On semi-automated matching and integration of database schemas

Ünal Karakaş, Ö.

Publication date
2010

Chapter 3

Heterogeneity

Heterogeneity corresponds to differences in a wide range of areas related to information systems, and is considered as the most challenging obstacle standing in the way of achieving seamless interoperability among independent information systems. This chapter first provides some introductory information about three dimensions, distribution, autonomy, and heterogeneity, under which information systems are categorized in relation to accessing information. Then Section 3.2 presents a number of taxonomies for heterogeneities proposed in the literature. Section 3.3 focuses on the main approaches for dealing with the schema heterogeneity, since this constitutes one of the main subjects of this thesis. Also provided under Section 3.3, different types of schema heterogeneities are exemplified. Finally, Section 3.4 concludes this chapter and emphasizes the importance of tackling schema heterogeneity problems with a semi-automated approach.

The research results presented in this chapter were partially published in the Journal of Knowledge and Information Systems (Unal & Afsarmanesh, 2010) and in Lecture Notes in Computer Science (Unal & Afsarmanesh, 2006b).

3.1 Related Concepts

From the viewpoint of accessing the information, the existing information systems are categorized under three dimensions, including: distribution, autonomy, and heterogeneity (Sheth & Larson, 1990), as further detailed below:

- *Distribution Dimension*: Data is typically distributed over different, usually geographically dispersed data sources. With the advances of the Web, these sources are now interlinked, hiding their physical locations, and thus making this dimension less challenging with regards to achieving database interoperability. However, exchanging large volumes of data over distributed networks have been another challenging aspect, which can now be easily dealt with through broad bandwidths.
- *Autonomy Dimension:* Each organization runs some information systems independently from others. For example, organizations may autonomously decide to share a part of their local resources or services with others. Furthermore, they may maintain autonomy on their local data and define/use their own data models. We identify four main types of autonomy that can be exercised in federated database systems: 1) *Design autonomy* that refers to a component's being independent from others in their information system design, including data model, query language, constraints, etc. 2) *Communication autonomy* that refers to a component's autonomy in deciding whether or not to communicate with others and when and how to communicate. 3) *Association autonomy* that refers to a component's autonomy in deciding which parts of its resources and functionalities to share with others.

4) *Execution autonomy* that enables a component to execute local operations and to decide on the order of these operations without interference of external systems.

- *Heterogeneity Dimension:* Heterogeneity arises due to autonomy of organizations. It corresponds to differences in numerous areas of information systems. Heterogeneity has been the most challenging dimension within the context of database interoperability, especially considering the large variety of conflicts that may exist among distinct data providers. Different types of classifications are proposed in the literature for this dimension, as addressed in the next section.

3.2 Taxonomy of Heterogeneity Resulted Conflicts

Different, but partially overlapping classifications for heterogeneity have been proposed in the literature. Some classifications only distinguish between the information and schema when it comes to heterogeneity, while some others consider several other types of heterogeneity in information systems. In this section, we present five different classifications defined in the literature, in relation to our research work:

Classification-1: As for conflicts that may exist among schemas, Batini et al. (Batini et al., 1986) defines two categories, as shown below in Figure 3.1, including: *name conflicts* and *structure conflicts*.

1- ***Name conflicts*** arise because of the fact that different database designers typically use different terminology for the same domain. Typically, there are two types of relationships among the names used that cause name conflicts:

- **Homonyms**: The same name is used for different concepts.
- **Synonyms**: The same concept is described by different names.

2- ***Structural conflicts*** arise because of using either different modeling constructs or different integrity constraints. Structure conflicts are classified by (Batini et al., 1986) as follows:

- **Type Conflicts** occur as a result of using different modeling constructs (e.g. using entity versus attribute) for representing the same concept.
- **Dependency Conflicts** arise when different schemas introduce different relationships among the same concepts, such as a 1-to-1 relationship is indicated between two concepts in one schema, while in another schema the concepts have a 1-to-m relationship.

