
ORACLE®

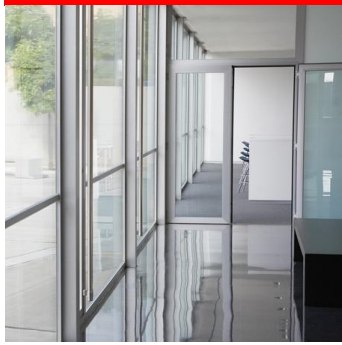
MySQL

ORACLE

Solving Common JDBC Problems with MySQL

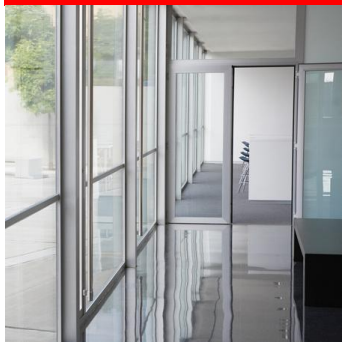
Todd Farmer

Program Agenda



- Common problems
- Recommended Configurations
- High Availability
- Load Distribution
- Extension Points
- Debugging MySQL JDBC Issues

Common Problems



- Classpath Exceptions
- CommunicationExceptions
- Auto-reconnect

Common Problems

Classpath Exceptions

- Most common problem seen in forums
- java.lang.ClassNotFoundException: com.mysql.jdbc.Driver
- Know where your application server expects to find JAR files
  - If using server components (e.g., JDBCRealms in Tomcat) make sure JAR file is in common lib

Common Problems

CommunicationException

- Very common problem
- Most often caused by:
  - Reusing idle (or even closed) connections without checking state
  - Network problems
  - MySQL Server or network configuration
- Connector/J and Server rely on TCP-level keepalive
  - When not actively executing commands, connection can appear “dead” to routers and firewalls

Common Problems

CommunicationExceptions - Debugging

- Upgrade! Versions greater than 5.1.13 have additional diagnostic information in error message

Common Problems

CommunicationExceptions - Debugging

- Upgrade! Versions greater than 5.1.13 have additional diagnostic information in error message

com.mysql.jdbc.exceptions.jdbc4.CommunicationsException: Communications link failure

The last packet successfully received from the server was 44,454 milliseconds ago. The last packet sent successfully to the server was 44,454 milliseconds ago.

Common Problems

CommunicationExceptions - Debugging

- Upgrade! Versions greater than 5.1.13 have additional diagnostic information in error message
- Check wait_timeout and interactive_timeout (if using interactiveClient option)

Common Problems

CommunicationExceptions - Debugging

- Upgrade! Versions greater than 5.1.13 have additional diagnostic information in error message
- Check wait_timeout and interactive_timeout (if using interactiveClient option)
- Confirm that connection pool is testing connections appropriately

  Use /* ping */ EXACTLY to issue lightweight connection check!

Common Problems

CommunicationExceptions - Debugging

- Upgrade! Versions greater than 5.1.13 have additional diagnostic information in error message
- Check wait_timeout and interactive_timeout (if using interactiveClient option)
- Confirm that connection pool is testing connections appropriately
- Double-check socketTimeout is not set (or exceeded if required)

Common Problems

CommunicationExceptions - Debugging

- Make sure any configurable network settings allow long idle times

Common Problems

CommunicationException - Prevention

- Always check connection state when using connections that may have been left idle

Common Problems

CommunicationException - Prevention

- Always check connection state when using connections that may have been left idle
  - Or implement retry logic

Common Problems

CommunicationException - Prevention

- Always check connection state when using connections that may have been left idle
  - Or implement retry logic
- Minimize duration that connections are left idle

Common Problems

Auto-reconnect

- What does autoReconnect/autoReconnectForPools do?

Common Problems

Auto-reconnect

- What does autoReconnect/autoReconnectForPools do?
  - Catch CommunicationExceptions and try to reconnect automatically

Common Problems

Auto-reconnect

- What does autoReconnect/autoReconnectForPools do?
  - Catch CommunicationExceptions and try to reconnect automatically
  - Throw SQLException to the application level

Why?

