JDBC Maintenance Release 4.5

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.

Changes:

1. **AutoClosable Support**

   The following interfaces now support the AutoClosable interface:

   - java.sql.Array
   - java.sql.Blob
   - java.sql.Clob
   - java.sql.NClob
   - java.sql.SQLXML

   To support this feature, the following close method has been added to each interface:

```
default void close() throws SQLException {
  free();
};
```

2. **java.sql.Connection changes**

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

   - 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

3. **java.sql.JDBCType and java.sql.Types changes**

   The following SQL types have been added to **java.sql.JDBCType and java.sql.Types**:

   - DECFLOAT
     - Types.java value: 2015
   - JSON
     - Types.java value: 2016

4. **java.sql.Timestamp hashCode changes:**

   The Timestamp `hashCode` method incorrectly indicated that the nanos value was not included in the calculation of the hash code.

5. **java.sql.SQLPermission changes:**

   java.sql.SQLPermission has been marked deprecated for removal.

6. **Selective Disabling of JDBC Escape Syntax processing and Parameter Markers (‘?’)**

   In a JDBC SQL String (i.e., a Java String literal), parameter markers are represented by the character ‘?’. With the introduction of the `MATCH_RECOGNIZE` operator in SQL:2016, the ‘?’ character can also appear as a token within SQL statements

   To accommodate this, JDBC allows for the disabling of parameter marker processing (for ‘?’) and the evaluation of JDBC escape syntax within a specified section of a JDBC SQL String. This section is defined by the delimiters "`{\\`" at the start and "`\\}`" at the end.

   The character sequence "\\\\" may be used to include a backslash within "{\\" and "\\}".

   The following is an example of a SQL query which selectively disables Escape processing:

```
String sql = """
    select T.firstW, T.lastZ, ? from tkpattern_S11
    MATCH_RECOGNIZE (
        MEASURES A.c1 as firstW, last(Z.c1) as lastZ
        ALL MATCHES
       {\\ PATTERN (A? X*? Y+? Z??)\\}
        DEFINE
            X as X.c2 > prev(X.c2),
            Y as Y.c2 < prev(Y.c2),
            Z as Z.c2 > prev(Z.c2)
    ) as T
""";
```

   The JDBC driver will not process the PATTERN clause "{\\ PATTERN (A? X*? Y+? Z??) \\}" and will result in:

```
PATTERN (A? X*? Y+? Z??)
```

Additional examples:

String Literal	Result
"{\\ _foo_\\}"	_foo_
"{\\ q'{\\\\ foo \\\\}'\\}"	q'{\ foo \}'
"{\\ {ts '1999-01-09 20:11:11.123455'} \\}"	{ts '1999-01-09 20:11:11.123455'}
"{\\ ? \\}"	?
"{\\ /* \\\\} */ \\}"	/* \} */
"{\\ // \\\\} \n hi\\}"	// \} ↵ hi
"{\\ '\\\\' \\}"	'\'
"{\\ \"\\\\\" \\}"	"\"
"{\\ '\\\\}' \\}"	'\}'
"{\\ \"\\\\}\" \\}"	"\}"
"{\\ \\ \\}"	SQLException thrown
"{\\ \\\\ \\}"	\
"{\ q'{\abc\}'\}"	Compilation error
"{\\ q'{\\abc\\}'\\}"	SQLException thrown
"{\\q'{\\\\abc\\\\}'\\}"	q'{\abc\}'
"{\ 'literal with {\..\}'\}"	Compilation error
"{\\'literal with {\\\\..\\\\}'\\}"	'literal with {\..\}'
"{\/* comment {\..\} */ \}"	Compilation error
""" {\\ // \\\\} hi\\} """	// \} hi

7. **Appendix B updates**

   The Data Type Conversion tables have been updated to address missing conversions as well as adding conversions for the JSON and DECFLOAT SQL Data types.

APPENDIX **B**

B. Data Type Conversion Tables

This appendix describes mappings and conversions that drivers must support. The following mappings and conversions are covered:

- JDBC Types Mapped to Java Types
- Java Types Mapped to JDBC Types
- JDBC Types Mapped to Java Object Types
- Java Object Types Mapped to JDBC Types
- Conversions by `setObject` and `setNull` from Java Object Types to JDBC Types
- Type Conversions Supported by `ResultSet` `getter` Methods

B.1 JDBC Types Mapped to Java Types

TABLE B-1 shows the conceptual correspondence between JDBC types and Java types. A programmer should write code with this mapping in mind. For example, if a value in the database is a `SMALLINT`, a short should be the data type used in a JDBC application.

