



IBM® DB2® Driver for JDBC and SQLJ

John Vonau



© Copyright IBM Corporation 2007

Disclaimer and Trademarks

The information contained in this presentation is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided “as is” without warranty of any kind, express or implied.

Information in this presentation about IBM's future plans reflect current thinking and is subject to change at IBM's business discretion and/or without notice. You should not rely on such information to make business plans. The use of this information is a customer responsibility.

IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other documentation. Nothing contained in this presentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM (or its supplies or licensors), or altering the terms and conditions of the applicable license agreement governing the use of IBM Software.



Disclaimer and Trademarks

IBM may have patents or pending patent applications covering subject matter in this presentation. The furnishing of this presentation does not imply giving license to these patents or patent applications.

IBM, the IBM logo, DB2, z/OS, Cloudscape, and DB2 Connect are trademarks of registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.



Disclaimer and Trademarks

The opinions, solutions, and advice in this presentation are from the author's experiences and are not intended to represent official communication from IBM or an endorsement of any products listed within. Neither the author nor IBM is liable for any of the contents in this article. The accuracy of the information in this article is based on the author's knowledge at the time of writing.

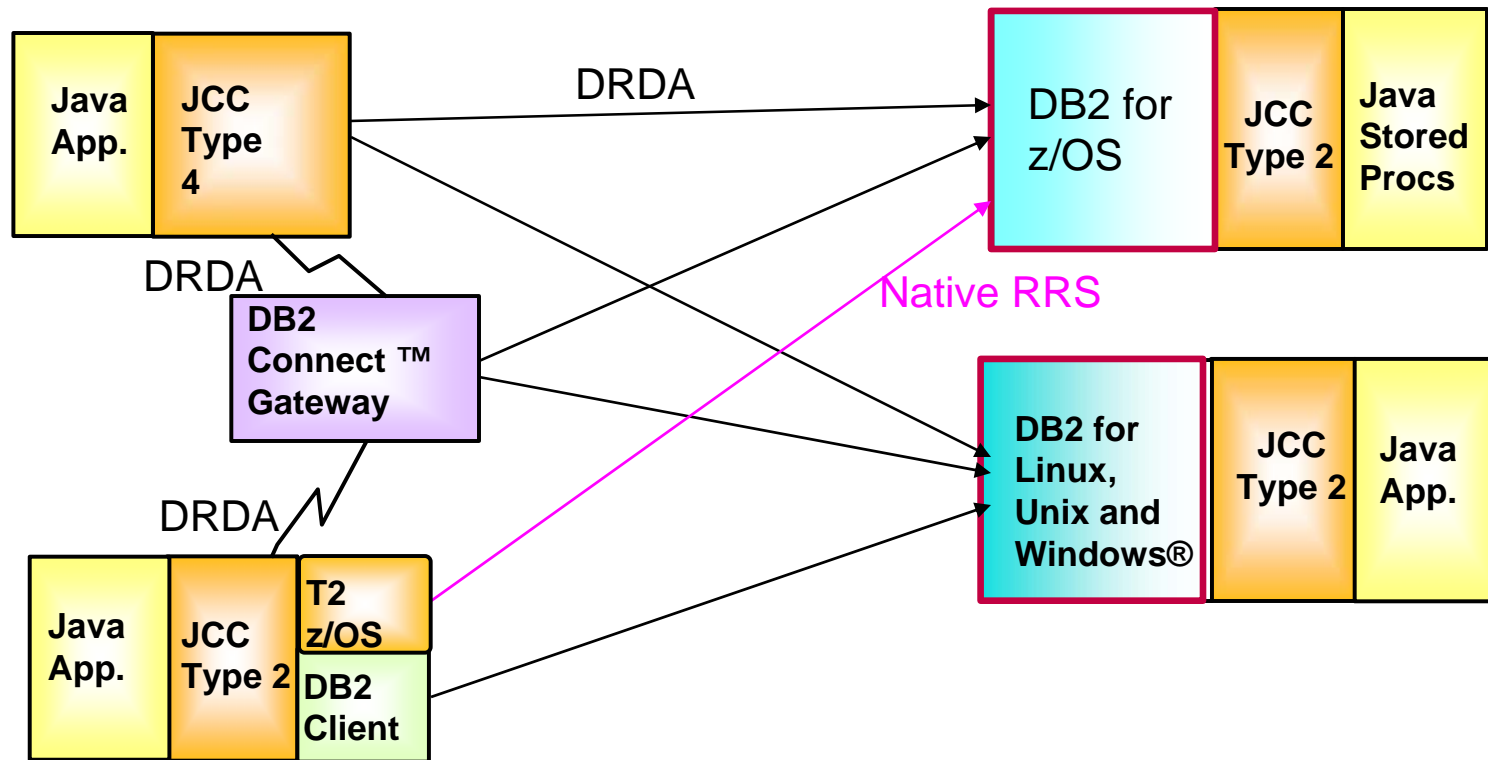


Introduction

- JDBC or SQLJ Access to Data
- Connectivity to various target servers: DB2, Cloudscape™
- Type 4 Connectivity
 - Distributed Relational Database Architecture (DRDA); pure Java™
- Type 2 connectivity
 - z/OS® Local Connectivity: Java and native C/C++. Extremely fast
 - Type 2 LUW Local and Remote Connectivity: DRDA
 - Also needed for Java stored procedures
- Direct access without gateway
- Very compact and supports light install
- Integrated closely with WAS and tools



