IBM Software Group

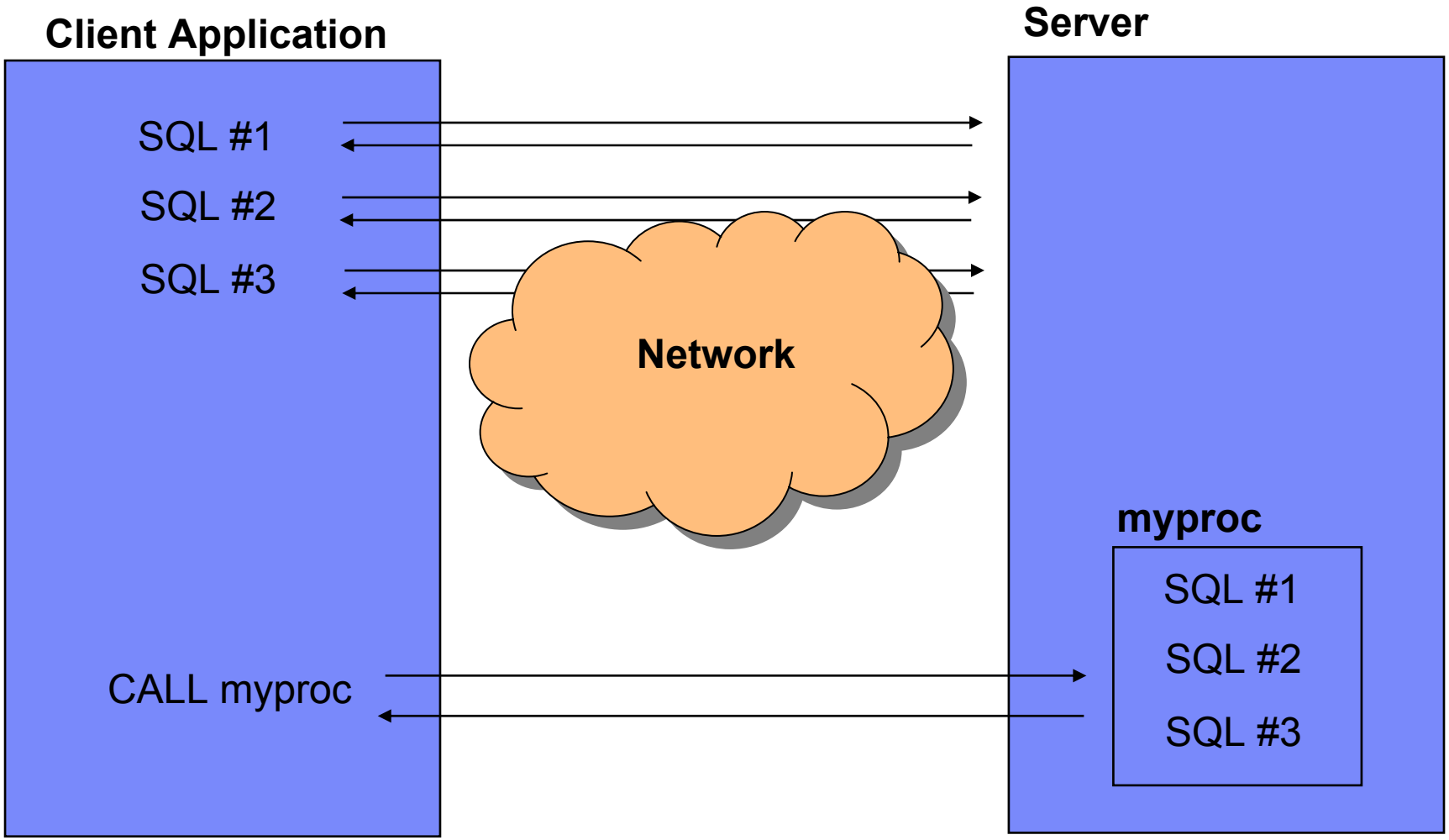
DB2 Express-C 9 overview course

SQL PL Stored Procedures, UDFs, Triggers

DB2 Information Management Software

ON DEMAND BUSINESS

Stored Procedures



Creating your first SQL PL Stored Procedure

- Using the Command Editor:

```
connect to sample
create procedure p1 begin end
```

- Using the DB2 Developer Workbench (Demo later)

Stored procedures IN, OUT and INOUT parameters

```
CREATE PROCEDURE P2 ( IN    v_p1 INT,
                      INOUT v_p2 INT,
                      OUT   v_p3 INT)
LANGUAGE SQL
SPECIFIC myP2
BEGIN
    -- my second SQL procedure
    SET v_p2 = v_p2 + v_p1;
    SET v_p3 = v_p1;
END
```

Stored procedure returning result sets

```
CREATE PROCEDURE set ()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE cur CURSOR WITH RETURN TO CLIENT
            FOR SELECT name, dept, job
            FROM staff
            WHERE salary > 20000;
  OPEN cur;
END
```

Calling Stored Procedures from a Java Application

```
try
{
  // Connect to sample database
  String url = "jdbc:db2:sample";
  con = DriverManager.getConnection(url);

  CallableStatement cs = con.prepareCall("CALL trunc_demo(?, ?)");

  // register the output parameters
  callStmt.registerOutParameter(1, Types.VARCHAR);
  callStmt.registerOutParameter(2, Types.VARCHAR);

  cs.execute();
  con.close();
}
catch (Exception e)
{
  /* exception handling logic goes here */
}
```

