14.9 <merge statement>

Function

Conditionally update rows of a table, or insert new rows into a table, or both.

Format

```
<merge statement> ::=
    MERGE INTO <target table> [ [ AS ] <merge correlation name> ]
    USING <table reference>
    ON <search condition> <merge operation specification>

<merge correlation name> ::= <correlation name>

<merge operation specification> ::= <merge when clause>...

<merge when clause> ::=
    <merge when matched clause>
  | <merge when not matched clause>

<merge when matched clause> ::=
    WHEN MATCHED THEN <merge update specification>

<merge when not matched clause> ::=
    WHEN NOT MATCHED THEN <merge insert specification>

<merge update specification> ::= UPDATE SET <set clause list>

<merge insert specification> ::=
    INSERT [ <left paren> <insert column list> <right paren> ]
    [ <override clause> ]
    VALUES <merge insert value list>

<merge insert value list> ::=
    <left paren>
    <merge insert value element> [ { <comma> <merge insert value element> }... ]
    <right paren>

<merge insert value element> ::=
    <value expression>
  | <contextually typed value specification>
```

Syntax Rules

1) Neither <merge when matched clause> nor <merge when not matched clause> shall be specified more than once.

2) Let *TN* be the <table name> contained in <target table> and let *T* be the table identified by *TN*. *T* is the *subject table* of the <merge statement>.

3) *T* shall be both updatable and insertable-into.

4) *T* shall not be an old transition table or a new transition table.

5) For each leaf generally underlying table of *T* whose descriptor includes a user-defined type name *UDTN*, the data type descriptor of the user-defined type *UDT* identified by *UDTN* shall indicate that *UDT* is instantiable.

6) If *T* is a view, then <target table> is effectively replaced by:

```
ONLY ( TN )
```

7) Case:

   a) If <merge correlation name> is specified, then let *CN* be the <correlation name> contained in <merge correlation name>.

   b) Otherwise, let *CN* be the <table name> contained in <target table>.

8) The scope of *CN* is <search condition> and <set clause list>.

9) Let *TR* be the <table reference> immediately contained in <merge statement>. *TR* shall not directly contain a <joined table>.

10) The <correlation name> or exposed <table name> that is exposed by *TR* shall not be equivalent to *CN*.

11) If the <insert column list> is omitted, then an <insert column list> that identifies all columns of *T* in the ascending sequence of their ordinal position within *T* is implicit.

12) Case:

   a) If *T* is a referenceable table or a table having an identity column whose descriptor includes an indication that values are always generated, then:

      i) Let *C* be the self-referencing column or identity column of *T*.

      ii) If *C* is an identity column, a system-generated self-referencing column or a derived self-referencing column and *C* is contained in <insert column list>, then <override clause> shall be specified; otherwise, <override clause> shall not be specified.

   b) Otherwise, <override clause> shall not be specified.

13) The <search condition> shall not generally contain a <routine invocation> whose subject routine is an SQL-invoked routine that possibly modifies SQL-data.

14) Each column identified by an <object column> in the <set clause list> is an *update object column*. Each column identified by a <column name> in the implicit or explicit <insert column list> is an *insert object column*. Each update object column and each insert object column is an *object column*.

15) Every object column shall identify an updatable column of *T*.

NOTE 375 — **The notion of updatable columns of base tables is defined in** Subclause 4.14, "Tables". The notion of updatable columns of viewed tables is defined in Subclause 11.22, "<view definition>".

16) No <column name> of *T* shall be identified more than once in in an <insert column list>.

17) Let *NI* be the number of <merge insert value element>s contained in <merge insert value list>. Let EXP_1, EXP_2, ... , EXP_{NI} be those <merge insert value element>s.

18) The number of <column name>s in the <insert column list> shall be equal to *NI*.

19) The declared type of every <contextually typed value specification> *CVS* in a <merge insert value list> is the data type *DT* indicated in the column descriptor for the positionally corresponding column in the explicit or implicit <insert column list>. If *CVS* is an <empty specification> that specifies ARRAY, then *DT* shall be an array type. If *CVS* is an <empty specification> that specifies MULTISET, then *DT* shall be a multiset type.

20) Every <merge insert value element> whose positionally corresponding <column name> in <insert column list> references a column of which some underlying column is a generated column shall be a <default specification>.

21) For 1 (one) ≤ *i* ≤ *NI*, the Syntax Rules of Subclause 9.2, "Store assignment", apply to the column of table *T* identified by the *i*-th <column name> in the <insert column list> and EXP_i as *TARGET* and *VALUE*, respectively.

