# DATA INTEGRATION SCHEMA ANALYSIS: AN APPROACH WITH INFORMATION QUALITY

(Research-in-Progress)

**Maria da Conceição Moraes Batista, Ana Carolina Salgado**
Centro de Informática, Universidade Federal de Pernambuco
{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 and consistency aspects and define some schema transformations to be applied in order to improve schema generation and, consequently, the quality of data integration query execution.

**Key Words**: Data Quality, Information Quality, Data Integration, Schema Quality

## 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 ([23], [19], [2]).

A data integration system 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. The propagation of data with lack of quality is a real problem in data integration and, in some cases, the integration step may not be executed if IQ problems are not fixed [4].

In general, information may be of poor quality because it does not reflect real world conditions or because it is not easily used and understood by users. The cost of poor information quality must be measured by its accordance with user requirements [22]. Even accurate information, if not interpretable and accessible by the user, is of little value.

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 focus on formally specifying the algorithms and definitions of schema IQ criteria – minimality and type consistency. We also defined an algorithm to carry out with schema minimality improvements.

The paper is organized as follows: section 2 introduces several issues related to schemas' IQ; section 3 discusses the schema representation; in section 4 we present the formal specification of the IQ criteria and in section 5 we discuss some examples of the analysis of these criteria; section 6 presents the schema improvement algorithm addressing minimality aspects; in section 7 we discuss existing IQ approaches

addressing data integration and schemas issues; in section 8 is our concluding remarks and the final considerations about the mentioned topics.

# 2. DATA INTEGRATION AND SCHEMA QUALITY

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. Commonly, the tasks of query processing involving query submission, planning, decomposition and results integration are performed by a software module called *mediator* [24]. Each source publishes a data source schema with the representation of its contents. The mediator reformulates a user query into queries that refers directly to schemas on the sources. To the reformulation step, a set of correspondence, 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.

As a starting point, we adopted IQ classifications proposed in previous works ([1], [13], [8], [16], [21], [23]) 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*. We defined these components as the core classes 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.

**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 |

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

**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 [9]. Thus, we believe that our minimality analysis will help decreasing the extra time spent by mediators accessing 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.

The quality analysis is performed by a software module called *IQ Manager* or *Information Quality Manager* which may be attached to a data integration system. 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, executed after the IQ evaluation, is presented in section 6.

**Table 2. IQ Criteria for schemas quality analysis**

| IQ Criteria | Definition | Metrics |
|---|---|---|
| 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] |

# 3. 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 [11] described in what follows. We also present the schema mappings with this notation.

## *3.1 X-Entity Model*

The X-Entity model is an extension of the Entity Relationship model [7], 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 X-Entity model is the entity type, which represents the structure of XML elements composed by other elements and attributes. 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.
- *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, $R(E_1, E_2, \{A_{11}, \ldots, A_{1n}\}, \{A_{21}, \ldots, A_{2n}\})$, where $R$ is the name of the relationship and 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 $E_2$.

## *3.2 Schema Mappings*

A data integration system is widely based on the existence of metadata describing individual sources and integrated schema, and on schema mappings [18] 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 and they have the same semantics.
- **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 world 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

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

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.

