75 Fifth Street NW, Suite 312
Atlanta, GA 30308, USA
voice: +1-404-592-6897
web: www.Intercax.com
email: info@Intercax.com

Author: Rose Yntema (rose@Intercax.com)
Date: Dec 20, 2015

Technote: Syndeia MySQL Capabilities

Abstract

This technote presents the capabilities of Intercax products Syndeia[1] and ParaMagic[2] to connect system architecture models (SysML) with MySQL databases. Three scenarios and a reference section are presented. In the first scenario, MySQL tables and their rows are used to generate SysML blocks and instances, and create connections between the two. Then, these connections are used to compare and sync SysML block value properties (or instance slots) and MySQL table rows bi-directionally throughout the development process. In the second, the reverse is demonstrated, using SysML blocks and instances to generate a table in MySQL. Finally, in the third scenario setup and execution of SysML parametric models for cost and mass rollups is demonstrated, using the values synchronized from MySQL tables. The reference section presents three methods for finding particular objects in a MySQL repository in Syndeia for the purposes of SysML generation, including filtering the rows of a table with a SQL query, a simple search capability for users without much knowledge of the SQL language, and an advanced

[1] Syndeia – www.intercax.com/syndeia
[2] ParaMagic – www.intercax.com/paramagic

search capability allowing users with sufficient knowledge of the SQL language to formulate complex queries for the database. These capabilities are important especially in the case of very large databases.

Introduction

MySQL databases are often used by engineers to store libraries of components that may be used across many diverse projects. Some of the information stored for the various components may be needed by system engineers to use in modeling throughout the system lifecycle in SysML. Syndeia from Intercax provides the ability to connect elements in a database with their counterparts in SysML, and ensure that values of key parameters such as mass and cost stay in sync, so that all team members are on the same page when using different software. For instance, the database may be updated with new cost or mass values from the manufacturer of externally sourced components, or the systems engineer may update the requirements for an in-house component in SysML that will need to be reflected in the database, and Syndeia will help by linking these two worlds. Also, very large databases with many tables and rows can become unwieldy, so Syndeia provides various ways of searching and filtering. Starting with a component library or equipment list in MySQL table form, perhaps filtered in one of these ways to find the relevant information, modelers may seed blocks and instances in SysML, or vice versa. Modelers may also build parametric mass and cost rollups of the components and values that were seeded from the database. This can be done using SysML parametric modeling along with the ParaMagic solving tool from Intercax. This technical note will walk through these scenarios and illustrate some of the possibilities available for using Syndeia and ParaMagic with MySQL and SysML. A reference section is also included demonstrating three ways to search and filter the MySQL repository

Scenario 1 – Creating SysML blocks and instances from a library of components in MySQL and maintaining the two in sync

Part of a simplified UAV (Unmanned Aerial Vehicle) component library table in MySQL is shown in Figure 1. Query 1 at the top gets all rows and columns of the Component table in the projects schema, and the results are shown below it. The column names with data types are also shown at bottom left.