Access Rules

1) Case:

   a) If <merge statement> is contained, without an intervening <SQL routine spec> that specifies SQL SECURITY INVOKER, in an <SQL schema statement>, then let *A* be the <authorization identifier> that owns that schema.

      i) The applicable privileges for *A* shall include UPDATE for each update object column.

      ii) The applicable privileges for *A* shall include INSERT for each insert object column.

      iii) If <target table> immediately contains ONLY, then the applicable privileges for *A* shall include SELECT WITH HIERARCHY OPTION on at least one supertable of *T*.

   b) Otherwise,

      i) The current privileges shall include UPDATE for each update object column.

      ii) The current privileges shall include INSERT for each insert object column.

      iii) If <target table> immediately contains ONLY, then the current privileges shall include SELECT WITH HIERARCHY OPTION on at least one supertable of *T*.

      NOTE 376 — "current privileges" and "applicable privileges" are defined in Subclause 12.3, "<privileges>".

General Rules

1) If the access mode of the current SQL-transaction or the access mode of the branch of the current SQL-transaction at the current SQL-connection is read-only, and *T* is not a temporary table, then an exception condition is raised: *invalid transaction state — read-only SQL-transaction*.

2) If there is any sensitive cursor *CR* that is currently open in the SQL-transaction in which this SQL-statement is being executed, then

   Case:

   a) If *CR* has not been held into a subsequent SQL-transaction, then either the change resulting from the successful execution of this statement shall be made visible to *CR* or an exception condition is raised: *cursor sensitivity exception — request failed*.

   b) Otherwise, whether the change resulting from the successful execution of this SQL-statement is made visible to *CR* is implementation-defined.

3) If there is any cursor *CR* that is currently open and whose <declare cursor> contained INSENSITIVE, then either the change resulting from the successful execution of this statement shall be invisible to *CR*, or an exception condition is raised: *cursor sensitivity exception — request failed*.

4) The extent to which an SQL-implementation may disallow independent changes that are not significant is implementation-defined.

5) Let *QT* be the table specified by the <table reference>. *QT* is effectively evaluated before update or insertion of any rows in *T*. Let *Q* be the result of evaluating *QT*.

6) For each <merge when clause>,

   Case:

   a) If <merge when matched clause> is specified, then:

      i) For each row *R1* of *T*:

         1) The <search condition> is applied to *R1* with the exposed <table name> of the <target table> bound to *R1* and to each row of *Q* with the exposed <correlation name>s or <table or query name>s of the <table reference> bound to that row. The <search condition> is effectively evaluated for *R1* before updating any row of *T* and prior to the invocation of any <triggered action> caused by the update of any row of *T* and before inserting any rows into *T* and prior to the invocation of any <triggered action> caused by the insert of any row of *T*. Each <subquery> in the <search condition> is effectively executed for *R1* and for each row of *Q* and the results used in the application of the <search condition> to *R1* and the given row of *Q*. If any executed <subquery> contains an outer reference to a column of *T*, then the reference is to the value of that column in the given row of *T*.

            Case:

            A) If <target table> contains ONLY, then *R1* is a subject row if *R1* has no subrow in a proper subtable of *T* and the result of the <search condition> is *True* for some row *R2* of *Q*. *R2* is the matching row.

            B) Otherwise, *R1* is a subject row if the result of the <search condition> is *True* for some row *R2* of *Q*. *R2* is the matching row.

            NOTE 377 — "outer reference" is defined in Subclause 6.7, "<column reference>".

         2) If *R1* is a subject row, then:

            A) Let *M* be the number of matching rows in *Q* for *R1*.

            B) If *M* is greater than 1 (one), then an exception condition is raised: *cardinality violation*.

            C) The <update source> of each <set clause> is effectively evaluated for *R1* before any row of *T* is updated and prior to the invocation of any <triggered action> caused by the update of any row of *T*. The resulting value is the update value.

D) A candidate new row is constructed by copying the subject row and updating it as specified by each <set clause> by applying the General Rules of Subclause 14.12, "<set clause list>".

ii) If *T* is a base table, then each subject row is also an object row; otherwise, an object row is any row of a leaf generally underlying table of *T* from which a subject row is derived.

NOTE 378 — The data values allowable in the object rows may be constrained by a WITH CHECK OPTION constraint. The effect of a WITH CHECK OPTION constraint is defined in the General Rules of Subclause 14.24, "Effect of replacing some rows in a viewed table".

iii) If any row in the set of object rows has been marked for deletion by any <delete statement: positioned> that identifies some cursor *CR* that is still open or updated by any <update statement: positioned> that identifies some cursor *CR* that is still open, then a completion condition is raised: *warning — cursor operation conflict*.

iv) Let *CL* be the columns of *T* identified by the <object column>s contained in the <set clause list>.

v) Each subject row *SR* is identified for replacement, by its corresponding candidate new row *CNR*, in *T*. The set of (*SR*, *CNR*) pairs is the replacement set for *T*.

NOTE 379 — Identifying a row for replacement, associating a replacement row with an identified row, and associating a replacement set with a table are implementation-dependent operations.

vi) Case:

1) If *T* is a base table, then

Case:

A) If <target table> specifies ONLY, then *T* is identified for replacement processing without subtables with respect to object columns *CL*.

