AG Heterogene Informationssysteme
Prof . Dr.-Ing. Stefan Deßloch
Fachbereich Informatik
Technische Universität Kaiserslautern

Recent Developments for Data Models – Solution to Exercise 3

Monday, June 4, 2012 – 15:30 to 17:00 – Room 36-336

1) SQL-invoked Routines Characteristics

SQL-invoked routines fall into three principle classes: procedures, functions and methods. Fill in the following table to summarize the more important differences!

	Procedures	Functions	Methods
Routine invocation	*CALL statement*	*Functional notation*	*Dot notation*
Associated with specif-ic type?	*No*	*No*	*Yes*
Schema of residence	*Any schema*	*Any schema*	*Schema of associated user-defined type*
Routine resolution	*Fully resolved at compile time.*	*Fully resolved at compile time.*	*Compilations resolves to set of candidate methods; final resolu-tion at runtime.*
Input parameters?	*Yes*	*Yes*	*Yes; additional implicit subject parameter (in-stance of the associated user-defined structured type)*
Output parameters?	*Yes. Result can also be returned in dy-namic result sets.*	*Only as return value of function.*	*Only as return value of method.*

2) SQL-invoked Routines

Reconsider the user-defined structured type hierarchy introduced by the first exercise sheet. Specify SQL-invoked routines for the following purposes.

a. A user-defined constructor method for the type `PublicationT` that takes the title, the URL, and the publication date as input parameters.

```
ALTER TYPE PublicationT
ADD CONSTRUCTOR METHOD PublicationT(title varchar(150),
    url varchar(150), pdate date)
RETURNS PublicationT


CREATE CONSTRUCTOR METHOD PublicationT(title varchar(150),
    url varchar(150), pdate date)
RETURNS PublicationT
FOR PublicationT
BEGIN
    SET self.title = title;
    SET self.url = url;
    SET self.pdate = pdate;
    RETURN self;
END
```

b. A method called `citeStr` for the type `PublicationT` that returns a character string of the following format: <title>, <year of publication>

```
ALTER TYPE PublicationT
ADD METHOD citeStr() RETURNS varchar(255);


CREATE METHOD citeStr()
RETURNS varchar(255)
FOR publicationT
RETURN self.title || ', ' ||
       CAST(year(self.pdate) AS CHAR(4));
```

c. A method called `citeStr` for the type `ContainedPublT` that overrides the method of its supertype and returns a character string of the following format: <title>, <first_page> - <last_page>, <year of publication>

```
ALTER TYPE ContainedPublT
ADD OVERRIDING METHOD citeStr() RETURNS varchar(255);


CREATE METHOD citeStr()
RETURNS varchar(255)
FOR ContainedPublT
RETURN self.title || ', ' ||
       TRIM(CHAR(self.first_page)) || ' - ' ||
       TRIM(CHAR(self.last_page)) || ', ' ||
       CAST(year(self.pdate) AS CHAR(4));
```

d. A function called `citeStr` that takes a parameter of the type `PublicationT` and returns a character string of the format described in b.

```
CREATE FUNCTION citeStr(publication publicationT)
RETURNS varchar(255)
RETURN publication.title || ', ' ||
       CAST(year(publication.pdate) AS CHAR(4));
```

e. A function called `citeStr` that takes a parameter of the type `Contained-PublT` and returns a character string of the format described in c.

```
CREATE FUNCTION citeStr(publication ContainedPublT)
RETURNS varchar(255)
RETURN publication.title || ', ' ||
       TRIM(CHAR(publication.first_page)) || ' - ' ||
       TRIM(CHAR(publication.last_page)) || ', ' ||
       CAST(year(publication.pdate) AS CHAR(4));
```

f. A function called `publishedBetween` that takes two `Date` parameters and returns those publications that have been published in the indicated time range.

```
CREATE FUNCTION publishedBetween(start date, end date)
RETURNS TABLE(publkey REF(PublicationT), title varchar(150),
              url varchar(150), pdate date)

RETURN SELECT publkey, title, url, pdate
       FROM Publication
       WHERE pdate >= start AND pdate <= end;
```

g. A procedure called `addAuthor` to add an author to a publication. The procedure shall take takes three input parameters of type `AuthorT`, `PublicationT`, and `Integer` to indicate the position of the author in the list.

```
CREATE PROCEDURE addAuthor(IN newAuid integer, IN newPublKey
                 varchar(20), IN newPosition integer)
MODIFIES SQL DATA
LANGUAGE SQL
BEGIN

    UPDATE publauthors SET position = position + 1
    WHERE publication = PublicationT(newPublKey)
      AND position >= newPosition;

    INSERT INTO publauthors
        VALUES (PublicationT(newPublKey), AuthorT(newAuid),
                newPosition);

END;
```

