JDBC Maintenance Release 4.3

Description:

Maintenance review of the JDBC 4.0 Specification

Maintenance Lead:

Lance Andersen, Oracle Corporation

Feedback:

Comments should be sent to jsr221-comments@jcp.org

Rationale for Changes:

The goal is to address several specification issues as well as several minor enhancements requested by the JDBC EG and user community.

Proposed Changes:

1. **Addition of Sharding Support**

Sharding is a scaling technique in which data is horizontally partitioned across independent databases.

The following example demonstrates the use of ShardingKeyBuilder to create a superShardingKey for an eastern region with a ShardingKey specified for the Pittsburgh branch office:

```
DataSource ds = new MyDataSource();
ShardingKey superShardingKey =
ds.createShardingKeyBuilder()
                .subkey("EASTERN_REGION",
JDBCType.VARCHAR)
                .build();
```

```
        ShardingKey shardingKey =
ds.createShardingKeyBuilder()
                        .subkey("PITTSBURGH_BRANCH",
JDBCType.VARCHAR)
                        .build();
        Connection con = ds.createConnectionBuilder()

.superShardingKey(superShardingKey)
                        .shardingKey(shardingKey)
                        .build();
```

To determine if a JDBC Driver supports sharding, an application may call DatabaseMetaData.supportsSharding.

2. **Addition of the java.sql.ConnectionBuilder Interface**

   A builder created from a DataSource object, used to establish a connection to the database that the data source object represents.

   java.sql.ConnectionBuilder contains the following methods:

   - Connection build() throws SQLException
   - ConnectionBuilder password(String password)
   - ConnectionBuilder shardingKey(ShardingKey shardingKey)
   - ConnectionbBuilder superShardingKey(ShardingKey superShardingKey)
   - ConnectionBuilder user(String username)

3. **Addition of the java.sql.ShardingKey Interface**

   This interface is used to indicate that this object represents a Sharding Key. A ShardingKey instance is only guaranteed to

be compatible with the data source instance that it was derived from. A ShardingKey is created using ShardingKeyBuilder.

4. **Addition of the java.sql.ShardingKeyBuilder Interface**

   A builder created from a DataSource or XADataSource object, used to create a ShardingKey with sub-keys of supported data types. Implementations must support JDBCType.VARCHAR and may also support additional data types.

   java.sql.ShardingKeyBuilder contains the following methods:

   - ShardingKey build() throws SQLException
   - ShardingKeyBuilder subkey(Object subkey, SQLType subkeyType)

5. **Addition of the javax.sql.XAConnectionBuilder Interface**

   A builder created from a XADataSource object, used to establish a connection to the database that the data source object represents.

   javax.sql.XAConnectionBuilder contains the following methods:

   - XAConnection build() throws SQLException
   - XAConnectionBuilder password(String password)
   - XAConnectionBuilder shardingKey(ShardingKey shardingKey)
   - XAConnectionbBuilder superShardingKey(ShardingKey superShardingKey)
   - XAConnectionBuilder user(String username)

6. **java.sql.Connection changes**

   The following methods have been added in java.sql.Connection:

- default void beginRequest throws SQLException
- default void endRequest throws SQLException
- default void setShardingKey(ShardingKey shardingKey) throws SQLException
- default void setShardingKey(ShardingKey shardingKey, ShardingKey superShardingKey) throws SQLException
- default void setShardingKeyIfValid(ShardingKey shardingKey, int timeout) throws SQLException
- default void setShardingKeyIfValid(ShardingKey shardingKey, ShardingKey superShardingKey, int timeout) throws SQLException

7. **java.sql.DriverManager changes**

   The following methods have been added to java.sql.DriverManager:

   - public static Stream<Driver> drivers()

   The following methods have been clarified in java.sql.DriverManager:

   - DriverManager overview
     - Clarifies how DriverManager will attempt to load available JDBC drivers during initialization
   - public static void deregisterDriver(Driver driver)
     - Clarifies the behavior when a security manager is present.
   - public static PrintStream getLogStream()
     - Specify the Java SE release when the method was deprecated.
   - public static void setLogStream(PrintStream out)
     - Specify the Java SE release when the method was deprecated.
     - Clarifies the behavior when a SecurityManager is present.

- public static void setLogWriter(PrintWriter out)
    - Clarifies the behavior when a SecurityManager is present.