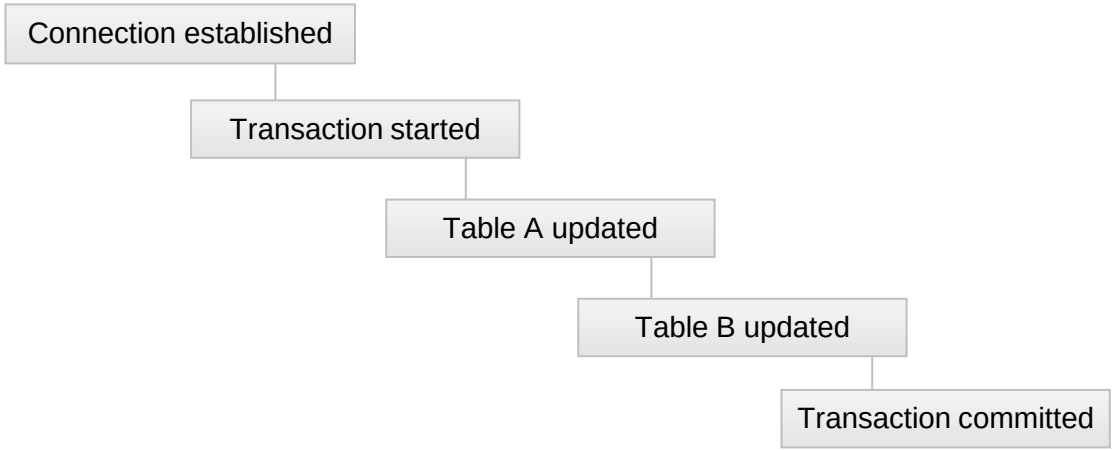
Common Problems

Auto-reconnect

- Session state
  - User variables
  - Temporary tables
  - Session-scoped server variables
- Transactional state
  - Even if auto-commit is enabled!

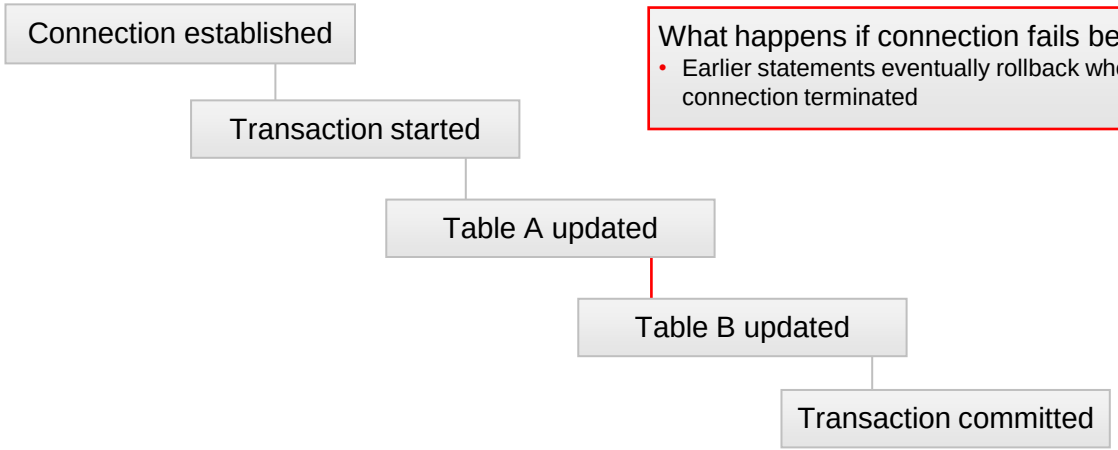
Auto-Reconnect

Example with transactions



Auto-Reconnect

Example with transactions

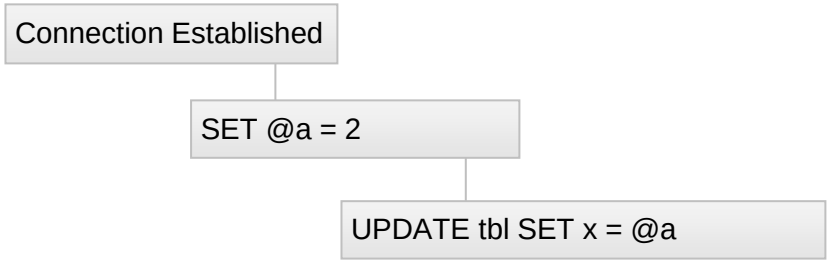


What happens if connection fails between UPDATEs?