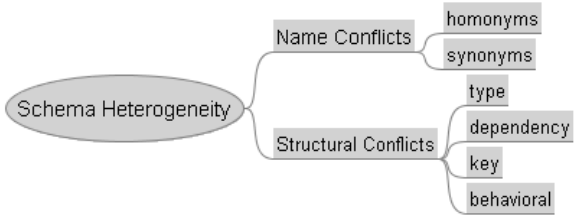

Fig. 3.1. Taxonomy of Schema Conflicts (Batini et al., 1986)

- **Key Conflicts** arise when different keys are introduced for the same concept in different schemas. For example, *employee* may have the *SSN* attribute as the key in one schema, whereas in another schema its key may be the *ID* attribute.
- **Behavioral Conflicts** arise due to different insertion / deletion policies introduced in different schemas for the same concept. For example, in one schema *player* data may exist without a related *team* data, but in another schema when a *player* is inserted it has to have a related *team* data.

Classification-2: In another study, (Sheth & Kashyap, 1992) defines a classification with the emphasis on the schematic heterogeneities and semantic similarities, as shown in Figure 3.2.

1- ***Domain Definition Incompatibility*** corresponds to differences in attribute domain definitions for representing semantically similar attributes and consists of types of conflicts listed below:

- **Naming Conflicts**: Using synonyms and homonyms for defining attributes.
- **Data Representation Conflicts**: Using different data types for semantically similar attributes. For example, an attribute may be defined of type *string* in one schema, whereas in another schema a similar attribute may be defined of type *integer*.
- **Data Scaling Conflicts**: Using different units or measures. For example, *price* attribute might have values in *dollar* in one database and in *euro* in another database.
- **Data Precision Conflicts**: Using different precisions. For example, the *grade* attribute may have a value between 1-100 in one schema, but it may have a letter value (A, B, C, etc.) in another schema (Sheth & Kashyap, 1992).
- **Default Value Conflicts**: Using different default values. For example, default value for the threshold attribute might be 0.5 in one schema and 0.6 in another schema.
- **Attribute Integrity Constraint Conflicts**: Using different integrity constraints that might not be consistent with each other. For example, an attribute *X* may have the constraint that *X>30* in one schema and *X<20* in another schema.

2- ***Data Value Incompatibility*** corresponds to using different values for data in different databases. This type of incompatibility depends on the database state and can arise as a result of the following inconsistencies:

- **Known Inconsistency**: Related to inconsistencies, cause of which are known. For example, it might be known that one database is more reliable than the other. In this case, the more reliable database can be used to resolve the inconsistency.
- **Temporal Inconsistency**: Related to inconsistencies, which are temporal. In other words, information stored in a database is time dependent. In this case, inconsistency is temporary.
- **Acceptable Inconsistency**: Related to inconsistencies that are within an acceptable range and considered tolerable. Therefore, for some types of queries, the errors in the values of inconsistent databases may be tolerable.

