CSIT115 Data Management and Security

CSIT882 Data Management Systems

Advanced DDL and DML statements

Subject Coordinators: Dr Chen Chen, Dr Thanh Le

School of Computing and Information Technology -
University of Wollongong

Advanced DDL and DML statements

Outline

- `TRUNCATE TABLE` statement
- `CREATE TABLE` statement with subquery
- `INSERT` statement with subquery
- `DELETE` statement with subquery
- `UPDATE` statement with subquery

`TRUNCATE TABLE` statement

`TRUNCATE TABLE` statement permanently deletes all rows from a relational table

`TRUNCATE TABLE` statement is a Data Definition Language (DDL) statement and because of that it cannot be reversed (in the future we shall learn how to reverse DML statements)

`TRUNCATE TABLE` statement does not have `WHERE` clause and because of that it can only delete ALL rows from a relational table

`TRUNCATE TABLE` statement deletes all rows much faster than `DELETE` statement because a database system does not need to save rollback information

`TRUNCATE TABLE` statement returns unused persistent storage to a pool of free persistent storage while `DELETE` statement does not do that

In HTML view press 'p' to see the lecture notes

TRUNCATE TABLE statement

For example, delete all courses from a relational table COURSE

TRUNCATE TABLE statement that deletes all rows from a table COURSE

```
TRUNCATE TABLE COURSE;
```

In HTML view press 'p' to see the lecture notes

Advanced DDL and DML statements

Outline

- `TRUNCATE TABLE` statement
- `CREATE TABLE` statement with subquery
- `INSERT` statement with subquery
- `DELETE` statement with subquery
- `UPDATE` statement with subquery

CREATE TABLE statement with subquery

CREATE TABLE statement with subquery creates a relational table and saves in the table the results of a given SELECT statement

CREATE TABLE statement with subquery is Data Definition Language (DDL) statement and its actions cannot be reversed

A relational table created by CREATE TABLE statement with subquery does not have any consistency constraints enforced except NULL/NOT NULL constraint

For example, a table, that contains the names of departments together with the total number of courses offered by each department can be created and correct data can be inserted into it in the following way

CREATE TABLE statement with a subquery

```
CREATE TABLE DCNT AS
( SELECT name, COUNT(cnum) totc
  FROM DEPARTMENT LEFT OUTER JOIN COURSE
                  ON DEPARTMENT.name = COURSE.offered_by
  GROUP BY name );
```

CREATE TABLE statement with subquery

The names of columns in a new relational table are taken from the names of columns in SELECT clause or from alias names attached to the columns names or functions

A relational table DCNT created and loaded with data has two columns: name and totc

The consistency constraints can be enforced with ALTER TABLE statements

ALTER TABLE statements that enforce primary and foreign key constraints

```
ALTER TABLE DCNT ADD CONSTRAINT DCNT_pkey PRIMARY KEY(name);
ALTER TABLE DCNT ADD CONSTRAINT DCNT_fkey FOREIGN KEY (name)
                                          REFERENCES DEPARTMENT(name);
```

Advanced DDL and DML statements

Outline

`TRUNCATE TABLE` statement

`CREATE TABLE` statement with subquery

`INSERT` statement with subquery

`DELETE` statement with subquery

`UPDATE` statement with subquery

INSERT statement with subquery

INSERT statement with subquery inserts into a relational table the rows retrived by a given SELECT statement

INSERT statement with subquery is Data Manipluation Language (DML) statement an because of that its actions can be reversed with ROLLBACK statement

For example, to enforce the consistency constraints first, we create a relational table DCNT with ordinary CREATE TABLE statement

CREATE TABLE statement

```
CREATE TABLE DCNT(
 name          VARCHAR(50)  NOT NULL,
 total_courses DECIMAL(2)    NOT NULL,
  CONSTRAINT DCNT_pkey PRIMARY KEY(name),
  CONSTRAINT DCNT_fkey FOREIGN KEY (name) REFERENCES DEPARTMENT(name) );
```

INSERT statement with subquery

And then we use INSERT statement with subquery to load data into a relational table DCNT

INSERT statement with a subquery

```
INSERT INTO DCNT
( SELECT name, COUNT(cnum)
  FROM DEPARTMENT LEFT OUTER JOIN COURSE
                  ON DEPARTMENT.name = COURSE.offered_by
  GROUP BY name );
```

Note, that in this case there is no need for an alias name following COUNT(cnum)

