Advanced Structured Query Language (SQL) – Part 1

Lecture 8

Learning Outcomes

- In this chapter, students will learn:
  - What is procedural SQL
  - Why procedural SQL is important
  - How to create and use triggers

Procedural SQL

- Shortcomings of SQL
  - SQL doesn't support execution of a stored set of procedures based on some **logical** condition
    - IF <condition>
      THEN <perform procedure>
      ELSE <perform alternate procedure>
  - SQL fails to support the **looping** operations
    - DO WHILE
      <perform procedure>
      END DO

Procedural SQL

- Solutions:
  - SQL statements can be inserted within the procedural programming language
  - **Procedural SQL (PL/SQL)**
    - **Triggers**
    - **Stored Procedures**
    - **PL/SQL Functions (in DB2, it is known as user-defined function)**

Procedural SQL

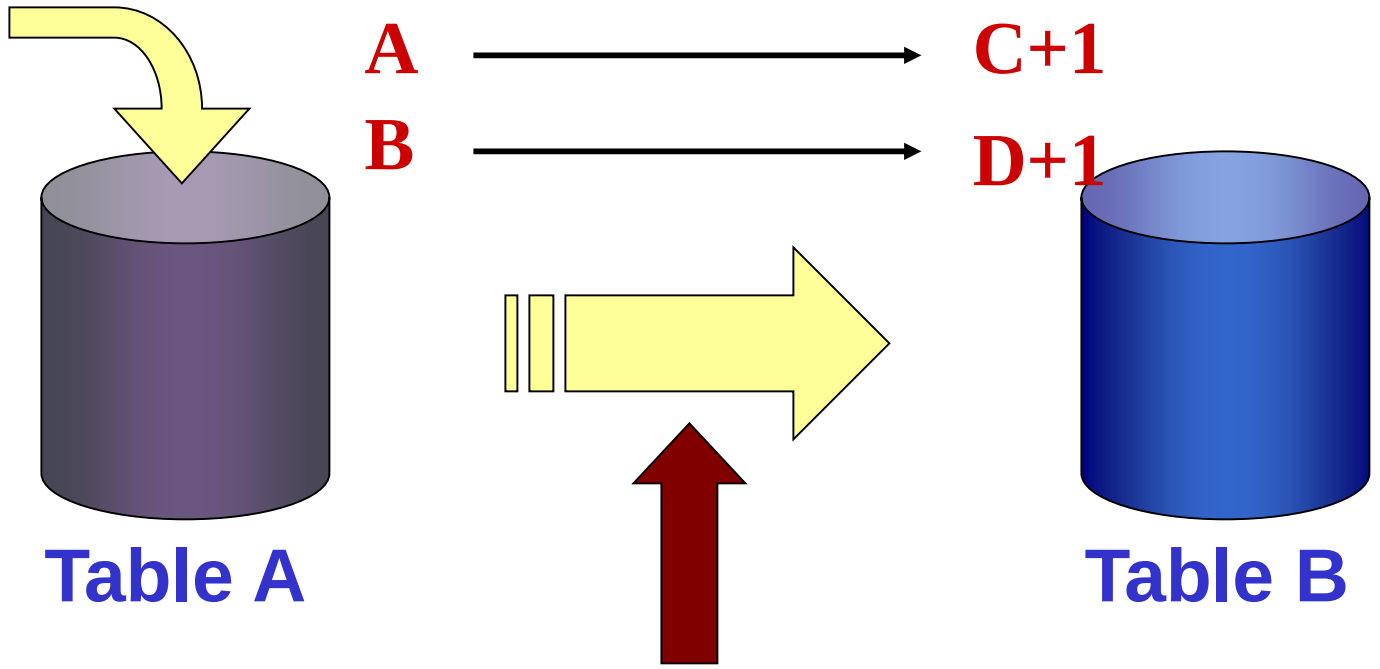
- **Procedural SQL (PL/SQL)**:
  - A language to store procedural code and SQL statements in database
  - Merge SQL and traditional programming constructs (e.g., IF-THEN-ELSE, FOR and WHILE loops, etc)
  - PL/SQL is executed by DBMS when invoked by the end user
    - Triggers
    - Stored procedures

