Information Quality Measurement in Data Integration Schemas

Maria da Conceição Moraes Batista, Ana Carolina Salgado
Centro de Informática, Universidade Federal de Pernambuco
Av. Professor Luis Freire s/n, Cidade Universitária
50740-540 Recife – PE, Brasil
Telephone Number: +55 81 2126.8430
{mcmb, acs}@cin.ufpe.br

ABSTRACT

Integrated access to distributed data is an important problem faced in many scientific and commercial applications. A data integration system provides a unified view for users to submit queries over multiple autonomous data sources. The queries are processed over a global schema that offers an integrated view of the data sources. Much work has been done on query processing and choosing plans under cost criteria. However, not so much is known about incorporating Information Quality analysis into data integration systems, particularly in the integrated schema. In this work we present an approach of Information Quality analysis of schemas in data integration environments. We discuss the evaluation of schema quality focusing in minimality, consistency and completeness aspects and define some schema transformations to be applied in order to improve schema design and consequently the quality of data integration query execution.

1. INTRODUCTION

Information quality (IQ) has become a critical aspect in organizations and, consequently, in Information Systems research. The notion of IQ has only emerged during the past ten years and shows a steadily increasing interest. IQ is a multidimensional aspect and it is based in a set of dimensions or criteria. The role of each one is to assess and measure a specific IQ aspect. One of these dimensions is the minimality aspect. This criterion defines that an element has good quality if it has no redundancies.

A data integration system based on a Global-as-view (GAV) approach [14] provides to users a unified view of several data sources, called *integrated schema*. In this kind of system, data is spread over multiple, distributed and heterogeneous sources and, consequently the query execution is an essential feature. To the best of our knowledge so far, not so much is known about the important problem of incorporating IQ aspects into usual data integration components and processes, like query results integration, schema maintenance, source selection, among others.

The primary contribution of this paper is the proposal of IQ criteria analysis in a data integration system, mainly related to the system's schemas.

The main goal we intend to accomplish is to improve the quality of query execution. Our hypothesis is that an acceptable alternative to optimize query execution would be the construction of good schemas, with high quality scores, and we have based our approach in this affirmative.

We focused our work in developing IQ analysis mechanisms to address schema generation and maintenance, specially the integrated schema. Initially we built a list of IQ criteria related to data integration aspects, but, due to space limitations, we decided to formally specify the algorithms and definitions of schema IQ criteria – *minimality*, *completeness* and *type consistency* – and guided the presented approach to this aspects. We also defined an algorithm to perform schema minimality improvements.

The paper is organized as follows: in section 2 we discuss approaches of Information Quality (IQ) and its use in data integration and schemas; the section 3 introduces the main issues related to schemas' IQ criteria; section 4 discusses the formalism of schema representation; in section 5 we present the formal specification of the chosen schemas IQ criteria and in section 6 we discuss some examples of these criteria; section 7 presents the schema improvement algorithm addressing minimality aspects and in section 8 is our concluding remarks and the final considerations about the mentioned topics.

2. APPROACHES OF IQ IN DATA INTEGRATION SYSTEMS

It has long been recognized that IQ is described or analyzed by multiple attributes or dimensions. During the past years, more and more dimensions and approaches were identified in several works ([11], [17]).

Naumann and Leser [17] define a framework addressing the IQ of query processing in a data integration system. This approach proposes the interleaving of query planning with quality considerations and creates a classification with twenty two dimensions divided into three classes: one related to the user preferences, the second class concerns the query processing aspects and the last one is related to the data sources.

Other relevant topic to consider in IQ and data integration is the set of quality criteria for schemas. These are critical due the importance of the integrated and data sources schemas for query processing. Some works are related to IQ aspects of schema equivalence and transformations. As in [2], where the authors exploit the use of normalization rules to improve IQ in conceptual database schemas.

VLDB '07, September 23-28, 2007, Vienna, Austria.

Some works are also relevant because are related to schema based data integration and schema correspondence definition as in [6], [15] and [1].

The work proposed by Herden [11] deals with measuring the quality of conceptual database schemas. In this approach, given a quality criterion, the schema is reviewed by a specialist in the mentioned criterion.

In [21] the authors propose IQ evaluation for data warehouse schemas focusing on the analyzability and simplicity criteria.

The work presented in [9] enhances our consistency point of view by identifying the measure, *column heterogeneity*, to quantify the data quality problems that can arise when merging data from different sources. Similarly, Halevy in [10] discusses the importance of detecting data types in handling schema heterogeneity for some tasks, including data integration
Our proposition is centered in IQ analysis for schemas in data integration systems with goals of query optimization and it is described in the next section.

3. SCHEMA QUALITY ISSUES

The main feature of a data integration system is to free the user from knowing about specific data sources and interact with each one. Instead, the user submits queries to an *integrated schema*, which is a set of views, over a number of data sources, designed for a particular data integration application. Each source must publish a data source schema with the representation of its contents. As a consequence, the data integration system must first reformulate a user query into queries that refers directly to schemas on the sources. To the reformulation step, a set of correspondences, called *schema mappings*, are required. There are also the user schemas that represent the requirements of information defined for one user or a group of users. The user requirements and their schema are not the focus of this work.

Commonly, the tasks of query processing involving query submission, planning, decomposition and results integration are performed by a software module called *mediator* [27].

As a starting point, we adopted IQ classifications proposed in previous works ([3], [17], [11], [20], [24], [25]) with discrete variations: some criteria are not considered (not applicable), and some were adapted to our environment. In our classification, the IQ aspects were adapted and associated to the main elements of a data integration system.

When considering any data integration task, component, process or element, (for example, a user query execution, data source selection or integrated schema generation), we perceive that each one can be associated with one of the three components: *data*, *schemas* and *data sources*. These components are the core of our IQ criteria classification. We classify as *data*, all the data objects that flow into the system. For example, query results, an attribute value, and so on. The *schemas* are the structures exported by the data sources (source schemas), the structures that are relevant for users to build queries (users' schema) and the mediation entities (integrated schema). The *data sources* are the origin of all data and schema items in the system.

All IQ criteria in the data integration system are associated with one of the three groups of elements according to the Table 1.

In this paper, we present our approach of schema maintenance with quality aspects, using three IQ criteria: *schema completeness*, *minimality* and *type consistency*.

Table 1. Data integration IQ criteria classification

Data Integration Element	IQ Criteria
Data Sources	Reputation, Verifiability, Availability, Response Time
Schema	Schema Completeness, Minimality, Type Consistency
Data	Data Completeness, Timeliness, Accuracy

Schema Completeness

The completeness can be measured as the percentage of real-world objects modeled in the integrated schema that can be found in the sources. Therefore, the *schema completeness* criterion is the number of concepts provided by the schema with respect to the application domain.

Minimality

Minimality is the extent in which the schema is compactly modeled and without redundancies. In our point of view, the minimality concept is very important to data integration systems because the integrated schema generated by the system may have redundancies. The key motivation for analyzing minimality is the statement that the more minimal the integrated schema is, the least redundancies it contains, and, consequently, the more efficient the query execution becomes [12]. Thus, we believe that our minimality analysis will help decreasing the extra time spent by mediator with access to unnecessary information represented by redundant schema elements.

Type Consistency

Type consistency is the extent in which the attributes corresponding to the same real world concept are represented with the same data type across all schemas of a data integration system. Table 2 lists each criterion with its definition and the metric used to calculate scores.

Table 2. IQ Criteria for schemas quality analysis

IQ Criteria	Definition	Metrics
Schema Completeness	The extent to which entities and attributes of the application domain are represented in the schema	1 – (#incomplete items / #total items)[1]
Minimality	The extent in which the schema is modeled without redundancies	1 – (#redundant schema elements/# total schema elements)[1]
Type Consistency	Data type uniformity across the schemas	1 – (#inconsistent schema elements / #total schema elements) [1]

The quality analysis is performed by a software module called *IQ Manager*. At the moment of integrated schema generation or update, this module proceeds with the criteria assessment and then, according to the obtained IQ scores, may execute adjustments over the schema to improve its design and, consequently, the query execution. This last step of schema tuning, after the IQ evaluation, is presented in section 7.

