Chapter 6
Models for Information Quality

6.1 Introduction

In the previous chapters, we introduced several dimensions that are useful to describe and measure information quality in its different aspects and meanings. Focusing on structured data, database management systems (DBMSs) represent data and relative operations on it in terms of a *data model* and a *data definition and manipulation language*, i.e., a set of structures and commands that can be represented, interpreted, and executed by a computer. We can follow the same process to represent, besides data, their quality dimensions. This means that in order to represent data quality, we have to extend data models.

Models are widely used in databases for various purposes, such as analyzing a set of requirements and representing it in terms of a conceptual description, called *conceptual schema*; such a description is translated into a *logical schema* in terms of which queries and transactions are expressed.

Models are also used in the wider area of information systems to represent business processes of organizations; processes are described in terms of activities, their inputs and outputs, causal relationships between them, and functional/non-functional requirements. Such models are needed in order to help the analyst, e.g., to analyze and foresee process behavior, measure performance, and design possible improvements.

In this chapter, we investigate the principal extensions of traditional models adopted for structured relational data and semistructured data to deal with data quality dimension issues. In Sect. 6.2, we investigate proposed extensions of conceptual and logical database models for structured data. Logical models are considered both from the perspective of data description models and as related to data manipulation and data provenance. Then we discuss models for semistructured information, with specific attention to XML schemas (Sect. 6.3). In Sect. 6.4, we move on to management information system models; here, we investigate two "orthogonal" issues: (1) extensions of models for process descriptions to issues

related to quality of sources, users involved in data checks, etc., and (2) proposals for joint representation of elementary and aggregated data and related qualities. In all the models that we are going to describe, we will see that the extensions of models to data quality issues lead to structures characterized by significant complexity.

6.2   Extensions of Structured Data Models

The principal database models are the Entity Relationship model, the most common for conceptual database design (see [38]), and the relational model, adopted by a wide range of DBMSs.

6.2.1   Conceptual Models

Several solutions exist for extending the Entity Relationship model with quality characteristics (see [594, 595]). The different proposals focus on *attributes*, the unique representation structure in the model with which data values may be associated. A possibility is to model the quality of attribute values as another attribute of the same entity. For example, if we want to express a dimension (e.g., accuracy or completeness) for the attribute `Address` of an entity `Person`, we may add (see Fig. 6.1) a new attribute `AddressQualityDimension` to the entity.

The drawback of this solution is that now the entity is no longer normalized, since the attribute `AddressQualityDimension` is dependent upon `Address`, which is dependent upon `Id`. Another problem is that if we want to define several dimensions for an attribute, we have to define a new attribute for each dimension, resulting in a proliferation of attributes.

A second possibility is to introduce two types of entities, explicitly defined for expressing quality dimensions and their values: a data quality dimension entity and a data quality measure entity.

The goal of the `DataQualityDimension` entity is to represent all possible pairs of dimensions and corresponding ratings; the pairs <DimensionName, Rating> constitute the set of dimensions and possible corresponding values resulting from measurements. In the previous definition, we have implicitly assumed that the scale of rating is the same for all attributes. If the scale depends on the attribute, then we have to extend the properties of the `DataQualityDimension` entity to <Dimension-Name, Attribute, Rating>.

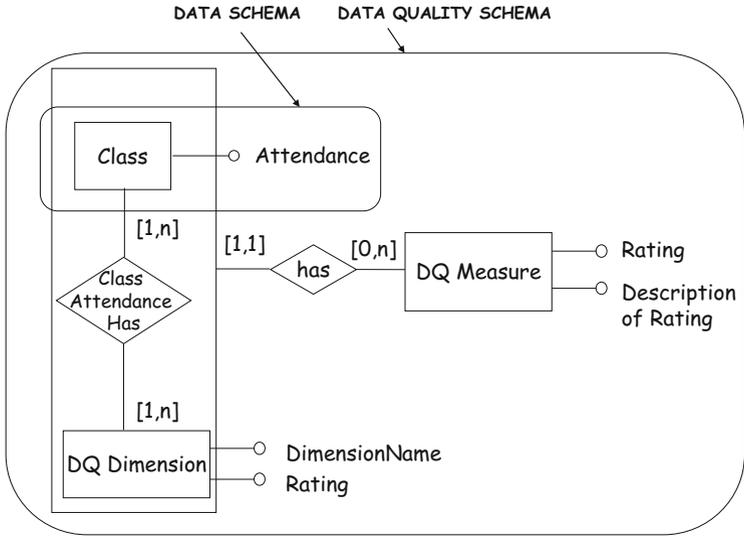**Fig. 6.1** A first example of quality dimension represented in the Entity Relationship model

Fig. 6.2 An example of Data Quality Schema as proposed in [595]

In order to represent metrics for dimensions, and their relationship with entities, attributes, and dimensions, we have to adopt a more complex structure than the one shown in Fig. 6.1, in which we introduce the `DataQualityMeasure` entity; its attributes are `Rating`, in which the values depend on the specific dimension modeled, and `DescriptionofRating`. The complete *data quality schema*, which we show by means of the example in Fig. 6.2, is made up of:

1. The original *data schema*, made in the example of the entity `Class` with all its attributes (here, we represent only the attribute `Attendance`).
2. The `DQ Dimension` entity with a pair of attributes <`DimensionName`, `Rating`>.
3. The relationship between the entity `Class`, the related attribute `Attendance`, and the `DQ Dimension` entity with a many-to-many relationship `ClassAttendanceHas`; a distinct relationship has to be introduced for each attribute of the entity `Class`.
4. The relationship between the previous structure and the `DQ Measure` entity with a new representation structure that extends the Entity Relationship model and relates entities and relationships.

