



I127/DBP08

IBM STG Technical Conference

Optimize Access to DB2 for i5/OS from Java and WebSphere

Jarek Miszczyk
DB2 for i5/OS Technology Team
IBM Rochester, MN

© 2007 IBM Corporation

IBM STG Technical Conference



Agenda

- DB2 JDBC drivers
 - IBM drivers that can be used to access DB2 for i5/OS
 - Choosing the driver for iSeries
- IBM JDBC Toolbox driver for System i
 - Connection Pooling/Statement Pooling
 - Extended Dynamic SQL support
 - Best Practices for i5/OS JDBC Programming
 - Reusable Open Data Path
 - Block Insert
 - Block Fetch
 - Massive Deletes
 - Troubleshooting

2

© 2007 IBM Corporation

Agenda cont.

- Native JDBC Driver
 - Connection Pooling/Statement Pooling JDBC 3.0 style
 - System Wide Cache
 - Performance Related Attributes
 - Troubleshooting
- Access from WebSphere
 - DB2 Object and Row Locking
 - WebSphere Locking Considerations
 - Resource Reference Configuration for Servlets and EJBs
 - Connection and Statement Pooling
 - Using extended DataSource attributes

Access From Java

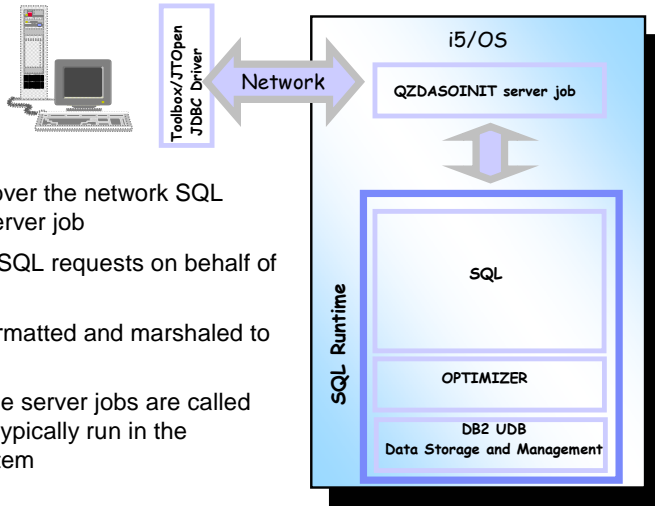
JDBC Drivers for iSeries Access

- IBM offers several JDBC drivers that support DB2 for i5/OS
- System i-optimized drivers from IBM Rochester
 - IBM Toolbox for Java & JOpen - Type 4
 - Native iSeries JDBC driver - Type 2
- DB2 UDB for Linux, Unix, Windows drivers
 - DB2 Universal JDBC Driver - Type 4
 - DB2 JDBC Type 2 Driver - Type 2

IBM Toolbox for Java & JOpen

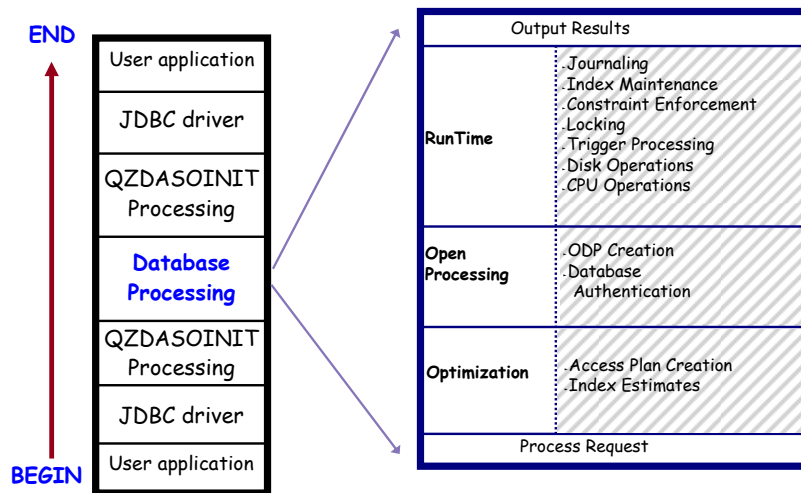
- Implemented in pure Java and optimized for accessing DB2 for i5/OS
- Uses native i5/OS protocol to communicate with the back-end database server job
- JOpen is open-source version of Toolbox driver

IBM Toolbox for Java & JTOpen



- JDBC client sends over the network SQL requests to i5/OS server job
- Server job runs the SQL requests on behalf of the client
- The results are reformatted and marshaled to the client
- The iSeries database server jobs are called QZDASOINIT, and typically run in the QUSRWRK subsystem

Components of JDBC request flow



Basic i5/OS Tuning

- Number of prestart jobs in the QUSRWRK subsystem
 - Use CHGPJE SBSD(QUSRWRK) PGM(QSYS/QZDASOINIT) to change parameters
- Number of disk arms in the i5/OS partition
 - Use WRKDSKSTS to check the disk configuration and to verify disk utilization
- CPU utilization
 - Use WRKSYSSTS command to check the CPU utilization
- Memory pool size and the max activity level values
 - Use WRKSYSSTS command
- Expert cache
 - Use WRKSYSSTS command