All `CallableStatement` getter methods except for `getObject` use this mapping. The `getObject` methods for both the `CallableStatement` and `ResultSet` interfaces use the mapping in TABLE B-3.

TABLE B-1 JDBC Types Mapped to Java Types

JDBC Type	Java Type
CHAR	String
VARCHAR	String
LONGVARCHAR	String
NUMERIC	java.math.BigDecimal
DECIMAL	java.math.BigDecimal
BIT	boolean
BOOLEAN	boolean
TINYINT	byte
SMALLINT	short
INTEGER	int
BIGINT	long
REAL	float
FLOAT	double
DOUBLE	double
BINARY	byte[]
VARBINARY	byte[]
LONGVARBINARY	byte[]
DATE	java.sql.Date
TIME	java.sql.Time
TIMESTAMP	java.sql.Timestamp

JDBC Type	Java Type
CLOB	java.sql.Clob
BLOB	java.sql.Blob
ARRAY	java.sql.array
DISTINCT	Mapping of underlying type
STRUCT	java.sql.Struct
REF	java.sql.Ref
DATALINK	java.net.URL
JAVA_OBJECT	Underlying Java class
ROWID	java.sql.RowId
NCHAR	String
NVARCHAR	String
LONGNVARCHAR	String
NCLOB	java.sql.NClob
SQLXML	java.sql.SQLXML
JSON	String
DECFLOAT	java.math.BigDecimal
TIMESTAMP_WITH_TIMEZONE	java.time.OffsetDatetime
TIME_WITH_TIMEZONE	java.time.OffsetTime

B.2 Java Types Mapped to JDBC Types

TABLE B-2 shows the mapping a driver should use for the updater methods in the `ResultSet` interface and for IN parameters. `PreparedStatement` setter methods and `RowSet` setter methods use this table for mapping an IN parameter, which is a Java type, to the JDBC type that will be sent to the database. Note that the `setObject` methods for these two interfaces use the mapping shown in TABLE B-4.

TABLE B-2 Standard Mapping from Java Types to JDBC Types

Java Type	JDBC Type
`String`	CHAR, VARCHAR, LONGVARCHAR, NCHAR, NVARCHAR, LONGNVARCHAR or JSON
`java.math.BigDecimal`	DECIMAL, NUMERIC or DECFLOAT
`boolean`	BIT or BOOLEAN
`byte`	TINYINT
`short`	SMALLINT
`int`	INTEGER
`long`	BIGINT
`float`	REAL
`double`	DOUBLE
`byte[]`	BINARY, VARBINARY, or LONGVARBINARY
`java.sql.Date`	DATE
`java.sql.Time`	TIME
`java.sql.Timestamp`	TIMESTAMP
`java.sql.Clob`	CLOB

Java Type	JDBC Type
java.sql.Blob	BLOB
java.sql.Array	ARRAY
java.sql.Struct	STRUCT
java.sql.Ref	REF
java.net.URL	DATALINK
Java class	JAVA_OBJECT
java.sql.RowId	ROWID
java.sql.NClob	NCLOB
java.sql.SQLXML	SQLXML

B.3 JDBC Types Mapped to Java Object Types

`ResultSet.getObject` and `CallableStatement.getObject` use the mapping shown in TABLE B-3 for standard mappings.

Note – The JDBC 1.0 specification defined the Java object mapping for the SMALLINT and TINYINT JDBC types to be `Integer`. The Java language did not include the `Byte` and `Short` data types when the JDBC 1.0 specification was finalized. The mapping of SMALLINT and TINYINT to `Integer` is maintained to preserve backwards compatibility.

TABLE B-3 Mapping from JDBC Types to Java Object Types

JDBC Type	Java Object Type
CHAR	String
VARCHAR	String
LONGVARCHAR	String
NUMERIC	java.math.BigDecimal
DECIMAL	java.math.BigDecimal
BIT	Boolean
BOOLEAN	Boolean
TINYINT	Integer
SMALLINT	Integer
INTEGER	Integer
BIGINT	Long
REAL	Float

JDBC Type	Java Object Type
FLOAT	Double
DOUBLE	Double
BINARY	byte[]
VARBINARY	byte[]
LONGVARBINARY	byte[]
DATE	java.sql.Date
TIME	java.sql.Time
TIMESTAMP	java.sql.Timestamp
DISTINCT	Object type of underlying type
ROWID	java.sql.RowId
NCHAR	String
NVARCHAR	String
LONGNVARCHAR	String
NCLOB	java.sql.NClob
SQLXML	java.sql.SQLXML
TIMESTAMP_WITH_TIMEZONE	java.time.OffsetDatetime
TIME_WITH_TIMEZONE	java.time.OffsetTime
DECFLOAT	java.math.BigDecimal
JSON	String

B.4 Java Object Types Mapped to JDBC Types

`PreparedStatement.setObject`, `PreparedStatement.setNull`, `RowSet.setNull` and `RowSet.setObject` use the mapping shown TABLE B-4 when no parameter specifying a target JDBC type is provided.

