SQLTools

Benutzerhandbuch

für die enitsys Datenbank Automatisierungs-Tools

SQL2csv	zur Erstellung einer CSV-Datei aus dem Ergebnis eines SQL-Select-Statements
SQL2file	zur Erstellung von Text-Dateien aus dem Ergebnis eines SQL-Select-Statements
SQL2xml	zur Erstellung von XML-Dateien aus dem Ergebnis von SQL-Select-Statements
CSV2sql	zur Befüllung einer Datenbanktabelle aus den Daten einer CSV-Datei
CSVAnalyser	zur Formatanalyse einer CSV-Datei
SQLAnalyser	zur Ergebnisanalyse eines SQL-Select-Statements
SQLcmd	zur Ausführung eines SQL Befehls
Encrypt	zur Verschlüsselung eines Passworts

Inhalt

1 Einführung

Bei der Entwicklung von Datenbankanwendungen für unsere Kunden benötigen wir immer wieder „kleine Helferlein“ für Automatisierungsaufgaben. Für einige Aufgaben gibt es Lösungen in verschiedensten Varianten im Netz, für andere könnte man auf proprietäre Lösungen einzelner Datenbankhersteller zurückgreifen und für wieder andere Aufgaben mussten wir eigene Lösungsansätze entwickeln.

Allen externen Lösungen gemein ist die Problematik, dass jedes Tool eine eigene Konfigurations-Syntax unterstützt und insbesondere eine untereinander inkompatible Bereitstellung der Verbindungsparameter erfordert, die zudem oft nicht mit den Sicherheitsanforderungen des Kunden vereinbar ist.

Über die Jahre hinweg haben wir deshalb ein kleines Set solcher Mini-Programme entwickelt und kontinuierlich verbessert, die unsere Anforderungen bei beliebigen Kunden ideal erfüllen.

SQL2csv dient der Erstellung einer CSV-Datei[1] aus dem Ergebnis eines SQL-Select-Statements. Wie bei solchen Programmen üblich, kann das Zielformat in weiten Bereichen frei definiert werden.

SQL2file dient der Erstellung mehrerer Dateien aus dem Ergebnis eines SQL-Select-Statements. Wir benutzen es beispielsweise um komplette Datenbank-Schemen „abzuziehen“. Ein solches Programm hatten wir nicht im Netz gefunden.

SQL2xml dient der Erstellung von XML-Dateien aus dem Ergebnis von SQL-Select-Statements. Wir benutzen es zur Datengewinnung zwecks Dokumentation von Datenbanken. Auch dieses Programm entstand, weil wir nichts Vergleichbares im Netz gefunden hatten.

CSV2sql dient der Befüllung einer Datenbanktabelle aus den Daten einer CSV-Datei. Diese Version zielt eher auf einen unkomplizierten Upload „mal eben abgezogener“ Daten als auf den „produktiven“ Einsatz.

CSVAnalyser dient der schnellen Formatanalyse einer CSV-Datei. Als Ergebnis liefert das Tool sowohl eine passende Konfigurationsdatei für **CSV2sql** als auch ein passendes `create table`-Statement.

SQLAnalyser dient der schnellen Ergebnisanalyse eines SQL-Select-Statements. Die Ausgabe des Tools ist für Regressionstests gedacht.

SQLcmd dient der Ausführung eines SQL Befehls im Batch-Betrieb.

Encrypt erzeugt ein verschlüsseltes Datenbankpasswort, das in den Verbindungsparametern benutzt werden kann.

[1] Als CSV-Datei bezeichnen wir hier nicht nur die Definition nach RFC 4180 sondern alle ähnlichen Formate zum Speichern strukturierter Daten in einer Textdatei.

2 Voraussetzungen für die Verwendung der SQLTools

2.1 Download

Die jeweils neueste Version von SQLTools steht unter http://www.enitsys.com/sqltools.html zum Download Verfügung.

Die aktuelle Version dieses Handbuchs finden Sie unter http://www.enitsys.com/SQLTools.pdf.

2.2 Java-Installation

Für die Verwendung der SQLTools ist eine Java-Runtimeumgebung (JRE) erforderlich. Mindestvoraussetzung ist Java 8.

Für den Aufruf der Beispiel-Windows-Batch-Files muss die JAVA_HOME Umgebungsvariable gesetzt sein.

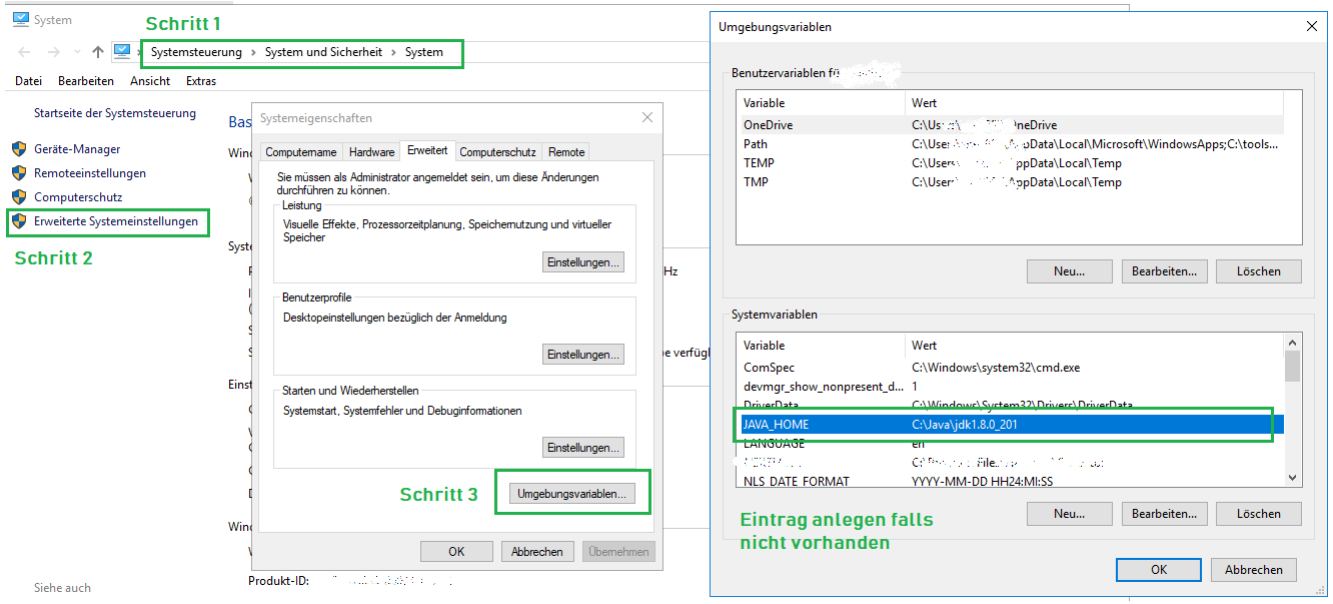

Abbildung 1: So wird die JAVA_HOME Umgebungsvariable gesetzt

2.3 Datenbank-Verbindungsparameter

Für die Verbindungsparameter verwenden alle SQLTools sogenannte Properties-Dateien.

Jede Datei sollte die folgenden 4 Properties setzen:

```
driver=Java-Classname des JDBC-Treibers
url=JDBC-Treiber-spezifische Datenbank-URL
user=Datenbank-User
password=Datenbank-Passwort
```

user und/oder **password** können entfallen, wenn sie für den Datenbankzugriff nicht benötigt werden. Das Datenbankpasswort kann auch verschlüsselt abgespeichert werden[2].

Die Datei kann weitere datenbankspezifische Einträge enthalten, die unverändert bei der Verbindungserstellung übergeben werden.

Sind die Properties **schema** oder **catalog** vorhanden, werden diese Eigenschaft in der Connection gesetzt.

Sicherheitshinweis: Diese Dateien enthalten unverschlüsselte Passworte. Sie müssen unbedingt in einem Verzeichnis gespeichert werden, **auf das nur der jeweilige Benutzer Zugriff hat**.

Standardmäßig sollten die Verbindungs-Dateien im Benutzerverzeichnis oder einem Unterverzeichnis davon gespeichert sein.

