Valéria Pequeno$^1$ and Helena Galhardas$^1$ and João Moura Pires$^2$

$^1$Av. Prof. Doutor Aníbal Cavaco Silva 2744-016 Porto Salvo, Portugal
$^2$CENTRIA, Departamento de Informática,
Faculdade de Ciências e Tecnologia, FCT, Universidade Nova de Lisboa
2829-516, Caparica, Portugal

This revision: October 17, 2013
Specifying Complex Correspondences between Relational Schemas in a Data Integration Environment

Valéria Magalhães Pequeno\(^1\) and Helena Galhardas\(^1\) and João Moura Pires\(^2\)

\(^1\) INESC-ID and Technical University of Lisbon
IST Tagus Park, Av Prof Cavaco Silva 2780-990 Porto Salvo, Portugal
vmp@inesc-id.pt, helena.galhardas@ist.utl.pt

\(^2\) CENTRIA, Departamento de Informática,
Faculdade de Ciências e Tecnologia, FCT, Universidade Nova de Lisboa
2829-516, Caparica, Portugal
jmp@di.fct.unl.pt

When dealing with the data integration problem, the designer usually encounters incompatible data models characterized by differences in structure and semantics, even in a context of a same organization. In this work, we propose a declarative and formal approach to specify correspondences between relational schema components. Our Correspondence Assertions (CAs) can specify 1-to-1 and m-to-n correspondences with semantics. We propose to adapt CAs to be able to express schema matching between relational schemas, as well as to extend this formalism with new types of CAs to deal with joins, outer-joins, and data-metadata relationships. Finally, we demonstrate how mapping expressions in the form of SQL queries can be generated from CAs.

**Keywords:** Schema Matching, Correspondence Assertions, Data Integration, Relational Model

1 INTRODUCTION

A Data Integration (DI) system aims at integrating a variety of data obtained from different data sources, usually autonomous and heterogeneous, and providing a unified view of these data, often using a global schema (also named mediated schema). The mediated schema makes a bridge between the data sources and the applications that access the DI system. Data in a DI system can be physically reconciled in a repository (materialized data integration approach), or can remain at data sources and is only consolidated when a query is posed to the DI system (virtual data integration approach). A data warehouse system [Kimball et al., 2008] is a typical example of the first approach. As examples of the second approach, we can cite federated information systems [Popfinger, 2006] and mediator systems [Andreas Langegger and Blöchl, 2008]. In the present work, both scenarios can be used, but in this paper we will focus on the virtual integration approach.

One of the hardest problems to solve in DI is to define mappings between the mediated schema (the target) and each data source schema, known as the schema mapping problem. It consists of two main tasks: a) schema matching to define/generate correspondences (a.k.a. matches) between schema elements (e.g., attributes, relation, XML tags, etc.); and b) schema mapping to find data transformations that, given data instances of a source schema, obtain data instances of the target schema.
The result of schema matching is a set of correspondences that relate elements of a source to elements of the target. Each correspondence specifies the elements that refer to the same real world entity [Doan et al., 2012]. Once the schema matching is performed, the correspondences are used to generate the schema mappings. For example, a schema mapping can be codified through an SQL query that transforms data from the source to the target.

Extensive research on schema matching has been carried out in recent years [Cruz et al., 2009, Seligman et al., 2010, Bonifati et al., 2011]. The majority of the works on this subject identifies 1-1 correspondences between elements of two schemas. For example, a 1-1 correspondence can specify that element title in one schema matches element film in another schema, or that relation genre matches relation category. This kind of schema matching is known in the literature as basic matching. Goods surveys can be found in [Rahm and Bernstein, 2001, Shvaiko and Euzenat, 2005].

While basic matching is common, it leaves out numerous correspondences of practical interest, in particular when we consider DI systems. Thus, more complex matches are necessary. A complex matching specifies 1:n, m:n, or n:1 correspondences between elements of two schemas. For example, it may specify that totalPrice corresponds to unitPrice * quantity; or that name matches concatenate(firstName, lastName), where concatenate is a function that applies to two strings and returns a concatenated string; or even that the average departmental salary avgWage corresponds to grouping the salaries (salary) of all employees (emp) by department (dept). [Doan, 2002, Massmann et al., 2011, Mork et al., 2008] are examples of approaches that propose formalisms to specify complex matches.

---

* Schema S1

MOVIEd.id, film, catId, year, summary
FK1(MOVIE, ⟨catId⟩), CATEGORY, ⟨catId⟩)
TAPE(number, format, id)
FK2(TAPE, ⟨id⟩), MOVIE, ⟨id⟩)
STARS(id, name, role)
FK3(STARS, ⟨id⟩), MOVIE, ⟨id⟩)
CATEGORY(catId, nameCat)

* Schema S2

FILM(id, title, year, rate)
SHOWTIME(id, location, time, city)
FK4(SHOWTIME, ⟨id⟩), FILM, ⟨id⟩)

* Schema M

MOVIE(title, genre, year, description)
CAST(movie, actor, director)
FK5(CAST, ⟨movie⟩), MOVIE, ⟨title⟩)
SCHEDULE(movie, cinema, startTime)
FK6(SCHEDULE, ⟨movie⟩), MOVIE, ⟨title⟩)
REMAKES(title, nvYear, ovYear)
FK7(REMAKES, ⟨title⟩), MOVIE, ⟨title⟩)
RATING(rate, quantity)

---

Fig. 1. Example of source schemas and a mediated schema.

---

1 We use bold to represent attribute names and UPPERCASE to represent relation names.
Some researchers go beyond dealing with complex matches and add semantics to the correspondences, in order to improve the overall matching quality. Consider as a running example the source schemas $S_1$ and $S_2$ in Figure 1, which contain information about movies. $S_1$ keeps a catalog of movies with information about different types of media (dvd, blue rays, etc.) in which the movies are available. In this paper, the relations of $S_1$ are: MOVIE, TAPE, STARS, and CATEGORY. The names of the relations and attributes are mostly self-explanatory. MOVIE keeps information about movies of different categories (drama, romance, western, etc.). TAPE stores information about format of tapes (vhs, dvd, blu-ray, etc.). STARS keeps information about movie stars (actors and directors). CATEGORY stores information about types of categories of the movies. The attributes in $S_1$ have the following meaning: id is the movie identifier, year is the year of a movie, film is the title of a movie, number is the tape identifier, format is the type of the tape (dvd, blu-ray, etc.), name can be an actor or a director name, role is the role of the star in the movie: director or actor, catId is the category identifier, and nameCat is the name of a movie category. $S_2$ stores general information about movies and places (in different cities) where movies are being shown. We assume that $S_1$ can store older movies than $S_2$. In this paper, the relations of $S_2$ are: FILM, and SHOWTIME. FILM keeps general information about movies. SHOWTIME contains information about the schedule of movies in various cities. The attributes in $S_2$ have the following meaning: id is the movie identifier, year is the year of a movie, title is the title of a movie, rate is the classification of a movie with regard the audience, location is the place where the movie is shown, time is the date when the movie is shown, and city is the name of the city where the movie is shown.