4. SCHEMA REPRESENTATION

Commonly, data integration systems use XML to represent the data and XML Schema to represent schemas. To provide a high-

level abstraction for XML schema elements, we use a conceptual data model, called X-Entity [15] described in what follows. We also present the schema mappings with this notation.

4.1 X-Entity Model

The X-Entity model is an extension of the Entity Relationship model [8], i.e., it uses some basic features of the ER model and extends it with some additional ones to better represent XML schemas.

The main concept of the model is the entity type, which represents the structure of XML elements composed by other elements and attributes. In the X-Entity model, relationship types represent element-subelement relationships and references between elements. An X-Entity schema S is denoted by $S=(E,R)$, where E is a set of entity types and R is a set of relationship types.

- *Entity type*: an entity type E, denoted by $E(\{A_1,\ldots,A_n\},\{R_1,\ldots,R_m\})$, is made up of an entity name E, a set of attributes $\{A_1,\ldots,A_n\}$ and a set of relationships $\{R_1,\ldots,R_m\}$. Each entity type may have attributes $\{A_1,\ldots,A_n\}$ that represents either a XML attribute or a simple XML element. In X-Entity diagrams, entity types are rectangles.[1]
- *Containment relationship*: a containment relationship between two entity types E and E_1, specifies that each instance of E contains instances of E_1. It is denoted by $R(E,E_1,(min,max))$, where R is the relationship name and (min,max) are the minimum and the maximum number of instances of E_1 that can be associated with an instance of E.
- *Reference relationship*: a reference relationship, denoted by $R(E_1,E_2,\ \{A_{11},\ldots,A_{1n}\},\{A_{21},\ldots,A_{2n}\})$, where R is the name of the relationship where the entity type E_1 references the entity type E_2. $\{A_{11},\ldots,A_{1m}\}$ and $\{A_{21},\ldots,A_{2n}\}$ are the referencing attributes between entities E_1 and E_2 such that the value of A_{1i}, $1 \leq i \leq n$, in any entity of E_1 must match a value of A_{2i}, $1 \leq i \leq n$, in E_2.

4.2 Schema Mappings

We use schema mappings to represent the correspondences between entities and attributes of distinct sources. Schema mappings, as defined in [22], are constraints used to assert that the semantics of some components in a schema is somehow related to the semantics of some components in another schema. In our approach, they are used to represent correspondences between X-Entity elements representing the same real world concept called *semantically equivalent*.

A data integration system is widely based on the existence of metadata describing individual sources and integrated schema, and on schema mappings [22] specifying correspondences between the integrated schema concepts and the source schemas concepts. There are several types of schema mappings to formally describe the associations between the concepts of X-Entity schemas. We consider an X-Entity element as an entity type, a relationship type or an attribute:

- **Entity schema mappings**: if E_1 and E_2 are entity types, the schema mapping $E_1 \equiv E_2$ specifies that E_1 and E_2 are *semantically equivalent*, i.e., they describe the same real world concept.
- **Attribute schema mappings**: are the mappings among attributes of semantically equivalent entities. The mapping $E_1.A_1 \equiv E_2.A_2$ indicates that the attributes A_1 and A_2 are semantically equivalent (correspond to the same real concept);
- **Path mappings**: specify special types of mappings between attributes and subentities of semantically equivalent entity types with different structures. Before defining a path mapping, it is necessary to define two concepts: *link* and *path*. A link between two X-Entity elements X_1 and X_2 ($X_1.X_2$) occurs if X_2 is an attribute of the entity type X_1, or X_1 is an entity of the relationship type X_2 (or vice-versa). If there is a multiple link, it is called a *path*. In this case it may occurs a normal path, $X_1.....X_n$ or an inverse path $(X_1.X_2.....X_n)^{-1}$. Entities attributes and relationships are represented by paths. A path mapping can occur in four cases as explained in the following (assuming P_1 and P_2 as being two paths):
  1. *Case 1*: $P_1=X_1.X_2...X_n$ and $P_2=Y_1.Y_2...Y_m$, where $X_1 \equiv Y_1$. The mapping $P_1 \equiv P_2$ specifies that the entity types X_n and Y_m are semantically equivalent.
  2. *Case2:* $P_1 = X_1.X_2...X_n.A$ and $P_2=Y_1.Y_2....Y_m.A'$, where $X_1 \equiv Y_1$. The mapping $P_1 \equiv P_2$ specifies that the attribute $A \in X_n$ and the attribute $A' \in Y_m$ are semantically equivalent.
  3. *Case 3*: $P_1 = X_1.X_2...X_n$ and $P_2 = (Y_1.Y_2...Y_n)^{-1}$, where $X_1 \equiv Y_n$. The mapping $P_1 \equiv P_2$ specifies that the entity types X_n and Y_1 are semantically equivalent.
  4. *Case 4*: $P_1 = X_1.X_2...X_n.A_k$ and $P_2 = (Y_1.Y_2...Y_n)^{-1}.A_k'$, where $X_1 \equiv Y_n$. The mapping $P_1 \equiv P_2$ specifies that the attribute $A_k \in X_n$ and the attribute $A_k' \in Y_1$ are semantically equivalent.

To illustrate the cases, consider the integrated and data source schemas presented in Figure 1.

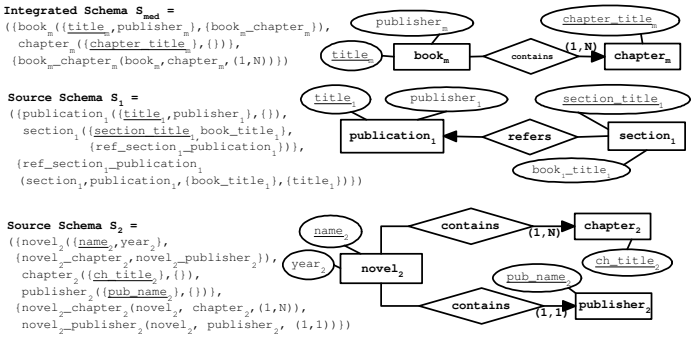


Figure 1. Integrated Schema (S_{med}) and Schemas of data sources (S_1 and S_2)

Table 3 presents the relevant schema mappings identified to compute $book_m$ and $chapter_m$. The mappings MP_1 to MP_{12} specify the semantic equivalences between the integrated and data source schema elements.

In data integration, the mappings are essential to assure the query processing over integrated schema. We assume that the mappings and schema elements equivalences are already defined automatically by the system or even manually by advanced users.

[1] **#** denotes the expression "*Number of*"

Particularly, in the environment exploited to experiment the proposed IQ evaluation ([5], [15]), there is a schema matcher component responsible to maintain equivalencies and define mappings among data sources and integrated schema.

Table 3. Schema mappings between the integrated schema S_{med} and the source schemas S_1 and S_2

MP_1: $book_m \equiv publication_1$
MP_2: $book_m.title_m \equiv publication_1.title_1$
MP_3: $book_m.publisher_m \equiv publication_1.publisher_1$
MP_4: $chapter_m \equiv section_1$
MP_5: $chapter_m.chapter\_title_m \equiv section_1.section\_title_1$
MP_6: $book_m.book_m\_chapter_m.chapter_m \equiv (section_1.section\_ref\_ publication_1.publication_1)^{-1}$
MP_7: $book_m \equiv novel_2$
MP_8: $book_m.title_m \equiv novel_2.name_2$
MP_9: $chapter_m \equiv chapter_2$
MP_{10}: $book_m.book_m\_chapter_m.chapter_m \equiv novel_2.novel_2\_chapter_2.chapter_2$
MP_{11}: $chapter_m.chapter\_title_m \equiv chapter_2.ch\_title_2$
MP_{12}: $book_m.publisher_m \equiv novel_2.novel_2\_publisher_2.publisher_2.pub\_name_2$

5. SCHEMA IQ CRITERIA

As previously mentioned, high IQ schemas are essential to accomplish our goal of improving integrated query execution.

