustrates time taken by users to complete each individual ranking task. We distinguish between the first round, where participants initially encounter the task and representation, from subsequent rounds where they are already familiar with the task. As seen in Figure 3a, participants spent significantly more time familiarizing themselves with the overall ranking task than with any of the specific representations of uncertainty. Furthermore, time taken per representation was relatively consistent across all forms of uncertainty; The slowest two in Figure 3b trials were both deterministic.

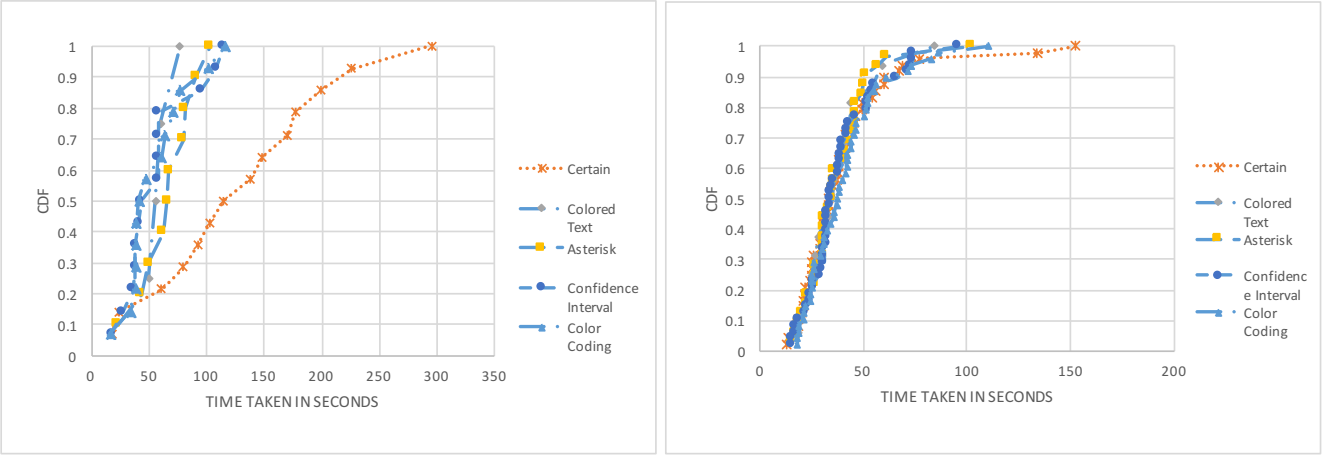
5. DISCUSSION

Participants were encouraged to verbalize their thought process. Based on this feedback, we were also also able to make several qualitative observations. In general participants considered consistency in the rating sources and products as a secondary source of feedback about data quality. For example, if Source 1 had low ratings for all three products, then some participants were more likely to discard it as uninformative and base their rating solely on the other two sources. If the range of ratings for a product was wide (4.5, 2, 1) then the product was considered unreliable by a few participants. Most of the participants explicitly stated that they were choosing based on the best two of, or the average of the three ratings.

Approximately half of the participants conveyed a strong negative emotional reaction to the color coding representation. Reactions ranged from participants who expressed a feeling of negative surprise on first seeing the value to participants indicating that the red boxes made them scared. By comparison, several participants suggested feelings of comfort associated with the additional information that the confidence interval supplied.

In addition to strong negative emotional responses, most participants indicated that they were ignoring values with a red background, except as a tiebreaker. This was true even for several participants who did not react in the same way toward the red text or asterisk representations.

Most participants exhibited risk-averse behavior. Given two similar choices, many participants stated a preference for products with more consistency in their ratings, as well as for products that did not include uncertain ratings. A frequent exception to this pattern was cases where uncertain



(a) First Round

(b) Second to Fifth Rounds

Figure 3: **Time taken per form of uncertainty. Graphs show cumulative distributions per-trial.**

values appeared at the low end of the rating spectrum — several participants indicated that the true value of a low, uncertain rating could only be greater than the value being shown.

In several instances, participants requested additional information, most frequently with the asterisk representation. It is possible that this is an artifact of the experimental protocol; The asterisk was the first form of uncertainty that many participants encountered. However, based on our efficacy analysis, it may also be the case that participants assumed that this representation signaled less significant errors. In future trials, we will use a random trial order and evaluate whether some representations are better at prompting users to seek out additional information.

For confidence bounds, users appeared to react to the presented uncertainty in one of two ways. One group appeared to first evaluate whether the uncertainty would make a significant impact on their deterministic ranking strategy (best 2 of 3 or average). The other group adopted a pessimistic view and plugged the lower bound into their deterministic strategy as a worst-case. For the experimental protocol used, both strategies typically resulted in the same outcome.

6. CONCLUSIONS AND FUTURE WORK

Data quality is becoming an increasingly painful challenge to scale. As a result of issues ranging from low-quality source data [8, 14, 11] to time-constrained execution [4, 9], the future is clear: Before long, imprecise database query results will be common. It is thus imperative that we learn how to communicate uncertainty in results effectively and efficiently. We presented our initial exploration of this space: a user study that examined four approaches to presenting attribute-level uncertainty. We plan to continue these efforts by exploring (1) other types of uncertainty in relational data (row-level and open-world), (2) qualitative feedback such as explanations [14], (3) giving the user mechanisms to dynamically control the level and complexity of uncertainty representation being shown, and (4) incorporating our findings into the Mimir on-demand curation system [14, 11].

Acknowledgements *This work was supported in part by a gift from Oracle. Opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of Oracle.*

7. REFERENCES

[1] G. Beskales, I. F. Ilyas, L. Golab, and A. Galiullin. Sampling from repairs of conditional functional dependency violations. *VLDBJ*, 23(1):103–128, 2014.

[2] M. Dallachiesa, A. Ebaid, A. Eldawy, A. Elmagarmid, I. F. Ilyas, M. Ouzzani, and N. Tang. NADEEF: A commodity data cleaning system. In *SIGMOD*, 2013.

[3] R. Fagin, P. G. Kolaitis, and L. Popa. Data exchange: Getting to the core. *TODS*, 30(1):174–210, Mar. 2005.

[4] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. *SIGMOD Rec.*, 26(2):171–182, June 1997.

[5] J. Huang, L. Antova, C. Koch, and D. Olteanu. MayBMS: A probabilistic database management system. In *SIGMOD*, 2009.

[6] R. Jampani, F. Xu, M. Wu, L. L. Perez, C. Jermaine, and P. J. Haas. MCDB: a monte carlo approach to managing uncertain data. In *SIGMOD*, 2008.

[7] S. R. Jeffery, M. J. Franklin, and A. Y. Halevy. Pay-as-you-go user feedback for dataspace systems. In *SIGMOD*.

[8] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. Wrangler: Interactive visual specification of data transformation scripts. In *CHI*, 2011.

[9] O. A. Kennedy and S. Nath. Jigsaw: Efficient optimization over uncertain enterprise data. In *SIGMOD*, 2011.

[10] Z. Khayyat, I. F. Ilyas, A. Jindal, S. Madden, M. Ouzzani, P. Papotti, J.-A. Quiané-Ruiz, N. Tang, and S. Yin. BigDansing: A system for big data cleansing. In *SIGMOD*, 2015.

[11] A. Nandi, Y. Yang, O. Kennedy, B. Glavic, R. Fehling, Z. H. Liu, and D. Gawlick. Mimir: Bringing CTables into practice. *CoRR*, abs/1601.00073, 2016.

[12] S. Roy, L. Orr, and D. Suciu. Explaining query answers with explanation-ready databases. *pVLDB.*, 9(4):348–359, Dec. 2015.

[13] D. Suciu, D. Olteanu, C. Ré, and C. Koch. Probabilistic databases. *Synthesis Lectures on Data Management*, 3(2):1–180, 2011.

[14] Y. Yang, N. Meneghetti, R. Fehling, Z. H. Liu, and O. Kennedy. Lenses: An on-demand approach to ETL. *pVLDB*, 8(12):1578–1589, Aug. 2015.