User-defined functions (UDFs)

Types of functions

- **Scalar functions**
  - Return a single value
  - Cannot change database state (i.e. no INSERT, UPDATE, DELETE statements allowed)
    - Example: COALESCE( ), SUBSTR( )
- **Table functions**
  - Return values in a table format
  - Called in the FROM clause of a query
  - Can change database state (i.e. allow INSERT, UPDATE, DELETE statements)
  - Example: SNAPSHOT_DYN_SQL( ), MQREADALL( )
- **Others type of functions (not covered in this course):**
  - Row functions
  - Column functions

Scalar function

- Scalar functions take input values and return a single value

- Example:

  if your application uses Oracle's NVL() function widely, it may be beneficial to simply map NVL( ) to DB2's COALESCE( ) function

```
CREATE FUNCTION NVL (p_var1 VARCHAR(30),
                     p_var2 VARCHAR(30))
SPECIFIC nvlvarchar30
RETURNS VARCHAR(30)
RETURN COALESCE(p_var1, p_var2)
```

Invoking Scalar UDFs

- Scalar UDFs can be invoked in SQL statements wherever a scalar value is expected, or in a VALUES clause

```
SELECT DEPTNAME, COALESCE(MGRNO, 'ABSENT')
  FROM DEPARTMENT

VALUES COALESCE('A','B')
```

Table UDFs

- Return a table of rows
- Used in the FROM clause of a query
- Similar to a view, except more powerful because data modification statements (INSERT/UPDATE/DELETE) can be performed
- Typically used to return a table and keep an audit record

Table function example

- A function which enumerates a set of employees of a department

```
CREATE FUNCTION getEnumEmployee(p_dept VARCHAR(3))
RETURNS TABLE
     (empno CHAR(6),
      lastname VARCHAR(15),
      firstnme VARCHAR(12))
SPECIFIC getEnumEmployee
RETURN
  SELECT e.empno, e.lastname, e.firstnme
    FROM employee e
    WHERE e.workdept=p_dept
```

Calling a Table function

- Used in the FROM clause of an SQL statement
- The TABLE() function must be applied and must be aliased.

```
SELECT * FROM
    TABLE (getEnumEmployee('E01')) T
```

DB2 Developer Workbench DEMO

Start → Programs → IBM DWB → IBM DB2 Developer Workbench 9.1 → Developer Workbench

Triggers

Types of Triggers

- **BEFORE**
  - activation before row is inserted, updated or deleted
  - Operations performed by this trigger cannot activate other triggers (i.e. INSERT, UPDATE, and DELETE operations are not permitted)
- **AFTER**
  - Activated after the triggering SQL statement has executed to successful completion
  - May activate other triggers (cascading permitted up to 16 levels)
- **INSTEAD OF**
  - Defined on views
  - Logic defined in the trigger is executed *instead of* the triggering SQL statement

A Simple BEFORE Trigger

if no value provided on insert, column is NULL

define qualifier for new values

```
CREATE TRIGGER default_class_end
NO CASCADE BEFORE INSERT ON cl_sched
REFERENCING NEW AS n
FOR EACH ROW
MODE DB2SQL
WHEN (n.ending IS NULL)
     SET n.ending = n.starting + 1 HOUR
```

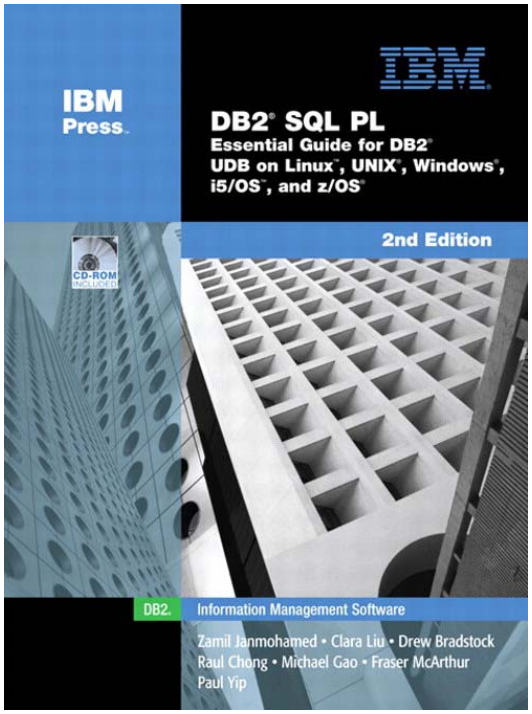
optional WHEN

A Simple AFTER Trigger

- Similar to BEFORE triggers, except that INSERT, UPDATE and DELETE are supported

```
CREATE TRIGGER audit_emp_sal
AFTER UPDATE OF salary ON employee
REFERENCING OLD AS o NEW AS n
FOR EACH ROW
MODE DB2SQL
  INSERT INTO audit VALUES (
  CURRENT TIMESTAMP, ' Employee ' || o.empno ||
    ' salary changed from ' || CHAR(o.salary) ||
    ' to ' || CHAR(n.salary) || ' by ' || USER)
```

Recommended Book



- DB2® SQL PL: Essential Guide for DB2® UDB on Linux™, UNIX®, Windows™, i5/OS™, and z/OS®, 2nd Edition
- More info:

  http://www.software.ibm.com/data/developer/sqlplbook

Order from any online book seller using

ISBN: 0-13-100772-6