HABILITATION À DIRIGER DES RECHERCHES

présentée devant

L'Université de Rennes 1
Spécialité : Informatique

par

Laure Berti-Équille

Quality Awareness for Managing and Mining Data

soutenue le 25 Juin 2007 devant le jury composé de :

Prof.	Olivier Ridoux	Président
Prof.	Tiziana Catarci	Rapporteur
Prof.	Mokrane Bouzeghoub	Rapporteur
Mme	Helena Galhardas	Rapporteur
Prof.	Jacky Akoka	Examinateur
Prof.	Henri Briand	Examinateur
Mme	Brigitte Trousse	Examinatrice

Acknowledgements - *Remerciements*

I would like to thank Tiziana Catarci, Helena Galhardas, and Mokrane Bouzeghoub for kindly accepting to serve as readers and reviewers for this dissertation of "Habilitation à Diriger des Recherches" and to participate to the jury.

Je remercie très chaleureusement Jacky Akoka, Henri Briand, Brigitte Trousse qui ont également accepté de participer à ce jury. Olivier Ridoux a présidé le jury de ma soutenance et je le remercie à ce titre.

Je le remercie également cette fois-ci en tant que directeur de l'IFSIC, comme je remercie son prédécesseur, Patrice Quinton, ainsi que Claude Labit, directeur de l'IRISA, qui, en 2004, ont été favorables à ma demande de délégation, me permettant ainsi d'être temporairement affranchie d'une tâche d'enseignement "chronophage" pour me consacrer entièrement à mon travail de recherche.

Je voudrais remercier toutes les personnes que j'ai pu cotoyer ces dernières années, en particulier, les personnels administratifs et services techniques de l'IRISA : les assistantes de projet, les ingénieurs et techniciens de l'atelier, le service des missions, le service administratif et financier, le service juridique, le service des relations extérieures et valorisation, ainsi que tous les autres services non cités ici qui facilitent admirablement le travail quotidien des (enseignants-)chercheurs de l'IRISA.

Travailler dans de telles conditions est une expérience exceptionnelle que j'ai su apprécier à sa juste valeur dès mon arrivée à l'IRISA.

Je tiens à remercier également mes collègues de l'ARA Masses de données QUADRIS pour nos échanges constructifs sur notre thème de la qualité des données et des systèmes d'information.

Finally, I thank my European and international colleagues for their stimulating collaborations, their enthusiasm and dynamism for promoting Data Quality Research.

Contents

Introduction

General Context

The maturity of database and web technologies has encouraged users to make data publicly available in large quantities, opening up the possibility of large-scale searches and comparative analyses over multi-source data. However, such analyses are possible in only a small number of domains due to the practical difficulties involved in integrating and comparing data from separately designed databases. Different storage technologies are used and different ways of representing the same data are adopted. To add to the problem, many of the available data sources overlap in terms of their content or purpose, making it difficult for users: *i)* to select the most appropriate data sets for a specific analysis or decisions, *ii)* to have clear means of distinguishing the various providers of the data sets, *iii)* to evaluate objectively the quality of the provided data at a given time, and *iv)* to make value-for-money decisions about which data provider to contract with and which corrective actions on data to set up or prioritize.

This context is comparable to a "data market", an environment in which multiple providers offer the data sets required by a consumer, but in different ways, through different services, and with various degrees of quality and trust. The user must compare and balance the features of each data provider and the quality of the data sets, to select the best ones in terms of measurable factors such as cost of access, speed of access, reliability, etc. One of the important means of distinguishing between multiple data providers or data sources in such a "data market" is the quality of the data (QoD) provided. For instance, users may be prepared to pay more for access to data when they are confident that it is both correct and complete, or they may be prepared to sacrifice (say) the currency of the data if it costs less.

Data quality problems such as duplicates, misspelling errors, outliers, contradictions, inconsistencies, missing or incomplete data are omnipresent and widespread in every governmental, industrial, commercial and personal information systems. Face to alarming situations and considerable financial consequences of the decisions based on low quality information, theoretical and pragmatic approaches are urgently required.

In the past decade, data quality has received increasing attention and became one of the hot research topics at the convergence of several academic and industrial communities: Database, Information Technology, Statistics, Knowledge Engineering and Discovery.

Many processes and applications - e.g., information system integration, information retrieval, and knowledge discovery from databases (KDD) - require various forms of data preparation and correction with several complex data processing techniques, because the data input to the application-specific algorithms is assumed to conform to "nice" data distributions, containing no missing, inconsistent or incorrect values. This leaves a large gap between the available "dirty" data and the available machinery to process the data for application purposes.

More and more systems have to integrate data coming from multiple data sources and provide the users with a uniform access to data. These systems, called multi-source information systems (MSIS), as illustrated in Figure 1, can be of several kinds:

- data integration systems (DIS) including virtual mediation systems (VMS), materialized integration systems (MIS), data warehouses or webhouses (DW), and cooperative information systems (CIS),
- replication systems (RS), and
- peer-to-peer database systems(P2P).

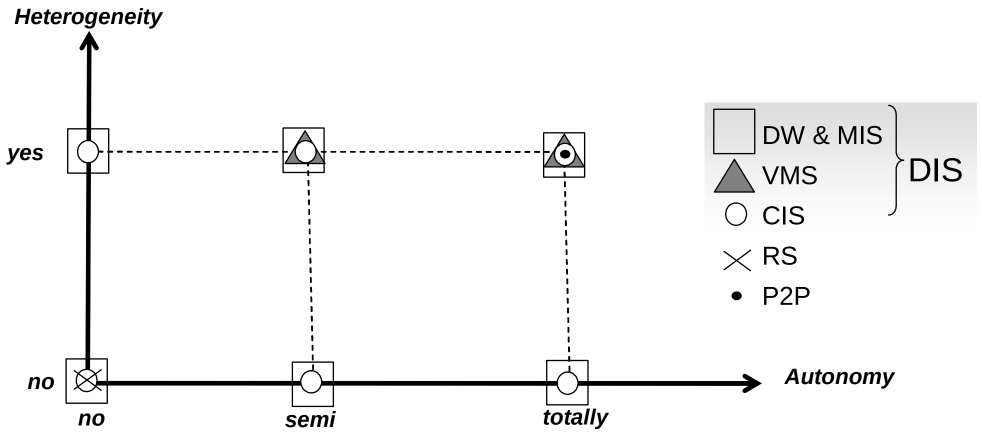


Figure 1: Data Sources Characteristics for MSISs

In the context of data integration systems, where data comes from various information sources, data quality issues grow in complexity. Maintaining traceability, freshness, non-duplication and consistency of very large data volumes for integration purposes is one of the major scientific and technological challenges today for research communities in information systems (IS) and databases (DB). Consequently, data quality consideration has to be brought to the center of IS development process.

From a technical perspective, data quality problems in data management systems (MSIS and monolithic centralized databases) have various causes. Let's just mention some of them together with the current scientific and technological challenges of Data Quality Research:

- During conceptual data modeling, the definitions of the database schema and of the attributes may have been insufficiently well-structured or standardized; when the conceptual database schema has not been validated or when some integrity constraints, triggers or stored procedures are missing for maintaining data consistency. This case is prevalent for legacy systems still in use in many organizations and in MSISs when schema mapping is perilous,
- During system and application development: requirements may have been incompletely specified, requirements may change and errors can be easily introduced during the design and development processes,
- When information gathering methods were not well specified or designed; systematic checking and measurement procedures, error detection techniques and duplicate elimination heuristics are missing or inappropriate,
- During the data collection process, because the technical equipments (e.g., sensors), or the survey methods are not accurate enough for the specifications, causing non stationary imprecision and incompleteness on produced raw data,
- When the companies do not have the software resources or manpower, neither for tracking the age of their data, nor for detecting the errors, updating, enriching their data, and eliminating obsolete and erroneous data,
- During integration or post-integration of several heterogeneous information sources: multi-source data can be overlapping, contradictory or inconsistent. Record linkage heuristics and integration techniques may be inappropriate. Each system may have different, changing and locally non homogeneous data quality depending on the considered dimension (e.g., a data source may provide very accurate data for a specific domain and low accuracy for another one, another source may offer fresher data but not as accurate as the latter one depending on the domain of interest),
- When data interpretations are inconsistent, in particular because of national or cultural differences for the usage of certain codes or symbols,
- During system migration: the conversion programs may introduce new errors. Reverse-engineering may not consider (or may lose) the whole (or some parts of the) context of definition, production or usage of data,
- When data are replicated asynchronously on various sites (e.g., in Web portals or P2P systems) and the secondary copies are not updated in conformance with the primary copy, etc.

One can bet that the great diversity of these problems will interest many of our colleagues both from R&D and academic IS and DB communities for still a long time. Data quality problems are complex, morphing and multi-dimensional. They have to be addressed from the very first starting steps of the information systems and database design to the final outputs of the decisional Information Supply Chain (ISC).

Contributions

The contributions of my work belong to the following research fields:

- *Data Mining and Knowledge Discovery*: I've adapted and used exploratory data mining techniques for the detection of data quality problems and anomaly patterns in large data sets and the measurement of various aspects of data quality dimensions,

- *Database and Information Systems engineering*, with a special focus on the data integration process and the design of multi-source information system, in particular, data warehouse and virtual data mediation systems. My contributions mainly address the following aspects of data quality management:

  i) Management of metadata that are measures, summaries, and sketches obtained from exploratory statistical techniques characterizing certain aspects of the quality of data,

  ii) Query processing with the proposition of a query language extension enabling the declaration of metrics and constraints on the various dimensions of data quality.

My research work is driven by the main following questions:

- How to adapt and use data mining techniques (such as clustering and association rule discovery) to implement data quality introspection and self-administration of data management systems?

- How to design or re-engineer the core of the data management system in order to integrate systematic data quality controls both on the data stored and on the data that have to be integrated and loaded?

- How to ensure guarantees on the quality of results with minimal costs when querying and mining data?

- How to quantify the impact of low data quality on the results of data mining and knowledge discovery processes?

- How to integrate QoD metadata for evaluating the quality of mining results?

My long-term objectives are to propose theoretically founded solutions for systematic evaluation and self-control of data quality in data management systems. These objectives are parts of a more general methodology I've adopted since the last decade. The methodology is organized as a matrix covering data quality problematics with four complementary approaches:

$\boxed{\textbf{A1}}$ **The preventive approach** focused on system-centric design and engineering for enabling the continuous self-control of the quality of data in data management systems.

$\boxed{\textbf{A2}}$ **The diagnostic approach** focused on statistical editing and data mining techniques for effective and efficient detection of anomalies in massive data sets (e.g., outliers, duplicates, inconsistencies, dubious data).

$\boxed{\textbf{A3}}$ **The cost-constrained approach** focused on cost-based prediction models to find optimal solutions ensuring the trade-off between result quality and cost of any data processing activity (e.g., querying and mining data).

$\boxed{\textbf{A4}}$ **The adaptive approach** focused on the extension of a query language and the optimization and adaptive processing of data quality-constrained queries.

The application of some of my contributions has been conducted on different data types, as shown in Figure 2: structured, semi-structured, object-oriented and stream data, and applied to different application domains, namely:

$\boxed{\textbf{D1}}$ Integration of XML biomedical data into an object-oriented data warehousing system,

$\boxed{\textbf{D2}}$ Mediation of relational CRM data for Business Intelligence applications,

$\boxed{\textbf{D3}}$ Monitoring Stream data in Telecom applications.

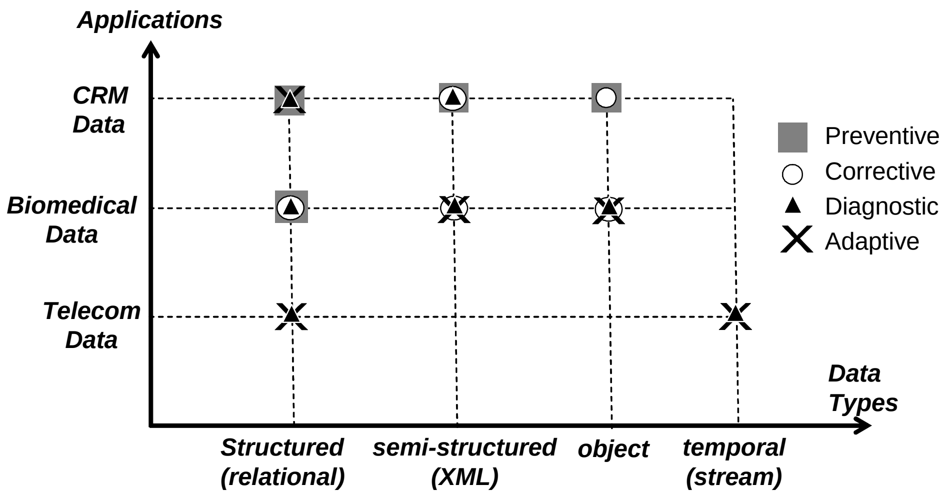


Figure 2: Applications and Data Types Coverage

Beyond the defense of the "*Habilitation à Diriger des Recherches (HDR)*", a wish would be that this dissertation may be useful for one- or two-semester courses on data quality in advanced undergraduate or graduate courses on database management and data warehouse design that are usually required in Information System, Business IT, or Computer Science curriculum.

Outline

The rest of the dissertation is organized as follows:

Chapter 1 **Survey and Recent Advances on Data Quality Research** introduces the field of Data Quality Research, examining the main data quality requirements such as uniqueness, consistency, certainty, completeness and freshness of data. This chapter provides a review of recent research contributions that propose measures, algorithms, languages, tools and models to cope with specific data quality problems. The chapter also presents the related research projects and highlights new research directions in the field.

Chapter 2 **Quality-Aware Data Management** presents an approach for designing analytic workflows for QoD evaluation where analytical functions compute measures characterizing various aspects of user-defined data quality dimensions. These measures are stored in a metadata repository whose underlying metamodel extending CWM metamodel is presented. A query language extension is also presented: it allows the declarative specification of data quality constraints that are taken into account in the query processing.

Chapter 3 **Data Quality-Aware Mining** proposes a generic framework for integrating data quality metadata into KDD processes and, in particular, it is focused on quality-aware association rule mining. A cost-based probabilistic model for selecting legitimately interesting rules is presented. Experiments have shown that variations on data quality have a great impact on the cost and quality of discovered association rules. This confirms our approach for the integrated management of data quality metadata into the KDD processes for ensuring the quality of data mining results.

Chapter 4 **Prototyping Data Quality Aware Applications** provides the brief descriptions of application scenarios and prototypes that have been developed by the students I supervised. These developments were based on different cases studies in the three application domains previously mentionned thanks to several ongoing collaborations with:

D1) the French public institute for biological, medical and public health research, INSERM (*Institut National de la Santé et de la Recherche Médicale*, Unit 522 - Rennes) involved in warehousing biomedical data related to liver pathologies,

D2) the French Company of Electricity Supply, EDF R&D, for the management of Customer Relationship Management (CRM) data in the QUADRIS project,

D3) the company GenieLog managing and monitoring stream Telecom data from Cegetel.

This chapter is consequently divided in three subsections respectively dedicated to quality-aware warehousing of biomedical data, quality-aware mediation of relational data, and quality monitoring for stream data.

The last chapter **Conclusions and Research Perspectives** concludes this dissertation by summarizing my contributions in the field of Data Quality Research and outlines my future work in this area.

Chapter 1

Survey and Recent Advances on Data Quality Research

Contents

1.1 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 to evaluate and check the quality of this data. While database technology has provided us with the basic tools for the efficient storage and lookup for large data sets, one of the big issues is how to help users, decision makers, data stewards, and DBAs understand, analyze data quality. Because taking the right decisions and triggering appropriately corrective actions on data capture / collection remain difficult tasks.

Maintaining a certain level of quality in a database is challenging and cannot be limited to one-shot approaches addressing (or separately circumscribing) simpler and more abstract versions of combined data quality problems. The commonly shared reality is that most databases simultaneously contain duplicated, inconsistent, imprecise, uncertain, incomplete, and outdated data.

A synthetic view of the main data quality problems with the current solutions proposed in the literature is given in Table 1.1.

Solving these problems often requires highly domain- and context-dependent information and together with human expertise (Dasu et al., 2003).

Classically, the database literature refers to data quality management as ensuring:

- *syntactic correctness*, e.g., constraints enforcement that prevent "*garbage data*" from being entered into the databases,

- *semantic correctness*, *i.e.*, data in the database that truthfully reflects the real-world entities and situations.

This traditional approach of data quality management has lead to techniques such as integrity constraints (Ramamritham & Chrysanthis, 1992), concurrency control, schema matching, and data integration for distributed and heterogeneous systems (Sayyadian et al., 2005). In this chapter, a broader vision of data quality management is presented with a database orientation and some incursions in Statistics for the description of relevant techniques of statistical data editing and anomalies detection.

Data quality is a "multidimensional, complex and morphing concept" (Dasu & Johnson, 2003). Since the 1990s decade, large bodies of research on information quality and data quality management have been initiated by several research communities: Statistics, Database, Information Technology, Machine Learning, Knowledge Engineering, Knowledge Management, and Discovery from Databases.

Numerous approaches and techniques have been proposed for modeling data quality dimensions, computing data quality indicators, proposing frameworks and methodologies for cleaning data and integrating voluminous, complex and heterogeneous data sets from large-scale distributed information systems (Batini & Scannapieco, 2006).

The classification of the various contributions in the field of Data Quality Research relies on three aspects:

Processing Step	**Data Quality Problems**	**Potential Solutions and References**
Data Creation, Capture Gathering Import	Manual entry, OCR Complex data type No standardized format / schema Duplicates Approximations Measurement Errors Hardware or software constraints Automatic massive data import	**Preemptive Approaches:** - Methodologies (English, 1999; Olson, 2003; Redman, 2001; Wang, 1998) (Wang et al., 2002) - Architectures for Data Quality Management (Ballou & Tayi, 1999) (Ballou & Tayi, 1989; Loshin, 2001) - Data audits, data stewardship **Retrospective and Corrective Approaches:** - Data Diagnosis : error and outliers (Breunig et al., 2000) (Knorr & Ng, 1998) - Data Cleaning: record linkage (Fellegi & Sunter, 1969), merge / purge problem (Hernández & Stolfo, 1998), object matching (Chaudhuri et al., 2003; Weis & Naumann, 2004), duplicate elimination (Ananthakrishna et al., 2002; Low et al., 2001) (Monge, 2000), citation matching (Bilenko & Mooney, 2003; McCallum et al., 2000), identity uncertainty (Pasula et al., 2002), entity identification (Lim et al., 1993), entity resolution (Benjelloun et al., 2005), approximate string join (Gravano et al., 2001), address and string matching (Navarro, 2001)
Data Delivery	Information destruction or mutilation by inappropriate pre-processing Data Loss: buffer overflows transmission problems No Checks	Data quality control (Liepins & Uppuluri, 1991) Data editing Data publishing Data aggregation Data squashing (DuMouchel et al., 1999) Use checksum Monitor data transmission, data integrity, data format Data mining techniques to check correctness of data transmissions
Data Storage	Metadata paucity and staleness Inappropriate data models Ad hoc Modifications HW / SW constraints	Metadata Management (Dasu et al., 2003; Mihaila et al., 2000) Plan ahead and customize for domain: Data Profiling, data browsing and monitoring (Dasu et al., 2002; Zhu & Shasha, 2002)
Data Integration	Multiple heterogeneous sources Time synchronization Atypical Data Legacy systems Sociological factors Ad-hoc Joins Random Matching Heuristics	Mandate accurate timestamps Data lineage (Cui & Widom, 2003) Data scrubbing Data profiling (Caruso et al., 2000) Commercial tools for data cleaning and migration Academic tools and language extensions for data cleaning: Potter's Wheel (Raman & Hellerstein, 2001), Bellman (Dasu et al., 2002), Arktos (Vassiliadis et al., 2001), Ajax (Galhardas et al., 2001) Febrl (Christen et al., 2004), and ClueMaker (Buechi et al., 2003) Quality-driven query processing (Naumann, 2002; Naumann et al., 1999) Academic tools for approximate join matching
Data Retrieval	Human errors Computational constraints, software limitations, incompatibility	Recall / Precision significance Feedback loop
Statistical Analysis and Data Mining	Issues of scale Performance, confidence guarantees Belief in black boxes and dart boards Attachment to a family of models Insufficient domain expertise Lack of familiarity with the data	Data Preparation for mining (Pearson, 2005; Pyle, 1999) Exploratory Data Mining - (EDM) (Dasu & Johnson, 2003) Greater accountability from analysts Continuous, ongoing analysis rather than one-shot solutions Sampling vs. full analysis Feedback loops

Table 1.1: Problems and Current Solutions for Data Quality Management

- *the scope*, *i.e.*, the quality of data instances, the quality of data models, the quality of processes or the quality of systems,
- *the system architecture*, *i.e.*, traditional and centralized database (DBMS), materialized integration (MIS) or data warehousing system (DW), virtual mediation system (VMS), cooperative information system (CIS), caching system (CS), and replication system (RS),
- *the level of abstraction* whose focus is respectively on:
    - the definition of data quality dimensions and metrics (Berenguer et al., 2005) depending on: *i)* the application context (Batini et al., 2004; Fox et al., 1994; Wang et al., 1995), *ii)* the audience type: practical (Redman, 2001), or more general (Kahn et al., 2002),
    - the design of database-oriented or statistical-based techniques for detecting anomalies (Winkler, 2004), transforming, cleaning data (Rahm & Do, 2000), and querying data with quality requirements (Naumann, 2002),
    - the proposition of (meta-)models (Santis et al., 2003), frameworks (Wang et al., 1995) and methodologies (English, 1999; Wang, 1998; Wang et al., 2002) for improving or assessing data quality in databases, in (cooperative) information systems or in data warehouse systems (Jarke et al., 1999).

Main classes of data quality problems addressed by DB researchers concern:

- **Duplicate and redundant data**. A wide range of techniques have been proposed for the "*merge/purge problem*" (Hernández & Stolfo, 1998), *i.e.*, the fusion and deletion of the multiple records that may describe the same real-world entity in order to keep one unique representative.
- **Imperfect data**. Inconsistency, imprecision, and uncertainty are some of the problems associated with data imperfection (Parsons, 1996). A significant amount of work in the areas of imprecise data management (Barga & Pu, 1993; Cheng et al., 2003; Dalvi & Suciu, 2004; Lazaridis & Mehrotra, 2004; McClean et al., 2001) and consistency constraints for measuring the internal validity and integrity of a database (Hou & Zhang, 1995) has been proposed in the literature.
- **Missing values and incomplete database**. The problem of handling incomplete information has been addressed in relational databases (Imielinski & Lipski, 1984) with emphasis on query rewriting to answer global queries on integrated information systems (Grahne, 2002; Naumann et al., 2004; Pang et al., 2005).
- **Stale data**. Refreshment techniques and synchronization policies influence the freshness of data and rely on the type of system (Peralta, 2006): data warehousing systems check the recentness of materialized views (Theodoratos &

Bouzeghoub, 2001; Zhuge et al., 1997); virtual mediation systems propose freshness guarantees for the query execution (Hull & Zhou, 1996); caching systems estimate the time-to-live of cached data before expiration and tune the caching policy, balancing response time and invalidation cycles for ensuring data currency (Cho & Garcia-Molina, 2000; Guo et al., 2005; Labrinidis & Roussopoulos, 2003; Li et al., 2003); replication systems manage the consistency of replicas in the presence of updates reducing the delay times of refresh transactions (Coulon et al., 2005; Olston & Widom, 2005).

Outline of the chapter. *In this chapter, we review the metrics and techniques proposed in the literature for the aforementioned data quality problems. Section 1.2 zooms in on the measures and algorithms respectively designed for: i) detecting and eliminating duplicate data, ii) handling inconsistent data, iii) managing imprecise or uncertain data, iv) handling missing or incomplete data, and v) improving data freshness. Section 1.3 describes tools and extensions to query languages that integrate data quality control and correction in data processing and management. Section 1.4 presents four recent related projects on data quality management. The last section of the chapter concludes and provides a panel of ongoing challenges in the field.*

1.2 Measures and Algorithms

1.2.1 Eliminating Duplicates

The principle of record linkage is to compare and bring together records from two (or more) sources that are believed to relate to the same real-world entity and whose (presumably overlapping) descriptions can be matched in such a way that they may be treated as a single record (Fellegi & Sunter, 1969).

These techniques are variously known as: record linkage (Fellegi & Sunter, 1969; Jaro, 1989; 1995; Monge & Elkan, 1996; Newcombe et al., 1959), object identification (Tejada et al., 2002), reference matching, reconciliation or disambiguation (Dong et al., 2005; Kalashnikov & Mehrotra, 2006; McCallum et al., 2000), duplicate elimination (Low et al., 2001), name disambiguation (Ananthakrishna et al., 2002; Bilenko & Mooney, 2003), entity resolution (Benjelloun et al., 2005), identity uncertainty (Pasula et al., 2002), fuzzy match (Chaudhuri et al., 2003), or approximate string join (Gravano et al., 2003; Navarro, 2001).

Record linkage is a necessary task in the context of data integration prior to warehousing, where data from distributed and heterogeneous data sources is combined, transformed and loaded. Deduplication of relational data received considerable attention in the DB community.

The existing methods can be classified depending on four aspects (Kalashnikov & Mehrotra, 2006), namely:

i) the goal and setting of the problem. Table 1.2.1 presents a simplified formalization of the various approaches in the field, *ii) the domain dependency*. Domain-specific solutions have been proposed for detecting duplicates in Census datasets (Winkler, 2004), medical data (Newcombe & Kennedy, 1962), genealogical data (Quass & Starkey, 2003) or bibliographic data (Bilenko & Mooney, 2003; McCallum et al., 2000). Domain-independent solutions include identity uncertainty (Pasula et al., 2002), entity identification (Mihaila et al., 2000), entity resolution (Benjelloun et al., 2005), or approximate string joins (Gravano et al., 2003).

iii) the type of similarity distance. Table A.1 given in Annexes (pages 150 and 151 presents several similarity distance functions currently used for string matching. In (Cohen et al., 2003), the authors surveys edit and common substring similarity metrics for name and record matching. They have compared the accuracy of these methods (including edit-distance like functions, as Levenshtein distance, Jaro metric, Jaro-Winkler, token-based distance functions, as TF-IDF, and hybrid string-distance functions). They concluded that the best-performing method is a hybrid scheme combining TF-IDF weighting scheme with the Jaro-Winkler string-distance scheme.

iv) the membership of a decision model class. Table 1.3 presents the classification of record linkage method proposed by (Batini & Scannapieco, 2006).

Notations	
Let $\mathscr{E} = \{e_1, e_2, \cdots, e_n\}$ be a set of real-world entities $\mathscr{T} = \{t_1, t_2, \cdots, t_m\}$ be the set of records (tuples) describing the real -world entities such as: a function *ref* exists and represents the 1-1 mapping between a real-world object and a tuple $ref(t_i) = e_j, \forall i = 1, \ldots, m$ and $j = 1, \ldots, n$. $Id(t_i, t'_i)$ is a Boolean function that returns true if t_i and t'_i values are identical. $Sim(t_i, t'_i)$ is a Boolean function that returns true if t_i and t'_i values are similar. $Merge(e_j, e'_j)$ is the inference function stating that e_j and e'_j are actually the same real-world entities.	
Problem	**Formulation**
Entity resolution or fusion	For given records $t_i, t'_i \in \mathscr{T}$, given entities $e_j, e'_j \in \mathscr{E}$, such as $ref(t_i) = e_j, ref(t'_i) = e'_j$, and $Sim(t_i, t'_i) = true$ then $Merge(e_j, e'_j)$.
Data reconciliation	If $ref(t_i) = ref(t'_i)$ and $Sim(t_i, t'_i) = true$, then Transform t_i such as $Id(t_i, t'_i)$ becomes true.
Data consolidation	For a given entity $e_j \in \mathscr{E}$, Find the set of records $\mathscr{T}_i = \{t_k \vert t_k \in \mathscr{T}, ref(t_k) = e_j\}$ and eventually proceed to data reconciliation.
Record linkage	For a given record t_i, Find the set of records $\mathscr{T}_i = \{t_k \vert t_k \in \mathscr{T}, ref(t_i) = ref(t_k) \wedge Sim(t_i, t_k) = true\}$.
Data disambiguation	For a given set of records $\mathscr{T}_i \subseteq \mathscr{T}$, Find the entity $e_j \in \mathscr{E}$ such as $ref(t_i) = e_j, t_i \in \mathscr{T}_i$.
Duplicate elimination	For a given subset $\mathscr{T}_i \subseteq \mathscr{T}$ of similar or identical records referring the same real-world entity, while $Card(\mathscr{T}_i) > 1$, for $t_i, t_j \in \mathscr{T}_i$ Proceed to data reconciliation and delete t_j from $\mathscr{T}_i$

Table 1.2: Problem Setting

Precise and unique categories of algorithms for entity resolution do not exist. Techniques successfully used range from well-known database-orientated techniques (such as approximate string join), statistical-based algorithms (such as classical inference and Bayesian methods), to *ad hoc* methods (such as templating and voting), or evolving techniques (such as adaptive neural networks) that are employed for entity fusion in signal processing and pattern recognition. Comparison of the complete range of techniques smacks a bit of an *apples and oranges* comparison, and any partitioning of algorithms into categories may be arguably arbitrary. Nevertheless, Figure 1.1 provides a conceptual taxonomy of the entity resolution techniques (not limited to DB domain) where two major categories are presented: feature-based inference techniques and cognitive-based models. Another taxonomy is proposed in the recent book of Batini and Scannapieco (Batini & Scannapieco, 2006).

Feature-based inference techniques are divided into two broad categories: (1) parametric techniques, which require *a priori* assumption about the statistical properties of the data (e.g., distributions), and (2) nonparametric techniques, which do not require *a priori* statistical information.

Parametric techniques include statistical techniques, classical inference, Bayesian inference, and the Dempster-Shafer method, as well as clustering methods. Nonparametric techniques include voting methods, correlation measure, entropy based techniques and similarity distance based algorithms.

Cognitive-based models are the second major category of entity resolution algorithms. These methods seek to mimic the inference processes of human analysts in recognizing entities. Techniques in this category include knowledge- or rule-based systems, fuzzy set theory, and logical templates. In one way or another, these methods are based on a perception of how humans process information to arrive at conclusions regarding identity of entities.

Most of record linkage methods proposed in the database literature belong to similar distance-based techniques with the following common steps (Batini & Scannapieco, 2006; Elmagarmid et al., 2007):

i) *Pre-processing* for coding, formatting and standardizing the data to compare

ii) *Selecting a blocking method* to reduce the search space by partitioning the datasets into mutually exclusive blocks to compare, e.g., with hashing, sorting keys, sorted nearest neighbors or windowing techniques over one or more keys (or attributes) (Baxter et al., 2003; Hernández & Stolfo, 1998).

iii) *Selecting and computing a comparison function*: this step consists of measuring the similarity distance between the pairs of records for string matching (Navarro, 2001). Many classes of similarity functions may be applied (see Table A.1 in Annexes, pages 150- 151):

  – *mono-attribute similarity*. Some are term-based such as TF-IDF measure or Jaccard coefficient, others are edit-based, such as Levenshtein distance, Soundex, Jaro, Jaro-Winkler, etc.

  – *multi-attribute similarity*. They are useful when relative importance of matching records exits along different attributes highly domain-dependent.

iv) *Selecting a decision model and validation of the method*: this step consists of assigning and classifying pairs of records as matching, non-matching or potentially matching records with a method that can be probabilistic (with or

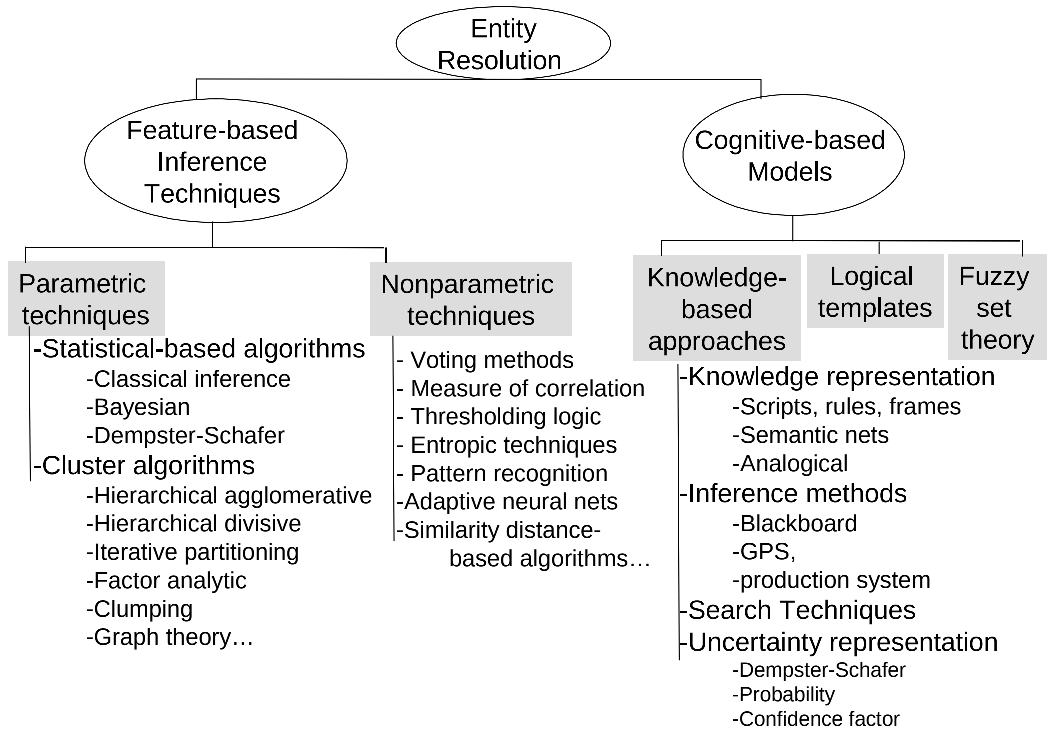


Figure 1.1: Taxonomy of Existing Techniques for Entity Resolution

without training datasets), knowledge-based or empirical. Table 1.3 presents the methods and tools proposed in the literature. The methods may be finally evaluated with recall, precision and F-measure on *ad hoc* data sets (Batini & Scannapieco, 2006).

We distinguish three categories of proposals in the diversity of similarity distance-based record linkage methods, namely: *i)* the approaches that are based on user-specified threshold, *ii)* the approaches based on threshold learning, and *iii)* the approaches based on graph and partitioning.

1.2.1.1 User-Defined Thresholds

The problem of identifying duplicate records in databases was originally identified by Newcombe *et al.* in 1959 (Newcombe et al., 1959) as record linkage on medical records for identifying the same individual over different time periods. The seminal paper of Fellegi and Sunter (Fellegi & Sunter, 1969) proposed a formal theory for *probabilistic record linkage* and offered a statistical method for estimating matching parameters and error rates. In Fellegi and Sunter model (FS), object identification is viewed as a classification problem. Previous work on object identification has either employed manual *ad hoc* methods to customize rules or transformations

Method *(System)*	**Authors**	**Model**
Error-based Model Expectation Maximization based Method Induction Clustering for Record Linkage *(Tailor)* 1-1 matching and Bridging File	(Fellegi & Sunter, 1969) (Dempster et al., 1977) (Bilenko & Mooney, 2003) (Elfeky et al., 2002) (Winkler, 2004)	Probabilistic
Sorted-Nearest Neighbors method XML Object Matching Hierarchical Structure *(Delphi)* Matching Prediction *(ClueMaker)*	(Hernández & Stolfo, 1998) (Weis & Naumann, 2004) (Ananthakrishna et al., 2002) (Buechi et al., 2003)	Empirical
Functional Dependencies Inference Transformation function *(Active Atlas)* Rules and sortednearest neighbors *(Intelliclean)*	(Lim et al., 1993) (Tejada et al., 2001) (Low et al., 2001)	Knowledge -based

Table 1.3: Decision Models for Handling Duplicates (Batini & Scannapieco, 2006)

for specific domains or has required the user to specify a fixed threshold to determine which objects are considered mapped together with a subsequent heavy user interaction for achieving high accuracy mapping.

In most applications, the edit distance model is derived by heuristics means, possibly including data-dependent tuning of parameters. For example, (Monge & Elkan, 1996) recognize duplicate corrupted records using an edit distance with tunable edit and gap costs. Among the empirical approaches, Hernández and Stolfo Hernández & Stolfo (1998) developed a windowing strategy, which sorts a relation on a key attribute and compares all records within the sliding window on the sorted order. This sorted neighborhood method is based on the use of domain-specific edit distance for limiting the number of potential duplicate pairs. This and other traditional approaches use a similarity measure with user-defined threshold to compare tuples' attribute values; tuples with similarity scores above a certain user-defined threshold are declared to be matches.

Ananthakrishna et al. (2002) exploit hierarchies on dimensional tables and use significant co-occurrences through other relations that exhibit equivalence errors and duplicates due to different representations of the same logical value (e.g., non unique and non standardized abbreviations), to resolve whether two entities are duplicates, they check for co-occurrence in the children sets of the entities.

With Q-gram set join, (Gravano et al., 2001) proposed an algorithm for approximate string join, which in principle can be adapted to detect duplicate records. The edit distance function is used to measure the closeness between tuples.

Fuzzy Match Similarity (*fms*) proposed by Chaudhuri et al. (2003) views a string as a sequence of tokens with weighted importance which is quantified by *inverse document frequency* (IDF). The goal is to approximately match erroneous input tuples with clean tuples from a reference relation. The similarity between two tuples depends on the minimum cost of token transformations (insertion, deletion, replacement) from one tuple to the other. The cost of each edit transformation is a function of the weights of tokens involved. An *error tolerant index (ETI)* comparable to a q-gram table is built from the reference relation from which a subset of all q-grams per tuple is probabilistically selected. Given a user-specified minimum similarity threshold, the algorithm efficiently retrieves the K reference tuples closest to the input tuple, according to the *fms* function.

Chaudhuri et al. (2006) recently proposed a primitive operator *SSJoin* which can be used as a foundation to implement similarity joins according to a variety of popular string similarity functions, and notions of similarity which go beyond textual similarity.

This approach as the previous ones imposes the user the non trivial task to define appropriate thresholds and set of parameters for setting up the record linkage method.

1.2.1.2 Learnt Similarity Thresholds

Work related to duplicate elimination can be classified into supervised and unsupervised approaches. Supervised approaches learn rules and patterns characterizing pairs of matching records from training data sets with known duplicates (Bilenko & Mooney, 2003; Sarawagi & Kirpal, 2004; Tejada et al., 2002) or with interactive user guidance. Supervised learning has been used for learning the parameters of string-edit distance metrics (Bilenko & Mooney, 2003; Ristad & Yianilos, 1998) applying a stochastic model for pattern recognition, and combining the results of different distance functions (Tejada et al., 2001).

While the basic edit distance models and algorithms are expressed in terms of single letter edits, in practice it is convenient to use a richer application specific set of edit operations, e.g. (Tejada et al., 2001; 2002) propose edit operations such as abbreviations and acronyms for record linkage.

Lots of approaches described to solve the problem of object identification are actually variants of the original FS model, typically based on logistic regression. A separate match decision is made for each candidate pair, followed by transitive closure to eliminate inconsistencies. For example, Winkler & Thibaudeau (1991) built upon the work of Fellegi and Sunter a probabilistic approach with a latent match variable which is estimated using Expectation-Maximization (EM). EM with an appropriate version of the forward-backward algorithm can be used to learn parameters that maximize the likelihood of a given training set of pairs of strings (Ristad & Yianilos, 1998).

Face to the problem of set containment joins, *Probe-Cluster* (Sarawagi & Kirpal, 2004) is an algorithm for joining set-valued data based on various thresholded similarity metrics, namely overlap set size, Jaccard coefficient, weighted match and cosine similarity. In contrast with other approaches, this work concentrates on returning exact join results to these join predicates. The optimized algorithm includes a threshold sensitive merge list procedure reducing the running time.

Bilenko & Mooney (2003) also use EM to train the probabilities in a simple edit transducer for one of the duplicate detection measure they evaluate. (Bilenko et al., 2005) proposed an online learning-based method for determining the similarity between record pairs from streaming data in Internet comparison shopping. The learnt similarity function is used in clustering to determine which records are coreferent and should be linked.

Tejada et al. (2001) developed a system that employs active learning methods for selecting record pairs that are informative for training the record-level classifier. The classifier combines similarity estimates from multiple fields across different metrics. In the object identification system of (Tejada et al., 2002), called *Active Atlas*, a set of domain-independent string transformations have been developed (e.g., stemming, soundex, abbreviation conversion or computation) and applied on the shared attributes of the objects to compare. The system learns to tailor the weights of the set of these general operations to specific application domains with minimal user intervention. To achieve accurate mapping, the system chooses the most informative candidate mappings from the training examples for the user to classify as *mapped* or *not mapped*. Decision trees are used to learn the *mapping rules* selected with an algorithm based on committee and majority votes.

The major limitation of supervised approaches for duplicate elimination is to assume that a training data set (or a corpus of properly labeled strings) is available and exhibits the variety and distribution of data errors and misspellings observed in realistic situations (e.g., in data integration scenarios).

Other approaches that do not have this limitation relies on association rule mining. Lee et al. (2004) designed a method based on association rule mining to disambiguate references using similarity of the context attributes. The authors use the discovered association rules and frequent co-occurrences of attribute values. Rules with high confidence and minimum support that contain spurious attributes in their antecedents are identified and attributes in their rule consequent constitutes the *context attributes*. A column-wise similarity measure is applied to the context attribute sets to find their overlap. If the context attributes of two sets of records are similar, then they are considered as referring the same real-world entity. However, the problem of determining the confidence and support thresholds for selecting the best association rules among a massive and possibly inconsistent set of rules still remains.

1.2.1.3 Graph-based and Clusterwise Deduplication

For the reference disambiguation problem, graph-based models have been recently proposed, e.g., (Bansal et al., 2002; Dong et al., 2005; Kalashnikov & Mehrotra, 2006).

RelDC (*Relationship-based Data Cleaning*) (Kalashnikov & Mehrotra, 2006) is a domain-independent approach based both on the notion of *context attraction principle* (CAP) and the measure of the *connection strength* between the references candidate for disambiguation. The database is viewed as an undirected entity-relationship graph composed of *regular nodes* and *choice nodes* representing possibly matching references. Connection strength between two nodes u and v is computed as the probability of reaching a node v from a node u via random walks in the graph.

Dong et al. (2005) proposed an algorithm based on propagating *reference-similarity* decisions in a *dependency graph* whose nodes represent similarities between pairs of *references* (*i.e.*, records or attribute values), and whose edges represent the dependencies between the reconciliation decisions.

Correlation clustering proposed by (Bansal et al., 2002) also consists of a graph-based representation of the data set with edges labeled as "*similarity*" or "*disagreement*" edges according to a similarity function between data (vertices). Given the corresponding fully-connected graph, the goal is to find the partition of the vertices into clusters that minimizes as much as possible the number of *disagreements* as illustrated in Figure 1.2; that is, to partition the graph into clusters with high similarity for intra-cluster edges and weak similarity (disagreement) between inter-cluster edges. The main advantages of the approach are to avoid the problematic specification of the number of clusters as input parameter for the clustering method, and to be extendable to the case of real-valued labels. Because the approach is NP-hard, the question is to find the appropriate approximation algorithm. As a solution, (Charikar et al., 2003) proposed a factor four algorithm for minimizing the *disagreements* in a complete general weighted graph. This approach (still impractical with $O(n^3)$ constraints over n clusters) uses a linear programming formulation of the problem, such as:

$$minimize \sum_{i,j \in edges} w_{ij} \cdot x_{ij} + \sum_{i,j \in edges} w_{ij} \cdot (1 - x_{ij})$$

such that $x_{ik} \leq x_{ij} + x_{jk}$ (triangular inequality) for all i, j, k vertices of the graph, $x_{ij} \in \{0, 1\}$: if i and j are in the same cluster x_{ij} is 1; otherwise x_{ij} is 0.

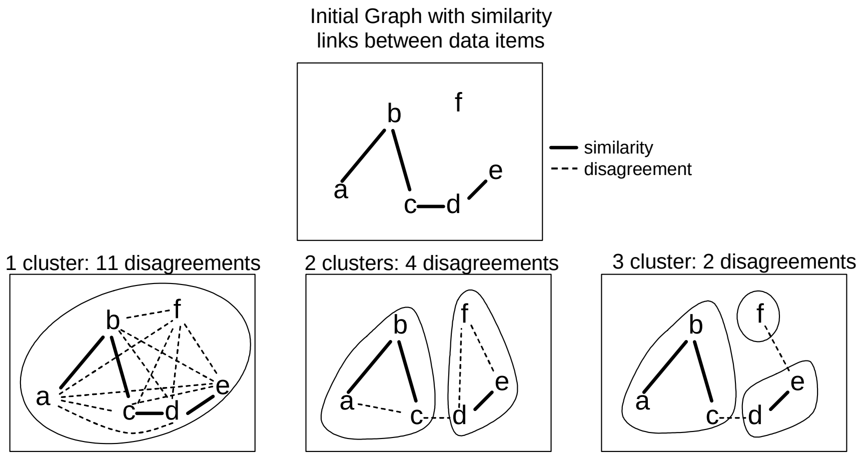

Initial Graph with similarity links between data items
similarity
disagreement
1 cluster: 11 disagreements
2 clusters: 4 disagreements
3 cluster: 2 disagreements
a
b
c
d
e
f

Figure 1.2: Correlation Clustering Example from (Bansal et al., 2002)

A method for online clustering in (Sarawagi & Kirpal, 2004) avoids the problem of data skew and redundant computation while creating clusters based on similarity along multiple strings rather than single strings, and it improves the efficiency when index searching with increasing threshold to return the most similar cluster.

Collective deduplication (Singla & Domingos, 2005) has been proposed to take advantage of information gleaned from one separate match decision to other matching decisions using *Conditional Random Fields* (CRFs) (McCallum et al., 2005). In this approach, decisions are made collectively, performing simultaneous inference for all candidate match pairs. This approach allows information propagation from one candidate match to another via their shared attributes (*fields*). CRFs (also known as *random fields* or *Markov networks*) are undirected graph models, trained to maximize the conditional probability of the outputs given the inputs. When the edges among the output variables form a linear chain, they correspond to conditionally-trained finite state machines. Three types of nodes are defined to build the collective model: *record-match nodes*, *field-match nodes*, and *field-similarity nodes*, also named *evidence nodes* (directly computed from data). The edges between two nodes represents the fact their values directly influence each other. All non-evidence nodes are Boolean-valued and the inference problem is reduced to a graph min-cut problem.

Among the unsupervised approaches for duplicate elimination, other clustering algorithms have been used to partition a relation into groups of duplicates (Bhattacharya & Getoor, 2004; Chaudhuri et al., 2005; Culotta & McCallum, 2005). Localized structural properties of clusters, such as the immediate vicinity of data that characterize the groups are important to consider for a clustering-based solution of the duplicate elimination problem. In (Chaudhuri et al., 2005), two criteria, namely *Compact Set* (CS) and *Sparse Neighborhood* (SN) are added to the formulation of duplicate elimination problem. They respectively reflect the facts that:

- *Compact Set* (CS): the set of duplicates must be a compact set of at least two mutual nearest neighbors: NN-relation is not symmetric (by opposition to global threshold approaches assuming transitivity). For example, if a is duplicate of b and b is duplicate of c, then a is not necessary duplicate of c;

- *Sparse Neighborhood* (SN): if the number of tuples in the larger sphere defined by the farer NN-distance around the tuple is small, the local neighborhood is sparse. A similar SN criterion is proposed in (Breunig et al., 2000). A two-phase approach is proposed by (Chaudhuri et al., 2005): *i)* the NN computation phase based on cosine metric, edit distance and fuzzy match similarity (Chaudhuri et al., 2003), and *ii)* the partitioning phase of an input relation into the minimum number of compact SN groups that satisfy specifications on the size and diameter of the group of nearest neighbors.

Clusterwise similarity metric proposed by (Culotta & McCallum, 2005) is used to measure the cohesion of a cluster and is learnt from a deduplicated training labeled database, by sampling positive and negative example clusters. Weighted, first-order features over clusters are then used to describe the compatibility of the nodes in each cluster. The weight of each predicate is estimated by maximizing the conditional log-likelihood of the training data. The optimal partitioning of a graph is then approximated with an agglomerative algorithm that greedily merges clusters based on their predicted compatibility scores.

For the bibliographical co-reference resolution problem, (Bhattacharya & Getoor, 2004) proposed an iterative process to refine the duplicate detection considering both the attributes of the objects (e.g., authors, titles, book titles in the paper citations represented as vectors) and the links between the objects (e.g., the references cited in the bibliography section of each paper). The thresholded distance between two references is a function combining the distance between the attributes and the distance measure between the groups of cited references (links). As new duplicates are discovered, the distances between groups of references are going to change, potentially leading to the discovery of more duplicates. Current sets of duplicates are represented as clusters. With each cluster is associated the set of groups that its references occur in. At each step, the algorithm re-evaluates the distances between the clusters (*i.e.*, groups of references) and merges the "nearest" cluster-pair until there are no more candidates.

1.2.2 Handling Inconsistencies

In database applications, integrity constraints (ICs) represent fundamental knowledge about the domain of interest. They are expressed as first-order formulae, and database instances are seen as first-order structures of finite relations. A database instance D is *consistent* with respect to a set IC of ICs if D satisfies IC (usually denoted by $D \models IC$). Since the genesis of the relational model, ICs have provided tremendous folder for database research (Ceri et al., 2000). Three categories of constraints are classically considered: *functional dependencies (FDs)*, *inclusion dependencies (IDs)*, and *key dependencies (KDs)*. Practically, the most common kinds of constraints are specified declaratively:

- *Key constraints*: Primary Keys (PK) - *i.e.*, non null attributes of a relation that uniquely identify an instance of the relation should be unique.

- *Referential consistency constraints*: Foreign Keys (FK) - *i.e.*, attributes of a relation that establish relationships among relations should correspond to PK in related relations, or ensure the existence of related relations; no value should be inserted in a table as FK without a corresponding column in the related tables and they should be updated simultaneous whenever update occurs. Particular actions to be taken upon violations, such as *cascaded delete* or *set null* can be specified.

- *Domain consistency rules* (consistency within column values): attribute values must fall within certain ranges or may assume only certain pre-defined values. Semantic integrity constraints and transition constraints are considered to be special cases of domain constraints.

In data integration scenarios, integrity constraints enrich the semantics of the global view of the set of data sources while such constraints may be locally violated at the sources (Fagin et al., 2003; Halevy, 2001). Maintaining consistency is then particularly difficult in the presence of semantic heterogeneity, which occurs when multiple information sources model overlapping portions of the real world in different ways (Ibrahim, 2002). This problem has been extensively studied in several works in the area of *inconsistent databases* (Bry, 1997). Various approaches to the management of ICs have been proposed, e.g., transposing the problem of reasoning with inconsistent databases as a belief revision problem (Calì et al., 2003; Lin & Mendelzon, 1998) or proposing a new semantic approach to cope with data inconsistencies in opposition to the *closed world assumption* of the traditional database theory (Arenas et al., 1999; Lembo et al., 2002; Lin & Mendelzon, 1998).

As illustrated in the following example from (Bertossi & Bravo, 2005), different IC enforcement mechanisms have been proposed for dealing with inconsistencies.

Consider a database schema consisting of two unary relations R and S and the following IC: $\forall x, \neg(R(x) \wedge\ S(x))$. Assume a database instance consists of the following facts: $R(a), S(a), R(b)$. One of the following approaches can be then adopted in this case of constraint violation:

- **Prevention** (usual constraint enforcement): such an instance could not arise: only one of $R(a)$ and $S(a)$ could be inserted into the database.

- **Ignorance** (constraint non-enforcement): no distinction is made between $R(a)$ and $R(b)$.

- **Isolation** (Bry, 1997): both $R(a)$ and $S(a)$ would be dropped or ignored in query answering.

- **Weakening** (Lin & Mendelzon, 1998): $R(a)$ and $S(a)$ would be replaced by $R(a) \vee S(a)$ or a disjunctive information.

- **Exception tolerance** (Arenas et al., 2000): the constraint is weakened as: $\forall x, \neg(R(x) \wedge\ S(x) \wedge x \neq a)$.

- **Materialized repairing** (Embury et al., 2001): the instance would be replaced by a consistent instance minimally different from the original one, as: $\{R(a), R(b)\}$ or $\{S(a), R(b)\}$.

- **Virtual repairing** (Arenas et al., 1999): returns consistent query answers, in this case, for all such x that $R(x)$ is true, returns only $x = b$.

- **Attack/support approach** (Pradhan, 2003): $R(a)$ attacks $S(a)$ and vice versa, and thus the support for both is lower than for $R(b)$.

1.2.2.1 Consistent Query Answering

We briefly survey the approaches focused on the problem of computing consistent query answers (CQA) in relational data integration under a *loose* semantics, *i.e.*, a semantics, which selects, among all possible database instances satisfying the integrity constraints expressed on the database schema, only the ones that are "*as close as possible*" to the actual database instance. We distinguish the approaches both on the properties of their semantics (exact, sound, complete, or loosely-exact, -sound, -complete) and on the ordering between databases (*i.e.*, cardinality-based, set-containment-based, and preference-based).

In (Lin & Mendelzon, 1998) the authors proposed an operator for merging databases with conflicting schemas under first-order formulae constraints; they compute the multisets [1] of the combination of the database instances that have the maximal cardinality and that are consistent with ICs.