The overall structure adopted in Fig. 6.2 has been proposed in [595]. The above example shows how complex a schema becomes extended with the above structures to describe qualities.

6.2.2   Logical Models for Data Description

The relational model is extended in [647, 649] with quality values associated with each attribute value, resulting in the *quality attribute model*. We explain the model with an example, shown in Fig. 6.3.

The figure shows a relational schema Employee, defined on attributes EmployeeId, DateofBirth, and others, and one of its tuples. Relational schemas are extended, adding an arbitrary number of underlying levels of *quality indicators* (only one level in the figure) to the attributes of the schema, to which they are linked through a quality key. In the example, the attribute EmployeeId is extended with three quality attributes, namely, accuracy, currency, and completeness, while the attribute DateofBirth is extended with accuracy and completeness, since currency is not meaningful for permanent data such as DateofBirth. The values of such quality attributes measure the quality dimensions' values associated with the whole relation instance (top part of the figure). Therefore, completeness equal to 0.7 for the attribute DateofBirth means that the 70 % of the tuples have a non-null value for such an attribute. Similar structures are used for the instance level quality indicator relations (bottom part of the figure); if there are n attributes of the relational schema, n quality tuples will be associated to each tuple in the instance.
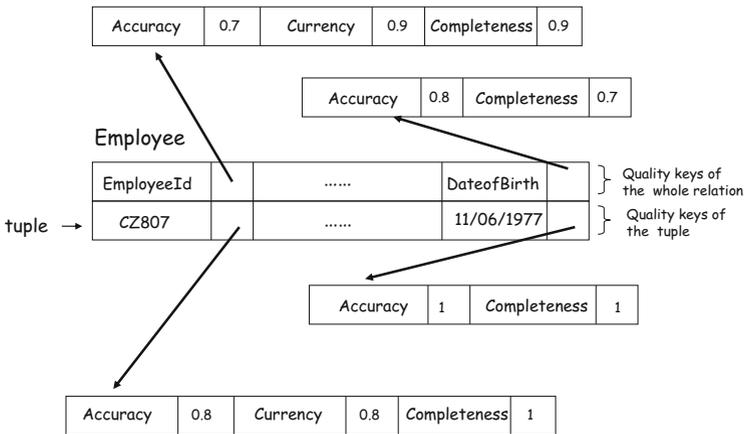

Fig. 6.3 An extension of the relational model

6.2.3   The Polygen Model for Data Manipulation

In principle, in every process of data collection and analysis, such as medical or biological experiments, data originating from different sources are manipulated in different stages; new data produced at each stage inherit the quality of ancestor data according to histories that depend on the execution plan. In Chap. 7, for several quality dimensions and relational algebra operations, we will investigate the functional relationships between the quality values of the input data and the quality values of the output data. In this section, we investigate an extension of the relational model, called *polygen model* [644, 649], proposed for explicitly tracing the origins of data and the intermediate sources. The model is targeted to heterogeneous distributed systems; the name of the model is derived from "multiple" "sources" (respectively, "poly" and "gen" in Greek). Now we briefly discuss the model, relevant for its pioneer role in the area. A *polygen domain* is a set of ordered triples composed of:

1. A datum drawn from a simple domain in a schema of a local database
2. A set of *originating databases* denoting the local databases from which the datum originates
3. A set of *intermediate databases* in which the data led to the selection of the datum

A *polygen relation* is a finite set of time-varying tuples, each tuple having the same set of attribute values from the corresponding polygen domains. A *polygen algebra* is a set of relational algebra operators whose semantics allows annotation propagation. The five primitive operators in the model are project, cartesian product, restrict, union, and difference. More precisely:

1. *Project*, *cartesian product*, *union*, and *difference* are extended from the relational algebra. The difference operator over two polygen relations r_1 and r_2 is extended as follows (for the remaining operators, see [644, 649]). A tuple t in r_1 is selected if the data part of t is not identical to those of the tuples of r_2. Since each tuple in r_1 has to be compared with all the tuples in r_2, it follows that all the originating sources of the data in r_1 are to be included in the intermediate source set produced by the difference operator.
2. The *restrict* operator is introduced to select tuples in a polygen relation that satisfy a given condition, and such tuples populate intermediate sources.
3. *Select* and *join* are defined in terms of the restrict operator, so they also involve intermediate sources.
4. New operators are introduced, e.g., *coalesce*, which takes two columns as input and merges them into one column (no inconsistency is admitted).

