

# Elixir Universe

Release 3.5.0



*Elixir Technology Pte Ltd*

---

# **Elixir Universe: Release 3.5.0**

*Elixir* Technology Pte Ltd

Published 2014

Copyright © 2014 Elixir Technology Pte Ltd

All rights reserved.

Java is a trademark or registered trademark of Oracle Corporation in the United States and other countries. Microsoft and Windows are trademarks of Microsoft Corporation.

---

---

# Table of Contents

1. Introduction .....	1
What is a Universe? .....	1
2. Universe Types .....	2
JDBC Universe .....	2
Repository Universe .....	2
Stave .....	2
3. Creating a Universe .....	3
Before You Begin .....	3
Creating a JDBC Universe .....	3
Creating a Repository Universe .....	5
Creating a Stave Universe .....	5
4. Universe Designer Walkthrough .....	8
Universe Designer .....	8
Examining Table Structure .....	8
Setting Attributes .....	9
Transforming Data .....	10
Examining Output Data .....	11
Inspecting Column Data .....	11
Viewing the Summary of Data .....	12
Exporting the Output Data .....	13
5. Using the Universe Designer .....	14
Repository Universe .....	14
Adding a New Table .....	14
Importing a Data Source .....	16
Settings Tab .....	17
Rows .....	24
Inspector .....	24
Summary .....	25
Exporting the Universe .....	26
Setting the Universe Access Permissions .....	28
JDBC Repository .....	29
Configuring a Connection Pool .....	30
Adding Tables .....	31
Pasting Tables .....	31
Viewing SQL .....	31
Other Actions .....	31
Stave Repository .....	31
Mapping Stave Columns to Logical Tables .....	32
Operations on Stave .....	32

---

## List of Figures

3.1. Menu to Add a Universe .....	4
3.2. Adding a JDBC Universe .....	4
3.3. Adding a Repository Universe .....	5
3.4. Adding a Stave Universe .....	6
3.5. Selecting a Stave Table .....	6
4.1. Universe Example .....	8
4.2. Examining Table Structure .....	9
4.3. Setting Attributes .....	10
4.4. Adding Transforms .....	11
4.5. Examining Output Data .....	11
4.6. Inspecting Column Data .....	12
4.7. Viewing the Data Summary .....	13
5.1. Universe Designer .....	14
5.2. Selecting a Data Source - Repository Universe .....	15
5.3. Adding a Column .....	15
5.4. Adding a Column Value .....	16
5.5. Importing a Data Source .....	17
5.6. Viewing a Schema .....	18
5.7. Setting Table Attributes .....	19
5.8. Setting Column Attributes .....	20
5.9. Setting Column Security .....	22
5.10. Rows Tab .....	24
5.11. Inspector Tab .....	25
5.12. Summary Tab .....	26
5.13. Exporting the Universe .....	27
5.14. Mapping the Universe .....	28
5.15. Setting the Universe Access Permissions .....	29
5.16. Universe Designer for a JDBC Universe .....	30
5.17. Configuring the Connection Pool .....	30
5.18. Universe Designer for a Stave Universe .....	31
5.19. Table Mapping for a Stave Universe .....	32
5.20. Mapped Table for a Stave Universe .....	33

---

# Chapter 1

## Introduction

---

---

### What is a Universe?

A universe is a collection of related data tables which provides access to all information needed to construct an ad-hoc report or dashboard.

In cases where the system knows the primary key and foreign key relationships between the tables, it automatically joins data from various tables and presents you the information seamlessly.

Logically, each Universe should represent a single domain. For example, in a vehicle store, there could be a Universe named Cars which may contain tables for model details, sales and service details. Similarly, there could be another Universe named Bikes containing all bike related information.

Using the above example, you can pull in statistics such as the number of cars sold versus the number of bikes sold, how reliable each car was, popularity for a particular vintage model of a car versus the same model of a bike and so on.

This manual describes the different kinds of universes and how to create and use them effectively.

---

# Chapter 2

## Universe Types

---

---

There are three types of universes that you can create. This chapter explains these types.

### JDBC Universe

The JDBC Universe allows data to be read from a relational database using a JDBC driver.

You can access your data using SQL statements. If you have all your data in a relational database, and a JDBC driver is available for your database, then use a JDBC Universe to access your data.

### Repository Universe

You can use any Elixir data source to read your data when using this Universe.

You can access data from a large number of data sources such as JDBC, Excel, XML and flat files.

You can mix data from any supported format. You can either embed the data sources in the Universe or maintain them as external links.

### Stave

The Stave universe allows data to be read from a column based Stave datastore.

Every column in a Stave table is already sorted and grouped. This enables lightning fast data retrieval, and facilitates better data compression.

Each column in Stave has the following characteristics:

- An attribute - one of: Boolean, Decimal, Double, Long or String.
- An encoder/decoder.
- A compressor/decompressor (gzip is the default).
- Attributes
- The row index acts as the Primary Key.

---

# Chapter 3

## Creating a Universe

---

---

### Before You Begin

Note the following points, when creating any Universe.

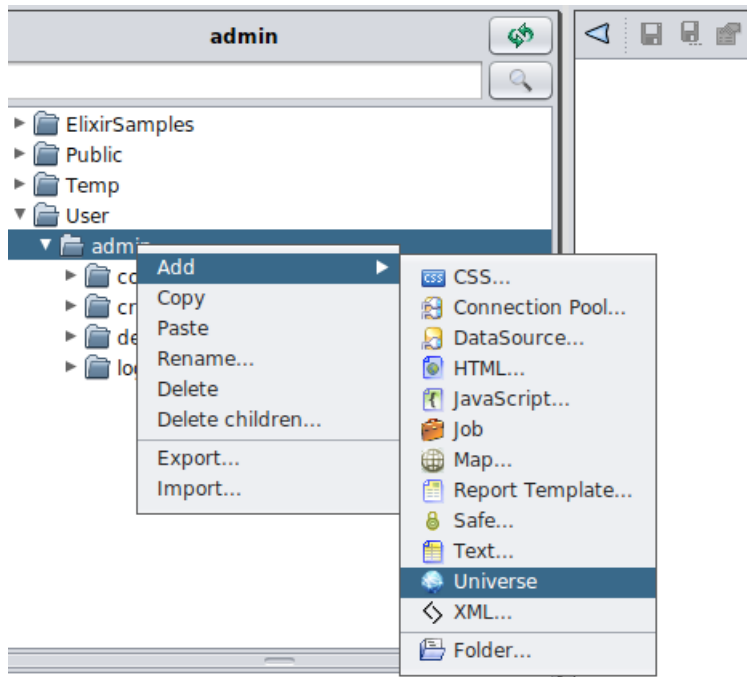
- Create a Universe file based on the origins and stability of the data. For example, you can create a Stave Universe if the data is stable and rarely changes. If your data comes from a JDBC database, use a JDBC Universe. Or, if you need to fetch data from multiple kinds of data sources, use a Repository Universe.
- If you want your Universe to be usable by others, do not save it in your user folder, instead save it in a publicly accessible folder such as `/Public`.
- Review the data from the Universe Designer by viewing it in the Rows tab, as described in [the section called “Rows”](#)
- Map the Universe file so that it can be referenced by other tools. This can only be done by an administrator. An example of Mapping is shown in [the section called “Exporting the Universe”](#).
- Assign access rights to the Universe, as desired. This is shown in [the section called “Setting the Universe Access Permissions”](#).
- Finally, save the Universe as a Published Template for Ad-Hoc Dashboard and Ad-Hoc Report. This is described in [the section called “Exporting the Universe”](#).

### Creating a JDBC Universe

To create a JDBC Universe:

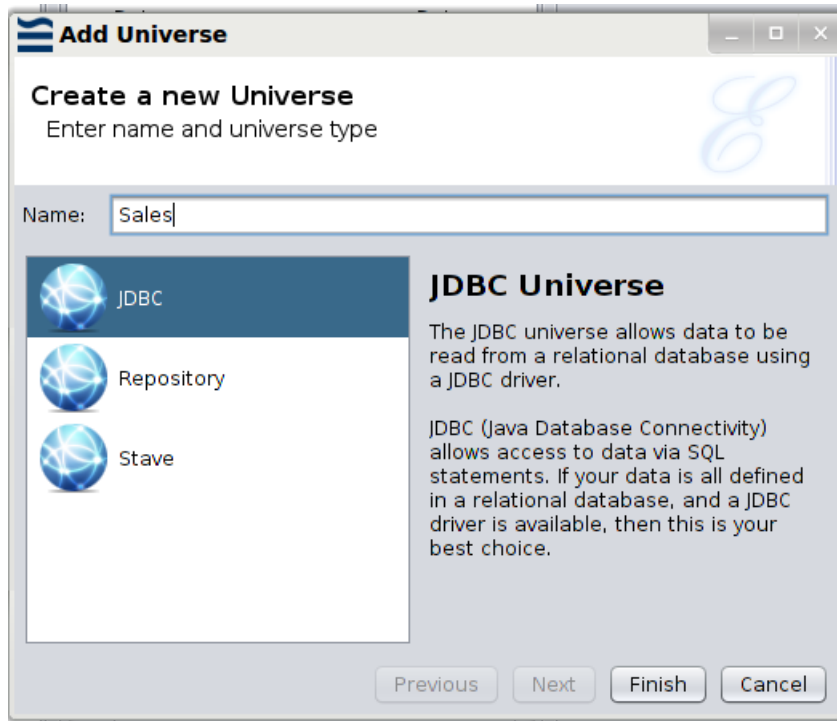
1. Login to the Ambience Designer.
2. Right click the location where you want to add the Universe. Select Add - Universe as shown in [Figure 3.1, “Menu to Add a Universe”](#):