- Earlier statements eventually rollback when server notices connection terminated

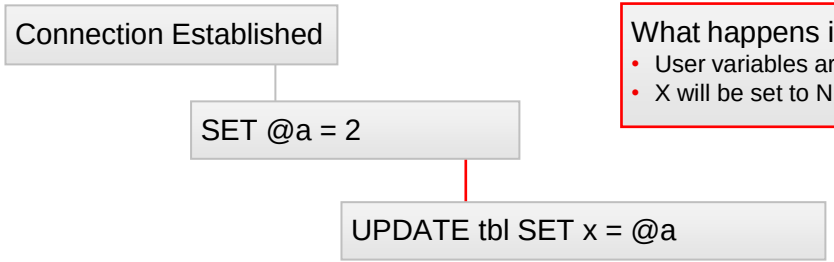
Auto-Reconnect

Example with user variables



Auto-Reconnect

Example with user variables



What happens if connection fails before UPDATE?

- User variables are tied to specific connections
- X will be set to NULL

But I don’t use variables!

Auto-Reconnect

Example with user variables

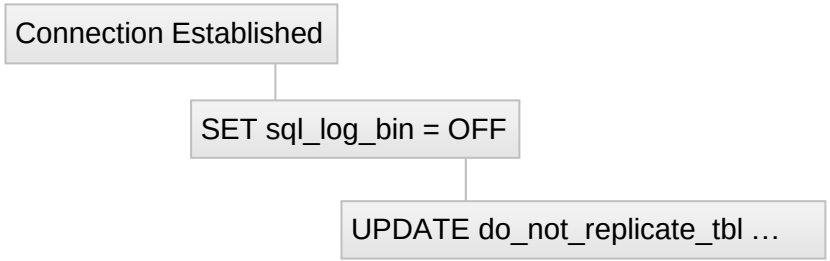
Sometimes the driver has to use variables internally.

Example from server general query log when calling a prepared statement with an INOUT parameter:

```
SET @com_mysql_jdbc_outparam_i='2'
CALL test_proc(@com_mysql_jdbc_outparam_i)
SELECT @com_mysql_jdbc_outparam_i
```

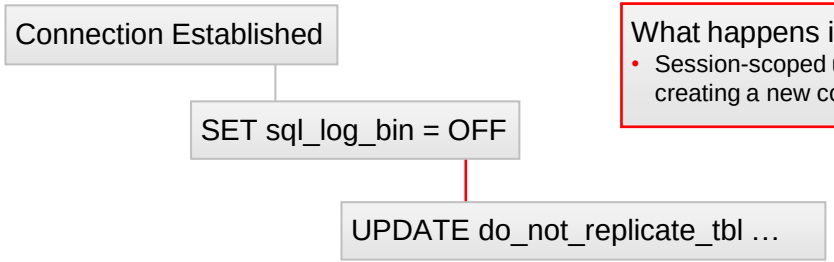
Auto-Reconnect

Example with session-scoped server variables



Auto-Reconnect

Example with session-scoped server variables

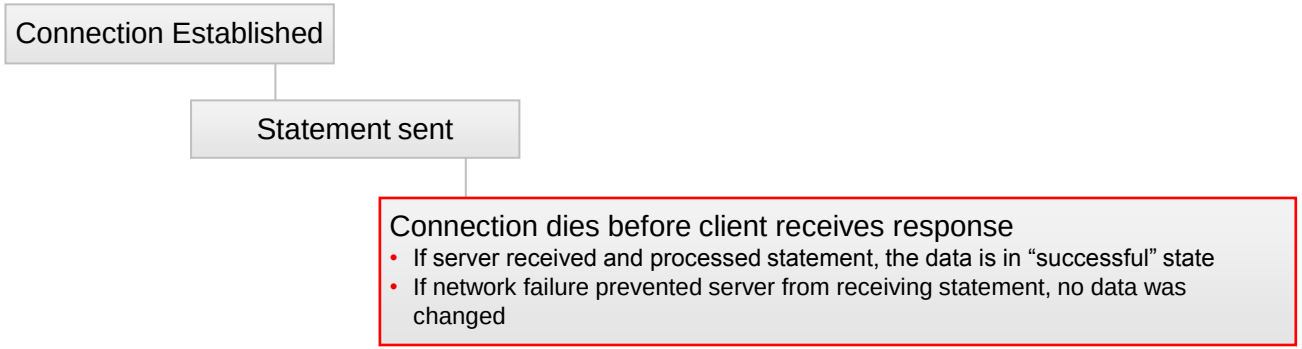


What happens if connection fails before UPDATE?

