JavaOne™

ORACLE®

MySQL Connector/J BOF

Internals and Optimization

CREATE
THE
FUTURE

Jess Balint
Java Specialist Principal Software Developer
MySQL Connectors Team
Sept 30, 2014

JavaOne
ORACLE

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.

Program Agenda

1. Connector/J Introduction
2. Internal Interfaces
3. Feature Support
4. Optimization
5. Resources

Program Agenda

1. Connector/J Introduction
2. Internal Interfaces
3. Feature Support
4. Optimization
5. Resources

What is Connector/J?

- Type-4 JDBC Driver, etc, etc
- Supports most major JDBC features
  - Prepared statements, BLOBs, etc
- A.k.a. C/J for short
- 5.1 branch stable for several years
  - Frequent maintenance releases

Sample of Interesting Connector/J Features

- Client and server prepared statements
  - Client-side aka “emulated” prepared statements
- Load-balanced connections
  - Supports replication, cluster, master-master, etc
  - JMX reporting and management
- Client-side batch insert rewrite
- Hugely configurable via properties, interceptors, custom APIs

Recently Introduced Features

- Pluggable authentication (MySQL 5.5)
  - SHA-256 using SSL or RSA (requires OpenSSL server)
- Support for MySQL Fabric
  - 5.1.30/5.1.31 supports Fabric 1.4
  - 5.1.32+ supports Fabric 1.5
- Connection attributes (MySQL 5.6)
  - Configurable via `connectionAttributes` property
  - Accessible via `performance_schema.session_connect_attrs`

Other Recent Improvements

- Load-balancing improvements
  - Live management of replication connections via JMX or `com.mysql.jdbc.ReplicationConnectionGroupManager` interfaces
  - Multi-master load-balanced connections with `address=(host=hostname)(type=[master|slave])` URL syntax
- Set read-only status on server (MySQL 5.6)
  - `set session transaction read only`
- Several low-level performance improvements in strings and arrays

4 Things You Might Not Know

But should!

- Connector/J supports UTF-8 by default – no special configuration
  - More on this later
- Exceptions are classified by error type
  - C/J-specific classes of JDBC4 exception classes
- Troubleshooting help and profiling facilities are integrated
  - Usage advisor, slow query explanations
- Several extension points available for customization
  - Exception, statement, connection interceptors

Program Agenda

1. Connector/J Introduction
2. Internal Interfaces
3. Feature Support
4. Optimization
5. Resources

Core Classes

Client/server connections

- com.mysql.jdbc.MysqlIO
  - Lowest-level communication with MySQL server
  - Implements client/server wire protocol
- com.mysql.jdbc.ConnectionImpl
  - Builds on MysqlIO to implement connection
  - Contains all properties allowed on connection
    - E.g. getAllowMultiQueries()
    - Inherits from ConnectionPropertiesImpl
  - Creates statements
  - Executes queries (via MysqlIO)

Core Classes

JDBC APIs

- com.mysql.jdbc.DatabaseMetaData
  - Implements JDBC metadata APIs using MySQL “SHOW” commands
  - DatabaseMetaDataUsingInfoSchema
    - Using SQL standard INFORMATION_SCHEMA
    - Enable with `useInformationSchema` property
- com.mysql.jdbc.StatementImpl
  - JDBC Statement operations and functionality used by prepared statements
  - com.mysql.jdbc.PreparedStatement implements client-side prepared statements
  - com.mysql.jdbc.ServerPreparedStatement implements server-side prepared statements
- com.mysql.jdbc.ResultSetImpl

Logical Connection Classes

Replication & Load-balancing

- com.mysql.jdbc.LoadBalancingConnectionProxy
  - JDK dynamic proxy
  - Implements load-balanced connections
  - Proxies JDBC objects (statements, result sets)
- com.mysql.jdbc.LoadBalancedMySQLConnection
  - Connection object used internally to avoid proxy overhead
- com.mysql.jdbc.ReplicationConnection
  - Implements master/slave selection logic

