Stored Procedures in MySQL 5.0

Per-Erik Martin, MySQL AB

San Jose, 2003-04-11

Content

- **Introduction**
- Overview of the Stored Procedure Implementation
- The SQL-99 Language
- The Implementation
- Example:
  - Compilation
  - Execution
- External Languages
- Current Status

Who Am I?

- Per-Erik ”pem” Martin
- Lives in Uppsala, Sweden
- Background in
  - Lisp/Prolog development (distant past)
  - Theoretical stuff at University (past)
  - Security (CAs, Authentication servers, etc)
- MySQL employee since June 2002

Who Are You?

- How many have used stored procedures in some other database?

Who Are You?

- How many have used stored procedures in some other database?
- How many need stored procedures in MySQL?

Who Are You?

- How many have used stored procedures in some other database?
- How many need stored procedures in MySQL?
- How many know anything about the internals of the MySQL server?

”Disclaimer”

- This presentation describes **work in progress!**
- Most things have been implemented...
- ...but might change
- A few things are not yet implemented...
- ...and might be different

Content

- Introduction
- **Overview of the Stored Procedure Implementation**
- The SQL-99 Language
- The Implementation
- Example:
  - Compilation
  - Execution
- External Languages
- Current Status

What Kind of SPs to Implement?

- The standard: SQL-99
  - Embedded SQL language
  - External language procedures

Why SPs are Good... and Bad

- Run on the server
  - save transmission time
  - ...but puts more load on the server
- Added security options
- External languages:
  - Access to the OS
  - Optimization
  - ...but less portable

More About the Standard

- The SP part of the standard touches a wide variety of different features of an SQL DB:
  - User Defined Types
  - Schemas/Modules
  - ”States” (exception handling)
- The implementation must be limited to what is relevant and possible for MySQL

Implementation Goals

- Find a useful subset of the standard
- Adopt to make it fit into the existing MySQL server
- ”Small steps”:
  - Don’t try to do everything at once
  - Essentials first, more features later
- Migration from other SQL dialects:
  - Features and syntax can be added as long as they’re not in conflict with the standard

What's Included?

- CREATE / DROP / ALTER
  - PROCEDURE
  - FUNCTION
- All the basic SQL language mechanisms:
  - BEGIN/END blocks
  - Local variables
  - IF, CASE
  - WHILE, LOOP, REPEAT, FOR, LEAVE, ITERATE
- External languages

What's Different?

- "Specific name" (overloading) is not implemented. Names must be unique
- The SET syntax is extended, and "merged" with the preexisting MySQL syntax for global/system variables
- CONDITIONs/HANDLERs: Limited implementation
- Type checking: optional
- Authorization:
  - The standard requires "setuid"
  - Will also support running with caller's id

What's Excluded?

- RESTRICT / CASCADE for ALTER / DROP
- METHODs (related to User Defined Types)
- MODULEs (related to Schema)

Various...

- No queries in a FUNCTION
  ...at least not to begin with:
  This limitation will probably go away in the future
- Semicolon is a separator in bodies
  ...and also query delimiter in the mysql client:
  Possible to change the delimiter in the 5.0 client

Content

- Introduction
- Overview of the Stored Procedure Implementation
- **The SQL-99 Language**
- The Implementation
- Example:
  - Compilation
  - Execution
- External Languages
- Current Status

The SQL Language

```
CREATE PROCEDURE
      foo([IN] x INT, INOUT y INT, OUT z FLOAT)
  ...body...
```

```
CREATE FUNCTION bar(x INT, y CHAR(8))
      RETURNS CHAR(16)
  ...body with RETURN...
```

```
CALL foo(1, 2, var)
```

```
SELECT * FROM table WHERE x = bar(y, ”foo”)
```

```
DROP PROCEDURE foo
```