Advanced DDL and DML statements

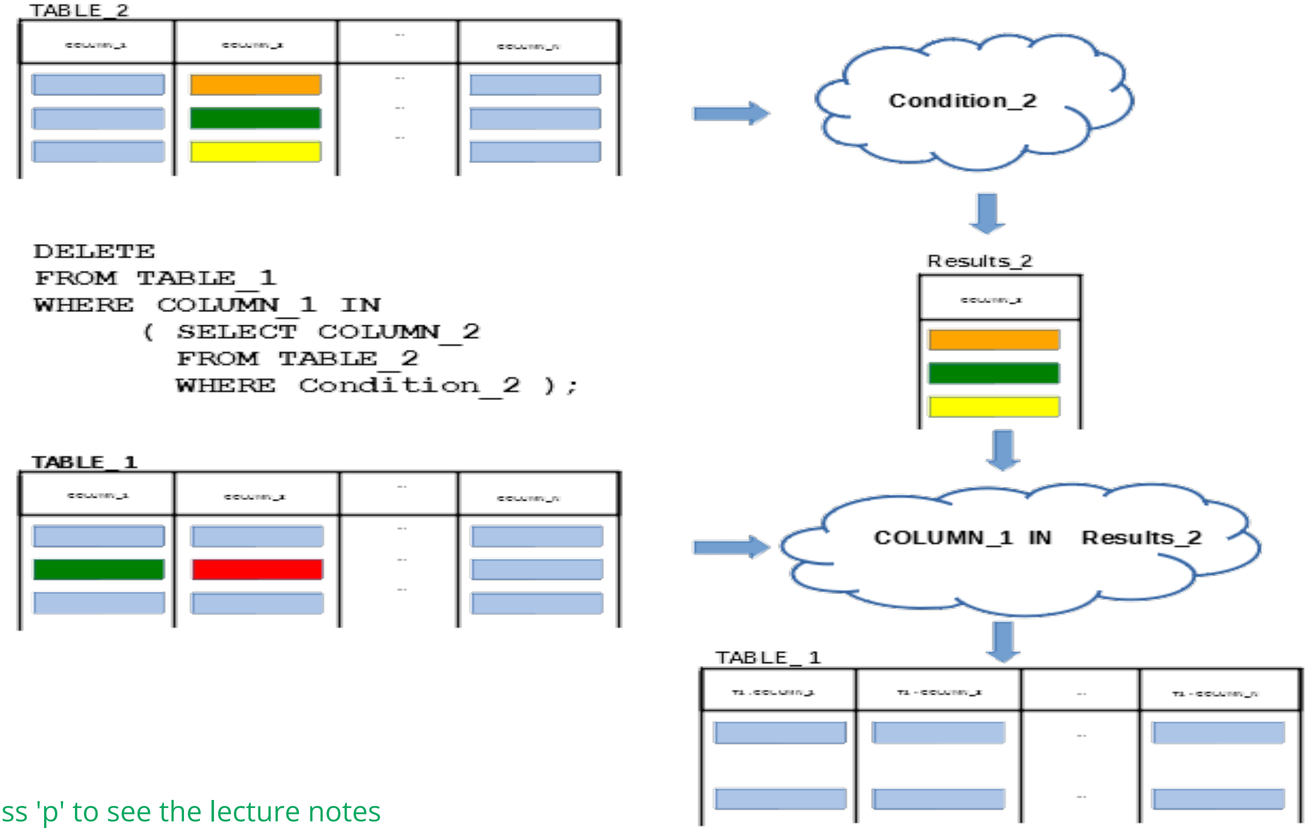
Outline

- `TRUNCATE TABLE` statement
- `CREATE TABLE` statement with subquery
- `INSERT` statement with subquery
- `DELETE` statement with subquery
- `UPDATE` statement with subquery

DELETE statement with subquery

DELETE statement with subquery deletes from a relational table all rows that satisfy WHERE condition

WHERE condition includes a subquery



```
DELETE
FROM TABLE_1
WHERE COLUMN_1 IN
      ( SELECT COLUMN_2
        FROM TABLE_2
        WHERE Condition_2 );
```

In HTML view press 'p' to see the lecture notes

DELETE statement with subquery

For example, we would like to delete all courses offered by a department chaired by Peter

DELETE statement with nested SELECT statement

```
DELETE FROM COURSE
WHERE offered_by = ( SELECT name
                     FROM DEPARTMENT
                     WHERE chair = 'Peter' );
```

For example, we would like to delete all departments that offer no courses

DELETE statement with negated existential quantifier EXISTS and nested SELECT statement

```
DELETE FROM DEPARTMENT
WHERE NOT EXISTS ( SELECT 'whatever'
                   FROM COURSE
                   WHERE COURSE.offered_by = DEPARTMENT.name );
```