Data Cleaning in the Wild: Reusable Curation Idioms from a Multi-Year SQL Workload

Shrainik Jain, Bill Howe
Computer Science and Engineering Department,
University of Washington,
Seattle, WA, USA
{shrainik, billhowe}@cs.washington.edu

ABSTRACT

In this work-in-progress paper, we extract a set of curation idioms from a five-year corpus of hand-written SQL queries collected from a Database-as-a-Service platform called SQLShare. The idioms we discover in the corpus include structural manipulation tasks (*e.g.*, vertical and horizontal recomposition), schema manipulation tasks (e.g., column renaming and reordering), and value manipulation tasks (e.g., manual type coercion, null standardization, and arithmetic transformations). These idioms suggest that users find SQL to be an appropriate language for certain data curation tasks, but we find that applying these idioms in practice is sufficiently awkward to motivate a set of new services to help automate cleaning and curation tasks. We present these idioms, the workload from which they were derived, and the features they motivate in SQL to help automate tasks. Looking ahead, we describe a generalized idiom recommendation service that can automatically apply appropriate transformations, including cleaning and curation, on data ingest.

1. INTRODUCTION

Data curation is increasingly recognized as the bottleneck to analytics. Researchers and practitioners report spending a high proportion of their time cleaning, restructuring, transforming or otherwise preparing data for analysis. Worse, the time and effort spent on these "janitorial" tasks are difficult to amortize over repeated analysis projects; requirements tend to vary widely from project to project.

Classical approaches to data integration are relevant to curation, but tend to emphasize the design of a mediated schema to subsume two or more existing schemas. Data warehouse cubes are also associated with significant up front design and engineering of a centralized schema and the ETL workloads to fill it. These heavyweight "once and for all" approaches are a poor fit in data science contexts, where small teams of analysts convert data into actionable insights in more or less real time, drawing together multiple sources to answer targeted questions using specialized methods. Recent systems aim to reduce the effort required during the data curation step (*e.g.*, format-busting and data profiling with Data Wrangler [1], enterprise integration with Tamr [18]), but scripts-and-files approaches are still dominant among data scientists: there is no time to amortize the up-front cost of warehouse design or global-as-view/local-as-view data integration exercises, and moreover, the data is rarely being extracted from a carefully engineered schema on which these methods tend to rely.

But the cost of this over-reliance on scripts and files is high: Our collaborators in the sciences report spending up to 90% of their time manipulating data [10], consistent with other anecdotal reports of the balance of time between data curation versus data analysis.

To reduce the burden of data janitorial work and improve reuse, we posit that databases can be naturally extended to support the entire data lifecycle, including preliminary data cleaning and curation from untrusted sources typically handled outside the database. That is, we argue that databases should be designed to *encourage* ingestion of dirty, weakly structured data (i.e., rows-and-columns but no engineered schema), and that curation should be performed directly in the database by writing SQL. In this paper, we provide evidence of how this approach of letting databases do the curation via SQL queries, actually worked in practice by finding and characterizing *'cleaning idioms'* in a multi-year query workload.

We see multiple benefits to letting databases do this heavy lifting: i) the data always resides at one place during the entire analysis lifecycle (Figure 1), ii) the cleanup steps become more scalable, reliable, and reusable, and iii) the raw data is always directly available for reprocessing and recleaning in new contexts.

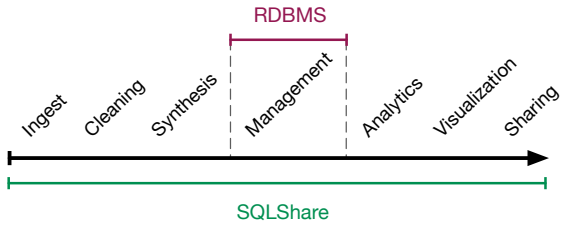


Figure 1: We find relational databases to be relevant at all stages in the scientific data lifecyle. SQLShare, a cloud-hosted database, empowers novice users by providing a system which handles use-cases across the data lifecyle.

The primary disadvantage of this approach is the SQL authorship: many common curation tasks, while expressible in SQL, are sufficiently awkward as to prevent uptake. Our hypothesis was that direct support for a set of common SQL data curation idioms can make SQL-based curation competitive with script-based curation.

To understand data curation tasks in practice, we analyzed the workload of the SQLShare system [12, 11, 10], a Database-as-a-Service system targeting scientists and engineers. SQLShare en-

courages users to upload uncurated datasets over the web "as is," write queries across any datasets in the system, and share the results as views. The goal is to reduce the overhead in using relational databases in ad hoc analytics scenarios by reducing or eliminating upfront costs associated with installation, configuration, schema design, tuning, and ingestion. SQLShare supports automated schema inference and tolerates dirty data; these features allowed users to switch from managing and sharing brittle, dynamic sequences of scripts to a single system where all the operations can be performed safely, reliably, and scalably.

Over the years, we collected the query logs on SQLShare and analyzed interesting use cases. One common use case, as we expected, was that of data cleanup and curation tasks. We show in this work how we can use these query logs to identify common cleanup tasks and provide them as suggestions for newer dataset uploads.

Furthermore, we envision how these cleanup *idioms* can be used to inform design of newer databases as follows:

1. Identify the clean up task from the query logs (hint: these are often the very first tasks performed on a dataset)
2. Generate templatized idioms for these cleanup tasks.
3. Upon newer dataset uploads, identify which idioms can be applied to the datasets.
4. Synthesize a clean up query from the selected idiom.

In this paper, we identify common curation patterns that appeared prominently in the SQLShare workload, we describe how they are used in practice, and how these patterns informed specific features in SQLShare to assist in data ingest and query authoring. Finally, we describe some ongoing work in semi-automatic data curation based on these idioms.

2. SQLSHARE WORKLOAD

SQLShare is a Database-as-a-Service system targeting scientists and engineers. Users upload datasets through a web interface as-is with no explicit schema, write queries across any datasets in the system, and share the results as views. The goal is to reduce the overhead in using relational databases in ad hoc analytics scenarios: installation, configuration, schema design, tuning, and data ingestion. Queries are submitted through a web interface, allowing collaborative query authoring and avoiding any software installation. The SQLShare interface facilitates and encourages the liberal use of views. Users frequently create deeply nested hierarchies of views to break down complex problems, clean and share intermediate datasets, and record provenance for complex results. SQLShare has been deployed in a number of scientific contexts and has proven to be useful even among SQL first-timers. Howe *et al.* describe the SQLShare architecture and motivation in detail in [9, 11, 12]. [10, 11] describes an in depth use cases of SQLShare view relational view sharing. SQLShare aims at reducing data management overhead in all stages of the data lifecycle shown in Figure 1. While there is no built-in support for visualization, Key *et al.* showed how the SQLShare can be extended to afford automatic visualization [14].

Analysis of this workload [3] showed that users were frequently expressing data curation and cleaning tasks directly in SQL. Since SQLShare encouraged users to upload datasets as is, a significant number of queries submitted to the system were intended to reshape the data, rename columns, remove errant values, and implement other data curation tasks. In this paper, we identify the common data cleaning idioms present in the SQLShare workload and consider how they can be generalized and applied automatically to incoming data.

Listing 1 Example queries for each idiom.

```
Vertical recompositioning:
"SELECT * from [gbc3].[sqlshare-exp.txt]
UNION ALL
SELECT * from [gbc3].[gen_sqlshare.txt]"
Horizontal recompositioning:
"SELECT * FROM [che].[m1]
FULL OUTER JOIN [che].[m3]
ON
[che].[m1].m1_loci_id=[che].[m3].m3_loci_id"
Column rename:
"SELECT column2 as sp, column3 as SPID,
column4 as Prot FROM
[userX].[uniprotolyblastx2.tab]"
NULL injection:
"SELECT CASE WHEN [400 avg NSAF] = 0
THEN NULL
ELSE [2800 avg NSAF]/[400 avg NSAF] END
FROM
[emma].[NSAFwithAve]"
```

Table 1: Frequency of observed idioms (total datasets: 4535)

Idiom	Datasets
Vertical recompositioning	100
Horizontal recompositioning	210
Column rename	720
NULL injection	420

3. CURATING IDIOMS

The ubiquity of weakly structured data in the science use cases required SQLShare to tolerate (and even embrace) upload of weakly structured data. SQLShare encourages users to write SQL queries to repair and reorganize data rather than relying on offline scripts. By mining the workload, we extract generalizable patterns used to perform these repairs and use them to design services to partially automate cleaning tasks.

The SQLShare query corpus presents rich evidence to support this hypothesis. By searching the corpus of 4535 derived datasets (views), we found specific SQL idioms that correspond to schematization tasks: cleaning, typecasting, and integration.