The SQL Language: Blocks

```
BEGIN ATOMIC
    ...(no COMMIT or ROLLBACK allowed)...
END
```

```
BEGIN [NOT ATOMIC]
    ...(COMMIT and ROLLBACK allowed)...
END
```

The SQL Language: Variables

```
BEGIN
    DECLARE s CHAR(8) DEFAULT ”foo”;
    DECLARE x, y INT;

    SET x  = 1;
    CALL foo(x, y, s);
END
```

```
DECLARE PROCEDURE
    foo(x INT, OUT y INT, INOUT s CHAR(8))
    ...
```

The SQL Language: Branching

```
IF expr1 THEN
    ...
ELSEIF expr2 THEN
    ...
ELSE
    ...
END IF
```

```
CASE expr
    WHEN val1 THEN  ...
    WHEN val2 THEN ...
    ELSE ...
END CASE
```

```
CASE
    WHEN expr1 THEN  ...
    WHEN expr2 THEN ...
    ELSE ...
END CASE
```

The SQL Language: Loops

```
foo:
LOOP
    ... LEAVE foo; ...
END LOOP foo
```

```
WHILE expression DO
    ...
END WHILE
```

```
bar:
REPEAT
    ... ITERATE bar; ...
UNTIL expression END REPEAT
```

The SQL Language: Cursors

```
BEGIN
    DECLARE CURSOR c FOR SELECT x FROM ...;
    DECLARE y INT;

    OPEN c;
    FETCH c INTO y;
    CLOSE c;
END
```

Conditions and Handlers

```
BEGIN
    DECLARE done BOOLEAN DEFAULT FALSE;
    DECLARE nomore CONDITION FOR ’02000’;
    DECLARE CONTINUE HANDLER FOR nomore
        SET done = TRUE;
    ...
    FETCH c INTO v;
    IF done THEN
        ...
    END IF;
    ...
END
```

The SQL Language: FOR loop

```
FOR x AS c CURSOR FOR
    SELECT ... FROM ...
DO
    ....
END FOR
```

Example: Withdrawal

```
CREATE PROCEDURE withdraw(
      p_amount DECIMAL(6,2),
      p_tellerid INTEGER,
      p_custid INTEGER)
MODIFIES SQL DATA
BEGIN ATOMIC
      UPDATE customers
            SET balance=balance - p_amount;
      UPDATE tellers
            SET cashonhand=cashonhand - p_amount
            WHERE tellerid = p_tellerid;
      INSERT INTO transactions
            VALUES ( p_custid, p_tellerid, p_amount );
END
```

Content

- Introduction
- Overview of the Stored Procedure Implementation
- The SQL-99 Language
- **The Implementation**
- Example:
  - Compilation
  - Execution
- External Languages
- Current Status

Recap: How Queries Are Executed in MySQL

- Two important structures:
  THD: The connection state
  LEX: The ”compiled” query
    Contains things like:
      - Command code (SELECT, INSERT, ...)
      - Tables, fields, conditions, ...
- Field names, constants, expressions, are all objects of the class ”Item”

Creating, dropping, etc

- Procedures and functions are stored in a system table: mysql.proc
- Fields are: name-type (key), definition (a blob), and other attributes (creation time, etc)
- The definition is the entire original ”CREATE” statement
- A procedure (or function) is fetched from the database and parsed on demand, and then kept in-memory

The Result of Compiling an SP

- A LEX structure containing a procedure ”head” object
- The head contains most of the table data and a sequence of ”instructions” generated by the parser
- The head has two ”execute” methods, one for PROCEDUREs and one for FUNCTIONs

”Instructions”

- An instruction is an object with an ”execute” method which returns a status code and a ”next instruction” (in the case of a jump)

- Different instructions are subclasses with different data and execute methods:
  1. Statement; contains the query’s LEX
  2. SET local variable; contains the frame offset and value (an Item)
  3. Unconditional jump
  4. Conditional jump; contains the value to test
  5. Return value; contains the value (for FUNCTIONs)

