What Is A Stored Procedure?

By Craig S. Mullins

Many RDBMSes provide support for stored procedures to assist developers in accessing stored data. But what exactly is a stored procedure? Those who have never used them may be inclined to dismiss them as nonessential. However, stored procedures provide a great deal of power when accessing relational data. Soon, any RDBMS that does not support stored procedures will be significantly hampered in the fast-paced client/server marketplace.

The Basics

Stored procedures are specialized pieces of code stored in the RDBMS. The motivating reason for stored procedure support is to move SQL code off the client and onto the database server. This results in less overhead because one client request invokes multiple SQL statements.

Stored procedures are like other database objects, such as tables, views and indexes, in that they are controlled by the DBMS. Depending on the particular RDBMS implementation, stored procedures may also physically reside in the RDBMS. However, a stored procedure is not "physically" associated with any other object in the database. It can access and/or modify data in one or more tables. Basically, stored procedures can be thought of as "programs" that "live" in the RDBMS.

All stored procedures are uniquely named and invoked by executing the procedure by name. This is a crucial point: A stored procedure must be directly called before it can be executed. It is not event-driven. Contrast this with the concept of database triggers, which are event-driven and never explicitly called. Triggers are automatically executed (or "fired") by the RDBMS as the result of an action; stored procedures are never automatically invoked (unless they are explicitly called from within a trigger). For more information on triggers, see "What Is A Trigger?" (*ESJ*, August 1994).

Why Use Stored Procedures?

The most compelling reason to use stored procedures is to promote code reusability. Instead of replicating code on multiple servers, stored procedures enable code to reside instead in a single place — the database server. Stored procedures can be called from client programs and, possibly, other stored procedures. This is preferable to cannibalizing sections of program code for each new application that must be developed. By coding a stored procedure, the logic can be invoked from multiple processes instead of being recoded into each new process every time the code is required.

Creating and using stored procedures increases consistency. If everyone is using the same stored procedures, then the DBA can be assured everyone is running *exactly* the same code. If users used their own individual batch scripts, no assurance could be given that the same code was being used by all. In fact, it is almost a certainty that inconsistencies *will* occur.

Stored procedures are particularly useful for reducing the overall code maintenance effort. Because the stored procedure exists in one place, changes can be made quickly without requiring propagation of the change to multiple workstations.

Another common reason to employ stored procedures is to enhance performance. A stored procedure may result in enhanced performance because it is typically stored in parsed (or compiled) format, thereby eliminating parser overhead. Additionally, stored

Stored procedures are a powerful feature of many RDBMS products today. They enable multiple data access statements to be executed with a single request.

procedures tend to reduce network traffic because multiple SQL statements can be invoked with a single execution of a procedure instead of sending multiple requests across the communication lines.

Security issues can be addressed using stored procedures. Consider a stored procedure that can access only a specific portion of the data. Certain individuals can be granted access to the database only through the stored procedure, thereby encapsulating database security within the procedure.

Additionally, stored procedures can be coded to support database integrity constraints, reduce code maintenance efforts and support remote data access (see the next section on RPC). There are probably more reasons to use stored procedures than can be adequately covered in an article of this length.

Remote Procedure Call (RPC)

Stored procedures are critical for supporting RPC according to the Open Software Foundation's Distributed Computing Environment (OSF/DCE). The basic premise behind RPC is that an algorithm can be executed from a workstation, but the actual code is processed on a remote machine.

An RDBMS supports RPC if it provides distributed data access and distributed stored procedures. RPC support enables a workstation application to invoke a stored procedure that executes on the server. The workstation and the server are different machines — remote from one another. This is an RPC.

RPC is not necessarily supported by an RDBMS product simply because it provides stored procedure support.

Creating Stored Procedures

Each RDBMS product that supports stored procedures does so in a different manner. There is no consistent, implemented methodology for creating stored procedures. There are, however, two basic types of stored procedures.

The first type of stored procedure is implemented using SQL alone. Typically, the RDBMS vendor will provide extensions to its SQL dialect to support looping, branching and other programming constructs. This facilitates the coding of stored procedures without requiring an external host language (such as C, COBOL or REXX). In this type of environment, stored procedures are usually created using DDL. The *create procedure* statement will allow the developer to name the procedure, along with any parameters, followed by the actual (extended) SQL code for the stored procedure.

The second type of stored procedure requires a host language. The stored procedure is coded in that host language (3GL or 4GL) and includes embedded SQL. Depending on the actual implementation, the SQL may be dynamic, static or both. Additionally, some RDBMS products support true embedded SQL while others require SQL API (such as ODBC) calls. Once created, the stored procedure is registered to the RDBMS and can be explicitly executed as desired.

Executing A Stored Procedure

Stored procedures are invoked by name in conjunction with a keyword. The keyword will depend on the implementation but is typically something like "execute" or "call." So, to invoke a stored procedure named "calc_salary" a statement such as the following may be used:

```
execute calc_salary parameters
```

Usually, stored procedures can be executed from within any of the following:

- Batches of SQL statements
- Other stored procedures
- Triggers
- Third-party programs
- Client application programs.