We focus on the following curation patterns extracted from the SQLShare logs. Along with each idiom, we present an example query from the logs in Listing 1, and a method for using the idiom to support curation-on-ingest. Table 1 shows the frequency of occurrences of these idioms.

- *Vertical recompositioning*: Datasets in SQLShare are often representative of scientific processes where one logical dataset arrives in the form of several distinct files arriving at different times. For example, one lab collected data daily from a sensor depoyed in a local estuary. The need to pre-establish a schema and load the data file-by-file makes databases unattractive in these contexts, but SQLShare helped eliminate steps during data ingest. However, users still needed to craft a UNION ALL query to assemble the results, sometimes reordering columns or casting types to align the derived schemas.

  Curation on ingest: By learning these schema alignment heuristics automatically from the data, and applying schema

matching methods, these UNION ALL queries can be automatically recommended and applied by the system upon data ingest. One such approach was describe in our previous work on automatically deriving example queries from base data [8].

- *Horizontal recompositioning*: This idiom pertains to horizontally partitioned datasets. As with vertical recompositioning, scientific processes generating the data sometimes produce horizontally split data. Sometimes different labs working on same samples generate different attributes about them. These cases appear in the logs as multiway 1:1 joins.

  Curation on ingest: Suggesting queries for horizontal recompositioning can be non-trivial. However, we can again use the approach shown in [8] to find potential for joins automatically. Automatic join finding using measures like jaccard similarity has been done in the past, combining this approach with a rich hand written query log to suggest data curation idioms is something that can finally make such approaches viable.

- *Column renaming*: It is common for datasets in SQLShare to have no column names in the source files. For this user scenario, SQLShare assigns default column names. Users are encouraged to write SQL to assign semantic names. We find evidence of 1996 uploaded tables, which is about 50% of the total tables, that had at least one default-assigned column name. The number of datasets for which all columns were assigned the default value is 1691. Almost 16% of datasets involve some kind of column renaming step, suggesting that users have adopted SQL as a tool for adding semantics to their data. We find this as sufficient evidence to back our hypothesis that the SQLShare workload contains a rich set of cleanup and curation queries.

  Curation on ingest: While identifying potential columns to rename is easy (columns with the default names are obvious candidates to begin, with a few false positives), suggesting valid renames is a very ambiguous problem. However, since we do have the advantage of having the previous tables and queries written on them. One approach could be to match the range of values of the column to rename to the range of values to previously existing and renamed columns. For example, for an attribute whose range is 0 to 360 and renamed to *'Angle'*, it might make sense to suggest for columns with values in the same domain. Another possible way could to be calculate the earth mover distance [17] between the histograms of column values and suggest rename to column with which this distance is least. There are other principled approaches as in WebTables [7] which uses the *attribute correlation statistics* to suggest schema auto-complete.

- *NULL injection and Type Coercion*: Sentinel values are routinely used to mark missing or inapplicable data; we see string values of "N/A" for example embedded in an otherwise numeric column. The SQL authors can use assemblies of CASE WHEN expressions, filtering, and type casting to replace these values with NULL or otherwise repair them. These constructs are conceptually trivial ("Across all columns, replace the value X with NULL") but hand-writing the corresponding query is tedious and error-prone. After removing bad tuples and replacing missing values with NULL, we find that about 200 of derived datasets used SQL CAST to introduce new types on existing columns.

  Curation on ingest: Our current implementation automatically infers data types based on a prefix of rows, and creates two table. The first table corresponds to the predicted type, and the second table holds non-conforming rows and has every column typed as a string. Finally, a view is created to union the 2 tables and is presented to the user, along with the information about the 2 base tables. This process helps separate the numeric data from the sentinel values, but does not automatically apply the CASE expressions.

3.1 Towards Idiom-Based Data Curation

So far we have shown the evidence of curation via SQL queries in the SQLShare workload. We discussed how these queries can be characterized into common curation idioms and finally we detailed the potential algorithms for curation on ingest.

Tying it all together, the idiom recommendation algorithm would work as follows:

- Identify the common curation idioms, the very first queries on a dataset are often representative of these idioms.
- Generate a query template for each idiom as shown in Qunits [16] and also in SQLShare analysis [11].
- At the time of data ingest, we use the **curation on ingest** techniques in section §3 to identify possible idioms.
- Synthesize the curation queries from the templates and provide them as suggestion to the users. The query synthesis problem has already been solved with multiple examples already available in the literature [6, 4, 5].

This approach of suggesting queries at ingest can save a lot of user time because writing these queries by hand can be repetitive and time consuming. The false positives don't hurt a lot because the user is always in loop and chooses which curation idiom, if any, she wants to apply to her dataset.

In our current implementation, we have a working analysis pipeline [11] and idiom detection. The next steps include integrating this pipeline with the SQLShare system and implementing a query synthesis algorithm. We are actively working on a demo system and hope to present in the immediate future.

4. RELATED WORK

Parsing of complex formats (messy data) to produce weakly structured data for further processing is a problem that has been approached previously, OpenRefine [2] and Wrangler [1] being popular examples of this approach. These tools do not offer support working with multiple datasets and have been shown to have dominant costs [13].

SnipSuggest [15] is an example of system which provides auto completion of queries and has been shown to enable non-experts to write complex SQL. Our work has similar aims, but our approach is to suggest complete queries, automatically synthesized based on previous queries.

WebTables [7] is another work in similar domain, but the focus is to provide automatic schema completion for myriad of documents of the web. We approach suggests a possible use of their schema completion algorithm, but goes beyond just schema completion and provides a richer set of curation idioms.

Akbarnejad *et al.* [5] used a similar approach, *i.e.* using history and preferences to recommend queries. We hope to extend their work and use it an a setting where user is automatically suggested queries on ingest, *i.e.*, the required interaction is minimal, while the queries are still relevant. However, we have one critical advantage in our proposed system, a workload with **real handwritten** queries.

One of our previous works [8] presents the notion of suggesting automatic starter queries, or queries by example, aimed at providing novice users with example and ease the ramping up process. This paper has a similar goal, but our approach has one major difference in that we learn the idioms we are suggesting from the query logs, these idioms are proven to be useful to users, since they have already used them and have the potential to suggest very complex queries (something which the previous approach lacked).

5. CONCLUSION AND FUTURE WORK

We presented a work in progress which uses handwritten queries from a five-year corpus of Database-as-a-Service platform called SQLShare to identify data clean up queries written in SQL. The design choices in SQLShare enabled the users to use databases all stages in the scientific data lifecycle. We present evidence of clean up and curation being done via SQL queries and discuss methods in which these query **idioms** can be used to suggest curation queries to users at the time of data ingest. We talked about the analysis of SQLShare workload and how we mined the queries related to cleaning and curation. We also identified some commonly used idioms which in our opinion should be better supported in databases. Currently we have set up workload analysis pipeline as shown in [11] and have a naive way to find out possible curation queries. Since the clean up queries are usually the very first queries on a dataset, our current methods looks for common idioms amongst these. Our immediate next plan is to extend this work to:

- Incorporate an idiom recommendation process into SQLShare
- Identify other query idioms for scientific use cases. This can be done by clustering embeddings for queries in a higher dimensional space and associating idioms with these clusters.

Our final vision is to have a system which takes in a dataset (plus JBOTs) and suggests curation & scientific analysis queries that can run on it.

6. REFERENCES

[1] Data wrangler. http://vis.stanford.edu/wrangler/.

[2] OpenRefine (formerly google refine). http://openrefine.org/.

[3] Sqlshare workload data release 1. https://uwescience.github.io/sqlshare/data_release.html.

[4] S. Abdul Khalek and S. Khurshid. Automated sql query generation for systematic testing of database engines. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, ASE '10, pages 329–332, New York, NY, USA, 2010. ACM.

[5] J. Akbarnejad, G. Chatzopoulou, M. Eirinaki, S. Koshy, S. Mittal, D. On, N. Polyzotis, and J. S. V. Varman. Sql querie recommendations. *Proceedings of the VLDB Endowment*, 3(1-2):1597–1600, 2010.

[6] N. Bruno, S. Chaudhuri, and D. Thomas. Generating queries with cardinality constraints for dbms testing. *Knowledge and Data Engineering, IEEE Transactions on*, 18(12):1721–1725, 2006.

[7] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. Webtables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment*, 1(1):538–549, 2008.

[8] B. Howe, G. Cole, N. Khoussainova, and L. Battle. Automatic example queries for ad hoc databases. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 1319–1322. ACM, 2011.

