# Support for MySQL database

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

Please note that the Java types for the MySQL 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.

|  |  |
| --- | --- |
| **MySQL Column Type** | **Default Java Type** |
| TINYINT,  INT1 | java.lang.Byte |
| TINYINT UNSIGNED,  INT1 UNSIGNED | java.lang.Short |
| SMALLINT,  INT2 | java.lang.Short |
| SMALLINT UNSIGNED,  INT2 UNSIGNED | java.lang.Integer |
| MEDIUMINT,  INT3 | java.lang.Integer |
| MEDIUMINT UNSIGNED,  INT3 UNSIGNED | java.lang.Integer |
| INTEGER,  INT4 | java.lang.Integer |
| INTEGER UNSIGNED,  INT4 UNSIGNED | java.lang.Long |
| BIGINT,  INT8 | java.lang.Long |
| BIGINT UNSIGNED,  INT8 UNSIGNED | java.math.BigInteger |
| DECIMAL(p,s),  NUMERIC(p,s) | If neither p or s are specified:   * java.lang.Long   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 9 < p <= 18: java.lang.Long * if p > 18: java.math.BigInteger |
| FLOAT(n),  FLOAT4(n),  FLOAT(n) UNSIGNED,  FLOAT4(n) UNSIGNED | If n is not specified:   * java.lang.Float   If n is specified:   * if n <= 24: java.lang.Float * if n >= 25: java.lang.Double |
| DOUBLE,  DOUBLE PRECISION,  REAL,  FLOAT8,  DOUBLE UNSIGNED,  DOUBLE PRECISION UNSIGNED,  REAL UNSIGNED,  FLOAT8 UNSIGNED | java.lang.Double |
| CHAR(n),  VARCHAR(n),  TINYTEXT,  TEXT,  MEDIUMTEXT,  LONGTEXT | java.lang.String |
| DATE,  YEAR | java.sql.Date |
| TIME | java.sql.Time |
| DATETIME,  TIMESTAMP | java.sql.Timestamp |
| TINYBLOB,  BLOB,  MEDIUMBLOB,  LONGBLOB | byte[] |
| ENUM | java.lang.String |
| SET | java.lang.String |

## 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).