Measuring and Constraining Data Quality with Analytic Workflows

Laure Berti-Équille [*]
Université de Rennes 1
Campus Universitaire de Beaulieu
35042 Rennes cedex, France
berti@irisa.fr

ABSTRACT

One challenging aspects of data quality modeling and management is to provide flexible, declarative and appropriate ways to express requirements on the quality of data. The paper presents a framework for specifying and checking constraints on data quality in RDBMS. The evaluation of quality of data (QoD) is based on the declaration of data quality metrics that are computed and combined into so-called *QoD analytic workflows.* These workflows are designed as a composition of statistical methods and data mining techniques used to detect patterns of anomalies in the data sets. As metadata they are used to characterize various quantifiable dimensions of data quality (e.g., completeness, freshness, consistency, accuracy). The paper proposes a query language extension for constraining data quality when querying both data and its associated QoD metadata. Probabilistic approximate constraints are checked to determine if the quality of data is (or not) acceptable to build quality-constrained query results.

1. INTRODUCTION

With the ever-growing data glut problem, our capabilities for collecting and storing data have far outpaced our abilities to analyze, summarize available data, and more critically, to evaluate and systematically check the quality of this data (QoD). While database technology has provided us with the basic tools for the efficient storage and lookup for large data sets, one of the current issues is how to measure, analyze and enforce data quality in databases. Data quality is known as a ”multidimensional, complex and morphing concept” [10]. Maintaining a high level of data quality in a database is challenging and cannot be limited to one-shot approaches addressing simpler and more abstract versions of the wide range of data quality problems. Technically, data quality management mainly refers to the detection and elimination of various data quality problems, such as:

- *Duplicate and redundant data.* A wide range of techniques have been proposed for record linkage [23] and entity resolution since the ”merge/purge problem” identified by Hernández and Stolfo [12],

- *Imperfect data.* Inconsistency, imprecision, and uncertainty are some of the problems associated with data imperfection [19]. A significant amount of work has been proposed in the areas of integrity constraint checking, imprecise data management, and uncertainty in DBMSs, e.g., [2, 4, 9],

- *Missing values and incomplete database.* The problem of handling incomplete information has also been addressed in information and database systems [17],

- *Stale data.* Various refreshment techniques and synchronization policies have been proposed for ensuring data freshness depending on the type of system architecture [6, 20]: e.g., data warehousing systems check the recentness of materialized views; caching systems estimate the time-to-live of cached data before expiration and tune the caching policy for ensuring data currency.

QoD dimensions are numerous with various definitions, interpretations and measurement methods depending on the considered application domain. One of our goals is to cope with this diversity in proposing a flexible way to express and measure the dimensions of data quality and to check constraints on these dimensions. Among other challenging research directions that have been recently identified in Data Quality Research [5] (e.g., pattern design for quality-aware IS engineering [1], new perspectives of methodological approaches for data quality management [3] or benchmarks for comparing objectively the research contributions to specific data quality problems [22]), QoD metadata modeling and management and QoD-aware query languages are indeed of particular interest for the community. There is currently no model that captures in a precise and easy way the semantics of all static and dynamic dimensions of data quality, and allows carrying out relevant automatic checking based on QoD metadata management. The existing propositions focus on one or two *hard-coded* QoD dimensions (often considered separately) [14, 11, 13, 20]. The approach which consists in defining *default* QoD dimensions and in basing

[*] Currently visiting researcher at AT&T Labs Research, NJ, USA supported by a Marie Curie International Fellowship within the 6th European Community Framework Programme (FP6-MOIF-CT-2006-041000).

VLDB ‘08, August 24-30, 2008, Auckland, New Zealand

the analysis on the Cartesian product or on the *ad-hoc* composition of some data quality dimensions do not make it possible to model, in a faithful way, the interdependencies between these QoD dimensions [1]. It is thus necessary to develop metadata models and declarative data manipulation languages which make it possible to represent combinations and interdependencies of user-defined QoD dimensions with relevant measurement techniques and constraints. As a first step in this direction, the paper presents a framework and a QoD-constrained language for integrating "natively" the specification, evaluation and checking of data quality in the data management system [5]. The emphasis of the paper is put on:

- **QoD Evaluation**. We define and implement *QoD analytic workflows* for computing of a set of statistical and data mining functions that evaluate various dimensions of data quality. The results of these functions are stored in a metadata repository as QoD measures characterizing data distribution properties, data quality problems, anomaly patterns, and any useful information reflecting QoD evaluation for different granularity levels of a RDBMS (*i.e.,* for the values, tuples, attribute domains, tables or the database),
- **QoD-aware data management**. We present a query language extension that allows the declaration and assignment of *contracts* on the quality of data as sets of constraints on the QoD measures resulting from the analytic workflows.

The paper is organized as follows: Section 2 presents the main steps of our approach based on the design of analytic workflows for QoD evaluation. Section 3 presents the syntax of the language extension we propose for measuring and checking constraints on data quality. Section 4 presents related work on declarative languages dedicated to data quality control and management. Section 5 concludes the paper and presents our research perspectives.

2. A FRAMEWORK FOR EVALUATING AND CHECKING QOD

2.1 Overview

Our approach can be summarized through the following steps:

1. **Definition and computation of analytical functions for QoD evaluation**. First, the design of analytic workflows consists of the definition (or eventual reuse) of functions that measure objectively various dimensions of the quality of data (e.g., freshness, consistency, completeness, accuracy); each dimension may characterize an aspect of the quality of a database object instance (*i.e.*, value, tuple, attribute domain, table or database). The computed measures are stored and managed as QoD metadata in a repository. One QoD dimension of a DB object instance can be characterized by many QoD measures in the metadata repository.
2. **Definition of probabilistic and approximate constraints on QoD measures**. To establish a QoD diagnostic, measured QoD values have to be compared to expected QoD values with a certain degree of tolerance. To do so, probabilistic and approximate constraints are specified and checked to determine if the quality is acceptable or not for each QoD dimension. The computed probabilities are stored in the metadata repository and refreshed by the system.
3. **Quality-constrained query declaration and processing**. Probabilities assigned to the QoD dimensions are then used in the query processing to build quality-aware query results (or QoD diagnostics) in conformance with the QoD constraints predefined in the analytic workflows.