In (Arenas et al., 1999) the authors define an algorithm for computing consistent query answers based on the notion of *residues* originally defined in the context of deductive database and semantic query optimization using the semantic knowledge about the domain that is contained in the ICs. The method is proved to be sound and complete only for the class of universally quantified binary constraints (BICs) (*i.e.*, constraints that involve two database relations). The authors also introduced the first mechanisms for computing consistent answers to first-order queries that did not appeal to explicit computation of repairs. According to (Arenas et al., 1999), a *repair* of a relational database instance D is an instance that satisfies the ICs, with the same schema as D, that in set theoretic terms, minimally differs from D with respect to whole tuples that are either deleted or inserted in order to restore consistency. However, repair mechanisms and semantics have been intensively studied since the first contribution of Arenas et al. (1999) on CQA in two main directions: *i)* repairs that minimally differ in cardinality from the original database (Arenas et al., 2003; Bertossi & Chomicki, 2003), and *ii)* repairs that minimize some aggregation function over the differences of attribute values between the repair and the original database (Bertossi & Bravo, 2005; Flesca et al., 2005; Wijsen, 2003).

In the area of inconsistent databases, several directions are emerging, as:

- *Preference-ordered repairs or preference-driven CQA* (Chomicki, 2006): a preference order over the database repairs is defined, in such way that only minimal repairs (in terms of set containment) are considered.

- *Null or default values consideration for repairs* (Bravo & Bertossi, 2003)

- *Intrinsic logic and compositionality properties of CQA* (Bertossi & Bravo, 2005): classical query answering in relational databases follows, essentially, a first-order logic (expressed through the relational calculus) that is monotonic and has advantageous compositional properties (e.g., answers to queries can be combined in order to give answers to more complex queries). In the case of CQA relying on non-monotonic formalisms (e.g., annotated predicate logic, logic programs with stable model semantics (Arenas et al., 2003), circumscription, or *analytic tableaux* (Bertossi & Schwind, 2004)), the compositionality of CQA has not been investigated and the answer set to a conjunctive query may not be the intersection of the answer sets.

- *Intractability and complexity of CQA*: tractable classes of CQA approaches have been identified by (Calì et al., 2003) but trade-offs between expressive power and complexity need to be further identified and studied together with approximate answers.

[1] A multiset is similar to a set except that a multiset allows duplicates in (Lin & Mendelzon, 1998).

1.2.2.2 Probabilistic Constraints

Statistical constraints manifest relationships among current attributes values in the database (Hou & Zhang, 1995). Since statistical constraints are probabilistic, the determination of the database correctness can be practiced at various degrees of strictness noted α, $0 \leq \alpha \leq 1$. The higher the α value, the more strict the system is or the smaller the discrepancy between the attribute value and the expected value will be tolerated. Statistical constraints can be considered as a weaker form of functional dependency in some respect.

The approach of Korn et al. (2003) is based on *probabilistic and approximate constraints* (*PACs*) that indicate, in a flexible way, the likelihood δ of a correctness constraint being satisfied within a tolerance ϵ as a cumulative probability distribution function (CDF). These constraints are specified as user-defined templates with tolerances and probability thresholds whose parameters are learnt using the statistical properties of the data. Three types of PACs are described in (Korn et al., 2003):

- *Domain PAC* specifies for a domain attribute D, that all attribute values x fall within ϵ of D with at least probability δ, that is: $Pr(x \in [D \pm \epsilon]) \geq \delta$.

- *Functional dependencies PAC* enforces, for the functional dependency $X \rightarrow Y$, that two tuples T_i and T_j must *approximately* agree on the values in the set of attributes Y if they agree in attributes X, that is:
  if $|T_i \cdot A_\ell - T_j \cdot A_\ell| \leq \Delta_\ell, \forall A_\ell \in X$, then $Pr(|T_i \cdot B_\ell - T_j \cdot B_\ell| \leq \epsilon_\ell) \geq \delta, \forall B_\ell \in Y$.

- *Unique key PAC* enforces that it is unlikely that more than one tuple exists with approximately the same values for the attributes A_ℓ that constitute a key in the table T, that is:
  $Pr(|T_i \cdot A_\ell - T_j \cdot A_\ell| \leq \epsilon_\ell) \leq \delta_\ell$ for each attribute A_ℓ composing the key.

In the context of real-time object management, (Ramamritham, 1993) introduced the notion of real-time data object (X_i) that is **absolutely temporally consistent** if its creation time (ct_i) plus the validity interval (V_i) of the data object (as the lifespan of the data value) is not smaller than current time, *i.e.*, $ct_i + V_i \geq t$. This defines practically the correctness criterion for temporal consistency of real-time data objects. If the lifespan has expired, the data value is temporally inconsistent and needs to be refreshed. The problem of temporal consistency maintenance (Xiong et al., 2006) is to efficiently generate periodic update transactions, which capture the latest status of the database and refresh the values of real-time data. Efficiently means reducing the update workload and thus, to know in advance the worse-case computation time of update transactions.

1.2.3 Managing Imprecise and Uncertain Data

A number of propositions has been devoted to capturing uncertainty in the context of relational databases (Barbará et al., 1990; Cavallo & Pittarelli, 1987; Gelenbe & Hébrail, 1986; Lakshmanan & Sadri, 1994; Lee, 1992; Re et al., 2006). Despite these efforts not all issues have been satisfactorily solved. Moreover, modeling uncertainty in other types of databases, such as XML databases is still in its childhood (Hung et al., 2003).

1.2.3.1 Probabilistic Data Models

In the relational context, Cavallo & Pittarelli (1987) described a model in which all tuples in the same table are disjoint (exclusive).

Lakshmanan & Sadri (1994) described a model in which probabilities are approximated by intervals.

More recently, Widom (2005) described a probabilistic system called Trio where probabilities and data lineage become first class citizens; the design space for probabilistic data models for Trio is described in Section 1.4.3.

These approaches are based on probability theory, and as a consequence they inherit the limitations of this theory. Probability theory is very suitable to capture uncertainty but not suitable to model ignorance. This has been noted and discussed in the work of Barbará et al. (1990) and Choenni et al. (2006). Choenni et al. (2006) proposed a framework for capturing uncertainty and ignorance in a unified way : inspired by the Dempster-Shafer theory, the authors assume that an attribute can assume a set of values instead of a single value and they assign a so-called *basic probability assignment (bpa)* to the attribute. In order to support joins and solve the problem of information loss, they extend the Dempster-Shafer theory with the notion of a dependent *bpa* and a combination rule. Such a *bpa* provides the possibility to take dependencies between data into account.

1.2.3.2 Possibilistic Data Models

Possibility theory provides an ordinal model for uncertainty where imprecision is represented by means of a preference relation encoded by a total order over the possible situations. This approach provides a unified framework for representing precise values, as well as imprecise ones (regular sets) or vague ones (fuzzy sets), and various null value situations. From a semantic point of view, a possibilistic database D can be interpreted as a set of usual databases (worlds), each of which being more or less possible; one of them is supposed to correspond to the actual state of the universe modeled. Any world W_i is obtained by choosing a candidate value in each possibility distribution appearing in D and its possibility degree is the minimum of those of the candidates chosen (according to the axioms of possibility theory since choices are assumed to be independent). The work of Bosc et al. (2006) deals with the querying of possibilistic relational databases, by means of queries called *generalized yes/no queries*, whose general form is: "to what extent is it possible that the answer to Q satisfies property P". More precisely, inclusion-based queries, where property P concerns the inclusion in the result of a set of tuples specified by the user, have been investigated. The possibility and necessity degrees, which constitute the answer to these queries, are respectively computed thanks to a "trial and error" procedure and an algorithm of linear complexity (Bosc et al., 2006).

However, most models make assumptions about data uncertainty that restricts applicability. Capturing correlations in the uncertainty model with a simple and intuitive semantics that is readily understood and defines precise answers to every query are two of the main current needs.

1.2.4 Handling Missing and Incomplete Data

The concept of data completeness may address a variety of issues from the very simple technical ones (e.g., problems of the data collecting campaign, loss of data during data exchange, format conversion or inappropriate user actions) to fundamental questions on the limits of our mental models and scientific concepts for representing real-world phenomena (e.g., inappropriate data modeling, conceptual fuzziness, impact of temporal changes on entity description completeness).

An added complication to the problem of missing or incomplete data is that the more data that are missing in a database, the more likely it is that you will need to address the problem of incomplete cases, yet those are precisely the situations where imputing or filling in values for the missing data points is most questionable due to the small proportion of valid data relative to the size of the data set.

1.2.4.1 Completeness of Relational Data Sources

Completeness in the relational model is classically defined with respect to :

- *the presence* (or absence) of null values,

- *the various possible interpretations*: *i)* values may exist but they are unknown, *ii)* values may not exist, and *iii)* no one knows whether values exist or not), and

- *the adopted paradigm*: the *open world assumption* (OWA) or the *closed world assumption* (CWA).

The OWA paradigm states that the real world is represented by the values present in the relational table and also by the negation of the values that are not represented in the relational schema. OWA assumes that the knowledge of the real world is incomplete. If something cannot be proved to be true, then it doesn't automatically become false. In the OWA, what is not stated is considered unknown, rather than wrong.

The CWA paradigm states that the real world is represented only by the values that are present in the relational schema. In other words, CWA is the presumption that what is not currently known to be true is false.

Under the OWA assumption, the completeness of a relation can be measured with respect to the size of a *reference relation*, also called the *universal relation*, denoted U. In this context, completeness has been defined by several authors. Naumann et al. (2004) defined the completeness measure for the size of a data source S as a combination of two orthogonal criteria, as:

- $coverage(S)$ is the measure for the relative number of tuples a source S provides compared to a universal relation U, as:
  $coverage(S) = \frac{|S|}{|U|}$,
- $density(S)$ is the measure for the average amount of data the source S provides for a single tuple t for a given attribute a, such as:
  $density(S) = \frac{1}{|A|} \sum_{a \in A} | \frac{\{t \in S | t[a] \neq \perp\}|}{|S|}$.

In (Motro & Rakov, 1998), the authors proposed a model for specifying soundness and completeness of relational database table instances, and a method of using these specifications to estimate the quality of query answers, working in an extension to relational algebra. Soundness measures the proportion of the stored information that is true and completeness measures the proportion of true information that is stored. The authors also show how their techniques can be used to assist in value-conflict resolution in the Multiplex multidatabase system.

Under the CWA assumption, the completeness can be defined depending on the type of the elements that are considered in the relation:

- *vertical completeness* concerns the number of null values of a specific attribute domain,
- *horizontal completeness* concerns the number of null values of a tuple or a set of tuples.

The notion of completeness in distributed query planning is also considered along with other quality criteria such as response time, accuracy and timeliness in (Naumann et al., 1999). A focus in (Naumann et al., 1999) is on exploiting completeness measures in query planning. However, the query language considered is much simpler than a complete data manipulation language. For example, it has no negation, so that soundness is not an issue, and the only merge operation in query planning is join.

In (Mihaila et al., 2000), an approach is presented to the selection and ranking of information sources and unions of sources based on the grounds of content and quality metadata provided by those sources. Completeness is one of the quality dimensions considered, although in contrast to the work of (Naumann et al., 1999), sources are only combined by union operations with no joins being performed.

1.2.4.2 Join Size Estimation

The estimation of the cardinalities of intermediate results in relational algebra expressions in the context of database query optimization has been intensively studied over the past decades.

Prediction of join result sizes is an important technique for cost-based query optimization in RDBMS. Classically, join result size is the size of the cross-product of the two relations in the join, multiplied with a *selectivity factor*.

Selectivity factors are statistical values stored in the data dictionary of the RDBMS (Mannino et al., 1988). Most of techniques for join size estimation adopt two simplifying assumptions: uniformity of attributes values and independence of attribute values. In (Najjar & Slimani, 1999), the authors use information from database profiles and statistical sampling to estimate cardinalities of derived relations in database query processing.

Peim et al. (2003) described a global-as-view distributed query processing system in which queries are formulated in the language, *ALCQI*. This paper presents a method for estimating soundness and completeness of *ALCQI* query plans, by estimating the cardinality of the extents of associated *ALCQI* expressions.

The problem of answering queries in the presence of limited access patterns to relations has been studied by Li (2003) for conjunctive queries, unions of conjunctive queries, and conjunctive queries with arithmetic comparisons. A method for data-dependent computability of the complete answer to a query is proposed, and a decision tree is used for guiding the process to compute the complete answer to a conjunctive query (Li, 2003).

The basic idea of (Nash & Ludäscher, 2004) is to avoid performing the computationally hard containment checks. They propose algorithms that use two efficiently computable approximate plans Q_u and Q_o, which respectively produce tight underestimates and overestimates of the actual query answer for Q, if possible, defer the query containment check. Finally, a runtime algorithm reports complete answers even in the case of infeasible plans, and quantifies the degree of completeness.

In (Petropoulos et al., 2006), the CLIDE interactive system uses a flag- and color-based scheme for query building. It leads the user toward feasible queries in a setting where the content and access methods of the sources are described by parameterized conjunctive views. The system checks whether a given query is supported (*i.e.*, feasible) and requires non-obvious rewriting algorithms, especially when the set of indirectly supported queries is enhanced via additional processing inside a mediator. A flag indicates whether the current query is feasible or not. If it is, colors indicate how to reach another feasible query, which will be a syntactic extension of the current one. The system also provides: *i)* guarantees of completeness (*i.e.*, every feasible query can be built by the system's suggested actions only), *ii)* minimality (*i.e.*, the minimal set of actions that preserves completeness is suggested by the system), and *iii)* rapid convergence (*i.e.*, the shortest sequence of actions from a query to any feasible query consists of the suggested actions).

1.2.4.3 Completion and Statistical Editing

The most appropriate way to handle missing or incomplete data depends upon the type of mechanisms underlying the generation of missing data. Three cases of missing data generation are classically identified in Statistics, as:

- *Missing Completely at Random (MCAR)*: cases with complete data are indistinguishable from cases with incomplete data.

- *Missing at Random (MAR)*: cases with incomplete data differ from cases with complete data, but the pattern of data missingness is traceable or predictable from other variables in the database rather than being due to the specific variable on which the data are missing.

- *Non-ignorable*: the pattern of data missingness is non-random and it is not predictable from other variables in the database. In contrast to the MAR situation outlined above where data missingness is explainable by other measured variables in a study, non-ignorable missing data arise due to the data missingness pattern being explainable - and only explainable - by the variable(s) on which the data are missing.

In practice it is usually difficult to meet the MCAR assumption. MAR is an assumption that is more often, but not always tenable. The more relevant and related predictors one can include in statistical models, the more likely it is that the MAR assumption will be met. A non exhaustive list of methods for handling missing data in data analysis is presented below. It covers some of the more widely recognized approaches to handling databases with incomplete data.

- *Listwise data deletion*: if a record has missing data for any one variable used in a particular analysis, the strategy is to omit that entire record from the analysis.
- *Pairwise data deletion*: For bivariate correlations or covariances, statistics are computed based upon the available pairwise data.
- *Mean substitution*: The variable's mean value computed from available data is substituted to fill in missing data values.
- *Regression methods*: A regression equation is developed on the basis of the complete data for a given variable, treated as the outcome using all other relevant variables as predictors. Then, for cases where data is missing, the available data are used into the regression equation as predictors.
- *Hot deck imputation*: the most similar tuple to the tuple with a missing value is identified and the most similar data value is substituted for the missing value.
- *Expectation Maximization (EM) approach*: EM is an iterative procedure that proceeds in two discrete steps. First, in the expectation (E) step, the expected value of the complete data log-likelihood is computed. In the maximization (M) step, the expected values for the missing data obtained from the E step are substituted and then the likelihood function is maximized as if no data were missing to obtain new parameter estimates. The procedure iterates through these two steps until convergence is obtained.
- *Raw maximum likelihood methods (Full Information Maximum Likelihood - FIML)*: all available data are used to generate maximum likelihood-based sufficient statistics consisting of a covariance matrix of the variables and a vector of means.

- *Multiple imputation*: Similar to the maximum likelihood method, except that multiple imputation generates actual raw data values suitable for filling in gaps in the considered database.

The review of these methods by Little & Rubin (1987) conclude that listwise, pairwise, and mean substitution methods for handling missing data are inferior when compared with maximum likelihood based methods such as raw maximum likelihood or multiple imputation. Regression methods are somewhat better, but not as good as hot deck imputation or maximum likelihood approaches. The EM method falls somewhere in between. It is generally superior to listwise, pairwise, and mean substitution approaches, but it lacks the uncertainty component contained in the raw maximum likelihood and multiple imputation methods.

1.2.5 Improving Data Freshness

Various factors and metrics have been proposed for characterizing data freshness depending on the type of the data management system: data integration and warehousing systems with materialized views (DIS/DW), virtual mediation systems (VMS), and caching systems (CS) (Peralta, 2006). The traditional freshness definition is called currency (Segev & Fang, 1990; Theodoratos & Bouzeghoub, 1999). It is related to view consistency when materializing data and describes how stale is data with respect to the original data sources. Currency captures the difference between query time and extraction time. Several proposals incorporate another notion of freshness, called timeliness, which describes how old is data. Timeliness captures how often data changes or how often new data is created in a source (the difference between query time and last update time) (Naumann et al., 1999). Therefore, freshness represents a family of quality factors, each one best suiting a particular problem or system architecture. For example, the freshness rate is the percentage of tuples in a materialized view that are up-to-date and these that have not been updated since data extraction time (Labrinidis & Roussopoulos, 2003). For search engines that copy portions of the Web, remote information sources are updated independently without pushing updates to clients that have a copy, so the clients must periodically poll the source to detect changes and refresh its copy. Cho & Garcia-Molina (2000) used the update frequency for measuring data freshness in a caching context.

A complete study on freshness is given in (Bouzeghoub & Peralta, 2004; Peralta, 2006). Bouzeghoub & Peralta (2004) analyzed related research works in terms of a taxonomy of the approaches for achieving data freshness in multi-source information systems. This taxonomy is based upon: *i)* the dynamic nature of data (*i.e.*, stable, long-term-changing, frequently-changing data), *ii)* the architecture type of the data management system, *i.e.*, virtual mediation, materialized integration, or caching systems), and *iii)* the synchronization policies between the queries sent by the system to local passive data sources (*pull mode*) or the data sent by active data sources (*push mode*). Based on this detailed analysis of data freshness and also on data accuracy quality factors, Peralta (2006) proposed a framework together with techniques and algorithms for the evaluation and enforcement of data freshness and data accuracy factors in data integration systems. The proposal includes a graph-based representation of DIS processes and the specification of quality evaluation algorithms as graph propagation methods applicable to various DIS applications. An extensible quality evaluation tool that implemented the framework has been prototyped. The framework components are specified in an abstract model, which supports the dynamic incorporation of new components to the tool, especially the inclusion of new quality factors and their evaluation algorithms.

Although data freshness has been largely studied (Cho & Garcia-Molina, 2000; Li et al., 2003; Segev & Fang, 1990; Zhuge et al., 1997), several problems still remain either unsolved or insufficiently treated.

1.2.5.1 View Maintenance and Update Propagation

Many incremental algorithms for view maintenance have been introduced for centralized database systems and also in distributed environments (Hull & Zhou, 1996; Zhuge et al., 1997). Numerous works have formed a spectrum of solutions ranging from a fully virtual approach to one end where no data is materialized and all user queries are answered by interrogating the source data, to a full replication at the other end where the whole data in the data source is copied to the data integration system so that the updates can be handled locally without consulting the sources. The two extreme solutions are inefficient in terms of communication and query response time in the former case, and storage space in the latter.

In the context of view materialization, a view is consistent if its state reflects an actual source state at some “recent time” (Zhuge et al., 1997), so the goal is assuring a certain degree of data currency. The view maintenance problem consists of updating a materialized view in response to changes arisen at source data. Most of the work concentrates in assuring data warehouse consistency for different types of views and refreshment strategies (Gupta et al., 1995). Another key problem concerns the selection of a set of views to materialize in order to optimize the query evaluation and the maintenance cost (Theodoratos & Bouzeghoub, 1999; 2001). In this context, freshness is implicitly considered when defining the update propagation processes.

Other works combine different properties and study the trade-off between them, such as the trade-off between execution time and storage constraints (Labrinidis & Roussopoulos, 2003).

1.2.5.2 Synchronization Policies

In caching systems, data is considered fresh when it is identical to data in the sources, so freshness is represented by the currency factor. An important problem is defining the refreshment policies in order to keep cache data up-to-date. Traditional cache proposals estimate the time-to-live (TTL) of an object to check whether this object is valid or not and when to get it from its source. In (Cho & Garcia-Molina, 2000), the authors study the synchronization policies for cache refreshment and experimentally verify their behavior. In (Li et al., 2003), the focus is in the fine tuning of the caching policy, balancing response time and invalidation cycles for assuring data currency. In (Bright & Raschid, 2002), the authors propose the use of latency recency profiles to adapt caching algorithms to user currency requirements, accessing the remote site only when the expected currency is not achieved.

The overview of Data Quality Research would not be complete without the description of the academic tools and query language extensions that have been proposed for handling data quality problems, correcting them and querying data with quality constraints or priorities defined explicitly in the query statements or integrated in the query planning.

1.3 Tools and Query Language Extensions

1.3.1 ETL Tools

Under the general acronym ETL, the Extraction-Transformation-Loading activities cover the most prominent tasks of data preparation before the data warehousing and mining processes (Rahm & Do, 2000; Vassiliadis et al., 2003). They include: *i)* the identification of relevant information at the source side, *ii)* the extraction of this information, *iii)* the transformation and integration of the information coming from multiple sources into a common format and, *iv)* the cleaning and correction of data.

Despite the specialized ETL tools (mainly dedicated to relational data) available on the market, data preparation and cleaning processes remain complex, costly and critical.

This area has raised a lot of interest from the research community (Rahm & Do, 2000; Vassiliadis et al., 2003) with a recent focus on semi-structured data (Weis & Naumann, 2004). Several academic tools have been proposed for data transformation and consolidation: AJAX (Carreira & Galhardas, 2004; Galhardas et al., 2000; 2001), Potter's Wheel (Raman & Hellerstein, 2001), ARKTOS (Vassiliadis et al., 2001), BELLMAN (Dasu et al., 2002), Telcordia (Caruso et al., 2000).

AJAX (Carreira & Galhardas, 2004; Galhardas et al., 2000; 2001) is an SQL extension for specifying each data transformation necessary to the cleaning process of relational data. The proposed transformation operators are *mapping, view, matching, clustering and merging* as defined in Table 1.4. The operators are combined in a *logical plan* to be applied to the input dirty data for improving data accuracy and format conformance. The transformations standardize data formats when possible and find pairs of records that most probably refer to the same real-world object. A thresholded distance function is used to decide which pairs of values are to be joined with the *matching* operator of AJAX. Associated to each logical plan, the best physical plan is then chosen to improve the execution of data transformations without affecting the required accuracy.

Potter's Wheel (Raman & Hellerstein, 2001) is an interactive data cleaning system that integrates transformation (see Table 1.4) and discrepancy detection in a single interface. Users can gradually build a transformation to clean the data by adding transforms as discrepancies are detected. Users can specify transforms and transformation domains through graphical operations or through examples, and see the effect instantaneously, thereby allowing easy experimentation with different transforms. Parsing strings using structures of user-defined transformation domains results in a general and extensible discrepancy detection mechanism. Such domains also provide a basis for specifying *Split* transformations through example values.

Table 1.4 presents the main ETL operators (e.g., *format, add, merge,* etc.) supported by Potter's Wheel and AJAX systems with, in the first column, the name of the operator and, in the second column, its definition.

ETL Operator	**Definition**
Format	Applies a function to every value in an attribute column of a relational table (such as regular-expression based substitutions and arithmetic operations or user-defined functions).
Add, Drop, Copy	Allow users to add a new column, or to drop or copy a column.
Merging	At the value level, concatenates values in two columns, optionally interposing a constant in the middle, to form a new column. At the relation level, partitions an input relation according to various grouping attributes, and collapses each partition into a single tuple using an arbitrary aggregation function. User-defined aggregation functions can be expressed and used.
Matching	Computes an approximate join between two tables depending on a thresholded similarity distance function.
Split	Splits a column into two or more parts, and is used typically to parse a value into its constituent parts. The split positions can be specified by character positions, regular expressions, or by interactively performing splits on example values.
Divide	Conditionally divides a column, sending values into one of two new columns based on a predicate.
Fold / Unfold	Flattens tables by converting one row into multiple rows, folding a set of columns together into one column and replicating the rest. Conversely Unfold unflattens tables: it takes two columns, collects that have the same values for all the other columns, and unfolds the two chosen columns.
Clustering	Groups the records of an input relation into a set of clusters computed by either the GROUP BY operator or a distance function

Table 1.4: Main Data Transformation Operators for ETL

In the context of data warehouse cleaning, ARKTOS (Vassiliadis et al., 2001) has been proposed for modeling and executing practical ETL scenarios by providing explicit primitives (such as, primary key violation, reference violation, NULL value existence, uniqueness violation, and domain mismatch). For capturing the common cleaning tasks, three functionalities have been designed: *i)* graphical and declarative facilities for the definition of data warehouse transformation and cleaning tasks, *ii)* measurement of the quality of data through specific quality factors, and *iii)* optimized execution of complex sequences of transformation and cleaning tasks.

BELLMAN (Dasu et al., 2002) profiles the database and computes concise statistical summaries of the contents of the database, to identify approximate keys, set resemblance frequent values of a field (often default values), joinable fields with estimates of join sizes paths, and to understand database dynamics (changes in a database over time).

Kettle / Spoon[2] is the open source ETL of the Pentaho Data Integration Suite that provides basic transformation operators, filters, and cleaning primitives to realize complete ETL scenarios with multi-source and multi-format datasets (XML, text, RDBMS, files, etc.).

Both ETL tools and cleaning algorithms operate in a batch and off-line manner but "active data warehousing" (also called "real time warehousing") refers to a new trend where higher levels of data freshness are required for data warehouses that must be updated as frequently as possible. This raises interesting research directions concerning the performance optimization and overloading issues of ETL tasks (Karakasidis et al., 2005; Simitsis et al., 2005).

1.3.2 Record Linkage Tools

Several academic tools are more specifically dedicated to record linkage: e.g., Febrl (Christen et al., 2004), IntelliClean (Low et al., 2001), Tailor (Elfeky et al., 2002), ClueMaker (Buechi et al., 2003), and Telcordia (Caruso et al., 2000).

Febrl (*Freely Extensible Biomedical Record Linkage*)[3] (Christen et al., 2004) allows data standardization, segmentation, probabilistic cleaning, and "fuzzy" matching of one or more files or data sources which do not share a unique record key or identifier. Febrl has particular probabilistic and rules-based cleaning and standardization routines, and various comparison functions for names, addresses, dates, localities, and telephone numbers, including approximate string comparisons, phonetic encodings, and geographical distance comparisons. Febrl includes two approximate string comparison methods (bag distance and compression based) and provides several blocking methods, including the traditional compound key blocking used in many record linkage programs. Probabilistic record linkage routines of Febrl are based on the classical Fellegi-Sunter model and a classifier allows a flexible definition of the weight calculation.

IntelliClean (Low et al., 2001) is a rule-based cleaning system providing representation standardization, duplicate elimination, anomaly detection and removal in dirty databases. Data records are first conditioned and scrubbed of any anomalies that can be detected at the pre-processing stage. Data type checks and format standardization can be performed. Inconsistent abbreviations used in the data can be resolved at this stage. Conditioned records are next fed into an expert system engine together with a set of rules. Each rule will fall into one of the following categories:

- *Duplicate identification rules*: These rules specify the conditions for two records to be classified as duplicates.
- *Merge/purge rules*: These rules specify how duplicate records are handled. A simple rule might specify that only the record with the least number of empty fields is to be kept in a group of duplicate records, and the rest deleted. More complex actions can be specified in a similar fashion.

[2]Pentaho Data Integration Suite, http://www.pentaho.com/products/data_integration/
[3]Febrl, http://datamining.anu.edu.au/software/febrl/febrldoc/

- *Update rules*: These rules specify the way data is to be updated in a particular situation.

- *Alert rules*: A user might want an alert to be raised when certain events occur.

Tailor (Elfeky et al., 2002) is a toolbox for comparing record linkage techniques and tools with a corresponding benchmarking process with tunable parameters. The four layers evaluated by Tailor correspond to the main steps of the record linkage strategy, namely the searching method, the comparison function, the decision model and the measurement (recall/precision) to estimate the performance of the method.

Based on the online study of Galhardas[4], Table 1.5 presents the research prototypes and open source systems developed for data cleaning, data preparation and analysis.

Functionalities	AJAX	Arktos	Bellman	Febrl	Flamingo	FraQL	IntelliClean	Potter's Wheel	Spoon/Pentaho
Data Standardization	✓	✓		✓		✓		✓	✓
Data Transformation	✓	✓		✓		✓		✓	✓
Data Cleaning	✓	✓		✓		✓			✓
Duplicate Elimination	✓			✓	✓	✓	✓		✓
Data Profiling			✓						✓
Data Analysis			✓					✓	

Table 1.5: Data Cleaning Prototypes

1.3.3 Extended Query Languages

Along with the development of Data Quality Research, several approaches have been proposed for expressing in a simply and declarative way constraints on data quality extending the query language (Benjelloun et al., 2005; Guo et al., 2004; 2005; Sampaio et al., 2005; Widom, 2005).

As the very first prototype, *Q-Data* (Sheth et al., 1993) checks if the existing data are correct and ensures data validation and cleanup by using a logical database language (LDL++). The system employs data validation constraints and data cleanup rules.

More recently, Guo et al. (2004; 2005) 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. A C&C constraint is a triple composed of:*i)* a currency bound, *ii)* a consistency class formed by tables, and *iii)* grouping rules from a consistency class into consistency groups. Both single-block and multi-block queries are supported in this approach. For example, consider the following relation:

[4]Data Quality Tools, http://web.tagus.ist.utl.pt/ nuno.campos/tfc/

```
AUTHORS(authorID*, name, city, state, phone)
```

The following query will retrieve the authors identified by 1, 2 and 3 whose values have been updated within the last 10 minutes (CURRENCY BOUND 10 min); these values have to be consistent since they are extracted from the same snapshot of the table AUTHORS of the database (*ON (A)*) and such that the records are present and complete in the cache (BY $key E1.1: $key = authorID).

```
SELECT *
FROM Authors A
WHERE authorID in (1,2,3)
CURRENCY BOUND 10 min ON (A)
BY  $key
E1.1: $key = authorID;
```

DQ^2L (*Data Quality Query Language*) (Sampaio et al., 2005) was designed to query relational data supporting a data quality aware query processing framework. A number of simple construct operators extend SQL with the operator “WITH QUALITY AS” to express quality requirements relating to a query, and to measure with used-defined functions some quality dimensions. For example, consider the following query:

```
SELECT *
FROM Authors
WHERE authorID in (1,2,3)
WITH QUALITY AS TIME_t_last(city)>'2006-04-01 12:00:00+00:00'
TIMELINESS(phone)>='0.50';
```

The result is built with taking into account the date of the last update of the value on the attribute $city$ and also the timeliness of the value of the attribute $phone$ that is computed by the user-defined function TIMELINESS($phone$).

Similarly to these approaches, our approach called *XQUAL* (Berti-Équille, 2001; 2003; 2004) proposes the syntax and the processing of SFW queries extended by an operator called “QWITH”. We offer the users the way to declare and manipulate measures and constraints on data quality at different granularity levels, namely CELL, COLUMN, ROW, TABLE and DATABASE. As we’ll see in the next chapter in details, *XQUAL* is based on the notion of *contract* that is declared by the user with methods calls and user-defined functions declarations that are applied and eventually combined for measuring certain aspects of data quality. An example of the syntax of a contract type is such as:

```
CREATE CONTRACTTYPE completeness (
density FLOAT ON COLUMN
   BY FUNCTION func_density IS JAVA NAME './XQLib/density.java',
authors_coverage FLOAT ON AUTHORS
   BY FUNCTION func_authors_coverage
```

```
     AS (SELECT sum((COUNT(A1.author_ID)/(select COUNT(*)
                     FROM AUTHORS)))
         FROM AUTHORS A1
         WHERE A1.name is not NULL group by A1.author_ID);
```

This contract type is composed of two user-defined factors characterizing completeness, as: *i) density* applied to every column of the database and computed by a java program named *density.java* and *ii) authors_coverage* applied on AUTHORS table and computed by a PL/SQL function named *func_authors_coverage*. When a contract type is created, the declared functions are called and executed. The results of each declared functions are stored is the repository as metadata associated to the specified database object instances (e.g., ON AUTHORS) or declared granularity levels, *i.e.*, values (*CELL*), tuples (*ROW*), attribute domains (*COLUMN*), tables (*TABLE*), or the database (*DATABASE*). At runtime, extended queries called QWITH queries are executed and the constraints declared in the QWITH part are checked to build the query results accordingly.

```
SELECT *
FROM AUTHORS
WHERE author_ID in (1,2,3)
QWITH completeness.density > .85,
AND completeness.authors_coverage > .90;
```

Query processing of QWITH queries and user-defined functions will be described in details in Chapter 3.

In the context of semi-structured data, *XClean* (Weis & Manolescu, 2007) is a modular, declarative system for native XML data cleaning. In XClean, cleaning processes are modeled using a set of cleaning operators, that can be combined in arbitrarily complex cleaning processes. XClean's operators are: *candidate selection* for selecting elements to be cleaned, *scrubbing* for removing errors in text (typos, format, ...), *enrichment* for specifying data that supports cleaning, *duplicate filtering* for filtering non-duplicate element pairs, *pairwise duplicate classification* for classifying pairs of elements as duplicates and non-duplicates, *duplicate clustering* for determine clusters of duplicates, *fusion* for creating unique representation of an entity, and *XML view* for creating XML view of clean data. The operators specification is expressed in a high-level operator definition language, called *XClean/PL*. Writing XClean programs is supported by a graphical user interface. An XClean/PL program is compiled into XQuery, to be executed on top of any XQuery processor.

1.3.4 Quality-Driven Query Processing

Among the solutions proposed in the literature for query planning and optimization of the distributed queries, some approaches use data quality indicators and metadata for selecting the best query plans to execute (Mihaila et al., 2000; Naumann, 2002; Naumann et al., 1999). In the context of mediation systems, Naumann (2002); Naumann et al. (1999) proposed *query correspondence assertions* (QCAs) that are used to specify the mapping between the local data sources supported by the wrappers and the global schema supported by the mediator. Quality scores are collected or pre-computed for characterizing the criteria that are specific to the quality of the sources (e.g., reputation), to the user's query (e.g., completeness) and to the QCA (e.g., price). The Data Envelopment Analysis method (DEA) is used to prune low quality sources from the query planning space based on the scores on source-specific criteria. QCAs are then used to create correct query plans for the Local As View mappings (LAV), as the queries on the global schema are defined in terms of queries on the local sources. The quality of both QCAs and plans is evaluated with a procedure that is similar to cost model estimation in traditional DBMSs. A corresponding execution tree is built for each query plan, with QCAs as leaves and join operators as inner nodes, and also quality scores computed for each node. A set of functions is used to merge the scores of the children nodes for join operator and for computing the final quality score of the provided query result. The simple additive method (SAW) is finally used to rank and select the best query plan to execute. Alternative and ranked plans are semantically correct but they will not necessarily provide equivalent results.

In the context of quality-driven query processing, (Braumandl et al., 2001) proposed 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). The authors also proposed mechanisms to follow the execution of the query and, if necessary, to cancel it or change the query plan execution.

FusionPlex (Motro & Anokhin, 2006) is a system for integrating multiple heterogeneous data sources that uses data fusion to resolve factual inconsistencies among the individual sources under the LAV mapping approach. To accomplish this, the system relies on *source features*, which are metadata on the merits of each information source (e.g., data recentness, accuracy, availability, or cost). Local schemas of the sources are assumed to be consistent whereas instances may represent the same real-world entity differently from one source to another due to errors and inconsistencies. The query processing is based on *query fragments* and on *contributing views* the sources offer to answer a given query. When the query fragments results are overlapping, a so-called *polyinstance* is built and needs conflict detection and resolution.

1.3.5 SQL-Based Conflict Resolution

MOMA *(Mapping-based Object Matching system)* (Thor & Rahm, 2007) is a flexible and domain-independent framework for mapping-based object matching. The design of MOMA allows the combined use of multiple match algorithms both attribute and context matchers, for a given schema match problem. A key feature of MOMA is that it is massively based on the notion of instance mappings. The output of a match task is represented as a so-called *same-mapping* indicating which input objects are considered semantically equivalent. MOMA provides a high flexibility to determine a tailored workflow for a given match task. In particular, it allows selection and combination of several matchers and the re-use and refinement of previously determined mappings. But this high flexibility can make it difficult even for experts to specify a suitable workflow and configuration. Similar to the E-Tuner approach for schema matching (Sayyadian et al., 2005), MOMA therefore provides self-tuning capabilities to automatically select matchers and mappings and to find optimal configuration parameters. Initially the focus is on optimizing individual matchers and combination schemes. For example, for attribute matching, choices must be made on which attributes to match, and which similarity function and similarity threshold to apply. For suitable training data these parameters can be optimized by standard machine learning schemes, e.g. using decision trees.

In *FusionPlex* (Motro & Anokhin, 2006), the fusion process is controlled with several parameters, such as: *i)* a vector of feature weights: each user defines an individual notion of data utility, *ii)* thresholds of acceptance: users ensure minimal performance of their data, excluding from the fusion process data that are too old, too costly, lacking in authority, or numeric data that are too high, too low, or obvious outliers; and, ultimately, and *iii)* the particular fusion function to be used for each attribute (for example, average, maximum, or simply any): users implement their own interpretation of fusion.

HumMer (Bilke et al., 2005) provides a subset of SQL as a query language, which consists of Select-Project-Join queries, and allows sorting, grouping, and aggregation. In addition, it specifically supports a *FUSE BY* statement. This statement is an extension of an SPJ statement specially designed for easy specification of data fusion. Conflict resolution is implemented as user-defined aggregation. However, the concept of conflict resolution is more general than the concept of aggregation, because it uses the entire query context to resolve conflicts. The query context consists not only of the conflicting values themselves, but also of the corresponding tuples, all the remaining column values, and other metadata, such as column name or table name. This extension enables authors of FUSE BY statements to employ many different and powerful resolution functions. In addition to the standard aggregation functions already available in SQL (min, max, sum,etc.), the following list gives further examples of functions that may be used for conflict resolution in HumMer. These functions cover most of the strategies to resolve data conflicts mentioned in the literature.

- *Choose(source)*: Returns the value supplied by the specific source.
- *Coalesce*: Takes the first non-null value appearing.
- *First/Last*: Takes the first / last value of all values, even if it is a null value.
- *Vote*: Returns the value that appears most often among the present values. Ties could be broken by a variety of strategies, e.g., choosing randomly.
- *Group*: Returns a set of all conflicting values and leaves resolution to the user.
- *(Annotated) Concat*: Returns the concatenated values, including annotations, such as the data source.

- *Shortest/Longest*: Chooses the value of minimum / maximum length according to a length measure.

- *Most Recent*: Recency is evaluated with the help of another attribute or other metadata.

1.4 Research Projects

Several database-oriented projects dedicated to data quality have been proposed in the past decade: the European *DWQ Project (Data Warehouse Quality)*, the Italian *DaQuinCis Project* (Santis et al., 2003) for data quality in cooperative information systems, and the *Trio Project* (Widom, 2005). Other projects such as the *TDQM project* (Wang et al., 1995), *TQdM* (English, 2002; 1999), guidelines and experiences for data quality project management (Redman, 2001) dealing with methodological and IT aspects of business and project management for ensuring or improving information quality in the organizations and the information system design will not be presented in this section. But we invite the reader to read Chapter 7 of (Batini & Scannapieco, 2006) to have a detailed description of strategies adopted in the main information quality management and assessment methodologies.

1.4.1 DWQ Project

The European **DWQ Project** (*Data Warehouse Quality*) (1996-1999) developed techniques and tools to support the design of data warehouses based on data quality factors. Starting from a definition of the basic DW architecture and the relevant data quality issues, the DWQ project goal was to define a range of alternative design and operational methods for each of the main architecture components of a data warehouse and associated quality factors. Since usually a combination of enabling technologies is required, contributions of DWQ project concerned the DW design (e.g., rich metadata representation and reasoning facilities) as well as the operational level (e.g., viewing DW contents as views over the underlying information sources, refreshment techniques and optimal handling of views with aggregate functions). Formal models of the DW architecture and services have been proposed together with associated tools for consistency checking, reuse by subsumption, view materialization strategies, and other components of the data warehousing software. In (Jarke et al., 1999) the authors have proposed an architectural framework for data warehouses and a repository of metadata which describes all the data warehouse components in a set of meta-models to which a quality meta-model is added, defining for each data warehouse meta-object the corresponding relevant quality dimensions and quality factors. Beside from this static definition of quality, DWQ project have also provided an operational complement that is a methodology on how to use quality factors and to achieve *user quality goals*. This methodology is an extension of the Goal-Question-Metric (GQM) approach, which permits to capture the inter-relationships between different quality factors and to organize them in order to fulfill specific quality goals.

1.4.2 DAQUINCIS Project

The Italian **DaQuinCIS project** (2001-2003) focused on studying how the cooperation among information systems can play a key role in improving the data quality of individual information systems. More specifically, an integrated methodology has been proposed for encompassing the definition of an *ad hoc* distributed architecture and specific methodologies for data quality measurement and error correction techniques (Santis et al., 2003). This methodology includes process- and data-based techniques used for data quality improvement in single information systems. The distributed architecture of DaQuinCIS system consisted of: *i)* the definition of the representation model for data quality information that flows between different cooperating organizations via cooperative systems (CIS) and *ii)* the design of a middleware that offers data quality services to the single organizations with the *data quality broker* which poses quality-enhanced queries over the global schema and selects data satisfying these requirements, the *quality notification service* which allows quality-based subscriptions for organizations to be notified on changes of the quality of data, and the *quality factory*, which evaluates the quality of internal data of each organization.

1.4.3 TRIO Project

The **Trio project** at Stanford InfoLab (Benjelloun et al., 2005; Widom, 2005) is a database system that manages not only data, but also the accuracy and lineage of the data. The goals of the Trio project are: *i)* to combine previous work on uncertain, probabilistic and imprecise data into a simple and usable model; *ii)* to design a query language as an understandable extension to SQL; *iii)* to build a working system that augments conventional data management with both accuracy and lineage as an integral part of the data. The Trio system aims at extending traditional data management by introducing components such as accuracy and lineage into both data and queries. As its query language, *TriQL*[5] is designed as an extension to SQL, for querying and modifying data in conformance with the Trio data model, also called *ULDB model (Uncertainty and Lineage DataBase)*. The underlying data model consists of four new constructs: *i)* tuple alternatives (*x-tuple*), representing uncertainty about the content of a tuple with alternative values (*or-sets*), *ii)* maybe ("?") annotations, representing uncertainty about the presence of a tuple, *iii)* numerical confidence values, optionally attached to alternatives and maybe *x-tuples*, *iv)* lineage, connecting tuple alternatives to other alternatives from which they were derived. A Boolean $\lambda(t)$ function for the *x-tuple* t represents the lineage of each alternative and how t has been derived.

TriQL queries return uncertain relations in the ULDB model, with lineage that connects query result data to the queried data. It includes three built-in predicates and functions: *Conf*, *Maybe*, and *Lineage*. Function *Conf* can be used to filter query results based on the confidence of the input data; the *Maybe* and *Lineage* predicates are incorporated into the query translation phase. Predicate *Maybe* is straight-

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

forward: it translates to a simple comparison between the queried attribute and the number of alternatives in the current *x-tuple*. Predicate *Lineage(X,Y)* is translated into one or more SQL subqueries that check if the lineage relationship holds: schema level lineage information is used to determine the possible table level paths from X to Y.

1.4.4 QUADRIS Project

Ensuring and maximizing the quality of data in data integration systems requires a clear understanding of the interdependencies between the various dimensions that characterize the quality of data (QoD), the quality of the underlying conceptual data model (QoM), and the quality of the processes (QoP) that manage or integrate data. Restricted to a single level (*i.e.*, data, model or process level), the improvement of one quality dimension (e.g., data accuracy, process performance or model expressiveness) may have negative consequences on other quality dimensions at the same level or on the other levels (e.g., improving the data model expressiveness may improve data consistency up to a certain degree but in the same time it may degrade the lisibility of the model; improving the cleaning processes with new, more sophisticated and complex operations may degrade factors related to data freshness by extending the delay between data extraction time and data loading and diffusion times). In this context, the goal of the **QUADRIS project**[6] (2005-2008) is to propose a framework for adopting quality improvement strategies on one or many dimensions of QoD, QoM or QoP with considering collateral effects and interdependencies between data quality factors and dimensions for each level (data, model and process) and transversally.

1.5 Conclusions

1.5.1 Summary

In this chapter we have examined the relevant issues of data quality and we have presented a non exhaustive review of the current solutions, ranging from data quality dimensions specifications and measures, to techniques and tools that have been proposed mainly in the DB community for evaluating and improving data quality in different types of (multi-source) information systems.

We specifically reviewed the measures and algorithms designed respectively for: *i)* detecting and eliminating duplicate data, *ii)* handling inconsistent data, *iii)* managing imprecise or uncertain data, *iv)* handling missing or incomplete data, and *v)* improving data freshness.

We have outlined the state of art and future development of the Data Quality Research area presenting the large diversity and richness of the approaches. But we would like to emphasize the gap between the solutions provided in the literature and the complexity and multidimensionality of the data quality problematic

[6]QUADRIS: http://www.irisa.fr/quadris/

that can not be circumscribed by one-shot approaches addressing a simpler and more abstract version of the problems undoubtedly occuring in real and operational data sets.

1.5.2 Ongoing Challenges

In the context of Data Quality Research, we have identified five topics with particularly challenging research directions, namely: 1) DB and IS quality-aware engineering, 2) QoD metadata modeling and management, 3) QoD-aware query languages and processing 4) algorithmics, and 5) methodological aproaches.

1. *Database and information systems design and engineering.* The current methods for designing, (reverse-)engineering information systems have to be profoundly revised in order to integrate the evaluation and the control of the various facets of data quality, conjointly with the evaluation and the control of the quality of the conceptual data model and the quality of the system design. The challenges in the design of (multi-source) information systems concern:

   - to specificy generic and reusable patterns for integrating as a part of the system design, the techniques and tools for the evaluation and improvement of the quality of data,
   - to specify metrics and methodologies to compare the quality of conceptual data models (and design alternatives) of IS and to control the degree of consistency between the data models, the models of software architecture, and the models of communication with respect to precise data quality objectives,
   - to design relevant UML profiles and patterns in order to help with the automatic generation of modules in charge of checking and controlling data quality aspects,
   - to quantify the impact that has a data model (or system architecture) on the quality of data to be managed by the system based on this model (architecture),
   - to develop tools as internal components of the system, to measure and control the quality of data,
   - to propose recommendations for the system design (or re-engineering) according to the results of quality measurements.

2. *Quality metadata modeling.* Interesting research directions are:

   - to study and define adequate semantic models for the representation of the quality of data: 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. Indeed, the existing models focus on one or two of these dimensions separately. The approaches which consist in basing the analysis on the Cartesian product or on the *ad-hoc* composition of data quality dimensions do not make it possible to model, in a faithful way, the interdependencies between these dimensions. This led to vagueness of analysis results, difficult to conciliate and whose practical utility is very low for a complete data quality diagnostic. It is thus necessary to develop symbolic notations, abstract domains, and metadata which make it possible to represent infinite sets of configuration to control the quality of data in an effective way.

- to define formal methods and tools for validating metadata exchange and fusion models, in particular, for ensuring the interoperability of the applications, guaranteeing data quality, *a fortiori* if data is heterogeneous and multi-source, and to facilitate quality-driven data integration and fusion of quality indicators.

3. *Data and quality declaration and manipulation languages.* Numerous challenges are related to the development of quality-enhanced languages and query processors able to integrate data quality awareness in the data management system with adaptive techniques providing dynamic data quality controls and optimizing quality-constrained queries and corrective actions on data.

4. *Algorithmic approaches.* Two main challenges can be identified, such as: *i)* to propose powerful and effective approaches (in terms of recall and precision) to detect various and combined types of anomalies (for example, to detect approximate duplicates despite partially complete records) with various degrees of approximation, *ii)* with statistical metadata computation, the index may become much bigger than the data volume and it is very largely higher than the integration capacities of the RAM. The algorithms of indexing must then take into account the various levels of memory hierarchy (register, masks, discs, RAM, etc.) or be distributed, or even, the index memory must be distributed.

5. *Methodological approaches and benchmarks*: The challenges are to propose and unify recommendations, formalisms, guidelines and benchmarks to integrate, evaluate, validate, and ensure the quality of data. No approach has been proposed yet to compare and possibly unify existing methodologies applied in different application domains for data quality improvement and assessment. Some proposals are in favor of the development of a shared ontology on information quality, but pragmatic initiatives for setting up benchmarks are needed in order to evaluate and validate, in a more precise and structured way, the contributions in this research field (Weis et al., 2006).

Added to these challenges specific to Data Quality Research, the following difficulties shared by other research fields have to be considered, such as:

- *Heterogeneity and variety of abstraction levels*: the data are often extracted from different information sources, then integrated, whereas they have already various levels of abstraction (ranging from raw data to data aggregates). The data are often integrated in a system which can thus contain data resulting from a potential superposition of several statistical processes. This requires a particular care in the use and the analysis of these data. Evaluation of the quality of these data is very difficult.

- *Scalability issues*: Lots of techniques for measurement and detection of data anomalies are not scalable for the currently available volumes of data. Although some statistical methods are well adapted to provide summaries characterizing the quality of numerical data (usually for a single attribute), they prove to be inoperative on multi-tables and multi-attribute or multi-dimensional data sets. For instance, mapping or embedding a set of data objects into points in a low-dimensional vector space can make similarity search more efficient for detecting potential duplicates (Hjaltason & Samet, 2003). But these techniques such as SparseMap, FastMap (Faloutsos & Lin, 1995), MetricMap and StringMap are designed for use in an offline setting and do not satisfy the efficiency requirements of the online match operation where the input tuples have to be quickly matched with target reference tuples before being loaded into a data warehouse.

- *Management of complex data*: very few metrics, theoretical and technical solutions exist today for detecting anomalies, approximate duplicates, or inferring missing data, checking consistency on multimedia data (combining audio, text, video, and image).

- *Meta level issues*, one can wonder about the quality of metadata: the database models a portion of the real world in constant evolution. However, in same time, measurements reflecting the quality of data may become obsolete and inaccurate. The process of quality evaluation and control of data must constantly be renewed according to the dynamics of the data and the modeled real world.

Chapter 2

Quality-Aware Data Management

Contents

2.1 Introduction

For ensuring data quality awareness in data management systems, one obviously needs to consider two relative aspects in evaluating the quality of data (QoD): the **actual quality** that can be evaluated at the data source and the **expected quality** that is required by the users at the users' views. This basically means that the system has to measure actual QoD values, collect expected QoD values, compare them, and determine if they match with respect to a user-defined constraints and to a particular quality goal (e.g., get query results with no data quality problems, clean the database, load valid data in the database, etc.).

As QoD is composed of various quality dimensions, the vision and goals of the user in establishing his quality requirements are usually different from the vision of the DBA in defining data quality measures that characterize certain factors and dimensions of data quality with various measurement methods and implementation techniques.

Under these considerations, several tasks must be achieved to provide data quality awareness, e.g., the selection, computation, and management of QoD measures, the flexible declaration of the user's constraints with expected QoD values, the comparison between the actual QoD measures and the expected QoD values, and the definition of strategies to adopt in the case that actual QoD measures do not satisfy constraints with respect to expected QoD values .

Based on my work in the field, the objective of this chapter is to present contributive solutions to these tasks for ensuring data quality awareness in data management systems. In this dissertation, a special focus is on:

1. the modeling and definition of measurable properties for characterizing aspects of actual QoD. Generic and user-defined functions can be defined and implemented for measuring, describing or predicting relevant data quality properties. For this purpose, we propose the flexible declaration and potential reuse of **analytical functions** associated to the main QoD dimensions. These functions are interpreted by the query processor, executed for computing QoD measures, and used for the careful quality-aware query analysis, the preparation of alternative query evaluation plans at compile time, and the selection of optimal quality-aware query plans at runtime. Measures and descriptions of computation methods with their parameters are stored as **QoD metadata**,

2. the appropriate indexing of QoD metadata,

3. the processing of queries extended by constraints on data quality,

4. the transparent and explainable presentation of results, so that the quality-aware query results can be understood and accepted by the users.

Each of these tasks represents a challenging field of research on its own that our solutions do not pretend to cover entirely. They rather offer interesting perspectives for more research works and investigations.

Outline of the chapter. *Section 2.2 presents a running example used in this chapter to illustrate our proposals. Section 2.4 presents the QoD metadata model we propose, as an extension to Common Warehouse Metamodel (CWM). Section 2.5 presents our general approach based on the design of analytic workflows dedicated to QoD evaluation. Analytical functions used in these workflows for computing QoD measures and generating the associated QoD metadata are also presented. Section 2.6 describes the solution we've adopted for indexing QoD metadata. Section 2.7 presents XQuaL, a declaration and manipulation language extension we've designed for quality-aware query processing.*

2.2 Illustrative Example

To motivate and illustrate the next sections of the chapter, we introduce a running example of a CRM database called CRM_DB. It is composed of three tables: PRODUCT, PRICE_QUOTE, and CUSTOMER that are linked by one-to-many relationships through the PROD_ID and CUST_ID fields, as sketched in Table 2.1.