It is important to notice that the proposed approach is not only to be applied in X-Entity schemas. The IQ aspects may be useful in any integrated schema to minimize problems acquired from schema integration processes, for example, the same concept represented more than once in a schema. The next section describes some definitions, required to introduce the minimality criterion.

It is important to point that semantic equivalence specification is not the focus of this paper. We use the approach presented in [15]. From now on, we assume that the integrated schema is already created and consequently, the equivalences between entities, attributes and relationships are already defined.

5.1 Definitions

More formally, a data integration system is defined as follows:

Definition 1 – Data Integration System (Ð)

A data integration system is a 4 element tuple, $Ð = <\delta, S_m, \rho, \varphi(Ð)>$ where:

- δ is the set of S_i data sources schemas, i.e. $\delta = <S_1, S_2, \ldots, S_w>$, where w is the total number of data sources participating in Ð;
- S_m is the integrated schema, generated by internal modules of Ð;
- ρ is the set of user schemas, $\rho = <U_1, U_2, \ldots, U_u>$ where *u* is the total number of users of Ð. Together with the data source schemas it is the basis of the integrated schema generation;
- The component $\varphi(Ð)$ is the set of all distinct concepts in the application domain of the data integration system, as stated in the next definition. This set can be extracted from the schema mappings between the data source schemas and the integrated schema.

In Ð, the following statements are true:

- S_m is a X-Entity integrated schema such as $S_m = <E_1, E_2, \ldots, E_{n_m}>$ where E_k is an integration or mediation entity $(1 \le k \le n_m)$, and n_m is the total number of entities in S_m;
- $\forall E_k \in S_m$, $E_k(\{A_{k1}, A_{k2}, \ldots, A_{ka_k}\}, \{R_{k1}, R_{k2}, \ldots, R_{kr_k}\})$, where:
  - $\{A_{k1}, A_{k2}, \ldots, A_{ka_k}\}$ is the set of attributes of E_k, a_k is the number of attributes in E_k, $(a_k > 0)$;
  - $\{R_{k1}, R_{k2}, \ldots, R_{kr_k}\}$ is the set of relationships of E_k, r_k is the number of relationships in E_k $(r_k \ge 0)$.
- If X_1 and X_2 are schema elements (attributes, relationships or entities), the schema mapping $X_1 \equiv X_2$ specifies that X_1 and X_2 are *semantically equivalent*, i.e., they describe the same real world concept and have the same semantics. More details about the definition of semantic equivalences can be found in [15].

Every information system (even a data integration one) is constructed from a number of requirements. Moreover, embedded in this set of requirements is the application domain information [13], very important to schemas construction.

Definition 2 – Domain Concepts Set

We define φ as the set of domain concepts, $\varphi(\beta) = <C_1, C_2, \ldots, C_{\sigma_\beta}>$:

β is even a given integrated schema S_m or a data integration system Ð;

σ_β as the number of real world concepts in β;

C_k is an application domain concept which is represented by an schema element Y in one of the two following ways:

i) $Y \in S_m$, if β is a integrated schema or;

ii) $Y \in \delta = <S_1, S_2, \ldots, S_w>$, if β is a the data integrated system.

Usually, the data integration system has mechanisms to generate and maintain the schemas. It is very difficult to guarantee that these mechanisms, specifically those concerning the schema generation, produce schemas without anomalies, e.g., redundancies. In data integration context, we define a schema as *redundant* if it has occurrences of redundant entities, attributes or relationships.

To contextualize schema aspects, we introduce the definitions 3 to 6.

Definition 3 – Redundant attribute in a single entity

An attribute A_{ki} of entity E_k, $A_{ki} \in \{A_{k1}, A_{k2}, \ldots, A_{ka_k}\}$ is *redundant*, i.e., $\mathbf{Red(A_{ki}, E_k) = 1}$, if:

$$\exists E_k.A_{kj},\ j \ne i,\ A_{kj} \in \{A_{k1}, A_{k2}, \ldots, A_{ka_k}\}$$
$$\text{such as } E_k.A_{ki} \equiv E_k.A_{kj},\ 1 \le i,j \le a_k$$

Definition 4 – Redundant attribute in different entities

An attribute A_{ki} of the entity E_k, $A_{ki} \in \{A_{k1}, A_{k2}, \ldots, A_{ka_k}\}$ is redundant, i.e. $\mathbf{Red(A_{ki}, E_k) = 1}$, if:

$$\exists E_o,\ o \ne k,\ E_o \in S_m,\ E_k \equiv E_o,$$
$$E_o(\{B_{o1}, B_{o2}, \ldots, B_{oa_o}\}),\ B_{oj} \text{ are attributes of } E_o$$
$$\text{and } \exists E_o.B_{oj},\ B_{oj} \in \{B_{o1}, B_{o2}, \ldots, B_{oa_o}\}$$

such as $E_k.A_{ki} \equiv E_o.B_{oj}$,

$1 \le i \le a_k$, $1 \le j \le a_o$.

If an attribute A_{ki}, $A_{ki} \in \{A_{k1}, A_{k2}, \ldots, A_{ka_k}\}$, and **$Red(A_{ki}, E_k) = 0$**, we say that A_{ki} is *non-redundant*.

Definition 5 – Entity Redundancy Degree

We say that a given entity E_k has a positive redundancy degree in schema S_m, i.e. $Red(E_k, S_m) > 0$, if E_k has at least one redundant attribute. The redundancy degree is calculated by the following formula:

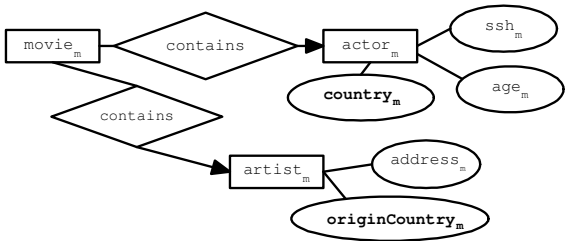
$$Red(E_k, S_m) = \frac{\sum_{i=1}^{a_k} Red(A_{ki}, E_k)}{a_k}$$, where

$\sum_{i=1}^{a_k} Red(A_{ki}, E_k)$ is the number of redundant attributes in E_k and

a_k is the total number of attributes in E_k.

An entity E_k is considered fully redundant when all of its attributes are redundant, i.e. **$Red(E_k, S_m) = 1$**. In this case, we assume that the entity E_k may be removed from the original schema S_m without lost of relevant information. Any existing relationship from E_k may be associated to a remaining equivalent entity E_o, as will be shown in section 7.

As an example of redundant attributes and the entity redundancy degree, suppose the schema and mappings of Figure 2:



$artist_m \equiv actor_m$

$originCountry_m \equiv country_m$

Figure 2. Schema with redundant attributes

The attribute $originCountry_m$ in $artist_m$ is redundant because it has a semantic correspondent in the entity $actor_m$ (attribute $country_m$), and both the entities $artist_m$ and $actor_m$ are semantically equivalent. Thus, we have the following:

$Red(originCountry_m, artist_m) = 1$

$Red(address_m, artist_m) = 0$

$Red(country_m, actor_m) = 0$

$Red(age_m, actor_m) = 0$

$Red(ssh_m, actor_m) = 0$

It is interesting to mention that $nationality_m \equiv country_m$, but only the first is classified as redundant. This occurs because only one must be marked as redundant and removed, while the other has to be kept in the schema to assure that domain information will not be lost.