TABLE B-4 Mapping from Java Object Types to JDBC Types

Java Object Type	JDBC Type
`String`	CHAR, VARCHAR, LONGVARCHAR, NCHAR, NVARCHAR, LONGNVARCHAR or JSON
`java.math.BigDecimal`	DECIMAL, NUMERIC or DECFLOAT
`Boolean`	BIT or BOOLEAN
`Byte`	TINYINT
`Short`	SMALLINT
`Integer`	INTEGER
`Long`	BIGINT
`Float`	REAL
`Double`	DOUBLE
`byte[]`	BINARY, VARBINARY, or LONGVARBINARY
`java.math.BigInteger`	BIGINT
`java.sql.Date`	DATE
`java.sql.Time`	TIME
`java.sql.Timestamp`	TIMESTAMP

Java Object Type	JDBC Type
java.sql.Clob	CLOB
java.sql.Blob	BLOB
java.sql.Array	ARRAY
java.sql.Struct	STRUCT
java.sql.Ref	REF
java.net.URL	DATALINK
Java class	JAVA_OBJECT
java.sql.RowId	ROWID
java.sql.NClob	NCLOB
java.sql.SQLXML	SQLXML
java.util.Calendar	TIMESTAMP
java.util.Date	TIMESTAMP
java.time.LocalDate	DATE
java.time.LocalTime	TIME
java.time.LocalDateTime	TIMESTAMP
java.time.OffsetTime	TIME_WITH_TIMEZONE
java.time.OffsetDatetime	TIMESTAMP_WITH_TIMEZONE

B.5 Conversions by `setObject` and `setNull` from Java Object Types to JDBC Types

TABLE B-5 shows which JDBC types may be specified as the target JDBC type to the methods `PreparedStatement.setObject`, `PreparedStatement.setNull`, `RowSet.setNull`, and `RowSet.setObject`.

TABLE B-5 Conversions Performed by `setObject` and `setNull` Between Java Object Types and Target JDBC Types

Java Object Type	Supported JDBC Type
`String`	TINYINT, SMALLINT, INTEGER, BIGINT, REAL, FLOAT, DOUBLE, DECIMAL, NUMERIC, BIT, BOOLEAN, CHAR, VARCHAR, LONGVARCHAR, BINARY, VARBINARY, LONVARBINARY, DATE, TIME, TIMESTAMP, NCHAR, NVARCHAR, LONGNVARCHAR, TIME_WITH_TIMEZONE, TIMESTAMP_WITH_TIMEZONE, JSON, DECFLOAT
`java.math.BigDecimal`	TINYINT, SMALLINT, INTEGER, BIGINT, REAL, FLOAT, DOUBLE, DECIMAL, NUMERIC, BIT, BOOLEAN, CHAR, VARCHAR, LONGVARCHAR, DECFLOAT
`Boolean`	TINYINT, SMALLINT, INTEGER, BIGINT, REAL, FLOAT, DOUBLE, DECIMAL, NUMERIC, BIT, BOOLEAN, CHAR, VARCHAR, LONGVARCHAR
`Byte`	TINYINT, SMALLINT, INTEGER, BIGINT, REAL, FLOAT, DOUBLE, DECIMAL, NUMERIC, BIT, BOOLEAN, CHAR, VARCHAR, LONGVARCHAR, DECFLOAT
`Short`	TINYINT, SMALLINT, INTEGER, BIGINT, REAL, FLOAT, DOUBLE, DECIMAL, NUMERIC, BIT, BOOLEAN, CHAR, VARCHAR, LONGVARCHAR, DECFLOAT