Der Dateiname sollte folgendermaßen aufgebaut sein: *Verbindungsname*.**properties**

Alle SQLTools sind auch dafür ausgelegt, auf einem gemeinsamen Laufwerk von mehreren Benutzern verwendet zu werden. In diesem Fall soll jeder Benutzer den gleichen relativen Verzeichnispfad (bezogen auf sein „Home-Verzeichnis“) für die Speicherung seiner Verbindungsdateien benutzen.

In vielen Projekten (und auch in den Beispielen) verwenden wir dafür den Verzeichnisnamen `jdbcconnection`.

Für eine lokale Oracle-Datenbank, sähe die Datei ähnlich wie diese aus:

```
driver=oracle.jdbc.OracleDriver
url=jdbc:oracle:thin:@//localhost:1521/TEST
user=Herbert
password=<geheim>
```

In den Beispielen erwarten wir datenbankspezifische Dateien namens `sqltoolsexample.<rdbms>.properties`.

Für die Oracle-Beispiele muss unser Beispiel-User Herbert mit seinem Home-Verzeichnis `C:\Users\Herbert` also eine Datei `C:\Users\Herbert\jdbcconnection\sqltoolsexample.Oracle.properties` anlegen.

[2] siehe Kapitel 2.4

2.4 Verschlüsselung des Datenbankpassworts

In einigen Fällen ist es sinnvoll, Passwörter in den Verbindungsparametern verschlüsselt abzulegen. Dies kann z.B. im Support-Fall erforderlich sein, wenn ein Dritter Einsicht in die Verbindungsparameter bekommen muss, ohne dass ihm das Passwort bekannt werden darf.

Hierzu bietet **SQLTools** einen einfachen Mechanismus.

Ein Verschlüsselungspasswort wird in einer separaten Datei **secret.properties** im gleichen Verzeichnis wie die Verbindungsparameter abgelegt.[3] Diese Datei muss folgenden Eintrag enthalten.

```
secret=Verschlüsselungs-Passwort
```

Mithilfe von **Encrypt** kann das unverschlüsselte Datenbankpasswort verschlüsselt werden und in den Verbindungsparametern durch das verschlüsselte Passwort ersetzt werden.

Die Datei mit den Verbindungsparametern hat dann (mindestens) die folgenden 4 Einträge:

```
driver=Java-Classname des JDBC-Treibers
url=JDBC-Treiber-spezifische Datenbank-URL
user=Datenbank-User
encrypted-password=verschlüsseltes Passwort
```

Enthält eine Datei mit Verbindungsparametern keine Property **`password`** aber eine Property **`encrypted-password`**, so wird das Datenbankpasswort mittels des Verschlüsselungspasswort wiederhergestellt.
Ist die Property **`password`** ebenfalls vorhanden, wird **`encrypted-password`** ignoriert.

Sicherheitshinweis 1: Die Datei **secret.properties** enthält ein unverschlüsseltes Passwort. Sie muss unbedingt in einem Verzeichnis gespeichert werden, **auf das nur der jeweilige Benutzer Zugriff hat**.

Sicherheitshinweis 2: Es ist nicht unmöglich, das Verschlüsselungspasswort zu rekonstruieren, wenn sowohl das verschlüsselte als auch das unverschlüsselte Passwort bekannt sind. Hierauf sollte insbesondere geachtet werden, wenn die unverschlüsselten Datenbankpasswörter mehreren Personen bekannt sind.

Sicherheitshinweis 3: Der Mechanismus ist primär als Verschleierung (Obfuscation) für den eingangs genannten oder ähnliche Anwendungsfälle konzipiert. **Für den Einsatz in sicherheitskritischen Anwendungen ist dieser Mechanismus nicht ausreichend.**

[3] Dateiname und Verzeichnis kann über die System-Property `secret.file` geändert werden.

3 SQLTools

3.1 SQL2csv

Mit **SQL2csv** wird eine CSV-Datei aus dem Ergebnis eines SQL-Select-Statements erstellt. Wie bei solchen Programmen üblich, kann das Zielformat in weiten Bereichen frei definiert werden.

Syntax:	`java [env] com.enitsys.sqltools.SQL2csv Task [Connection [OutDir]]`

env	Java VM Optionen (z.B. Classpath -cp und System Properties -D)
Task	Name des auszuführenden Tasks (siehe: Taskeigenschaften definieren)
Connection	Name der Datenbankverbindung (siehe: Datenbankverbindung definieren)
OutDir	Name des Zielverzeichnisses (siehe: Zielverzeichnis und Ausgabedatei definieren)

Zur Definition des SQL-Select-Statements siehe: SQL-Statements definieren.

Zur Definition des CSV-Formats siehe: CSV-Dateiformat definieren sowie Spaltenformate definieren.

3.2 SQL2file

Mit **SQL2file** werden mehrere Textdateien aus den vorformatierten Inhalten eines SQL-Statements in ein Zielverzeichnis geschrieben.

Das SQL-Select-Statements muss mindestens zwei Spalten zurückliefern:

1. Spalte *Dateiname*
   Der Dateinamen kann absolut oder relativ angegeben werden. Ist er relativ angegeben, bezieht er sich auf das Zielverzeichnis.
2. Spalte *Dateiinhalt*
   Der Dateiinhalt wird unverändert in die jeweilige Zieldatei übernommen.

Syntax: `java [env] com.enitsys.sqltools.SQL2file Task [Connection [OutDir]]`

env	Java VM Optionen (z.B. Classpath -cp und System Properties -D)
Task	Name des auszuführenden Tasks siehe: Taskeigenschaften definieren
Connection	Name der Datenbankverbindung siehe: Datenbankverbindung definieren
OutDir	Name des Zielverzeichnisses siehe: Zielverzeichnis und Ausgabedatei definieren

Zur Definition des SQL-Select-Statements siehe: SQL-Statements definieren.

3.3 SQL2xml

Mit **SQL2xml** wird eine XML-Datei aus den Ergebnissen eines oder mehrerer SQL-Statements erstellt.

Syntax:	`java [env] com.enitsys.sqltools.SQL2xml Task [Connection [OutDir]]`

env	Java VM Optionen (z.B. Classpath -cp und System Properties -D)
Task	Name des auszuführenden Tasks siehe: Taskeigenschaften definieren
Connection	Name der Datenbankverbindung siehe: Datenbankverbindung definieren
OutDir	Name des Zielverzeichnisses siehe: Zielverzeichnis und Ausgabedatei definieren

Zur XML-Definition siehe: XML-Elemente definieren.

3.4 CSV2sql

Mit **CSV2sql** wird eine Datenbanktabelle aus den Daten einer CSV-Datei befüllt.

Syntax:	`java [env] com.enitsys.sqltools.CSV2sql Task [Connection [InFile]]`

env	Java VM Optionen (z.B. Classpath -cp und System Properties -D)
Task	Name des auszuführenden Tasks (siehe: Taskeigenschaften definieren)
Connection	Name der Datenbankverbindung (siehe: Datenbankverbindung definieren)
InFile	Name des CSV-Datei (siehe: Quelldatei/Inputdatei definieren)

Zur Definition des CSV-Formats siehe: CSV-Dateiformat definieren sowie Spaltenformate definieren.

Zur Definition der Zieltabelle siehe: Zieltabelle definieren.

Zur Definition des Mappings siehe: Spalten einer CSV-Datei identifizieren und mappen

3.5 CSVAnalyser

Mit **CSVAnalyser** wird eine CSV-Datei analysiert. Ergebnis sind ein „passendes“ create table- Statement sowie eine Konfigurationsdatei um die Datei mit **CSV2sql** verarbeiten zu können.

Syntax:	`java [env] com.enitsys.sqltools.CSVAnalyser Task [OutDir [InFile]]`

env	Java VM Optionen (z.B. Classpath -cp und System Properties -D)
Task	Name des auszuführenden Tasks (siehe: Taskeigenschaften definieren)
OutDir	Name des Zielverzeichnisses siehe: Zielverzeichnis und Ausgabedatei definieren (Es wird nur die Property out.dir unterstützt)
InFile	Name des CSV-Datei (siehe: Quelldatei/Inputdatei definieren)

Zur Definition des CSV-Formats siehe: CSV-Dateiformat definieren sowie Spaltenformate definieren.

