The University Database Integration: An AutoMed Example

AutoMed Technical Report 12: Draft Version A

Peter McBrien
Dept. of Computing, Imperial College, pjm@doc.ic.ac.uk

Abstract

This reports describes a contrived set of database schemas, and how they are integrated using the AutoMed approach. It serves to illustrate several common techniques used in a AutoMed integration of databases.

1 Introduction

AutoMed [2] supports many methodologies for performing data integration and hence forming a **network** of pathways joining schemas together. Here we describe a simple methodology based on forming union-compatible schemas, the general structure of which is illustrated in Figure 1. Each of the n local schemas LS_i is first transformed into a "union" schema US_i. These n union schemas are syntactically identical, and this is asserted by creating a sequence of id transformation steps between each pair of union schemas US_i and US_{i+1}, of the form id $(US_i : c, US_{i+1} : c)$ for each schema construct c. These id transformations between pairs of union schemas are generated automatically by the AutoMed software. An arbitrary one of the union schemas can then be selected for further transformation into the global schema GS. This two stage process reflects first schema conformance, followed by schema integration and restructuring.

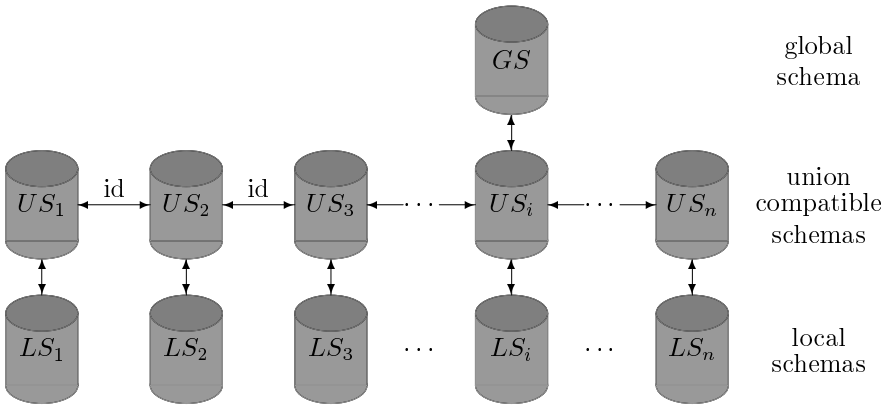

Figure 1: A general AutoMed Integration

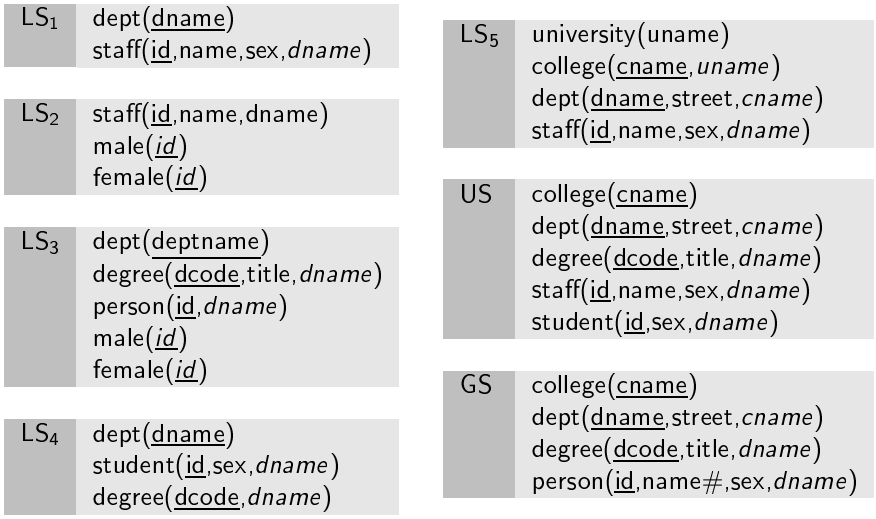

Figure 2: Example schemas

2 The University Global Database Example

Figure 2 gives some specific schemas that illustrate the integration approach of Figure 1. Primary key attributes are underlined, foreign key attributes are in italics and nullable attributes are suffixed by #.