Note, a reference to `DEPARTMENT.name` in `WHERE` condition

It is the same reference to a relational table `DEPARTMENT` as in a correleated nested query

Advanced DDL and DML statements

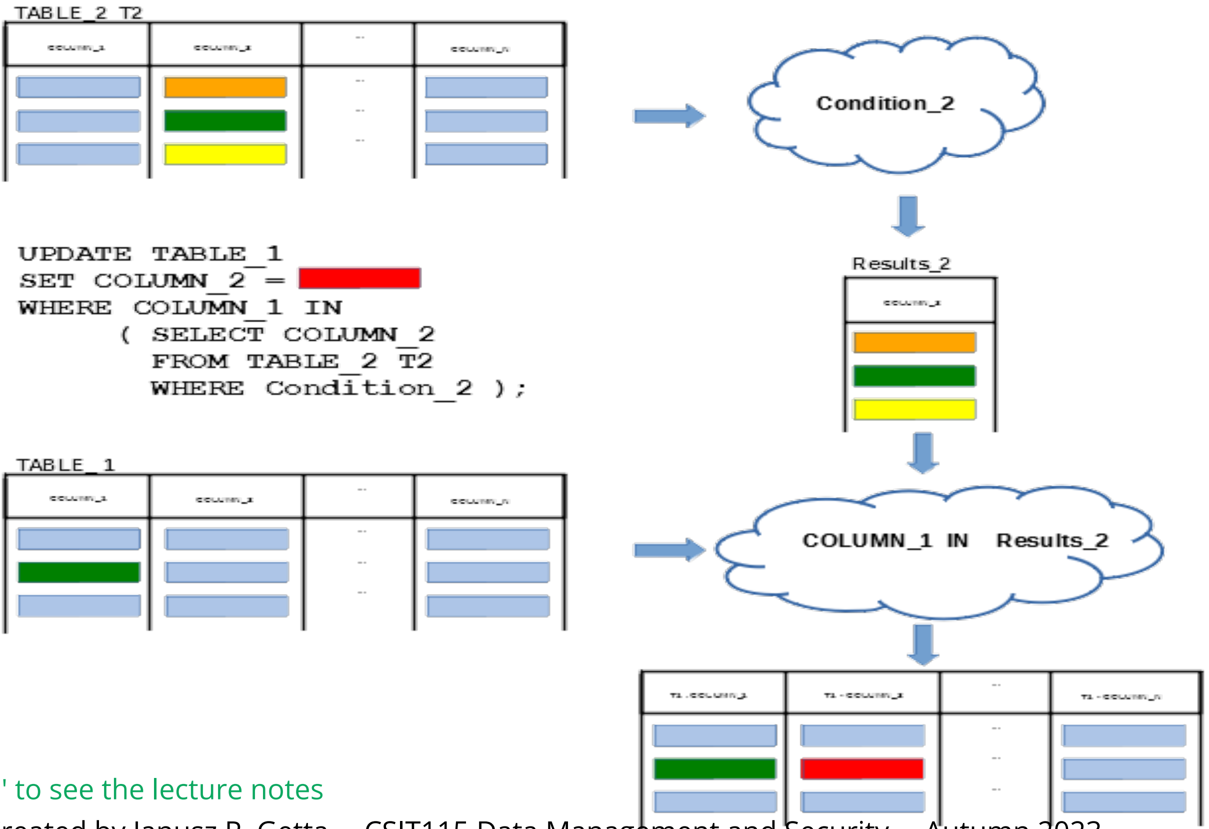
Outline

- `TRUNCATE TABLE` statement
- `CREATE TABLE` statement with subquery
- `INSERT` statement with subquery
- `DELETE` statement with subquery
- `UPDATE` statement with subquery

UPDATE statement with subquery

UPDATE statement with subquery updates in a relational table all rows that satisfy WHERE condition with the values determined in SET clause

