Lecture Notes for Lecture 8 of CS 5200 (Database Management System) for the Summer 1, 2019 session at the Northeastern University Silicon Valley Campus.

Stored Procedures and Functions

Philip Gust,
Clinical Instructor
Department of Computer Science

Lecture 7 Review

- In this lecture we will discussed conditional mechanisms for automating operations in a database:
- Foreign key constraints enable the database to perform simple actions as table rows are added, altered and removed.
- Triggers enable the database to perform more general actions to ensure that constraints are maintained as table rows are added, altered and removed.
- Stored procedures and functions allow new operations to be defined that can be invoked during a query or as part of a triggered event.

Stored Procedures and Functions

Today's Lecture

- We touched briefly in the the previous lecture on the concept of stored procedures, and briefly introduced their purpose.
- In today's lecture, we will take an in-depth look at the mechanism of stored procedures, and the JDBC bindings that enable stored procedures to be defined.
- We will work through examples of stored procedures in JavaDB as Java functions, how data is passed between SQL and Java functions, and how SQL can be executed within the functions.
- We will also discuss how stored procedures are implemented in other DBMS products and look at an example using MySQL.

Stored Procedures and Functions

- A stored procedure or function performs a particular task, and encapsulates a set of operations or queries to execute on a database server.
- For example, operations on an employee database (hire, fire, promote, lookup) could be coded as stored procedures and functions executed by application code.
- Stored procedures and functions can be compiled and executed with different parameters and results, and they can have any combination of input, output, and input/output parameters.

Stored Procedures and Functions

- Stored procedures and functions are supported by most DBMSs, but there is a fair amount of variation in their syntax and capabilities
- A stored procedure is invoked using a special CALL statement, and can only return values through output parameters
- A stored function can be invoked anywhere that a SQL expression can appear, and returns a value as its function result.

Stored Procedures and Functions

- This table shows places that stored procedures and functions can be used within SQL. A stored function cannot be invoked using CALL and a stored procedure cannot be invoked as an expression using VALUES.

Feature	**Procedure**	**Function**
Execute in a trigger	yes	yes
Return result set(s)	yes	no
Process OUT / INOUT Params	yes	no
Execute SQL select	yes	yes
Execute SQL update/insert/delete	yes	no
Execute DDL (create/drop)	yes	no
Execute in a SQL expression	no	yes

Stored Procedures and Functions

Stored Procedures

- The CREATE PROCEDURE statement in Derby allows creating a Java stored procedure, which can then be call using the CALL PROCEDURE statement.
  - Syntax

    CREATE PROCEDURE procedure-Name(

    [ProcedureParameter [, ProcedureParameter]] *

)

    [ProcedureElement] *
- Dropping a procedure follows the form for other resources
  - Syntax

    DROP PROCEDURE procedure-Name

Stored Procedures and Functions

Stored Procedures

- Example:

```
CREATE PROCEDURE sendEmail(
        IN recipient varchar(64),
        IN sender varchar(64),
        OUT status boolean
)
PARAMETER STYLE JAVA
LANGUAGE JAVA
DETERMINISTIC
NO_SQL
EXTERNAL NAME
        ‘CustomerDemo. sendEmail'
```

Stored Procedures and Functions

Stored Procedures

- Example:

```
public class CustomerDemo {
    …

    /**
     * Send mail to recipient from sender
     * @param recipient the recipient email address
     * @param sender the sender email address
     * @param status the return status
     */
    public static void sendEmail(String recipient, String sender, boolean status[]) {
        System.out.printf("sendEmail: To: %s  From: %s\n", recipient, sender);
        status[0] = true;
    }
    …
}
```

Stored Procedures and Functions

Stored Procedures

- Example:

```
// prepared statement for calling sendEmail procedure with 3 params
CallableStatement cstmt = conn.prepareCall("call sendEmail(?,?,?)");
cstmt.setString(1,  “the_receiver@google.com”);     // sender IN param
cstmt.setString(2, "the_sender@yahoo.com");         // receiver IN param
cstmt.registerOutParameter(3, Types.BOOLEAN);     // status OUT param
cstmt.execute();

boolean result = cstmt.getBoolean(3);       // pick up result value
System.out.printf("status: %b\n", result);
cstmt.close();  // free resources when done
```

Stored Procedures and Functions

Stored Procedures

- Details of stored procedure name and parameters
  - procedure-Name

    [schemaName.] SQL92Identifier
  - If schema-Name is not provided, the current schema is the default schema. If a qualified procedure name is specified, the schema name cannot begin with SYS.
  - ProcedureParameter

    [{ IN | OUT | INOUT }] [parameter-Name] DataType

    Note: The default value for a parameter is IN, and the parameter-Name must be unique within a procedure.