The mediated schema $M$, also shown in Figure 1, provides a unified user view of movies currently shown in cinemas. It is populated by information from schemas $S_1$ and $S_2$. By combining $S_1$ and $S_2$ information, we can present a unified and more complete view to the user, since some movie information may be partially and overlapping over $S_1$ and $S_2$. The relation $M$.MOVIE stores movies shown currently at a cinema\(^4\). The relation $M$.CAST keeps information about movie stars. The relation $M$.SCHEDULE contains information about the schedule of movies shown in Lisbon. The relation $M$.REMAKES keeps the years of movies for which there is at least one remake. The relation $M$.RATING stores the classification of movies with regard to suitability audience. The attributes in $M$ have the following meaning: title is the title of a movie, year is the year of the movie, description is the summary of a movie, cinema is the place where the movie is shown, genre is the category of a movie, nyYear is the year of the most recent version of a movie, ovYear is the year of the older versions of a movie, rate is the classification of the movie with regard the audience (M/4, M/6, etc. in portuguese classification), quantity is the total of movies with the same rating.

Let the schemas $S_1$, $S_2$, and $M$, we can consider the correspondences between the source schemas $S_1$ and $S_2$, and the target schema $M$. In a first example, we can state that $M$.SCHEDULE corresponds to $S_2$.SHOWTIME, because both relations store information regarding the same real world concept. However, in this correspondence, it is not clear that $M$.SCHEDULE only keeps

\(^4\)We use a path representation: an attribute A of a given relation R in a given database schema D is referred to as D.R.A. For simplicity, we omit the database schema when the context is clear.
schedules about movies shown in Lisbon. The additional information: M.SCHEDULE corresponds to S₂.SHOWTIME when S₂.SHOWTIME.city = “Lisbon”, specifies better the matching.

As a second example, consider the attributes S₂.SHOWTIME.id and M.SCHEDULE.movie, which uniquely identify a movie in the corresponding relations. The domain of the former is an integer (the identifier of the movie id), while the domain of the latter is a string (the title of the movie movie). Since M.SCHEDULE corresponds to S₂.SHOWTIME, we need to relate M.SCHEDULE.movie to an element of S₂.SHOWTIME that identifies a movie. However, S₂.SHOWTIME does not store the title of movies. Moreover, S₂.SHOWTIME.id, which is used to identify a movie in S₂.SCHEDULE, has type and domain that are different from the type and domain of M.SCHEDULE.movie. Thus, S₂.SHOWTIME.id and M.SCHEDULE.movie cannot be matched. Knowing that S₂.SHOWTIME.id is a foreign key that refers to S₂.FILM and that S₂.FILM.title stores the title of a movie, the correct matching would then be: M.SCHEDULE.movie corresponds to S₂.FILM.title, when S₂.SHOWTIME.id = S₂.FILM.id.

The works reported in [Pequeno and Pires, 2009, Gal, 2011, Bohannon et al., 2006, Pequeno, 2011, Vidal and Lóscio, 1999] propose schema matching approaches that can specify correspondences to deal with situations as required in the first example, while [Pequeno and Pires, 2009, Pequeno, 2011, Vidal and Lóscio, 1999] propose correspondences to deal with the matching required in the second example. The reader can see other proposals to add semantics to schema matching in [Massmann et al., 2011, Dhamankar et al., 2004, Magnani et al., 2005]. However, the following situations have not been fully covered yet:

1. **Correspondences between relations involving a join with inequality conditions:** Consider the relation M.REMAKES that keeps a list of remakes with the years of the oldest versions. Knowing that S₂.FILM keeps current movies and S₁.MOVIE may contain older versions of the same movie, we want to indicate which of the current movies are remakes and store this information in M.REMAKES. The correspondence between these relations can be specified as: M.REMAKES corresponds to S₂.FILM join S₁.MOVIE where S₂.FILM.title = S₁.MOVIE.film and S₂.FILM.year > S₁.MOVIE.year. Usual schema matching approaches cannot specify this correspondence, because join conditions are not explicitly defined in schema matching. Moreover, join paths are normally automatically discovered in the schema mapping phase [Yan et al., 2001], and the algorithms used can only find equi-join conditions, so they cannot automatically discover the condition S₂.FILM.year > S₁.MOVIE.year. Hence, we need a schema matching approach that makes it possible to specify the join between relations and allows general join conditions containing operators different from equality.

2. **Correspondences between relations involving outer-joins (full, left, or right):** We want to indicate how M.MOVIE is related to source schemas S₁ and S₂. M.MOVIE and S₂.FILM represent the same concept of the real world (i.e., both relations store current movies shown at a cinema). However, it is not enough to specify that M.MOVIE matches S₂.FILM, because there are attributes in S₁.MOVIE (namely, category and summary) that contain information required in the schema of M.MOVIE. Hence, we should specify that M.MOVIE is related to both S₁.MOVIE and S₂.FILM. However, it is not correct we simply match M.MOVIE to S₁.MOVIE because S₁.MOVIE can store movies that are not being shown in a cinema anymore and M.MOVIE can store recent movies that are not available in dvds yet. In resume, we should
specify that: \textsc{M.Movie} corresponds to \textsc{S}_2.\textsc{Film left outer-join} \textsc{S}_1.\textsc{Movie} on \textsc{S}_2.\textsc{Film}.\textsc{film} = \textsc{S}_1.\textsc{Movie}.\textsc{film} and \textsc{S}_2.\textsc{Film}.\textsc{year} = \textsc{S}_1.\textsc{Movie}.\textsc{year}. Note that the condition \textsc{S}_2.\textsc{Film}.\textsc{film} = \textsc{S}_1.\textsc{Movie}.\textsc{film} and \textsc{S}_2.\textsc{Film}.\textsc{year} = \textsc{S}_1.\textsc{Movie}.\textsc{year} guarantees that we refer to a same movie stored in both \textsc{S}_1.\textsc{Movie} and \textsc{S}_2.\textsc{Film}. Again, we cannot specify this type of correspondence since joins (and their variants) are not explicitly defined in current schema matching approaches.

3. Correspondences between data and metadata: Consider the relations \textsc{S}_1.\textsc{Stars} and \textsc{M.Cast}. Both keep information about the relationship between a movie, an actor, and a director. We want to indicate that \textsc{M.Cast} corresponds to \textsc{S}_1.\textsc{Stars} since they represent the same concept in the real world. In addition, we want to specify the correspondences between the attributes of these relations. Knowing that \textsc{S}_1.\textsc{Stars}.\textsc{name} can be an actor name or a director name, we would like to specify that \textsc{M.Cast}.\textsc{actor} corresponds to \textsc{S}_1.\textsc{Stars}.\textsc{name} when \textsc{S}_1.\textsc{Stars}.\textsc{role} = “actor” and that \textsc{M.Cast}.\textsc{director} corresponds to \textsc{S}_1.\textsc{Stars}.\textsc{name} when \textsc{S}_1.\textsc{Stars}.\textsc{role} = “director”. However, we cannot specify these correspondences using traditional schema matching approaches, because these correspondences involve semantics not covered yet by these approaches. Actually, we can only specify that \textsc{M.Cast}.\textsc{actor} matches to \textsc{S}_1.\textsc{Stars}.\textsc{name} and \textsc{M.Cast}.\textsc{director} matches to \textsc{S}_1.\textsc{Stars}.\textsc{name}.

In order to deal with these situations, we propose to use a formalism based on CAs [Pequeno and Pires, 2009, Pequeno and Aparício, 2005, Pequeno, 2011]. Using CAs, we can declaratively specify basic and complex mappings with semantics. We propose to adapt CAs to be able to express schema matching between relational schemas, as well as to extend this formalism with new types of CAs to deal with joins, outer-joins, and data-metadata relationships. Finally, we demonstrate how mapping expressions in the form of SQL queries can be generated from CAs.