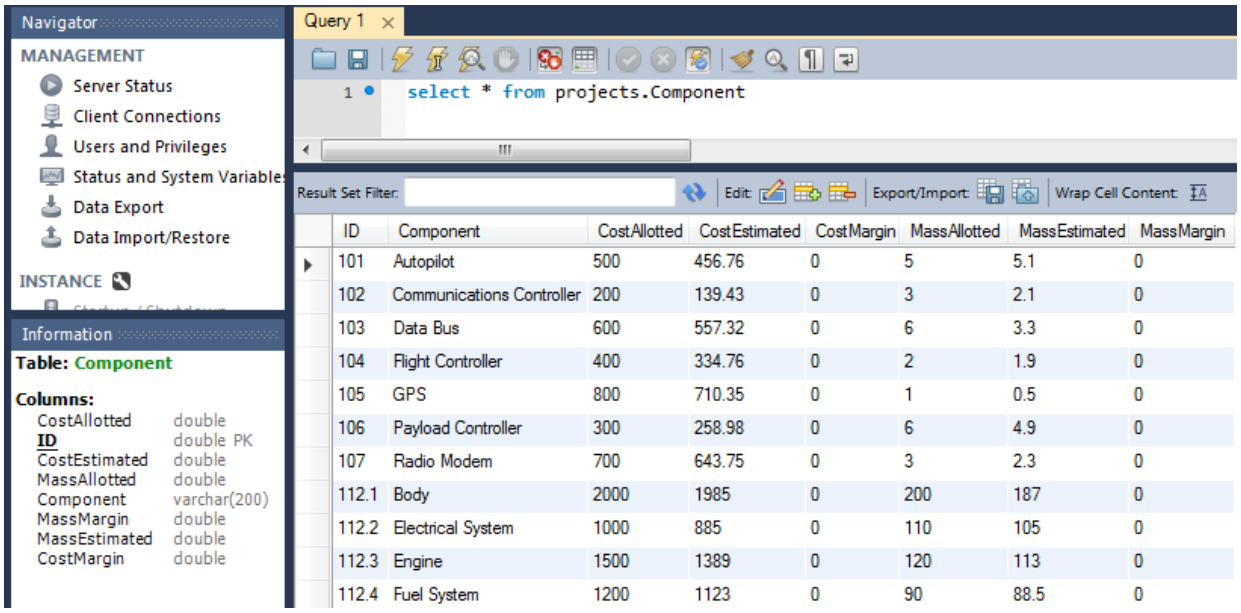


Figure 1 Partial component table in MySQL with name defined (MySQL Workbench)

Figure 2 shows the Syndeia dashboard, Connection Manager tab. The MySQL repository has been loaded on the right side and expanded to show the Component table under the projects schema. Each row in the table (Figure 1) shows up in the Syndeia Dashboard (Figure 2, RHS) in *column_name (value)* format. Before beginning the process of generating objects in SysML, we may want to filter this table, or search the database for tables and rows that match specific criteria. For information on these filter and search capabilities, see the Reference – Ways to search the MySQL repositorysection that follows Scenario 3. To begin the process of generating elements in SysML from the table shown in Figure 2 using Syndeia, we select Model Transform as the connection type (middle panel) and drag the Component table (RHS) to a SysML package on the left. This creates a Component block with value properties corresponding to the columns of the table, as can be seen on the left hand side of the figure below. The Model Transform connection type is used here is so that later on structure-level changes, such as addition or removal of block value properties and Excel table columns, can be detected and synchronized bi-directionally.

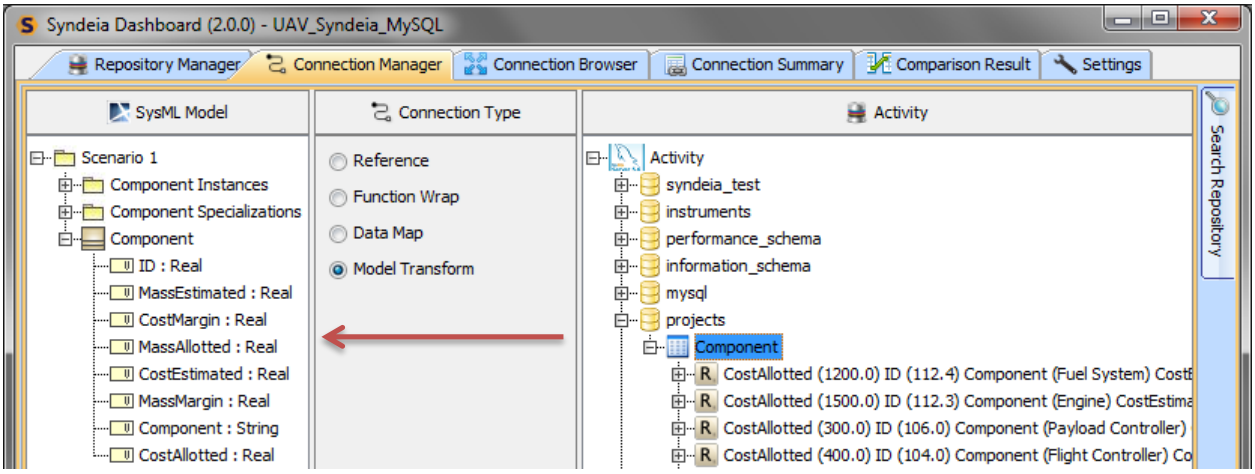


Figure 2 Syndeia dashboard Connection Manager view of Component table and corresponding block

We can now begin to use rows in the table to generate either block instances in SysML with populated slot values, or specializations of the Component block with default values for value properties, corresponding to the values in the table rows. For this we select Data Map as the connection type and begin dragging the MySQL table rows to the destination SysML packages, and select whether to generate an instance or a block specialization. The Data Map connection type is used at this point because it will facilitate compare and sync of attribute values later on. The instances generated in this way are shown below in Figure 3 and the block specializations are shown in Figure 4, expanded to show the values and generalization relations below. Data map connections have now been created between these instances / blocks and the corresponding MySQL table rows, and they can be renamed without losing this connection.