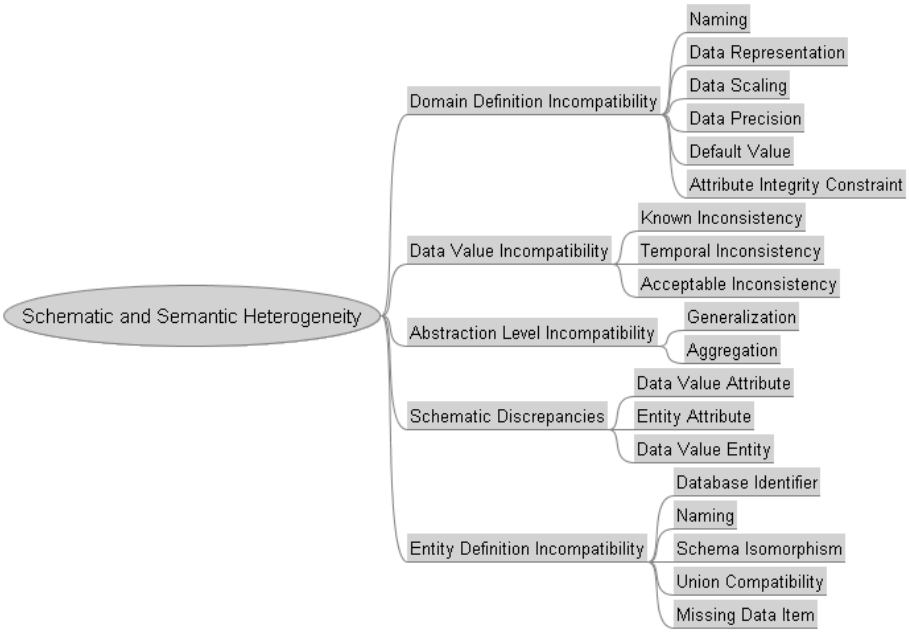

Fig. 3.2. Taxonomy of Schema Conflicts (Sheth & Kashyap, 1992)

3- ***Abstraction Level Incompatibility*** corresponds to differences in levels of abstraction that different databases use to represent semantically similar entities and can be of two types:

- **Generalization Conflicts**: Using different levels of generalization. For example, *car* entity may be represented by the *vehicle* entity in one schema and *car* entity in another schema.
- **Aggregation Conflicts**: Using aggregation in one database to identify a set of entities in another database. For example, *team* in one schema is a set of *player*s in another schema.

4- ***Schematic Discrepancies*** arise when data in one database corresponds to metadata in another database and consist of three types of conflicts:

- **Data Value Attribute Conflict**: Arises when the value of an attribute in one database corresponds to an attribute in another database. For example, letter grades (A, B, C, D, F) of a student may be stored in a *grade* attribute in schema S1, whereas in *A*, *B*, *C*, *D*, and *F* attributes in schema S2.
- **Entity Attribute Conflict**: Arises when an entity is modeled as an attribute in one database, whereas as a relation in another database. Considering the example in the Data Value Attribute Conflict, suppose that another schema, S3, has separate relations for each grade, in the form of A(date, name_of_student,..), B(date, name_of_student,..), etc. In this case, there is an entity-attribute conflict between the schemas S3 and S1.

- **Data Value Entity Conflict**: Arises when the value of an entity in one database corresponds to a relation in another database. Considering the examples in data value attribute and entity attribute conflicts, there is a data value entity conflict between the schemas S2 and S3.

5- ***Entity Definition Incompatibility*** arises when incompatible entity descriptors are used and involves the following types:

- **Database Identifier Conflicts**: Using semantically different identifier records. For example, *employee* entity may have the attribute *ssn* as the key in one schema and *name* as the key in another schema.
- **Naming Conflicts**: Using synonyms and homonyms for defining entities. Name conflicts here exist among entities, whereas the name conflicts under "Domain Definition Incompatibility" exist among attributes.
- **Schema Isomorphism Conflicts**: Using different number of attributes for semantically similar entities.
- **Union Compatibility Conflicts**: Having semantically unrelated set of attributes for semantically similar entities. For example, *employee* entity having *ssn, name,* and *address* attributes in one schema and *ssn, name,* and *salary* attributes in another schema are union incompatible.
- **Missing Data Item Conflicts**: Arise when one of the semantically similar elements has a missing attribute. For example, *vehicle* entity may have *type* attribute (for the types of vehicle, such as car, bus, etc.) in one schema, but in another schema, *car* entity may not have this attribute.

Classification-3: Another classification, shown in Figure 3.3, is proposed by Sheth for defining different types of heterogeneity in information systems (Sheth, 1998):

1- ***Information Heterogeneity:*** Corresponds to differences in information that involves *semantic*, *structural* and *schematic*, and *syntactic* heterogeneities.

