SQL/SE - A Query Language Extension for Databases Supporting Schema Evolution

John F. Roddick
School of Computer and Information Science,
University of South Australia,
The Levels, SA 5095, South Australia

Abstract

The incorporation of a knowledge of time within database systems allows for temporally related information to be modelled more naturally and consistently. Adding this support to the meta-database further enhances its semantic capability and allows elaborate interrogation of data. This paper presents SQL/SE, an SQL extension capable of handling schema evolution in relational database systems.

Introduction

Recent proposals to incorporate a knowledge of time within database systems has allowed for temporally related information to be modelled more effectively. Numerous models which accommodate various aspects of time have been proposed and a number of bibliographies and surveys have been compiled[1-7]. The concept of time management within database systems can be extended further by incorporating temporal support within the meta-database thus permitting the modelling of changing schemas. More precisely it allows the history of changes made to a schema to be recorded without overwriting existing schema definitions and therefore allows the *shape* of the database to be remembered. Discussion on schema evolution has principally centred around the relational[8-10] and object-oriented paradigms[11-16].

Associated with the support for schema evolution within database systems is the requirement for a query language which is able to accommodate the enhanced semantics. This can manifest itself as a richer query language structure and/or more appropriate output. For instance, one of the facilities afforded by schema evolution support is the ability, through a (sufficiently powerful) query language, to reissue queries made at a previous point in time with the assurance of identical results. Such assurances are not possible within temporal databases unless a static schema is maintained.

The ANSI and ISO standards committees have proposed an extension to the SQL standard to provide for limited temporal support[17]. The adoption of SQL in this paper does not imply that it is considered the best vehicle for the inclusion of schema evolution support, indeed others support the temporal dimension more completely[18-25]. The proposed extensions are applicable generally and SQL has been adopted because of its universal familiarity. A working knowledge of SQL is assumed throughout the paper.

1. Schema Evolution in Database Systems

In the majority of the literature to date, temporal support has been investigated only within the database itself. A significant exception is the *Grammatical Database Model* of Laine *et al* [8] which uses the same temporally augmented language to describe both the structure of the database and meta-database. McKenzie and Snodgrass[9] propose an extension to the relational algebra to support schema evolution. Roddick[10] investigates the ramifications of incorporating temporal support within the meta-database and thus the accommodation of schema evolution.

Relation Meta-Relation

	Valid Time		Transaction Time	
Rel. Name	Start Date	End Date	Start Date	End Date
Empl.Salary	1-Mar-1991	∞	1-Mar-1991	∞

Attribute Meta-Relation

		Valid Time		Transaction Time	
Att. Name	Domain	Start Date	End Date	Start Date	End Date
Name	Text	1-Mar-1991	∞	1-Mar-1991	∞
Salary	Dollars	1-Mar-1991	∞	1-Mar-1991	∞
Dept	Departments	5-Jun-1991	∞	5-Jun-1991	∞

Relation/Attribute Meta-Relation

		Valid Time		Transaction Time	
Rel.Name	Att.Name	Start Date	End Date	Start Date	End Date
Empl.Salary	Name	1-Mar-1991	∞	1-Mar-1991	∞
Empl.Salary	Salary	1-Mar-1991	∞	1-Mar-1991	∞
Empl.Salary	D-O-Birth	1-Mar-1991	1-Oct-1991	1-Mar-1991	∞
Empl.Salary	Dept	10-Jan-1992	∞	7-Jan-1992	∞

Figure 1 - Evolutionary Schema

The meta-database holds the structure of the database. Changes to the structure of the database results in an addition to or a change in the semantics of the information held. In the case of a populated database, these changes can affect the inferences drawn from pre-existing data.

Figure 1 represents an evolutionary schema in the form of a set of relations. The dates associated with each tuple within the schema give the time the change was recorded (the transaction time) and the time the change became effective (the valid time). Thus the schema in Figure 1 shows that `D-O-Birth` has since been deleted from the `Empl.Salary` relation and that `Dept` was defined as an attribute in the database many months before it was incorporated into the `Empl.Salary` relation. Further, the addition of `Dept` to the `Empl.Salary` relation was predated by three days (as shown by the valid and transaction times). The referent for transaction and valid times within the database and the meta-database can be represented as in Figure 2.

