Reduce, Reuse, Recycle : Practical Approaches to Schema Integration, Evolution and Versioning

John F. Roddick and Denise de Vries

School of Informatics and Engineering
Flinders University,
PO Box 2100, Adelaide, South Australia 5001
roddick@infoeng.flinders.edu.au

Abstract. Three themes are apparent in recent schema integration, evolution and versioning research. First, the need to reduce the number of schema changes that are necessary. The approach here has been to build into the conceptual and data models the scope to accommodate modest changes to definition. Second, research that aims to reuse the current schema definition through procedures that mask the changes through sophisticated wrappers or techniques for multiple extensional data. Finally, techniques that enable schema change to be accommodated as seamlessly and as painlessly as possible. All these approaches have their limitations and strengths. This paper investigates each of these approaches and outlines the current research directions in schema integration, evolution and versioning.

1 Introduction

Changes in the functional requirements for software systems commonly result in changes to the underlying database. Similarly, changes to the storage schema for a database often necessitates amendments to the software systems to accommodate the changes. Aside from any data loss that may result, these changes cause both application and database to age resulting in higher maintenance and redevelopment costs [1]. Schema evolution and versioning research thus aims to reduce the effect of changes to schema on the system. Schema versioning is related to schema evolution in that it aims to provide not only evolution (change) capability but also the provision of multiple schema instances for data update and retrieval.

Schema versioning requires data integration which takes data held under multiple local schemata (perhaps different versions) and provides a unified view of those data, while schema (or view) integration takes multiple user views and provides a common conceptual schema. Thus in many respects, schema and data integration and schema evolution and versioning can be viewed as cognate problems in that there are multiple sources of data which are held, and which need to be viewed, on demand, through multiple schemata [2].

Research in these fields has been relatively steady now for two decades, as illustrated in Figure 1 and can broadly be classified under a *3R-hierarchy* which is broadly speaking, in order of desirability:

J.F. Roddick et al. (Eds.): ER Workshops 2006, LNCS 4231, pp. 209–216, 2006.

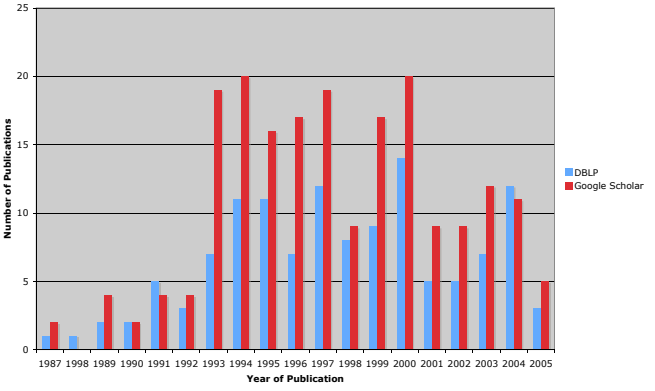

Fig. 1. Approximately publications by year in schema evolution and schema versioning (source DBLP, Google Scholar)

- **Reduce.** Those strategies that attempt to reduce the number of database schema changes that need to be made by building into the conceptual and data models the capability to accommodate changes to definition.
- **Reuse.** Those strategies that reuse the schema definition (and the data held under that definition) by providing translation mechanisms, such as wrappers.
- **Recycle.** Those strategies that ease the pain of integration and evolution, including techniques for data coercion.

This paper explores those three themes. Given the surveys that are available [3,4,5,6,7], the paper focusses specifically on techniques that have been developed in the past few years. Despite useful work in a variety of other areas (see for example [8,9,10]) we limit ourselves here to a discussion on those approaches to database schema management.

2 The Practical and Theoretical Limits of Schema Evolution

As discussed by Qian [11], Miller *et al.* showed that in order to update data stored under two different schemata using the opposite schemata, theoretically they must be equivalent – all valid instances of some schema S_1 must be able to be stored under S_2 and vice-versa [12,13]. Specifically, $S_1 \equiv S_2$ iff $I(S_1) \rightarrow I(S_2)$ is bijective where $I(S)$ is the set of all valid instances of S – essentially the *information capacity* of S. This means that, in theory, *full* schema versioning is unattainable and much research in the area adopts the weaker concept of *partial schema versioning* in which data stored under any historical schema may be viewed through any other schema but may only be updated through one specified schema version - normally the current or active schema. This is essentially the old *view update* problem.