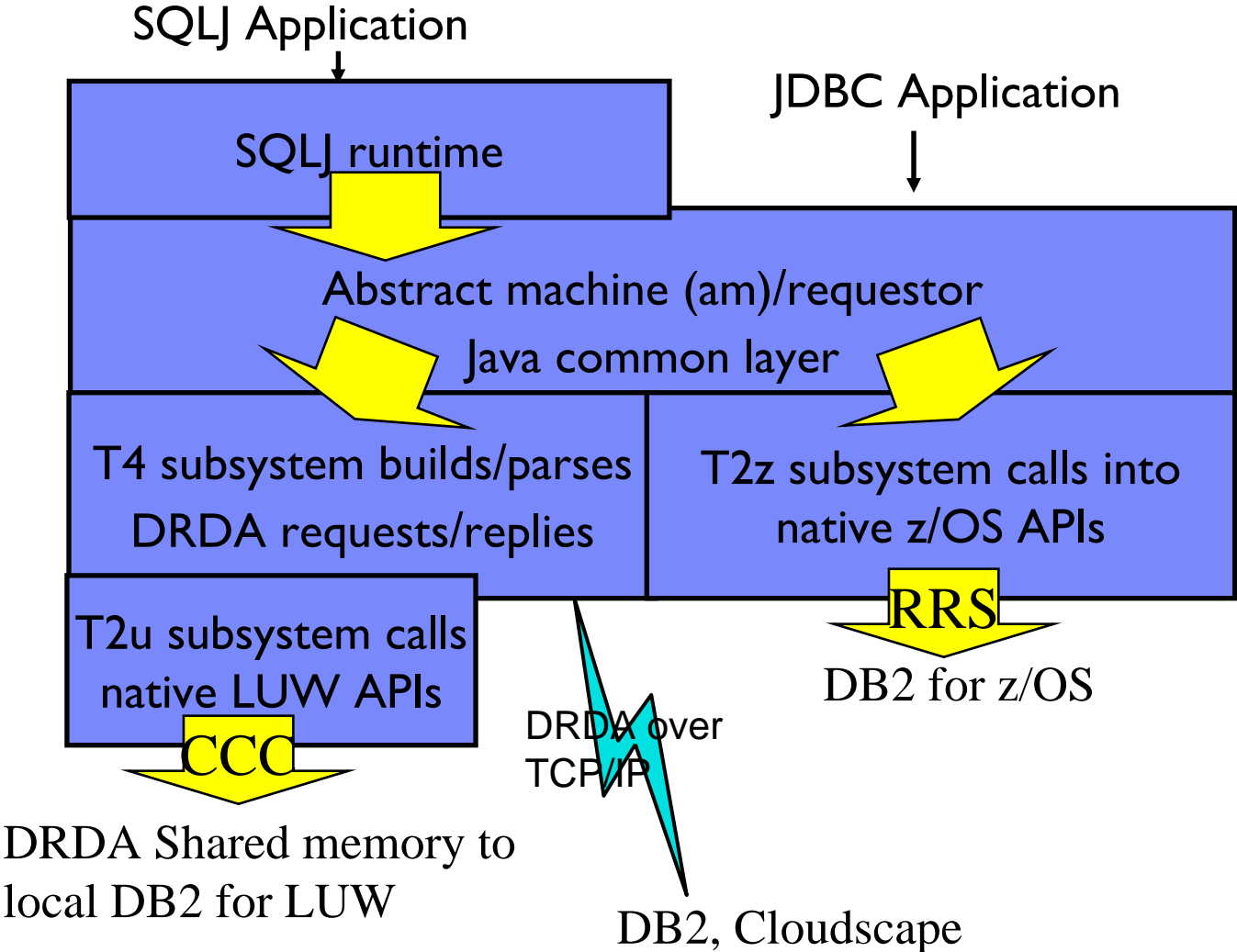
Typical Usage



Java App. - Java application or AppServer

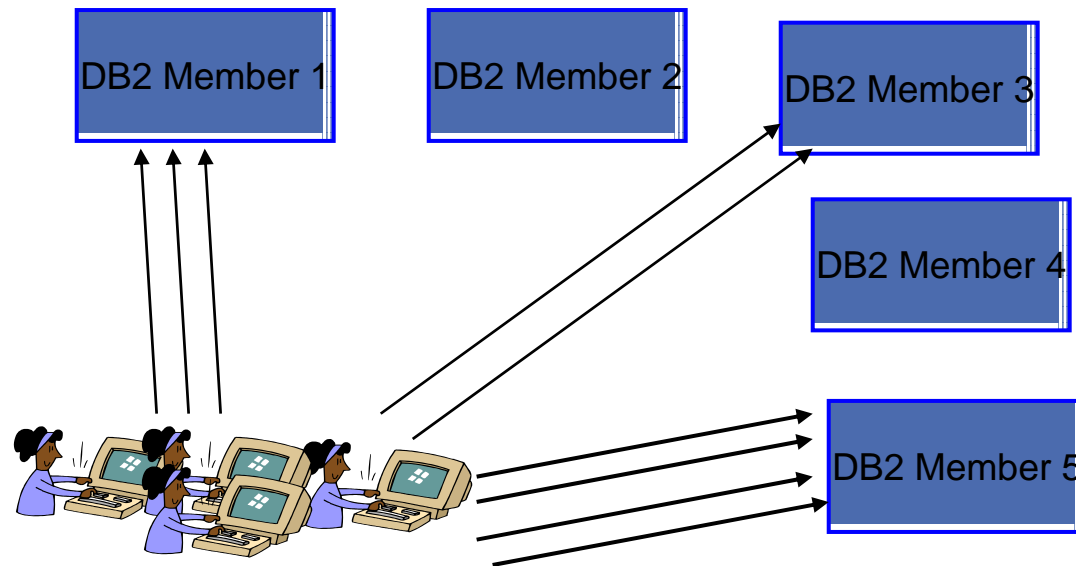


Driver Architecture



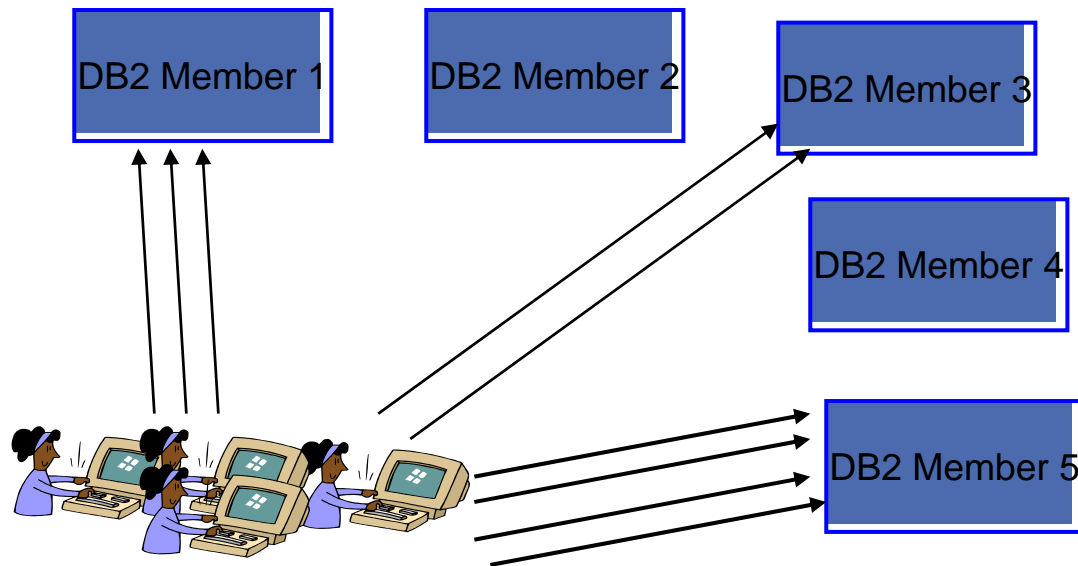
DB2 for z/OS Sysplex Balancer

- High Availability feature added for Type 4 connectivity in DB2 LUW v8.1 FP10
- The driver maintains physical connections, known as transports, to the various members of a DB2 for z/OS parallel sysplex
- Using feedback received from WLM, the driver balances work for new transactions across the data sharing members by associating connections with various transports