PRODUCT Table

PROD_ID	CUST_ID	P_DESTINATION	ORDER_DATE	SHIP_DATE	STATUS	PRICE
P1	C1	Reading, UK	NOV-10-2006	NOV-01-2006	CLOSED	77
P2	C2	Reading, NJ, US	NOV-10-2006	NOV-20-2006	CLOSED	77
P3	C2	Reading, NJ, US	NOV-10-2006	NOV-10-2006	ACTIVE	7777
P4	NULL	Reading, NJ, US	NULL	NOV-10-2006	CLOSED	7

CUSTOMER Table

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

PRICE_QUOTE Table

PROD_ID	PRICE_DATE	OPN	MAX	MIN	CLO
P3	NOV-09-2006	70	86	65	78
P3	NOV-10-2006	72	85	67	79
P3	NOV-11-2006	75	90	70	81

Table 2.1: CRM_DB Example

At a first glance on CRM_DB database, several data quality problems may be observed (or suspected). Null values occur in CUSTOMER and PRODUCT tables. Three records ($P1$, $P2$ and $P4$) are missing in Table PRICE_QUOTE. The price ('7777') of $P3$ in PRODUCT table is a probable outlier and it should be at least in the interval [67, 85] given for the same day in PRICE_QUOTE table. Several inconsistencies occur on STATUS values of PRODUCT table: the status should be 'CLOSED' if the SHIP_DATE is passed, otherwise 'ACTIVE' if the shipping date is planned after the current date, e.g., 'ACTIVE' status for $P3$ is inconsistent. SHIP_DATE values that are before the ORDER_DATE values are also incorrect, e.g., 'NOV-01-2006' for P_1. Of course, such constraints should have been specified when creating the database schema. Nevertheless, as we'll see further, relevant constraints (e.g., correlations or frequent itemsets) may be inferred statistically and anomaly patterns may be detected as data values deviate from the statistical rules.

Join over FK attributes considered as correct. Ignoring these observations, we assume that the data are correct, fresh, consistent, complete and without duplicates, the join of tables PRODUCT and CUSTOMER over the CUST_ID attribute would result in a table consisting of 3 rows, as shown in Table 2.2. For the sake of simplicity, only PROD_ID, CUST_IDs, P_DESTINATION and CUST_CITY are projected in the join result of tables PRODUCT and CUSTOMER, respectively renamed as P and C.

$\Pi(PRODUCT \bowtie CUSTOMER)$

PROD_ID	**P.CUST_ID**	**C.CUST_ID**	**P_DESTINATION**	**CUST_CITY**
P1	C1	C1	Reading, UK	Reading, UK
P2	C2	C2	Reading, NJ, US	Reading, NJ, US
P3	C2	C2	Reading, NJ, US	Reading, NJ, US

Table 2.2: Join over FK attributes considered as correct

Join over incorrect FK attributes. If a value of the joining field is incorrect, then, assuming that referential integrity constraints have been enforced, the join would result in a row that, while incorrect, at least should be there. To see this, suppose that in the $P3$ row of PRODUCT table, the correct value $C2$ is replaced by an incorrect value, say $C3$. Then, the join operation would generate an incorrect row for a correct one, as shown in Table 2.3.

PRODUCT Table ↓

PROD_ID	**CUST_ID**	**P_DESTINATION**	**ORDER_DATE**	**SHIP_DATE**	**STATUS**	**PRICE**
P1	C1	Reading, UK	NOV-10-2006	NOV-01-2006	CLOSED	77
P2	C2	Reading, NJ, US	NOV-10-2006	NOV-20-2006	CLOSED	77
P3	~~C2~~ $\boxed{\textbf{C3}}$	Reading, NJ, US	NOV-10-2006	NOV-10-2006	ACTIVE	7777
P4	NULL	Reading, NJ, US	NULL	NOV-10-2006	CLOSED	7

CUSTOMER Table

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

$\Pi(PRODUCT \bowtie CUSTOMER)$

PROD_ID	**P.CUST_ID**	**C.CUST_ID**	**P_DESTINATION**	**CUST_CITY**	
P1	C1	C1	Reading, UK	Reading, UK	
P2	C2	C2	Reading, NJ, US	Reading, NJ, US	
P3	$\boxed{\textbf{C3}}$	$\boxed{\textbf{C3}}$	Reading, NJ, US	$\boxed{\textbf{Reading, UK}}$	←

Table 2.3: Join over incorrect FK attributes

Join over non FK attributes considered as correct. If the joining field is not a foreign key, then the result could contain for each incorrect joining value multiple rows that would not exist. For instance, in Table 2.1, consider joining the two tables over the P_DESTINATION field of PRODUCT table and the CUST_CITY field of CUSTOMER table. The result would yield five rows, as illustrated in Table 2.4.

$\Pi(PRODUCT \bowtie CUSTOMER)$

PROD_ID	**P.CUST_ID**	**C.CUST_ID**	**P_DESTINATION**	**CUST_CITY**
P1	C1	C1	Reading, UK	Reading, UK
P1	C1	C3	Reading, UK	Reading, UK
P2	C2	C2	Reading, NJ, US	Reading, NJ, US
P3	C2	C2	Reading, NJ, US	Reading, NJ, US
P4	NULL	C2	Reading, NJ, US	Reading, NJ, US

Table 2.4: Join over non FK attributes considered as correct

Join over incorrect non FK attributes. Now assume that the value 'Reading, NJ, US' in the $P3$ row of PRODUCT table is incorrectly replaced by the value 'Reading, UK'. The resulting join would now have six rows as shown in Table 2.5, two of which would be incorrect.

PRODUCT Table

PROD_ID	**CUST_ID**	↓ **P_DESTINATION**	**ORDER_DATE**	**SHIP_DATE**	**STATUS**	**PRICE**
P1	C1	Reading, UK	NOV-10-2006	NOV-01-2006	CLOSED	77
P2	C2	Reading, NJ, US	NOV-10-2006	NOV-20-2006	CLOSED	77
P3	C2	~~Reading, NJ, US~~ $\boxed{\textbf{Reading, UK}}$	NOV-10-2006	NOV-10-2006	ACTIVE	7777
P4	NULL	Reading, NJ, US	NULL	NOV-10-2006	CLOSED	7

CUSTOMER Table

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

$\Pi(PRODUCT \bowtie CUSTOMER)$

PROD_ID	**P.CUST_ID**	**C.CUST_ID**	**P_DESTINATION**	**CUST_CITY**	
P1	C1	C1	Reading, UK	Reading, UK	
P3	C2	$\boxed{\textbf{C1}}$	Reading, UK	$\boxed{\textbf{Reading, UK}}$	←
P1	C1	C3	Reading, UK	Reading, UK	
P3	C2	$\boxed{\textbf{C3}}$	Reading, UK	$\boxed{\textbf{Reading, UK}}$	←
P2	C2	C2	Reading, NJ, US	Reading, NJ, US	
P4	NULL	C2	Reading, NJ, US	Reading, NJ, US	

Table 2.5: Join over incorrect non FK attributes

Another type of error in that field could lead to multiple missing rows. To see this, consider the case where, in $P3$ row of the PRODUCT table, the value 'Reading, NJ, US' is incorrectly replaced by 'London'. In this case, the resulting join would four rows instead of five.

Join over FK attribute of duplicates. Now consider that $C1$, $C2$, and $C3$ records of CUSTOMER table have high probability of being duplicates. They may actually refer to the same person who used to leave in UK but recently moved to New Jersey, USA. To get a correct answer to the previous query joining FK attributes of PRODUCT and CUSTOMER tables, one would first merge duplicates $C1$ and $C3$ and keep the single record $C2$, for example, with considering timeliness of data values for CUST_CITY, and then propagate $C2$ value as FK instead of $C1$ in $P1$

row of PRODUCT table. The correct answer is illustrated in Table 2.6 instead of the one given in Table 2.2 for P1.

PRODUCT Table ↓

PROD_ID	CUST_ID	P_DESTINATION	ORDER_DATE	SHIP_DATE	STATUS	PRICE
P1	~~C1~~ $\boxed{C2}$	Reading, UK	NOV-10-2006	NOV-01-2006	CLOSED	77
P2	C2	Reading, NJ, US	NOV-10-2006	NOV-20-2006	CLOSED	77
P3	C2	Reading, NJ, US	NOV-10-2006	NOV-10-2006	ACTIVE	7777
P4	NULL	Reading, NJ, US	NULL	NOV-10-2006	CLOSED	7

CUSTOMER Table

CUST_ID	FN	LN	CUST_CITY	SHIP_TAX	
~~C1~~	~~Joe~~	~~Smith~~	~~Reading, UK~~	~~10.56~~	←
C2	Joy	Smith	Reading, NJ, US	0.30	
~~C3~~	~~John~~	~~Smitt~~	~~Reading, UK~~	~~NULL~~	←

$\Pi(PRODUCT \bowtie CUSTOMER)$

PROD_ID	P.CUST_ID	C.CUST_ID	P_DESTINATION	CUST_CITY	
P1	$\boxed{C2}$	$\boxed{C2}$	Reading, UK	$\boxed{\text{Reading, NJ, US}}$	←
P2	C2	C2	Reading, NJ, US	Reading, NJ, US	
P3	C2	C2	Reading, NJ, US	Reading, NJ, US	

Table 2.6: Join over Deduplicated FK attributes

Join over non FK attributes of duplicates. The result of joining PRODUCT and CUSTOMER over the non FK attributes P_DESTINATION and the CUST_CITY once the duplicates have been removed in CUSTOMER table would be composed of three rows (instead of five rows) as illustrated in Table 2.7.

$\Pi(PRODUCT \bowtie CUSTOMER)$

PROD_ID	P.CUST_ID	C.CUST_ID	P_DESTINATION	CUST_CITY
P2	C2	C2	Reading, NJ, US	Reading, NJ, US
P3	C2	C2	Reading, NJ, US	Reading, NJ, US
P4	NULL	C2	Reading, NJ, US	Reading, NJ, US

Table 2.7: Join over Deduplicated non FK attributes

Considering the CRM_DB example, suppose we can characterize and compute the accuracy probability of each joining attribute value. Intuitively, we could use it to estimate the accuracy probability of the result of join over FK and non FK attributes. The main idea is to compute the accuracy probability for each cell of the tables to join and to propagate probabilities to the result considering the fact that if the join is based on attribute values with high probabilities of being inaccurate, then the accuracy probability of the result should be proportionally penalized. Similarly, we apply this intuitive idea to any other data quality dimension.

2.3 General Approach

From the CRM_DB example, several general observations can be made from the user's perspective:

1. when querying the database, the user doesn't want to ignore data quality problems, but rather to be informed of; the system should provide this additional information when probable data anomalies are detected or when they are previsible in the query result,

2. the user may want to exclude from the query result data that does not meet data quality requirements in terms of freshness, accuracy, consistency, completeness, and uniqueness (*i.e.*, absence of duplicates),

3. the user may want to choose the level of granularity or the database object instances on which his data quality requirements and constraints will be applied,

4. the user should be able to declare his own data quality requirements as data quality factors and measurement methods from a panel of functions he can reuse, adapt or extend with new functions for evaluating various aspects of QoD dimensions on the database

From the system perspective, in our approach, these requirements have been translated into three main components for including quality-awareness in the data management system: *i)* an extensible library of functions for measuring various aspects characterizing aspects of QoD dimensions, *ii)* a metadata manager that stores, indexes, searches and resfreshes QoD measures and related descriptive metadata, and *iii)* a query engine that allows declaration and manipulation of data and constraints on the associated QoD metadata.

Our general approach is centered on the design of **analytic workflows** and based on five steps:

1. **Definition or reuse of analytical functions for QoD evaluation**. The analytic workflow first consists of the definition, selection, and reuse of the functions that measure objectively several (possibly user-defined) factors for characterizing QoD dimensions of database object instances.

2. **Computation of QoD measures**. The computed measures are stored and managed as QoD metadata in a repository. Several QoD measures characterize one QoD dimension, which they are associated to.

3. **Definition of probabilistic and approximate constraints on QoD measures**. Because actual QoD measures have to be compared to QoD values expected by the user, constraints on actual QoD measures are declared and used for searching actual QoD metadata that best satisfy the constraints or that are the most similar to expected QoD values.

4. **Constraints checking and probabilities assignment**. Checking the constraints on the QoD measures associated to one QoD dimension results in the computation of the probability that the QoD dimension is acceptable (exactly or with a certain degree of approximation). The computed probabilities are also stored in the metadata repository and refreshed by the system as the constraints may change.

5. **Quality-extended query declaration**. Probabilities assigned to QoD dimensions are used in the query processing to build quality-aware results that are in conformance with the QoD constraints defined by the user.

In our approach, the central components of a quality-aware data management system are: *i)* the QoD metadata manager, *ii)* the analytical function library and analytic workflows manager, and *iii)* the quality-extended query engine.

In the next sections, we zoom in on the main features of these key components, starting with a focus on the conceptual model underlying the management of QoD metadata in the repository (Section 2.4). Then, we present in details the analytical functions we use in the design of analytic workflows dedicated to QoD evaluation (Section 2.5). Since QoD measures and descriptive metadata are stored in the metadata repository, we present the method adopted for indexing and searching these metadata (Section 2.6). Finally, the syntax of XQuaL, a query language extension is presented (Section 2.7).

2.4 Modeling Quality Metadata

Metadata plays an important role in information systems as it usually contains data dictionary with the definitions of the databases being maintained and the relationships between database objects, data flows, data transformations, and data usage statistics. One important problem is to handle complexities of the different views of metadata (possibly integrated from disparate sources) and to provide the infrastructure with standard access mechanisms and application programming interfaces that enables users to control and manage the access, interchange, and manipulation of data and metadata as well. Defining and managing new kinds of metadata is also an important requirement in the design of a metadata repository. In the last decade, several specifications, e.g., *MDIS - Metadata Interchange Specification, CWM - Common Warehouse Metamodel* [1] (OMG, 2003; Poole et al., 2003), languages, e.g., *Telos* (Mylopoulos et al., 1990), and implementations, e.g., *Microsoft Repository* (Bernstein et al., 1999), *CWM MetaStore* (Poole et al., 2003) have been proposed for these purposes.

The storage and exploitation of metadata requires an appropriate modeling framework that allows data representation with the logical and physical views gathering the complete set of metadata describing the available data resources expressed in different paradigms (e.g., object-oriented, relational, XML, or multidimensional data) and also the transformations and data mining processes that may be applied on these data resources.

In this context, considering our modeling requirements, it appears to be particularly relevant to use and extend CWM metamodel for the management of QoD metadata: first, because CWM metamodel covers the description of the complete range of resources available through different technologies and secondly, because CWM integrates the complete description of data mining functions and processes that we instantiate in our analytic approach for QoD evaluation.

[1] CWM Specification, v1.1, http://www.omg.org/technology/documents/formal/cwm.htm

2.4.1 The CWM Metamodel

Today, as a metadata integration standard for data warehouse field proposed by Object Management Group (OMG), CWM has been accepted as a prevailing standard for interchanging metadata between warehouse tools, warehouse platforms and warehouse metadata repositories in distributed heterogeneous environments.

Considering the OMG metadata model architecture, CWM is a complete M2-level layered metamodel divided in a number of different but closely related metamodels. CWM metamodel is based on the use of shared metadata expressed with the following technologies:

- *MOF (Meta Object Facility)*, an OMG metamodeling standard that defines an extensible framework for defining models for metadata, and providing tools to store and access metadata in a repository,
- *UML (Unified Modeling Language)*, an OMG modeling standard that defines a rich, object-oriented modeling language,
- *XMI (XML Metadata Interchange)*, an OMG metadata interchange standard that allows metadata to be interchanged as streams or files with an XML format.

CWM is organized in 21 separate packages for metadata generation and management grouped into five stackable layers as illustrated in Figure 2.1.

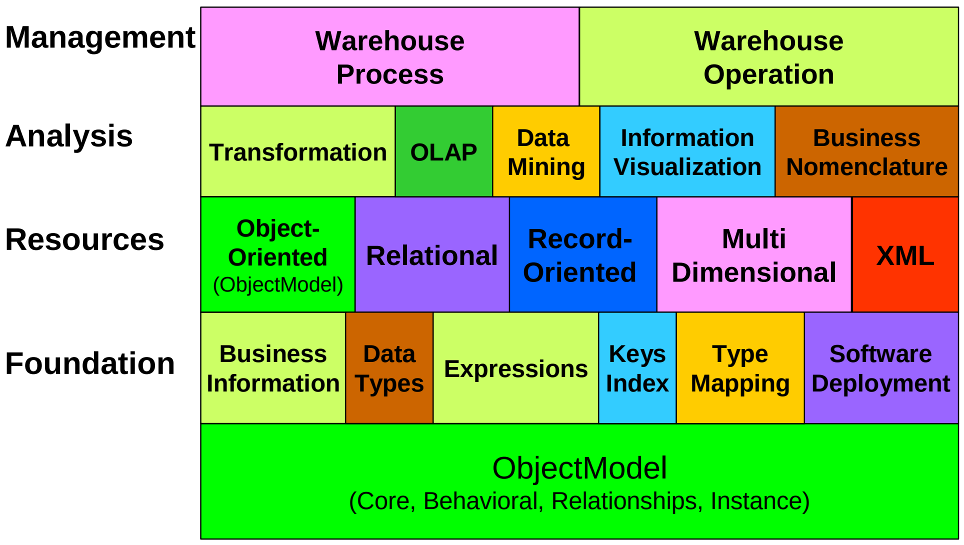


Figure 2.1: CWM Packages

These layers cover the complete *Information Supply Chain* (ISC) for business intelligence applications. The layers of the CWM integrate together different sorts of packages:

1. the *Foundation* layer contains general services that are shared by other packages, defining business information, data types, expressions, indexing scheme, type mapping and software deployment,

2. the *Resources* layer contains data metamodels (object-oriented, relational, record-oriented, XML, and multidimensional) used for the description of data sources,

3. the *Analysis* layer provides metamodels supporting logical services that may be mapped onto data stores defined by *Resources* layer packages. For example, the *Transformation* metamodel supports the definition of transformations between data warehouse sources and targets. *Data Mining* metamodel offers a complete description of the elements involved in a data mining process (e.g., mining functions, settings, parameters, etc.),

4. the *Management* layer metamodels support the operation of data warehouses by allowing the definition and scheduling of operational tasks (*Warehouse Process* package) and by recording the activity of warehouse processes and related statistics (*Warehouse Operation* package).

As illustrated in Figure 2.1, each layer uses respectively the core model elements (e.g., data types, index, expressions), the resources available in different technologies and formats (object-oriented, relational DB, record-oriented, multidimensional or XML), the operations and transformations used for data analysis, and the management of the data warehouse.

In this section, we present how we extend and instantiate the CWM metamodel to include and associate QoD metadata to the *Data Instance* metamodel. We illustrate our proposition by examples in the relational context, but this approach can be easily applied to any other type of resource using the appropriate packages of CWM metamodel from the *Resources* layer.

CWM has been specified for data integration systems and particularly data warehousing systems but it can be easily applied to any other MSIS and also to conventional and centralized DB.

Our examples are based on CWM *Relational* metamodel that is presented in Figures A.1 and A.2 in Annexes, pages 152-153). For a complete and detailed description of CWM metamodel, the reader is invited to read the detailed CWM specification[2] (OMG, 2003; Poole et al., 2003).

Example 1. *Considering the CRM_DB database, the instance of the CWM logical model is given in Figure 2.2. It characterizes the relational database object instances of CRM_DB database.*

A key aspect of our approach is to privilege the use analytical functions to compute QoD measures and generate QoD metadata for characterizing various aspects of QoD dimensions. As we'll see in Section 2.5, analytical functions are basic counts, summary statistics, and data mining techniques. Their results provide

[2]CWM Specification, v1.1, http://www.omg.org/technology/documents/formal/cwm.htm

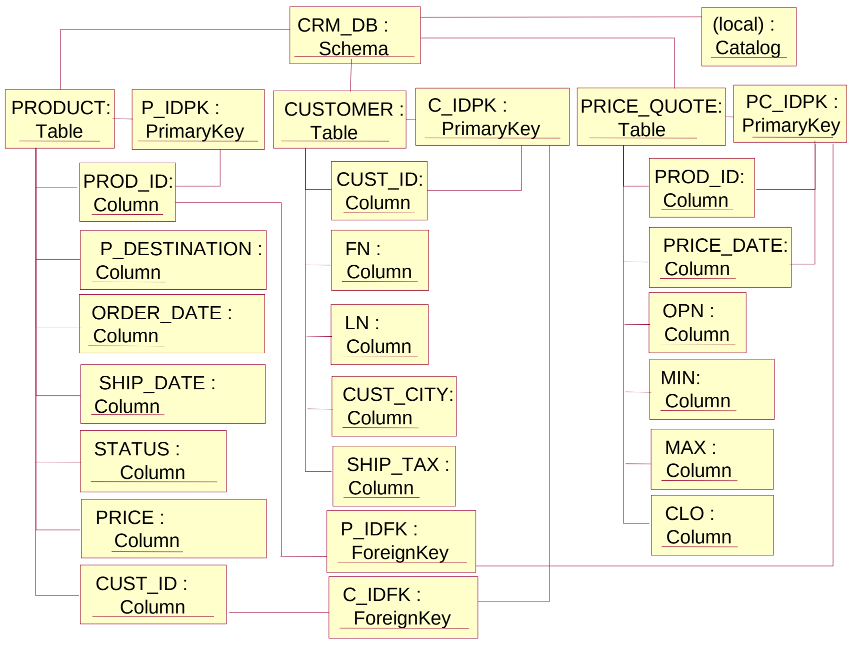


Figure 2.2: Relational Metamodel of CRM_DB

additional descriptive information that is manifest in inherent patterns or relations between the data and that can be used for QoD evaluation.

As previously mentioned, CWM metamodel includes a relevant *Data Mining* metamodel that is composed of seven conceptual areas: A *Core Mining* metamodel (upon which the other areas depend), and metamodels representing the data mining subdomains of *Clustering*, *Association Rules*, *Supervised*, *Classification*, *Approximation*, and *Attribute Importance*. Each area is represented by the *Data Mining* metamodel packages. Each package defines the basic *Mining Model* from which all model objects inherit as the result of a mining build task. The *Mining Model* metamodel is illustrated in Figures A.3 in Annexes, page 154. Each *Mining Model* has a signature that defines the characteristics of the data required by the model. In our approach, we instantiate CWM *Mining model* with a panel of functions that characterize various aspects of QoD dimensions.

2.4.2 CWM Extension for QoD Metadata Management

Because QoD measures can only be computed from instances of database objects with respect to a particular quality goal, we have extended CWM *Data Instance* metamodel that describes the instances of various model of resources (e.g., object-oriented, XML, relational, multidimensional, record-oriented). Figure 2.3 illustrates CWM *Data Instance* metamodel with specialization for relational resource instances.

Data values may be stored in one of two alternative ways. The *Slot class* is a generic container that can hold either *DataValue* or *Object* instances what ever the resource type is. One way of storing a data value is to create a *Slot* instance and place a *DataValue* instance ’in’ the Slot via the *SlotValue association*. The alternate way is to create an instance of *DataSlot*, storing the value into its *dataValue attribute*. The former method is more general while the latter creates fewer total objects.

Modeling the relational resource instances, Figure 2.3 shows how a *Rowset* inherits from *Extent*, from the *Foundation* package. It represents all the data comprised in a *ColumnSet*. A *RowSet* can only be owned by a *ColumnSet* or any derived class. A *RowSet* contains *Rows*. *Row* inherits from *Object*. Its structure is defined by the corresponding *ColumnSet* and its *Columns* from the relational metamodel given in Annexes, Figure A.1 and A.2. (pages 152 et 153). Each *Row* is divided into *ColumnValues*, which match the value of a relational table, at the intersection of a row and a column. *ColumnValue* inherits from *DataValue* from *ObjectModel*.

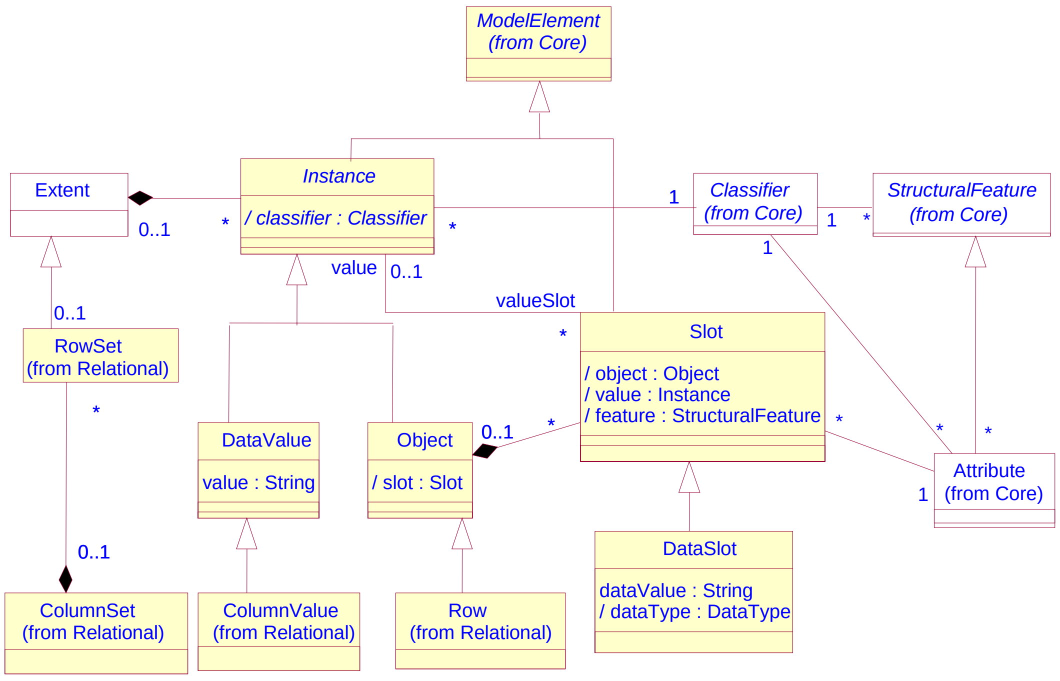


Figure 2.3: CWM Relational Data Instance Metamodel (OMG, 2003)

The quality extension we propose coherently supplements the CWM *Data Instance* metamodel. QoD measures that correspond to specific quality requirements and goals are defined over the *Extent* of a specific element of the CWM instance metamodel. In our case for the relational resource layer, a QoD measure can be defined over one specific database object instance, namely over the schema, table, column, rowset, columnset, column, row, and slot (*i.e.*, attribute value).

QoDMeasure class incorporates the name of the measure, the resulting measured value, the method that is employed for computing the measure, the computation timestamp, and the constraint with the expected value. The measurement methods are defined as classes from other packages of CWM metamodel: QoD measures may be computed from *MiningFunctions*, *Triggers*, stored *Procedures*, or *Constraints*.

Quality dimensions are used as the vocabulary to define abstractly various aspects of quality, as the user may perceive it. Of course, each user might have a different vocabulary, different preferences and goals underlying quality specifications in the various dimensions of quality. In order to aggregate the QoD measures related to one quality dimension, we define the *QoDDimension* class composed of the associated *QoDMeasures*. QoD dimension has a name, a scoring method, and a *acceptability* value. The scoring method is used to compute the acceptability value with possibly weighting the degree of constraint satisfaction of the QoD measures associated to the QoD dimension and compared to expected values. Acceptability of a QoD dimension evaluated from a data object instance is the probability that each value of its associated QoD measures satisfies the specified constraint.

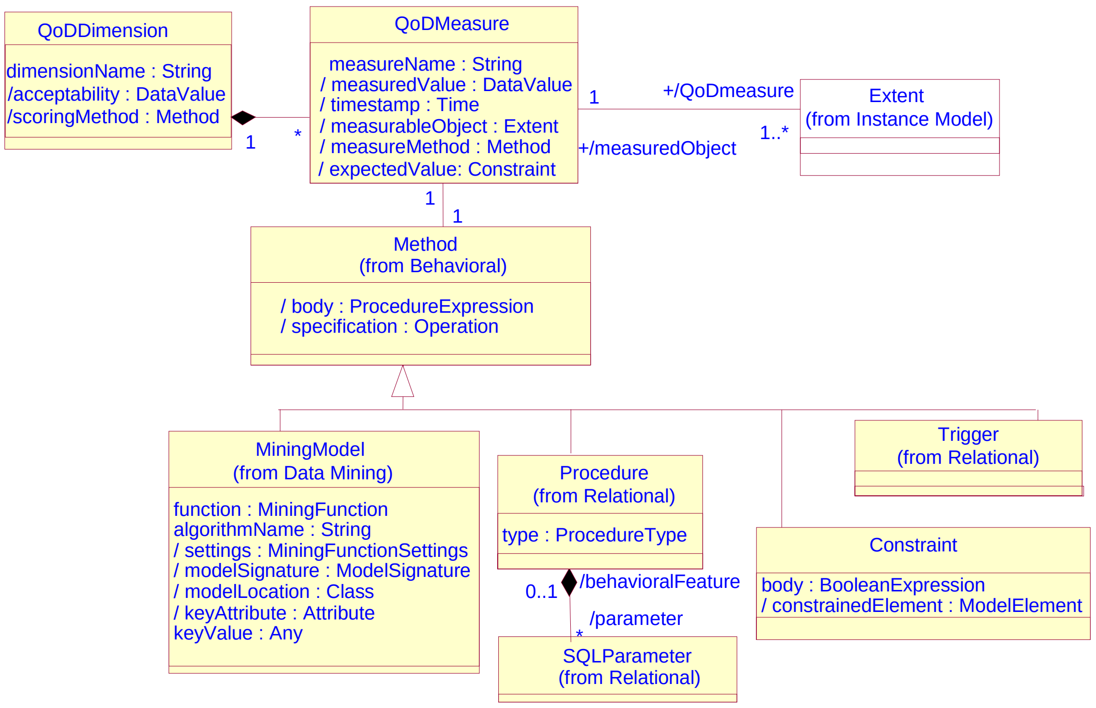


Figure 2.4: QoD Extension to CWM Data Instance Metamodel

Example 2. *For PRODUCT table of CRM_DB example, an instantiation of the extended CWM metamodel includes QoD measures stored as QoD metadata in the metadata repository. For the sake of clarity, only a subset of QoD measures is illustrated in Figure 2.5.*

Each QoD measure is associated to only one QoD dimension and one database object instance. For example, for PRICE column, nullValues% is a QoD measure associated to Completeness QoD dimension. This QoD measure is calculated at a given time (May 15th, 2006) by procedure plsql_id6 and returns the value 75% whereas the expected value for this measure equals 100%.

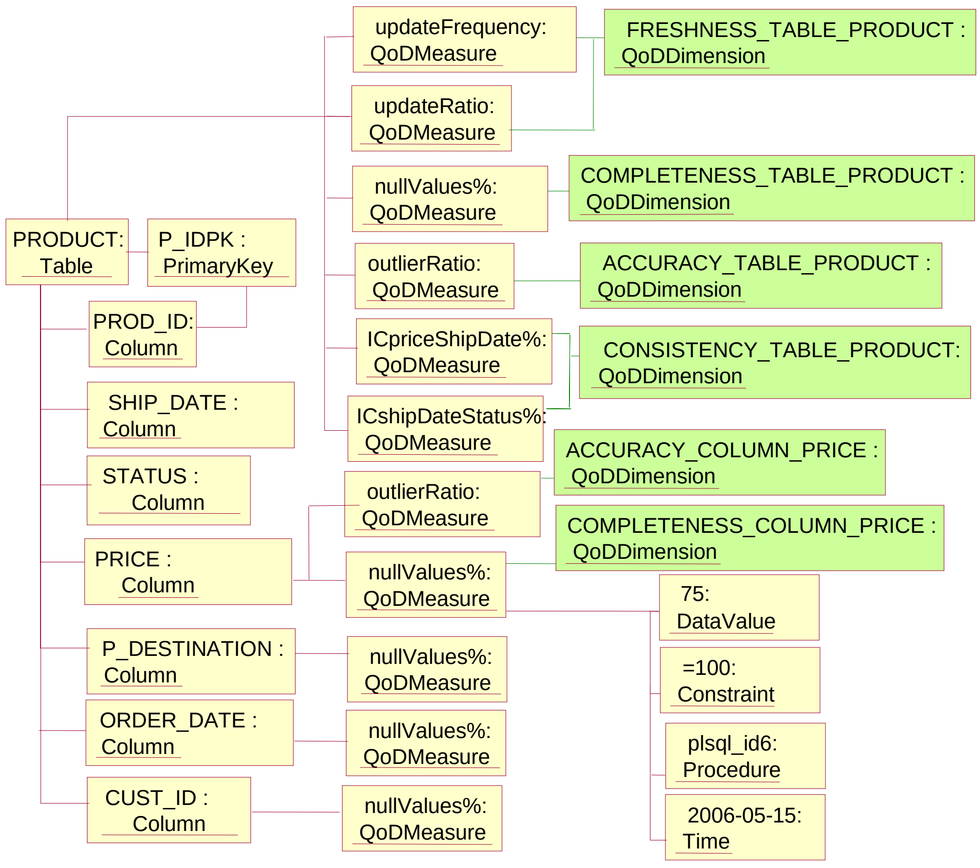


Figure 2.5: Example of QoD Metadata Associated to CRM_DB PRODUCT table

In this example, we observe that QoD measures associated to one DB object instance are computed by user-defined functions (e.g., *plsql_id6* procedure), they are associated to one QoD dimension. QoD dimensions may have many associated measures. A constraint is based on the comparison between actual and expected values for each QoD measure composing the QoD dimension. The degrees to which these constraints are satisfied are aggregated in order to compute the acceptability value assigned to the QoD dimension. This represents the probability that the QoD dimension is acceptable considering that the values of its associated QoD measures are in conformance with expected values defined in their respective constraints.

In the example, FRESHNESS_TABLE_PRODUCT, the acceptability of freshness dimension for PRODUCT table may be computed by a scoring function taking as input the values of two QoD measures (`updateFrequency` and `updateRatio)`). Once the constraint defined in each of these QoD measures are checked (comparing actual QoD measure values with expected values), the degree of satisfaction of the constraint is used as input by the scoring function to compute the acceptability value of the dimension for the database object instance PRODUCT.

2.5 Computing Quality Metadata

To evaluate the quality of data and the quality of possible query results, measures that characterize the quality of database object instances are computed and compared to expected values specified in the QoD measure specification. Again, each QoD measure is associated to one particular QoD dimension. QoD measure values are results from functions, we called **analytical functions**, that can be precomputed or computed on-demand after the arrival of a query.

The consideration of data quality requirements (expressed as constraints with expected values) may drive the query planning and it can be taken into account for building advantageously a quality-aware query result.

On-demand computation within the query execution is superior to precomputation in that suitable query plans may be determined using the latest QoD measurement of the database object instances. However, the delay incurred in computing QoD measures and determining a suitable quality-optimal query plan on-demand may be unacceptable. Quality measures precomputation (or off-line computation) overcomes the drawback of long delay of on-demand computation. In addition, quality measures precomputation may be preferred over on-demand computation for scalability issues, when the data volume or the number of quality metadata and constraints is very large.

The work presented in this section mainly focuses on the off-line computation techniques for QoD measurement.

For QoD evaluation, we mainly focus on QoD measures generated by non supervised techniques for data analysis, ranging from basic counts, statistics, and synopses computation to sketch-based computation and data mining techniques. These techniques are used to characterize several properties of data distribution and data quality. The resulting QoD measures and the detailed description of the analytical function (e.g., settings, parameters, etc.) are stored as QoD metadata in the repository.

For the sake of clarity and for the progressive description of our approach, we propose a classification of the analytical functions we use in four levels based on the function abstraction level and its computation cost:

Level I - QoD Profiling Functions are 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).

Level II - QoD Constraint-Based Functions can be specified as a set of global or application-specific integrity rules and 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.

Level III - QoD Synopses Functions are 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.

Level IV - QoD Exploratory Mining Functions are 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., outliers, duplicates, anomaly patterns, and dubious data), but their computation and maintenance costs are much higher.

In the following subsections, we review the analytical functions we use to compute relevant QoD metadata for each of the aforementioned levels.

2.5.1 Level I: QoD Profiling Functions

Data quality profiling is the process of analyzing quantitatively a database to identify and prioritize data quality problems. The results include simple summaries (mainly counts) describing completeness of fields and records, data freshness, and data problems existing in records (e.g., invalid format).

The structure and internal relationships of the database are analyzed to determine basically:

- the tablespaces, tables and fields used, and the keys (PK and FK),
- the number of rows, distinct values, extreme values, and popular values for each field,
- relationships between tables (*i.e.*, functional dependencies),
- columns copied or derived from other columns, and data lineage,

- how tables and columns are populated and changed, using history logs,

- the access frequency of the tables and columns (e.g., to identify the most cache intensive SQL statements).

Table 2.8 provides examples of the basic Level I functions we use for computing measures related to the following data quality dimensions: completeness (CP), consistency (CT), and freshness (F). The computed measures are respectively associated to the instances of the level of granularity which the measure is computed from.

DQD	Function	Output	DB Objects
CP	nullValues%	Returns a percentage that represents the quantity of null values in the instances of the DB object	D,T,R,A
CT	Rare Value	Returns the rare values of the attribute	A
	Extreme Value	Returns the extreme values of the attribute	A
F	Extraction Period (Currency)	Returns a decimal that represents the quantity of hours passed between the time when the instance of DB object instance has been extracted from its original source.	D,T,R,V
	Age	Returns a decimal that represents the quantity of hours passed between the time when the instance of DB object has been delivered and the time when DB object instance has been created.	D,T,R,V
	Last Update Period (Timeliness)	Returns a decimal that represents the quantity of hours passed between the time when the instance of DB object has been delivered and the time when it has been updated.	D,T,R,V
	Obsolescence	Returns a number of updates of the instance of DB object since the extraction time from its original source.	T,R,A,V
	Freshness Ratio	Returns a decimal that represents the percentage of DB subobject instances that have not been updated since the extraction time.	D,T,R,A
	Update Ratio	Returns a decimal that represents the percentage of DB subobject instances that have been not updated since the creation time.	D,T,R,A
	Average Loading Period	Returns a decimal that represents the average quantity of hours passed between the starting times of the loading processes (period between loadings).	D,T,R
	Average Loading Duration	Returns a decimal that represents the average quantity of hours that passes between the starting times of the loading processes and their ending times.	D,T,R
	Last Loading Time	Returns the date and time when the last loading process was started for the instance of DB object.	D,T,R,V
	Last Update Time	Returns the date and time since the last update time for the instance of DB object.	D,T,R,V
	Update Frequency	Returns a decimal that indicates the number of updates since the creation time for the instance of DB object.	D,T,R,V
	Loading Frequency	Returns a decimal that indicates the number of loadings since the creation time for the instance of DB object.	D,T

Table 2.8: Examples of Level I Basic Functions

As previously mentioned, we use five granularity levels for the assignment of QoD metadata to relational database object instances. They correspond to the main structural elements of the relational model: metadata are thus associated to each DB object instance depending on its granularity level, namely the value (V), record (R), attribute (A), table (T) and database (D) granularity levels (see the last column in Table 2.8).

2.5.2 Level II: QoD Constraint-Based Functions

In order to enhance accuracy and correctness of data through integrity checking, we use three types of constraints:

i) *conventional integrity constraints (IC)* including key constraints, referential constraints, and attribute domain consistency constraints defined in the database schema,

ii) *consistency constraints (CC)* defined by *ad-hoc* and application-specific rules,

iii) *statistical constraints (SC)* defined by the conformance of data values to statistical or probabilistic models (e.g., linear regression model).

A database, as a model of some parts of the real world, is expected to faithfully reflect reality. Therefore, it is likely to see statistical relationships existing among attributes of a relation as they are often embodied in the real world. Unlike conventional integrity constraints used for consistency checking, which specify the ranges of legal attributes values (perhaps with regard to legitimate combinations of attribute values), statistical constraints manifest embedded relationships among attributes values. They may be used to detect potential errors not easily detected by the conventional ICs or attributes dependencies.

The term accuracy is frequently used to describe many different aspects related to the data itself. Semantic correctness, syntactic correctness, precision and fuzziness are some of the concepts related to accuracy (Peralta, 2006). In our context, we consider the definition of accuracy as the probability of an attribute value to be correct and we mainly use SCs to verify and evaluate the accuracy over attributes values of the database.

In order to illustrate this idea, consider a set of X attribute values. One can estimate the corresponding Y value and draw inference about whether a combination of X and Y values is likely or unlikely to be correct. Under a linear regression model, an estimator $\hat{y}$ of Y can be computed together with a confidence interval and a confidence level. The confidence interval is an interval of plausible values for the parameter being estimated and the confidence level is the degree of plausibility of such an interval, noted α. Since statistical constraints are probabilistic constraints, the determination of accuracy can be practiced at various degrees of strictness for α, $0 \leq \alpha \leq 1$. A most plausible interval (for numerical dependent attribute) or a most probable set of values (for categorical dependent attribute) can be computed. This interval (or set of values) is expected to cover the set of correct values with probability $1 - \alpha$. As α increases, the more strict the system is or the smaller the discrepancy between attribute value and expected value will be tolerated. For numerical dependent attributes, the decision rule is that if the attribute value y of a tuple does not fall within the interval defined by: $[\hat{y} \pm t_{\alpha/2,n-m}\sigma_{\hat{y}}]$, then it is inaccurate with m, the number of explanatory attributes of X involved in the relationship; n, the number of sample tuples considered; $\hat{y}$, the estimated dependent attribute value of Y explained by X values, perhaps with some function performed on it; $\sigma_{\hat{y}}$, the standard error of $\hat{y}$, and $t_{\alpha/2,n-m}$, the t distribution with parameters $\alpha/2$ and $n - m$.

Example 3. *In CRM_DB example, a statistical constraint SC may indicate the most probable SHIP_TAX charged to a customer buying a product depending on the distance to shipping destination P_DESTINATION and the PRICE value of the product. Assume that it has been found that SHIP_TAX (the dependent attribute) is closely related to a function $dist$ on P_DESTINATION and PRICE (the explanatory attributes).*

The SHIP_TAX relationship can be approximated by:
$\log(SHIP\_TAX) = \beta_1 dist(P\_DESTINATION) + \beta_2 PRICE.$

Generic examples of Level II Functions we use for characterizing aspects of consistency (CT) and accuracy (AC) dimensions are given in Table 2.9.

Function	**Output**	**DB Objects**
IC Violation	Returns a Boolean that is true when IC associated to the DB object instance is not satisfied.	D,T,A,R,V
SC Violation	Returns a Boolean and a set of parameters: the Boolean is true when SC associated to the DB object instance is not satisfied.	D,T,A,R,V
Semantic Correctness	Returns a decimal number between 0 and 1 that represents the probability that the instance of DB object is correct and correctly represents the real world.	R,V
Syntactic Correctness	Returns a decimal number between 0 and 1 that represents the percentage of format discordances, syntactical errors and misspellings of the instance of DB object.	D,T,A,R,V
Control Charts	Return univariate and bivariate plotted statistical summaries collected from several samples with acceptable bound (σ-limits) that defines the control regions and out of control regions with potential outliers (e.g., R-Chart,$\hat{\chi}$-chart)	A
Frequency Table	Considering two attribute domains divided into intervals, frequency table presents the counts of data values that fall into the corresponding intervals	A
Cumulative Distribution Function (CDF)	Returns the proportion of data values N_p whose value a falls below any given value t of the attribute A, such as the estimate is: $\widehat{CDF(t)} = \frac{N_p \mid a \leq t}{N_{total}}$	A

Table 2.9: Examples of Level II Functions

To complete Table 2.9, we provide several ways on how to define various SCs. For example, to monitor the relationship between two attributes that should roughly be linearly correlated with each other, the well-known Pearson correlation coefficient can be used to describe this relationship.

The regression control chart contains a regression line that summarizes the linear relationship between the two variables of interest. Around the regression line a confidence interval is established within which we would expect a certain proportion (e.g., 95%) of samples to fall. Outliers in this plot may indicate samples where, for some reason, the common relationship between the two variables of interest does not hold. Other types of quality control charts may be used. They are often classified according to the type of characteristics they are supposed to monitor, e.g.:

- in *X-bar chart*: the sample means are plotted in order to control the mean value of a variable.
- in *R chart*: the sample ranges are plotted in order to control the variability of a variable.
- in *S chart*: the sample standard deviations are plotted in order to control the variability of a variable.

Frequency or one-way tables represent the simplest method for analyzing categorical (nominal) data. They are often used as one of the exploratory procedures to review how different categories of values are distributed in the data sample. Customarily, if a data set includes any categorical data, then one of the first steps in the data analysis is to compute a frequency table for those categorical variables. Crosstabulation is a combination of two (or more) frequency tables arranged such that each cell in the resulting table represents a unique combination of specific values of crosstabulated variables. Thus, crosstabulation allows us to examine frequencies of observations that belong to specific categories on more than one attributes. By examining these frequencies, we can identify relations between crosstabulated variables. Only categorical (nominal) variables or variables with a relatively small number of different meaningful values should be crosstabulated. Note that in the cases where we do want to include a continuous variable in a crosstabulation (e.g., PRICE), we can first recode it into a particular number of distinct ranges (e.g., low, medium, high). The values in the margins of the table are simply one-way frequency tables for all values in the table. They are important in that they help us to evaluate the arrangement of frequencies in individual columns or rows. The differences between the distributions of frequencies in individual rows (or columns) in the respective margins informs us about the relationship between the crosstabulated attributes. This information can be used for specifying SCs and detect potential data anomalies.

Other tests, univariate, bivariate, and multivariate statistics can be used to characterize data distribution properties, preliminarily to the evaluation of consistency and accuracy by means of SCs. These are included in the set of Level II QoD functions whose examples are presented in Table 2.10.

2.5.3 Level III: QoD Synopses Functions

Response times and scalability issues are very important for quickly generating summaries and statistical metadata over very large data sets with multiple variables. That's the reason why we've considered the techniques proposed for approximate query processing, focusing on specific forms of aggregate queries. The crucial advantage of these techniques is the employed data reduction mechanism to obtain synopses of the data. The methods explored in our context include sampling, histograms, and string similarity sketches. They provide preliminary analysis results or reduce the scope of the analysis as they are a very useful way to characterize aspects of QoD dimensions such as uniqueness (for approximate duplicate detection) and consistency (for outlier detection).

Function	Definition
Measures of location	Typical Values that are representative of the population (mean, median, mode)
Measures of dispersion	Values that quantify the extent of spread of the attribute around the core or typical value (variance, co-variances, dispersion matrix, standard and absolute deviation, quantile, range, IQR)
Skewness coefficient	Value that quantifies the form and symmetry of the attribute value distribution
Kurtosis coefficient	Value that quantifies the form and flatness of the attribute value distribution
Homoscedasticity	Variance equality of an attribute on several samples: e.g., Levene test (nonnormality robust), Bartlett test for normal distribution, or Fisher test.
Normality tests	Nonparametric tests that validate (or not) the hypothesis that the attribute value distribution is a normal distribution (Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors, Cramer-von Mises, Anderson-Darling tests)
Measures of attribute relationships	Simple summaries such as counts and sums used to infer inter-relationships between attributes: e.g., correlation coefficient for linear association between two attributes, Q-Q plots for characterizing the shape of an attribute distribution, nonparametric Chi-Square test for characterizing the independency of two categorical attributes or exact Fisher test and contingency table.
(M)ANOVA tests	For $k \geq 3$ data samples (with normality and homoscedasticity), ANOVA method tests the equality of means of continuous variables that can be dependent (ANOVA) or independent (MANOVA). It generalizes the Student test and is used to compare data sets.
Factor Analytic Methods	Methods form a correlation matrix of similarities among cases. Classical factor analysis is then performed on the $N \times N$ correlation matrix. Data are assigned to clusters based on their factor loadings.

Table 2.10: Example of Statistics for Specifying SCs on Attributes Values

- **Sampling-based techniques** are based on the use of random samples as synopses for very large data sets. Random samples of a data collection typically provide accurate estimates for aggregate quantities (e.g., counts or averages) when they are difficult to compute accurately with limited memory.

- **Histogram-based techniques** have been studied extensively in the context of query selectivity estimation and, also, as a tool for providing approximate query answers. Certain classes of histograms can provide higher-quality approximate answers compared to random sampling, when considering low-dimensional data (with only one or two variables). It is a well-known fact, however, that histogram-based approaches become problematic when dealing with high-dimensional data sets. Simple statistics such as the mean and variance are both insufficiently descriptive and highly sensitive to data anomalies in real world data distributions. Most database management systems maintain order statistics, *i.e.*, quantiles on the contents of their database relations. Medians (half-way points) and quartiles (quarter-way points) are elementary order statistics. More generally, the ϕ-quantiles (for small real-valued $\phi > 0$) of an ordered sequence of N data items are the values with rank $k\phi N$, for $k = 1, 2, \ldots, \frac{1}{\phi - 1}$. Quantiles can summarize massive database relations more robustly than simple statistics or histogram-based approaches and they provide a quick similarity check in coarsely comparing relations, which may be very useful for data cleaning.

- **Wavelet-based techniques** provide a mathematical tool for the hierarchical decomposition of functions. Briefly, the idea is to apply wavelet decomposition to the input data collection to obtain a compact data synopsis that comprises a selected small collection of wavelet coefficients. Wavelets can be very effective in handling aggregates over high-dimensional data, while avoiding the high construction costs and storage overheads of histogram techniques. These techniques and variants have not been used in our study, but they constitute an interesting perspective for extending our library of analytical functions for QoD measurement.

- **String similarity sketches** provide quick way to compare textual fields and find textually similar attributes, for example with computing min hash samples on the q-grams of the attribute values, rather than the attributes values themselves. A min hash sample of an attribute returns a small number of hash values (signatures). The resemblance of two attributes can be computed from the comparison of the signatures of the two attributes, as the size of the intersection of the set of unique values of the attributes divided by the size of their union. Various techniques and string similarity distance functions may be used for computing set resemblance (see Table A.1 in Annexes, pages 150 and 151).

Table 2.11 presents examples of Level III functions we use for computing data synopses as a preliminary step to detect anomalies in very large data sets or approximates duplicates.

2.5.4 Level IV: QoD Mining Functions

Most existing work in the detection of outliers and deviation analysis lies in the field of statistics (e.g., mean and variance, upper or lower outliers in an ordered sample). Statistical measures such as average, standard deviation, and range to identify outliers are very often employed. However, the main problem with these approaches is that in a number of situations, the user simply does not have enough knowledge about the underlying data distribution. For this reason, nonparametric statistical methods and non-supervised techniques (clustering and association rule mining) have been privileged in our approach for characterizing such anomalies in data sets.

2.5.4.1 Classification

Classification is a grouping method based on the measure of resemblance or dissimilarity of the characteristics of the DB object instances. In general, a classification analysis can be carried out by first dividing the tuples into distinct groups based on their dependent attribute values. Then the dissimilarity (or resemblance) coefficient of a tuple with respect to each individual group is computed. Based on

Function	**Description**	**DB Objects**
Counting Sampling	Estimate of the frequency of a value in a sample. In simple random sampling, each element has an equal probability of being selected. Other probalities are used in stratified sampling, multistage sampling, systematic or clustering sampling.	D,T,A
Sample mean	Estimate of the mean from a sample $x_1, x_2 \ldots, x_N$ of N values randomly selected in of the attribute domain X in the interval $[\mu - t_\alpha \frac{\sigma}{\sqrt{N}}, \mu + t_\alpha \frac{\sigma}{\sqrt{N}}]$ with $\mu = \frac{1}{n}\sum_i x_i$ and sample standard deviation $\sigma = \sqrt{\frac{\sum_i (x_i - \mu)^2}{n-1}}$.	D,T,A
Min Hash Sampling	Method that randomly permutes row and applies a hash function $h(A)$ that indexes the first row with 1 for each attribute A with the following property: $Sim(A,B) = Pr[h(A) = h(B)]$. The similarity of two attributes A and B is based on the observations of min-hash signatures similarity. Let $Sim_H(sig(A), sig(B))$ be the fraction of permutations where Minhash signature values agree ($sig(A) = \min_{a \in A} h(a)$), we observe empirically that $Sim_H(sig(A), sig(B)) = Sim(A,B)$.	D,T
Equi-spaced Histograms	Approximations of the frequency distribution of the values of the attribute domain divided into intervals (or bins) of equal length.	A
Equi-depth Histograms	Approximations of the frequency distribution of the values of the attribute domain divided into buckets such that counts per bucket are equal.	A
String Similarity Sketches	Dimensionally reduced representation of strings with q-gram vectors used for computing q-gram similarity, L2 distance of the frequency distribution of the q-grams of an attribute. The q-gram vector $QV(A)$ of the attribute A is a normalized count of number of times a q-gram appears in the attribute, such as: $QV(A)[i] = \frac{Nboccur(q_i, Qgram(A))}{\sqrt{\sum_i Nboccur(q_i, Qgram(A))^2}}$ where q_i is the i^{th} q-gram, $Nboccur(x, X)$ the number of times the element x occur in X.	D,T,A
Set resemblance	Measure of how similar two attribute domains A and B are, as: $\rho = \frac{\|A \cap B\|}{\|A \cup B\|}$.	D,T,A

Table 2.11: Examples of Level III Synopses Techniques

the dissimilarity coefficients, the membership probability of a tuple belonging to a group can be estimated.