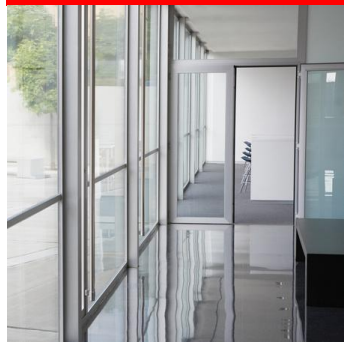
- Session-scoped user variables are reset to defaults when creating a new connection

Auto-Reconnect

Example without transactions

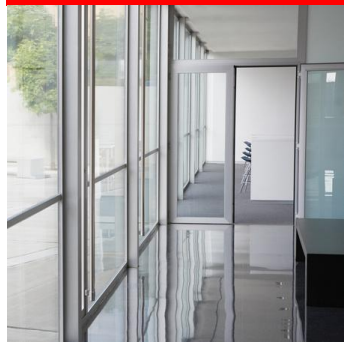


Program Agenda



- Common problems
- Recommended Configurations
- Load Balancing
- Extension Points
- Debugging MySQL JDBC Issues

Recommended Configurations



- Standard Bundles
- Performance-related options
- Network-related options
- Timeout-related options
- Availability options

Recommended Configurations

Standard Bundles

- Found in com/mysql/jdbc/configs directory
  - maxPerformance.properties
  - solarisMaxPerformance.properties
  - fullDebug.properties
  - coldFusion.properties
- Use these as starting point
- Include them with useConfigs=maxPerformance

Performance-Related Options

cachePrepStmts

- Caches parsing for client-side prepared statements
- Caches references to server-side prepared statements
- Caches results of “suitability check” for server-side prepared statements
- Stored in a connection-specific HashMap of configurable size (prepStmtCacheSize, default 25), uses LRU purge
- SQL statements larger than prepStmtCacheSqlLimit (default 256 characters) will not be cached

Performance-Related Options

cacheCallableStmts

- NOT cacheCallableStatements (bug in standard bundle)
- Caches parameter information by schema/SQL
- Stored in a connection-specific HashMap of configurable size (callableStmtCacheSize, default 100), uses LRU to purge (was LRI until fixed in 5.1.18)
- Client pulls CallableStatement metadata from server
  - INFORMATION_SCHEMA
  - SHOW CREATE
  - mysql.proc table

Performance-Related Options

cacheServerConfiguration

- Caches server-side state information
  - COLLATION
  - VARIABLES
- Cache is static (per JVM, not per Connection object)
  - Lives on even if MySQL Server is restarted!
  - Requires application restart if MySQL Server configurations are changed

Performance-Related Options

useLocalSessionState

- Assumes use of standard JDBC operations:
  - setTransactionIsolation()
  - setAutocommit()
  - setCatalog()
- Avoids roundtrip to confirm state with server

Performance-Related Options

elideSetAutoCommits

- Prevents sending auto-commit state command to server if new state matches last server-reported state
  - Earlier Server versions have bug which returns state out of query cache, so not guaranteed to be accurate ☹
- Avoids roundtrip to set state on server

Performance-Related Options

alwaysSendSetIsolation

- Defaults to true
- If set to false, will not set transaction isolation level on server if new state matches existing client state
- useLocalSessionState=true trumps alwaysSendSetIsolation and triggers same behavior

Performance-Related Options

cacheResultSetMetaData

- Caches ResultSet metadata per SQL statement, per connection
- Default cache size is 50, uses LRU (LRI until 5.1.18)

Performance-Related Options

enableQueryTimeouts

- Used in Statement.setQueryTimeout()
  - Starts second thread to establish second physical connection and issue KILL statement against slow connection
- Uses RAM, even when timeout threshold is not reached
- Disable if not using Statement.setQueryTimeout() in high-load environment

Network-Related Options

Push-down options for Java Sockets

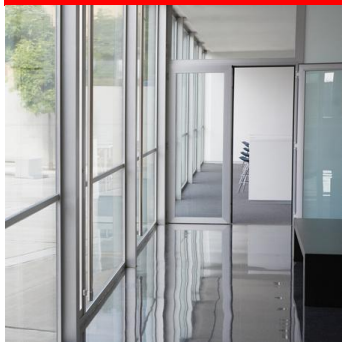
- Generally won't want to change these, unless you have very specific network needs and support:
  - tcpKeepAlive
  - tcpNoDelay
  - tcpRcvBuf
  - tcpSndBuf
  - tcpTrafficClass

Network-Related Options

maxAllowedPacket

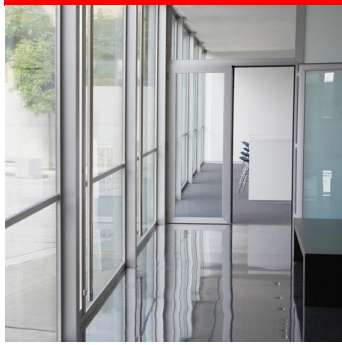
- See server documentation for option:
  - http://dev.mysql.com/doc/refman/5.5/en/server-system-variables.html#sysvar_max_allowed_packet
- Cannot be set larger than server max_allowed_packet
- Set when dealing with large BLOBs, etc.

