DBSeeder - Relational Database Data Generator.

Table of Contents

1. Introduction
1.1 RDBMS Overview
1.2 RDBMS Directory
1.3 Performance Example
2. Data
2.1 Database Schema
2.2 Construction of the Dummy Data Content
3. Installation
4. Operating Instructions
4.1 Scripts
4.2 Operation Possibilities
4.3 Control Parameters
4.4 Statistics
5. RDBMS Specific Technical Details
6. trino

1. Introduction

DBSeeder allows the flexible generation of large amounts of anonymised random dummy data for selected relational database systems (RDBMS) - useful e.g. for stress testing.

The database schema underlying the data generation can be freely defined. The names of the database, the schema and the user can be freely chosen, unless the respective database management system contains restrictions. If the selected database, schema or user already exist, they are deleted with all including data. **DBSeeder** then creates the selected database, schema or user and generates the desired dummy data. A maximum of 2 147 483 647 rows can be generated per database table. The database schema to be used, that is, the required database tables can be user defined using a JSON file. Details can be found here: 2.1 Database Schema.

Currently, depending on the capabilities of the specific RDBMS, the following functionalities and data types are supported:

- constraints
  - foreign (referential) key
  - not null constraint
  - primary key
  - unique (alternate) key
- data types
  - BIGINT - large integer
  - BLOB - large binary object
  - CLOB - large character Object
  - TIMESTAMP - timestamp including date
  - VARCHAR - variable text

The database systems considered meet the following conditions:

1. The database system is freely available in a documented docker image for testing purposes.
2. The database system provides a well documented JDBC interface.
3. A complete documentation of the SQL commands is available.

1.1 RDBMS Overview

RDBMS	Ticker Symbol(s)	RDBMS Versions	Latest JDBC
AgensGraph	agens	v2.1.1 - v2.1.3	1.4.2-c1
Apache Derby	derby, derby_emb	10.15.2.0	10.15.2.0
CockroachDB	cockroach	v20.2.5 - v21.1.2	see PostgreSQL
CrateDB	cratedb	4.1.6 - 4.5.1	2.6.0

RDBMS	Ticker Symbol(s)	RDBMS Versions	Latest JDBC
CUBRID	cubrid	10.2 - 11.0	11.0.1.0286
Exasol	exasol	6.2.8-d1 - 7.0.10	7.0.7
Firebird	firebird	3.0.5 - v4.0.0rc1	4.0.3.java11
H2 Database Engine	h2, h2_emb	1.4.200	1.4.200
HSQLDB	hsqldb, hsqldb_emb	2.5.1 - 2.6.0	2.6.0
IBM Db2 Database	ibmdb2	11.5.1.0 - 11.5.5.1	11.5.5.0
IBM Informix	informix	14.10 FC3DE - 14.10.FC5DE	4.50.4.1
MariaDB Server	mariadb	10.4.13 - 10.6.1	2.7.3
Mimer SQL	mimer	v11.0.3c - v11.0.5a	3.40
MonetDB	monetdb	Jun2020-SP1 - Oct2020-SP5	3.0.jre8
MySQL Database	mysql	8.0.20 - 8.0.25	8.0.25
OmniSciDB	omnisci	5.6.1	5.6.0
Oracle Database	oracle	12c - 19c	21.1.0.0
Percona Server for MySQL	percona	8.0.23-14	see MySQL
PostgreSQL	postgresql	12.3 - 13.3	42.2.20
SQL Server	sqlserver	2019-latest	9.2.1.jre15
SQLite	sqlite	3.32.0 - 3.32.3	3.34.0
trino	mysql_trino,	339 - 358	358
	oracle_trino,		
	postgresql_trino,		
	sqlserver_trino		
VoltDB	voltdb	9.2.1	10.1.1
YugabyteDB	yugabyte	2.2.2.0-b15 - 2.7.1.1-b1	42.2.7-yb-3

1.2 RDBMS Directory

The following database systems are included in the current version of **DBSeeder**:

- AgensGraph
  - client only version
  - commercial, open source
  - derived from PostgreSQL
  - property graph model and relational model
  - **see technical details here**
- Apache Derby
  - client and embedded version
  - open source
  - relational model
  - **see technical details here**
- CockroachDB
  - client only version
  - commercial, open source
  - compatible with PostgreSQL JDBC
  - relational model
  - **see technical details here**
- CrateDB
  - client only version
  - commercial, open source
  - compatible with PostgreSQL

    - relational model
    - **see technical details here**
- CUBRID
    - client only version
    - compatible with MySQL
    - open source
    - relational model
    - **see technical details here**
- Exasol
    - client only version
    - commercial
    - in-memory, column-oriented, relational model
    - **see technical details here**
- Firebird
    - client and embedded (not supported here) version
    - open source
    - relational model
    - **see technical details here**
- H2 Database Engine
    - client and embedded version
    - compatible with HSQLDB, PostgreSQL
    - open source
    - relational model
    - **see technical details here**
- HSQLDB
    - client and embedded version
    - open source
    - relational model
    - **see technical details here**
- IBM Db2 Database
    - client only version
    - commercial
    - relational model
    - **see technical details here**
- IBM Informix
    - client only version
    - commercial
    - relational model
    - **see technical details here**
- MariaDB Server
    - client only version
    - derived from MySQL
    - open source
    - relational model
    - **see technical details here**
- Mimer SQL
    - client only version
    - commercial
    - relational model
    - **see technical details here**
- MonetDB
    - client only version
    - open source
    - column-oriented relational model
    - **see technical details here**
- MySQL Database
    - client only version
    - open source
    - relational model
    - **see technical details here**
- OmniSciDB

    - client only version
    - commercial, open source
    - GPU and CPU version
    - relational model
    - **see technical details here**
- Oracle Database
    - client only version
    - commercial
    - relational model
    - **see technical details here**
- Percona Server for MySQL
    - client only version
    - commercial, open source
    - derived from MySQL
    - relational model
    - **see technical details here**
- PostgreSQL
    - client only version
    - open source
    - relational model
    - **see technical details here**
- SQL Server
    - client only version
    - commercial
    - derived from Adaptive Server Enterprise
    - relational model
    - **see technical details here**
- SQLite
    - commercial, open source
    - embedded only version
    - relational model
    - **see technical details here**
- trino
    - compatible with Accumulo, Cassandra, Elasticsearch, Hive, Kudu, MongoDB, MySQL, Pinot, PostgreSQL, Redis, Redshift
    - distributed query engine
    - open source
    - **see technical details here**

For the RDBMS MySQL, Oracle, PostgreSQL and SQL Server the JDBC driver from trino can optionally be used instead of the original JDBC driver. The prerequisite for this is that trino is either installed locally (Linux) or is available as a Docker container (Linux and Windows). Details can be found here: 6. trino.

- VoltDB
    - client only version
    - commercial, open source
    - derived from H-Store, HSQLDB
    - in-memory relational model
    - **see technical details here**
- YugabyteDB
    - client only version
    - commercial, open source
    - compatible with Cassandra, PostgreSQL, Redis
    - derived from PostgreSQL, RocksDB
    - inspired by Cloud Spanner
    - relational model
    - **see technical details here**

1.3 Performance Example

An interesting side effect of working with **`DBSeeder`** is the ability to compare the performance of the data generation (`INSERT`) between the individual RDBMSs (e.g. Version 2.9.1 Windows 10):

ticker symbol	DBMS	db type	runtime in ms	start time	end time	host name	no. cores	operating system
hsqldb_emb	HSQLDB	embedded	3081	11.06.2021 13:23	11.06.2021 13:23	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
h2_emb	H2 Database Engine	embedded	3301	11.06.2021 13:22	11.06.2021 13:22	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
sqlite	SQLite	embedded	4138	11.06.2021 14:43	11.06.2021 14:43	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
postgresql	PostgreSQL	client	4264	11.06.2021 14:18	11.06.2021 14:18	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
mariadb	MariaDB Server	client	4287	11.06.2021 13:33	11.06.2021 13:34	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
agens	AgensGraph	client	4288	11.06.2021 13:13	11.06.2021 13:13	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
exasol	Exasol	client	4515	11.06.2021 13:19	11.06.2021 13:19	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
sqlserver	MS SQL Server	client	5210	11.06.2021 14:27	11.06.2021 14:27	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
cratedb	CrateDB	client	5440	11.06.2021 13:14	11.06.2021 13:15	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
cockroach	CockroachDB	client	5657	11.06.2021 13:14	11.06.2021 13:14	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
mysql	MySQL Database	client	6463	11.06.2021 13:36	11.06.2021 13:36	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
cubrid	CUBRID	client	6881	11.06.2021 13:15	11.06.2021 13:15	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
percona	Percona Server for MySQL	client	7064	11.06.2021 14:17	11.06.2021 14:18	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
h2	H2 Database Engine	client	8087	11.06.2021 13:22	11.06.2021 13:22	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
mimer	Mimer SQL	client	9631	11.06.2021 13:34	11.06.2021 13:34	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
oracle	Oracle Database	client	12854	11.06.2021 13:57	11.06.2021 13:57	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
yugabyte	YugabyteDB	client	13744	11.06.2021 14:44	11.06.2021 14:45	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
voltdb	VoltDB	client	14344	11.06.2021 14:44	11.06.2021 14:44	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
derby	Apache Derby	client	15441	11.06.2021 13:16	11.06.2021 13:16	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
derby_emb	Apache Derby	embedded	16526	11.06.2021 13:16	11.06.2021 13:17	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
hsqldb	HSQLDB	client	18594	11.06.2021 13:23	11.06.2021 13:23	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
informix	IBM Informix	client	20650	11.06.2021 13:32	11.06.2021 13:33	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
monetdb	MonetDB	client	21775	11.06.2021 13:34	11.06.2021 13:35	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
ibmdb2	IBM Db2 Database	client	24832	11.06.2021 13:28	11.06.2021 13:29	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
omnisci	OmniSciDB	client	39068	11.06.2021 13:54	11.06.2021 13:55	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
firebird	Firebird	client	96473	11.06.2021 13:20	11.06.2021 13:21	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
postgresql_trino	PostgreSQL	trino	431792	11.06.2021 14:19	11.06.2021 14:26	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
sqlserver_trino	MS SQL Server	trino	896016	11.06.2021 14:28	11.06.2021 14:43	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
mysql_trino	MySQL Database	trino	929016	11.06.2021 13:37	11.06.2021 13:53	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0
oracle_trino	Oracle Database	trino	981940	11.06.2021 14:00	11.06.2021 14:16	jfww-win-dt-21	32	amd64 / Windows 10 / 10.0

2. Data

2.1 Database Schema

The underlying database schema is defined in a JSON-based parameter file and the associated program code is generated and compiled with the script `scripts/run_db_seeder_generate_schema`. To validate the database schema in the JSON parameter file, the JSON schema file `db_seeder_schema.schema.json` in the directory `src/main/resources` is used.

2.1.1 Structure of the Database Schema Definition File

The definition of a database schema consists of the object `global` with the global parameters and the array `tables`, which contains the definition of the database tables.

2.1.1.1 `globals` - Global Parameters

- `defaultNumberOfRows` - default value for the number of table rows to be generated, if no value is specified in the table definition
- `encodingISO_8859_1` - a string with Western Latin characters is inserted into generated character columns
- `encodingUTF_8` - a string with simplified Chinese characters is inserted into generated character columns specified in the table definition
- `nullFactor` - determines the proportion of NULL values in optional columns and must be between 2 and 99 (inclusive): 2 means 50%, 4 means 25%, 10 means 10%, etc., default value is 4

2.1.1.2 `tables` - Database Table Definitions

- `tableName` - database table name
- `numberOfRows` - number of table rows to be generated
- `columns` - an array of column definitions
  - `columnName` - column name
  - `dataType` - data type, is one of BIGINT, BLOB, CLOB, TIMESTAMP or VARCHAR
  - `size` - for data type VARCHAR the maximum size of the column value
  - `precision` - currently not used
  - `notNull` - is a NULL value allowed ?
  - `primaryKey` - is this the primary key column ?
  - `references` - an array of foreign key definitions
    - `referenceTable` - name of the reference database table
    - `referenceColumn` - name of the reference column
  - `defaultValueInteger` - default value for integer columns
  - `defaultValueString` - default value for alphanumeric columns

    - `lowerRangeInteger` - lower limit for an integer column, requires also an upper limit
    - `lowerRangeString` - lower limit for an alphanumeric column, requires also an upper limit
    - `upperRangeInteger` - upper limit for an integer column
    - `upperRangeString` - upper limit for an alphanumeric column
    - `validValuesInteger` - valid values for an integer column
    - `validValuesString` - valid values for an alphanumeric column
- `tableConstraints` - an array of table constraint definitions
    - `constraintType` - constraint type, is one of FOREIGN, PRIMARY or UNIQUE
    - `columns` - an arry with the names of the affected columns
    - `referenceTable` - name of the reference database table, only for foreign keys
    - `referenceColumns` - an arry with the names of the affected reference columns, only for foreign keys

Only either a range restriction (`lowerRange...`, `upperRange...`) or a value restriction (`validValues...`) may be specified for each column.

2.1.2 Mapping of Data Types in the JDBC Driver

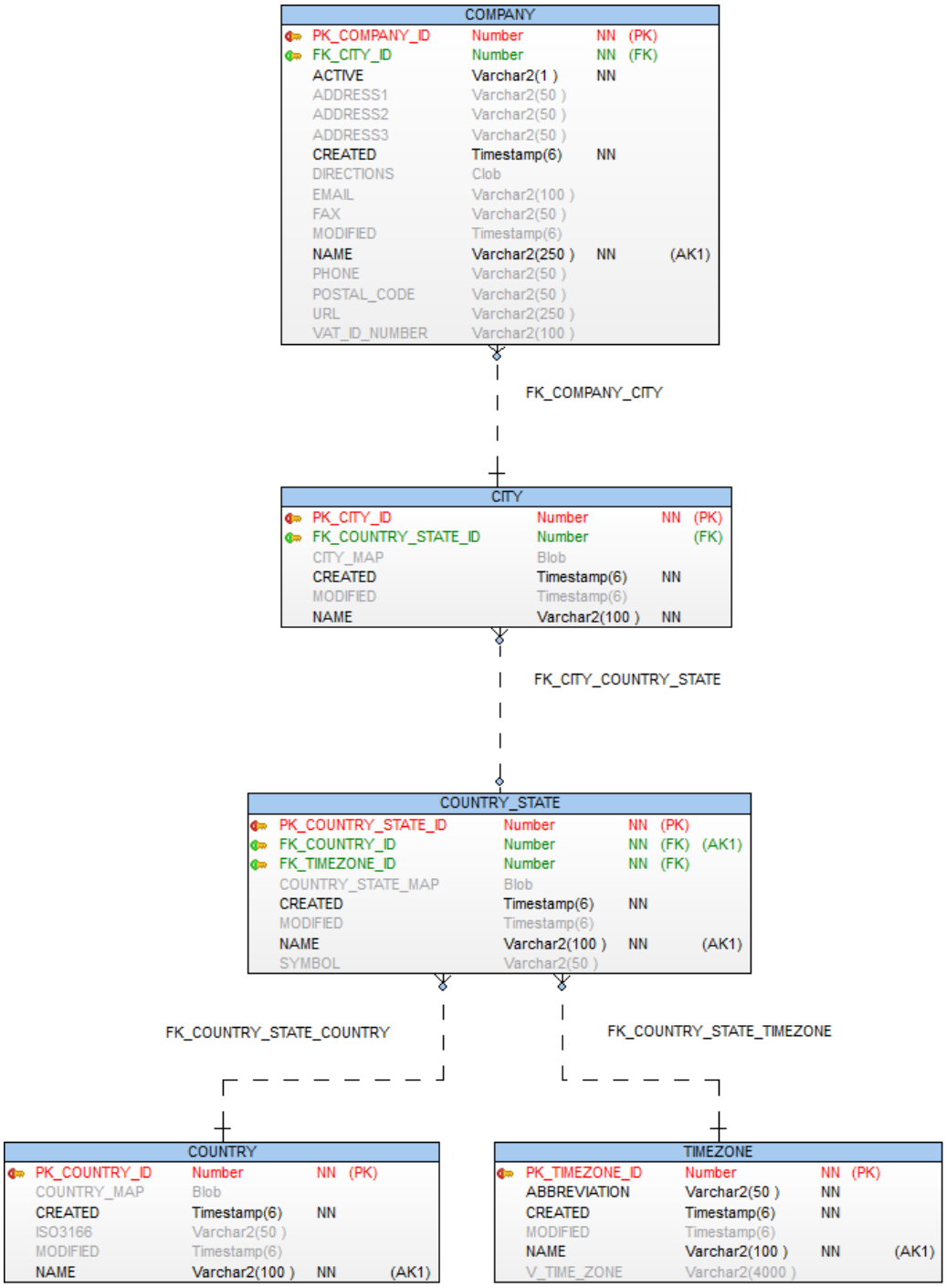
Data Type	JDBC Method
`BIGINT`	`setLong`
`BLOB`	`setBytes`
`CLOB`	`setString`
`TIMESTAMP`	`setTimestamp`
`VARCHAR`	`setNString` (Firebird, MariaDB, MS SQL SERVER and Oracle)
	`setString` (else)

2.1.3 Example File `db_seeder_schema.company.json` in the Directory `resources/json`

This file contains the definition of a simple database schema consisting of the database tables CITY, COMPANY, COUNTRY, COUNTRY_STATE and TIMEZONE.

The abbreviations in the following illustration (created with Toad Data Modeler) mean:

- (AK1) - alternate key (unique key)
- FK - foreign key
- NN - not null
- PK - primary key



2.2 Construction of the Dummy Data Content

The proportion of `NULL` values in optional columns is defined by the global parameter `nullFactor`.

All methods for generating column contents can be overwritten if necessary.

2.2.1 BIGINT

Java method: `getContentBigint`

- If the column parameter `validValuesInteger` is defined in the database schema, a random value is taken from it.
- If the column parameters `lowerRangeInteger` and `upperRangeInteger` are defined in the database schema, a random value is taken from this interval.
- Otherwise the counter for the current row (row number) is used.

2.2.2 BLOB

Java method: `getContentBlob`

- The content of the file `blob.png` from the resource directory (`src/main/resources`) is loaded into these columns.This file contains the company logo of Konnexions GmBH.

2.2.3 CLOB

Java method: `getContentClob`

- The content of the file `clob.md` from the resource directory (`src/main/resources`) is loaded into these columns. This file contains the text of the Konnexions Public License (KX-PL).

2.2.4 TIMESTAMP

Java method: `getContentTimestamp`

- A randomly generated timestamp is assigned to all columns that can contain temporal data.