[9] B. Howe, G. Cole, E. Souroush, P. Koutris, A. Key, N. Khoussainova, and L. Battle. Database-as-a-service for long-tail science. In *Scientific and Statistical Database Management*, pages 480–489. Springer, 2011.

[10] B. Howe, F. Ribalet, D. Halperin, S. Chitnis, and E. V. Armbrust. Sqlshare: Scientific workflow via relational view sharing. *Computing in Science & Engineering, Special Issue on Science Data Management*, 15(2), 2013.

[11] S. Jain, D. Moritz, D. Halperin, B. Howe, and E. Lazowska. Sqlshare: Results from a multi-year sql-as-a-service experiment. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pages 281–293, New York, NY, USA, 2016. ACM.

[12] S. Jain, D. Moritz, and B. Howe. High variety cloud databases. In *Proceedings of the 2016 IEEE Cloud Data Management Workshop.*, 2016.

[13] S. Kandel, A. Paepcke, J. Hellerstein, and J. Heer. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3363–3372. ACM, 2011.

[14] A. Key, B. Howe, D. Perry, and C. Aragon. Vizdeck: self-organizing dashboards for visual analytics. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 681–684. ACM, 2012.

[15] N. Khoussainova, Y. Kwon, M. Balazinska, and D. Suciu. Snipsuggest: Context-aware autocompletion for sql. *Proceedings of the VLDB Endowment*, 4(1):22–33, 2010.

[16] A. Nandi and H. Jagadish. Qunits: queried units in database search. *arXiv preprint arXiv:0909.1765*, 2009.

[17] Y. Rubner, C. Tomasi, and L. J. Guibas. A metric for distributions with applications to image databases. In *Computer Vision, 1998. Sixth International Conference on*, pages 59–66. IEEE, 1998.

[18] M. Stonebraker, D. Bruckner, I. F. Ilyas, G. Beskales, M. Cherniack, S. B. Zdonik, A. Pagan, and S. Xu. Data curation at scale: The data tamer system. In *CIDR*, 2013.

Data Quality for Semantic Interoperable Electronic Health Records

Shivani Batra
Jaypee Institute of Information Technology University, Sector-128, Noida, India
ms.shivani.batra@gmail.com

Shelly Sachdeva
Jaypee Institute of Information Technology University, Sector-128, Noida, India
shelly.sachdeva@jiit.ac.in

ABSTRACT
The current study considers an example of healthcare domain from a BIG DATA perspective to address the issues related to data quality. Healthcare domain frequently demands for timely semantic exchange of data residing at disparate sources. It aids in providing support for remote medical care and reliable decision making. However, an efficient semantic exchange needs to address challenges such as, data misinterpretation, distinct definition and meaning of underlying medical concept and adoption of distinct schemas. The current research aims to provide an application framework that aids in syntactic, structural and semantic interoperability to resolve various issues related to semantic exchange of electronic health records data. It introduces a new generic schema which is capable of capturing any type of data without a need of modifying existing schema. Moreover, proposed schema handles sparse and heterogeneous data efficiently. The generic schema proposed is built on the top of relational database management system (RDBMS) to aid in providing high consistency and availability of data. For having a deep analysis of proposed schema considering timeliness parameter of data quality, experiments have been performed on two flavours of RDBMS namely row oriented (MySQL) and column oriented (MonetDB). Results achieved favours adoption of column oriented RDBMS over row oriented RDBMS under various tasks performed in current research for timely access of data stored in proposed generic schema.

Keywords: Data Interoperability, Data Quality, Electronic Health Records (EHRs), Generic schema, Column oriented RDBMS, Row oriented RDBMS.

1. INTRODUCTION
In healthcare domain, millions and billions of patient records are recorded on a daily basis. This leads to BIG DATA and thus, demands to handle 3V's (volume, velocity and variety) related to BIG DATA. Current study is focused on handling 3V's for semantic interoperable Electronic Health Records (EHRs). Semantic interoperable EHRs constitute medical data related to various patients that possess same meaning to all users using the same set of attributes. Considering large volume of EHRs, a major portion of storage space accounts for null values. So, handling sparseness can greatly reduce storage space requirements. Moreover, EHRs being related to life of patients, demands real time access to information. This requires various optimization techniques which will help in attaining high velocity. Another important aspect is variety since, EHRs constitutes ambiguous heterogeneous data. Apart from dealing with volume, velocity and variety parameters of BIG DATA, healthcare domain often demands for data interoperability. Frequent data access and semantic exchange among distant healthcare organizations aids in improving quality of care. However, this semantic exchange of data needs to address following challenges.

1. *Data misinterpretation:* An entity in medical domain may have different meaning to different organizations. For example, a hospital (APPOLO) may store body temperature data in degree Fahrenheit and other hospital (FORTIS) stores same data in degree Celsius as shown in Figure 1. When data of APPOLO and FORTIS are exchanged, one can misinterpret data regardless of the fact that both data were presenting a correct state of body temperature for some patient. For handling data misinterpretation, all organizations involved in data exchange must follow same data semantics, i.e., there should be semantic interoperability.

Hospital Name	Body Temperature
APPOLO	104
FORTIS	40

Figure 1. Body Temperature recording at two Hospitals

2. *Distinct set of attributes for same medical concept:* Various organizations have their own mechanism for recording data. For example, a medical concept named as 'Blood Pressure' consists of four types of pressures namely 'Systolic', 'Diastolic', 'Mean arterial' and 'Pulse pressure'. 'Systolic' and 'Diastolic' are two types of pressure recorded using medical instrument while 'Mean arterial' and 'Pulse pressure' stores a calculated value using 'Systolic' and 'Diastolic' data. APPOLO might depicts blood pressure condition (Low, Normal and High) of a patient based on 'Systolic' and 'Diastolic' pressure value. While, FORTIS performs same task of predicting blood pressure condition of a patient based on 'Mean arterial' pressure value. This again leads to difficulty in semantic exchange of data. To have same set of attributes there should be a mechanism that provides structural interoperability. Moreover, each attribute should depict the same set of constraints such as, data type, i.e., system must be syntactic interoperable.

3. *Distinct local schemas*: Every organization customizes his schema to capture the data based on local requirements. For example, APPOLO stores value of 'Systolic' and 'Diastolic' pressure and not 'Mean arterial' pressure. Whereas, FORTIS reserves a column in local database table for 'Mean arterial' pressure and not for 'Systolic' and 'Diastolic' pressure. Thus, semantic exchange requires to perform an operation such as, JOIN to construct a maximal schema that can capture each and every detail. However, this approach is not suitable since it require changes in existing database schema.
4. *Timeliness*: In an emergency situation, extracting patient's medical history from local database without any time delay is crucial. Absence of right data at right time can adversely affect treatment given to the patient.

One solution to avoid data misinterpretation and follow same set of attributes for one medical concept is to adopt a standard based EHRs system. To resolve distinct local schema, all organizations need to commit to follow a common standard schema for data storage for smooth semantic exchange. Moreover, schema adopted should be rich enough in terms of capturing all existing and future data requirements without any restructuring of schema. Current research introduces a new generic schema (in Section 3.2) which helps in achieving schema interoperability. Also, proposed generic schema can also be easily expanded to provide data security to enhance data quality. Physical storage approach (row oriented or column oriented) adopted for the proposed generic schema affect the timely access of data. Experimentations has been done to account for this effect.

1.1 Key Contributions

Core aim of current research is to address issues (data misinterpretation, distinct set of attributes for same medical concept and distinct local schema) related to data interoperability. Key highlights of the work done in this paper are as follows:

1. A new generic persistence model.
2. Minimizing storage requirement by eliminating the need of storing null values.
3. Handling heterogeneous data.
4. Maintenance of various quality parameters such as, accuracy and validity, believability, reliability, security, completeness, accessibility, consistency and fitness for use.
5. Experimental evaluation of proposed generic storage on two variants of RDBMS viz. row oriented RDBMS (MySQL) and column oriented RDBMS (MonetDB) under various tasks performed to have an insight of timeliness as a data quality parameter.

This will help in building an EHR system which is capable of dealing large volume with high velocity and can store variety of data.