8. **java.sql.DatabaseMetaData changes**

   The following methods have been added to to java.sql.DatabaseMetaData

   - default boolean supportsSharding() throws SQLException

   The following methods have been clarified in java.sql.DatabaseMetaData

   - ResultSet getTables(String catalog, String schemaPattern, String tableNamePattern, String[] types)
       - The returned REMARKS column may be null

9. **java.sql.Date changes**

   The following methods have been clarified in java.sql.Date

   - public Date(in year, int month, int day)
       - Specify the Java SE release when the method was deprecated
   - public int getHours()
       - Specify the Java SE release when the method was deprecated
   - public int getMinutes()
       - Specify the Java SE release when the method was deprecated
   - public int getSeconds()
       - Specify the Java SE release when the method was deprecated
   - public void setHours(int i)
       - Specify the Java SE release when the method was deprecated
   - public void setMinutes(int i)

    - Specify the Java SE release when the method was deprecated
- public void setSeconds(int i)
    - Specify the Java SE release when the method was deprecated
- public java.time.instant toInstant()
    - Remove the errant @return information
- public java.time.LocalDate toLocalDate()
    - Clarify that the LocalDate instance is created using the Year, Month, Day from the Date instance

10. **java.sql.Time changes**

The following methods have been clarified in java.sql.Time:

- public Time (int hour, int minute, int second, int nano)
    - Specify the Java SE release when the method was deprecated
- public int getDate()
    - Specify the Java SE release when the method was deprecated
    - Remove the errant @return information
- public int getDay()
    - Specify the Java SE release when the method was deprecated
    - Remove the errant @return information
- public int getMonth()
    - Specify the Java SE release when the method was deprecated
    - Remove the errant @return information
- public int getYear()
    - Specify the Java SE release when the method was deprecated
    - Remove the errant @return information
- public void setDate(int i)
    - Specify the Java SE release when the method was deprecated
    - Remove the errant @param information

- public void setMonth(int i)
    - Specify the Java SE release when the method was deprecated
    - Remove the errant @param information
- public void setYear(int i)
    - Specify the Java SE release when the method was deprecated
    - Remove the errant @param information
- public Instant toInstant()
    - Remove the errant @return inform
- public Instant toLocalTime()
    - Clarify that the nanosecond LocalTime field will be set to zero
- public Instant valueOf(LocalTime time)
    - Clarify that the nanosecond LocalTime field will not be part of the newly created Time Object

11. **java.sql.Timestamp changes**

The following methods have been clarified in java.sql.Timestamp:

- public Timestamp(int year, int month, int date, int hour, int minute, int second, int nano)
    - Specify the Java SE release when the method was deprecated
- public int hashCode()
    - Clarified the behavior of how the hashCode is calculated
- public toString()
    - Clarify the nanosecond precision

12. **java.sql.Statement changes**

The following methods have been added to java.sql.Statement:

- default String enquoteIdentifier(String identifier, Boolean

alwaysQuote) throws SQLException
- default String enquoteLiteral(String val) throws SQLException
- default String enquoteNCharLiteral(String val) throws SQLException
- default boolean isSimpleIdentifier(String identifier) throws SQLException

13. **java.sql.CallableStatement changes**

The following methods have been clarified in java.sql.CallableStatement:

- BigDecimal getBigDecimal(int parameterIndex, int scale) throws SQLException
    - Specify the Java SE release when the method was deprecated

14. **java.sql.PreparedStatement changes**

The following methods have been clarified in java.sql.PreparedStatement

- void setUnicodeStream(int parameterIndex, inputStream x, int length) throws SQLException
    - Specify the Java SE release when the method was deprecated

15. **java.sql.ResultSet changes**

The following methods have been clarified in java.sql.ResultSet:

- BigDecimal getBigDecimal(int parameterIndex, int scale) throws SQLException

    - Specify the Java SE release when the method was deprecated
- BigDecimal getBigDecimal(String columnLabel, int scale) throws SQLException
    - Specify the Java SE release when the method was deprecated
- InputStream getUnicodeStream(int columnIndex) throws SQLException
    - Specify the Java SE release when the method was deprecated
- InputStream getUnicodeStream(String columnLabel) throws SQLException
    - Specify the Java SE release when the method was deprecated

16. **javax.sql.CommonDataSource changes**

The following methods have been added to javax.sql.CommonDataSource:

- default ShardingKeyBuilder createShardingKeyBuilder() throws SQLException

17. **javax.sql.ConnectionPoolDatasource changes**

The following methods have been added to javax.sql.ConnectionPoolDataSource:

- default PooledConnectionBuilder createPooledConnectionBuilder () throws SQLException

18. **javax.sql.PooledConnection changes**

Clarified that if the connection pool manager wraps or provides a proxy to the logical handle returned from a call to PooledConnection.dgetConnection, the pool manager must do

one of the following when the application calls Connection.close:

- call endRequest on the logical Connection handle
- call close on the logical Connection handle

19. javax.sql.XADataSource changes

The following methods have been added to javax.sql.XADataSource:

- default XAConnectionBuilder createXAConnectionBuilder () throws SQLException