Note that in general in heterogeneous multidatabase systems, the values coalesced may be inconsistent. This issue is not considered in the polygen approach; it will be discussed in detail in Sect. 10.4.3 dedicated to instance-level conflict resolution techniques.

6.2.4   Data Provenance

The polygen model is a first attempt to represent and analyze the provenance of data, which has been recently investigated in a more general context. *Data provenance* is defined in [98] as the "description of the origins of a piece of data and the process by which it arrived in the database." We will provide a detailed discussion on provenance in the context of Web data in Chap. 14, while in this chapter, the focus is on structured data in a database.

The typical mechanism to trace the provenance is the use of *annotations* that can be exploited to represent a wide spectrum of information about data, such as comments or other types of metadata, and, in particular, data representing the quality of data. Annotations can be used in a variety of situations including:

1. Systematically trace the provenance and flow of data, namely, even if the data has undergone a complex process of transformation steps, we can determine the origins by examining the annotations.
2. Describe information about data that would otherwise have been lost in the database, e.g., an error report about a piece of data.
3. Enable the user to interpret the data semantics more accurately and to resolve potential conflicts among the data retrieved from different sources. This capability is useful in the field of data integration (see Chap. 10), where we are interested in understanding how data in different databases with heterogeneous semantics and different quality levels can be integrated.
4. Filter the data retrieved from a database according to quality requirements.
5. Improve the management of data trustworthiness through annotations referring to the reputation of a source or to a certification procedure.

Two types of provenance are defined in the literature, *why provenance* and *where provenance* (see [98, 158], and [133] as the main references in this area). We introduce them by means of an example. Assume we issue the following query:

```
SELECT StudentId, LastName, Sex
FROM Student
WHERE Age > SELECT AVERAGE Age FROM Student
over the relational schema Student (StudentId, LastName, Sex, Age).
```

If the output is the tuple `<03214, Ngambo, Female>`, the provenance of the tuple can be related to two distinct data items:

1. The set of tuples in the input relation that contributed to the final result. In this case, all the tuples have to be selected as contributing tuples, since any modification in one tuple may affect the presence of `<03214, Ngambo, Female>` in the result. This kind of provenance is called *why provenance*, since we are looking for the tuples that explain the shape of the output.
2. The tuple(s) in the input relation that originated the values `03214, Ngambo,` and `Female` in the output tuple. In this case, the set is made up of the unique tuple with `StudentId = 03214`. This kind of provenance is called

where provenance, since in this case, we are interested in finding from where annotations are propagated. In the case of a join between two tuples, both would be considered part of the input set.

The where provenance is particularly useful in the data quality context. In the case where annotations represent quality values, control of the process of quality dimension propagation is allowed by identifying the sources that are responsible for quality degradation. For the above reasons, in the following, we focus on the where provenance.

We will discuss the concept of the where provenance and its different meanings in the following context: given a relational database D, with a set of annotations associated with tuples in D, and a query Q over D, compute the provenance of an output tuple t in the result of Q.

If we think of possible meanings, i.e., methods to compute the where provenance (similar considerations can be made for the why provenance), two different approaches exist: the *reverse query* (or lazy) approach and the *forward propagation* (or eager) approach.

In the *reverse query approach* (see [98, 158]), a "reverse" query Q′ is generated in which the result is the tuple or set of tuples that contribute, when Q has been executed, in producing it.

In the *forward propagation approach*, when applying Q, an enriched query Q* is generated and executed that computes how annotations are propagated in the result of Q. The approach is called *eager*, since provenance is immediately made available, together with the output of Q. The forward propagation approach, in turn, has three possible types of execution or *propagation schemes* [133], called the *default scheme*, the *default-all scheme*, and the *custom propagation scheme*. We introduce the three schemes by means of an example. Assume (see Fig. 6.4) we have a database of clients made up of two different tables, Client1 and Client2 and a mapping

Client1

Id	Description
071 [ann$_1$]	Cded [ann$_2$]
358 [ann$_3$]	Hlmn [ann$_4$]
176 [ann$_5$]	Stee [ann$_6$]

Client2

Id	Last Name
E3T [ann$_7$]	Nugamba [ann$_8$]
G7N [ann$_9$]	Mutu [ann$_{10}$]

MappingRelation

Id	Client1Id	Client2Id
1 [ann$_{11}$]	071 [ann$_{12}$]	E3T [ann$_{13}$]
2 [ann$_{14}$]	358 [ann$_{15}$]	G7N [ann$_{16}$]

Fig. 6.4 Two Client relations and a mapping relation

table between identifiers of clients in `Client1` and `Client2` (a typical situation in many organizations).