The entities in Figure 2 have the following entity redundancy degrees in schema S_m:

$$Red(artist_m, S_m) = \frac{1 + 0}{2} = 0.5$$

$$Red(actor_m, S_m) = \frac{0 + 0 + 0}{3} = 0$$

The entity $artist_m$ is 50% redundant because it has only two attributes, and one of them is redundant.

Definition 6 – Redundant Relationship

Consider a relationship $R \in S_m$ between the entities E_k and E_y represented by the path $E_k.R.E_y$, then: $R \in \{R_{k1}, \ldots, R_{kr_k}\}$ and $R \in \{T_{y1}, \ldots, T_{yr_y}\}$,where $\{R_{k1}, \ldots, R_{kr_k}\}$ is the set of relationships of the entity E_k and $\{T_{y1}, \ldots, T_{yr_y}\}$ is the set of relationships of the entity E_y. Thus, the relationship R connects E_k and E_y if and only if $R \in \{R_{k1}, \ldots, R_{kr_k}\}$ and $R \in \{T_{y1}, \ldots, T_{yr_y}\}$.

We define R as a *redundant relationship* in S_m, i.e. **$Red(R, S_m) = 1$** if:

$\exists P_1$, where $P_1 = E_k.R_j.\ldots.T_s.E_y$ is a path with

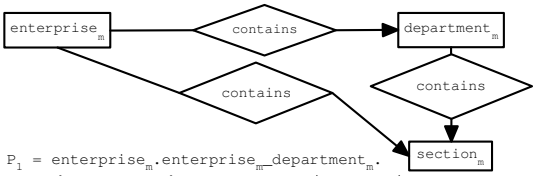
$R_j \in \{R_{k1}, \ldots, R_{kr_k}\}$ and

$T_s \in \{T_{y1}, \ldots, T_{yr_y}\}$

such as $P_1 \equiv R$.

In other words, a relationship between two entities is redundant if there are other semantically equivalent relationships which paths are connecting the same two entities.

Consider the schema and mappings illustrated in Figure 3.



$P_1 = enterprise_m.enterprise_m\_department_m.$
$department_m.department_m\_section_m.section_m$

$P_2 = enterprise_m.enterprise_m\_section_m.section_m$

$P_1 \equiv P_2$

Figure 3. Schema with redundant relationship

The relationship connecting $enterprise_m$ and $section_m$ is redundant ($Red(enterprise_m, section_m, (1,N)) = 1$) because it has a semantically equivalent correspondent represented by P_1.

We agree with the study presented in [28], where Zhang states that redundancy is not a symmetric metric. An entity E_j may cause E_k to be assigned as redundant, but if the comparison order is changed, E_j may not be redundant when related to E_k. A simple example is the case where an entity E_1 is entirely contained in E_2, E_1 is redundant but E_2 is not.

5.2 Minimality

A major problem of conceptual schema design is to avoid the generation of a schema with redundancies. A schema is minimal if all of the domain concepts relevant for the application are described only once. In other words, a minimal schema may represent each application requirement only once ([12], [23], [16], [19]). Thus, we can say that the minimality of a schema is the degree of absence of redundant elements in the schema. Likewise our point of view, Kesh [12] argues that a more minimal

(or *concise*) schema will make itself more efficient, and consequently improve the effectiveness of operations and queries over it. We state that if the integrated schema is minimal, query execution will be improved. Redundant elimination (or minimality increasing) avoids the query processor to spend extra time querying redundant elements.

Therefore, to measure the minimality, we must first determine the redundancy degree of the schema. To each one of the next redundancy definitions (6 and 7), we assume the following:

- n_{rel} is the total number of relationships in S_m;
- n_m is the total number of entities in S_m;
- r_k is the number of relationships of each entity E_k in S_m.

Definition 7 – Entity Redundancy of a Schema

The total entity redundancy of a schema S_m is computed by the formula:

$$ER(S_m) = \frac{\sum_{k=1}^{n_m} Re\,d(E_k, S_m)}{n_m}, \text{ where } Red(E_k, S_m) \text{ is the redundancy degree of each } E_k \text{ in } S_m.$$

Definition 8 – Relationship Redundancy of a Schema

The relationship redundancy degree of S_m is measured by the equation:

$$RR(S_m) = \frac{\sum_{1=1}^{n_{rel}} \#\,Re\,d(R_1, S_m)}{n_{rel}}, \text{ where } \sum_{1=1}^{n_{rel}} \#\,Re\,d(R_1, S_m) \text{ is the number of redundant relationships in } S_m \text{ as stated in Definition 6.}$$

Definition 9 – Schema Minimality

We define the overall redundancy of a schema in a data integration system as the sum of the aforementioned redundancy values: entities (ER) and relationships (RR). The schema minimality is measured by the formula:

$$Mi_{S_m} = 1 - [ER(S_m) + RR(S_m)]$$

5.3 Schema Completeness

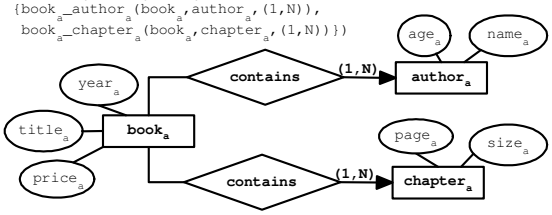
The schema completeness is the percentage of domain concepts represented in the integrated schema when related to the concepts represented in all data source schemas ([12], [23], [16], [19]). For instance, suppose that a given data integration system has a total of 10 distinct domain concepts (described by entities and relationships) in all data sources' published schemas. If the integrated schema have only 6 representations of these concepts, we can say that the integrated schema is 60% complete related to the current set of data sources.

To introduce schema completeness metrics, we assume that the data integration system (**Ð**) has a minimal integrated schema S_m.

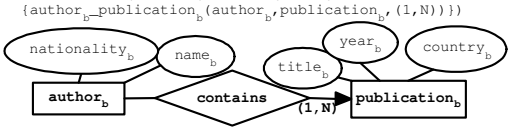
As mentioned in Definition 2, a data integration system is constructed over a domain, and this domain aggregates a number of real world concepts (obtained from the requirements) that are represented by the schemas elements. Every relationship, entity or attribute in a user, data source or integrated schema are (part of) a real world concept. Sometimes, the same concept can be found in more than one element, in one or more schemas. In this case the concept is replicated and the corresponding schema elements are semantically equivalent. For example, in a given schema ***A*** the name of a movie director can be an attribute `director.name` and the same concept may be the element `movie.director` (i.e., `director.name ≡ movie.director`) in other schema ***B***.

Let consider **Ð** with two data sources (S_a and S_b) and its respective schemas and mappings as in Figure 4.

Schema of data source S_a =
({book$_a$({title$_a$,price$_a$,year$_a$},{book$_a$_author$_a$,book$_a$_chapter$_a$}),
author$_a$({name$_a$, age$_a$},{}),
chapter$_a$({size$_a$,page$_a$},{})},
{book$_a$_author$_a$(book$_a$,author$_a$,(1,N)),
book$_a$_chapter$_a$(book$_a$,chapter$_a$,(1,N))})



Schema of data source S_b =
({author$_b$({name$_b$,nationality$_b$},{author$_b$_publication$_b$}),
publication$_b$({title$_b$,country$_b$,year$_b$},{ })},
{author$_b$_publication$_b$(author$_b$,publication$_b$,(1,N))})



Schema Mappings =
MP$_1$: book$_a$.year$_a$ ≡ publication$_b$.year$_b$
MP$_2$: book$_a$.title$_a$ ≡ publication$_b$.title$_b$
MP$_3$: author$_a$.name$_a$ ≡ author$_b$.name$_b$
MP$_4$: book$_a$.book$_a$_author$_a$.author$_a$ ≡
(author$_b$.author$_b$_publication$_b$.publication$_b$)$^{-1}$

Figure 4. Data source schemas and schema mappings in Ð

With respect to the schemas in Figure 4, after investigating and analyzing the schemas and mappings, it is possible to find the set of distinct concepts ($\varphi(Ð) = \langle C_1, C_2, \ldots, C_{\sigma_Ð} \rangle$) as in the following:

$\varphi(Ð) =$ <book, book.year, book.title, book.price, author, author.age, author.name, chapter, chapter.page, chapter.size, book.country,
(book,author,1,N),(book,chapter,1,N)>

$\Rightarrow \sigma_Ð = 13$, where

$\sigma_Ð$ is the number of concepts in **Ð** (Definition 2)

It is important to emphasize that the relationship `(author, publication,1,N)` is not present in the domain set of concepts because it is equivalent to the relationship `(book,author,1,N)` e.g., both represents the same concept.