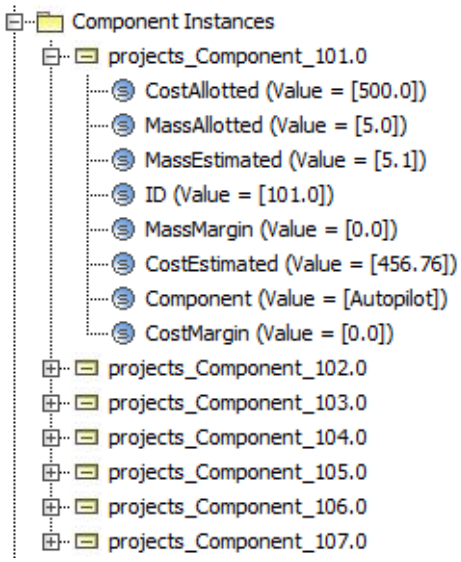


Figure 3 Instances of Component from MySQL rows

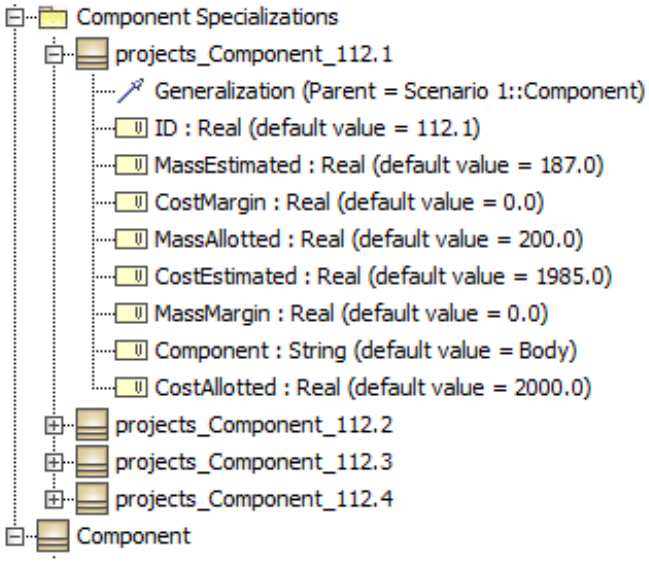


Figure 4 Specializations of Component from MySQL rows

Now we can exercise the power of such data map connections by making changes on one side or the other, and then using Syndeia to compare and update the other end. Consider the situation where new information from a supplier has been received that the Autopilot component has been upgraded, which has caused a change in estimated mass and cost. To update these values in the MySQL database, we can begin with the results of the query in Figure 1, and double-click to change the values in the MassEstimated and CostEstimated columns of the 101/Autopilot row. After editing these, we click Apply as shown in Figure 6, then review the SQL script shown in the prompt and *Apply* again to save.

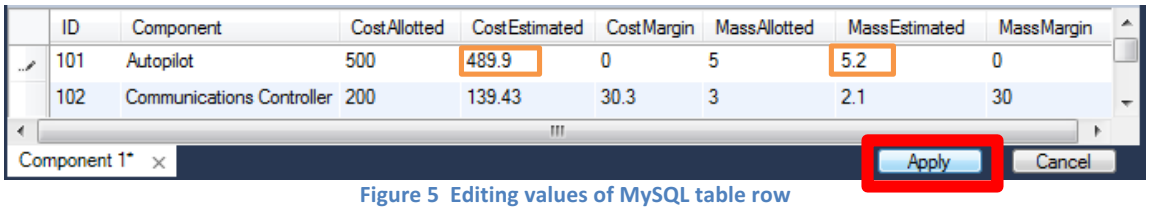


Figure 5 Editing values of MySQL table row

Then the systems engineer, either during a routine check, or prompted by the database manager, can compare SysML and MySQL to see the changes highlighted in the results with a message about what is different as shown in Figure 6.