- Rebalancing to another member will occur in case of a member failure and is transparent if the failure occurs on a transaction boundary



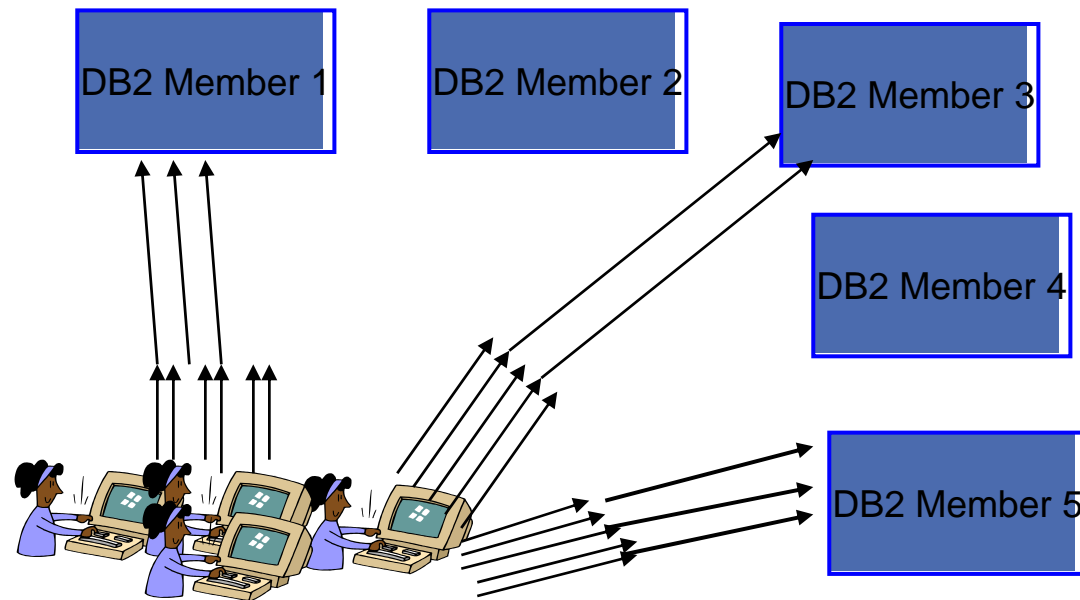
DB2 for z/OS Sysplex Balancer

- Enabled via the DataSource property enableSysplexWLB
- Transport object pool is configured by
 - DataSource property - maxTransportObjects
 - Global Properties
 - db2.jcc.maxTransportObjects
 - db2.jcc.minTransportObjects
 - db2.jcc.maxTransportObjectIdleTime
 - db2.jcc.maxTransportObjectWaitTime