Program Agenda



- Common problems
- Recommended Configurations
- High Availability
- Load Distribution
- Extension Points
- Debugging MySQL JDBC Issues

High Availability



- Deployment options
- Implementation notes
- Semi-automated failover
- Dual-master, replication ring and active cluster

High Availability

Deployment Options

- How are your MySQL servers deployed?
  - Single server
  - Semi-automated server failover
    - Replication slave promotion
    - O/S HA options
    - Oracle VM or DRBD
    - Windows Cluster
  - Dual master/replication ring/active cluster
- MySQL Server HA whitepapers:
  - http://mysql.com/products/enterprise/high_availability.html

High Availability

Implementation Notes

- HA does not mean never any Exceptions!
- HA means quick to recover and resume processing
- Application needs to be prepared to handle failover situations
  - Implement retry logic, if appropriate
  - See earlier discussion of CommunicationExceptions for details

High Availability

Semi-automated failover

- Only route to secondary server(s) if primary is offline
  - Use jdbc:mysql://master,secondary,etc/ URL format
- Set failOverReadOnly=false if write operations are OK
- Will failover entail creating new Connection objects?
  - If so, leaving autoReconnect=false is fine
- Failing back
  - secondsBeforeRetryMaster and queriesBeforeRetryMaster throttle how often Driver will try master again

High Availability

Dual master, replication ring, active cluster

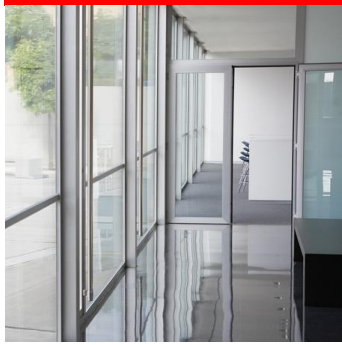
- Use jdbc:mysql:loadbalance://host1,host2 JDBC URL
- failOverReadOnly is useless – all hosts are read/write
- Connection object retains physical connections, max one for each defined host (internal connection pool)
  - Make sure to check connection state after rebalancing opportunity

High Availability

Dual master, replication ring, active cluster

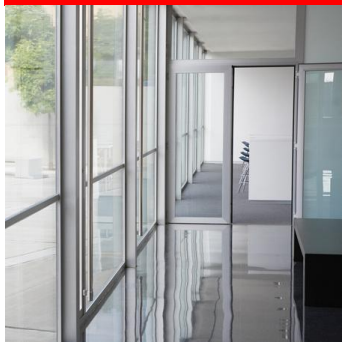
- Host management:
  - loadBalanceBlacklistTimeout keeps hosts from being retried
  - loadBalanceValidateConnectionOnSwapServer checks connections before choosing
  - loadBalanceEnableJMX and loadBalanceConnectionGroup allow live manipulation of hosts
    - Useful to add or remove nodes for scaling or maintenance

Program Agenda



- Common problems
- Recommended Configurations
- High Availability
- Load Distribution
- Extension Points
- Debugging MySQL JDBC Issues

Load Distribution



- Distributing read load
- Distributing read/write load

Load Distribution

Distributing read load

- Useful for offloading read-only load to replication slaves
- Beware of replication lag
- Use Connection.setReadOnly(boolean) to trigger direction to slaves
- Under the hood, ReplicationDriver uses loadbalancing to distribute load between slaves

Load Distribution

Distributing read/write load

- Using loadbalancing option
- Single Connection object is effectively a connection pool
  - Uses one physical connection at a time
  - Chooses new connection at transaction commit/rollback or SQLException
    - Behavior is configurable/extensible with options