More About Parsing

The procedure is actually parsed in two different modes:

1. At creation time; syntax check only
2. At first call time; check if called procedures exist, check parameter count and types, etc

Additional checks possible at each invocation, (e,g, authorization)

Local Variables

- Local variables must appear as Items, just like any other data
- But, the same variable Item (stored in a LEX) must have different values for different callers
- The values are kept in a procedure context (call frame) in the caller's THD structure
- The Item contains the frame offset and defers all method calls to the actual Item in the current frame

The Parser Context

- During parsing, a parser context in the procedure's LEX is used to keep track of local variable's positions, types, modes, and label references
- This is also used at invocation time to get the modes (IN/INOUT/OUT) of parameters

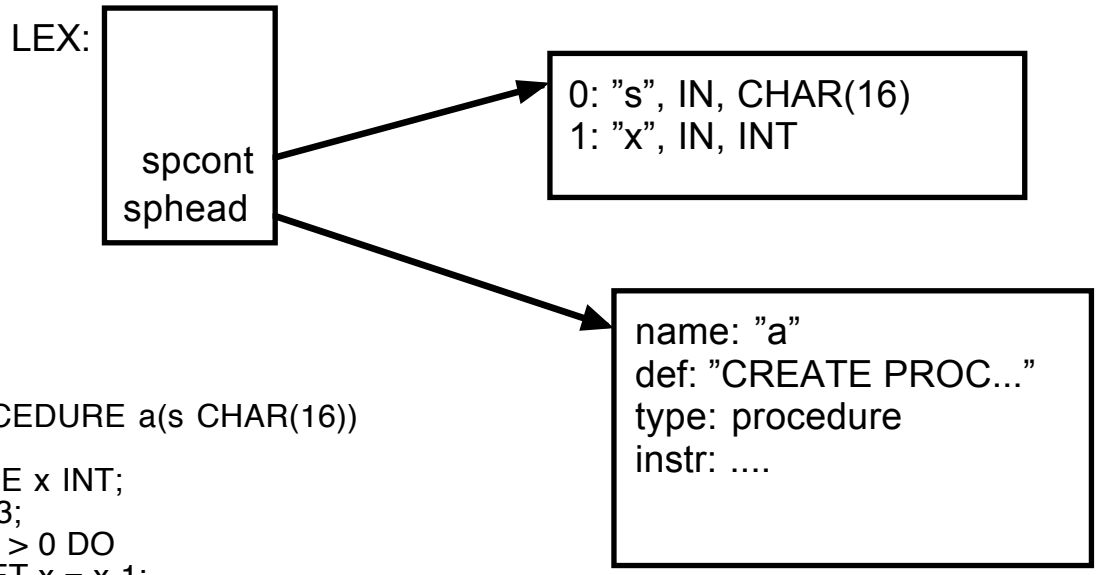
Content

- Introduction
- Overview of the Stored Procedure Implementation
- The SQL-99 Language
- The Implementation
- Example:
  - **Compilation**
  - Execution
- External Languages
- Current Status

Compilation Example (1)

```
CREATE PROCEDURE a(s CHAR(16))
BEGIN
    DECLARE x INT;
    SET x = 3;
    WHILE x > 0 DO
        SET x = x-1;
        INSERT INTO db.tab VALUES (x, s);
    END WHILE;
END
```

Compilation Example (2)




```
CREATE PROCEDURE a(s CHAR(16))
BEGIN
        DECLARE x INT;
        SET x = 3;
        WHILE x > 0 DO
                SET x = x-1;
                INSERT INTO db.tab VALUES (x, s);
        END WHILE;
END
```

Compilation Example (3)