Zur Definition des Mappings siehe: Spalten einer CSV-Datei identifizieren und mappen

Die Ausgabedateien heißen

- ***Task*.properties** Konfigurationsdatei für **CSV2sql**.
  Dies Datei enthält auch detaillierte Hinweise zur Konfiguration.
- ***Task*.sql** Zu den Daten passendes create table Statement.
  Dieses kann beispielsweise mit **SQLcmd** ausgeführt werden.

Durch Angabe der Task Property `db.productname` (siehe: Datenbankeigenschaften) wird die korrekte Syntax für das **`create table`** Statement definiert.

3.6 SQLAnalyser

SQLAnalyser ermittelt Kennzahlen aus dem Ergebnis eines SQL-Select-Statements erstellt. Diese Kennzahlen sind insbesondere für Regressionstests geeignet.

Syntax:	`java [env] com.enitsys.sqltools.SQLAnalyser Task [Connection [OutDir]]`

env	Java VM Optionen (z.B. Classpath -cp und System Properties -D)
Task	Name des auszuführenden Tasks (siehe: Taskeigenschaften definieren)
Connection	Name der Datenbankverbindung (siehe: Datenbankverbindung definieren)
OutDir	Name des Zielverzeichnisses (siehe: Zielverzeichnis und Ausgabedatei definieren)

Zur Definition des SQL-Select-Statements siehe: SQL-Statements definieren.

3.7 SQLcmd

Mit **SQLcmd** wird ein SQL Befehl im Batchbetrieb ausgeführt.

Hinweis: **SQLcmd** ist für die Ausführung einzelner, eher einfacher SQL Befehle konzipiert. Für größere SQL Skripte sind andere Tools besser geeignet.

Syntax:	`java [env] com.enitsys.sqltools.SQLcmd Task [Connection]`

env	Java VM Optionen (z.B. Classpath -cp und System Properties -D)
Task	Name des auszuführenden Tasks (siehe: Taskeigenschaften definieren)
Connection	Name der Datenbankverbindung (siehe: Datenbankverbindung definieren)

Zur Definition des SQL Statements siehe: SQL-Statements definieren

3.8 Encrypt

Mit **Encrypt** wird ein verschlüsseltes Datenbankpasswort erzeugt, das in den Verbindungsparametern statt des unverschlüsselten Passworts benutzt werden kann.

Das Tool ist zur manuellen Ausführung gedacht. Es gibt einen Eintrag der Form

encrypted-password=<*encrypted-password*>

auf der Konsole aus, der dazu bestimmt ist, den entsprechenden Eintrag für das Datenbankpasswort in den Datenbankverbindungsparametern zu ersetzen.

Syntax:	**java** [*env*] **com.enitsys.sqltools.Encrypt** *password*

env	Java VM Optionen (z.B. Classpath -cp und System Properties -D)
password	Unverschlüsseltes Datenbank-Passwort

4 Beispiele

Die Distribution enthält zahlreiche Beispielanwendungen für verschiedene RDBMS.

4.1 Einrichtung und Besonderheiten der RDBMS

Zur Ausführung der Beispiele sind JDBC Treiber für das jeweilige RDBMS erforderlich. Diese sind in der „normalen“ SQLTools -Distribution nicht enthalten. Es steht eine erweiterte Distribution sqltools_incl_example_libs.zip zur Verfügung, in der alle erforderlichen jar-Files enthalten sind. Deren Weitergabe erfolgt ausdrücklich nur zu Demonstrationszwecken.

4.1.1 Oracle (und OracleCS)

SQLTools wurden ursprünglich für Oracle Datenbanken entwickelt und die meisten Funktionalitäten sind ständig mit verschiedenen Versionen von Oracle Datenbanken im produktiven Einsatz.

Für Oracle Datenbanken existieren zwei Sets an Beispielen:

- Oracle,
  die traditionelle Verwendung mit Nicht-casesensitiven Tabellen- und Spaltennamen und
- OracleCS,
  wo die Verwendung casesensitiver Tabellen und Spaltennamen demonstriert wird.

Die Beispielanwendungen erfordern eine Datenbank mit einem Schemauser mit CONNECT-Rolle und der Berechtigung Tabellen, Views und Prozeduren anzulegen.

Die Verbindungsparameter müssen entsprechend angepasst werden (url, user, password).

Hinweis: Wird das gleiche Schema für Oracle und OracleCS genutzt, existieren jeweils 2 Versionen der Beispieltabellen, die sich nur in der Groß-/Kleinschreibung unterscheiden. Dies führt zu einem Abbruch bei der abschließenden Erstellung der HTML-Schemadokumentation im Beispiel für SQL2xml unter Windows, da Windows-Dateinamen nicht casesensitiv sind. Die Funktionalität von SQL2xml ist davon nicht betroffen.

4.1.2 Google BigQuery

Das Beispiel ist eher akademischer Natur. Auch wenn wir zeigen, dass es funktioniert, erwarten wir nicht, dass SQLTools tatsächlich für dieses DBMS eingesetzt wird.

Die Beispielanwendung erfordern ein Google Cloud Projekt mit einem BigQuery Dataset. (Ist dafür eine andere Location als „us“ konfiguriert, muß diese in den Verbindungsparametern angegeben werden.)

In der Beispiel-Url ist die Authentifizierungsmethode mittels Google Service Account konfiguriert. Das war für uns am praktikabelsten. Hierzu wird ein solcher Account mit den entsprechenden Projekt-Berechtigungen benötigt und der Dateipfad zu den Schlüsselinformationen muss in der Verbindungsurl hinterlegt werden.

4.1.3 Microsoft Access

Ursprünglich hielten wir auch dieses Beispiel für eine akademische Übung. Eine entsprechende Bemerkung stieß allerdings auf hefti-

gen Protest bei einigen unserer Kunden, die SQLTools im regelmäßigen Einsatz für Access Datenbanken verwenden.

Die Beispiele sollten mit jeder Access Datenbank funktionieren. Im Ordner /examples/jdbcconnection befindet sich auch zwei leere Datenbanken sqltools-example.accdb und sqltools-example-encrypted.accdb.

Letztere ist passwortgeschützt (siehe Beispiel der Verbindungsparameter). Für den im Beispiel verwendeten JDBC-Treiber muss in diesem Fall ein zusätzlicher „Opener“ definiert werden. Dieser ist in `sqltools.jar` enthalten.

4.2 Beispielprogramme

Die Beispielprogramme sind als Windows-Batchdateien mit der Dateinamenserweiterung .bat im Verzeichnis `examples` verfügbar. Der Name des jeweiligen RDBMS dient als Präfix für den Dateinamen. Also beispielsweise `oracle_01_CSVAnalyzer.bat`.

4.2.1 00_SQLcmd_HelloWorld

Dieses HelloWorld Program öffnet eine JDBC-Connection zu einer Oracle Datenbank. Über diese Verbindung wird ein PLSQL Programm ausgeführt. Dieses ruft `DBMS_OUTPUT.PUT_LINE("Hello World")` auf. Die Datenbankausgabe wird von SQLcmd im Kommandozeilenfenster ausgegeben.

Das Beispiel ist nur für Oracle und OracleCS verfügbar.

4.2.2 01_CSVAnalyzer

Dieses Program analysiert 3 mitgelieferte csv-Dateien und erstellt für jede dieser csv-Dateien jeweils zwei Ergebnisdateien.

Die sql-Dateien enthält ein create table Statement zur Erzeugung einer Tabelle in dem jeweiligen RDBMS, in die die Daten importiert werden könnten. Die properties-Dateienthält eine Konfiguration mit der CSV2sql diese Datei in eine solche Tabelle importieren könnte.

Das Beispiel ist für alle RDBMS verfügbar.

4.2.3 02_SQLcmd_installSchema

Dieses Programm öffnet eine JDBC-Connection zu einer Datenbank. Über diese Verbindung wird ein SQL-Skript ausgeführt, das die Beispieltabellen (und -Views) in der Datenbank erzeugt.

Das Beispiel ist für alle RDBMS verfügbar, das SQL-Skript unterscheidet sich je nach RDBMS.

Die Beispielview Bewohner wird nur dann erzeugt, wenn auch das Beispiel 05_SQL2file_generateFiles unterstützt wird.

In den meisten RDBMS werden eventuell vorhandene gleichnamige Tabellen überschrieben.