Java Object Type	Supported JDBC Type
Integer	TINYINT, SMALLINT, INTEGER, BIGINT, REAL, FLOAT, DOUBLE, DECIMAL, NUMERIC, BIT, BOOLEAN, CHAR, VARCHAR, LONGVARCHAR, DECFLOAT
Long	TINYINT, SMALLINT, INTEGER, BIGINT, REAL, FLOAT, DOUBLE, DECIMAL, NUMERIC, BIT, BOOLEAN, CHAR, VARCHAR, LONGVARCHAR, DECFLOAT
Float	TINYINT, SMALLINT, INTEGER, BIGINT, REAL, FLOAT, DOUBLE, DECIMAL, NUMERIC, BIT, BOOLEAN, CHAR, VARCHAR, LONGVARCHAR, DECFLOAT
Double	TINYINT, SMALLINT, INTEGER, BIGINT, REAL, FLOAT, DOUBLE, DECIMAL, NUMERIC, BIT, BOOLEAN, CHAR, VARCHAR, LONGVARCHAR, DECFLOAT
byte[]	BINARY, VARBINARY, or LONGVARBINARY
java.math.BigInteger	BIGINT, CHAR, VARCHAR, LONGVARCHAR, DECFLOAT
java.sql.Date	CHAR, VARCHAR, LONGVARCHAR, DATE, TIMESTAMP
java.sql.Time	CHAR, VARCHAR, LONGVARCHAR, TIME, TIMESTAMP
java.sql.Timestamp	CHAR, VARCHAR, LONGVARCHAR, DATE, TIME, TIMESTAMP
java.sql.Array	ARRAY
java.sql.Blob	BLOB

Java Object Type	Supported JDBC Type
java.sql.Clob	CLOB
java.sql.Struct	STRUCT
java.sql.Ref	REF
java.net.URL	DATALINK
Java class	JAVA_OBJECT
java.sql.RowId	ROWID
java.sql.NClob	NCLOB
java.sql.SQLXML	SQLXML
java.util.Calendar	CHAR, VARCHAR, LONGVARCHAR, DATE, TIME, TIMESTAMP, ARRAY
java.util.Date	CHAR, VARCHAR, LONGVARCHAR, DATE, TIME, TIMESTAMP, ARRAY
java.time.LocalDate	CHAR, VARCHAR, LONGVARCHAR, DATE
java.time.LocalTime	CHAR, VARCHAR, LONGVARCHAR, TIME
java.time.LocalDateTime	CHAR, VARCHAR, LONGVARCHAR, DATE, TIME, TIMESTAMP

Java Object Type	Supported JDBC Type
java.time.OffsetTime	CHAR, VARCHAR, LONGVARCHAR, TIME_WITH_TIMEZONE
java.time.OffsetDatetime	CHAR, VARCHAR, LONGVARCHAR, TIME_WITH_TIMEZONE, TIMESTAMP_WITH_TIMEZONE

B.6 Type Conversions Supported by `ResultSet` getter Methods

TABLE B-6 shows which JDBC types may be returned by `ResultSet` getter methods. This table also shows the conversions used by the `SQLInput` reader methods, except that they use only the recommended conversions.

TABLE B-6 Use of `ResultSet` getter Methods to Retrieve JDBC Data Types

Java Object Type	Recommended JDBC Type	Supported JDBC Type
`getByte`	`TINYINT`	`TINYINT`, `SMALLINT`, `INTEGER`, `BIGINT`, `REAL`, `FLOAT`, `DOUBLE`, `DECIMAL`, `NUMERIC`, `BIT`, `BOOLEAN`, `CHAR`, `VARCHAR`, `LONGVARCHAR`, `ROWID`, `DECFLOAT`
`getShort`	`SMALLINT`	`TINYINT`, `SMALLINT`, `INTEGER`, `BIGINT`, `REAL`, `FLOAT`, `DOUBLE`, `DECIMAL`, `NUMERIC`, `BIT`, `BOOLEAN`, `CHAR`, `VARCHAR`, `LONGVARCHAR`, `DECFLOAT`
`getInt`	`INTEGER`	`TINYINT`, `SMALLINT`, `INTEGER`, `BIGINT`, `REAL`, `FLOAT`, `DOUBLE`, `DECIMAL`, `NUMERIC`, `BIT`, `BOOLEAN`, `CHAR`, `VARCHAR`, `LONGVARCHAR`, `DECFLOAT`
`getLong`	`BIGINT`	`TINYINT`, `SMALLINT`, `INTEGER`, `BIGINT`, `REAL`, `FLOAT`, `DOUBLE`, `DECIMAL`, `NUMERIC`, `BIT`, `BOOLEAN`, `CHAR`, `VARCHAR`, `LONGVARCHAR`, `DECFLOAT`
`getFloat`	`REAL`	`TINYINT`, `SMALLINT`, `INTEGER`, `BIGINT`, `REAL`, `FLOAT`, `DOUBLE`, `DECIMAL`, `NUMERIC`, `BIT`, `BOOLEAN`, `CHAR`, `VARCHAR`, `LONGVARCHAR`, `DECFLOAT`