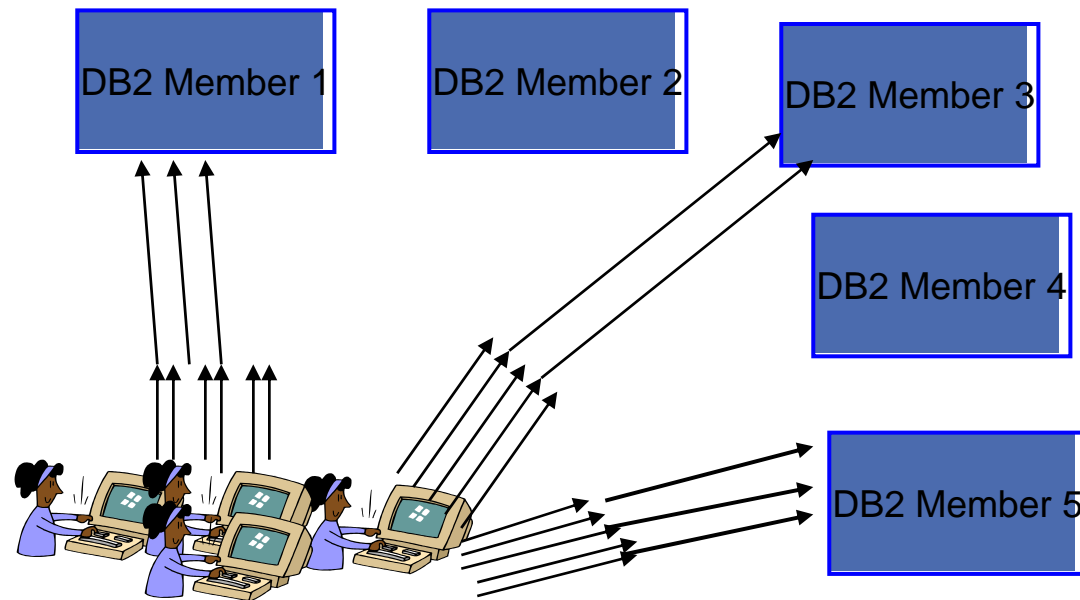
DB2 for z/OS Connection Concentrator

- High Availability feature added for Type 4 Connectivity in DB2 LUW v8.1 FP10
- Allows a given JVM to map multiple connections onto a smaller number of physical connections known as transports
- Helps reduce total number of physical connections to a database server
- Automatically associates connection with a new transport in case of a connectivity failure
- Failure is transparent if it occurs on a transaction boundary



DB2 for z/OS Connection Concentrator

- Enabled via the DataSource property enableConnectionConcentrator
- Level of concentration is determined by configuring the transport object pool
 - DataSource property - maxTransportObjects
 - Global Properties
 - db2.jcc.maxTransportObjects
 - db2.jcc.minTransportObjects
 - db2.jcc.maxTransportObjectIdleTime
 - db2.jcc.maxTransportObjectWaitTime
- Connection Concentrator and Sysplex Balancer may be enabled at the same time



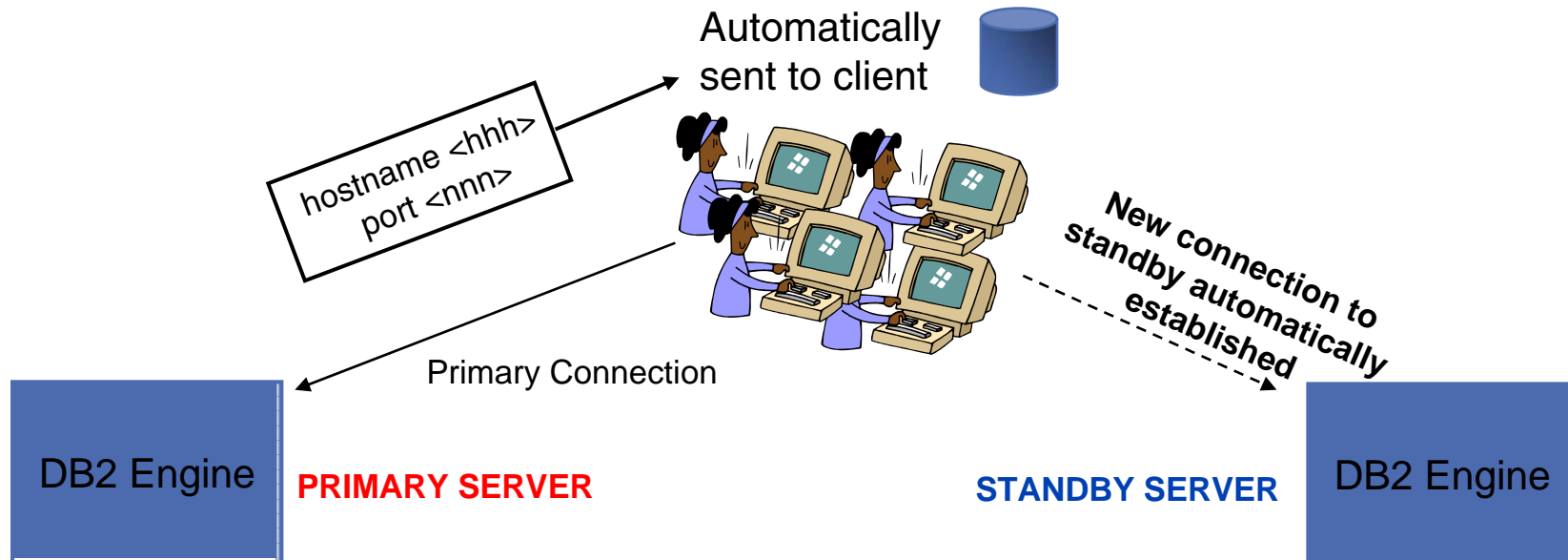
Transport Object Pool Monitoring

- Traces can be enabled to assist with monitoring the pool
- A pool statistics trace is enabled and configured via the following global properties
 - db2.jcc.dumpPool
 - db2.jcc.dumpPoolStatisticsOnSchedule
 - db2.jcc.dumpPoolStatisticsOnScheduleFile
- The DB2PoolMonitor can also be obtained and used
 - DB2PoolMonitor monitor =
DB2PoolMonitor.getPoolMonitor(DB2PoolMonitor.TRANSPORT_OBJECT)
- DB2PoolMonitor methods:
 - int getMonitorVersion()
 - int totalRequestsToPool()
 - int successfulRequestsFromPool()
 - int numberOfRequestsBlocked()
 - long totalTimeBlocked()
 - int lightWeightReusedObjectCount()
 - int heavyWeightReusedObjectCount()
 - int createdObjectCount()
 - int agedOutObjectCount()
 - int removedObjectCount()
 - int totalPoolObjects()