To determine the *schema completeness* metrics, we introduce the following definition.

Definition 10 – Schema Completeness

The overall schema completeness degree in a given schema $S_x \in$ **Ð** is obtained by the following average:

$$SC(S_x) = \frac{\sigma_{S_x}}{\sigma_Ð},$$

where S_x can be either a data source schema or the integrated schema;

σ_{S_x} is the number of distinct concepts in the schema S_x and;

$\sigma_{Đ}$ is the is the number of distinct concepts contained in all the schemas of the data integration system **Đ**.

5.4 Type Consistency

In databases, the consistency property states that only valid data will be written to the database. The stored data must adhere to a number of consistency rules. If, for some reason, a transaction is executed that violates the database's consistency rules, the entire transaction will be rolled back and the database will be restored to a consistent state according to those rules. On the other hand, if a transaction successfully executes, it will take the database from a consistent state with the rules to another state that is also consistent with the rules. These affirmatives are related to data consistency, but they can be extended to adequately represent data type consistency constraints [20].

A data type is a constraint placed upon the interpretation of data in a type system in computer programming. Common types of data in programming languages include primitive types (such as integers, floating point numbers or characters), tuples, records, algebraic data types, abstract data types, reference types, classes and function types. A data type describes representation, interpretation and structure of values manipulated by algorithms or objects stored in computer memory or other storage device. The type system uses data type information to check correctness of computer programs that access or manipulate the data [7].

When an integrated schema management system experiences problems with consistency, the same data type of information is recorded in more than one way. The first step in resolving this consistency problem is to determine which alternative data type is preferable. This approach would then be defined as the standard, namely, the accepted way of recording the information.

In this case, a schema element is called consistent if it adheres to the standard data type. If it not adheres, commonly the data type conversion is a difficult process performed by mediator, and then achieving consistency could be both time-consuming and expensive. As in [4] we have based the consistency metric in an essential factor: the number of semantically equivalent attributes in schema that adhere to the standard data type defined for the attribute.

We approximate consistency rules and data types to create the *Type Consistency* IQ concept. The use of different coding schemes to record the same attribute falls into the lack of IQ in this category. We use the *Type Consistency* criterion to investigate which data elements in the schemas are always represented with the same type, or adhering to a consistency standard. This is an indicator of quality and query improvement, once the query processor is not going to perform type conversions for a schema element in order to access its correspondences in data sources schemas. For type consistency measurement, we use a metric similar to the one presented in [24].

X-Entity is a high level abstraction for XML schema structures, it is necessary to define the concept of *type* for an X-Entity attribute.

Definition 11 – X-Entity Attribute Data Type

A data type T_{kj} for the attribute A_{kj}, where $A_{kj} \in E_k$, is a domain element or structural metadata associated with the attribute data as defined in previous works [7]. As the data integration system is concerned with XML data, then every T_{kj} may be one of the valid *datatypes* defined for XML Schema (including the user defined ones). From the XML Schema specification [18], we import the concept of *datatype* as follows:

A *datatype* T is a tuple $<\alpha, \lambda, \gamma>$, consisting of:

- α is a set of distinct values, called the *value space,* containing the values that the elements of the type can have**;**
- λ is a set of lexical representations, called the *lexical space* and;
- γ is a set of facets that characterize properties of the value space, individual values or lexical items;
- **T ∈ £,** where **£** is the set of all XML schema datatypes.

In our work, to use datatypes, it is only necessary to refer to the α set of valid values in the datatype specification.

To determine the type *consistency* criterion, we define the following:

Definition 12 – Attribute

$\forall E_k \in S_m$, every attribute A_{kj} $(A_{kj} \in E_k)$ is defined by the tuple (T_{kj}, v_{kj}), where:

- $T_{kj} = <\alpha_{kj}, \lambda_{kj}, \gamma_{kj}>$ is the datatype of attribute A_{kj} $(1 \le j \le a_k)$;
- v_{kj} is the value of attribute A_{kj} $(1 \le j \le a_k)$ and $v_{kj} \in \alpha_{kj}$.

Definition 13 – Data Type Consistency Standard

The data type consistency standard is the alternative data type which is more appropriate to an attribute. This data type is defined as the standard, namely, the accepted way of recording the attribute. Formally, a data type consistency standard is an X-Entity attribute data type such as:

$\forall E_k.A_{kj}, E_k \in$ **Đ**, $A_{kj} \in \{A_{k1}, A_{k2}, \ldots, A_{ka_k}\}$ ^

$\exists \mathbf{T_{std}}, \mathbf{T_{std}} = <\alpha_{std}, \lambda_{std}, \gamma_{std}>$ ^

$\exists E_x.A, E_x \in$ **Đ** ^ $E_x.A \equiv E_k.A_{kj}$

A = (T_{std}, v) where $\mathbf{T_{std}}$ is the most frequently data type used in Đ for attribute A and its equivalents

Definition 14 – Attribute Type Consistency

In a given a set of data source schemas S_i $(1 \le i \le w)$ and a mediation schema S_m, we say that an attribute $A_{pj} = (T_{pj}, v_{pj})$ (T_{pj} is a valid datatype as in Definition 12) of an entity $E_p \in S_p$ ($S_p = S_i$ or $S_p = S_m$) is ***consistent*** i.e. **Con(A_{pj}, S_p) = 1** if it appears in another entity or even in the same entity with other datatype: $\exists T_{std} \in$ **Đ**, **T ∈ £** such as $A_{pj} = (T_{std}, v_{pj})$

Definition 15 – Schema Data Type Consistency

The overall schema type consistency score in a given data integration system $(Con(S_m, Đ))$ is obtained by the following calculation:

$$Con(S_m, Đ) = \frac{\sum_{k=1}^{n_m} \sum_{j=1}^{a_k} Con(A_{kj}, Đ)}{\sum_{k=1}^{n_m} a_k},$$

where

$\sum_{k=1}^{n_m} \sum_{j=1}^{a_k} Con(A_{kj}, Đ)$ is the total number of consistent attributes in **Đ**;

$A_{kj} \in$ **Đ**;

n_m is the total number of entities in the schema **Đ**;

a_k is the number of attributes of the entity E_k.

6. Examples

In this section we present practical examples of proposed criteria evaluation in schemas. For each one of the IQ criteria, one schema with anomalies in the referred aspect is presented, and the evaluation process is detailed.

6.1 Minimality Analysis

Consider the redundant schema of Figure 5 for minimality example.

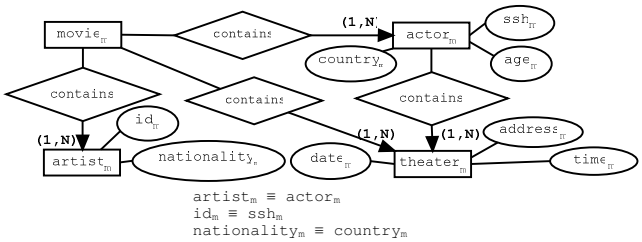


Figure 5. Schema with redundant elements

The entity $artist_m$, is redundant because it is semantically equivalent to $actor_m$ and all its attributes have a semantically equivalent correspondent in $actor_m$.