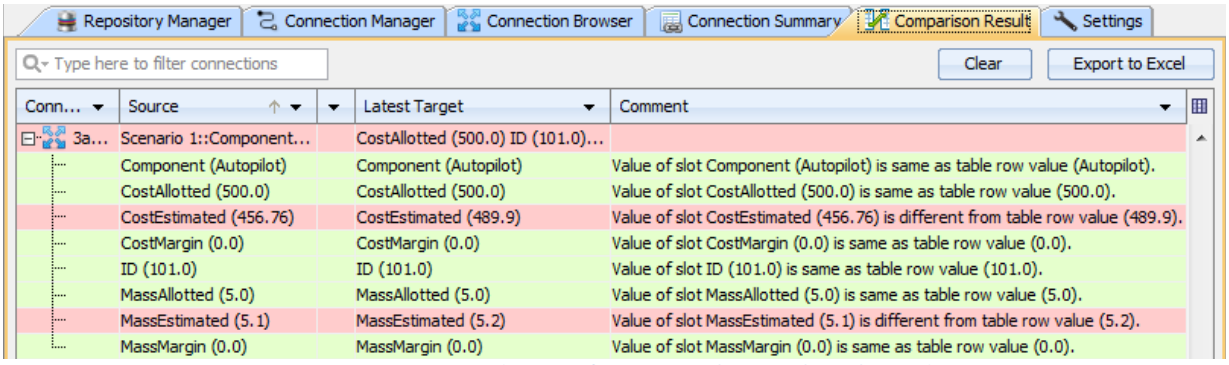


Figure 6 Compare SysML and MySQL for a row with two values changed in MySQL

Figure 7 shows the Syndeia Connection Browser before syncing values from MySQL to SysML, and Figure 8 shows the instance in the MagicDraw containment tree after syncing. At this point, any parametric analyses that use these values (such as those that will be shown in Scenario 3) will have to be performed again.

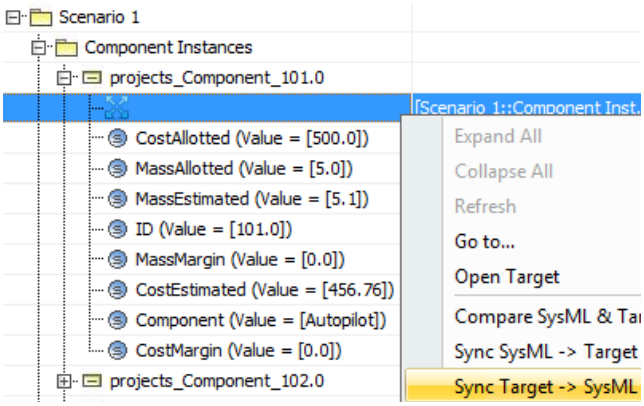


Figure 7 Instance values before syncing Target (MySQL) -> SysML

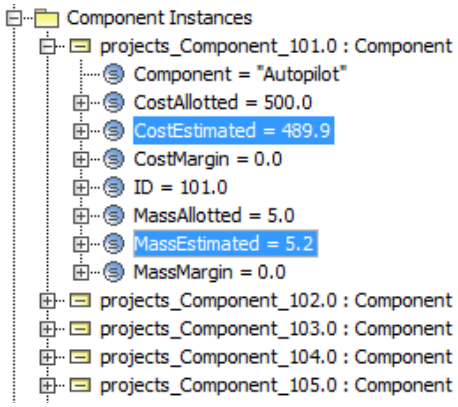
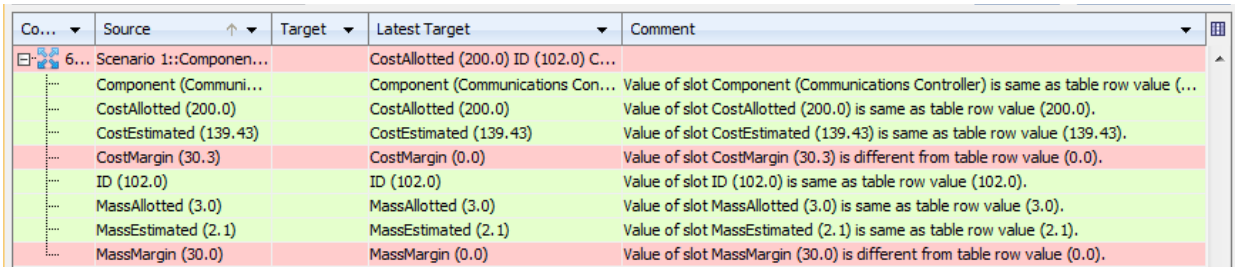


Figure 8 Instance after syncing Target (MySQL) -> SysML

Now consider the case where the systems engineer changes some values in SysML, perhaps by calculating the margins using parametrics, and then wishes to compare and sync them to the MySQL database. The steps are the same as above, except reversed in direction. The system engineer updates the values of Cost Margin and Mass Margin of a component in the SysML model, and then uses Syndeia to compare with MySQL database, which may have also changed as is often the case in concurrent development. The comparison results are shown in Figure 9.