Internal Interfaces

Connection Properties

- com.mysql.jdbc.ConnectionProperties
  - Getter/setter interface for all connection properties
- com.mysql.jdbc.ConnectionPropertiesImpl
  - Private member for each property
    - Property values accessed with accessor/mutator methods
  - Property definitions
  - Base class for most connection classes

Internal Interfaces

Connections

- com.mysql.jdbc.Connection
  - Vendor-specific extension to JDBC Connection interface
  - Documented & stable
- com.mysql.jdbc.MySQLConnection extends ConnectionProperties
  - Internal interface all connection classes implement
  - Exposes all operations internals need to perform on connections
  - Implemented different for physical direct connections (ConnectionImpl) than for logical connections such as load-balanced connections (LoadBalancedMySQLConnection)

Internal Interfaces

Fabric & Load-balanced Connections

- com.mysql.jdbc.LoadBalancedConnection
  - Extends MySQLConnection
  - Interface for load-balanced connections
  - Add/remove hosts
- com.mysql.fabric.jdbc.FabricMySQLConnectionProperties
  - Extends ConnectionProperties
  - Adds additional Fabric-specific connection properties
- com.mysql.fabric.jdbc.FabricMySQLConnection
  - Adds operations relevant to Fabric connections
  - Documented for application use

Extensions

- com.mysql.jdbc.StatementInterceptorV2
  - Allows altering statement execution
  - Pre/post process statement
    - If result is returned in pre-processing, query execution is skipped
  - Callbacks implemented in MysqlIO
- com.mysql.fabric.jdbc.ConnectionLifecycleInterceptor
  - Interceptor receives events at transaction boundaries and connection open/close
- Others
  - ExceptionInterceptor
  - LoadBalanceExceptionChecker
  - BalanceStrategy (load-balancing)