Toolbox JDBC Connection Pooling - Deployment

- Obtaining a database connection is resource intensive
- Connection pooling improves the response time
 - Connection pool manager can locate and use an existing connection
- Built-in connection pool manager example
 - Make sure you have current Service Pack



Current SP

```
// Register an AS400JDBCManagedConnectionPoolDataSource object.
AS400JDBCManagedConnectionPoolDataSource tkcpds =
new AS400JDBCManagedConnectionPoolDataSource(); [1]
tkcpds.setServerName(serverName) ;
tkcpds.setDatabaseName(databaseName) ;
tkcpds.setUser(userName) ;
tkcpds.setPassword(password) ;
tkcpds.setSavePasswordWhenSerialized(true);
tkcpds.setPrompt(false);
tkcpds.setMinPoolSize(1);
tkcpds.setInitialPoolSize(3);
tkcpds.setMaxPoolSize(10);
ctx.rebind("ToolkitManagedDataSource", tkcpds); [2]
```

Toolbox JDBC Connection Pooling - Deployment cont.

- AS400JDBCManagedDataSource provides built-in connection pool manager
 - Implements the Referenceable interface and thus can be bound through JNDI services
 - Supports homogenous pooling (connections to a single back-end system).
 - Does not support statement caching
You may consider SQL Package local caching as alternative

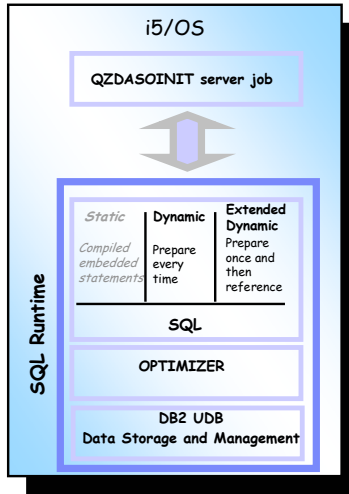
```
AS400JDBCManagedDataSource tkpools = new
AS400JDBCManagedDataSource(); [1]
tkpools.setDescription("DataSource with built-in connection manager");
tkpools.setDataSourceName("ToolkitManagedDataSource"); [2]
ctx.rebind("ConnectionPoolingDataSource", tkpools);
```

Toolbox JDBC Driver Connection Pooling - Execution

- At execution time application obtains a reference to a DataSource that provides built-in connection pool manager
 - Connection pooling is transparent to the application

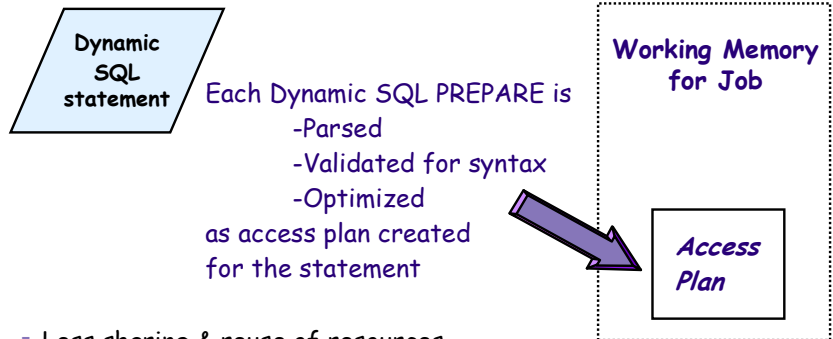
```
// Obtain the reference to a DataSource with built-in connection pooling
ds = (DataSource) ctx.lookup("ConnectionPoolingDataSource");
```

SQL Interfaces used by Toolbox



Access Plans

Dynamic SQL View



- Less sharing & reuse of resources

Generic plan quickly generated on Prepare
 Complete, optimized plan on Execute/Open

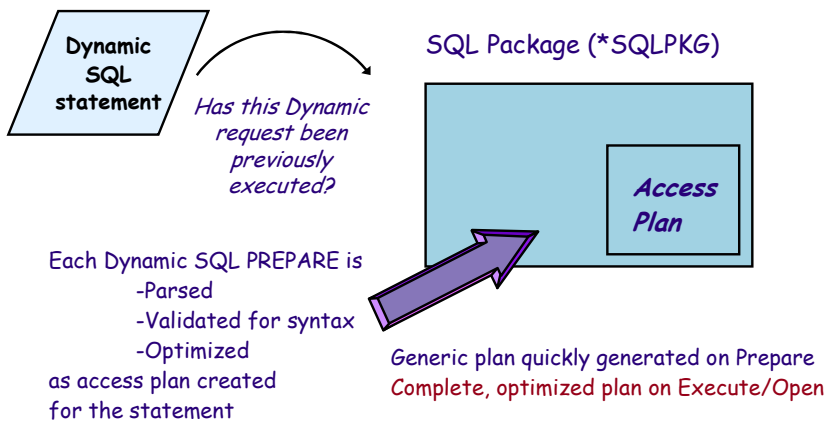
Access Plans

Plan Contents

- A control structure that contains info on the actions necessary to satisfy each SQL request
- These contents include:
 - Access Method
 - Access path ITEM used for file 1
 - Key row positioning used on file 1
 - Info on associated tables and indexes
 - Used to determine if access plan needs to be rebuilt due to table changes or index changes
 - EXAMPLE: a column has been removed from a table since the last time the SQL request was executed
 - Any applicable program and/or environment info
 - Examples: Last time access plan rebuilt, DB2 SMP feature installed