The remainder of the paper is structured as follows. In Section 2, we present the necessary background in CAs. In Section 3, we propose new CAs to deal with join operators and metadata. Section 4 shows how generate mapping expressions from CAs. Section 5 describes the related work. Finally, Section 6 concludes and describes future work.

2 Background

In this section, we present the basic terminology used in this paper. We also review the different classes of CAs, and adapt them to the Relational Data Model (RDM).\footnote{In this work, the information sources and mediated schemas are defined using the RDM as proposed by Codd in [Codd, 1970] that only allows first normal form relations.}

2.1 Basic concept and notation

We assume that the reader is familiar with the relational concepts. We denote a relation scheme as \textsc{R}(\textsc{A}_1, \textsc{A}_2, \ldots, \textsc{A}_n). We use, as in [Vidal et al., 2013], the notation \textsc{FK}(\textsc{R}:\textsc{L}, \textsc{S}:\textsc{K}) to denote a foreign key, named \textsc{FK}, where \textsc{R} and \textsc{S} are relation names and \textsc{L} and \textsc{K} are list of attributes from \textsc{R} and \textsc{S}, respectively, with the same length. We also say that \textsc{FK} relates \textsc{R} and \textsc{S}.\footnote{In this work, the information sources and mediated schemas are defined using the RDM as proposed by Codd in [Codd, 1970] that only allows first normal form relations.}
A relational schema is a pair $S = (R, \Omega)$, where $R$ is a set of relation schemes and $\Omega$ is a set of relational constraints such that: (i) $\Omega$ has a unique primary key for each relation scheme in $R$; (ii) if $\Omega$ has a foreign key of the form $FK(R:L, S:K)$, then $\Omega$ also has a constraint indicating that $K$ is the primary key of $S$. Given a relation scheme $R(A_1, A_2, \ldots, A_n)$ and a tuple variable $t$ over $R$, we use $t[A_i]$ to denote the projection of $t$ over $A_i$.

Let $S = (R, \Omega)$ be a relational schema and $R$ and $T$ be relation names of relation schemes of $S$. We denote $\varrho = FK_1 \bullet FK_2 \bullet \ldots \bullet FK_{n-1}$ a path from $R$ to $T$ if there is a list $R_1, \ldots, R_n$ of relation schemes in $S$ such that $R_1 = R$, $R_n = T$, and $FK_i$ relates $R_i$ and $R_{i+1}$. We say that tuples of $R$ references tuples of $T$ through $\varrho$. For instance, consider the schema $S_1$ in Figure 1, $\varrho = FK_2 \bullet FK_1$ is a path from TAPE to CATEGORY.

### 2.2 Definition of correspondence assertions

We use Correspondence Assertions (CAs) in order to express schema matchings between schema elements. CAs are formal expressions of the general form $\psi: T \leftarrow S$, where $\psi$ is the name of the CA, $T$ is an expression formed by elements of the target schema, and $S$ is an expression formed by elements of a source schema. The symbol “$\leftarrow$” means “is matched from”.

In accordance to [Pequeno and Pires, 2009, Pequeno, 2011], there are four types of CAs: Relation Correspondence Assertion (RCA), Attribute Correspondence Assertion (ACA), Summation Correspondence Assertion (SCA), and Aggregation Correspondence Assertion (GCA). RCAs and SCAs specify the relationship between relations of distinct schemas, while ACAs and GCAs specify the relationship between attributes of relations of distinct schemas. We now shortly describe each type of CA, adapting them to the RDM.

**Definition 1.** Let $S_i$ be relational schemas, $R_i$ relation schemes of $S_i$ (for $1 \leq i \leq n$), and $\sigma$ is a selection over $R_2$. A Relation Correspondence Assertion (RCA) is an expression of one of the following forms:

1. $\psi: S_i[R_1] \leftarrow S_2[R_2]$. (equivalence)
2. $\psi: S_i[R_1] \leftarrow S_2[R_2][\sigma]$. (selection)
3. $\psi: S_i[R_1] \leftarrow S_2[R_2] - S_3[R_3]$. (difference)
4. $\psi: S_i[R_1] \leftarrow S_2[R_2] \cup S_3[R_3] \cup \cdots \cup S_n[R_n]$. (union)
5. $\psi: S_i[R_1] \leftarrow S_2[R_2] \cap S_3[R_3] \cap \cdots \cap S_n[R_n]$. (intersection)

We say that $\psi$ matches $R_1$ and $R_i$, $2 \leq i \leq n$. □

RCAs express the different kinds of semantic equivalent relations. Two relations $R_1$ and $R_2$ are semantically equivalent if they represent the same real concept and there is one-to-one correspondence between their instances. For instance, $\psi_1$, shown in Figure 2, is an example of a RCA.

$\psi_1$ specifies that $M: SCHEDULE$ is semantically equivalent to $S_2:SHOWTIME$ when the condition $S_2:SHOWTIME:city = \text{"Lisbon"}$ is satisfied. This means that only a subset of tuples of

---

6 In [Pequeno and Pires, 2009, Pequeno, 2011] the CAs are named, respectively, Extension Correspondence Assertion, Property Correspondence Assertion, Summation Correspondence Assertion, and Aggregation Correspondence Assertion. Here we rename them to conform to the relational concept.
ACAs specify the relationship between the attributes of relations that are matched by a RCA. They allow to define 1:1, 1:n, n:1, or m:n relationships between attributes of relations of different schemas. For a simple example see the ACA \( \psi_1 \) presented in Figure 2. It specifies a one-to-one relationship between the attributes \( \text{M}.\text{SCHEDULE}.\text{cinema} \) and \( \text{S}_2.\text{SHOWTIME}.\text{location} \). \( \psi_3 \) is a more complex match example. It specifies the correspondence between \( \text{M}.\text{SCHEDULE}.\text{movie} \) and \( \text{S}_2.\text{FILM}.\text{title} \) through a path from \( \text{SHOWTIME} \) to \( \text{FILM} \). \( \psi_4 \) specifies how to identify tuples of \( \text{M}.\text{SCHEDULE} \) and tuples of \( \text{S}_2.\text{SHOWTIME} \) when these represent the same instance in the real world (named tuples semantically equivalent). In our example, the attributes \text{movie} \) and \text{cinema} \) of \( \text{M}.\text{SCHEDULE} \) and the attributes \text{title} \) of \( \text{S}_2.\text{FILM} \) and \text{location} \) of \( \text{S}_2.\text{SHOWTIME} \) are used to identify the tuples semantically equivalent. This type of ACA is named same as.

**Definition 3.** Let \( \mathbf{S}_i \) be relational schemas, \( \mathbf{R}_i \) relation schemes of \( \mathbf{S}_i \), and \( \sigma \) is a selection over \( \mathbf{R}_2 \). A Summation Correspondence Assertion (SCA) is an expression of one of the following forms:

1. \( \psi: \mathbf{S}_i[\mathbf{R}_1] \leftarrow \text{groupby}(\mathbf{S}_2[\mathbf{R}_2](\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_n)) \).  

Fig. 2. Examples of 1:1 correspondence assertions.

\( \mathbf{S}_2.\text{SHOWTIME} \), those that satisfy the condition \( \mathbf{S}_2.\text{SHOWTIME}.\text{city} = \text{“Lisbon”} \), are involved in the match. \( \psi_1 \) is an example of a RCA of selection. The RCA of equivalence specifies that a relation is semantically equivalent to another relation. The RCA of union, intersection, and difference were taken from the set theory.
We say that \( \psi \) matches \( R_1 \) and \( R_2 \).