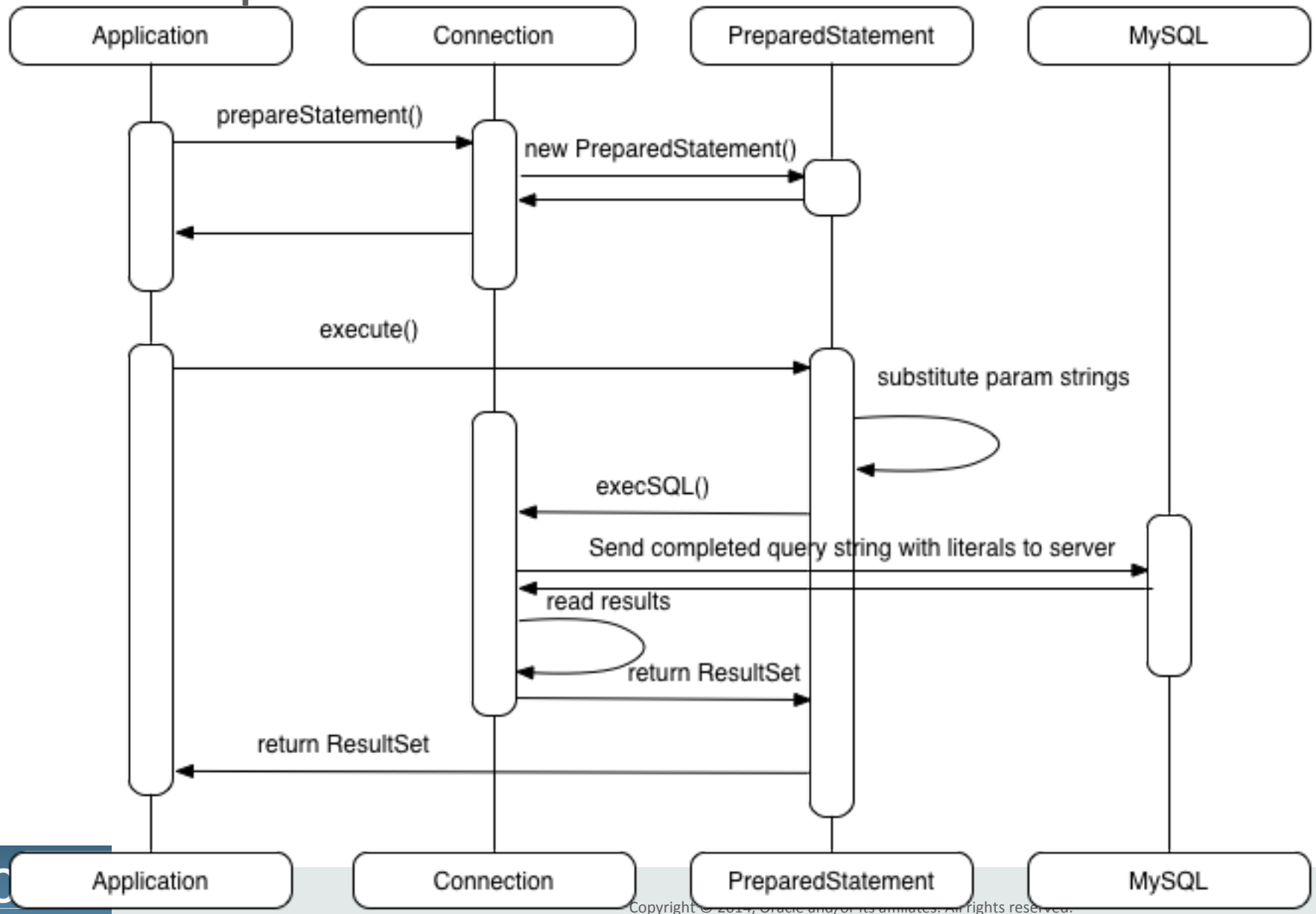
Program Agenda

1. Connector/J Introduction
2. Internal Interfaces
3. Feature Support
4. Optimization
5. Resources

Client-side Prepared Statements and the Text Protocol

- Connection creates a PreparedStatement object on prepareStatement()
- Upon execution:
  - Bound parameters are converted to strings and placeholders in the query string are substituted with string representations
  - Query is sent to the server without any placeholders
- Results are read as text and parsed/converted when requested by the application
  - E.g. resultSet.getInt(1)