Nested Procedures

Some RDBMS products allow one stored procedure to call another stored procedure. However, this is not a requirement for stored procedures, so check your particular product to see if this feature is supported.

Nested procedures can be tricky to code and debug. This is particularly true if stored procedures can recursively call themselves. Before implementing nested procedures, be sure to document which procedures can be called by other procedures.

Generally, there is an upper limit on the levels of nested stored procedure calls that can be made.

Procedure Cache And Query Plans

Stored procedures are often stored in a parsed or compiled format. The manner in which the RDBMS provides this will differ. For example, DB2 for AIX (formerly DB2/6000) requires the stored procedure program to be compiled before it will be executed. SQL Server, though, provides a more interesting scenario.

SQL Server parses the stored procedure and stores it in the system catalog when it is created. Upon its first execution, the parsed code is read from the system catalog, optimized, compiled and loaded into an area of storage called procedure cache. Because the procedure is optimized the first time it is read from disk, the query plan can be reused by subsequent processes as long as it remains in the procedure cache.

Although optimization of the procedure on initial access aids reusability, the first query plan created may not be appropriate for all users. Different users may have different access requirements dictating that new access paths be formulated. In this case, each time the query is run by a different user, a new query plan should be created.

To reoptimize a stored procedure for each and every execution, it can be cre-

VSAMEXEC™

- REXX Command Interface to access VSAM Data Sets
- Supports KSDS, ESDS, RRDS and LDS data sets
- Key, Address and Control Interval access
- Interface to ENQ/DEQ
- Available on TSO/ISPF, BATCH, NETVIEW, etc.

$ 2950.00 OTC

FOLIUM Inc.
207 East Bay St., Suite 300
Charleston, SC 29401

1-803-853-2241

30 DAYS FREE TRIAL

CIRCLE #112 on Reader Service Card ▲

ated using the "with recompile" option. Every time the procedure is executed, a new query plan will be formulated.

Alternately, the stored procedure could be created as usual (without the "with recompile" parameter), and the "with recompile" option can be specified on the exec statement instead. Using this approach, some users can reuse the current query plan, and others (usually only extremely performance-sensitive executions) can specify "with recompile" at execution time to reevaluate access paths.

Stored Procedure Restrictions

There are typically restrictions on the type of processing that can be accomplished by stored procedures. Once again, however, this will vary based on the RDBMS implementation.

In general, it is quite common for stored procedures to limit DDL and DCL, use of transaction processing statements such as COMMIT/ROLLBACK and certain database commands. Additionally, there may be a limit on the overall size (number of bytes) of the stored procedures and the number of parameters that can be specified.

EXAMPLE 1

Sample Stored Procedure

```
create proc verify_book (@t_id char(6)=null)
as
declare @msg varchar(60)

if @t_id = null
begin
        print "Usage is: verify_book title_id"
        return
end
else
begin
        if exists (select 1 from titles where title_id =@t_id)
              select title_id, title, pubdate
              where title_id = @t_id
        else
        begin
              select @msg="title_id" + @t_id + "not on file."
              print @msg
              return
        end
end
return
```

Using Parameters In Stored Procedures

Most stored procedures use parameters to pass information to and from the procedure. Parameters make stored procedures more flexible. The same stored procedure can retrieve, update, delete and/or insert different data values based on parameterized input.

For an example of a SQL Server stored procedure refer to Example 1. This stored procedure, named "verify_book", accepts a parameter named @t_id. The stored procedure accepts a title_id as input and verifies a book identified by the title_id exists.

Note the parameter is given a default value, in this case "null." It is always a good idea to supply a default value for parameters. When the stored procedure is executed, a value should be passed to the parameter. If no value is passed, the default value will remain. The stored procedure can check for the default value and prompt the user to supply an appropriate value.

Stored Procedure Tips

Before saving a stored procedure in the database, test the SQL by first coding it into a batch. When all of the bugs have been worked out, the SQL can be converted into a stored procedure and saved in the database. Following this approach can save testing and debugging time.

Additionally, be sure to sufficiently document the purpose and processing

Run Legacy CICS Programs Above The 16MB Line with XAbove/CICS

Compatible with COBOL and PL/I Command Level programs

- Eliminate MRO overhead
- Make more room for programs to run below the line
- Run all programs resident
- Increase program buffer sizes
- Eliminate CICS compression

Available for VSE/ESA, MVS/XA, and MVS/ESA

For more information, or to order your 30-day free trial, please write or call today

SOFTWARE PURSUITS

1420 Harbor Bay Parkway, Suite 200 • Alameda, CA 94502
510-769-4900
Toll Free 800-367-4823, Fax 510-769-4944

CIRCLE #43 on Reader Service Card ▲

flow of stored procedures using comments. SQL comments are typically coded using two dashes ("--") preceding the SQL. Use the commenting syntax of the appropriate development language if the stored procedure is not implemented in SQL but instead in a 3GL or 4GL.