From the system-centric perspective, our approach has been translated into three main components that pragmatically integrate data quality awareness in the system: *i)* an extensible library of functions and statistical methods for measuring various aspects of QoD, *ii)* a quality metadata manager that stores, indexes, searches, and refreshes QoD measures and related descriptive metadata, and *iii)* an extended query engine that allows declaration and manipulation of data with probabilistic approximate constraints on QoD metadata. The paper mainly focuses on the components *i)* and *iii)* for declaring and manipulating constraints on QoD.

2.2 Designing QoD Analytic Workflows

In order to analyze and evaluate the quality of data, a relevant sequence of tasks are specified and planned depending on the underlying goal of the QoD evaluation. For this purpose, we define *QoD analytic workflows* as sets of parameters interacting with each other. These include: a goal (G), a set of tasks (T), a set of resources (R), a set of function allocations (A), and a set of limitations (L). A QoD analytic workflow, W, is then a function of all the sets interacting with each other, as: $W = \{G, T, R, A, L\}$. The goal G of an analytic workflow is defined for a DB object instance resource in order to characterize it and to analyze a variety of data quality issues for a specific purpose. A goal has three components: *i)* the set of resources (in our case, data instance objects such as a set of rows or a set of values depending on the considered granularity), *ii)* the data quality issue (e.g., data consistency, freshness or completeness), and *iii)* the purpose (e.g., detect outliers or correct anomalies). The set of tasks (T) are the building blocks of the analytic workflow. Tasks can be broken down into smaller tasks through task refinement. This activity continues until a satisfied level of abstraction has been achieved for that particular QoD analytic workflow being modeled. The set of resources (R) include the set of inputs of the analytic workflow, such as the set of DB object instances to analyze (e.g., attribute domain, tuple, value) and the set of output QoD metadata including QoD measures and descriptive metadata (*i.e.,* detailed description of the settings and parameters of the functions). The set of function allocations (A) define the set of analytical functions to be applied for QoD evaluation. A set of limitations (L) may be specified and associated to a task or to a function allocation stating for example that the function can only be applied to categorical data sets.

Table 1 gives a classification of the analytical functions that are used for QoD evaluation; the mention from level I to level IV indicates the increasing range of complexity of the methods. It also provides examples of the functions used

Level	Category	Description	Examples		
			QoD	Functions	DB object
I	QoD Profiling Functions	Simple counts computed from the database dictionary, look-up tables, control, log or trace files, usually with single-pass algorithms. These metadata can be used to characterize some aspects of database completeness and freshness depending on the level of granularity of the database object instances (*i.e.*, value, record, column, table or database in the relational context)	CP	`nullValues%` returns a percentage that represents the quantity of null values.	D,T,R,A
			F	`updateFreqStdDev` returns a decimal in [0,1] that represents how frequently the DB object instance has been updated since its creation time compared to the average frequency of the objects of the same granularity.	D,T,R,A
II	QoD Constraint-Based Functions	Sets of global or application-specific integrity rules, consistency constraints or inferred rules from statistical techniques that characterize the most plausible relationships between data instances or that compute the deviations from the rules. These constraints are verified at runtime. Constraint violations indicate errors or dubious data.	CT	`SyntacticCorrectness` returns a decimal number in [0,1] that represents the percentage of format discordances, syntactical errors and misspellings.	D,T,A,R,V
III	QoD Synopses Functions	Statistical summaries, aggregates or parametric estimations computed as approximate answers with deterministic error bounds or probabilistic guarantees that the approximate answer is the actual one. Basic synopses are samples, equi-depth histograms, quantile computation. Advanced synopses are computed from sketch-based computation techniques (e.g., V-optimal histograms, wavelets). They are useful to quickly reveal unlikely values that are artifacts or inconsistent patterns from samples of very large data sets before going into deeper and more expensive analysis.	AC	`outlierProb` returns a decimal number in [0,1] based on IQR that represents the probability of being an outlier in the attribute domain.	V
IV	QoD Exploratory Mining Functions	Data mining results obtained from techniques such as clustering, association rule discovery, and decision trees. These techniques have the same goal as the previous methods, in the sense that they can be used to detect data glitches (e.g., duplicates, anomaly patterns, and dubious data), but their computation and maintenance costs are much higher.	U	`DupDetectionProb` returns a decimal number in [0,1] that represents the probability of the DB object instance to be a duplicate after clustering and association rule mining on a combination of attributes whose values are identical or similar.	R

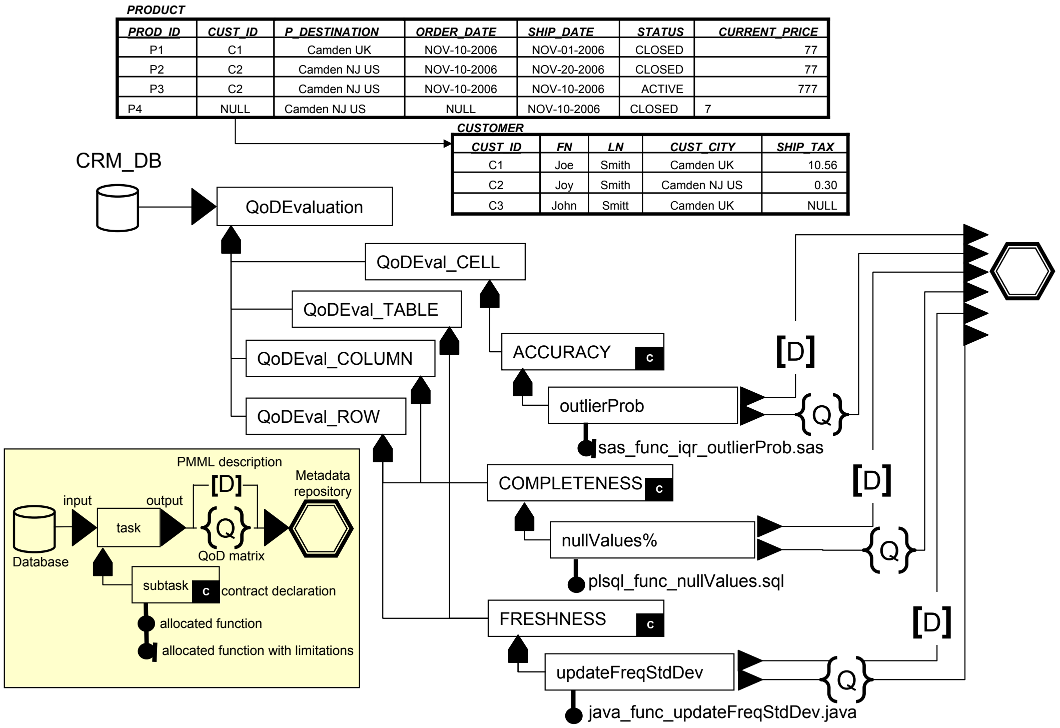
Table 1: Categories of Functions for QoD Evaluation