Intuitively, the default propagation scheme propagates annotations of data according to where data is copied from. Assume that the following query Q_1 is computed on the database of Fig. 6.4:

```
SELECT DISTINCT c.Id, c.Description
FROM Client1 c
WHERE c.Id = 071
```
The result of Q_1 executed against the relation `Client1` in the default propagation scheme is the unique tuple
$< 071[\text{ann}_1]; \text{Cded}[\text{ann}_2] >$

The semantics of the default scheme is quite natural, but it has a drawback, in that two equivalent queries (i.e., queries that return the same output for every database) may not propagate the same annotations to the output. Consider the two queries, Q_2:

```
SELECT DISTINCT c2.Id AS Id, c2.LastName AS LastName
FROM Client2 c2, MappingRelation m
WHERE c2.Id = m.Client2Id
```

and Q_3:

```
SELECT DISTINCT m.Id AS Id,  c2.LastName AS LastName
FROM Client2 c2 , MappingRelation m
WHERE c2.Id = m.Client2Id
```

The results of running Q_2 and Q_3 under the default propagation scheme are shown in Fig. 6.5. For Q_2, the annotations for the `Id` attribute are from the `Client2` relation, while for Q_3, the annotations for the `Id` attribute are from the `MappingRelation`.

The *default scheme* propagates the annotation for equivalent queries differently. We need a second propagation scheme, where propagations are invariant under equivalent queries. This scheme is called the *default-all propagation scheme* in [133]; it propagates annotations according to where data is copied from among all equivalent formulations of the given query. In case a user wants to bear the responsibility to specify how annotations should propagate, a third scheme can be adopted, the *custom scheme*, where annotation propagations are explicitly declared in the query.

Output of Q2

Id	Last Name
E3T [ann_7]	Nugamba [ann_8]
E3T [ann_9]	Muto [ann_{10}]

Output of Q3

Id	Last Name
E3T [ann_{13}]	Nugamba [ann_8]
E3T [ann_{16}]	Muto [ann_{10}]

Fig. 6.5 The output of two queries

The above schemes can be applied flexibly, whatever the type of the annotated information, i.e., it could be the source relation, the exact location within the source, or a comment on the data.

6.3   Extensions of Semistructured Data Models

In [548], a model for associating quality values to data-oriented XML documents is proposed. The model, called *data and data quality* (D^2Q), is intended to be used in the context of a cooperative information system (CIS). In such systems, the cooperating organizations need to exchange data with each other, and it is therefore critical for them to be aware of the quality of such data. D^2Q can be used in order to certify the accuracy, consistency, completeness, and currency of data. The model is semistructured, thus allowing each organization to export the quality of its data with a certain degree of flexibility. More specifically, quality dimension values can be associated with various elements of the data model, ranging from the single data value to the whole data source. The main features of the D^2Q model are summarized as follows:

- A *data class* and a *data schema* are introduced to represent the *domain* data portion of the D^2Q model, namely, the data values that are specific to a given cooperating organization's domain.
- A *quality class* and a *quality schema* correspond to the quality portion of the D^2Q model.
- A *quality association function* that relates nodes of the graph corresponding to the data schema to nodes of the graph corresponding to the quality schema. Quality associations represent biunivocal functions among all nodes of a data schema and all non-leaf nodes of a quality schema.

In Fig. 6.6, an example of a D^2Q schema is shown. On the left-hand side of the figure, a data schema is shown representing enterprises and their owners. On the right-hand side, the associated quality schema is represented. Specifically, two quality classes, `Enterprise_Quality` and `Owner_Quality` are associated with the `Enterprise` and `Owner` data classes. Accuracy nodes are shown for both data classes and related properties. For instance, `Code_accuracy` is an accuracy node associated with the `Code` property, while `Enterprise_accuracy` is an accuracy node associated with the data class `Enterprise`. The arcs connecting the data schema and the quality schema with the `quality labels` represent the quality association functions.

The D^2Q model is intended to be easily translated into the XML data model. This is important for meeting the interoperability requirements that are particularly
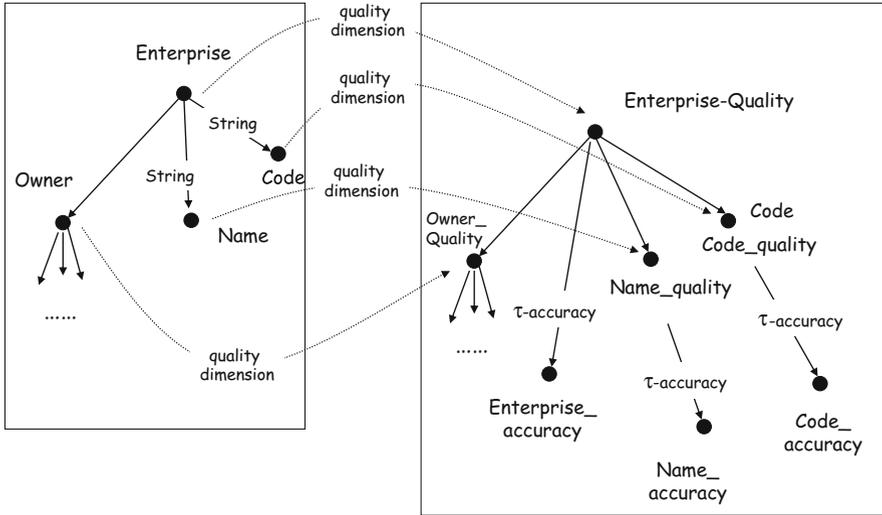
Fig. 6.6 Example of D^{2}Q quality schema