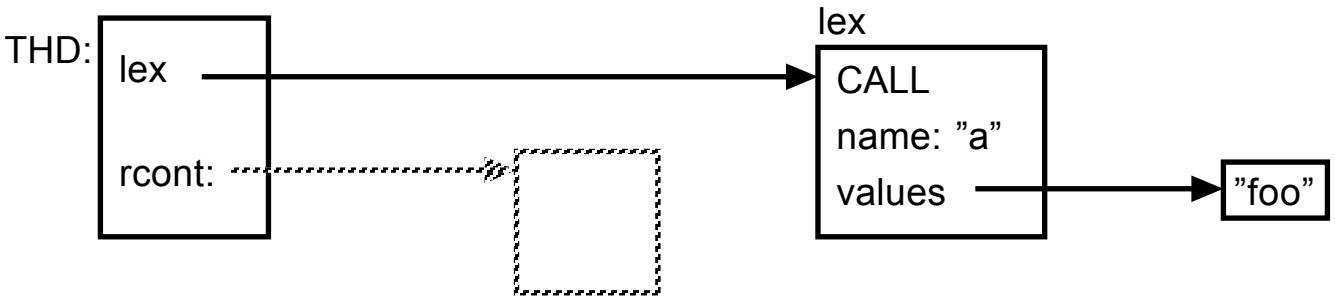
```
CREATE PROCEDURE a(s CHAR(16))
BEGIN
    DECLARE x INT;
    SET x = 3;
    WHILE x > 0 DO
        SET x = x-1;
        INSERT INTO ...
    END WHILE;
END
```

```
0:  set(1, '3')
1:  jump_if_not('x>0', 5)
2:  set(1, 'x-1')
3:  statement('INSERT...')
4:  jump(1)
5:  <end>
```