**Figure 3.1. Menu to Add a Universe**



3. Enter a name for the Universe and select the type as JDBC, as shown in Figure 3.2, “Adding a JDBC Universe”:

**Figure 3.2. Adding a JDBC Universe**



4. Click **Finish**. The universe is then created and opened in the Universe Designer.

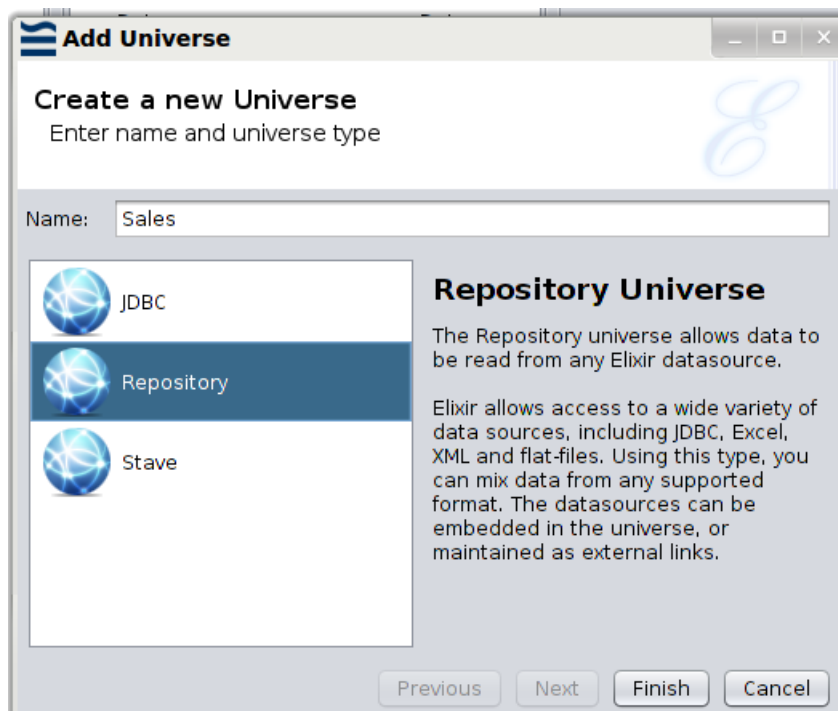


## Creating a Repository Universe

To create a Repository Universe:

1. Login to the Ambience Designer.
2. Right click the location where you want to add the Universe. Select Add - Universe as shown in Figure 3.1, “Menu to Add a Universe”.
3. Enter a name for the Universe and select the type as Repository, as shown in Figure 3.3, “Adding a Repository Universe”:

**Figure 3.3. Adding a Repository Universe**



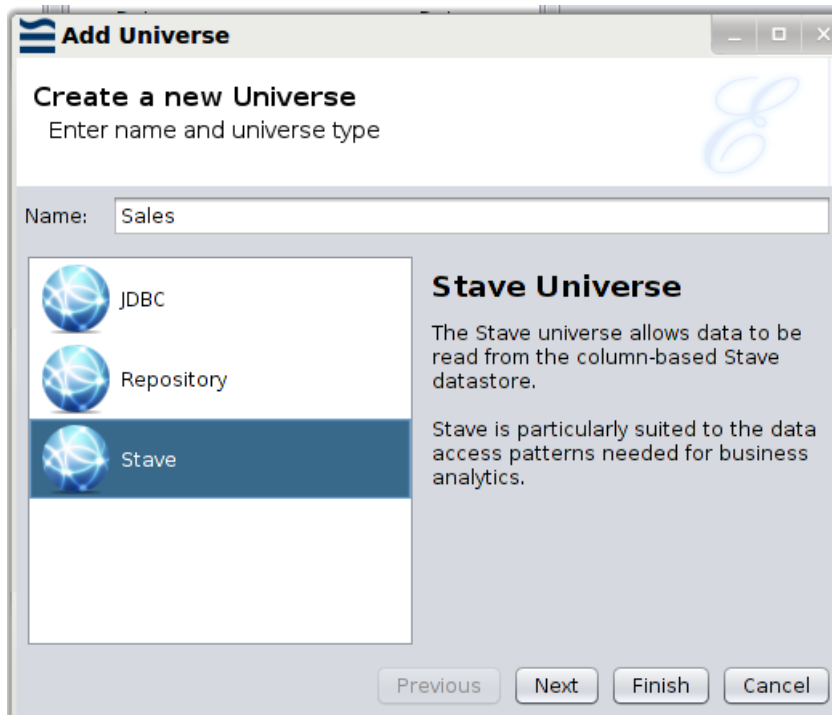
4. Click **Finish**. The universe is then created and opened in the Universe Designer.

## Creating a Stave Universe

To create a Stave Universe:

1. Login to the Ambience Designer.
2. Right click the location where you want to add the Universe. Select Add - Universe as shown in Figure 3.1, “Menu to Add a Universe”.
3. Enter a name for the Universe and select the type as Stave, as shown in Figure 3.4, “Adding a Stave Universe”:

**Figure 3.4. Adding a Stave Universe**



4. Click **Next**. Select the Stave table that contains the data for the universe as shown in [Figure 3.5, “Selecting a Stave Table”](#).

**Figure 3.5. Selecting a Stave Table**



See the Data Designer manual to know how to create a Stave table.

5. Click **Finish**. The universe is then created and opened in the Universe Designer.

---

# Chapter 4

## Universe Designer Walkthrough

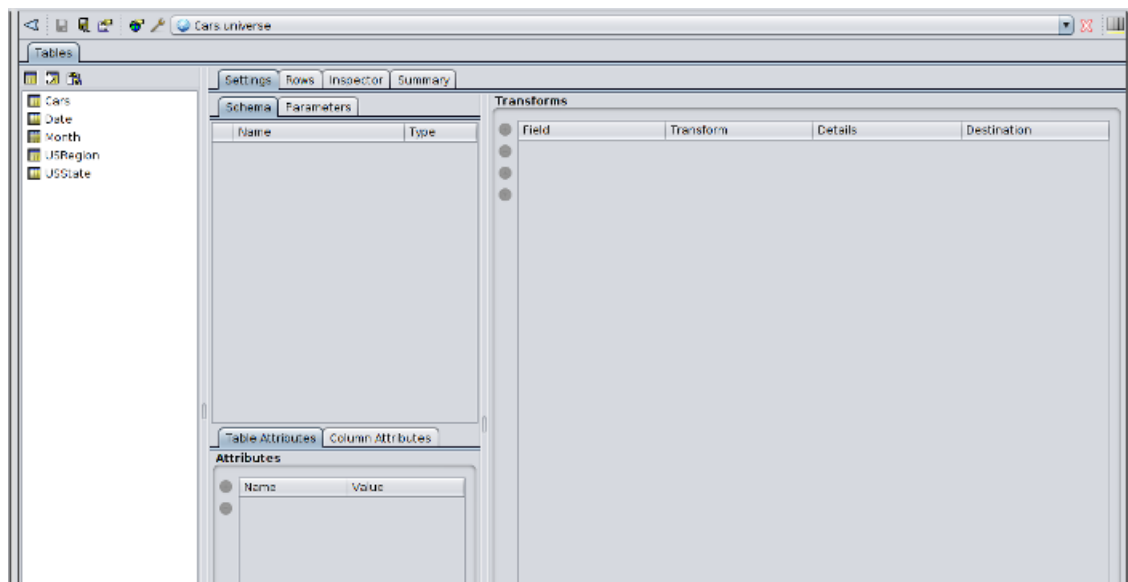
---

This chapter illustrates how a Universe looks in the Universe designer, and the various functionality it supports. You can easily create your Universe in a full graphical setting and export it for use.

### Universe Designer

An example of the Cars Universe that is shipped with Ambience, is shown in [Figure 4.1, “Universe Example”](#):

**Figure 4.1. Universe Example**



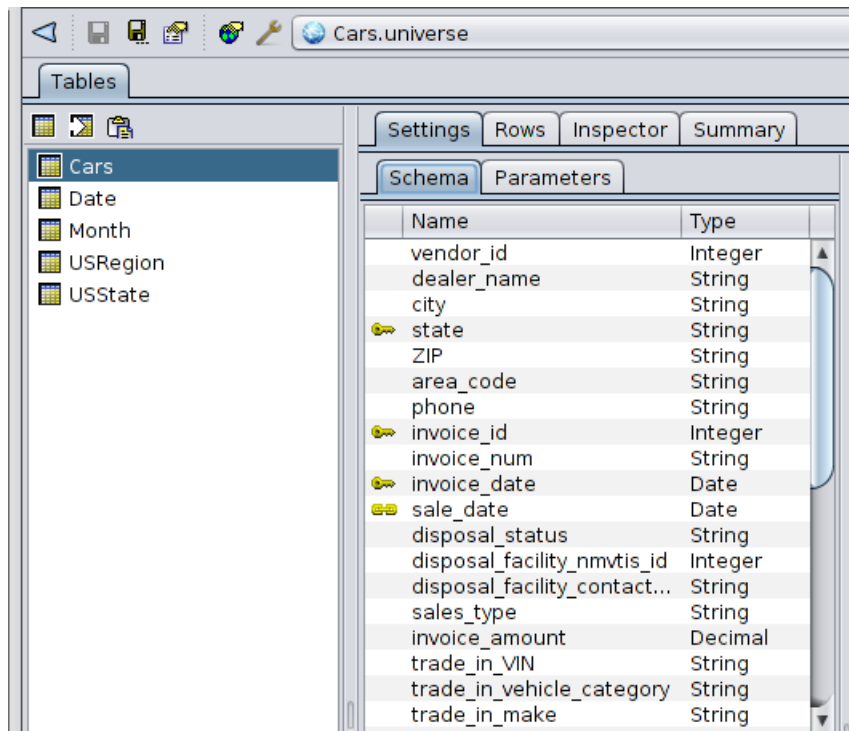
Once you create a Universe, the relevant tables are present on the left pane.

You can examine the contents of each table, inspect the various columns, manipulate data using Transforms, set various data attributes and finally export the Universe for use by Ad-Hoc Dashboard and Ad-Hoc Report.

### Examining Table Structure

You can view the columns that are in a table in the Schema tab.