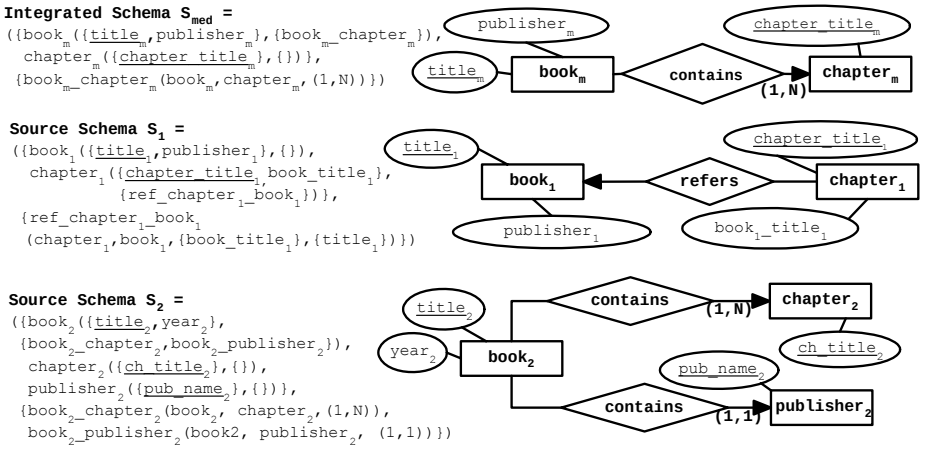



**Figure 1. Integrated Schema ($S_{med}$) and Schemas of data sources ($S_1$ and $S_2$)**

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

| |
|---|
| **$MP_1$**: $book_m \equiv book_1$ |
| **$MP_2$**: $book_m.title_m \equiv book_1.title_1$ |
| **$MP_3$**: $book_m.publisher_m \equiv book_1.publisher_1$ |
| **$MP_4$**: $chapter_m \equiv chapter_1$ |
| **$MP_5$**: $chapter_m.chapter\_title_m \equiv chapter_1.chapter\_title_1$ |
| **$MP_6$**: $book_m.book_m\_chapter_m.chapter_m \equiv (chapter_1.chapter\_ref\_book_1.book_1)^{-1}$ |
| **$MP_7$**: $book_m \equiv book_2$ |
| **$MP_8$**: $book_m.title_m \equiv book_2.title_2$ |
| **$MP_9$**: $chapter_m \equiv chapter_2$ |
| **$MP_{10}$**: $book_m.book_m\_chapter_m.chapter_m \equiv book_2.book_2\_chapter_2.chapter_2$ |
| **$MP_{11}$**: $chapter_m.chapter\_title_m \equiv chapter_2.ch\_title_2$ |
| **$MP_{12}$**: $book_m.publisher_m \equiv book_2.book_2\_publisher_2.publisher_2.pub\_name_2$ |

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. Particularly, in the environment exploited to experiment the proposed IQ evaluation ([5], [11]), there is a schema generator component responsible to maintain equivalencies and define mappings among data sources and integrated schema.

Our proposition, centered in IQ analysis for schemas in data integration systems, has goals of query

optimization and it is detailed in the following sections.

# 4. SCHEMA IQ ASPECTS

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.

From now on, we assume that the integrated schema is already created, and, consequently, the equivalences between entities, attributes and relationships are already defined.

## *4.1 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, $\mathbf{Ð = <\delta, S_m, \rho, \varphi(Ð)>}$ where:

- $\delta$ is the set of $S_i$ data sources schemas, i.e. $\delta = \mathbf{<S_1, S_2, \ldots, S_w>}$, where w is the total number of data sources participating in **Ð**;
- $\mathbf{S_m}$ is the integrated schema, generated by internal modules of **Ð**;
- $\rho$ is the set of user schemas, $\rho = \mathbf{<U_1, U_2, \ldots, U_u>}$ 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, $S_m = <E_1, E_2, \ldots, E_{n_m}>$ where $E_k$ is an integration or mediation entity $(1 \leq k \leq n_m)$, 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 $E_k$ $(r_k \geq 0)$.
- 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.

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 [10], very important to schemas construction.

### Definition 2 – Domain Concepts Set

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

- $\beta$ is even a given integrated schema $S_m$ or a data integration system **Ð**;
- $\sigma_\beta$ is the number of real world concepts in $\beta$;
- $\mathbf{C_k}$ is an application domain concept which is represented by an schema element **Y** in one of the two following ways:
  - i) $\mathbf{Y} \in S_m$, if $\beta$ is a integrated schema or;
  - ii) $\mathbf{Y} \in \delta = \mathbf{<S_1, S_2, \ldots, S_w>}$, if $\beta$ is the data integrated system.

Usually, the data integration system has mechanisms to generate and maintain the integrated schema. 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:

$$\exists E_k.A_{kj},\ j \neq i,\ A_{kj} \in \{A_{k1}, A_{k2}, \ldots, A_{ka_k}\} \text{ and } E_k.A_{ki} \equiv E_k.A_{kj},\ 1 \leq i,j \leq a_k$$

### 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:

$\exists E_o,\ o \neq k,\ E_o \in S_m,\ E_k \equiv E_o$,

$E_o(\{B_{o1}, B_{o2}, \ldots, B_{oa_o}\})$, $B_{oj}$ are attributes of $E_o$ and $\exists E_o.B_{oj},\ B_{oj} \in \{B_{o1}, B_{o2}, \ldots, B_{oa_o}\}$

and $E_k.A_{ki} \equiv E_o.B_{oj}, 1 \leq i \leq a_k,\ 1 \leq j \leq a_o$.

