04. Update Operations using SQL

[PM Jat, DAIICT, Gandhinagar]

SQL Data Manipulation Language (DML)

studid character	name character va	progid character	batch character	cpi numeri
1011	Shally	BCS	2010	8.0
1018	Jitendra	BEE	2010	8.0
1110	Rohit	BCS	2011	7.9
1113	Pharhan	BEE	2011	8.0
1118	Aesha	BEE	2011	8.0
1120	Manav	BCS	2011	7.9
1215	Mehul	BCS	2012	7.9
1218	Megh	BEE	2012	8.0

- **INSERT** – add tuple
- **UPDATE** – update tuple
- **DELETE** – delete tuple
- **SELECT** – Query Relations

- First three modifies relation; where as,
- Last one just reads relations

Update Operations

Update operation modify state of a *relation*.

There are three types of update operations. Their names are drawn from their keyword in SQL communicates what each operation should be doing.

INSERT: add new tuples to a relation.

UPDATE: modify data of existing tuples of a relation.

DELETE: remove tuples from a relation.

Since update operations change data; and data to be written may not agree with database constraints. Therefore, DBMS needs to be active while update operations are performed. If any operation leads to constraint violation, operation is not allowed to be performed.

INSERT:

INSERT lets us add one or more tuple in a relation.

Parameters: relation name, tuple values

Result: Specified tuples are added to the operand relation, provided that new tuple(s) do not violate any database constraint.

Tuple values can be specified in one of following two formats:

(1) Ordered List: in this case we need to supply values for all attributes. For example: suppose we want to add a new tuple in student relation, shown here. Following logical expression would add grayed new tuple to student relation shown here.

studid character	name character va	progid character	batch character	cpi numeri
1011	Shally	BCS	2010	8.0
1018	Jitendra	BEE	2010	8.0
1110	Rohit	BCS	2011	7.9
1113	Pharhan	BEE	2011	8.0
1118	Aesha	BEE	2011	8.0
1120	Manav	BCS	2011	7.9
1215	Mehul	BCS	2012	7.9
1218	Megh	BEE	2012	8.0
1219	Misri	MCS	2012	6.8
1312	Amita	MCS	2013	7.0

INSERT STUDENT <1312, 'Amrita', 'MCS', 2013, 7.0>

Terminology Note: in example here, INSERT is operation; student is name of operand relation; values, as set of tuples, are parameters to the operations.

Values in specified tuple need to be in "ordinal" order, i.e., order in which they appear in corresponding CREATE table statement of the operand relation.

In this format it is necessary that we supply values for all attributes of relation. For the attributes, we do not have values, null is specified as placeholder; for example, if in above tuple, if we do not cpi value, then operation can be specified as following-

Logical: INSERT STUDENT <1312, 'Amrita', 'MCS', 2013, NULL>

(2) Set of attribute value pair. We specify new tuple as attribute value pairs as shown in example below.

INSERT STUDENT {<studId, 1219>, <progid, 'MCS '>, <name,' Misri '>, <batch,2012>}

It should be noted that attribute pair need not to be in any order. It is also not necessary that we specify all attributes.

INSERT in SQL

SQL provides INSERT INTO statement for inserting tuples. INSERT INTO allows specifying tuple value in same two ways that we have looked in previous section.

1. First, **specify tuple as ordered list of values** (require all attribute values)
   General syntax:

```
INSERT INTO <table-name> VALUES (<list of values>);
```

   Note that, here we require specifying values for all attributes of relation schema, and in the same order in which they appear in create table statement of the relation.
   Example:

```
INSERT INTO student VALUES (1219,'Misri','MCS',2012,6.8);
```

   We can specify multiple tuples in an INSERT statement; as shown following

```
SQL
INSERT INTO student VALUES
(1219, 'Misri', 'MCS', 2012, 6.8),
(1312, 'Amrita', 'MCS', 2013, 7.0);
```

Take a note of specifying format of data values. Character values are enclosed in apostrophe, while numeric values are not required to be.