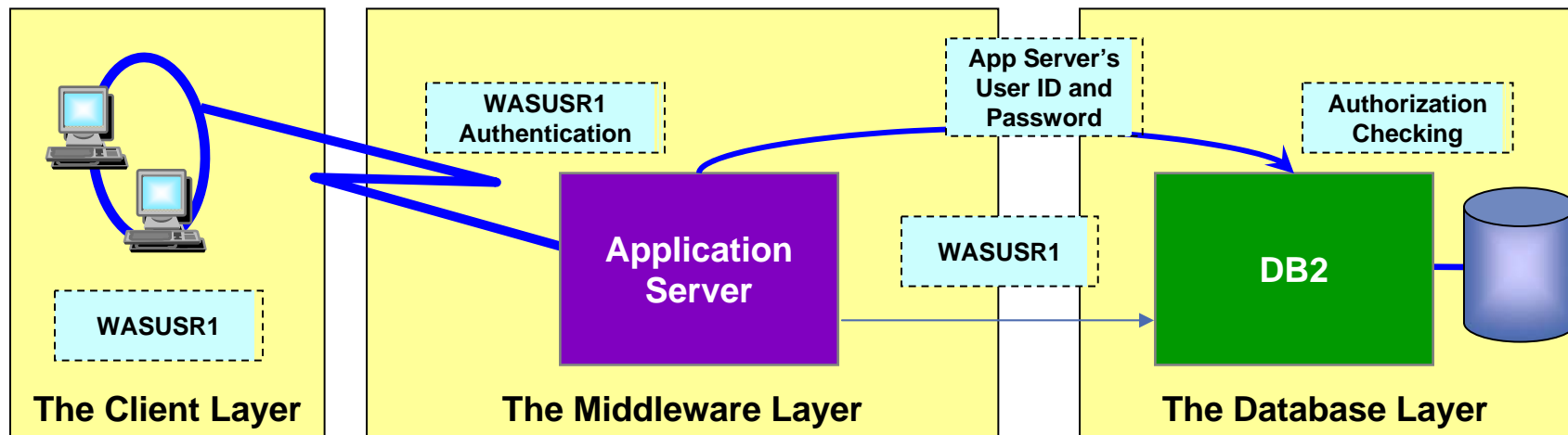
Client Reroute

- Automatic, transparent connection to alternate server when primary connection fails
 - If there is a currently executing SQL statement, it will fail with -4498
 - Transaction can then be re-driven without re-establishing a connection
- Alternate information updated on client
 - System database directory for CLI
 - JNDI and/or driver memory for JCC t4
 - DNS is not updatable but may also be used for alternates



Trusted Authentication in a Three-tier Architecture

- The application server's user ID and password are used to establish the trusted connection.
- The user is switched in the trusted connection and client user ID is propagated to the server
- The client authorization ID's privileges are checked for database access



Progressive Streaming

- LOBs can be either fully materialized or streamed
- Feature is enabled by default when supported by target server
- Explicit enablement via the property `progressiveStreaming`
- The `streamBufferSize` property helps determine when LOBs are fully materialized or streamed. `streamBufferSize` default is 1M
 - LOB size \leq `streamBufferSize`
 - Fully-materialized data
 - Server returns entire LOB data in one flow
 - LOB size $>$ `streamBufferSize`
 - Streamed data
 - Server returns a subset of the LOB data in multiple flows
 - `streamBufferSize` is the data size
- Streaming data is cursor based
- Streamed data is sequential-access based



Progressive Streaming (continued)

fullyMaterializeLobData		true	false
		progressiveStreaming	
YES or NOT_SET	data size <= streamBufferSize	Fully-materialized data	Fully-materialized data
	data size > streamBufferSize	Streaming data	Streaming data
NO		Fully-materialized data	Locators

(YES, NO, and NOT_SET are constants defined on DB2BaseDataSource)



XML Data

- No standard support for XML as of JDBC 3.0
- PreparedStatement setters can be used with XML Data
 - PreparedStatement.setBytes(1, xmlBytes)
- ResultSet getters can be used with XML Data
 - ResultSet.getBinaryStream(1)
- DB2Xml is introduced for use with XML Data
 - Data without declaration: DB2Xml.getDB2String()
 - Data with declaration: DB2Xml.getDB2XmlString()
 - Data with UTF-8 encoding: DB2Xml.getDB2Bytes()
 - Data with other encoding: DB2Xml.getDB2XmlBytes("EUC-JP")



Decimal Floating Point

- Decimal floating point is supported in JRE 5.0 as part of BigDecimal
- Access to a DECFLOAT column requires JRE 5.0
- DECFLOAT is mapped to BigDecimal

SQL	DECFLOAT(16)	DECFLOAT(34)
Java	BigDecimal with MathContext.DECIMAL64	BigDecimal with MathContext.DECIMAL128



Decimal Floating Point Rounding Mode

- Property `decimalRoundingMode` configures rounding mode used with DECFLOAT
- `ROUND_UNSET` (default) indicates no explicit setting
 - `ROUND_HALF_EVEN` will be used
 - No attempt to sync-up with server rounding
- If `decimalRoundingMode` is set to one of the five supported rounding modes, `SET CURRENT DECFLOAT ROUNDING MODE` statement will be flowed to DB2
- Continue to truncate (round down) on DECIMAL to match the DB2 behavior



