Database Systems

Lecture #10

Sang-Wook Kim
Hanyang University

Data
And
Knowledge
Engineering

Objectives

- To learn more other SQL commands
  - INSERT
  - DELETE
  - UPDATE
  - Views and indexes

Outline

- INSERT Statement
- DELETE Statement
- UPDATE Statement
- Views
- Indexes

INSERT Statement

- Used to add new tuples to a table
  - Modifies the table
- Types
  - Adding a single tuple
  - Adding a set of tuples

INSERT Statement

- Adding a single tuple
  - **INSERT INTO** <table name>
    VALUES (<list of attributes values>);
  - Values should be listed *in the same order* in which the corresponding attributes were specified in the CREATE TABLE command

INSERT Statement

- Adding a single tuple
  - Example

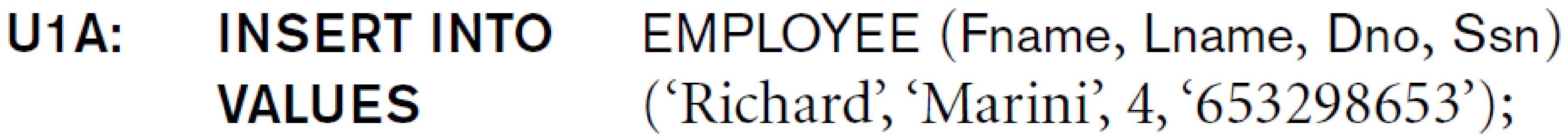
```
U1:   INSERT INTO   EMPLOYEE
      VALUES        ( 'Richard', 'K', 'Marini', '653298653', '1962-12-30', '98
                      Oak Forest, Katy, TX', 'M', 37000, '653298653', 4 );
```

INSERT Statement

- Adding a single tuple
  - Can specify some attributes partially
  - **INSERT INTO** <table name>(<list of attributes>) **VALUES** (<list of attribute values>);
  - Those attributes with NULL allowed or DEFAULT values can be *left out*
    - Will have NULL or DEFAULT value

INSERT Statement

- Adding a single tuple
  - Example



```
U1A:    INSERT INTO    EMPLOYEE (Fname, Lname, Dno, Ssn)
        VALUES         ('Richard', 'Marini', 4, '653298653');
```

INSERT Statement

- Adding a set of tuples
  - Allows users to insert multiple tuples
    - From the *result of a query*
  - **INSERT INTO** <table name>(<list of attributes>) <select statement>;

INSERT Statement

- Adding a set of tuples
  - Example

```
U3A:   CREATE TABLE     WORKS_ON_INFO
       ( Emp_name       VARCHAR(15),
         Proj_name      VARCHAR(15),
         Hours_per_week DECIMAL(3,1) );

U3B:   INSERT INTO   WORKS_ON_INFO ( Emp_name, Proj_name,
                     Hours_per_week )
       SELECT        E.Lname, P.Pname, W.Hours
       FROM          PROJECT P, WORKS_ON W, EMPLOYEE E
       WHERE         P.Pnumber=W.Pno AND W.Essn=E.Ssn;
```

INSERT Statement

- Adding a set of tuples
  - WORKS_ON_INFO table may not be up-to-date
    - Changes to PROJECT, WORKS_ON, or EMPLOYEE table will not change WORKS_ON_INFO automatically
  - Use *views* to automatically reflect the changes of base tables

DELETE Statement

- Removes tuples from a relation
  - Modifies the table
- Uses a WHERE clause to select the tuples to be deleted
- From only one table at a time

DELETE Statement

- **DELETE FROM** <table name>
  WHERE <condition>;
- Missing WHERE clause
  - Unconditionally delete tuples
  - All the tuples in the relation are deleted

DELETE Statement

- Examples

```
U4A:    DELETE FROM    EMPLOYEE
        WHERE          Lname='Brown';

U4B:    DELETE FROM    EMPLOYEE
        WHERE          Ssn='123456789';

U4C:    DELETE FROM    EMPLOYEE
        WHERE          Dno=5;

U4D:    DELETE FROM    EMPLOYEE;
```