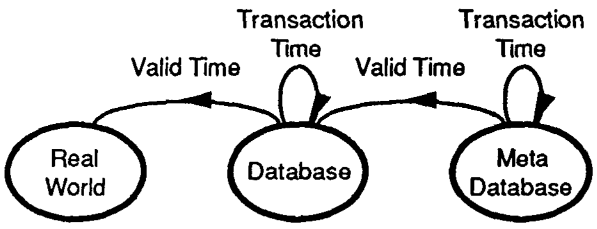

Figure 2 - Referent for Transaction and Valid Time in Database and Meta-Database

Temporal support within databases augments the semantic capability of a model. It is suggested that the accommodation of changing schema provides significant semantic enhancement over *ordinary* (non-schema evolutionary) temporal databases. In particular, schema evolution offers the ability to:

- intelligently apply null logic to queries where the data held are incomplete;
- retain historical information without retaining an obsolete database structure;
- interpret database queries in their proper temporal context;
- model the changes to the schema for later evaluation.

Application areas include distributed databases where database queries may be issued asynchronously with database schema modifications[26] and version control within system development environments[27].

2. Query Language Extensions for Schema Evolution Support

The augmentation of a query language to support schema evolution involves functional, economic and pragmatic considerations. Functional issues only are discussed and an extended syntax is suggested. The extensions to the grammar outlined below are given in BNF and augment that given in Date[28].

2.1. Completed Relations

Clifford and Warren[29] examine the concept of the *completed relation* as a relation containing a tuple for every key that has existed in the relation. This concept can be applied to the meta-relations to create a *completed schema* with relations containing every attribute that has ever been defined for them. For instance, given the relation/attribute meta-relation in Figure 1, the completed schema would contain an `Empl.Salary` relation with four attributes despite the schema never having more than three at any point in time. This simplifies exhaustive extraction of data from the database by obviating the need to issue multiple queries, each targetting different time periods. This can be accommodated by altering the SELECT statement as follows:

```
query-spec ::=
      SELECT [ ALL | DISTINCT ]
      selection table-exp

selection ::=
      scalar-exp-commalist | * | **
```

where ** is interpreted as providing completed relations for all relations specified in `table-exp`. Alternatively completed relations can be specified on a individual basis:

```
table-exp ::= from-clause
              [ where-clause ]
              [ group-by-clause ]
              [ having-clause ]

from-clause ::= table-ref
              { , table-ref }

table-ref  ::= [ COMPLETED ] table
              [ range-variable ]
```

2.2. Null Values

The introduction of undefined attributes introduces new semantics to null values in the database. A three-valued null logic has been proposed[30, 31] with semantics of:

- attribute applicable but value unknown (unk);
- attribute inapplicable (dne);
- nothing known about value or applicability (ni).

These proposals presuppose static schemas. The concept of null values in defined attributes retains the semantics suggested. For attributes *not currently* defined their value is dependent on the reason behind their non-existence. Intuitively, an employee does not cease to have a date of birth simply because the information is no longer collected. Thus in this case a value unknown (unk) null would be an appropriate interpretation for the missing data. However a change in the object system may result in additional attributes being introduced. In this case an attribute inapplicable (dne) would be appropriate.

Given criticisms in the literature of SQL's null handling ability[32], significant changes to the schema definition language to cater for enhanced null value semantics are beyond the scope of this paper. However it is clear that, given the adoption of the three-valued null logic discussed above, changes to the schema definition language should include the ability to define the semantics of attributes outside of their defined interval. These additional semantics can be accommodated within the query language by amending the test for null values currently available:

```
test-for-null ::=
    column-ref IS
    [ NOT ] ( NULL | DEFINED )
```

The specification of NOT DEFINED returns those tuples for which the schema lacks the specified attribute(s) at that time.

2.3. Dual Times Lines

Historical (valid-time) support within the meta-database allows past and future schemas to be accessed for the retrieval of data. Rollback (transaction-time) support allows temporal information with respect to past changes to the schema to be recorded. While the accommodation of both types of support would be useful, the necessity of implementing transaction-time handling is less clear. An incorporation of schema rollback support would allow the use of a schema *as seen* from any other point in the past. For instance, if a subsequently corrected error had been made in the database definition, rollback support would allow the view applicable to the erroneous subschema to be presented. While it is acknowledged that this facility may not merit the cost of implementation, SQL/SE includes proposals for both historical and rollback support.

The use of past or future schemas may be accommodated by the specification of a *schema-time* reference:

```
table-exp ::= from-clause
            [ where-clause ]
            [ group-by-clause ]
            [ having-clause ]
            [ schema-time-clause ]

schema-time-clause  ::=
    SCHEMA-FOR  time-ref

time-ref  ::=
    date-time | NOW | QUERY
```

This would have a default of SCHEMA-FOR NOW which would select the currently active schema. SCHEMA-FOR QUERY would utilise the schema applicable to the temporal reference of the query while SCHEMA-FOR date-time would allow selection of the schema by specification of an explicit date and time. The output produced by the various schemas is discussed in more detail in Section 2.4. Embedded SQL applications could utilise this facility by substituting the compile date-time for date-time thus shielding the application against subsequent schema changes. This would be useful in effecting online schema modifications, even in a distributed environment.