Co...	Source	Target	Latest Target	Comment
6...	Scenario 1::Componen...		CostAllotted (200.0) ID (102.0) C...	
	Component (Communi...		Component (Communications Con...	Value of slot Component (Communications Controller) is same as table row value (...
	CostAllotted (200.0)		CostAllotted (200.0)	Value of slot CostAllotted (200.0) is same as table row value (200.0).
	CostEstimated (139.43)		CostEstimated (139.43)	Value of slot CostEstimated (139.43) is same as table row value (139.43).
	CostMargin (30.3)		CostMargin (0.0)	Value of slot CostMargin (30.3) is different from table row value (0.0).
	ID (102.0)		ID (102.0)	Value of slot ID (102.0) is same as table row value (102.0).
	MassAllotted (3.0)		MassAllotted (3.0)	Value of slot MassAllotted (3.0) is same as table row value (3.0).
	MassEstimated (2.1)		MassEstimated (2.1)	Value of slot MassEstimated (2.1) is same as table row value (2.1).
	MassMargin (30.0)		MassMargin (0.0)	Value of slot MassMargin (30.0) is different from table row value (0.0).

Figure 9 Compare SysML and MySQL for a row with two values changed in SysML

The system engineer uses Syndeia to sync from source (SysML) to target (MySQL). The MySQL table after syncing new margin values is shown in Figure 10. New rows may also be added to the table, corresponding to new components. As long as the model transform connection is maintained between the Component block and corresponding Component table, these new rows can be dragged from MySQL to SysML (or vice versa) as needed to create new instances or specializations

ID	Component	CostAllotted	CostEstimated	CostMargin	MassAllotted	MassEstimated	MassMargin
101	Autopilot	500	489.9	0	5	5.2	0
102	Communications Controller	200	139.43	30.3	3	2.1	30

Figure 10 MySQL table row after syncing SysML -> Target

Scenario 2 – Generating a table in MySQL from blocks and instances or specializations in SysML

Our second scenario begins with a Component block in SysML with value properties of either type Real or String, as shown at the top of Figure 11, specializations of this block that have the same value properties redefined with default values added where applicable, as shown at the bottom of Figure 11, and instances of the same block with some slot values populated, as shown in Figure 12.