Für Access ist das nicht der Fall.

4.2.4 03_CSV2sql_uploadData

Dieses Programm importiert die Daten aus 3 CSV-Dateien in die 3 Beispieltabellen, die mit 02_SQLcmd_installSchema erstellt wurden.

Dabei werden verschiedene Arten der Konfiguration verwendet.

- Zunächst eine generische Minimalkonfiguration, die für unterschiedliche Tabellen verwendet werden kann. Der Tabellenname wird dabei als System Property übergeben.
- Dann eine typische Konfiguration, wenn Spaltennamen in Tabelle und Datei übereinstimmen. Hier werden einzelne Feldformate der Datei definiert.
- Schließlich eine ausführlichere Konfiguration, wenn kein 1-zu-1 Mapping zwischen den Spalten in Datei und Tabelle möglich ist.

Das Beispiel ist für alle RDBMS verfügbar.

4.2.5 04_SQL2csv_exportData

Dieses Programm exportiert die Daten aus den 3 Beispieltabellen in CSV-Dateien.

Das Ausgabeformat soll dem Eingabeformat des Programms 03_CSV2sql_uploadData entsprechen.

Das Beispiel ist für alle RDBMS verfügbar.

4.2.6 05_SQL2file_generateFiles

Dieses Programm erzeugt für jeden Datensatz der View Bewohner eine Textdatei. Dabei wird für jede Stadt ein Verzeichnis erzeugt.

Das Beispiel ist nur für Oracle und OracleCS verfügbar.

4.2.7 06_SQL2xml_documentSchema

Dieses Programm erzeugt eine xml-Datei mit Meta-Informationen zu den Datenbankobjekten des Beispielschemas. Anschließend wird diese xml-Datei mittels xslt in eine einfache html-Schemadokumentation transformiert.

Das Beispiel ist nur für Oracle und OracleCS verfügbar.

5 Konfiguration/Referenz

Allen SQLTools wird beim Aufruf als erstes Programmargument der Name des auszuführenden Tasks übergeben. Unter diesem Namen wird eine Konfigurationsdatei im Konfigurationsverzeichnis erwartet, das gegebenenfalls durch eine System Property definiert wird.

`configuration.dir`
: Diese System Property bestimmt das Konfigurationsverzeichnis. Sie kann eine relative oder absolute Pfadangabe enthalten. Eine relative Pfadangabe bezieht sich auf das Basisverzeichnis (Current Dir).
Voreinstellung ist ein leerer Eintrag, was bedeutet, dass die TaskProperties-Datei im Basisverzeichnis (Current Dir) abgelegt ist.

`config$import`
: Diese Task Property definiert eine weitere zu importierende Property Datei. Der Dateiname ist relativ zum Konfigurationsverzeichnis.

5.1 Taskeigenschaften definieren

Für einen Task mit dem Namen MyTask wird eine Konfigurationsdatei namens „***`MyTask`*`.properties`**“ in dem erwähnten „Konfigurations-Verzeichnis“ erwartet.

Ist eine solche Datei nicht vorhanden, werden die System Properties zur Task-Konfiguration verwendet.

`config$imports`
: Diese Task Property definiert eine durch Komma oder Semikolon getrennte Liste von ImportDefinitionen.

`config$import.`*`ImportDefinition`*`.`
: Diese Task Property definiert eine weitere zu importierende Property Datei. Der Dateiname ist relativ zum Konfigurationsverzeichnis. ImportDefinition muss in der Task Property config$imports definiert sein. Der Eintrag ist optional. Voreinstellung ist ImportDefinition.

In den Beispielen wird `configuration.dir=config` verwendet.

5.2 Datenbankverbindung definieren

Die Datenbankverbindung wird immer mithilfe der oben beschriebenen Datenbank-Verbindungsparameter-Dateien hergestellt.

Üblicherweise geschieht dies dadurch, dass der Name der Datenbankverbindung beim Programmaufruf als (zweites) Argument übergeben wird, während das „Connection-Verzeichnis“, in dem sich die Dateien befinden gegebenenfalls über System Properties definiert wird.

Für eine Datenbankverbindung mit dem Namen ***MyConnection*** wird eine Datei mit den Datenbank-Verbindungsparametern namens „***MyConnection*.properties**“ in dem erwähnten „Connection-Verzeichnis“ erwartet.

connection.dir
: Diese System Property bestimmt das „Connection-Verzeichnis“. Kann eine relative oder absolute Pfadangabe enthalten. Eine relative Pfadangabe bezieht sich auf das mit connection.home angegebene Basisverzeichnis.
Voreinstellung ist ein leerer Eintrag, was bedeutet, dass die Dateien mit den Datenbank-Verbindungsparametern direkt in dem mit connection.home angegebenen Verzeichnis abgelegt sind.

connection.home
: Diese System Property bestimmt das Basisverzeichnis für das JDBCConnection-Verzeichnis. Kann eine relative oder absolute Pfadangabe enthalten.
Eine relative Pfadangabe bezieht sich auf das Basisverzeichnis (Current Dir) !
Voreinstellung ist das Benutzerverzeichnis[4].

connection.properties
: Diese System Property bestimmt den Dateinamen der ConnectionProperties-Datei wenn das Argument Connection beim Programmaufruf nicht angegeben ist. Kann nur eine relative Pfadangabe enthalten. Diese bezieht sich auf das „Connection-Verzeichnis“.

secret.file
: Diese System Property bestimmt den Dateinamen der Secret-Datei, die das Verschlüsselungspasswort für verschlüsselte Datenbankpasswörter enthält. Kann eine relative oder absolute Pfadangabe enthalten. Eine relative Pfadangabe bezieht sich auf das „Connection-Verzeichnis“.
Voreinstellung ist `secret.properties`.

In den Beispielen heißt die Connection `exampleconnection` und als Connection Verzeichnis wird `connection.dir=jdbcconnection` verwendet.

[4] Unter Windows wird das Benutzerverzeichnis aus den Umgebungsvariablen `%HOMEDRIVE%%HOMEPATH%` ermittelt, ansonsten aus der System Property `user.home`.

5.3 Quelldatei/Inputdatei definieren

Eine Quelldatei wird üblicherweise durch die Angabe beim Programmaufruf (als drittes Argument) als absoluter oder relativer Pfad definiert. Alternativ kann sie auch als Task Property in.file angegeben werden.

`in.dir`

Diese Property kann als System Property oder als Task Property definiert werden und bestimmt das Inputverzeichnis. Sie kann eine relative oder absolute Pfadangabe enthalten. Eine relative Pfadangabe bezieht sich auf das Basisverzeichnis (Current Dir). Die Angabe als System Property hat Vorrang vor der Angabe als Task Property.
Voreinstellung ist ein leerer Eintrag, was bedeutet, dass die Quelldateien im Basisverzeichnis (Current Dir) abgelegt sind.

`in.file`

Diese Task Property bestimmt den Dateinamen der Quelldatei. Sie kann eine relative oder absolute Pfadangabe enthalten. Eine relative Pfadangabe bezieht sich auf das mit in.dir angegebene Inputverzeichnis. Die Property wird ignoriert wenn die Inputdatei als Argument beim Programmaufruf übergeben wird.

Ist die Quelldatei als relativer Pfad angegeben, bezieht sich dieser auf das Inputverzeichnis, das gegebenenfalls als System Property oder Task Property in.dir definiert wird.

`in.encoding`

Diese Task Property bestimmt das Encoding der Inputdatei.
Der Inhalt muss der Name eines in Java verfügbaren Charactersets sein[5].
Voreinstellung ist UTF-8.

In den Beispielen:

```
in.dir=in
```

[5] https://docs.oracle.com/javase/8/docs/api/java/nio/charset/Charset.html.

5.4 Zielverzeichnis und Ausgabedatei definieren

Einzelne Ausgabedateien werden üblicherweise als Task-Property und immer relativ zum Zielverzeichnis definiert.

Der Ausgabedateiname kann **Ersetzungen** enthalten: In (einzelnen) eckigen Klammern angegebene Namen von Task Properties oder System Properties werden durch deren Inhalt ersetzt.