1.2 Organization of paper

The paper is divided into various sections. Section 2 describes layered approach used by various standard organizations to aid in standardized EHRs. Section 3 ushers the proposed approach and presents the way syntactic, structural and semantic interoperability is achieved. Moreover, it introduces a new generic persistence to handle sparseness and heterogeneous data. Section 4 explains various data quality parameters achieved in current research. Section 5 highlights the experimental setup and results attained considering timeliness parameter of data quality. Section 6 finally concludes the work done.

2. STANDARDIZED EHRs

Electronic Health Records (EHRs) have a complex structure that may include data from about 100-200 parameters [2][16] such as, temperature, blood pressure and body mass index. Individual parameters will have their own contents. Each contains an item, such as, 'data' (e.g., captured for a blood pressure observation). It offers complete knowledge about a clinical context, (i.e., attributes of data), 'state' (context for interpretation of data), and 'protocol' (information regarding gathering of data), as shown in Figure 2 (depicting completeness). Standardized EHRs aid in providing same context among all organization. For example, value of blood pressure recorded in sitting position might vary from the value recorded in standing position. Using standardized EHRs, state description can be captured to provide same context to data while exchanging.

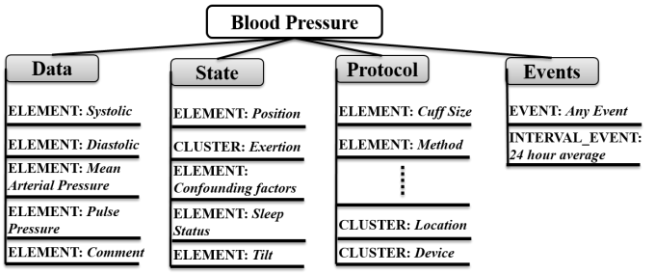


Figure 2. Blood Pressure as a concept (at document level)

Many standard organizations are making constant successful efforts to achieve standardization in healthcare domain for the purpose of semantic interoperability. Some of these famous organizations are Health Level 7 (HL7) [10], European Committee of standardization Technical committee 251 (CEN TC251) [6], International standard organization (ISO) [12] and openEHR [14]. These organizations adopt a layered approach for providing semantic interoperability among EHRs.

2.1 Layered Approach for Standardization

In past, single layer approach was used to design an application. Incorporating single layer hides segregation between programming and domain specific concepts, making an application difficult to modify. Thus, application needs to be rebuild from scratch to accommodate required amendments. For quality enhancement and reduction in repeated efforts for building an updated application, multi-layered approach was introduced.

Multi-layer methodology divides architecture of building an application in multiple layers. Each layer highlights issues related to one of various concepts related to developed application. Considering EHRs, multilevel model approach [1] is widely adopted, aiming for standardization. Standardization is key to achieving fitness for use i.e. communicating same interpretation of data to everyone as originally planned. Layered approach is presented in Figure 3. It divides the application architecture in

different layers termed as Reference Model (RM) [3], Archetype Model (AM) [5] and Service Model (SM) [4].

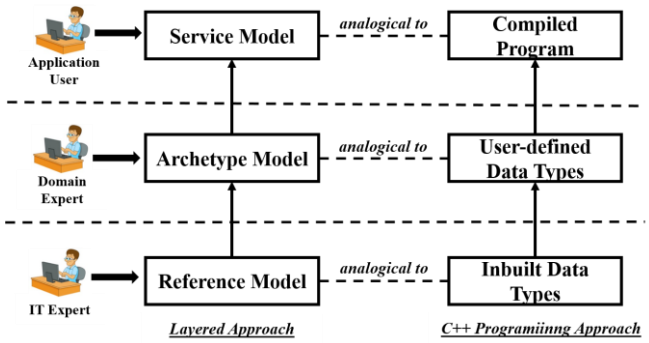


Figure 3. Analogy of layered approach to C++ programming framework

Layered approach can be well-understood through its analogy to the C++ object oriented programming approach. There are inbuilt data types such as, 'int', 'float' and 'char' that are already defined and the application programmer needs to make use of inbuilt data types to define structure or classes as per their needs. RM layer of layered approach depicts inbuilt data types and AM layer depicts user defined data types of C++ object oriented programming approach. SM layer deals with the overall application provided to end user similar to a compiled program in C++ environment.

RM in healthcare domain provides all technical information in terms of data types or data structures to be adopted by final application. This requires involvement of information technology (IT) expert for definitions of various aspects used in RM. RM is stable in nature, similar to inbuilt datatypes in C++ environment.

AM defines domain specific knowledge in form of small modules called archetypes [4]. All knowledge regarding a medical concept resides in an archetype. For example, Blood pressure archetype defines five data attributes termed as 'Systolic', 'Diastolic', 'Mean Arterial', 'Pulse pressure' and 'Comment' with their complete definition (as shown in Figure 2). Various aspects depicting completeness includes data type followed by attribute, domain of attribute, magnitude units in which attribute is defined and links to standard terminologies such as, SNOMED-CT [11] and LONIC [13]. Making use of standard terminologies aids in semantic interoperability by providing a common definition of various terms used in healthcare domain. At this level, domain expert utilizes the information provided in RM to define requirement specific aspects within an archetype. Once an archetype is defined it can be saved in archetype repository and, reused on demand. Healthcare is an expanding domain. As the domain knowledge expands, a new archetype is build using core classes defined in stable RM. Any modification in an archetype or building a new archetype will not require any change at RM or SM. For instance, incorporating any new user defined structure or class in C++ environment will not impact existing inbuilt data types and programs.

SM makes use of archetypes to deliver an application which can further be reused to build any number of user specific applications. For instance, any number of programs can be built in C++ environment using existing inbuilt and user defined data types.

Layered approach was initially proposed by openEHR. Later on other organizations such as, ISO 13606 and HL7 adopted dual model approach for standardization. Interoperability among standards help in providing ability to communicate between various applications built based on various standards. Numerous researches exist [8] that provides framework for switching between different standards. An organization that adopts openEHR, HL7 or ISO 13606 can easily migrate to any standard of choice using such frameworks. Thus, it is very easy for various organisations to adopt same standard (openEHR in our case). Current research focuses on openEHR for achieving data quality and experimentation.

3. SOLUTION APPROACH FOR SEMANTIC EXCHANGE OF EHRs

Semantic exchange of data in healthcare domain is highly demanded to enhance quality of care. The current study aims to resolve the challenges addressed in section 1 for a reliable semantic exchange of data.

1. Data misinterpretation is resolved by linking to standard medical terminologies such as, SNOMED-CT and LONIC. Archetype constraints metric (units) of each attribute that helps in overcoming any misinterpretation. Hence, adopting standard facilitates semantic interoperability.
2. Same set of attributes for same medical concept following same data semantics is achieved through the use of archetypes.
3. Problem of distinct local schema is handled via proposing a new generic schema. Schema adopted capture existing and future data requirements. Simultaneously, it handles sparse and heterogeneous data.

Current research proposes to use application built on archetypes defined by openEHR standard and to persist data in a generic persistence as shown in Figure 4. Each hospital may download any number of archetype from Clinical Knowledge Manager (CKM) [7] based on their local requirements and store them in a local archetype repository. Archetype repository of various hospitals can have distinct set of archetypes, exactly same set of archetypes or overlapping set of archetypes depending upon their local requirements. Based on local archetype repository, a customized clinical application is built for the corresponding hospital.

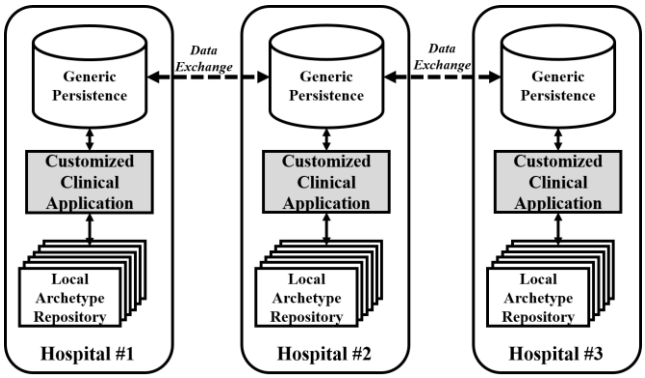


Figure 4. Proposed Approach

Data captured by local application is thus stored in a generic schema proposed in current research (Section 3.2). Generic schema allows migration of data from various organizations without making any changes at schema level. Moreover, proposed generic schema can be enhanced to incorporate security mechanism that in turn helps in enhancing data quality.