2- ***System Heterogeneity:*** Corresponds to differences in information systems, namely in *digital media repository management systems* and *database management systems*

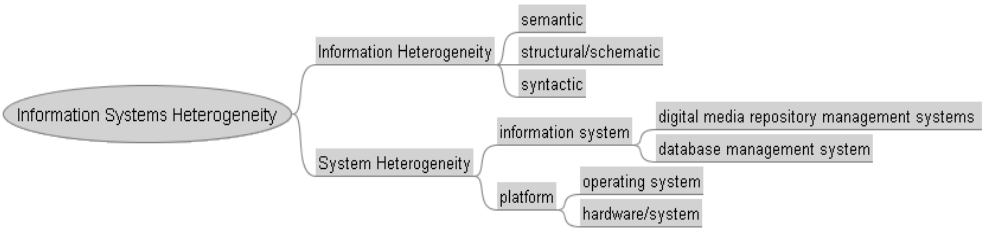

Fig. 3.3. Taxonomy of Information Systems Heterogeneity (Sheth, 1998)

(DBMS), as well as disparities in *platforms*, namely *operating systems* and *hardware/systems*.

Classification-4: (Busse et al., 1999) uses another classification, shown in Figure 3.4, which divides the heterogeneity into three: syntactical, data model and logical heterogeneity, as explained below.

1. ***Syntactical Heterogeneity*** is divided into two main types; technical and interface heterogeneity:
    - **Technical Heterogeneity**: Differences in technical aspects such as operating systems, protocols, etc.
    - **Interface Heterogeneity**: Heterogeneity that arises as a result of using different access languages. Differences in the following subjects cause interface heterogeneity:
        - *Language Heterogeneity*: Use of different query languages or language restrictions.
        - *Query Restrictions*: Allowing only certain types of queries, such as the maximum number of joins allowed.
        - *Binding Restrictions*: Allowing only certain attribute values in queries.
2. ***Data Model Heterogeneity***: Related to different data models' having different semantics for their concepts.
3. ***Logical Heterogeneity***: Classified in three sub-categories:
    - **Semantic Heterogeneity**: Differences in semantics of data and schema. Different semantic schema conflicts can occur: equal names can denote different concepts (homonyms), different names can be used for the same concept (synonyms), and so on.
    - **Schematic Heterogeneity**: Differences in encoding of concepts at different elements of a data model. Attribute name-relationship and attribute name-attribute value conflicts in relational databases are examples of this kind of heterogeneity.
    - **Structural Heterogeneity**: Occurs if elements are structured in different ways in

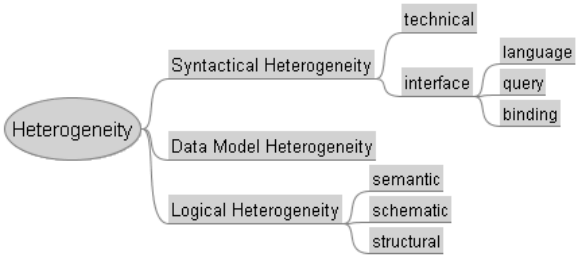

Fig. 3.4. Taxonomy of Information Heterogeneity (Busse et al., 1999)

different schemas. For example, grouping attributes into different tables in two schemas results in this type of heterogeneity.

Classification-5: Another classification of heterogeneity is proposed by (Kahng & Mcleod, 2001), who divides information heterogeneity into two, as depicted in Figure 3.5:

1. ***Data Model Heterogeneity:*** Differences in collections of structures, constraints, and operations that information systems use to describe and manipulate data.
2. ***Semantic Heterogeneity:*** Differences in specifications of data. Semantic conflicts among information systems that use object-based data model are listed by (Kahng & Mcleod, 2001) as follows:
    - **Category***:* Two objects coming from different information sources may have equivalence, sub-concept/super-concept, or partially overlapping relationships when they represent same or similar real world entities.
    - **Structure***:* Two objects coming from compatible categories may have different structures. For example one object in one system may have one attribute but another object in another system might not have it.
    - **Unit**: Two objects coming from the compatible categories and having the same structures may use different units.
    - **Terminology**: Use of synonyms and homonyms may cause semantic heterogeneities.
    - **Universe of Discourse**: Semantics hidden in the context may result in semantic heterogeneities.