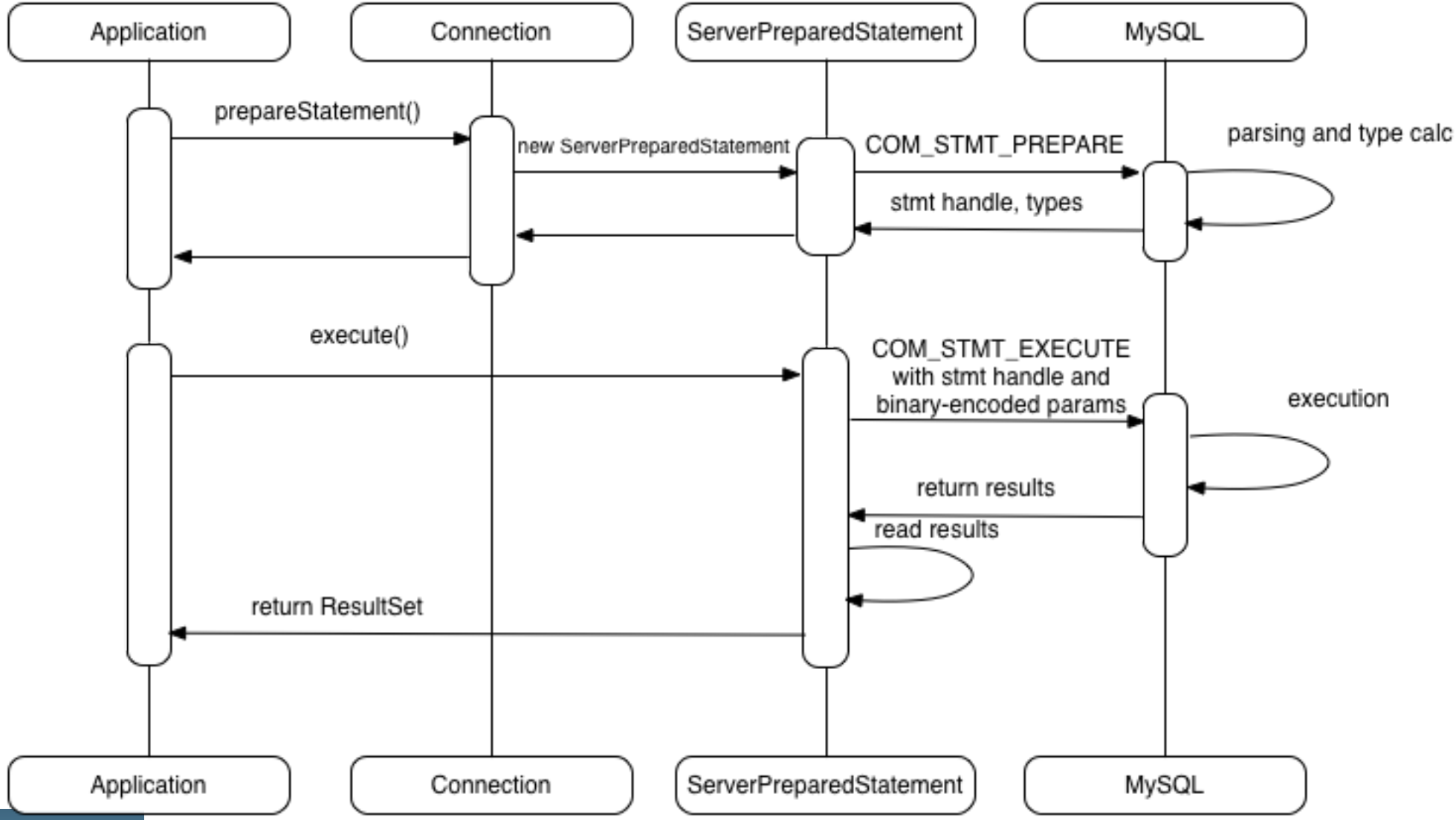
Client-side Prepared Statement Execution



Server-side Prepared Statements and the Binary Protocol

- Connection creates a ServerPreparedStatement object on prepareStatement()
  - Statement object sends the query text with placeholders to the server for parsing
  - Reads statement handle and inferred parameter and result types as response
- Upon execution:
  - Bound parameters are binary-encoded and sent with statement handle for exec
- Results are read in the same binary format as the parameters

Server-side Prepared Statement Execution



JDBC 3 and JDBC 4 – One Jar!

- Classloader unable to resolve JDBC 4 classes if loaded in JDBC 3 env
- Two-stage build
  - JDBC 3 classes compiled with Java 5
  - JDBC 4 classes compiled with Java 6/7/8
- JDBC 4 implementations are subclasses of JDBC 3 implementations
- Runtime selection of implementation via reflection

Internal Caching Opportunities

- Result set metadata cache
  - Types of results indexed by query text
- Prepared statement and callable statement cache
  - Avoids re-preparing same statements across connections
- Server configuration cache
  - Server configuration needed by driver indexed by

Load-balanced Connections

- Logical connection
  - One or more physical connections contained in logical connection
- Evaluates and possibly replaces “current” physical connection at transaction boundaries
  - Configurable balancing strategies
- Statement objects delegate to this logical connection which delegates to the “current” physical connection
- Created statements, result sets, etc are proxied to intercept errors which may trigger failover (changing the physical connection)

Replication Connections

- Logical connection
  - Does NOT implement MySQLConnection
- Uses load-balanced connections internally
  - Both master and slave
- Connection selection is based on readOnly mode of the logical connection
- Runtime configurable via JMX and public methods
  - No separate interface
  - Delegates to runtime configurability of load-balanced connections

Logical Connection Behavior

- Load-balanced connections maintain state on the client (db, tx isolation, etc)
  - Propagated to physical connection when made “current”
- Load-balanced connections are transparent to statements
  - Statements are executed on current physical connection
- Replication connections are NOT transparent to statements
  - Statements are pinned to master/slave connection upon creation

Contributions Encouraged!