for computing measures related to the following data quality dimensions: completeness (CP), consistency (CT), accuracy (AC), freshness (F) and uniqueness (U) (*i.e.,* absence of duplicates). The computed measures are respectively associated to the instances with the level of granularity the measure is computed from, namely the value (V), record (R), attribute (A), table (T) and database (D) (see the last column of Table 1). The granularity levels correspond to the main structural elements of the relational DB; metadata are thus associated to each DB object instance depending on its granularity level.

The reader is invited to read the chapter 2 of [5] to have a more detailed description of the functions that are used in the QoD analytic workflows.

The set of resource allocations (A) defines the relationship of the tasks and the relationship of the resources, as well as the task/resource allocations. These relationships are given at the time the resources and tasks are defined. The set of limitations (L) defines any limitations or restrictions imposed on the tasks and resources. Such restrictions may include scheduling restrictions (with precedence constraints in task planning), resource restrictions (e.g., resources A and B are mutually exclusive), resource allocation restrictions, (e.g., a function f cannot be applied for task t), and so on. Once sets of resources and their relations are defined in the QoD analytic workflow model, the data and functions become inputs to the tasks and QoD metadata become output. Output from one analytical function could serve as input to other functions; the same resource can serve either as input or output, or both depending on which task it applies to.

Figure 1 presents the example of a QoD analytic workflow designed for evaluating the quality of data of the database named CRM_DB composed of two tables: PRODUCT and CUSTOMER. This includes several tasks of evaluation on three QoD dimensions, namely freshness, accuracy, and completeness at different granularity levels (e.g., cell, row, column, table). Data object instances are the inputs of the tasks. Each task may be composed of several subtasks with allocated functions for the computation of the QoD measures. Summary statistics for numerical values are computed, out-of-range data values may also be detected with univariate statistics and percentiles (IQR) e.g., '`sas_func_iqr_outlierProb.sas`' of the subtask `outlierProb` composing the task `ACCURACY` evaluates the accuracy at the `CELL` granularity level of CRM_DB. The execution of each allocated function generates QoD measures that are stored in the metadata repository as arrays of QoD values represented in Figure 1 as **{Q}**. The detailed description of each function is stored as descriptive metadata in the repository using



PRODUCT

PROD_ID	CUST_ID	P_DESTINATION	ORDER_DATE	SHIP_DATE	STATUS	CURRENT_PRICE
P1	C1	Camden UK	NOV-10-2006	NOV-01-2006	CLOSED	77
P2	C2	Camden NJ US	NOV-10-2006	NOV-20-2006	CLOSED	77
P3	C2	Camden NJ US	NOV-10-2006	NOV-10-2006	ACTIVE	777
P4	NULL	Camden NJ US	NULL	NOV-10-2006	CLOSED	7

CUSTOMER

CUST_ID	FN	LN	CUST_CITY	SHIP_TAX
C1	Joe	Smith	Camden UK	10.56
C2	Joy	Smith	Camden NJ US	0.30
C3	John	Smitt	Camden UK	NULL

Figure 1: Example of QoD Analytic Workflow for CRM_DB

PMML[1] represented as [**D**] in Figure 1. PMML is used to describe in details the parameters, thresholds or initial conditions that are necessary for executing the specified analytical functions. For traceability reasons, the PMML descriptions of the functions are stored in the metadata repository mainly to allow the users to keep the complete and detailed history of the functions used in the workflows and also to be able to eventually change the initial parameters or thresholds for computing new QoD measures.

Analytic workflows specify and combine multiple analytical functions as the ones illustrated in the example. Functions may be implemented in different ways (stored procedure, SAS script, C or Java program) depending on the set of limitations specified in the workflow. Of course, many other analytical functions may be added to the library. The panel of analytical functions used for computing QoD measures and generating QoD metadata gives relevant indications for characterizing potential data quality problems and understanding anomaly patterns. Nevertheless, the overall process is a 2-step based approach: in the first off-line step, the analytic functions declared in the contract types of the workflow compute QoD measures from the database object instances; these measures are stored in the metadata repository and in the second step, constraints on data quality are declared in the contract instances or in the QWITH queries and they compare the pre-computed measures from the metadata repository to expected thresholds in order to build the query result.

3. DECLARATION OF CONSTRAINTS ON DATA QUALITY

In our approach, constraints on data quality are grouped and expressed by means of *quality contract types* and *quality contracts instances*:

1. A **quality contract type** defines a set of quality dimensions, measures and functions associated to a particular database object instance. It corresponds to the computation tasks of a particular QoD analytic workflow.

2. A **quality contract instance** is a set of one-sided range constraints defined on the QoD dimensions declared in the contract type.

3.1 Declaration of QoD Contract Types

The syntax for creating a quality contract type is given in Figure 2. A contract type is named (`ct_name`) and assigned to a given schema of the database. Each contract type is composed of a list of named measurable dimensions, the datatype of the output result, the granularity level which the measure is associated to (e.g., `ON CELL, ROW, COLUMN, TABLE, DATABASE`) and the identifier or name of the function that computes the measure following the `BY FUNCTION` statement. QoD measures are stored in the metadata repository and they are assigned either to: *i)* a global granularity level, *i.e.* `ON CELL, ROW, COLUMN, TABLE`, or `DATABASE`, or to: *ii)* a specific DB object instance, *i.e.*, an existing table, column, record, or cell. For each row in the database, the ROWID pseudo column returns a row's address.