The relationship $movie_m\_artist_m(movie_m, artist_m, (1,N))$ is also redundant because it has a semantically equivalent relationship $movie_m\_actor_m(movie_m, actor_m, (1,N))$ and $actor_m \equiv artist_m$.

The schema minimality value will be obtained as illustrated in Figure 6. The minimality of schema S_m is 75%, what means that the schema has 25% of redundancy that can possibly be eliminated.

```
Red(movie_m,S_m)    = 0
Red(actor_m,S_m)    = 0
Red(theater_m,S_m)  = 0
Red(artist_m,S_m)   = 1
ER(S_m)= 1/(4 + 4) = 0,125
RR(S_m)= 1/(4 + 4) = 0,125
Mi(S_m)= 1 -(0,125 + 0.125) = 0,75
```

Figure 6. Schema minimality score

6.2 Schema Completeness Analysis

For the completeness evaluation, consider the integrated schema presented in Figure 7 and the data source schemas presented in Figures 8 to 10.

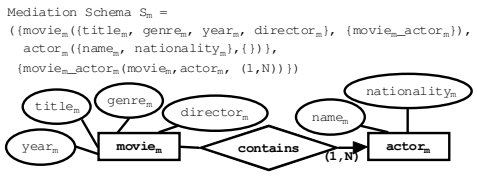


Figure 7. Integrated schema S_m

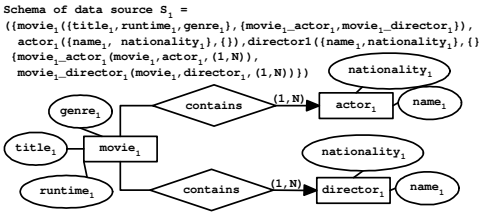


Figure 8. Data Source schema S_1

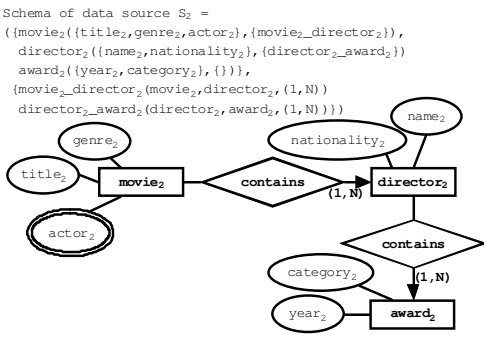


Figure 9. Data Source schema S_2

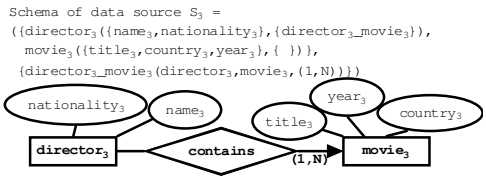


Figure 10. – Data Source schema S_3

The schema mappings between the schemas are in Table 4.

Table 4. Schema mappings between the integrated schema S_m and the source schemas S_1 , S_2 and S_3

SM_1: $movie_m \equiv movie_1$
SM_2: $movie_m \equiv movie_2$
SM_3: $movie_m \equiv movie_3$
SM_4: $movie_m.genre_m \equiv movie_1.genre_1$
SM_5: $movie_m.genre_m \equiv movie_2.genre_2$
SM_6: $movie_m.title_m \equiv movie_1.title_1$
SM_7: $movie_m.title_m \equiv movie_2.title_2$
SM_8: $movie_m.title_m \equiv movie_3.title_3$
SM_9: $movie_m.year_m \equiv movie_3.year_3$
SM_{10}: $movie_m.director_m \equiv director_1.name_1$
SM_{11}: $movie_m.director_m \equiv director_2.name_2$
SM_{12}: $movie_m.director_m \equiv director_3.name_3$
SM_{13}: $movie_m.movie_m\_actor_m.actor_m \equiv movie_1.movie_1\_actor_1.actor_1$
SM_{14}: $movie_m.movie_m\_actor_m.actor_m.name_m \equiv movie_1.movie_1\_actor_1.actor_1.name_1$
SM_{15}: $movie_m.movie_m\_actor_m.actor_m.name_m \equiv movie_2.actor_2$
SM_{16}: $movie_m.movie_m\_actor_m.actor_m.nationality_m \equiv movie_1.movie_1\_actor_1.actor_1.nationality_1$

Analyzing and compiling the schemas and mappings, it is possible to say that **Ð** has the following set of distinct concepts (**φ(Ð)**):

φ(Ð) = <movie,
movie.year,movie.title, movie.genre,
movie.runtime, director,
director.name, director.nationality,
actor, actor.name, actor.nationality,
award, award.category, award.year,
(movie,actor,1,N),
(movie,director,N,N),
(director,award,1,N)>

⇒ $\sigma_{Ð}$ = 17

Analogously, examining the integrated schema of Figure 7, it is possible to identify the following concepts:

φ(S_m) = <$movie_m$, $movie_m.title_m$,
$movie_m.genre_m$, $movie_m.year_m$,
$movie_m.director_m$, $actor_m$, $actor_m.name_m$,
$actor_m.nationality_m$,
($movie_m$,$actor_m$,1,N)>

⇒ σ_{S_m} = 9

Thus, for our example the overall score of S_m (Figure 7) schema completeness will be obtained as follows:

$$SC(S_m) = \frac{\sigma_{S_m}}{\sigma_{Ð}} = \frac{9}{17} = 0,5294$$

Therefore, the completeness of S_m is 52,94%, what means that the schema has a 47,06% of the domain concepts missing in the integrated schema. Improvements in schema completeness can be done by the insertion of a set of tasks to investigate the data source schemas seeking for concepts that are not in the current integrated schema. After that, the system must generate schema mappings and propagate the new concepts converted into entities and relationships to the integrated schema. This can be done, for example, by applying the techniques presented in [15].

6.3 Type Consistency Analysis

To an example of type consistency evaluation, assume an hypothetic schema with the following attribute equivalencies:

SM_1: $actor_m.birthdate_m \equiv actor_1.birth_1$
SM_2: $movie_m.birthdate_m \equiv actor_2.birth_2$
SM_3: $movie_m.birthdate_m \equiv actor_3.bd_3$

Suppose that the data type of the attribute $actor_1.birth_1$ is `String` and the data type of attributes $actor_m.birthdate_m$, $actor_2.birth_2$ and $actor_3.bd_3$ is `Date`. We have three `Date` occurrences versus one single occurrence of `String` data type for the same attribute. Thus, the *IQ Manager* will consider the data type `Date` as the data type consistency standard:

T_{std} = Date
$Con(actor_m.birthdate_m,Ð) = 1$
$Con(actor_1.birth_1,Ð) = 0$
$Con(actor_2.birth_2,Ð) = 1$
$Con(actor_3.bd_3,Ð) = 1$

The attributes of type Date are consistent and the attribute of type String is inconsistent. To compute the consistency degree of a given schema it is necessary to sum the consistency values of each attributes in the schema, dividing the result by the total number of attributes as stated in Definition 15.

7. SCHEMA MINIMALITY IMPROVEMENT

After detecting the schema IQ anomalies, it is possible to restructure it to achieve better IQ scores [2]. In order to improve minimality scores, redundant elements must be removed from the schema. In this section, we present an algorithm with schema improvement actions to be executed after the integrated schema generation or update. The sequence of steps is specified in the algorithm of Table 5.

It is important to declare that we can accomplish a *total minimality* schema score, or a schema with no redundancies, by removing redundant elements until the value of minimality equal to 1 is achieved.