An example is shown in [Figure 4.2, “Examining Table Structure”](#):

**Figure 4.2. Examining Table Structure**

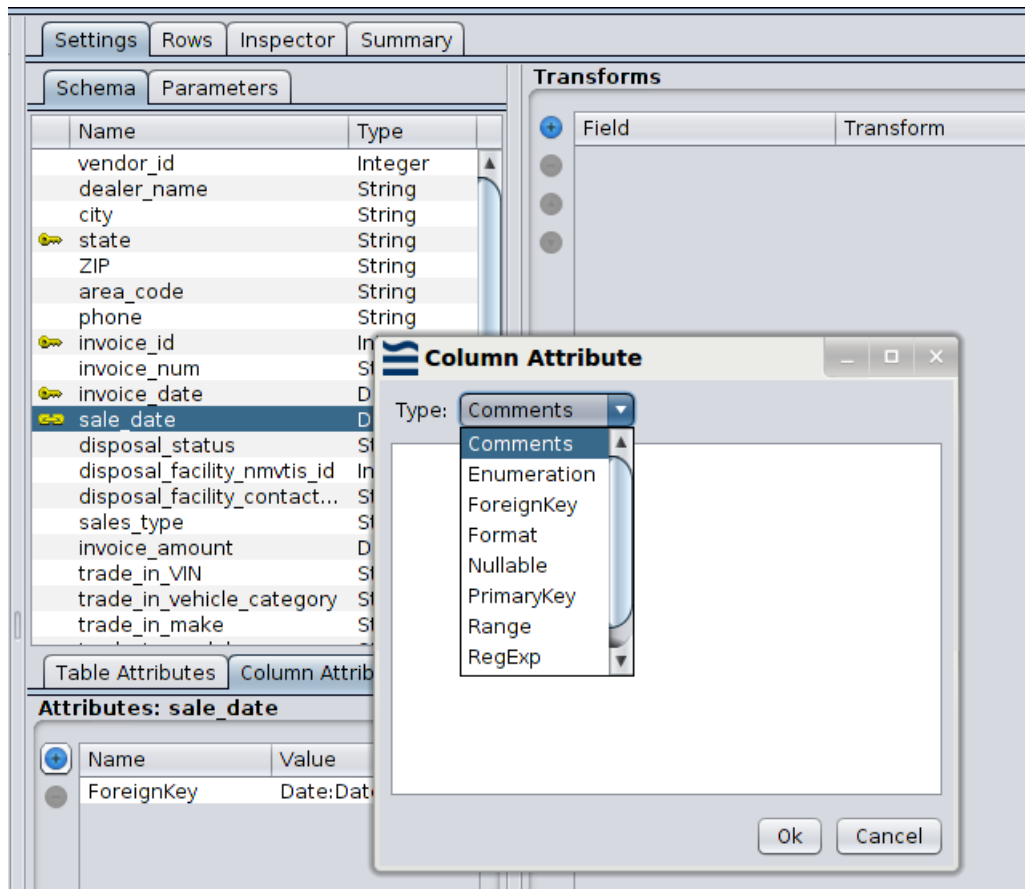
The data parameters if any, are displayed in the `Parameter` tab.

## Setting Attributes

You can set various attributes such as table security, primary and foreign keys and the format of the data.

You can set attributes that are common to a table as a whole, and also individual column attributes.

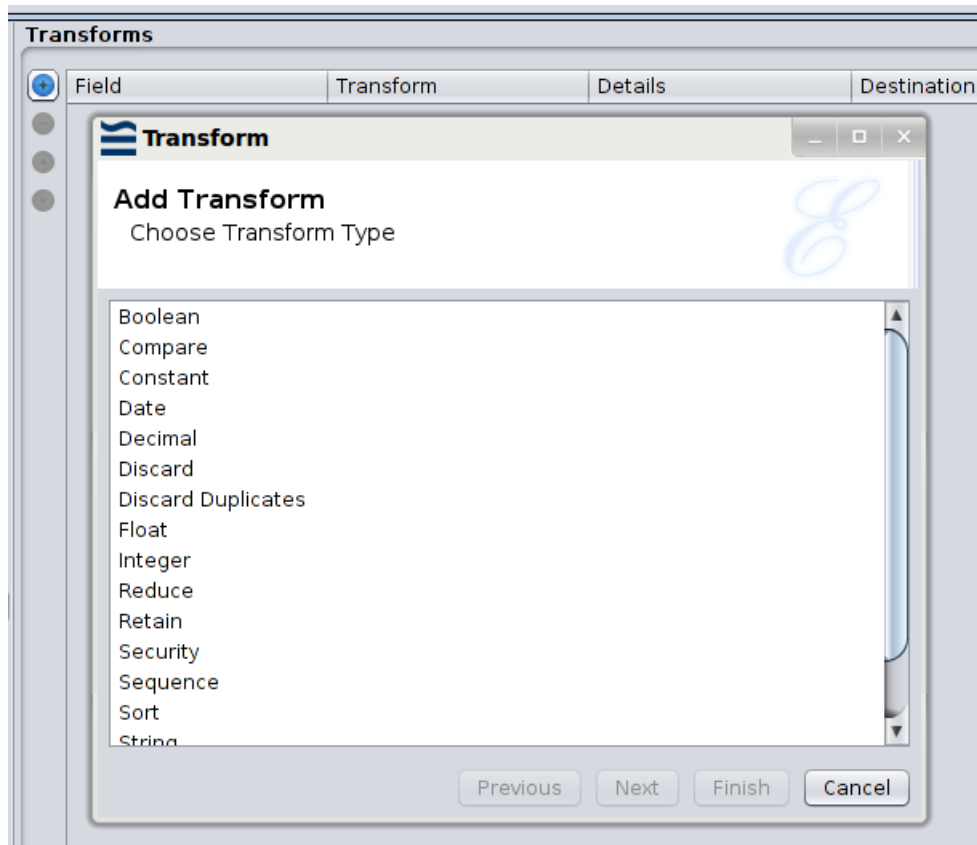
An example of setting attributes is show in [Figure 4.3, “Setting Attributes”](#)

**Figure 4.3. Setting Attributes**

## Transforming Data

You can add various transforms to manipulate the output as desired. For an in-depth explanation of transforms, see the [Elixir Transform User Manual](#).

An example of adding transforms is shown in [Figure 4.4, “Adding Transforms”](#)

**Figure 4.4. Adding Transforms**

## Examining Output Data

The Rows tab displays the data that will be output. This output data includes entities that have been transformed by any transforms you have added.

An example of the data in Rows is show in [Figure 4.5, “Examining Output Data”](#)

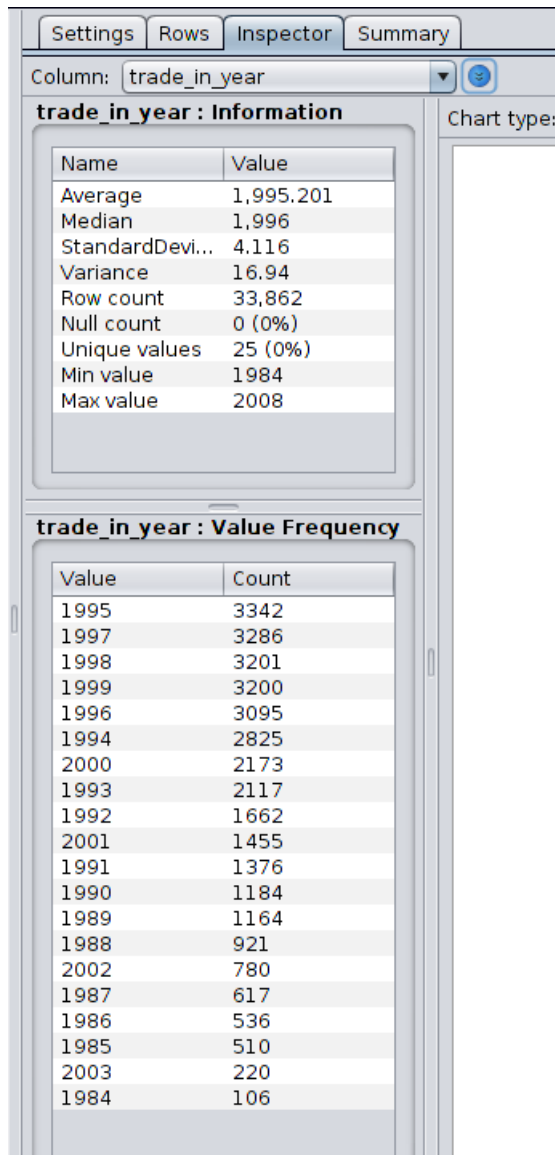
**Figure 4.5. Examining Output Data**

vendor_id (Integer)	dealer_name (String)	city (String)	state (String)
1	HUDIBURG MOTORS, INC.	Midwest City	OK
1	HUDIBURG MOTORS, INC.	Midwest City	OK
1	HUDIBURG MOTORS, INC.	Midwest City	OK
1	HUDIBURG MOTORS, INC.	Midwest City	OK
1	HUDIBURG MOTORS, INC.	Midwest City	OK
1	HUDIBURG MOTORS, INC.	Midwest City	OK
1	HUDIBURG MOTORS, INC.	Midwest City	OK
1	HUDIBURG MOTORS, INC.	Midwest City	OK

## Inspecting Column Data

The Inspector tab enables you to review the contents of a column. It displays column information, value frequency based on the data type, as well as the chart type and plot.

An example of the data in the Inspector tab is show in [Figure 4.6, “Inspecting Column Data”](#)

**Figure 4.6. Inspecting Column Data**

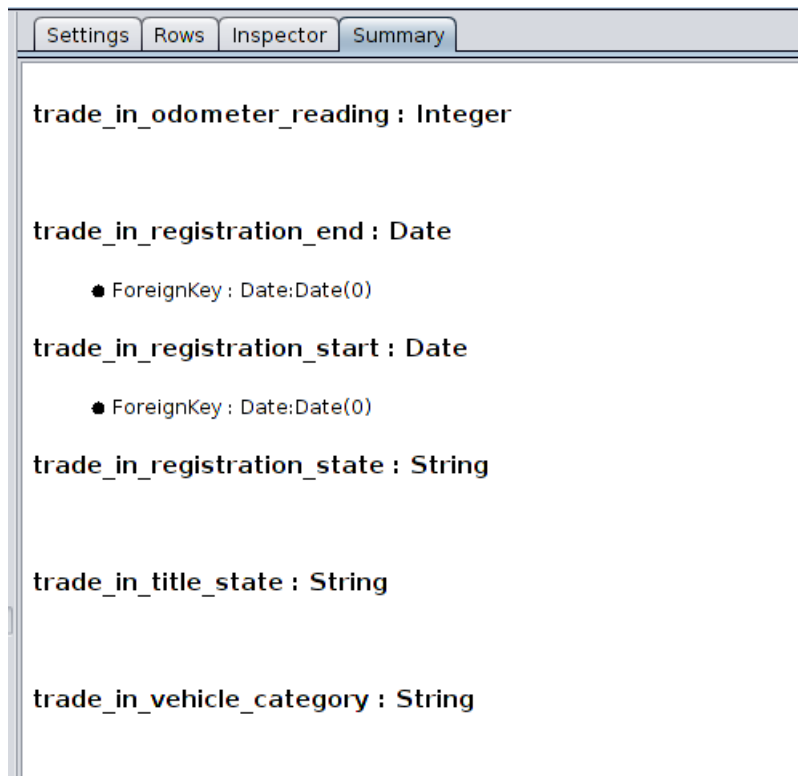
The example displays statistical data of the entities in the selected column, and the number of cars that were traded in for each year.