Stored Procedures and Functions

Stored Procedures

- Details of stored procedure PrecedureElement
  - ProcedureElement

    {

    | [DYNAMIC] RESULT SETS INTEGER

    | LANGUAGE { JAVA }

    | EXTERNAL NAME string

    | PARAMETER STYLE JAVA

    | { NO SQL | MODIFIES SQL DATA | CONTAINS SQL | READS SQL DATA }

    }
  - The procedure elements may appear in any order, but each type of element can only appear once. A procedure definition must contain these elements:
    - LANGUAGE
    - PARAMETER STYLE
    - EXTERNAL NAME

Stored Procedures and Functions

Stored Procedures

- Details of stored procedure PrecedureElement
  - Dynamic Result Sets integer

    Indicates the estimated upper bound of returned result sets for the procedure. Default is no (zero) dynamic result sets.
  - Language

    JAVA- the database manager will call the procedure as a public static method in a Java class.
  - EXTERNAL NAME

    string describes the Java method to be called when the procedure is executed, and takes the following form:

    class_name.method_name

    The External Name cannot have any extraneous spaces.

Stored Procedures and Functions

Stored Procedures

- Details of stored procedure PrecedureElement
  - PARAMETER STYLE

    JAVA - The procedure will use a parameter-passing convention that conforms to the Java language and SQL Routines specification.

    - INOUT and OUT parameters will be passed as single entry arrays to facilitate returning values.
    - Result sets are returned through additional parameters to the Java method of type java.sql.ResultSet [] that are passed single entry arrays.

    Derby does not support long column types (for example Long Varchar, BLOB, and so on). An error will occur if you try to use one of these long column types.

Stored Procedures and Functions

Stored Procedures

- Details of stored procedure PrecedureElement
  - NO SQL, CONTAINS SQL, READS SQL DATA, MODIFIES SQL DATA

    Indicates whether the stored procedure issues any SQL statements and, if so, what type.
    - **NO SQL**: Indicates that the stored procedure cannot execute any SQL statements
    - **CONTAINS SQL**: Indicates that SQL statements that neither read nor modify SQL data can be executed by the stored procedure. Statements that are not supported in any stored procedure return a different error.
    - **READS SQL DATA**: Indicates that some SQL statements that do not modify SQL data can be included in the stored procedure. Statements that are not supported in any stored procedure return a different error.
    - **MODIFIES SQL DATA**: Indicates that the stored procedure can execute any SQL statement except statements that are not supported in stored procedures. MODIFIES SQL DATA is the default value.

Stored Procedures and Functions

Stored Functions

- The CREATE FUNCTION statement in Derby allows you to create Java stored functions, which you can then call using the normal function call syntax.
- Syntax

  CREATE FUNCTION function-Name(

  [FunctionParameter [, FuctionParameter]] *

) RETURNS ReturnDataType

  [FunctionElement] *
- Dropping a funciton follows the form for other resources
  - Syntax

    DROP FUNCTION function-Name

Stored Procedures and Functions

Stored Functions

- Example:

```
CREATE FUNCTION isEmail(
        email VARCHAR(64)
)  RETURNS BOOLEAN
PARAMETER STYLE JAVA
LANGUAGE JAVA
DETERMINISTIC
NO_SQL
EXTERNAL NAME
        ‘CustomerDemo. isEmail'
```

Stored Procedures and Functions

Stored Functions

- Example:

```
public class CustomerDemo {
    …

    /**
     * Determines whether 'email' is a valid email address.
     *
     * @param email the email address
     * @return true if 'email' is a valid email address
     */
    public static boolean isEmail(String email) {
        return email.matches(
            "^[\\p{L}\\p{N}\\._%+-]+@[\\p{L}\\p{N}\\.\\-]+\\.[\\p{L}]{2,}$");
    }
    …
}
```

Stored Procedures and Functions

- Example:

```
// prepared statement for calling isEmail procedure with 1 param
PreparedStatement invoke_isEmail =
            conn.prepareStatement("values ( isEmail(?) )”);

// validate sender email address
String sender = "the_sender@yahoo.com";
invoke_isEmail.setString(1, sender);
ResultSet rs = invoke_isEmail.executeQuery();
if (rs.next()) {
      boolean status = rs.getBoolean(1);
      System.out.printf("sender: %s isEmail: %b\n", sender, status);
}
invoke_isEmail.close();
rs.close();
```

Stored Procedures and Functions

Stored Functions

- Example:

```
CREATE TABLE Customer (
        Id INT NOT NULL,"
        FamilyName VARCHAR(32) NOT NULL,
        GivenName VARCHAR(32) NOT NULL,
        Email VARCHAR(64) NOT NULL,
        PRIMARY KEY (Id)
        CHECK (isEmail(Email))
);
```

Stored Procedures and Functions

Stored Functions

- Details of stored function name and parameters
  - function-Name

    [schemaName.] SQL92Identifier

    If schema-Name is not provided, the current schema is the default schema. If a qualified procedure name is specified, the schema name cannot begin with SYS.
  - FunctionParameter

    [parameter-Name] DataType

    Note: Function parameters are always input only, and the parameter-Name must be unique within a function.

