

A Simple Case Study for Schema Integration Tools

AutoMed Technical Report 1, Version 1

Peter McBrien

Wednesday 14th February 2007

1 Introduction

The purpose of this document is to record a simple small example of database schema integration, which contains enough complexity to test all key aspects of the AutoMed achitecture [?] as it is developed. In particular, the example is used to test the representation of modelling languages in the MDR and schemas and transformations in the STR of the AutoMed repositories [?, ?]. The example is a development of that presented in [?], and is presented in Figure ?? as three extensional databases (*i.e.* are actual databases with stored data) **s1**, **s2** and **s3** in a variant of the ER modelling language.

The version of the ER modelling language we shall be using is given in Table ?. It a slight extension of the ER language used in the well-known survey paper on schema integration [?], and similar to the well known **enhanced-ER (EER)** model [?]. The key features of this ER modelling language are that

- only binary relationships are permitted,
- relationships may not have an attribute added,
- attributes may be key, notnull or null, and
- generalisations may be total or partial.

The following sections present a number of integration scenarios, which illustrate common techniques used in the BAV approach.

2 Integration of Two Instance Equivalent Schemas

Suppose that it can be determined that the **staff** entity in **s1** always contains the same set of people as the **person** entity in **s2**, and that the **id** key of **staff** is the same type as the **pid** attribute of **person**. It follows the we have a synonym conflict between the schemas, and should rename **staff** to **person** (or *visa versa*) and rename **pid** to **id** (or *visa versa*).

Also suppose that it can be determined that the instances of the **dept** entity (and hence its **dname** key attribute) in **s1** are always the same set as the values of the **dname** attribute of **person** in **s2**. It follows that we have a structural conflict between the schemas, and should transform the **dname** attribute in **s2** into a new entity **dept** and key attribute **dname** connected to **person** by a relationship called **worksin**. In this particular example, we could alternatively have transformed the **worksin** relationship, **dept** entity and **dname** key attribute into a single **dname** attribute of **staff**,

but this would not be the case if the **dept** entity had any other associations (*i.e.* it had any other attributes, relationships, subset involvements or generalisation involvements).

Finally, suppose that it can be determined that the instances of the **sex** attribute of **staff** in **s1** only takes the string values 'm' and 'f', and that always the set of **staff** which have the value 'm' is the same as the set of **male** in **s2**, and the set of **staff** which have the value 'f' is the same as the set of **female**. It follows that we have a structural conflict between the schemas, and should transform the **sex** attribute of **staff** into a total generalisation hierarchy with two subset entities **male** and **female**. In this particular case, we could alternatively have made the inverse transformation on **male** and **female** in **s2** since they have no other associations.

Given these suppositions, we can use the following transformations to transformation **s1** and **s2** into a new global schema **s4**. The queries of the transformations use the AutoMed IQL language [?]. The semantics of $\langle\langle\text{male}\rangle\rangle$ are that the extent of $\langle\langle\text{male}\rangle\rangle$ is a list of unary tuples, taken as the projection of the first attribute of the tuples of $\langle\langle\text{person}, \text{sex}\rangle\rangle$ where the second attribute equates to the value 'm'. The semantics of $\langle\langle\text{female}\rangle\rangle$ are that the binary tuples for the $\langle\langle\text{person}, \text{sex}\rangle\rangle$ attribute can be covered by associating the first attribute of the tuple an instance of $\langle\langle\text{male}\rangle\rangle$ and the second to the string constant 'm', concatenated (by the ++ operator) with associating the first attribute to an instance of $\langle\langle\text{female}\rangle\rangle$ and the second to 'f'. Note that $\langle\langle\text{female}\rangle\rangle$ has also a constraint added to it, stating that this transformation is only valid when the extent of $\langle\langle\text{person}, \text{sex}\rangle\rangle$ is just ['m', 'f'].