## Viewing the Summary of Data

The **Summary** tab displays the data type of each column, along with the primary and foreign key relationships between the columns.

An example of the data in the **Summary** tab is show in [Figure 4.7, “Viewing the Data Summary”](#)



**Figure 4.7. Viewing the Data Summary**

## Exporting the Output Data

You can easily export the output data for use by the Ad-Hoc Dashboard and the Ad-Hoc Report applications. Refer to [the section called “Exporting the Universe”](#) for the details.

---

# Chapter 5

## Using the Universe Designer

---

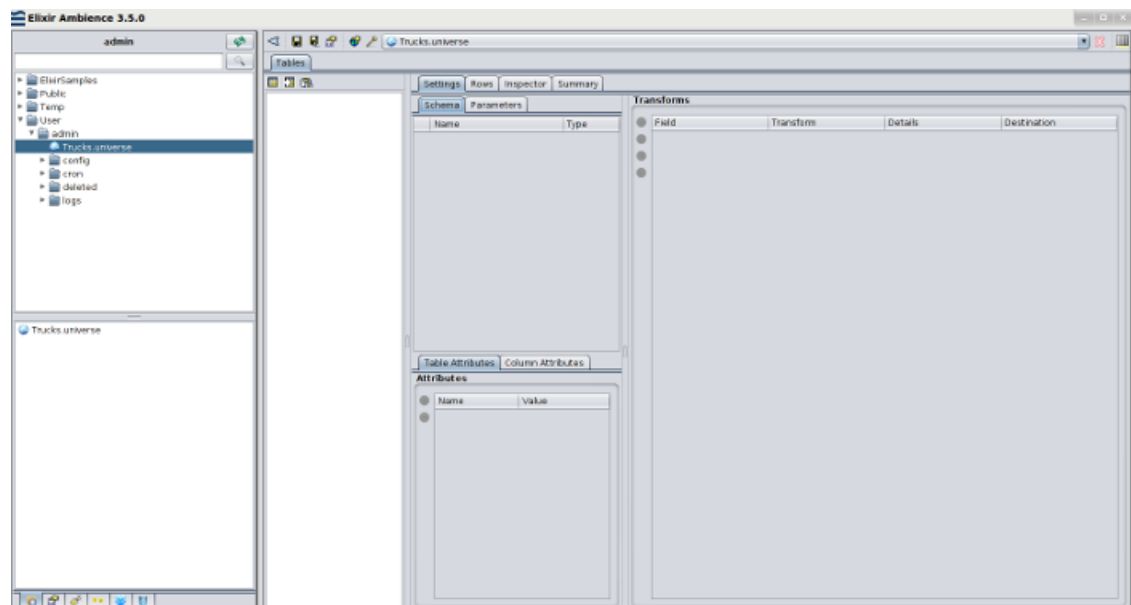
The Universe Designer allows you to create or import data sources into the universe, examine the rows in a table, set various table and column attributes, select the data to be transformed or displayed and create various transforms to transform the data to your desired results.

### Repository Universe

Double click a repository universe from the folder tree to open it in the Universe Designer.

The Universe Designer displays as shown in [Figure 5.1, “Universe Designer”](#):

**Figure 5.1. Universe Designer**




### Adding a New Table

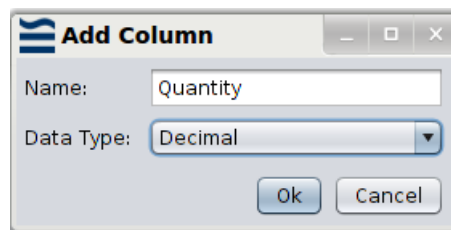
A Repository Universe allows you to select data from various table columns across multiple data sources. If the system deduces the relationships between the various columns and tables, based on the primary and foreign keys, it seamlessly joins the data and outputs the results.


To add a new table column to the Universe:

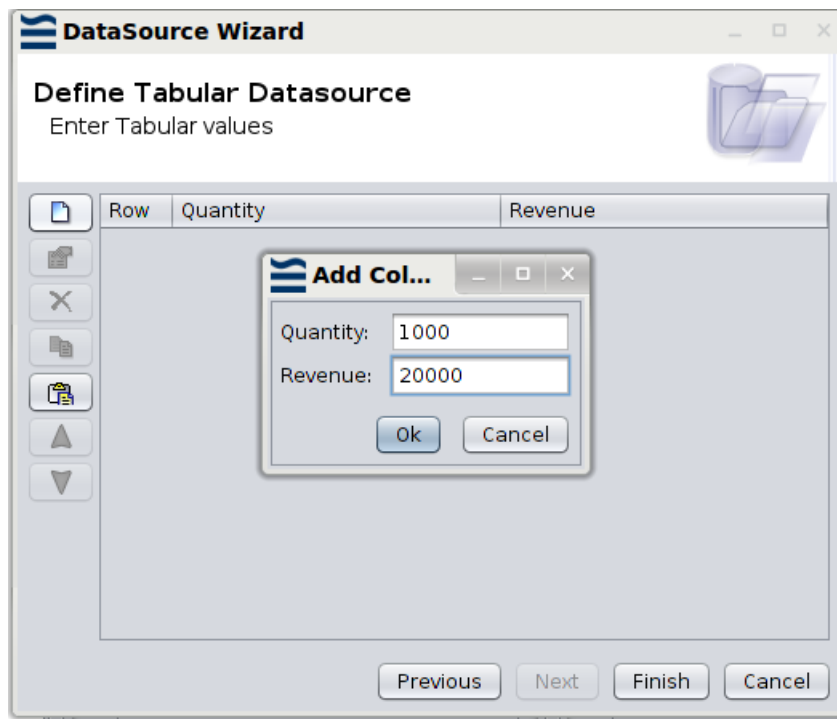
1. Click the Add Table icon (📄).
2. Select the data source for which you want to add a table. Repository universe supports a large number of data sources. In the example shown in [Figure 5.2, “Selecting a Data Source - Repository Universe”](#) we select a Tabular data source.:

**Figure 5.2. Selecting a Data Source - Repository Universe**

3. Enter a Name and a Description for the data source.
4. Click the Add icon (.
5. Enter a name for the column and select its data type as shown in [Figure 5.3, “Adding a Column”](#):

**Figure 5.3. Adding a Column**

6. Click **OK** to save the column.
7. Click the Add icon again, and repeat the process to add as many columns as you need.
8. Click **Next**.
9. You need to enter the values for the columns you have saved. Click the Add icon (.
10. Enter the values for the columns you have saved, as shown in [Figure 5.4, “Adding a Column Value”](#):

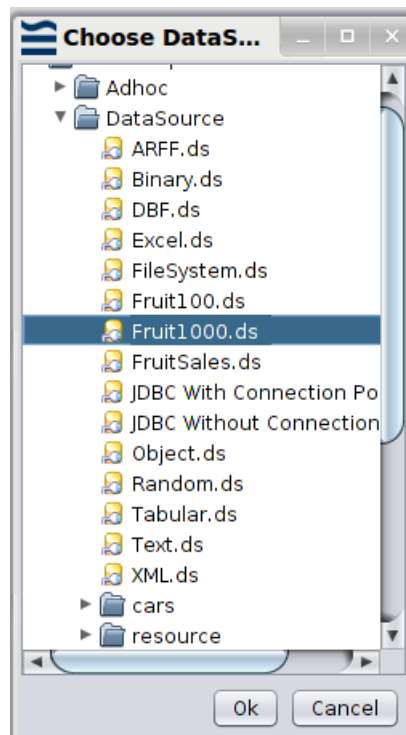
**Figure 5.4. Adding a Column Value**

11. Click **OK** to save the column.
12. Click the Add icon again and repeat the process to add as many values as you need.
13. Click **Finish** to add the data source to the Universe.

## Importing a Data Source

To import an existing data source into the Universe:

1. Click the Import data source icon (📄).
2. Select the data source to be imported, as shown in Figure 5.5, "Importing a Data Source":

**Figure 5.5. Importing a Data Source**

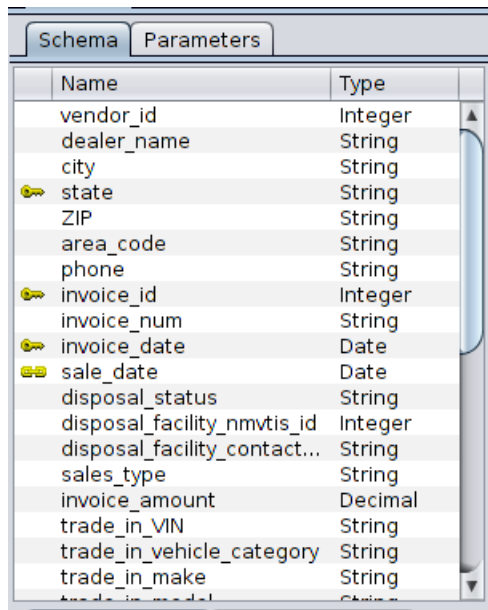
3. Click **OK**.

## Settings Tab

The **Settings** tab displays the structure of the data source. You can add table and column attributes to each field in the data source and add transforms to manipulate the data as desired before output.

## Schema Tab

Click the **Schema** tab to view the database schema of the data source. An example is shown in [Figure 5.6, "Viewing a Schema"](#):

**Figure 5.6. Viewing a Schema**


Name	Type
vendor_id	Integer
dealer_name	String
city	String
state	String
ZIP	String
area_code	String
phone	String
invoice_id	Integer
invoice_num	String
invoice_date	Date
sale_date	Date
disposal_status	String
disposal_facility_nmvtis_id	Integer
disposal_facility_contact...	String
sales_type	String
invoice_amount	Decimal
trade_in_VIN	String
trade_in_vehicle_category	String
trade_in_make	String
trade_in_model	String

## Parameters

The **Parameters** tab displays the parameters of the data source if any.

The Parameter processor retrieves parameters from one input and applies them to another flow. This allows flows to be developed and reused with different sets of parameters. Parameters can be maintained independent of the flows.

For more information on Parameters, see **Parameter Processor** in the **Data Designer Manual**.

## Attributes

This section allows you to add table and column attributes. You can define Primary Key and Foreign Key relationships between the various columns. You can set table level security, set up a table to be cached in memory, and set dimension parameters.

### Table Attributes

This allows you to set the following attributes for the table as a whole.

Click the Add icon (  ).

The various table attributes are shown in [Figure 5.7, “Setting Table Attributes”](#).

**Figure 5.7. Setting Table Attributes**

1. **Change Dimension Capture:** This is used to determine and track the data that has changed, so that suitable actions can be taken using the changed data.