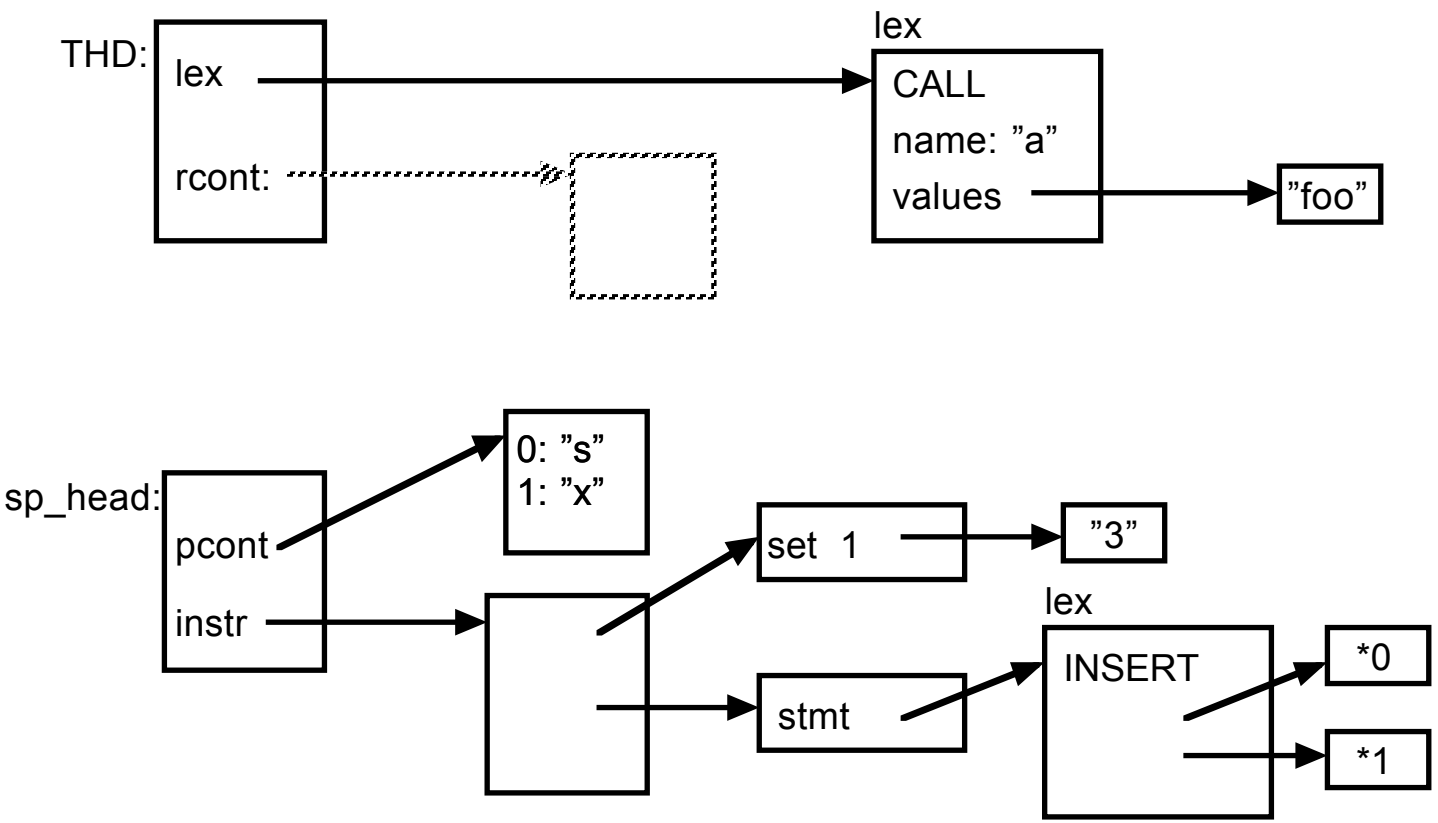
Content

- Introduction
- Overview of the Stored Procedure Implementation
- The SQL-99 Language
- The Implementation
- Example:
  - Compilation
  - **Execution**
- External Languages
- Current Status

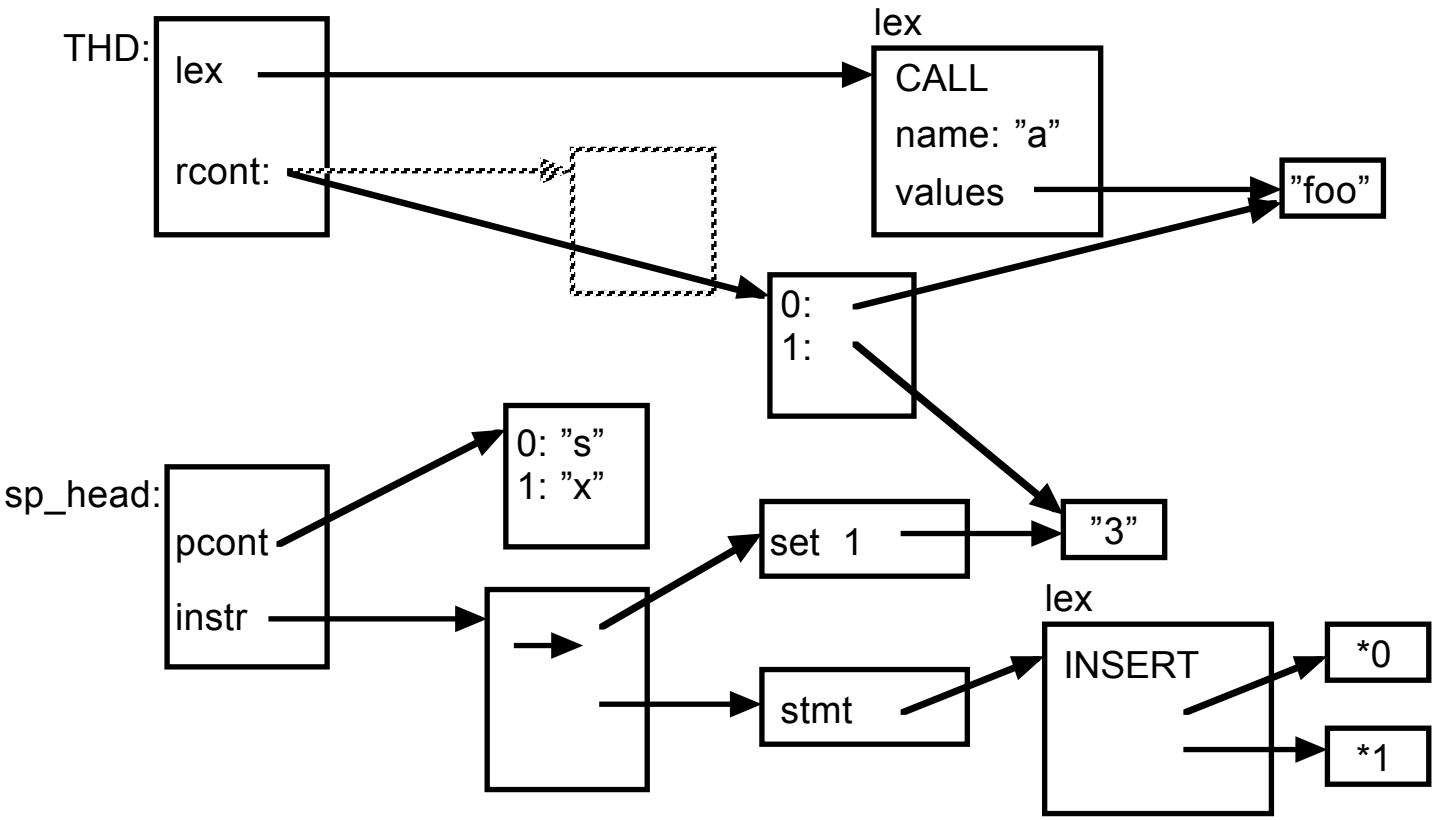
Calling a Procedure (1)