UPDATE Statement

- Modify attribute values of one or more tuples selected
  - Modifies the table
- Includes a WHERE clause to select the tuples to be modified
- From only one table at a time
- Additional SET clause in the UPDATE command
  - Specifies attributes to be modified and new values

UPDATE Statement

- **UPDATE** <table name>
  SET <list of 'attribute = value' pairs>
  WHERE <condition>;
- Missing WHERE clause
  - Unconditionally modify tuples
  - All the tuples in the relation are modified

UPDATE Statement

◆ Examples

```
U5:     UPDATE     PROJECT
        SET        Plocation = 'Bellaire', Dnum = 5
        WHERE      Pnumber=10;
```

UPDATE Statement

- Examples

```
U6:     UPDATE    EMPLOYEE
        SET       Salary = Salary * 1.1
        WHERE     Dno = 5;
```

- New values for Salary is calculated by referring to the old values for Salary *before modification*

Views

- Concept of a view in SQL
  - A single table *derived* from other tables
  - Considered to be a *virtual* table

Views

- Characteristics of view
  - View does not necessarily exist in a physical form
    - Limits the update operations that can be applied to views
  - Querying views does not have any limitations
    - Retrieve related tuples from base tables

Creating Views

- CREATE VIEW statement
  - **CREATE VIEW** <view name>
    AS <select statement>
  - Note that the <view name> table does not exist physically
  - Querying views retrieves tuples from <select statement>
  - Basically, the attribute names for the view would be the same as those of the defining tables

Creating Views

- Examples

```
V1:     CREATE VIEW     WORKS_ON1
        AS SELECT       Fname, Lname, Pname, Hours
           FROM         EMPLOYEE, PROJECT, WORKS_ON
           WHERE        Ssn=Essn AND Pno=Pnumber;
```

WORKS_ON1

Fname	Lname	Pname	Hours

Creating Views

◆ Examples

```
V2:     CREATE VIEW    DEPT_INFO(Dept_name, No_of_emps, Total_sal)
        AS SELECT      Dname, COUNT (*), SUM (Salary)
           FROM        DEPARTMENT, EMPLOYEE
           WHERE       Dnumber=Dno
           GROUP BY    Dname;
```

DEPT_INFO

Dept_name	No_of_emps	Total_sal

● Attribute names are specified explicitly

Querying Views

- Queries on a view can be specified in the same way as a table
- Example
  - Retrieve the last name and first name of all employees who work on the 'ProductX' project

```
QV1:    SELECT    Fname, Lname
        FROM      WORKS_ON1
        WHERE     Pname='ProductX';
```

Querying Views

◆ Example

- Without WORKS_ON1 view, QV1 is specified in a more complicate form

```
SELECT    Fname, Lname
FROM      EMPLOYEE, PROJECT, WORKS_ON
WHERE     Ssn=Essn AND Pno=Pnumber
          AND Pname='ProductX';
```

Data
And
Knowledge
Engineering

Querying Views

- Advantages of views
  - Simplify the specification of certain queries
  - Can be used as a security and authorization mechanism
    - Showing only a part of the physical tables
  - Always up-to-date

Querying Views

- Disadvantage of views
  - Can cause a performance issue when a view is defined via a time-consuming query
  - Limited updates

Dropping Views

- Dispose of a view when it is not needed any more
- **DROP VIEW** <view name>;
- Example

```
V1A:    DROP VIEW    WORKS_ON1;
```

Updating Views

- View update problem
  - Updating views can be interpreted as updating underlying base tables
    - Only for some view updates
    - Can be ambiguous
  - Research issue in the database field

Updating Views

- Examples
  - Update the PNAME attribute of `John Smith' from `ProductX' to `ProductY'

```
UV1:    UPDATE  WORKS_ON1
        SET     Pname = 'ProductY'
        WHERE   Lname='Smith' AND Fname='John'
                AND Pname='ProductX';
```

Updating Views

- Examples: Approach 1
  - Change the name of the 'ProductX' tuple in the PROJECT relation to 'ProductY'

```
UPDATE PROJECT    SET    Pname = 'ProductY'
WHERE       Pname = 'ProductX';
```