The schema US_1 is the simplist to integrate into US since the only constructs it contains are exactly the same in extent as the same constructs in US. Hence the pathway $\mathsf{LS}_1 \rightarrow \mathsf{US}$ in Example 1 involves only *extend* transformations to add each of the tables student, college and degree, plus the two attributes of dept which are present in US but not LS_1, and *add* tramsformations to assert the missing primary and foreign key constraints:

Pathway 1 $\mathsf{LS}_1 \rightarrow \mathsf{US}$

① extendTable(⟨⟨student⟩⟩)
② extendField(⟨⟨student,id⟩⟩)
③ extendField(⟨⟨student,sex⟩⟩)
④ extendField(⟨⟨student,dname⟩⟩)
⑤ addPK(⟨⟨student,⟨⟨student,id⟩⟩⟩⟩)
⑥ addFK(⟨⟨student,⟨⟨student,dname⟩⟩,dept,⟨⟨dept,dname⟩⟩⟩⟩)
⑦ extendTable(⟨⟨college⟩⟩)
⑧ extendField(⟨⟨college,cname⟩⟩)
⑨ addPK(⟨⟨college,⟨⟨college,cname⟩⟩⟩⟩)
⑩ extendTable(⟨⟨degree⟩⟩)
⑪ extendField(⟨⟨degree,dcode⟩⟩)
⑫ extendField(⟨⟨degree,title⟩⟩)
⑬ extendField(⟨⟨degree,dname⟩⟩)
⑭ addPK(⟨⟨degree,⟨⟨degree,dcode⟩⟩⟩⟩)
⑮ addFK(⟨⟨degree,⟨⟨degree,dname⟩⟩,dept,⟨⟨dept,dname⟩⟩⟩⟩)
⑯ extendField(⟨⟨dept,street⟩⟩)
⑰ extendField(⟨⟨dept,cname⟩⟩)
⑱ addFK(⟨⟨dept,⟨⟨dept,cname⟩⟩,college,⟨⟨college,cname⟩⟩⟩⟩)

In Example 2, transformations ⑲ –㉚ use *extend* transformations to state that the tables student, college and degree in US cannot be derived from LS_2. Then ㉝ –㊱ use the dname attribute of person to derive the dept table in US, and use *extend* transformations for the two attributes street and cname that cannot be derived from LS_2. Finally, in ㊶ –㊾ the male and female relations of LS_2 are restructured into the single sex attribute of staff.