s1 $\rightarrow$ **s4**

- ① renameEntity($\langle\langle\text{staff}\rangle\rangle$, $\langle\langle\text{person}\rangle\rangle$)
- ② addEntity($\langle\langle\text{male}\rangle\rangle$, $\{\{x\} \mid \{x, y\} \leftarrow \langle\langle\text{person}, \text{sex}\rangle\rangle; (=) y \text{ 'm'}\}$)
- ③ addEntity($\langle\langle\text{female}\rangle\rangle$, $\{\{x\} \mid \{x, y\} \leftarrow \langle\langle\text{person}, \text{sex}\rangle\rangle; (=) y \text{ 'f'}\}$)
- ④ addGeneralisation(**sex**, **total**, **person**, **male**, **female**)
- ⑤ delAttribute($\langle\langle\text{person}, \text{sex}\rangle\rangle$,
 $\{\{x, y\} \mid \{x\} \leftarrow \langle\langle\text{male}\rangle\rangle; (=) y \text{ 'm'}\} ++ \{\{x, y\} \mid \{x\} \leftarrow \langle\langle\text{female}\rangle\rangle; (=) y \text{ 'f'}\}$
 $([\text{'m'}, \text{'f'}] = \{\{x\} \mid \{x, y\} \leftarrow \langle\langle\text{person}, \text{sex}\rangle\rangle\})$)

s2 $\rightarrow$ **s4**

- ⑥ addEntity($\langle\langle\text{dept}\rangle\rangle$, $\{\{y\} \mid \{x, y\} \leftarrow \langle\langle\text{person}, \text{dname}\rangle\rangle\}$)
- ⑦ addAttribute($\langle\langle\text{dept}, \text{dname}, \text{key}\rangle\rangle$, $\{\{y, y\} \mid \{x, y\} \leftarrow \langle\langle\text{person}, \text{dname}\rangle\rangle\}$)
- ⑧ addRelationship($\langle\langle\text{worksin}, \text{person}, \text{dept}, 1:1, 1:N\rangle\rangle$, $\{\{x, y\} \mid \{x, y\} \leftarrow \langle\langle\text{person}, \text{dname}\rangle\rangle\}$)
- ⑨ delAttribute($\langle\langle\text{person}, \text{dname}, \text{nonnull}\rangle\rangle$, $\{\{x, y\} \mid \{x, y\} \leftarrow \langle\langle\text{worksin}, \text{person}, \text{dept}\rangle\rangle\}$)
- ⑩ renameAttribute($\langle\langle\text{person}, \text{pid}, \text{key}\rangle\rangle$, $\langle\langle\text{person}, \text{id}, \text{key}\rangle\rangle$)

3 Integration of Overlapping Schemas

Suppose it can be determined that all schema components that appear in both **s4** and **s3** always share the same sets of values, *i.e.* the entity **person** in **s4** has the same set of values as **person** in **s3**, the attribute **id** in **s4** has the same set of values as **id** in **s3**, *etc.*

Also suppose that it can be determined that any construct that does not appear in one schema cannot be derived from the remaining components in that schema, *i.e.* the extension of the missing **name** attribute of **person** in **s3** cannot be derived from any other components in **s3**, and the extension of the missing **degree** entity and associated constructs in **s4** cannot be derived from any other constructs in **s4**.

Given these suppositions, we can use extend transformations to build a new global schema **s5** that contains all information available from the three extensional schemas **s1**, **s2** and **s3**.

s3 $\rightarrow$ **s5**

- ⑪ extendAttribute($\langle\langle\text{person}, \text{name}, \text{nonnull}\rangle\rangle$)