`out.dir`
Diese Property kann als System Property oder als Task Property definiert werden und bestimmt das Zielverzeichnis. Es kann eine relative oder absolute Pfadangabe enthalten. Eine relative Pfadangabe bezieht sich auf das Basisverzeichnis (Current Dir). Die Angabe als System Property hat Vorrang vor der Angabe als Task Property. Die Property wird ignoriert wenn das Zielverzeichnis als Argument beim Programmaufruf übergeben wird.
Voreinstellung ist ein leerer Eintrag, was bedeutet, dass die Ergebnisdateien im Basisverzeichnis (Current Dir) abgelegt werden.

`out.file`
Diese Task Property bestimmt den Dateinamen einer einzelnen Ausgabedatei.
Für **SQLAnalyser** ist die Voreinstellung ***`Task`*`.txt`**.

`out.encoding`
Diese Task Property bestimmt das Encoding der Ausgabedatei. Der Inhalt muss der Name eines in Java verfügbaren Charactersets sein[6].
Voreinstellung ist UTF-8.

Das Zielverzeichnis kann als Argument beim Programmaufruf, als System Property oder als Task Property definiert werden.

`out.tempdir`
Diese Task Property definiert ein temporäres Verzeichnis. Es kann eine relative oder absolute Pfadangabe enthalten. Eine relative Pfadangabe bezieht sich auf das Basisverzeichnis (Current Dir).
Ist ein temporäres Verzeichnis angegeben, wird die Ausgabedatei zunächst im temporären Verzeichnis erzeugt und erst nach erfolgreicher Vollendung in das Zielverzeichnis verschoben.
Temporäres Verzeichnis und Zielverzeichnis müssen sich im gleichen Laufwerk und in der gleichen Partition befinden.
Temporäre Verzeichnisse werden nur von **SQL2csv**, **SQL2file** und **SQL2xml** unterstützt. Per Voreinstellung wird kein temporäres Verzeichnis benutzt.

`out.zip.file`
Bestimmt den Dateinamen einer Zip-Datei, in der die Ausgabedateien gespeichert werden (experimentell).
Zip-Dateien werden nur von **SQL2csv**, **SQL2file** und **SQL2xml** unterstützt.

In den Beispielen:

```
out.dir=out
```

[6] https://docs.oracle.com/javase/8/docs/api/java/nio/charset/Charset.html.

5.5 CSV-Dateiformat definieren

Eine Definition des CSV-Formats findet sich in https://www.ietf.org/rfc/rfc4180.txt. Aus verschiedenen Gründen wird von der dortigen Spezifikation häufig abgewichen.

Prinzipiell besteht ein solches einfaches Text-Format für strukturierte Daten aus folgenden Elementen:

- Der Zeilenumbruch trennt die einzelnen Datensätze voneinander.
- Ein Separatorzeichen trennt die einzelnen Spalten voneinander.
- Textbegrenzungszeichen fassen einzelne Spaltenwerte optional ein. Dies bezeichnen wir als Quoting. Es ist dann erforderlich, wenn ein Spaltenwert das Separatorzeichen enthält.
- Mit einem Escapezeichen wird das rechte Textbegrenzungszeichen innerhalb eines eingefassten Spaltenwertes als Inhalt kenntlich gemacht. Das Escapezeichen selbst wird durch Verdopplung als Inhalt gekennzeichnet.

SQLTools unterstützen dabei folgende Varianten:

Separator:	Jegliche Zeichenfolge mit mindestens einem Zeichen
Textbegrenzer:	Jegliche Zeichenfolgen mit mindestens einem Zeichen, linke und rechte Begrenzung können unterschiedlich sein.
Escape:	Einzelnes Zeichen

SQLTools unterstützt keine Zeilenumbrüche in Spaltenwerten oder Strukturierungselementen. **SQL2csv** entfernt \r und \n aus Feldinhalten.

`csv.header`

Diese Task Property bestimmt, ob die erste Zeile einer csv-Datei Spaltennamen enthält.
Möglicher Inhalt:

`true`:	Erste Zeile der csv-Datei enthält Spaltennamen oder
`false`:	Erste Zeile der csv-Datei enthält Werte

In **SQL2csv** ist die Voreinstellung `true`.

`csv.rectangle`

Diese Task Property bestimmt, ob eine Zeile nach dem letzten nicht-leeren Wert mit Spaltenseparatoren für leere Felder aufgefüllt wird.
Möglicher Inhalt:

`true`:	Alle Zeilen haben die gleiche Anzahl an Spalten oder
`false`:	Zeilen enden nach dem letzten nicht-leeren Wert.

In **SQL2csv** ist die Voreinstellung `true`.

`csv.separator`

Diese Task Property bestimmt den Spaltenseparator einer CSV-Datei.
In **SQL2csv** ist die Voreinstellung ein Semikolon „;“.

csv.suppressEmptyLines

Diese Task Property bestimmt, ob eine Zeile geschrieben wird, die ausschließlich leere Felder enthält.
Möglicher Inhalt:
`true`: Leere Zeilen werden unterdrückt oder
`false`: leere Zeilen werden geschrieben.
In **SQL2csv** ist die Voreinstellung `true`.

csv.quoting

Diese Task Property steuert das Quoting in **SQL2csv** und kann folgende Ausprägungen haben:

NEVER Kein Quoting.
Separatorzeichen werden aus Feldinhalten entfernt.
SMART Quoting, wenn der Feldinhalt das Separatorzeichen enthält oder mit dem linken Textbegrenzer beginnt.
NONEMPTYSTRING Quoting für jedes nicht-leere STRING-Feld.
STRING Quoting für jedes STRING-Feld.
NONEMPTY Quoting für jedes nicht-leere Feld.
ALLWAYS Quoting für jedes Feld.

Die Voreinstellung ist **NEVER** wenn keine Textbegrenzer definiert sind, ansonsten **SMART**.
Im Falle von **STRING** und **NONEMPTYSTRING** werden die Inhalte von Nicht-String-Feldern nicht geändert, wenn sie das Separatorzeichen enthalten.

csv.quotes.left

Diese Task Property bestimmt den linken (optionalen) Textbegrenzer für Spalten einer CSV-Datei.

csv.quotes.right

Diese Task Property bestimmt den rechten (optionalen) Textbegrenzer für Spalten einer CSV-Datei.

csv.quote.header

Diese Task Property bestimmt, ob die (optionalen) Spaltennamen gequotet werden.
Möglicher Inhalt:
`true`: Leere Zeilen werden unterdrückt oder
`false`: leere Zeilen werden geschrieben.
In **SQL2csv** ist die Voreinstellung `true`., wenn `csv.quoting` **ALLWAYS** oder **NONEMPTY** ist, ansonsten `false`.

csv.escape

Diese Task Property bestimmt das Escape-Zeichen für von Textbegrenzern eingefasste Spaltenwerte einer CSV-Datei.

5.6 Zieltabelle definieren

out.table

Diese Property definiert den Namen der Zieltabelle.
Die Eingabe als System Property hat Vorrang vor der Eingabe als Task Property. Voreinstellung ist der Name des auszuführenden Tasks.

out.batchsize

Diese Task Property bestimmt die Batchsize bei Ausführung und commit des Input-Statements. Voreinstellung ist 1000.

out.case

Diese Property bestimmt die Case Sensitivity für Tabellen- und Spaltennamen. Gültige Werte sind **SENSITIVE** und **UPPER**.
Die Eingabe als System Property hat Vorrang vor der Eingabe als Task Property. Die Voreinstellung ist Datenbankabhängig.

5.7 SQL-Statements definieren

Die Definition eines SQL-Statements kann entweder direkt in einer **Task Property** oder in einer separaten **SQL-Datei** erfolgen.

Wird eine SQL-Datei verwendet, muss diese im **SQL-Verzeichnis** abgelegt werden, das durch die System Property sql.dir (s.u.) bestimmt wird. Voreinstellung für das SQL-Verzeichnis ist das Konfigurationsverzeichnis.

Die Angabe als Task Property sql.stmt hat Vorrang vor der Definition in einer separaten Datei. Der Dateiname wird über die Task Property sql.file angegeben. Sofern für die Datei ein anderes Encoding als UTF-8 verwendet wird, muss dieses mit der Task Property sql.encoding angeben werden.

Das SQL-Statement kann **Ersetzungen** enthalten: In doppelten eckigen Klammern angegebene Namen von Task Properties oder System Properties werden durch deren Inhalt ersetzt.