2.2.5 VARCHAR

Java method: `getContentVarchar`

- If the column parameter `validValuesString` is defined in the database schema, a random value is taken from it.
- If the column parameters `lowerRangeString` and `upperRangeString` are defined in the database schema, a random value is taken from this interval.
- Otherwise content of the column is constructed depending on the row number and the encoding flags as follows:
  - ASCII (all rows where the index modulo 3 is 0):
    - column name in capital letters
    - underscore _
    - current row number left-justified
  - ISO 8859 1 (all rows where the index modulo 3 is 1) :
    - column name in capital letters
    - underscore _
    - a string containing specific Western European characters with accent (e.g. French, Portuguese or Spanish)
    - underscore _
    - current row number left-justified
  - the ISO 8859 1 version can be prevented by choosing `encodingISO_8859_1 false` in the database schema definition
  - UTF-8 (all rows where the index modulo 3 is 2):
    - column name in capital letters
    - underscore _
    - a string containing simplified Chinese characters
    - underscore _
    - current row number left-justified
  - the UTF-8 version can be prevented by choosing `encodingUTF_8 false` in the database schema definition
  - If the resulting value exceeds the permissible column size, the value is shortened accordingly from the left

2.2.6 Examples

1. Table CITY

CITY | Enter a SQL expression to filter results (use Ctrl+Space)

Grid | Text

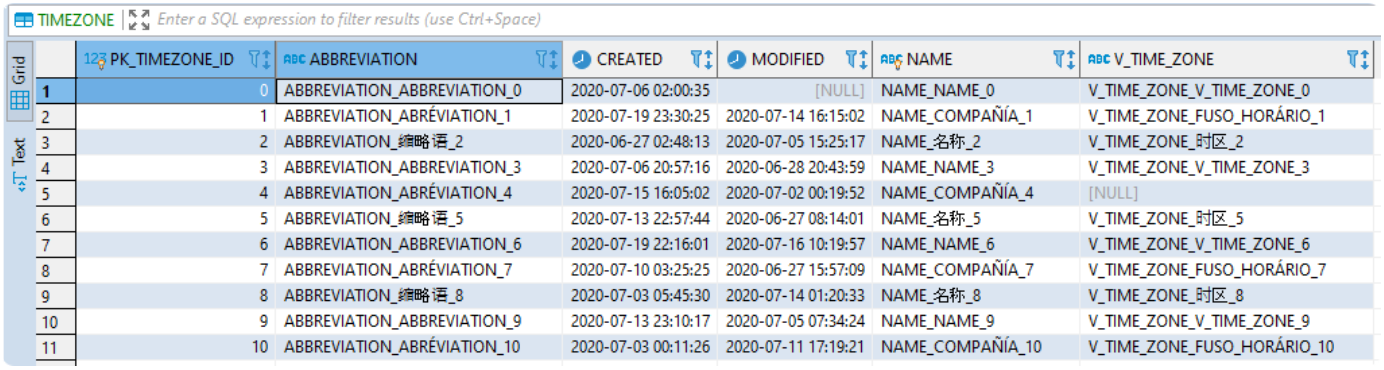
	PK_CITY_ID	FK_COUNTRY_STATE_ID	CITY_MAP	CREATED	MODIFIED	NAME
1	0	417	PNG IHDR Õ Ü çOä... [19664]	2020-07-07 21:16:36	2020-07-09 13:58:19	NAME_NAME_0
2	1	154	[NULL]	2020-07-01 15:29:56	2020-07-11 13:22:42	NAME_COMPAÑÍA_1
3	2	307	PNG IHDR Õ Ü çOä... [19664]	2020-07-03 17:37:20	[NULL]	NAME_名称_2
4	3	270	PNG IHDR Õ Ü çOä... [19664]	2020-07-03 08:17:33	2020-07-03 09:21:59	NAME_NAME_3
5	4	201	PNG IHDR Õ Ü çOä... [19664]	2020-07-09 02:37:50	[NULL]	NAME_COMPAÑÍA_4
6	5	119	PNG IHDR Õ Ü çOä... [19664]	2020-07-13 23:50:00	2020-07-12 20:48:13	NAME_名称_5
7	6	[NULL]	[NULL]	2020-07-20 18:08:54	[NULL]	NAME_NAME_6
8	7	[NULL]	PNG IHDR Õ Ü çOä... [19664]	2020-07-10 02:51:42	2020-07-13 22:11:40	NAME_COMPAÑÍA_7
9	8	261	PNG IHDR Õ Ü çOä... [19664]	2020-07-13 14:53:57	[NULL]	NAME_名称_8
10	9	584	[NULL]	2020-07-13 20:23:48	[NULL]	NAME_NAME_9
11	10	[NULL]	PNG IHDR Õ Ü çOä... [19664]	2020-07-04 17:37:34	[NULL]	NAME_COMPAÑÍA_10
12	11	35	PNG IHDR Õ Ü çOä... [19664]	2020-07-19 01:25:20	2020-06-27 19:33:15	NAME_名称_11
13	12	553	PNG IHDR Õ Ü çOä... [19664]	2020-06-28 16:59:18	[NULL]	NAME_NAME_12
14	13	401	PNG IHDR Õ Ü çOä... [19664]	2020-07-09 15:50:28	2020-07-11 00:37:42	NAME_COMPAÑÍA_13
15	14	127	PNG IHDR Õ Ü çOä... [19664]	2020-07-15 07:32:40	2020-07-01 18:59:11	NAME_名称_14
16	15	296	PNG IHDR Õ Ü çOä... [19664]	2020-07-10 12:51:31	2020-07-17 11:12:31	NAME_NAME_15
17	16	[NULL]	PNG IHDR Õ Ü çOä... [19664]	2020-07-18 19:15:29	2020-07-11 12:23:04	NAME_COMPAÑÍA_16
18	17	523	PNG IHDR Õ Ü çOä... [19664]	2020-07-08 23:38:14	2020-07-09 19:58:29	NAME_名称_17
19	18	230	PNG IHDR Õ Ü çOä... [19664]	2020-07-01 01:32:05	2020-07-11 15:13:29	NAME_NAME_18
20	19	275	[NULL]	2020-07-02 19:47:10	2020-06-28 15:22:39	NAME_COMPAÑÍA_19
21	20	274	PNG IHDR Õ Ü çOä... [19664]	2020-07-19 23:43:47	2020-07-04 22:07:54	NAME_名称_20
22	21	[NULL]	[NULL]	2020-07-11 22:24:56	[NULL]	NAME_NAME_21
23	22	[NULL]	PNG IHDR Õ Ü çOä... [19664]	2020-07-08 03:42:43	2020-07-01 21:37:16	NAME_COMPAÑÍA_22
24	23	[NULL]	PNG IHDR Õ Ü çOä... [19664]	2020-06-27 08:43:04	2020-06-29 23:47:39	NAME_名称_23
25	24	158	PNG IHDR Õ Ü çOä... [19664]	2020-06-28 20:05:01	2020-07-09 22:41:36	NAME_NAME_24

2. Table COUNTRY

COUNTRY | Enter a SQL expression to filter results (use Ctrl+Space)

	PK_COUNTRY_ID	COUNTRY_MAP	CREATED	ISO3166	MODIFIED	NAME
1	0	PNG IHDR Õ Ü çOä... [19664]	2020-06-28 19:36:46	[NULL]	2020-06-30 00:33:11	NAME_NAME_0
2	1	PNG IHDR Õ Ü çOä... [19664]	2020-07-19 07:16:11	ISO3166_CÓDIGO 3166_1	2020-06-27 23:12:01	NAME_COMPAÑÍA_1
3	2	PNG IHDR Õ Ü çOä... [19664]	2020-07-08 22:39:47	ISO3166_ISO 3166标准_2	2020-07-16 18:37:01	NAME_名称_2
4	3	[NULL]	2020-07-21 14:58:16	ISO3166_ISO3166_3	2020-07-16 01:14:12	NAME_NAME_3
5	4	PNG IHDR Õ Ü çOä... [19664]	2020-07-01 19:00:23	ISO3166_CÓDIGO 3166_4	2020-06-29 07:27:24	NAME_COMPAÑÍA_4
6	5	PNG IHDR Õ Ü çOä... [19664]	2020-07-05 00:54:26	ISO3166_ISO 3166标准_5	2020-07-08 02:20:09	NAME_名称_5
7	6	PNG IHDR Õ Ü çOä... [19664]	2020-07-06 22:36:40	ISO3166_ISO3166_6	2020-07-04 08:24:18	NAME_NAME_6
8	7	PNG IHDR Õ Ü çOä... [19664]	2020-07-20 07:33:21	ISO3166_CÓDIGO 3166_7	2020-06-28 06:10:36	NAME_COMPAÑÍA_7
9	8	PNG IHDR Õ Ü çOä... [19664]	2020-07-21 14:34:52	ISO3166_ISO 3166标准_8	[NULL]	NAME_名称_8
10	9	PNG IHDR Õ Ü çOä... [19664]	2020-07-08 08:27:06	ISO3166_ISO3166_9	2020-07-16 00:17:52	NAME_NAME_9
11	10	[NULL]	2020-06-29 04:43:50	ISO3166_CÓDIGO 3166_10	2020-07-14 19:08:27	NAME_COMPAÑÍA_10
12	11	PNG IHDR Õ Ü çOä... [19664]	2020-06-29 11:03:58	ISO3166_ISO 3166标准_11	[NULL]	NAME_名称_11
13	12	PNG IHDR Õ Ü çOä... [19664]	2020-07-16 15:39:56	ISO3166_ISO3166_12	2020-06-27 16:50:58	NAME_NAME_12
14	13	PNG IHDR Õ Ü çOä... [19664]	2020-07-12 01:44:07	[NULL]	[NULL]	NAME_COMPAÑÍA_13
15	14	PNG IHDR Õ Ü çOä... [19664]	2020-07-12 07:01:06	ISO3166_ISO 3166标准_14	2020-07-18 06:54:10	NAME_名称_14
16	15	PNG IHDR Õ Ü çOä... [19664]	2020-07-09 15:54:56	[NULL]	[NULL]	NAME_NAME_15
17	16	PNG IHDR Õ Ü çOä... [19664]	2020-07-16 12:48:24	[NULL]	[NULL]	NAME_COMPAÑÍA_16
18	17	PNG IHDR Õ Ü çOä... [19664]	2020-07-06 08:49:55	ISO3166_ISO 3166标准_17	[NULL]	NAME_名称_17
19	18	[NULL]	2020-07-19 16:21:58	[NULL]	2020-07-02 11:22:09	NAME_NAME_18
20	19	PNG IHDR Õ Ü çOä... [19664]	2020-07-17 04:08:40	ISO3166_CÓDIGO 3166_19	2020-07-04 04:18:40	NAME_COMPAÑÍA_19
21	20	PNG IHDR Õ Ü çOä... [19664]	2020-07-09 16:02:53	[NULL]	2020-07-19 14:04:43	NAME_名称_20
22	21	PNG IHDR Õ Ü çOä... [19664]	2020-06-29 20:47:52	[NULL]	[NULL]	NAME_NAME_21
23	22	[NULL]	2020-07-12 01:36:41	[NULL]	2020-07-03 14:32:35	NAME_COMPAÑÍA_22
24	23	[NULL]	2020-07-19 09:13:19	ISO3166_ISO 3166标准_23	2020-07-12 17:02:15	NAME_名称_23
25	24	PNG IHDR Õ Ü çOä... [19664]	2020-07-02 10:15:05	[NULL]	2020-07-06 00:07:07	NAME_NAME_24

3. Table TIMEZONE



TIMEZONE — Enter a SQL expression to filter results (use Ctrl+Space)

	PK_TIMEZONE_ID	ABBREVIATION	CREATED	MODIFIED	NAME	V_TIME_ZONE
1	0	ABBREVIATION_ABBREVIATION_0	2020-07-06 02:00:35	[NULL]	NAME_NAME_0	V_TIME_ZONE_V_TIME_ZONE_0
2	1	ABBREVIATION_ABRÉVIATION_1	2020-07-19 23:30:25	2020-07-14 16:15:02	NAME_COMPAÑÍA_1	V_TIME_ZONE_FUSO_HORÁRIO_1
3	2	ABBREVIATION_缩略语_2	2020-06-27 02:48:13	2020-07-05 15:25:17	NAME_名称_2	V_TIME_ZONE_时区_2
4	3	ABBREVIATION_ABBREVIATION_3	2020-07-06 20:57:16	2020-06-28 20:43:59	NAME_NAME_3	V_TIME_ZONE_V_TIME_ZONE_3
5	4	ABBREVIATION_ABRÉVIATION_4	2020-07-15 16:05:02	2020-07-02 00:19:52	NAME_COMPAÑÍA_4	[NULL]
6	5	ABBREVIATION_缩略语_5	2020-07-13 22:57:44	2020-06-27 08:14:01	NAME_名称_5	V_TIME_ZONE_时区_5
7	6	ABBREVIATION_ABBREVIATION_6	2020-07-19 22:16:01	2020-07-16 10:19:57	NAME_NAME_6	V_TIME_ZONE_V_TIME_ZONE_6
8	7	ABBREVIATION_ABRÉVIATION_7	2020-07-10 03:25:25	2020-06-27 15:57:09	NAME_COMPAÑÍA_7	V_TIME_ZONE_FUSO_HORÁRIO_7
9	8	ABBREVIATION_缩略语_8	2020-07-03 05:45:30	2020-07-14 01:20:33	NAME_名称_8	V_TIME_ZONE_时区_8
10	9	ABBREVIATION_ABBREVIATION_9	2020-07-13 23:10:17	2020-07-05 07:34:24	NAME_NAME_9	V_TIME_ZONE_V_TIME_ZONE_9
11	10	ABBREVIATION_ABRÉVIATION_10	2020-07-03 00:11:26	2020-07-11 17:19:21	NAME_COMPAÑÍA_10	V_TIME_ZONE_FUSO_HORÁRIO_10

3. Installation

The easiest way is to download a current release of **DBSeeder** from the GitHub repository. You can find the necessary link here.

To download the repository Git is needed and for compilation the Gradle Build Tool and the open-source JDK are needed. For changes to the **DBSeeder** repository it is best to use an editor (e.g. Vim) or an IDE (e.g. Eclipse IDE). For using the Docker Image based databases in operational mode, Docker Desktop must also be installed. For the respective software versions, please consult the document release notes.

4. Operating Instructions

4.1 Scripts

4.1.1 Script `run_db_seeder`

Using the **DBSeeder** development and operational Docker image from Docker Hub (see here) eliminates the need to install the runtime environment.

With the script `run_db_seeder` the complete functionality of the **DBSeeder** application can be used:

- Creating a suitable database
- Generation of any number of dummy data.

All scripts are available in a Windows version (`cmd` / `.bat`) as well as in a Unix version (`bash` / `.sh`). To run the scripts, apart from the prerequisites as release notes (`ReleaseNotes.md`), only the libraries in the `lib` directory and the corresponding script of `run_db_seeder` are required. The creation of the databases also requires a working access to Docker Hub.

All control parameters used in **DBSeeder** (see section 4.3) can be adapted in the scripts to specific needs.

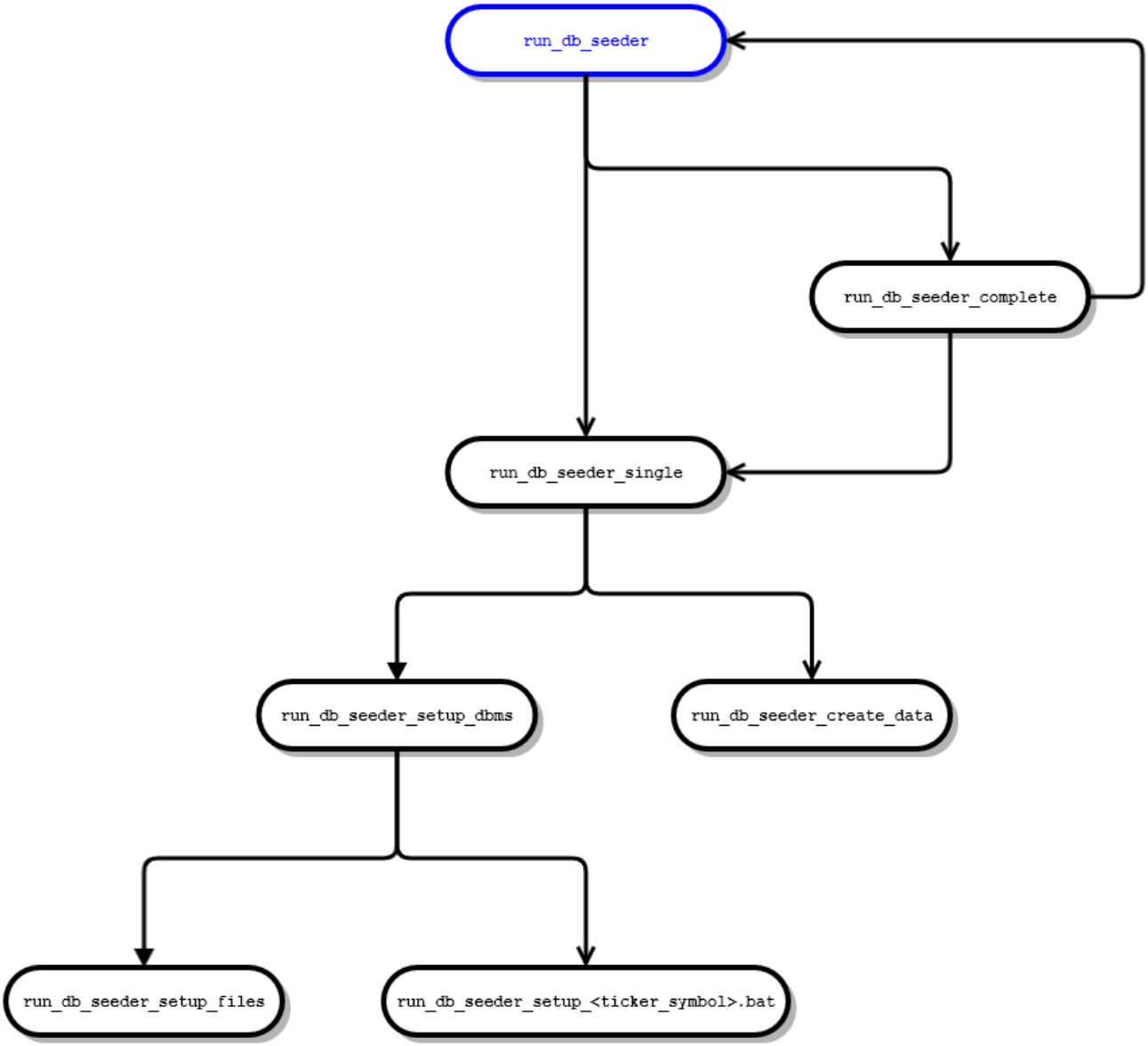
The `run_db_seeder` script is controlled by the following script parameters::

- `DB_SEEDER_DBMS`: the ticker symbol of the desired database management system (default value `sqlite`) or `complete` for all implemented RDBMS.
- `DB_SEEDER_SETUP_DBMS`: should an empty database be created:
  - `yes`: a new database is created based on a suitable Docker image
  - otherwise: no database is created
- `DB_SEEDER_NO_CREATE_RUNS`: Number of dummy data generation runs:
  - 1: one run
  - 2: two runs
  - otherwise: no run

For the run variants `complete`, `complete_client`, `complete_emb` and `complete_trino`, statistics files with the following data name structure are created in the file directory `resources/statistics` by default:

```
db_seeder_<bash | cmd>_<run variant>_unknown_<DBSeeder release>.tsv
```

An overview of the structure of the scripts used can be taken from the following diagram:



4.1.2 Script `scripts/run_db_seeder_statistics`

This script aggregates the existing statistics files into a single overall file. The file name of this overall file is defined with parameter `db_seeder.file.statistics.summary.name` and the existing statistics files are searched in the file directories according to parameter `db_seeder.file.statistics.summary.source`. The file format `csv` or `tsv` depends on the parameter `db_seeder.file.statistics.delimiter`.