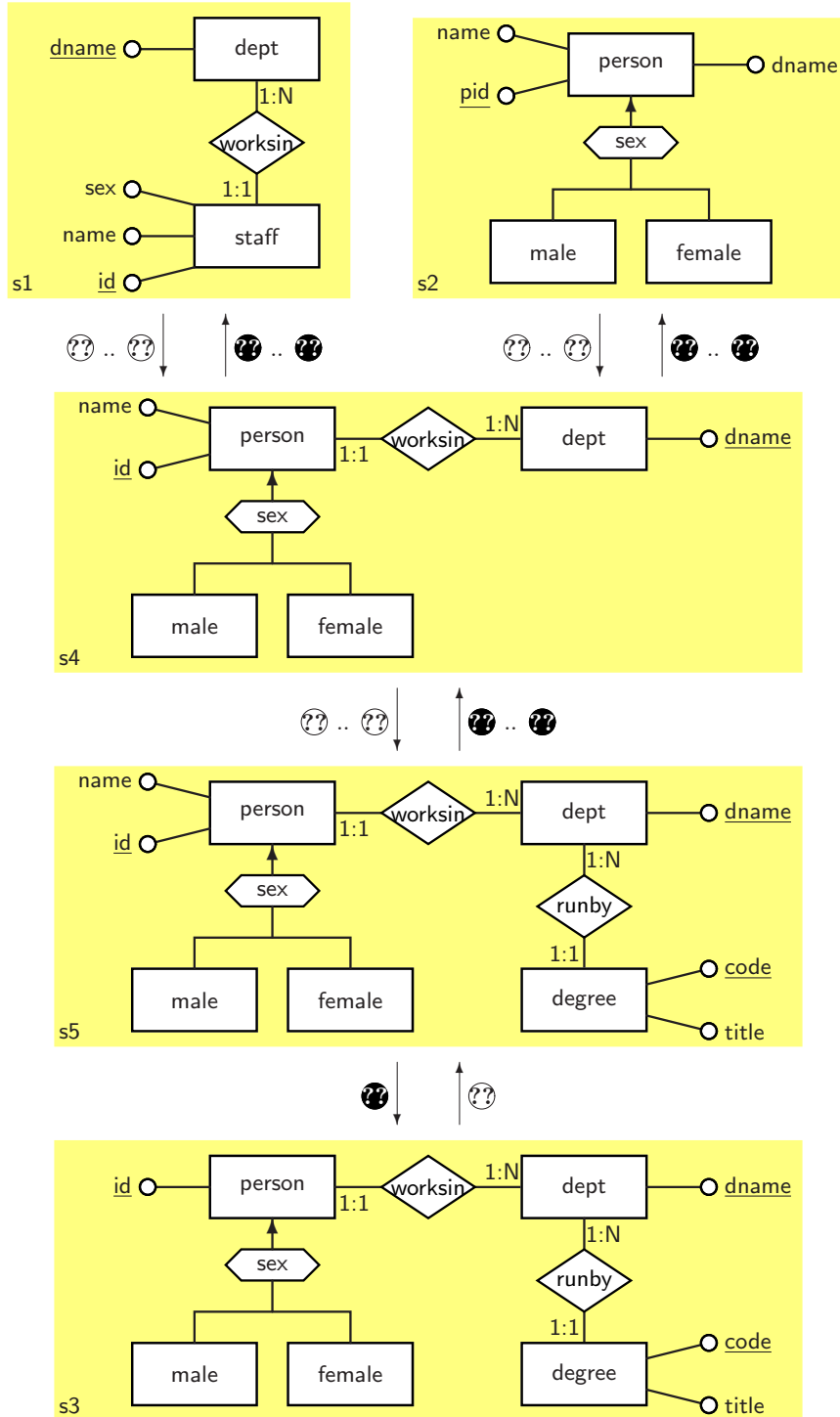


Figure 1: Example ER schemas, and the transformations that relate them

s4 → s5

- ⑫ extendEntity(⟨⟨degree⟩⟩)
- ⑬ extendAttribute(⟨⟨degree, code, key⟩⟩)
- ⑭ extendAttribute(⟨⟨degree, title, notnull⟩⟩)
- ⑮ extendRelationship(⟨⟨runby, degree, dept⟩⟩)

Note that these transformations may be composed to form a direct transformation from s3 to s4 as follows (where white on black text for the transformation number indicates an inverse of the black on white transformation):

s3 → s4

- ?? extendAttribute(⟨⟨person, name, notnull⟩⟩)
- ?? contractRelationship(⟨⟨runby, degree, dept⟩⟩)
- ?? contractAttribute(⟨⟨degree, title, notnull⟩⟩)
- ?? contractAttribute(⟨⟨degree, code, key⟩⟩)
- ?? contractEntity(⟨⟨degree⟩⟩)

4 Relational Models and Sample Data

Figure ?? illustrates three relational schemas equivalent to the three EDB schemas of Figure ??, on the basis that the ER modelling language being used identifies instances of entities by using the key attribute of the entity.