BIGINT, BINARY, and VARBINARY

- **BIGINT**
 - An eight-byte integer
 - Range: -9223372036854775808 to 9223372036854775807
 - Mapped to long or Long in Java
- **BINARY**
 - Fixed-length
 - Range: 1 to 255 bytes
 - Mapped to byte[] in Java
- **VARBINARY**
 - Varying-length
 - Range: 1 to 32704 bytes
 - Mapped to byte[] in Java

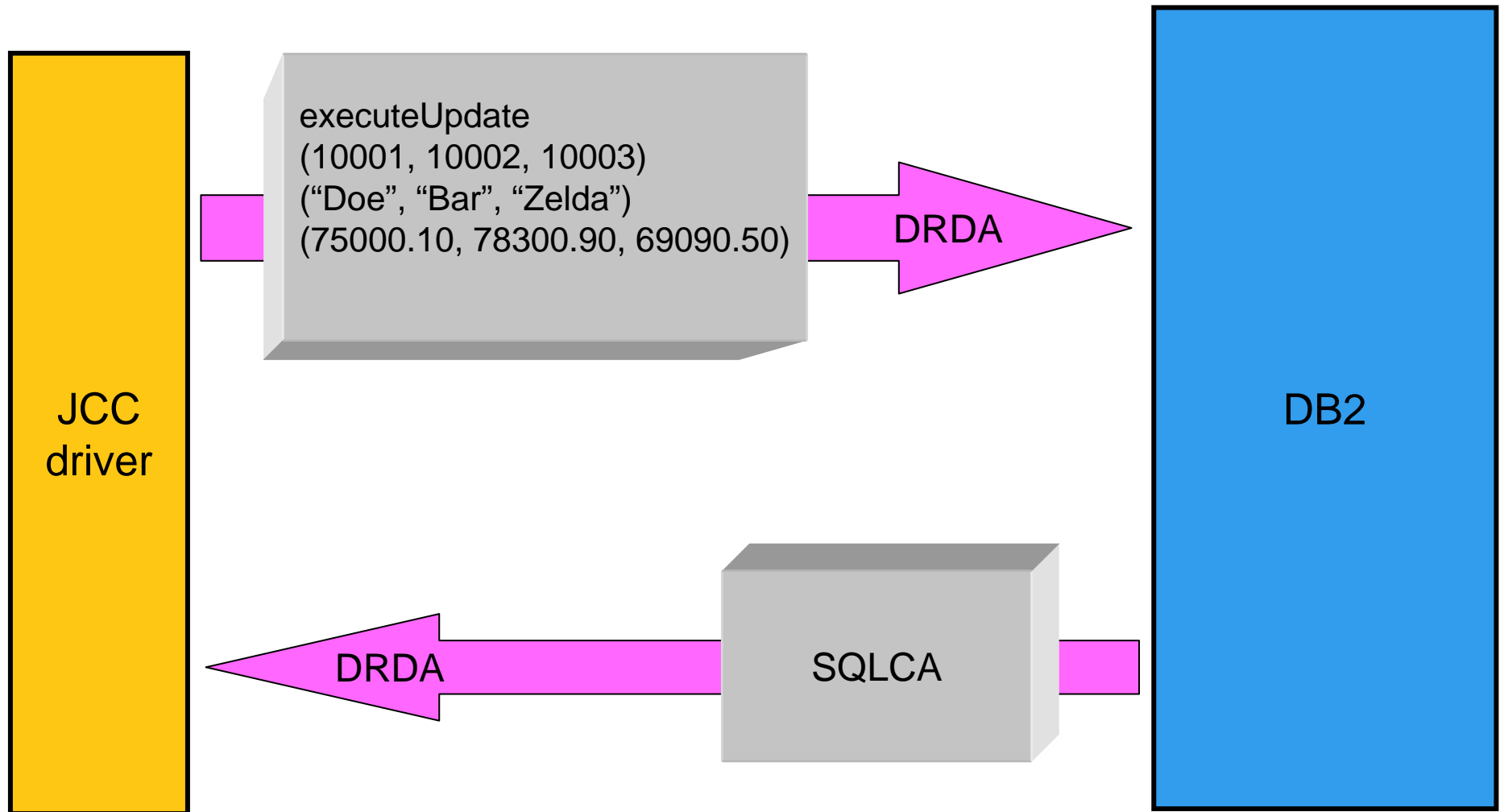


Date/Time Formats

Format	Date	Time
ISO	yyyy-mm-dd 2007-02-08	hh:mm:ss 15:00:23
JIS	yyyy-mm-dd 2007-02-08	hh:mm:ss 15:00:23
USA	mm/dd/yyyy 02/08/2007	hh:mm (am/pm) 03:00 pm
EUR	dd.mm.yyyy 08.02.2007	hh.mm.ss 15.00.23



Multi-row Insert



Multi-row Fetch

- Enabled by default for scrollable cursors
- Explicit enablement via `useRowsetCursor` property
- When enabled, driver uses multi-row fetch for scrollable cursors if the feature is also supported by the target server
- Uses Prepare attributes:
 - “WITH ROWSET POSITIONING”
- Property access methods
 - `public void setUseRowsetCursor (boolean useRowsetCursor);`
 - `public boolean getUseRowsetCursor ();`



Tracing

- Modified time format facilitates diagnostics
 - Old: 1054850798671
 - New: 2004-03-16-14.44.19.427
- Trace contains LUWID information
- Trace includes system monitor information
 - Previously available from DB2SystemMonitor
 - TRACE_SYSTEM_MONITOR traceLevel

**[ibm][db2][jcc][SystemMonitor:stop] core: 565.67ms | network:
211.695ms | server: 207.771ms**



Additional Highlights

- Security Enhancements
- Enhanced Binder Options
- Support for IPv6
- Ability to detect active connection
- XA Enhancements
- Enhanced support for Call statement literals
- Transaction Isolation Property
- Generic Package Utility