If an attribute $A_{ki}$, $A_{ki} \in \{A_{k1}, A_{k2}, \ldots, A_{ka_k}\}$, and $\mathbf{Red(A_{ki}, E_k) = 0}$, 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:

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

$\sum_{i=1}^{a_k} Red(A_{ki}, E_k)$ 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. $\mathbf{Red(E_k, S_m) = 1}$. 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 6. As an example of redundant attributes and the entity redundancy degree, suppose the schema and mappings of Figure 2.

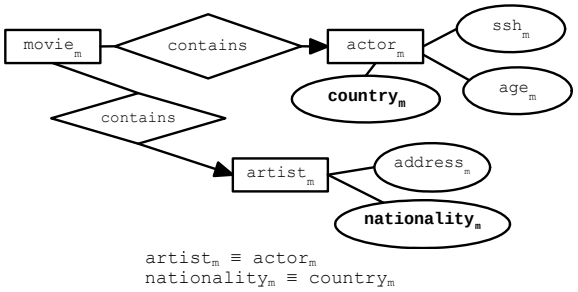



**Figure 2. Schema with redundant attributes**

The attribute $nationality_m$ 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:

$Red(nationality_m, artist_m) = 1$
$Red(address_m, artist_m) = 0$
$Red(country_m, actor_m) = 0$
$Red(age_m, actor_m) = 0$
$Red(ssh_m, actor_m) = 0$

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 $S_m$:

$Red(artist_m, S_m) = (1 + 0)/2 = 0.5$
$Red(actor_m, S_m) = (0 + 0 + 0)/3 = 0$

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 $E_y$ iif $R \in \{R_{k1}, \ldots, R_{kr_k}\}$ and $R \in \{T_{y1}, \ldots, T_{yr_y}\}$.

We define $R$ as a *redundant relationship* in $S_m$, i.e. $\mathbf{Red(R, S_m) = 1}$ 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 $T_s \in \{T_{y1}, \ldots, T_{yr_y}\}$ and $P_1 \equiv R$.

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. 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 $P_1$.

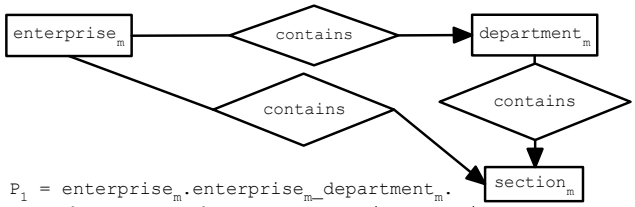



**Figure 3. Schema with redundant relationship**

## *4.1 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. These affirmatives are related to data consistency, but they can be extended to adequately represent data type consistency constraints [16].
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 [6].
When the same data type is recorded in more than one way, an integrated schema management system experiences problems with consistency. The first step in resolving this consistency problem is to

determine which alternative data type is preferable. This approach would then be defined as a standard, namely, the accepted way of recording the information. In this case, a schema element is called consistent if it adheres to the defined standard data type. On the other side, when there are a variety of types for the same information, the conversion may be a difficult process, and achieving consistency could be both time-consuming and expensive. As in [3], we have based the consistency metric in an essential factor: in a set of semantically equivalent attributes, the number elements that adhere to the standard data type defined for the group.

We approximate consistency rules and data types to create the *Type Consistency* concept, and associate it with the degree in which an attribute is recorded with different types. We use the *Type Consistency* criterion to investigate which data elements in the schemas are represented with the same type, adhering to a consistency standard. This is an indicator of quality and query improvement, once the query processor is not going to perform a number of 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 [21].

As 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 7 – 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 [6]. 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 [14], we import the concept of *datatype* as follows:

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

- $\alpha$ is a set of distinct values, called the *value space*, containing the values that the elements of the type can have**;**
- $\lambda$ is a set of lexical representations, called the *lexical space* and;
- $\gamma$ is a set of facets that characterize properties of the value space, individual values or lexical items;
- $\mathbf{T \in £}$, where **£** is the set of all XML schema datatypes.

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

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

### Definition 8 – 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 9 – 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:

$\forall E_k.A_{kj}, E_k \in \mathbf{Ð},$

$A_{kj} \in \{A_{k1}, A_{k2}, \ldots, A_{ka_k}\}$ and $\exists \mathbf{T_{std}}, \mathbf{T_{std}} = <\alpha_{std}, \lambda_{std}, \gamma_{std}>$ and

$\exists E_x.A, E_x \in \mathbf{Ð} \wedge E_x.A \equiv E_k.A_{kj}, \mathbf{A = (T_{std}, v)}$ where

$\mathbf{T_{std}}$ is the most frequently data type used in **Ð** for attribute A and its equivalents.