3.1 Handling Interoperability

As specified above, different organizations must adhere to same set of attributes for same medical concept. Current research proposes to use archetypes defined by openEHR for this purpose. Archetypes are agreed formal representation of a medical concept. It defines consensus on maximal representation of a medical concept. Archetypes are being defined through Archetype Definition Language (ADL) [5].

For creation and versioning (modifying an existing) of an archetype, a prototype process is followed that involves teams constituting various medical experts. After many review iterations, archetype is agreed to be published in a standard online library such as, CKM. Currently, archetypes available on CKM are followed by 87 countries [7]. Any organization that wishes to make use of an archetype can download it from any online library related to any standard. An archetype downloaded from a standard online library based on one standard can be easily transformed in other standard using tools such as, POSEACLE converter [8] using ontology-based archetype transformation process. POSEACLE converter provides online functionality to transform an ISO 13606 based archetype into an equivalent openEHR based archetype. Providing archetypes in a standard online library enables various organizations involved to easily download archetype anywhere and anytime for their local archetype repository.

As described in Section 2, archetypes are built on a stable structure defined RM. All archetypes related to a standard (openEHR in our case) will follow same RM. This aids in providing syntactic interoperability. Use of archetype provides same set of attributes for same medical concept irrespective of organization adopting it. This provides a mechanism for achieving structural interoperability. Moreover, archetypes are linked with standard terminologies such as, SNOMED-CT and LONIC that aids in providing common data semantics. Achieving common data semantics provides semantic interoperability and resolves issue of data misinterpretation. Thus, archetype provides business rules, ontology, terminology binding, F-logic, versioning mechanism, standardized data definition, content and structure, and language translation capability [17].

Considering all above mentioned features related to semantic exchange, the current research makes use of archetype to propose extension to EAV model.

3.2 Proposed Generic Schema

Various healthcare organizations store data in their own schema as per their local requirements. This creates issues while semantically exchanging data from various independent resources. To overcome this problem, current research proposes to use a generic persistence. Existing solutions for generic persistence in healthcare domains [9] recommend use of Entity-Attribute-Value (EAV) model [15]. EAV constitutes three columns named as Entity, Attribute and Value. One row in EAV corresponds to one attribute value of particular entity (as per relation table). EAV model depicts the same logical representation as relational table through metadata table which reserves information such as, name of all attributes. Use of metadata provides information regarding null values that are not stored in EAV model. Only non-null values are stored in EAV model to limit the storage wastage due to presence of sparse entries. EAV suffers from the issue of heterogeneity due to presence of single value column that constitutes data related to one data type only.

Current research proposes an extension to EAV model namely Archetype Entity Attribute Value (AEAV) to capture archetype based data. AEAV provides a generic persistence for archetype based applications that is more secure than EAV. Firstly, AEAV extends basic three column structure of EAV to four column structure for capturing archetype details also as shown in Figure 5.

Entity	Archetype_Name	Attribute_Name	Value_int
1	Blood Pressure	Systolic	105
1	Blood Pressure	Diastolic	85
2	Body Weight	Weight	70
3	Body Mass Index	Body Mass Index	40
2	Blood Pressure	Systolic	125
2	Blood Pressure	Diastolic	100
4	Body Weight	Weight	65

Entity	Archetype_Name	Attribute_Name	Value_string
2	Blood Pressure	Comment	High
1	Blood Pressure	Comment	Normal

Figure 5. EAV extended to store archetype details

To deal with heterogeneity, AEAV table is divided in multiple tables' segregated based on type of data in value column. Similar to EAV, AEAV also stores only non-null values. After defining the extended EAV storage structure, next step is to define numeric coding for archetype names and attribute names using a manually designed mapping dictionary as shown in Figure 6. Mapping dictionary also serve role of metadata table (as in EAV).

Mapping dictionary defined for AEAV has various advantages as follows:

1. *Improved storage:* Apart from storage enhancements achieved by not storing null values, AEAV optimizes space by not storing long names of archetypes and attributes. It reduces the storage space by replacing the need of storing archetype redundantly and attribute name textually to numeric codes that consumes less space.

2. *Improved searching speed:* Search efficiency in finding codes related to one archetype is high by making use of index structure.

3. *No prior knowledge:* Adding new archetypes in existing system require no prior knowledge about existing codes of attributes since each archetype reuses same set of codes. Combination of archetype name code and attribute name code will define a unique code that serves as a unique identifier in main table.

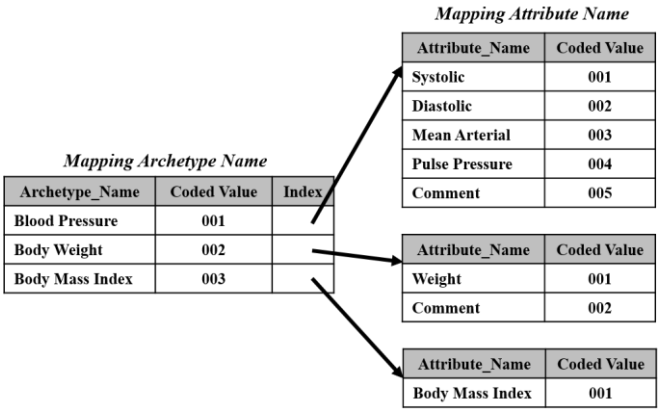


Figure 6. Mapping dictionary for archetype and attribute names

Once mapping dictionary is defined, Archetype_Name and Attribute_Name columns of extended EAV table are replaced by their corresponding codes. Finally, Archetype_Name and Attribute_Name columns are combined as one column named "ArchAtt" using steps as follows.

1. Convert numeric code of Archetype_Name into equivalent 8 bit binary code.
2. Append '00000000', i.e. eight 0 bits to the end of 8 bit Archetype_Name code to make it a 16 bit code.
3. Convert the 16 bit code into an equivalent decimal and replace existing Archetype_Name value with this new value.
4. Add decimal values of Archetype_Name and Attribute_Name columns and replace Archetype_Name and Attribute_Name columns are with one column named ArchAtt containing this summation value.

Final outcome of the above process on initial tables, i.e., AEAV model is shown in Figure 7.

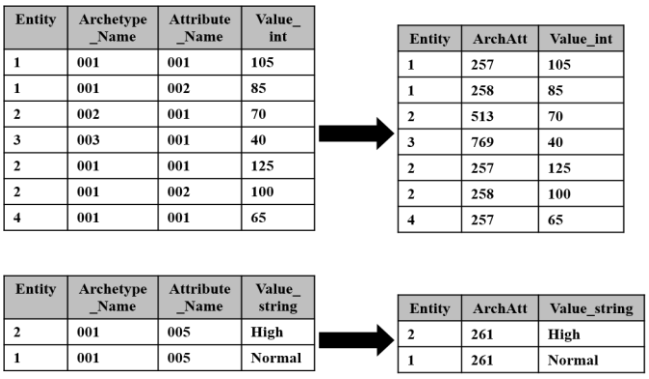


Figure 7. Archetype Entity Attribute Value (AEAV) model

Addition of coded 'ArchAtt' (of AEAV) in place of 'Attribute' column (of EAV) makes AEAV more secure than EAV. AEAV is meaningless until related coding mechanism is known and mapping dictionary are available. Thus, any attack on data will not be able to understand data in absence of mapping dictionary and algorithm followed to combine Archetype_Name and Attribute_Name in ArchAtt.

AEAV can be modified to enhance security feature. To accomplish this, 8 bit code can be replaced with an 'n' bit code in step 2 of coding algorithm. Different organizations can adopt different values of 'n'. This make coded values of one organization distinguished and insignificant to other organizations. Information regarding 'n' can be sent to the organization involved in data exchange through a well-defined secured data encryption mechanism.

Although EAV/AEAV eliminates the need to store null values, it does introduce an overhead of storing entity/attribute code. Thus, there is a trade-off between amount of sparseness and overhead introduced in EAV/AEAV. Larger the amount of sparseness, lower will be the overhead. Hence, EAV/AEAV should be preferred for domain constituting huge volume of null values. Healthcare is one such domain and thus, AEAV is suitable to be adopted for EHRs. Moreover, EHRs are very crucial in terms of ethical and legal issues related to it. This demands for a secure transfer of information. AEAV is step towards the secure transfer of EHRs data.

So far, in current research various challenges identified for semantic exchange of EHRs and solution proposed are summarized in Table 1.