Access Plans

Extended Dynamic SQL View



Enabling Extended Dynamic Support

- By default the extended dynamic support in Toolbox is disabled
 - use properties to enable

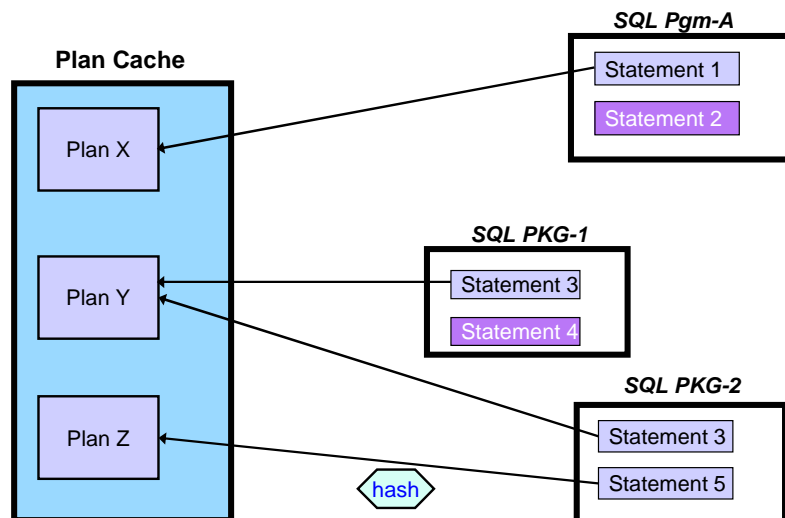
DataSource Example:

```
AS400JDBCManagedConnectionPoolDataSource tkcpds = new
AS400JDBCManagedConnectionPoolDataSource();
...
tkcpds.setExtendedDynamic(true); [1]
tkcpds.setPackage("JDBCDS"); [2]
tkcpds.setPackageCriteria("select"); [3]
```

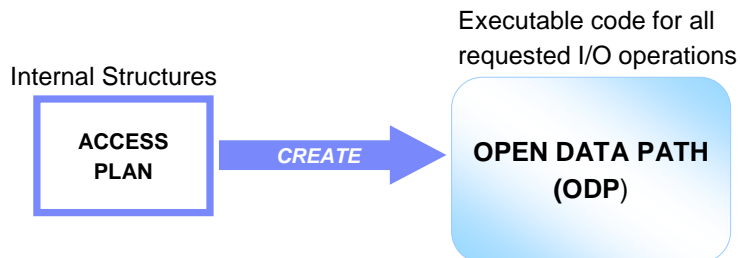
Driver Example:

```
Class.forName("com.ibm.as400.access.AS400JDBCdriver");
Connection conn =
DriverManager.getConnection("jdbc:as400://iSeries;extended
dynamic=true;package=JDBCDRV;package criteria=select;", "db2user",
"db2pwd");
```

More on plan caching - SQE Plan Cache



Access Plan to ODP



- Create process is **EXPENSIVE**
 - Longer execution time the first time an SQL statement is executed
- Emphasizes the need of **REUSABLE** ODPs

ODP Tuning

- Avoid full opens by adopting "prepare once, execute many" programming paradigm
- A PREPARE does NOT automatically create a new statement and full open on each execution
 - DB2 performs caching on Dynamic SQL PREPAREs within a job

Prepare Once, Execute Many Example:

```
PreparedStatement pstmt = con.prepareStatement("INSERT INTO COFFEES
VALUES( ?, ?, ?, ?, ?)");[1]
for (int i = 0; i < outerNumOfLoops; i++) {
    for (int j = 0; j < numOfLoops; j++) {
        pstmt.setString(1, "Lavaza_" + Integer.toString(j));
        ...
        pstmt.addBatch();
    }
    int [] updateCounts = pstmt.executeBatch();[2]
    con.commit();
}
```

Blocking for performance

BLOCK FETCH

- Multiple rows of data from a table are retrieved into the application in a single request
- SQL blocking of fetches can be improved with the following:
 - In general, let the database determine the optimal blocking size
 - The Default blocking size of 32kB can be tuned

```
AS400JDBCManagedConnectionPoolDataSource tkcpds = new
AS400JDBCManagedConnectionPoolDataSource();
tkcpds.setBlockSize(128);
```

- PRIOR, CURRENT, and RELATIVE options should not be used with multiple row fetch due to their random nature

```
s = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_READ_ONLY);
s.setFetchSize(1);
ResultSet rs = s.executeQuery("SELECT t.* FROM t");
rs.absolute(25);
rs.relative(25);
```