Stored Procedures and Functions

Stored Functions

- Details of stored function FunctionElement
  - FunctionElement

```
{
| LANGUAGE JAVA
|   { DETERMINISTIC | NOT DETERMINISTIC }
|   EXTERNAL NAME string
|   PARAMETER STYLE { JAVA | DERBY_JDBC_RESULT_SET | DERBY }
|   EXTERNAL SECURITY { DEFINER | INVOKER }
|   { NO SQL | CONTAINS SQL | READS SQL DATA }
|   { RETURNS NULL ON NULL INPUT | CALLED ON NULL INPUT }
}
```

    Function elements are similar to those for a ProcedureElement.

Stored Procedures and Functions

Stored Functions

- Details of stored function FunctionElement
  - DETERMINISTIC, NOT DETERMINISTIC
    - **DETERMINISTIC** declares that the function is deterministic, meaning that with the same set of input values, it always computes the same result.
    - The default is **NOT DETERMINISTIC**. Derby cannot recognize whether an operation is actually deterministic, so you must take care to specify this element correctly.

Stored Procedures and Functions

Stored Functions

- Details of stored function FunctionElement
  - PARAMETER STYLE
    - **JAVA**: The function will use a parameter-passing convention that conforms to the Java language and SQL Routines specification. INOUT and OUT parameters will be passed as single entry arrays to facilitate returning values.
    - **DERBY_JDBC_RESULT_SET**: The PARAMETER STYLE is DERBY_JDBC_RESULT_SET if and only if this is a Derby-style table function, that is, a function which returns *tableType* and which is mapped to a method which returns a JDBC *ResultSet*.
    - **DERBY**: The PARAMETER STYLE must be DERBY if and only if an ellipsis (...) appears at the end of the argument list.

Stored Procedures and Functions

Stored Functions

- Stored Functions for the publication database
  - Several stored functions will be useful for our publication database, including:
    - issnToString()
    - parseIssn()
    - isIssn()
    - orcidToString()
    - parseOrcid()
    - isOrcId()
    - isDoi()

Stored Procedures and Functions

Stored Functions

- Stored Functions for the publication database
  - The isIssn() function can be used to check the ISSN parameter when inserting a new Publisher

```
create table Journal {
    …
    ISSN int not null check (isIssn(ISSN)),
    …
}
```

  - The issnToString() function can be used in a query to return the ISSN string, rather than their internal int representations.

```
select Title, issnToString(ISSN) from Journal
```

Stored Procedures and Functions

Stored JAR Files in the Database

- Derby allows storing Java JAR files for stored functions and procedures in the database.
  - Loading JAR file ‘Biblio.jar’ into database:

```
CALL SQLJ.install_jar
        (Biblio.jar', 'APP.BiblioJar', 0)
```

  - Setting database class path to include stored JAR file:

```
CALL SYSCS_UTIL.SYSCS_SET_DATABASE_PROPERTY
        ('derby.database.classpath', 'APP.BiblioJar')
```

Stored Procedures and Functions

Stored Procedures and Functions in MySQL

- MySQL also supports defining and using stored procedures and functions, using a proprietary procedural programming language.
- Here is how to define a procedure cleanupJournals() to removes any journals whose ISSN field is negative (i.e between than 8000-0000 and 9999-999X)

```
CREATE PROCEDURE cleanupJournals()
BEGIN
  DELETE FROM Journal where ISSN < 0;
END
```

- The stored procedure can be invoked as

```
CALL cleanupJournals()
```

Stored Procedures and Functions

Stored Procedures and Functions in MySQL

- Here is an example of a stored function that calculates profits based on cost and price for a product database.

```
CREATE FUNCTION calcProfit(cost FLOAT, price FLOAT) RETURNS DECIMAL(9,2)
BEGIN
  DECLARE profit DECIMAL(9,2);
  IF price > cost THEN
        SET profit = price - cost;
  ELSE
        SET profit = 0;
  END IF;
  RETURN profit;
END
```

- The stored function can be invoked as

```
SELECT *, calcProfit(prod_cost,prod_price) AS profit FROM products;
```

Stored Procedures and Functions

Stored Procedures and Functions in MySQL

- When entering a stored procedure or function from the *mysql* shell, it is necessary to delimit the definition to prevent MySQL from processing the definition too soon.

```
DELIMITER $$

CREATE PROCEDURE cleanupJournals()
BEGIN
  DELETE FROM Journal where ISSN < 0;
END$$

DELIMITER ;
```

- The DELIMITER command at the end of these statements returns processing to normal.

Stored Procedures and Functions

Stored Procedures and Functions in MySQL

- Challenge:
  - Create a set of stored procedures and function that provide equivalent functionality to the Java functions in Biblio.java
  - Note that MySQL provides extended regular expression processing in addition to the standard SQL string matching operations.