# Support for SAP Adaptive Server (ex-Sybase) database

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

Please note that the Java types for the SAP ASE 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.

|  |  |
| --- | --- |
| **SAP ASE Column Type** | **Default Java Type** |
| BIT | java.lang.Byte |
| TINYINT | java.lang.Byte |
| UNSIGNED TINYINT | java.lang.Byte\* |
| SMALLINT | java.lang.Short |
| UNSIGNED SMALLINT | java.lang.Integer |
| INT,  INTEGER | java.lang.Integer |
| UNSIGNED INT,  UNSIGNED INTEGER | java.lang.Long |
| BIGINT | java.lang.Long |
| UNSIGNED BIGINT | java.math.BigInteger |
| DECIMAL(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),  REAL,  DOUBLE PRECISION | java.lang.Double \*\* |
| CHAR(n),  VARCHAR(n),  UNICHAR(n),  UNIVARCHAR(n),  NCHAR(n),  NVARCHAR(n),  TEXT,  UNITEXT,  SYSNAME,  LONGSYSNAME | java.lang.String |
| DATE | java.sql.Date |
| DATETIME,  SMALLDATETIME,  BIGDATETIME | java.sql.Timestamp |
| TIME,  BIGTIME | java.sql.Timestamp |
| BINARY(n),  VARBINARY(n|MAX),  IMAGE | byte[] |

\* The SAP ASE JDBC driver does not provide information to differentiate TINYINT from UNSIGNED TINYINT. If you happen to have an UNSIGNED TINYINT column you may want to use the custom type java.lang.Short for it, instead of the default type java.lang.Byte. Or, try to avoid using the UNSIGNED TINYINT type altogether, if possible.

\*\* The SAP ASE JDBC driver does not provide information to differentiate float of different precisions. FLOAT with precision 15 or less could be treated as a Java java.lang.Float. However, since there’s no way to find out the precision of the FLOAT the default type is, regardless of their precision, Double.

## 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).