Load Distribution

Options

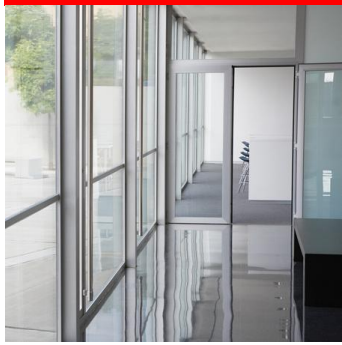
- loadBalanceBlacklistTimeout
- loadBalancePingTimeout
- loadBalanceSQLExceptionSubclassFailover
- loadBalanceSQLStateFailover
- loadBalanceValidateConnectionOnSwapServer

Load Distribution

Options for auto-commit

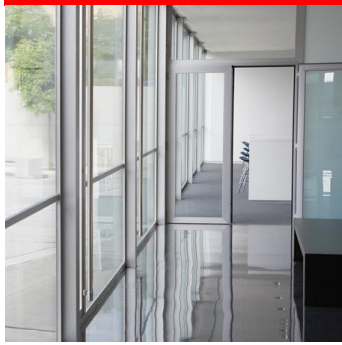
- Useful for distributing read-only replication load when auto-commit is enabled
- Key options (use one, not both!):
  - loadBalanceAutoCommitStatementRegex
  - loadBalanceAutoCommitStatementThreshold

Program Agenda



- Common problems
- Recommended Configurations
- High Availability
- Load Distribution
- Extension Points
- Debugging MySQL JDBC Issues

Extension Points



- Lifecycle Interceptors
- Statement Interceptors
- Exception Interceptors
- Loadbalancing Strategies

Extension Points

Lifecycle Interceptors

- Implement com.mysql.jdbc.ConnectionLifecycleInterceptor
- Intercept JDBC method calls, return false to suppress driver standard behavior
  - Commit()
  - Rollback()
  - setAutoCommit()
  - Close()
  - setCatalog()

Extension Points

Statement Interceptors

- Implement com.mysql.jdbc.StatementInterceptorV2
- Override pre or post-execution method hooks
  - Return null if no changes
  - Return implementation of ResultSetInternalMethods to override actual response from server.
- Useful to change behavior of without application-level changes
- Can be chained for different use cases

Extension Points

Exception Interceptors

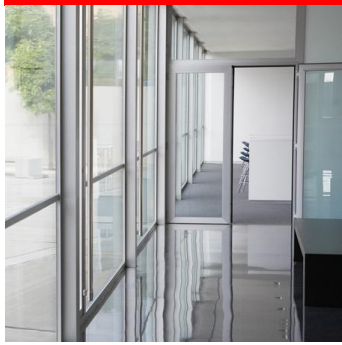
- Implement com.mysql.jdbc.ExceptionInterceptor
- Override interceptException() method
  - Return the SQLException the application should receive
  - Useful to shove additional diagnostic information into error message.

Extension Points

Load balance stragegies

- Implement com.mysql.jdbc.BalanceStrategy
  - Key method is pickConnection(), returning ConnectionImpl
- Connector/J has several examples:
  - RandomBalanceStrategy
  - BestResponseTimeBalanceStrategy

Program Agenda



- Common problems
- Recommended Configurations
- High Availability
- Load Distribution
- Extension Points
- Debugging MySQL JDBC Issues

Debugging MySQL JDBC Issues

Gather performance data from Connector/J

- gatherPerfMetrics
- profileSQL
- slowQueryThresholdMillis
- useUsageAdvisor
- logSlowQueries
- autoSlowLog

Debugging MySQL JDBC Issues

Dealing with Exceptions

- dumpQueriesOnException
- includeInnodbStatusInDeadlockExceptions
- includeThreadDumpInDeadlockExceptions

Debugging MySQL JDBC Issues

Network diagnostics

- traceProtocol – Log network packets

Debugging MySQL JDBC Issues

Outside your Java application stack

- MySQL general query log
- MySQL slow query log
- MySQL Enterprise Monitor
  - Query Analyzer
- PROCESSLIST output
- MySQL variables

Debugging MySQL JDBC Issues

Network diagnostics

- traceProtocol – Log network packets

Resources

- MySQL Java forums:
  - http://forums.mysql.com/list.php?39
  - http://forums.mysql.com/list.php?46
- MySQL Connector/J documentation:
  - http://dev.mysql.com/doc/refman/5.5/en/connector-j-reference-configuration-properties.html
  - http://dev.mysql.com/doc/refman/5.5/en/connector-j-reference.html
- http://mysqlblog.fivefarmers.com

Q&A

Hardware and Software

Engineered to Work Together


ORACLE®