Pathway 2 $LS_2 \rightarrow US$

⑲ extendTable(⟨⟨student⟩⟩)
⑳ extendField(⟨⟨student,id⟩⟩)
㉑ extendField(⟨⟨student,sex⟩⟩)
㉒ extendField(⟨⟨student,dname⟩⟩)
㉓ addPK(⟨⟨student,⟨⟨student,id⟩⟩⟩⟩)
㉔ extendTable(⟨⟨college⟩⟩)
㉕ extendField(⟨⟨college,cname⟩⟩)
㉖ addPK(⟨⟨college,⟨⟨college,cname⟩⟩⟩⟩)
㉗ extendTable(⟨⟨degree⟩⟩)
㉘ extendField(⟨⟨degree,dcode⟩⟩)
㉙ extendField(⟨⟨degree,title⟩⟩)
㉚ extendField(⟨⟨degree,dname⟩⟩)
㉛ addPK(⟨⟨degree,⟨⟨degree,dcode⟩⟩⟩⟩)
㉜ addFK(⟨⟨degree,⟨⟨degree,dname⟩⟩,dept,⟨⟨dept,dname⟩⟩⟩⟩)
㉝ addTable(⟨⟨dept⟩⟩, [x | (y,x) ← ⟨⟨staff,dname⟩⟩])
㉞ addField(⟨⟨dept,dname⟩⟩, [(x,x) | x ← ⟨⟨dept⟩⟩])
㉟ extendField(⟨⟨dept,street⟩⟩)
㊱ extendField(⟨⟨dept,cname⟩⟩)
㊲ addFK(⟨⟨degree,⟨⟨degree,dname⟩⟩,dept,⟨⟨dept,dname⟩⟩⟩⟩)
㊳ addFK(⟨⟨staff,⟨⟨staff,dname⟩⟩,dept,⟨⟨dept,dname⟩⟩⟩⟩)
㊴ addFK(⟨⟨student,⟨⟨student,dname⟩⟩,dept,⟨⟨dept,dname⟩⟩⟩⟩)
㊵ addFK(⟨⟨dept,⟨⟨dept,cname⟩⟩,college,⟨⟨college,cname⟩⟩⟩⟩)
㊶ addField(⟨⟨staff,sex⟩⟩, [(x,'M') | x ← ⟨⟨male⟩⟩] ++ [(x,'F') | x ← ⟨⟨female⟩⟩])
㊷ deleteFK(⟨⟨male,⟨⟨male,id⟩⟩⟩⟩, ⟨⟨person,⟨⟨person,id⟩⟩⟩⟩)
㊸ deletePK(⟨⟨male,⟨⟨male,id⟩⟩⟩⟩)
㊹ deleteField(⟨⟨male,id⟩⟩, [(x,x) | x ← ⟨⟨male⟩⟩])
㊺ deleteTable(⟨⟨male⟩⟩, [x | (x,'M') ← ⟨⟨staff,sex⟩⟩])
㊻ deleteFK(⟨⟨female,⟨⟨female,id⟩⟩⟩⟩, ⟨⟨person,⟨⟨person,id⟩⟩⟩⟩)
㊼ deletePK(⟨⟨female,⟨⟨female,id⟩⟩⟩⟩)
㊽ deleteField(⟨⟨female,id⟩⟩, [(x,x) | x ← ⟨⟨female⟩⟩])
㊾ deleteTable(⟨⟨female⟩⟩, [x | (x,'F') ← ⟨⟨staff,sex⟩⟩])

The example uses the **Intermediate Query Language (IQL)**, which is the defualt query language supported by the AutoMed implementation. In IQL ++ is the bag union operator and the construct $[e \mid Q_1; \ldots Q_n]$ is a **comprehension** [1]. The expressions Q_1 to Q_n are termed **qualifiers**, each qualifier being either a **filter** or a **generator**. A filter is a boolean-valued expression. A generator has syntax $p \leftarrow c$ where p is a **pattern** and c is a bag-valued expression. In IQL, the patterns p are restricted to be single variables or tuples of variables.

The pathway $LS_3 \rightarrow US$ contains *extend* steps ㊿ –㊼ to add the missing student and college tables, which are textually the same as ⑲ –㉕ . It then renames deptname, adds the missing attributes of dept, renames person to staff, and adds the missing name attribute. Finally, in steps ㊹ –㋇ it does the same restructuring as steps ㊶ –㊾ of $LS_2 \rightarrow US$, converting the male and female relations into the single sex attribute of staff.