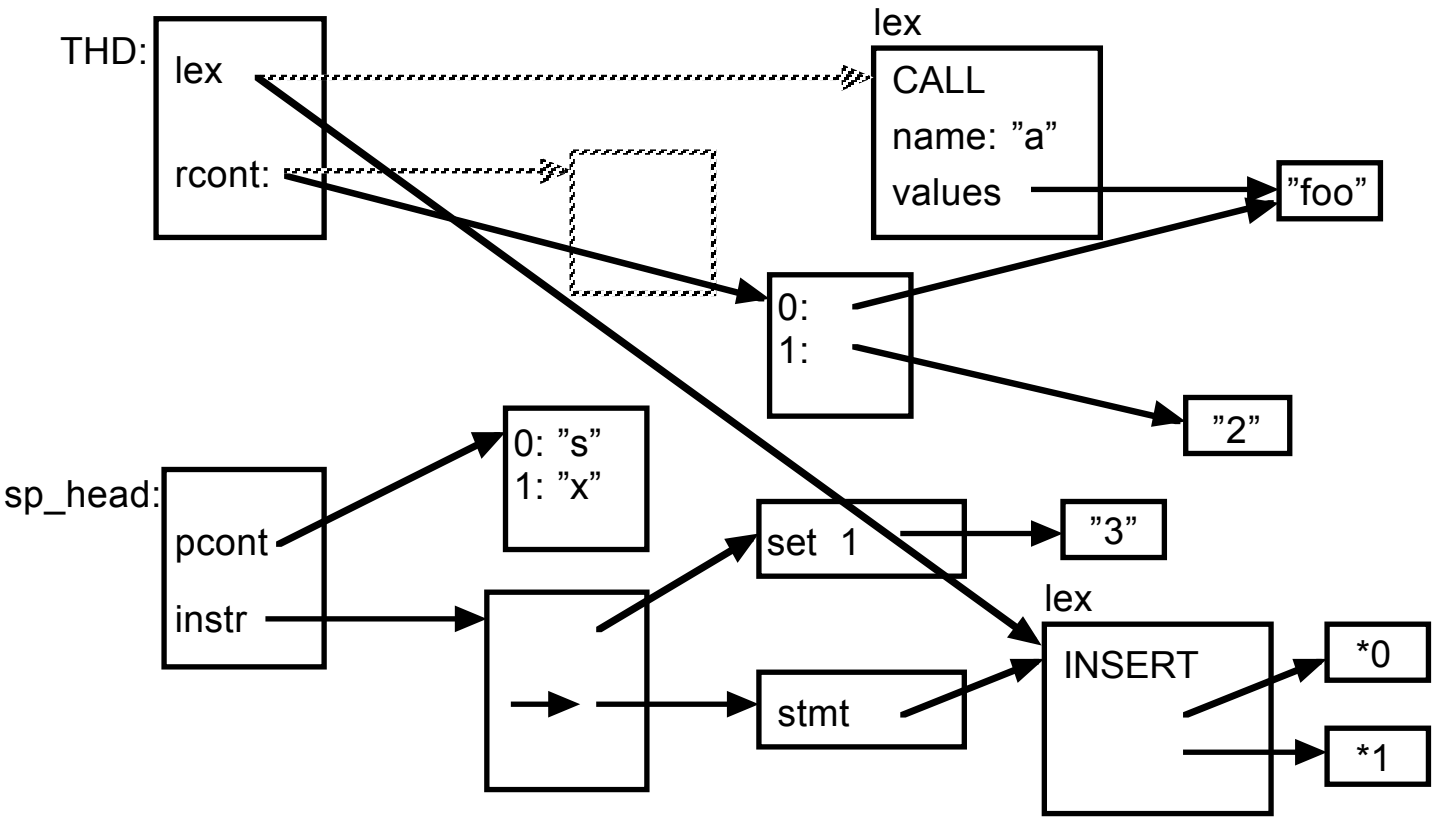
Calling a Procedure (2)