SCAs specify 1:n, n:1, m:n relationships between relations of distinct schemas. Here we use the symbol “\( \leftarrow \) instead of “\( \rightarrow \)” in order to emphasize that the correspondence is not 1:1 as is usual in the most part of schema matching approaches. SCAs are used to describe the summary of a relation whose tuples are related to the tuples of another relation by breaking them into logical groups. This means that a SCA has only the necessary information to indicate which grouping field is involved in the relationship and indicates the process used to grouping the tuples. \( \psi_4 \) shown in Figure 3 is a good example of a SCA.

![Fig. 3. Examples of m:n correspondence assertions.](image-url)

\( \psi_5 \) specifies that there is a relationship between \( M.RATING \) and \( S_2.FILM \), since \( M.RATING \) keeps a list of total of movies by rating and \( S_2.FILM \) keeps current movies with their rates (\( \text{rate} \)). In particular, \( \psi_5 \) specifies that \( M.RATING \) is grouped by \( \text{rate} \) of \( S_2.FILM \).

**Definition 4.** Let \( S_1, S_2 \) relational schemas; \( R_1, R_2 \) relation scheme of, respectively, \( S_1 \) and \( S_2 \); \( A_i, A'_i \) attributes of, respectively, \( R_1 \) and \( R_2 \) (for \( 1 \leq i \leq n \)), \( \varrho \) a path from \( R_2 \) to \( R_k \), with \( R_k \) being a relation of schema \( S_2 \); \( A_k \) an attribute of \( R_k \); and \( v \) a value. Consider also \( p \) a boolean condition over attributes of \( R_2 \) and \( E \) an expression with one of the following forms: i) \( S_2[R_2] \cdot A'_2; \) ii) \( S_2[R_2] \cdot \varrho/A_k; \) iii) \( \varphi(E) \). A Grouping Correspondence Assertion (GCA) is an expression of one of the following forms:

1. \( \psi: S_1[R_1] \cdot A_1 \leftarrow S_2[R_2] \cdot A'_2 \). (compatible domain)
2. \( \psi: S_1[R_1] \cdot A_1 \leftarrow S_2[R_2] \cdot \varrho/A_k \). (denormalisation)
3. \( \psi: S_1[R_1] \cdot A_1 \leftarrow \varphi(E_1, E_2, \ldots, E_n) \). (calculated attribute)
4. \( \psi: S_1[R_1] \cdot A_1 \leftarrow (E_1, p_1); \ldots; (E_i, p_i); v \). (case based)
5. \( \psi: S_1[R_1] \cdot A_1 \leftarrow \gamma(S_2[R_2] \cdot A_2) \). (aggregation)
6. \( \psi: S_1[R_1] \cdot A_1 \leftarrow \gamma(S_2[R_2] \cdot \varrho/A_k) \). (aggregation)
7. \( \psi: S_1[R_1] \cdot A_1 \leftarrow \gamma(\varphi(E_1, E_2, \ldots, E_n)) \). (aggregation)
8. \( \psi: S_1[R_1] \cdot A_1 \leftarrow \gamma(S_2[R_2] \cdot A_2, p) \). (aggregation)
9. \( \psi: S_1[R_1] \cdot A_1 \leftarrow \gamma(S_2[R_2] \cdot \varrho/A_k, p) \). (aggregation)
10. \( \psi: S_1[R_1] \cdot A_1 \leftarrow \gamma(\varphi(E_1, E_2, \ldots, E_n), p) \). (aggregation)

Where \( \varphi \) is a function over attributes of \( R_2 \), and \( \gamma \) is one of the aggregate function: sum (summation), max (maximum), min (minimum), avg (average), or count. We say that \( \psi \) matches \( R_1 \) and \( R_2 \).
GCAs specify the relationship 1:1, 1:n, n:1, or m:n between attributes of relations that are matched by a SCA. For instance, the correspondence between the attributes of \( \text{M.rating} \) and \( \text{S}_2.\text{film} \) are specified through GCAs \( \psi_6 \) and \( \psi_7 \) (shown in Figure 3). \( \psi_6 \) specifies that \( \text{M.rating}.\text{rate} \) corresponds to each distinct value of \( \text{S}_2.\text{film}.\text{rate} \). \( \psi_7 \) specifies that \( \text{M.rating}.\text{quantity} \) corresponds to the counting of all distinct value of \( \text{S}_2.\text{film}.\text{rate} \).

**Definition 5.** A matching between a schema \( S \) and a schema \( T \) is a set \( A \) of CAs such that:

1. if \( A \) has an ACA \( \psi \) such that \( \psi \) matches \( R_1 \) and \( R_2 \), then \( A \) have a RCA \( \psi' \) that matches \( R_1 \) and \( R_2 \).
2. if \( A \) has a GCA \( \psi \) such that \( \psi \) matches \( R_1 \) and \( R_2 \), then \( A \) have a SCA \( \psi' \) that matches \( R_1 \) and \( R_2 \).
3. if \( A \) has a RCA \( \psi \) such that \( \psi \) matches \( R_1 \) and \( R_2 \), then \( A \) have an ACA “same as” \( \psi' \) that matches \( R_1 \) and \( R_2 \).

### 3 Specifying new CAs

In Section 1, we identify three types of relationships between schemas elements that are not properly handled in current schema matching approaches: 1) matches involving explicit join conditions; 2) matches involving outer-joins; and 3) matches involving data-metadata. Join (and outer joins) relationships can express one-to-one or one-to-many correspondences between the relations involved. Matches involving data-metadata can express many-to-many correspondences between the relations involved. So, we extend our previous definitions of RCA and SCA in order to better specify these types of matchings. In the following text consider \( S_i \) relational schemas, \( R_i \) relation schemes of \( S_i \) (for \( 1 \leq i \leq 3 \)), \( \theta \) is a join condition between \( R_2 \) and \( R_3 \), and \( A_j \) attributes of \( R_2 \) (for \( 1 \leq j \leq n \)).

**Definition 6.** A Relation Correspondence Assertion (RCA) is an expression of one of the following forms:

1. Expressions as those defined in Definition 1.
2. \( \psi: S_i[R_1] \leftarrow S_2[R_2] \bowtie S_i[R_3]\theta. \) (inner-join)
3. \( \psi: S_i[R_1] \leftarrow S_2[R_2] \bowtie S_i[R_3]\theta. \) (left outer join)
4. \( \psi: S_i[R_1] \leftarrow S_2[R_2] \bowtie S_i[R_3]\theta. \) (right outer join)
5. \( \psi: S_i[R_1] \leftarrow S_2[R_2] \bowtie S_i[R_3]\theta. \) (full outer join)

**Definition 7.** A Summation Correspondence Assertion (SCA) is an expression of one of the following forms:

1. Expressions as those defined in 3.
2. \( \psi: S_i[R_1] \leftarrow S_2[R_2] \bowtie S_i[R_3]\theta. \) (inner-join)
3. \( \psi: S_i[R_1] \leftarrow S_2[R_2] \bowtie S_i[R_3]\theta. \) (left outer join)
4. \( \psi: S_i[R_1] \leftarrow S_2[R_2] \bowtie S_i[R_3]\theta. \) (right outer join)
5. \( \psi: S_i[R_1] \leftarrow S_2[R_2] \bowtie S_i[R_3]\theta. \) (full outer join)
6. \( \psi: S_1[R_1] \iff \text{metadata}(S_2[R_2](A_1, A_2, \ldots, A_n)) \). 