Example content:

```
ticker symbol   RDBMS version   creator db type schema  runtime in ms   start time  end time    host name   no.
cores   operating system    file_name
agens   AgensGraph  v2.6.0  bash    client  unknown 14  2020-10-05 16:09:36.618076382   2020-10-05
16:09:51.570013623  ubuntu  2   amd64 / Linux / 5.4.0-48-generic    db_seeder_bash_client_unknown_2.6.0
cratedb CrateDB v2.6.0  bash    client  unknown 24  2020-10-05 16:11:40.160409347   2020-10-05
16:12:04.695790414  ubuntu  2   amd64 / Linux / 5.4.0-48-generic    db_seeder_bash_client_unknown_2.6.0
cubrid  CUBRID  v2.6.0  bash    client  unknown 50  2020-10-05 16:13:22.287362093   2020-10-05
16:14:12.339067275  ubuntu  2   amd64 / Linux / 5.4.0-48-generic    db_seeder_bash_client_unknown_2.6.0
```

4.2 Operation Possibilities

DBSeeder is tested under Ubuntu and Microsoft Windows. In addition, tests are always performed in Windows with Ubuntu under the Windows Subsystem for Linux (WSL). Besides one of the two operating systems, these are the minimum requirements for running **DBSeeder**:

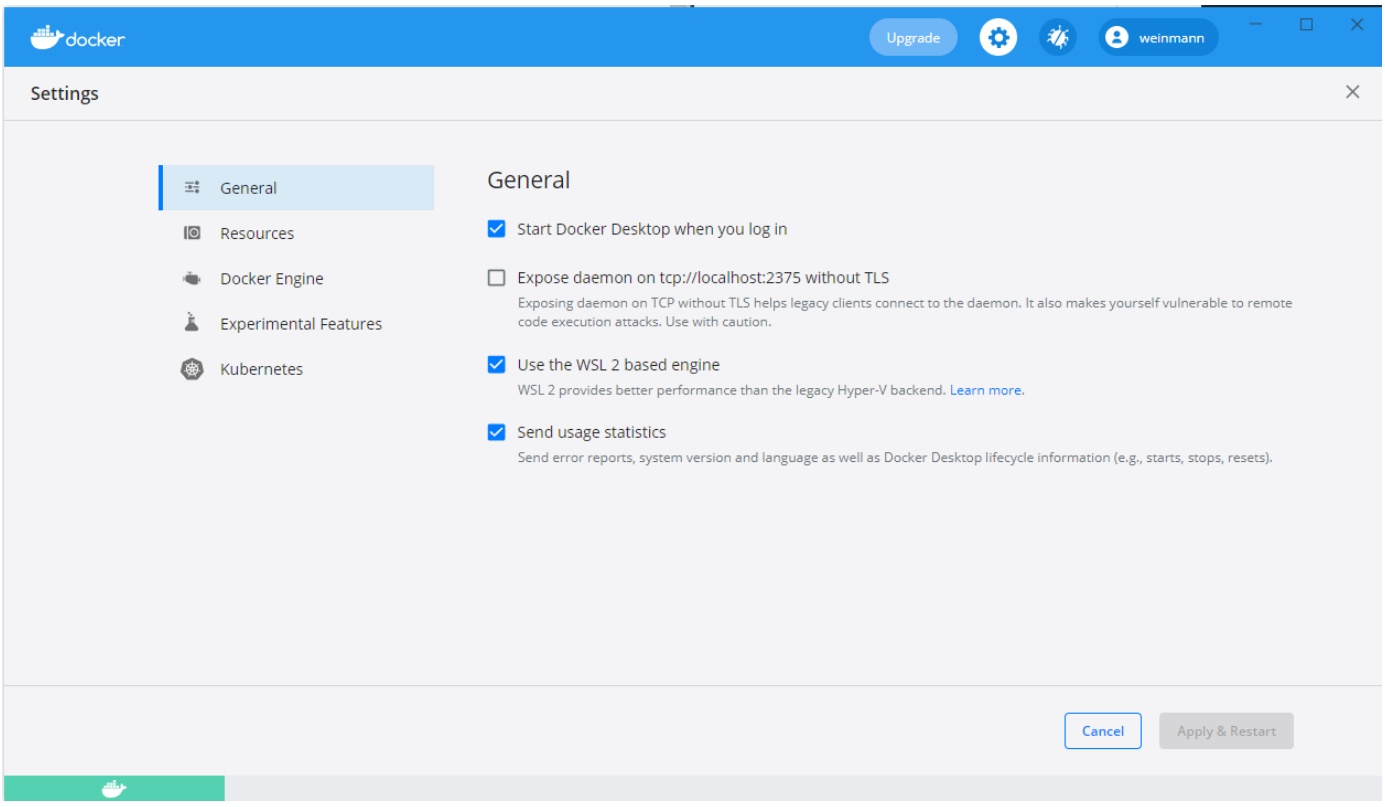
- Docker Desktop Community

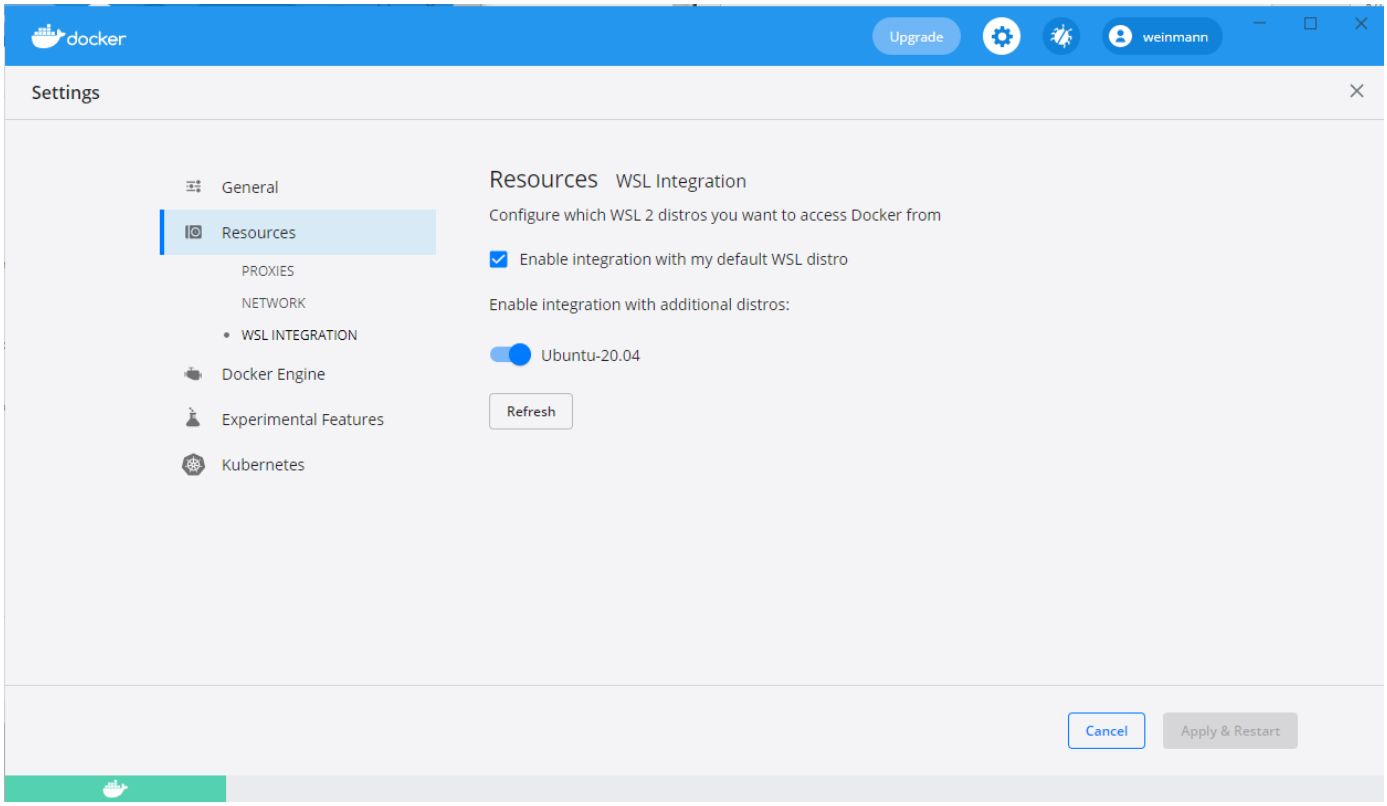
- Eclipse IDE
- Gradle Build Tool
- Java Development Kit

Details on the required software versions can be found in the release notes.

Special Features for the Operation with Ubuntu

- A suitable image is available on Docker Hub for development and operation, see here.
- In the directory `scripts` are the two scripts `run_install_4_vm_wsl2_1.sh` and `run_install_4_vm_wsl2_1.sh` with which an Ubuntu environment can be prepared for development and operation.
    - Ubuntu 20.04 installed directly or via VMware
    - run `sudo apt update`
    - run `sudo apt install dos2unix git`
    - run `git clone https://github.com/KonnexionsGmbH/db_seeder` (cloning the **`DBSeeder`** repository)
    - run `cd db_seeder`
    - run `./scripts/run_install_4_vm_wsl2_1.sh`
    - close the Ubuntu shell and reopen it again
    - run `cd db_seeder`
    - run `./scripts/run_install_4_vm_wsl2_2.sh`
    - run `gradle copyJarToLib`
    - run `./run_db_seeder.sh`
- If the Windows Subsystem for Linux (WSL) is to be used, then the `WSL INTEGRATION` for Ubuntu must be activated in Docker





4.3 Control Parameters

4.3.1 Supported Parameters

The flow control parameters for **`DBSeeder`** are stored in the properties file `src/main/resources/db_seeder.properties` and can all be overridden by the environment variables defined in the scripts. The following control parameters are currently supported:

```
db_seeder_character_set_server=
db_seeder_collation_server=
db_seeder.connection.host=
db_seeder.connection.host.trino=
db_seeder.connection.port=0
db_seeder.connection.port.trino=0
db_seeder.connection.prefix=
db_seeder.connection.service=
db_seeder.connection.suffix=

db_seeder.database.sys=
db_seeder.database=

db_seeder.file.configuration.name=
db_seeder.file.json.name=resources/json/db_seeder_schema.company.json
db_seeder.file.statistics.delimiter=\t
db_seeder.file.statistics.header=ticker symbol;RDBMS;db type;runtime in ms;start time;end time;host name;no.
cores;operating system
db_seeder.file.statistics.name=resources/statistics/db_seeder_local.tsv
db_seeder.file.statistics.summary.name=resources/statistics/db_seeder_summary.tsv
db_seeder.file.statistics.summary.source=resources/statistics;Transfer

db_seeder.password.sys=
db_seeder.password=

db_seeder.schema=

db_seeder.user.sys=
db_seeder.user=
```

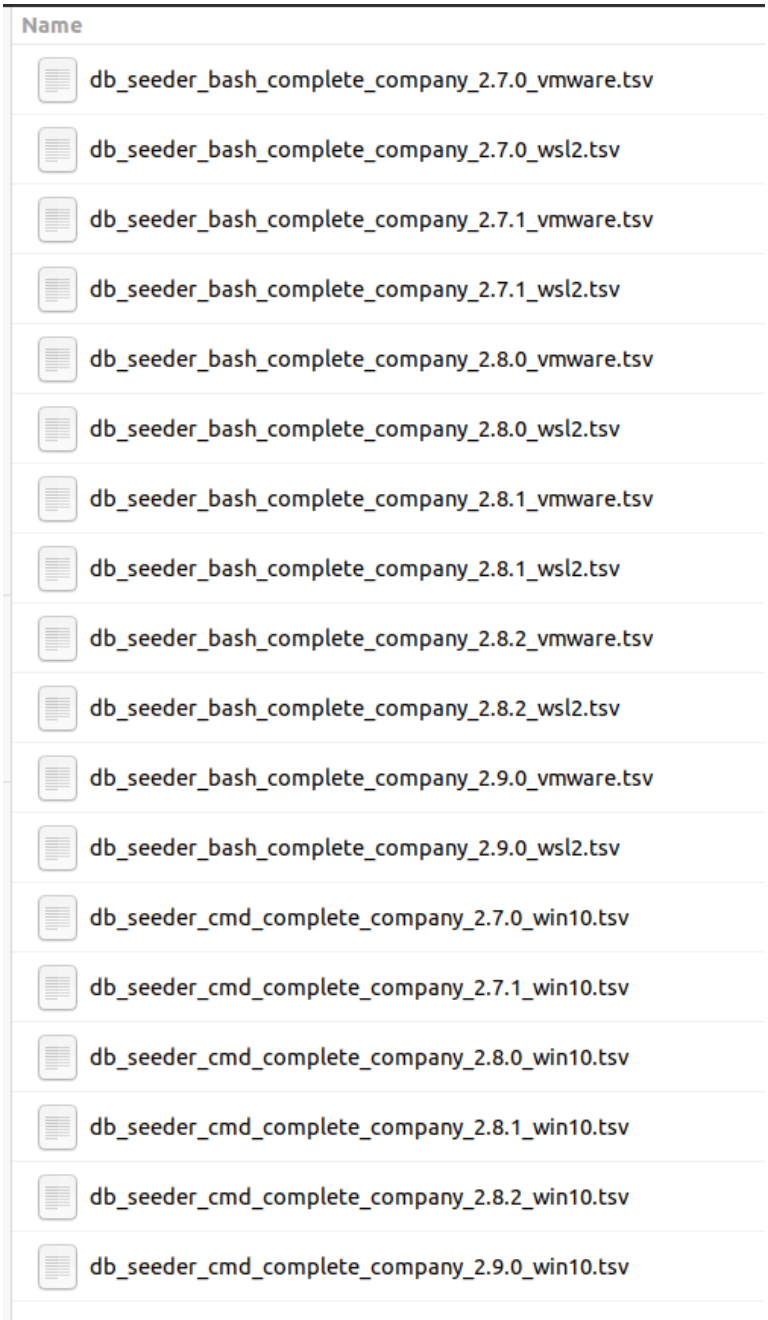
4.3.2 Explanation and Cross-reference

Property incl. Default Value [db.seeder.]	Environment Variable [DB_SEEDER_]	Used By	Description
character.set.server=<x...x>	CHARACTER_SET_SERVER	mariadb	default server character set
collation.server=<x...x>	COLLATION_SERVER	mariadb	default server collation
connection.host=<x...x>	CONNECTION_HOST	all client RDBMS	host name or ip address of the database server
connection.host_trino=<x...x>	CONNECTION_HOST_TRINO	trino	host name or ip address of the trino
connection.port=<9...9>	CONNECTION_PORT	all client RDBMS	port number of the database server
connection.port_trino=<9...9>	CONNECTION_PORT_TRINO	trino	port number of the trino
connection.prefix=<x...x>	CONNECTION_PREFIX	all RDBMS	prefix of the database connection string
connection.service=<x...x>	CONNECTION_SERVICE	oracle	service name of the database connection string
connection.suffix=<x...x>	CONNECTION_SUFFIX	firebird, hsqldb, mysql, percona, voltdb	suffix of the database connection string
database.sys=<x...x>	DATABASE_SYS	agens, cockroach, informix, mariadb, mimer, monetdb, mysql, omnisci, percona,	privileged database name
		postgresql, sqlserver, yugabyte	
database=<x...x>	DATABASE	all RDBMS except cratedb, exasol, monetdb, oracle, voltdb	database name
file.configuration.name=<x...x>	FILE_CONFIGURATION_NAME	n/a	directory and file name of the `DBSeeder` configuration file
file.json.name=<x...x>	FILE_JSON_NAME	scripts/run_db_seeder_generate_schema	directory and file name of the JSON file containing the database schema
file.statistics.delimiter=<x...x>	FILE_STATISTICS_DELIMITER	all RDBMS	separator of the statistics file created in `run_db_seeder`
file.statistics.header=<x...x>	FILE_STATISTICS_HEADER	all RDBMS	header line of the statistics file created in `run_db_seeder`
file.statistics.name=<x...x>	FILE_STATISTICS_NAME	all RDBMS	file name of the statistics file created in `run_db_seeder`
file.statistics.summary.name=<x...x>	FILE_STATISTICS_SUMMARY_NAME	all RDBMS	file name of the summary statistics file created in `run_db_seeder_statistics`
file.statistics.summary.source=<x...x>	FILE_STATISTICS_SUMMARY_SOURCE	all RDBMS	directory name(s) (separated by semicolon) of the source directories containing statistics files
password.sys=<x...x>	PASSWORD_SYS	agens, exasol, firebird, ibmdb2, informix, mariadb, mimer, monetdb, mysql, omnisci,	password of the privileged user
		oracle, percona, postgresql, sqlserver	password of the privileged user
password=<x...x>	PASSWORD	all RDBMS except cockroach, derby, ibmdb2, informix	password of the normal user

Property incl. Default Value [db.seeder.]	Environment Variable [DB_SEEDER_]	Used By	Description
schema=kxn_schema	SCHEMA	agens, derby, exasol, h2, hsqldb, ibmdb2, monetdb, postgresql, sqlserver, yugabyte	schema name
user.sys=<x...x>	USER_SYS	all RDBMS except derby, voltdb	name of the privileged user
user=kxn_user	USER	all RDBMS except derby, ibmdb2, informix	name of the normal user

4.4 Statistics

Performance data for the different versions of **DBSeeder** can be found in the file directory `resources/statistics`:



The different file name patterns result from the following operating system environments:

- `..._vmware.tsv`: Ubuntu with VMware Workstation Player on Windows
- `...._win10.tsv`: Windows
- `....._wsl2.tsv`: Ubuntu LTS with Windows Subsystem for Linux on Windows

5. RDBMS Specific Technical Details

DBeaver is a great tool to analyze the database content. Below are also DBeaver based connection parameter examples for each database management system.

AgensGraph / **Apache Derby** / **CockroachDB** / **CrateDB** / **CUBRID** / **Exasol** /
Firebird /
H2 Database Engine /
HSQLDB /
IBM Db2 Database / **IBM Informix** / **MariaDB Server** / **Mimer SQL** / **MonetDB** / **MySQL Database** / **OmniSciDB** / **Oracle Database** / **Percona Server for MySQL** / **PostgreSQL** / **SQL Server** / **SQLite** / **trino** / **VoltDB** / **YugabyteDB**

5.1 AgensGraph

- **data types**:

`DBSeeder` Type	AgensGraph Database Type
BIGINT	BIGINT
BLOB	BYTEA
CLOB	TEXT
TIMESTAMP	TIMESTAMP
VARCHAR	VARCHAR

- **DDL syntax**:
  - CREATE DATABASE: see PostgreSQL
  - CREATE SCHEMA: see PostgreSQL
  - CREATE TABLE: see PostgreSQL
  - CREATE USER: see PostgreSQL
- **Docker image (latest)**:
  - pull command: `docker pull bitnine/agensgraph:v2.1.3`
  - DockerHub
- **encoding**: see PostgreSQL
- **issue tracking**: GitHub
- **JDBC driver (latest)**:
  - version 1.4.2-c1
  - Maven repository
- **source code**: GitHub

5.2 Apache Derby

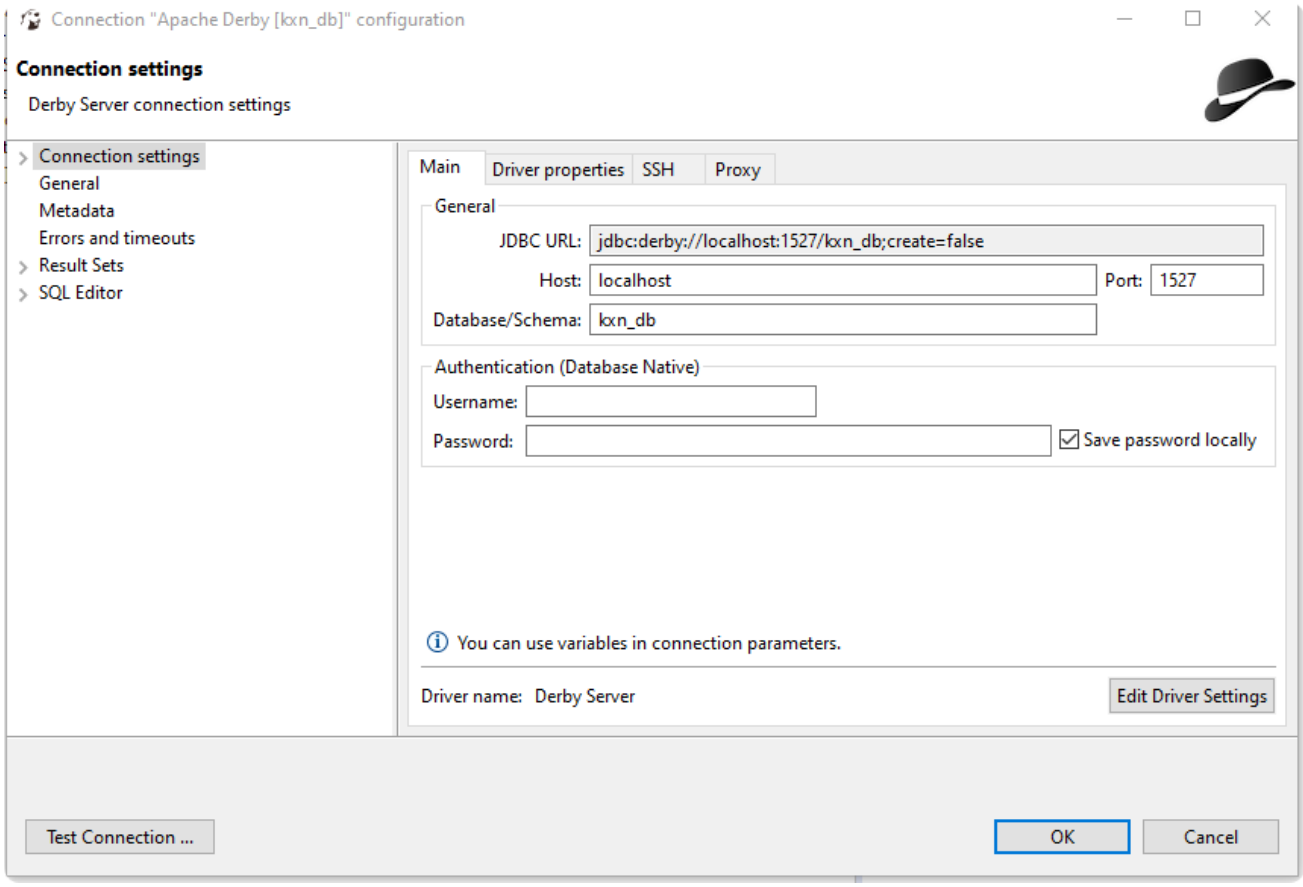
- **data types**:

`DBSeeder` Type	Apache Derby Type
BIGINT	BIGINT
BLOB	BLOB
CLOB	CLOB
TIMESTAMP	TIMESTAMP
VARCHAR	VARCHAR

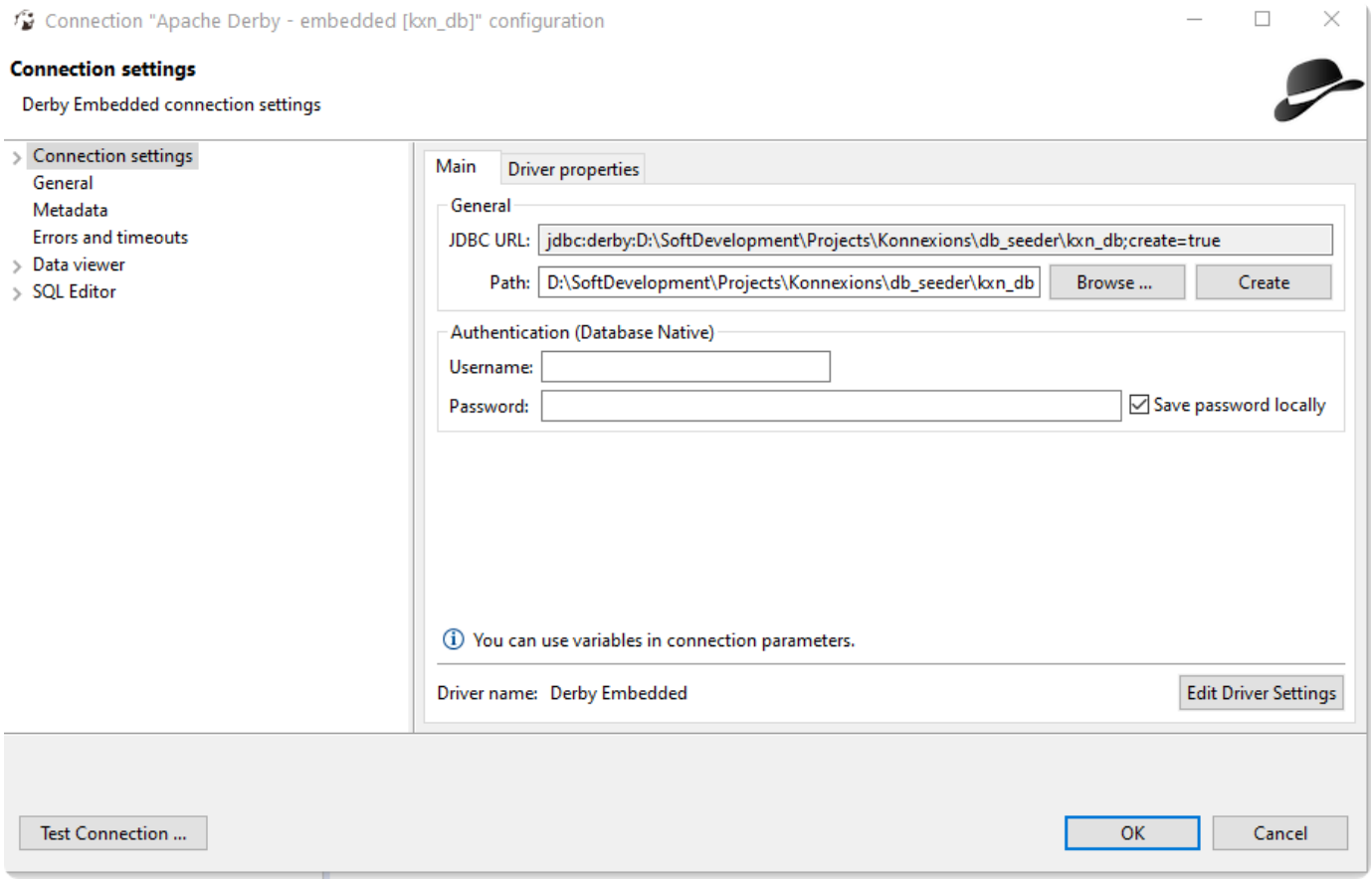
- **DDL syntax**:
  - CREATE DATABASE - n/a
  - CREATE SCHEMA
  - CREATE TABLE
  - CREATE USER - n/a

- **Docker image (latest - only client version``)**:
  - pull command: `docker pull konnexionsgmbh/apache_derby:10.15.2.0`
  - DockerHub
- **encoding**: by using the following JVM parameter: `-Dderby.ui.codeset=UTF8`
- **issue tracking**: Jira
- **JDBC driver (latest)**:
  - version 10.15.2.0
  - client version: Maven repository
  - embedded version: Maven repository
- **source code**: Apache Derby
- **DBeaver database connection settings**:

  -- client version:



-- embedded version:



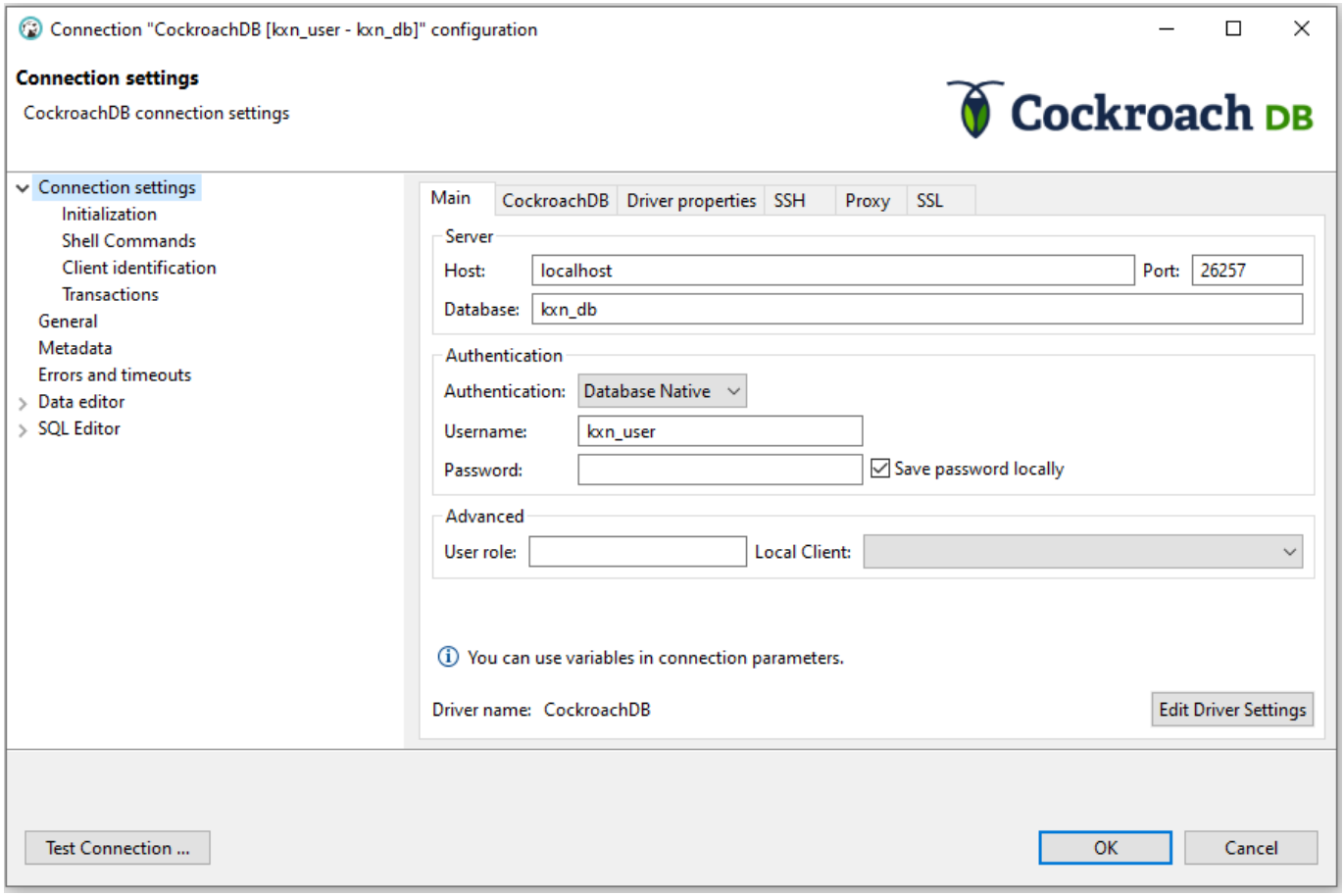
5.3 CockroachDB

- **data types**:

`DBSeeder` Type	CockroachDB Type
BIGINT	INT
BLOB	BYTES
CLOB	STRING
TIMESTAMP	TIMESTAMP
VARCHAR	STRING

- **DDL syntax**:
  - CREATE DATABASE
  - CREATE SCHEMA
  - CREATE TABLE
  - CREATE USER
- **Docker image (latest)**:
  - pull command: `docker pull cockroachdb/cockroach:v21.1.2`
  - DockerHub
- **encoding**: by default `utf8` encoding
- **issue tracking**: GitHub
- **JDBC driver (latest)**:
  - same as PostgreSQL
- **privileged database access**: user `root`
- **source code**: GitHub

- **DBeaver database connection settings**:



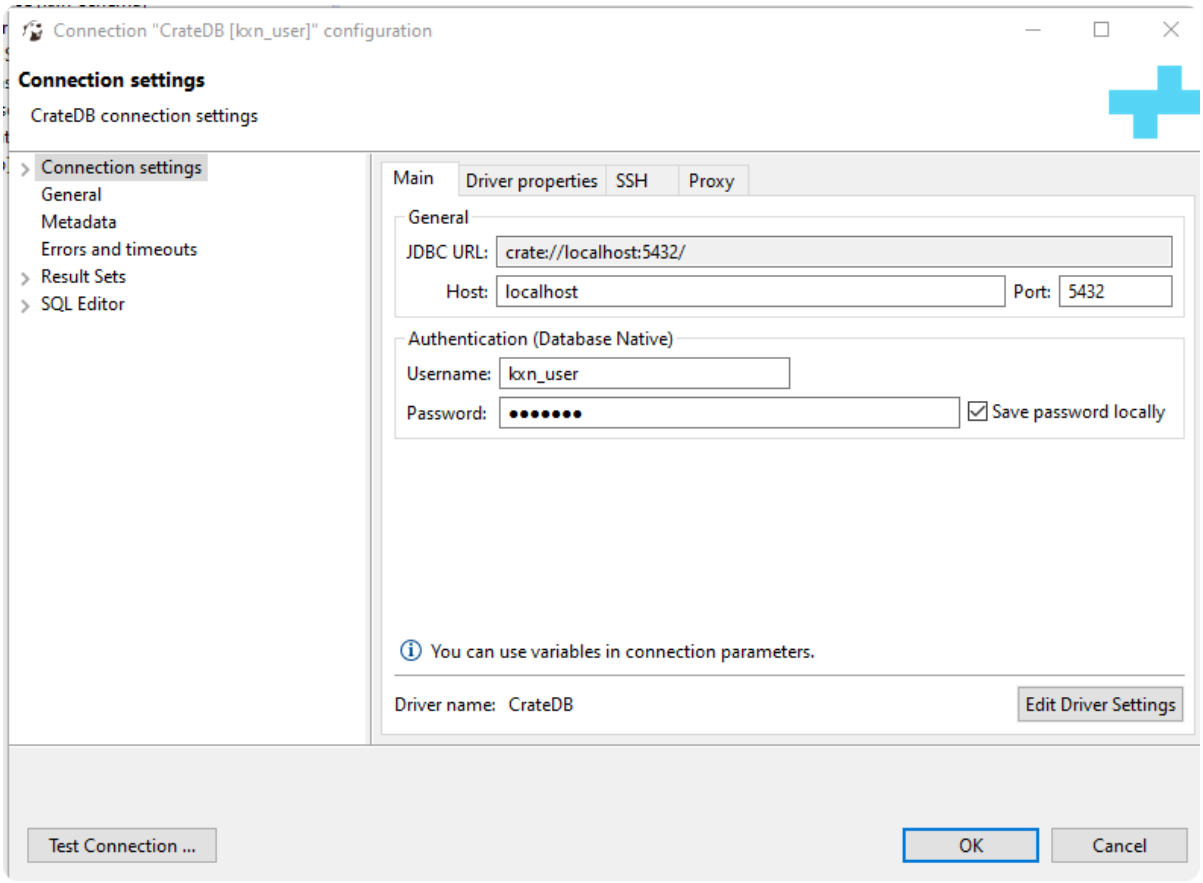
5.3 CrateDB

- **data types**:

`DBSeeder` Type	CrateDB Type
BIGINT	BIGINT
BLOB	OBJECT
CLOB	TEXT
TIMESTAMP	TIMESTAMP
VARCHAR	TEXT

- **DDL syntax**:
  - CREATE DATABASE - n/a
  - CREATE SCHEMA - n/a
  - CREATE TABLE
  - CREATE USER
- **Docker image (latest)**:
  - pull command: `docker pull crate:4.5.1`
  - DockerHub
- **encoding**: by default `utf8` encoding
- **issue tracking**: GitHub
- **JDBC driver (latest)**:
  - version 2.6.0
  - JFrog Bintray repository

- **privileged database access**: user `crate`
- **restrictions**:
  - no constraints (e.g. foreign keys or unique keys)
  - no transaction concept
  - no triggers
  - only a very proprietary BLOB implementation
- **source code**: GitHub
- **DBeaver database connection settings**:



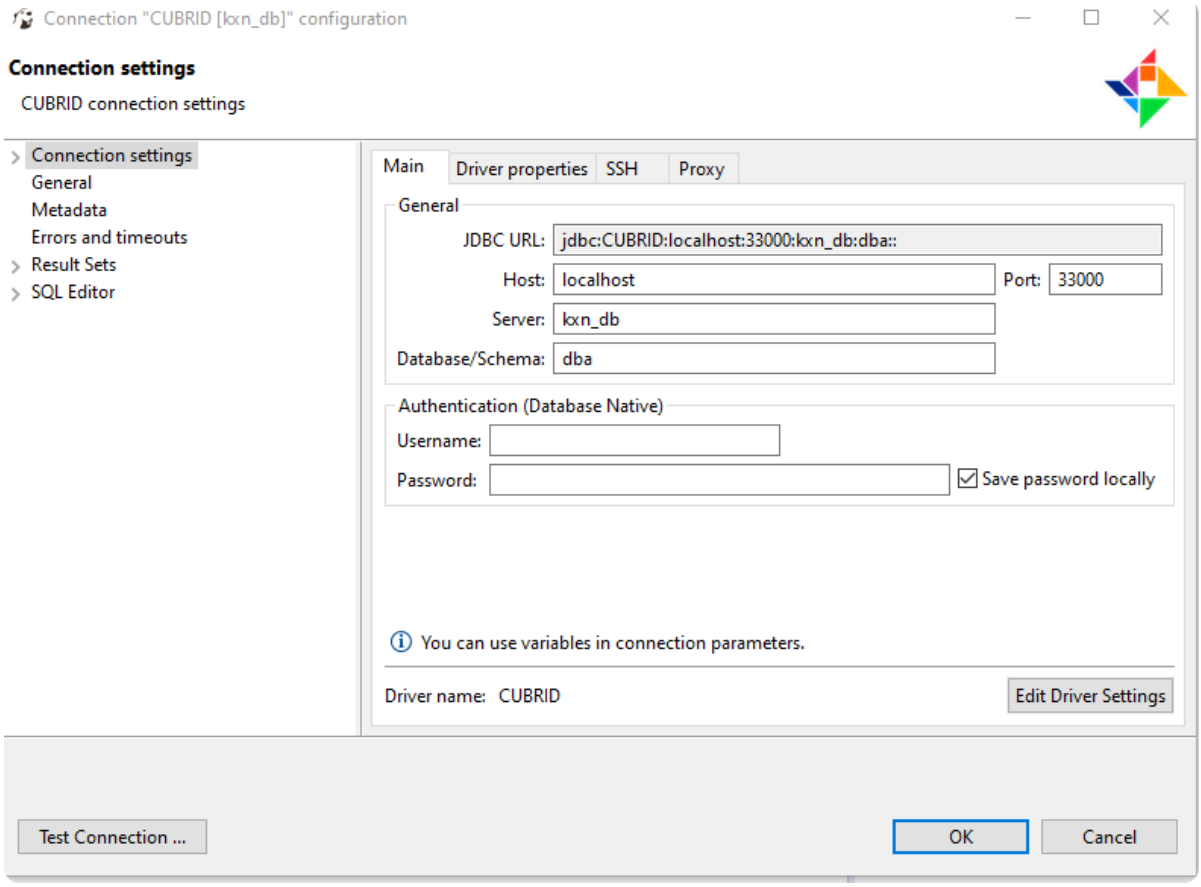
5.4 CUBRID

- **data types**:

`DBSeeder` Type	CUBRID Type
BIGINT	INT
BLOB	BLOB
CLOB	CLOB
TIMESTAMP	TIMESTAMP
VARCHAR	VARCHAR

- **DDL syntax**:
  - CREATE DATABASE - n/a
  - CREATE SCHEMA - n/a
  - CREATE TABLE
  - CREATE USER
- **Docker image (latest)**:
  - pull command: `docker pull cubrid/cubrid:11.0`
  - DockerHub

- **encoding**: by specifying after the database name when database is created: `kxn_db de_DE.utf8`
- **issue tracking**:
  - Jira
- **JDBC driver (latest)**:
  - version 11.0.1.0286
  - Maven repository
- **privileged database access**: users `DBA` and `PUBLIC`
- **restrictions**: no full UTF-8 support
- **source code**: GitHub
- **DBeaver database connection settings**:



5.5 Exasol

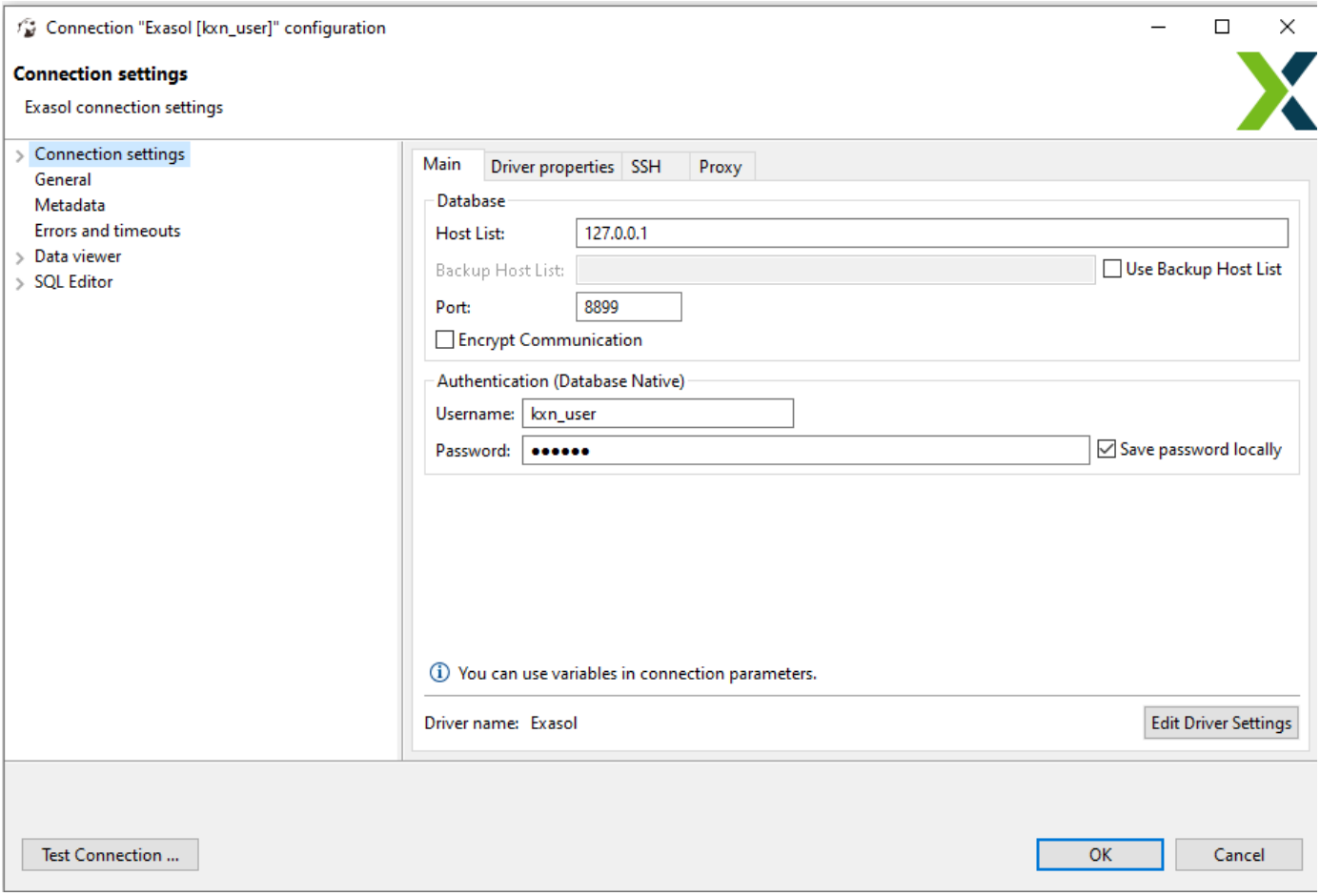
- **data types**:

`DBSeeder` Type	Exasol Type
BIGINT	BIGINT
BLOB	VARCHAR(2000000)
CLOB	VARCHAR(2000000)
TIMESTAMP	TIMESTAMP
VARCHAR	VARCHAR

- **DDL syntax**:
  - CREATE DATABASE - n/a
  - CREATE SCHEMA
  - CREATE TABLE

    - CREATE USER

- **Docker image (latest)**:
    - pull command: `docker pull exasol/docker-db:7.0.10`
    - DockerHub
- **JDBC driver (latest)**:
    - version 7.0.7
    - Maven repository
- **privileged database access**: user `sys` password `exasol`
- **DBeaver database connection settings**:



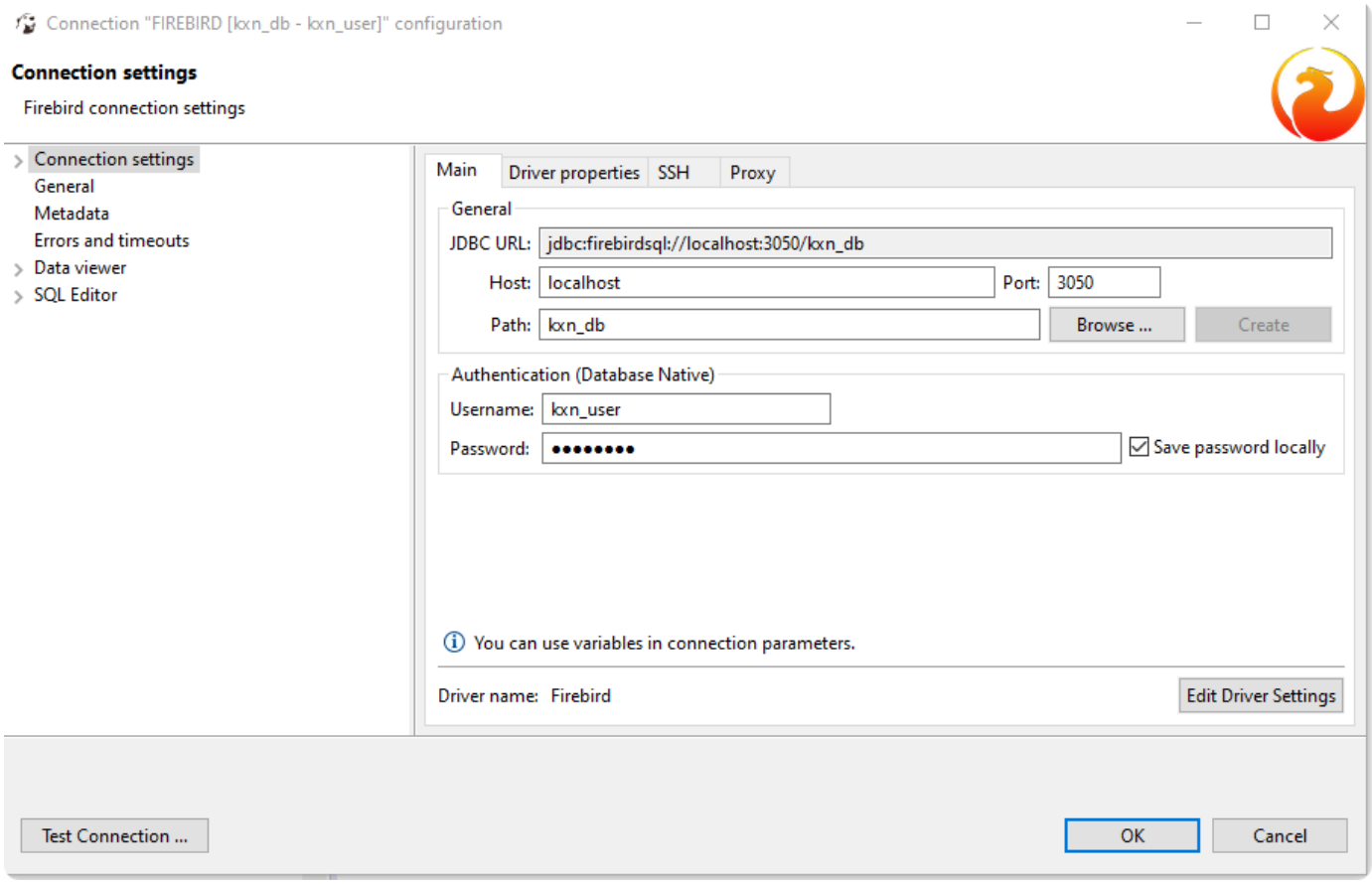
5.6 Firebird

- **data types**:

DBSeeder Type	Firebird Type
BIGINT	INTEGER
BLOB	BLOB
CLOB	BLOB SUB_TYPE 1
TIMESTAMP	TIMESTAMP
VARCHAR	VARCHAR

- **DDL syntax**:
    - CREATE DATABASE
    - CREATE SCHEMA - n/a
    - CREATE TABLE
    - CREATE USER

- **Docker image (latest)**:
  - pull command: `docker pull jacobalberty/firebird:v4.0.0rc1`
  - DockerHub
- **encoding**: by using the following JDBC URL parameter: `encoding=UTF8`
- **issue tracking**: GitHub
- **JDBC driver (latest)**:
  - version 4.0.3.java11
  - Maven repository
- **privileged database access**: user SYSDBA
- **source code**: GitHub
- **DBeaver database connection settings**:



5.7 H2 Database Engine

- **data types**:

DBSeeder Type	H2 Database Engine Type
BIGINT	BIGINT
BLOB	BLOB
CLOB	CLOB
TIMESTAMP	TIMESTAMP
VARCHAR	VARCHAR

- **DDL syntax**:
  - CREATE DATABASE - n/a
  - CREATE SCHEMA

    - CREATE TABLE
    - CREATE USER

- **Docker image (latest)**:

    - pull command: `docker pull konnexionsgmbh/h2_database_engine:1.4.200`
    - DockerHub

- **encoding**: H2 internally uses Unicode, and supports all character encoding systems and character sets supported by the virtual machine you use.

- **issue tracking**: GitHub

- **JDBC driver (latest)**:

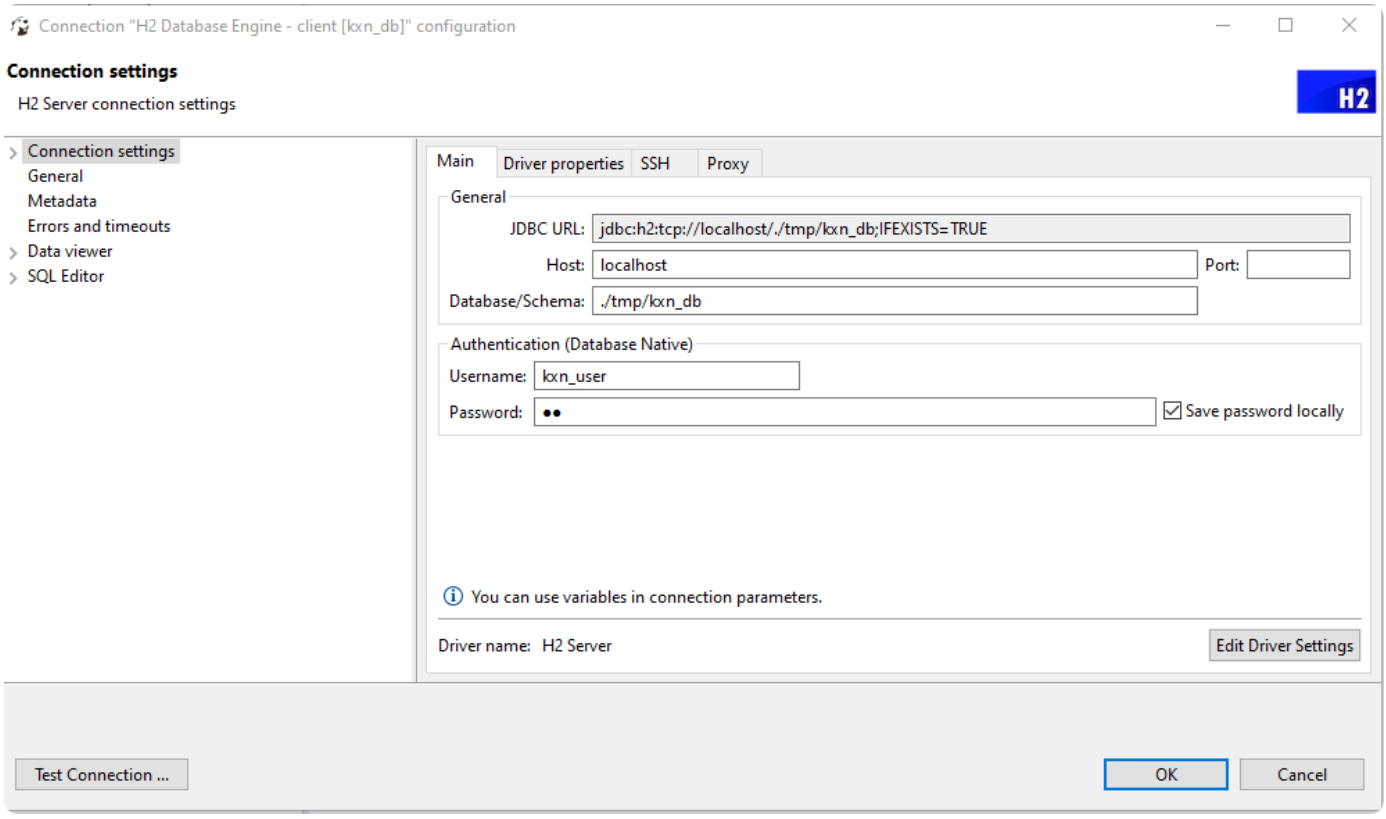
    - version 1.4.200
    - Maven repository

- **privileged database access**: user `sa`

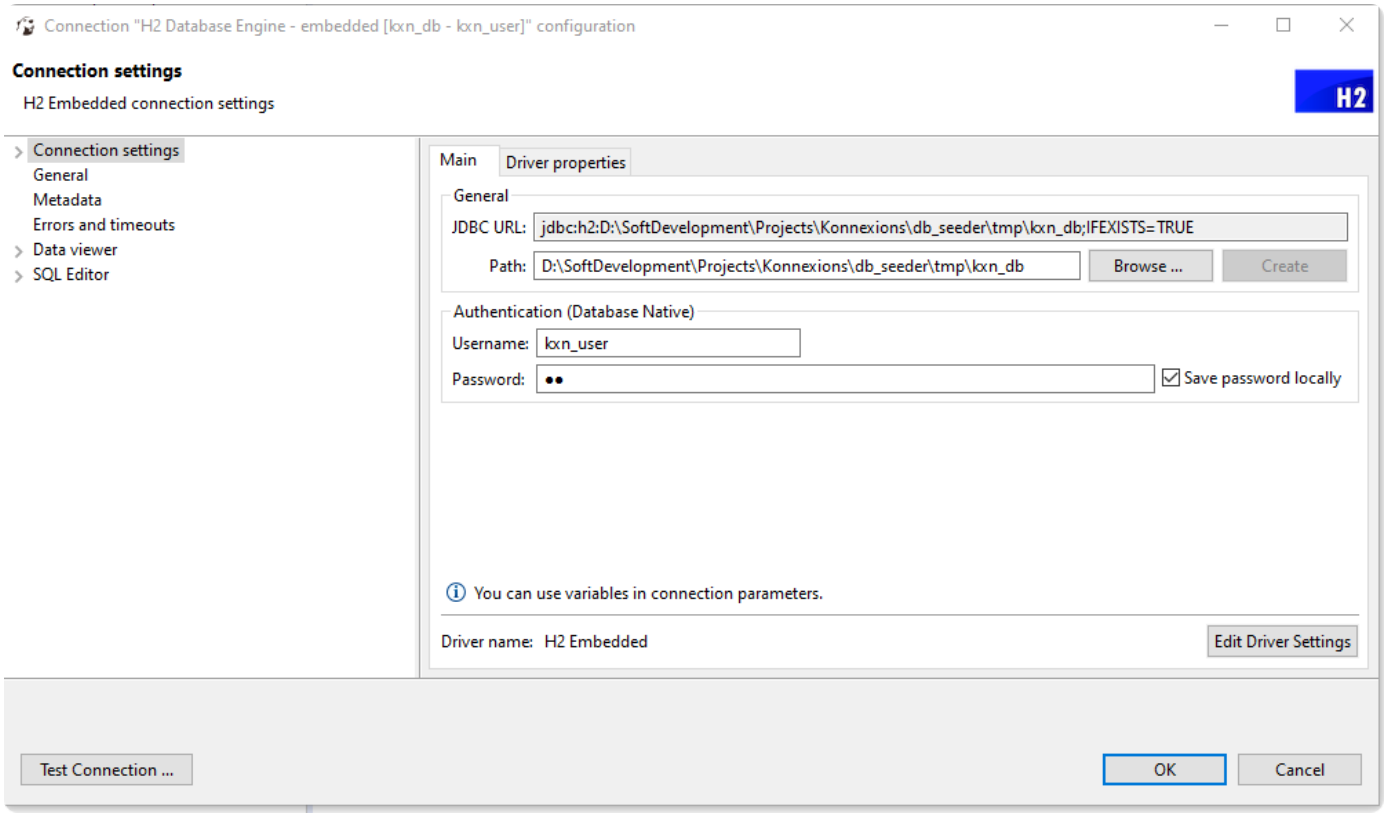
- **source code**: GitHub

- **DBeaver database connection settings**:

  -- client version:



-- embedded version:



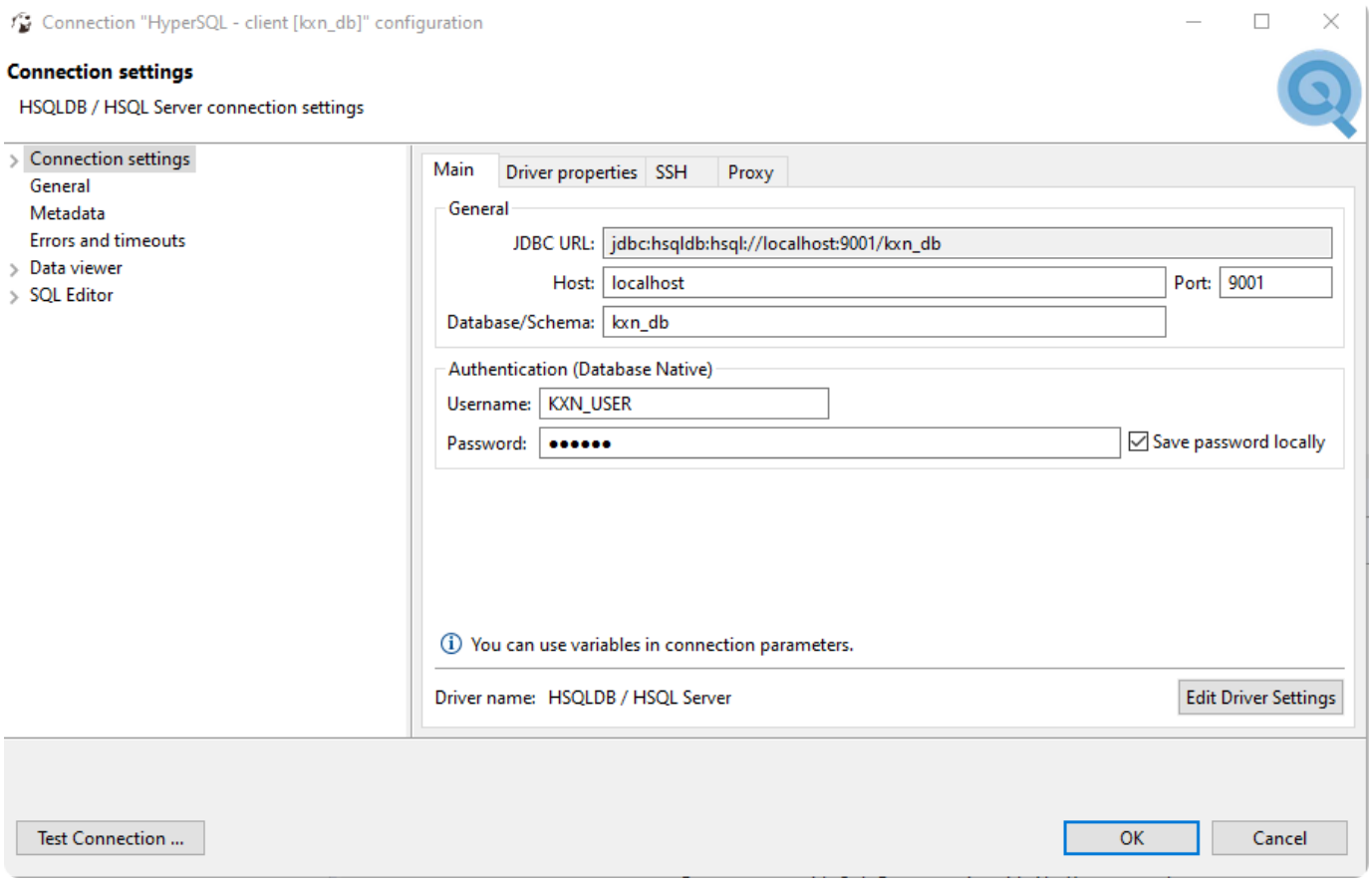
5.8 HSQLDB

- **data types**:

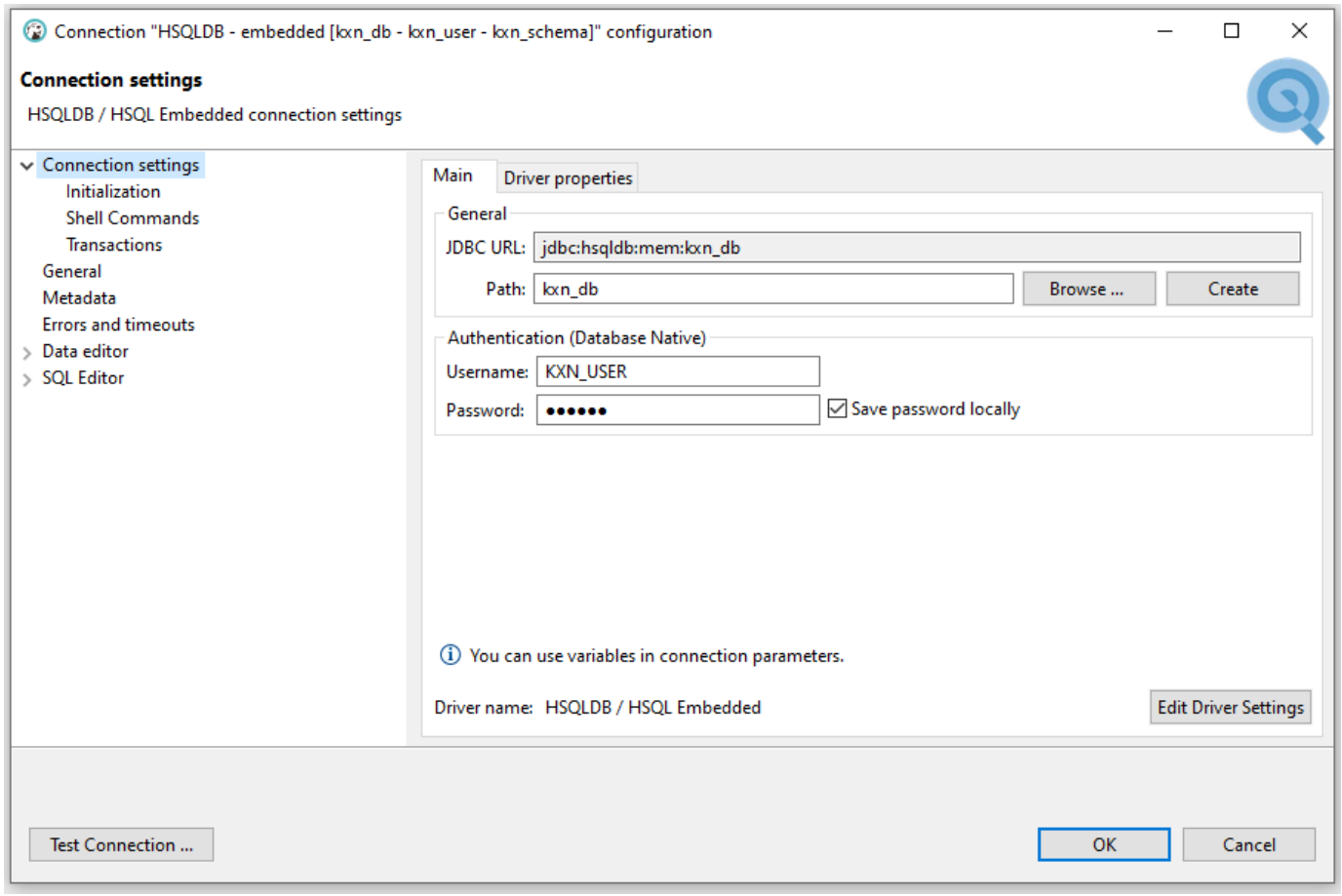
DBSeeder Type	HSQLDB Type
BIGINT	BIGINT
BLOB	BLOB
CLOB	CLOB
TIMESTAMP	TIMESTAMP
VARCHAR	VARCHAR

- **DDL syntax**:
  - CREATE DATABASE - n/a
  - CREATE SCHEMA
  - CREATE TABLE
  - CREATE USER
- **Docker image (latest)**:
  - pull command: `docker pull konnexionsgmbh/hypersql_database:2.6.0`
  - DockerHub
- **encoding**: by using the following system property `sqlfile.charset=UTF-8`.
- **issue tracking**: SourceForge
- **JDBC driver (latest)**:
  - version 2.6.0
  - Maven repository
- **privileged database access**: user SA
- **source code**: SourceForge
- **DBeaver database connection settings**:

-- client version:



-- embedded version:

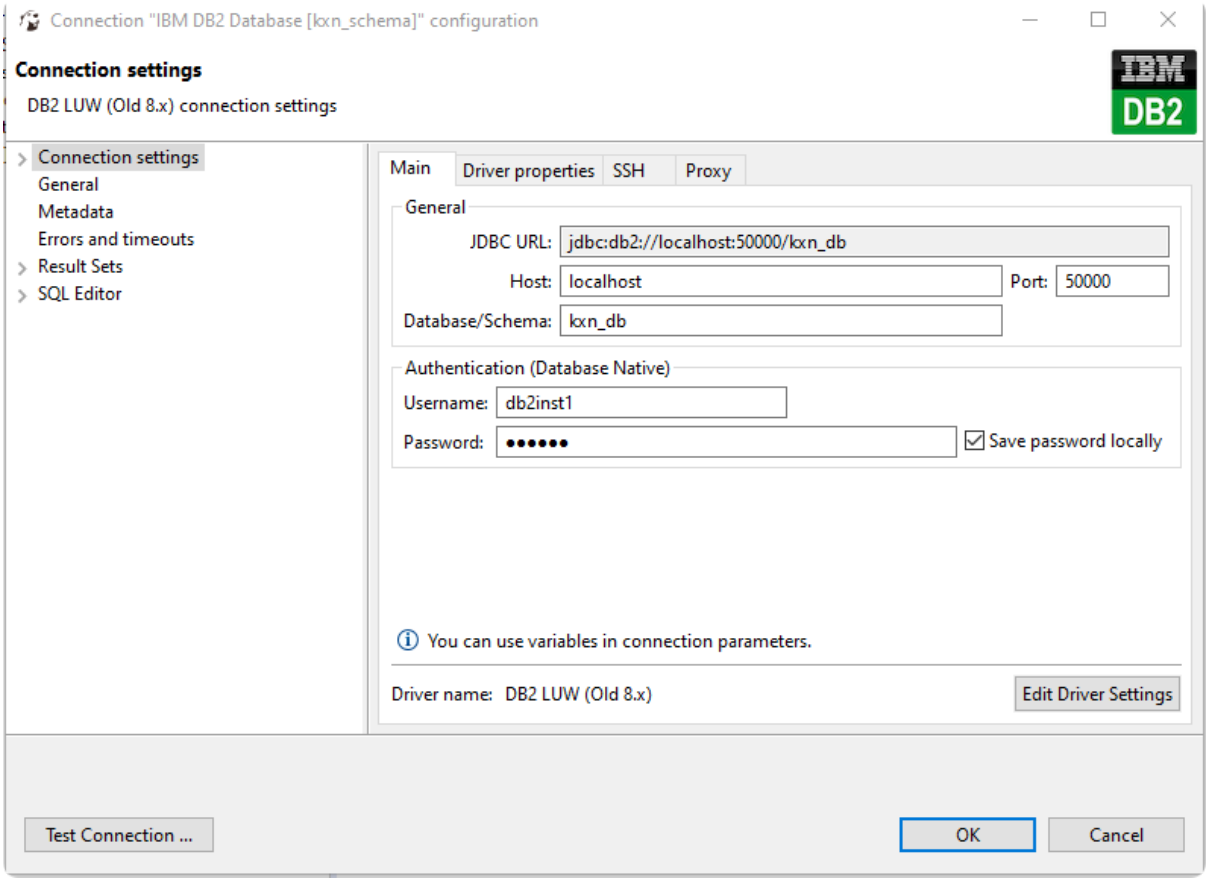


5.9 IBM Db2 Database

- **data types**:

`DBSeeder` Type	IBM Db2 Database Type
BIGINT	BIGINT
BLOB	BLOB
CLOB	CLOB
TIMESTAMP	TIMESTAMP
VARCHAR	VARCHAR

- **DDL syntax**:
  - CREATE DATABASE
  - CREATE SCHEMA
  - CREATE TABLE
  - CREATE USER
- **Docker image (latest)**:
  - pull command: `docker pull ibmcom/db2:11.5.5.1`
  - DockerHub
- **encoding**:
  - by using the CCSID clause in the CREATE statements for any of the following objects:
    - Database
    - Table space
    - Table
    - procedure or function
- **JDBC driver (latest)**:
  - version 11.5.54.0
  - Maven repository
- **privileged database access**: user `db2inst1`
- **restrictions**: the IBM Db2 DBMS only accepts operating system accounts as database users
- **DBeaver database connection settings**:



5.10 IBM Informix

- **data types**:

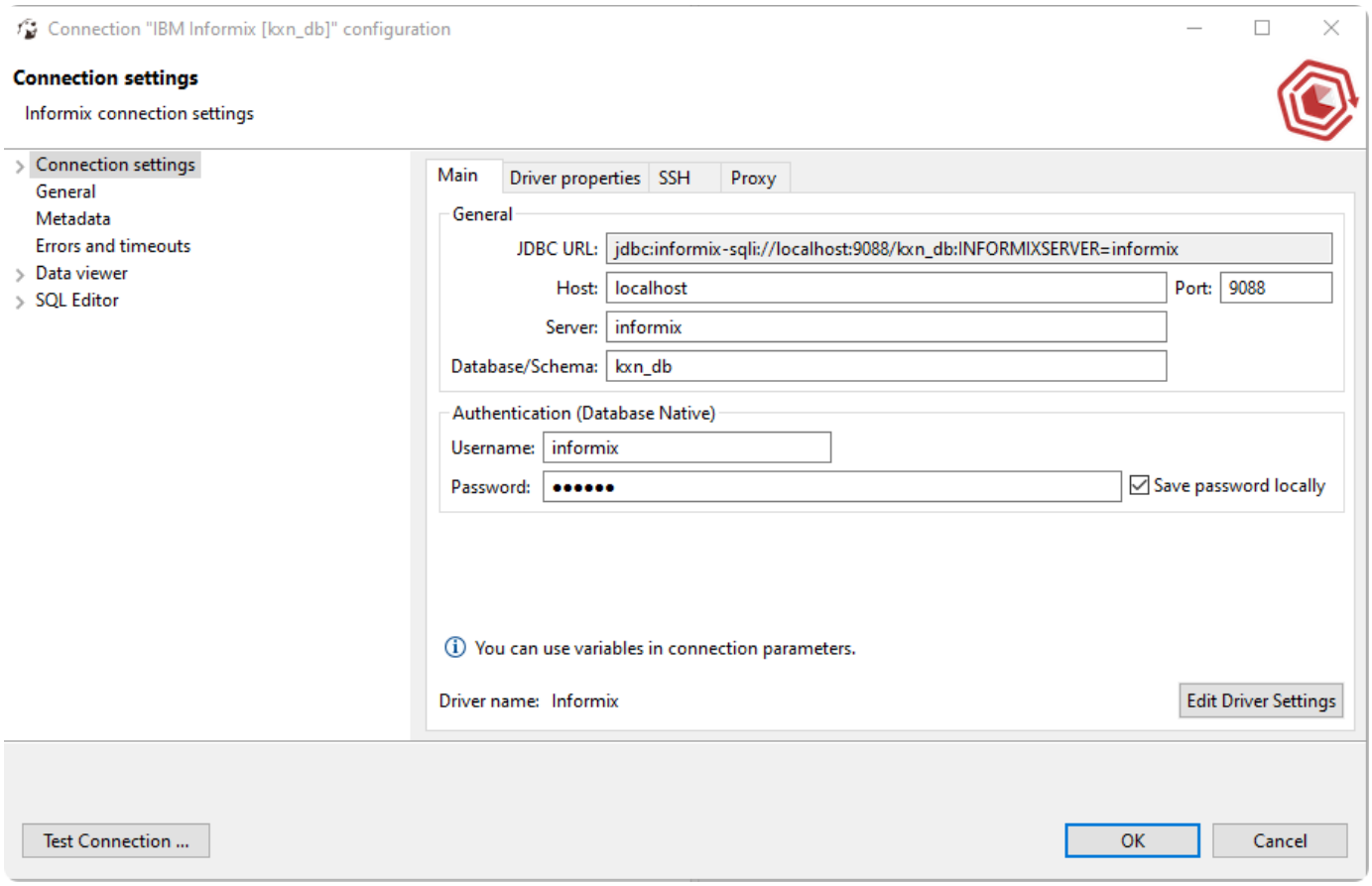
DBSeeder Type	IBM Informix Database Type
BIGINT	BIGINT
BLOB	BLOB
CLOB	CLOB
TIMESTAMP	DATETIME YEAR TO FRACTION
VARCHAR	VARCHAR (1-254) / LVARCHAR

- **DDL syntax**:
  - CREATE DATABASE
  - CREATE SCHEMA - n/a
  - CREATE TABLE
  - CREATE USER
- **Docker image (latest)**:
  - pull command: `docker pull ibmcom/informix-developer-database:14.10.FC5DE`
  - DockerHub
- **encoding**:
  - code-set conversion value is extracted from the DB_LOCALE value specified at the time the connection is made
- **JDBC driver (latest)**:
  - version 4.50.4.1
  - Maven repository
- **privileged database access**:
  - user `informix`

    - password `in4mix`
    - database / schema `sysmaster`
    - INFORMIXSERVER `informix`

- **restrictions**: the IBM Informix DBMS only accepts operating system accounts or users mapped to operating system accounts as database users

- **DBeaver database connection settings**:



5.11 MariaDB Server

- **data types**:

`DBSeeder` Type	MariaDB Type
BIGINT	BIGINT
BLOB	LONGBLOB
CLOB	LONGTEXT
TIMESTAMP	DATETIME
VARCHAR	VARCHAR

- **DDL syntax**:

    - CREATE DATABASE
    - CREATE SCHEMA - n/a
    - CREATE TABLE
    - CREATE USER

- **Docker image (latest)**:

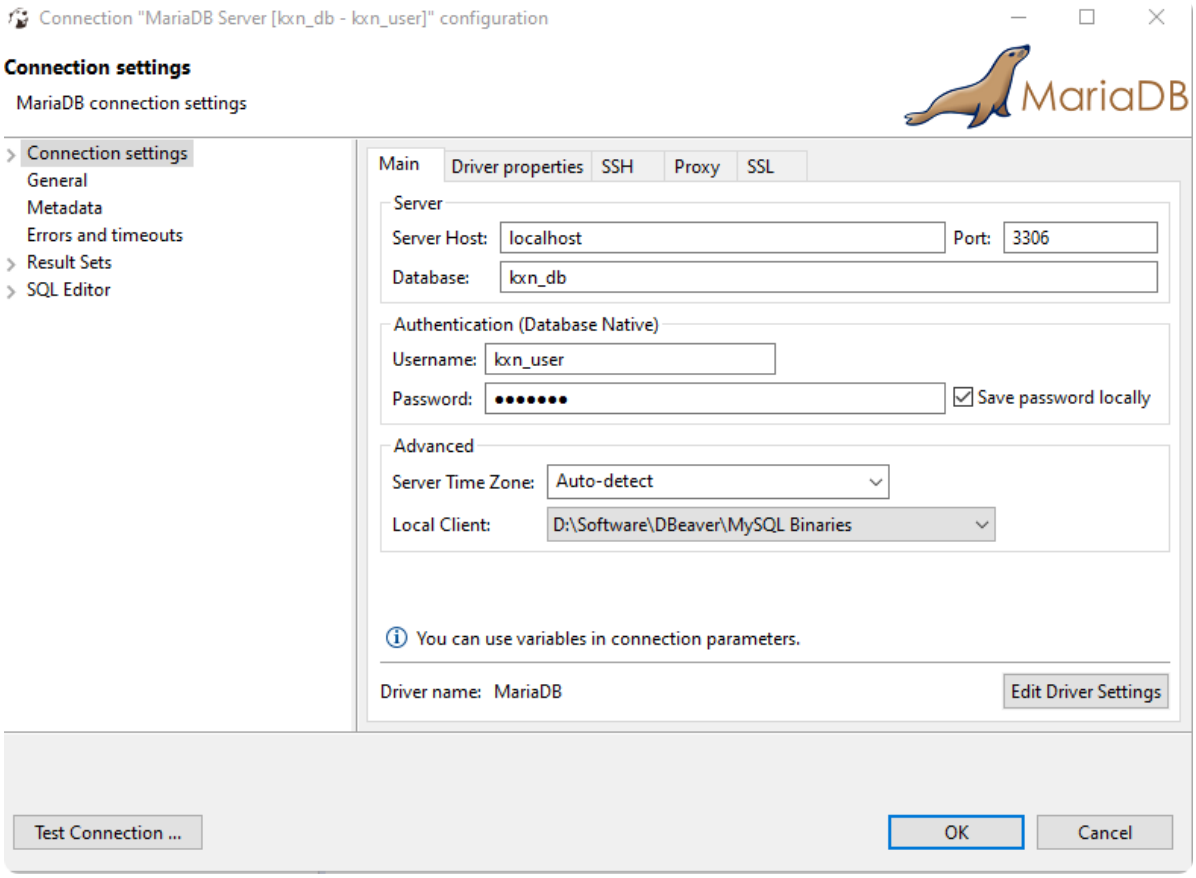
    - pull command: `docker pull mariadb:10.6.1`
    - DockerHub

- **encoding**:

    - server level: `SET character_set_server = 'latin2';`

    - database level: `CHARACTER SET = 'keybcs2'`
    - table level: `CHARACTER SET 'utf8'`
    - column level: `CHARACTER SET 'greek'`

- **issue tracking**: Jira
- **JDBC driver (latest)**:
    - version 2.7.2
    - Maven repository
- **privileged database access**:
    - user: `mysql`
    - password; `root`
- **source code**: GitHub
- **DBeaver database connection settings**:



5.12 Mimer SQL

- **data types**:

DBSeeder Type	MimerSQL Type
BIGINT	BIGINT
BLOB	BLOB
CLOB	CLOB
TIMESTAMP	TIMESTAMP
VARCHAR	NVARCHAR

- **DDL syntax**:
    - CREATE DATABASE
    - CREATE SCHEMA - n/a

    - CREATE TABLE
    - CREATE USER

- **Docker image (latest)**:

    - pull command: `docker pull mimersql/mimersql_v11.0.5a`
    - DockerHub

- **encoding**: NCHAR, NVARCHAR

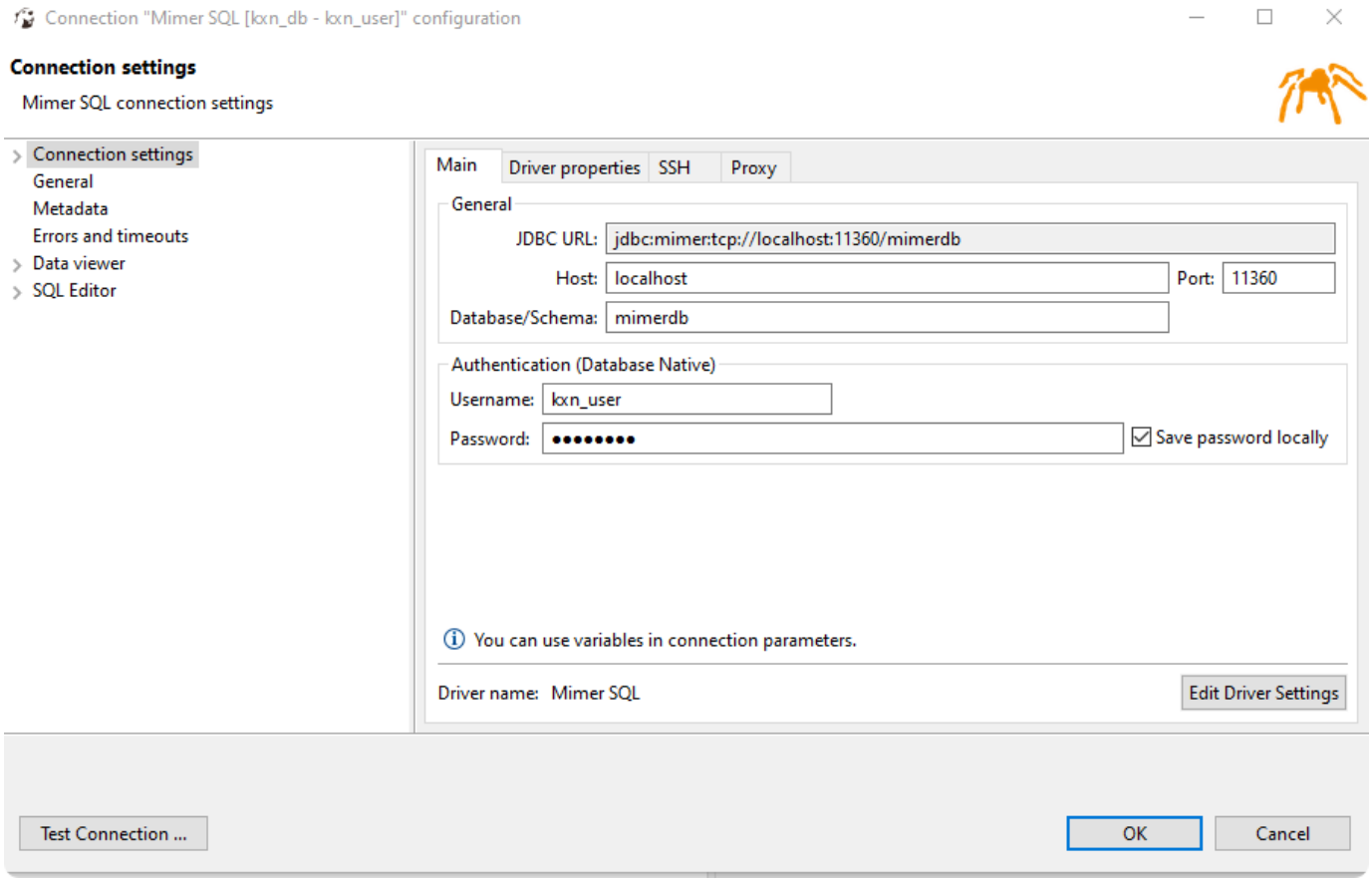
- **JDBC driver (latest)**:

    - version 3.41a
    - Mimer Website

- **privileged database access**:

    - database; `mimerdb`
    - user: `SYSADM`

- **DBeaver database connection settings**:



5.13 MonetDB

- **data types**:

`DBSeeder` Type	MonetDB Type
BIGINT	BIGINT
BLOB	BLOB
CLOB	CLOB
TIMESTAMP	TIMESTAMP
VARCHAR	VARCHAR

- **DDL syntax**:

    - CREATE DATABASE - n/a

    - CREATE SCHEMA
    - CREATE TABLE
    - CREATE USER

- **Docker image (latest)**:

    - pull command: `docker pull monetdb/monetdb:Oct2020-SP5`
    - DockerHub

- **encoding**: no special configuration should be needed

- **issue tracking**: GitHub

- **JDBC driver (latest)**:

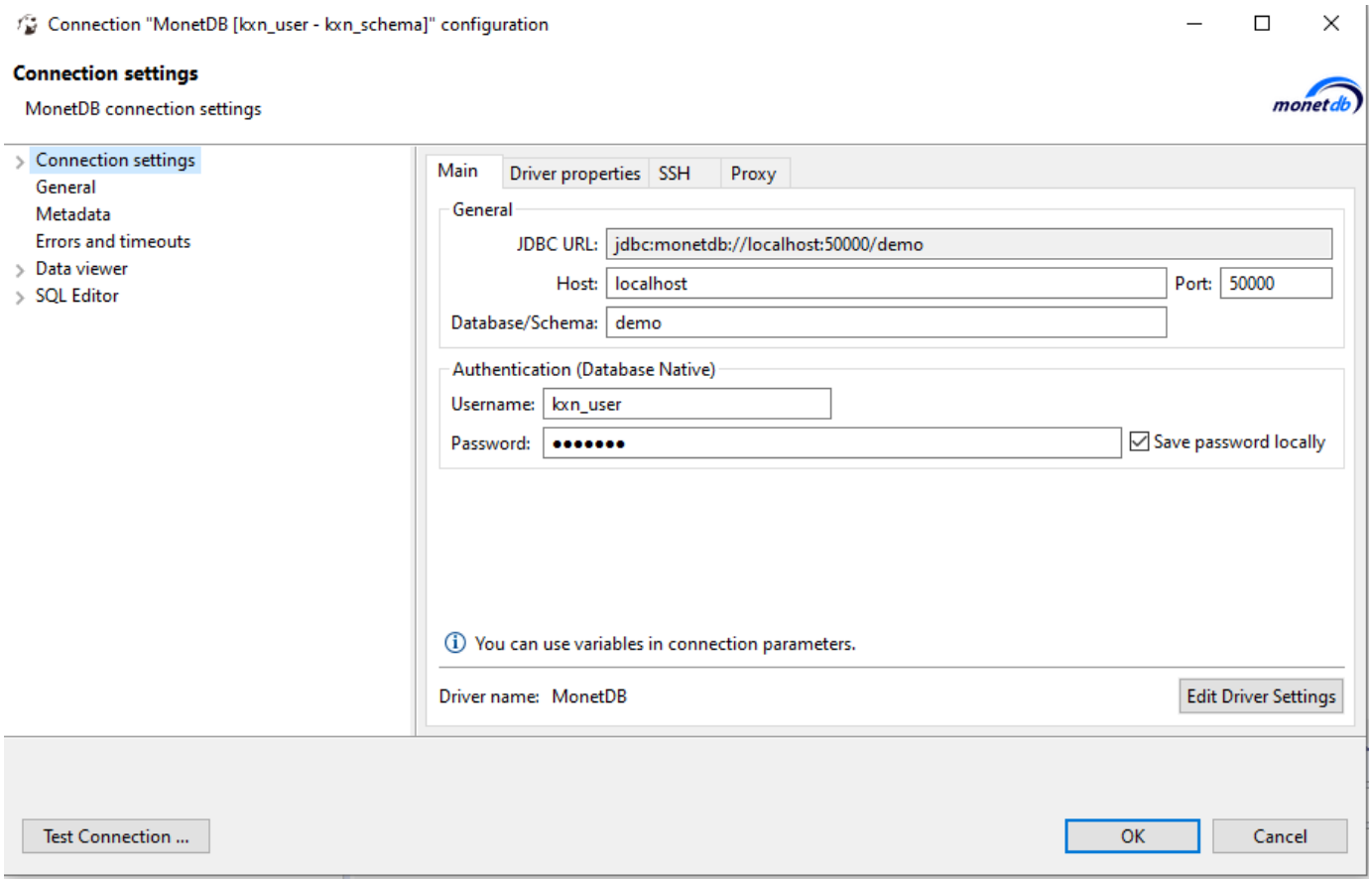
    - version 3.0.jre8
    - MonetDB Java Download Area

- **privileged database access**:

    - database: `demo`
    - user: `monetdb`
    - password: `monetdb`

- **source code**: GitHub

- **DBeaver database connection settings**:



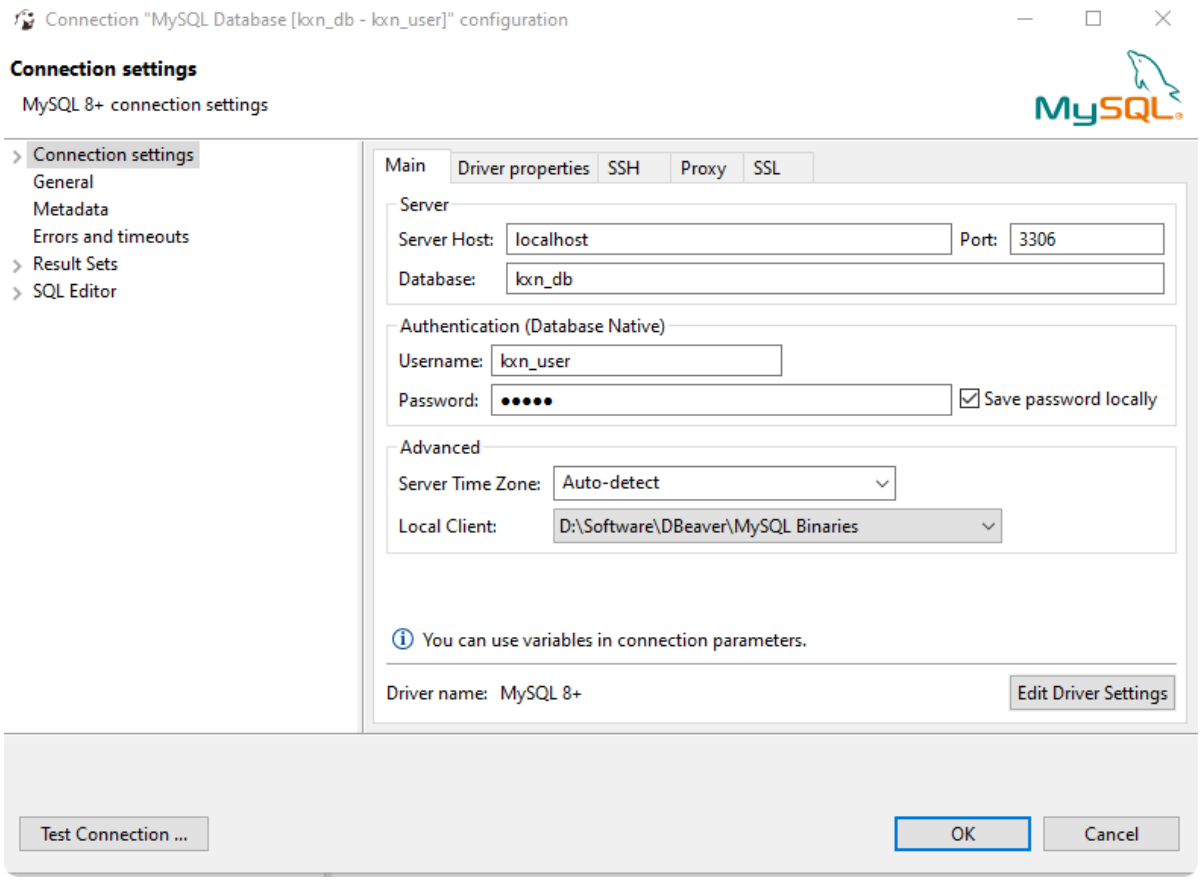
5.14 MySQL Database

- **data types**:

`DBSeeder` Type	MySQL Database Type
BIGINT	BIGINT
BLOB	LONGBLOB
CLOB	LONGTEXT

`DBSeeder` Type	MySQL Database Type
TIMESTAMP	DATETIME
VARCHAR	VARCHAR

- **DDL syntax**:
  - CREATE DATABASE
  - CREATE SCHEMA - n/a
  - CREATE TABLE
  - CREATE USER
- **Docker image (latest)**:
  - pull command: `docker pull mysql:8.0.25`
  - DockerHub
- **encoding**: for applications that store data using the default MySQL character set and collation (utf8mb4, utf8mb4_0900_ai_ci), no special configuration should be needed
- **JDBC driver (latest)**:
  - version 8.0.25
  - Maven repository
- **privileged database access**:
  - database: `sys`
  - user: `root`
- **source code**: GitHub
- **DBeaver database connection settings**:

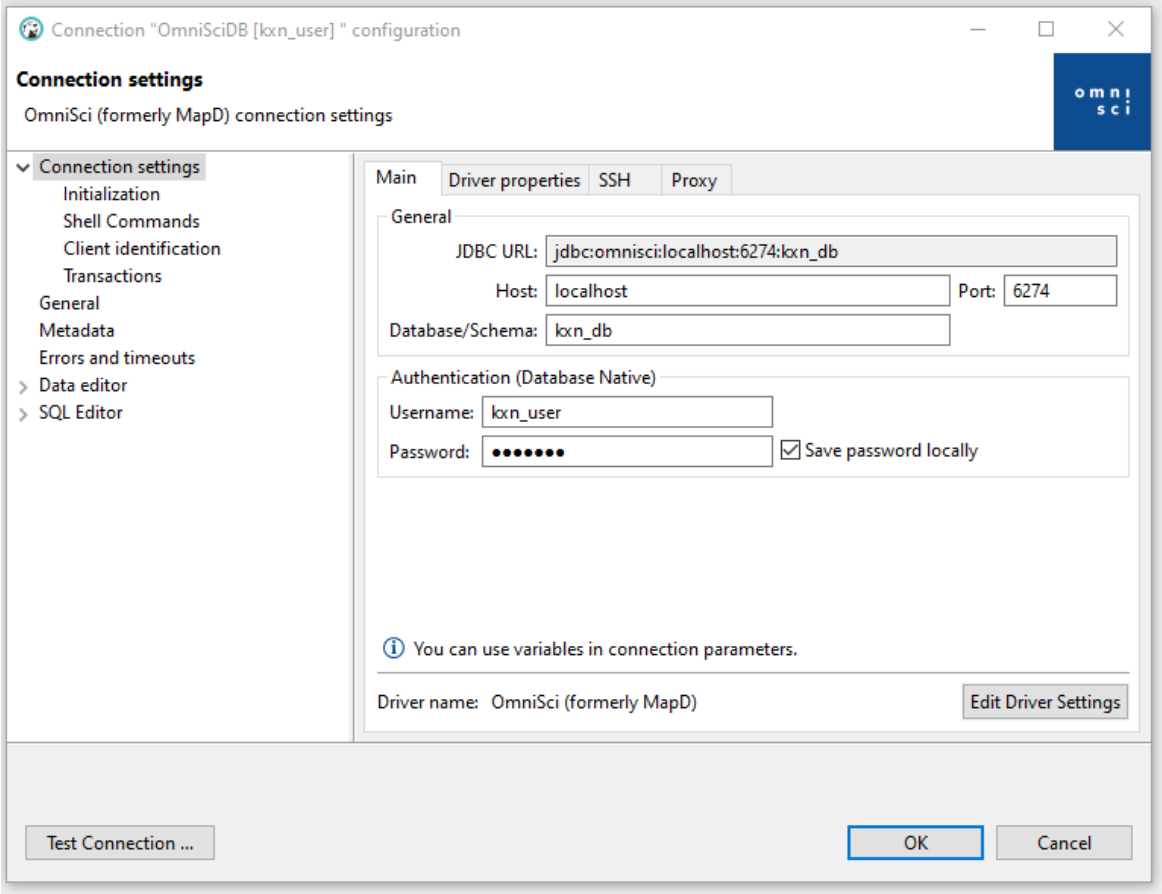


5.15 OmniSciDB

- **data types**:

DBSeeder Type	OmniSciDB Type
BIGINT	BIGINT
BLOB	TEXT ENCODING NONE
CLOB	TEXT ENCODING NONE
TIMESTAMP	TIMESTAMP(0)
VARCHAR	TEXT ENCODING NONE

- **DDL syntax**:
  - CREATE DATABASE
  - CREATE SCHEMA - n/a
  - CREATE TABLE
  - CREATE USER
- **Docker image (latest)**:
  - pull command: `docker pull omnisci/core-os-cpu`
  - DockerHub
- **encoding**: no special configuration should be needed
- **issue tracking**: GitHub
- **JDBC driver (latest)**:
  - version 5.6.0
  - Maven repository
- **privileged database access**:
  - database: `omnisci`
  - user: `admin`
- **restrictions**:
  - column and table names case sensitive
  - max. column length 32767 bytes
  - no binary columns
  - no constraints, e.g. unique keys
  - no foreign / referential keys
  - no primary key
  - no triggerss
- **source code**: GitHub
- **DBeaver database connection settings**:

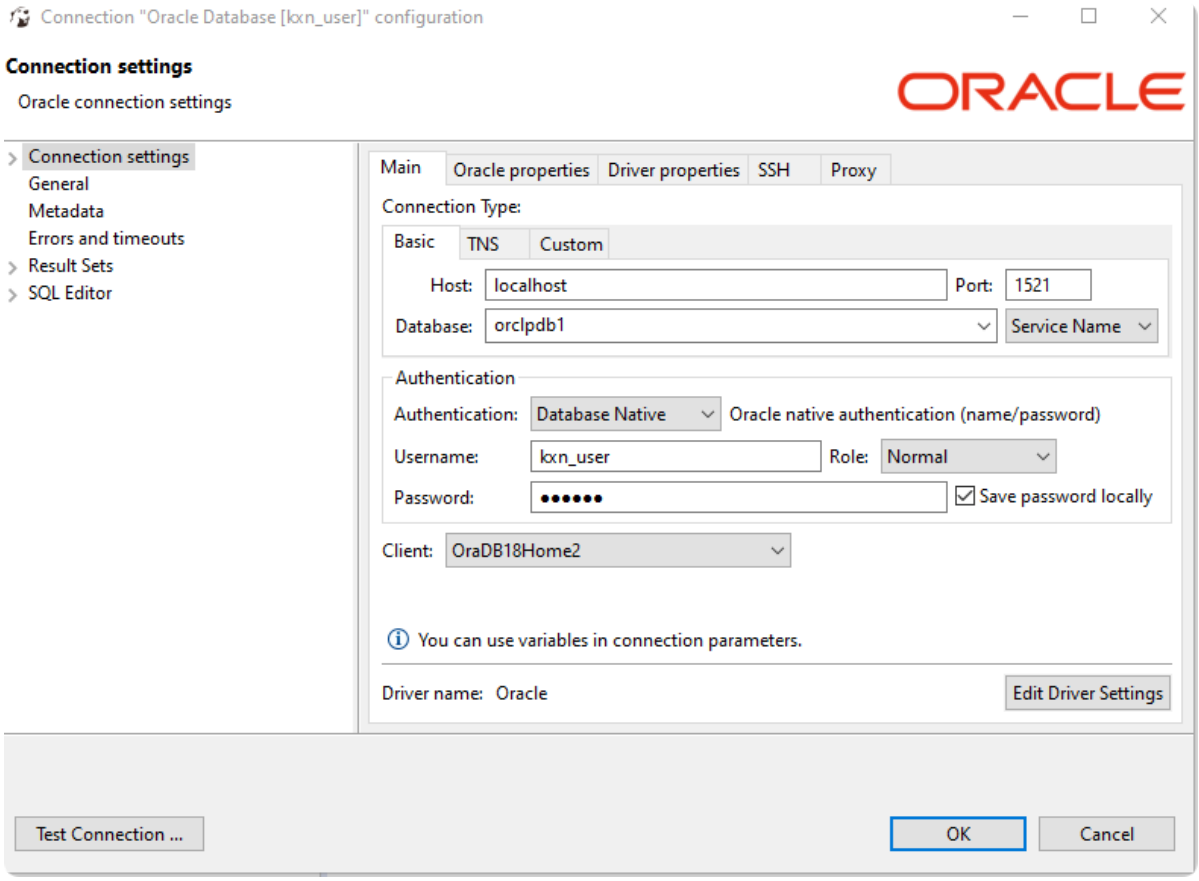


5.16 Oracle Database

- **data types**:

`DBSeeder` Type	Oracle Database Type
BIGINT	NUMBER
BLOB	BLOB
CLOB	CLOB
TIMESTAMP	TIMESTAMP
VARCHAR	VARCHAR2

- **DDL syntax**:
  - CREATE DATABASE - n/a
  - CREATE SCHEMA - n/a
  - CREATE TABLE
  - CREATE USER
- **Docker image**: DockerHub
- **encoding**: since Oracle Database 12c Release 2 the default database character set used is the Unicode character set AL32UTF8
- **JDBC driver (latest)**:
  - version 21.1.0.0
  - Maven repository
- **privileged database access**:
  - database: `orclpdb1`
  - user: `SYS AS SYSDBA`
- **DBeaver database connection settings**:



5.17 Percona Server for MySQL

- **data types**:

`DBSeeder` Type	Percona Sercver Type
BIGINT	BIGINT
BLOB	LONGBLOB
CLOB	LONGTEXT
TIMESTAMP	DATETIME
VARCHAR	VARCHAR