Calling a Procedure (3)



Calling a Procedure (4)



Content

- Introduction
- Overview of the Stored Procedure Implementation
- The SQL-99 Language
- The Implementation
- Example:
  - Compilation
  - Execution
- **External Languages**
- Current Status

External Languages (1)

- Managed like SQL language procedures, with additional attributes, e.g.:
  - LANGUAGE (C | PERL | ...)
  - EXTERNAL NAME ...
- An external procedure often needs additional information, e.g. the name of a DLL, shared object file, or script
  - How to specify this is unspecified(!)

External Languages (2)

- Two possibilities:
  - Run in a separate process (IPC)
  - Link into the mysqld process
- First choice: Separate process is slower, but safe and robust
- Second choice: Fast and dangerous

External Languages (3)

- The plan:
  - Aim for the safe method, but will probably have both
  - Implement one or two model languages
  - Provide an extendable interface to make it easy to add new languages

Content

- Introduction
- Overview of the Stored Procedure Implementation
- The SQL-99 Language
- The Implementation
- Example:
  - Compilation
  - Execution
- External Languages
- **Current Status**

Current Status

- The SQL-99 Language is implemented as described...
- ... but no cursors or handlers yet
- ... BEGIN [[NOT] ATOMIC] not yet implemented
- ... not yet the performance it will have (in-memory cache is being implemented now)
- ... attributes and ALTER are being done this month
- ... still only run with the caller's priviledges
- ... external language implementation design is in progress

Questions?

The Source

- Available at:
  - http://mysql.bkbits.net/
  - http://mysql.bkbits.net:8080/mysql-5.0
- clone with:
  - bk clone http://mysql.bkbits.net/mysql-5.0
- mysql-5.0/sql/
  - sp.{h,cc}
  - sp_*.{h,cc}
  - sql_yacc.yy
- mysql-5.0/mysql-test/t/
  - sp.test
  - sp-error.test