Consider the three examples about join, left outer-join, and data-metadata correspondences described in Section 1. The correspondence between \( M \text{.REMAKES} \) and both \( S_1 \text{.MOVIES} \) and \( S_2 \text{.FILM} \) can be specified by the SCA \( \psi_8 \) shown in Figure 4. The relationship between the attributes of \( M \text{.REMAKES} \), \( S_1 \text{.MOVIE} \) and \( S_2 \text{.FILM} \) can be specified using the usual GCAs and are not presented here.

\[
\begin{array}{l|c}
\psi_8: & M \text{[REMAKES]} \iff S_2 \text{[FILM]} \bowtie S_1 \text{[MOVIE]} \Rightarrow (S_2 \text{[FILM]} \cdot \text{title} = S_1 \text{[MOVIE]} \cdot \text{film} \\
 & \text{and} \quad S_2 \text{[FILM]} \cdot \text{year} > S_1 \text{[MOVIE]} \cdot \text{year} ) & \text{SCA} \\
\psi_9: & M \text{[MOVIE]} \iff S_2 \text{[FILM]} \Rightarrow S_1 \text{[MOVIE]} \Rightarrow (S_2 \text{[FILM]} \cdot \text{title} = S_1 \text{[MOVIE]} \cdot \text{film} \\
 & \text{and} \quad S_2 \text{[FILM]} \cdot \text{year} = S_1 \text{[MOVIE]} \cdot \text{year} ) & \text{RCA} \\
\psi_{10}: & M \text{[CAST]} \iff \text{promoteAttribute}(S_1 \text{[STARS]} \cdot \text{id}) & \text{SCA} \\
\end{array}
\]

\textbf{Fig. 4.} Examples of CAs involving joins, outer-joins and data-metadata.

The correspondence between \( M \text{.MOVIE} \) and both \( S_2 \text{.FILM} \) and \( S_1 \text{.MOVIE} \) involves a left outer-join that can be specified by the RCAs \( \psi_9 \), shown in Figure 4. \( \psi_{10} \) to \( \psi_{14} \), shown in Figure 5, are examples of ACAs between the attributes of \( M \text{.MOVIE}, S_2 \text{.FILM}, \) and \( S_1 \text{.MOVIE} \).

\[
\begin{array}{l|c}
\psi_{11}: & M \text{[MOVIE]} \cdot \text{title} \iff S_2 \text{[FILM]} \cdot \text{title} & \text{ACA} \\
\psi_{12}: & M \text{[MOVIE]} \cdot \text{year} \iff S_2 \text{[FILM]} \cdot \text{year} & \text{ACA} \\
\psi_{13}: & M \text{[MOVIE]} \cdot \text{genre} \iff S_1 \text{[MOVIE]} \cdot \text{category} & \text{ACA} \\
\psi_{14}: & M \text{[MOVIE]} \cdot \text{description} \iff S_1 \text{[MOVIE]} \cdot \text{summary} & \text{ACA} \\
\psi_{15}: & M \text{[CAST]} \cdot \text{actor} \iff (S_1 \text{[STARS]} \cdot \text{name}, S_1 \text{[STARS]} \cdot \text{role} = "actor") & \text{GCA} \\
\psi_{16}: & M \text{[CAST]} \cdot \text{director} \iff (S_1 \text{[STARS]} \cdot \text{name}, S_1 \text{[STARS]} \cdot \text{role} = "director") & \text{GCA} \\
\psi_{17}: & M \text{[CAST]} \cdot \text{movie} \iff S_1 \text{[STARS]} \cdot \text{FK4} \cdot \text{film} & \text{GCA} \\
\end{array}
\]

\textbf{Fig. 5.} Examples of ACAs and GCAs.

The correspondence between \( M \text{.CAST} \) and \( S_1 \text{.STARS} \) involves a data-metadata relationship. In our approach it can be specified through a SCA, and their attributes must be specified using GCAs. Thus, the relationship between \( M \text{.CAST} \) and \( S_1 \text{.STARS} \) is specified by the SCA \( \psi_{10} \). \( \psi_{10} \) indicates that \( S_1 \text{.STARS} \) matches \( M \text{.CAST} \), being that each group of values in \( S_1 \text{.STARS} \) with the same \text{id} will correspond to one tuple in \( M \text{.CAST} \). The correspondences between attributes of \( M \text{.CAST} \) and \( S_1 \text{.STARS} \) are specified through GCAs \( \psi_{15} \) to \( \psi_{17} \), shown in Figure 5. \( \psi_{15} \), for example, is a case-base GCA that matches \( M \text{.CAST} \cdot \text{actor} \) to \( S_1 \text{.STARS} \cdot \text{name} \). It indicates that values should be stored in \( M \text{.CAST} \cdot \text{actor} \) only when \( S_1 \text{.STARS} \cdot \text{role} = "actor" \).
Once the schema matching is finished, the CAs generated can be used, for example, to generate mapping expressions that convert data sources into data target. This is the subject of the next section.

4 From CAs to mapping expressions

In our proposal, the process to create queries to transform data from a schema to another one consists of three steps:

1. Indicate the source schemas and the mediated schema using a high-level data model. In our case, we use the RDM.
2. Run the schema matching and identify the CAs that formally specify the relationships between the mediated schema and the source schemas.
3. Generate a set of queries based on the CAs generated in the step 2.

In order to illustrate our approach, consider the mediated schema $M$ and the sources schemas $S_1$ and $S_2$ shown in Figure 1. The next step, in the process to create queries to transform data from a schema to another, is generate the CAs. This process consists of the following steps:

1. To each relation $R_T$ of the target do:
   (a) Identify the correspondences at a relation level (i.e., if there is a RCA or a SCA matching a target relation $R_T$ and some source relation $R_S$).
   (b) Identify the correspondences at an attribute level: 1) identify the ACAs between the attributes of $R_T$ and $R_S$ (if there is a RCA between $R_T$ and $R_S$); 2) identify the GCAs between the attributes of $R_T$ and $R_S$ (if there is a SCA between $R_T$ and $R_S$).
   (c) Determine which RCAs and SCAs can be combined to form a single CA.

Some examples of RCAs, ACAs, SCAs, and GCAs between elements of the mediated schema $M$ and the source schemas $S_1$ and $S_2$ can be found in Figures 4 and 5. The final step in the process of creating queries to transform data from a schema to another is the generation of the queries. In our proposal, they are defined based on the definition of schemas and the CAs. In this work, we decide use the SQL:2011 [Taylor, 2013] for the definition of our queries.

Let $A$ be a set of CAs that defines a matching between the schemas $S_1$, $S_2$ and $T$, that is, $A$ satisfies the conditions stated in Definition 5. Table 1 shows the statements of SQL queries induced by, respectively, the RCAs and ACAs in $A$.

In Table 1, we assume that $K_2$ and $K_3$ are short expressions to indicate primary keys. Later they will be properly changed to expressions of the form $A_1 = A_1' \ and \ldots \ and \ A_n = A_n'$, with $K_2 = \langle A_1, \ldots, A_n \rangle$ and $K_3 = \langle A_1', \ldots, A_n' \rangle$. $\varphi()$ can be a user-defined function or a SQL function. We assume that $K$ and $FK$ are short expressions to indicate, respectively, a primary key and a foreign key. The symbol $v$ indicates a value (integer, string, float, etc.). The expression $[\text{where } | \text{ on}]$ in item 5 indicates that the ACA “same as” can be used in a where clause or in an on clause.