Pathway 3 $LS_3 \rightarrow US$

⑤⓪ extendTable(⟨⟨student⟩⟩)
⑤① extendField(⟨⟨student,id⟩⟩)
⑤② extendField(⟨⟨student,sex⟩⟩)
⑤③ extendField(⟨⟨student,dname⟩⟩)
⑤④ addPK(⟨⟨student,⟨⟨student,id⟩⟩⟩⟩)
⑤⑤ addFK(⟨⟨student,⟨⟨student,dname⟩⟩,dept,⟨⟨dept,deptname⟩⟩⟩⟩)
⑤⑥ extendTable(⟨⟨college⟩⟩)
⑤⑦ extendField(⟨⟨college,cname⟩⟩)
⑤⑧ addPK(⟨⟨college,⟨⟨college,cname⟩⟩⟩⟩)
⑤⑨ renameField(⟨⟨dept,deptname⟩⟩, ⟨⟨dept,dname⟩⟩)
⑥⓪ extendField(⟨⟨dept,street⟩⟩)
⑥① extendField(⟨⟨dept,cname⟩⟩)
⑥② renameTable(⟨⟨person⟩⟩, ⟨⟨staff⟩⟩)
⑥③ extendField(⟨⟨staff,name⟩⟩)
⑥④ addField(⟨⟨staff,sex⟩⟩, [(x,′ M′) | x ← ⟨⟨male⟩⟩] ++ [(x,′ F′) | x ← ⟨⟨female⟩⟩])
⑥⑤ deleteFK(⟨⟨male,⟨⟨male,id⟩⟩⟩⟩, ⟨⟨person,⟨⟨person,id⟩⟩⟩⟩)
⑥⑥ deletePK(⟨⟨male,⟨⟨male,id⟩⟩⟩⟩)
⑥⑦ deleteField(⟨⟨male,id⟩⟩, [(x, x) | x ← ⟨⟨male⟩⟩])
⑥⑧ deleteTable(⟨⟨male⟩⟩, [x | (x,′ M′) ← ⟨⟨staff,sex⟩⟩])
⑥⑨ deleteFK(⟨⟨female,⟨⟨female,id⟩⟩⟩⟩, ⟨⟨person,⟨⟨person,id⟩⟩⟩⟩)
⑦⓪ deletePK(⟨⟨female,⟨⟨female,id⟩⟩⟩⟩)
⑦① deleteField(⟨⟨female,id⟩⟩, [(x, x) | x ← ⟨⟨female⟩⟩])
⑦② deleteTable(⟨⟨female⟩⟩, [x | (x,′ F′) ← ⟨⟨staff,sex⟩⟩])

The pathway $\mathsf{LS}_4 \rightarrow \mathsf{US}$ contains a sequence of *extend* steps for its missing information.

Pathway 4 $\mathsf{LS}_4 \rightarrow \mathsf{US}$

⑦③ extendTable(⟨⟨staff⟩⟩)
⑦④ extendField(⟨⟨staff,id⟩⟩)
⑦⑤ extendField(⟨⟨staff,name⟩⟩)
⑦⑥ extendField(⟨⟨staff,sex⟩⟩)
⑦⑦ extendField(⟨⟨staff,dname⟩⟩)
⑦⑧ addPK(⟨⟨staff,⟨⟨staff,id⟩⟩⟩⟩)
⑦⑨ addFK(⟨⟨staff,⟨⟨staff,dname⟩⟩,dept,⟨⟨dept,dname⟩⟩⟩⟩)
⑧⓪ extendTable(⟨⟨college⟩⟩)
⑧① extendField(⟨⟨college,cname⟩⟩)
⑧② addPK(⟨⟨college,⟨⟨college,cname⟩⟩⟩⟩)
⑧③ extendField(⟨⟨degree,title⟩⟩)
⑧④ extendField(⟨⟨dept,street⟩⟩)
⑧⑤ extendField(⟨⟨dept,cname⟩⟩)

The pathway $\mathsf{LS}_5 \rightarrow \mathsf{US}$ contains a sequence of *extend* steps for its missing information and also three *contract* steps to remove the university relation and its attributes.

Pathway 5 $\mathsf{LS}_5 \rightarrow \mathsf{US}$

⑧⑥ renameTable(⟨⟨person⟩⟩, ⟨⟨staff⟩⟩)
⑧⑦ extendTable(⟨⟨student⟩⟩)
⑧⑧ extendField(⟨⟨student,id⟩⟩)

89 extendField(⟨⟨student,sex⟩⟩)
90 extendField(⟨⟨student,dname⟩⟩)
91 addPK(⟨⟨student,⟨⟨student,id⟩⟩⟩⟩)
92 addFK(⟨⟨student,⟨⟨student,dname⟩⟩,dept,⟨⟨dept,dname⟩⟩⟩⟩)
93 extendField(⟨⟨degree⟩⟩)
94 extendField(⟨⟨degree,dcode⟩⟩)
95 extendField(⟨⟨degree,title⟩⟩)
96 extendField(⟨⟨degree,dname⟩⟩)
97 addPK(⟨⟨degree,⟨⟨degree,dcode⟩⟩⟩⟩)
98 addFK(⟨⟨degree,⟨⟨degree,dname⟩⟩,dept,⟨⟨dept,dname⟩⟩⟩⟩)
99 deleteFK(⟨⟨college,⟨⟨college,uname⟩⟩,university,⟨⟨university,uname⟩⟩⟩⟩)
100 contractField(⟨⟨college,uname⟩⟩)
101 deletePK(⟨⟨university,⟨⟨university,uname⟩⟩⟩⟩)
102 contractField(⟨⟨university,uname⟩⟩)
103 contractTable(⟨⟨university⟩⟩)