stringent in cooperative systems. Once translated into XML, the model can be queried by means of an extension of the XQuery language that queries quality values in the model. XQuery allows users to define new functions. Quality values represented according to the D^{2}Q model can be accessed by a set of XQuery functions, called *quality selectors*. Quality selectors are defined for accuracy, completeness, consistency, and currency and for the overall set of quality values that can be associated with a data node.

In Fig. 6.7, the implementation of the quality selector *accuracy*() is shown as an example. Searchroot is a function defined to reach the root of a document containing the input node.

The D^{2}Q model represents quality values to be associated with generic data. XML is used as a language for modeling quality dimensions in a growing number of contributions. For example, see in [428] a proposal for modeling quality of data by means of six quality measures meaningful in the biological domain. Being domain specific, such a proposal also includes metrics that allow the computation of node quality values across the XML graph, by considering the interdependencies between quality values of the various nodes in the graph.

```
define function accuracy($n as node*) as node* {
let $root := searchroot($n), qualitydoc:=document(string($root/@qualityfile))
for $q in $n/@quality
for $r in $qualitydoc//*[@qOID eq $q]/accuracy
return $r }
```

Fig. 6.7 Accuracy selector implementation as an XQuery function

6.4   Management Information System Models

In this section, we discuss management information system models in their relation to data quality issues. We discuss process models in Sects. 6.4.1 and 6.4.2, introducing the information production map (IP-MAP) model and its extensions. Issues related to data models are discussed in Sect. 6.4.3.

6.4.1   Models for Process Description: The IP-MAP Model

The IP-MAP model [563] is based on the principle that information can be seen as a particular product of a manufacturing activity, and so descriptive models (and methodologies) for information quality can be based on models conceived in the last two centuries for manufacturing traditional products. The IP-MAP model is centered on the concept of *information product (IP)*, introduced in Chap. 1.

An *IP-MAP* is a graphical model designed to help people comprehend, evaluate, and describe how an information product such as an invoice, customer order, or prescription is assembled in a business process. The IP-MAP is aimed at creating a systematic representation for capturing the details associated with the manufacturing of an information product. IP-MAPs are designed to help analysts to visualize the information production process, identify ownership of process phases, understand information and organizational boundaries, and estimate time and quality metrics associated with the current production process. There are eight types of construct blocks that can be used to form the IP-MAP. Each construct block is identified by a unique name and is further described by a set of attributes (metadata). The content of metadata varies depending on the type of construct block. In Fig. 6.8, the possible types of construct blocks are shown, together with the symbol used for their representation.

An example of IP-MAP is shown in Fig. 6.9. Information products (IP in the figure) are produced by means of processing activities and data quality checks on *raw data* (RD) and semi-processed information or *component data* (CD). In the example, we assume that high schools and universities of a district have decided to cooperate in order to improve their course offering to students, avoiding overlappings and being more effective in the education value chain. To this end, high schools and universities have to share historical data on students and their curricula. Therefore, they perform a record linkage activity (we will address in depth the issue of record linkage in Chaps. 8 and 9) that matches students in their education life cycle. To reach this objective, high schools periodically supply relevant information on students; in case it is in paper format, the information has to be converted in electronic format. At this point, invalid data are filtered and matched with the database of university students. Unmatched students are sent back to high schools for clerical checks, and matched students are analyzed; the result of the analysis on curricula and course topics are sent to the advisory panel of the universities.
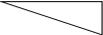
| Concept name	Symbol	Description
Source (raw input data)		Represents the source of each raw (input) data that must be available in order to produce the information product expected by the customer
Customer (output)		Represents the consumer of the information product. The consumer specifies the data elements that constitute the "finished" information products.
Data quality		Represents the checks for information quality on those data items that are essential in producing a "defect-free" information product.
Processing		Represents any calculations involving some or all of the raw input data items or component data items required to ultimately produce the information block.
Data Storage		It is any data item in a database.
Decision		It is used to describe the different decision conditions to be avaluated and the corresponding procedures for handling the incoming data items,based on the evaluation.
Business Boundary		Specifies the movement of the information product accross departmental or organization boundaries.
Information system boundary		Reflects the changes to the raw data items or component data items as they move form one information system to another type of information system. These system changes could be inter or intra business units.