### Definition 10 – 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 7) of an entity $E_p \in S_p$ ($S_p = S_i$ or $S_p = S_m$) is

***consistent*** i.e. $\mathbf{Con(A_{pj}, S_p) = 1}$ if it appears in another entity or even in the same entity with other datatype:

$\exists T_{std} \in$ **Ð**, $\mathbf{T \in £}$ and $A_{pj} = (T_{std}, v_{pj})$

**Definition 11 – Schema Data Type Consistency**

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

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

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

$n_m$ is the total number of entities in the schema **Ð**;

$a_k$ is the number of attributes of the entity $E_k$.

In this section we have presented the formal specifications introduced for type consistency criterion. We strongly believe that improving the consistency of the information across the schemas in **Ð** will help the query execution, once it will be required less conversion steps between equivalent attributes.

## *4.2 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 ([9], [20], [12], [15]). 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 [9] 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, we assume the following:

- $n_{rel}$ is the total number of relationships in $S_m$;
- $n_m$ is the total number of entities in $S_m$.

**Definition 12 – Schema Minimality**

We define the overall redundancy of a schema in a data integration system as the sum of the aforementioned redundancy values: entities and relationships. The schema minimality is measured by the computation of the following:

$$Mi(S_m) = 1 - \left( \frac{\sum_{k=1}^{n_m} Re\, d(E_k, S_m)}{n_m} + \frac{\sum_{l=1}^{n_{rel}} Re\, d(R_l, S_m)}{n_{rel}} \right), \text{ where}$$

$\frac{\sum_{k=1}^{n_m} Re\, d(E_k, S_m)}{n_m}$ is the entity redundancy score of schema $S_m$;

$\frac{\sum_{l=1}^{n_{rel}} Re\, d(R_l, S_m)}{n_{rel}}$ is the relationship redundancy score of schema $S_m$.

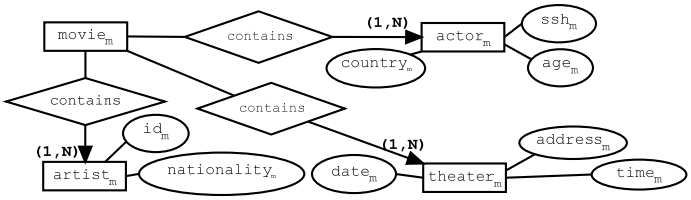
This section discussed the minimality specification. If the data integration system has minimal schemas, the query execution step can be simplified once the system will not spend time querying redundant elements. Analogously, the query result may have good quality scores as an effect of the absence of redundant items in its composition.

# 5. 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.

## *5.1 Minimality Analysis*

For an example of minimality evaluation, assume the redundant schema of Figure 4 and its equivalencies.




$artist_m \equiv actor_m$
$id_m \equiv ssh_m$
$nationality_m \equiv country_m$

**Figure 4. 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 5.

$Red(movie_m, S_m) = 0$
$Red(actor_m, S_m) = 0$
$Red(theater_m, S_m) = 0$
$Red(artist_m, S_m) = 1$
$Red(movie_m\_actor_m, S_m) = 0$
$Red(movie_m\_artist_m, S_m) = 1$
$Red(movie_m\_theater_m, S_m) = 0$
$ER(S_m) = 1/4 = 0,25$
$RR(S_m) = 1/3 = 0,333$
$Mi(S_m) = 1 - (0,25 + 0.333) = 0,417$

**Figure 5. Schema minimality score**

The minimality of schema $S_m$ is 41,70%, what means that the schema has 58,30% of redundancy that can possibly be eliminated.

## *5.2 Type Consistency Analysis*

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

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

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
$Con(actor_m.birthdate_m, Đ) = 1$
$Con(actor_1.birth_1, Đ) = 0$
$Con(actor_2.birth_2, Đ) = 1$

$Con(actor_3.bd_3, Đ) = 1$

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 11.

# 6. SCHEMA MINIMALITY IMPROVEMENT

After detecting the schema IQ anomalies, it is possible to restructure it to achieve better IQ scores [1]. 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 4.

It is important to see that is possible to 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. It will occur when all the redundancies have been eliminated.

