ANSI X3H2-96-582

I S O
INTERNATIONAL ORGANIZATION FOR STANDARDIZATION
ORGANISATION INTERNATIONALE DE NORMALISATION

November 22, 1996

Subject: SQL/Temporal

Status: Change Proposal

Title: Addendum to Valid- and Transaction-time Proposals

Source: ANSI Expert's Contribution

Authors: Richard T. Snodgrass

Abstract: This change proposal augments the valid and transaction time proposals to address some minor inconsistencies.

References

[1] Melton, J. (ed.) *SQL/Temporal*. July, 1996. (ISO/IEC JTC 1/SC 21/WG 3 DBL-MCI-0012.)

[2] Snodgrass, R. T., M. H. Böhlen, C. S. Jensen and A. Steiner *Adding Valid Time to SQL/Temporal*, ANSI X3H2-96-501r2, ISO/IEC JTC 1/SC 21/WG 3 DBL-MAD-146r2, November, 1996.

[3] Snodgrass, R. T., M. H. Böhlen, C. S. Jensen and A. Steiner *Adding Transaction Time to SQL/Temporal*, ANSI X3H2-96-502r2, ISO/IEC JTC 1/SC 21/WG 3 DBL-MAD-147r2, November, 1996.

1 Introduction

This addendum addresses some inconsistencies identified by Hugh Darwen. Note: I am still corresponding with Hugh on these changes. I expect to produce and post R1 of this paper shortly, as soon as we reach closure on these details; I will announce that revision as soon as it is available.

2 Overview of Required Changes

Hugh noted that SR3 of Subclause 7.4 <query expression> [2] didn't consistently cover all the boundary cases. The intuition of this SR is that sequenced queries (assertions, constraints), which are specified with `VALIDTIME SELECT` are evaluated "over" tables with valid-time support. There is one exception: query expressions in the from clause are evaluated separately. The reason for this exception is that it is desirable to use a query expression in a from clause to convert a table without valid-time support (e.g., a table with `When` column) into a table with valid-time support for the purposes of the query.

Here is the SR in question, as it appears in [2].

> 3. (Insert this SR) If VALIDTIME is specified and NONSEQUENCED is not specified in the <validtime option> that is contained in the <time option> that is simply contained in <query expression>, then each exposed table, query, or correlation name that is contained in the <query expression body> without an intervening <from clause> shall identify a table with valid-time support and with identical precision P.

We propose to replace it with the following SR.

> 3. (Insert this SR) If VALIDTIME is specified and NONSEQUENCED is not specified in the <validtime option> that is contained in the <time option> that is simply contained in <query expression> QE, then for each explicit or implicit <item qualifier> IQ with a scope clause of QE or of a <query expression> that is contained in QE without an intervening <from clause>, IQ shall be associated with a table with valid-time support and with identical precision P.

The difference is primarily one of emphasizing <item qualifier>s of the scope, whether or not they actually appear elsewhere in the <query expression>. Similar changes are required to analogous SRs of other clauses.

During the course of this discussion, we also realized that there was an oversight in the specification of the `INSERT` statement. There are four cases to be considered.

1. Subject table does not have valid-time support and row(s) to be inserted do not have valid-time support.
2. Subject table has valid-time support and row(s) to be inserted do not have valid-time support.
3. Subject table does not have valid-time support and row(s) to be inserted have valid-time support.
4. Subject table has valid-time support and row(s) to be inserted have valid-time support.

In [2], cases 1 and 2 were incorrectly merged and case 3 was not considered. Case 4 needs no special attention. In this addendum, we provide the correct semantics for all cases.

Finally, in this addendum we correct a typo in an SR or Subclause 12.4.

3 Some Examples

Let's first examine a series of sixteen examples that Hugh provided that illustrate many of the boundary cases. While these examples are by no means comprehensive, they are quite instructive. We identify places where the current change proposals [2, 3] require tweaking.

We assume that `T` is a table without valid-time support and `VT` is a table *with* valid-time support. For each, we state first whether the statement is allowed under [2], then whether it is allowed given the changes in this addendum.

1. `VALIDTIME SELECT C FROM T`

   Disallowed→Disallowed

   `T` is in the scope of a sequenced <query expression> (indicated with `VALIDTIME`). Hence `T` must have valid-time support.

2. `VALIDTIME SELECT 1 FROM T`

   Allowed→Disallowed

   This is disallowed for the same reason. However, it was legal under [2], the reason being that SR3 of <query expression> is in terms of correlation names that were present, rather than correlation names in the scope of the <query expression>. Since `T` was not used in this query, any SR that restricts its use will not apply. In this addendum, we will rephrase the SR to discuss <item qualifier>s in the scope of the <query expression>, rather than correlation names that actually appear. The result is to make this case and the one equivalent with respect to syntactic correctness.

3. `VALIDTIME SELECT COUNT(*) AS CARD FROM T`

   Allowed→Disallowed

   This case is similar to the previous one.