h. A procedure called `publAuthorsKeywords` that takes a publication key as parameter and returns the associated authors and keywords (in two result sets).

```
CREATE PROCEDURE publAuthorsKeywords(IN lkey varchar(20))
DYNAMIC RESULT SETS 2
LANGUAGE SQL
READS SQL DATA
BEGIN ATOMIC

    DECLARE acurs CURSOR WITH RETURN TO CLIENT FOR
        SELECT author->name, author->firstName
        FROM publauthors
        WHERE publication = PublicationT(lkey)
        ORDER BY position;

    DECLARE kcurs CURSOR WITH RETURN TO CLIENT FOR
        SELECT DISTINCT keyw->topic
        FROM publkeywords
        WHERE publication = PublicationT(lkey);

    OPEN acurs;

    OPEN kcurs;

END
```

3) Subject Routine Determination

Function overloading allows creating multiple functions with equal names. After a function invocation, the database manager must decide which of the equally named functions “fits best”. This function is referred to as *subject function*.

Assume that the function `func` is invoked with three parameters of the types `ArticleT`, `INTEGER`, and `CHAR(50)`. Further assume that the current session `PATH` is set to `'A, B'` and that execute privileges are granted for all functions.

a. Consider the following function signatures.

```
1  FUNCTION A.foo (ArticleT, SMALLINT, CHAR(50))
2  FUNCTION A.func(BookT, INTEGER, VARCHAR(50))
3  FUNCTION A.func(PublicationT, REAL, INTEGER)
4  FUNCTION C.func(ArticleT, INTEGER, CHAR(50))
5  FUNCTION B.func(ContainedPublT, INTEGER)
6  FUNCTION A.func(ArticleT, DECIMAL(5, 2), VARCHAR(50))
7  FUNCTION A.func(ContainedPublT, INTEGER, CHAR(50))
8  FUNCTION B.func(ArticleT, DECIMAL(5,2), DATE)
9  FUNCTION B.func(PublicationT, INTEGER, CHAR(50))
10 FUNCTION B.func(ArticleT, DOUBLE, CLOB)
```

Which steps are performed to determine the subject function? Which functions are eliminated in each of these steps? Which function is eventually determined to be the subject function?

In the first step functions 1, 4, and 5 are eliminated. Function 1 is not named func, function 4 is neither in schema A nor B, and function 5 has not the required number of parameters.

In the second step functions 2, 3, and 8 are eliminated. The type of the first parameter of function 2 (`BookT`) is not in the type precedence list of `ArticleT`. The type of the third parameter of function 3 (`INTEGER`) is not in the type precedence list of `CHAR`. The type of the third parameter of function 8 (`DATE`) is not in the type precedence list of `CHAR`.

In the third step the types of the parameters of the remaining functions are examined left-to-right. Looking at the first parameter, functions 7 and 9 are eliminated, because `ContainedPublT` and `PublicationT` appear later in the type precedence list of `ArticleT` than `ArticleT` itself. The remaining functions are thus 6 and 10. Looking at the second parameter, function 10 is eliminated, because `DOUBLE` appears later in the type precedence list of `INTEGER` than `DECIMAL`. At this point only function 6 remains and is thus selected as the subject function.

b. Consider the following function signatures.

```
1  FUNCTION A.func(AuthorT, REAL, CHAR(50))
2  FUNCTION A.func(ArticleT, DECIMAL(5, 2), CHAR(50))
3  FUNCTION B.func(ArticleT, FLOAT, VARCHAR(50))
4  FUNCTION A.func(ArticleT, INTEGER, VARCHAR(50))
5  FUNCTION A.func(TechReportT, INTEGER)
6  FUNCTION B.func(ArticleT, INTEGER, CLOB)
7  FUNCTION B.func(ArticleT, SMALLINT, CHAR(50))
8  FUNCTION A.func(ContainedPublT, INTEGER, INTEGER)
9  FUNCTION B.func(ArticleT, INTEGER, VARCHAR(50))
10 FUNCTION A.func(PublicationT, INTEGER, CHAR(50))
```

Which function is determined to be the subject function?

In the first step function 5 is eliminated, because it does not have the required number of parameters.

In the second step functions 1,7 and 8 are eliminated. The type of the first parameter of function 1 (`AuthorT`) is not in the type precedence list of `ArticleT`. The type of the second parameter of function 7 (`SMALLINT`) is not in the type precedence list of `INTEGER`. The type of the third parameter of function 8 (`INTEGER`) is not in the type precedence list of `CHAR`.