However, in practice, many schema changes that expand or reduce the information capacity of a schema can be done without loss of information. This is the case for example, for domains defined too large for the data, for the creation of subclass relations from a single relation where the subclass type attribute already exists, and so on. It is common practice where there is some ambiguity in the requirements definition of a system to allow for data that once implemented never materialises and, as a result, changes to schemata to adhere to the data collected are not uncommon [7].

Thus the limits for *practical schema versioning* in a database $\mathfrak{D}$ are that $S_1 \overset{p}{\equiv} S_2$ iff $I'(\mathfrak{D}|S_1) \rightarrow I'(\mathfrak{D}|S_2)$ is bijective where $I'(\mathfrak{D}|S_n)$ is the set of all instances of S_n inferrable from $\mathfrak{D}$ given the constraints of S_n. This means that whether the integration of two schema is possible is dependent on the data held as well as the schema definition and while this makes the ability to undertake wholesale change less predictable, it may provide an acceptable level of support in many practical situations. A number of schema matching techniques have been suggested [14] and the ones detailing instance matching [15,16] may be able to be modified to determine whether instance level equivalence exists. The use of Armstrong relations [17] or induced dependencies [18] may also assist. This is an area for further research.

3 Reducing the Requirement for Schema Change

Since it is changes that result in system and schema modification and thus in the ageing of a system, one approach is to avoid the need for change by developing schemata that can handle a modest level of change through data modification. One approach which has been proposed as a solution to this is the concept of mesodata [19,20]. Mesodata aims to add power to the relational data model by providing greater semantics to the *domain* (note, not the type) of an attribute by allowing attributes to be defined over domains with complex data structures. The mesodata concept does not add complex data types to the relational model – the *type* of an attribute remains a simple scalar type while the *domain* of the attribute allows the values taken by the attribute to be placed within some complex structure. For example, while the code for a disease might be defined as `CHAR(5)`, disease codes exist within an agreed international classification (such as ICD10) a tree-structure that relates diseases and other observations by group.

When a schema changes two events typically occur - the application is modified and recompiled to deal with the changes and the data may be converted to a new format [21,22]. By adding a mesodata layer to the structure, the domains over which the data are defined may be modified leaving untouched both the data and the attribute definition. Even simple country, organisation or region name changes may be one-to-one, one-to-many, many-to-one or many-to-many and more complex changes might involve the reorganisation of the entity.

The typical solution is to convert all old values and replace them with new values. This could be an ongoing task and results can result in an irreversible loss of information. The mesodata solution would be to use the mesodata type TREE

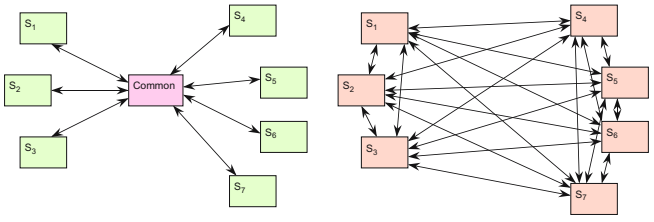

Fig. 2. Basic Architectures for Schema Reuse

to map old values to the new values. Extended SQL operators could be used to retrieve data values with the advantage that the original values are not lost.

4 Reusing Schema Definition and Data

Coercing data to a common schema is often not a practical solution and thus the largest group of techniques focusses on retaining data and schema definition and providing interfaces using either indirect *Source-Common-Target* (SCT) [23,24] or direct *Source-Target* (ST) architectures. The SCT architecture benefits from requiring fewer interfaces than the SC architecture ($2n$ as opposed to $n^2/2$, where n is the number of schemata) but can suffer from a greater level of information loss as the common schema is often non-maximal. Wrappers [25,26] and XML-based markups are effectively a version of the SCT approach in which the interface provided adheres, at least in principle, to a more or less agreed commonality of schema.