Rollback support may be accommodated in a similar fashion.

```
table-exp ::= from-clause
            [ where-clause ]
            [ group-by-clause ]
            [ having-clause ]
            [ schema-time-clause ]
            [ schema-as-at-clause ]

schema-as-at-clause  ::=
            SCHEMA-AS-AT  time-ref
```

with a default of SCHEMA-AS-AT NOW which would utilise the current view of the schema history. Other SCHEMA-AS-AT time references would allow use of different views of the schema history.

2.4. Modifications to Query Language Output

By adopting a default of NOW in both valid and transaction time support for schema evolution, the addition of schema evolution support does not impact users who do not require such support; output will be in accordance with the present view of the current schema. The only suggested enhancement is the use of more explanatory error messages when currently inactive attributes are specified in queries.

Empl.Salary Relation

			Valid Time		Transaction Time	
Name	**Salary**	**Dept**	**Start Date**	**End Date**	**Start Date**	**End Date**
Smith, A.	$34,300	?	12-Mar-1991	∞	15-Mar-1991	18-Mar-1991
Smith, A.	$34,350	?	12-Mar-1991	1-Nov-1991	18-Mar-1991	∞
Smith, A.	$35,000	?	1-Nov-1991	10-Jan-1992	17-Oct-1991	∞
Smith, A.	$35,000	CS	10-Jan-1992	∞	12-Jan-1992	∞

Figure 3

The specification of SCHEMA-FOR date-time can also be handled in a straight forward manner with the system utilising a static, past version of the schema.

The use of SCHEMA-FOR QUERY allows the extra potential of databases with schema evolution support to be employed. For instance, use of this option would provide output tailored to the shape of the schema at the reference time specified in the query. For instance given the data shown in Figure 3 [*] if :

```
SELECT * FROM Empl.Salary ES
    WHEN ES OVERLAP "December 1991"
```

returns:

```
NAME         SALARY   DEPT   START  END
----------   -------  ----   ------ ------
Smith, A.      35000  ?      011191 100192
```

then given the schema described earlier, (Figure 1) :

```
SELECT * FROM Empl.Salary ES
    WHEN ES OVERLAP "December 1991"
            SCHEMA-FOR QUERY
```

would return:

```
NAME         SALARY   START  END
----------   -------  ------ ------
Smith, A.      35000  011191 100192
```

Queries specifying intervals relating to more than one version of the schema would need to utilise a semi-completed relation with all attributes active during the interval being returned, possibly with null values in some columns. For instance:

```
SELECT * FROM Empl.Salary ES
    WHEN ES OVERLAP "January 1992"
            SCHEMA-FOR QUERY
```

would return:

```
NAME         SALARY   DEPT   START  END
----------   -------  ----   ------ ------
Smith, A.      35000  ?      011191 100192
Smith, A.      35000  CS     100192   inf.
```

3. Conclusions and Outstanding Problems

An extension to SQL to permit query language support for databases with schema evolution has been presented. In particular, considerations relating to null value semantics, time lines and the concept of completed relations have been discussed.

While the extensions provide for historical and rollback support of a schema's definition, it is acknowledged that rollback support in particular may be too costly to implement in many cases. Nevertheless, query language support for schema evolution enables the

[*] A temporal SQL extension similar to TQUEL is presented here for the purposes of explaining schema evolution support only. This query selects Empl.Salary tuples with a valid time overlapping the interval defined by "December 1991".

accommodation of a number of advanced data modelling concepts, such as the retention of historical data without the retention of an obsolete database structure and the ability to model changes to a schema. Furthermore it is suggested that such support results in limited changes to, and can generally be provided on top of, current query languages.

REFERENCES

1. J.A. Bubenko. The temporal dimension in information modelling, in *Architecture and Models in Data Base Management Systems*, G. Nijssen (ed.), North-Holland Amsterdam, (1977).

2. A. Bolour, T.L. Anderson, L.J. Dekeyser and H.K.T. Wong. The role of time in information processing: A survey. *SIGMOD RECORD*, **12**(3), (1982).