In the third step the types of the parameters of the remaining functions are examined left-to-right. Looking at the first parameter, functions 10 is eliminated, because `PublicationT` appears later in the type precedence list of `ArticleT` than `ArticleT` itself. Looking at the second parameter, functions 2 and 3 are eliminated, because `DECIMAL` and `FLOAT` appear later in the type precedence list of `INTEGER` than `INTEGER` itself. *Looking at the third parameter, functions 6 is eliminated, because `CLOB` appears later in the type precedence list of `CHAR` than `VARCHAR`. At this point the functions 4 and 9 remain. Because schema A appears first in the current path, function 4 is selected as the subject function.*

c. What are the major differences between subject *function* resolution and subject *method* resolution?

Subject function resolution and subject method resolution are preformed in a similar way with one important exception. If multiple methods qualify as subject methods, the current path is NOT considered to make a final choice. In fact, the decision which method is to be invoked is made at runtime based on the most-specific type of the instance value.

4) External SQL-invoked Routines

SQL-invoked routines can either be written in SQL or in any of several general-purpose programming languages, such as C or Java. These routines are referred to as *external* routines. Discuss the advantages and drawbacks of external routines as compared to SQL routines!

Advantages:

- *You may already own routines written in a host language that were created for a different purpose.*
- *External routines may perform computationally intensive tasks more efficiently than SQL routines.*
- *External routines provide the ability to invoke identical program code within the database and in other parts of your application.*
- *External routines are usually able to access services provided by the underlying operating system.*

Disadvantages:

- *Moving data between an external routine and the database system involves the well-known impedance mismatch with regard to both, data types and set-orientation versus single-datum orientation.*
- *Creation of new SQL sessions for the execution of external routines that contain SQL code may be expensive (context switching overhead).*

5) SQLJ-1

SQLJ-1 allows for the implementation of external routines using the Java Programming language. The Java application may read and modify database data system using JDBC or SQLJ.

Create a java class containing a method that takes a publication key as input parameter and returns the number of keywords assigned to the publication! What steps are required to create an external function based on this method using DB2?

1. Create a Java class that contains the required method. Note that a JDBC connection to the database is obtained using the URL `jdbc:default:connection`.

```
package extJavaFunc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class myFunctions {

      public static int keywCnt(String keyword) throws SQLException {
            Connection connection = null;
            int cnt = 0;
            String url = "jdbc:default:connection";
            connection = DriverManager.getConnection(url);
            String query = "select count (*)  " +
                        "from publkeywords " +
                        "where publication = PublicationT(?)";
            PreparedStatement preparedStatement = connection
                        .prepareStatement(query);
            preparedStatement.setString(1, keyword);
            ResultSet resultSet = preparedStatement.executeQuery();
            if (resultSet.next()) {
                  cnt = resultSet.getInt(1);
            }
            resultSet.close();
            preparedStatement.close();
            return cnt;
      }

}
```

2. Compile the Java class and generate a JAR archive.

```
jar cf extJavaFunc.jar extJavaFunc\myFunctions.class
```

3. Install the JAR file to the database server using a system-defined routine.

```
call sqlj.install_jar('file:///.../extJavaFunc.jar', 'extJava-
FuncJAR');
```

4. Create an external function based on the Java method.

```
create function javaKeywCnt(keyword varchar(25))
returns integer
reads sql data
external name 'extJavaFuncJAR:extJavaFunc.myFunctions!keywCnt'
language java
parameter style java;
```

6) User-defined Transforms

Transform functions are used to exchange structured type values with host language programs and with external functions and methods. Pairs of such transform functions are called transform groups; the so called `TO SQL` function is invoked when transferring a structured type instance from the host language side of the interface to the SQL side, and the other, called `FROM SQL` function, is used for transferring from the SQL side to the host language side.

Reconsider the `AccessCntT` structured type introduced by the first exercise sheet. Define a pair of user-defined functions to convert instances of this type to a string representation (simply separate the two integer values by a comma) and vice versa! Furthermore, create a transform group using these functions!

```
create function accessCntToStr (a AccessCntT)
returns VARCHAR(20)
language sql
return trim(char(a.localAccesses)) || ', ' ||
       trim(char(a.remoteAccesses));


create function strToAccessCnt (a Varchar(20))
returns AccessCntT
language sql
return accessCntT(integer(substr(a, 1, locate(',', a)-1)),
                  integer(substr(a, locate(',', a)+1)));

CREATE TRANSFORM FOR AccessCntT
DB2_PROGRAM(
    TO SQL WITH FUNCTION strToAccessCnt,
    FROM SQL WITH FUNCTION accessCntToStr
);
```

Note that DB2_PROGRAM is DB2's default transform group.