# 16

# Modification

**T. Y. Cliff Leung, Christian S. Jensen, and Richard T. Snodgrass**

## 1 Introduction

This chapter contains an informal specification of the TSQL2 modification statements.

## 2 The Insert Statement

We consider a valid-time employee table as an example.

```
CREATE TABLE Employee (Name CHAR(10), Dept CHAR(10),
   Salary INTEGER)
AS VALID
```

Next, a tuple is inserted which records Ben's salary of 30K and indicates that he was in the Toy department in January of 1993 and from April 1, 1993 until the end of time (`forever`). Conceptually, the table shown next results.

| Name | Department | Salary | V |
|---|---|---|---|
| Ben | Toy | 30 | [1 Jan 1993, 31 Jan 1993] ∪ [1 Apr 1993, forever] |

Suppose that we then record that Ben was in Book department from February 1, 1993 until February 28, 1993.

```
INSERT INTO Employee
VALUES ('Ben', 'Book', 30)
VALID PERIOD '[1 Feb 1993, 28 Feb 1993]'
```

Note that the use of valid clause is similar to its use in valid-time projection—it specifies when the fact is/was valid. After the insertion, the following table results.

| Name | Department | Salary | V |
|---|---|---|---|
| Ben | Toy | 30 | [1 Jan 1993, 31 Jan 1993] ∪ [1 Apr 1993, forever] |
| Ben | Book | 30 | [1 Feb 1993, 28 Feb 1993] |

Note that inserting new tuples is allowed even if there exists a value-equivalent tuple in the table whose time element overlaps with the new tuples, unless explicit constraints prohibit this type of insertion. In this situation, the two tuples will be coalesced, i.e., a single, value-equivalent tuple is created which has as its timestamp the union of the valid-time elements of the two argument tuples. The next example illustrates this.

```
INSERT INTO Employee
VALUES ('Ben', 'Book', 30)
VALID PERIOD '[15 Feb 1993, 15 Mar 1993]'
```

This insertion produces the following table.

| Name | Department | Salary | V |
|---|---|---|---|
| Ben | Toy | 30 | [1 Jan 1993, 31 Jan 1993] ∪ [1 Apr 1993, forever] |
| Ben | Book | 30 | [1 Feb 1993, 15 Mar 1993] |

Tuples may be inserted into a table even without the `VALID` clause. In such cases, the inserted tuples will be assigned a default valid-time element. This default is specified as part of the table definition.

If no default valid-time element is specified for the table, an element `[present, now]` is assumed. For example,

```
INSERT INTO Employee
VALUES ('Ben', 'Sales', 30)
```

results in this table when issued on March 20, 1993.

| Name | Department | Salary | V |
|---|---|---|---|
| Ben | Toy | 30 | [1 Jan 1993, 31 Jan 1993] ∪ [1 Apr 1993, forever] |
| Ben | Book | 30 | [1 Feb 1993, 15 Mar 1993] |
| Ben | Sales | 30 | [20 Mar 1993, now] |

Insert statements with subqueries are treated as above. For example, the following statement inserts tuples of all employees currently in the Toy department.

```
INSERT INTO NewDept
SELECT Name, 'Newtoy', Salary
VALID PERIOD '[1 Feb 1993, 31 Dec 1993]'
FROM Employee
WHERE Dept = 'Toy'
```

## 3 The Delete Statement

There are two different situations, depending on whether or not the `VALID` clause is used. First, let us consider a delete statement with the `VALID` clause.

Suppose that Ben was not an employee during `PERIOD '[1 Feb 1993, 15 Mar 1993]'`.

```
DELETE FROM Employee
WHERE Name = 'Ben'
VALID PERIOD '[1 Feb 1993, 15 Mar 1993]'
```

The information to be deleted is that in tuples with Name value Ben and valid sometime during the specified interval. Thus, for tuple that satisfy the `WHERE` condition, those parts of their elements that overlap with the interval [1 Feb 1993, 15 Mar 1993] are deleted.

Note that deletion may (or may not) result in empty valid-time elements (when the original element is covered by the element specified in the `VALID` clause).

A tuple with with an empty valid-time element is removed from the table—no tuple can have an empty valid-time element as it contains no information. Also observe that the original elements of tuples are not required to cover the valid-time element specified in the `VALID` clause. For example, if the original element is [1 Mar 1993, 31 Mar 1993], the resulting time element is [16 Mar 1993, 31 Mar 1993].

As another example, the following delete statement asserts that Ben no longer works in Toy department, effective April 1, 1993.

```
DELETE FROM Employee
WHERE Name = ’Ben’ AND Dept = ’Toy’
VALID PERIOD ’[1 Apr 1993, forever]’
```

Like in SQL2, no tuple will be removed and an error code is returned if no tuple satisfies the matching qualification.

Finally, we consider the case where the delete statement does not include a `VALID` clause.

```
DELETE FROM Employee
WHERE Name = ’Ben’
```

This delete statement contains no explicit `VALID` clause, but uses implicitly a default valid-time element for deletion. The default [present, forever] is assumed, and the statement is thus equivalent to the following.

```
DELETE FROM Employee
WHERE Name = ’Ben’
VALID PERIOD(CURRENT_DATE, TIMESTAMP ’forever’)
```

## 4 The Update Statement

Updating non-time attributes is treated the same as regular SQL2 update. For example, the following statement changes Ben’s department from ’Toy’ to ’Book’:

```
UPDATE Employee
SET Dept TO ’Book’
WHERE Name = ’Ben’ AND Dept = ’Toy’
```

Note that the valid-time elements of qualifying tuples remain unchanged. Thus, updating a tuple can be modeled as deleting the original tuple and inserting a tuple with new data values, except that the valid-time element value comes from the original tuple.

Updating the valid-times of tuples requires the use of the `VALID` clause. For example, the following statement changes Ben’s department from ’Toy’ to ’Book’ effective during [1 May 93, 31 May 93].

```
UPDATE Employee
SET Dept TO ’Book’ VALID PERIOD ’[1 May 93, 31 May 93]’
WHERE Name = ’Ben’ AND Dept = ’Toy’
```

In this case, updating a tuple can be modeled as deleting the original tuple and inserting a tuple with new data values including the new valid-time element as specified in the `VALID` clause.