**Table 4. 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.

## *6.1 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 $\exists R_y$, $R_y \in E_2$ and $R_x \equiv R_y$ then $R_x$ is deleted;
- If $R_x \notin E_2$ and there is no $R_y$, $R_y \in E_2$ and $R_x \equiv R_y$, 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 13 – Substitute Entity

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

- $E_k$ $(\{A_{k1}, \ldots, A_{ka_k}\}, \{R_{k1}, \ldots, R_{kr_k}\})$ $A_{kx}$ 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 13 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 14 – 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, to avoid the schema of losing relevant 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 $\exists E_s.R_{sp}$, $E_k.R_{kj} \equiv E_s.R_{sp}$ 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 $\nexists E_s.R_{sp}$, $E_k.R_{kj} \equiv E_s.R_{sp}$ 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 6 and 7.

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 $\{movie_m\_actor_m(movie_m, actor_m, (1,N))\}$.

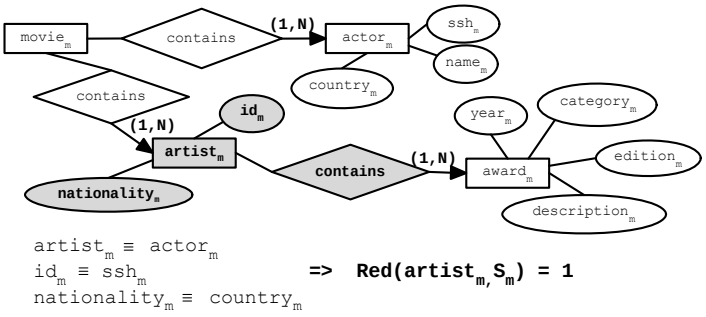



**Figure 6. Redundant entity detection**

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 operation, it is possible to obtain a no redundant schema as illustrated in Figure 7.

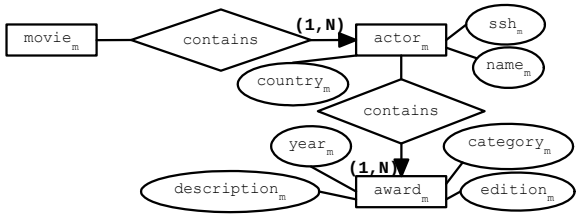



**Figure 7. Relationship relocation**

## *6.2 Redundant Relationships Elimination*

After removing redundant entities and possibly performing the necessary relationship relocations, the *IQ Manager* identifies remaining redundant relationships to eliminate. This can be accomplished by merely deleting from the schema, the relationships identified as redundant. Considering the example of Figure 8, the relationship $\{enterprise_m\_section_m(enterprise_m, section_m, (1,N))\}$ is redundant because it has a semantically equivalent correspondent represented by $P_1$.

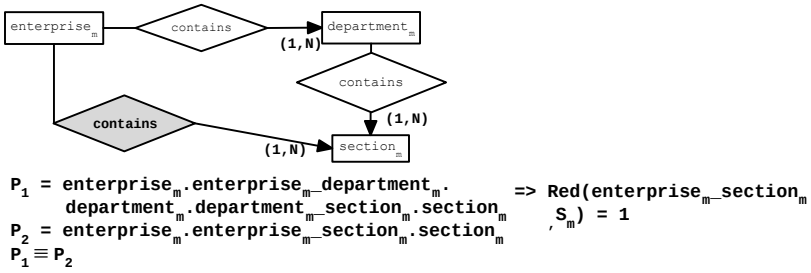



**Figure 8. Redundant relationship detection**

After eliminating the relationship {enterprise$_m$_section$_m$(enterprise$_m$, section$_m$,(1,N))}, the schema with no relationship redundancies is showed in Figure 9.

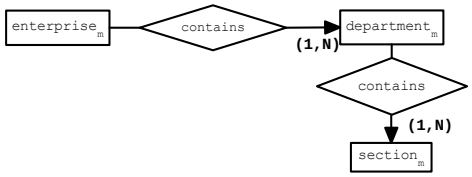



**Figure 9. Redundant relationship elimination**

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 useful to assist the query processing.

## *6.3 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 10, the attribute nationality$_m$ is removed because there is a semantically equivalent attribute country$_m$, which will substitute it.

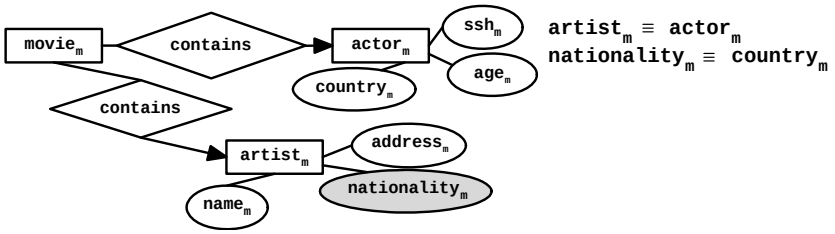



**Figure 10. Redundant attribute detection**

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.