Fig. 6.8 IP-MAP construct blocks
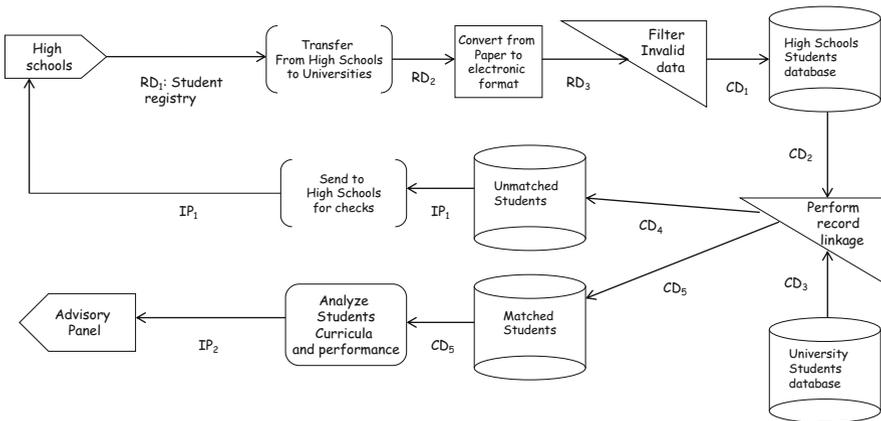
Fig. 6.9 An example of IP-MAP

6.4.2   Extensions of IP-MAP

The IP-MAP model has been extended in several directions. First, more powerful
mechanisms have been provided in [501, 549], called *event process chain diagrams*
representing the *business process overview*, the *interaction model* (how company
units interact), the *organization model* (who does what), the *component model* (what
happens), and the *data model* (what data is needed). This is done by modeling the
following:

- The event that triggers the use of data by a process
- The communication structure between sources, consumers, and organizational
  groups
- The hierarchy of organizational groups/functions
- The relationship between products, storages, and other data components
- Logical relationships between events and processes

A modeling formalism is proposed in [549], called IP-UML, extending UML
with a data quality profile based on IP-MAP. The use of UML instead of the IP-
MAP formalism has the following advantages:

1. UML is a standard language, and computer-aided tools have been implemented
   for it.
2. UML is a language supportive of analysis, design, and implementation artifacts,
   so the same language can be used in all the phases of analysis and development.
3. The expressive power of UML is higher with reference to the process modeling
   constructs.

We briefly recall that in UML (see [192, 243]), the specification of analysis and
design elements is based on the notion of a *model element*, defined as an abstraction
drawn from the system being modeled; the principal model elements are *classes*
and *relationships* between classes. A *constraint* is a semantic restriction that can
be attached to a model element. A *tag definition* specifies new kinds of properties
that may be attached to model elements. A *tagged value* specifies the actual values
of tags of individual model elements. A *stereotype* is a new model element that
extends previously defined model elements through a precise semantics. According
to the UML specification [477], "a coherent set of such extensions, defined for a
specific purpose, constitutes a *UML profile*".
The starting concepts of IP-UML are the ones defined in the IP-MAP framework;
the result of the proposed extension is a UML profile called data quality profile.
The *data quality profile* consists of three different models, namely, the data analysis
model, the quality analysis model, and the quality design model.

The *data analysis model* specifies which data are important for consumers, as
its quality is critical for the organization's success. In the data analysis model
information products, raw data and component data are represented as a stereotyped
UML class. A *quality data class* is a class labeled with this stereotype generalizes
information product classes, raw data classes, and component data classes. The

quality analysis model consists of modeling elements that can represent quality requirements of data, related to one of the dimensions typically defined for data quality. The set of dimensions proposed consists of four categories; for example, the *intrinsic information quality category* includes accuracy, objectivity, believability, and reputation. In order to model the overall set of dimension-related requirements, the following stereotypes are introduced:

1. A *quality requirement* class generalizes the set of quality requirements that can be specified on a quality data class.
2. A *quality association* class associates a quality requirement class with a quality data class. Quality requirements on data need to be verified so that, if they are not satisfied, improvement actions can be taken; therefore, a constraint is specifically introduced on the quality association.

The specification of a distinct stereotype for each quality requirement has the advantage of clearly fixing the types of requirements that can be associated with data.

The *quality design model* specifies IP-MAPs. The IP-MAP dynamic perspective, in which processes are described together with exchanged data, can be obtained by combining *UML!activity diagrams* with *UML object flow diagrams*. Activity diagrams are a special case of state diagrams in which the states are action or subactivity states and in which the transitions are triggered by completion of the actions or subactivities in the source states. Object flows are diagrams in which objects that are input or output from an action may be shown as object symbols. The following UML extensions need to be introduced, to represent IP-MAP elements:

• *Stereotyped activity*, to represent processing and data quality blocks
• *Stereotyped actor*, to represent customer, source, and data storage blocks
• *Stereotyped dependency relationship*, to give a precise semantics to the relationships between some elements

Notwithstanding the rich set of new structures introduced in the extensions of IP-MAP, such extensions suffer from different limitations, discussed in the next section, with new models that attempt to override such limitations.