Blocking for performance

BLOCK INSERT

- Applications that perform many INSERT statements in succession or via a single loop may be improved by bundling all the new rows into a single request
- Fill a buffer with new rows and then pass the buffer on single executeBatch() method call

```
pstmt = con.prepareStatement("INSERT INTO " + tableName + "
VALUES( ?, ?, ?)");[1]
for (int j = 0; j < numOfLoops; j++) {
    pstmt.setString(1, "Kona_" + Integer.toString(j));
    pstmt.setInt(2, 150);
    pstmt.setFloat(3, 9.99f);
    pstmt.addBatch();[2]
}
int [] updateCounts = pstmt.executeBatch();[3]
```

executeBatch() vs. executeUpdate() comparison

- DB2 for i5/OS access through Java JTOpen JDBC
- 5000 records insert
- Communications gear: 1Gb Virtual LAN

Iteration	executeBatch() ms	executeUpdate() ms
1	4771	18758
2	912	9062
3	547	7937
4	563	8750
5	542	9532

Miscellaneous considerations

- Although SELECT * is very easy to code, it is far more effective to explicitly list the columns that are actually required by the application
 - Minimizes the amount of resource needed
 - Example, SELECT DISTINCT or SELECT UNION requires columns to be sorted
 - Improves the query optimizer's decision making
 - Improves chances of Index Only Access method
- Example: JDBC program that executed a statement 20 times that really only needed 3 out of the 20 total columns
 - "SELECT *" caused the JDBC driver to call the database 800 times
 - "SELECT col1, col2, col3" caused driver to call the database 120 times

Toolbox Driver Troubleshooting

- Troubleshooting a multi-tier application is non-trivial
 - need to deal with software components that reside in separate often heterogeneous environments
- The following tools are particularly useful for troubleshooting
 - Database Monitor
 - Joblog messages for QZDASOINIT database server job
 - Toolbox trace utility

Programmatic Trace Collection

- Database monitor captures implementation of SQL statements
 - Describe implementation methods such as use of indexes, join order, ODP implementation (reusable versus non-reusable), ...
- Connection property can be use to enable database monitor collection & debug messages
 - Debug Messages: Server_Trace_Debug_Server_Job or server trace=4

DataSource Example:

```
AS400JDBCManagedConnectionPoolDataSource tkcpds = new
AS400JDBCManagedConnectionPoolDataSource();
tkcpds.setServerTraceCategories(
    AS400JDBCDataSource.SERVER_TRACE_START_DATABASE_MONITOR);
```

Driver Example:

```
Class.forName("com.ibm.as400.access.AS400JDBCdriver");
Connection conn =
DriverManager.getConnection("jdbc:as400://iSeries;server trace=2;",
"db2user", "db2pwd");
```

Toolbox Trace Utility

- Toolbox trace utility collects detailed client-side traces
- Use connection property to enable client-side tracing

DataSource Example:

```
AS400JDBCManagedConnectionPoolDataSource tkcpds = new
AS400JDBCManagedConnectionPoolDataSource();
tkcpds.setTrace(true);
```

Driver Example:

```
Class.forName("com.ibm.as400.access.AS400JDBCDriver");
Connection conn =
DriverManager.getConnection("jdbc:as400://iSeries;trace=true;",
"db2user", "db2pwd");
```

Toolbox Trace Utility Sample

```
Console [ <terminated> C:\Program Files\IBM\WebSphere Studio\Application Developer\v5.1\ eclipse\rel\bin\javaw.exe (3/26/04 1:32 PM)
1112765323 open. Parent: Connection K005TEK (2011706250) .
1112765323 : Escape processing = "true".
1112765323 : Fetch direction = "1000".
1112765323 : Fetch size = "0".
1112765323 : Max field size = "0".
1112765323 : Max rows = "0".
1112765323 : Query timeout = "0".
1112765323 : Result set concurrency = "1007".
1112765323 : Result set holdability = "-9999".
1112765323 : Result set type = "1003".
1112765323 : Behavior Override = "0".
1112765323 : Data to correlate statement with cursor Cursor CRSR0001 (1136325515) .
1112765323 : Executing query, SQL Statement -->[SELECT t.* FROM JAREK.COFFEES t].
1112765323 : Prepared STMT0001*, SQL Statement -->[SELECT t.* FROM JAREK.COFFEES t].
6325515) open.
1308029835 open. Parent: Statement STMT0001 (1112765323) .
1308029835 : Concurrency = "1007".
1308029835 : Fetch direction = "1000".
1308029835 : Fetch size = "0".
1308029835 : Max rows = "0".
1308029835 : Type = "1003".
1112765323 : Executed STMT0001*, SQL Statement --> [SELECT t.* FROM JAREK.COFFEES t].
1112765323 : Update count = -1.
1112765323 : Result set = true.
1112765323 : Number of result sets = 0.
1112765323 : Row count estimate = 606.

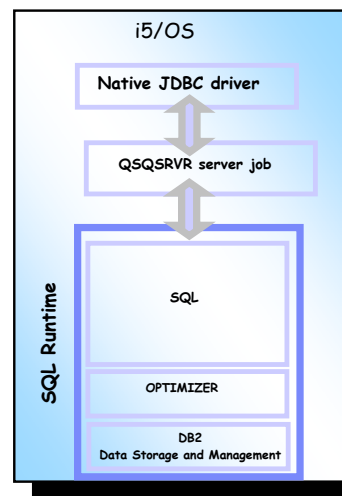
2011706250) : Fetching a block of data from the server.
2011706250) : Fetching a block of data from the server.
2011706250) : Fetching a block of data from the server.
2011706250) : Fetching a block of data from the server.
```