## *6.4 Experimental Results*

We implemented the IQ Manager in 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 data integration system is a real application in a health care domain. The main concepts are doctors, patients, diseases, treatments, among other.

The following steps were executed: (i) initially, the queries were submitted over an integrated schema with a 26% degree of redundancy and the execution times were measured; (ii) second, the redundancy elimination algorithm was executed over the redundant integrated schema generating a completely

minimal schema; (iii) the same queries used in step (i) were executed. The query performance was improved and results obtained with these experiments have been satisfactory.

# 7. RELATED WORKS

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 ([8], [13]). Naumann and Leser [13] 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 [1], where the authors exploit the use of normalization rules to improve IQ in conceptual database schemas.

The work proposed by Herden [8] 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 [17] the authors propose IQ evaluation for data warehouse schemas focusing on the analyzability and simplicity criteria.

The differential of our approach is the proposal of processes for management of data integration schemas associated with IQ analysis features and design improvements. The main contributions are: (i) consolidation of using IQ in data integration systems through the classification of a set of criteria specifically selected for this kind of environment; (ii) specification of the relevant IQ criteria in the context of a data integration system and; (iii) analysis of system's components according to the specified IQ criteria. We presented specifically the IQ analysis of schemas associated with an algorithm for IQ improvement.

# 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 an Information 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 to improve the schema's minimality score.

We have specified the *IQ Manager* as a module of a data integration system ([5], [11]) to proceed with all schemas IQ analysis and also the execution of improvement actions by eliminating the redundant items.

Similarly as done with the minimality criterion, we are working to formally describe and implement the algorithms to evaluate the others specified IQ criteria, and proceed with schema IQ improvement.

# REFERENCES


[1] 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.

[2] BALLOU, D.P. and PAZER, H.L. Modeling Data and Process Quality in Multi-input, Multi-output Information Systems. *Management Science* 1985.

[3] 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.

[4] BOUZEGHOUB, M. and LENZERINI M. Introduction to the Special Issue on Data Extraction, Cleaning, and Reconciliation. *Information Systems*, 26(8):535–536, 2001.

[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] CARDELLI, L. and WEGNER, P. On Understanding Types, Data Abstraction, and Polymorphism. *ACM Computing Surveys*, Vol.17, No.4, Dec. 1985.
[7] CHEN, P.P. The Entity-Relationship Model: Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1976.
[8] 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.
[9] KESH, S. Evaluating the Quality of Entity Relationship Models. *Inform. Software Technology*. 1995.
[10] KOTONYA, G and SOMMERVILLE, I. Requirements Engineering: Processes and Techniques. 1st Edition, Wiley & Sons, 1997.
[11] 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.
[12] 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 *11th European Conference on Information Systems*, 2003.
[13] NAUMANN, F. and LESER, U. Quality-driven Integration of Heterogeneous Information Systems. In Proc. of the 25th *VLDB*. 1999.
[14] 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/.
[15] PIATTINI, M., GENERO, M. and CALERO, C. Data Model Metrics. In *Handbook of Software Engineering and Knowledge Engineering: Emerging Technologies*, *World Scientific*, 2002.
[16] SCANNAPIECO, M. Data Quality at a Glance. *Datenbank-Spektrum 14*, 6–14. 2005.
[17] 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.
[18] SPACCAPIETRA, S. and PARENT, C. View Integration: a Step Forward in Solving Structural Conflicts, *IEEE Transactions on Knowledge and Data Engineering*, 1994.
[19] TAYI, G. K. and BALLOU, D. P. Examining Data Quality. Communications of the ACM 41(2), 1998.
[20] 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.
[21] WAND, Y. and WANG, R.Y. Anchoring Data Quality Dimensions in Ontological Foundations. *Communications of the ACM* 39(11), 86—95, 1996.
[22] WANG, R., KON, H. and MADNICK, S. Data Quality Requirements Analysis and Modelling, 9th *International Conference of Data Engineering*, Vienna, Austria, 1993.
[23] WANG R.Y. and STRONG D.M. Beyond Accuracy: What Data Quality Means to Data Consumers. *Journal of Management Information Systems*, 1996.
[24] WIEDERHOLD, G., 1992. Mediators in the Architecture of Future Information Systems. *IEEE Computer*. 2.