6.4.3   Information Models

A first limitation of IP-MAP (and IP-MAP extensions) lies in the fact that it does not distinguish between or provide specific formalisms for *operational processes*, which make use of *elementary information*, and *decisional processes*, which use *aggregated data*. The information system of an organization is composed of both types of data that present different quality problems. So, it seems relevant to enrich models for management information systems to explicitly provide a uniform formalism to represent both types of information and their quality dimensions.

Fig. 6.10 Organizations, processes, and information flows in a cooperative information system
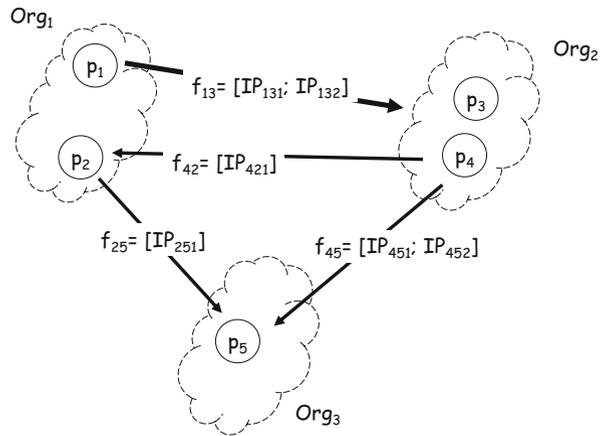
Secondly, IP-MAP does not take specific features of CIS into account. In a CIS, as Fig. 6.10 shows, an organization can be modeled as a collection of processes that transform input information flows into output information flows and that carry a stream of information products. In Fig. 6.10, three organizations are represented that exchange four information flows: two of them are composed of two information products each; the two remaining flows exchange one single information product. In the domain of a specific organization, an input flow to a process can be transformed into (1) an internal flow, (2) an input to another intraorganizational process, or (3) an output flow to one or more external organizations.

In [443–445], a comprehensive approach to overcome the above limitations is presented, discussed in the following sections.

6.4.3.1   Modeling Information Flows of an Organization

We first distinguish two different roles for organizations exchanging information flows in a CIS, namely, a *producer (organization)* when it produces flows for other organizations and a *consumer (organization)* when it receives flows from other organizations. Every organization usually plays both roles. Following traditional manufacturing practice, we characterize the quality of the individual items produced on the producer side; by extension, we associate a *quality offer profile* to a producer organization. Such a profile represents the quality that the organization is willing to offer to its customers, i.e., to other consumer organizations that require that information for use in a cooperative process. Symmetrically, on the consumer side, we define the notion of *quality demand profile* to express acceptable quality levels for the information items that consumers will acquire. Ultimately, we frame the

problem of managing information quality within an organization as the problem of matching the quality profile offered by that organization to the quality requested by the consumers of the organization. At this point, we are able to define a framework for expressing quality offer and demand in a CIS context. The framework models both the structure of a cooperative organization (*information schema*) and its quality profiles (*quality schema*; see next section) in a uniform, hierarchical way.

We start by associating quality profiles with the elementary information items that the organization produces and consumes during the execution of processes (see Fig. 6.11 for the metaschema of the information schema, represented with a class diagram in UML).

An *information flow* f is a sequence of *physical information items* (PII) that are streamed from a producer process to one or more consumer processes. For instance, given a domain entity Address, and its instance 4 Heroes Street (suitably identified using keys defined for Address), a PII would be a specific copy of J. Smith's address, produced at a particular time t by a process p_1 and sent to a process p_2 over flow f. All PIIs produced by any process at any time, referring to the same data, homogeneous in meaning, are associated with a single *logical information item* 4 Heroes Street.
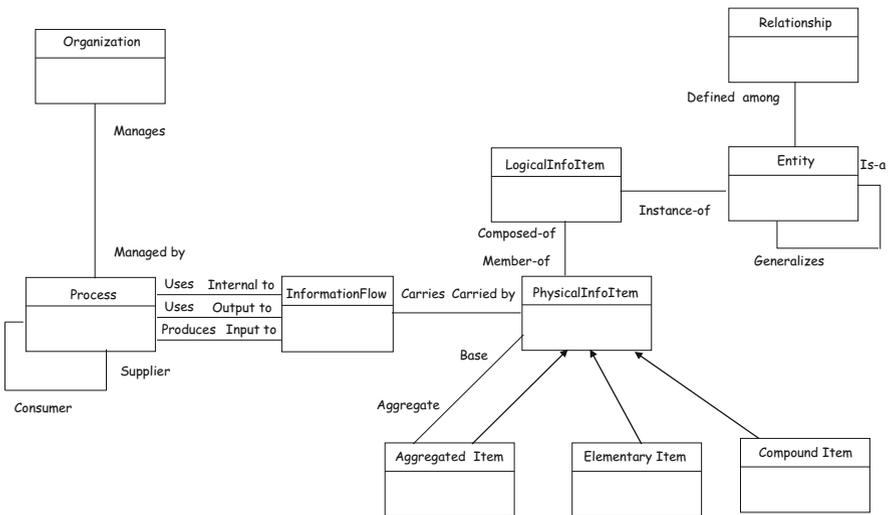

Fig. 6.11 Data, process, and organization schema

PII and logical information items describe *atomic* (or *elementary*) information items and their flow in time. As the metaschema in Fig. 6.11 shows, a *compound item* is obtained recursively from other compound or elementary items using composition functions, such as the record-type function (e.g., an Address is composed of Street, City, and ZipCode). An *aggregated item* is obtained from a collection of elementary and compound items by applying an aggregation function to them (e.g., the average income of taxpayers in a given town).