Native DB2 for i5/OS JDBC Driver

- Shipped as part of the iSeries Developer Kit for Java (5722-JV1)
- Implemented as Type 2 JDBC Driver
 - makes native method calls to the CLI (Call Level Interface) APIs
 - runs only on i5/OS JVM

Native JDBC Driver Access to DB2

- Uses SQL Server mode
 - SQL requests routed to a separate prestart job QSQSRVR
 - allows for multiple database connections
 - each connection associated with a separate QSQSRVR job
 - each connection has its own commitment definition
 - may improve performance of multithreaded applications
 - different connection handle per thread to get better performance



Native JDBC Driver Connection Pooling - Deployment

- Supports JDBC 3.0 compliant connection pooling
 - Provides a DataSource that implements ConnectionPoolDataSource used by application servers that provide connection pooling manager functionality (e.g. WebSphere)
 - Provides also a DataSource with built-in connection pool manager

```
// Create a ConnectionPoolDataSource implementation
UDBConnectionPoolDataSource cpds = new UDBConnectionPoolDataSource();
cpds.setDescription("Connection Pooling");
ctx.rebind("ConnectionSupport", cpds)[1];

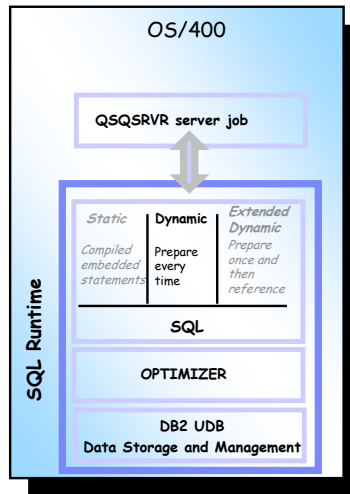
// Create a standard datasource that references it.
UDBDataSource ds = new UDBDataSource();
ds.setDescription("DataSource supporting statement pooling");
ds.setDataSourceName("ConnectionSupport")[2];
ctx.rebind("ConnectionPoolingDataSource", ds)[3];
```

Native JDBC Driver Connection Pooling - Execution

- At execution time application obtains a reference to a DataSource that provides built-in connection pool manager
 - Connection pooling is transparent to the application

```
// Do the work with statement pooling added.
ds = (DataSource) ctx.lookup("ConnectionPoolingDataSource");
```

SQL Interfaces used by Native JDBC Driver



Dynamic SQL Tuning - System Wide Statement Cache

- DB2 for i5/OS also caches access plans for Dynamic SQL requests in the SystemWide Statement Cache (SWSC)
 - Only access plans are reused (No ODP reuse)
- SWSC requires no administration
 - Cache storage allocation & management handled by DB2
 - Cache is created from scratch each IPL
 - Cache churn and contention avoided by allowing limited access plan updates
 - In some cases, optimizer will build a temporary access plan to use instead of the cached access plan
 - Might think about system IPL after your database is tuned
 - Cache contents cannot be viewed, max of 165,000+ statements
- SWSC cache does interact with the job cache
- SWSC cache lives in the Switchable IASP (V5R2), if one is used

Native JDBC driver Statement Pooling

- Supports JDBC 3.0 compliant statement pooling
 - allows an application to reuse a PreparedStatement object
 - transparent to the application code
 - PreparedStatement objects scoped to a PooledConnection
 - Statements can potentially be reused by multiple logical connections
 - Not supported when using DB2Driver to obtain a connection

DataSource Example:

```
UDBConnectionPoolDataSource cpds = new UDBConnectionPoolDataSource();
cpds.setDescription("Connection Pooling with Statement pooling");
cpds.setMaxStatements(10)[1];
ctx.rebind("StatementSupport", cpds);
```

Block Insert Considerations

- executeBatch() method provides superior performance
- Block insert needs to be explicitly enabled

DataSource Example:

```
...
UDBDataSource ds = new UDBDataSource();
ds.setDescription("DataSource supporting statement pooling");
ds.setDataSourceName("ConnectionSupport");
ds.setUseBlockInsert(true);[1]
ctx.rebind("ConnectionPoolingDataSource", ds);
```

Driver Example:

```
Class.forName("com.ibm.db2.jdbc.app.DB2Driver");
Connection conn =
DriverManager.getConnection("jdbc:db2://*LOCAL;use block insert=true:[1]",
"db2user", "db2pwd");
```