4. `VALIDTIME SELECT COUNT(*) AS CARD FROM T GROUP BY C`

   Allowed→Disallowed

   This case is also similar. Again, the problem is that SR3 was in terms of existing correlation names.

5. `INSERT INTO T VALIDTIME VALUES(1)`

   Disallowed→Allowed

   There are no <item qualifier>s to violate the SR, so the <query expression> is legal. However, [2] requires that only rows without valid-time support be inserted into a table without valid-time support. It is preferable that this insertion be allowed, to ensure temporal upward compatibility. This will be addressed with a new GR in Subclause 12.5 <insert statement>.

6. `INSERT INTO VT VALIDTIME VALUES(1)`

   Allowed→Allowed

   No changes are required here.

7. `INSERT INTO VT VALUES(1)`

   Allowed→Allowed

   No changes are required here. Rows without valid-time support can be inserted into tables with valid-time support.

8. `CREATE ASSERTION A VALIDTIME CHECK NOT EXISTS ( SELECT * FROM T )`

   Allowed→Disallowed

   This is now disallowed, consistent with case 2. The problem with [2] is again that it focuses on existing correlation names rather than possible <item qualifier>s.

9. `CREATE ASSERTION A VALIDTIME CHECK NOT EXISTS ( SELECT C FROM T )`

   Disallowed→Disallowed

   No change is required here.

10. `CREATE ASSERTION A VALIDTIME CHECK NOT EXISTS ( SELECT * FROM VT )`

    Allowed→Disallowed

    This should be consistent with the previous case.

11. `CREATE ASSERTION A VALIDTIME CHECK NOT EXISTS ( VALIDTIME SELECT * FROM VT )`

    Allowed→Disallowed

    `VALIDTIME` is not allowed in a subclause of a <query expression>. This restriction should have also been applied consistently to assertions and check constraints.

    Note that the position of the <time option> is before `CHECK` in an `ASSERTION` statement and within <query expression> in an `INSERT` statement.

12. `CREATE ASSERTION A CHECK NOT EXISTS ( SELECT * FROM VT )`

    Allowed→Allowed

    This takes the current state from `VT`, to ensure temporal upward compatibility.

13. `CREATE ASSERTION A CHECK NOT EXISTS ( VALIDTIME SELECT * FROM VT )`

    Allowed→Disallowed

    `VALIDTIME` should not be allowed inside any subclause.

14. `VALIDTIME SELECT ONE FROM ( SELECT 1 AS ONE FROM T ) X`

    Disallowed→Disallowed

    No changes are required here.

15. `VALIDTIME SELECT ONE FROM ( VALIDTIME SELECT 1 AS ONE FROM T ) X`

    Allowed→Disallowed

    `X` now has valid-time support, so the outer <query expression> is legal. However, the inner <query expression> is now disallowed, as in case 2.

16. `SELECT ONE FROM ( VALIDTIME SELECT 1 AS ONE FROM T ) X`

    Allowed→Disallowed

    This is now disallowed, due to the inner <query expression>, as in the previous example. The outer <query expression> is fine.

3.1 Summary

We make four fairly minor changes to [2, 3] in this addendum.

1. We rephrase the sequenced syntactic restriction in terms of scope. There are numerous SRs that need to be changed, all in a similar manner.
2. To be consistent with <query expression>, we disallow `VALIDTIME` within a `CHECK` clause. This was a simple oversight.
3. We replace GR1 in Subclause 12.5 <insert statement> with two GRs that handle the cases correctly. Cases 1 and 4 were already handled; we specify GRs for cases 2 and 3.
4. SR5 in Subclause 12.4 <delete statement: search> had seventeen words accidentally omitted.

4 Proposed Language Extensions

The syntax is given as extensions to "Database Language SQL — Part 7: Temporal" [1], to be applied after [2, 3].

5 Clause 7 Query expressions

5.1 Subclause 7.4 <query expression>

1) Replace SR3 with the following SR.

3. (Insert this SR) If VALIDTIME is specified and NONSEQUENCED is not specified in the <validtime option> that is contained in the <time option> that is simply contained in <query expression> QE, then for each explicit or implicit <item qualifier> IQ with a scope clause of QE or of a <query expression> that is contained in QE without an intervening <from clause>, IQ shall be associated with a table with valid-time support and with identical precision P.

2) Replace SR8 with the following SR.

8. (Insert this SR) If TRANSACTIONTIME is specified and NONSEQUENCED is not specified in the <transactiontime option> that is contained in the <time option> that is simply contained in <query expression> QE, then for each explicit or implicit <item qualifier> IQ with a scope clause of QE or of a <query expression> that is contained in QE without an intervening <from clause>, IQ shall be associated with a table with transaction-time support.

6 Clause 10 Schema definition and manipulation

6.1 Subclause 10.3 <column definition>

1) Replace SR2b with the following.

