Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
Running JavaScript Stored Programs Inside MySQL Server

Øystein Grøvlen
Vojin Jovanovic
Farhan Tauheed

Oracle
April 2018
Program Agenda

1. Stored Programs in MySQL
2. GraalVM: Run Programs Faster Anywhere
3. MySQL MLE: Multilingual Environment in MySQL
4. Project Status and Future Work
5. Q&A
Stored Programs in MySQL

• SQL/PSM
  – Procedural language defined in part 4 of SQL standard

• Functions
  – Called from inside a statement like any built-in function
  – Returns a scalar value
    
    mysql> SELECT f1(c1) FROM t1 WHERE f2(c2) = 1;

• Procedures
  – Invoked by a CALL statement
    
    mysql> CALL myproc('myparam');
  – May return values in output parameters
Why Stored Programs?

• Performance
  – Reduced data volume to client (server-side filtering)
  – Fewer round-trips client/server

• Security
  – Control access to data

• Maintainability
  – Isolation of business rules
  – Easier to understand effects of schema changes
Stored Programs – Disadvantages

• Productivity
  – Unfamiliar programming language
  – Few available 3rd party libraries
  – Poor development tools

• Portability
  – Different language dialects

• Performance
  – Interpreted code

```
create procedure p(x int, out err int, out nulls int)
begin
  declare count int default 0;
  set nulls = 0;
  begin
    declare c cursor for select name from t1;
    declare exit handler for not found close c;
    open c;
    loop
      begin
        declare n varchar(20);
        declare continue handler for sqlexception set err=1;
        fetch c into n;
        if isnull(n) then
          set nulls = nulls + 1;
        else
          set count = count + 1;
          update t2 set idx = count where name=n;
        end if;
      end loop;
  end;
end;
```
User-Defined Functions (UDF)

- Compiled as library files
- Added and removed dynamically

```sql
mysql> CREATE FUNCTION udf_func RETURNS INTEGER SONAME mylib.so;
```

- Written in C/C++
- Supports user-defined aggregate functions

- NB! **Programming errors in function may take down your server!**
  - Memory leaks
  - Segmentation faults
Program Agenda

1. Stored Programs in MySQL
2. GraalVM: Run Programs Faster Anywhere
3. MySQL MLE: Multilingual Environment in MySQL
4. Project Status and Future Work
5. Q&A
GraalVM: Run Programs Faster Anywhere

GraalVM: Completeness and Performance

82.0% 
ECMA Script 2018  
Node.js 8.9.3 LTS scores 81%

99.9% 
Ruby Language  
Comparable to JRuby

Performance relative to: JRuby, GNU R, V8
GraalVM in Native Projects

- Native library for embedding GraalVM
- Unified interface for all languages
- Debugging and profiling support
- High-performance implementation

libpolyglot.so
Native API
Program Agenda

1. Stored Programs in MySQL
2. GraalVM: Run Programs Faster Anywhere
3. MySQL MLE: Multilingual Environment in MySQL
4. Project Status and Future Work
5. Q&A
MySQL MLE: GraalVM in MySQL 8

- Developed as a language plugin
- Only JavaScript in the first release
- Functions and procedures defined in SQL
Demo: "Hello, JS World!" and Stored Functions
MySQL MLE: Module Deployment

- JS has little value without its ecosystem

```javascript
import * as validator from "validator";

function isEmail(input: string): boolean {
    return validator.isEmail(input);
}

module.exports.isEmail = isEmail;
```
Demo: JavaScript Module Deployment
MySQL MLE: Stored Procedures

- MySQL exposes a language-agnostic database connector

- Connector calls directly to the MySQL internals
  - No round-trips for every query
  - No data transfer over the network
  - No need to copy data in memory

```sql
CREATE PROCEDURE word_count_js(OUT count INT) LANGUAGE JS
function() {
  const mle = Polyglot.import('mle');
  const sql = mle.syncConnector;
  const context = { count: 0 };
  const callbacks = {
    getString: function(str, context) {
      context.count += str.split(' ').length;
    }
  }
  sql.execute("SELECT text FROM tweets;", context, callbacks);
  return { count: context.count };}
};$$
MySQL MLE: Client-Code Portability

- Deployment and stored procedures allows client-code running in MySQL
- We developed an npm module called mle:
  - Allows to distinguish between the client and the database
  - Allows fetching the MLE environment, e.g., the MySQL connector
- Implementation for the client MySQL connectors

Module mle

```javascript
const mle = require('mle');
if (mle.enabled()) {
  const connector = mle.env().connector({
    synchronous: true
  });
  connector.execute(...);
}
```

The MySQL Community Connector

```javascript
const mysql = require('mysql');
const conn = mysql.createConnection(...);
conn.connect();
conn.query(
  'SELECT text from tweets;', tokenize);
conn.end();
```
Demo: Stored Procedures and Client-Code Portability
MySQL MLE: Debugging and Tools

• MySQL MLE supports debugging

• Debugging with Chrome DevTools

• Possible support for other instruments
  – Profiling
  – Code coverage
Demo: Debugging Stored Programs with Chrome DevTools
Program Agenda

1. Stored Programs in MySQL
2. GraalVM: Run Programs Faster Anywhere
3. MySQL MLE: Multilingual Environment in MySQL
4. Project Status and Future Work
5. Q&A
Project Status

• Open-source MySQL Labs release

• Support for stored functions and stored procedures

• Support for primitive types and strings

• Basic support for JavaScript client connectors (mysql and mysql2)
Future Work

• Short term
  – Better coverage of the MySQL data types
  – Tighter integration with MySQL: query abort, monitoring, etc.
  – Full support for MySQL JavaScript connectors (mysql and mysql2)

• Mid term
  – Support for Windows
  – Support for all data types

• Long term
  – Support for Python
  – Performance improvements
Demo: "Hello, Polyglot World!"
Conclusions

• Writing stored programs in a familiar language

• High-performance stored programs

• Take MySQL MLE for a spin—we need your feedback:
  https://github.com/graalvm/mysql-mle-demos
Program Agenda

1. Stored Programs in MySQL
2. GraalVM: Run Programs Faster Anywhere
3. MySQL MLE: Multilingual Environment in MySQL
4. Project Status and Future Work
5. Q&A
Integrated Cloud
Applications & Platform Services