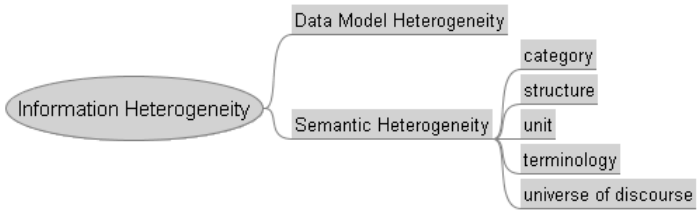

Fig. 3.5. Taxonomy of Information Heterogeneity (Kahng & Mcleod, 2001)

3.3 Challenges for Schema Matching

As it is clear from the existence of a large number of classifications presented in Section 3.2 above, heterogeneity has been one fundamental problem against enabling interoperability among information systems. Despite the existence of different classifications of heterogeneity, there are many commonalities in the terminology that these classifications employ. However, categories and the terms used in each classification are confusing. For example, Classification-

2 places “Naming Conflicts” under both “Domain Definition Incompatibility” and “Entity Definition Incompatibility”. The first one refers to the attributes, whereas the second one refers to the entities. Furthermore, for the “Schematic Discrepancies” category, it is difficult to infer by just looking at the name to which the category is related for its data-metadata conflicts. As another example, Classification-4 has a category called “Schematic Heterogeneity”, but another category called “Semantic Heterogeneity” is also defined. But there is some overlap between these two categories. Furthermore, some classifications miss to consider heterogeneities corresponding to the different forms of names used in the schemas, such as using abbreviated vs. extended names.

Since the focus of this thesis is on schema matching and schema integration, we concentrate on analyzing schema heterogeneities. Hence, the heterogeneities related to instance data and underlying systems, such as those of the database management system are not considered in our classification. Furthermore, we aim to define a classification that is clear and simple (as opposed to those other classifications that are confusing), but at the same time broad enough to cover a wide variety of different types of database schema conflicts. In this respect, the research explained in this thesis covers both the structural and linguistic heterogeneities, to capture the semantic, syntactic, and structural schema conflicts, as indicated in Figure 3.6.

A number of examples are provided below, each of which representing a different kind of schema conflict that belongs to one of the categories: structural, linguistic, or belong to a combination of the two. These examples are taken from two specific example schemas S1 and S2, defined for a university domain, encompassing courses, students, etc., as shown in Figure 3.7. As described in these examples, structural conflicts are more difficult to resolve than linguistic conflicts. Clearly, the varieties of types of conflicts that exist among databases are not limited to these given here. However, the conflicts shown in the examples are the ones which frequently occur in database schemas, and cause bottlenecks for automatic schema matching and integration. Note that for simplicity reasons, in the example cases only partial schemas are shown in a simple format. If relevant to the example, some primary and foreign keys are also shown in the schemas.

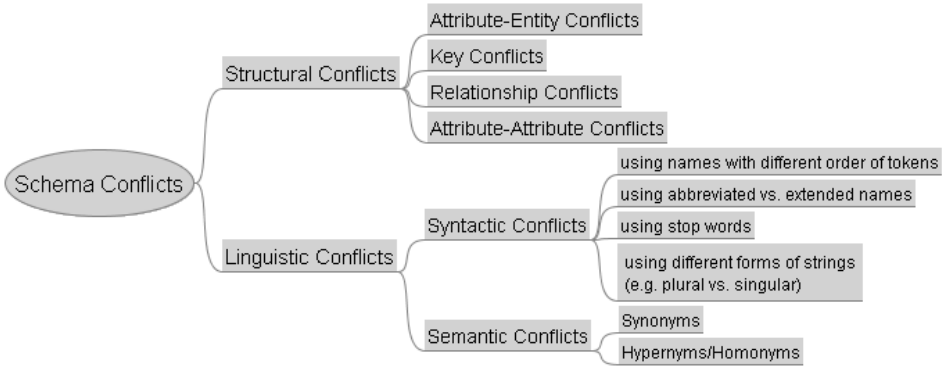

Fig. 3.6. Classification of Schema Conflicts Considered in the Thesis

S1
Person (ssn (PK), fname, lname, rank, salary, office, phone, child, depid (FK))
Student (stid (PK), fname, lname, birth_date, sex, address, class, gpa, depid (FK))
GradStudent (stid (PK, FK))
Department (dptID (PK), name_of_department, dphone, doffice, college)
Course (cno, cname, description, section, day, ssn (FK), depid (FK))
Time (timeid, start, end, cno(FK))
Campus (cid (FK), location)