Given this relational model and data we can derive instances of the ER model. For example, s2 has the following extent:

⟨⟨person⟩⟩ = [{1}, {2}, {3}, {4}, {5}]
 ⟨⟨person, pid⟩⟩ = [{1, 1}, {2, 2}, {3, 3}, {4, 4}, {5, 5}]
 ⟨⟨person, name⟩⟩ = [{1, 'Alex'}, {2, 'Dimitri'}, {3, 'Mike'}, {4, 'Nerissa'}, {5, 'Peter'}]
 ⟨⟨person, dept⟩⟩ =
 [{1, 'CS-BBK'}, {2, 'CS-BBK'}, {3, 'Comp-IC'}, {4, 'Comp-IC'}, {5, 'Comp-IC'}]
 ⟨⟨male⟩⟩ = [{2}, {3}, {5}]
 ⟨⟨female⟩⟩ = [{1}, {4}]

staff				dept
<u>id</u>	name	sex	workins	<u>dname</u>
1	Alex	F	CS-BBK	CS-BBK
2	Dimitri	M	CS-BBK	Comp-IC
3	Mike	M	Comp-IC	
4	Nerissa	F	Comp-IC	
5	Peter	M	Comp-IC	

(a) s1

person			male	female
<u>pid</u>	name	dname	<u>id</u>	<u>id</u>
1	Alex	CS-BBK	2	1
2	Dimitri	CS-BBK	3	4
3	Mike	Comp-IC	5	
4	Nerissa	Comp-IC		
5	Peter	Comp-IC		

(b) s2

person		male	female
<u>id</u>	worksin	<u>id</u>	<u>id</u>
1	CS-BBK	2	1
2	CS-BBK	3	4
3	Comp-IC	5	
4	Comp-IC		
5	Comp-IC		

dept	degree		
<u>dname</u>	<u>dcode</u>	title	runby
CS-BBK	1234	MSc Computing	CS-BBK
Comp-IC	G500	BEng Computing	Comp-IC
	G500	MEng Computing	Comp-IC

(c) s3

Figure 2: Relational models and data

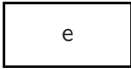
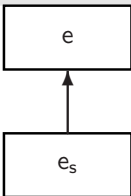
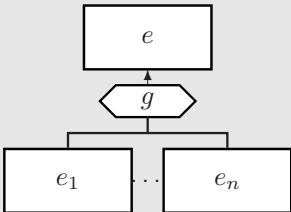
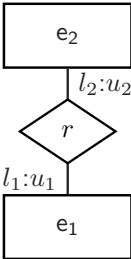
Name	Symbol	Description
entity set		Describes a set of entities in the UoD which share some common properties. We denote the entity by the scheme $\langle\langle e \rangle\rangle$ and can enumerate the values of the entity in variable x by the expression $x \leftarrow \langle\langle e \rangle\rangle$.
attribute		A set of attributes which are of a certain type, and are associated with members of an entity set. We denote the attribute by the scheme $\langle\langle e, a \rangle\rangle$. Each instance of the entity must be associated an instance of the attribute, and we can enumerate. We can enumerate the values of the association be entity and attributes by the expression $\{x, y\} \leftarrow \langle\langle e, a \rangle\rangle$.
key attribute		An attribute which forms part of the key for the entity.
null attribute		An attribute for which not every instance of $\langle\langle e \rangle\rangle$ appears as instance of $\langle\langle a \rangle\rangle$.
subset		Denotes that any instance of the subset entity $\langle\langle e_s \rangle\rangle$ is also a member of the superset entity $\langle\langle e \rangle\rangle$.
generalisation		Denotes a series of subset entities $\langle\langle e_1 \rangle\rangle \dots \langle\langle e_n \rangle\rangle$ of superset $\langle\langle e \rangle\rangle$, which have the additional property that the instances of $\langle\langle e \rangle\rangle$ must be also instances of exactly one of $\langle\langle e_1 \rangle\rangle \dots \langle\langle e_n \rangle\rangle$. If an asterisk is put by the g , then the above is relaxed such that some instances of e may exist which are not instances of $\langle\langle e_1 \rangle\rangle \dots \langle\langle e_n \rangle\rangle$.
relationship		A set of binary relationships between entities, where the relationships are all of a certain type. We denote the relationship by $\langle\langle r, e_1, e_2 \rangle\rangle$. The cardinality constraint $l_1:u_1$ gives the minimum and maximum number of instances of e_2 that may be associated with any one instance e_1 , and $l_2:u_2$ gives the minimum and maximum number of e_1 instances that may be associated to a particular instance of e_2 . We can enumerate the values of the association be entity and attributes by the expression $\{x, y\} \leftarrow \langle\langle r, e_1, e_2 \rangle\rangle$.

Table 1: Constructs and declarative semantics of an entity-relationship model