Implicit in some work in schema versioning is the concept of a completed schema C (similar to the completed relation of Clifford and Warren [27]) which contains the minimal union of all attributes which have been defined during the life of the relation. The domain of each attribute in C is considered syntactically general enough to hold all data stored under every version of the relation scheme R and the implicit primary key of C is defined as the maximal set of key attributes for the scheme over time. Versions of the schema (denoted S_{time}) can be seen to be views of C. Thus the relation scheme active during the interval $t_1, R_{t_1} = (S_{t_1}, T)$. The current schema through which updates may be performed is denoted S_{now}. A view function V_{t_1} maps C to a subset of the attributes in a schema S_{t_1} active during t_1 with a converse function W_{t_2} mapping from S_{t_2} to C.

Grandi [28,29] provides a solution that enhances this view by providing a *multi-pool* solution. In this proposal, each schema version can have different extensional data and thus the same object might have an independent representation.

Rosenthal et al. [30] recommend that we must go beyond after-the-fact semantic integration to actively guiding semantic choices with semantic management by:

- producing new areas of semantic agreement not just correspondences between existing systems,
- considering the needs of people in multiple roles (e.g. owners, architects, users, developers) to have a greater share of what the data mean,

– broadening the definition of *semantics* to describe what data instances are collected and desired, not just concept definitions and relationships.

For temporal databases, which lend themselves well to schema versioning through the provision of a temporally-aware schema, data coercion is inapplicable due to their write-once, append-only nature. Work in this area has included Jensen and Böhlen [31,32] who discuss two versions of a lossless method of schema versioning – first, a full but exponentially complex, multi-dimensional conditionally evolving schema (MD-CES) and, secondly, a restriction on this that is tractable for one and two-dimensional temporal datasets.

5 Strategies for Accommodating Schema Modification

As would be expected, the desire to retain information means that data coercion is avoided in many proposals. However, it is a popular practical solution and, although not accommodated in some models, we would argue that there are some circumstances in which data coercion would be the most desirable mechanism of evolution, particularly when a consistent interpretation of the source data is required and/or there is confidence that historical information will not be required.

Early approaches to data coercion investigated either a strict conversion [33] or a lazy conversion [22] of data to the new format. Recent work has improved on this by providing a more detailed view of how this may occur. The TVSE Model [34], for example, uses temporal and versioning concepts to manage schema evolution in object-oriented databases. The model assists users to manage the evolution history of both intensional and extensional databases and proposes a language to derive and modify schema versions and to update the data associated with them.

It is also useful if the decisions made during database design can be consulted and the users can interact with schema changes at a high level. Hick and Hainaut [35] investigate how requirements changes can be propagated to database schemas, to data and to programs through reference to the design process. Other work, particularly that of Bernstein et al. [36,37] and Madhavan and Halevy [38], provides support for meta data management that provides a higher level view in which models can be mapped to each other.

6 Conclusions

Despite the research to date, schema integration, evolution and versioning are far from being solved as evidenced by the almost total lack of functionality in commercial DBMS. This lack of support may be for a number of reasons. First, the lack of strategies for dealing with schema versioning in practice which may be because the motivation for schema change is seldom examined and, given many possible solutions, the best outcome is often related to the motivation for change. Second, the need for manual intervention and the incompleteness of most solutions. Many solutions deal with specific situations only and most proposals, if implemented, would require at least a check that the process has provided an

adequate solution. Finally, many solutions target the object-oriented (cf. [39,40]) rather than the (commercially more popular) relational model cf. [28]).

We would argue that one of the next steps in schema handling is to transfer the body of research from theory to practice including extensions to commercial DBMS and SQL. To do this the user's motivation needs to be involved at a more fundamental level with, for example, links back to the original conceptual design.