- **DDL syntax**:
  - CREATE DATABASE: see MySQL Database
  - CREATE SCHEMA - n/a
  - CREATE TABLE: see MySQL Database
  - CREATE USER: see MySQL Database
- **Docker image (latest)**:
  - pull command: `docker pull percona/percona-server:8.0.23-14`
  - DockerHub
- **encoding**: for applications that store data using the default MySQL character set and collation (utf8mb4, utf8mb4_0900_ai_ci), no special configuration should be needed
- **issue tracking**: Jira
- **JDBC driver (latest)**:
  - version 8.0.23
  - Maven repository
- **privileged database access**:

    - database: `sys`
    - user: `root`

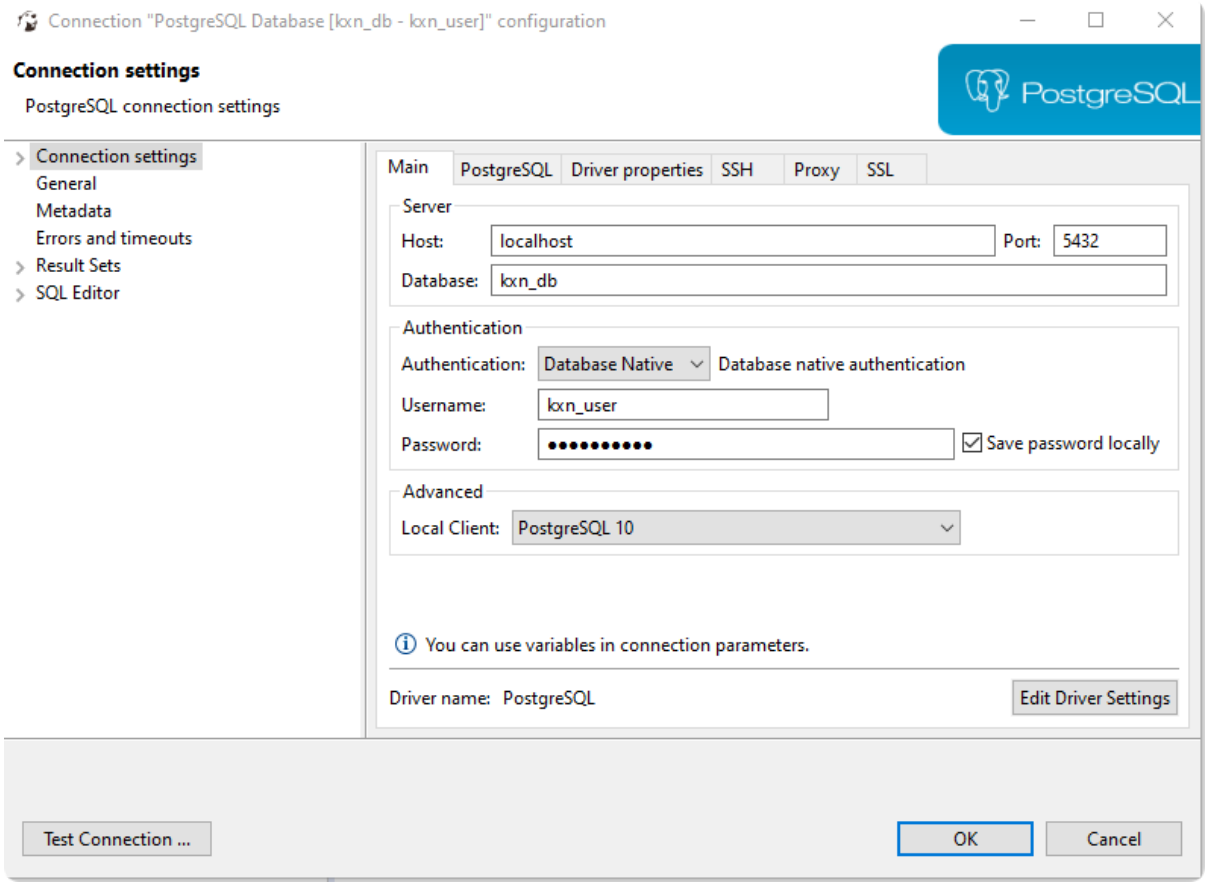
- **source code**: GitHub

5.18 PostgreSQL

- **data types**:

`DBSeeder` Type	PostgreSQL Type
BIGINT	BIGINT
BLOB	BYTEA
CLOB	TEXT
TIMESTAMP	TIMESTAMP
VARCHAR	VARCHAR

- **DDL syntax**:
    - CREATE DATABASE
    - CREATE SCHEMA
    - CREATE TABLE
    - CREATE USER
- **Docker image (latest)**:
    - pull command: `docker pull postgres:13.3-alpine`
    - DockerHub
- **encoding**: when creating the database: `CREATE DATABASE testdb WITH ENCODING 'EUC_KR' ...`
- **issue tracking**: PostgreSQL
- **JDBC driver (latest)**:
    - version 42.2.19
    - Maven repository
- **source code**: GitHub
- **DBeaver database connection settings**:



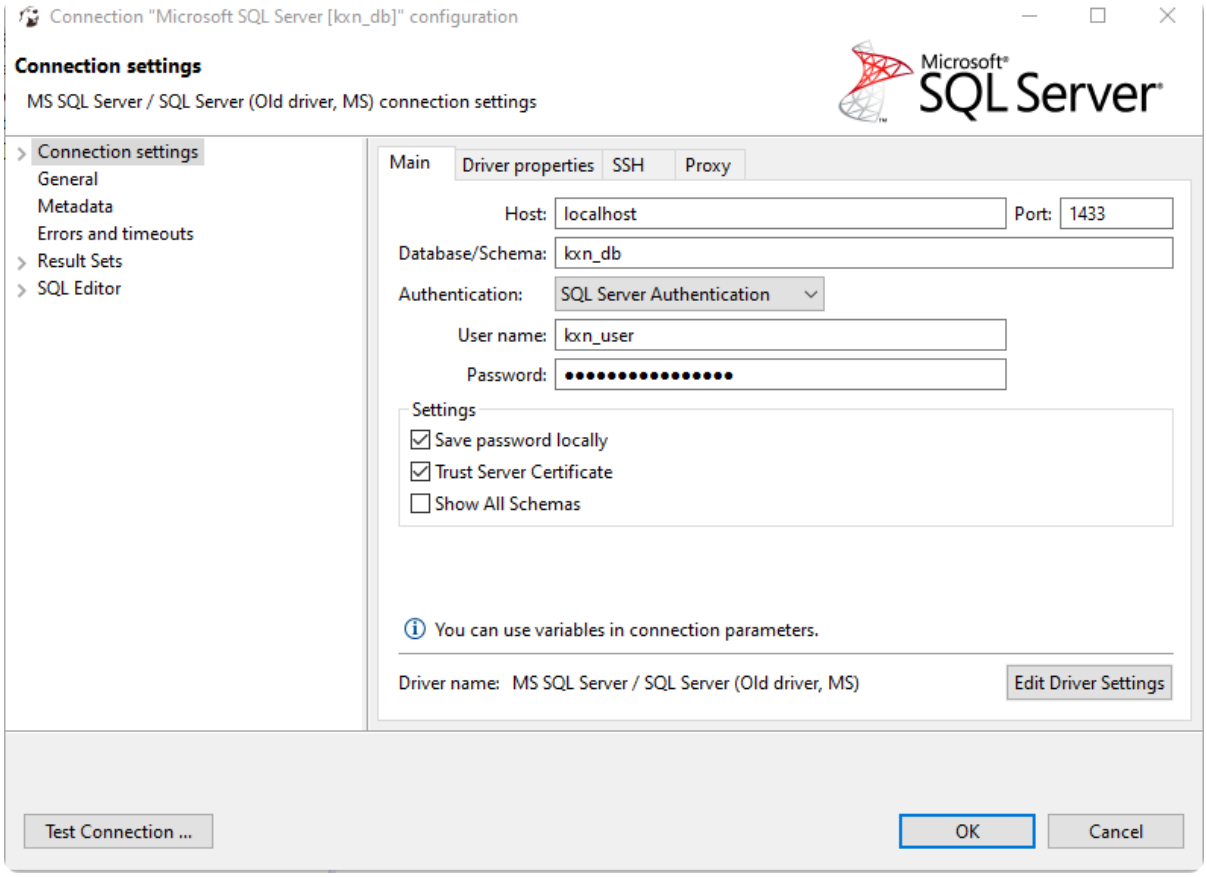
5.19 SQL Server

- **data types**:

`DBSeeder` Type	SQL Server Type
BIGINT	BIGINT
BLOB	VARBINARY (MAX)
CLOB	VARCHAR (MAX)
TIMESTAMP	DATETIME2
VARCHAR	VARCHAR

- **DDL syntax**:
  - CREATE DATABASE
  - CREATE SCHEMA
  - CREATE TABLE
  - CREATE USER
- **Docker image (latest)**:
  - pull command: `docker pull mcr.microsoft.com/mssql/server:2019-latest`
  - DockerHub
- **encoding**: to use the UTF-8 collations that are available in SQL Server 2019 (15.x), you must select UTF-8 encoding-enabled collations (_UTF8)
- **JDBC driver (latest)**:
  - version 9.2.1.jre15
  - Maven repository
- **privileged database access**:
  - database: `master`
  - user: `sa`

- **restrictions**: no full UTF-8 support in the given Docker images
- **DBeaver database connection settings**:



5.20 SQLite

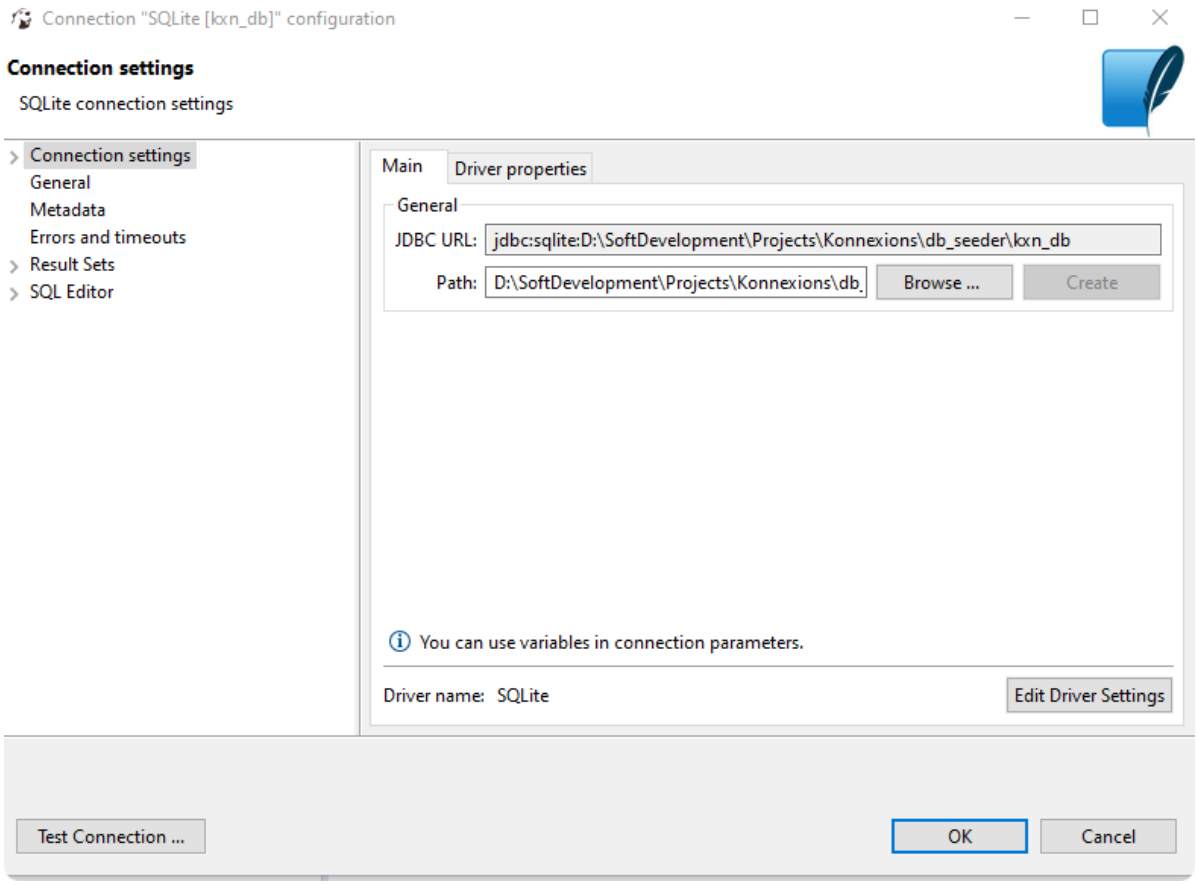
- **data types**:

DBSeeder Type	SQLite Type
BIGINT	INTEGER
BLOB	BLOB
CLOB	CLOB
TIMESTAMP	DATETIME
VARCHAR	VARCHAR2

- **DDL syntax**:
  - CREATE DATABASE - n/a
  - CREATE SCHEMA - n/a
  - CREATE TABLE
  - CREATE USER - n/a
- **encoding**: by using the following parameter: `PRAGMA encoding='UTF-8';`
- **issue tracking**: SQLite
- **JDBC driver (latest)**:
  - version 3.34.0
  - Maven repository
  - determines also the DBMS version
- **restrictions**:
  - no Docker image necessary, hence not available

    - no user management

- **source code**: SQLite
- **DBeaver database connection settings**:



5.21 trino

- **data types**:

`DBSeeder` Type	trino Type
BIGINT	BIGINT
BLOB	BLOB
CLOB	CLOB
TIMESTAMP	TIMESTAMP
VARCHAR	VARCHAR

- **DDL syntax**:

    - CREATE DATABASE - n/a
    - CREATE SCHEMA
    - CREATE TABLE
    - CREATE USER - n/a

- **Docker image (latest)**:

    - pull command: `docker pull trinodb/trino:358`
    - DockerHub

- **encoding**: full support of UTF-8 (see here)
- **issue tracking**: GitHub
- **JDBC driver (latest)**:

    - version 358
    - Maven repository

- **source code**: GitHub

5.22 VoltDB

- **data types**:

DBSeeder Type	VoltDB Type
BIGINT	BIGINT
BLOB	VARBINARY(1048576)
CLOB	VARCHAR(1048576)
TIMESTAMP	TIMESTAMP
VARCHAR	VARCHAR

- **DDL syntax**:

    - CREATE DATABASE - n/a
    - CREATE SCHEMA - n/a
    - CREATE TABLE
    - CREATE USER - n/a

- **Docker image (latest)**:

    - pull command: `docker pull voltdb/voltdb-community:9.2.1`
    - DockerHub

- **issue tracking**: Jira

- **JDBC driver (latest)**:

    - version 10.1.1
    - Maven repository

- **source code**: GitHub

5.23 YugabyteDB

- **data types**:

DBSeeder Type	YugabyteDB Database Type
BIGINT	BIGINT
BLOB	BYTEA
CLOB	TEXT
TIMESTAMP	TIMESTAMP
VARCHAR	VARCHAR

- **DDL syntax**:

    - CREATE DATABASE
    - CREATE SCHEMA
    - CREATE TABLE
    - CREATE USER

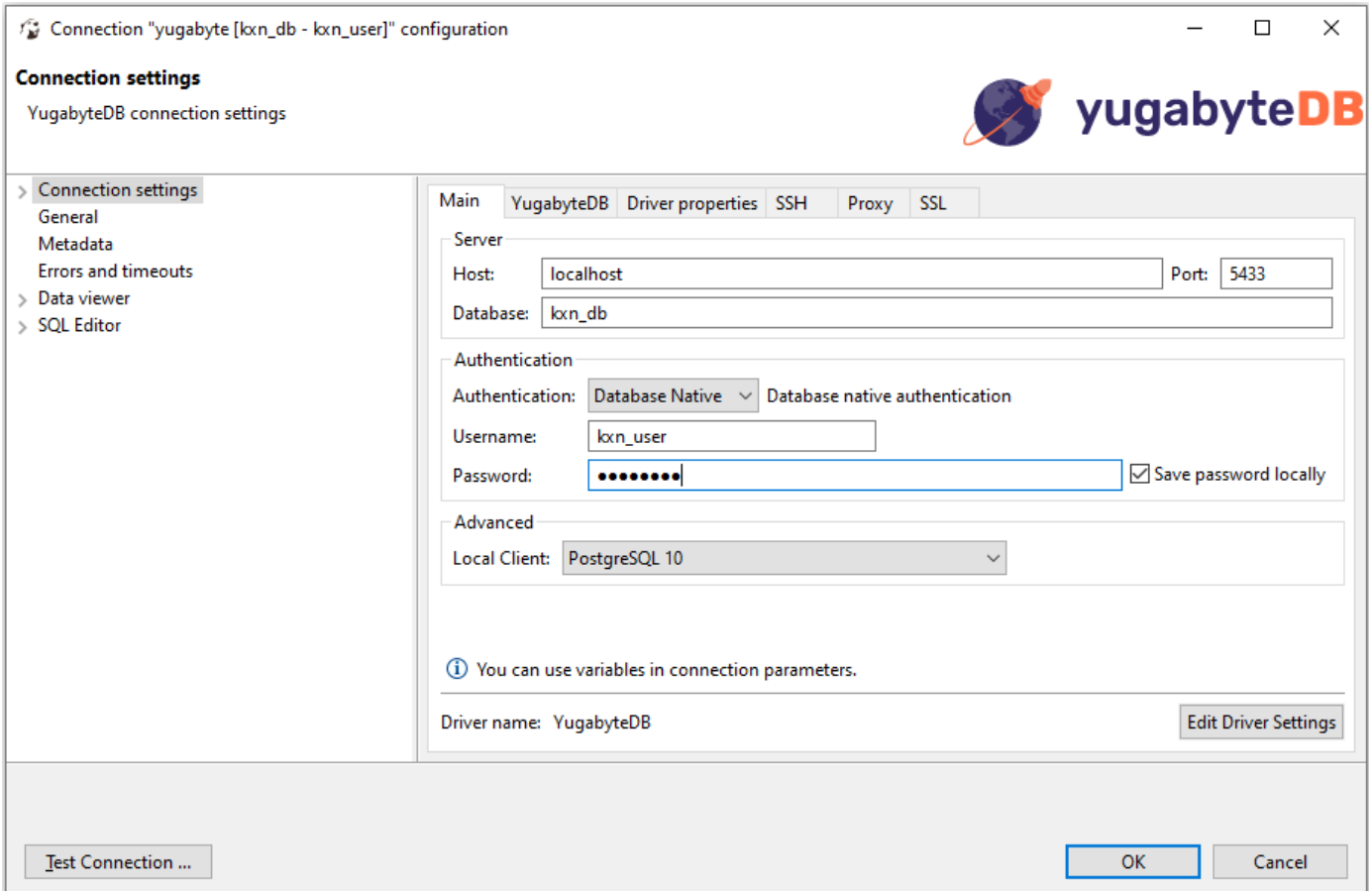
- **Docker image (latest)**:

    - pull command: `docker pull yugabytedb/yugabyte:2.7.1.1-b1`
    - DockerHub

- **encoding**: see PostgreSQL

- **issue tracking**: GitHub

- **JDBC driver (latest)**:
  - version 42.2.7-yb-3
  - Maven repository
- **source code**: GitHub
- **DBeaver database connection settings**:



6. trino

trino can integrate the following DBMS, among others:

- MySQL via the MySQL Connector,
- Oracle via the Oracle Connector, and
- PostgreSQL via the PostgreSQL Connector.
- SQL Server via the SQL Server Connector,

`DBSeeder` makes it possible to use trino's JDBC driver and the corresponding connectors as an alternative to the JDBC drivers of the DBMS suppliers. To use the trino JDBC driver, a trino server is required. With the script `db_seeder_trino_environment` a trino server can be set up. Since trino does not support the Windows operating system, a suitable Docker image is created for Windows. For Linux, e.g. Ubuntu, the script can alternatively be used to perform a local installation of the trino server.