SQL language: basics

Update commands

Update commands

- Introduction
- The INSERT command
- The DELETE command
- The UPDATE command

Update commands

Introduction

Update commands (1/3)

- Inserting tuples
- Deleting tuples
- Modifying tuples

Update commands (2/3)

- INSERT
  - inserting new tuples into a table
- DELETE
  - deleting tuples from a table
- UPDATE
  - modifying the content of tuples in a table

Update commands (3/3)

- Update operations alter the state of the database
  - integrity constraints must be checked to ensure that they are still verified
- Each command may update the contents of a single table

Update commands

The INSERT command

The INSERT command

- Inserting a single tuple
  - assignment of a constant value to each attribute
- Inserting multiple tuples
  - read from other tables by means of a SELECT command

Inserting a tuple

INSERT INTO *TableName*
[(ColumnList)]
VALUES *(ConstantList);*

Inserting a tuple: example (no.1)

- Insert product P7 with Name: Jumper, Color: Purple, Size: 40, Store: Helsinki

```
INSERT INTO P (PId, PName, Color, Size, Store)
        VALUES ('P7', 'Jumper', 'Purple', 40, 'Helsinki');
```

- A new tuple is inserted into table P with the specified values

Inserting a tuple: example (no.1)

- Insert product P7 with Name: Jumper, Color: Purple, Size: 40, Store: Helsinki

```
INSERT INTO P (PId, PName, Color, Size, Store)
        VALUES ('P7', 'Jumper', 'Purple', 40, 'Helsinki');
```

- Omitting the field list is equivalent to specifying all fields, according to the column order specified upon table creation
  - If the table schema changes, the INSERT command is no longer valid

Inserting a tuple: example (no.2)

- Insert product P8 with Store: Istanbul, Size: 42

```
INSERT INTO P (PId, Store, Size)
        VALUES ('P8', 'Istanbul', 42);
```

- A new tuple is inserted into table P with the specified values
  - PName and Color are assigned the NULL value
- For all attributes whose values are not specified, the domain of the attribute must allow the NULL value

Referential integrity with insertions

- Insert a new supply for supplier S20, product P20 and quantity 1000

```
INSERT INTO SP (SId, PId, Qty)
        VALUES ('S20', 'P20', 1000);
```

- Referential integrity constraint
  - P20 and S20 must already be present in the P and S tables respectively
  - if the constraint is not satisfied, the insertion should not be executed

Inserting multiple records

INSERT INTO *TableName*
[(ColumnList)]
Query;

- All tuples selected by query *Query* are inserted into table *TableName*
- *Query* is an arbitrary SELECT statement
  - it may not include an ORDER BY clause

Inserting multiple records: example

TOTAL-SUPPLIES (PId, TotalQty)

- For each product, insert the overall supplied quantity into table TOTAL-SUPPLIES
  - aggregate data extracted from table SP

```
SELECT PId, SUM(Qty)
FROM SP
GROUP BY PId
```

Inserting multiple records: example

TOTAL-SUPPLIES (PId, TotalQty)

- For each product, insert the overall supplied quantity into table TOTAL-SUPPLIES

```
INSERT INTO TOTAL-SUPPLIES (PId, TotalQty)
        (SELECT PId, SUM(Qty)
         FROM SP
         GROUP BY PId);
```

Update commands

The DELETE command

The DELETE command

DELETE FROM *TableName*
[WHERE *predicate];*

- Deletion of all tuples satisfying the predicate from table *TableName*
- It must be ensured that the deletion does not cause the violation of referential integrity constraints

The DELETE command: example (no.1)

- Delete all supplies

```
DELETE FROM SP;
```

- If no WHERE clause is specified, all tuples satisfy the selection predicate
  - the contents of table SP are deleted
  - the table itself is *not* deleted

The DELETE command: example (no.2)

- Delete the tuple corresponding to the supplier with code S1

```
DELETE FROM S
WHERE SId='S1';
```

- If SP includes supplies related to the deleted suppliers, the database loses its integrity
  - a violation of the referential integrity constraint between SP and S occurs
  - the deletion must be propagated

The DELETE command: example (no.2)

- Delete the tuple corresponding to the supplier with code S1

```
DELETE FROM S
WHERE SId='S1';

DELETE FROM SP
WHERE SId='S1';
```

- To maintain consistency, the deletion operations must be completed on both tables

The DELETE command: a complex example

- Delete the suppliers based in Paris

```
DELETE FROM S
WHERE City='Paris';
```

- If SP includes supplies referring to the deleted suppliers, the referential integrity constraint between SP and S is violated
  - such supplies must also be deleted from SP

The DELETE command: a complex example

- Delete the suppliers based in Paris

```
DELETE FROM S
WHERE City='Paris';
```

```
DELETE FROM SP
WHERE SId IN (SELECT SId
              FROM S
              WHERE City='Paris');
```

- In which order should the two deletion operations be executed?

The DELETE command: a complex example

- Correct order of execution

```
DELETE FROM SP
WHERE SId IN (SELECT SId
              FROM S
              WHERE City='Paris');
```

```
DELETE FROM S
WHERE City='Paris';
```

Update commands

The UPDATE command

The UPDATE command

```
UPDATE TableName
SET column = expression
        {, column=expression}
[ WHERE predicate];
```

- All records in table *TableName* satisfying the predicate are modified according to the assignments *column=expression* in the SET clause

Updating a tuple

- Update the features of product P1: assign Yellow to Color, increase the size by 2 and assign NULL to Store

```
UPDATE P
SET Color = 'Yellow',
      Size=Size+2,
      Store = NULL
WHERE PId='P1';
```

- The tuple identified by code P1 is updated

Multiple updates

- Update all suppliers based in Paris by doubling the number of employees

```
UPDATE S
SET #Employees=2*#Employees
WHERE City='Paris';
```

- All tuples selected by the predicate in the WHERE clause are updated

Update with nested query

- Update to 10 the quantity of supplied products for all suppliers based in Paris

```
UPDATE SP
SET Qty = 10
WHERE SId IN (SELECT SId
                    FROM S
                    WHERE City='Paris');
```

Updating multiple tables

- Change the code of supplier S2 to S9

```
UPDATE S
SET SId='S9'
WHERE SId='S2';
```

- If SP includes supplies related to the updated suppliers, the referential integrity constraint is violated
  - such supplies must also be updated in SP

Updating multiple tables

- Change the code of supplier S2 to S9

```
UPDATE S
SET SId='S9'
WHERE SId='S2';

UPDATE SP
SET SId='S9'
WHERE SId='S2';
```

- To keep consistency, the update must be completed on both tables