S2
Instructor (id (PK), name, telephone, children, dcode (FK))
Student (studentID (PK), degree, date_birth, name, gender, city, state, zip,
GradePointAverage, dptcode (FK))
GradeReport (studentID (PK, FK), section_number (FK), letter_grade, numeric_grade)
Course (cnumber, coursename, coursedesc, credits, dcode (FK))
Section (section_number (PK), semester, year, instID (FK), day, cnumber (FK))
Time (timeid, start, end, section_number (FK))
Department (dptCode (PK), nameDepartment, officeNumber, officePhone, college)
Campus (cid (PK), city, zipcode)
Apply (studentID (PK, FK), cid (PK, FK))

Fig. 3.7. Example Schemas from University Domain

1) ***Examples of Structural Conflicts:*** Structural conflicts exist due to the fact that different organizations use different constructs and integrity constraints to represent the same concepts.

- **Attribute-Entity Conflict***:* This conflict arises when a concept is represented as an attribute in one schema and as a separate entity in the second schema. For example, as shown in Figure 3.8, "section" information is represented by the 'section' column of the "Course" table in schema S1, while schema S2 represents it as a separate table "Section".

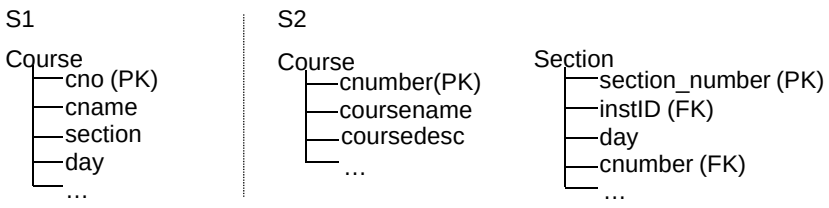

Fig. 3.8. Attribute-Entity conflict

- **Key Conflict:** This conflict arises when different keys are assigned for the same entity in different schemas. In the example shown in Figure 3.9, the key of the "Department" table in schema S1 is "dptID", while that of the "Department" table in schema S2 is "dptCode"

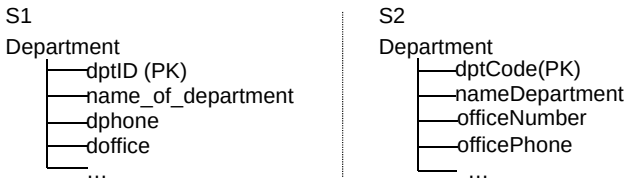

Fig. 3.9. Key conflict

- **Relationship Conflict:** This conflict is related to the case, in which relationships between entities in different schemas are defined differently, as exemplified in Figure 3.10. In S1, there is a one-to-many relationship between the "Student" and "Campus" tables, while in S2, this relationship is many-to-many, thus introducing a third table "Apply", to represent this relationship.

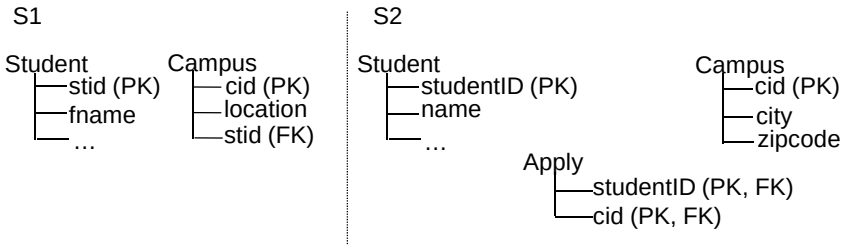

Fig. 3.10. Relationship conflict

- **Attribute-Attribute Conflict**: This conflict arises when the same concept is represented by using different number of attributes in different schemas. For the example shown in Figure 3.11, "address" data is stored in one column of the "Student" table in S1, while in S2, it is spread over three columns.