Table 5. Schema adjustment algorithm

1	Calculate minimality score and if minimality = 1, then stop;
2	Search for fully redundant entities in S_m;
3	If there are fully redundant entities then eliminate the redundant entities from S_m;
4	Search for redundant relationships in S_m;
5	If there are redundant relationships then eliminate the redundant relationships from S_m;
6	Search for redundant attributes in S_m;
7	If there are redundant attributes then eliminate the redundant attributes from S_m;
8	Go to Step 1

The detection of redundant elements processes in steps 2, 4 and 6. are already described in previous definitions. The next sections describe the proposed redundancies elimination actions executed in steps 3, 5 and 7 of the improvement algorithm.

In the following, we present details about schema adjustments, performed when the *IQ Manager* has to remove redundant elements.

7.1 Redundant Entities Elimination

It is important to point that, after removing a redundant entity E, its relationships must be relocated to a semantic equivalent remaining entity.

When removing a redundant entity E_1 ($E_1 \equiv E_2$), the *IQ Manager* transfers the relationships of E_1 to the remaining equivalent entity E_2. Three different situations may occur when moving a relationship R_x, $R_x \in E_1$:

- If $R_x \in E_2$ then R_x is deleted because it is no longer necessary;
- If $R_x \notin E_2$ but $\exists R_y$, $R_y \in E_2$ such as $R_x \equiv R_y$ then R_x is deleted;
- If $R_x \notin E_2$ and there is no R_y, $R_y \in E_2$ such as $R_x \equiv R_y$, then R_x is connected to E_2.

The first and second situations are not supposed to cause any schema modification besides the entity deletion. However, the third case needs more attention, once the redundant relationships of the removed entity have to be relocated.

Definition 16 – Substitute Entity:

We say that E_k is a fully redundant entity, if and only if $Red(E_k, S_m) = 1$ and E_k has at least one *Substitute Entity* E_s, i.e. $Subst(E_k) = E_s$, such as:

- E_k $(\{A_{k1}, \ldots, A_{ka_k}\}, \{R_{k1}, \ldots, R_{kr_k}\})$ A_{kx} are attributes and R_{ky} are relationships of E_k and;
- E_s $(\{A_{s1}, \ldots, A_{sa_s}\}, \{R_{s1}, \ldots, R_{sr_s}\})$ A_{sz} are attributes and R_{st} are relationships of E_s and
- $E_k \equiv E_s$ and $\forall E_k.A_{ki} \in \{A_{k1}, \ldots, A_{ka_k}\}$,
- $\exists E_s.A_{sj} \in \{A_{s1}, \ldots, A_{sa_s}\}$ with $E_k.A_{ki} \equiv E_s.A_{sj}$, $1 \leq i,j \leq a_k$

The Definition 16 states that an entity E_k is considered fully redundant when all of its attributes are redundant ($Red(E_k, S_m) = 1$) and it must have a substitute entity E_s in S_m. All the attributes of E_k are contained in E_s. In this case, E_k may be removed from the original schema S_m without lost of relevant information if it is replaced by its *substitute entity* E_s. Any existing relationship from E_k may be associated to E_s, as stated in the following definition.

Definition 17 – Relationship Relocation:

In a schema S_m, if $Subst(E_k) = E_s$, then E_k can be eliminated from S_m. In this case, in order to do not lose any information, E_k relationships are relocated to E_s according to the following rules, i.e. $\forall E_k.R_{kj}$:

i. If $E_k.R_{kj} \in \{R_{s1}, \ldots, R_{sr_s}\}$ then R_{kj} must be deleted because it is no longer useful;

ii. If $E_k.R_{kj} \notin \{R_{s1}, \ldots, R_{sr_s}\}$ but $\exists E_s.R_{sp}$, such that $E_k.R_{kj} \equiv E_s.R_{sp}$ then $E_k.R_{kj}$ must be deleted because it has an equivalent relationship in E_s;

iii. If $E_k.R_{kj} \notin \{R_{s1}, \ldots, R_{sr_s}\}$ and $\nexists E_s.R_{sp}$ such as $E_k.R_{kj} \equiv E_s.R_{sp}$ then, E_s is redefined as $E_s = (\{A_{s1}, \ldots, A_{sa_s}\}, \{R'_{s1}, \ldots, R'_{sr_s}\})$, A_{sz} are attributes and R'_{st} are relationships of E_s and $\{R'_{s1}, \ldots, R'_{sr_s}\} = \{R_{s1}, \ldots, R_{sr_s}\} \cup R_{kj}$.

The first and second case above do not imply in schema relevant changes, only the relationship removal. The third one, where the relationship relocation occurs, can be exemplified in Figures 11 and 12.

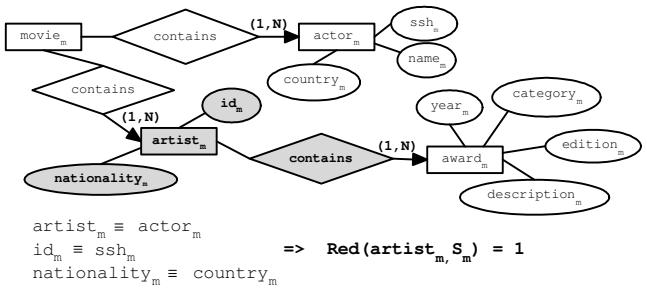


Figure 11. Redundant entity detection

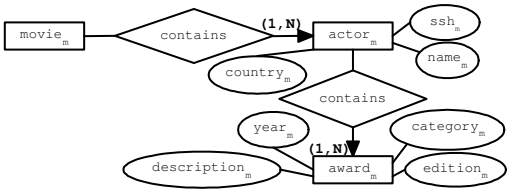


Figure 12. Relationship relocation

The fully redundant entity $artist_m$ (with its attributes) is removed and it is substituted by the semantically equivalent $actor_m$. Consequently, the relationship $\{movie_m\_artist_m(movie_m, artist_m, (1,N))\}$ may be deleted because it can be replaced by the remaining equivalent relationship $\{movie_m\_actor_m(movie_m, actor_m, (1,N))\}$.

The relationship $\{artist_m\_award_m(artist_m, award_m, (1,N))\}$ is relocated to $actor_m$, turning into the new relationship $\{actor_m\_award_m(actor_m, award_m, (1,N))\}$. With this operations, it is possible to obtain a no redundant schema as illustrated in Figure 12.

7.2 Redundant Relationships Elimination

After removing redundant entities and possibly performing the necessary relationship relocations, the *IQ Manager* discovers remaining redundant relationships to eliminate them. This can be accomplished by merely deleting from the schema, the relationships identified as redundant. Considering the example or Figure 13, the relationship $\{enterprise_m\_section_m(enterprise_m, section_m, (1,N))\}$ is redundant because it has a semantically equivalent correspondent represented by P_1.

After eliminating the relationship $\{enterprise_m\_section_m(enterprise_m, section_m, (1,N))\}$, the schema with no relationship redundancies is showed in Figure 14.

It is important to note that the remaining schema after the relationship eliminations, do not lose relevant information. Instead, without redundancies, it has better IQ scores, and consequently it is more usefulness to assist the query processing.

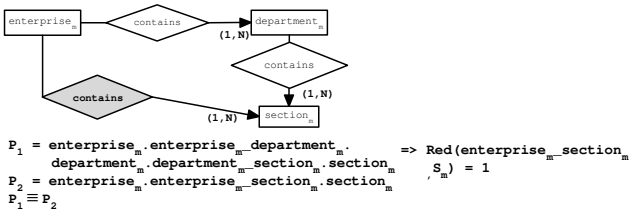


Figure 13. Redundant relationship detection

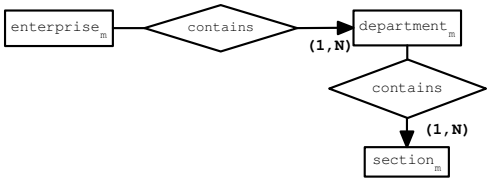


Figure 14. Redundant relationship elimination

7.3 Redundant Attributes Elimination

The last step of schema improvement algorithm consists in investigating and eliminating remaining redundant attributes in schema. Similarly to the redundant relationships removal step, these attributes may merely be deleted from schema. This occurs because the schema always has semantically equivalent attributes to substitute the redundant ones. In Figure 15, the attribute $\texttt{nationality}_m$ is removed because there is a semantically equivalent attribute $\texttt{country}_m$, which will substitute it.