Figure 6 presents the definition of the query to transform data from $S_1.\text{MOVIE}$ and $S_2.\text{FILM}$ to $M.\text{MOVIE}$ induced by the RCA $\psi_9$ and ACAs $\psi_{11}$, $\psi_{12}$, $\psi_{13}$, and $\psi_{14}$. The “select” clause (in line 2) is denoted based on ACAs $\psi_{11}$, $\psi_{12}$, $\psi_{13}$ and $\psi_{14}$. The “from” clause (in line 3) correctly
Table 1. SQL Statements induced by RCAs.

<table>
<thead>
<tr>
<th>Step</th>
<th>Statement</th>
<th>From</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>$S_1[R_1] \leftarrow S_2[R_2]$</td>
<td>$S_2,R_2$ as $T_2$</td>
</tr>
<tr>
<td>2.</td>
<td>$S_1[R_1] \leftarrow S_2[R_2(\sigma)]$</td>
<td>$S_2,R_2$ as $T_2$ where $p$</td>
</tr>
<tr>
<td>3.</td>
<td>$S_1[R_1] \leftarrow S_2[R_2] \bowtie S_3[R_3]$</td>
<td>$S_2,R_2$ as $T_1$</td>
</tr>
<tr>
<td></td>
<td>$S_1[R_1] \leftarrow S_2[R_2] \bowtie \bowtie S_3<a href="%5Ctheta">R_3</a>$</td>
<td>left join $S_3,R_3$ as $T_3$ on $T_2.K_2 = T_3.K_3$ where $T_3.K_3$ is null</td>
</tr>
<tr>
<td>4.</td>
<td>$S_1[R_1] \leftarrow S_2[R_2] \cup S_3[R_3]$</td>
<td>$S_2,R_2$ as $T_2$</td>
</tr>
<tr>
<td>5.</td>
<td>$S_1[R_1] \leftarrow S_2[R_2] \cap S_3[R_4]$</td>
<td>$S_2,R_2$ as $T_2$</td>
</tr>
<tr>
<td>6.</td>
<td>$S_1[R_1] \leftarrow S_2[R_2] \bowtie S_3<a href="%5Ctheta">R_3</a>$</td>
<td>$S_2,R_2$ as $T_2$</td>
</tr>
<tr>
<td></td>
<td>$S_1[R_1] \leftarrow S_2[R_2] \bowtie \bowtie S_3<a href="%5Ctheta">R_3</a>$</td>
<td>inner join $S_3,R_3$ as $T_3$ on $\theta$</td>
</tr>
<tr>
<td>7.</td>
<td>$S_1[R_1] \leftarrow S_2[R_2] \bowtie S_3<a href="%5Ctheta">R_3</a>$</td>
<td>$S_2,R_2$ as $T_2$</td>
</tr>
<tr>
<td></td>
<td>$S_1[R_1] \leftarrow S_2[R_2] \bowtie \bowtie S_3<a href="%5Ctheta">R_3</a>$</td>
<td>left join $S_3,R_3$ as $T_3$ on $\theta$</td>
</tr>
<tr>
<td>8.</td>
<td>$S_1[R_1] \leftarrow S_2[R_2] \bowtie \bowtie S_3<a href="%5Ctheta">R_3</a>$</td>
<td>$S_2,R_2$ as $T_2$</td>
</tr>
<tr>
<td>9.</td>
<td>$S_1[R_1] \leftarrow S_2[R_2\bowtie S_3<a href="%5Ctheta">R_3</a>]$</td>
<td>$S_2,R_2$ as $T_2$</td>
</tr>
<tr>
<td></td>
<td>$S_1[R_1] \leftarrow S_2[R_2\bowtie S_3<a href="%5Ctheta">R_3</a>]$</td>
<td>outer join $S_3,R_3$ as $T_3$ on $\theta$</td>
</tr>
<tr>
<td>10.</td>
<td>$S_1[R_1] \bowtie A_1 \leftarrow S_2[R_2] \bowtie A_2$</td>
<td>select $T_2.A_2$ as $A_1$</td>
</tr>
<tr>
<td>11.</td>
<td>$S_1[R_1] \bowtie A_1 \leftarrow S_2[R_2] \bowtie \bowtie A_n$</td>
<td>select $T_3.A_3$ as $A_1$</td>
</tr>
<tr>
<td></td>
<td>$S_1[R_1] \bowtie A_1 \leftarrow S_2[R_2] \bowtie \bowtie A_n$</td>
<td>from $S_2,R_2$ as $T_2$</td>
</tr>
<tr>
<td></td>
<td>$S_1[R_1] \bowtie A_1 \leftarrow S_2[R_2] \bowtie \bowtie A_n$</td>
<td>inner join $S_3,R_3$ as $T_3$ on $T_2.FK_2 = T_3.K_3$</td>
</tr>
<tr>
<td>12.</td>
<td>$S_1[R_1] \bowtie A_1 \leftarrow \varphi(S_2[R_2] \bowtie A_2, \ldots)$</td>
<td>select $\varphi(S_2,R_2,A_2, \ldots)$ as $A_1$</td>
</tr>
<tr>
<td>13.</td>
<td>$S_1[R_1] \bowtie A_1 \leftarrow (S_2[R_2] \bowtie A_2, p_1); \ldots; v$</td>
<td>select case when $p_1$ then $S_2,R_2,A_2$ when $p_2$ then $\ldots$ else $\text{end } A_1$</td>
</tr>
<tr>
<td>14.</td>
<td>$S_1[R_1](A_1, \ldots, A_n) \leftarrow (A_1', \ldots, A_n')$</td>
<td>[where $\mid$ or] $T_2.A_1' = T_1.A_1$ and $T_2.A_2' = T_1.A_2$ and $\ldots$ and $T_2.A_n' = T_1.A_n$</td>
</tr>
</tbody>
</table>

The “on” clause (in line 4) is based on the join condition indicated in the end of $\psi_0$.

Table 2 shows the statements of SQL queries induced by, respectively, the SCAs and GCAs in $A$.

In Table 2, $\sigma$ is a selection over $R_2$. In item 3, the SQL query contains temporary relations $X_1$ and $X_2$. There will be one temporary relation to each GCA of case-based. $\varphi()$ can be a user-defined function or a SQL function, and $\gamma()$ is an aggregation function. We assume that $K$ and $FK$ are short expressions to indicate, respectively, a primary key and a foreign key.
\( Q_{\text{MOVIE}}: \)

01. `insert into M.Movie (title, year, genre, description)`
02. `select T2.title as title, T2.year as year, T3.category as genre, T3.summary as description`
03. `from S2.Film as T2 left join S1.Movie as T3 on T2.title = T3.film and T2.year = T3.year;`

**Fig. 6.** Query definition to load \( M\).MOVIE from \( S\).FILM and \( S\).MOVIE.

\( Q_{\text{CAST}}: \)

01. `insert into M.Cast (movie, actor, director)`
02. `select movie, actor, director`
03. `from (select T3.film as movie, T2.name as actor, T2.id from S1.Stars as T2 inner join S1.Movie as T3 on T2.id = T3.id)`
04. `where T2.role = 'actor') as X1`
05. `outer join`
07. `(select T3.film as movie, T2.name as director, T2.id from S1.Stars as T2 inner join S1.Movie as T3 on T2.id = T3.id)`
08. `where T2.role = 'director') as X2`
10. `on (X1.id = X2.id);`

**Fig. 7.** Query definition to load \( M\).CAST from \( S\).STARS.

Figure 7 presents the definition of the query to transform data from \( S\).STARS to \( M\).CAST. This query is more complex than the previous one. For this query, we have to define a nested select statement to each case-base GCA that relates attributes of \( S\).STARS to attributes of \( M\).CAST. Each nested select statement must be joined through a full outer join in order to guarantee both: i) that duplicate tuples will be merged properly, and 2) not duplicate tuples will be stored in \( M\).CAST. Thus, the clauses “from”, “outer join”, and “on” (respectively, lines 3, 6, and 10) correctly implement the data-metadata relationship specified by the SCA \( \psi_{10} \). The “on” clause (line 10) is based on the attribute indicated in \( \psi_{10} \). The first nested select statement (lines 3 to 5) is defined based on the GCAs \( \psi_{15} \) and \( \psi_{17} \). Lines 2 and 5 are based on \( \psi_{15} \) (see table ??, item 4). The second nested select statement (lines 7 to 9) is similar to the first one, but now it is based on \( \psi_{16} \) and \( \psi_{17} \). The “select” clause in line 1 is based on the GCAs \( \psi_{15}, \psi_{16} \) and \( \psi_{17} \).