The analytical function may be a PL/SQL[2] procedure or the call specification of a Java, C or SAS program. The creation of a

[1] PMML - Predictive Model Markup Language, Version 3.1: http://www.dmg.org/

[2] Although our implementation was initially made to access Oracle databases with providing a set of appropriate

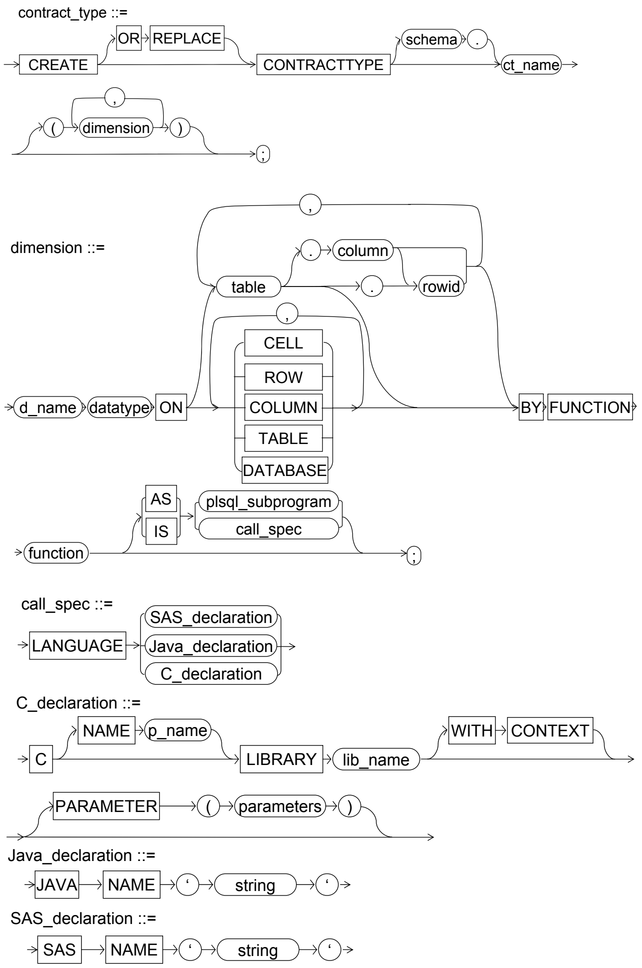


Figure 2: Syntax of Quality Contract Type Creation

task in the QoD analytic workflow can be associated to the creation of a contract type on a database object instance. This leads to the execution of the declared analytical functions, the computation and storage of the measured QoD values and the associated PMML descriptions of the functions in the metadata repository. As an example, Table 2 gives the syntax for creating the quality contract types resulting from the tasks of the illustrated QoD analytic workflow.

For instance, in `FRESHNESS` contract type, `updateFreqStdDev` is a float returned by a JAVA function named '`java_func_updateFreqStdDev.java`' that indicates how frequent the rows and tables have been updated since their creation. In `ACCURACY` contract type, `outlierProb` is a float returned by a SAS function named '`sas_func_iqr_prob.sas`' that returns the probability of being outlier for each value of the database. In `COMPLETENESS` contract type, the PL/SQL procedure `plsql_func_nullValues` returns a percentage of missing values for each row, column, and table of CRM_DB.

3.2 Declaration of QoD Contract Instances

The syntax of creation of a quality contract is given in Figure 3. Each contract declaration refers to an existing contract type (`ct_name`). It defines the constraints on each contract type dimension with simple or composed expressions using basic binary operators. Consider again the CRM_DB database and the QoD analytic workflow given in Figure 1, three quality contract types have been defined for characterizing freshness, accuracy, and completeness on CRM_DB.

```
CREATE CONTRACTTYPE FRESHNESS(
 updateFreqStdDev FLOAT ON ROW, TABLE
  BY FUNCTION java_func_updateFreqStdDev
  IS LANGUAGE JAVA
  NAME 'java_func_updateFreqStdDev.java');
CREATE CONTRACTTYPE ACCURACY(
 outlierProb FLOAT ON CELL
  BY FUNCTION sas_func_iqr_prob
  IS LANGUAGE SAS
  NAME 'sas_func_iqr_prob.sas');
CREATE CONTRACTTYPE COMPLETENESS(
 nullValues% FLOAT
   ON ROW, COLUMN, TABLE
   BY FUNCTION IS plsql_func_nullValues);
```

Table 2: Example of Quality Contract Type Declaration

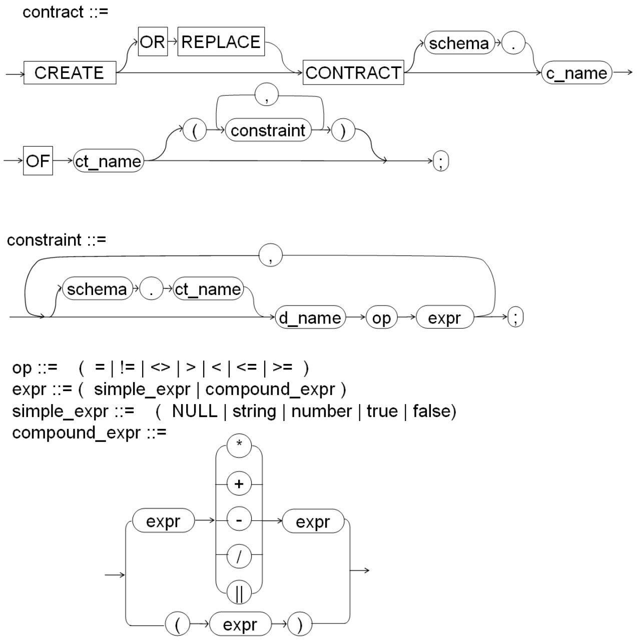


Figure 3: Syntax of Quality Contract

Table 3 gives examples of quality contract instances, named `fresh`, `accurate`, and `complete` whose definition is based on their respective contract types: `FRESHNESS`, `ACCURACY`, and `COMPLETENESS` given previously. Contract instances describe the one-sided range constraints to be checked in conformance with the expected values on each dimension declared in the corresponding quality contract type.

```
CREATE CONTRACT fresh OF FRESHNESS(
   updateFreqStdDev > (.50,.50));
CREATE CONTRACT accurate OF ACCURACY(
   outlierProb < .06;
CREATE CONTRACT complete OF COMPLETENESS(
   nullValues% <= (.20,.20,.35);
```

Table 3: Example of Contract Declaration

PL/SQL stored procedures for characterizing various data quality dimensions, an interesting extension would be to provide the ability to the user to call or declare any stored procedure in the proprietary language of the database management system he/she is connected to.