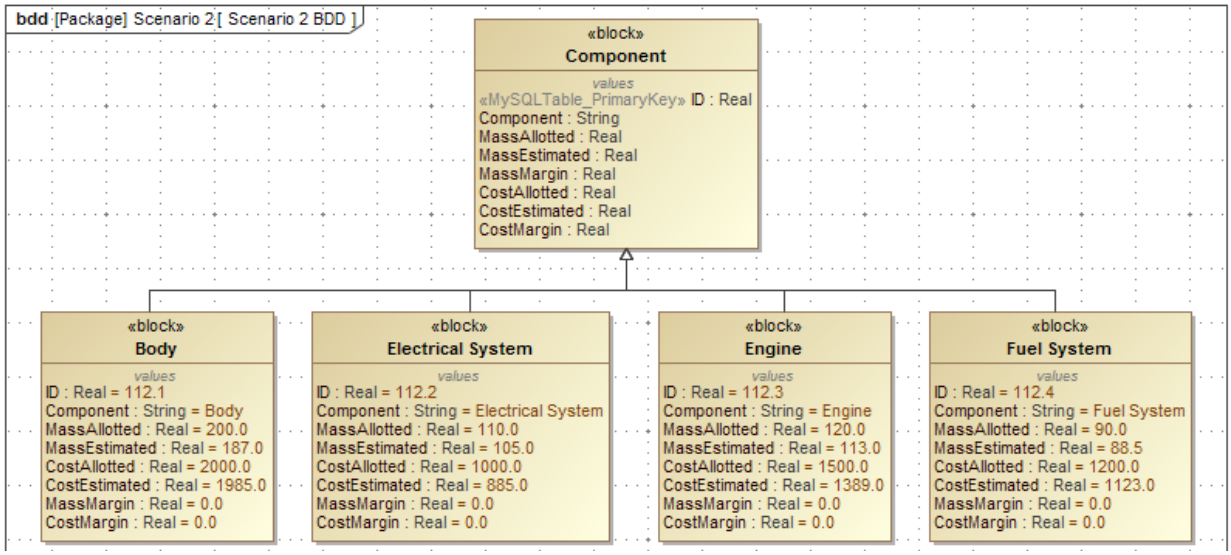


Figure 11 Component block with block specializations for generation in MySQL

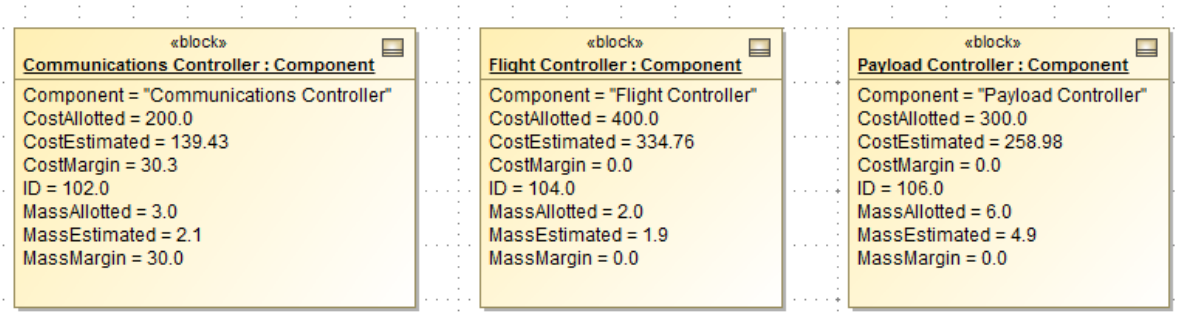


Figure 12 Instances of the Component block for generation in MySQL

Now simply drag the Component block to a MySQL schema as shown in Figure 13.

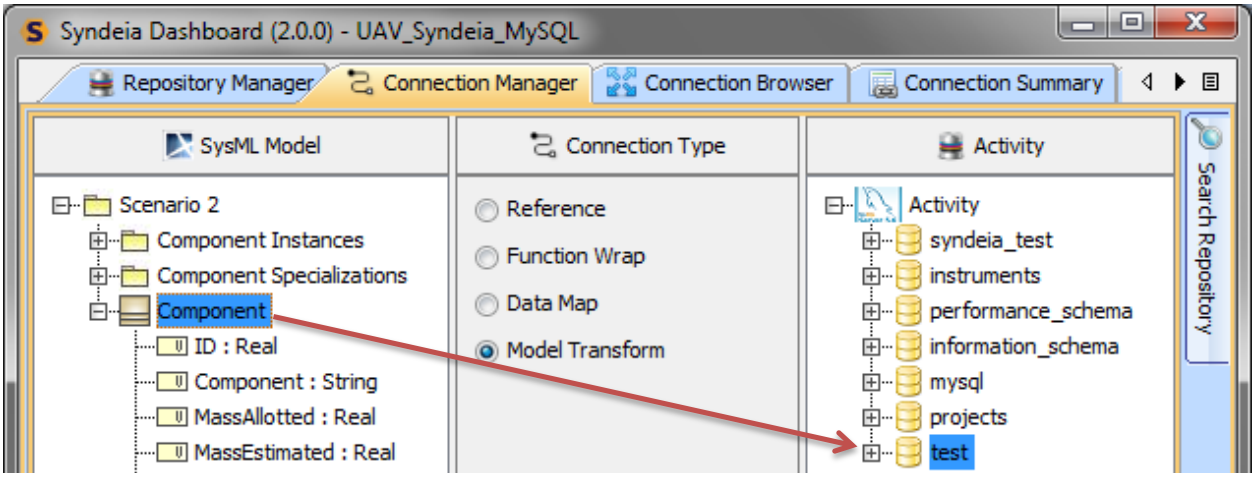


Figure 13 Creating component table in MySQL from Component block in SysML

From here, selecting Data Map connection type in Syndeia, we can drag both instances and block specializations to this new table. This will create a new table row in MySQL that is connected to the corresponding block instance or specialization for downstream sync, as shown in Figure 14.