Consider $X_1, X_2, \cdots, X_m$ be a set of numerical explanatory attributes and $(x_{1i}, x_{2i}, \cdots, x_{mi})$ be the $X_1, X_2, \cdots, X_m$ values of a given tuple t_i.

Let $(\widehat{x_{1i}}, \widehat{x_{2i}}, \cdots, \widehat{x_{mi}})$ be the centroid of the k^{th} group, where $\widehat{x_j}$ are the mean X_j values of the k^{th} group ($j = 1, 2, \cdots, m$). The centroid vector indicates the average characteristics of the concerned group. The dissimilarity of a tuple t_i with respect to the k^{th} group is usually measured by the *squared-distance*, defined as (χ^2)-statistic:

$$D_{ik}^2 = (x_{1i} - \widehat{x_1}, \cdots, x_{mi} - \widehat{x_m}) C_k^{-1} (x_{1i} - \hat{x_1}, \cdots, x_{mi} - \widehat{x_m})^T$$

where C_k^{-1} is the inversed covariance matrix of the k^{th} group.

Both the centroid $(\widehat{x_{1i}}, \widehat{x_{2i}}, \cdots, \widehat{x_{mi}})$ and the covariance matrix C_k of a group can be obtained from (a sample of) the relation.

The larger the (χ^2)-statistic, the farther away, in the generalized distance sense, the point $(x_{1i}, x_{2i}, \cdots, x_{mi})$ is from the centroid $(\widehat{x_{1i}}, \widehat{x_{2i}}, \cdots, \widehat{x_{mi}})$ of the reference group. Thus, the tuple may be said to be more deviant from the average member of the group.

Conversely, a small (χ^2)-statistic indicates that the tuple resembles the group closely. With the computed (χ^2) value and the frequency of each group population, one can further calculate the membership probability of a tuple and provide metadata that characterizes certain aspects associated to the accuracy dimension (e.g., to find probable outliers) or to the uniqueness dimension (e.g., to determine probable duplicates).

Table 2.12 presents the description of the main classification techniques (with the underlined ones we used).

Function	**Description**	**Methods**	**DB Objects**
Hierarchical Agglomerative Methods (top-down)	1) For each observations pairs, compute the similarity measure, 2) Use the linkage rules (e.g., single linkage, complete linkage, and average linkage) to cluster the observation pairs that are the most similar, 3) Continue hierarchically comparing/clustering observations-to-observations, observations-to-clusters, clusters-to-cluster until no additional clustering is feasible via the clustering rules. Hierachical methods differ only in their definition of *between-cluster* similarity. They do not scale well: $O(n^2)$	AGNES (Kaufman & Rousseeuw, 1990), BIRCH (Zhang et al., 1996)	D,T,A
Hierarchical Divisive Methods	Initially all data are assigned to a single cluster. This cluster is divided into successively smaller chunks using either a monothetic or a polythetic strategy. Monothetic clusters are defined by certain variables on which certain scores are necessary for membership. Polythetic clusters are groups of entities in which subsets of the variables are sufficient for cluster membership.	DIANA (Kaufman & Rousseeuw, 1990)	D,T,A
Iterative Partitioning Methods	1) Begin with an initial partition of the data set into a specific number of nonempty clusters, 2) Compute seed points as the cluster centroids (for K-means) or medoids (representative objects for K-medoids) of the current partition, 3) Allocate each data point to the cluster with the nearest seed point (centroid or medoid), 4) Compute new centroids of the clusters; alternate steps 2 and 3 until no data points change clusters.	K-means, K-medoids-PAM, CLARA (Kaufman & Rousseeuw, 1990)	D,T,A
Density Search Methods	Algorithms search the feature space for natural modes in the data that represent volumes of high density. Strategies include a variant of single-linkage clustering and methods based on multivariate probability distributions.	Two-Way Joining	D,T,A

Table 2.12: Examples of Level IV Classification and Partitioning Methods

When a data set is partitioned into clusters, a data point can be then identified by its cluster membership and various characteristics that may define the cluster properties (e.g., cluster center, diameter, etc.). These properties are stored as descriptive metadata in the repository as they describe the parameters of the functions and the criteria for evaluating the classification result quality as presented in Table 2.13.

Outlier detection is an important issue in data clustering. Numerous work have proposed hierarchical, density-based, and distance-based clustering methods that gather data into clusters, detect and handle (or reject) outliers. However, most of them only consider numerical data. They regard the entire data record as a whole and they usually do not consider the deviation of one attribute value with respect to other attribute values. Many outlier detection algorithms have been proposed to find abnormal data, but they usually assume that the data is uniformly distributed which, in general, is not true. Clustering also requires a fair amount of domain expertise and experience with the algorithms and the tuning parameters,

Output Metadata	**Description**
R-Squared (RSQ or R^2)	Variance proportion that is explained by clusters (interclass inertia) $RSQ \rightarrow 1$ for good classification (for any type of methods).
Semi-partial R-Squared (SPRSQ)	Measure of the R-squared reduction when regrouping two clusters: a peak for k clusters and a hollow for k+1 clusters is expected for a good hierarchical classification for k+1 clusters.
Pseudo t^2 (PST2)	Measure of the separation between two clusters recently merged: a peak for k clusters and a hollow for k+1 clusters is expected for a good hierarchical classification of k+1 clusters
Cubic Clustering Criterion (CCC)	Indicator that represents the risk that the classification (Ward method or K-means) can be affected by outliers (if $CCC < 0$). CCC is not applicable to hierarchical clustering methods.
Pseudo F	Measure of the separation between k clusters over n observations $PseudoF = \frac{\frac{R^2}{k-1}}{\frac{1-R^2}{n-c}}$. Pseudo F is not applicable to hierarchical clustering methods.
Cluster Characteristics	For each cluster#, the cluster characteristics include intra-class inertia, the closest cluster#, the distance between the closest cluster, the radius as the max distance between the centroid of the cluster and the farest point, the coordinates of the centroid.
Data points Characteristics	For each data point, the distance to its centroid and the cluster#

Table 2.13: Descriptive Metadata for Partitioning Methods

and might prove to be computationally expensive.

2.5.4.2 Association Rule Mining

Association rule discovery may be used to specifically search and find inconsistencies and contradictions that occur frequently or that follow a certain pattern. In the particular context of QoD evaluation, we used association rule mining to discover four types of anomaly patterns: *i)* patterns of null values, *ii)* patterns of duplicates, *iii)* patterns of contradictions over overlapping data sets, and *iv)* patterns of dubious values. Dubious data refers to data values that appear legitimate on their own. But, upon examination together with other related attribute values, their values become questionable.

More formally, let $\mathcal{I}$ be a set of literals. An *association rule* R is an expression $LHS \rightarrow RHS$ with $(Conf, Supp)$, where $LHS, RHS \subseteq \mathcal{I}$ and $LHS \cap RHS = \emptyset$.

LHS and RHS are conjunctions of variables such as the extension of the Left-Hand Side LHS of the rule is: $g(LHS) = x_1 \wedge x_2 \wedge \ldots \wedge x_n$ and the extension of the Right-Hand Side RHS of the rule is $g(RHS) = y_1 \wedge y_2 \wedge \ldots \wedge y_{n'}$.

The support of a rule measures the occurrence frequency of the pattern in the data set (*i.e.*, how frequently items appear together) while the confidence is the measure of the strength of implication (*i.e.*, how frequently they conditionally appear). The problem of mining association rules is to generate all association rules that have support and confidence greater than user-specified minimum support and confidence thresholds. Other *interestingness measures* have been proposed in the literature for knowledge quality evaluation with the purpose of supplying alternative indicators to the user in the understanding and use of the discovered rules.

In our context, the semantics of the rules we are interested in can be defined as follows:

- *Patterns of null values*: a certain combination of attribute values may lead frequently to null values for a particular attribute. We are looking for interesting rules such as: $LHS \rightarrow RHS$ with $LHS = x_1 \wedge x_2 \wedge \ldots \wedge x_n$ and the extension of RHS only having $NULL$ values.

- *Duplicate detection patterns*: a certain combination of attribute values that are similar may frequently lead to the detection of duplicates, *i.e.*, we are looking for interesting rules such as: $Similar(LHS) \rightarrow [duplicate = true]$. $Similar(LHS)$ is the combination of attributes and range of values composing the extension of LHS whose values are frequently similar. The comparison distance between similar values is greater than a similarity threshold defined for each attribute domain. $duplicate$ is a Boolean function indicating that if the attributes of LHS belong to a set of matching criteria, then the records are considered to be duplicates.

  For example, $Similar(address) \rightarrow [duplicate = false]$ will not be among the interesting patterns for detecting duplicates whereas:

  $Similar(address) \wedge Similar(SSN) \wedge Similar(Phone) \rightarrow [duplicate = true]$ can be used to prevent duplicates by selecting the relevant attributes for preclustering their values. Moreover, the records that support the following rule:

  $Similar(address) \wedge Similar(SSN) \wedge Similar(Phone) \rightarrow \neg Similar(name)$ would be worth examining in order to prevent tuples that are similar on the matching criteria and should refer the same real world entity but they don't.

- *Patterns of contradictions for overlapping data sets*: we are looking for interesting rules such as: $matching(LHS) \rightarrow \neg Similar(RHS)$. $matching(LHS)$ corresponds to the attributes whose values are identical and known to refer the same real world object and they should have similar description values for RHS attributes but they don't. We are interested in LHS matching attributes that imply frequent conflicts (as non similar values) over RHS attributes values. If matching tuples for the best rules have frequent conflict on certain attribute values, then the tuples are considered contradictory. For example, when two overlapping databases DB1 and DB2 have to be merged at the instance level (suppose the mappings at the schema level have been resolved), it is interesting to determine the combinations of attributes that frequently generate conflicting values. For instance, consider the following rule:

  $matching(DB1.CUSTOMER.address(c1), DB2.CLIENT.address(c1)) \rightarrow \neg Similar(DB1.CUSTOMER.phone(c1), DB2.CLIENT.phone(c1))$

  This rule indicates that for the same real world entity (the same customer $c1$ identified in DB1 and DB2), the two sources agree on the address but do not on the phone numbers, showing the fact that they do not manage phone numbers the same way. Although both may have inaccurate or out-of-date phone numbers, one source may be more reliable that the other.

- *Patterns of dubious values*: we observe that dubious data values can be detected via data correlation (among other statistical techniques), which measures the relationship between associated variables. Such correlations can be also captured by using interval association rules. In our work, we use interval association rules to identify dubious data for both equality predicates ($Attr = val$) as well as range predicates ($Attr = [val_1, val_2]$). Data is first partitioned into meaningful clusters, *i.e.*, intervals for numeric attributes or fixed values for categorical attributes composing LHS, then interval association rules are generated such as:

  $Preclustered(LHS) \rightarrow NonExpectedValue(RHS)$. If the value of the tuples frequently deviates from the predicted value with respect to a particular function (e.g., linear regression), then data is considered as dubious data.

The aforementioned rules will be selected if their support and confidence values are greater than predefined thresholds, $(Conf, Supp) \geq (Min_{conf}, Min_{supp})$ and they will be stored together with the detailed description of their parameters as descriptive metadata in the repository (e.g., minimum support and confidence thresholds, numbers of rules, algorithm name, number of items, etc.).

2.5.5 Designing Analytic Workflows for QoD Evaluation

Once we have defined the panel of analytical functions that can be possibly used for QoD evaluation, an important step of our approach is to organize, plan and execute a relevant sequence of tasks depending on the specific goal driving the QoD measurement activities. For this purpose, we define and design analytic workflows.

Definition 2.5.1. Analytic workflow. *An analytic workflow can be described in terms of several 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 resource allocations (A),*
- *a set of limitations (L).*

An 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 data 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 data instance resource, e.g., a set of rows, *ii)* the data quality issue, e.g., accuracy, and *iii)* the purpose, e.g., detect anomalies.

The set of tasks (T) are the building blocks of a whole analytic workflow. The user defines the set of tasks for evaluating relevant aspects of data quality for a specific domain, application or goal. These tasks can be defined at different levels based on which the worklfow is to be modeled. 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 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, the set of analytical functions to be applied for QoD evaluation, and the set of output QoD metadata including QoD measures (outputs of analytical functions) and descriptive metadata (detailed description of the settings and parameters of the functions).

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.

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 analytic workflow model, the data and functions become inputs to the tasks and QoD metadata become output.

Depending on the layer of task representation, this weight can be used to represent different parameters. For example, if the task duration is the parameter that we are interested in, then we can use the weight of the edge, e, to represent the task duration of the vertex which the edge is incident to, i.e. vertex T_j .

Example 4. *For example, Figure 2.6 presents the analytic workflow designed for evaluating the quality of CRM_DB data. This includes several tasks of evaluation by user-defined measures on four QoD dimensions, namely freshness, accuracy, consistency and completeness at different granularity levels (e.g., cell, row, rowset, column, table, database). Data object instances are the inputs of tasks. Each task may be composed of several subtasks with allocated analytic functions for the computation of the QoD measures of our example. Summary statistics for numerical values (e.g., for PRICE of PRODUCT table) are computed, out-of-range data values may also be detected with univariate statistics and percentiles (IQR) (e.g., by function* `'plsql_func_iqr_ratio.sql'` *for the subtask* `t121` *named* `outlierRatio` *composing the task* `t12:AccuracyEval` *that evaluates the accuracy at the TABLE granularity level of CRM_Db. The percentage of null values for each table are computed by function* `'plsql_func_nullValues%'` *for the parent task* `t14:CompletenessEval`*. The execution of each allocated function generates QoD mea-*

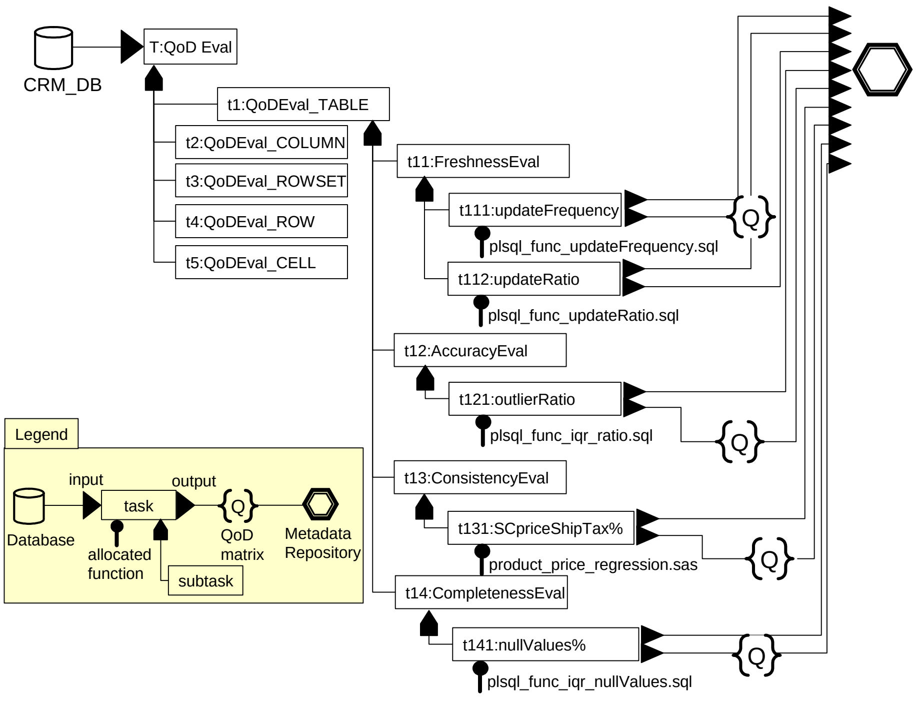


Figure 2.6: Example of Analytic Workflow for QoD evaluation of CRM_DB

sures whose values compose QoD matrices (represented in the figure as {Q}) as descriptive metadata.

The detailed description of each task involving at least one analytical function is stored as descriptive metadata in the repository. This detailed description of analytical functions can be specified with PMML, the Predictive Model Markup Language[3] as illustrated in Table A.2 in Annexes (page 155).

PMML is an XML standard for specifying the parameters and preliminary transformations of data mining models. PMML and CWM *Data Mining* metamodel are complementary in that PMML specifies the detailed content of mining models, whereas CWM *Data Mining* metamodel specifies data mining process metadata such as settings and scoring output.

Analytic workflows combine multiple analytical functions as the ones presented in the previous section (from Level I to Level IV). As illustrated in Tables A.3 and A.4 in Annexes (pages 156-157) by two examples for detecting outliers or data that significantly deviate from statistical rules, functions may be implemented in different ways depending on the set of limitations specified in the workflow. Of course, many other functions can be added to the library.

[3]PMML Version 3.1: http://www.dmg.org/v3-0/GeneralStructure.html

2.5.6 Computing and Assigning Probabilities to QoD Dimensions

The panel of analytical functions used for computing QoD measures and generating QoD metadata gives relevant indications for characterizing potential data quality problems and understand anomaly patterns. Some are preliminary inputs to other tasks in the analytic workflow.

Considering the set of QoD metadata generated by analytical functions associated to the evaluation of one QoD dimension over DB object instances, a probability that a data quality problem exists can be assigned to this QoD dimension. This probability indicates the degree to which the considered QoD dimension is acceptable with respect to the set of constraints (and expected values) defined on its associated QoD measures computed by analytical functions at a given time for a particular database object instance.

Consider a collection of DB object instances $\mathscr{O}$, a set of metadata $\mathscr{M}$, and a set of analytical functions $\mathscr{F}_i = \{f_{i1}, \cdots, f_{ip}\}$ associated to the QoD dimension noted Q_i. Each function computes from a DB object instance measures that are stored in the repository and associated to the QoD dimension Q_i.

Definition 2.5.2. Quality Matrix. *Given o, a DB object instance in $\mathscr{O}$, m_{ij}, a QoD measure computed from o by the function $f_{ij}(o)$ associated to QoD dimension Q_i. Considering k QoD dimensions and p analytical functions, the complete quality evaluation of the object o is represented as a $k \times p$ matrix, as: $QoD(o) = (m_{ij}), m_{ij} \in \mathscr{M}$.*

$$QoD(o) = \begin{pmatrix} m_{11} & \cdots & m_{1p} \\ \vdots & \ddots & \vdots \\ m_{k1} & \cdots & m_{kp} \end{pmatrix} \tag{2.1}$$

Example 5. *Consider the PRODUCT table and a set of QoD measures that are computed for characterizing freshness (F), accuracy (A), completeness (CP), and consistency (CT) dimensions. An example of quality matrix computed from PRODUCT table is:*

$$QoD(PRODUCT) =$$

$$\begin{pmatrix} F.updateFrequency & F.updateRatio & - \\ A.outlierRatio & - & - \\ CP.nullValues\% & - & - \\ ICorderShipDate\% & ICshipDateStatus\% & SCpriceShipTax\% \end{pmatrix}$$

In our approach, we adopt a probabilistic semantics for exploiting QoD measures obtained from descriptive and predictive analytical functions. There are of course different ways of assigning such probabilities.

The way we adopted is based on *conditional probabilistic and approximate constraints* to compute the **acceptability** value of a QoD dimension evaluated from a Db object instance.

Definition 2.5.3. QoD dimension Acceptability of a DB object instance *indicates the likelihood δ_i that the QoD dimension Q_i is acceptable for the database object instance o given the constraints on its associated QoD measures m_{ij} being satisfied with respect to the expected values M_{ij} within a tolerance ϵ_{ij}. Acceptability of QoD dimension Q_i on the DB object instance o, noted $Pr_i(o)$ is defined as:*

$$Pr_i(o) = Pr(Q_i(o) | \bigwedge_j m_{ij} \in [M_{ij} \pm \epsilon_{ij}]) = \delta_i. \tag{2.2}$$

δ_i is a scoring method that combines all the measures obtained from the analytical functions that characterize the considered QoD dimension Q_i for the database object instance. δ_i may be defined as the weighted sum of distances between actual and expected measure values associated to QoD dimension Q_i, as:

$$\delta_i = \sum_j w_{ij}.d(m_{ij}, M_{ij}).$$

$d(m_{ij}, M_{ij})$ also denoted d_{ij} is null when the QoD measure m_{ij} exactly satisfies the constraint. If the constraint is not satisfied even with the approximation tolerance noted ϵ_{ij}, then d_{ij} equals 1; otherwise d_{ij} is computed as the normalized distance between the expected value and the actual QoD measure value for the object instance o.

Definition 2.5.4. Quality Acceptability of a DB object instance. *Quality acceptability of a Db object instance is defined as the vector of probabilities over the k QoD dimensions as:*

$$Acceptability(o) = \begin{pmatrix} Pr_1(o) \\ \cdots \\ Pr_k(o) \end{pmatrix} \tag{2.3}$$

Table 2.14 gives a generic procedure to compute and assign probabilities to database object instances.

```
Algorithm AssignQoDProbabilities
Input:
        a collection of database object $\mathscr{O}$
        a set of quality dimension $\mathscr{Q} = \{Q_1, Q_2, \dots, Q_n\}$
        a set of metadata $\mathscr{M}$
        a set of functions associated to quality dimension $Q_i$
          applied to a database object instance $o$, $F_i(o) = \{f_{i1}(o), \dots, f_{ip}(o)\}$
        $M_{ij}$ the expected value and $\epsilon_{ij}$ the approximation for metadata $m_{ij}$ in $\mathscr{M}$
        a set of approximate constraints on metadata for quality dimension $Q_i$
          $C_i = \{C_{ij} | m_{ij} \in [M_{ij} \pm \epsilon_{ij}], \forall m_{ij} \in \mathscr{M}\}$
        a normalized weight $w_{ij}$ assigned to constraint $C_{ij}$, as $\sum_j w_{ij} = 1$

Output:
        For every database object $o$ in $\mathscr{O}$,
        a conditional probability $P_i(o)$ is associated to quality dimension $Q_i$,
        $P_i(o) = prob(Q_i(o) | m_{ij} \in [M_{ij} \pm \epsilon_{ij}])$

Main Procedure:
        For each $o \in \mathscr{O}$,
          For each quality dimension $i = 1 \cdots n$:
            For each analytical function $j = 1 \cdots p$:
              Execute $f_{ij}(o)$: $f_{ij}(o) \rightsquigarrow m_{ij}$
              /* each function execution leads to the storage of QoD measure $m_{ij}$ in the repository */
              For each $m_{ij}$:
                Check the constraints $C_{ij}$:
                 if $C_{ij}$ is true, then:
                    if $\epsilon_{ij} = 0$, then: $d_{ij} = 0$
                         else $d_{ij} = Normdist(m_{ij}, M_{ij})$
                 else $d_{ij} = 1$
                Compute $Pr_i(o) = \sum_j w_{ij} \cdot d_{ij}$
```

Table 2.14: Assigning Probabilities to QoD Dimension for a DB Object Instance

This is the evaluation process we applied to each QoD dimension in order to determine the probability that the QoD dimension is acceptable with respect to a set of constraints with expected values and a tolerance interval. This evaluation process is triggered by the analytic workflow (represented as **{Q}** in Figure 2.6) and it is also one of its main output result.

2.6 Indexing Quality Metadata

2.6.1 Requirements

For quality-aware query processing, we use the existing index structure for data but we need to index associated QoD metadata and process range or similarity search on QoD metadata to find the quality measures that best match quality requirements (usually expressed as one-sided range constraints) and get the corresponding DB object instances. The problem addressed in this section is how to efficiently access data and metadata to efficiently support quality-aware query processing. Since we do not want to limit ourselves to a particular domain, the solution we seek for metadata indexing should satisfy the following requirements:

- **Adaptability**: Current DBMSs do not allow specific indexes to be "plugged-in". For classical data, the B+tree has been widely accepted as the standard index structure offered by RDBMSs. Our objective is not to change the current data index of existing databases but rather index generated QoD metadata with the appropriate index structure because quality requirements are expressed as similarity search or range queries over associated QoD measures. Such queries can be better performed with multidimensional index structures (e.g., R-tree). Nevertheless, when we access data, we need to access associated QoD metadata simultaneously.

- **Flexibility**: We would also retain the possibility to submit queries over only a subset of QoD metadata at different granularity levels of the database, as well as to vary at query time the relevance of some QoD criteria (*i.e.*, weights or different constraints on QoD metadata from one query to another) for producing on-demand "customizable quality"-aware results.

- **Extensibility**: The solution should be able to deal with different metric spaces, because QoD measures have to be compared with expected values of the user-defined constraints using various distance functions and can not be limited to vector spaces. The solution should be able to index QoD measures by using many distance functions.

Fulfilling all these requirements constitute a challenging research direction in the area of multidimensional indexing and meta-indexing. In this dissertation, we only present the operational solution we've adopted so far for indexing QoD metadata.

2.6.2 Range-Encoded Bitmap Index for QoD measures

For data sets with a moderate number of dimensions, a common way to the answer multidimensional and *ad-hoc range* queries is to use one of the multidimensional indexing methods (e.g., R-Trees or kd-trees). These approaches have two notable shortcomings. Firstly, they are effective only for data sets with modest number of dimensions (< 7). Secondly, they are only efficient for queries involving all indexed attributes. However, in many applications only some of the attributes are used in the queries. In these cases, the conventional indexing methods are often not efficient.

The compressed bitmap index can be very efficient in answering one-dimensional range queries (Stockinger, 2002; Wu et al., 2006). Since answers to one-dimensional range queries can be efficiently combined to answer arbitrary multidimensional range queries, compressed and bitmap indices are known to be efficient for any range query. There are a number of indexing methods, including B*-tree and B+-tree that are theoretically optimal for one-dimensional range queries, but most of them cannot be used to efficiently answer arbitrary multidimensional range queries.

In our context of QoD measure indexing, we use **range-encoded bitmap index with binning strategy** to reduce the number of bitmaps. Range encoding requires at most one bitmap scan for evaluating range queries.

The basic idea of binning is to build a bitmap for a bin rather than each distinct attribute value. This strategy disassociates the number of bitmaps from the attribute cardinality and allows one to build a bitmap index of a prescribed size, no matter how large the attribute cardinality is. A clear advantage of this approach is that it allows one to control the index size.

However, it also introduces some uncertainty in the answers if one only uses the index. To generate precise answers, one may need to examine the original data records (*i.e.*, the QoD measure value) to verify that the user's specified conditions defined in the sided constraint are satisfied (candidate check).

As illustrated in Table 2.15, the values of the QoD measure m_{ij} are given in the second leftmost column. The identifier of the DB object instance which the m_{ij} measure value is associated to is given in the fist column. The range of possible values of m_{ij} is partitioned into five bins $[0, 20), [20, 40)$, etc. A "1-bit" indicates that the measure value falls into a specific bin. On the contrary, a "0-bit" indicates that the measure value does not fall into the specific bin.

OID#	m_{ij}	0	1	2	3	4	← bitmap identifier
	values	[0;20)	[20;40)	[40;60)	[60,80)	[80;100)	← bin ranges
1	32.4	0	1	0	0	0	
2	82.7	0	0	0	0	1	
3	79.2	0	0	0	1	0	
4	65.3	0	0	0	1	0	
5	55.6	0	0	1	0	0	

Table 2.15: Range-Encoded Bitmap Index with Binning for QoD Measures

Take the example of evaluating the following query: "Give the most accurate PRODUCT price value". The query is equivalent to the selection the DB object instance values of PRICE attribute in PRODUCT table from our CRM_DB example and search for accuracy measures m_{ij} in the interval $[80, 100)$. The correct result should be the OID# 2.

For the query: "Give PRODUCT price values with accuracy measure greater than 70%". The query searches in the interval $[60, 80)$ and $[80, 100)$. We see that the range in the query overlaps with bins 3 and 4. We know for sure that all rows that fall into bin 4 definitely qualify (*i.e.*, they are hits). On the other hand, rows that fall into bin 3 possibly qualify and need further verification. In this case, edge bins are 3 and 4. The DB object instances that fall into edge bins are candidates and need to be checked against the constraint on accuracy measure.

In this example, there are two candidates, namely OID# 2 from bin 4 and OID# 3 from bin 3. The candidate check process needs to read these rows from disk and examine their values to see whether or not they satisfy the user-specified conditions.

2.7 Extending the Syntax of a Query Language

2.7.1 Declaration of Quality Requirements

In (Berti-Équille, 2003; 2004), we have proposed a first version of *XQuaL*, a generic query language extension to SQL for describing and manipulating in a flexible way data and associated QoD metadata. Our approach consists of including data quality requirements that extend the traditional select-from-where query (SFW-query) with constraints on QoD measures.

A quality-extended query, called **QWITH query** is a SFW-query followed by a `QWITH` operator used to declare data quality constraints required by the user. Data quality constraints are expressed by means of *quality contract types* and *quality contracts*:

- A **quality contract type** defines a set of quality dimensions, measures and functions associated to a particular database object instance or a granularity level of the database. A quality contract type is instantiated once the QoD measures are computed by analytical functions call or declaration; the results are then stored as QoD measures and assigned to the database object instance or granularity level (*i.e.*, value, record, attribute, table, database).

- A **quality contract** is a set of constraints (usually one-sided range constraints) defined on the dimensions declared in the related contract type.

2.7.1.1 Declaration of Quality Contract Type

The syntax of creation of a quality contract type is given in Figure 2.7. A contract type is assigned to the schema of the database and it is composed of a set of named dimensions with the type declaration of the output QoD measure computed by a named analytical function that is declared or invoked.

Measures are stored in the QoD metadata repository and they are assigned to:

- *a global granularity level*, *i.e.*, the considered database, table, column, record or value, respectively expressed by ON CELL, COLUMN, TABLE, ROW, DATABASE statement,

- *a specific DB object instance*, *i.e.*, an existing table, column, record (ROWID), set of records (ROWSET) or cell (COLUMN.ROWID) expressed by ON <table_name>.[<column_name>].[<rowid>] statement.

For each row in the database, the ROWID pseudocolumn returns a row's address. ROWID values contain information necessary to locate a row, *i.e.*, the data object instance number, the data block in the datafile, the row in the data block (first row is 0), and the datafile (first file is 1). The file number is relative to the tablespace. A ROWID value uniquely identifies a row in the database.

The function may be a PL / SQL procedure or a call specification of a Java, C or SAS program (se Figure 2.8):

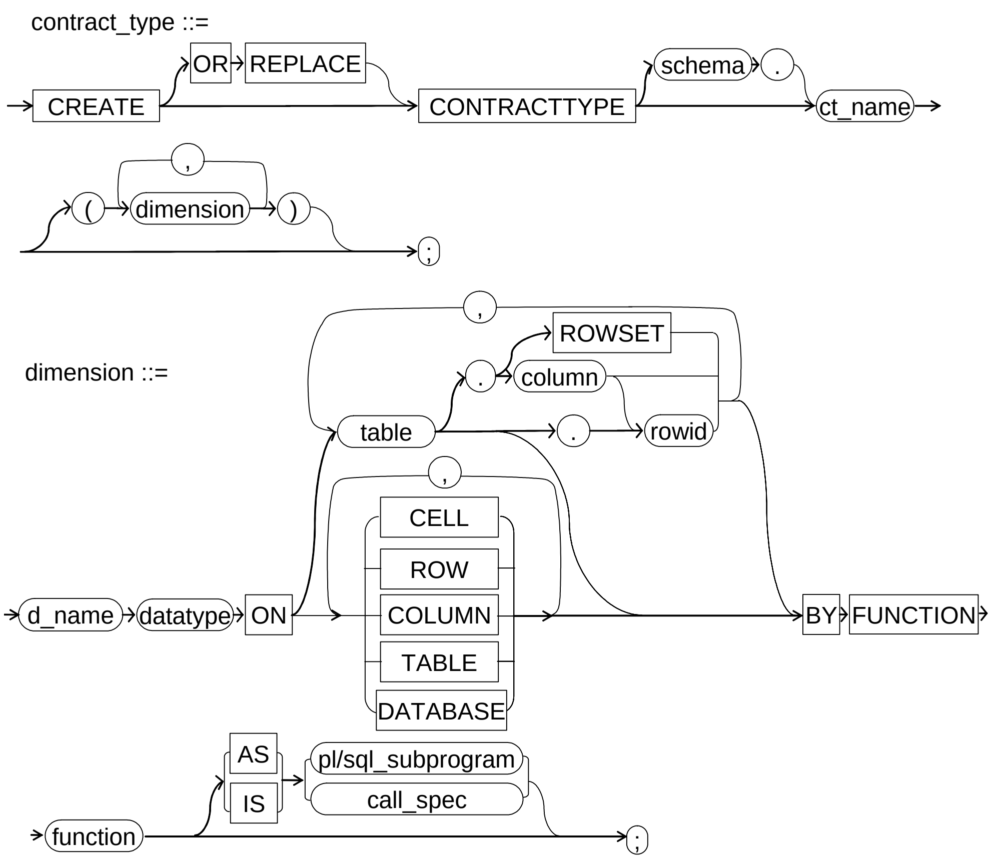


Figure 2.7: Syntax of Quality Contract Type Creation

- 'pl/sql_subprogram' is the declaration of the procedure in a PL/SQL subprogram body.

- 'call_spec' is used to map a Java, SAS or C method name, parameter types, and return type to their SQL counterparts.

- In 'Java_declaration' or 'SAS_declaration', 'string' identifies the name of the implemented method.

The creation of a contract type associated to a current database object instance implies the execution of the declared analytical functions, *i.e.*, computation and storage of the resulting QoD measures and descriptive metadata in the repository.

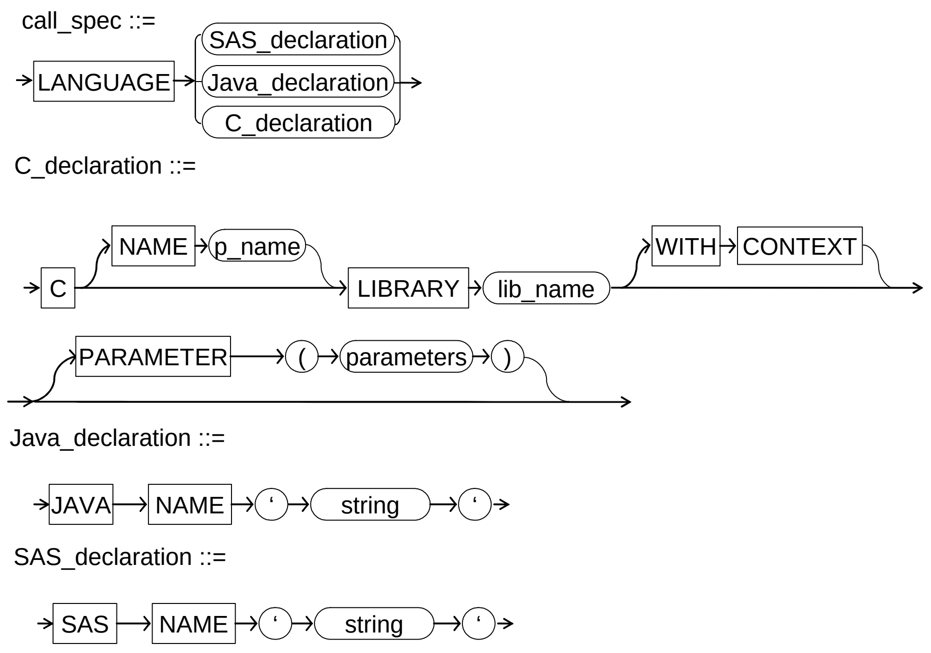


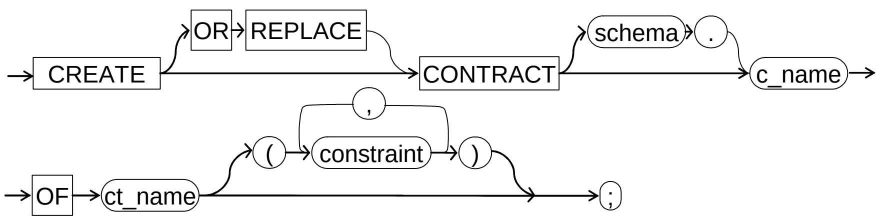
Figure 2.8: Syntax of Call Specification in Quality Contract Type Declaration

2.7.1.2 Declaration of Quality Contract

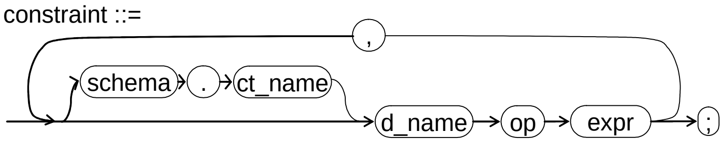
The syntax of creation of a quality contract is given in Figure 2.9. Each contract declaration refers to an existing contract type. It defines the constraints on each contract type dimension with simple or composed expressions using basic binary operators.

The creation of contract instances assigned to selected granularity levels of the database is a necessary step before submitting a quality-extended query.

contract ::=



constraint ::=



op ::= (= | != | <> | > | < | <= | >=)
expr ::= (simple_expr | compound_expr)
simple_expr ::= (NULL | string | number | true | false)
compound_expr ::=

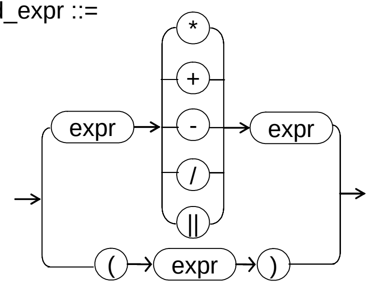


Figure 2.9: Syntax of Quality Contract Creation

Example 6. *Consider again the CRM_DB example. Using XQuaL, we declare four quality contract types for characterizing freshness, accuracy, completeness, and consistency on the instances of CRM_DB database. Table 2.16 presents the creation script of these quality contract types.*

Each contract type is composed of a list of named measurable dimensions (e.g., `age` *for the contract type named* `FRESHNESS`*), the type of the output result* `FLOAT`*, the granularity level which the measure is associated to, noted* `ON CELL, COLUMN, ROW, TABLE`*, and the identifier and name of the function that computes the measure following the* `BY FUNCTION` *statement.*

The contract type `FRESHNESS` has five dimensions, named `age`, `timeliness`, `lastUpdateTime`, `updateFrequency`, and `updateRatio`. These functions have

```
CREATE CONTRACTTYPE FRESHNESS(
   age FLOAT ON CELL BY FUNCTION func_age IS LANGUAGE JAVA
                            NAME './XQLib/func_age.java',
   timeliness FLOAT ON CELL,ROW BY FUNCTION plsql_func_timeliness,
   lastUpdateTime date ON ROW BY FUNCTION plsql_func_lastUpdateTime,
   updateFrequency FLOAT ON TABLE BY FUNCTION plsql_func_updateFrequency,
   updateRatio FLOAT ON TABLE BY FUNCTION plsql_func_updateRatio);

CREATE CONTRACTTYPE ACCURACY(
   outlierStatus boolean ON CELL BY FUNCTION sas_outlier_check IS LANGUAGE SAS
                                    NAME './XQLib/sas_outlier_check.sas',
   outlierRatio FLOAT ON ROW, COLUMN, TABLE BY FUNCTION plsql_func_iqr_ratio,
   approximateDuplicate FLOAT on CUSTOMER.ROWSET BY FUNCTION plsql_func_sim_sketch);

CREATE CONTRACTTYPE COMPLETENESS(
   nullValues% FLOAT ON ROW, COLUMN, TABLE, DATABASE
                            BY FUNCTION plsql_func_nullValues%);

CREATE CONTRACTTYPE CONSISTENCY(
   ICshipDateStatus boolean ON PRODUCT.ROWSET BY FUNCTION plsql_IC1,
   ICorderShipDate% FLOAT ON PRODUCT BY FUNCTION plsql_IC2,
   ICshipDateStatus% FLOAT ON PRODUC BY FUNCTION plsql_IC3,
   SCpriceShipTax% FLOAT ON PRODUCT BY FUNCTION regression IS LANGUAGE SAS
                            NAME './XQLib/product_price_regression.sas');
```

Table 2.16: Examples of Quality Contract Type Declaration

been previously defined in Table 2.8. For instance, the type of the measure computed for the dimension `updateFrequency` is `real` and it is computed by the PL/SQL function identified by `plsql_func_updateFrequency` applied on each table `ON TABLE` of the CRM_DB database. `age` of data values is computed by a Java program named '`func_age.java`'.

In `ACCURACY` contract type, `outlierStatus` is a Boolean value returned by the SAS function named '`sas_outlier_check.sas`' derived from the one given in Table A.4 in Annexes (page 157). `outlierStatus` is true if the value of the cell belongs to the set of outliers. `outlierRatio` computed by the function `plsql_func_iqr_ratio` counting the fraction of the number of outliers over the total number of values for a given (set of) attribute domain(s). This function is derived from `sql_func_iqr` given in Table A.3 in Annexes (page 156).

In `COMPLETENESS` contract type, the PL/SQL procedure `plsql_func_nullValues%` returns a percentage of missing values for each row, column, table, and for the whole database.

In `CONSISTENCY` contract type, `ICshipDateStatus` is a Boolean value returned by the PL/SQLfunction '`plsql_IC1`'. The constraint returns true if the status is 'CLOSED' when the shipping date is passed or 'ACTIVE' if the shipping date is planned after the current date for each row of PRODUCT table. Two integrity and one statistical constraints are also declared on PRODUCT table. They are implemented with PL/SQL procedures assigned to PRODUCT table and return the percentage of rows that do not satisfy the constraint: for example, for `ICorderShipDate%`, the constraint checks if the date of product order is before the date of product shipping; for `SCpriceShipTax%`, a statistical constraint is based on the result of the logistic regression given in Table A.2 in Annexes (page 155). It returns percentage of rows whose values significantly deviate from the computed estimators.

Example 7. *Based on the contract type structure, the user can instantiate his data quality requirements and define constraints. Table 2.17 gives user-defined examples of quality contracts, respectively named `fresh`, `accurate`, `complete`, and `consistent` whose definition is based on their respective contract types: `Freshness`, `Accuracy`, and `Completeness`. Contract instances describe the constraints that have to be checked on the dimensions declared in the corresponding quality contract type. `age` of data values defined in `fresh` contract, instance of `FRESHNESS` contract type has to be lower than .42 hours.*

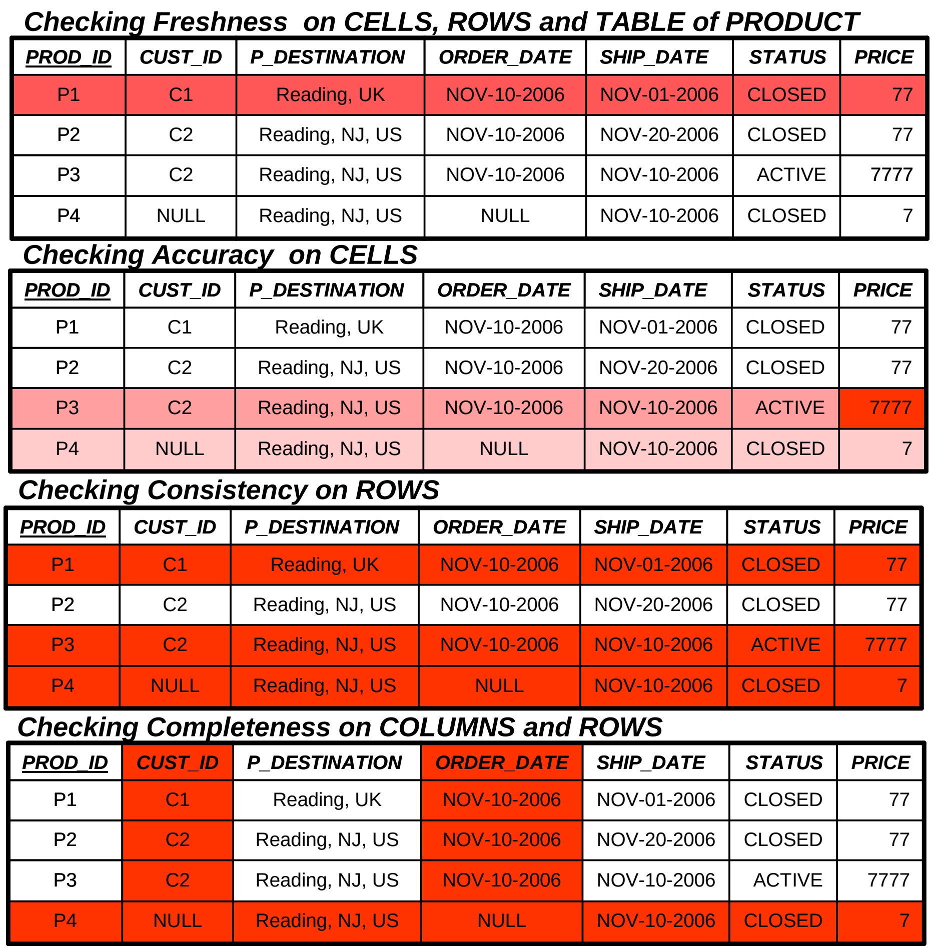
```
CREATE CONTRACT fresh OF FRESHNESS(
   age < .42,
   timeliness > .50,
   lastUpdateTime > '2007-JAN-01',
   updateFrequency >= 5,
   updateRatio > .50);

CREATE CONTRACT accurate OF ACCURACY(
   outlierStatus = false,
   outlierRatio < .05,
   approximateDupplicate <.50);

CREATE CONTRACT complete OF COMPLETENESS(
   nullValues% <= .80);

CREATE CONTRACT consistent OF CONSISTENCY(
   ICshipDateStatus=true,
   ICorderShipDate% <.05,
   ICshipDateStatus% <.05,
   SCpriceShipTax%< .05);
```

Table 2.17: Example of Contract Declaration

Declaring quality contract types, our objective is to incorporate a set of analytical functions for precomputing 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. Constraints are checked on the defined granularity levels or particular database object instances immediately after the validation of the contract declaration. The result of constraints checking indicating that the constraints are satisfied is stored as QoD temporary matrices in the metadata repository.

The corresponding values of the QoD matrix of PRODUCT table is given as example in Example 5.

Example 8. *Suppose the quality contracts given in Table 2.17 are applied to CRM_DB. For each CRM_DB object instance, the constraints declared in the quality contracts are checked. For each granularity level, Figure 2.10 illustrates in red color the DB object instances that do not satisfy exactly the constraints expressed in the previous contracts (i.e., whose acceptability probabilities are in]0,1]).*



Checking Freshness on CELLS, ROWS and TABLE of PRODUCT

PROD_ID	CUST_ID	P_DESTINATION	ORDER_DATE	SHIP_DATE	STATUS	PRICE
P1	C1	Reading, UK	NOV-10-2006	NOV-01-2006	CLOSED	77
P2	C2	Reading, NJ, US	NOV-10-2006	NOV-20-2006	CLOSED	77
P3	C2	Reading, NJ, US	NOV-10-2006	NOV-10-2006	ACTIVE	7777
P4	NULL	Reading, NJ, US	NULL	NOV-10-2006	CLOSED	7

Checking Accuracy on CELLS

PROD_ID	CUST_ID	P_DESTINATION	ORDER_DATE	SHIP_DATE	STATUS	PRICE
P1	C1	Reading, UK	NOV-10-2006	NOV-01-2006	CLOSED	77
P2	C2	Reading, NJ, US	NOV-10-2006	NOV-20-2006	CLOSED	77
P3	C2	Reading, NJ, US	NOV-10-2006	NOV-10-2006	ACTIVE	7777
P4	NULL	Reading, NJ, US	NULL	NOV-10-2006	CLOSED	7

Checking Consistency on ROWS

PROD_ID	CUST_ID	P_DESTINATION	ORDER_DATE	SHIP_DATE	STATUS	PRICE
P1	C1	Reading, UK	NOV-10-2006	NOV-01-2006	CLOSED	77
P2	C2	Reading, NJ, US	NOV-10-2006	NOV-20-2006	CLOSED	77
P3	C2	Reading, NJ, US	NOV-10-2006	NOV-10-2006	ACTIVE	7777
P4	NULL	Reading, NJ, US	NULL	NOV-10-2006	CLOSED	7

Checking Completeness on COLUMNS and ROWS

PROD_ID	CUST_ID	P_DESTINATION	ORDER_DATE	SHIP_DATE	STATUS	PRICE
P1	C1	Reading, UK	NOV-10-2006	NOV-01-2006	CLOSED	77
P2	C2	Reading, NJ, US	NOV-10-2006	NOV-20-2006	CLOSED	77
P3	C2	Reading, NJ, US	NOV-10-2006	NOV-10-2006	ACTIVE	7777
P4	NULL	Reading, NJ, US	NULL	NOV-10-2006	CLOSED	7

Figure 2.10: Checking QoD Constraints on CRM_DB Granularity Levels

At this stage, when the set of constraints C declared in the contracts are checked on the DB object instance o, a temporary matrix $temp$ is generated from the quality matrix $QoD(o)$ associated to o. Each element of the temporary matrix indicates the degree of satisfaction of the related constraint: δ_{ij} in [0,1].

$$\begin{aligned} temp(o) &= QoD(o).C(o) \\ &= \begin{pmatrix} m_{11} & \cdots & m_{1p} \\ \vdots & \ddots & \vdots \\ m_{k1} & \cdots & m_{kp} \end{pmatrix} \cdot \begin{pmatrix} c_{11} & \cdots & c_{1p} \\ \vdots & \ddots & \vdots \\ c_{k1} & \cdots & c_{kp} \end{pmatrix} = \begin{pmatrix} \delta_{11} & \cdots & \delta_{1p} \\ \vdots & \ddots & \vdots \\ \delta_{k1} & \cdots & \delta_{kp} \end{pmatrix} \end{aligned}$$

with

$$\delta_{ij} = \begin{cases} 1 & \text{if } c_{ij}\text{=false} \\ 0 & \text{if } c_{ij}\text{=true and } \epsilon_{ij} = 0 \\ Normdist(m_{ij}, M_{ij}) & \text{if } c_{ij}\text{=true for non null } \epsilon_{ij} \end{cases}$$

As we previously mentioned, a probability can be assigned for the object o with respect to the particular QoD dimension declared in the contrat. This probability, we called *quality acceptability* of DB object o is computed based on a scoring function (e.g., weighted sum) to represent the relative importance of the QoD measures associated to the QoD dimension. Although, in line of principle, any kind of scoring function would do the job, in this work we only consider monotonic scoring functions.

$$Acceptability(o) = temp(o) \cdot W = \begin{pmatrix} \delta_{11} & \cdots & \delta_{1p} \\ \vdots & \ddots & \vdots \\ \delta_{k1} & \cdots & \delta_{kp} \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ \vdots \\ w_p \end{pmatrix} = \begin{pmatrix} Pr_1(o) \\ \cdots \\ Pr_k(o) \end{pmatrix}$$

Example 9. *To complete Figure 2.10 that illustrated in red color the DB object instances that do not satisfy the constraints expressed in the declared contracts. Table 2.18 gives the computed probabilities of some DB object instances of CRM_DB (illutrated with various nuances of red). Again, null probability means acceptable QoD dimension; 1 means unacceptable. We suppose that the probabilities of the other CRM_DB object instances are null.*

	PRODUCT Table				**CUSTOMER Table**	
Acceptability	**CELL('7777')**	**ROW('P1')**	**ROW('P3')**	**ROW('P4')**	**CELL('C2')**	**ROW('C2')**
Pr_{fresh}	0	.4	0	0	0	0
$Pr_{accurate}$	1	0	.143	.06	.6	.5
$Pr_{complete}$	-	0	0	.286	-	0
$Pr_{consistent}$	-	1	1	1	-	0

Table 2.18: Examples of Acceptability Values per QoD Dimension

2.7.2 Manipulation of Data and Quality Metadata

Once declared, one (or several) contract(s) or constraints may be used in the `QWITH` part of the *XQUAL* queries and applied to cells, rows, attributes, tables or the database declared in the query. The syntax of a quality-extended query is given in Figure 2.11.

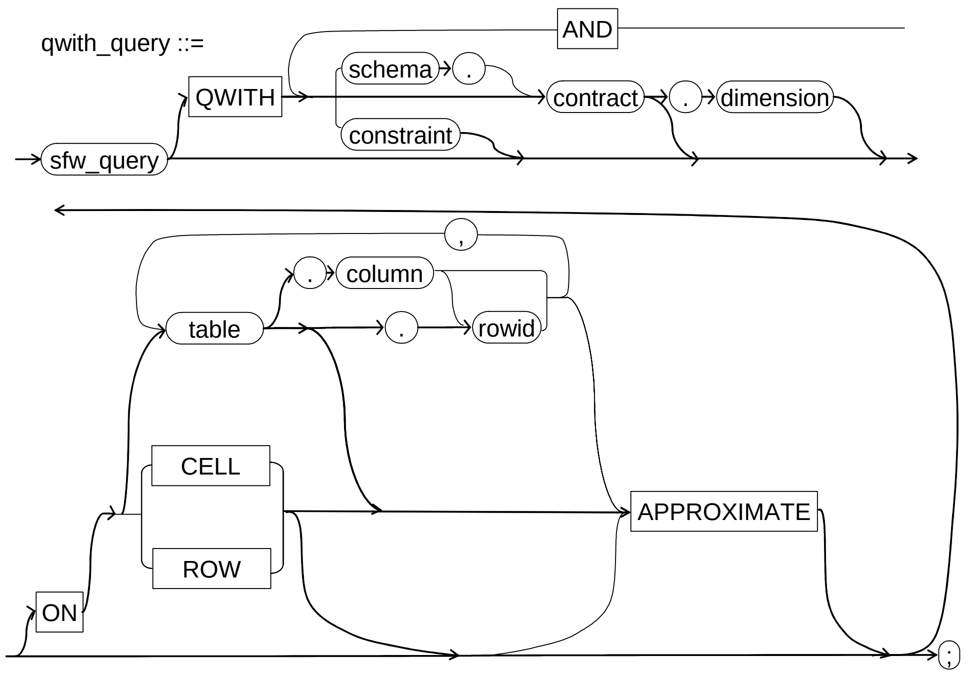


Figure 2.11: Syntax of QWITH queries

'`sfw_query`' represents the classical SELECT-FROM-WHERE statement of the query. In the QWITH statement, pre-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.