- Technical issues can be discussed in the Connector/J forum on forums.mysql.com
- Extensive regression test suite in src/testsuite
- Patches for bugs or small improvements should be submitted on bugs.mysql.com for review
  - Must sign the OCA (Oracle Contributor Agreement)

Program Agenda

1. Connector/J Introduction
2. Internal Interfaces
3. Feature Support
4. Optimization
5. Resources

How to Optimize?

- Awareness of application performance bottlenecks
- C/J and server both profile several useful tools
- Different strokes for different folks
  - Network vs CPU vs memory concerns

What to Optimize?

- Figure out where time is spent
  - Narrow by use case
- C/J and MySQL server both can log slow queries
  - C/J can include an query plan from EXPLAIN
    - `logSlowQueries`, `explainSlowQueries`
- C/J usage advisor can help narrow bottlenecks
  - Find unclosed JDBC objects (connections, statements)
  - Find short-lived connections, single-use prepared statements
  - Find large result sets, unused columns, unnecessary type conversions

Reducing Network Traffic (1)

- Goal is to reduce round-trips and amount of data transferred
- Server-prepared statements
  - Avoid sending query text for repeatedly executed queries
  - Pinned to the physical connection
- Eliminate unnecessary server communication
  - Caching of configuration information (cacheServerConfiguration)
  - Disable/minimize connection attributes (MySQL 5.6+)
- Employ JDBC batching
  - Optionally with rewrites (doesn't help when using server-prepared statements)

Reducing Network Traffic (2)

- Don’t send auto-commit or transaction isolation levels
  - elideSetAutoCommits, useLocalSessionState, useLocalTransactionState, alwaysSendSetIsolation=false
- Use compressed protocol
- Use connection pooling
  - Can generate additional statements
- Use ping for connection pool validation

Reducing Network Traffic
(3)

- Use connection pool PreparedStatement cache

- Use common sense
  - Don’t retrieve more data than necessary – rows OR fields
  - Push data aggregation and filtering to the server
  - Keep transaction size reasonable

Reducing Memory Footprint

- Goal is to reduce the amount of data stored on the client
- Streaming result sets
  - com.mysql.Statement.enableStreamingResults()
- Server-prepared statements (useServerPrepStmts)
- Connection pool configuration
  - Minimize idle connections
- Caching server configurations
- Don’t cache prepared statements unless necessary

Reducing CPU Usage

- Goal is to reduce the amount of work done
- Don’t use compression
- Cache result set metadata
- Binary protocol
- Use and cache prepared statements
  - Client or server side
- Solaris => useConfigs=solarisMaxPerformance

Program Agenda

1. Connector/J Introduction
2. Internal Interfaces
3. Feature Support
4. Optimization
5. Resources

More on C/J

- Browse the codebase
  - There's a lot to learned from getting familiar with the nitty gritty code
- Discuss C/J on the forums
  - http://forums.mysql.com
- MySQL Connector/J Developer Guide
  - Main source of documentation

C/J Blogs

- Todd Farmer – http://mysqlblog.fivefarmers.com
  - Typically focused on load-balance and replication connections
- Mark Matthews – http://www.jroller.com/mmatthews/
  - Description of many C/J features and tips on using them
- Jess Balint – http://blogs.oracle.com/jbalint
  - Connector/J announcements, tips, tutorials
- Connectors Team Java – http://blogs.oracle.com/mysqlconnectors-java
  - New!

JavaOne
ORACLE

JavaOne™

ORACLE®