Updating Views

- Examples: Approach 2
  - Relate `John Smith' to the `ProductY' PROJECT tuple instead of the `ProductX' PROJECT tuple

```
UPDATE WORKS_ON
SET     Pno =    ( SELECT   Pnumber
                   FROM     PROJECT
                   WHERE    Pname='ProductY' )
WHERE   Essn IN  ( SELECT   Ssn
                   FROM     EMPLOYEE
                   WHERE    Lname='Smith' AND Fname='John' )
        AND
        Pno =    ( SELECT   Pnumber
                   FROM     PROJECT
                   WHERE    Pname='ProductX' );
```

Updating Views

- Examples: Impossible update

```
UV2:    UPDATE  DEPT_INFO
        SET     Total_sal=100000
        WHERE   Dname='Research';
```

- Total_sal is defined to be the sum of the individual employee salaries
  - Not stored in the physical base table

Updating Views

- Summary
  - Cannot guarantee successful updates for some queries on a view
  - Possibilities
    - A view with a single defining table is updatable if the view attributes contain the primary key of the base relation
    - Views defined on multiple tables using joins are generally not updatable.
    - Views defined using grouping and aggregate functions are not updatable

Indexes

- Additional meta-data to support fast retrieval
- Indexing attributes:
  - Attributes used to index a table
- If indexing attributes are used in the condition of a retrieval query, the query can be processed very fast

Creating Indexes

- **CREATE INDEX** <index name>
  ON <table name>(<list of attributes>);
- Create an index on <list of attributes> for <table name>

Creating Indexes

- Example
  - **CREATE INDEX** LnameIndex
    ON EMPLOYEE (Lname);

Creating Indexes

- Order of the attribute values in indexes
  - Ascending order (ASC) by default
  - Descending order can be specified by DESC keyword

Creating Indexes

- Example
  - **CREATE INDEX** LnameIndex
    ON EMPLOYEE (Lname **DESC**);

Creating Indexes

- Index can be created on multiple attributes
- Example
  - **CREATE INDEX** NameIndex
    ON EMPLOYEE (Lname, Fname, Minit);

UNIQUE Keyword

- Used for specifying key constraint on indexing attributes
- Example
  - **CREATE UNIQUE INDEX** SsnIndex
    ON EMPLOYEE (Ssn);

CLUSTER Keyword

- Used when the index to be created should also sort the data file records on the indexing attributes
- Can improve query performance if a query requires join or ranged condition on the indexing attributes

CLUSTER Keyword

- Example

```
CREATE  INDEX DnoIndex
ON EMPLOYEE (Dno)
CLUSTER ;
```

Dropping Indexes

- Drop an index when it is not needed any more
- **DROP INDEX** <index name>;
- Example
  - **DROP INDEX** DnoIndex;

Summary

- Database update commands
  - INSERT
  - DELETE
  - UPDATE
- Views
- Indexes

References

1. Dayal, Umeshwar, and Philip A. Bernstein. "On the Updatability of Relational Views." *VLDB*. Vol. 78. 1978.
2. Keller, Arthur M. *Updates to relational databases through views involving joins.* IBM Research Division, 1981.
3. Langerak, Rom. "View updates in relational databases with an independent scheme." *ACM Transactions on Database Systems (TODS)* 15.1 (1990): 40-66.
4. Blakeley, José A., Neil Coburn, and Per Larson. "Updating derived relations: Detecting irrelevant and autonomously computable updates." *ACM Transactions on Database Systems (TODS)* 14.3 (1989): 369-400.

References

5. Negri, Mauro, Giuseppe Pelagatti, and Licia Sbattella. "Formal semantics of SQL queries." *ACM Transactions on Database Systems (TODS)* 16.3 (1991): 513-534.
6. Melton, Jim, and Alan R. Simon. *SQL: 1999: understanding relational language components*. Morgan Kaufmann, 2001.
7. Melton, Jim. *Advanced SQL: 1999: Understanding object-relational and other advanced features*. Morgan Kaufmann, 2002.

Have a nice day!