# Support for HyperSQL (HSQLDB) database

The HotRod HyperSQL 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 HyperSQL column. In yellow is the DAO property type. In parenthesis the actual object type returned by the HyperSQL JDBC driver, that on occasions may be different.

Please note that the Java types for the HyperSQL 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.

|  |  |
| --- | --- |
| **HyperSQL Column Type** | **Default Java Type** |
| TINYINT | java.lang.Byte |
| SMALLINT | java.lang.Short |
| INTEGER | java.lang.Integer |
| BIGINT | java.lang.Long |
| DECIMAL(p,s)  NUMERIC(p,s) | If neither p or s are specified:   * java.math.BigInteger   If s is specified and different from zero the Java type is:   * java.math.BigDecimal   if s is not specified or specified with a value of 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 |
| REAL, FLOAT,  DOUBLE | java.lang.Double |
| CHAR(n),  CHARACTER(n),  VARCHAR(n),  CHARACTER VARYING(n),  CHAR VARYING(n),  LONGVARCHAR(n),  CLOB(n) | java.lang.String |
| DATE | java.sql.Date |
| TIME(n),  TIME(n) WITH TIME ZONE | java.sql.Time |
| TIMESTAMP(n),  TIMESTAMP(n) WITH TIME ZONE | java.sql.Timestamp |
| BINARY(n),  VARBINARY(n),  BLOB(n) | byte[] |
| BOOLEAN | java.lang.Boolean |
| OTHER | java.lang.Object |
| <type> ARRAY | java.lang.Object |
| INTERVAL <qualifier> | java.lang.Object |
| BIT(n),  BIT VARYING(n) | java.lang.Object |

## 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).