A.4 MySQL 5.7 FAQ: Stored Procedures and Functions

A.4.1.	Does MySQL 5.7 support stored procedures and functions?
	Yes. MySQL 5.7 supports two types of stored routines—stored procedures and stored functions.
A.4.2.	Where can I find documentation for MySQL stored procedures and stored functions?
	See Section 21.2, “Using Stored Routines (Procedures and Functions)”.
A.4.3.	Is there a discussion forum for MySQL stored procedures?

	Yes. See http://forums.mysql.com/list.php?98.
A.4.4.	Where can I find the ANSI SQL 2003 specification for stored procedures?
	Unfortunately, the official specifications are not freely available (ANSI makes them available for purchase). However, there are books—such as *SQL-99 Complete, Really* by Peter Gulutzan and Trudy Pelzer—which give a comprehensive overview of the standard, including coverage of stored procedures.
A.4.5.	How do you manage stored routines?
	It is always good practice to use a clear naming scheme for your stored routines. You can manage stored procedures with `CREATE [FUNCTION|PROCEDURE]`, `ALTER [FUNCTION|PROCEDURE]`, `DROP [FUNCTION|PROCEDURE]`, and `SHOW CREATE [FUNCTION|PROCEDURE]`. You can obtain information about existing stored procedures using the `ROUTINES` table in the `INFORMATION_SCHEMA` database (see Section 22.20, “The INFORMATION_SCHEMA ROUTINES Table”).
A.4.6.	Is there a way to view all stored procedures and stored functions in a given database?

	Yes. For a database named ***`dbname`***, use this query on the `INFORMATION_SCHEMA.ROUTINES` table:
	`SELECT ROUTINE_TYPE, ROUTINE_NAME`
	`    FROM INFORMATION_SCHEMA.ROUTINES`
	`    WHERE ROUTINE_SCHEMA='dbname';`
	
	For more information, see Section 22.20, “The INFORMATION_SCHEMA ROUTINES Table”.
	
	The body of a stored routine can be viewed using `SHOW CREATE FUNCTION` (for a stored function) or `SHOW CREATE PROCEDURE` (for a stored procedure). See Section 14.7.5.9, “SHOW CREATE PROCEDURE Syntax”, for more information.

A.4.7.	Where are stored procedures stored?
	In the `proc` table of the `mysql` system database. However, you should not access the tables in the system database directly. Instead, use `SHOW CREATE FUNCTION` to obtain information about stored functions, and `SHOW CREATE PROCEDURE` to obtain information about stored procedures. See Section 14.7.5.9, “SHOW CREATE PROCEDURE Syntax”, for more information about these statements. You can also query the `ROUTINES` table in the `INFORMATION_SCHEMA` database—see Section 22.20, “The INFORMATION_SCHEMA ROUTINES Table”, for information about this table.
A.4.8.	Is it possible to group stored procedures or stored functions into packages?

	No. This is not supported in MySQL 5.7.
A.4.9.	Can a stored procedure call another stored procedure?
	Yes.
A.4.10.	Can a stored procedure call a trigger?
	A stored procedure can execute an SQL statement, such as an `UPDATE`, that causes a trigger to activate.
A.4.11.	Can a stored procedure access tables?
	Yes. A stored procedure can access one or more tables as required.
A.4.12.	Do stored procedures have a statement for raising application errors?
	Yes. MySQL 5.7 implements the SQL standard `SIGNAL` and `RESIGNAL` statements. See Section 14.6.7, “Condition Handling”.
A.4.13.	Do stored procedures provide exception handling?
	MySQL implements `HANDLER` definitions according to the SQL standard. See Section 14.6.7.2, “DECLARE ... HANDLER Syntax”, for details.
A.4.14.	Can MySQL 5.7 stored routines return result sets?
	Stored procedures can, but stored functions cannot. If you perform an ordinary `SELECT` inside a stored procedure, the result set is returned directly to the client. You need to use the MySQL 4.1 (or above) client/server protocol for this to work. This means that—for instance—in PHP, you need to use the `mysqli` extension rather than the old `mysql` extension.
A.4.15.	Is `WITH RECOMPILE` supported for stored procedures?
	Not in MySQL 5.7.
A.4.16.	Is there a MySQL equivalent to using `mod_plsql` as a gateway on Apache to talk directly to a stored procedure in the database?
	There is no equivalent in MySQL 5.7.
A.4.17.	Can I pass an array as input to a stored procedure?
	Not in MySQL 5.7.