Native JDBC Driver Troubleshooting

- Troubleshooting methodology
 - The joblog messages for QSQRV database server job
 - Several levels of JDBC tracing
 - User level tracing
 - Similar to Toolbox tracing utility
 - enabled by "trace" connection property
 - JDBC trace
 - robust and more comprehensive trace utility
 - enabled by a JVM property or through an environment variable
 - Native JDBC Dynamic Tracing
 - useful in environments where the job cannot be ended to start or stop a trace
 - enabled by JVM properties
 - CLI and Malloc Tracing

Native JDBC Trace

- Trace options can be enabled using JVM properties
 - Some trace options can also be set by using environment variables
- Trace settings are read when a JDBC connection is made
- The trace ends when all connections being traced are closed

JVM Property	Environment Variable	Description
jdbc.db2.trace	QIBM_JDBC_TRACE_LEVEL	Traces JDBC errors. Optionally traces all JDBC calls and internal calls.
jdbc.db2.trace.config	QIBM_JDBC_TRACE_CONFIG	File name to direct trace output to. Default is stdout.

Supported trace levels:

Trace Level	Setting	Description
No Trace	'0'	
Error Level	'1'	Failures only
Info Level All	'3'	JDBC calls and internal calls.
Extended Level	'4'	All JDBC trace points except CLI and malloc.

Native JDBC Trace Examples

```
//Setting the trace in QSHELL
Java -Djdbc.db2.trace=4
-Djdbc.db2.trace.config=file:///adirectory/jdbcLog.txt TestJDBCdriver

//Setting the trace in RUNJVA
RUNJVA CLASS(TestJDBCdriver) PROP( (jdbc.db2.trace '4')
(jdbc.db2.trace.config 'file:///adirectory/jdbcLog.txt'))

//Setting the trace using Environment variables for current job
ADDENVVAR ENVVAR(QIBM_JDBC_TRACE_LEVEL) VALUE('4') LEVEL(*JOB)
ADDENVVAR ENVVAR(QIBM_JDBC_TRACE_CONFIG) +
VALUE('file:///adirectory/jdbcLog.txt') LEVEL(*JOB)

//Setting the trace using Environment variables for all jobs
ADDENVVAR ENVVAR(QIBM_JDBC_TRACE_LEVEL) VALUE('4') LEVEL(*SYS)
ADDENVVAR ENVVAR(QIBM_JDBC_TRACE_CONFIG) +
VALUE('file:///adirectory/jdbcLog.txt') LEVEL(*SYS)
```

Choosing the JDBC Driver for System i Access

- Native JDBC Driver is the best choice to access DB2 for i5/OS
 - optimized for DB2 access - best overall performance
 - use when running directly on i5/OS
 - supports a range of specific properties
- IBM Toolbox for Java
 - uses native i5/OS protocol - more efficient than non-native protocols
 - use when accessing DB2 from a separate system/partition
 - runs on any JVM
 - supports extended dynamic SQL and other i5/OS properties
- DB2 Universal JDBC Driver
 - Considered when access to different DB2 platform is required
 - Requires a DRDA (DB2 Connect) connection to System i
 - not as efficient as IBM Toolbox for Java driver

Locking and Concurrency for Java Developers

Locking 101

- Java jobs and threads use database objects to perform transactional work
- DB2 locks these objects to ensure data integrity and the highest possible level of concurrent access
- The object-level and row-level locks are held for the duration of a database transaction
 - The object-level locks are obtained based on the intended use (read or update) and sharing capability (single or multiple job/thread access)
 - Row-level locks are obtained to disallow conflicting updates and prevent reading of uncommitted changes
- Transaction allows you to group multiple SQL statements together
 - The current transaction context is called a "commitment definition."
 - The transaction processing, locking, and concurrent access are managed by DB2 commitment control

Online Article: *DB2 Locking and Concurrency for Java Developers*
<http://www.mcpressonline.com/mc/1@1.ZHP2dh9fEGR.0@.6b3ce83b>

Decrypting Isolation Levels Naming

- DB2 uses a combination of object- and record-level locks to enforce various isolation levels
- In addition to four ANSI standard isolation levels, DB2 for i5/OS supports a non-standard isolation level of NONE
- The ANSI SQL isolation level naming differs from the naming adopted by the DB2 family
- DB2 for i5/OS, in turn, uses the traditional operating system naming

ANSI SQL Isolation Level	DB2 Isolation Level	i5/OS Isolation Level (Commit Level)
	NO COMMIT (NC)	*NC
READ UNCOMMITTED	UNCOMMITTED READ (UR)	*CHG
READ COMMITTED	CURSOR STABILITY (CS)	*CS
REPEATABLE READ	READ STABILITY (RS)	*ALL
SERIALIZABLE	REPEATABLE READ (RR)	*RR

Isolation Levels and Locking

- DB2 implements the isolation levels by locking
- Commitment control provides additional functionality also implemented by locking
 - For example: support for updateable cursors
- DB2 locking guarantees the ANSI standard compliance
 - Locks acquired to provide additional functionality do not break the compliance because the disallowed phenomena do not occur



Compatibility of Object and Row locks

Object Level Locks

*SHRRD

Lock shared for read.