Tables whose changes must be captured usually have a column that indicates the time of last change. Any row in any table that has a timestamp in that column that is more recent than the last time data was captured is considered to have changed.

Enter the name of the columns that contain the time of the last change.



This attribute is reserved for future use.

2. **Slowly Changing Dimension:** Dimension is a logical grouping of data such as customer or product information. With Slowly Changing Dimension (SCD), data changes slowly, rather than changing on a time-based, regular schedule.

For example, assume you have a dimension that tracks the sales from each salesman for each region.

If a Salesman moves to a different region at the end of a month, what happens to the sales data from the next month? Should all the data (including historical data) for the salesman be reflected in the new region? Slowly Changing Dimension takes care of these situations.

Type 1 - This option overwrites all old data with the new. Historical data is not preserved.

Type 2 - This option preserves historical data by creating multiple records for a given natural key in the dimensional tables with separate surrogate keys and different version numbers.

Enter the surrogate key for type 1, along with the start and end fields for which the SCD is valid.



This attribute is reserved for future use.

3. **Static Cache:** For increased performance, you can cache in memory, tables (data sources) that rarely change.

Marking a data source as `Static Cache` causes its entities to be cached in memory.

Some examples of data sources that can be cached are a table of zip codes and the periodic table.

4. **Table Security:** Enter the names of the users who are allowed to view this table. User names can be separated by spaces, tabs or new lines.



By default (leaving this blank), all users are allowed to view this table.



You can also set Column level as well as Row level security by selecting the **Security Transform**. For detailed information on transform categories, refer to the **Elixir Transform User Manual**.

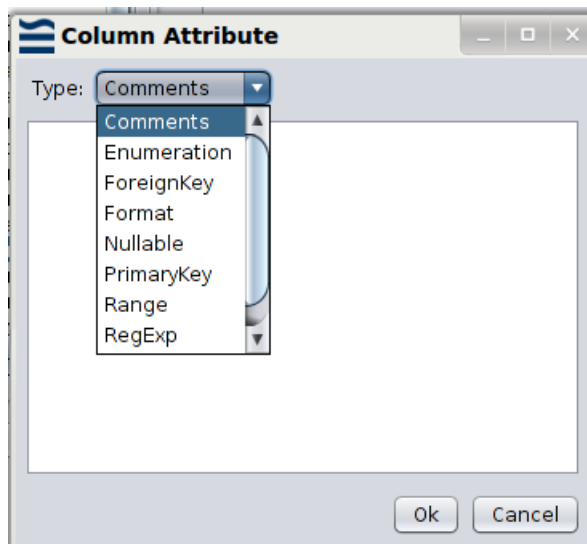
For JDBC and Stave Universes, you can set the Column level and Row level security, from within the **Column Attributes** tab. See the following Column Attributes section for more information.

## Column Attributes

This section allows you to set attributes for a single column in the table. You can enumerate values, set the format for display, set the column as a primary key or as a foreign key, and set the column values to be in a defined range.

The various column attributes are shown in [Figure 5.8, "Setting Column Attributes"](#).

**Figure 5.8. Setting Column Attributes**



The following are the various column attributes:

1. **Comments:** Add any comments that you desire, for your own understanding. These comments are not displayed in the Ad Hoc report or dashboard when published.
2. **Enumeration:** By defining the attributes of individual fields, the system can validate the record values using the cleansing process. For example, you might specify that the field "Gender" is a nominal enumeration of "M" and "F". The cleansing process would then warn you of any records containing "m" or "Female".



Ordinal enumeration attributes allow you to specify the order in which to display data, while retaining the original order. For example, you might specify that the field "Fruit" is an ordinal enumeration of "Apple" and "Orange". Only "Apple" and "Orange" records can pass the validation in cleansing process.

3. **Foreign Key:** A foreign key is a field (or collection of fields) in one table that uniquely identifies a row of another table. Enter the name of the referenced table and the name of the referenced column.

In case of a compound foreign key where the foreign key is made up of two or more columns, the sequence number defines the order of the columns that make up the key.

4. **Format:** Enter the format of the data in this column.
5. **Nullable:** Specify whether the data can be Null values.
6. **Primary Key:** A Primary Key is necessary for all tables in the Repository Universe. Ensure that each table has a unique Primary Key column. This is important for Ad-Hoc Dashboard. For most cases, a table has only one Primary Key, therefore the sequence number is 0.

If you are working on a datasource without an obvious Primary Key - for example - `FruitSales.ds`, add a Sequence Transform to create a new ID column, and add a Column Attribute to mark it as a Primary Key. Since `FruitSales.ds` uses a composite Primary Key, an alternative approach is to add a Column Attribute to mark `Company` as `PrimaryKey:0`, and add another Column Attribute to mark `Fruit` as `PrimaryKey:1`.

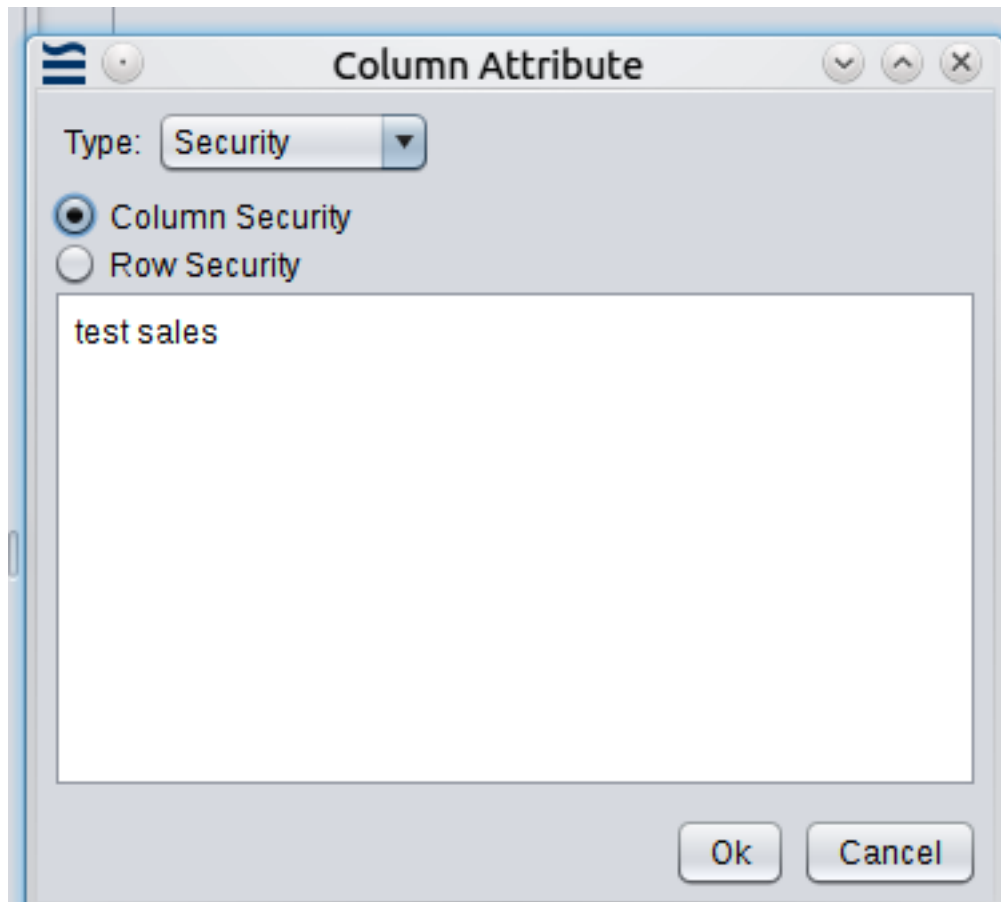
7. **Range:** Specify the start value and end value for the data in the column.
8. **RegExp:** Define and test the regular expression syntax.

Either enter the regular expression, or select from the list. This helps ensure that the data is correctly entered, before processing.

9. **Security (Only applicable for JDBC and Stave Universes):** Allows you to set the Column level and Row level security, for JDBC and Stave Universes.

Enter the names of users and groups who are granted the right to view the column or row data. User names can be separated by spaces, tabs or new lines.

An example of setting Column security is shown in [Figure 5.9, "Setting Column Security"](#).

**Figure 5.9. Setting Column Security**

Note the following points when setting up Column or Row security.

- If you do not set up security, all users are allowed to view the column or row data.
- Make sure that the users or groups to whom you grant access, have at least the `SignIn`, `Adhoc Report View` and `Adhoc Dashboard View` permissions.
- The users or groups also need access to the JDBC or Stave Universe.

The following is a demonstration of column security:

- a. Create either a JDBC Cars Universe or a Stave Cars Universe. Row and Column security do NOT work with Repository Universe yet.
- b. Add some new users and groups as follows:
  - Users: `test2`, `WY`
  - Groups: `CA`
- c. Make user `test2` a member of the group `CA`.
- d. Ensure that users `admin`, `test`, `test2` and `WY` have the following privileges:
  - `SignIn` privileges
  - `Adhoc Report View` and `Adhoc Dashboard View` privileges
  - Access to the JDBC or Stave Universe

- e. Sign in to Ambience as user `admin`.
- f. Create an Adhoc Report in `/Public` with a Table band that contains these columns: (all from `Cars` table) `city`, `dealer_name`, `sales_type` and `state`.
- g. You will be able to view all columns as selected.
- h. Next, open up Designer, edit the Universe, and set the column security for the `sales_type` column to `test2`.
- i. Save the Universe.
- j. Now from Ambience as `admin`, refresh your report. You will not be able to view the values in `sales_type` column as they are hidden since you do not have the permission to view this column.
- k. Log out of Ambience as `admin` and log back in as user `test2`. You should now be able to view all columns in the report (including the `sales_type`) column.

The above procedure demonstrates column security. You can only view the columns that are protected, if you are a user who has the access privilege set in the column security settings. The whole column is protected based on `username + groups`.

The following procedure demonstrates row security.


For this demo, we use the same Universe, report, users and groups and their privileges as in the column security demo.

- a. Remove the column security restriction set earlier in the column security demo.
- b. Open Designer, edit the Universe and set the row security for the `state` column to the word: `test`.
- c. Save the Universe.
- d. Open Ambience and try logging in as the following users to view the report:
  - `test` -> can view all the rows in the table.
  - `WY` -> can view only the `state=WY` rows in the table.
  - `test2` -> can view only the `state=CA` rows in the table (because `test2` is a member of the `CA` group.)
  - `admin` -> cannot view any rows in the table (no state matches the word `admin`).