Für **SQL2xml** wird den Propertynamen der Elementbezeichner (**`root.`** bzw. **`element.`*`ElementDef`*`.`**) des jeweiligen XML-Elements vorangestellt.

`sql.dir`
: Diese System Property bestimmt das SQL-Verzeichnis. Sie kann eine relative oder absolute Pfadangabe enthalten. Eine relative Pfadangabe bezieht sich auf das Basisverzeichnis (Current Dir). Voreinstellung ist ein leerer Eintrag, was bedeutet, dass die SQL-Datei im Konfigurationsverzeichnis abgelegt ist.

`sql.encoding`
: Diese Task Property bestimmt das Encoding der SQL-Datei. Der Inhalt muss der Name eines in Java verfügbaren Charactersets sein[7]. Voreinstellung ist UTF-8.

`sql.file`
: Diese Task Property bestimmt die SQL-Datei. Die Pfadangabe ist relativ und bezieht sich auf das SQL-Verzeichnis. Diese Property wird nur ausgewertet, wenn das SQL-Statement nicht mit der Property sql.stmt angegeben wird.
Für **SQL2csv** ist die Voreinstellung **`Task.sql`**.

`sql.stmt`
: Diese Task Property bestimmt das SQL-Statement.

`sql.fetchsize`
: Mit dieser Task Property kann die FetchSize für ein Select-Statement gesetzt werden.[8] Der Inhalt muss eine nicht-negative Integer-Zahl sein. Als Voreinstellung bleibt die Default-FetchSize unverändert.

In den Beispielen:

```
sql.dir=sql
```

[7] https://docs.oracle.com/javase/8/docs/api/java/nio/charset/Charset.html.

[8] https://docs.oracle.com/javase/8/docs/api/java/sql/Statement.html.

5.8 Spalten einer CSV-Datei identifizieren und mappen

Spalten einer CSV-Datei können entweder anhand ihres Spaltennamens oder anhand ihrer Position (beginnend bei 1) identifiziert werden.

Soll die Identifizierung anhand des Namens erfolgen, muss allen relevanten Spalten ein Name zuordenbar sein. Wenn die CSV-Datei keine Spaltennamen enthält, kann dies auch durch entsprechende Task Properties geschehen.

`columns`
Diese Task Property bestimmt, ob die Spalten einer CSV-Datei per Namen oder Position definiert werden sollen. Gültige Werte sind „**name**“ und „**position**“.
Die Voreinstellung ist **name**, wenn die CSV-Datei Spaltennamen enthält (csv.header=true), ansonsten **position**.

`column.`*`CSVColumnName`*`.name`
Diese Task Property weist der über ihren Namen bestimmten Spalte in der CSV-Datei einen neuen Namen zu. Nur sinnvoll Im Falle von csv.header=true. Der Spaltenname aus der ersten Zeile der CSV-Datei kann damit korrigiert werden.

`column.`*`CSVColumnPosition`*`.name`
Diese Task Property weist der über ihre Position bestimmten Spalte in der CSV-Datei einen Namen zu. Im Falle von csv.header=true wird damit der Spaltenname aus der ersten Zeile der CSV-Datei anhand der Position überschrieben. Die Angabe hat Vorrang vor **`column.`*`CSVColumnName`*`.name.`**

`columns.mandatory`
Wenn der Inhalt dieser Task Property true ist, sind alle mit **`map.`*`DBColumnName`*** definierten Mappings per Voreinstellung obligatorisch.
Voreinstellung ist **false**.

`columns.map-auto`
Wenn der Inhalt dieser Task Property **true** ist und **`columns=name`**, werden als Voreinstellung alle gleichnamigen Spalten von CSV-Datei und Zieltabelle gemappt. Die Property `out.case` wird hierbei berücksichtigt. Wird ignoriert, wenn `columns=`**position**.

`map.`*`DBColumnName`*
Diese Task Property weist der über ihren Namen bestimmten Tabellenspalte in der Zieltabelle eine Spalte der CSV-Datei zu. Inhalt ist bei
columns=name, der Name der Spalte in der CSV-Datei und bei columns=position, die Position der Spalte in der CSV-Datei.

5.9 XML-Elemente definieren

SQL2xml erlaubt es, in Task Properties XML-Elemente frei zu definieren. Dabei wird die Elementbezeichnung der jeweiligen Task Property als Präfix vorangestellt. Die Elementbezeichnung für das root Element lautet **`root.`**, für andere Elemente **`element.`*`ElementDef`*`.`**.

Die Verwendung ist in den Konfigurationsdateien der Beispiele ausführlich beschrieben.

`xmlns`
: Kürzel des XML Namespaces. Optional. Wird als Attribut im Root-Element hinzugefügt und jedem ElementNamen vorangestellt.

`nsuri`
: URI des mit xmlns definierten Namespaces. Optional. Wenn kein Namespace mit xmlns angegeben wurde, definiert der Eintrag den Default Namespace.

`xml.encoding`
: Diese Task Property bestimmt das Attribut encoding in der XML Deklaration. Der Inhalt muss ein hierfür gültiger Characterset-Name sein. Voreinstellung ist das Encoding der Ausgabedatei.

***`ElementBezeichner`*`.name`**
: Name des XML Elements. Voreinstellung ist ***`ElementDef`*** bzw. **`root`**.

***`ElementBezeichner`*`.attributes`**
: Attribut-Liste statischer Attribute.

***`ElementBezeichner`*`.attribute.`*`AttributeName`**
: Inhalt des mit ***`AttributeName`*** bezeichneten statischen Attributs. Voreinstellung ist ein leerer Eintrag.

***`ElementBezeichner`*`.elements`**
: Statische Element-Liste.

***`ElementBezeichner`*`.sql.stmt`**
***`ElementBezeichner`*`.sql.encoding`**
***`ElementBezeichner`*`.sql.fetchsize`**
***`ElementBezeichner`*`.sql.file`**
: Definition des für dieses Element auszuführenden SQL-Statements. Optional.

***`ElementBezeichner`*`.sql.attributes`**
: Attribut-Liste der Spalten des SQL Statements die XML-Attribute bezeichnen.

***`ElementBezeichner`*`.sql.elements`**
: Element-Liste Statische Element-Liste.

***`ElementBezeichner`*`.sql.dyn-element`**
: Name des dynamischen Elements.

***`ElementBezeichner`*`.sql.key`**
: Liste der Key-Spalten des SQL-Statements.

***`ElementBezeichner`*`.sql.filter`**
: Filter-Liste der Spalten des vom Parent übergebenen SQL-Statements.

***`ElementBezeichner`*`.sql.filter.`*`FilterName`**
: Filter Wert der jeweiligen Spalte.

5.10 Spaltenformate definieren

5.10.1 Basis-Datentypen

SQLTools unterscheidet zwischen folgenden Basis-Datentypen:

DATE *(lokales[9]) Datum*
Java-Repräsentation: `java.time.LocalDate`
Standardformat: `yyyy-MM-dd`
sprintf-Voreinstellung: `%1$tF`

TIME *(lokale) Uhrzeit (Genauigkeit: Sekunden)*
Java-Repräsentation: `java.time.LocalTime`
Standardformat: `HH:mm:ss`
sprintf-Voreinstellung: `%1$tT`

DATETIME *(lokales) Datum mit Uhrzeit (Genauigkeit: Sekunden)*
Java-Repräsentation: `java.time.LocalDateTime`
Standardformat: `yyyy-MM-dd HH:mm:ss`
sprintf-Voreinstellung: `%1$tF %1$tT`

TIMESTAMP *(lokaler) Zeitstempel bestehend aus Datum, Uhrzeit und Sekundenbruchteilen*
Java-Repräsentation: `java.time.LocalDateTime`
Standardformat: `yyyy-MM-dd HH:mm:ss.SSSSSSSSS`
sprintf-Voreinstellung: `%1$tF %1$tT.%1$tN`

NUMBER *Dezimalzahl*
Java-Repräsentation: `java.math.BigDecimal`
Standardformat: `9.9`
sprintf-Voreinstellung: `%1$f`

INTEGER *Integerzahl*
Java-Repräsentation: `java.math.BigInteger`
Standardformat: `9`
sprintf-Voreinstellung: `%1$d`

STRING *Alphanumerische Zeichenkette*
Java-Repräsentation: `java.lang.String`
sprintf-Voreinstellung: `%1$s`

[9] SQLTools unterstützt derzeit keine Datums- und Zeitformate mit Zeitzonenangaben.

