# Support for Microsoft SQL Server database

The HotRod SQL Server adapter automatically maps known database column types to DAO Java types. In most of the cases this default Java type is well suited to handle the database values. However, when needed the default Java type of a property can be overridden by a custom type if it's provided by the developer.

## Default Java Types

If a custom Java type is not specified HotRod will use the following rules to decide which Java type to use for each SQL Server column. In yellow is the DAO property type. In parenthesis the actual object type returned by the SQL Server JDBC driver, that on occasions may be different.

Please note that the Java types for the SQL Server columns may vary depending on the specific version and variant of the RDBMS, the operating system where the database engine is running, and the JDBC driver version.

|  |  |
| --- | --- |
| **SQL Server Column Type** | **Default Java Type** |
| BIT | java.lang.Byte |
| TINYINT | java.lang.Byte |
| SMALLINT | java.lang.Short |
| INT | java.lang.Integer |
| BIGINT | java.lang.Long |
| DECIMAL(p,s),  DEC(p,s),  NUMERIC(p,s) | If neither p or s are specified, i.e. DECIMAL(18,0):   * java.lang.Long   If s is specified and different from zero the Java type is:   * java.math.BigDecimal   if s is not specified or it's zero:   * if p <= 2: java.lang.Byte * if 2 < p <= 4: java.lang.Short * if 4 < p <= 9: java.lang.Integer * if 8 < p <= 18: java.lang.Long * if p > 18: java.math.BigInteger |
| MONEY,  SMALLMONEY | java.math.BigDecimal |
| FLOAT(n) | If n is not specified, i.e. a FLOAT(53):   * java.lang.Double   if n is specified:   * if n <= 24: java.lang.Float * if n >= 25: java.lang.Double |
| REAL | java.lang.Float  **Note**: REAL is equivalent to FLOAT(24), |
| CHAR(n),  CHARACTER(n),  VARCHAR(n|MAX),  CHARVARYING(n|MAX),  CHARACTERVARYING(n|MAX),  NCHAR(n),  NATIONAL CHAR(n),  NATIONAL CHARACTER(n),  NVARCHAR(n|MAX),  NATIONAL CHAR VARYING(n|MAX),  NATIONAL CHARACTER VARYING(n|MAX),  TEXT,  NTEXT | java.lang.String |
| DATE | java.sql.Date |
| DATETIME,  SMALLDATETIME | java.util.Date |
| DATETIME2(n),  DATETIMEOFFSET(n) | java.sql.Timestamp |
| TIME(n) | If n is not specified, i.e. TIME(7):   * java.sql.Timestamp   If n is specified:   * If n <=3: java.sql.Time * If n >=4: java.sql.Timestamp |
| BINARY(n),  VARBINARY(n|MAX),  IMAGE | byte[] |
| HIERARCHYID | byte[] |
| ROWVERSION | java.lang.Object Cannot insert, nor update by PK. Selects and deletes work normally. Rows can be “updated by example” when excluding this column. |
| UNIQUEIDENTIFIER | java.lang.String |
| SQL\_VARIANT | This type is not supported by the JDBC driver 4.0 provided by SQL Server. A workaround, at least to read it, is to cast this column to a different supported type (maybe using a view or a select) as in th expression: CAST(**<column>** AS <**type>**) |
| XML | java.lang.String \* |
| GEOGRAPHY | byte[] \*\* |
| GEOMETRY | byte[] \*\* |
| (pseudo column) <col> as <expression> | Type depends on expression type. Cannot insert, nor update by PK. Selects and deletes work normally. Rows can be “updated by example” when excluding this column. |

\* Must be a well-formed XML String. Depending on the column definition it may also need to be a valid XML String.

\*\* These data types represent well-formed binary data as specified by the “[MS-SSCLRT]: Microsoft SQL Server CLR Types Serialization Formats” document at https://msdn.microsoft.com/en-us/library/ee320529.aspx.

## Custom Java Types

To override the default java type see the reference section for the tables, views, and selects. The example Custom DAO Property Java Types shows a some cases where a custom type overrides the default type. To do it add a <column> tag in a <table>, <view>, or <select> definition as in:

<table name=*"my\_table"*>

<column name=*"price"* java-type=*"java.lang.Double"* jdbc-type=*"NUMERIC"* />

</table>

This configuration will force the property type to java.lang.Double instead of java.math.BigDecimal (the default type).