A.4.18.	Can I pass a cursor as an `IN` parameter to a stored procedure?
	In MySQL 5.7, cursors are available inside stored procedures only.
A.4.19.	Can I return a cursor as an `OUT` parameter from a stored procedure?
	In MySQL 5.7, cursors are available inside stored procedures only. However, if you do not open a cursor on a `SELECT`, the result will be sent directly to the client. You can also `SELECT INTO` variables. See Section 14.2.9, “SELECT Syntax”.
A.4.20.	Can I print out a variable's value within a stored routine for debugging purposes?
	Yes, you can do this in a *stored procedure*, but not in a stored function. If you perform an ordinary `SELECT` inside a stored procedure, the result set is returned directly to the client. You will need to use the MySQL 4.1 (or above) client/server protocol for this to work. This means that—for instance—in PHP, you need to use the `mysqli` extension rather than the old `mysql` extension.
A.4.21.	Can I commit or roll back transactions inside a stored procedure?
	Yes. However, you cannot perform transactional operations within a stored function.
A.4.22.	Do MySQL 5.7 stored procedures and functions work with replication?
	Yes, standard actions carried out in stored procedures and functions are replicated from a master MySQL server to a slave server. There are a few limitations that are described in detail in Section 21.7, “Binary Logging of Stored Programs”.
A.4.23.	Are stored procedures and functions created on a master server replicated to a slave?
	Yes, creation of stored procedures and functions carried out through normal DDL statements on a master server are replicated to a slave, so the objects will exist on both servers. `ALTER` and `DROP` statements for stored procedures and functions are also replicated.
A.4.24.	How are actions that take place inside stored procedures and functions replicated?
	MySQL records each DML event that occurs in a stored procedure and replicates those individual actions to a slave server. The actual calls made to execute stored procedures are not replicated. Stored functions that change data are logged as function invocations, not as the DML events that occur inside each function.
A.4.25.	Are there special security requirements for using stored procedures and functions together with replication?
	Yes. Because a slave server has authority to execute any statement read from a master's binary log, special security constraints exist for using stored functions with replication. If replication or

	binary logging in general (for the purpose of point-in-time recovery) is active, then MySQL DBAs have two security options open to them: 1. Any user wishing to create stored functions must be granted the `SUPER` privilege. 2. Alternatively, a DBA can set the `log_bin_trust_function_creators` system variable to 1, which enables anyone with the standard `CREATE ROUTINE` privilege to create stored functions.
A.4.26.	What limitations exist for replicating stored procedure and function actions?
	Nondeterministic (random) or time-based actions embedded in stored procedures may not replicate properly. By their very nature, randomly produced results are not predictable and cannot be exactly reproduced, and therefore, random actions replicated to a slave will not mirror those performed on a master. Declaring stored functions to be `DETERMINISTIC` or setting the `log_bin_trust_function_creators` system variable to 0 will not allow random-valued operations to be invoked. In addition, time-based actions cannot be reproduced on a slave because the timing of such actions in a stored procedure is not reproducible through the binary log used for replication. It records only DML events and does not factor in timing constraints. Finally, nontransactional tables for which errors occur during large DML actions (such as bulk inserts) may experience replication issues in that a master may be partially updated from DML activity, but no updates are done to the slave because of the errors that occurred. A workaround is for a function's DML actions to be carried out with the `IGNORE` keyword so that updates on the master that cause errors are ignored and updates that do not cause errors are replicated to the slave.
A.4.27.	Do the preceding limitations affect MySQL's ability to do point-in-time recovery?
	The same limitations that affect replication do affect point-in-time recovery.
A.4.28.	What is being done to correct the aforementioned limitations?
	You can choose either statement-based replication or row-based replication. The original replication implementation is based on statement-based binary logging. Row-based binary logging resolves the limitations mentioned earlier. Mixed replication is also available (by starting the server with `--binlog-format=mixed`). This hybrid, “smart” form of replication “knows” whether statement-level replication can safely be used, or row-level replication is required. For additional information, see Section 18.2.1, “Replication Formats”.