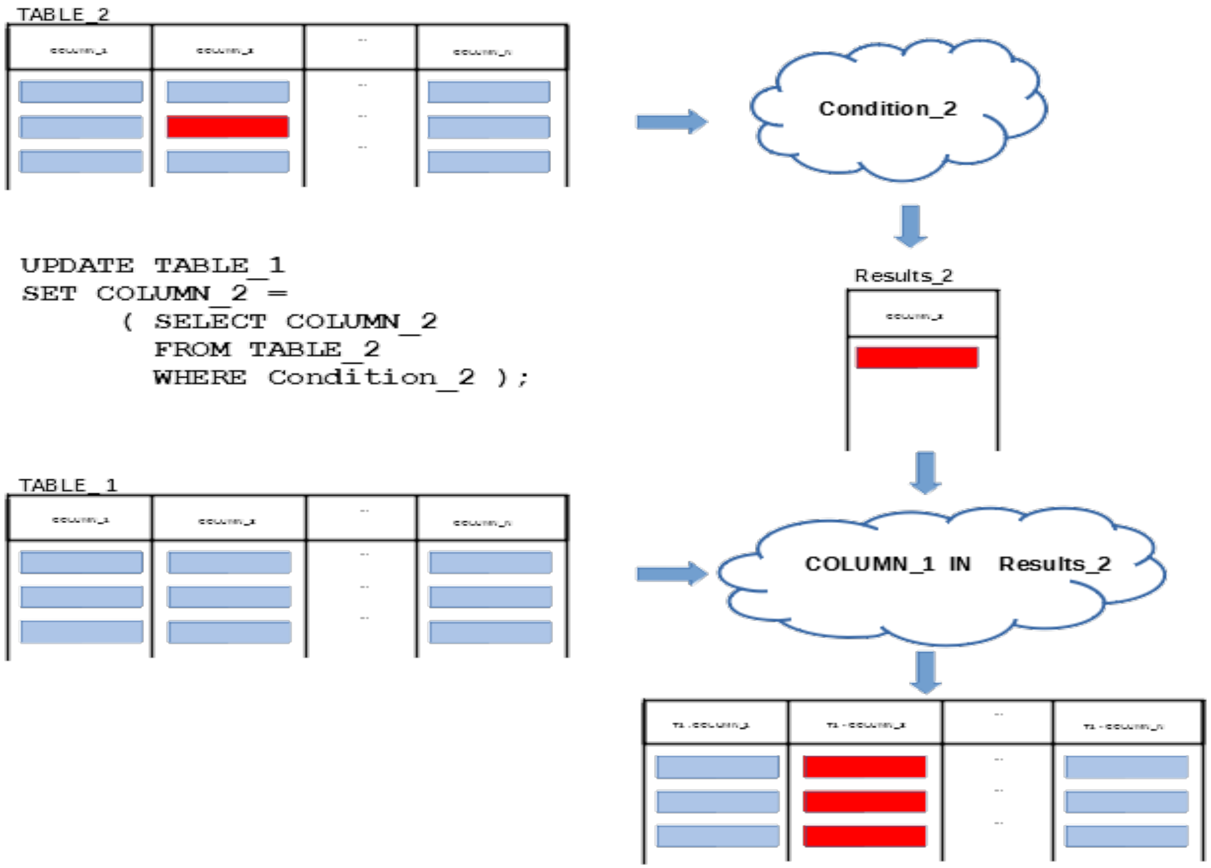
UPDATE statement with subquery can use a subquery in WHERE clause



```
UPDATE TABLE_1
SET COLUMN_2 = 🟥
WHERE COLUMN_1 IN
      ( SELECT COLUMN_2
        FROM TABLE_2 T2
        WHERE Condition_2 );
```

UPDATE statement with subquery

UPDATE statement with subquery can use a subquery in SET clause or in both WHERE and SET clauses



```
UPDATE TABLE_1
SET COLUMN_2 =
      ( SELECT COLUMN_2
        FROM TABLE_2
        WHERE Condition_2 );
```

In HTML view press 'p' to see the lecture notes

UPDATE statement with subquery

For example, we would like to increase the total number of staff members by 5 in all departments that offer more than 20 courses

UPDATE statement with nested SELECT statement

```
UPDATE DEPARTMENT
SET total_staff_number = total_staff_number + 5
WHERE name IN ( SELECT offered_by
                FROM COURSE
                GROUP BY offered_by
                HAVING COUNT(cnum) > 20 );
```

For example, we would like to add to table DEPARTMENT a column that contains the total number of courses offered by each department and insert the correct values into the column

ALTER TABLE statement that adds an attribute

```
ALTER TABLE DEPARTMENT ADD ( total_courses DECIMAL(2) );
```

UPDATE statement with nested SELECT statement

```
UPDATE DEPARTMENT
SET total_courses = ( SELECT COUNT(title)
                      FROM COURSE
                      WHERE COURSE.offered_by = DEPARTMENT.name );
```

In HTML view press 'p' to see the lecture notes

References

C. Coronel, S. Morris, A. Basta, M. Zgola, Data Management and Security, Chapters 5, 7, Cengage Compose eBook, 2018, eBook: Data Management and Security, 1st Edition

How to ... ? Cookbook, Recipe 7.2 How to perform advanced data manipulations ?