A user can belong to multiple groups - for example `test2` could be made a member of the `CA` and `AK` groups and see both sets of records.


The above procedure demonstrates row security. Rows are removed based on `access_rights + field value`.

To set an attribute for a column:


1. Click a column from the Schema.
2. Click the Column Attributes tab.
3. Click the Add Attribute icon (.
4. Select the Attribute type.

5. Enter the parameters as appropriate.
6. Click **OK** to save the attribute.

## Adding Transforms

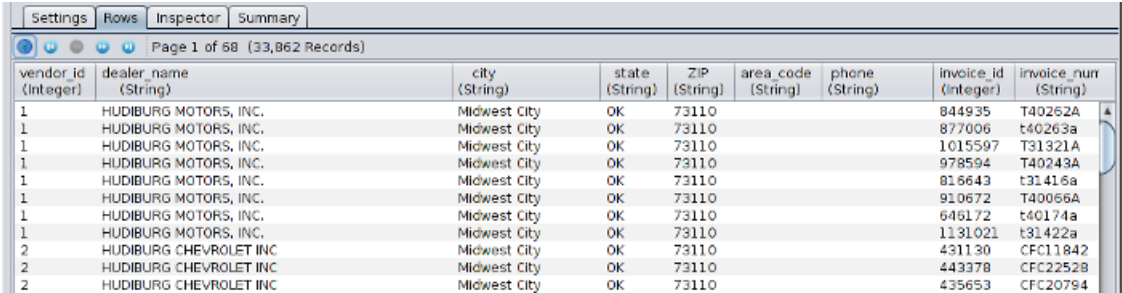
The Transforms tab enables you to create and edit transforms to a schema. Click the Add icon () to invoke the Transform Wizard. For detailed information on transform categories, refer to the *Elixir Transform User Manual*.

## Rows

The Rows tab displays either the original data of a table, or the output of column data if there are transforms created. Click the Load Data icon () to display the rows.

An example Rows tab is shown in [Figure 5.10, “Rows Tab”](#).


**Figure 5.10. Rows Tab**



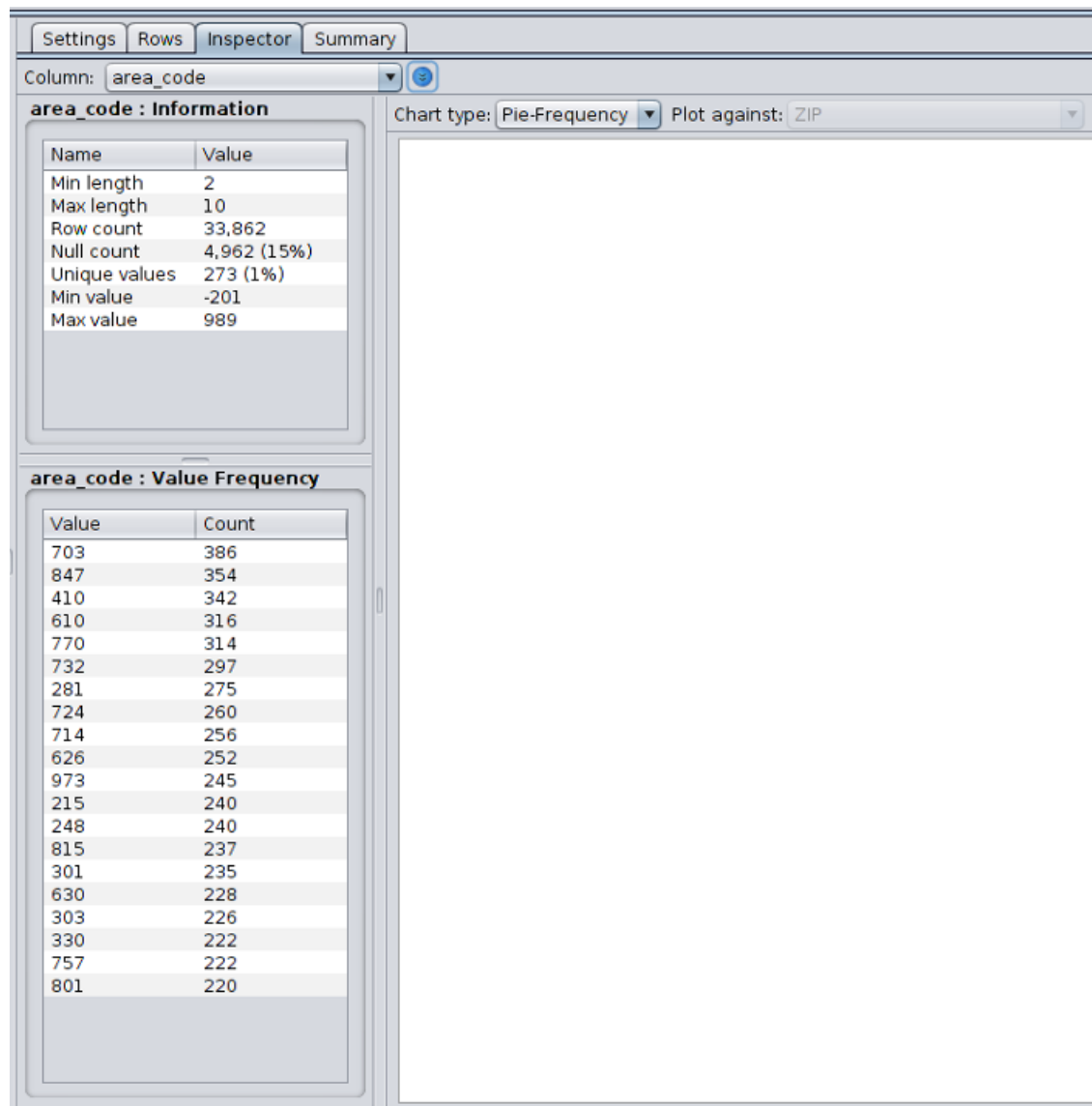
vendor_id (Integer)	dealer_name (String)	city (String)	state (String)	ZIP (String)	area_code (String)	phone (String)	invoice_id (Integer)	invoice_nurr (String)
1	HUDIBURG MOTORS, INC.	Midwest City	OK	73110			844935	T40262A
1	HUDIBURG MOTORS, INC.	Midwest City	OK	73110			877006	t40263a
1	HUDIBURG MOTORS, INC.	Midwest City	OK	73110			1015597	T31321A
1	HUDIBURG MOTORS, INC.	Midwest City	OK	73110			978594	T40243A
1	HUDIBURG MOTORS, INC.	Midwest City	OK	73110			816643	t31416a
1	HUDIBURG MOTORS, INC.	Midwest City	OK	73110			910672	T40066A
1	HUDIBURG MOTORS, INC.	Midwest City	OK	73110			646172	t40174a
1	HUDIBURG MOTORS, INC.	Midwest City	OK	73110			1131021	t31422a
2	HUDIBURG CHEVROLET INC	Midwest City	OK	73110			431130	CFC11842
2	HUDIBURG CHEVROLET INC	Midwest City	OK	73110			443378	CFC22528
2	HUDIBURG CHEVROLET INC	Midwest City	OK	73110			435653	CFC20794

## Inspector

The Inspector tab enables you to review the contents of a column. It displays column information, value frequency based on the data type, as well as the chart type and plot.

Select the column to inspect and click the Load Data icon ()

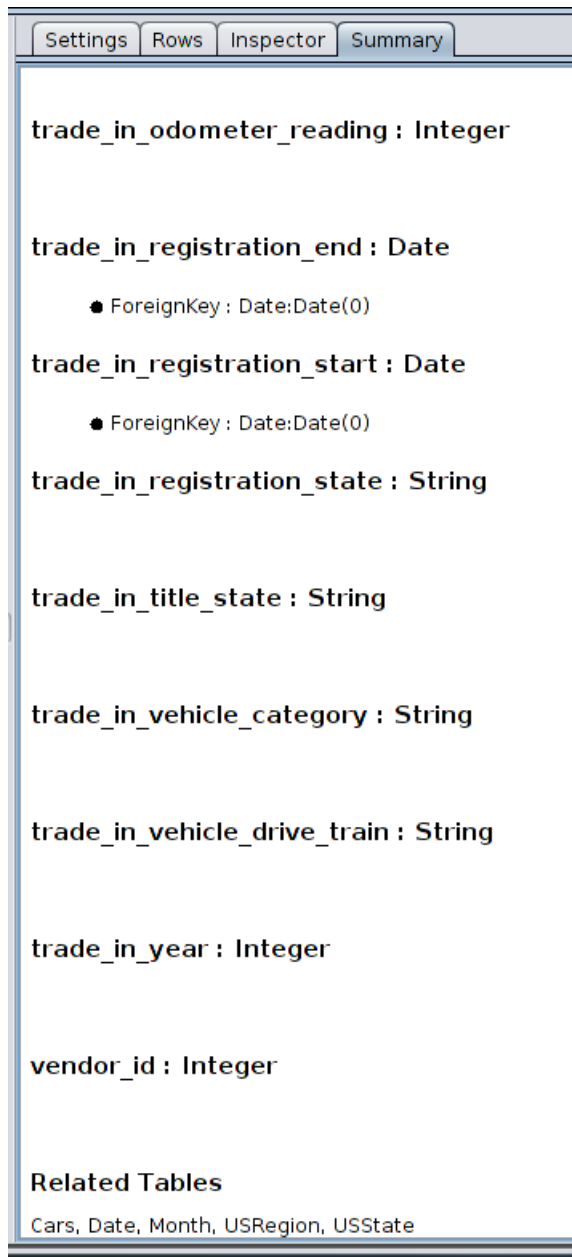
An example Inspector tab is shown in [Figure 5.11, “Inspector Tab”](#).

**Figure 5.11. Inspector Tab**

## Summary



The Summary tab displays the column names, data types and Primary and Foreign Key information.