For specifying date values, we require little bit extra effort; we cast a string to date type; as shown in an example below (in red color):

```
INSERT INTO employee VALUES
(102, 'Kirit', date '1985-12-08', 'M', 40000, 105, 5);
```

2. Second, **specify tuple as set of attribute/value pair**
   General Syntax

```
INSERT INTO <table-name> (<attribute-list>) VALUES (<value-list>);
```

The value list should be in the order of attribute-list and should match with corresponding domain.

SQL
```
INSERT INTO student (studId, progid, name, batch)
VALUES (1219, 'MCS', 'Misri', 2012);
```

Here also we can specify value for multiple tuples.

INSERT and integrity constraints

We supply values of a tuple as parameter to insert operation. Insert operation will not succeed, if any of following is true in regards to values in tuple being inserted -

1. If value for Primary Key (all attributes in PK) is not specified, or null has been specified for any attribute that is part of key; no attribute that is part of primary key can be null.
2. PK value of tuple being added is already there of some existing tuple – is not allowed ==> if allowed will lead to duplicate values for PK?
3. If we do not specify values for attributes with NOT NULL constraints.
4. If new data will cause duplicate values in attributes with UNIQUE constraint
5. Value specified for FK does not refer to an existing tuple in referred relation. For example, if we attempt inserting a tuple <1501, 'Amrita', 'BEC', 2013, 7.0>; will fail because value in FK progid 'BEC' refers to non-existent value in program relation.

DELETE

Delete operation lets you delete tuples from a relation.

Parameter: row selection criteria; all rows matching the criteria will be deleted from the relation.

Result: all affected rows are deleted from the table provided they do not violate any database constraint.

Example in SQL, should be self-explanatory-

SQL
```
DELETE FROM student WHERE name='Kiran'; --S1
DELETE FROM department WHERE did='CS'; --S2
DELETE FROM student; --S3: deletes all rows from student
```

DELETE operation and integrity constraints

Deletion may violate referential integrity constraints; for example, in statement S2 above might not be successful if the row being deleted here (from department relation) is referred by some program tuples. If a referred tuple is allowed to be deleted; value in FKs of program relation will be violating referential integrity.

As we have seen DDL allows us specifying appropriate action when a tuple referred by other tuples is getting deleted, using ON DELETE clause of FK constraint.

Recall that default action is "NO ACTION" and in that case delete operation violating the constraint is rejected.

Update (modify)

This operation allows modifying data of existing tuple. Here we modify values of selected attributes of specified rows.

Parameters:

1. List of attributes to be modified
2. New values for all specified attributes
3. Row selection criteria

Result: new values are set for the attributes in rows meeting the criteria.

Following few SQL examples, making some updates, should be communicating their intent and effect:

SQL

```
UPDATE student SET cpi=7.5 WHERE studid = 1234; --S1

UPDATE employee
    SET salary=50000, dno=5
    WHERE eno = 101; --S2

UPDATE student SET progid='BCE' WHERE studid = 1234; --S3
```

Update operation and integrity constraints

Update operation may also violate integrity constraints.

A new value for an attribute while updating, should not violate any integrity constraint. An update operation will fail if any of following occurs-

- Modifying PK (values) that leads to duplicate values
- Modifying UNIQU attributes that leads to duplicate value
- Modifying PK that is referred by value in some FK; operation will fail if NO ACTION has been specified in ON UPDATE action in FK definition.

  Note an update operation on attribute that are not part of any constraint is not a problem.

For all constraint checking purposes, modifying a PK attribute is seen as two-step process-

- DELETE tuple for OLD PK value
- INSERT new tuple with new PK value and all other data from old tuple.

Therefore, all rules (or actions) defined for INSERT and DELETE are applied here as applicable.

Also note: Modifying PK values are expensive as well in efficiency terms as physically (on disk) also if will be executed as two-step process; delete and insert.

TRUNCATE Command

TRUNCATE command removes all rows from a table. This command is not part of standard SQL but found in most popular RDBMS 'es because of its efficiency reasons.

Functionally it is equivalent to DELETE FROM …. (i.e., delete all tuples) – that is empties the table; expressed as following:

```
TRUNCATE employee;
```

However, it is implemented for executing much faster; few of the reasons it is faster is because, it can perform the operation in bulk without caring about indexes updates and all.

Whereas DELETE goes through access paths and subject to update indexes, "constraint check", and all.