The declaration of contract types and instances is a part of the QoD analytic workflow specification. Our objective is to incorporate and use a set of analytical functions for computing QoD measures that can be easily extended by other user-defined functions. The call and execution of functions is triggered immediately after the validation of the contract type declaration in the workflow design.

3.3 Checking Constraints on QoD Measures

Once the contracts type and instances have been declared and validated in the workflow execution, constraints on QoD measures are checked at the specified granularity levels or for the particular database object instances. The constraints are based on the comparison between measured and expected values for each QoD dimension. The degree to which these constraints are satisfied is computed as an aggregated value called *QoD acceptability* combining the checking result of the constraints on every QoD dimension. This value represents the probability that the QoD dimension is acceptable considering that the values of its associated QoD measures are in conformance with expected bounds given in the constraints of the contract instance. More formally, given a particular QoD dimension Q_i, the QoD acceptability of a DB object instance indicates the likelihood δ_i that the QoD dimension measured for the database object instance o is acceptable considering the constraints c_{ij} on its associated QoD measures m_{ij} being satisfied with respect to an interval of expected values M_{ij} with a tolerance ϵ_{ij}. Acceptability of QoD dimension Q_i on the DB object instance o, noted $Acc_i(o)$ is then defined as:

$$Acc_i(o) = Pr(Q_i(o)| \bigwedge_j c_{ij} : m_{ij} \in [M_{ij} \pm \epsilon_{ij}]) = \delta_i. \qquad (1)$$

Because all the facets (and corresponding measures) of a QoD dimension may not be equally considered, δ_i is defined as the weighted sum of all distances between measured and expected values that characterize the QoD dimension Q_i, as:

$\delta_i = \sum_j w_{ij}.d_{ij}$ with $\sum_j w_{ij} = 1$ and

$$d_{ij} = \begin{cases} 0 & \text{if } c_{ij} \text{ is satisfied wit } \epsilon_{ij} = 0 \\ Normdist(m_{ij}, M_{ij}) & \text{if } c_{ij} \text{ is satisfied with } \epsilon_{ij} > 0 \\ 1 & \text{if } c_{ij} \text{ is not satisfied with } \epsilon_{ij} > 0 \end{cases}$$

d_{ij} is null when the measured QoD value m_{ij} fully satisfies the constraint c_{ij} with respect to the expected interval M_{ij} without ϵ_{ij}. If the constraint is not satisfied even within the interval of the tolerance noted $[M_{ij} \pm \epsilon_{ij}]$, then d_{ij} equals 1; otherwise d_{ij} is computed as the normalized distance between the measured QoD value and the expected value for the object instance o. Suppose that a row has the measure for `updateFreqStdDev` equals to .42 which does not satisfied exactly the declared constraint ($.42 < .50$), the possibility to soften the constraint may be interesting in order to provide results to the user that do not exactly meet their requirements in terms of quality but still are close.

Suppose the quality contracts given in Table 3 are applied to CRM_DB. For each CRM_DB object instance, the constraints declared in the quality contracts are checked. For each granularity level, Table 4 gives the computed probabilities for the DB object instances of the PRODUCT table of CRM_DB. Again, null probability means acceptable QoD dimension; 1 means unacceptable. We suppose that the probabilities of the other CRM_DB object instances are null.

3.4 Constraining Data Quality in the query

We have designed a query language extension named XQuaL for manipulating and checking constraints on the quality of data in RDBMS. The syntax of a quality-extended query is given in Figure 4. Once declared, one (or several) contract(s) or constraints may be used in the QWITH part of the XQuaL queries.

`'sfw_query'` represents the classical SELECT-FROM-WHERE statement of the query. In the QWITH statement, declared contract instances are identified and the constraints defined in the declared contracts are checked on ROW or CELL granularity levels specified after the statement ON. In the QWITH declaration, two alternatives are possible for checking constraints:

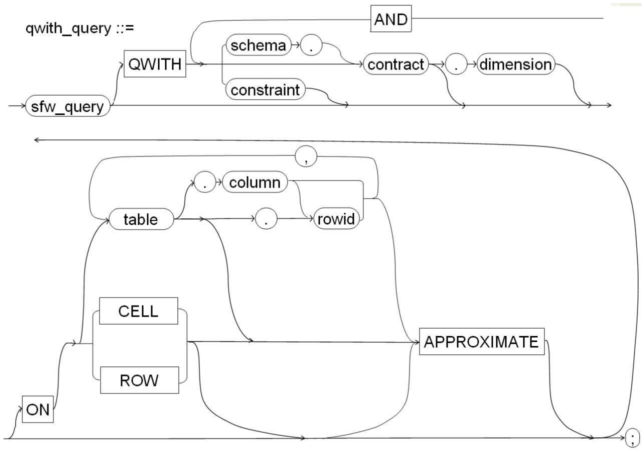


Figure 4: Syntax of QWITH queries

- *Exact checking (default).* Only DB object instances that exactly satisfy (with $\epsilon = 0$) the constraints declared in the contracts are used in the query processing for building the query result,
- *Approximate checking.* DB object rows and cells that satisfy approximately (with non null ϵ) the constraints defined in the contracts of the QWITH query will be considered for elaborating the query result.

Consider the query that retrieves all the products whose PRICE is greater than $10 in CRM_DB. QoD metadata of PRODUCT table involved in this query have been defined by means of the quality contract types and instances previously declared in Tables 2 and 3. A *quality-blind* query would return three rows: $P1$, $P2$ and $P3$. Different quality-aware queries are given in Tables 5 and 6 based on the declared quality contracts that check data freshness, accuracy, and completeness at the ROW and CELL granularity levels. Results are presented in both EXACT and APPROXIMATE modes along with the acceptability values of the result for each QoD dimension considered in the QWITH part of the query. These queries lead to different results depending on the constraints required on the chosen QoD dimensions.