3. E. McKenzie. Bibliography : temporal databases. *SIGMOD RECORD*, 40-52, (1986).

4. J. Clifford and G. Ariav. Temporal data management : models and systems, in *New Directions for Database Systems Ch. 12*, ABLEX Publishing Co., 168-185. (1986).

5. R.B. Stam and R. Snodgrass. A bibliography on temporal databases. *Data Engineering*, 7(4), 53-61, (1988).

6. D.H.O. Ling and D.A. Bell. Taxonomy of time models in databases. *Information and Software Technology*, **32**(3), 215-224, (1990).

7. J. Roddick and J. Patrick. Temporal semantics - a survey, Tech. Rep. 1/91, Deakin University, Victoria, (1991).

8. H. Laine, O. Maanavilja and E. Peltola. Grammatical database model. *Information Systems*, **4**, 257-267, (1979).

9. E. McKenzie and R. Snodgrass. Scheme evolution and the relational algebra. *Information Systems*, **15**(2), 207-232, (1990).

10. J.F. Roddick. Dynamically changing schemas within database models. *Australian Computer Journal*, **23**(3), 105-109, (1991).

11. J. Banerjee, H.-T. Chou, J.F. Garza, W. Kim, D. Woelk and N. Ballou. Data model issues for object-oriented applications. *ACM Transactions on Office Information Systems*, **5**(1), 3-26, (1987).

12. J. Banerjee, H.-T. Chou, H.J. Kim and H.F. Korth. Semantics and implementation of schema evolution in object-oriented databases. *ACM SIGMOD conference*, San Francisco, CA, (1987).

13. D.J. Penney and J. Stein. Class modification in the GemStone object-oriented DBMS. *SIGPLAN Notices*, **22**, (1987).

14. W. Kim and H.-T. Chou. Versions of schema for object-oriented databases. *ACM SIGMOD Int. Conf. Very Large DataBases*, (1988).

15. S.L. Osborn. The role of polymorphism in schema evolution in an object-oriented database. *IEEE Transactions on Knowledge and Data Engineering*, **1**(3), 310-317, (1989).

16. B.S. Lerner and A.N. Habermann. Beyond schema evolution to database reorganisation. *SIGPLAN Notices*, **25**(10), 67-76, (1990).

17. J. Melton. SQL2 (the sequel): a emerging standard. *Database Programming and Design*, 3(11), 24-32, (1990).

18. J. Clifford. A logical framework for the temporal semantics and natural language querying of historical databases, Ph.D. thesis, State University of New York at Stony Brook, (1982).

19. S.K. Gadia and J. Vaishnav. A query language for a homogeneous temporal database. *4th Annual ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, Portland, Oregon, 51-56, (1985).

20. A.U. Tansel and M.E. Arkun. HQUEL, A query language for historical databases. *3rd International Workshop on Statistical and Scientific Database Management*, Luxembourg, (1986).

21. R. Snodgrass. The temporal query language TQUEL. *ACM Transactions on Database Systems*, **12**(2), 247–298, (1987).

22. S.K. Gadia. A homogeneous relational model and query languages for temporal databases. *ACM Transactions on Database Systems*, **13**(4), (1988).

23. S.B. Navathe and R. Ahmed. Temporal relational model and a query language. *Information Sciences*, **49**, 147-175, (1989).

24. N.L. Sarda. Algebra and query language for a historical data model. *Computer Journal*, 33(1), 11-18, (1990).

25. J. Kim, H. Yoo and Y. Lee. Design and implementation of a temporal query language with abstract time. *Information Systems*, **15**, 349-357, (1990).

26. L. Marinos, M.P. Papazoglou and M. Norrie. Towards the design of an integrated environment for distributed databases, in *Parallel Processing and Applications*, E. Chiricozzi and A. D'Amico (ed.), Elsevier Science Publishers B.V. (North-Holland), 283-288. (1988).

27. R.H. Katz. Toward a unified framework for version modeling in engineering databases. *ACM Computer Surveys*, **22**(4), 375-408, (1990).

28. C.J. Date. *A guide to the SQL standard*, 2nd Ed., Addison-Wesley Publ. Co., (1989).

29. J. Clifford and D.S. Warren. Formal semantics for time in databases. *ACM Transactions on Database Systems*, 8(2), 214–254, (1983).

30. C. Zaniolo. Database relations with null values. *Journal of Computer and System Sciences*, **28**, 142-166, (1984).

31. M.A. Roth, H.F. Korth and A. Silberschatz. Null values in nested relational databases. *Acta Informatica*, **26**, 615-642, (1989).

32. C.J. Date. Null values in database management, in *Relational database : selected writings*, Addison-Wesley, (1986).