Enhancements For DB2 Viper 2

John Vonau



© Copyright IBM Corporation 2007

Enhancements – Viper 2 Focus

- JDBC 4 compliance – new driver JAR
- IDS support
- Effective support of new server features
- New driver functionality
- Supportability improvements and others

JAR file	Driver version	Level of JDBC support	Minimum level of Java
db2jcc.jar	JCC 3.50	JDBC 3.0 and earlier	1.4
db2jcc4.jar	JCC 4.0	JDBC 4.0 and earlier	1.6



JDBC 4 Enhancements – SQLXML, ROWID

- SQLXML interface for XML data manipulation
- Get/Set support for SQLXML

```
SQLXML sqlxml = resultSet.getSQLXML(1);
preparedStatement.setSQLXML(1, sqlxml);
```
- Support for RowId Interface
- Get/Set support for RowId

```
RowId rowid = resultSet.getRowId(1);
byte[] bytes = rowid.getBytes();
ps.setRowId(1, rowid);
```
- DatabaseMetaData getRowIdLifetime() indicates RowId lifetime



JDBC 4 Enhancements - LOBs

- Creation of large objects through factory methods

```
Clob clob = conn.createClob();
```

- Partial large object retrieval as a stream

```
Reader reader = clob.getCharacterStream (position,  
length);
```

- Allow large object setting without a length

```
preparedStatement.setCharacterStream (1, reader);
```

- Explicit release of resources

```
clob.free();
```



JDBC 4 Enhancements - Client Information

- Associate client-specific information with a connection

```
conn.setClientInfo("ApplicationName", "fooApp");  
String app = conn.getClientInfo("ApplicationName");
```

- Monitoring tools can display this information for pinpointing the problematic connection

- Retrieve a list of supported properties

```
ResultSet rs = conn.getMetaData().getClientInfoProperties();  
while (rs.next()) {  
    String name = rs.getString(1);  
    int max_len = rs.getInt(2);  
    String default_value = rs.getString(3);  
    String description = rs.getString(4);  
}
```



JDBC 4 Enhancements - Connection Validation

- Determine if a connection is still alive
- The connection pool manager may take advantage of this information

```
do {
```

```
    Connection c = ...; // get a connection from pool
```

```
    if (c.isValid(timeout)) return c;
```

```
} while (hasNextConnectionInPool)
```

- Similar to proprietary `isDB2Alive()`



JDBC 4 Enhancements - Statement Events and Pooling Hints

- Statement Events
 - A listener is notified of statement closing and statement error events
 - The event contains the statement, its associated connection, and the exception about to be thrown
 - Statement Pool Hints
 - Provides a hint to the statement pool manager about whether a statement should be pooled or not
 - By default, when created, Statement is not poolable, and PreparedStatement and CallableStatement are poolable
- statement.[setPoolable](#)(true);
- Boolean poolable = statement.[isPoolable](#)();



JDBC 4 Enhancements - Wrapper Interface

- Vendor-specific resources may be wrapped in some environments, such as in an application server
- The relationship between a wrapper and a underlying resource might be inheritance, containment, or else
- Gain access to the vendor-specific resource in a standard way
- Most JDBC interfaces, such as Statement, ResultSet, and Connection, now extend the Wrapper interface

```
if (s.isWrapperFor(DB2Statement.class)) {  
    DB2Statement ds = s.unwrap(DB2Statement.class);  
    ds.setDB2ClientProgramId("...");  
}
```



JDBC 4 Enhancements - Service Provider Mechanism

- No need to load a driver using `Class.forName` method
- Auto-loading of a JDBC driver is through the service provider mechanism
 - A driver implementation needs to provide a `META-INF/services/java.sql.Driver` file in the driver JAR file, such as `db2jcc4.jar`
 - The file contains a single line to identify the driver implementation class, such as `com.ibm.db2.jcc.DB2Driver`



JDBC 4 Enhancements - SQLException

- For-each loop support allows SQLException navigation without coding getNextException()
for (**Throwable e : sqle**)
 e.printStackTrace();
- SQLNonTransientException: A retry of the same operation would not succeed
 - 6 subclasses of SQLNonTransientException
- SQLTransientException: A retry of the same operation might succeed
 - 3 subclasses of SQLTransientException



New Server Features – ARRAY

- Array objects from Connection factory methods

```
String[] nameStr = {"John", "Doe"};  
java.sql.Array name = c.createArrayOf("VARCHAR", nameStr);
```
- Invoke a stored procedure with ARRAY parameters

```
callableStatement.setArray(1, name);  
callableStatement.registerOutParameter(2, java.sql.Types.ARRAY);  
callableStatement.execute();  
java.sql.Array phones = callableStatements.getArray(2);
```
- Retrieve the Array object as a Java array

```
String[] phonesStr = (String[])phones.getArray();
```
- Retrieve the Array object as a ResultSet

```
java.sql.ResultSet phonesResult = phones.getResultSet();
```



New Server Features - Optimistic Locking

- Convenience methods introduced to obtain RID and ROW CHANGE TIMESTAMP

```
DB2ResultSet rs =  
    (DB2ResultSet)s.executeDB2OptimisticLockingQuery("SELECT *  
    FROM employee WHERE id = 1",  
    DB2Statement.RETURN_OPTLOCK_COLUMN_ALWAYS);  
rs.next();  
Object rid = rs.getDB2RID();  
long rct = rs.getDB2RowChangeToken();
```