In the EXACT mode, only DB object rows and cells that satisfy exactly (with $\epsilon = 0$) all the constraints defined in the contracts invoked in the QWITH query will be considered for elaborating the query result. For Q1 query, the contract `fresh` is defined on two granularity levels (ROW, TABLE) as given in Table 2. The constraints are checked on data freshness for each cell and each row involved in the SFW query processing. Probabilities resulting from the checking are given in the fourth column of the tables. In the EXACT constraint checking mode, only the rows and cells involved in the query with null probability will be considered for building the query result. Consequently, $P1$ row will be excluded ($Pr_{fresh}(ROW('P1')) = .42$ in Table 4) whereas in the APPROXIMATE mode, $P1$ row will be included in the result ($.42 + .15 > .50$).

For Q2, the query result will not include product $P3$ in the EXACT mode and also in the APPROXIMATE mode as well because its PRICE value ('7777') has a high probability of being inaccurate with respect to the contract `accurate` applied to CELL granularity level ($Pr_{accurate}(CELL('7777')) = .35$ in Table 4). The result presentation includes the probabilities of every cell returned in the result.

For Q3 constraining freshness and completeness on rows and accuracy on cells, only $P2$ satisfies the constraints and the query result will exclude $P1$ and $P3$ in the EXACT mode but it will include $P1$ in the APPROXIMATE mode.

Consider a query that retrieves the list of product identifiers and the city of their purchaser, PROD_ID and CUST_CITY joining PRODUCT and CUSTOMER tables on CUST_ID field. A

	CELL	**ROW**		**COLUMN**		**TABLE**
Acceptability	**7777**	$P1$	$P4$	**CUST_ID**	**ORDER_DATE**	**PRODUCT**
P_{fresh}	-	.42	0	0	-	.56
$P_{accurate}$	.35	-	-	.06	-	-
$P_{complete}$	-	0	.286	.25	.25	.071

Table 4: Acceptability Values per QoD Dimension in PRODUCT tables of CRM_DB

Q#	**Query**	**Results**			
		Exact		**Approximate ($\epsilon = .15$)**	

Q1	SELECT PROD_ID, PRICE
	FROM PRODUCT
	WHERE PRICE > 10
	QWITH fresh ON ROW;
		P2,77	$P_{fresh}(ROW(\text{P2})) = 0$	P1,77	$P_{fresh}(ROW(\text{P1})) = .42$

		P3,7777	$P_{fresh}(ROW(\text{P3})) = 0$	P2,77	$P_{fresh}(ROW(\text{P2})) = 0$
				P3,7777	$P_{fresh}(ROW(\text{P3})) = 0$

Q2	SELECT PROD_ID, PRICE
	FROM PRODUCT
	WHERE PRICE > 10
	QWITH accurate ON CELL;
		P1,77	$P_{accurate}(CELL(\text{P1})) = 0$
			$P_{accurate}(CELL(77)) = 0$
				P1,77	$P_{accurate}(CELL(\text{P1})) = 0$
					$P_{accurate}(CELL(77)) = 0$

		P2,77:	$P_{accurate}(CELL(\text{P2})) = 0$ $P_{accurate}(CELL(77)) = 0$	P2,77:	$P_{accurate}(CELL(\text{P2})) = 0$ $P_{accurate}(CELL(77)) = 0$

Q3	SELECT PROD_ID, PRICE
	FROM PRODUCT
	WHERE PRICE > 10
	QWITH fresh ON ROW
	AND accurate ON CELL;
	AND complete ON ROW;
		P2,77	$P_{fresh}(ROW(\text{P2}))) = 0$
			$P_{accurate}(CELL(\text{P2})) = 0$
			$P_{accurate}(CELL(77)) = 0$
			$P_{complete}(ROW(\text{P2})) = 0$
				P1,77	$P_{fresh}(ROW(\text{P1})) = .42$
					$P_{accurate}(CELL(\text{P1})) = 0$
					$P_{accurate}(CELL(77)) = 0$
					$P_{complete}(ROW(\text{P1})) = 0$

				P2,77	$P_{fresh}(ROW(\text{P2}))) = 0$ $P_{accurate}(CELL(\text{P2})) = 0$ $P_{accurate}(CELL(77)) = 0$ $P_{complete}(ROW(\text{P2})) = 0$

Table 5: Examples of simple QWITH Queries on PRODUCT table

quality-blind query would return $P1$, $P2$, and $P3$ PROD_IDs. Different join quality-aware queries may be formulated with respect to the previous quality contracts on freshness, accuracy, and completeness.

Suppose that all the acceptability values of CUSTOMER table are null except for cell $C2$ and its associated row such as $P_{fresh}(CELL('\text{C2}')) = .6$, $P_{accurate}(CELL('\text{C2}')) = .2$ and $P_{complete}(ROW('\text{C2}')) = .5$.

For Q4 in the EXACT mode, the join of PRODUCT and CUSTOMER tables only consider the joining cells that have a null probability of being inaccurate with respect to the contract `accurate`, then $P2-C2$ and $P3-C2$ joins will be excluded from the final join result because the probability of $C2$ cell from CUSTOMER table is $Pr_{accurate}(ROW('C2') = .2$ with respect to contract `accurate`. But the in the APPROXIMATE mode, this cell will be considered for joining the tables and leads to a result including $P2-C2$ and $P3-C2$ rows.

For Q5, the join of PRODUCT and CUSTOMER tables only consider the rows of both tables having null probability with respect to the contracts `fresh` and `complete`. $P1$ will then be rejected in the EXACT mode but included in the APPROXIMATE mode ($Pr_{fresh}(ROW('P1') = .42 < .50$ in Table 4).

4. RELATED WORK

In Data Quality Research, several contributions have been proposed for expressing in a simply and declarative way constraints on data quality with extending the query languages. Different approaches have used data quality indicators and metadata for selecting the best source [15] or the best query plans to execute [18] in a distributed environment. As the first prototype, Q-Data [21] checks if the existing data is correct and ensures data validation and cleanup by using a logical database language (LDL++). More recently, Guo et al. [11] have proposed a model for expressing currency and consistency constraints (C&C) in the queries on replicated and cached data by the means of a new clause on data currency that extends SQL. DQ^2L (*Data Quality Query Language*) [14] was designed to query relational data supporting a data quality aware query processing framework. In the context of quality-driven query processing, Braumandl et al. [8] propose to take into account data quality estimates when evaluating the user's query and deciding the best manner of carrying out the query (which sources to reach, which server to use, etc). TriQL[3], the query language of the Trio project [16] is an extension of SQL including three built-in predicates for checking confidence, accuracy and lineage of queried data and extending traditional data management.