Two alternatives are possible for constraints checking in the QWITH declaration:

- *Exact* (default). only DB object instances that exactly satisfy the constraints declared in the contracts are used in the query processing.

- *Approximate.* DB object instances that satisfy the constraints within a non null approximation range of the expected values are used in the query processing.

Example 10. *Consider the query that retrieves all the products whose PRICE is greater than $10. QoD metadata of PRODUCT table involved in this query have been defined by means by the quality contract types and instances previously declared in Tables 2.16 and 2.17.*

But different quality-aware queries as illustrated in Table 2.19 may be defined based on the declared quality contracts checking freshness, accuracy, completeness, and consistency requirements at different granularity levels of CRM_DB. In Table 2.19, the third column indicates the result obtained with EXACT constraint checking together with the acceptability values of the result for the QoD dimensions considered in the QWITH part *of the query. Of course, these queries lead to different results depending on the constraints imposed on the QoD dimensions evaluated from database object instances.*

Query#	**Query**	**Results**	

Q1	SELECT PROD_ID, PRICE
	FROM PRODUCT
	WHERE PRICE > 10
	QWITH fresh;
		P2,77	$Pr_{fresh}(CELL('77'))$:0

		P3,7777	$Pr_{fresh}(CELL('7777'))$:0
			$Pr_{fresh}(ROW('P2'))$:0
			$Pr_{fresh}(ROW('P3'))$:0
			$Pr_{fresh}(TABLE('PRODUCT'))$:.5

Q2	SELECT PROD_ID, PRICE
	FROM PRODUCT
		P1,77	$Pr_{accurate}(CELL('P1'))$:0
			$Pr_{accurate}(CELL('77'))$:0

	FROM PRODUCT
	WHERE PRICE > 10
	QWITH accurate ON CELL;
		P2,77	$Pr_{accurate}(CELL('P2'))$:0
			$Pr_{accurate}(CELL('77'))$:0

Q3	SELECT PROD_ID, PRICE	P2,77	$P_{fresh}(ROW('P2')$:0

	FROM PRODUCT
	WHERE PRICE > 10
	QWITH fresh ON ROW;
		P3,7777	$P_{fresh}(ROW('P3'))$:0

Q4	SELECT PROD_ID, PRICE
	FROM PRODUCT
	WHERE PRICE > 10
	QWITH fresh ON CELL
	AND consistent ON ROW;
		P2,77	$P_{fresh}(CELL('77'))$:0
			$P_{fresh}(CELL('P2'))$:0
			$P_{consistent}(ROW('P2'))$:0

Table 2.19: Examples of Simple QWITH Queries with EXACT Constraint Checking Mode

2.7.2.1 Exact Constraint Checking

In the EXACT mode for checking constraints, 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, respectively:

- For Q1 query, the complete contract `fresh` is defined on three granularity levels (CELL, ROW, TABLE) as given in Table 2.16. The constraints expressed in Table 2.17 are checked for each cell and each row involved in the SPJ query processing. Probabilities resulting from the checking are given in the last column of Table 2.18. In the exact constraint checking mode, only the row and cell instances involved in the query with null probability will be considered for building the query result. Consequently, $P1$ row will be excluded ($Pr_{fresh}(ROW('P1')) = .4$ in Table 2.18). Invoking a contract that has been declared on TABLE (or COLUMN) granularity level does not actually change the elaboration of the query result but only the presentation of the probabilities associated to the table involved in the query.

- For Q2 query, the query result will not include product $P3$ because its PRICE value (e.g., '7777') has a non null probability of being inaccurate with respect to the contract `accurate` applied on CELL ($Pr_{accurate}(CELL('7777')) = 1$ in Table 2.18). The result presentation also includes the probabilities of every cell returned in the result.

- For Q3 query, the query result will not include $P1$ product because $P1$ row has a non null probability of being out-of-date with respect to the contract `fresh` $((Pr_{fresh}(ROW('P1')) = .4$ in Table 2.18) due to the non satisfaction of at least one of the constraints declared on `timeliness` and `lastUpdateTime` values.
- For Q4 query, $P1$ and $P3$ product rows will not be included in the result because they have non null probabilities of being inconsistent with respect to `consistent` contract (e.g., $Pr_{consistent}(ROW('P1')) = Pr_{consistent}(ROW('P3')) = 1$ in Table 2.18) due to the non satisfaction of the constraint on `ICshipDateStatus` measure. None of the cells is rejected by invoking the contract `fresh`.

Example 11. *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 "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 checked in the* <u>*EXACT*</u> *mode.*

Query#	Query	Results	
Q5	SELECT PROD_ID, CUST_CITY FROM PRODUCT P, CUSTOMER C WHERE P.CUST_ID=C.CUST_ID QWITH accurate ON ROW;	P1, 'Reading, UK'	$P_{accurate}(ROW('\text{P1'}))$:0 $P_{accurate}(ROW('\text{C1'}))$:0
Q6	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, 'Reading, NJ, US'	$P_{fresh}(ROW('\text{P2'}))$:0 $P_{fresh}(ROW('\text{C2'}))$:0 $P_{complete}(ROW('\text{P2'}))$:0 $P_{complete}(ROW('\text{C2'}))$:0
		P3, 'Reading, NJ, US'	$P_{fresh}(ROW('\text{P3'}))$:0 $P_{fresh}(ROW('\text{C2'}))$:0 $P_{complete}(ROW('\text{P3'}))$:0 $P_{complete}(ROW('\text{C2'}))$:0

Table 2.20: Examples of Join QWITH Queries with EXACT Constraint Checking Mode

For the sake of clarity, we suppose that all probabilities of CUSTOMER table are null except the ones given in Table 2.18, as $P_{accurate}(CELL('\text{C2'})) = .6$ and $P_{accurate}(ROW('\text{C2'})) = .5$.

- In Q5 query, the join of PRODUCT and CUSTOMER tables only consider the joining rows 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 $C2$ row with respect to contract `accurate` is $Pr_{accurate}(ROW('C2') = .5$ (see Table 2.18).
- In Q6 query, 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 ($Pr_{fresh}(ROW('P1') = .4$ in

Table 2.18). The probabilities of each row involved in the query are presented in the result.

2.7.2.2 Approximate Constraint Checking

In the APPROXIMATE mode, DB object rows and cells that satisfy approximately the constraints defined in the contracts of the QWITH query, *i.e.*, having a probability in [0,1[, will be considered for elaborating the query result. In our example, we arbitrarily fixe $\epsilon = \pm 5\%$ for numerical expected values expressed in the constraints. Several examples of QWITH queries with APPROXIMATE constraint checking mode are given in Table 2.21.

- For Q7 query, $P3$ will be included in the query result as it satisfied approximately the constraints on `outlierRatio` measure defined in the `accurate` contract with probability $P_{accurate}(ROW('\text{P3}'))$:0.143.
- For Q8 query, $P1$ will not be rejected as it satisfied approximately the constraints on `timeliness` or `lastUpdateTime` measures defined in the `fresh` contract with a probability $P_{fresh}(ROW('\text{P2}'))$:.4. It is presented at the end of the result list ordered by increasing freshness acceptability values.
- For Q9 query, $P2$ and $P3$ rows will be included in the result (by opposition to Q5 query) considering non null probabilities are still acceptable.

Query#	Query	Results	

Q7	SELECT PROD_ID, PRICE
	FROM PRODUCT
	WHERE PRICE > 10
	QWITH accurate ON ROW
	APPROXIMATE;
		P1,77
		P2,7777
		P3,7777
			$P_{accurate}(ROW('\text{P1}'))$:0
			$P_{accurate}(ROW('\text{P2}'))$:0
			$P_{accurate}(ROW('\text{P3}'))$:0.143

Q8	SELECT PROD_ID, CUST_CITY
	FROM PRODUCT P,
	CUSTOMER C
	WHERE
	P.CUST_ID=C.CUST_ID
	QWITH fresh ON ROW
	APPROXIMATE;
		P2, 'Reading, NJ, US'	$P_{fresh}(ROW('\text{P2}'))$:0
			$P_{fresh}(ROW('\text{C2}'))$:0

		P3, 'Reading, NJ, US'	$P_{fresh}(ROW('\text{P3}'))$:0 $P_{fresh}(ROW('\text{C2}'))$:0
		P1, 'Reading, UK'	$P_{fresh}(ROW('\text{C1}'))$:0 $P_{fresh}(ROW('\text{P2}'))$:.4

Q9	SELECT PROD_ID, CUST_CITY
	FROM PRODUCT P,
	CUSTOMER C
	WHERE
	P.CUST_ID=C.CUST_ID
	QWITH accurate ON ROW
	APPROXIMATE;
		P1, 'Reading, UK'	$P_{accurate}(ROW('\text{P1}'))$:0
			$P_{accurate}(ROW('\text{C1}'))$:0

		P2, 'Reading, NJ, US'	$P_{accurate}(ROW('\text{P2}'))$:0 $P_{accurate}(ROW('\text{C2}'))$:.5
		P3, 'Reading, NJ, US'	$P_{accurate}(ROW('\text{P3}'))$:0 $P_{accurate}(ROW('\text{C2}'))$:.5

Table 2.21: Examples of QWITH Queries in the Exact Constraint Checking Mode

2.7.3 Quality-Extended Query Processing

The processing of QWITH queries includes the four following steps, as illustrated in Figure 2.12:

1. *Query Parsing* - In this first step, the processor parses the SQL string representing the query and generates a tree representing the tables, attributes and expression that form part of the query. It also parses the QWITH part of the query.

2. *Metadata Discovery and QoD Checking* - In this second step, the processor accesses the catalog in search for the metadata describing the tables, types, analytical functions, contract types and contracts that are needed to solve the two parts of the query. At this stage the query is validated to make sure it is syntactically and semantically correct. The processor fetches the QoD matrix of each DB object instances at the ROW and CELL granularity levels that will be considered in the query processing. This step results in a query tree that is annotated by temporary QoD submatrices for the rows and cells strictly involved in the query.

3. *Query Optimization* - The query optimizer embedded in the processor generates an execution plan to solve the SQL query. Depending on the mode (EXACT or APPROXIMATE), temporary QoD submatrices associated to query tree nodes are used to prune the rows that will not be considered for the final result because they don't satisfy the constraints expressed in the contracts.

4. *Query Plan Execution* - The processor starts executing the query plan. All these results and associated probabilities are then send to the processor for further presentation processing, and then processor forwards the final values to the user.

2.8 Conclusion

2.8.1 Summary

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 query results. 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 chapter, 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. Analytical functions generate measures that are intended to characterize user-defined aspects of QoD dimensions. These measures are stored as metadata and indexed in a repository. They are checked at the query time 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

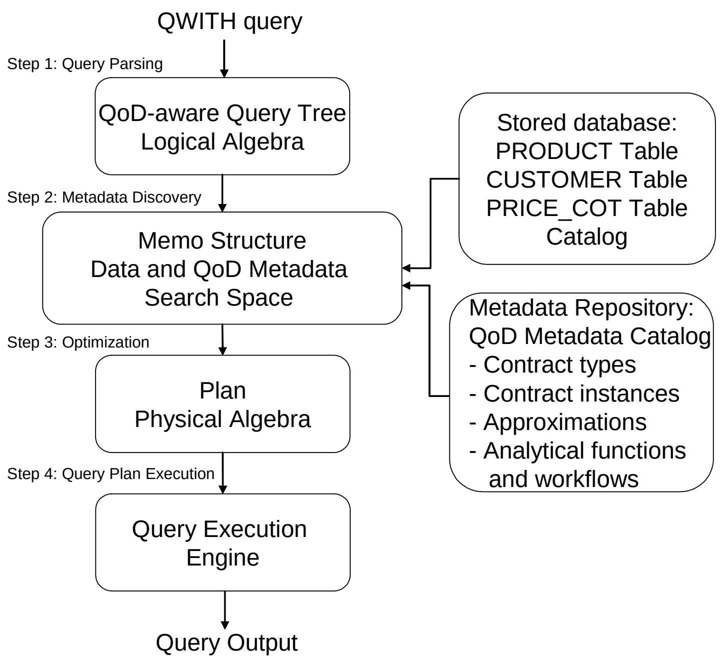


Figure 2.12: Quality-Extended Query Processing

QoD dimension. A scoring function is used to compute the probability that a given QoD dimension is acceptable with respect to a quality contract defining the set of (usually one-sided range) constraints on its associated QoD measures.

The query result is elaborated only with the database object instances (rows and cells) whose QoD measures match (exactly ou approximately) the constraints defined in the invoked contracts.

The main contributions presented in this chapter concern the key elements of the analytic approach adopted in XQuaL. In particular, the chapter put the emphasis on our work related to:

- *QoD Metadata modeling.* The proposed QoD metadata model is an extension to CWM metamodel. This metamodel is used to organize and store metadata generated by analytic workflows for QoD evaluation. They are stored in the metadata repository and currently indexed with ranged-encoded bitmap index.

- *QoD Evaluation*. We design analytic workflows dedicated to QoD evaluation and use several types of analytical functions in order to evaluate various aspects of QoD dimensions. The results of analytical functions are stored in the metadata repository as QoD measures characterizing data distribution properties, data quality problems, anomaly patterns, and any useful information for QoD evaluation at different granularity levels of the database (CELL, ROW, COLUMN, TABLE, DATABASE).

- *QoD-aware query language.* In this chapter, we present a query language extension, XQuaL, that allows the user to declare quality contracts, assign analytical functions for QoD evaluation, and express quality requirements as constraints with expected values. When querying the database with XQuaL, the user may extend his queries with a QWITH operator that invokes the existing quality contracts and constraints. When the QWITH part of the query is processed, depending on the mode (EXACT or APPROXIMATE) for checking the constraints defined on each QoD measure, the query engine builds a query tree whose nodes are annotated with the probabilities that the database object instances (rows and cells) involved in the QWITH query have acceptable QoD dimensions with respect to the user-defined contracts invoked in the query.

2.8.2 Research Perspectives

Many issues and research perspectives are raised by our proposals and in the following section, the more salient ones are exposed.

2.8.2.1 QoD metadata and analytic workflow modeling

During the metadata creation based on CWM, metadata may evolve in use, the inconsistencies of metadata, such as: content conflicts or the violation of constraints in the metamodel, redundancies of metadata, or inconsistencies arise inevitably. Nevertheless, CWM metamodel offers interesting perspectives for reasoning on metadata (Zhao & Huang, 2006). In particular, inconsistencies could be detected to improve the reliability of metadata, and also, the reliability of designed analytic workflows for QoD evaluation.

2.8.2.2 Design of analytic workflows

QoD evaluation can be based on the results of typically data-centric analyses. They require a sequence of activities and components for data retrieval, computation, and visualization, 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.), or even external, accessed via web services. In addition to providing users with a mechanism to compose and configure themselves components, and analytical functions dedicated to the detection of data anomalies and to QoD evaluation, our 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 intermediate results, QoD measures and descriptive metadata, and monitoring of analytic workflow execution. In this context, we also want to propose various alternatives assisting the user in the design of analytic workflows depending on resource limitations with possibly proposing workflow templates and frames for QoD evaluation.

2.8.2.3 Multidimensional indexing of QoD metadata

The constraints specified in the contract are currently one-sided range constraints or equality constraints over numerical or Boolean fields as totally ordered metadata. To extend and refine the proposal to partially ordered metadata in other metric spaces, we need to manipulate a indexing method with the two main following advantages: *i)* the method should operates in a *generic* metric space where information on data distribution are not necessarily required to derive the access cost model and predict the access performance, *ii)* the method should not be limited to a particular distance type by opposition to vector (Cartesian) space requiring the Euclidean distance, as we want to compare expected and actual QoD measure values possibly in different metric spaces.

2.8.2.4 QWITH query optimization

Since QWITH query optimization is an ongoing work with burning issues, a deliberate choice has been made to not include in this dissertation the detailed description of my proposals (not yet finalized) concerning: *i)* the heuristics used for building algebraic annotated QWITH query trees, *ii)* the technique for query rewriting, *iii)* the QWITH query cost model, and *iv)* an algorithm for relaxing the constraints of QWITH queries.

These aspects constitute the immediate perspectives of research and development for improving the QWITH query engine.

Chapter 3

Quality-Aware Data Mining

Contents

3.1 Introduction

The quality of data mining results and the validity of results interpretation essentially rely on the data preparation process and on the quality of the analyzed data sets. Indeed, data mining processes and applications require various forms of data preparation, correction and consolidation, combining complex data transformation operations and cleaning techniques. This is because the data input to the mining algorithms is assumed to conform to "nice" data distributions, containing no missing, inconsistent or incorrect values. This leaves a large gap between the available "dirty" data and the available machinery to process and analyze the data for discovering added-value knowledge and decision making.

In error-free data warehousing systems with perfectly clean data, knowledge discovery techniques can be relevantly used as decision making processes to automatically derive new knowledge patterns and new concepts from data.

Unfortunately, most of the time, this data is neither rigorously chosen from the various information sources with different degrees of quality and trust, nor carefully controlled and evaluated on the various quality dimensions. Deficiencies in data quality are a burning issue in many application areas, and become acute for practical applications of data mining and knowledge discovery (Pearson, 2005). Metadata are usually excluded (or largely ignored) from the classical data analysis process.

In this dissertation, we consider two dual aspects in the use of data mining techniques:

1. The first aspect concerns the appropriate use of statistical and data mining techniques in order to: *i)* measure or estimate factors characterizing data quality dimensions, *ii)* detect potential data quality problems, and *iii)* discover anomaly patterns in the data sets. For these purposes, we've designed relevant analytic workflows for data quality measurement. This aspects has been presented in Chapter 2.

2. The second aspect concerns the evaluation and improvement of data mining results in an application-driven decisional context with taking advantage of metadata that characterize data quality. This aspect is presented in this chapter with emphasis on association rule discovery and how QoD metadata may improve the results evaluation, interpretation, and usage with data quality awareness.

Among traditional descriptive data mining techniques, association rules discovery (Agrawal et al., 1993) identifies intra-transaction patterns in a database and describes how much the presence of a set of attributes in a database's record (or a transaction) implicates the presence of other distinct sets of attributes in the same record (respectively in the same transaction). The quality of association rules is commonly evaluated by the support and confidence measures (Agrawal et al., 1993). The support of a rule measures the occurrence frequency of the pattern in the rule while the confidence is the measure of the strength of implication. The problem of mining association rules is to generate all association rules that have support and confidence greater than user-specified minimum support and confidence thresholds. Besides support and confidence, other measures for knowledge quality evaluation (called interestingness measures) have been proposed in the literature with the purpose of supplying alternative indicators to the user in the understanding and use of the new discovered knowledge (Lavrač et al., 1999; Tan et al., 2002; Vaillant et al., 2004). But, to illustrate the impact of low-quality data over discovered association rule quality, one might legitimately wonder whether a so-called "interesting" rule, noted $LHS \rightarrow RHS$[1] is meaningful when 30% of the LHS data are not up-to-date anymore, 10% of the RHS data are not accurate, 15% of the LHS data are inconsistent with respect to a set of constraints, and there are 3% of approximate duplicates in the whole data set.

The contribution of this chapter is threefold:

1. a step-by-step method for integrating data quality awareness in the KDD process is proposed,

2. a method for scoring the quality of association rules that combines QoD measures is described,

3. a probabilistic cost model for estimating the cost of selecting *legitimately interesting* association rules based on data with *acceptable quality* is defined. The model gives the thresholds of three decision areas for the predicted class of the discovered rules (*i.e., legitimately interesting, potentially interesting*, or *not interesting*).

Our experiments on the KDD-Cup-98 data set confirmed our original assumption that is: interestingness measures of association rules are not self-sufficient and the quality of association rules depends on the quality of the data which the rules are computed from. Data quality includes various dimensions (such as data freshness, accuracy, completeness, etc.) which should be combined, integrated, explored, and presented together with data mining results.

Outline of the chapter. *The rest of the chapter is organized as follows. Section 3.2 presents a quality-aware KDD framework. In the context of quality-aware association rule mining, Section 3.3 describes a probabilistic decision model that can be used for post-processing the discovered association rules with tacking into account the quality of data they are discovered from. Section 3.4 presents an experimental study on quality-aware association rule discovery from the KDD-Cup-98 data set that shows the relevancy of our approach with respect to the cost of selecting rules (wrongly considered as interesting) when they are based on low-quality data. Section 3.5 provides concluding remarks and perspectives.*

[1]with the following semantics: Left-Hand Side implies Right-Hand Side

3.2 Data Quality Awareness for KDD Process

This section describes a pragmatic framework for data quality awareness preceding and during a classical knowledge discovery process. The grand view of the framework is depicted in Figure 3.1. This framework is divided into two parts: upstream and downstream of the KDD process (respectively left-hand and right-hand sides of Figure 3.1). It consists of five steps from $\boxed{\text{U1}}$ to $\boxed{\text{U5}}$ for KDD upstream activities and seven steps from $\boxed{\text{D1}}$ to $\boxed{\text{D7}}$ for KDD downstream activities, as they are described in the next sections.

3.2.1 KDD Upstream Quality-Aware Activities

In the KDD downstream side, main activities are parts of a quality-oriented data warehousing process. As previously presented in Chapter 2, this process has the particularity to compute QoD measures and store them in a repository as QoD metadata (with many other metadata describing the complete warehousing process). This can be based on the CWM metamodel and its quality management package extension we proposed in Chapter 2.

The first upstream step denoted $\boxed{\text{U1}}$ in Figure 3.1 consists of: *i)* selecting the data sources from which data will be extracted by automatic and massive import procedures and *ii)* defining a clear, consistent set of data quality requirements for the input data sources, the data warehouse, and if possible with respect to the decisionsal tasks and risks along the whole information supply chain.

In the $\boxed{\text{U2}}$ step, it is necessary to provide the basis for computing QoD measures characterizing relevant data quality dimensions depending on the quality goals and quality requirements specified in the first step $\boxed{\text{U1}}$. Data items do not have the same importance, they may not be equivalent neither from a strategic nor from a decisional point of view for a company, and they do not have to be considered in a uniform way but prioritized fo any processing activity (e.g., for scheduling and planning ETL tasks or data mining activities). The data quality measurement step $\boxed{\text{U2}}$ should provide precise and complete specifications of data quality measurement procedures in conformance with data quality requirements and goals expressed in step $\boxed{\text{U1}}$.

In our case, these specifications are expressed via *quality contract types* and instanciated with constraints declared in *quality contracts* instances. Once executed, the QoD measurement methods called in the contracts generate QoD measures that are stored as QoD metadata in the metadata repository of the system as illustrated in Figure 3.1. From a more technical and system-centered perspective, different levels of periodic QoD measurement and control can be implemented in a data warehouse as listed from A. to J. in Figure 3.2. The $\boxed{\text{U2}}$ step is applied before and after loading the multi-source data inside the data warehousing system. In the first case (pre-loading), this step consists of computing QoD measures from raw data extracted from the local sources into the data staging area with pre-validation methods that may be declared in the contract types (see H. in Figure 3.2). QoD metadata on local data sources may be used to drive the data loading process (for example, for loading only the most accurate and up-to-date data in the data warehousing system). In the second case (post-loading), QoD measures are computed *a posteriori* from the data that have been already integrated and stored in the system.

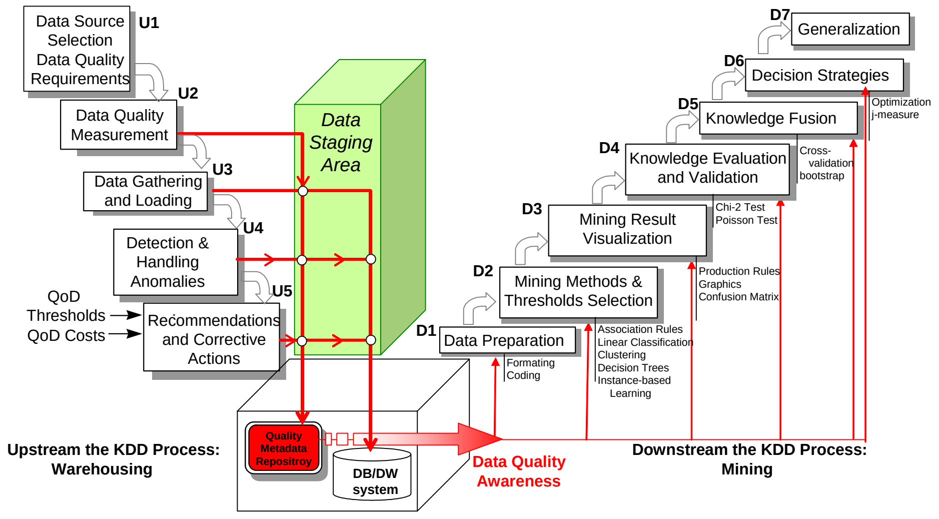

Data Source Selection Data Quality Requirements
U1
U2
Data Quality Measurement
U3
Data Gathering and Loading
U4
Detection & Handling Anomalies
U5
QoD Thresholds
QoD Costs
Recommendations and Corrective Actions
Data Staging Area
Quality Metadata Repositroy
DB/DW system
Upstream the KDD Process: Warehousing
Data Quality Awareness
Downstream the KDD Process: Mining
D1
Data Preparation
Formating Coding
D2
Mining Methods & Thresholds Selection
Association Rules Linear Classification Clustering Decision Trees Instance-based Learning
D3
Mining Result Visualization
Production Rules Graphics Confusion Matrix
D4
Knowledge Evaluation and Validation
Chi-2 Test Poisson Test
D5
Knowledge Fusion
Cross-validation bootstrap
D6
Decision Strategies
Optimization j-measure
D7
Generalization

Figure 3.1: General Framework of Data Quality Awareness for the KDD Process

The $\boxed{\text{U3}}$ step consists of possibly cleaning and reconciling data of the staging area with appropriate ETL tools and record linking strategies and loading them into the data warehouse. Different ETL operators are used for formatting the records, mapping or merging the matching records to load and store in the data warehousing system.

The goal of the $\boxed{\text{U4}}$ step is: *i)* to check QoD measures and detect data quality problems on integrated data with respect to pre-defined acceptance thresholds using post-validation programs (see I. in Figure 3.2), *ii)* to maintain and refresh the QoD metadata, and *iii)* more globally, to analyze the causes and costs of data quality problems.

Based on QoD metadata stored in the repository, the purpose of the $\boxed{\text{U5}}$ step is to propose corrective actions and recommendations for data quality improvements both on the data extracted from the sources and temporarily stored in the data staging area and on the integrated data in the system.

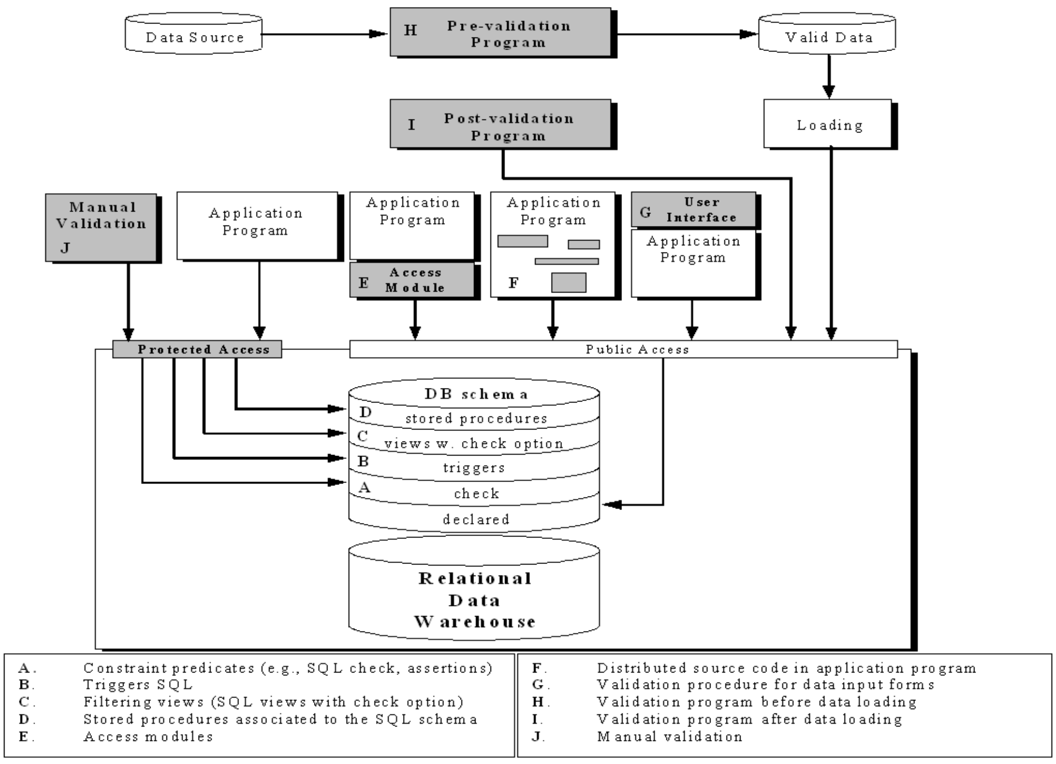

Data Source
H Pre-validation Program
Valid Data
I Post-validation Program
Loading
Manual Validation J
Application Program
Application Program
E Access Module
Application Program
F
G User Interface
Application Program
Protected Access
Public Access
DB schema
D stored procedures
C views w. check option
B triggers
A check
declared
Relational Data Warehouse
A. Constraint predicates (e.g., SQL check, assertions)
B. Triggers SQL
C. Filtering views (SQL views with check option)
D. Stored procedures associated to the SQL schema
E. Access modules
F. Distributed source code in application program
G. Validation procedure for data input forms
H. Validation program before data loading
I. Validation program after data loading
J. Manual validation

Figure 3.2: Different Levels for Measuring Data Quality

3.2.2 KDD Downstream: Quality Metadata Exploitation

Starting the KDD process, the classical first step is data preparation noted $\boxed{\text{D1}}$ for downstream of the KDD process in Figure 3.1. It consists of a succession of tasks, such as:

i) selecting data sets and object instances. Ideally, this task should be done with considering a stratification of the data sets with respect to the data quality required for a specific decisional context,

ii) selecting and weighting the variables and features,

iii) (re-)coding data,

iv) analyzing missing values. This task can be based on the measures stored in the metadata repository that are related to completeness dimension. Different kinds of missing values may also be distinguished (e.g., unknown, not applicable or nonexistent),

v) detecting data anomalies: again, this task may could take advantage of the QoD metadata stored in the repository as they characterize potential data quality problems (e.g., outliers, accuracy or consistency anomalies detected by statistical constraints). When setting up the methods and data to use for the analysis, it's very important to be aware of the side-effects of the strategy that will omit (or include) outliers or missing data from the analysis,

vi) homogenizing the data files,

vii) discretizing the continuous attributes, and

viii) using (if needed) quantization strategies for real variables (e.g., defining quartiles or deciles) or high-dimension reduction techniques, as their results may have already been computed and stored in the metadata repository.

The next steps $\boxed{\text{D2}}$ and $\boxed{\text{D3}}$ in Figure 3.1 consist of selecting the mining methods, configuration parameters and thresholds and, therefore the knowledge representation for visualizing the mining results (e.g., decision tables, decision trees, classification rules, association rules, instance-based learning representation or clusters).

For quality-awareness, these steps (in particular $\boxed{\text{D3}}$) should also provide the visualization of a relevant set of data quality measures previously computed in step $\boxed{\text{U2}}$.

The added-value of QoD metadata consists in their exploitation and timely presentation together with mining results. QoD metadata are additional explanatory information for evaluating the quality and validity of discovered knowledge. They are useful for validating a single mining process (step $\boxed{\text{D4}}$) or a combination of the results of several mining techniques (step $\boxed{\text{D5}}$). They also may drive the generalization of results and the decision strategies (steps $\boxed{\text{D6}}$ and $\boxed{\text{D7}}$).

3.2.3 Ilustrative Example in Marketing

The principle of marketing is matching products and advertising to customers. The initial set of goals of marketing are to interpret data from a variety of sources that may include census data, life-style clusters, consumer panels (with looking inside shopping bags for instance) and point-of-sale information. The outcome of this interpretation is a market segmentation which is then used as a basis for product positioning and further market planning and execution. Each consumer household may also use a special identification card at specific stores each time a purchase is made. Use of this card triggers a detailed record of the purchases made and is then stored. This detailed purchase information may then be related to other details about the household, previous purchases, and previous promotions and advertisements that members of the household were exposed to. If the information obtained in marketing research is to help decision making, it should be relevant, cost-effective, **timely**, and **valid**.

Consider some of the problems that may arise in a typical scenario. A marketing research team sets up a booth in a busy shopping mall. People are given a demonstration or sample of a new product and are then asked how it compares to competitor products. This information may be collected either as a questionnaire or through a formal interview. This information is presumably relevant because the shoppers are a fairly typical cross-section of people who buy the product in the mall. The marketing researchers may actually screen people according to criteria such as age, gender, or type of clothing, etc. Data are stored in tables as illustrated in Table 3.1.

CUSTOMER Table

CUST_ID	FN	LN	GENDER	AGE	CUST_CITY	CUST_COUNTRY
C1	Joe	Smith	M	NULL	Reading	UK
C2	Joy	Smith	NULL	30	Reading, NJ	US
C3	John	Smitt	F	33	Reading	NULL

INTERVIEW_CAMPAIGN Table

PROD_ID	CUST_ID	TV_VIEWER	REBATE_CLAIM
P1	C1	TRUE	TRUE
P2	C2	FALSE	FALSE
P3	C2	TRUE	TRUE
P4	C3	TRUE	TRUE

PRODUCT_SALES_VIEW Table

PROD_ID	CASH_REBATE	MONTH	SALES
P1	12	JULY-04	4526
P2	12	DEC-04	5630
P3	16	OCT-04	5447
P4	100	OCT-04	3267

Table 3.1: Marketing Example

The first upstream step of our approach $\boxed{\text{U1}}$ in Figure 3.1 consists of: *i)* selecting the relevant sources of data (e.g., product and purchase data extracted from CRM_DB, point-of-sale data collected by bar code scanning, and census data), and *ii)* defining a clear, consistent set of data quality requirements for the marketing database. For the example, the considered data quality dimensions for the input data sets are the following:

- Data consistency measure is defined in this example as the percentage of values that satisfy a set of pre-defined integrity constraints on one (or a set of) record(s) (e.g., the values "FN = 'John', LN='Smitt', gender = F" of record $C3$ does not satisfy a constraint between the gender and the first name). The constraints may involve additional knowledge (e.g., a thesaurus or a dictionary of first names and gender conformance).

- Data completeness measure is defined as the percentage of data fields having non-null values.

- Data accuracy measure is defined as the percentage of data having values that fall within their respective domain of allowable values (e.g., the CASH_REBATE has to be in]0,20]).

- Data timeliness measure is defined by a decimal that represents the quantity of hours elapsed since the last time the data has been updated.

In the $\boxed{\text{U2}}$ step, the measures, statistical controls or EDM summaries characterizing data quality dimensions are computed and stored in the metadata repository.

The $\boxed{\text{U3}}$ step consists of linking records and aggregating data before loading data into the marketing database. Before integration, the records coming from CRM_DB may be transformed with respect to the schema of MARKETING_DB presented in Table 3.1.

The goal of the $\boxed{\text{U4}}$ step is to detect the problems of data quality using post-validation programs (see I. in Figure 3.2). Then the quality dimensions such as consistency, completeness, accuracy, and freshness are computed.

In the $\boxed{\text{U5}}$ step, two corrective actions may be proposed to avoid the usage of one of the input data sources or to improve the way the answers of the questionnaire are collected and entered in the database.

The approach proposed in the framework is to combine the data quality awareness with the data mining process itself. The quality measures stored as QoD metadata in the metadata repository are used and combined to characterize the quality of the discovered association rules (decision trees or clusters), resulting from a data mining process. There are as many repeated downstream processes as the number of data mining models. Different mining objectives can lead to different models and different quality measures.

Starting the KDD process, the first step of data preparation noted $\boxed{\text{D1}}$ in Figure 3.1 consists of selecting the data sets, and variables, (re-)coding and normalizing the data, and analyzing missing values (using the pre-computed completeness measure). For instance, deviations from the standard and from typical correlations are detected (completing the accuracy measure).

For the next steps $\boxed{\text{D2}}$ and $\boxed{\text{D3}}$ in Figure 3.1, consider that association rule discovery has been selected as one of the mining methods for prediction. An example of applied rule-based prediction is the prediction of demand for stock items on the basis of historical marketing data. In this case, the premise of a predictive rule is a set of conditions such as promotional and advertising campaigns undertaken, and the conclusion of the rule is the number of orders for a given item within a time period (e.g., available in PRODUCT_SALES_VIEW table). Here is an example of discovered rule:

```
IF    $10 \leq CASH\_REBATE \leq 15$ AND
      $3,000 \leq SALES \leq 6,000$ AND
      $300,000 \leq TV\_VIEWER \leq 600,000$
THEN           $5,000 \leq REBATE\_CLAIM \leq 8,000$
         WITH confidence: 100% AND
         applicable months: July 2004, Sept. 2004,
                  Oct. 2004, Nov. 2004, Dec. 2004, Jan. 2005.
```

The interpretation of this rule is "if an active rebate between \$10 and \$15 is in place and the number of product sales are currently between 3,000 and 6,000 and between 300,000 to 600,000 TV viewers are reached, then further sales between

5,000 and 8,000 taking advantage of the rebate can be expected". This rule can be discovered from historical sales data for products. By using this rule one can perform better material resource planning, anticipate demand, and know when to run rebate programs.

Data quality specific issues which can affect a predictive rule are: *i)* the historical time period from which the rule is discovered (timeliness), *ii)* the lack of completeness or consistency of some of the analyzed data fields, *iii)* the presence of invalid, inaccurate or incorrect data that may are considered in the rule discovery process. In this particular example, consistency, completeness, and accuracy of the data values of each attribute involved in the rule can be characterized by percentages as previously defined. Then, the rule interpretation makes a sense if and only if data the rule has been computed from are 100% consistent, 100% complete, and 100% accurate. If it is not the case, adding QoD metadata would certainly help decision makers to cautiously and appropriately use the discovered rules.

Predictive stability and validity are not just problems for rule-based prediction. It affects all forms of prediction. One strategy for dealing with these problems is to continuously test and update rules. Rules should be scored according to how well they predict new data. Rules that fall below a certain level of performance should be discarded and replaced by new rules that provide a better level of prediction.

- One way of selecting new rules in this situation is to deliberately discard the data that supported the faulty rule and carry out rule discovery on the same conclusion in order to identify the replacement rule. Thus rule-based prediction requires continuous testing, validation, and refinement in order to be maximally effective.

- Another complementary way is to first scrutinize data involved in the faulty rule, and then exploit data quality measures computed from the data sets composing the premise and the conclusion of the rule in order to characterize and understand the quality of the data which the rule has been computed from. These metadata may explain why the rule even with good confidence may be faulty and useless for prediction.

3.3 Quality-Aware Rule Mining

Our assumption is that the quality of an association rule depends on the quality of the data which the rule is computed from. In order to prove this assumption, this section presents the formal definitions of our approach that uses precomputed QoD measures and combines them for determining the quality of association rules.

3.3.1 Preliminary Definitions for Association Rule Quality

Let $\mathcal{I}$ be a set of literals, called *items* in the set of database object instances, $\mathcal{I} \subseteq \mathcal{O}$. An *association rule* R is an expression $LHS \rightarrow RHS$, where $LHS, RHS \subseteq \mathcal{I}$ and $LHS \cap RHS = \emptyset$. LHS and RHS are conjunctions of variables such as the extension of the L̲eft-H̲and S̲ide LHS of the rule is:

$g(LHS) = x_1 \wedge x_2 \wedge \ldots \wedge x_n$
and the extension of the Right-Hand Side RHS of the rule is:
$g(RHS) = y_1 \wedge y_2 \wedge \ldots \wedge y_{n'}$.

Let $j(j = 1, 2, \ldots, k)$ be the k dimensions of data quality (e.g., data completeness, freshness, accuracy, consistency, etc.). As defined in Chapter 2, a probablility may be associated to each QoD dimension based on the satisfaction of constraints on QoD measures. Any scoring function combining the QoD measures associated to one QoD dimension may also be used. We denoted QoD score, the value assigned to the evaluation of one QoD dimension.

Let $q_j(\mathcal{I}) \in [0, 1]$ be the QoD scores associated to the quality dimension j computed from the set $\mathcal{I}$. The structure, that keeps the probability values of all quality dimensions for each data set $\mathcal{I}$ is called *quality vector* denoted $q(\mathcal{I})$. The set of all possible quality vectors is called *quality space* $\mathcal{Q}$.

Definition 3.3.1. *The quality of the association rule R is defined by a fusion function denoted $\otimes_j$ specific for each quality dimension j that merges the components of the quality vectors of the data sets constituting the extension of the right-hand and left-hand sides of the rule. The quality of the rule R is defined as a k-dimensional vector such as:*

$$\begin{aligned} q(R) &= \begin{pmatrix} q_1(R) \\ q_2(R) \\ \ldots \\ q_k(R) \end{pmatrix} = \begin{pmatrix} q_1(LHS) \otimes_1 q_1(RHS) \\ q_2(LHS) \otimes_2 q_2(RHS) \\ \ldots \\ q_k(LHS) \otimes_k q_k(RHS) \end{pmatrix} \\ &= \begin{pmatrix} q_1(x_1) \otimes_1 \ldots \otimes_1 q_1(x_n) \otimes_1 q_1(y_1) \otimes_1 \ldots \otimes_1 q_1(y_{n'}) \\ q_2(x_1) \otimes_2 \ldots \otimes_2 q_2(x_n) \otimes_2 q_2(y_1) \otimes_2 \ldots \otimes_2 q_2(y_{n'}) \\ \ldots \\ q_k(x_1) \otimes_k \ldots \otimes_k q_k(x_n) \otimes_k q_k(y_1) \otimes_k \ldots \otimes_k q_k(y_{n'}) \end{pmatrix} \end{aligned} \quad (3.1)$$

The average quality of the association rule R denoted $\overline{q}(R)$ can be computed by a weighted sum of the quality vector components of the rule as follows:

$$\overline{q}(R) = \sum_{j=1}^{k} w_j \cdot q_j(R) \quad (3.2)$$

with w_j the weight of the quality dimension j. We assume the weights are normalized:

$$\sum_{j=1}^{k} w_j = 1 \quad (3.3)$$

Definition 3.3.2. *Let T be the domain of values of the quality score $q_j(\mathcal{I})$ for the data set $\mathcal{I}$ on the quality dimension j. The fusion function denoted $\otimes_j$ is commutative and associative such as: $\otimes_j$: $T \times T \to T$. The fusion function may have different definitions depending on the considered quality dimension j in order to suit the properties of each quality dimension.*

Based on the main data quality dimensions (e.g., freshness, accuracy, completeness, and consistency), Table 3.2 presents several examples of fusion function allowing the combination of quality measures per quality dimension for two data sets noted x and y in the rule $x \rightarrow y$.

j	**Quality Dimension**	**Fusion Function $\otimes_j$**	**Quality Dimension of the rule $x \rightarrow y$**
1	Freshness	$\min[q_1(x), q_1(y)]$	The freshness of the association rule $x \rightarrow y$ is estimated pessimistically as the lower score of freshness of the two data sets composing the rule.
2	Accuracy	$q_2(x) \cdot q_2(y)$	The accuracy of the association rule $x \rightarrow y$ is estimated as the probability of accuracy of the two data sets x and y of the rule.
3	Completeness	$q_3(x) + q_3(y)$ $-q_3(x) \cdot q_3(y)$	The completeness of the association rule $x \rightarrow y$ is estimated as the probability that one of the two data sets of the rule is complete.
4	Consistency	$\max[q_4(x), q_4(y)]$	The consistency of the association rule $x \rightarrow y$ is estimated optimistically as the higher score of consistency of the two data sets composing the rule.

Table 3.2: Fusion Function Examples for Scoring Quality Dimensions of Association Rule

3.3.2 Probabilistic Decision Model for Quality and Cost Optimal Association Rule Mining

In this section, we present our approach that was initially inspired from the approach of Verykios et al. (2003) who first defined a Bayesian decision model for record matching. We've extended the previous approach so that it considers the misclassification problem that may occur in the classification decision. We apply our approach in the totally different context for association rule selection considering the quality of the data which the rule is computed from is probably acceptable with respect to each dimension that characterizes data quality. The underlying key concept is the notion of *data quality acceptability*. Data quality with respect to one specific dimension is acceptable (or not) based on the measures that have been computed from data and that are in conformance with precise constraints. Some measurement procedures may fail in detecting certain errors and anomalies or their scope does not cover the range of possible data quality problems. This problem of misclassification is considered in our approach and applied for post-processing of discovered association rules.

Based both on good interestingness measures and on the actual quality status of the data sets composing the left-hand and right-hand sides of the rule, we consider that selecting an association rule (among the top N) is a decision that designates the rule as:

- *legitimately interesting* (noted D_1), the rule has good support and confidence measures and it is worth being selected considering the acceptable quality of data which the rule is computed from and the cost of the decision if it is made based on the rule,

- *potentially interesting* (D_2), the rule has good support and confidence measures and it is potentially worth being selected depending on the quality of data it is computed from and the cost of decision,
- *not interesting* (D_3), the rule has good support and confidence measures but it is not worth being selected because it relies on data with low quality (non acceptable) and the cost of the decision.

Consider the data item $x \in LHS \cup RHS$ of a given association rule, we use:

- $P_{UE}(x)$ denotes the probability that the quality of data x will be classified Unacceptable due to Erroneous or "low-quality" data,
- $P_{AC}(x)$ denotes the probability that the quality of data x will be classified as Acceptable with Correct data (*i.e.*, in the range of acceptable values),
- $P_{AE}(x)$ represents the probability that the quality of data x is classified as Acceptable with "actually erroneous" (AE) data,
- $P_{UC}(x)$ represents the probability that the quality of data is classified as Unacceptable with "actually Correct" (UC) data (see Figure 3.3).

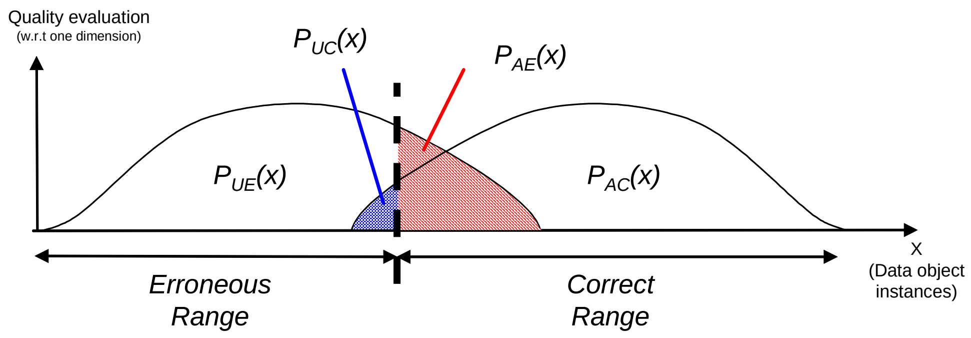


Figure 3.3: Classification Probabilities

For $\overline{q}(R) \in \mathcal{Q}$, an arbitrary average quality vector of the rule $R : LHS \rightarrow RHS$ defined on the data sets in $LHS \cup RHS$, we denote by $P(\overline{q} \in \mathcal{Q}|AC)$ or $f_{AC}(\overline{q})$ the conditional probability that the average quality vector $\overline{q}$ corresponds to the data sets that are classified as correct (AC), that is with acceptable quality dimensions. Similarly, we denote by $P(\overline{q} \in \mathcal{Q}|UE)$ or $f_{UE}(\overline{q})$ the conditional probability that the average quality vector $\overline{q}$ appropriately reflects the data sets that are classified erroneous (UE).

We denote by d the decision of the predicted class of the rule, *i.e., legitimately interesting* (D_1), *potentially interesting* (D_2), or *not interesting* (D_3), and by s the actual status of quality of the data sets upon which the rule has been computed.

Let us denote by $P(d = D_i, s = j)$ and $P(d = D_i|s = j)$ correspondingly, the joint and the conditional probability that the decision D_i is taken, when the actual status of data quality is j (*i.e.*, AC, UE, AE, UC).

We also denote by c_{ij} the cost of making a decision D_i for classifying an association rule with the actual data quality status j of the data sets composing the two parts of the rule.

Example 12. *As an illustrative example, Table 3.3 shows tentative unit costs developed by the staff of the direct marketing department on the basis of consideration of the consequences of the decisions on selecting and using the discovered association rules in both cases: with and without misclassification. In Table 3.3,*

- c_{10} *is the cost of a confident decision (D_1) for the selection of a legitimately interesting rule based on data with acceptable quality data (AC).*

- c_{21} *is the cost of a neutral decision (D_2) for the selection of a potentially interesting rule based on low-quality data (UE).*

- c_{33} *is the cost of a suspicious decision (D_3) for the selection of a rule that is not legitimately interesting because it is based on data with low-quality but actually detected as correct (UC).*

Decision for Rule Selection	Cost#	Data Quality Status	Cost($) **without** misclassification	Cost($) **with** misclassification
D_1	c_{10}	AC	0	0
	c_{11}	UE	1000	1000
	c_{12}	AE	0	1000
	c_{13}	UC	0	500
D_2	c_{20}	AC	50	50
	c_{21}	UE	50	50
	c_{22}	AE	0	500
	c_{23}	UC	0	500
D_3	c_{30}	AC	500	500
	c_{31}	UE	0	0
	c_{32}	AE	0	500
	c_{33}	UC	0	1000

Table 3.3: Example of Costs of Various Decisions for Classifying Association Rules Based on Data Quality

Based on the example presented in Table 3.3, we can see how the cost of decisions could affect the result of the selection among interesting association rules. And we need to minimize the mean cost $\overline{c}$ that results from making such a decision. The corresponding mean cost $\overline{c}$ is written as follows:

$$\begin{aligned}\overline{c} &= c_{10}.P(d=D_1, s=AC) + c_{20}.P(d=D_2, s=AC) + c_{30}.P(d=D_3, s=AC)\\ &+c_{11}.P(d=D_1, s=UE) + c_{21}.P(d=D_2, s=UE) + c_{31}.P(d=D_3, s=UE)\\ &+c_{12}.P(d=D_1, s=AE) + c_{22}.P(d=D_2, s=AE) + c_{32}.P(d=D_3, s=AE)\\ &+c_{13}.P(d=D_1, s=UC) + c_{23}.P(d=D_2, s=UC) + c_{33}.P(d=D_3, s=UC)\end{aligned} \tag{3.4}$$

From the Bayes theorem, the following is true:

$$P(d=D_i, s=j) = P(d=D_i|s=j).P(s=j) \tag{3.5}$$

where $i = 1, 2, 3$ and $j = AC, UE, AE, UC$.

The mean cost $\overline{c}$ in Eq. 3.4 based on Eq. 3.5 is written as follows:

$$\begin{aligned}\overline{c} &= c_{10}\cdot P(d=D_1|s=AC)\cdot P(s=AC) + c_{20}\cdot P(d=D_2|s=AC)\cdot P(s=AC)\\ &+ c_{30}\cdot P(d=D_3|s=AC)\cdot P(s=AC) + c_{11}\cdot P(d=D_1|s=UE)\cdot P(s=UE)\\ &+ c_{21}\cdot P(d=D_2|s=UE)\cdot P(s=UE) + c_{31}\cdot P(d=D_3|s=UE)\cdot P(s=UE)\\ &+ c_{12}\cdot P(d=D_1|s=AE)\cdot P(s=AE) + c_{22}\cdot P(d=D_2|s=AE)\cdot P(s=AE)\\ &+ c_{32}\cdot P(d=D_3|s=AE)\cdot P(s=AE) + c_{13}\cdot P(d=D_1|s=UC)\cdot P(s=UC)\\ &+ c_{23}\cdot P(d=D_2|s=UC)\cdot P(s=UC) + c_{33}\cdot P(d=D_3|s=UC)\cdot P(s=UC)\end{aligned} \tag{3.6}$$

Let us also assume that $\overline{q}$ is the average quality vector drawn randomly from the space of all quality vectors of the item sets of the rule. The following equality holds for the conditional probability $P(d=D_i|s=j)$:

$$P(d=D_i|s=j) = \sum_{\overline{q}\in Q_i} f_j(\overline{q}) \tag{3.7}$$

where $i = 1, 2, 3$ and $j = AC, UE, AE, UC$.

f_j is the probability density of the quality vectors when the actual quality status is j.

We also denote the a priori probability of AC or else $P(s=AC)$ as π^0, the a priori probability of $P(s=UC) = \pi^0_{UC}$, the a priori probability of $P(s=AE) = \pi^0_{AE}$ and the a priori probability of $P(s=UE) = 1-(\pi^0+\pi^0_{AE}+\pi^0_{UC})$.

Without misclassification region $P(s=UE)$ could be simplified as $1-\pi^0$.