B) Otherwise, *T* is identified for replacement processing with subtables with respect to object columns *CL*.

NOTE 380 — Identifying a base table for replacement processing, with or without subtables, is an implementation-dependent mechanism. In general, though not here, the list of object columns can be empty.

2) If *T* is a viewed table, then the General Rules of Subclause 14.24, "Effect of replacing some rows in a viewed table", are applied with <target table> as *VIEW NAME*.

vii) The General Rules of Subclause 14.22, "Effect of replacing rows in base tables", are applied.

b) If <merge when not matched clause> is specified, then:

i) Let *TR1* be the <target table> immediately contained in <merge statement>, let *TR2* be the <table reference> immediately contained in <merge statement>, and let *SC1* be the <search condition> immediately contained in <merge statement>. If <merge correlation name> is specified, let *MCN* be "AS <merge correlation name>"; otherwise, let *MCN* be a zero-length string. Let *S1* be the result of

```
SELECT *
FROM TR1 MCN, TR2
WHERE SC1
```


ii) Let *S2* be the collection of rows of *Q* for which there exists in *S1* some row that is the concatenation of some row *R1* of *T* and some row *R2* of *Q*.

iii) Let *S3* be the collection of rows of *Q* that are not in *S2*. Let *SN3* be the effective distinct name for *S3*. Let *EN* be the exposed <correlation name> or <table or query name> of *TR2*.

iv) Let *S4* be the result of:

```
SELECT EXP1, EXP2, ... , EXPNI
FROM SN3 AS EN
```

v) *S4* is effectively evaluated before insertion of any rows into or update of any rows in *T*.

vi) For each row *R* of *S4*:

1) A candidate row of *T* is effectively created in which the value of each column is its default value, as specified in the General Rules of Subclause 11.5, "<default clause>". The candidate row consists of every column of *T*.

2) If *T* has a column *RC* of which some underlying column is a self-referencing column, then

Case:

A) If *RC* is a system-generated self-referencing column, then the value of *RC* is effectively replaced by the REF value of the candidate row.

B) If *RC* is a derived self-referencing column, then the value of *RC* is effectively replaced by a value derived from the columns in the candidate row that correspond to the list of attributes of the derived representation of the reference type of *RC* in an implementation-dependent manner.

3) For each object column in the candidate row, let C_i be the object column identified by the *i*-th <column name> in the <insert column list> and let SV_i be the *i*-th value of *R*.

4) For every C_i for which one of the following conditions is true:

A) C_i is not marked as unassigned and no underlying column of C_i is a self-referencing column.

B) Some underlying column of C_i is a user-generated self-referencing column.

C) Some underlying column of C_i is a self-referencing column and OVERRIDING SYSTEM VALUE is specified.

D) Some underlying column of C_i is an identity column and OVERRIDING SYSTEM VALUE is specified.

the General Rules of Subclause 9.2, "Store assignment", are applied to C_i and SV_i as *TARGET* and *SOURCE*, respectively.

NOTE 381 — The data values allowable in the candidate row may be constrained by a WITH CHECK OPTION constraint. The effect of a WITH CHECK OPTION constraint is defined in the General Rules of Subclause 14.21, "Effect of inserting a table into a viewed table".

vii) Let *S* be the table consisting of the candidate rows.

Case:

1) If *T* is a base table, then *T* is *identified for insertion of source table S*.

NOTE 382 — Identifying a base table for insertion of a source table is an implementation-dependent operation.

2) If *T* is a viewed table, then the General Rules of Subclause 14.21, "Effect of inserting a table into a viewed table", are applied with *S* as *SOURCE* and *T* as *TARGET*.

viii) The General Rules of Subclause 14.19, "Effect of inserting tables into base tables", are applied.

7) If *Q* is empty, then a completion condition is raised: *no data*.

Conformance Rules

1) Without Feature F781, "Self-referencing operations", conforming SQL language shall not contain a <merge statement> in which a leaf generally underlying table of *T* is generally contained in a <query expression> immediately contained in the <table reference> except as the <table or query name> or <correlation name> of a column reference.

2) Without Feature F781, "Self-referencing operations", conforming SQL language shall not contain a <merge statement> in which a leaf generally underlying table of *T* is an underlying table of any <query expression> generally contained in the <search condition>.

3) Without Feature S024, "Enhanced structured types", conforming SQL language shall not contain a <merge statement> that does not satisfy the condition: for each column *C* identified in the explicit or implicit <insert column list>, if the declared type of *C* is a structured type *TY*, then the declared type of the corresponding column of the <query expression> or <contextually typed table value constructor> is *TY*.

4) Without Feature F312, "MERGE statement", conforming SQL language shall not contain a <merge statement>.