SQL	PL/SQL
SQL is a single query that is used to perform DML and DDL operations.	PL/SQL is a block of codes that used to write the entire program blocks/ procedure/ function, etc.
It is declarative, that defines what needs to be done, rather than how things need to be done.	PL/SQL is procedural that defines how the things needs to be done.
Execute as a single statement.	Execute as a whole block.
Mainly used to manipulate data.	Mainly used to create an application.
Cannot contain PL/SQL code in it.	It is an extension of SQL, so it can contain SQL inside it.

Triggers

- A procedural SQL code that is **automatically** invoked by RDBMS on data manipulation event
- Example of use:
  - Automatically update the vendor when product inventory drops below its minimum quantity on hand
- Trigger is invoked **before/after** a data row is
  - **Inserted**
  - **Updated**
  - **Deleted**

Trigger - Insert Trigger



After data is inserted to Table A then do something to Table B

Trigger

- Role of triggers
  - To enforce constraints that cannot be enforced at the design and implementation levels.
  - Add functionality:
    - automating critical actions
    - providing appropriate warnings and suggestions
  - Can be used to update table values, insert records in tables, and call other stored procedures

Trigger Example

- Syntax:
  - **CREATE TRIGGER** <triggername>
    [BEFORE/AFTER] [DELETE/INSERT/UPDATE columnname]
    ON <tablename>
    FOR EACH ROW mode db2sql
    UPDATE <tablename>
    SET <conditions>

The PRODUCT Table

PRODUCT

	P_CODE	P_DESCRIPT	P_QOH	P_MIN	P_PRICE	P_REORDER
1	A0001	Book	8	5	12.67	0
2	A0002	Pencil	15	10	0.5	0
3	A0003	Ruler	18	12	0.8	0
4	A0004	Pen	15	8	0.3	0
5	A0005	Pen	23	5	1.2	0

Trigger Example

- Example:

```
CREATE TRIGGER trg_Product
AFTER INSERT ON Product
FOR EACH ROW mode db2sql
  UPDATE Product
  SET P_Reorder = 1
  WHERE P_QOH <= P_Min;
```

PRODUCT

	P_CODE	P_DESCRIPT	P_QOH	P_MIN	P_PRICE	P_REORDER
1	A0001	Book	8	5	12.67	0
2	A0002	Pencil	15	10	0.5	0
3	A0003	Ruler	18	12	0.8	0
4	A0004	Pen	15	8	0.3	0
5	A0005	Pen	23	5	1.2	0
6	B0002	Hammer	10	15	50	1

- To execute:
- **INSERT INTO** Product **VALUES**('B0002', Hammer', 10, 15, 50);

Trigger Example

- Example:
  - **CREATE TRIGGER** trg_Product
    AFTER UPDATE OF P_QOH **ON** Product
    FOR EACH ROW mode db2sql
    UPDATE Product
    SET P_Reorder = 1
    WHERE P_QOH <= P_Min;

PRODUCT

	P_CODE	P_DESCRIPT	P_QOH	P_MIN	P_PRICE	P_REORDER
1	A0001	Book	8	5	12.67	0
2	A0002	Pencil	5	10	0.5	1
3	A0003	Ruler	18	12	0.8	0
4	A0004	Pen	15	8	0.3	0
5	A0005	Pen	23	5	1.2	0
6	B0002	Hammer	10	15	50	1

- To execute:

UPDATE Product **SET** P_QOH = 5
WHEREP_CODE = ‘A0002';

Trigger Example

```
CREATE TRIGGER trg_Product
AFTER INSERT or UPDATE OF P_QOH ON Product
FOR EACH ROW mode db2sql
BEGIN
        IF INSERTING THEN
            UPDATE Product SET P_Reorder = 1
            WHERE P_QOH <= P_Min;
        ELSEIF UPDATING THEN
            UPDATE Product SET P_Reorder = 2
            WHERE P_QOH <= P_Min;
        END IF;
END@
```