References

1. Lientz, B.: Issues in software maintenance. ACM Computing Surveys **15** (1983) 271–278
2. McBrien, P., Poulovassilis, A.: Schema evolution in heterogeneous database architectures, a schema transformation approach. In: CAiSE'02, Birkbeck College and Imperial College (2002)
3. Lautemann, S.E.: An introduction to schema versioning in OODBMS. Database and Expert Systems Applications, 1996. Proceedings., Seventh International Workshop on (1996) 132–139
4. Lemke, T.: Schema evolution in OODBMS: A selective overview of problems and solutions. Technical Report IDEA.WP.22.O.002, University of Bonn (1994)
5. Li, X.: A survey of schema evolution in object-oriented databases. In: 31st International Conference on Technology of Object-Oriented Language and Systems, Nanjing, China, IEEE (1999) 362–371
6. Roddick, J.F.: A survey of schema versioning issues for database systems. Information and Software Technology **37** (1995) 383–393
7. Shankaranarayanan, G., Ram, S.: Research issues in database schema evolution - the road not taken. Technical Report Technical Report 2003-15, University of Arizona (2003)
8. Fan, H., Poulovassilis, A.: Schema evolution in data warehousing environmentsa schema transformation-based approach. In Atzeni, P., Chu, W., Lu, H., Zhou, S., Ling, T.W., eds.: International Conference on Conceptual Modeling. Volume 3288., Shanghai, Springer (2004) 639–653
9. Noy, N.F.E., Klein, M.E.: Ontology evolution: Not the same as schema evolution. Knowledge and Information Systems **6** (2004) 428–440
10. Haase, P., Stojanovic, L.: Consistent evolution of OWL ontologies. In Gómez-Pérez, A., Euzenat, J., eds.: Second European Semantic Web Conference. Volume 3532 of LNCS., Heraklion, Greece, Springer (2005) 182–197
11. Qian, X.: Correct schema transformations. In Apers, P.M.G., Bouzeghoub, M., Gardarin, G., eds.: Advances in Database Technology - EDBT'96, 5th International Conference on Extending Database Technology. Volume 1057 of LNCS., Avignon, France, Springer (1996) 114–128
12. Miller, R., Ioannidis, Y., Ramakrishnan, R.: The use of information capacity in schema integration and translation. In Agrawal, R., Baker, S., Bell, D., eds.: 19th International Conference on Very Large Data Bases, VLDB'93, Dublin, Ireland, Morgan Kaufmann (1993) 120–133
13. Miller, R.J., Ioannidis, Y.E., Ramakrishnan, R.: Schema equivalence in heterogeneous systems: Bridging theory and practice. Information Systems **19** (1994) 3 31
14. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB Journal **10** (2001) 334:350

15. Li, W.S., Clifton, C.: SemInt: a tool for identifying attribute correspondences in heterogeneous databases using neural network. Data and Knowledge Engineering **33** (2000) 49–84
16. Doan, A.H., Domingos, P., Halevy, A.Y.: Reconciling schemas of disparate data sources: a machine-learning approach. ACM SIGMOD International Conference on Management of Data (2001) 509–520
17. Armstrong, W.W.: Dependency structures of data base relationships. In: 6th International Federation for Information Processing Congress (IFIP). Volume 74., North-Holland, Amsterdam (1974) 580–583
18. Roddick, J.F., Craske, N.G., Richards, T.J.: Handling discovered structure in database systems. IEEE Transactions on Knowledge and Data Engineering **8** (1996) 227–240
19. de Vries, D., Roddick, J.F.: Facilitating database attribute domain evolution using mesodata. In Grandi, F., ed.: 3rd International Workshop on Evolution and Change in Data Management (ECDM2004). Volume 3289 of LNCS., Shanghai, China, Springer (2004) 429–440
20. de Vries, D.: Mesodata : Engineering Domains for Attribute Evolution and Data Integration. PhD thesis, Flinders University (2006)
21. Ferrandina, F., Meyer, T., Zicari, R.: Implementing lazy database updates for an object database system. In Bocca, J.B., Jarke, M., Zaniolo, C., eds.: 20th International Conference on Very Large Data Bases, VLDB'94, Santiago, Chile, Morgan Kaufmann (1994) 261–272
22. Tan, L., Katayama, T.: Meta operations for type management in object-oriented databases - a lazy mechanism for schema evolution. In Kim, W., Nicolas, J.M., Nishio, S., eds.: 1st International Conference on Deductive and Object-Oriented Databases, DOOD'89, Kyoto, Japan, North-Holland (1989) 241–258
23. Bergamaschi, S., Castano, S., Vincini, M.: Semantic integration of semistructured and structured data sources. SIGMOD Record **28** (1999) 54–59
24. Cavalnese, D., De Giacomo, G., Lenzerini, M., Nardi, D., Rosati, R.: Information integration: Conceptual modeling and reasoning support. In: 3rd IFCIS International Conference on Cooperative Information Systems (CoopIS), New York City, NY (1998) 280–291
25. Chawathe, S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J.D., Widom, J.: The TSIMMIS project: Integration of heterogeneous information sources. In: 16th Meeting of the Information Processing Society of Japan, Tokyo, Japan (1994) 7–18
26. Hammer, J., Garcia-Molina, H., Nestorov, S., Yerneni, R., Breunig, M., Vassalos, V.: Template-based wrappers in the TSIMMIS system. SIGMOD Record **26** (1997) 532–535
27. Clifford, J., Warren, D.: Formal semantics for time in databases. ACM Transactions on Database Systems **8** (1983) 214–254
28. Grandi, F.: A relational multi-schema data model and query language for full support of schema versioning. In: National Conference on Advanced Database Systems, Isola d'Elba, Italy (2002) 323–336
29. Grandi, F.: SVMgr: A tool for the management of schema versioning. In Atzeni, P., Chu, W., Lu, H., Zhou, S., Ling, T.W., eds.: 23rd International Conference on Conceptual Modeling (ER2004). Volume 3288., Shanghai, China, Springer-Verlag (2004) 860–861
30. Rosenthal, A., Seligman, L.J., Renner, S.: From semantic integration to semantics management: case studies and a way forward. SIGMOD Record **33** (2004) 44–50