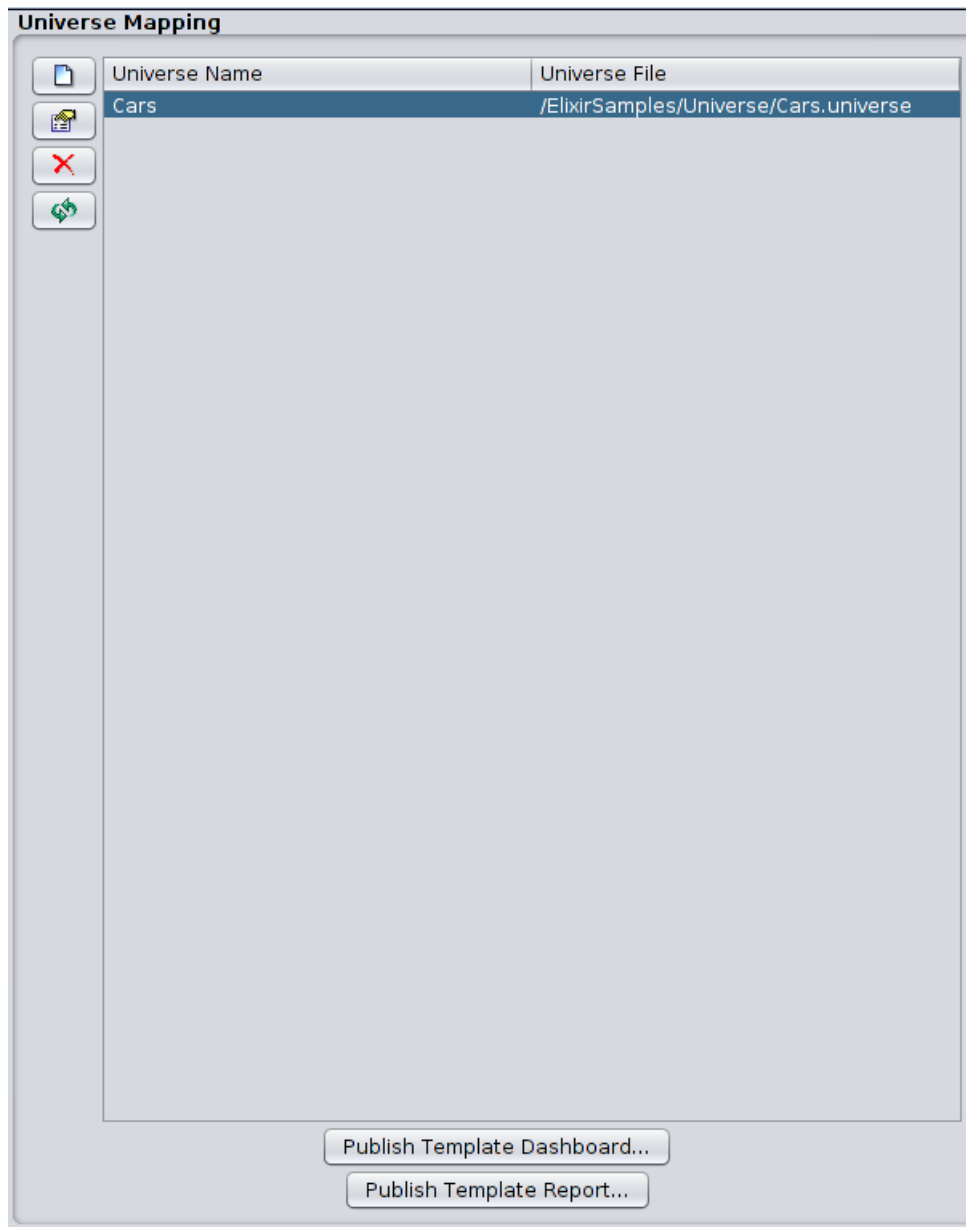
An example Summary tab is shown in [Figure 5.12, "Summary Tab"](#).


**Figure 5.12. Summary Tab**

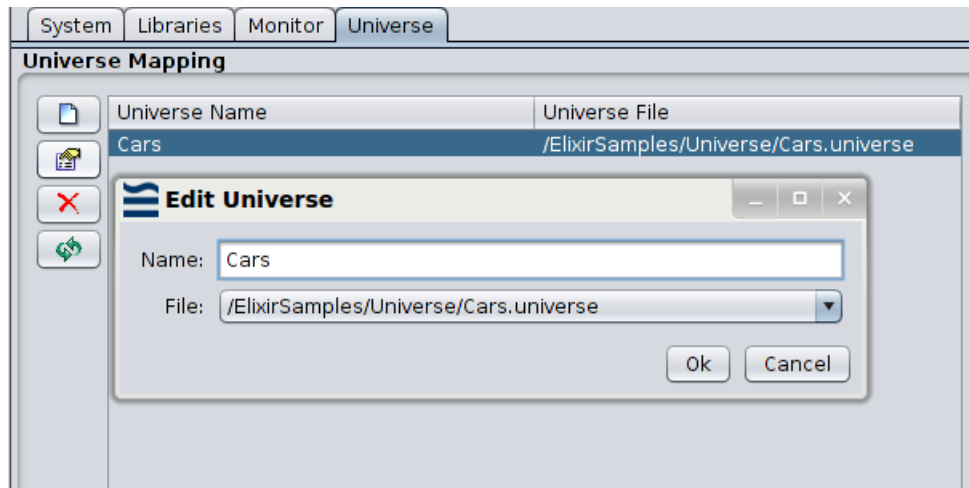
## Exporting the Universe

To export the Universe for use by Ad-Hoc Dashboard and Ad-Hoc Report:

1.  Click the Admin Tools (  ) icon.
2. Select the Universe as shown in Figure 5.13, “Exporting the Universe”.

**Figure 5.13. Exporting the Universe**

3. Optionally, click the Edit (  ) icon and enter your desired Universe name. The Universe is mapped with this name and can now be referred to in the Ad-Hoc Dashboard and Ad-Hoc Report applications. An example of mapping is shown in [Figure 5.14, "Mapping the Universe"](#)


**Figure 5.14. Mapping the Universe**

4. Click either **Publish Template Dashboard** or **Publish Template Report**, to save the Universe in a format suitable for use by Ad-Hoc Dashboard and Ad-Hoc Report respectively.

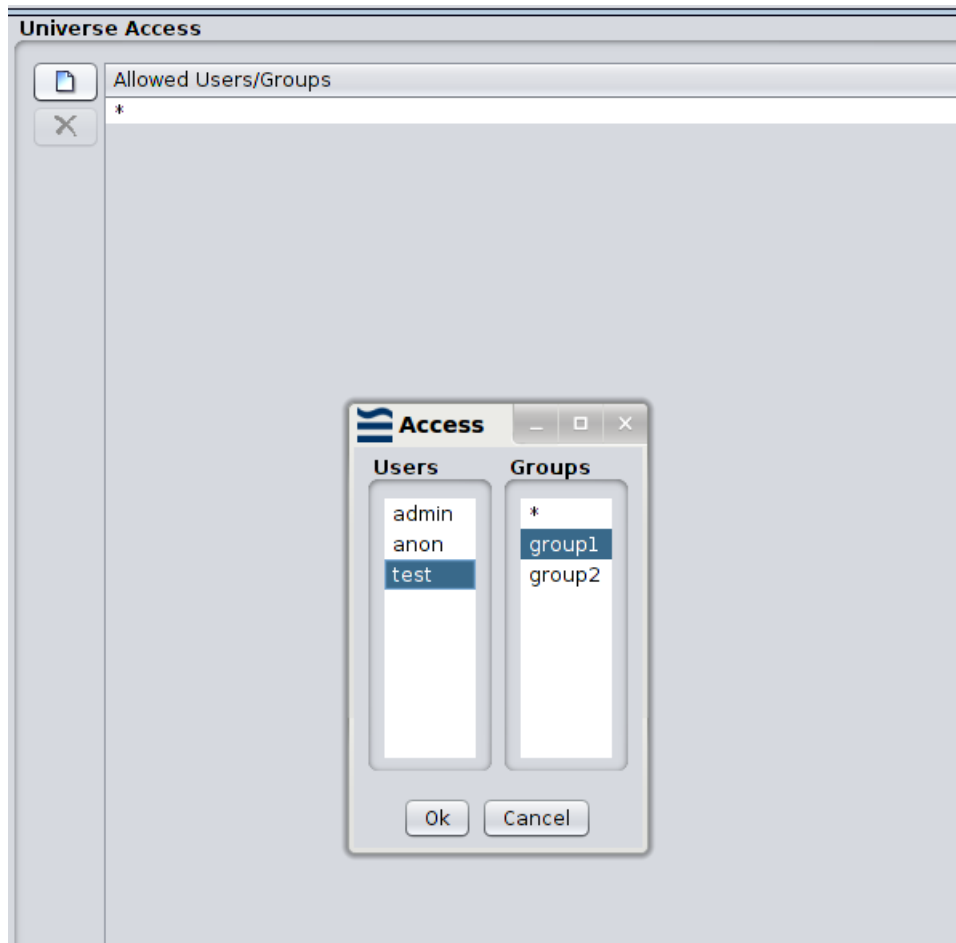
## Setting the Universe Access Permissions

By default, everyone can access the Universe provided it is saved in a publicly accessible folder, i.e. not in any User folder.

Optionally, to set restrict the users and groups who can access the Universe:

1. Click the Universe Access (  ) icon.
2. Select the Users and Groups who are allowed to access the Universe, as shown in [Figure 5.15](#), “Setting the Universe Access Permissions”.



**Figure 5.15. Setting the Universe Access Permissions**

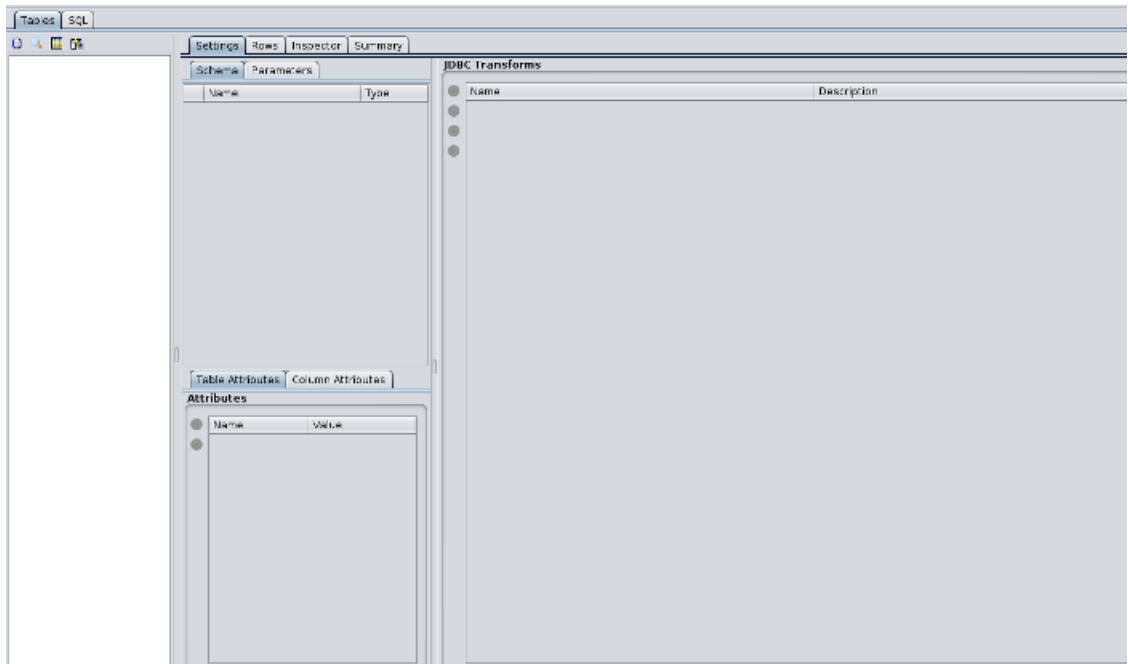
3. Click **OK** to save the access settings.