After executing the schema improvement steps, the *IQ Manager* can recalculate and analyze minimality scores in order to determine if the desired IQ is accomplished.

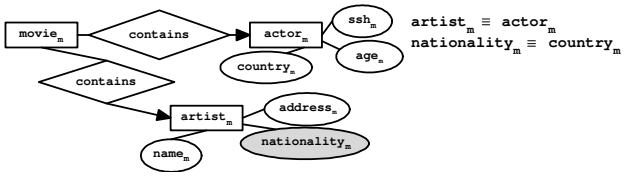


Figure 15. Redundant attribute detection

7.4. Implementation Issues

We implemented the IQ Manager as a module of an existing mediator-based data integration system. More details about the system can be found in [5]. The module was written in Java and the experiment used two databases – MySQL and PostgreSQL – to store the data sources. As mentioned before, the data in the system is XML and the schemas are represented with XML Schema.

The experiment was done in the following steps: (i) initially, the queries were submitted over an integrated schema 26% redundant and the execution times were measured; (ii) the redundancy elimination algorithm was executed over the redundant integrated schema generating a minimal schema (100% of minimality); (iii) the same queries of step (i) were executed. The results obtained with these experiments have been satisfactory.

8. CONCLUSION

Data integration systems may suffer with lack of quality in produced query results. They can be outdated, erroneous, incomplete, inconsistent, redundant and so on. As a consequence, the query execution can become rather inefficient. To minimize the impact of these problems, we propose a quality approach that serves to analyze and improve the integrated schema definition and consequently, the query execution.

It is known that a major problem in data integration systems is to execute user queries efficiently. The main contribution of the presented approach is the specification of IQ criteria assessment methods for the maintenance of high quality integrated schemas with objectives of achieving better integrated query execution. We also proposed an algorithm used to improve the schema's minimality score.

We have specified the *IQ Manager* module to proceed with all schemas IQ analysis and also the execution of improvement actions by eliminating the redundant items.

As future work, similarly as done with the minimality criterion, we must formally describe and implement the algorithms to evaluate the others IQ criteria and to execute the schema IQ improvement actions for each one.

9. REFERENCES

[1] ANHAI, D., DOMINGOS, P. and HALEVY, A. Learning to Match the Schemas of Data Sources: A Multistrategy. *Machine Learning, 50(3),* 2003.

[2] ASSENOVA, P. and JOHANNESON, P. Improving quality in conceptual modeling by the use of schema transformations, Proceedings 15th Int. Conf. of Conceptual Modeling (ER´96), Cottbus, Germany, 1996.

[3] BALLOU, D.P. and PAZER, H.L. Modeling data and process quality in multi-input, multi-output information systems. *Management Science* 1985.

[4] BALLOU, D. P. and PAZER H.: Modeling Completeness versus Consistency Tradeoffs in Information Decision Contexts. IEEE Transactions on Knowledge and Data Engineering , vol. 15, no. 1, 2003.

[5] BATISTA, M. C., LÓSCIO, B. F. AND SALGADO, A. C. Optimizing Access in a Data Integration System with Caching and Materialized Data. In Proc. of 5th ICEIS, 2003.

[6] CALI, A., CALVANESE, D., DE GIACOMO, G. and LENZERINI, M. Data integration Under Integrity Constraints. In Proc. Conference on Advanced Information Systems Engineering, 2002.

[7] CARDELLI, L. and WEGNER, P. On Understanding Types, Data Abstraction, and Polymorphism. ACM Computing Surveys, Vol.17, No.4, Dec. 1985.

[8] CHEN, P.P. The Entity-Relationship Model: Toward a unified view of data, ACM Transactions on Database Systems, 1976.

[9] DAI, B., T., KOUDAS, N., OOI, B. C., SRIVASTAVA, D. and VENKATASUBRAMANIAN, S. Column Heterogeneity as a Measure of Data Quality, in proceedings of 1st Int'l VLDB Workshop on Clean Databases, 2006.

[10] HALEVY, A. Why Your Data Don't Mix. *ACM Queue, 3(8),* 2005.

[11] HERDEN, O. Measuring Quality of Database Schema by Reviewing - Concept, Criteria and Tool. In Proc. 5th Intl Workshop on Quantitative Approaches in Object-Oriented Software Engineering, 2001.

[12] KESH, S. Evaluating the Quality of Entity Relationship Models. Inform. Software Technology. 1995.

[13] KOTONYA, G and SOMMERVILLE, I. Requirements Engineering: Processes and Techniques. 1st Edition, Wiley & Sons, 1997.

[14] LENZERINI, Maurizio. Data integration: A theoretical perspective. In *Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002)*, pages 233–246, 2002.

[15] LÓSCIO, B. F., Managing the Evolution of XML-Based Mediation Queries. Tese de Doutorado. Curso de Ciência da Computação. Centro de Informática, UFPE, Recife, 2003.

[16] MOODY, D. Measuring the Quality of Data Models: An Empirical Evaluation of the Use of Quality Metrics in Practice, New Paradigms in Organizations, Markets & Society: Proc. of the 11th European Conference on Information Systems, 2003.

[17] NAUMANN, F. and LESER, U. Quality-dri ven Integration of Heterogeneous Information Systems. In Proc. of the 25th VLDB. 1999.

[18] PETERSON, D., BIRON, P. V., MALHOTRA, A. and SPERBERG-MCQUEEN., C. M. *XML Schema 1.1 Part 2: Data Types* – W3C Working Draft, 2006. http://www.w3.org/TR/xmlschema11-2/.

[19] PIATTINI, M., GENERO, M. and CALERO, C. Data Model Metrics. In Handbook of Software Engineering and Knowledge Engineering: Emerging Technologies, World Scientific, 2002.

[20] SCANNAPIECO, M. Data quality at a glance. Datenbank-Spektrum *14*, 6–14.

[21] SI-SAID, S. C. and PRAT, N. Multidimensional Schemas Quality: Assessing and Balancing Analyzability and Simplicity, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2814, 140—151. 2003.

[22] SPACCAPIETRA, S. and PARENT, C. View integration: a step forward in solving structural conflicts, IEEE Transactions on Knowledge and Data Engineering, 1994.

[23] VARAS, M. Diseño Conceptual de Bases de Datos: Un enfoque Basado en la Medición de la Calidad", Actas Primer Workshop Chileno de Ingeniería de Software, Punta Arenas, 2001.

[24] WAND, Y. and WANG, R.Y.: Anchoring data quality dimensions in ontological foundations. Communications of the ACM 39(11), 86--95. (1996).

[25] WANG R.Y. and STRONG D.M.: Beyond Accuracy: What Data Quality Means to Data Consumers. Journal of Management Information Systems,1996.

[26] WANG, Y. and HU, J. Detecting tables in HTML documents. In Lecture Notes in Computer Science, volume 2423, pages 249–260. Springer-Verlag, 2002.

[27] WIEDERHOLD, G., 1992. Mediators in the Architecture of Future Information Systems. IEEE Computer. 2.

[28] ZHANG, Y., CALLAN, J. and MINKA. T. Novelty and Redundancy Detection in Adaptive Filtering. In Proc. of the 25 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), 2002.