31. Jensen, O.G., Böhlen, M.H.: Evolving relations. In: Database Schema Evolution and Meta-Modeling, Proc. International Workshop on Foundations of Models and Languages for Data and Objects. Volume 2065 of LNCS. Springer (2001) 115
32. Jensen, O.G., Böhlen, M.H.: Multitemporal conditional schema evolution. In Grandi, F., ed.: 3rd International Workshop on Evolution and Change in Data Management (ECDM2004). Volume 3289 of LNCS., Shanghai, China, Springer (2004) 441–456
33. Penney, D., Stein, J.: Class modification in the gemstone object-oriented dbms. OOPSLA '87 (SIGPLAN Notices) **22** (1987) 111–117
34. Edelweiss, N., Moreira, A.: Temporal and versioning model for schema evolution in object-oriented databases. Data and Knowledge Engineering **53** (2005) 99–128
35. Hick, J.M., Hainaut, J.L.: Database application evolution: A transformational approach. Data and Knowledge Engineering **Article in Press** (Preprint)
36. Bernstein, P.A.: Applying model management to classical meta data problems. Conference on Innovative Data Systems Research (CIDR) (2003) 209–220
37. Melnik, S., Rahm, E., Bernstein, P.A.: Rondo: a programming platform for generic model management. In: 2003 ACM SIGMOD International Conference on Management of data, San Diego, California, ACM Press (2003) 193–204
38. Madhavan, J., Halevy, A.Y.: Composing mappings among data sources. In Freytag, J.C., Lockemann, P.C., Abiteboul, S., Carey, M.J., Selinger, P.G., Heuer, A., eds.: 29th International Conference on Very Large Data Bases (VLDB), Berlin, Germany, Morgan Kaufmann (2003) 572–583
39. Franconi, E., Grandi, F., Mandreoli, F.: A semantic approach for schema evolution and versioning in object-oriented databases. In Lloyd, J.W., Dahl, V., Furbach, U., Kerber, M., Lau, K.K., Palamidessi, C., Pereira, L.M., Sagiv, Y., Stuckey, P.J., eds.: 1st International Conference on Computational Logic, CL'00. Volume 1861., London, UK, Springer (2000) 1048–1062
40. Grandi, F., Mandreoli, F.: A formal model for temporal schema versioning in object-oriented databases. Data and Knowledge Engineering **46** (2003) 123–167