5.10.2 Erkennung von SQL-Datentypen

SQL-Datentypen werden standardmäßig über den java.sql.Type den Basis-Datentypen zugeordnet.

Diese Zuordnung kann mit der folgenden SQL Property bzw. den folgenden Task Properties angepasst werden:

`treat.`*`ColumnTypeName`*
: Diese SQL Property und Task Property ändert das Mapping des per ColumnTypeName[10] angegebenen SQL-Datentyps zu einem Basis-Datentyp.
Der Inhalt muss der Name eines der oben genannten Basis-Datentypen sein. Die Angabe als Task Property hat Vorrang vor der Angabe als SQL Property.

`treat.column.`*`ColumnName`*
: Diese Task Property ändert das Mapping der per ColumnName angegebenen SQL-Spalte zu einem Basis-Datentyp.
Der Inhalt muss der Name eines der oben genannten Basis-Datentypen sein.

5.10.3 Erkennung von CSV-Datentypen

Datentypen in CSV-Input-Dateien werden zusammen mit ihrem Format für jede einzelne Spalte spezifiziert.
Die Voreinstellung ist **STRING**.

Für unbekannte CSV-Dateien ist diese Zuordnung in aller Regel sehr mühsam und fehleranfällig. Hierzu kann der **CSVAnalyser** genutzt werden.

`column.`*`ColumnName`*`.format`
: Diese Task Property ändert das Mapping der per ColumnName angegebenen CSV-Spalte zu einer Format-Definition.
Der Inhalt muss der Name eines vordefinierten Formats sein.

`column.`*`ColumnPosition`*`.format`
: Diese Task Property ändert das Mapping der per ColumnPosition angegebenen CSV-Spalte zu einer Format-Definition.
Der Inhalt muss der Name eines vordefinierten Formats sein.

Zur Erweiterung der Formatdefinitionen kann die mitgelieferte Formatdefinitionsdatei angepasst werden (siehe unten).

[10] Rückgabewert der Java-Funktion `java.sql.ResultSetMetaData.getColumnTypeName`.

5.10.4 CSV-Ausgabeformat

Das CSV-Ausgabeformat wird über eine „sprintf-Formatangabe“ definiert.[11]

Für jeden Basis-Datentyp existiert hierzu eine Voreinstellung (siehe oben).

Dieses Format kann über die folgenden Task-Properties geändert werden:

`format.`*`BasisTypeName`*

Diese Task Property ändert das CSV-Ausgabeformat des per BasisTypeName angegebenen Basis-Datentyps.
Der Inhalt muss eine gültige „sprintf-Formatangabe“ sein. (Das mit %1 angegebene Argument enthält den Wert.) Alternativ kann der Wert „toString“ angegeben werden um die Standard-Java-Textrepräsentation zu verwenden. Für den Basis-Datentyp **NUMBER** ist ausserdem der Wert „significant“ zulässig, der alle signifikanten Vor- und Nachkommastellen ausgibt.

`column.`*`ColumnName`*

Diese Task Property ändert das CSV-Ausgabeformat der per ColumnName angegebenen SQL-Spalte.
Der Inhalt muss eine gültige „sprintf-Formatangabe“ sein. (Das mit %1 angegebene Argument enthält den Wert.) Alternativ kann der Wert „toString“ angegeben werden um die Standard-Java-Textrepräsentation zu verwenden. Für Spalten mit dem Basis-Datentyp **NUMBER** ist ausserdem der Wert „significant“ zulässig, der alle signifikanten Vor- und Nachkommastellen ausgibt.

---

[11] Siehe hierzu: https://docs.oracle.com/javase/8/docs/api/java/util/Formatter.html.

5.11 CSV-Formatdefinitionsdatei

Zum Parsen von CSV-Dateien benutzen SQLTools eine Formatdefinitionsdatei. Um die vordefinierten Formate zu erweitern oder anzupassen, kann die mitgelieferte Formatdefinitionsdatei geändert oder ersetzt werden.

Der Dateiname der Formatdefinitionsdatei ist CSVFormatDefinitions.txt, kann jedoch mittels der Task Property csv.formats geändert werden.

Die Formatdefinitionsdatei wird in folgender Reihenfolge gesucht:

- Angegebene Datei im Dateisystem.
- Wenn relative Pfadangabe: Datei im Classpath.
- Mitgelieferte Formatdefinitionsdatei[12]

Es wird nur die erste so gefundene Datei berücksichtigt.

`csv.formats`

Diese Task Property bestimmt den Dateinamen der CSV-Formatdefinitionsdatei. Sie kann eine relative oder absolute Pfadangabe enthalten. Eine relative Pfadangabe bezieht sich auf das mit configuration.dir angegebene Konfigurationsverzeichnis sowie den Java-Classpath.

Die Formatdefinitionsdatei ist eine Textdatei im UTF-8 Zeichensatz die folgende Einträge enthält[13]:

SEPARATOR Definition möglicher CSV-Spalten-Separatoren

`<WHITE*>`***Name***`<WHITE+>SEPARATOR`***Separator***

Name	*Eindeutiger Name.*
Separator	*Alle Zeichen bis zum Zeilenende.*

---

[12] Die mitgelieferten Konfigurationsdateien befinden sich im **`SQLTool.jar`** im Verzeichnis **`com/enitsys/sqltools`**.

[13]
- `<WHITE*>`: Beliebig viele Leerzeichen oder Tabulatorzeichen (optional).
- `<WHITE+>`: Ein oder mehrere Leerzeichen oder Tabulatorzeichen.
- `<TAB+>`: Ein oder mehrere Tabulatorzeichen.
- `[<TAB>…]`: Tabulator gefolgt von beliebigen weiteren Zeichen (optional).
- `[…]`: Beliebige weitere Zeichen (optional).

QUOTES Definition möglicher CSV-Textbegrenzungszeichen

```
<WHITE*>Name<WHITE+>QUOTES<TAB+>Links<TAB+>Rechts[<TAB>…]
```

Name	*Eindeutiger Name.*
Links	*Linke Textbegrenzung.*
Rechts	*Rechte Textbegrenzung.*

ESCAPE Definition möglicher Escape-Zeichen

```
<WHITE*>Name<WHITE+>ESCAPE<TAB+>Escape[…]
```

Name	*Eindeutiger Name.*
Escape	*Einzelnes Zeichen.*

Datums- und Zeit-Formate mit Angabe des Basis-Datentyps

```
<WHITE*>Name<WHITE+>DATE<TAB+>RegEx<TAB+>Locale<TAB+>Format[<TAB>…]
<WHITE*>Name<WHITE+>DATETIME<TAB+>RegEx<TAB+>Locale<TAB+>Format[<TAB>…]
<WHITE*>Name<WHITE+>TIME<TAB+>RegEx<TAB+>Locale<TAB+>Format[<TAB>…]
<WHITE*>Name<WHITE+>TIMESTAMP<TAB+>RegEx<TAB+>Locale<TAB+>Format[<TAB>…]
```

Name	*Eindeutiger Name.*
RegEx	*Regular Expression zur Identifikation des Formats.*
Locale	*Anzuwendende Locale[14] für den Format-String.*
Format	*DateTimeFormatter-Pattern[15].*

Numerische Formate

```
<WHITE*>Name<WHITE+>NUMBER<TAB+>RegEx<TAB+>Conversion[<TAB>…]
<WHITE*>Name<WHITE+>INTEGER<TAB+>RegEx<TAB+>Conversion[<TAB>…]
```

Name	*Eindeutiger Name.*
RegEx	*Regular Expression zur Identifikation des Formats.*
Conversion	*Anzuwendende Operationen[16]:*

c	*entferne alle Kommas*
d	*entferne alle Punkte*
w	*entferne alle Leerzeichen*
C	*ersetze Kommas durch Punkte*
0	*Setzt ein fehlendes 0 vor dem Punkt ein falls erforderlich*

[14] IETF BCP 47 Language Tag zur Übergabe an die Funktion `java.util.Locale.forLanguageTag`.