2b) If <column constraint> is <check constraint definition> CCD, then for each explicit or implicit <item qualifier> IQ with a scope clause of CCD or of a <query expression> that is contained in CCD without an intervening <from clause>, IQ shall be associated with a table with valid-time support and with identical precision P.

2) Replace SR9b with the following.

9b) If <column constraint> is <check constraint definition> CCD, then for each explicit or implicit <item qualifier> IQ with a scope clause of CCD or of a <query expression> that is contained in CCD without an intervening <from clause>, IQ shall be associated with a table with transaction-time support.

6.2 Subclause 10.4 <table constraint definition>

3) Replace SR3 with the following SR.

3. (Insert this SR) If VALIDTIME is specified and NONSEQUENCED is not specified in <table constraint definition> TCD, then for each explicit or implicit <item qualifier> IQ with a scope clause of TCD or of a <query expression> that is contained in TCD without an intervening <from clause>, IQ shall be associated with a table with valid-time support and with identical precision P.

4) Replace SR7 with the following SR.

3. (Insert this SR) If TRANSACTIONTIME is specified and NONSEQUENCED is not specified in <table constraint definition> TCD, then for each explicit or implicit <item qualifier> IQ with a scope clause of TCD or of a <query expression> that is contained in TCD without an intervening <from clause>, IQ shall be associated with a table with transaction-time support.

6.3 Subclause 10.9 <assertion definition>

5) Replace SR1 with the following SR.

1. (Insert this SR) If VALIDTIME is specified and NONSEQUENCED is not specified in the <validtime option> that is contained in <time option>, then for each explicit or implicit <item qualifier> IQ with a scope clause of the <search condition> SC that is simply contained in <triggered assertion> or of a <query expression> that is contained in SC without an intervening <from clause>, IQ shall be associated with a table with valid-time support and with identical precision P.

6) Replace SR3 with the following SR.

1. (Insert this SR) If TRANSACTIONTIME is specified and NONSEQUENCED is not specified in the <transactiontime option> that is contained in <time option>, then for each explicit or implicit <item qualifier> IQ with a scope clause of the <search condition> SC that is simply contained in <triggered assertion> or of a <query expression> that is contained in SC without an intervening <from clause>, IQ shall be associated with a table with transaction-time support.

7) Insert this new Subclause, "<check constraint definition>", to SQL/Temporal immediately following Subclause 10.9, "<assertion definition>".

6.4 Subclause 10.10 <check constraint definition>

Function

Specify a condition for the SQL-data.

Format

No additional Format Items.

Syntax Rules

1. (Insert this SR) Any <query expression> simply contained in <search condition> without an intervening <from clause> shall not simply contain a <time option>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

7 Clause 12 Data manipulation

7.1 Subclause 12.2 <select statement: single row>

1) Replace SR1 with the following SR.

1. (Insert this SR) If VALIDTIME is specified and NONSEQUENCED is not specified in the <validtime option> that is contained in <time option>, then for each explicit or implicit <item qualifier> IQ with a scope clause of the <table expression> TE or of a <query expression> that is contained in TE without an intervening <from clause>, IQ shall be associated with a table with valid-time support and with identical precision P.

2) Replace SR4 with the following SR.

4. (Insert this SR) If TRANSACTIONTIME is specified and NONSEQUENCED is not specified in the <transactiontime option> that is contained in <time option>, then for each explicit or implicit <item qualifier> IQ with a scope clause of the <table expression> TE or of a <query expression> that is contained in TE without an intervening <from clause>, IQ shall be associated with a table with transaction-time support.

7.2 Subclause 12.4 <delete statement: searched>

3) Replace SR5 with the following SR.

5. (Insert this SR) If VALIDTIME is specified and NONSEQUENCED is not specified in <validtime option> that is contained in <time option>, then for each explicit or implicit <item qualifier> IQ with a scope clause of <search condition> SC or of a <query expression> that is contained in SC without an intervening <from clause>, IQ shall be associated with a table with valid-time support and with identical precision P.

7.3 Subclause 12.5 <insert statement>

4) Replace GR1 with the following GR.

1. (Insert this GR) If R is a table without valid-time support and T is a table with valid-time support, then valid-time support is added to each row of T, by associating with that row a valid-time period from the current timestamp to the end of time with a precision of P.

5) Add the following GR.

2. (Insert this GR) If R is a table with valid-time support and T is a table without valid-time support, then R is replaced with its current valid-time state.

7.4 Subclause 12.7 <update statement: searched>

6) Replace SR5 with the following SR.

5. (Insert this SR) If VALIDTIME is specified and NONSEQUENCED is not specified in the <validtime option> that is contained in <time option>, then for each explicit or implicit <item qualifier> IQ with a scope clause of the <search condition> SC or of a <query expression> that is contained in SC without an intervening <from clause>, IQ shall be associated with a table with valid-time support and with identical precision P.