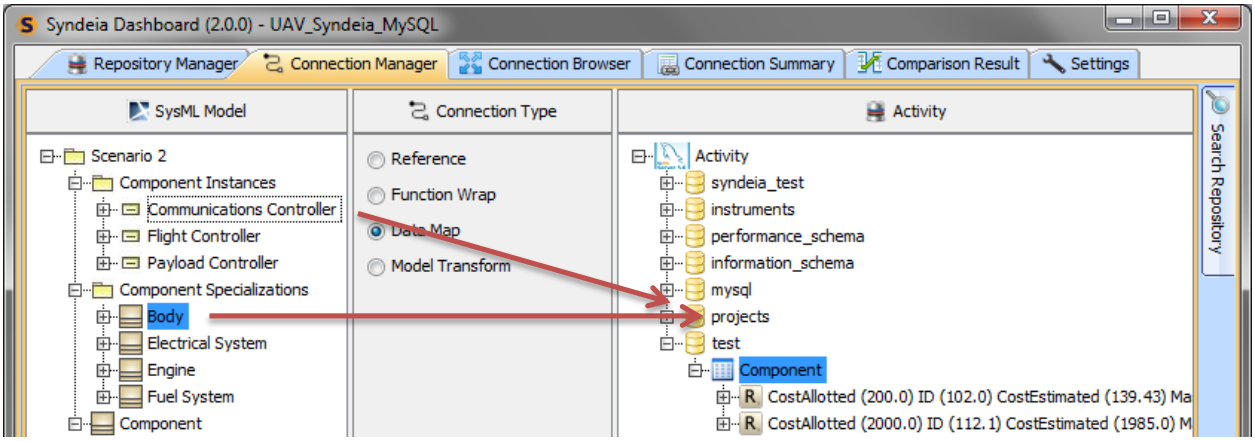


Figure 14 Creating table rows in MySQL from instances and block specializations in SysML

The connections created by dragging SysML -> MySQL are the same as connections created by dragging MySQL -> SysML, so the same compare and sync operations shown in Scenario 1 may be repeated for Scenario 2.

Scenario 3 – Mass and Cost rollups for components taken from MySQL library in Scenario 1

In this last scenario, we will build on the Component block and instances created in Scenario 1 to run some simple mass and cost rollup calculations. First of all, we create a UAV System block and add a directed composition from UAV System to Component block to create a *component* part property, and set the multiplicity of this property as one-to-many (1..*). Then, we create an instance of the UAV System (UAV config 1) and link it with the Component block instances generated from MySQL in Scenario 1. Both the block and instance structure may be seen in Figure 15.