For the sake of clarity, we define three variables denoted κ_{AE}, κ_{UC} and κ_{UE} as:

$$\kappa_{AE} = \frac{f_{AE}}{f_{AC}}\cdot\frac{\pi^0_{AE}}{\pi^0} \tag{3.8}$$

$$\kappa_{UC} = \frac{f_{UC}}{f_{AC}}\cdot\frac{\pi^0_{UC}}{\pi^0} \tag{3.9}$$

$$\kappa_{UE} = \frac{f_{UE}}{f_{AC}} \cdot \frac{1-(\pi^0 + \pi^0_{UC} + \pi^0_{AE})}{\pi^0} \tag{3.10}$$

By using Eq. 3.7 and by dropping the dependent vector variable $\overline{q}$, Eq. 3.6 becomes:

$$\begin{aligned}\overline{c} = \pi^0 \cdot f_{AC} \cdot \quad & \Big(\textstyle\sum_{\overline{q}\in\mathcal{Q}_1} \quad [c_{10} + c_{11}\cdot\kappa_{UE} + c_{12}\cdot\kappa_{AE} + c_{13}\cdot\kappa_{UC}] \\ & + \textstyle\sum_{\overline{q}\in\mathcal{Q}_2} \quad [c_{20} + c_{21}\cdot\kappa_{UE} + c_{22}\cdot\kappa_{AE} + c_{23}\cdot\kappa_{UC}] \\ & + \textstyle\sum_{\overline{q}\in\mathcal{Q}_3} \quad [c_{30} + c_{31}\cdot\kappa_{UE} + c_{32}\cdot\kappa_{AE} + c_{33}\cdot\kappa_{UC}]\Big)\end{aligned} \tag{3.11}$$

Every point $\overline{q}$ in the quality space $\mathcal{Q}$ belongs to the partitions of quality $\mathcal{Q}_1$ or $\mathcal{Q}_2$ or $\mathcal{Q}_3$ that correspond respectively to the partitions of the decision space: D_1, or D_2 or D_3 in such a way that its contribution to the mean cost is minimum. This will lead to the optimal selection for the three sets of rules which we denote by D^0_1, D^0_2, and D^0_3.

3.3.2.1 Cost Optimal Selection of Rule with Misclassification

In the case of misclassification, Eq. 3.11 will lead to the optimal selection for the three sets of rules which we denote by D^0_1, D^0_2 and D^0_3. In order to minimize the cost, a point $\overline{q}$ is assigned to one of the three optimal areas as follows:
To D^0_1 if:

$$\begin{aligned} & c_{10}+ \quad c_{11}\cdot\kappa_{UE} + c_{12}\cdot\kappa_{AE} + c_{13}\cdot\kappa_{UC} \\ & \qquad \le c_{30} + c_{31}\cdot\kappa_{UE} + c_{32}\cdot\kappa_{AE} + c_{33}\cdot\kappa_{UC} \\ \text{and, } & c_{10}+ \quad c_{11}\cdot\kappa_{UE} + c_{12}\cdot\kappa_{AE} + c_{13}\cdot\kappa_{UC} \\ & \qquad \le c_{20} + c_{21}\cdot\kappa_{UE} + c_{22}\cdot\kappa_{AE} + c_{23}\cdot\kappa_{UC}\end{aligned}$$

To D^0_2 if:

$$\begin{aligned} & c_{20}+ \quad c_{21}\cdot\kappa_{UE} + c_{22}\cdot\kappa_{AE} + c_{23}\cdot\kappa_{UC} \\ & \qquad \le c_{30} + c_{31}\cdot\kappa_{UE} + c_{32}\cdot\kappa_{AE} + c_{33}\cdot\kappa_{UC} \\ \text{and, } & c_{20}+ \quad c_{21}\cdot\kappa_{UE} + c_{22}\cdot\kappa_{AE} + c_{23}\cdot\kappa_{UC} \\ & \qquad \le c_{10} + c_{11}\cdot\kappa_{UE} + c_{12}\cdot\kappa_{AE} + c_{13}\cdot\kappa_{UC}\end{aligned}$$

To D^0_3 if:

$$\begin{aligned} & c_{30}+ \quad c_{31}\cdot\kappa_{UE} + c_{32}\cdot\kappa_{AE} + c_{33}\cdot\kappa_{UC} \\ & \qquad \le c_{10} + c_{11}\cdot\kappa_{UE} + c_{12}\cdot\kappa_{AE} + c_{13}\cdot\kappa_{UC} \\ \text{and, } & c_{30}+ \quad c_{31}\cdot\kappa_{UE} + c_{32}\cdot\kappa_{AE} + c_{33}\cdot\kappa_{UC} \\ & \qquad \le c_{20} + c_{21}\cdot\kappa_{UE} + c_{22}\cdot\kappa_{AE} + c_{23}\cdot\kappa_{UC}\end{aligned}$$

The three decision areas for rule selection are then defined as follows:

$$D^0_1 = \left\{ \begin{array}{l} \overline{q} : \kappa_{UE} \le \frac{(c_{30}-c_{10})}{(c_{11}-c_{31})} + \kappa_{AE}\cdot\frac{(c_{32}-c_{12})}{(c_{11}-c_{31})} + \kappa_{UC}\cdot\frac{(c_{33}-c_{13})}{(c_{11}-c_{31})}) \\ \text{and, } \kappa_{UE} \le \frac{(c_{20}-c_{10})}{(c_{11}-c_{21})} + \kappa_{AE}\cdot\frac{(c_{12}-c_{22})}{(c_{11}-c_{21})} + \kappa_{UC}\cdot\frac{(c_{13}-c_{23})}{(c_{11}-c_{21})}) \end{array}\right.$$

$$D_2^0 = \begin{cases} \overline{q} : \kappa_{UE} \leq \frac{(c_{30}-c_{20})}{(c_{21}-c_{31})} + \kappa_{AE} \cdot \frac{(c_{32}-c_{22})}{(c_{21}-c_{31})} + \kappa_{UC} \cdot \frac{(c_{33}-c_{23})}{(c_{21}-c_{31})}) \\ \text{and, } \kappa_{UE} \geq \frac{(c_{20}-c_{10})}{(c_{11}-c_{21})} + \kappa_{AE} \cdot \frac{(c_{12}-c_{22})}{(c_{11}-c_{21})} + \kappa_{UC} \cdot \frac{(c_{13}-c_{23})}{(c_{11}-c_{21})}) \end{cases}$$

$$D_3^0 = \begin{cases} \overline{q} : \kappa_{UE} \geq \frac{(c_{30}-c_{10})}{(c_{11}-c_{31})} + \kappa_{AE} \cdot \frac{(c_{32}-c_{12})}{(c_{11}-c_{31})} + \kappa_{UC} \cdot \frac{(c_{33}-c_{13})}{(c_{11}-c_{31})}) \\ \text{and, } \kappa_{UE} \geq \frac{(c_{30}-c_{20})}{(c_{21}-c_{31})} + \kappa_{AE} \cdot \frac{(c_{33}-c_{23})}{(c_{21}-c_{31})} + \kappa_{UC} \cdot \frac{(c_{32}-c_{22})}{(c_{21}-c_{31})}) \end{cases}$$

In the case of misclassification these inequalities give rise to three different threshold values λ, ρ and ν (respectively for *legitimately*, *potentially* and *not interesting* rules) in the decision space as illustrated in Figure 3.4.

$$\lambda = \frac{1}{(c_{11} - c_{21})} \cdot (c_{20} - c_{10} + \kappa_{AE} \cdot (c_{12} - c_{22}) + \kappa_{UC} \cdot (c_{13} - c_{23})) \tag{3.12}$$

$$\rho = \frac{1}{(c_{11} - c_{31})} \cdot (c_{30} - c_{10} + \kappa_{AE} \cdot (c_{32} - c_{12}) + \kappa_{UC} \cdot (c_{33} - c_{13})) \tag{3.13}$$

$$\nu = \frac{1}{(c_{21} - c_{31})} \cdot (c_{30} - c_{20} + \kappa_{AE} \cdot (c_{33} - c_{23}) + \kappa_{UC} \cdot (c_{32} - c_{22})) \tag{3.14}$$

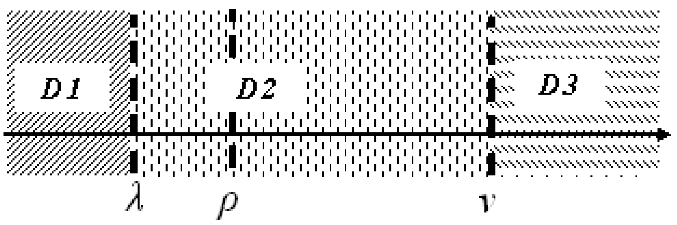


Figure 3.4: Decision Areas for Rule Post-Selection

3.3.2.2 Cost Optimal Selection of Rule without Misclassification

For the sake of simplicity, let us now consider the case of the absence of the misclassification region. f_{UC}, f_{AE}, κ_{UC}, and κ_{AE}, π^0_{AE}, and π^0_{UC} are null. We also assume that the a priori probability that a vector belongs to AC is equal to the a priori probability that the same vector belongs to UE. κ_{UE} is equal to $\frac{f_{UE}}{f_{AC}}$ and we can thus simplify the inequalities above:

$$D_1^0 = \{\overline{q} : \frac{f_{UE}}{f_{AC}} \leq \frac{\pi^0}{1-\pi^0} \cdot \frac{c_{30}-c_{10}}{c_{11}-c_{31}} \text{ and, } \frac{f_{UE}}{f_{AC}} \leq \frac{\pi^0}{1-\pi^0} \cdot \frac{c_{20}-c_{10}}{c_{11}-c_{21}}\} \tag{3.15}$$

$$D_2^0 = \{\overline{q}: \frac{f_{UE}}{f_{AC}} \leq \frac{\pi^0}{1-\pi^0} \cdot \frac{c_{30}-c_{20}}{c_{21}-c_{31}} \text{ and, } \frac{f_{UE}}{f_{AC}} \geq \frac{\pi^0}{1-\pi^0} \cdot \frac{c_{20}-c_{10}}{c_{11}-c_{21}}\} \tag{3.16}$$

$$D_3^0 = \{\overline{q} : \frac{f_{UE}}{f_{AC}} \geq \frac{\pi^0}{1-\pi^0} \cdot \frac{c_{30}-c_{10}}{c_{11}-c_{31}} \text{ and, } \frac{f_{UE}}{f_{AC}} \geq \frac{\pi^0}{1-\pi^0} \cdot \frac{c_{30}-c_{20}}{c_{21}-c_{31}}\} \tag{3.17}$$

The inequalities (3.15), (3.16) and (3.17) give rise to three different threshold values λ, ρ and ν (respectively for *legitimately, potentially and not interesting* rules) in the decision space that define concretely the decision regions based on the cost of rule selection decision with the following relationship:

$$\lambda = \frac{\pi^0}{1-\pi^0} \cdot \frac{c_{20}-c_{10}}{c_{11}-c_{21}} \leq \rho = \frac{\pi^0}{1-\pi^0} \cdot \frac{c_{30}-c_{10}}{c_{11}-c_{31}} \leq \nu = \frac{\pi^0}{1-\pi^0} \cdot \frac{c_{30}-c_{20}}{c_{21}-c_{31}} \quad (3.18)$$

Additionally to the interestingness measures these thresholds can be used for quality awareness in association rule mining for a predictive selection of legitimately interesting rules based on the data quality scores.

3.4 Experiments on Quality-Aware Rule Mining

In order to evaluate our decision model (in both cases with and without misclassification), we built an experimental system. The system relies on a data generator that automatically generates data quality measures. This system also allows us to perform controlled studies so as to establish measures and variations for each data quality dimension computed both on data sets and discovered association rules.

In this section we present a set of experiments using the KDD-CUP-98 data set from the UCI repository[2]. The KDD-Cup-98 data set contains 191,779 records about individuals contacted in the 1997 mailing campaign. Each record is described by 479 non-target variables and two target variables indicating the "respond"/"not respond" classes and the actual donation in dollars. About 5% of records are "respond" records and the rest are "not respond" records. The KDD-Cup-98 competition task was to build a prediction model of the donation amount. The participants were contested on the sum of actual profit $\sum($*actual donation*$-\$0.68)$ over the validation records with predicted donation greater than the mailing cost \$0.68 (see (Wang et al., 2005) for details). Because we ignored the quality of the data collected during this campaign, we generated synthetic data quality measures.

In this experiment, our goal is to demonstrate that data quality variations may have a great impact on the significance of KDD-Cup-98 results (*i.e.*, the top ten discovered "respond" rules and profit predictions). Although data quality indicators do not affect the top ten list of discovered association rules, they significantly change the reliability (and the quality) of the mining result and also the cost of the decisions relying on these rules.

The names, definitions, generated quality indicators for four data quality dimensions (*i.e.*, freshness, accuracy, completeness, and consistency), average quality scores, and estimated probabilities per variable of the KDD-Cup-98 data set are given in Table 3.4. For the sake of simplicity, we generated the quality dimension scores such as they are uniformly representative of the quality dimension of the

[2]http://kdd.ics.uci.edu/databases/kddcup98/kddcup98.html for the data set and http://www.kdnuggets.com/meetings/kdd98/kdd-cup-98.html for the results

values in the attribute domain. The average quality $\overline{q}$ per variable in Table 3.4 is computed from the equi-weighted function given in Eq. 3.2.

f_{AC} in Table 3.4 (also noted $f_{AC}(\overline{q}(\mathcal{I}))$ in our formalism) is the probability density that the data set $\mathcal{I}$ has acceptable quality and correct data when the average quality score of the variable $\mathcal{I}$ is $\overline{q}(\mathcal{I})$. f_{UE} (also noted $f_{UE}(\overline{q}(\mathcal{I}))$) is the probability density that the data set $\mathcal{I}$ has unacceptable quality due to erroneous data when the average quality score of $\mathcal{I}$ is $\overline{q}(\mathcal{I})$.

Variable	Definition	Quality					f_{AC}	f_{UE}
		Fresh.	Accur.	Compl.	Cons.	$\overline{q}$		
AGE904	Average Age of Population	0.50	0.21	0.39	0.73	0.46	0.90	0.05
CHIL2	% Children Age 7 - 13	0.16	0.99	0.75	0.71	0.65	0.95	0.10
DMA	DMA Code	0.49	0.58	0.16	0.95	0.55	0.95	0.01
EIC16	% Employed in Public Administration	0.03	0.56	0.33	0.61	0.38	0.98	0.01
EIC4	% Employed in Manufacturing	0.17	0.37	0.87	0.15	0.39	0.90	0.20
ETH1	% White	0.21	0.76	0.50	0.53	0.50	0.55	0.15
ETH13	% Mexican	0.52	0.77	0.87	0.79	0.74	0.90	0.60
ETHC4	% Black $\leq$ Age 15	0.84	0.52	0.32	0.35	0.51	0.95	0.45
HC6	% Owner occupied structures built since 1970	0.47	0.96	0.74	0.11	0.57	0.98	0.03
HHD1	% Households w/ Related Children	0.61	0.95	0.27	0.08	0.48	0.96	0.41
HU3	% Occupied Housing Units	0.07	0.40	0.18	0.57	0.30	0.94	0.53
HUPA1	% Housing Units w/2 thru 9 at the address	0.76	0.85	0.96	0.93	0.88	0.95	0.52
HVP5	% Home Value $\geq$ $50.000	0.99	0.88	0.38	0.95	0.80	0.94	0.05
NUMCHLD	Number of children	0.44	0.23	0.53	0.50	0.42	0.96	0.17
POP903	Number of Households	0.77	0.52	0.74	0.61	0.66	0.87	0.15
RAMNT_22	Dollar amount of the gift for 95XK	0.37	0.95	0.95	0.75	0.76	0.84	0.25
RFA_11	Donor's RFA status as of 96X1 promotion date	0.59	0.34	0.34	0.76	0.51	0.95	0.12
RFA_14	Donor's RFA status as of 95NK promotion date	0.60	0.69	0.24	0.10	0.41	0.95	0.13
RFA_23	Donor's RFA status as of 94FS promotion date	0.34	0.01	0.23	0.63	0.30	0.97	0.55
RHP2	Average Number of Rooms per Housing Unit	0.66	0.72	0.08	0.26	0.43	0.98	0.20
TPE11	Mean Travel Time to Work in minutes	0.20	0.26	0.78	0.32	0.39	0.85	0.05
WEALTH2	Wealth Rating	0.24	0.82	0.41	0.58	0.51	0.87	0.05

Table 3.4: Quality Measures and Estimated Probabilities of Selected Attributes of the KDD-Cup-98 Data Set

The top ten a priori association rules discovered by (Wang et al., 2005) are given in Table 3.5 with the confidence, the support (in number of records), and the quality scores. Table 3.5 shows the score per quality dimension and the average quality score for each association rule. The scores are computed from the definitions of the quality dimensions given in Table 3.2 and the data quality scores previously given per attribute in Table 3.4.

Rule#	Association Rule	(Conf.; Supp.)	Quality				
			Fresh.	Accur.	Compl.	Cons.	$\overline{q}$
R1	ETHC4=[2.5,4.5], ETH1=[22.84,29.76], HC6=[60.91,68.53]	(0.11; 13)	0.21	0.38	0.79	0.53	0.48
R2	RFA_14=f1d, ETH1=[29.76,36.69]	(0.17; 8)	0.21	0.52	0.62	0.53	0.47
R3	HHD1=[24.33,28.91], EIC4=[33.72,37.36]	(0.12;12)	0.17	0.35	0.90	0.15	0.39
R4	RFA_23=s2g, ETH13=[27.34,31.23]	(0.12;16)	0.34	0.01	0.90	0.79	0.51
R5	EIC16=[11.25,13.12], CHIL2=[33,35.33], HC6=[45.69,53.30]	(0.16;11)	0.03	0.53	0.77	0.71	0.51
R6	RHP2=[36.72,40.45], AGE904=[42.2,44.9]	(0.16;7)	0.50	0.15	0.44	0.73	0.46
R7	HVP5=[56.07,63.23], ETH13=[31.23,35.61], RAMNT_22=[7.90,10.36]	(0.14;10)	0.37	0.65	0.68	0.95	0.66
R8	NUMCHLD=[2.5,3.25], HU3=[66.27,70.36]	(0.08;31)	0.07	0.09	0.61	0.57	0.34
R9	RFA_11=f1g, DMA=[743,766.8], POP903=[4088.208,4391.917], WEALTH2=[6.428571,7.714286]	(0.25;8)	0.24	0.08	0.72	0.95	0.50
R10	HUPA1=[41.81+,], TPE11=[27,64,31.58]	(0.23;9)	0.20	0.22	0.99	0.93	0.59

Table 3.5: The Top 10 "Respond" Rules with Confidence, Support and Quality Scores

In the next subsections, we study the impact of data quality variations on the decision cost of rule selection respectively in the two cases: with and without misclassification. We use the decision costs arbitrarly defined in Table 3.3 for classifying the rules based on the quality of their data.

3.4.1 Quality and Cost of Association Rules without Misclassification

First, we identify the value of the a priori probability that implies the largest amplitude of decision costs for rule selection based on Table 3.3 and Eq. 3.11 for the top ten rules discovered by (Wang et al., 2005). Figure 3.5 shows this case for the a priori probability $\pi^0 = 0.200$ in the absence of misclassification region (*i.e.*, $\pi^0_{AE} = \pi^0_{UC} = 0$).

By using Eq. 3.18, we compute the values of the three decision thresholds for rule selection with the a priori probability $\pi^0 = 0.200$. We obtain the following thresholds: $\lambda = 0.0131579$, $\rho = 0.125$, and $\nu = 2.25$. In order to be consistent with the conditional independency of the quality vector components we also need to take the logarithms of the thresholds values. By doing this we obtain:

$\log(\lambda) = -1.8808, \log(\rho) = -0.9031$ and $\log(\nu) = 0.3522$.

Based on the values for these thresholds, we can assign each rule to one of the three decision areas. Table 3.6 shows the profit per rule predicted by (Wang et al., 2005), the decision cost of rule selection computed from Table 3.3 and the decision area per rule.

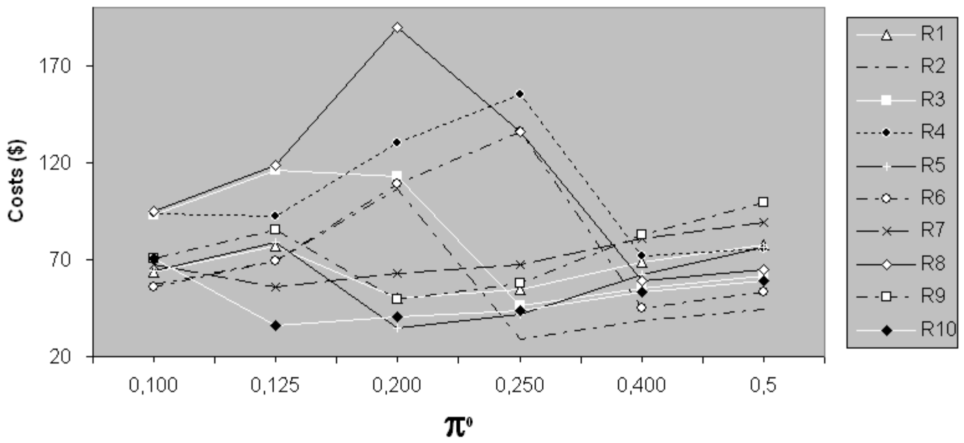


Figure 3.5: Decision Costs for Rule Selection with a Priori Probability in [0.1,0.5] without Misclassification

Rule#	Profit($)	Cost ($)	Decision Area
R1	81.11	53	potentially
R2	61.73	109.5	not
R3	47.07	113	not
R4	40.82	130	not
R5	35.17	34.7	potentially
R6	28.71	109	not
R7	24.32	62.8	potentially
R8	19.32	190	not
R9	17.59	49.6	potentially
R10	9.46	40.8	potentially

Table 3.6: The Top 10 "Respond" Rules with Profit, Cost, and Decision Area for $\pi^0 = 0.200$ without Misclassification

We observe that only 5 rules (*i.e.*, R1, R5, R7, R9, R10) are *potentially interesting* among the top ten rules considering the quality of data they are computed from. With data quality-awareness, the other rules (R2, R3, R4, R6, R8) are not interesting despite a good rank in the top ten list. It's also interesting to notice that the profit per rule predicted by (Wang et al., 2005) may be considerably counterbalanced by the cost of the rule computed from low-quality data (although it depends from initial costs defined in Table 3.3). The second best rule R2 whose predicted profit is $61.73 has a cost of $109.5 and thus is classified as *not interesting* due to the low quality of its data sets.

Let us now introduce different variations on the average quality of the data sets composing the rules. Based on the costs in Table 3.3, Figure 3.6 shows the behavior of the decision cost for rule selection when data quality varies from the initial average quality down to -10%, -30%, and -50% and up to +10%, +30% and +50% for the a priori probability $\pi^0 = 0.200$ without misclassification.

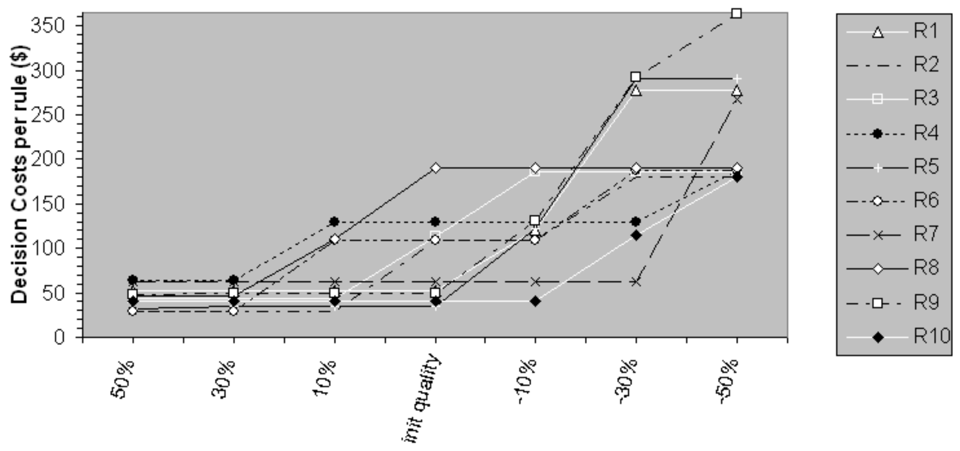


Figure 3.6: Decision Costs for Rule Selection with Different Data Quality Variations without Misclassification for the a Priori Probability $\pi^0 = 0.200$

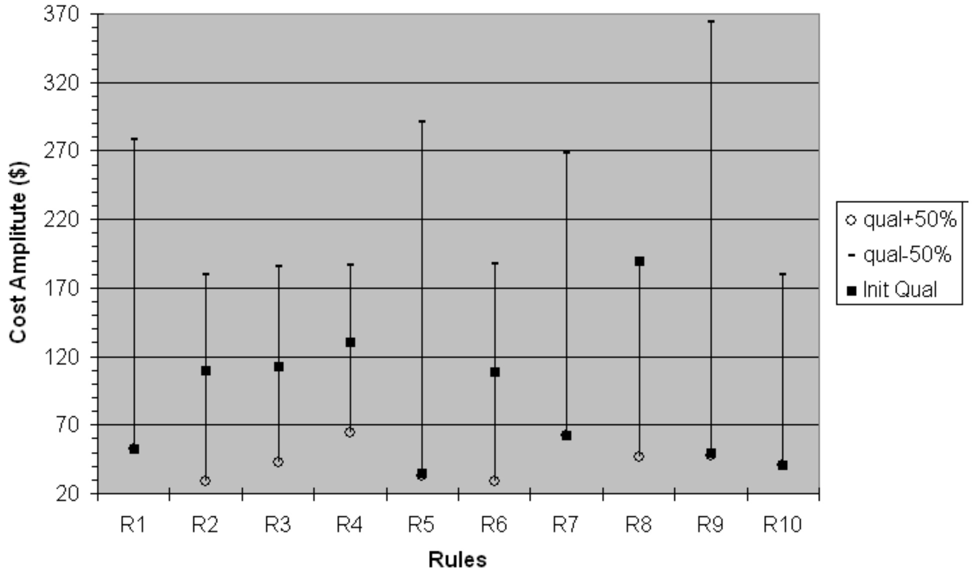


Figure 3.7: Amplitude of Cost Variations Depending on Data Quality Variations without Misclassification for the a priori Probability $\pi^0 = 0.200$

In Figure 3.6 we observe that the quality degradation of the data sets composing the rules increases the cost of these rules with various amplitudes shown in Figure 3.7 (with a maximal quality degradation noted qual-50% and a maximal quality amelioration noted qual+50%). Data quality amelioration implies a stabilization trend of the decision cost for *legitimately interesting* rule selection.

Another interesting result is shown in Figure 3.8 where the decisions for rule selection change simultaneously with the data quality variations. Among the top ten interesting rules discovered by (Wang et al., 2005) with the initial data quality (noted Init Qual), 5 rules (R1, R5, R7, R9 and R10) are potentially worth being selected based on their average data quality and 5 rules are *not interesting* (R2, R3, R4, R6 and R8). While increasing data quality up to +30%, 3 rules become *legitimately interesting* (R5, R7 and R9).

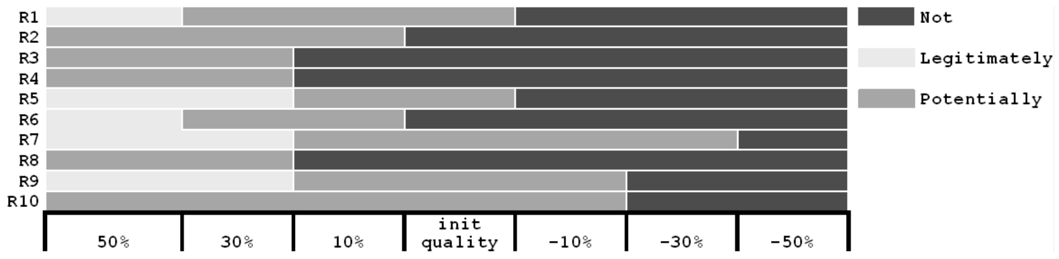


Figure 3.8: Decision Status on Rule Selection for Data Quality Variations without Misclassification for $\pi^0 = 0.200$

3.4.2 Quality and Cost of Association Rules with Misclassification

In the case of misclassification (with $f_{UC} = f_{AE} = f_{AC}$) we observe that the amplitude of the decision cost per rule depending on the a priori probability is reduced (see Figure 3.9) and the rule costs are stratified per rule.

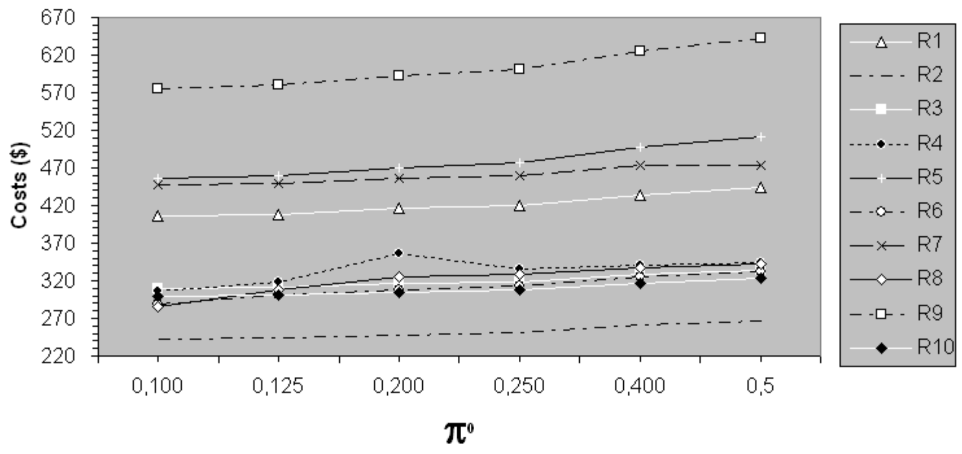


Figure 3.9: Decision Costs for Rule Selection with a Priori Probability in [0.1,0.5] with Misclassification

While keeping the a priori probability $\pi^0 = 0.200$ and using Eq. 3.13, 3.12 and 3.14, we compute the values of the three decision thresholds for rule selection with misclassification and we obtain: $\lambda = 0.1053$, $\rho = 0.1667$, and $\nu = 4.6667$. Based on the values for these thresholds, we can assign the rules to one of the three decision areas (see Table 3.6).

Rule#	Profit($)	Cost ($)	Decision Area
R1	81.11	415.70	potentially
R2	61.73	248.40	potentially
R3	47.07	315.90	not
R4	40.82	356.00	not
R5	35.17	469.80	potentially
R6	28.71	308.30	potentially
R7	24.32	455.80	legitimately
R8	19.32	325.00	not
R9	17.59	592.30	potentially
R10	9.46	305.10	potentially

Table 3.7: The Top 10 “Respond” Rules with Profit, Cost, and Decision Area for $\pi^0 = 0.200$ with Misclassification

In the case of misclassification with the a priori probability $\pi^0 = 0.200$ it’s interesting to notice that the cost per rule may be increased from 1.7 times to 13.5 times (respectively for R8 and for R5) compared to the case of correct classification. This is mainly due to the cost of: *i)* confident decisions for rule selection computed from low-quality data that are incorrectly classified, and *ii)* suspicious decision for rule selection computed from correct-quality that are incorrectly classified. With different variations on the average quality of the data sets composing the rules (from -10%, -30%, down to -50% and from +10%, +30% up to +50%) and based on the costs given in Table 3.3 in the case of misclassification, we study the behavior of the decision cost for rule selection.

Figure 3.10 shows that the costs are relatively stable with smaller amplitudes and more distinct and staggered cost ranges than in the case without misclassification (see Figures 3.10 and 3.11 compared to Figures 3.6 and 3.7) except at the maxima of data quality variations (*i.e.*, $\pm 50\%$) when the misclassification has more impact on decision costs.

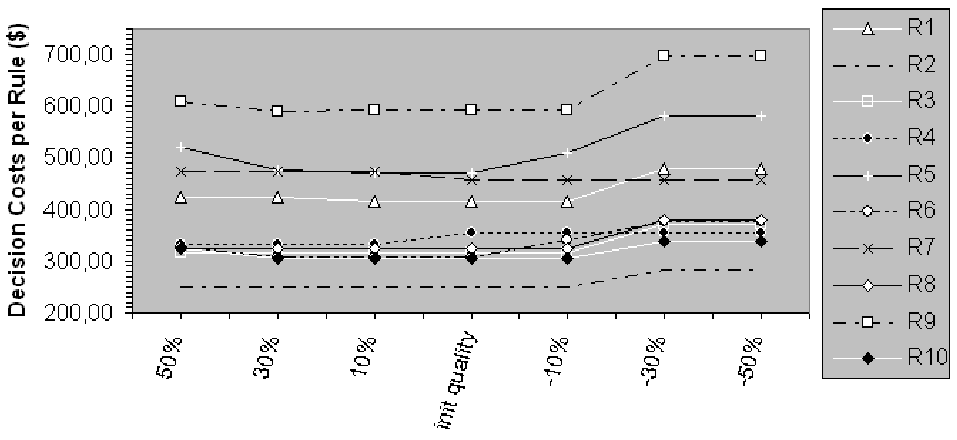


Figure 3.10: Decision Costs for Rule Selection with Different Data Quality Variations with Misclassification for the a Priori Probability $\pi^0 = 0.200$

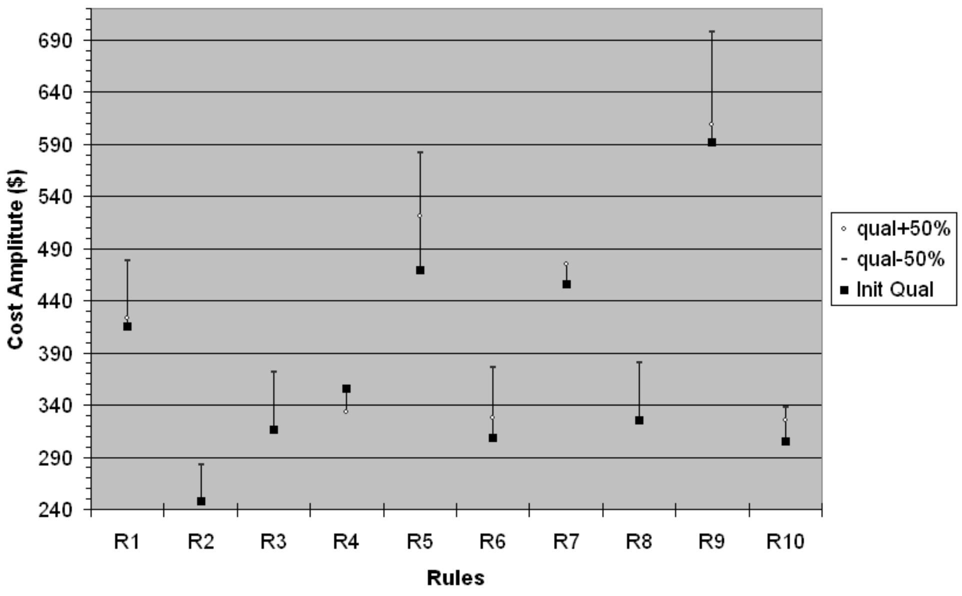


Figure 3.11: Amplitude of Cost Variations Depending on Data Quality Variations with Misclassification for the a Priori Probability $\pi^0 = 0.200$

In Figure 3.12 only R7 is *legitimately interesting* among the top ten rules discovered by (Wang et al., 2005) with the initial data quality (noted Init Qual). Three rules are not interesting (R8, R3 and R4) and the other 6 rules are potentially interesting. Misclassification globally attenuates the "verdict" that classifies each rule correspondingly to one of the decision areas for the *legitimately*, *potentially* or *not interesting* rules. Some rules (e.g., R8) keep the same behavior with or without misclassification when data quality varies.

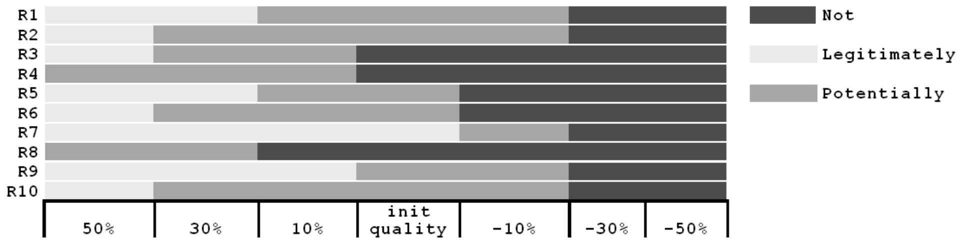


Figure 3.12: Decision Status on Rule Selection for Data Quality Variations with Misclassification for $\pi^0 = 0.200$

3.5 Conclusion

3.5.1 Summary

The quality of discovered association rules is commonly evaluated by interestingness measures (commonly support and confidence) with the purpose of supplying indicators to the user in the understanding and use of the new discovered knowledge. Low-quality data sets have a very bad impact over the quality of the discovered association rules, and one might legitimately wonder if a so-called "interesting" rule noted $LHS \rightarrow RHS$ is meaningful when 10% of the LHS data are not up-to-date anymore, 15% of the RHS data are not accurate, 10% of the LHS data are inconsistent, and there are 3% of approximate duplicates in the data set.

This chapter first presents a framework for integrating quality awareness in the knowledge discovery process. For the particular case of association rule mining, we propose to integrate data quality measures and estimate the cost of selecting interesting rules that are based on unacceptable data quality. In this study, our probabilistic model computes the cost of selecting legitimate versus non legitimate interesting rules. Experiments on the challenging KDD-CUP-98 data set show that variations on data quality may have an important impact on the cost and quality of discovered association rules. This confirms our approach and our argumentation in favor of the integrated management of data quality metadata into the KDD process for ensuring the quality of data mining results.

3.5.2 Research Perspectives

The observations made on this set of experiments offer two immediate interesting research perspectives for both association rule mining and data quality improvement:

- a first direction would be to propose a post-filtering rule process based on data quality metadata and optimal decision costs for rule selection,

- a second direction concerns the optimal scheduling of data quality improvement activities (with data cleaning techniques and ETL tools) in the data preparation KDD step. Cleaning tasks could relevantly be driven and target specific data sets depending on the cost and criticality requirements of mining results.

To make the correct decisions based on the knowledge discovery results that may be available in a timely manner, automatic means are needed to determine accurate data sources and to be able to detect malicious or compromised data sources to prevent them from influencing the decision making processes. Thus, mining techniques should be aware of completeness, accuracy, trustworthiness and interdependency of data sources for ensuring critical data and decisions. Using relevant QoD metadata constitutes a necessary step in KDD post-processing in order to improve and guarantee the quality of data mining results.

As a part of the validation of our approach on quality-aware rule mining, our future plans regarding this work are to study the optimality of our decision model and to propose error estimation.

As a mid-term objective, we'll extend and adapt this idea to another type of data mining techniques: the clustering methods. To achieve this objective, an in-depth and careful comparative study of the robustness of the clustering methods with respect to specific data quality problems (e.g, outliers, duplicates, and missing data) is required. As a prospective work, this study has been initiated by Ravi Jain (INRIA Rennes Post-Doc position since February, 2007).

Chapter 4

Prototyping Data Quality-Aware Applications

Contents

4.1 Introduction

This chapter illustrates how data quality awareness has been integrated in three domain-specific applications. For taking into account various dimensions of data quality relevant for these domains, several tools that have been implemented for three different types of multi-source information systems, namely, a data integration and warehousing system, a data mediation system, and a data stream monitoring prototype system.

The contributions presented in this chapter are parts of operational cases studies and ongoing developments that have been conducted thanks to several collaborations respectively with:

- the French public institute for biological, medical and public health research, INSERM (*Institut National de la Santé et de la Recherche Médicale*) Unit 522 in Rennes and in particular with Fouzia Moussouni (Associate Prof., University of Rennes 1) leading the GEDAW project dedicated to the integration and warehousing of biomedical data related to liver pathologies,
- the French Company of Electricity Supply, EDF R&D, for Customer Relationship Management (CRM) data,
- GenieLog, a French company managing, monitoring and mining stream Telecom data of Cegetel.

The development of prototypes have been achieved by several graduate and undergraduate students involved in each project and whose valuable contribution is worth being mentioned. After exposing problem statement and related work for each application domain, this chapter describes the contributions and main technical features of each project.

4.2 Quality-Aware Integration of Biomedical Data

4.2.1 Problem Statement

In life sciences, researchers extensively collaborate with each other, sharing biomedical and genomic data and their experimental results. This necessitates dynamically integrating different databases and warehousing them into a single repository. Overlapping data sources may be maintained in a controlled way, such as replication of data on different sites for load balancing or for security reasons. But uncontrolled overlaps are very frequent cases in available biomedical databanks.

Moreover, scientists need to know how reliable the data is if they are to base their research on it, because pursuing incorrect theories and experiments costs time and money. The current solution to ensure biomedical data quality is verification by human experts with the two main drawbacks that are: *i)* on-line data sources are autonomous and rapidly evolving; sources may provide excellent reliability for specific areas at a given time, but low-quality data on other topics, and data reliability may change over time and domain of interest, and *ii)* verification is a manual process of data accreditation by specialists that slows data integration; this process is also not free from conflicts of interest.

In such a competitive research domain, biological databank providers will not directly support data quality evaluations to the same degree since there is no equal motivation for them to do so, and there is currently no standard for evaluating and comparing biomedical data quality.

In the context of biomedical data integration, the major problems can be summarized as follows:

- *Lack of Data quality control.* Anyone is able to submit biological information to public on-line databanks with a more or less formalized submission protocols that usually do not include names standardization, inconsistency checking, and other data quality controls. Erroneous data may be easily entered and cross-referenced. The available data sources have overlapping scopes with different levels of data quality and trust. As we usually ignore the precise quality levels of each source, quality-driven data integration is perilous. As micro-array laboratories are scaling up their facilities, manual assessment of chip images becomes cumbersome and prone to subjective criteria. Automatic, impartial, and independent data quality evaluation methods and tools are needed for such a large diversity of biomedical data, ranging from structured (relational, object-oriented) or semi-structured (XML) to multimedia data types (text, image, video, audio).

- *Bio-entity resolution.* Even if some tools propose clustering techniques to gather records which possibly identify the same biological concept across different biological databanks for being semantically related, biologists still must validate the correctness of the clusters and resolve interpretation differences among the records. At the instance-level, these differences may be intrinsic and come from distinct (or conflicting) visions or knowledge states from one discipline to another in life sciences. This makes the process of verification and entity resolution very complex, usually involving several specialists searching for consensus.

- *Data mapping and transformation.* At the schema-level, the problem of format heterogeneity between publicly available databanks and "home-made" databases, data warehouses or laboratory information management systems (LIMS) obviously requires the translation and transformation of multi-source data, so that extracted data subscribe to the data model used by the biologist. Although the translation problem is inherent in all the data integration approaches, it becomes much more complex in the biological domain, again because different (and sometimes not formalized yet) biological interpretations are made based on rapidly evolving knowledge in various disciplines (e.g., involving the metabolic, chemical or functional views of a biological concept). This reinforces the perpetual evolution of any global schema for the system that is intended to integrate all the available information at a given time and for the biologist's focus of interest.

4.2.2 Related Work

In the context of biological data management systems, a survey of representative data integration systems is given in (Lacroix & Critchlow, 2003). Current solutions are mostly based on data warehouse architectures (e.g., GIMS[1], DataFoundry[2]) or a federation approach with physical or virtual integration of data sources that are based on the union of the local schemas which have to be transformed to a uniform schema (e.g., TAMBIS[3], P / FDM[4], DiscoveryLink[5]).

Very little work has been done on biological data cleaning and it is usually carried out in proprietary or *ad-hoc* manner, sometimes even manual. Systematic processes are lacking. From among the few examples proposed for the bio-entity resolution problem, Thangavel (1999) uses stringent selection criteria to select 310 complete and unique records of Homo Sapiens splice sites from the 4300 raw records in EMBL database[6]. Febrl (*Freely Extensible Biomedical Record Linkage*)[7] (Christen et al., 2004) allows data standardization, segmentation, and probabilistic and rules-based cleaning. Müller & Naumann (2003) examine the production process of genome data and identified common types of data errors. Mining for patterns of contradictions in overlapping databases has been proposed by Müller et al. (2004) for the cases where the entity identification problem has been already solved by specialists. But rigorous elimination of erroneous data or approximate duplicates may result in loss of rare information, at the first glance considered as dubious but actually critical for competing researchers.

More specific to data quality evaluation in the biomedical context, Martinez & Hammer (2005) propose a semi-structured model with quality measures that are biologically-relevant, objective (*i.e.*, with no ambiguous interpretation when assessing the value of the data quality measure), and easy to compute. Six criteria are defined and stored as quality metadata for each XML record of the genomic databank RefSeq[8]:

- *Stability*: magnitude of changes applied to a record in the databank
- *Density*: number of attributes and values describing a data item
- *Time since last update*
- *Redundancy*: fraction of redundant information contained in a data item and its sub-items

[1]GIMS, http: / / www.cs.man.ac.uk / img / gims /
[2]DataFoundry, http: / / www.llnl.gov / CASC / datafoundry /
[3]TAMBIS, http: / / imgproj.cs.man.ac.uk / tambis /
[4]P / FDM, http: / / www.csd.abdn.ac.uk / gjlk / mediator /
[5]DiscoveryLink, http: / / www.research.ibm.com / journal / sj / 402 / haas.html
[6]EMBL, European Molecular Biology Laboratory: http: / / www.embl-heidelberg.de /
[7]Febrl, http: / / datamining.anu.edu.au / software / febrl / febrldoc /
[8]NCBI References Sequences http: / / www.ncbi.nlm.nih.gov / RefSeq /

- *Correctness*: degree of confidence that the data represents true information
- *Usefulness*: utility of a data item defined as a function combining density, correctness, and redundancy.

The authors also propose algorithms for updating the scores of quality measures when navigating, inserting or updating / deleting a node in the record.

4.2.3 Contributions and Perspectives

4.2.3.1 Design of Quality-Aware Integration Process for Biomedical Data Warehousing

My work with INSERM U522 focuses on the problems of biological data integration in the project called GEDAW, *Gene Expression DAta Warehouse* initiated in 2000 and leaded by Fouzia Moussouni, Associate Professor of University of Rennes 1.

Since 2001, seven graduate students have contributed under my supervision to this project and its related research directions (Berti-Équille & Moussouni, 2005; Berti-Équille et al., 2001; Guérin et al., 2001; 2005).

Biologists and more specifically medical science and health researchers need to focus on specific data such as a given metabolism or pathology and to confront them to several experiment results on same genes and same focus of interest. In addition to data delivered by home experiments, they may wish to confront them to data issued by public transcriptome experiments, having a close related interest: same organ, same pathology, same specie, or whatever. Users' requirements are not exhaustive but do lead certainly to collect intensively data sets and knowledge on genes gathered from various on-line databanks, along with experiment results.

A selected part of this extremely rich and available knowledge on the expressed genes needs to be integrated before analyzing for optimization sake. In fact, to get the information, biologists of INSERM U522 spent a considerable time and effort to seek relevant information on the Internet. The challenge was how to automatically capture, organize and integrate data of interest along with capitalized biological knowledge. It is clear that to answer to all the issues posed by the biologist, the functionality of such environment had to include: data integration, management, and mining, data quality evaluation, and synthetic query result presentation.

Considering the different data quality and data integration issues previously mentioned, GEDAW system has been designed to support complex analysis on integrated data and tends to be a unified infrastructure for different bioinformatics technologies to measure gene expression of the genes involved in liver pathologies.

XML as an exchange format has been used for integrating data coming from multiple and semi-structured sources into a unified ontology-based data description model implemented in the object-oriented data warehousing system of GEDAW. The concept of ontology is the keystone of GEDAW system for integrating both genomic data available on public databanks, as well as experimental data on genes delivered from laboratory experiments and clinical statements.

GEDAW description model includes three major packages corresponding to the management and integration of three data domains that are: *i)* experimental data domain, *i.e.*, gene expression measurements through several physiopathological conditions, *ii)* genomic data domain, *i.e.*, DNA gene, mRNA, protein sequences and their annotations, and *iii)* biomedical knowledge, *i.e.*, biological and medical concepts that annotate the genes with concepts of ontologies like GO[9] and UMLS[10], using BioMeKE[11], the application developed at INSERM U522 by Marquet et al. (2003).

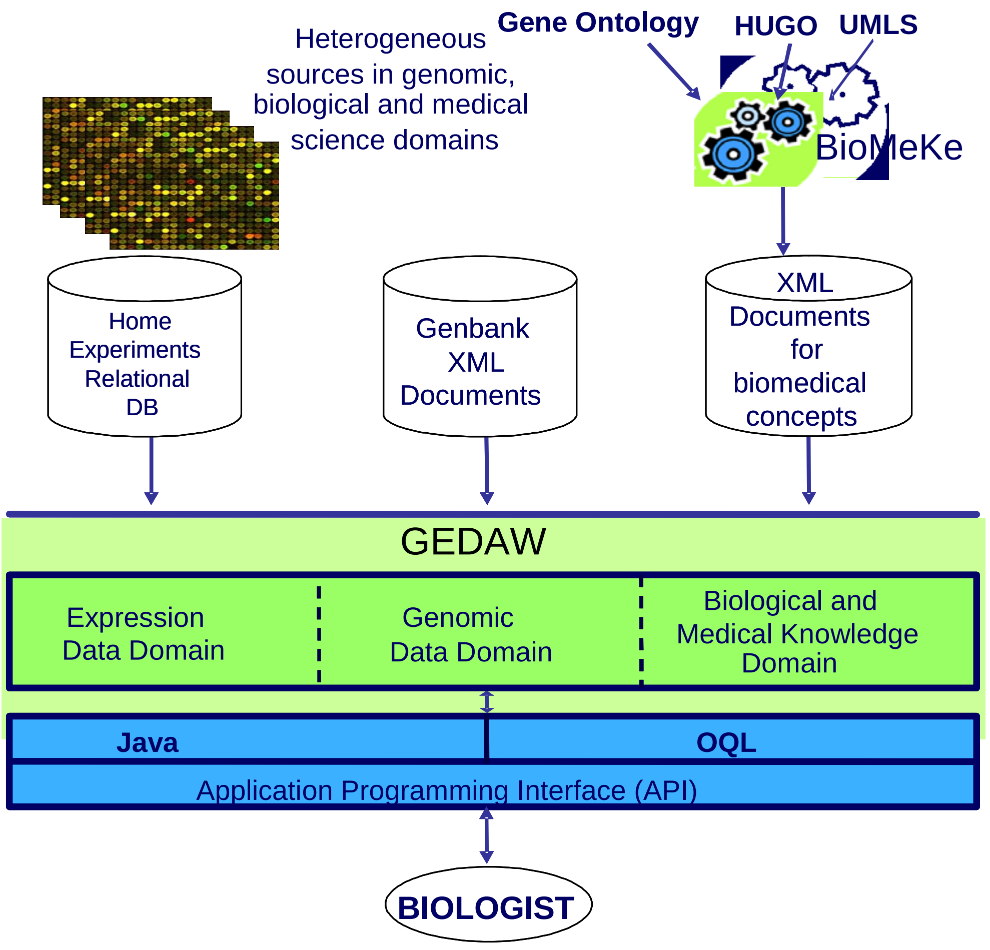

Heterogeneous sources in genomic, biological and medical science domains
Gene Ontology
HUGO
UMLS
BioMeKe
Home Experiments Relational DB
Genbank XML Documents
XML Documents for biomedical concepts
GEDAW
Expression Data Domain
Genomic Data Domain
Biological and Medical Knowledge Domain
Java
OQL
Application Programming Interface (API)
BIOLOGIST

Figure 4.1: GEDAW Architecture

As illustrated in Figure 4.1, the data sources used during integration are local or spread world wide and hosted on different systems, each having its own schema:

[9]Gene Ontology, GO: http://www.geneontology.org/
[10]Unified Medical Language System, UMLS: http://www.nlm.nih.gov/research/umls/
[11]Biological and Medical Knowledge Extraction system, BioMeKe: http://www.med.univ-rennes1.fr/biomeke/

1. An in-house relational database to store more details on micro-array experiments along with their results,

2. XML records from GenBank[12] have been used to instantiate the genomic domain of GEDAW,

3. Medical concepts from UMLS and genomic concepts from GO ontology are also delivered by BioMeKE (Marquet et al., 2003) for all the genes as an XML document and used to integrate the biomedical knowledge associated to the genes.

GEDAW system is an object-oriented data warehouse implemented using *FastObjects* (Versant) that provides a direct description of the conceptual model on genes and their expression with Java (or C++) as a binding language. Objects are made persistent into the object-oriented database, that is the central element of GEDAW. Data representation for browsing and user interface programming is quite intuitive.

We designed and developped the process for integrating genomic data in GEDAW data warehouse system. It is based on the four following steps:

1. *Coordinating*: The system internally searches for the data objects stored in the data warehouse whose description is incomplete, and finds the objects related to a gene whose attributes values are missing. Then, it accesses the GenBank databank to start the importation and integration process based on the accession numbers (identifiers of the records describing a gene in the databank). At this stage, several records that may refer the same gene may provide various alternative descriptions for completing the missing fields in the warehouse. Only the ones with matching characteristics and accession numbers have to be identified and selected with various heuristics we proposed.

2. *Importing*: Starting from the accession number of each gene involved in the transcriptome experiments (*i.e.*, that is present and expressed under certain experimental conditions on micro-arrays), the system imports the XML files from the GenBank databank into GEDAW data staging area.