Java Object Type	Recommended JDBC Type	Supported JDBC Type
getDouble	FLOAT, DOUBLE	TINYINT, SMALLINT, INTEGER, BIGINT, REAL, FLOAT, DOUBLE, DECIMAL, NUMERIC, BIT, BOOLEAN, CHAR, VARCHAR, LONGVARCHAR, DECFLOAT
getBigDecimal	DECIMAL, NUMERIC	TINYINT, SMALLINT, INTEGER, BIGINT, REAL, FLOAT, DOUBLE, DECIMAL, NUMERIC, BIT, BOOLEAN, CHAR, VARCHAR, LONGVARCHAR, DECFLOAT
getBoolean	BIT, BOOLEAN	TINYINT, SMALLINT, INTEGER, BIGINT, REAL, FLOAT, DOUBLE, DECIMAL, NUMERIC, BIT, BOOLEAN, CHAR, VARCHAR, LONGVARCHAR
getString	CHAR, VARCHAR	TINYINT, SMALLINT, INTEGER, BIGINT, REAL, FLOAT, DOUBLE, DECIMAL, NUMERIC, BIT, BOOLEAN, CHAR, VARCHAR, LONGVARCHAR, BINARY, VARBINARY, LONVARBINARY, DATE, TIME, TIMESTAMP, DATALINK, NCHAR, NVARCHAR, LONGNVARCHAR, TIME_WITH_TIMEZONE, TIMESTAMP_WITH_TIMEZONE, JSON, DECFLOAT
getNString	NCHAR, NVARCHAR	TINYINT, SMALLINT, INTEGER, BIGINT, REAL, FLOAT, DOUBLE, DECIMAL, NUMERIC, BIT, BOOLEAN, CHAR, VARCHAR, LONGVARCHAR, BINARY, VARBINARY, LONVARBINARY, DATE, TIME, TIMESTAMP, DATALINK, NCHAR, NVARCHAR, LONGNVARCHAR, TIME_WITH_TIMEZONE, TIMESTAMP_WITH_TIMEZONE, JSON, DECFLOAT
getBytes	BINARY, VARBINARY	BINARY, VARBINARY, LONGVARBINARY

Deleted:

Formatted: Font color: Text 1

Formatted: Font color: Text 1

Java Object Type	Recommended JDBC Type	Supported JDBC Type
getDate	DATE	CHAR, VARCHAR, LONGVARCHAR, DATE, TIMESTAMP
getTime	TIME	CHAR, VARCHAR, LONGVARCHAR, TIME, TIMESTAMP
getTimestamp	TIMESTAMP	CHAR, VARCHAR, LONGVARCHAR, DATE, TIME, TIMESTAMP
getAsciiStream	LONGVARCHAR	CHAR, VARCHAR, LONGVARCHAR, BINARY, VARBINARY, LONGVARBINARY, CLOB, NCLOB
getBinaryStream	LONGVARBINARY	BINARY, VARBINARY, LONGVARBINARY
getCharacterStream	LONGVARCHAR	CHAR, VARCHAR, LONGVARCHAR, BINARY, VARBINARY, LONGVARBINARY, CLOB, NCHAR, NVARCHAR, LONGNVARCHAR, NCLOB, SQLXML
getNCharacterStream	LONGNVARCHAR	CHAR, VARCHAR, LONGVARCHAR, BINARY, VARBINARY, LONGVARBINARY, CLOB, NCHAR, NVARCHAR, LONGNVARCHAR, NCLOB, SQLXML
getClob	CLOB	CLOB, NCLOB

Java Object Type	Recommended JDBC Type	Supported JDBC Type
getNClob	NCLOB	CLOB, NCLOB
getBlob	BLOB	BLOB
getArray	ARRAY	ARRAY
getRef	REF	REF
getURL	DATALINK	DATALINK
getObject	STRUCT, JAVA_OBJECT	TINYINT, SMALLINT, INTEGER, BIGINT, REAL, FLOAT, DOUBLE, DECIMAL, NUMERIC, BIT, BOOLEAN, CHAR, VARCHAR, LONGVARCHAR, BINARY, VARBINARY, LONVARBINARY, DATE, TIME, TIMESTAMP, CLOB, BLOB, ARRAY, REF, DATALINK, STRUCT, JAVA_OBJECT, ROWID, NCHAR, NVARCHAR, LONGNVARCHAR, NCLOB, SQLXML, TIME_WITH_TIMEZONE, TIMESTAMP_WITH_TIMEZONE, JSON, DECFLOAT
getRowId	ROWID	ROWID

Formatted: Font color: Text 1

Java Object Type	Recommended JDBC Type	Supported JDBC Type
getSQLXML	SQLXML	SQLXML

APPENDIX B Data Type Conversion Tables

APPENDIX B Data Type Conversion Tables