- Locks are released and row is later updated using RID and ROW CHANGE TIMESTAMP



New Server Features – Additional Items

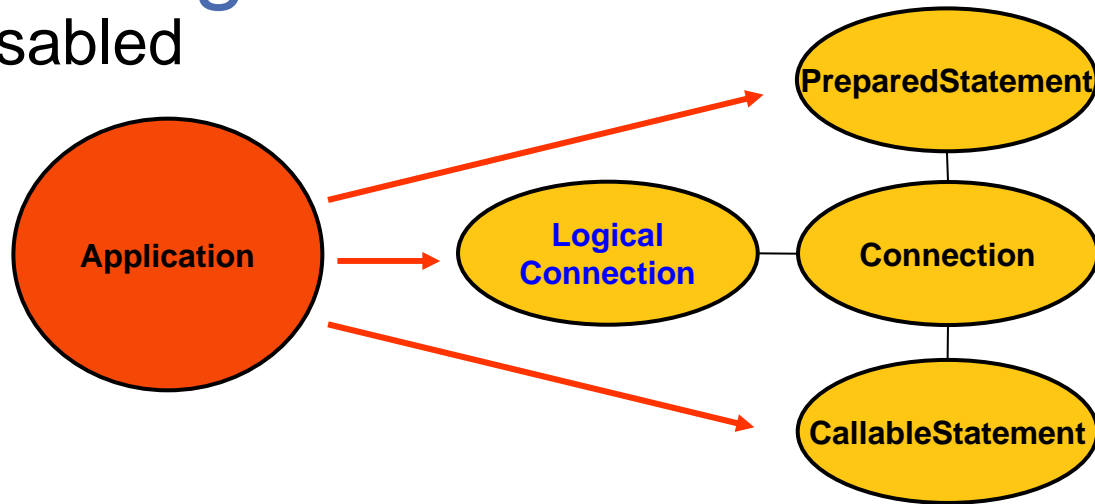
- Driver is now default for DB2 LUW stored procedure
- Long Identifier Support
- XML Data Encryption
- API for XML Schema Updates
- Progressive Streaming against DB2 LUW
- DECFLOAT support against DB2 LUW
- DNS Support for Client Reroute



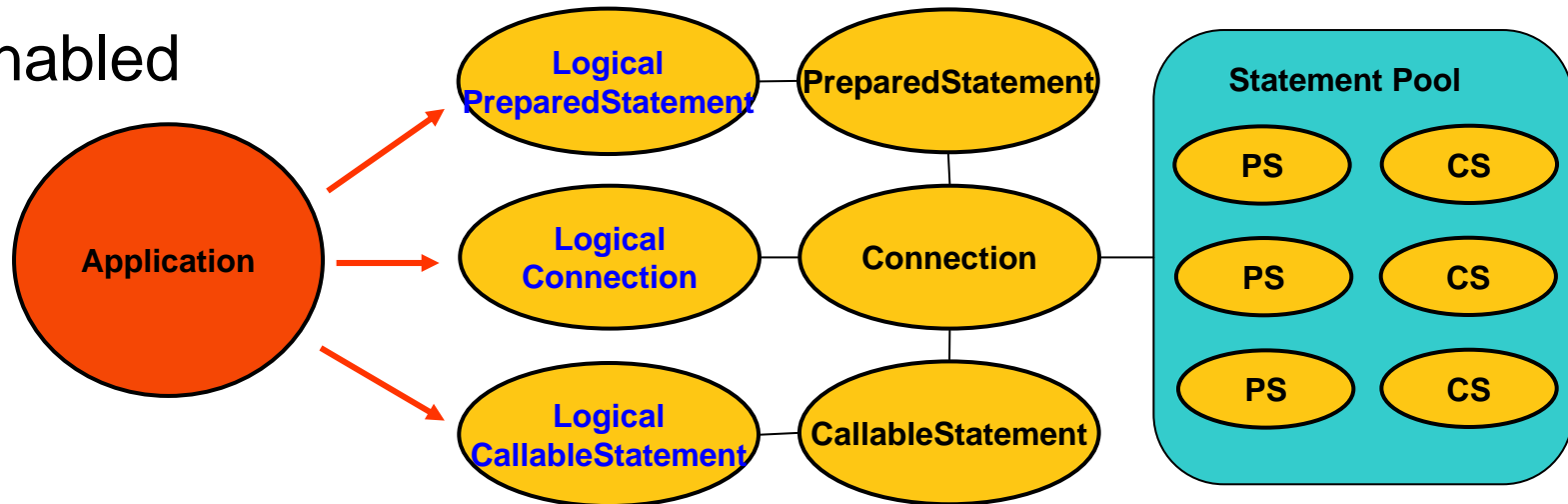
New Driver Functionality - Statement

Pooling

Disabled



Enabled



New Driver Functionality - Binder Enhancements

- Ability to drop the JDBC packages *intelligently*
- A negative code is now returned if DB2Binder fails
 - First digit identifies the error category
 - 1: Missing bind option
 - 2: Invalid bind option
 - 3: Unsupported bind option
 - 4: Initialization issues
 - 5: Bind issues
 - Next two digits identify the reason that caused this error
 - -101: “url” option was not specified
 - -202: “user” value is invalid
 - -310: “reopt” option is not supported by the target server
 - -403: Database metadata could not be retrieved
 - -502: An existing package is not valid
- A new runtime API is first introduced in JCC 3.50 on DB2Binder to *programmatically* add, replace, and drop the JDBC packages



New Driver Functionality – Additional Items

- insertRow() API
- ResultSet LOB updates
- Driver manager support for sysplex and client reroute
- Get local catalogued databases API
- Property to override resultSetHoldability of metadata ResultSets
- XA transaction timeout API



Supportability and Other Features

- Supportability improvements
 - Remote trace controller
 - Trace Enhancements – new trace levels
- Miscellaneous
 - Define SQL codes and SQL states for all driver error messages vs. -99999
 - JDBC 4 SQL State Mappings
 - Alternative semantics for `getColumnName()` and `getColumnLabel()`