With the above representation structures, we are able to model both information flows made of elementary items and flows made of aggregated items. Finally, we associate information flows between processes and processes with organizations. Information flows are of three types: input to, output from, and internal to processes. We enrich the set of representation structures with other structures, typical of a conceptual model, such as *entity*, *relationship* among entities, and *generalization* among entities, as done in the schema in Fig. 6.11, with usual meanings in the Entity Relationship model.

6.4.3.2 A Quality Profile Model

In order to represent and compute quality profiles, associated with all the classes in the previous schema, we model the quality profile of an organization as a *data cube* on a given set of dimensions, using the *multidimensional database model* proposed in [10]. We view the quality profile of a single item as one point in a *multidimensional cube*, in which the axes include a hierarchy of entities consisting of physical and logical information items, flows, processes, organizations, and quality dimensions.

The information carried by each quality point in the resulting *quality cube* is the single quality measurement at the finest level of granularity, i.e., the quality descriptor associated with a single PII and for a single dimension. Figure 6.12 shows the *star schema*, according to the data warehouse modeling approach; it has the quality values as fact entity and the remaining ones as the dimension entities; attributes of fact and dimension entities are not shown.

The quality profiles for information flows, processes, and entire organizations are computed as appropriate aggregations from a base quality cube. Thus, once an appropriate set of *aggregation functions* (e.g., average) is defined over quality descriptors, quality profiles at each level of granularity in an organization are described within an established framework for multidimensional data. As an example, consider again Fig. 6.10, where two organizations, five processes, and four flows are defined. We may aggregate quality values along the following chain: (1) PII, (2) information flow, (3) process, and (4) organization; and, using aggregation functions, we may associate quality values with each one of the above information flows, processes, and organizations, according to the perspective we choose.
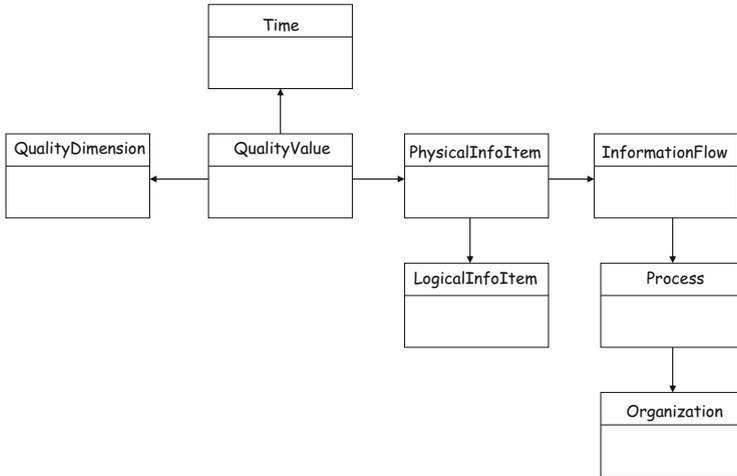
Fig. 6.12 Star schema of the data quality cube

6.5   Summary

In this chapter, we have seen several proposals for extending data, information, and process models, to provide them with structures for representing quality dimensions and for using them to measure and improve quality profiles of single information flows, processes, and entire organizations. In the following chapters, we will address the core topics of research in and experience with information quality, i.e., techniques and methodologies proposed for IQ measurement and improvement. We anticipate that such techniques and methodologies seldom rely on the proposals presented in this chapter on model extensions, with the distinctive exception of the IP-MAP model. Furthermore, only a few prototypical DBMSs have experienced the adoption of some of the approaches mentioned for data quality, among them [20]. This feeble connection is due to the complexity of the overall equipment of the representational structures proposed in the different approaches and the lack of consolidated tools and DBMSs to manage them. From an application perspective, rather than articulated data models, quality metadata descriptions can be used to document quality. As an example, in the Official Statistical domain, Eurostat provides guidelines for quality reports to be associated to datasets produced by National Statistical Institute. Such guidelines include the identification of specific quality metadata characterizing quality dimensions, i.e., accuracy and reliability, timeliness and punctuality, coherence and comparability, and accessibility and clarity [208].

The future of research on models appears to be in provenance and trustworthiness issues, as we will discuss in Chap. 14. In cooperative and Web information systems, knowing the provenance and the trustworthiness of data is crucial for the user, who may trace the history of data and increase his or her awareness in accessing and using them.