*SHRUPD

Lock shared for update.

*SHRNUP

Lock shared, no update.

*EXCLRD

Lock exclusive, read allowed.

*EXCL

Lock exclusive, no read allowed.

Object Level Lock	Incompatible Locks (locks that cannot be obtained by other processes)
EXCL	EXCL, EXCLRD, SHRUPD, SHRNUP, SHRRD
EXCLRD	EXCL, EXCLRD, SHRUPD, SHRNUP
SHRUPD	EXCL, EXCLRD, SHRNUP
SHRNUP	EXCL, EXCLRD, SHRUPD
SHRRD	EXCL

Row Level Locks

Read - The record is locked for read. Another job may read the same record but cannot lock the record for update intent. The record cannot be changed by another job as long as one job holds a read lock on the record.,

Update - The record is locked for update intent. Another job may read the record but may not obtain a read or update lock on it until the lock is released

Row Level Lock	Incompatible Row Level Locks (locks that cannot be obtained by other processes)
READ	UPDATE
UPDATE	READ, UPDATE



Object and Row Locking for Select Cursor Attributes

Isolation Level	SQL Operation	Statement Clause	Member Lock	Data Space Lock	Row Lock
READ_COMMITTED	DELETE (fast delete)	NO WHERE clause	*EXCL	-	-
READ_COMMITTED	DELETE	WITH WHERE (searched Delete)	*SHRRD	*SHRUPD	UPDATE Deleted rows
READ_COMMITTED	INSERT		*SHRRD	*SHRUPD	UPDATE Inserted rows
READ_COMMITTED	SELECT		*SHRRD	*SHRRD	READ Short-lived at the time the cursor is opened
READ_COMMITTED	UPDATE		*SHRRD	*SHRUPD	UPDATE Updated rows
REPEATABLE_READ	DELETE (fast delete)	NO WHERE clause	*EXCL	-	-
REPEATABLE_READ	DELETE	WITH WHERE	*SHRRD	*SHRRD	UPDATE Deleted rows
REPEATABLE_READ	INSERT		*SHRRD	*SHRUPD	UPDATE Inserted rows
REPEATABLE_READ	UPDATE		*SHRRD	*SHRUPD	UPDATE Updated rows
REPEATABLE_READ	SELECT		*SHRRD	*SHRRD	READ On open rows cached by the ResultSet implementation

Object and Row Locking for Select Cursor Attributes

Line	Isolation Level	Cursor Attributes	Statement Clause	Member Lock	Data Space Lock	Record Lock
1	READ_COMMITTED	READ_ONLY FORWARD_ONLY Blocked Fetch		*SHRRD	*SHRRD	READ Short lived - at the time the SQ fetches the row to build a result set
2	READ_COMMITTED	READ_ONLY FORWARD_ONLY 1 row fetch	FOR UPDATE	*SHRRD	*SHRUPD	UPDATE The row on which the cursor is currently positioned
3	READ_COMMITTED	UPDATABLE FORWARD_ONLY 1 row fetch		*SHRRD	*SHRUPD	UPDATE The row on which the cursor is currently positioned
4	READ_COMMITTED	READ_ONLY FORWARD_ONLY Blocked Fetch	WITH CS KEEP LOCKS	*SHRRD	*SHRRD	READ ALL rows read by the cursor (block of rows at a time)
5	REPEATABLE_READ	READ_ONLY FORWARD_ONLY Blocked Fetch		*SHRRD	*SHRRD	READ ALL rows read by the cursor (block of rows at a time)
6	REPEATABLE_READ	READ_ONLY FORWARD_ONLY 1 row fetch	FOR UPDATE	*SHRRD	*SHRUPD	UPDATE ALL rows accessed by the cursor
7	REPEATABLE_READ	UPDATABLE FORWARD_ONLY 1 row fetch	FOR READ ONLY	*SHRRD	*SHRRD	READ ALL rows read by the cursor, a batch of locks is obtained
8	REPEATABLE_READ	UPDATABLE FORWARD_ONLY 1 row fetch		*SHRRD	*SHRUPD	UPDATE ALL rows accessed by the cursor, locks obtained one row at a time
9	Any	READ_ONLY FORWARD_ONLY Blocked Fetch	WITH RS USE AND KEEP EXCLUSIVE LOCKS			

Access From WebSphere

WebSphere Locking Considerations

- WAS-managed Transactions use a default isolation level of Repeatable Read, that is NOT the most efficient isolation level
 - NOT the most scalable isolation level since rows are locked exclusively until the end of a transaction
 - Resetting isolation level property on DataSource object will NOT change the default isolation level used
 - Resource referencing must be used instead
 - Resource reference method differs depending if referenced used in a servlet or EJB
 - WebSphere Studio can be used to define the resource references
 - More Details on using resource reference for isolation level configuration can be found in the "Accessing DB2 UDB for iSeries data from AIX Applications" white paper (ibm.com/servers/enable/site/education/ibo/view.html?wp#db2)

WebSphere Locking Considerations

- Servlet & Resource Reference Configuration
 - Open Deployment Descriptor editor for project containing the servlet
 - Select the References tab and then select Resource tab
 - Under Resource References section, select the Add button
 - Sample reference created for iSeries DataSource, TestJDBCDB

References

EJB | EJB Local | Resource | Resource Environment | JSP tag libraries

Resource References

The following resource references are used in this web application:

TestJDBC12C

Add Remove

Details

Details of the selected resource reference

Type: Browse...