In resume, the process to generate SQL queries based on CAs consists of the following steps:

1. To each RCA \( \psi \) that matches \( R_T \) and \( R_S \) do:
   - (a) Obtain the `INSERT INTO` statement (based on \( R_T \) definition).
   - (b) Identify the SQL query statement assigned to \( \psi \) in accordance to Table 1 (item 1 to 9).
   - (c) Identify all ACAs \( \psi' \) that matches \( R_T \) and \( R_S \).
Table 2. SQL Statements induced by SCAs.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ( \psi : S_1[R_1] \leftarrow groupby[S_2[R_2]([A_1, \ldots, A_n])] )</td>
<td>from ( S_2, R_2 ) as T2 group by ( A_1, \ldots, A_n )</td>
</tr>
<tr>
<td>2. ( \psi : S_1[R_1] \leftarrow groupby[S_2[R_2(\sigma)]([A_1, \ldots, A_n])] )</td>
<td>from ( S_2, R_2 ) as T2 where ( \sigma ) group by ( A_1, \ldots, A_n )</td>
</tr>
<tr>
<td>3. ( \psi : S_1[R_1] \leftarrow metadata(S_2[R_2]([A_1, \ldots, A_n])) )</td>
<td>from (select T2.A1, \ldots, T2.A_n from ( S_2, R_2 ) as T2) as X1 outer join (select T2.A1, \ldots, T2.A_n from ( S_2, R_2 ) as T2) as X2 on (X1.A1 = X2.A1, \ldots, X1.A_n = X2.A_n)</td>
</tr>
<tr>
<td>4. ( \psi : S_1[R_1] \bullet A_1 \leftarrow S_2[R_2] \bullet A_2 )</td>
<td>same as item 1. in Table 2</td>
</tr>
<tr>
<td>5. ( \psi : S_1[R_1] \bullet A_1 \leftarrow S_2[R_2] \bullet \varphi[A_n] )</td>
<td>same as item 2. in Table 2</td>
</tr>
<tr>
<td>6. ( \psi : S_1[R_1] \bullet A_1 \leftarrow \varphi(S_2[R_2] \bullet A_2, \ldots) )</td>
<td>same as line 3. in Table 2</td>
</tr>
<tr>
<td>7. ( \psi : S_1[R_1] \bullet A_1 \leftarrow (S_2[R_2] \bullet A_2, p_1); \ldots; p )</td>
<td>select T2.A2 as A1 from ( S_2, R_2 ) as T2 where p</td>
</tr>
<tr>
<td>8. ( \psi : S_1[R_1] \bullet A_1 \leftarrow \gamma(S_2[R_2] \bullet A_2) )</td>
<td>select ( \gamma(T2.A_2) ) as A1</td>
</tr>
<tr>
<td>9. ( \psi : S_1[R_1] \bullet A_1 \leftarrow \gamma(S_2[R_2] \bullet \varphi[A_n]) )</td>
<td>select ( \gamma(T3.A_n) ) as A1 from ( S_2, R_2 ) as T2 inner join S3.R3 as T3 on T2.FK2 = T3.K3</td>
</tr>
<tr>
<td>10. ( \psi : S_1[R_1] \bullet A_1 \leftarrow \gamma(\varphi(S_2[R_2] \bullet A_2, \ldots)) )</td>
<td>select ( \gamma(\varphi(T2.A_2)) ) as A1 from ( S_2, R_2 ) as T2</td>
</tr>
<tr>
<td>11. ( \psi : S_1[R_1] \bullet A_1 \leftarrow \gamma(S_2[R_2] \bullet A_2, p) )</td>
<td>select ( \gamma(\text{case when } p \text{ then } T2.A_2 \text{ else null end}) ) as A1</td>
</tr>
<tr>
<td>12. ( \psi : S_1[R_1] \bullet A_1 \leftarrow \gamma(S_2[R_2] \bullet \varphi[A_n], p) )</td>
<td>select ( \gamma(\text{case when } p \text{ then } T3.A_n \text{ else null end}) ) as A1 from ( S_2, R_2 ) as T2 inner join S3.R3 as T3 on T2.FK2 = T3.K3</td>
</tr>
<tr>
<td>13. ( \psi : S_1[R_1] \bullet A_1 \leftarrow \gamma(\varphi(S_2[R_2] \bullet A_2, \ldots), p) )</td>
<td>select ( \gamma(\text{case when } p \text{ then } \varphi(T2.A_2) \text{ else null end}) ) as A1 from ( S_2, R_2 ) as T2 inner join S3.R3 as T3 on T2.FK2 = T3.K3</td>
</tr>
</tbody>
</table>

(d) For each ACA \( \psi' \) do:
   i. identify the SQL query statement assigned to \( \psi' \) in accordance to Table 1 (item 10 to 14).

(e) Combine all statements of the SQL query in order to obtain the whole SQL query.

2. To each SCA \( \psi \) that matches \( R^T \) and \( R^S \) do:
   (a) Obtain the `INSET INTO` statement (based on \( R^T \) definition).
   (b) Identify the SQL query statement assigned to \( \psi \) in accordance to Table 2 (item 1 to 3).
   (c) Identify all GCAs \( \psi' \) that matches \( R^T \) and \( R^S \).
   (d) For each GCA \( \psi' \) do:
      i. identify the SQL query statement assigned to \( \psi' \) in accordance to Table 2 (item 4 to 13).
   (e) Combine all statements of the SQL query in order to obtain the whole SQL query.
5 Related Work

Schema matching is an important step of the data integration process [Fernando Filho, 2010]. Typically, 1:1 correspondences between two different schemas are manually defined using a GUI or are (semi-) automatically discovered using matchers (usually through heuristics). Each correspondence, in general, only specifies which elements refer to a same attribute or relation in the real world [Doan et al., 2012]. Cupid [Madhavan et al., 2001], AgreementMaker [Cruz et al., 2009], and OII Harmony [Seligman et al., 2010, Mork et al., 2008] are some examples of tools for schema matching. Cupid [Madhavan et al., 2001] is a generic schema matching, outside of any particular data model, and schema-based only (i.e., matchers consider only schema information, not instance data). AgreementMaker [Cruz et al., 2009] can match schemas and ontologies whose matchers are schema-based and instance-based. OII Harmony [Seligman et al., 2010, Mork et al., 2008] supports XML schemata, entity-relationship schemata from ER Win (a popular modeling tool), and relational schemata. It combines multiples matchers algorithms, each of which identifies correspondences using a different strategy, in order to generate correspondences with higher quality matching between attributes of compared schemas. Rahm in [Rahm, 2011] provides a brief comparison of some match tools, while authors in [Bonifati et al., 2011] present, in a systematic way, some metrics to evaluate matching and mapping tools.