[15] Siehe https://docs.oracle.com/javase/8/docs/api/java/time/format/DateTimeFormatter.html.

[16] Die Operationen werden in der angegebenen Reihenfolge ausgeführt. Die verbleibende Zeichenfolge muss der String-Repräsentation eines `java.math.BigDecimal` entsprechen und durch den String Constructor dieser Klasse instanziierbar sein. Siehe auch https://docs.oracle.com/javase/8/docs/api/java/math/BigDecimal.html.

5.12 Datenbankabhängige Konfigurationsdateien

Trotz aller Standardisierung verlangen die Datenbanken unterschiedlicher Hersteller manchmal eine leicht unterschiedliche Syntax bzw. Konfiguration. Hierzu werden spezielle Dateien im Konfigurationsverzeichnis oder im Classpath hinterlegt. Die Suchreihenfolge ist jeweils:

- Datei ***typ***[17]**.*RDBMS*[18]**.properties** im Konfigurationsverzeichnis
- Datei ***typ*****.*RDBMS*****.properties** im Classpath
- Mitgelieferte Datei[19] ***typ*****.*RDBMS*****.properties**
- Datei ***typ*****.ansi.properties** im Classpath
- Mitgelieferte Datei ***typ*****.ansi.properties**

Es wird nur die erste so gefundene Datei berücksichtigt.

5.12.1 SQL Properties

SQL Properties unterstützen ausschließlich die Zuordnung von SQL Datentypen zu Basis Datentypen[20].

5.12.2 DML Properties

DML Properties werden von **CSV2sql** für die Erstellung der insert Statements verwendet. Die Syntax ist in den mitgelieferten Konfigurationsdateien erläutert.

5.12.3 DDL Properties

DDL Properties werden von **CSVAnalyser** für die Erstellung der create table Statements verwendet. Die Syntax ist in den mitgelieferten Konfigurationsdateien erläutert.

5.12.4 Script Properties

Script Properties werden von **SQLcmd** für das Parsen multipler SQL Statements in einer Skript-Datei verwendet. Die Syntax ist in den mitgelieferten Konfigurationsdateien erläutert.

5.12.5 Produkterkennung

Zur Erkennung des RDBMS für die datenbankabhängige Konfiguration verwendet **SQLTools** den vom JDBC-Treiber zurückgelieferten Datenbankproduktname[21]. Dieser kann gegebenfalls über die Datei Productnames.txt auf den gewünschten db.productname geändert werden.

Die Datei wird in folgender Reihenfolge gesucht:

- Im Konfigurationsverzeichnis.
- Im Classpath.
- Mitgelieferte Formatdefinitionsdatei.

Es wird nur die erste so gefundene Datei berücksichtigt.

Das Format ist in der mitgelieferten Datei beschrieben.

---

[17] Je nach Art der Konfigurationsdatei ist dies das Literal **sql**, **dml, ddl** oder **script**.

[18] Siehe hierzu Beschreibung Datenbank Eigenschaft **db.productname**.

[19] Die mitgelieferten Konfigurationsdateien befinden sich im **SQLTools.jar** im Verzeichnis **com/enitsys/sqltools**.

[20] Siehe **treat.*ColumnTypeName***

[21] Rückgabewert der Java-Funktion `java.sql.DatabaseMetaData.getDatabaseProductName()`.

5.13 Sonstige Konfigurationen und Eigenschaften

5.13.1 Datenbank Eigenschaften

`db.productname`

Im **CSVAnalyser** definiert diese Task Property das Ziel-RDBMS für die DDL-Erstellung. Inhalt ist der Produktname[22]. Voreinstellung ist „Oracle“. In den anderen Tools wird diese Property automatisch bei der Verbindungserstellung gesetzt und steht als Ersetzung zur Verfügung. Sie wird ausserdem benutzt um die RDBMS-spezifischen DML- und SQL-Properties zu laden.

`db.productversion`

Diese Property enthält die Produktversion der Datenbank[23], wird automatisch bei der Verbindungserstellung gesetzt und steht als Ersetzung zur Verfügung.

5.13.2 Logging Konfiguration

`logging.format`

Diese System Property kann auf den Wert simple gesetzt werden, um eine übersichtliche Log-Ausgabe zu definieren. Dies ist auch die Voreinstellung. Jeder andere Inhalt führt dazu, dass das Logging im ausführlicheren Standardformat erfolgt.[24]

`logging.level`

Diese Property setzt das Logging Level für alle Logger und Handler auf das angegebene Level. Der Inhalt muss ein gültiges Logging Level sein[25]. Per Voreinstellung ist dies normalerweise INFO. **SQLTools** benutzen die Level CONFIG und FINE für noch detailliertere Ausgaben. Mit WARNING, SEVERE und OFF kann die Log-Ausgabe weiter reduziert werden.

5.13.3 Sonstige Eigenschaften

`sqltools.classname`

Diese Property enthält den Java Classname des aufgerufenen **SQLTools**, wird automatisch gesetzt und steht als Ersetzung zur Verfügung.

`sqltools.version`

Diese Property enthält die SQL Tools Version, wird automatisch gesetzt und steht als Ersetzung zur Verfügung.

`sqltools.today`

Diese Property enthält das aktuelle Datum im Format yyyyMMdd, wird automatisch gesetzt und steht als Ersetzung zur Verfügung.

`sqltools.now`

Diese Property enthält das aktuelle Datum und Uhrzeit im Format yyyyMMddHHmmss, wird automatisch gesetzt und steht als Ersetzung zur Verfügung.

---

[22] Siehe Abschnitt Produkterkennung.

[23] Rückgabewert der Java-Funktion `java.sql.DatabaseMetaData.get getDatabaseProductVersion()`.

[24] Wenn die System Property `java.util.logging.SimpleFormatter.format` genutzt werden soll um ein eigenes LoggingFormat zu definieren muss `logging.format` überschrieben werden da diese Property ansonsten wieder überschrieben würde.

[25] Siehe hierzu https://docs.oracle.com/javase/8/docs/api/java/util/logging/Level.html .

6 Lizenz und Haftungsausschluss

SQLTools werden Ihnen von enitsys kostenlos unter der Apache License, Version 2.0 zur Verfügung gestellt.

Diese Lizenz ist als Open Source Lizenz von der OSI zertifiziert und unter http://www.apache.org/licenses/LICENSE-2.0 einsehbar.

Die Software wird auf einer „AS IS“-Basis ohne Garantien oder Gewährleistungen jeglicher Art, weder ausdrücklich noch stillschweigend, zur Verfügung gestellt.

Eigene Werke, die auf SQLTools basieren, müssen

- eine Kopie der in der Textdatei LICENSE enthaltenen Apache-Lizenz beinhalten,
- bei modifizierten Dateien an auffälliger Stelle angeben, dass sie modifiziert sind,
- in der Quellform alle Original-Urheberrechtsvermerke beibehalten sowie
- die in der Textdatei NOTICE enthaltenen Urhebervermerke auf eine in der Lizenz genauer vorgeschriebene Art und Weise enthalten.

Mit Ausnahme vorsätzlich verursachter Schäden und der Haftung bei Körperschäden haftet der Produktanbieter für keinerlei Schäden die durch die Verwendung des Produkts oder die Unmöglichkeit der Verwendung verursacht worden sind.

Dies gilt ohne Ausnahme auch für entgangenen Geschäftsgewinn, Betriebsunterbrechungen, entgangene Geschäftsinformationen oder anderen wirtschaftlichen Verlust, auch wenn der Produktanbieter vorher auf die Möglichkeit eines solchen Schadens hingewiesen wurde.

Er haftet insbesondere auch nicht für Verletzungen der Datenschutz- und Geheimhaltungspflichten aus einer Weitergabe oder ungesicherten Aufbewahrung des Produkts oder mit ihm hergestellter Daten und Dokumente.

In den Beispielprogrammen werden Java Bibliotheken Dritter verwendet. Diese sind in der „normalen“ SQLTools Distribution nicht enthalten. In einer erweiterten Distribution erfolgt deren Weitergabe - in Übereinstimmung mit den jeweiligen Lizenzen - ausdrücklich nur zu den genannten Demonstrationszwecken. Eine darüber hinausgehende Benutzung oder Weitergabe ist im Rahmen dieser Lizenz nicht gestattet.