4. DATA QUALITY CONSIDERATIONS

In addition to various parameters (Accuracy and validity, believability, reliability, security, accessibility, completeness, and consistency) of data quality identified in [16], authors in current research commit to achieve fitness for use as a data quality parameter while semantic exchange of data.

1. *Accuracy and Validity:* Use of archetype entitles solution for accurate and valid data. Archetype constitute various business rules to precise data domain and mathematical logics. Applications built on top of archetypes must adhere to these business rules and thus, achieves accuracy and validity.
2. *Believability:* Involvement of domain expert at AM level ensures believability of user in application developed.
3. *Reliability:* Current research adopts dual model approach which segregates responsibilities of an IT expert from a domain expert. This segregation restrict any communication gap between an IT expert and domain expert while designing an application. Moreover, archetypes available on standard online library (such as, CKM) follows a rigorous approach for their development, modification and deployment.
4. *Security:* A generic persistence named AEAV is proposed in current research. Coding of archetype name and attribute name creates a meaningless information in absence of mapping dictionary, number of bits used for coding, and coding algorithm.
5. *Consistency:* Consistency is achieved in current research by adopting RDBMS for persistence of data. RDBMS commits to ACID (Atomicity, Consistency, Isolation and Durability) properties. Any system built on RDBMS, automatically commits to ACID properties.

Table 1. Solution to challenges identified

#	`Challenge	Proposed Solution
1	Data Misinterpretation	• Solved through adoption of archetype based system. • Using archetypes aids in capturing maximum possible information about medical concept. • Archetypes provide links to standard medical terminologies such as, SNOMED-CT and LOINC.
2	Distinct set of attribute for same medical concept	• Archetypes define standard set of attribute for a medical concept. • Archetypes following one standard can be transformed to archetype following another standard using online tools such as, POSEACLE convertor.
3	Distinct Local Schema	• Proposed generic schema, AEAV handles this issue. • Schema is capable to capture all existing and future data requirements without making any changes in schema.
4	Sparseness	• AEAV doesn't store any null value. • AEAV reduces space by eliminating need of storing long archetype and attributes names.

6. *Completeness:* Archetypes are designed to capture maximal consensus on data definition related to a medical concept. This ensure completeness of data. Healthcare domain has one related problem of sparseness, i.e., most of the values are kept null due to many reasons such as, patient don't want to reveal personal information and some parameters are not applicable in certain situations. AEAV is designed to avoid storage of null values and thus, excel in saving storage space. To achieve completeness, mapping dictionary can be used to construct complete set of attributes constituting a medical concept. Knowledge of complete set of attributes facilitates in identifying null values, i.e., the attributes for which no value is found, will be null.
7. *Accessibility:* Current research aims to provide interoperability of data. This enhances accessibility of data.
8. *Fitness of use:* Approach in current research suggests to adopt a standardized EHRs system and generic persistence. Involvement of standardized EHRs restrict to use same set of attribute for same concept. Moreover, generic persistence proposed in current research helps in providing schema interoperability. Thus, data ported from one site to another will depict same meaning to both.

5. EXPERIMENTS AND RESULTS

So far, authors achieved syntactic, structural and semantic interoperability by adopting a standard based system for developing clinical application. Moreover, generic schema proposed in Section 3.2 provides reduced storage requirement (thus, handling more volume), support for variety of (heterogeneous) data and schema interoperability. This section is devoted to show the impact of different physical storage of RDBMS viz. row-oriented and column-oriented on timeliness of data stored as per AEAV. Absence of right information at right time might cause severe losses in terms of patient's health and life. Thus, retrieval of desired data at a very high speed is very crucial in healthcare domain.

5.1 Hardware and Software Used

We implemented AEAV schema using MySQL Workbench 6.0 CE (row-oriented) and MonetDB 5 (column-oriented). All experiments are executed on a pair of 2.66 GHz dual-core Intel Xeon processors with 16 GB RAM running Mac OS X 10.4.11.

5.2 Dataset Description

Data set on which experiments are performed are collected using two different methodologies.

Method #1: Healthcare applications were designed using three archetypes provided by openEHR at CKM, namely openEHR-EHR-OBSERVATION.lab_test-liver_function.v1, openEHR-EHR-OBSERVATION.lab_test-thyroid.v1 and openEHR-EHR-OBSERVATION.blood_pressure.v1 and deployed to three private clinics for collection of data.

Method #2: Liver Disorder dataset and Thyroid dataset were downloaded from the UCI machine learning repository [18-19]. However, it was not standardized. So, user interfaces were developed for related archetypes (openEHR-EHR-OBSERVATION.lab_test-liver_function.v1 and openEHR-EHR-OBSERVATION.lab_test-thyroid.v1). Downloaded data was manually fed in these forms to have dataset as per AEAV schema.

To have persistence as per AEAV in both of the above methodologies, hibernate layer is modified to accommodate coding algorithm of AEAV. Mapping dictionaries were manually populated and provided in the application to achieve coding of archetype name and attribute name at hibernate layer. Attributes present in archetypes but not in UCI datasets were kept null.

In total 75K records were collected as per the relational model. To have reliable and accurate results, we removed redundant records. After removal of redundant data, 75K records were reduced to 50K records.

5.3 Results

Timeliness behavior of AEAV is tested through seven distinguished tasks by formulating queries in MySQL and MonetDB.

Table 2. Impact of row-oriented and column-oriented storage approach on Timeliness of AEAV modelled data

ID	Task	Task Description	Time Taken (seconds)	
			Row-Oriented Storage	Column-Oriented Storage
Q1	Extracting Complete Column Details	Extracting details of Systolic pressure	0.377	0.359
		Extracting Systolic pressure, Diastolic pressure and overall interpretation of all patients	0.618	0.419
		Extracting ALP, AST, ALT, Albumin and Globulins of all patients	5.429	0.577
Q2	Extracting Complete Row Details	Extracting data of all patients	63.684	1.482
		Extracting data of all patients having Total T3 greater than 2	0.499	0.374
		Extracting data of all patients having Systolic pressure greater than 100, Diastolic pressure less than 100 and overall interpretation as Hypotension	0.755	0.569
Q3	Extracting Selected Column Details of Selected Rows	Extracting Systolic pressure, Diastolic pressure and overall interpretation of all patients having Patient ID greater than 4500 and Systolic pressure greater than 100	2.805	1.091
		Extracting ALP, AST, ALT, Albumin and Globulins of all patients having Patient ID less than 5000 and AST greater than 100	4.47	3.962
Q4	Performing Statistical Analysis	Extracting the average value of albumin among people tested for Liver	1.36	0.687
		Extracting number of patients tested for BP and diagnosed with Hypotension	0.396	0.232
		Group the patients tested for Liver according to Albumin values	12.635	0.952
Q5	Adding data	Insert data of one patient	0.774	0.234
Q6	Deleting data	Delete data of one patient	0.372	0.218
Q7	Modifying data	Update data of one patient	0.315	0.297

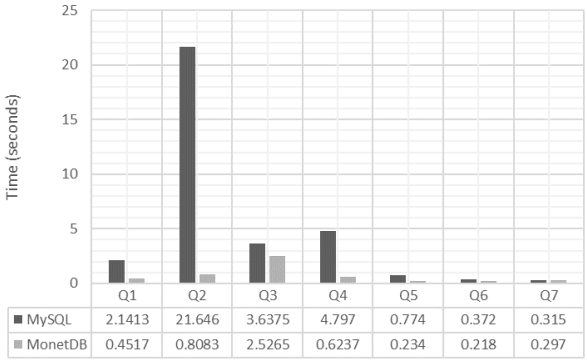


Figure 6. Results of various tasks performed to observe timeliness behavior of AEAV

Seven tasks are extracting complete column details, extracting complete row details, extracting complete row details of selected rows, performing statistical analysis, adding data, deleting data and modifying data. Time taken for performing different tasks is presented in Table 2. Execution time taken by various queries coded for the underlying tasks (presented in Table 2) are averaged and correspondingly presented graphically in Figure 6.

Column details are mostly enquired in EHRs domain to perform analysis of population. The analysis performed might deliver new knowledge that can add value to existing healthcare practices. Storing EHRs data as per AEAV in column oriented physical storage can help in performing faster analysis. Change in physical storage approach of AEAV has shown a drastic performance change in case of extracting rows, i.e., accessing data of specific patients. In summary, under all query scenarios, column-oriented storage approach (MonetDB) outperforms row-oriented storage approach (MySQL).