Correspondences such as those defined/generated in [Cruz et al., 2009, Seligman et al., 2010, Mork et al., 2008, Madhavan et al., 2001] do not provide the necessary information for discovering expressions to transform data sources in data target (i.e., the mapping expressions), the next phase in the schema mapping process. Richer models for specifying correspondences between schemas were proposed by [Doan, 2002, Massmann et al., 2011, Gal, 2011, Bohannon et al., 2006, Vidal and Lóscio, 1999, Dhamankar et al., 2004, Magnani et al., 2005, Giunchiglia et al., 2005]. These approaches allow to define one-to-one or many-to-one attribute correspondences (i.e., association between attributes of two schemas). [Doan, 2002, Dhamankar et al., 2004] allow to semi-automatically match one attribute to various attributes (e.g., totalPrice matches to unitPrice*quantity), incorporating machine learning, statistics, and heuristics to evaluate candidate matches. COMA and COMA++ [Massmann et al., 2011] are generic prototypes for schema and ontology matching, schema-based and instance-based, and support a semi-automatic or manual enrichment of simple 1:1 correspondences into more complex mapping expressions including functions to support data transformations. [Dhamankar et al., 2004] describes the IMAP system, which semi-automatically discover basic and complex matches. Each match in [Dhamankar et al., 2004] (there named search) discovers specific types of complex matches, using different kind of information such as domain knowledge, and domain integrity constraints to improve matching accuracy. [Gal, 2011, Bohannon et al., 2006] allow to express conditional correspondences (i.e., the value of an attribute A is the same of an attribute B if a given condition is satisfied). [Giunchiglia et al., 2005] and [Magnani et al., 2005] set correspondences with semantic relationships, such as equivalence, containment, subsumption, disjointness, and unknown (a special relationship retuned by the matching algorithm when none of the others relationships hold). More closely to our approach is the work in [Vidal and Lóscio, 1999] and [Vidal et al., 2013]. In [Vidal and Lóscio, 1999], the authors allow to manually specify one-to-one correspondence assertions between elements of Entity Relationship models. Although they cannot specify
many-to-many matches, they correspondences have some semantic and allow to specify relationships such as: equivalence, union, intersection, and filtering (there named selection). In [Vidal et al., 2013], the authors allow to specify correspondences between relational schemas and RDF schemas, but they are basically 1-to-1 correspondences referring to project-selection-equi-join queries. The unique 1:m correspondence that they deal with relates one attribute to various attributes through concatenation.

Authors in [Pequeno and Pires, 2009, Pequeno, 2011] can specify one-to-one and many-to-many basic, complex, and semantic matches between elements of object-relational schemas. They can specify most part of the correspondences specified in [Vidal and Lóscio, 1999] and other more complex. For example, they can deal with aggregate functions, denormalisations, and grouping (i.e., group by in SQL). Joins and outer-joins are implicitly defined based on the integrity constraints or match functions. A distinguished feature of [Pequeno and Pires, 2009, Pequeno, 2011] is that it allows to match, in a same correspondence, relations and attributes of two or more schemas. Yet, the information they provide is not sufficient, since they do not explicitly allow to specify join paths and its variants, nor deal with data-metadata relationships.

Data-metadata translations between elements of different relational schemas have been studied extensively, being SchemaSQL [Lakshmanan et al., 1996] and FIRA/FISQL [Wyss and Robertson, 2005] maybe the most notable works in this subject. SchemaSQL [Lakshmanan et al., 1996] is a SQL-like metadata query language that uses view statement to restructure one column of values of a relation into metadata in another one. FISQL [Wyss and Robertson, 2005] is a successor of SchemaSQL. It is a query language that expresses data-metadata transformations using metavariables that range over relation names and column names. In addition, FISQL is equivalent to the query algebra FIRA. FIRA, in addition to the usual relational operators, defines simple operators for data-metadata querying such as: “↑” to promote metadata, “↓” to demote metadata, and “→” to dereference data. Furthermore, FIRA/FISQL are capable of producing fully dynamic output schemas, where the exact name of the relation schema or the attribute name is only known in run-time. Both SchemaSQL and FIRA/FISQL are proposed to provide interoperability in relational multi-database systems. Our new SCAs of metadata were based on the promote operator of the FIRA.

In other proposals, such as those in [Haas et al., 2005, Bonifati et al., 2008], correspondences are basic matching (although sometimes they allow to specify selection conditions and combination of attributes) and the aim is automatically generate mapping expressions from the correspondences. Both Clio in [Haas et al., 2005] and Spicy in [Bonifati et al., 2008] define mapping expressions as logical formulas (tuple-generated dependences - tgds, and equality-generated dependences - egds). A drawback of approaches that create tgds automatically, as in [Haas et al., 2005] and [Bonifati et al., 2008], is that tgds do not deal with, for example, groupings, aggregations, many-to-many attribute correspondences, and data-metadata transformations. In addition, the quality of the mapping generated depends on the quality of the correspondences.

Tgds are only used to relational instances. Clip [Raffio et al., 2008] (an extension of Clio) and ++Spicy [Marnette et al., 2011] (an extension of spicy) proposed extensions to tgds, named nested tgds, in order to be used in instances with hierarchical nesting (i.e., XML instances). Their

---

7 Match functions are functions that determine if two different instances represent the same concept in the real world.
proposal allow generate mappings between relational and XML schemas and allow to express grouping, aggregation and many-to-many attribute correspondences. Yet, they do not deal with data-metadata relationships, nor specify the condition that can be used for combining relations as well as join variants (e.g., right outer join and left outer join). Mad mapping [Papotti and Torlone, 2009] is an extension of Clio that deal with data-metadata relationships. Its approach is similar to the SchemaSQL approach [Lakshmanan et al., 1996], and, in some aspects to the FISQL. Due to the correspondences used as input do not provide the necessary information for deriving the mappings that encodes the semantic desired by the user, the solution proposed in [Raffio et al., 2008], [Marnette et al., 2011] in [Papotti and Torlone, 2009] can generate mapping expressions which populate a target relation with duplicate information. In our proposal, we specify correspondences that provide information that is sufficient for determining and deriving the desired schema mappings. In addition, different of the most approaches of schema matching and schema mapping, CAs allow to specify, in a same correspondences, the relationships between several source schemas and one target schema. This contribute to reduce the number of duplicate information in the target schema.

6 Conclusion

This paper focused on present Correspondence Assertions (CAs) that deal with 1:1 and m:n matchings between schemas components, including correspondences involving aggregations, joins, and metadata. Using CAs, we shown how SQL queries can be automatically generated in order to load relations (or views) of a mediated schema.

We emphasize that, in our approach, the CAs can specify basic and complex correspondences with semantics. The semantically rich and formal representation of our Correspondence Assertions (CAs) allows generate mapping expressions that are easy to reuse and maintain when some change occurs in a schema definition.

We are currently working on the development of a mechanism to discover new CAs from previous ones using inferences and heuristics. This mechanism intend to help the designer in the definition of new CAs, given insights and suggestions in an interactive way. We also are working on the development of a tool to specify the CAs, and later, infer new CAs.

ACKNOWLEDGEMENTS

This work has been supported by the scholarship (SFRH/BPD/76024/2011) granted by FCT.

References