All these approaches have proposed an extension of SQL query language in order to include in different ways constraints on specific and fixed dimensions of the quality of data mainly for query answering. They may suffer from the drawback that they are neither flexible nor modular from the user perspective in the way that QoD is evaluated by hard-coded functions. In our approach, an extendable library of functions can be used to define and compute QoD measures. The composition of functions is required through the design of analytic workflows that can be set up by the user to automatically check and enforce various aspects of data quality particularly important for his/her specific needs and goals. In terms of development, XQuaL prototype is currently being implemented on Eclipse Platform 3.3.2 (JRE1.5.0_12) using SQLExplorer Plug-in 3.5.0 and based on Kepler[4][7] that is built upon Ptolemy II framework[5] developed at the University of California, Berkeley. Kepler is a scientific workflow management system that allows specifying and executing data-intensive analysis. A workflow in Kepler can be graphically designed by chaining together tasks, where each task may take input data from previous tasks, parameter settings, and data coming from external data sources. The contribution of the XQuaL project in terms of development concerns: *i)* the implementation of the extended QWITH query engine, *ii)* the development of the library of functions for QoD evaluation, *iii)* a QoD contract editor invoking C, SAS or PL/SQL functions extending the design of Kepler workflows for QoD analysis, *iv)* the coupling of the QWITH query engine with the workflows and the specification of contract types and instances, and *v)* the development of a QoD metadata repository for storing QoD measures.

[3] TriQL : http://infolab.stanford.edu/widom/triql.html

[4] Kepler project: http://www.kepler-project.org/

[5] Ptolemy II: http://ptolemy.eecs.berkeley.edu/ptolemyII/

Query#	Query	Results			
		Exact		Approximate ($\epsilon = .15$)	

Q4	SELECT PROD_ID,
	CUST_CITY
	FROM PRODUCT P,
	CUSTOMER C
	WHERE
	P.CUST_ID=C.CUST_ID
	QWITH accurate ON CELL;
		P1,Camden UK	$P_{accurate}(CELL(\text{P1})) = 0$
			$P_{accurate}(CELL(\text{Camden UK})) = 0$
				P1,Camden UK	$P_{accurate}(ROW(\text{P1})) = 0$
					$P_{accurate}(ROW(\text{Camden UK})) = 0$

				P2,Camden NJ US	$P_{accurate}(CELL(\text{P2})) = 0$ $P_{accurate}(CELL(\text{Camden NJ US})) = 0$
				P3,Camden NJ US	$P_{accurate}(CELL(\text{P3})) = 0$ $P_{accurate}(CELL(\text{Camden NJ US})) = 0$

Q5	SELECT PROD_ID,
	CUST_CITY
	FROM PRODUCT P,
	CUSTOMER C
	WHERE
	P.CUST_ID=C.CUST_ID
	QWITH fresh ON ROW
	AND complete ON ROW;
		P2,Camden NJ US	$P_{fresh}(ROW(\text{P2})) = 0$
			$P_{complete}(ROW(\text{P2})) = 0$
				P1,Camden UK	$P_{fresh}(ROW('\text{P1}')) = .42$
					$P_{complete}(ROW(\text{P1})) = 0$

		P3,Camden NJ US	$P_{fresh}(ROW(\text{P3})) = 0$ $P_{complete}(ROW(\text{P3})) = 0$	P2,Camden NJ US	$P_{fresh}(ROW(\text{P2})) = 0$ $P_{complete}(ROW(\text{P2})) = 0$
				P3,Camden NJ US	$P_{fresh}(ROW(\text{P3})) = 0$ $P_{complete}(ROW(\text{P3})) = 0$

Table 6: Examples of QWITH JOIN Queries

5. CONCLUSION AND PERSPECTIVES

Since, realistically, it is difficult to evaluate with certainty the quality of data in a large database, we propose an analytical and probabilistic approach that allows the evaluation of the quality of database object instances and takes into account this evaluation for elaborating and presenting alternative query results depending on constraints on data quality. Since it is also important to accommodate *ad-hoc* queries with taking into account possible data quality requirements, which of course, *a priori* are unknown, our objective is to evaluate data quality in the context of known uses (e.g., at the query time). In this paper, we address this by proposing a complete approach based on the use of various functions, statistics, and data mining techniques that are combined into analytic workflows dedicated to QoD evaluation for quality-aware query processing and QoD diagnostics. Analytical functions generate measures that are intended to characterize user-defined dimensions of QoD. These measures are stored as metadata in a repository. They are checked with respect to user-defined constraints that are associated to the database object instances and defined by means of contracts. Several QoD measures may be associated to one QoD dimension. A scoring function is used to compute the probability that for a given DB object instance, a QoD dimension is acceptable with respect to a quality contract that defines the set of constraints on its associated QoD measures.

QoD evaluation can be based on the results of typical data-centric analysis. They require a sequence of activities and components for data retrieval computation, and presentation, assembled together into a single executable data analysis pipeline. The components may be part of the data management system, part of another application (invoked through system calls, executing SQL or SAS scripts, etc.). In addition to providing users with a mechanism to define and compose themselves QoD analysis tasks, our long-term research perspectives are to design and support end-to-end analytic workflows for QoD evaluation, e.g., through tools for accessing external resources (data sources or functions), archival of metadata (QoD measures and descriptive metadata), optimizing and monitoring of QoD analytic workflow execution. In this context, we also want to propose various alternatives for assisting the user in the design of QoD analytic workflows depending on resource limitations.

Since QWITH query optimization is our ongoing work, the interesting issues concerning: *i)* the heuristics we've defined for building the algebraic annotated QWITH query trees, *ii)* the techniques for query rewriting, *iii)* the cost model for QWITH query processing, and *iv)* the algorithms we propose for relaxing the constraints of QWITH queries, will remain out of the scope of this paper. These important aspects constitute our immediate perspectives of research and development for improving the QWITH query engine. As our current work, this includes more precisely the elaboration of reasonable solutions for the main following issues: *i)* the on-line computation of analytic functions, *ii)* the checking of dual or overlapping constraints in the sense that they partition the data set into mutual exclusive or inclusive subsets when answering data quality-constrained queries, and *iii)* the automatic relaxation of the constraints (determination of ϵ) when no data satisfies the constraints on data quality whereas a query result is still possible.