Finally, we list below the pathway from the union schema US to the global schema GS:

Pathway 6 US → GS

104 addTable(⟨⟨person⟩⟩, ⟨⟨staff⟩⟩ ++ ⟨⟨student⟩⟩)
105 addField(⟨⟨person,id⟩⟩, ⟨⟨staff,id⟩⟩ ++ ⟨⟨student,id⟩⟩)
106 addField(⟨⟨person,name⟩⟩, ⟨⟨staff,name⟩⟩)
107 addField(⟨⟨person,sex⟩⟩, ⟨⟨staff,sex⟩⟩ ++ ⟨⟨student,sex⟩⟩)
108 addField(⟨⟨person,dname⟩⟩, ⟨⟨staff,dname⟩⟩ ++ ⟨⟨student,dname⟩⟩)
109 addPK(⟨⟨person,⟨⟨person,id⟩⟩⟩⟩)
110 addFK(⟨⟨person,⟨⟨person,dname⟩⟩,dept,⟨⟨dept,dname⟩⟩⟩⟩)
111 deleteFK(⟨⟨student,⟨⟨student,dname⟩⟩,dept,⟨⟨dept,dname⟩⟩⟩⟩)
112 deletePK(⟨⟨student,⟨⟨student,id⟩⟩⟩⟩)
113 deleteField(⟨⟨student,id⟩⟩, [(x, y) | x ← ⟨⟨student⟩⟩; (x, y) ← ⟨⟨person,id⟩⟩])
114 deleteField(⟨⟨student,sex⟩⟩, [(x, y) | x ← ⟨⟨student⟩⟩; (x, y) ← ⟨⟨person,sex⟩⟩])
115 deleteField(⟨⟨student,dname⟩⟩, [(x, y) | x ← ⟨⟨student⟩⟩; (x, y) ← ⟨⟨person,dname⟩⟩])
116 deleteTable(⟨⟨student⟩⟩), [x | x ← ⟨⟨person⟩⟩; not(member⟨⟨staff⟩⟩x)])
117 deleteFK(⟨⟨staff,⟨⟨staff,dname⟩⟩,dept,⟨⟨dept,dname⟩⟩⟩⟩)
118 deletePK(⟨⟨staff,⟨⟨staff,id⟩⟩⟩⟩)
119 deleteField(⟨⟨staff,id⟩⟩, [(x, y) | x ← ⟨⟨staff⟩⟩; (x, y) ← ⟨⟨person,id⟩⟩])
120 deleteField(⟨⟨staff,name⟩⟩, [(x, y) | x ← ⟨⟨staff⟩⟩; (x, y) ← ⟨⟨person,name⟩⟩])
121 deleteField(⟨⟨staff,sex⟩⟩, [(x, y) | x ← ⟨⟨staff⟩⟩; (x, y) ← ⟨⟨person,sex⟩⟩])
122 deleteField(⟨⟨staff,dname⟩⟩, [(x, y) | x ← ⟨⟨staff⟩⟩; (x, y) ← ⟨⟨person,dname⟩⟩])
123 deleteTable(⟨⟨staff⟩⟩, [x | x ← ⟨⟨person⟩⟩; (x, y) ← ⟨⟨person,name⟩⟩])

3 Conclusions

See `http://www.doc.ic.ac.uk/automed/` for more details of AutoMed.

References

[1] P. Buneman *et al.* Comprehension syntax. *ACM SIGMOD Record*, 23(1):87–96, 1994.

[2] P.J. McBrien and A. Poulovassilis. Data integration by bi-directional schema transformation rules. In *Proc. ICDE'03 (to appear)*, 2003.