Authentication:

Sharing:

Description:

WebSphere Extensions

Connection management:

Isolation level:

WebSphere Bindings

JNDI Name:

Overview | Servlets | Filters | Listeners | Security | Environment | References | Pages | Parameters | MIME | Extensions | Source

WebSphere Locking Considerations

- Servlet & Resource Reference Configuration (continued)
 - Once the resource reference defined, the Servlet source code needs to be changed to reflect the fact that a resource reference is used to obtain a DataSource from the JNDI service

```
try{
    ctx = new InitialContext();
    ds = (DataSource)ctx.lookup("java:comp/env/TestJDBCJ2C");
} catch(Exception e) {
e.printStackTrace(out);
}
```

WebSphere Locking Considerations

- EJB & Resource Reference Configuration
 - Open Deployment Descriptor editor for project containing the EJB
 - In References section on the left, locate and select the EJB that needs a default isolation level change, Click Add
 - New resource definition needs to be further modified - click on the (+) sign next to the EJB in the References list

The screenshot shows the 'References' configuration window in WebSphere. On the left, a tree view shows the project structure: 'Coffees' (expanded) contains 'ResourceRef TestJDBCJ2C' and 'TestJDBC'. The right pane is for editing the selected 'ResourceRef TestJDBCJ2C'. It includes fields for Name (TestJDBCJ2C), Description, Type (javax.sql.DataSource), Authentication (CONTAINER), and Sharing scope (Shareable). Below these are sections for 'WebSphere Bindings' (JNDI name: jdbc/TestJDBCDB) and 'WebSphere Extensions' (Isolation level: TRANSACTION_READ_COMMITTED, Connection policy: Default). At the bottom, there are 'Add...' and 'Remove' buttons and a navigation bar with 'Overview', 'Beans', 'Assembly Descriptor', 'References', 'Access', and 'Source' tabs.

WebSphere Considerations

- Connection Pooling and Statement Caching
 - WebSphere provides its own connection pooling and statement caching mechanism that is, by default, enabled
- Extended Dynamic SQL
 - non-JTA connections use `com.ibm.as400.access.AS400JDBCConnectionPoolDataSource` class supports a range of iSeries specific properties that can be used to fine tune JDBC performance
 - These properties can be set through the WebSphere Application Server admin console

Additional Information

- DB2 for i5/OS home page - ibm.com/series/db2
- Newsgroups
 - USENET: `comp.sys.ibm.as400.misc`, `comp.databases.ibm-db2`
 - System i Network DB2 Forum - <http://systeminetwork.com/isnetforums/forumdisplay.php>
- Education Resources - Classroom & Online
 - ibm.com/series/db2/gettingstarted.html
 - ibm.com/servers/enable/site/education/ibo/view.html?oc#db2
- DB2 for i5/OS Publications
 - White Papers: ibm.com/servers/enable/site/education/ibo/view.html?wp#db2
 - Online Manuals: ibm.com/series/db2/books.html
 - Porting Help: ibm.com/servers/enable/site/db2/porting.html
 - DB2 for i5/OS Redbooks (<http://ibm.com/redbooks>)
 - [Stored Procedures, Triggers, and UDFs on DB2 UDB for iSeries \(SG24-6503-02\)](#)
 - [Preparing for and Tuning the SQL Query Engine on DB2 for i5/OS \(SG24-6598-01\)](#)
 - [Modernizing iSeries Application Data Access \(SG24-6393\)](#)
 - [SQL Performance Diagnosis on DB2 for i5/OS \(SG24-6654\)](#)



Trademarks

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries. For a complete list of IBM Trademarks, see www.ibm.com/legal/copytrade.shtml: AS/400, DBE, e-business logo, ESCO, eServer, FICON, IBM, IBM Logo, Series, MVS, OS/390, pSeries, RS6000, S/390, VM/ESA, VSE/ESA, Websphere, xSeries, z/OS, zSeries, z/VM

The following are trademarks or registered trademarks of other companies

Lotus, Notes, and Domino are trademarks or registered trademarks of Lotus Development Corporation
Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries
Linux is a registered trademark of Linux Torvalds
UNIX is a registered trademark of The Open Group in the United States and other countries.
Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.
SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.
Intel is a registered trademark of Intel Corporation.
* All other products may be trademarks or registered trademarks of their respective companies.

NOTES:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

References in this document to IBM products or services do not imply that IBM intends to make them available in every country.

Any proposed use of claims in this presentation outside of the United States must be reviewed by local IBM country counsel prior to such use.

The information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.