6. REFERENCES

[1] J. Akoka, L. Berti-Équille, O. Boucelma, M. Bouzeghoub, I. Comyn-Wattiau, M. Cosquer, V. Goasdoué-Thion, Z. Kedad, S. Nugier, V. Peralta, and S. Sisaïd-Cherfi. A Framework for Quality Evaluation in Data Integration Systems. In *Proc. of the 9th Intl. Conf. on Enterprise Information Systems, ICEIS 2007*, pages 170–175, Funchal, Madeira, Portugal, June 12-16, 2007.

[2] P. Andritsos, A. Fuxman, and R. J. Miller. Clean Answers over Dirty Databases: A Probabilistic Approach. In *Proc. of the 22nd Intl. Conf. on Data Engineering, ICDE 2006*, page 30, Atlanta, GA, USA, April 3-8, 2006.

[3] C. Batini and M. Scannapieco. *Data Quality: Concepts, Methodologies and Techniques.* Data-Centric Systems and Applications. Springer-Verlag, 2006.

[4] O. Benjelloun, A. D. Sarma, A. Y. Halevy, M. Theobald, and J. Widom. Databases with uncertainty and lineage. *VLDB J.*, 17(2):243–264, 2008.

[5] L. Berti-Équille. Quality Awareness for Data Management and Mining. Habilitation à Diriger des Recherches, Université de Rennes 1, France, June 2007, [available online].

[6] M. Bouzeghoub and V. Peralta. A Framework for Analysis of Data Freshness. In *Proc. of the 1st Intl. ACM SIGMOD 2004 Workshop on Information Quality in Information Systems, IQIS 2004*, pages 59–67, Paris, France, June 18, 2004.

[7] S. Bowers and B. Ludäscher. Actor-Oriented Design of Scientific Workflows. In *Proc. of the 24th Intl. Conf. on Conceptual Modeling, ER 2005*, volume 3716 of *Lecture Notes in Computer Science*, pages 369–384, Klagenfurt, Austria, October 24-28, 2005.

[8] R. Braumandl, M. Keidl, A. Kemper, D. Kossmann, S. Seltzsam, and K. Stocker. ObjectGlobe: Open Distributed Query Processing Services on the Internet. *IEEE Data Eng. Bull.*, 24(1):64–70, 2001.

[9] N. N. Dalvi and D. Suciu. Management of Probabilistic Data: Foundations and Challenges. In *Proc. of the 26th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 1–12, Beijing, China, June 11-13, 2007.

[10] T. Dasu and T. Johnson. *Exploratory Data Mining and Data Cleaning.* John Wiley, 2003.

[11] H. Guo, P.-Å. Larson, and R. Ramakrishnan. Caching with 'Good Enough' Currency, Consistency, and Completeness. In *Proc. of the 31st Intl. Conf. on Very Large Data Bases, VLDB 2005*, pages 457–468, Trondheim, Norway, August 30 - September 2, 2005.

[12] M. A. Hernández and S. J. Stolfo. Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem. *Data Min. Knowl. Discov.*, 2(1):9–37, 1998.

[13] D. Lembo, M. Lenzerini, and R. Rosati. Source Inconsistency and Incompleteness in Data Integration. In *Proc. of the 9th Intl. Workshop on Knowledge Representation meets Databases, KRDB 2002*, volume 54, Toulouse, France, April 19-21, 2002.

[14] S. d. F. Mendès-Sampaio, C. Dong, and P. Sampaio. Incorporating the Timeliness Quality Dimension in Internet Query Systems. In *Proc. of the Intl. Workshop on Web Information Systems Engineering, WISE 2005*, pages 53–62, New York, NY, USA, November 20-22 2005.

[15] G. A. Mihaila, L. Raschid, and M.-E. Vidal. Source Selection and Ranking in the WebSemantics Architecture: Using Quality of Data Metadata. *Advances in Computers*, 55:89–119, 2001.

[16] M. Mutsuzaki, M. Theobald, A. de Keijzer, J. Widom, P. Agrawal, O. Benjelloun, A. D. Sarma, R. Murthy, and T. Sugihara. Trio-One: Layering Uncertainty and Lineage on a Conventional DBMS (Demo). In *Proc. of 3rd Biennial Conf. on Innovative Data Systems Research*, pages 269–274, Asilomar, CA, USA, January 7-10, 2007.

[17] F. Naumann, J. C. Freytag, and U. Leser. Completeness of Integrated Information Sources. *Inf. Syst.*, 29(7):583–615, 2004.

[18] F. Naumann, U. Leser, and J. C. Freytag. Quality-Driven Integration of Heterogenous Information Systems. In *Proc. of the 25th Intl. Conf. on Very Large Data Bases, VLDB 1999*, pages 447–458, Edinburgh, Scotland, UK, September 7-10, 1999.

[19] S. Parsons. Current Approaches to Handling Imperfect Information in Data and Knowledge Bases. *IEEE Trans. Knowl. Data Eng.*, 8(3):353–372, 1996.

[20] V. Peralta. *Data Quality Evaluation in Data Integration Systems.* PhD thesis, Université de Versailles, France & Universidad de la República, Uruguay, 2006.

[21] A. P. Sheth, C. Wood, and V. Kashyap. Q-Data: Using Deductive Database Technology to Improve Data Quality. In *Proc. of the Intl. Workshop on Programming with Logic Databases, ILPS*, pages 23–56, Vancouver, BC, Canada, October 26-29, 1993.

[22] M. Weis, F. Naumann, and F. Brosy. A Duplicate Detection Benchmark for XML (and Relational) Data. In *Proc. of the 3rd Intl. ACM SIGMOD 2006 Workshop on Information Quality in Information Systems, IQIS 2006*, Chicago, IL, USA, June 30, 2006.

[23] W. E. Winkler. Overview of Record Linkage and Current Research Directions. Technical report, Statistical Research Division, U.S. Census Bureau, February 2006.