3. *Cleaning*: The system parses the XML files, extracts the information that has to be integrated into the warehouse. The cleaning script uses the XML *QueryEngine* component to seek the relevant elements through XML documents to extract. The queries are formulated to declare the path to reach the relevant element, using filters and selection operators based on GenBank's DTD. The tags defined in GenBank's DTD have a recursive structure, therefore the depth of a same element may vary from one record to another. Redundancies and inconsistencies are removed from the temporary XML files which only keeps the elements to be loaded in the data warehouse.

[12]NCBI, National Center for Biotechnology Information: http://www.ncbi.nlm.nih.gov/

4. *Storage*: The system parses the temporary files of extracted and cleaned information and ensures the data storage and persistence into the object-oriented data warehouse of GEDAW.

4.2.3.2 Profiling Database and Exploring Biomedical Data Quality

One of the major problems of biomedical data integration comes from the heterogeneity and rapid evolution of database schemas: both data sources and data warehouse schemas need to adapt constantly to the growing and evolving knowledge of this domain.

Recent advances in biotechnology have produced a massive amount of raw biological data which are accumulating at an exponential rate. Errors, redundancy and discrepancies are prevalent in the raw data, and there is a serious need for systematic approaches towards biological database auditing, profiling, and cleaning.

As a first step in this direction, we have built *Q-DEX*, *Quality-Driven Database Exploration*, a plugged-in tool on top of the application layer of GEDAW data warehousing system. Q-DEX is a generic tool whose GUI allows the user to flexibly build query workflows on any database schema given the XMI data model.

Off-line Q-DEX fetches and analyzes the XMI database schema and dynamically generates and configure the interface for graphically querying the database. Once Q-DEX is connected to the database, user-defined graphical queries and profiling scenarios are processed on the database object instances. Generic functions that compute elementary QoD measures for completeness, freshness and consistency dimensions may be used to evaluate final or intermediate query results in the query scenarios.

Q-DEX provides an intuitive way to formulate queries and data quality constraints that follow the reasoning of biologists, assist them in the elaboration of their queries, and complete the obtained results by additional information describing the quality of data.

Using Q-DEX interface, biologists desgin and combine query workflows on any database.

By having an immediate glance on his intermediate or final query results that can be browsed on the *Q-DEX Result Viewer*, the user may modify and re-execute his queries when needed. He is also able to save a query workflow for ulterior reuse on different data, or export effective resulting data for an eventual use on external tools (clustering, spreadsheet, etc.) making it quite flexible and attractive for the biologists, as confirmed by our validation tests by users at INSERM U522.

Actually, using Q-DEX, much more possibilities are offered to the user to compose various query workflows on integrated data objects in GEDAW. The user can apply predefined functions (Level I and II as defined in Chapter 2) that compute generic QoD measures to get the quality of intermediate and final results of his query workflows. Other more specific functions may be flexibly added to the function library of Q-DEX.

The immediate perspectives of this project are threefold:

- to extend the QoD metadata management package of Q-DEX including more sophisticated analytical functions (Level III and IV) and a CWM-compliant metadata repository,

- to integrate query-driven loading and profiling activities so that query workflows for biomedical data integration could be designed through Q-DEX,
- to design on-line profiling techniques that first fetch selected data source schema and object data instances from accessible biomedical databanks, and profile them for driving data extraction and loading into GEDAW warehousing system.

4.3 Quality-Driven Query in Mediation Systems

4.3.1 Problem Statement

In classical mediation environments, data sources usually do not export information describing their quality of service (e.g., resource accessibility, query processing cost, reliability, etc.), nor information describing the quality of their content (e.g., data accuracy, availability, freshness, completeness, etc.). Different data sources may thus answer a global query with different response times, query costs and various levels of data quality. Because one user may accept a query result of lower quality (if it is cheaper or has a shorter response time than if the query cost is higher), it's necessary to consider both query cost and users' quality requirements, and make trade-offs between cost and quality when building the query result. One major difficulty is to adapt existing query processing techniques to environments where resource availability, allocation, query cost, and data quality may be not evaluated at compile time.

4.3.2 Related Work

The problem of designing multi-source information systems taking into account information about quality has been addressed by several approaches that propose techniques to select the data sources based on the metadata characterizing their content and quality, e.g., (Braumandl et al., 2001; Mihaila et al., 2000; Naumann, 2002; Naumann et al., 1999).

Among the projects that have been proposed for considering data quality in distributed query processing, *HiQIQ B&B (High Quality Branch and Bound Algorithm)* proposed by Naumann (2002) is a distributed query planning algorithm that enumerates query plans in such way that it finds the best N query plans after computing only a fraction of the total number of query plans. Upper quality bounds for partial query plans are constructed and thereby non-promising sub-plans are early pruned in the search tree.

In *ObjectGlobe* (Braumandl et al., 2001) the query processing follows a multi-step strategy. First, a lookup service locates data from each source that are relevant to the query by consulting a metadata repository. It also gathers statistical cost information. In the second step, the optimizer enumerates alternative query execution plans using a System-R-like dynamic algorithm and an optimal plan is built based on a cost model using the previously gathered information. In the last step, the query execution plan is distributed and executed using an iterator model. Users can specify quality constraints on the execution of their query. Constraints are defined on results (e.g., size of the result), cost (*i.e.*, how much the user is ready to pay), and time (e.g., time to first results). Quality of Service (QoS) management is introduced as part of the query processor. The quality constraints are treated in all the phases of querying processing. If they cannot be fulfilled, the query plan is dynamically adapted or the query is aborted. Based on that QoS concept, the optimizer's goal is to maximize the percentage of successful queries and abort any query that cannot fulfill its QoS constraints as soon as possible.

4.3.3 Contributions and Perspectives

In (Berti-Équille, 2001; 2003) we proposed a first version of XQuaL query processor and developed the main modules of a rudimentary mediation architecture as illustrated in Figure 4.2.

From 2001 to 2003, four undergraduate and six graduate students have contributed under my supervision to the development of the Java-based mediator and wrappers architecture in Java.

- **Before the query time.** From the application layer, the user is able to declare quality contract types and instances (defined in Chapter 2, Section 2.7) to the mediator that forwards the contract types et constraints specifications to the wrappers. Each wrapper has a metadata repository including a set of analytical functions to characterize several data quality dimensions. Wrappers apply the specifications defined with ON DATABASE, TABLE, COLUMN, ROW, CELL statement and check the contract declaration on their respective source, compute QoD measures, store them as QoD metadata in their local repository, and associate them to the specified DB object instances and granularity levels.

- **At the query time.** As illustrated in Figure 4.2, the user can submit a global quality-extended QWITH query to the XQUAL mediator. The mediator translates the global QWITH query into a local QWITH query with respect to the schema of the source that may be able to answer and sends the corresponding local QWITH query to the *Query Router*. When the *Query Router* receives a QWITH query, it invokes the *Quality Contract Manager* that checks the invoked contract types and constraints, and broadcasts the contracts definition. The broadcast method invokes wrappers to update locally their QoD metadata repository, checking constraints on the various quality dimensions and QoD measures declared in the quality contract types. Each wrapper creates a new *Controller* in order to execute the contract type. The next step is to get the registered wrappers by invoking the *Quality Contract Manager*. This method returns a set of wrappers able to answer the router queries. As the router also broadcasts the QWITH statement with the regular part of the query to all available wrappers, it waits for their response. Finally, the ones of selected wrappers that maximize the satisfaction degrre of quality constraints specified in the contract will be chosen to execute the SQL statement. Assume that the constraints are too strict and no wrapper can satisfy them, in this case the relaxation module will renegotiate the contract terms with selected wrappers; it will actually reduce the strictnes of the constraints defined in the contract of the QWITH query and broadcasts the modified part of the contract to the wrappers able to answer. The wrappers that conform the most to the modified contract will be selected to answer the query.

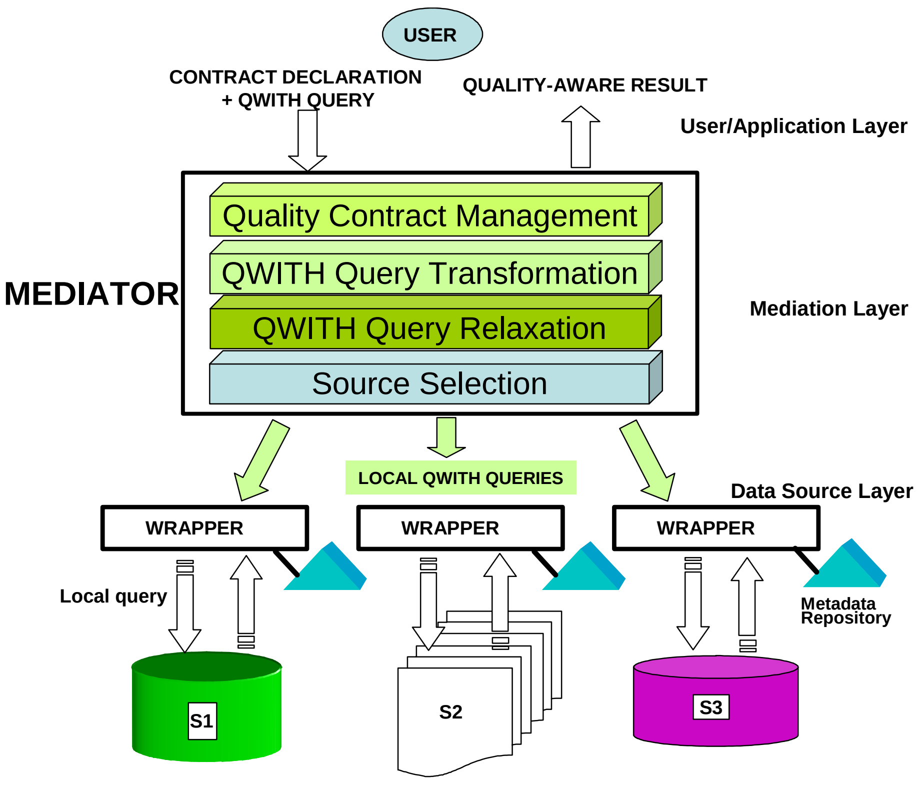


Figure 4.2: Mediation Architecture with XQuaL

For the technical validation of the prototype, several overlapping CRM relational databases have been generated and populated with synthetic data based on overlapping parts of the logical schema given in the TPC-W (V1.8) benchmark[13]. Although many technical aspects of this prototype still need to be fixed, it offers

[13]TPC Benchmark: http://www.tpc.org/

a first platform for implementing our proposals regarding QWITH query processing.

Among the numerous perspectives related to this project, our priorities mainly concern:

- the automatic generation of quality-aware wrappers including contract and QoD metadata management and

- the optimization and relaxation of QWITH queries at the mediator side.

Each of these priorities constitutes on its own a very challenging research direction to explore in the context of mediation architecture.

4.4 Monitoring the Quality of Stream Data

4.4.1 Problem Statement

In emerging data streams, data quality problems are likely to be manifold (Aggarwal, 2007). For example, in network databases, it is unrealistic to set integrity constraints to detect missing, duplicate, irregular or out-of-range values and stop processing a high speed data feed for each such violation and anomaly detection.

The continuous arrival of transient data in multiple, rapid time-varying, possibly unpredictable and unbounded streams appears to yield some fundamentally new research problems. Approximation and adaptivity are key alternatives to performance and storage stringent constraints in executing continuous queries and performing on-line data analysis. Backtracking over a data stream is not feasible. On-line stream processing or mining algorithms are restricted to making only one pass over data.

In many of data stream monitoring applications, each tuple of the data stream may be composed by data collected from many places, such as sensors, network equipments, etc. These data are collected through a wired or wireless network. Unfortunately, both the devices and the network transmission are vulnerable to errors. Hence, on-line data cleaning and active warehousing are indispensable. In these applications, user queries are typically continuous queries, which are evaluated in an on-line fashion. Hence, streams have to be cleaned on the fly, which requires that the anomaly detection and cleaning algorithms should be unblocking and can be pipelined.

For traditional passive data sets, data cleaning is an independent preprocessing phase, which can be performed off-line. Hence, specialized cleaning tools can be developed without interacting with the query processing engine. However, this is not true for stream processing where the cleaning tools have to interact with the query engine on the fly. In this context, human interactions are only possible at the phase of setting up the cleaning process. After the cleaning activities are set up, they should run automatically as the streams will continuously come into the system in a very high speed. Of course, one can revise the setup on the fly, but the cleaning should run automatically for most of the time.

Data stream anomaly detection and on-line cleaning may have large computational cost, which may impair the system's responsiveness to high speed data streams.

4.4.2 Prospective Work

In an operational scenario of a Telecom company, a centralized management system continuously collects stream data from a number of servers and equipments and it integrates them into a global data warehouse.

Malfunctions of the various devices and errors of transmissions happen frequently causing lots of problems in streamed data (e.g., missing, invalid values, duplicate records, inconsistencies, etc.)

Real-time monitoring of these data is compelling for a lot of applications, e.g. technical or administration management within the Telecom company, marketing, criminal and terrorism detection involving data from other companies across the world, etc.

In this context, a prospective work has recently been initiated in 2006 with GenieLog / Cegetel to review the related work on mining techniques and algorithms that are applicable to data streams for detecting anomalies and data quality problems, and inferring situations where they are likely to occur.

Recent results in (approximate) algorithms for on-line streaming include computing signatures and representative trends, decision trees (Domingos & Hulten, 2001; Hulten et al., 2001), k-medians clustering (Guha et al., 2001), regression or CPA analysis (Papadimitriou et al., 2005), similarity search (Faloutsos, 2002). On-line stream mining operators must be incrementally updatable without making multiple passes over the data. The objectives of the project are to study these techniques, evaluate their adaptability to the problem of stream data quality evaluation and monitoring with several criteria related to performance, scalability in realistic and operational scenarios, and recall / precision for anomaly detection.

Independently from this starting project, a prototype implementing an adaptation of XQuaL query language extension for continuous queries has been developed upon TelegraphCQ v0.2[14] by Yongluan Zhou, a PhD student from National University of Singapore, during his six months of internship at INRIA Rennes.

4.5 Conclusion

This chapter has presented the projects and prospective work in three application domains where several aspects of data quality awareness have been implemented. Main technical features of the prototypes that have been developed in response to realistic scenarios have been presented. Case studies are supported by several ongoing collaborations, respectively dedicated to: *i)* quality-aware warehousing

[14]TelegraphCQ is a open source stream processing engine based on PostgreSQL v7.3. It is available at: http: / / telegrap.cs.berkeley.edu / telegraphcq / v0.2 /

of biomedical data, *ii)* quality-aware mediation of CRM databases, and *iii)* quality monitoring for stream telecom data.

Involved in the design and software development of these prototypes, each one integrating particular aspects of data quality awareness in the managenent or integration of different types of data (relational, XML, object), I've observed many commonly shared features, good and bad practices in the design and development of such prototypes. As effective system design requires considering issues that may not become visible until later in the system / application implementation, usage or maintenance.

My observation (that is also an interesting R&D directions) is that patterns for integrating quality-awareness in data management applications need be clearly specified in order provide general solutions, documented in a format that doesn't require specifics tied to a particular problem.

Reusing QoD management patterns for IS design would considerably help to prevent data quality problems. The detailed study and specification of a set of *quality-aware design patterns* starts with describing the context in which the pattern is used, the problems within the context that the pattern seeks to resolve, the suggested solution and side-effects, and all the information necessary for its appropriate use. This constitutes a very challenging research direction.

Conclusions and Research Perspectives

Summary

Data quality is considered as a notoriously complex combination of problems. Taken globally it is very often viewed as technically intractable, even though some problems can be addressed individually by various methods from many disciplines: Databases, Statistics, Knowledge Discovery, and Knowledge Engineering.

Real-world databases inevitably have various anomalies and poor quality data such as incorrect, incomplete, inconsistent, redundant, and out-of-date data. In this context of “dirty” data management, the ultimate goal is ideally to create a general and reusable solution that can automatically scan and analyze the data sets, create a set of data quality metadata and constraints, isolate the records that do not meet this process, and trigger appropriate corrective actions on data.

At the end of the Information Supply Chain, databases contain a variety of patterns to discover, but few of them are of much interest. A pattern is interesting to the degree that is not only accurate but that is also useful with respect to the end-user’s knowledge and objectives. A critical issue in knowledge discovery is how well the database is designed, and maintained. Obviously, the effectiveness of knowledge discovery processes and the quality of the discovered knowledge are strongly dependent on the quality of data.

In the last decade, my contributions to Data Quality Research have been gradually oriented to the convergence point of the two following disciplines:

1. **Database and Information Systems Engineering**, with a special focus on the data integration processes and the design of multi-source information systems, in particular, data warehousing and virtual data mediation systems for integrating the management of quality metadata. The central questions of my work are the following: How to design the core of the data management system in order to integrate systematic quality measurement and control of integrated and stored data? How to ensure guarantees on the quality of query or mining results with minimal costs?

2. **Data Mining and Knowledge Discovery**, with a special focus on associa-

tion rule discovery and clustering techniques. On the one hand, because one of the main challenges of Data Quality Research is in measuring data quality factors and managing appropriately and efficiently these measures, my contribution is to propose analytic workflows for QoD evaluation that use and combine statistical and data mining techniques to compute measures characterizing the quality of data. On the other hand, the evaluation of the quality of data mining results that depend on the quality of analyzed data is the dual problem that needed to be considered in my approach. In this context, the central questions of my work are the following: How to adapt and use data mining techniques to implement data quality introspection and self-administration in data management systems? How to quantify the cost of low data quality on the results of data mining and knowledge discovery processes?

The contributions reported in this dissertation can be summarized as follows:

- *Management of metadata*: analytic workflows are designed for QoD evaluation; they generate metadata that characterize various aspects of data quality with computing statistical measures, summaries, and sketches obtained from exploratory data mining and statistical techniques,

- *Quality-aware query processing*: an extension of a query language for the declaration of factors, measures, and constraints on data quality dimensions has been proposed,

- *Quality-aware minig*: integration and exploitation of QoD metadata for the post-processing and validation of data mining results are proposed.

My dissertation has been organized as follows:

- **Chapter 1** introduces the field of Data Quality Research, reviewing literature with respect to each of the main quality dimensions, namely uniqueness, consistency, completeness, and freshness of data. This chapter has presented main measures, algorithms, languages, and tools that have been proposed in the literature for the main data quality problems. It has also presented recent research projects and new research directions in the field.

- **Chapter 2** presents our metamodel extension to CWM for modeling, computing measures characterizing several aspects of data quality. This chapter has proposed techniques for generating data quality metadata stored and indexed in a repository, and also a query language extension based on the declaration of quality contracts for querying data with user-defined constraints on quality metadata.

- **Chapter 3** proposes a generic framework for integrating data quality measures into the KDD process. In the context of association rule mining where result quality is classically evaluated by interestingness measures, our basic premise is that the quality of mining results also relies on the quality of the data which the discovered rules have been computed from. A cost-based probabilistic model for selecting legitimately interesting rules is presented. Experiments have shown that variations on data quality have a great impact on the cost and quality of discovered association rules. This confirms our approach for the integrated management and analysis of data quality metadata into the KDD processes.

- **Chapter 4** presents the issues, contributions, prospective works, and main technical features of prototype architectures developped in the context of three application domains for implementing data quality-awareness, respectively for: *i)* integration and warehousing biomedical data, *ii)* mediation of multi-source CRM data, and *iii)* monitoring stream Telecom data.

Our next research directions concern the following topics, as described in the next section:

1. *Quality-aware query processing.* General and particular challenges will be exposed.
2. *Statistical metadata computation*
3. *QoD-introspective data management system*
4. *KDD post-processing.*

Perspectives on Quality-Aware Query Processing for MSIS

General Challenges

Providing efficient access to information sources received a sustained interest since several decades but relatively few approaches have been proposed to deal with the various issues of quality-aware query processing in distributed environments. These issues are particularly challenging due the characteristics of the sources, including autonomy, volatility, amounts of data, large heterogeneity spectrum, e.g., on data type (multimedia, XML, relational records, etc.), database schema, and data quality. In this context, one of the challenge we've identified is to build a quality-aware query processing infrastructure for multi-source information systems. Building such an infrastructure requires addressing several interesting research issues and investigating in the following directions:

- *QoD- and QoS-based query languages.* The idea is to devise a declarative data query language that operates on data with quality of data and quality of service (QoS) constraints. The advantage is that the same quality-constrained query specification holds whatever underlying information is available.

- *Computation model.* The resolution of any quality-extended query may involve an iterative process between the different distributed systems within the infrastructure. We need to devise a computation model for the interaction of the different (sub-)systems (e.g., wrapper / mediator systems, sources / data warehouse, peers, Web portals / Web services / Web providers, etc.).

- *Optimization model.* Performance has a prime importance in successfully deploying a quality-aware query processing infrastructure over distributed systems. It mainly relates to query optimization. One challenge is to define appropriate metrics to characterize and measure QoD and QoS dimensions depending on the application domain, the systems capabilities, and the limited resources / performances. The different query planning strategies focus generally on finding feasible and optimal sub-goal orderings based on available bindings and supported conditions at the information sources. Proposed techniques assume a full knowledge of the query capabilities of every source participating in a multi-source information system. They rely heavily on the way that information sources are described and the objective function of the optimizer (e.g., number of sources, response time, etc.). Using the same source description and data quality description models may not always be possible across a large spectrum of information sources.

- *Optimization heuristics.* General randomized search heuristics are a popular tool for query optimization. Other kinds of algorithms such as randomized hill-climbing, Tabu search (that uses memory to guide the search towards optimal / near-optimal solutions, by dynamically managing a list of forbidden moves) or genetic algorithms (that emulate the evolutionary behaviour of biological systems to create subsequent generations) could be used or adapted for quality-aware query optimization. In most of the real world applications, it is quite natural that quality-extended query should meet a number of different and conflicting quality dimensions. Optimizing a particular objective function may sacrifice optimization of another dependent and conflicting objective. An interesting direction is the study the quality-extended query processing problem from the perspective of multi-objective optimization.

- *Quality-aware adaptive query processing.* Another interesting trend is the use of adaptive or dynamic approaches in dealing with quality-aware query optimization. This is motivated by the intrinsic dynamics of the distributed and autonomous sources where unpredictable events may occur during the execution of a query. The types of actions that are proposed in these approaches fall into one of the following cases: *i)* change the query execution plans in order to privilege data quality of query results, *ii)* change the scheduling of operations in the same query execution plan or in different concurrent query plans, *iii)* introduce new operators to cater for the unpredictable events (e.g., QoD or QoS degradation), or *iv)* modify the order of inputs of binary operators. Adaptive techniques have yet to demonstrate their applicability to various real applications with large numbers of information sources. There is also a need to show how they react under heavy quality of data and quality of service fluctuations. To the best of our knowledge, the issues of data quality-awareness in on-line query processing have not been much investigated and constitutes a very challenging perspective of research.

Particular Challenges

Relational query optimization has traditionally relied upon table cardinalities when estimating the cost of query plans they consider. While this approach has been and continues to be successful, the need to consider the various dimensions of data quality for query execution requires a particular approach. The dual problem is to fix the query cost and search for the "best quality" result, or to fix the result quality and optimize the query cost. Data quality awareness when querying a single or several distributed data sources in a dynamic and distributed environment raises several interesting questions such as:

- *Selecting dynamically the adequate data source.* Different data sources may answer a global query with different response times, query costs and various levels of data quality. How to define strategies for selecting adaptively the most appropriate sources for answering a query with the "most acceptable" data quality?

- *Defining semantically and qualitatively correct distributed query plans.* The result of a global query is classically built depending on the particular order for the execution of subquery plans. For ensuring data quality awareness, this technique must combine in a coherent way both information and meta-information from the various data sources (*i.e.*, data quality metadata if available). Data quality levels are often unknown, heterogeneous from one source to another, more or less aggregated or locally non uniform (*i.e.*, a source may provide excellent data reliability for one specific area, data subset or data type but not for the others). In this context, one of the problems is to merge the data quality indicators in a consistent way.

- *Making trade-offs between the cost of the query and the measurable quality of the result.* Because one may accept a query result of lower quality (if it is cheaper or has a shorter response time than if the query cost is higher), it's necessary to adapt the query cost to users' quality requirements. The objective is to measure and optimally reduce the cost and bargain query situations where the system searches for solutions that "squeeze out" more gains (in terms of data quality of the query result) than the query without data quality constraints.

- *Developing quality-aware query cost models.* It is important to evaluate whether the expected benefits from a quality-extended query compensate for the cost of computing or predicting quality measures and collecting feedbacks from the source and the environment during execution time. The difficulty is to adapt existing query processing techniques to environments where resource availability, allocation, query cost and data quality may be not decidable at compile time.

Algorithmic Challenges for Statistical Metadata Computation

Exploratory data mining summaries are typically averages, standard deviations, medians or other quantiles on several data sets or samples. They characterize the distribution of attribute values or describe value dispersion (form, density, symmetry, etc.). But as user-defined functions, the computation cost of these method that are used to characterize certain dimensions of the quality of data sets has to be precisely studied in order to choose adequately:

- the starting time when it is the most convenient to start the execution of the computation method,
- the type and properties of the computation methods (*i.e.*, precomputation, on the fly, on-demand, incremental) that is preferable depending on the application, data set size, CPU cost, the semantics, properties of the measurement, and characteristics of implementation methods,
- the relevancy of the QoD measures with respect to a quality goal that is either to observe, control or detect anomalies in the data set.
- the techniques used to reduce the volume of data to analyze (pre-selection or sampling) and the size of the data set targeted by the computation method. Because it is not realistic to compute all the methods on the whole data set, it is relevant to propose various strategies for data subset selection, sampling, and generalization depending on the semantics of the measurement and on the properties of the computation method. The final objective is to propose a cost model for executing analytic worflows.

The goal of this study is to define parameters: *i)* to trigger the most appropriate techniques for efficient QoD measure computation, such as loading anticipation or *prefetching* of the data necessary to precomputation, *ii)* to drive index selection strategies on indexed data and metadata, and *iii)* to propose recomputation and refreshment strategies for existing QoD measures.

Derived from these considerations, other algorithmic challenges can be identified as:

- to study data and metadata indexing algorithmics: with metadata computation, the index may become much bigger than the data volume and it is very largely higher than the integration capacities of the RAM. The algorithms of indexing must then take into account the various levels of memory hierarchy (register, masks, discs, RAM, etc.) or be distributed, or even, the index memory must be distributed.

- to reduce the volume of data to be analyzed for detecting data anomalies as quickly and effectively as possible.

- to study the decidability and the complexity of the approaches combining the detection of several types of anomalies: programs for the detection of duplicates, outliers, missing values, etc., and those for the cleaning or the refreshment of data have a great number of configurations (values of the parameters and variables, etc.). This raises theoretical problems (indecidability in a general way), and practical as well (size and complexity of the models and their sets of possible configurations).

Perspectives for QoD-Introspective Data Management Systems

A mid-term objective of my research project consists of proposing and designing a *quality introspective data management system* monitoring data quality by exploratory data mining techniques on very large volumes of data (*i.e.,* for several hundreds of Terabytes and billions of records).

The idea to use data mining techniques to extract the knowledge useful for database system administration is a very promising approach to reduce the tasks of database maintenance, administration, and tuning. Self-administration systems aim at managing and adapting themselves automatically without performance loss or even with performance benefit. My next research directions are motivated by the idea to extend the auto-administration primitives of data management systems with the management and control of data quality, and to develop the concept of **introspective systems**, as well as the techniques for their implementation.

For a taking into account “natively” data quality awareness, it is necessary to reconsider totally the traditional architectures of data management systems and, in particular, to modify the core of the database management system. This requires the modification of the query processor and the data storage subsystem in order to maintain (and recover) indexed structures of metadata that characterize the quality of data (e.g., with sketches and summaries, results of statistical analyses or probabilistic constraints), and ensure frequent refreshment.

One of the objective is to recommend strategies for the configuration and selection of data and QoD metadata indexes and the materialization of quality-constrained views in order to optimize the access time to data and QoD metadata.

Selection of data / metadata indexes and views can be very useful for the efficient sharing of the disk space allotted to these structures’ storage. In this area, my future work intends to propose indexing techniques that precompute quality-aware joins and adapt various selection strategies for quality-constrained view materialization.

Perspectives for Knowledge Discovery Post-Processing

The interest about knowledge post-processing has grown as a research topic in recent years. This is motivated mainly by the intensification of extracted knowledge uses in practical applications, and for the great volume of this knowledge that makes impracticable its manual analysis and validation. When extracting patterns with association rule discovery for example, the considerable amount of discovered rules raises an important problem of expressivity and result quality evaluation. The use and combination of objective measures in the post-processing step of rule evaluation has as a main purpose to establish filters for rule selection and validation. These measures supply an indication of the hypothetical strength association between the two parts of the rules, and also may prioritize the analysis of the rules set with higher contribution to the measures as they are considered for having higher potential to be interesting.

In this context, interestingness measures are not self-sufficient for evaluating mining result, and the joint analysis and exploration of metadata that characterize the quality of data which the rules are computed from, offers interesting perspectives for KDD post-processing. Metadata exploration could indeed be advantageously used for each of the main categories of post-processing methods used in data mining, as:

- *Knowledge filtering*: when data is noisy, the mining algorithm "overgenerates" leaves of the resulting decision tree or decision rules or association rules so that they also cover a very small number of noisy objects. To overcome this problem a tree or a decision set of rules must be shrunk, by either post-pruning (decision trees) or truncation (decision rules). Similar approach is adopted for association rules filtering with refining interestingness measures thresholds. Nevertheless, this process could be driven with a bottom-up approach, by analyzing the metadata that characterize several quality dimensions of the data set. Knowledge that is discovered from low-quality data could be filtered.

- *Interpretation and explanation*. In the cases where the acquired knowledge is directly used for prediction or where the knowledge discovery process is performed for an end-user, the derived results must be precisely documented and easily understandable through the appropriate visualization of results. For decision-making, end-users also need to know the quality of the data which the knowledge has been extracted from and possibly reconsider the reliability of results and estimate the risk of their decision if it is based on results that are discovered from low or unacceptable quality data.

- *Evaluation*. There are several criteria for evaluating the discovered knowledge: interestingness measures for discovered association rules, classification accuracy, comprehensibility for learning systems that induce concept hypotheses (or models), computational complexity, etc. The evaluation of several dimensions (as freshness, consistency, or accuracy) of th quality of data which the knowledge has been discovered from has been largely ignored from data analysis, because analysts seem to have confidence in data preparation step purging every "visible" data problem. Actually, metadata provide additional information (e.g., describing data freshness or detecting approximate duplicates) that may be relevant to capture the "dark side" of data sets. Consequently, they should be considered and appropriately explored as new criteria for knowledge evaluation.

Annexes

Distance	Definition and Main Characteristics
Hamming distance	Number of bits which need to be changed to turn one string s_1 into another string s_2. This can be extended into a vector space approach where the terms within a string are compared, counting the number of terms in the same positions. This approach is only suitable for exact length comparisons (e.g. SSN, Zip Code)
Levenshtein distance	Basic edit distance function whereby the distance is given simply as the minimum edit distance which transforms string1 into string2. Edit Operations are listed as follows: copy character from string s_1 over to string s_2 (cost 0), delete a character in s_1 (cost 1), insert a character in s_2 (cost 1), substitute one character for another (cost 1). $d(i, j)$ is a function whereby $d(c_1, c_2) = 0$ if $c_1 = c_2$, 1 else. $D(i,j) = \begin{cases} D(i-1,j-1)+d(s_i,t_j) & subst/copy \\ \min(D(i-1,j)+1) & insert \\ D(i,j-1)+1 & delete \end{cases}$
Needleman-Wunch distance Sellers Algorithm	Similar to the Levenshtein distance, this adds an variable cost adjustment to the cost of a gap noted G, *i.e.* insert/ deletion, in the distance metric. $D(i,j) = \begin{cases} D(i-1,j-1)+d(s_i,t_j) & subst/copy \\ \min(D(i-1,j)+G) & insert \\ D(i,j-1)+G & delete \end{cases}$ $d(c, d)$ is an arbitrary distance function on characters (e.g., related to typographic frequencies, amino acid substitutability).
Smith-Waterman distance	Similar to the to Levenshtein distance, this was developed to identify optimal alignments between related DNA and protein sequences. This has two main adjustable parameters a function for an alphabet mapping to cost values for substitutions and costs, (the d function). This also allows costs to be attributed to a gap G, (insert or delete). $D(i,j) = \begin{cases} 0 & startover \\ D(i-1,j-1)-d(s_i,t_j) & subst/copy \\ \max(D(i-1,j)-G) & insert \\ D(i,j-1)-G & delete \end{cases}$
Monge Elkan distance (Monge & Elkan, 1996)	Recursive field matching algorithm: each subfield A is evaluated against the most similar subfield B in the comparison string using the Gotoh distance between the fields this is combined, as: $match(A,B) = \frac{1}{\|A\|}\sum_{i=1}^{\|A\|}\max_{j=1}^{\|B\|} match(A_i, B_j)$
Jaro distance metric (Jaro, 1989; 1995)	This takes into account typical spelling deviations: for two strings s and t, let s' be the characters in s that are "common with" t, and let t' be the characters in t that are "common with" s; a character a in s is "in common" with t if the same character a appears in about the place in t. Let $T_{s',t'}$ be the measure of the number of transpositions of characters in s' relative to t'. The Jaro similarity metric for s and t is: $Jaro(s,t) = \frac{1}{3}\cdot(\frac{\|s'\|}{\|s\|}\frac{\|t'\|}{\|t\|}+\frac{\|s'\|-T_{s',t'}}{2\|s'\|})$
Jaro-Winkler distance (Winkler, 1999)	Extension of the Jaro distance metric that modifies the weights of poorly matching pairs s, t that share a common prefix, as: $JW(s,t) = Jaro(s,t) + (pLength * pScale * (1.0f - Jaro(s,t)))$ where $pLength$ is the length of common prefix at the start of the string, $pScale$ is a constant scaling factor for how much the score is adjusted upwards for having common prefix's.
Soundex distance	Term-based conversion where each phoneme is given a Soundex code. A Soundex code consists of the first consonant of the term, followed by three digits based on the consonants as in the following enumeration: 1: B,P,F,V - 2: C,S,K,G,J,Q,X,Z - 3: D,T - 4: L - 5: M,N - 6: R. The vowels are not used. If there are not three digits after the consonants are convert, the code is filled out with zeros (e.g., the code of "John" and "Jon" is J500).
Matching Coefficient	Token-based vector space similarity measure which counts the number of terms (dimensions), on which both vectors are non zero. For vector set X and set Y the matching coefficient is $\|X \cap Y\|$. This can be seen as the vector based count of co-referent terms.
Dice's Coefficient	Term-based similarity measure (0-1) whereby the similarity measure is defined as twice the number of terms common to compared entities divided by the total number of terms in both tested entities. The coefficient result of 1 indicates identical vectors as where a 0 equals orthogonal vectors. CT is the number of common terms, $N1$ and $N2$ the numbers of terms respectively in strings s_1 and s_2: $DicesCoef = (2 * CT)/(N1 + N2)$
q-gram (Gravano et al., 2001)	Approximate string matching using "sliding" a window of length q over the characters of a string to create a number of q length grams for matching. A match is then rated as number of q-gram matches within the second string over possible q-grams. The positional q-grams of length q=3 for string "john-doe" are f(1,##j), (2,#jo), (3,joh), (4,ohn), (5,hn-), (6, n-d), (7,-do), (8,doe), (9,oe$), (10,e$$), where '#' and '$' indicate the beginning and end of the string.

Table A.1: Main Characteristics of Distance Functions

Distance	Main Characteristics
Block distance or L1 distance or Manhattan distance	First order Minkowski distance between two real vector x and y, $x, y \in \Re^n$: $L_1(x,y) = \sum_{i=1}^{n} \|(x_i - y_i)\|$
Euclidean distance or L2 distance	Geometric distance between two real vector x and y, $x, y \in \Re^n : L_2(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$
Weighted Euclidean distance	Euclidean distance weighted by W: $[(x-y)W(x-y)^T]^{1/2}$
Minkowski distance	Generalized Euclidean distance of order p where $p \geq 1 : L_p(x,y) = [\sum_{i=1}^{n} \|x_i - y_i\|^p]^{1/p}$
l_∞ distance	Generalized Euclidean distance:$L_\infty(x,y) = \max_i \|x_i - y_i\|$
Mahalanobis distance	Weighted Euclidean distance with weight equal to inverse covariance matrix Cov: $L_M(x,y) = [(x-y)^T Cov^{-1}(x-y)]^{1/2}$
Hausdorff distance	Distance between two sets of points or clusters, P and Q in $\Re^n$: $H(P,Q) = \max(h(P,Q), h(Q,P))$ where $h(P,Q) = \max_{p\in P} \min_{p\in Q} \\|p - q\\|$ is the forward distance, and $h(Q,P) = \max_{q\in Q} \min_{p\in P} \\|p - q\\|$ is the backward distance.
Bhattacharyya distance (Basu et al., 1997) or Hellinger distance	Generalization of Mahalanobis distance between two sets of points or clusters, P and Q in $\Re^n$ calculated from the means (μ_p and μ_q) and covariance matrices, Cov_P and Cov_Q of the two clusters: $\frac{1}{8}(\mu_p - \mu_q)^T\{(Cov_P + Cov_Q)/2\}^{-1}(\mu_p - \mu_q) + \frac{1}{2}\ln\frac{\|Cov_P + Cov_Q\|/2}{\sqrt{\|Cov_P\|\|Cov_Q\|}}$
Chernoff distance	Generalization of Mahalanobis distance, where $0 < \sigma < 1$ allows unequal covariance matrices R_x and R_y: $\sigma\frac{1-\sigma}{2}(x-y)^T[\sigma R_x + (1-\sigma)R_y]^{-1}(x-y) + \frac{1}{2}\ln\frac{\|\sigma R_x + (1-\sigma)R_y\|}{\|R_x\|^\sigma \|R_y\|^{1-\sigma}}$
Jaccard Similarity or Jaccard Coefficient or Tanimoto coefficient	Token-based vector space similarity measure that uses word sets from the comparison instances to evaluate similarity. Each instance is represented as a Jaccard vector similarity function. The Jaccard similarity between two vectors X and Y is: $Jaccard(X,Y) = (X*Y)/(\|X\|\|Y\| - (X*Y))$ where $(X*Y)$ is the inner product of X and Y, and $\|X\| = (X*X)^{1/2}$, *i.e.*, the Euclidean norm of X. This can more easily be described as $(\|X \cap Y\|)/(\|X \cup Y\|)$
Overlap Coefficient	This is a measure whereby if a set X is a subset of Y or the converse then the similarity coefficient is a full match. This is similar to Dice coefficient. Overlap coefficient is defined as so: $Overlap(X,Y) = (\|X \cup Y\|)/\min\{\|X\|, \|Y\|\}$
Kullback-Leibler divergence or information gain or relative entropy	Natural distance measure from a "true" probability distribution P to an arbitrary probability distribution Q. Typically P represents data, observations, or a precise calculated probability distribution. $KL(P\|\|Q) = \sum_i P(i) \log\frac{P(i)}{Q(i)}$
Smith-Waterman-Gotoh distance	Comparing two sequences $A = (a_1a_2a_3\cdots a_n)$ and $B = (b_1b_2b_3\cdots b_m)$ $D_{ij} = \max[D_{i-1,j-1} + d(a_i,b_j), \max[D_{i-k,j} - W_k], \max[D_{i,j-l} - W_l]]$ D_{ij} is the maximum similarity of two segments ending in a_i and b_j respectively.
Gotoh Distance or	Extension from the Smith Waterman distance by allowing affine gaps within the sequence. The Affine Gap model includes variable gap costs typically based upon the length of the gap l.
Cosine similarity	Token-based vector space similarity measure similar to Dice coefficient, whereby the input string is transformed into vector space so that the Euclidean cosine rule can be used to determine similarity between two vectors q and r, as: $cos(q,r) = \sum_y q(y)r(y)/\sqrt{\sum_y \sqrt{q(y)^2}\sum_y \sqrt{r(y)^2}}$.
Variational distance	Measure used to quantify the difference (or divergence) between probability distributions. This is a form of the Kullback-Leibler divergence, as: $V(P\|\|Q) = \sum_{i=1}^{n} \|p_i - q_i\|$.
Information Radius or Jensen-Shannon divergence	Measure is used to quantify the difference between two (or more) probability distributions, as: $JS(q,r) = \frac{1}{2}\big[D\big(q\|\|avg(q,r)\big) + D\big(r\|\|avg(q,r)\big)\big]$.
Harmonic Mean	Measure used to quantify the difference (sometimes called divergence) between probability distributions. This could more accurately be called the Harmonic Geometric Mean: $M(P\|\|Q) = \sum_{i=1}^{n}\frac{2p_iq_i}{p_i+q_i}$
Skew divergence (Lee, 2001)	Extended from the Kullback-Leibler divergence, the skew divergence is defined as: $s_\alpha(q,r) = D(r\|\|\alpha q + (1-\alpha)r)$.
Confusion Probability	Estimates the substitutability of two given strings, is again an approximation of the Kullback-Leibler divergence, as: $conf(q,r,P(x')) = P(x')\sum_y q(y)r(y)/P(y)$
Tau	Approximation of the Kullback-Leibler divergence, as: $\tau(q,r) = \sum_{y_1,y_2} sign[(q(y_1) - q(y_2))(r(y_1) - r(y_2))]/(\frac{\|V\|}{2})$

Table A.1: (continued) Main Characteristics of Distance Functions

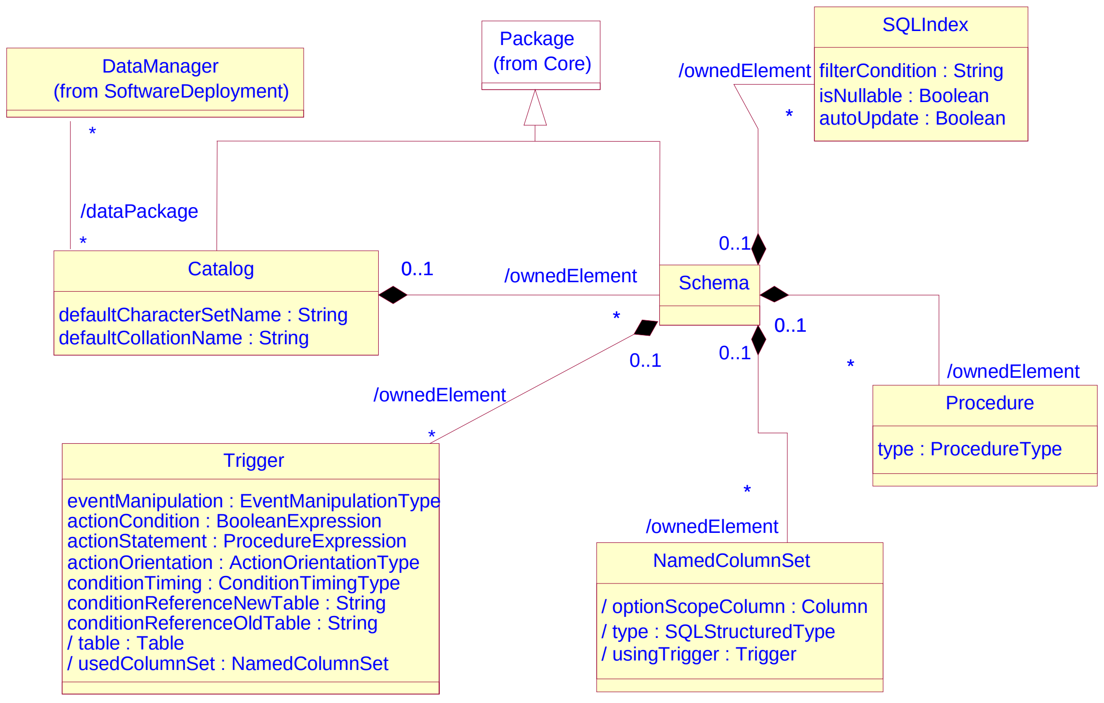

DataManager
(from SoftwareDeployment)
Package
(from Core)
SQLIndex
filterCondition : String
isNullable : Boolean
autoUpdate : Boolean
/ownedElement
/dataPackage
Catalog
defaultCharacterSetName : String
defaultCollationName : String
0..1
/ownedElement
Schema
0..1
*
Procedure
type : ProcedureType
Trigger
eventManipulation : EventManipulationType
actionCondition : BooleanExpression
actionStatement : ProcedureExpression
actionOrientation : ActionOrientationType
conditionTiming : ConditionTimingType
conditionReferenceNewTable : String
conditionReferenceOldTable : String
/ table : Table
/ usedColumnSet : NamedColumnSet
NamedColumnSet
/ optionScopeColumn : Column
/ type : SQLStructuredType
/ usingTrigger : Trigger

Figure A.1: Extract of CWM Relational Model: Catalogue and Schema (OMG, 2003)

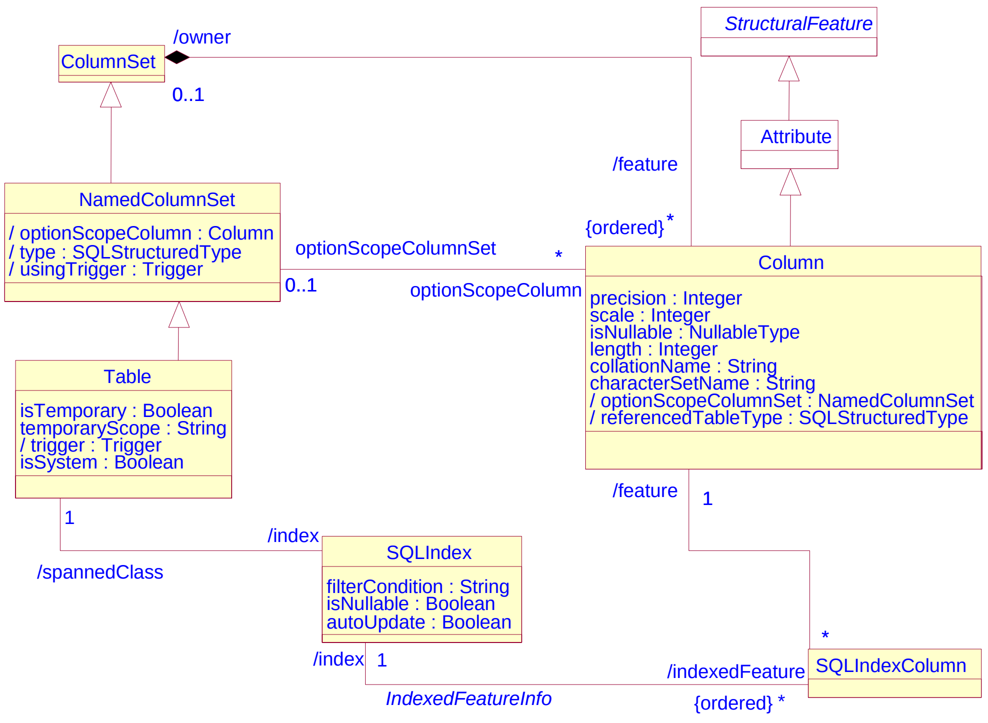


Figure A.2: Extract of CWM Relational Model: Table, column and data type classes (OMG, 2003)

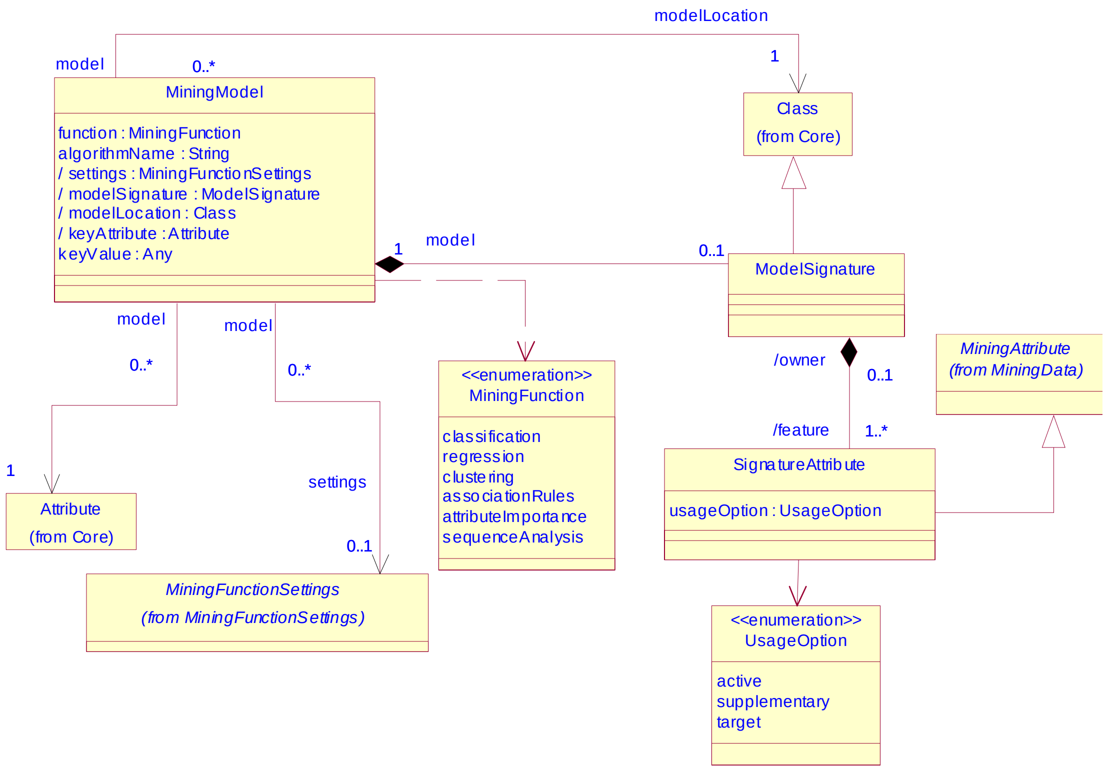


Figure A.3: Extract of CWM Mining Metamodel (OMG, 2003)

```
---------------------------------------------------------------------------
-- Descritive metadata for logistic regression
-- on PRODUCT PRICE, P_DESTINATION and SHIP_TAX
-- SAS procedure product_price_regression.sas
-- PROC LOGISTIC DATA=PRICE-P_DESTINATION-SHIP_TAX
-- MODEL SHIP_TAX = PRICE | dist(P_DESTINATION) / SELECTION=forward; RUN;
---------------------------------------------------------------------------
<PMML version="3.0">
    <DataDictionary numberOfFields="3">
      <DataField name="PRICE" optype="continuous"/>
      <DataField name="SHIP_TAX" optype="continuous"/>
      <DataField name="dist(P_DESTINATION)" optype="categorical">
        <Value value="<10"/>
        <Value value="[10,50]"/>
        <Value value=">50"/>
      </DataField>
    </DataDictionary>
    <RegressionModel
       modelName="Logistic regression"
       functionName="classification"
       algorithmName="logisticRegression"
       normalizationMethod="softmax"
       targetFieldName="SHIP_TAX">
      <MiningSchema>
        <MiningField name="PRICE"/>
        <MiningField name="dist(P_DESTINATION)"/>
        <MiningField name="SHIP_TAX" usageType="predicted"/>
      </MiningSchema>
      <RegressionTable intercept="46.418">
        <NumericPredictor name="PRICE" exponent="1" coefficient="-0.132"/>
        <CategoricalPredictor name="dist(P_DESTINATION)"
                              value="<10" coefficient="41.1"/>
        <CategoricalPredictor name="dist(P_DESTINATION)"
                              value="[10,50]" coefficient="12.1"/>
        <CategoricalPredictor name="dist(P_DESTINATION)"
                             value=">50" coefficient="25.03"/>
      </RegressionTable>
```

Table A.2: PMML Specifications of Logistic Regression on a data subset of CRM_DB

```
----------------------------------------------------------------------------
-- (1) PL/SQL procedure for statistics on numerical attributes
-- and PRICE outlier detection based on IQR (plsql_func_iqr.sql)
----------------------------------------------------------------------------
DECLARE
v_ownername varchar2(8);
v_tablename varchar2(50);
v_columnname varchar2(50);
v_sigma_value number;
type n_arr1 is varray(5) of number;
type num_table1 is table of number;
s1 dbms_stat_funcs.summaryType;
BEGIN
v_ownername := 'BERTI';
v_tablename := 'PRODUCT';
v_columnname :='PRICE';
v_sigma_value := 3;
dbms_stat_funcs.summary(p_ownername=>v_ownername, p_tablename=>v_tablename,
        p_columnname=> v_columnname, p_sigma_value=>v_sigma_value, s=> s1);
END;
----
CREATE TABLE price_outliers (pid varchar2(4), price number);
DECLARE
s dbms_stat_funcs.summaryType;
BEGIN
dbms_stat_funcs.summary('BERTI','PRODUCT','PRICE',3,s);
dbms_output.put_line('SUMMARY STATISTICS');
dbms_output.put_line('Count: '||s.count);
dbms_output.put_line('Min: '||s.min);
dbms_output.put_line('Max: '||s.max);
dbms_output.put_line('Range: '||s.range);
dbms_output.put_line('Mean:'||round(s.mean));
dbms_output.put_line('Mode:'||s.cmode(1));
dbms_output.put_line('Variance:'||round(s.variance));
dbms_output.put_line('Stddev:'||round(s.stddev));
dbms_output.put_line('Quantile 5 '||s.quantile_5);
dbms_output.put_line('Quantile 25 '||s.quantile_25);
dbms_output.put_line('Median  '||s.median);
dbms_output.put_line('Quantile 75 '||s.quantile_75);
dbms_output.put_line('Quantile 95 '||s.quantile_95);
dbms_output.put_line('Extreme Count:'||s.extreme_values.count);
dbms_output.put_line('Top 3:'||s.top_5_values(1)||',
         '||s.top_5_values(2)||','||','||s.bottom_5_values(3));
insert into price_outliers select prod_id,price  from PRODUCT
                     where (price > s.quantile_95) or (price < s.quantile_5);
END;
/
```

Table A.3: PRICE Outlier Detection in PRODUCT table (PL/SQL)

```
*-------------------------------------------------------------
*-- (2) SAS procedure to detect outlier
*-- on PRICE of PRODUCT (sas_outlier_check.sas)
*-------------------------------------------------------------
PROC UNIVARIATE DATA=PRICE NOprint;
VAR p;
OUTPUT OUT=metadata Q1=q1 Q3=q3 QRANGE=iqr;
RUN;
DATA _null_;  SET metadata; CALL SYMPUT("q1",q1);
CALL SYMPUT("q3",q3); CALL SYMPUT("iqr",iqr);
RUN;
* save the PRICE outliers;
DATA price_outliers;
SET PRICE; LENGTH severity $2;
severity=" ";
IF (p <= (&q1 - 1.5*&iqr)) OR (p >= (&q3 + 1.5*&iqr))
   THEN severity="*";
IF (p <= (&q1 -   3*&iqr)) OR (p >= (&q3 +   3*&iqr))
   THEN severity="**";
IF severity IN ("*", "**") THEN OUTPUT price_outliers;
RUN;
```

Table A.4: PRICE Outlier Detection in PRODUCT table (SAS)

Bibliography

AGGARWAL, CHARU. 2007. *Data Streams: Models and Algorithms.* Springer.