Conclusion

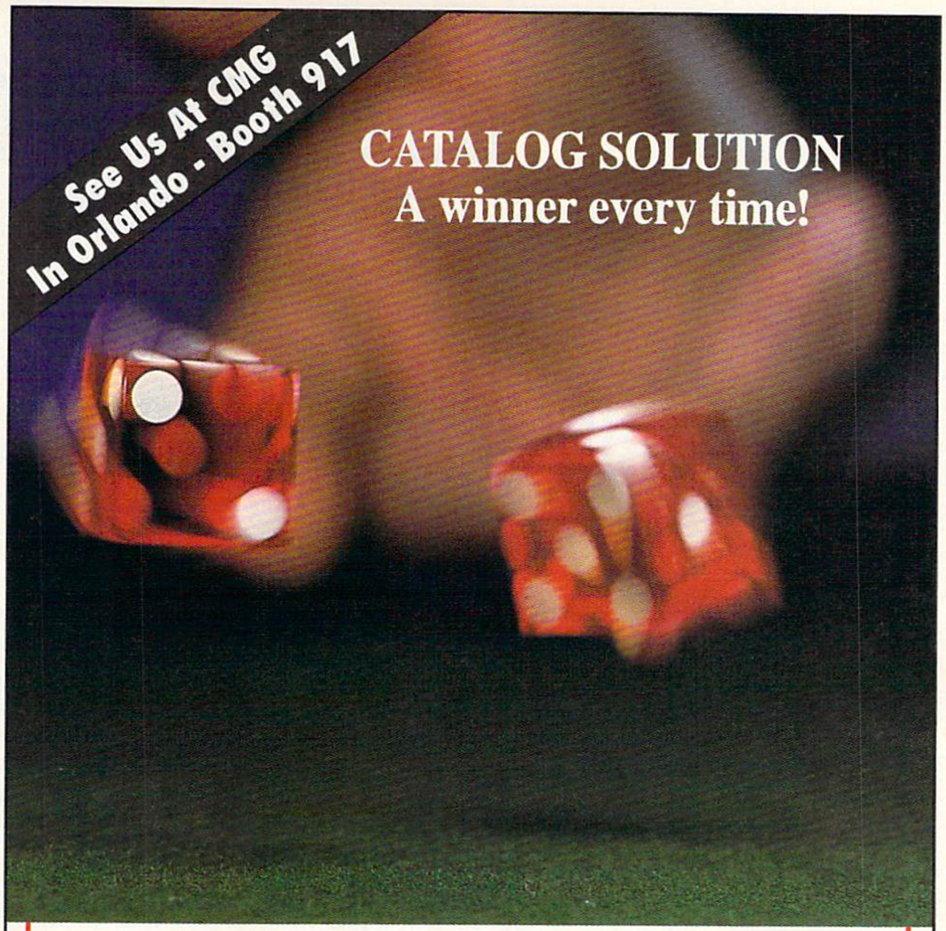
Stored procedures are a powerful feature of many RDBMS products today. They enable multiple data access statements to be executed with a single request. Additionally, they are controlled

Instead of replicating code on multiple servers, stored procedures enable code to reside instead in a single place — the database server.

and managed by the RDBMS, providing a consistent and reusable point of reference for frequently executed database code. Many products (such as DB2 for AIX, SQL Server and Oracle 7) support stored procedures today. Many others (such as DB2 for MVS) will support them in the near future. It is a wise course of action to learn about stored procedures and what they can provide today, before it is too late!

ABOUT THE AUTHOR

Craig S. Mullins is manager of education technology at Platinum technology, inc., where he specializes in Sybase SQL Server, the DB2 family and client/server. Platinum technology, inc., 1815 S. Meyers Rd., Oakbrook Terrace, IL 60181, (708) 620-5000. He can also be reached through CompuServe, 70410,237.



Without a facility to recover from Catalog/VTOC/VVDS problems, you're leaving your system availability to the roll of the dice.

Keep the odds in your favor with **Catalog Solution**,™ the premier tool for catalog diagnosis, management, recovery, maintenance and contingency planning.

- Repair broken VSAM data sets and catalogs
- Reduce downtime from hours to minutes
- Prepare BCS & VVDS for SMS implementation
- Identify over-allocated data sets
- Backup and restore catalogs
- Identify data sets with excessive CI/CA splits and/or extents
- Identify duplicate VVDS records
- Completely uplevel VVDS records

Don't gamble with your catalog environment. Call Softworks today and discover what Catalog Solution can do for you and your data center.

SOFTWORKS
Think Solutions. Think Softworks.

5845 Richmond Highway, Alexandria, Virginia 22303
U.S.A. 1-800-727-4422 • International 1-703-317-2424

CIRCLE #29 on Reader Service Card ▲