## JDBC Repository


A JDBC Universe allows data to be read from a relational database using a JDBC driver. If your data is defined in a relational database, and a JDBC driver is available, then JDBC Universe is recommended. Before getting started, add the JDBC driver into `/opt/elixir/lib/`, to make it available to all Java Virtual Machines.

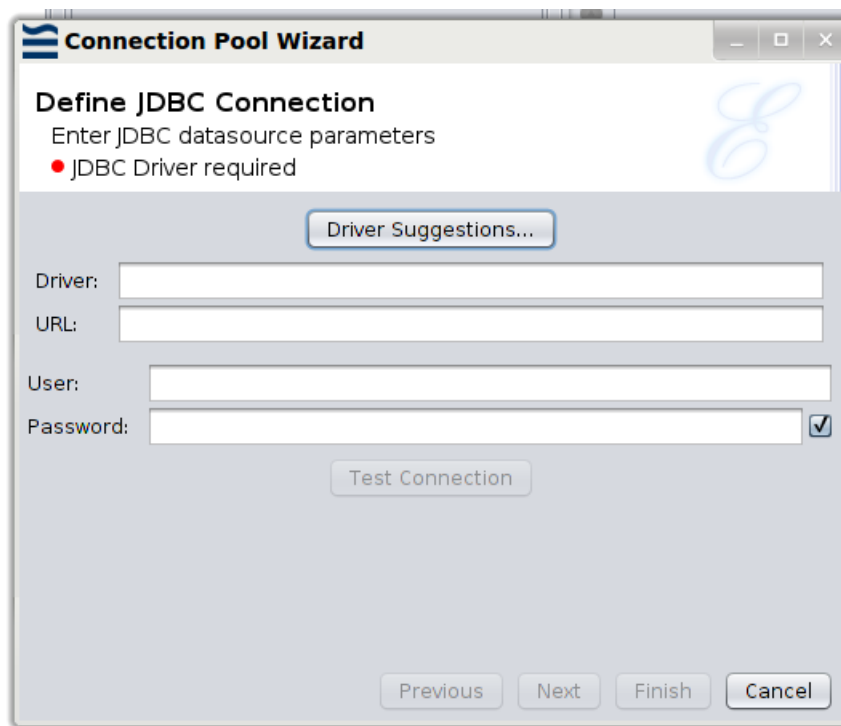
Double click a JDBC universe from the folder tree to open it in the Universe Designer.

The Universe Designer displays as shown in [Figure 5.16, “Universe Designer for a JDBC Universe”](#):

**Figure 5.16. Universe Designer for a JDBC Universe**


## Configuring a Connection Pool

JDBC allows access to data through SQL statements. To configure the connection pool, click the  icon. For example, you can install MySQL Workbench and use `com.mysql.jdbc.Driver` for the JDBC Universe. Modify the host and database name in the **URL** field, enter the user name and password, and click **Test Connection**.

**Figure 5.17. Configuring the Connection Pool**


If the connection test succeeds, you have the option to define the Connection Pool Parameters on the next page. On the Connection Pool JDBC Properties page, you can set any customized properties for your JDBC driver by setting the keys and values. After you complete the settings, click **Finish**.

## Adding Tables



Click the Add Tables  icon. Tables from the relational datasource will display in the Add Tables window. You can choose from these tables, or invert your selection. Click **OK**.

The tables are joined automatically. You can create custom tables by using expressions.

## Pasting Tables

Click the Paste  icon to copy tables from another JDBC Universe and paste them here.

## Viewing SQL

Click the SQL tab. This displays the SQL that will be executed for the table. Click the Load Data  (  ) icon to load the table contents.

## Other Actions

All other actions that you can perform on a JDBC Universe are the same as in [the section called “Repository Universe”](#)

## Stave Repository

Stave is a column based repository. Data is stored in columns. Stave offers better data compression, and extremely fast data retrieval.

Double click a Stave Universe from the folder tree to open it in the Universe Designer.

The Universe Designer displays as shown in [Figure 5.18, “Universe Designer for a Stave Universe”](#):

**Figure 5.18. Universe Designer for a Stave Universe**

Stave Column	Type	Universe Table	Universe Column
Company	String		
DataStore_StaveIndex	Integer		
Date	Date		
Date_day	Integer		
Date_dow	Integer		
Date_month	Integer		
Date_quarter	Integer		
Date_year	Integer		
Fruit	String		
Id	Integer		
Quantity	Integer		
Unit Cost	Decimal		
WeightKG	Decimal		

## Mapping Stave Columns to Logical Tables

Before the Stave columns can be used, you need to map them to a logical table.

Click the **Mapping** tab. For each row, enter a name for the Universe table and a name for the column as shown in [Figure 5.19, “Table Mapping for a Stave Universe”](#).

**Figure 5.19. Table Mapping for a Stave Universe**

Stave Column	Type	Universe Table	Universe Column
Company	String	Fruits	Company
DataStore_StaveIndex	Integer	Fruits	Index
Date	Date	Fruits	Date
Date_day	Integer	Fruits	Day
Date_dow	Integer	Fruits	DayOfWeek
Date_month	Integer	Fruits	Month
Date_quarter	Integer	Fruits	Quarter
Date_year	Integer	Fruits	Year
Fruit	String		
Id	Integer	Fruits	Id
Quantity	Integer	Fruits	Quantity
Unit Cost	Decimal	Fruits	Cost
WeightKKG	Decimal	Fruits	Weight



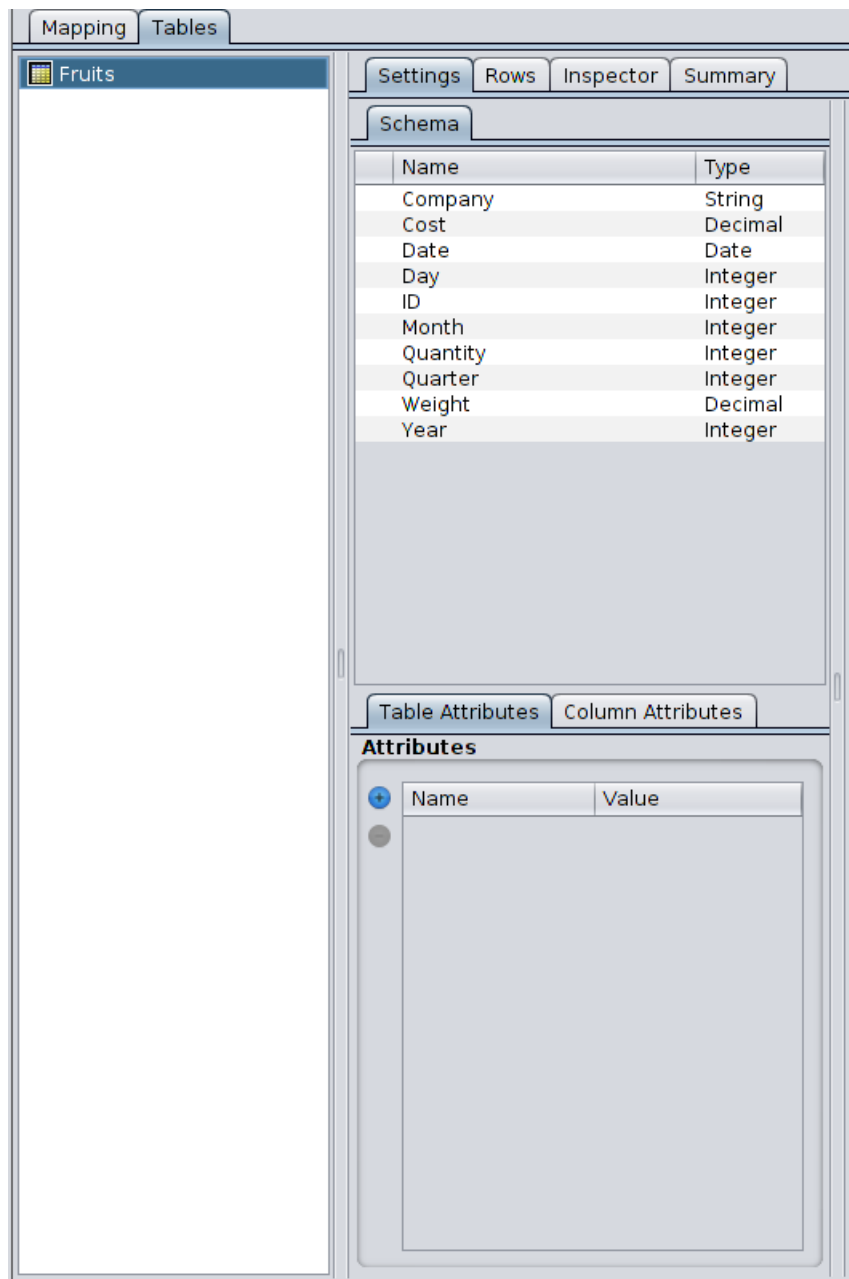
Observe how Stave automatically names the Date columns once you enter a name for the Universe table for the column marked **Date**. This occurs since Date in Stave is internally represented as multiple rows, one each for Day, Day of Week, Month, Quarter and Year.



Similarly, Stave automatically names the Time columns once you enter a name for the Universe table for the column marked **Time**. This occurs since Time in Stave is internally represented as multiple rows, one each for Hour and Minute.

## Operations on Stave

Once you finish Mapping, click the **Tables** tab. You will see the mapped Universe table as shown in [Figure 5.20, “Mapped Table for a Stave Universe”](#).

**Figure 5.20. Mapped Table for a Stave Universe**

Click the table and proceed with the operations as specified in the section called “Repository Universe”.