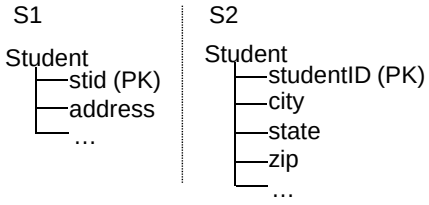

Fig. 3.11. Attribute-Attribute conflict

2) ***Examples of Linguistic Conflicts*****:** Linguistic conflicts arise because of different terminology and names that different organizations use to refer to the same data. Linguistic conflicts can be of two types: syntactic and semantic. Below are the examples for the syntactic

and semantic conflicts. Since these types of schema conflicts are due to differences in representation of "names" of columns or tables, we have not provided example schemas but only some related but different names in the following examples.

- **Syntactic Conflicts: These conflicts arise** due to using different forms of the names and can be of the following types:
    - Using names that consist of more than 1 token (word) with different order of tokens, e.g. *birth_date* vs. *date_birth*.
    - Using abbreviated vs. extended names, e.g. *GPA* vs. *GradePointAverage.*
    - Existence of stop words, e.g. *name_of_department vs. nameDepartment*
    - Using different forms of a string (plural, singular, etc.), e.g. *child* vs. *children.*
    - Similar to the use of abbreviated vs. extended names, use of short forms of strings, e.g. *phone* vs. *telephone.*
- **Semantic Conflicts: These conflicts arise** due to differences in semantics of names and can be of the following types:
    - Use of synonyms, e.g. *gender* vs. *sex.*
    - Use of hypernymy / hyponymy (representing IS-A), e.g. *person* vs. *instructor*.

3) ***Example of Combined Structural and Linguistic Conflicts*****:** The two types of conflicts addressed above may occur in a combined form in some parts of the schema. Following figure (Figure 3.12) shows an example for this case, where the "location" information in S1 can be matched with the concatenation of "city" and "zipcode" columns of the "Campus" table in S2, if we apply both the linguistic "IS-A hierarchy" and the structural "one attribute in the first schema matches two attributes in the second" (attribute-attribute conflict) conflict resolution techniques.

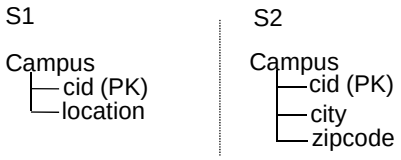

Fig. 3.12. Combined Structural and Linguistic conflicts case

3.4 Conclusion

The term heterogeneity, also referred to as conflict in several related reported research, is used in different contexts, such as those addressing schema heterogeneity, information heterogeneity, etc. It is generally used to refer to the diversity or difference, which exists among distinct elements within a domain. Heterogeneity is a big obstacle to interoperability. In information systems, with the increasing number of efforts during the last years, which aim

to enable interoperability, heterogeneity has been addressed as an important issue common among them.

Different classifications of heterogeneities in information systems have been proposed in the related research literature. Considering the subject of this thesis, schema heterogeneity is the one, on which we have focused. We divide schema heterogeneity into two main classes of: *structural* and *linguistic*. *Structural conflicts* exist due to the fact that different organizations use different constructs and different integrity constraints to represent the same or similar concepts. The most frequently occurring structural conflicts include: *attribute-entity conflicts*, *key conflicts*, *relationship conflicts*, and *attribute-attribute conflicts*. *Linguistic conflicts* on the other hand arise because of the different terminology and names that different organizations use to refer to the same or similar concept. Linguistic conflicts can be of two types: *syntactic* and *semantic*. Structural conflicts are more difficult to identify and resolve than linguistic conflicts.

In any approach that attempts to enable data sharing among heterogeneous databases, both linguistic and structural conflicts need to be considered. Considering that the complete semantics of the concepts introduced in an environment are not and/or cannot be fully expressed in database schemas, human intervention is deemed to be necessary, as the decision maker to resolve ambiguities in this area. Therefore, fully-automated conflict resolution does not seem realistic for the time being, and hence we hypothesize that semi-automatic approaches need to be devised to more effectively deal with the problem of heterogeneity.