AGRAWAL, RAKESH, IMIELINSKI, TOMASZ, & SWAMI, ARUN N. 1993. Mining Association Rules Between Sets of Items in Large Databases. Pages 207–216 of: *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data.* Washington, DC, USA.

ANANTHAKRISHNA, ROHIT, CHAUDHURI, SURAJIT, & GANTI, VENKATESH. 2002. Eliminating Fuzzy Duplicates in Data Warehouses. Pages 586–597 of: *Proceedings of 28th International Conference on Very Large Data Bases, VLDB 2002.* Hong Kong, China.

ARENAS, MARCELO, BERTOSSI, LEOPOLDO E., & CHOMICKI, JAN. 1999. Consistent Query Answers in Inconsistent Databases. Pages 68–79 of: *Proceedings of the 18th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems.* Philadelphia, PA, USA.

ARENAS, MARCELO, BERTOSSI, LEOPOLDO E., & CHOMICKI, JAN. 2000. Specifying and Querying Database Repairs Using Logic Programs with Exceptions. Pages 27–41 of: *Proceedings of the 4th International Conference on Flexible Query Answering Systems, FQAS 2000.* Warsaw, Poland.

ARENAS, MARCELO, BERTOSSI, LEOPOLDO E., & CHOMICKI, JAN. 2003. Answer Sets for Consistent Query Answering in Inconsistent Databases. *Theory and Practice of Logic Programming (TPLP)*, **3**(4-5), 393–424.

BALLOU, DONALD P., & TAYI, GIRI KUMAR. 1989. Methodology for Allocating Resources for Data Quality Enhancement. *Commun. ACM*, **32**(3), 320–329.

BALLOU, DONALD P., & TAYI, GIRI KUMAR. 1999. Enhancing Data Quality in Data Warehouse Environments. *Commun. ACM*, **42**(1), 73–78.

BANSAL, NIKHIL, BLUM, AVRIM, & CHAWLA, SHUCHI. 2002. Correlation Clustering. Page 238 of: *Proceedings of 43rd Symposium on Foundations of Computer Science, FOCS 2002.* Vancouver, BC, Canada.

BARBARÁ, DANIEL, GARCIA-MOLINA, HECTOR, & PORTER, DARYL. 1990. A Probalilistic Relational Data Model. Pages 60–74 of: *Proceedings of the 2nd International Conference on Extending Database Technology, EDBT 1990*. Lecture Notes in Computer Science, vol. 416. Venice, Italy.

BARGA, ROGER S., & PU, CALTON. 1993. Accessing Imprecise Data: An Approach Based on Intervals. *IEEE Data Eng. Bull.*, **16**(2), 12–15.

BASU, AYANENDRANATH, HARRIS, IAN R., & BASU, SRABASHI. 1997. Minimum Distance Estimation: The Approach Using Density-Based Distances. *Handbook of Statistics*, **15**, 21–48.

BATINI, CARLO, & SCANNAPIECO, MONICA. 2006. *Data Quality: Concepts, Methodologies and Techniques*. Data-Centric Systems and Applications. Springer-Verlag.

BATINI, CARLO, TIZIANA, CATARCI, & SCANNAPIECO, MONICA. 2004. A Survey of Data Quality Issues in Cooperative Systems. In: *Tutorial of the 23rd International Conference on Conceptual Modeling, ER 2004*. Shanghai, China.

BAXTER, ROHAN A., CHRISTEN, PETER, & CHURCHES, TIM. 2003. A Comparison of Fast Blocking Methods for Record Linkage. Pages 27–29 of: *Proceedings of the KDD'03 Workshop on Data Cleaning, Record Linkage and Object Consolidation*. Washington, DC, USA.

BENJELLOUN, OMAR, SARMA, ANISH DAS, HALEVY, ALON, & WIDOM, JENNIFER. 2005 (June). *The Symbiosis of Lineage and Uncertainty*. Technical Report 2005-39. Stanford InfoLab, Stanford University, CA, USA.

BERENGUER, GEMA, ROMERO, RAFAEL, TRUJILLO, JUAN, SERRANO, MANUEL, & PIATTINI, MARIO. 2005. A Set of Quality Indicators and Their Corresponding Metrics for Conceptual Models of Data Warehouses. Pages 95–104 of: *Proceedings of the 7th International Conference on Data Warehousing and Knowledge Discovery, DaWaK 2005*. Lecture Notes in Computer Science, vol. 3589. Copenhagen, Denmark.

BERNSTEIN, PHILIP A., BERGSTRAESSER, THOMAS, CARLSON, JASON, PAL, SHANKAR, SANDERS, PAUL, & SHUTT, DAVID. 1999. Microsoft Repository Version 2 and the Open Information Model. *Inf. Syst.*, **24**(2), 71–98.

BERTI-ÉQUILLE, LAURE. 1999b. Qualité des données multi-sources et recommandation multi-critère. Pages 185–204 of: *Actes du congrès francophone INFormatique des ORganisations et Systèmes d'Information Décisionnels, INFORSID 1999*. Toulon, France.

BERTI-ÉQUILLE, LAURE. 1999c. Quality and Recommendation of Multi-Source Data for Assisting Technological Intelligence Applications. Pages 282–291 of: *Proceedings of the International Conference on Database and Expert Systems Applications, (DEXA'99)*. Lecture Notes in Computer Science, vol. 1677. Florence, Italy.

BERTI-ÉQUILLE, LAURE. 2001. Integration of Biological Data and Quality-driven Source Negotiation. Pages 256–269 of: *Proceedings of the 20th International Conference on Conceptual Modeling, ER'2001*. Lecture Notes in Computer Science, vol. 2224. Yokohama, Japan.

BERTI-ÉQUILLE, LAURE. 2002. Annotation et recommandation collaboratives de documents selon leur qualité. *Revue Ingénieire des Systèmes d'Information (ISI-NIS), Numéro Spécial "Recherche et Filtrage d'Information"*, **7**(1-2 / 2002), 125–156.

BERTI-ÉQUILLE, LAURE. 2003. Quality-Extended Query Processing for Distributed Sources. In: *Proceedings of the 1rst International Workshop on Data Quality in Cooperative Information Systems, DQCIS'2003*. Siena, Italy.

BERTI-ÉQUILLE, LAURE. 2003a. Quality-based Recommendation of XML Documents. *Journal of Digital Information Management*, **1**(3), 117–128.

BERTI-ÉQUILLE, LAURE. 2004. Quality-Adaptive Query Processing over Distributed Sources. Pages 285–296 of: *Proceedings of the 9th International Conference on Information Quality, ICIQ 2004*. Massachusetts Institute of Technology, Cambridge, MA, USA.

BERTI-ÉQUILLE, LAURE, & MOUSSOUNI, FOUZIA. 2005. Quality-Aware Integration and Warehousing of Genomic Data. Pages 442–454 of: *Proceedings of the 10th International Conference on Information Quality, ICIQ 2005*. Massachusetts Institute of Technology, Cambridge, MA, USA.

BERTI-ÉQUILLE, LAURE, MOUSSOUNI, FOUZIA, & ARCADE, ANNE. 2001. Integration of Biological Data on Transcriptome. *Revue ISI-NIS, Numéro Spécial Interopérabilité et Intégration des Systèmes d'Information*, **6**(3 / 2001), 61–86.

BERTI-ÉQUILLE, LAURE. 2006a. Data Quality Awareness: a Case Study for Cost-Optimal Association Rule Mining. *Knowl. Inf. Syst.*, **11**(2), 191–215.

BERTI-ÉQUILLE, LAURE. 2006b. Qualité des données. *Techniques de l'Ingénieur*, **H3700**, 1–19.

BERTI-ÉQUILLE, LAURE. 2006c. Quality-Aware Association Rule Mining. Pages 440–449 of: *Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining, (*PAKDD *2006)*. Lecture Notes in Artificial Intelligence, vol. 3918. Springer.

BERTOSSI, LEOPOLDO E., & BRAVO, LORETO. 2005. Consistent Query Answers in Virtual Data Integration Systems. Pages 42–83 of: *Inconsistency Tolerance, Dagstuhl Seminar*. Lecture Notes in Computer Science, vol. 3300. Schloss Dagstuhl, Germany.

BERTOSSI, LEOPOLDO E., & CHOMICKI, JAN. 2003. Query Answering in Inconsistent Databases. Pages 43–83 of: *Logics for Emerging Applications of Databases, Dagstuhl Seminar*. Schloss Dagstuhl, Germany.

BERTOSSI, LEOPOLDO E., & SCHWIND, CAMILLA. 2004. Database Repairs and Analytic Tableaux. *Ann. Math. Artif. Intell.*, **40**(1-2), 5–35.

BHATTACHARYA, INDRAJIT, & GETOOR, LISE. 2004. Iterative Record Linkage for Cleaning and Integration. Pages 11–18 of: *Proceedings of the 9th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, DMKD 2004*. Paris, France.

BILENKO, MIKHAIL, & MOONEY, RAYMOND J. 2003. Adaptive Duplicate Detection Using Learnable String Similarity Measures. Pages 39–48 of: *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Washington, DC, USA.

BILENKO, MIKHAIL, BASU, SUGATO, & SAHAMI, MEHRAN. 2005. Adaptive Product Normalization: Using Online Learning for Record Linkage in Comparison Shopping. Pages 58–65 of: *Proceedings of the 5th IEEE International Conference on Data Mining, ICDM 2005*. Houston, TX, USA.

BILKE, ALEXANDER, BLEIHOLDER, JENS, BÖHM, CHRISTOPH, DRABA, KARSTEN, NAUMANN, FELIX, & WEIS, MELANIE. 2005. Automatic Data Fusion with HumMer. Pages 1251–1254 of: *Proceedings of the 31st International Conference on Very Large Data Bases, VLDB 2005*. Trondheim, Norway.

BOSC, PATRICK, LIETARD, NADIA, & PIVERT, OLIVIER. 2006. About Inclusion-Based Generalized Yes / No Queries in a Possibilistic Database Context. Pages 284–289 of: *Proceedings of the 16th International Symposium on Foundations of Intelligent Systems, ISMIS 2006*. Bari, Italy.

BOUZEGHOUB, MOKRANE, & PERALTA, VERÓNIKA. 2004. A Framework for Analysis of Data Freshness. Pages 59–67 of: *Proceedings of the 1st International ACM SIGMOD 2004 Workshop on Information Quality in Information Systems, IQIS 2004*. Paris, France.

BRAUMANDL, REINHARD, KEIDL, MARKUS, KEMPER, ALFONS, KOSSMANN, DONALD, SELTZSAM, STEFAN, & STOCKER, KONRAD. 2001. ObjectGlobe: Open Distributed Query Processing Services on the Internet. *IEEE Data Eng. Bull.*, **24**(1), 64–70.

BRAVO, LORETO, & BERTOSSI, LEOPOLDO E. 2003. Logic Programs for Consistently Querying Data Integration Systems. Pages 10–15 of: *Proceedings of the 18th International Joint Conference on Artificial Intelligence, IJCAI-03*. Acapulco, Mexico.

BREUNIG, MARKUS M., KRIEGEL, HANS-PETER, NG, RAYMOND T., & SANDER, JÖRG. 2000. LOF: Identifying Density-Based Local Outliers. Pages 93–104 of: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. Dallas, TX, USA.

BRIGHT, LAURA, & RASCHID, LOUIQA. 2002. Using Latency-Recency Profiles for Data Delivery on the Web. Pages 550–561 of: *Proceedings of 28th International Conference on Very Large Data Bases, VLDB 2002*. Hong Kong, China.

BRY, FRANÇOIS. 1997. Query Answering in Information Systems with Integrity Constraints. Pages 113–130 of: *Integrity and Internal Control in Information Systems, IFIP TC11 Working Group 11.5, First Working Conference on Integrity and Internal Control in Information Systems: Increasing the confidence in Information Systems, IICIS*. Zurich, Switzerland.

BUECHI, MARTIN, BORTHWICK, ANDREW, WINKEL, ADAM, & GOLDBERG, ARTHUR. 2003. ClueMaker: A Language for Approximate Record Matching. Pages 207–223 of: *Proceedings of the 8th International Conference on Information Quality, ICIQ 2003*. MIT, Cambridge, MA, USA.

CALÌ, ANDREA, LEMBO, DOMENICO, & ROSATI, RICCARDO. 2003. On the Decidability and Complexity of Query Answering Over Inconsistent and Incomplete Databases. Pages 260–271 of: *Proceedings of the 22nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS*. San Diego, CA, USA.

CARREIRA, PAULO J. F., & GALHARDAS, HELENA. 2004. Execution of Data Mappers. Pages 2–9 of: *Proceedings of the 1st International ACM SIGMOD 2004 Workshop on Information Quality in Information Systems, IQIS 2004*. Paris, France.

CARUSO, FRANCESCO, COCHINWALA, MUNIR, GANAPATHY, UMA, LALK, GAIL, & MISSIER, PAOLO. 2000. Telcordia's Database Reconciliation and Data Quality Analysis Tool. Pages 615–618 of: *Proceedings of 26th International Conference on Very Large Data Bases, VLDB 2000*. Cairo, Egypt.

CAVALLO, ROGER, & PITTARELLI, MICHAEL. 1987. The Theory of Probabilistic Databases. Pages 71–81 of: *Proceedings of 13th International Conference on Very Large Data Bases, VLDB 1987*. Brighton, England.

CERI, STEFANO, COCHRANE, ROBERTA, & WIDOM, JENNIFER. 2000. Practical Applications of Triggers and Constraints: Success and Lingering Issues. Pages 254–262 of: *Proceedings of 26th International Conference on Very Large Data Bases, VLDB 2000*. Cairo, Egypt.

CHARIKAR, MOSES, GURUSWAMI, VENKATESAN, & WIRTH, ANTHONY. 2003. Clustering with Qualitative Information. Pages 524–533 of: *Proceedings of 44th Symposium on Foundations of Computer Science, FOCS 2003*. Cambridge, MA, USA.

CHAUDHURI, SURAJIT, GANJAM, KRIS, GANTI, VENKATESH, & MOTWANI, RAJEEV. 2003. Robust and Efficient Fuzzy Match for Online Data Cleaning. Pages 313–324 of: *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*. San Diego, CA, USA.

CHAUDHURI, SURAJIT, GANTI, VENKATESH, & MOTWANI, RAJEEV. 2005. Robust Identification of Fuzzy Duplicates. Pages 865–876 of: *Proceedings of the 21st International Conference on Data Engineering, ICDE 2005*. Tokyo, Japan.

CHAUDHURI, SURAJIT, GANTI, VENKATESH, & KAUSHIK, RAGHAV. 2006. A Primitive Operator for Similarity Joins in Data Cleaning. Page 5 of: *Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006*. Atlanta, GA, USA.

CHENG, REYNOLD, KALASHNIKOV, DMITRI V., & PRABHAKAR, SUNIL. 2003. Evaluating Probabilistic Queries over Imprecise Data. Pages 551–562 of: *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*. San Diego, CA, USA.

CHO, JUNGHOO, & GARCIA-MOLINA, HECTOR. 2000. Synchronizing a Database to Improve Freshness. Pages 117–128 of: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. Dallas, TX, USA.

CHOENNI, SUNIL, BLOK, HENK ERNST, & LEERTOUWER, ERIK. 2006. Handling Uncertainty and Ignorance in Databases: A Rule to Combine Dependent Data. Pages 295–309 of: *Proceedings of 11th International Conference on Database Systems for Advanced Applications, DASFAA 2006*. Lecture Notes in Computer Science, vol. 3882. Singapore.

CHOMICKI, JAN. 2006. Consistent Query Answering: Opportunities and Limitations. Pages 527–531 of: *Proceedings of 2nd International Workshop on Logical Aspects and Applications of Integrity Constraints, LAAIC 2006*. Krakow, Poland.

CHRISTEN, PETER, CHURCHES, TIM, & HEGLAND, MARKUS. 2004. Febrl - A Parallel Open Source Data Linkage System. Pages 638–647 of: *Proceedings of the 8th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, PAKDD 2004*. Lecture Notes in Computer Science, vol. 3056. Sydney, Australia.

COHEN, WILLIAM W., RAVIKUMAR, PRADEEP, & FIENBERG, STEPHEN E. 2003. A Comparison of String Distance Metrics for Name-Matching Tasks. Pages 73–78 of: *Proceedings of IJCAI-03 Workshop on Information Integration on the Web, IIWeb-03*. Acapulco, Mexico.

COULON, CÉDRIC, PACITTI, ESTHER, & VALDURIEZ, PATRICK. 2005. Consistency Management for Partial Replication in a High Performance Database Cluster. Pages 809–815 of: *Proceedings of 11th International Conference on Parallel and Distributed Systems, ICPADS 2005*, vol. 1. Fuduoka, Japan.

CUI, YINGWEI, & WIDOM, JENNIFER. 2003. Lineage Tracing for General Data Warehouse Transformations. *VLDB J.*, **12**(1), 41–58.

CULOTTA, ARON, & MCCALLUM, ANDREW. 2005. Joint Deduplication of Multiple Record Types in Relational Data. Pages 257–258 of: *Proceedings of the 2005 ACM International Conference on Information and Knowledge Management, CIKM 2005*. Bremen, Germany.

DALVI, NILESH N., & SUCIU, DAN. 2004. Efficient Query Evaluation on Probabilistic Databases. Pages 864–875 of: *Proceedings of the 30th International Conference on Very Large Data Bases, VLDB 2004*. Toronto, ON, Canada.

DASU, TAMRAPARNI, & JOHNSON, THEODORE. 2003. *Exploratory Data Mining and Data Cleaning.* John Wiley.

DASU, TAMRAPARNI, JOHNSON, THEODORE, MUTHUKRISHNAN, S., & SHKAPENYUK, VLADISLAV. 2002. Mining Database Structure or How To Build a Data Quality Browser. Pages 240–251 of: *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data.* Madison, WI, USA.

DASU, TAMRAPARNI, VESONDER, GREGG T., & WRIGHT, JON R. 2003. Data Quality Through Knowledge Engineering. Pages 705–710 of: *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2003.* Washington, DC, USA.

DEMPSTER, ARTHUR PENTLAND, LAIRD, NAN M., & RUBIN, DONALD B. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, **39**, 1–38.

DOMINGOS, PEDRO, & HULTEN, GEOFF. 2001. Catching up with the Data: Research Issues in Mining Data Streams. In: *Proceedings of 2001 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, DMKD2001.* Santa Barbara, CA, USA.

DONG, XIN, HALEVY, ALON Y., & MADHAVAN, JAYANT. 2005. Reference Reconciliation in Complex Information Spaces. Pages 85–96 of: *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data.* Baltimore, MD, USA.

DUMOUCHEL, WILLIAM, VOLINSKY, CHRIS, JOHNSON, THEODORE, CORTES, CORINNA, & PREGIBON, DARYL. 1999. Squashing Flat Files Flatter. Pages 6–15 of: *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 1999.* San Diego, CA, USA.

ELFEKY, MOHAMED G., ELMAGARMID, AHMED K., & VERYKIOS, VASSILIOS S. 2002. TAILOR: A Record Linkage Tool Box. Pages 17–28 of: *Proceedings of the 18th International Conference on Data Engineering, ICDE 2002.* San Jose, CA, USA.

ELMAGARMID, AHMED K., IPEIROTIS, PANAGIOTIS G., & VERYKIOS, VASSILIOS S. 2007. Duplicate Record Detection: A Survey. *IEEE Trans. Knowl. Data Eng.*, **19**(1), 1–16.

EMBURY, SUZANNE M., BRANDT, SUE M., ROBINSON, JOHN S., SUTHERLAND, IAIN, BISBY, FRANK A., GRAY, W. ALEX, JONES, ANDREW C., & WHITE, RICHARD J. 2001. Adapting Integrity Enforcement Techniques for Data Reconciliation. *Inf. Syst.*, **26**(8), 657–689.

ENGLISH, LARRY. 2002. Process Management and Information Quality: How Improving Information Production Processes Improves Information (Product) Quality. Pages 206–209 of: *Proceedings of the Seventh International Conference on Information Quality, ICIQ 2002.* MIT, Cambridge, MA, USA.

English, Larry P. 1999. *Improving Data Warehouse and Business Information Quality*. Wiley.

Fagin, Ronald, Kolaitis, Phokion G., Miller, Renée J., & Popa, Lucian. 2003. Data Exchange: Semantics and Query Answering. Pages 207–224 of: *Proceedings of 9th International Conference on Database Theory, ICDT 2003*. Lecture Notes in Computer Science, vol. 2572. Siena, Italy.

Faloutsos, Christos. 2002. Sensor Data Mining: Similarity Search and Pattern Analysis. In: *Tutorial of 28th International Conference on Very Large Data Bases, VLDB 2002*. Hong Kong, China.

Faloutsos, Christos, & Lin, King-Ip. 1995. FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets. Pages 163–174 of: *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*. San Jose, CA, USA.

Fellegi, Ivan P., & Sunter, A.B. 1969. A Theory for Record Linkage. *Journal of the American Statistical Association*, **64**, 1183–1210.

Flesca, Sergio, Furfaro, Filippo, & Parisi, Francesco. 2005. Consistent Query Answers on Numerical Databases Under Aggregate Constraints. Pages 279–294 of: *Proceedings of 10th International Symposium on Database Programming Languages, DBPL 2005*. Trondheim, Norway.

Fox, Christopher J., Levitin, Anany, & Redman, Thomas. 1994. The Notion of Data and Its Quality Dimensions. *Inf. Process. Manage.*, **30**(1), 9–20.

Galhardas, Helena, Florescu, Daniela, Shasha, Dennis, & Simon, Eric. 2000. AJAX: An Extensible Data Cleaning Tool. Page 590 of: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. Dallas, TX, USA.

Galhardas, Helena, Florescu, Daniela, Shasha, Dennis, Simon, Eric, & Saita, Cristian-Augustin. 2001. Declarative Data Cleaning: Language, Model, and Algorithms. Pages 371–380 of: *Proceedings of 27th International Conference on Very Large Data Bases, VLDB 2001*. Roma, Italy.

Gelenbe, Erol, & Hébrail, Georges. 1986. A Probability Model of Uncertainty in Data Bases. Pages 328–333 of: *Proceedings of the Second International Conference on Data Engineering, ICDE 1986*. Los Angeles, CA, USA.

Grahne, Gösta. 2002. Information Integration and Incomplete Information. *IEEE Data Eng. Bull.*, **25**(3), 46–52.

Gravano, Luis, Ipeirotis, Panagiotis G., Jagadish, H. V., Koudas, Nick, Muthukrishnan, S., Pietarinen, Lauri, & Srivastava, Divesh. 2001. Using q-grams in a DBMS for Approximate String Processing. *IEEE Data Eng. Bull.*, **24**(4), 28–34.

GRAVANO, LUIS, IPEIROTIS, PANAGIOTIS G., KOUDAS, NICK, & SRIVASTAVA, DIVESH. 2003. Text Joins for Data Cleansing and Integration in an RDBMS. Pages 729–731 of: *Proceedings of the 19th International Conference on Data Engineering, ICDE 2003*. Bangalore, India.

GUÉRIN, EMILIE, MOUSSOUNI, FOUZIA, & BERTI-ÉQUILLE, LAURE. 2001. Intégration des données sur le transcriptome. Pages 219–228 of: *Actes de la journée de travail bi-thématique du GDR-PRC I3*. Lyon, France.

GUÉRIN, EMILIE, MARQUET, GWENAELLE, BURGUN, ANITA, LORÉAL, OLIVIER, BERTI-ÉQUILLE, LAURE, LESER, ULF, & MOUSSOUNI, FOUZIA. 2005. Integrating and Warehousing Liver Gene Expression Data and Related Biomedical Resources in GEDAW. Pages 158–174 of: *Proceedings of the 2nd International Workshop on Data Integration in the Life Sciences, DILS 2005*. San Diego, CA, USA.

GUHA, SUDIPTO, RASTOGI, RAJEEV, & SHIM, KYUSEOK. 2001. Cure: An Efficient Clustering Algorithm for Large Databases. *Inf. Syst.*, **26**(1), 35–58.

GUO, HONGFEI, LARSON, PER-ÅKE, RAMAKRISHNAN, RAGHU, & GOLDSTEIN, JONATHAN. 2004. Relaxed Currency and Consistency: How to Say "Good Enough" in SQL. Pages 815–826 of: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. Paris, France.

GUO, HONGFEI, LARSON, PER-ÅKE, & RAMAKRISHNAN, RAGHU. 2005. Caching with 'Good Enough' Currency, Consistency, and Completeness. Pages 457–468 of: *Proceedings of the 31st International Conference on Very Large Data Bases, VLDB 2005*. Trondheim, Norway.

HALEVY, ALON Y. 2001. Answering Queries Using Views: A Survey. *VLDB J.*, **10**(4), 270–294.

HERNÁNDEZ, MAURICIO A., & STOLFO, SALVATORE J. 1998. Real-world Data is Dirty: Data Cleansing and The Merge / Purge Problem. *Data Min. Knowl. Discov.*, **2**(1), 9–37.

HJALTASON, GÍSLI R., & SAMET, HANAN. 2003. Properties of Embedding Methods for Similarity Searching in Metric Spaces. *IEEE Trans. Pattern Anal. Mach. Intell.*, **25**(5), 530–549.

HOU, WEN-CHI, & ZHANG, ZHONGYANG. 1995. Enhancing Database Correctness: a Statistical Approach. Pages 223–232 of: *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*. San Jose, CA, USA.

HULL, RICHARD, & ZHOU, GANG. 1996. A Framework for Supporting Data Integration Using the Materialized and Virtual Approaches. Pages 481–492 of: *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*. Montreal, Quebec, Canada.

HULTEN, GEOFF, SPENCER, LAURIE, & DOMINGOS, PEDRO. 2001. Mining time-changing data streams. Pages 97–106 of: *Proceedings of the 7th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2001*. San Francisco, CA, USA.

HUNG, EDWARD, GETOOR, LISE, & SUBRAHMANIAN, V. S. 2003. PXML: A Probabilistic Semistructured Data Model and Algebra. Page 467 of: *Proceedings of the 19th International Conference on Data Engineering, ICDE'03*. Bangalore, India.

IBRAHIM, HAMIDAH. 2002. A Strategy for Semantic Integrity Checking in Distributed Databases. Pages 139–144 of: *Proceedings of 9th International Conference on Parallel and Distributed Systems, ICPADS 2002*. Taiwan, ROC.

IMIELINSKI, TOMASZ, & LIPSKI, WITOLD JR. 1984. Incomplete Information in Relational Databases. *J. ACM*, **31**(4), 761–791.

JARKE, MATTHIAS, JEUSFELD, MANFRED A., QUIX, CHRISTOPH, & VASSILIADIS, PANOS. 1999. Architecture and Quality in Data Warehouses: An Extended Repository Approach. *Inf. Syst.*, **24**(3), 229–253.

JARO, MATTHEW A. 1989. Advances in Record Linking Methodology as Applied to the 1985 Census of Tampa Florida. *Journal of the American Statistical Society*, **64**, 1183–1210.

JARO, MATTHEW A. 1995. Probabilistic Linkage of Large Public Health Data File. *Statistics in Medicine*, **14**, 491–498.

KAHN, BEVERLY K., STRONG, DIANE M., & WANG, RICHARD Y. 2002. Information Quality Benchmarks: Product and Service Performance. *Commun. ACM*, **45**(4), 184–192.

KALASHNIKOV, DMITRI V. & MEHROTRA, SHARAD. 2006. Domain-Independent Data Cleaning via Analysis of Entity-Relationship Graph. *ACM Transactions on Database Systems*, **31**(2), 716–767.

KARAKASIDIS, ALEXANDROS, VASSILIADIS, PANOS, & PITOURA, EVAGGELIA. 2005. ETL Queues for Active Data Warehousing. Pages 28–39 of: *Proceedings of the 2nd International ACM SIGMOD 2005 Workshop on Information Quality in Information Systems, IQIS 2005*. Baltimore, MA, USA.

KAUFMAN, L., & ROUSSEEUW, PETER J. 1990. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley.

KNORR, EDWIN M., & NG, RAYMOND T. 1998. Algorithms for Mining Distance-Based Outliers in Large Datasets. Pages 392–403 of: *Proceedings of 24rd International Conference on Very Large Data Bases, VLDB 1998*. New York City, NY, USA.

KORN, FLIP, MUTHUKRISHNAN, S., & ZHU, YUNYUE. 2003. Checks and Balances: Monitoring Data Quality Problems in Network Traffic Databases. Pages 536–547 of: *Proceedings of 29th International Conference on Very Large Data Bases, VLDB 2003*. Berlin, Germany.

LABRINIDIS, ALEXANDROS, & ROUSSOPOULOS, NICK. 2003. Balancing Performance and Data Freshness in Web Database Servers. Pages 393–404 of: *Proceedings of 29th International Conference on Very Large Data Bases, VLDB 2003*. Berlin, Germany.

LACROIX, ZOE, & CRITCHLOW, TERENCE (eds). 2003. *Bioinformatics: Managing Scientific Data*. Morgan Kaufmann.

LAKSHMANAN, LAKS V. S., & SADRI, FEREIDOON. 1994. Modeling Uncertainty in Deductive Databases. Pages 724–733 of: *Proceedings of the 5th International Conference on Database and Expert Systems Applications, DEXA'94*. Lecture Notes in Computer Science, vol. 856. Athens, Greece.

LAVRAČ, NADA, FLACH, PETER A., & ZUPAN, BLAZ. 1999. Rule Evaluation Measures: A Unifying View. Pages 174–185 of: *Proceedings of the Intl. Workshop on Inductive Logic Programming, ILP 1999*. Bled, Slovenia.

LAZARIDIS, IOSIF, & MEHROTRA, SHARAD. 2004. Approximate Selection Queries over Imprecise Data. Pages 140–152 of: *Proceedings of the 20th International Conference on Data Engineering, ICDE 2004*. Boston, MA, USA.

LEE, LILLIAN. 2001. On the Effectiveness of the Skew Divergence for Statistical Language Analysis. *Artificial Intelligence and Statistics*, 65–72.

LEE, MONG-LI, HSU, WYNNE, & KOTHARI, VIJAY. 2004. Cleaning the Spurious Links in Data. *IEEE Intelligent Systems*, **19**(2), 28–33.

LEE, SUK KYOON. 1992. An Extended Relational Database Model for Uncertain and Imprecise Information. Pages 211–220 of: *Proceedings of the 18th International Conference on Very Large Data Bases, VLDB 1992*. Vancouver, Canada.

LEMBO, DOMENICO, LENZERINI, MAURIZIO, & ROSATI, RICCARDO. 2002. Source Inconsistency and Incompleteness in Data Integration. In: *Proceedings of the 9th International Workshop on Knowledge Representation meets Databases, KRDB 2002*, vol. 54. Toulouse, France.

LI, CHEN. 2003. Computing Complete Answers to Queries in the Presence of Limited Access Patterns. *VLDB J.*, **12**(3), 211–227.

LI, WEN-SYAN, PO, OLIVER, HSIUNG, WANG-PIN, CANDAN, K. SELÇUK, & AGRAWAL, DIVYAKANT. 2003. Freshness-Driven Adaptive Caching for Dynamic Content Web Sites. *Data Knowl. Eng.*, **47**(2), 269–296.

LIEPINS, GUNAR E., & UPPULURI, V. R. 1991. *Data Quality Control: Theory and Pragmatics*. New York, NY, USA: Marcel Dekker, Inc. 0-8247-8354-9.

LIM, EE-PENG, SRIVASTAVA, JAIDEEP, PRABHAKAR, SATYA, & RICHARDSON, JAMES. 1993. Entity Identification in Database Integration. Pages 294–301 of: *Proceedings of the 9th International Conference on Data Engineering, ICDE 1993*. Vienna, Austria.

LIN, JINXIN, & MENDELZON, ALBERTO O. 1998. Merging Databases Under Constraints. *Int. J. Cooperative Inf. Syst.*, **7**(1), 55–76.

LOSHIN, D. 2001. *Enterprise Knowledge Management: The Data Quality Approach*. Morgan Kaufmann.

LOW, WAI LUP, LEE, MONG-LI, & LING, TOK WANG. 2001. A Knowledge-Based Approach for Duplicate Elimination in Data Cleaning. *Inf. Syst.*, **26**(8), 585–606.

MANNINO, MICHAEL V., CHU, PAICHENG, & SAGER, THOMAS. 1988. Statistical Profile Estimation in Database Systems. *ACM Comput. Surv.*, **20**(3), 191–221.

MARQUET, GWENAELLE, BURGUN, ANITA, MOUSSOUNI, FOUZIA, GUÉRIN, EMILIE, LE DUFF, FRANCK, & LORÉAL, OLIVIER. 2003. BioMeKe: an Ontology-Based Biomedical Knowledge Extraction System Devoted to Transcriptome Analysis. *Studies in Health Technology and Informatics*, **95**, 80–86.

MARTINEZ, ALEXANDRA, & HAMMER, JOACHIM. 2005. Making Quality Count in Biological Data Sources. Pages 16–27 of: *Proceedings of the 2nd International ACM SIGMOD 2005 Workshop on Information Quality in Information Systems, IQIS 2005*. Baltimore, MA, USA.

MCCALLUM, ANDREW, NIGAM, KAMAL, & UNGAR, LYLE H. 2000. Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching. Pages 169–178 of: *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2000*. Boston, MA, USA.

MCCALLUM, ANDREW, BELLARE, KEDAR, & PEREIRA, FERNANDO. 2005. A Conditional Random Field for Discriminatively-trained Finite-state String Edit Distance. Pages 388–396 of: *Proceedings of the 21rst Conference in Uncertainty in Artificial Intelligence, UAI'05*. Edinburgh, Scotland, UK.

MCCLEAN, SALLY I., SCOTNEY, BRYAN W., & SHAPCOTT, MARY. 2001. Aggregation of Imprecise and Uncertain Information in Databases. *IEEE Trans. Knowl. Data Eng.*, **13**(6), 902–912.

MIHAILA, GEORGE A., RASCHID, LOUIQA, & VIDAL, MARIA-ESTHER. 2000. Using Quality of Data Metadata for Source Selection and Ranking. Pages 93–98 of: *Proceedings of the 3rd International Workshop on the Web and Databases, WebDB 2000*. Dallas, TX, USA.

MONGE, ALVARO E. 2000. Matching Algorithms within a Duplicate Detection System. *IEEE Data Eng. Bull.*, **23**(4), 14–20.

MONGE, ALVARO E., & ELKAN, CHARLES. 1996. The Field Matching Problem: Algorithms and Applications. Pages 267–270 of: *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, KDD 1996*. Portland, OR, USA.

MOTRO, AMIHAI, & ANOKHIN, PHILIPP. 2006. FusionPlex: Resolution of Data Inconsistencies in the Integration of Heterogeneous Information Sources. *Information Fusion*, **7**(2), 176–196.

MOTRO, AMIHAI, & RAKOV, IGOR. 1998. Estimating the Quality of Databases. Pages 298–307 of: *Proceedings of the 3rd International Conference on Flexible Query Answering Systems, FQAS'98*. Roskilde, Denmark.

MÜLLER, HEIKO, & NAUMANN, FELIX. 2003. Data Quality in Genome Databases. Pages 269–284 of: *Proceedings of the 8th International Conference on Information Quality, ICIQ 2003*. MIT, Cambridge, MA, USA.

MÜLLER, HEIKO, LESER, ULF, & FREYTAG, JOHANN CHRISTOPH. 2004. Mining for Patterns in Contradictory Data. Pages 51–58 of: *Proceedings of the 1st International ACM SIGMOD 2004 Workshop on Information Quality in Information Systems, IQIS 2004*. Paris, France.

MYLOPOULOS, JOHN, BORGIDA, ALEXANDER, JARKE, MATTHIAS, & KOUBARAKIS, MANOLIS. 1990. Telos: Representing Knowledge About Information Systems. *ACM Trans. Inf. Syst.*, **8**(4), 325–362.

NAJJAR, FAÏZA, & SLIMANI, YAHYA. 1999. Cardinality Estimation of Distributed Join Queries. Pages 66–70 of: *Proceedings of the 10th International DEXA Workshop on on Parallel & Distributed Databases: Innovative Applications & New Architectures*. Florence, Italy.

NASH, ALAN, & LUDÄSCHER, BERTRAM. 2004. Processing Unions of Conjunctive Queries with Negation under Limited Access Patterns. Pages 422–440 of: *Proceedings of the 9th International Conference on Extending Database Technology, EDBT 2004*. Lecture Notes in Computer Science, vol. 2992. Heraklion, Crete, Greece.

NAUMANN, FELIX. 2002. *Quality-Driven Query Answering for Integrated Information Systems*. Lecture Notes in Computer Science, vol. 2261. Springer-Verlag.

NAUMANN, FELIX, LESER, ULF, & FREYTAG, JOHANN CHRISTOPH. 1999. Quality-Driven Integration of Heterogenous Information Systems. Pages 447–458 of: *Proceedings of the 25th International Conference on Very Large Data Bases, VLDB 1999*. Edinburgh, Scotland, UK.

NAUMANN, FELIX, FREYTAG, JOHANN CHRISTOPH, & LESER, ULF. 2004. Completeness of Integrated Information Sources. *Inf. Syst.*, **29**(7), 583–615.

NAVARRO, GONZALO. 2001. A Guided Tour to Approximate String Matching. *ACM Comput. Surv.*, **33**(1), 31–88.

NEWCOMBE, H.B., KENNEDY, J.M., AXFORD, S.J., & JAMES, A.P. 1959. Automatic Linkage of Vital Records. *Science*, 954–959.

NEWCOMBE, HOWARD B., & KENNEDY, JAMES M. 1962. Record Linkage: Making Maximum Use of the Discriminating Power of Identifying Information. *Commun. ACM*, **5**(11), 563–566.

OLSON, JACK E. 2003. *Data Quality: The Accuracy Dimension*. Morgan Kaufmann.

OLSTON, CHRISTOPHER, & WIDOM, JENNIFER. 2005. Efficient Monitoring and Querying of Distributed, Dynamic Data via Approximate Replication. *IEEE Data Eng. Bull.*, **28**(1), 11–18.

OMG. 2003 (March). *Common Warehouse Metamodel (CWM), Specification Version 1.1*. Tech. rept. Object Management Group.

PANG, HWEEHWA, JAIN, ARPIT, RAMAMRITHAM, KRITHI, & TAN, KIAN-LEE. 2005. Verifying Completeness of Relational Query Results in Data Publishing. Pages 407–418 of: *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*. Baltimore, MD, USA.

PAPADIMITRIOU, SPIROS, SUN, JIMENG, & FALOUTSOS, CHRISTOS. 2005. Streaming Pattern Discovery in Multiple Time-Series. Pages 697–708 of: *Proceedings of the 31st International Conference on Very Large Data Bases, VLDB 2005*. Trondheim, Norway.

PARSONS, SIMON. 1996. Current Approaches to Handling Imperfect Information in Data and Knowledge Bases. *IEEE Trans. Knowl. Data Eng.*, **8**(3), 353–372.

PASULA, HANNA, MARTHI, BHASKARA, MILCH, BRIAN, RUSSELL, STUART J., & SHPITSER, ILYA. 2002. Identity Uncertainty and Citation Matching. Pages 1401–1408 of: *Proceedings of Advances in Neural Information Processing Systems 15, NIPS 2002*. Vancouver, BC, Canada.

PEARSON, RONALD K. 2005. *Mining Imperfect Data: Dealing with Contamination and Incomplete Records*. Philadelphia: SIAM.

PEIM, MARTIN, FRANCONI, ENRICO, & PATON, NORMAN W. 2003. Estimating the Quality of Answers When Querying Over Description Logic Ontologies. *Data Knowl. Eng.*, **47**(1), 105–129.

PERALTA, VERÓNIKA. 2006 (November). *Data Quality Evaluation in Data Integration Systems*. Ph.D. thesis, Université de Versailles, France & Universidad de la República, Uruguay.

PETROPOULOS, MICHALIS, DEUTSCH, ALIN, & PAPAKONSTANTINOU, YANNIS. 2006. Interactive Query Formulation over Web Service-Accessed Sources. Pages 253–264 of: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. Chicago, IL, USA.

POOLE, JOHN, CHANG, DAN, TOLBERT, DOUGLAS, & MELLOR, DAVID. 2003. *Common Warehouse Metamodel Developer's Guide*. New York: John Wiley & Sons Inc.

PRADHAN, SHEKHAR. 2003. Argumentation Databases. Pages 178–193 of: *Proceedings of 19th International Conference on Logic Programming, ICLP 2003*. Mumbai, India.

PYLE, DORIAN. 1999. *Data Preparation for Data Mining*. Morgan Kaufmann.

QUASS, DALLAN, & STARKEY, P. 2003. A Comparison of Fast Blocking Methods for Record Linkage. Pages 40–42 of: *Proceedings of the KDD 2003 Workshop on Data Cleaning, Record Linkage and Object Consolidation*. Washington, DC, USA.

RAHM, ERHARD, & DO, HONG HAI. 2000. Data Cleaning: Problems and Current Approaches. *IEEE Data Eng. Bull.*, **23**(4), 3–13.

RAMAMRITHAM, KRITHI. 1993. Real-Time Databases. *Distributed and Parallel Databases*, **1**(2), 199–226.

RAMAMRITHAM, KRITHI, & CHRYSANTHIS, PANOS K. 1992. In Search of Acceptability Citeria: Database Consistency Requirements and Transaction Correctness Properties. Pages 212–230 of: *Proceedings of International Workshop on Distributed Object Management, IWDOM 1992*. Edmonton, AL, Canada.

RAMAN, VIJAYSHANKAR, & HELLERSTEIN, JOSEPH M. 2001. Potter's Wheel: An Interactive Data Cleaning System. Pages 381–390 of: *Proceedings of 27th International Conference on Very Large Data Bases, VLDB 2001*. Roma, Italy.

RE, CHRISTOPHER, DALVI, NILESH N., & SUCIU, DAN. 2006. Query Evaluation on Probabilistic Databases. *IEEE Data Eng. Bull.*, **29**(1), 25–31.

REDMAN, THOMAS. 2001. *Data Quality: The Field Guide*. Digital Press, Elsevier.

RISTAD, ERIC SVEN, & YIANILOS, PETER N. 1998. Learning String-Edit Distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, **20**(5), 522–532.

SAMPAIO, SANDRA DE F. MENDES, DONG, CHAO, & SAMPAIO, PEDRO. 2005. Incorporating the Timeliness Quality Dimension in Internet Query Systems. Pages 53–62 of: *Proceedings of the International Workshop on Web Information Systems Engineering, WISE 2005*. New York, NY, USA.

SANTIS, LUCA DE, SCANNAPIECO, MONICA, & CATARCI, TIZIANA. 2003. Trusting Data Quality in Cooperative Information Systems. Pages 354–369 of: *Proceedings of CoopIS, DOA, and ODBASE - OTM Confederated International Conferences*. Catania, Sicily, Italy.

SARAWAGI, SUNITA, & KIRPAL, ALOK. 2004. Efficient Set Joins on Similarity Predicates. Pages 743–754 of: *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*. Paris, France.

SAYYADIAN, MAYSSAM, LEE, YOONKYONG, DOAN, ANHAI, & ROSENTHAL, ARNON. 2005. Tuning Schema Matching Software using Synthetic Scenarios. Pages 994–1005 of: *Proceedings of the 31st International Conference on Very Large Data Bases, VLDB 2005*. Trondheim, Norway.

SEGEV, ARIE, & FANG, WEIPING. 1990. Currency-Based Updates to Distributed Materialized Views. Pages 512–520 of: *Proceedings of the 6th International Conference on Data Engineering, ICDE 1090*. Los Angeles, CA, USA.

SHETH, AMIT P., WOOD, CHRISTOPHER, & KASHYAP, VIPUL. 1993. Q-Data: Using Deductive Database Technology to Improve Data Quality. Pages 23–56 of: *Proceedings of the International Workshop on Programming with Logic Databases, ILPS*. Vancouver, BC, Canada.

SIMITSIS, ALKIS, VASSILIADIS, PANOS, & SELLIS, TIMOS K. 2005. Optimizing ETL Processes in Data Warehouses. Pages 564–575 of: *Proceedings of the 21st International Conference on Data Engineering, ICDE 2005*. Tokyo, Japan.

SINGLA, PARAG, & DOMINGOS, PEDRO. 2005. Collective Object Identification. Pages 1636–1637 of: *Proceedings of the 19th International Joint Conference on Artificial Intelligence, IJCAI-05*. Edinburgh, Scotland, UK.

STOCKINGER, KURT. 2002. Bitmap Indices for Speeding Up High-Dimensional Data Analysis. Pages 881–890 of: *Proceedings of the 13th International Database and Expert Systems Applications Conference, DEXA 2002*. Lecture Notes in Computer Science, vol. 2453. Aix-en-Provence, France.

TAN, P.N., KUMAR, V., & SRIVASTAVA, J. 2002. Selecting the Right Interestingness Measure for Association Patterns. Pages 32–41 of: *Proceedings of the 8th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2002*. Edmonton,AL, Canada.

TEJADA, SHEILA, KNOBLOCK, CRAIG A., & MINTON, STEVEN. 2001. Learning Object Identification Rules for Information Integration. *Inf. Syst.*, **26**(8), 607–633.

TEJADA, SHEILA, KNOBLOCK, CRAIG A., & MINTON, STEVEN. 2002. Learning Domain-Independent String Transformation Weights for High Accuracy Object Identification. Pages 350–359 of: *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2002*. Edmonton, AL, Canada.

THANGAVEL, ALPHONSE THANARAJ. 1999. A Clean Data Set of EST-confirmed Splice Sites from Homo Sapiens and Standards for Clean-up Procedures. *Nucleic Acids Res.*, **27**(13), 2627–2637.

THEODORATOS, DIMITRI, & BOUZEGHOUB, MOKRANE. 1999. Data Currency Quality Factors in Data Warehouse Design. Page 15 of: *Proceedings of the International Workshop on Design and Management of Data Warehouses, DMDW'99*. Heidelberg, Germany.

THEODORATOS, DIMITRI, & BOUZEGHOUB, MOKRANE. 2001. Data Currency Quality Satisfaction in the Design of a Data Warehouse. *Int. J. Cooperative Inf. Syst.*, **10**(3), 299–326.

THOR, ANDREAS, & RAHM, ERHARD. 2007. MOMA - A Mapping-based Object Matching System. Pages 247–258 of: *Proceedings of the 3rd Biennial Conference on Innovative Data Systems Research, CIDR 2007*. Asilomar, CA, USA.

VAILLANT, BENOÎT, LENCA, PHILIPPE, & LALLICH, STÉPHANE. 2004. A Clustering of Interestingness Measures. Pages 290–297 of: *Proceedings of the 7th International Conference on Discovery Science, DS 2004*. Padova, Italy.

VASSILIADIS, PANOS, VAGENA, ZOGRAFOULA, SKIADOPOULOS, SPIROS, KARAYANNIDIS, NIKOS, & SELLIS, TIMOS K. 2001. ARKTOS: Towards the Modeling, Design, Control and Execution of ETL Processes. *Inf. Syst.*, **26**(8), 537–561.

VASSILIADIS, PANOS, SIMITSIS, ALKIS, GEORGANTAS, PANOS, & TERROVITIS, MANOLIS. 2003. A Framework for the Design of ETL Scenarios. Pages 520–535 of: *Proceedings of the 15th International Conference on Advanced Information Systems Engineering, CAiSE 2003*. Klagenfurt, Austria.

VERYKIOS, VASSILIOS S., MOUSTAKIDES, GEORGE V., & ELFEKY, MOHAMED G. 2003. A Bayesian Decision Model for Cost Optimal Record Matching. *VLDB J.*, **12**(1), 28–40.

WANG, KE, ZHOU, SENQIANG, YANG, QIANG, & YEUNG, JACK MAN SHUN. 2005. Mining Customer Value: from Association Rules to Direct Marketing. *Data Min. Knowl. Discov.*, **11**(1), 57–79.

WANG, RICHARD Y. 1998. A Product Perspective on Total Data Quality Management. *Commun. ACM*, **41**(2), 58–65.

WANG, RICHARD Y., STOREY, VEDA C., & FIRTH, CHRISTOPHER P. 1995. A Framework for Analysis of Data Quality Research. *IEEE Trans. Knowl. Data Eng.*, **7**(4), 623–640.

WANG, RICHARD Y., ZIAD, MOSTAPHA, & LEE, YANG W. 2002. *Data Quality*. Advances in Database Systems, vol. 23. Kluwer Academic Publishers.

WEIS, MELANIE, & MANOLESCU, IOANA. 2007. XClean in Action. Pages 259–262 of: *Proceedings of the 3rd Biennial Conference on Innovative Data Systems Research, CIDR 2007*. Asilomar, CA, USA.

WEIS, MELANIE, & NAUMANN, FELIX. 2004. Detecting Duplicate Objects in XML Documents. Pages 10–19 of: *Proceedings of the First International ACM SIGMOD 2004 Workshop on Information Quality in Information Systems, IQIS 2004*. Paris, France.

WEIS, MELANIE, NAUMANN, FELIX, & BROSY, FRANZISKA. 2006. A Duplicate Detection Benchmark for XML (and Relational) Data. In: *Proceedings of the 3rd International ACM SIGMOD 2006 Workshop on Information Quality in Information Systems, IQIS 2006*. Chicago, IL, USA.

WIDOM, JENNIFER. 2005. Trio: A System for Integrated Management of Data, Accuracy, and Lineage. Pages 262–276 of: *Proceedings of 2nd Biennial Conference on Innovative Data Systems Research.* Asilomar, CA, USA.

WIJSEN, JEF. 2003. Condensed Representation of Database Repairs for Consistent Query Answering. Pages 378–393 of: *Proceedings of 9th International Conference on Database Theory, ICDT 2003.* Siena, Italy.

WINKLER, WILLIAM E. 1999. *The State of Record Linkage and Current Research Problems.* Tech. Rept. Statistics of Income Division, Internal Revenue Service Publication R99 / 04. U.S. Bureau of the Census, Washington, DC, USA.

WINKLER, WILLIAM E. 2004. Methods for Evaluating and Creating Data Quality. *Inf. Syst.*, **29**(7), 531–550.

WINKLER, WILLIAM E., & THIBAUDEAU, YVES. 1991. *An Application of the Fellegi-Sunter Model of Record Linkage to the 1990 U.S. Decennial Census.* Tech. Rept. Statistical Research Report Series RR91 / 09. U.S. Bureau of the Census, Washington, DC, USA.

WU, KESHENG, OTOO, EKOW J., & SHOSHANI, ARIE. 2006. Optimizing Bitmap Indices with Efficient Compression. *ACM Trans. Database Syst.*, **31**(1), 1–38.

XIONG, MING, LIANG, BIYU, LAM, KAM-YIU, & GUO, YANG. 2006. Quality of Service Guarantee for Temporal Consistency of Real-Time Transactions. *IEEE Trans. Knowl. Data Eng.*, **18**(8), 1097–1110.

ZHANG, TIAN, RAMAKRISHNAN, RAGHU, & LIVNY, MIRON. 1996. BIRCH: An Efficient Data Clustering Method for Very Large Databases. Pages 103–114 of: *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data.* Montreal, Quebec, Canada.

ZHAO, XIAOFEI, & HUANG, ZHIQIU. 2006. A Formal Framework for Reasoning on Metadata Based on CWM. Pages 371–384 of: *Proceedings of 25th International Conference on Conceptual Modeling, ER 2006.* Lecture Notes in Computer Science, vol. 4215. Tucson, AZ, USA.

ZHU, YUNYUE, & SHASHA, DENNIS. 2002. StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time. Pages 358–369 of: *Proceedings of 28th International Conference on Very Large Data Bases, VLDB 2002.* Hong-Kong, China.

ZHUGE, YUE, GARCIA-MOLINA, HECTOR, & WIENER, JANET L. 1997. Multiple View Consistency for Data Warehousing. Pages 289–300 of: *Proceedings of the 13th International Conference on Data Engineering, ICDE 1997.* Birmingham, UK.

List of Figures

List of Tables