6. CONCLUSIONS

Considering large scale healthcare data, a clinical information system must be able to handle the 3V's (Volume, Variety and Velocity) of BIG DATA and should simultaneously support data interoperability. Current research propose framework for clinical information system that can handle data misinterpretation, provide same set of attribute for same medical concept in appropriate context and generic schema for persistence. Thus, proposed framework enhances data quality while semantic exchange of data from one organization to another. Moreover, adopting proposed approach will

1. supports syntactic, structural and semantic interoperability,
2. refers a generic schema capable of capturing all current and future data requirements without making any changes in schema,
3. eliminates the need of storing null values to save storage space,
4. supports storage of heterogeneous data,
5. achieves accuracy and validity, believability, reliability, security, completeness, accessibility, consistency and fitness for use as data quality parameters, and

6. improves search efficiency by utilizing optimization techniques of MonetDB.

Current research adopts standard based EHR system to achieve syntactic, structural and semantic interoperability. Despite of attaining syntactic, structural and semantic interoperability, various organizations can adopt their own local schema which is tailored as per their requirements. Thus, efficient data interoperability demands for a generic schema. Hence, a new generic schema is proposed in current research to attain schema interoperability. The generic schema proposed is rich enough to accommodate any existing and future data demands while eliminating the need of storing null values and handling heterogeneous data. Apart from providing schema interoperability, the proposed schema also offers improved security to enhance data quality.

Considering timely access of data, current research observes impact of different physical storage variants of RDBMS, i.e., row-oriented (MySQL) and column-oriented (MonetDB) on proposed generic schema. Experiments are conducted to show the impact on timeliness of data by adopting different variants of RDBMS to store data as per proposed generic schema. Results achieved clearly favors the adoption of column oriented RDBMS over row oriented RDBMS.

7. REFERENCES

[1] Atalag, K. and Bilgen, S. Multi-Level Modeling and the Role of Archetypes in the Design of Health Information Systems: A Modeling Example in Endoscopy. *HIBIT '07 Proceedings of the International Symposium on Health Informatics and Bioinformatics*, May2007.

[2] Atalag, K. and Yang, H.Y. From openEHR domain models to advanced user interfaces: a case study in endoscopy. *In Health Informatics New Zealand Conference*, November 2010.

[3] Beale, T. and Frankel, H. *The openEHR Reference Model. Extract Information Model*. The openEHR release, 1, 2007.

[4] Beale, T. and Heard, S. *openEHR architecture overview*. openEHR Foundation. London, UK, 2008.

[5] Beale, T. and Heard, S. *The openEHR archetype model-archetype definition language ADL 1.4*. openEHR release, 1(2), 2008.

[6] CEN European committee for Standardization: www.cen.eu [online] (Accessed 01/16).

[7] Clinical Knowledge Manager: http://www.openehr.org/ckm/ [online] (Accessed 01/16).

[8] Costa, C.M., Menárguez-Tortosa, M. and Fernández-Breis, J.T. Clinical data interoperability based on archetype transformation. *Journal of biomedical informatics*, 44(5), 2011, 869-880.

[9] Dinu, V. and Nadkarni, P. Guidelines for the effective use of entity–attribute–value modeling for biomedical databases. *International journal of medical informatics*, 76(11), 2007, 769-779.

[10] Health Level 7 International - Homepage: www.hl7.org [online] (Accessed 02/16).

[11] International Health Terminology Standards Development Organization. Systematized Nomenclature of Medicine-Clinical Terms (SNOMED CT). Available from: http://www.ihtsdo.org/snomed-ct/ [online] (Accessed 04/16).

[12] ISO organization: www.iso.org [online] (Accessed 01/16).

[13] Logical Observation Identifiers Names and Codes – LOINC: https://loinc.org/ [online] (Accessed 04/16).

[14] openEHR Foundation: www.openehr.org [online] (Accessed 01/16).

[15] Patrick, J., Ly, R. and Truran, D. Evaluation of a persistent store for openEHR. *HIC 2006 and HINZ 2006: Proceedings*, 2006, 83-89.

[16] Sachdeva, S. and Bhalla, S. Semantic interoperability in standardized electronic health record databases. *Journal of Data and Information Quality (JDIQ)*, 3(1), 2012, 1-37.

[17] Sachdeva, S., Yaginuma, D., Chu, W., & Bhalla, S. Dynamic generation of archetype-based user interfaces for queries on electronic health record databases. *In Databases in Networked Information Systems, Springer Berlin Heidelberg*, 2011, 109-125.

[18] UCI Machine Learning Repository: Liver Disorders Data Set: https://archive.ics.uci.edu/ml/datasets/Liver+Disorders (Accessed 06/15).

[19] UCI Machine Learning Repository: Thyroid Disease Data Set: https://archive.ics.uci.edu/ml/datasets/Thyroid+Disease (Accessed 06/15).

Towards Rigorous Evaluation of Data Integration Systems - It's All About the Tools

Boris Glavic
Illinois Institute of Technology
bglavic@iit.edu

ABSTRACT

Given the maturity of the data integration field it is surprising that rigorous empirical evaluations of research ideas are so scarce. We identify a major roadblock for empirical work - the lack of tools that aide a researcher in generating the inputs and gold standard outputs for their integration tasks in a controlled, effective, and repeatable manner. In this talk, I will give an overview of our efforts for developing such tools and highlight how they have been used for streamlining the empirical evaluation of a wide variety of integration systems. Particularly, the talk will focus on two systems: iBench and BART. iBench is a metadata generator that can be used to evaluate a wide-range of integration tasks (data exchange, mapping creation, mapping composition, schema evolution, among many others). The system permits control over the size and characteristics of the metadata it generates (schemas, constraints, and mappings). BART (Benchmarking Algorithms for data Repairing and Translation) is a scalable system for introducing errors into clean databases for the purpose of benchmarking data-cleaning algorithms. The presentation will include a short live demonstration of both systems.

Three Semi-Automatic Advisors for Data Exploration

Thibault Sellam
CWI
thibault.sellam@cwi.nl

ABSTRACT

In data exploration, users query a database to discover its content. Typically, explorers operate by trial and error. They write a query, observe the results and reiterate. When the data is small, this approach is perfectly acceptable. But what if the database contains 100 s of columns and 100,000s of tuples? During this talk, I will introduce Blaeu, Claude and Ziggy, three "advisors" for data exploration. The main idea is to use simple machine learning models to help users navigate the space of all possible queries and views. I will present practical use cases, discuss the main ideas behind each assistant and describe open research problems.

Graph-based Exploration of Non-graph Datasets

Udayan Khurana
IBM Research
ukhurana@us.ibm.com

ABSTRACT

Graphs or networks provide a powerful abstraction to view and analyze relationships among different entities present in a dataset. However, much of the data of interest to analysts and data scientists resides in non-graph forms such as relational databases, JSON, XML, CSV and text. The effort and skill required in identifying and extracting the relevant graph representation from data is often the prohibitive and limits a wider adoption of graph-based analysis of non-graph data. In this paper, we demonstrate our system called GraphViewer, for accelerated graph-based exploration and analysis. It automatically discovers relevant graphs implicit within a given non-graph dataset using a set of novel rule-based and data-driven techniques, and optimizes their extraction and storage. It computes several node and graph level metrics and detects anomalous entities in data. Finally, it summarizes the results to support interpretation by a human analyst. While the system automates the computationally intensive aspects of the process, it is engineered to leverage human domain expertise and instincts to fine tune the data exploration process.

Data Quality Management in Data Exchange Platforms An Approach for the Industrial Data Space in Germany

Christoph Quix
Fraunhofer FIT
christoph.quix@fit.fraunhofer.de

ABSTRACT

Data quality plays an important role in data marketplaces as a value is assigned to the data and customers pay for the received data. It is known that data quality problems arise especially in data integration projects, when data (from one organization) is used in a different context than originally planned. This problem is aggravated in a setting where data is exchanged between different organizations as in a data marketplace. In addition, data consumers expect a high data quality as they pay for the data. Research in data quality has derived many issues from quality management for classical products and transferred this to the case of data management. An open question is how results from quality assurance and pricing models in the classical product world can be transferred to data. In this talk, we will review the state of the art in the area of data quality management and pricing in data marketplaces and report on the initiative "Industrial Data Space" in Germany, in which open platform for data exchange between industrial organizations is being developed.