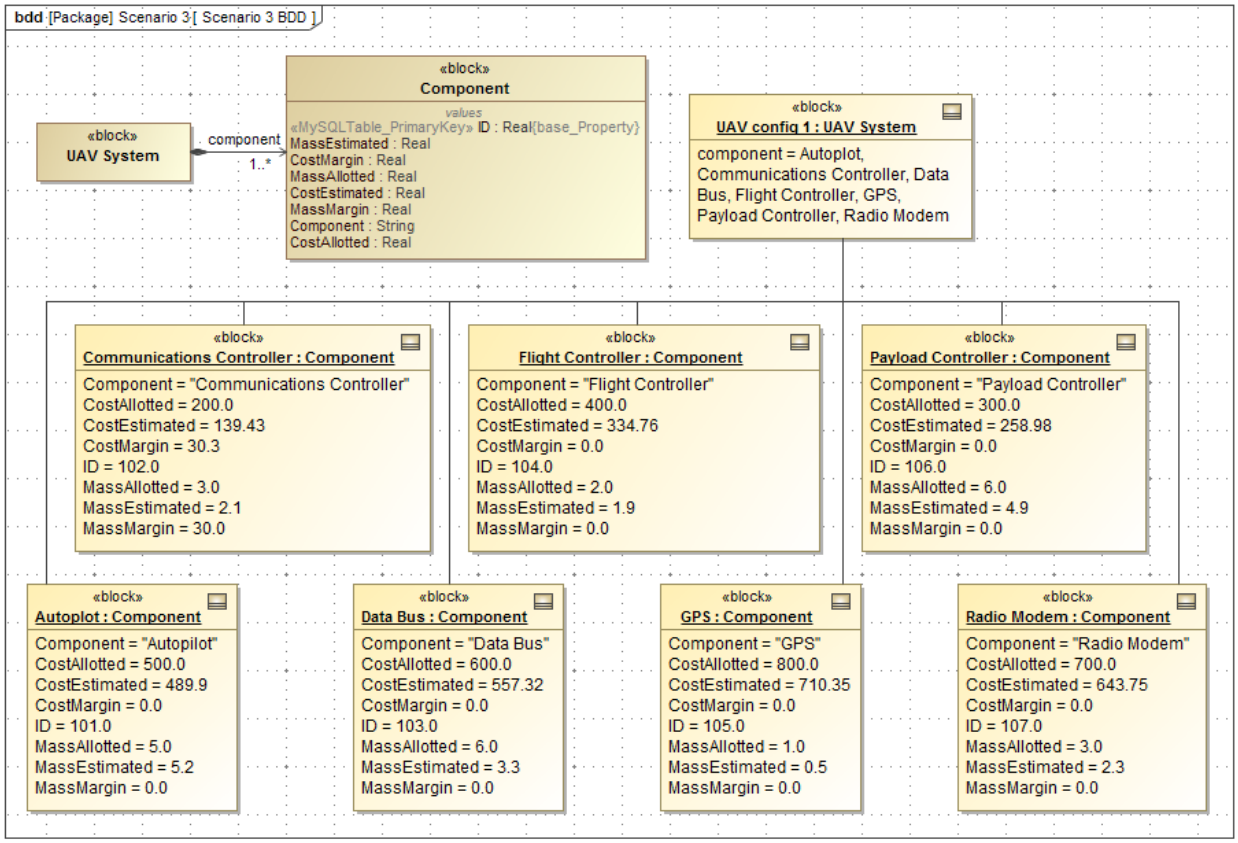


Figure 15 Enhanced block and instance structure with a top-level UAV System block and UAV config 1 instance

Next we create a parametric diagram for the UAV System block, which uses the component part property and the mass and cost value properties of the Component block (Figure 16). Value properties are created in the UAV System block to contain the rolled-up values. Then we use the *Rollup* constraint block with equation "y = sum(x)" four times to roll up allotted and estimated cost and mass.

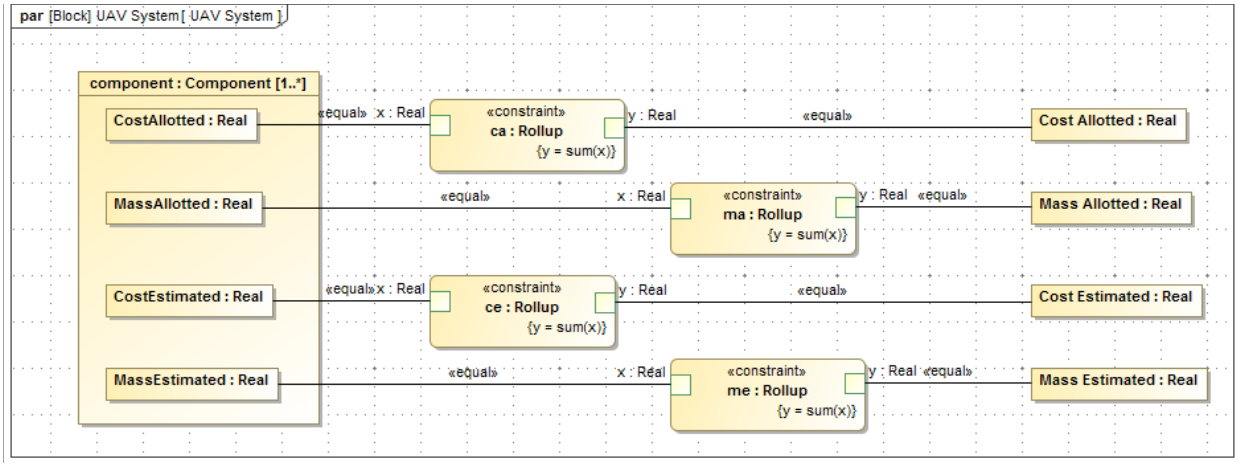


Figure 16 Parametric diagram for the UAV System block, showing rollup constraint properties

From here we simply browse the UAV config 1 instance in ParaMagic, set the Allotted and Estimated Cost and Mass values as targets for solving, and solve, as shown in Figure 17.

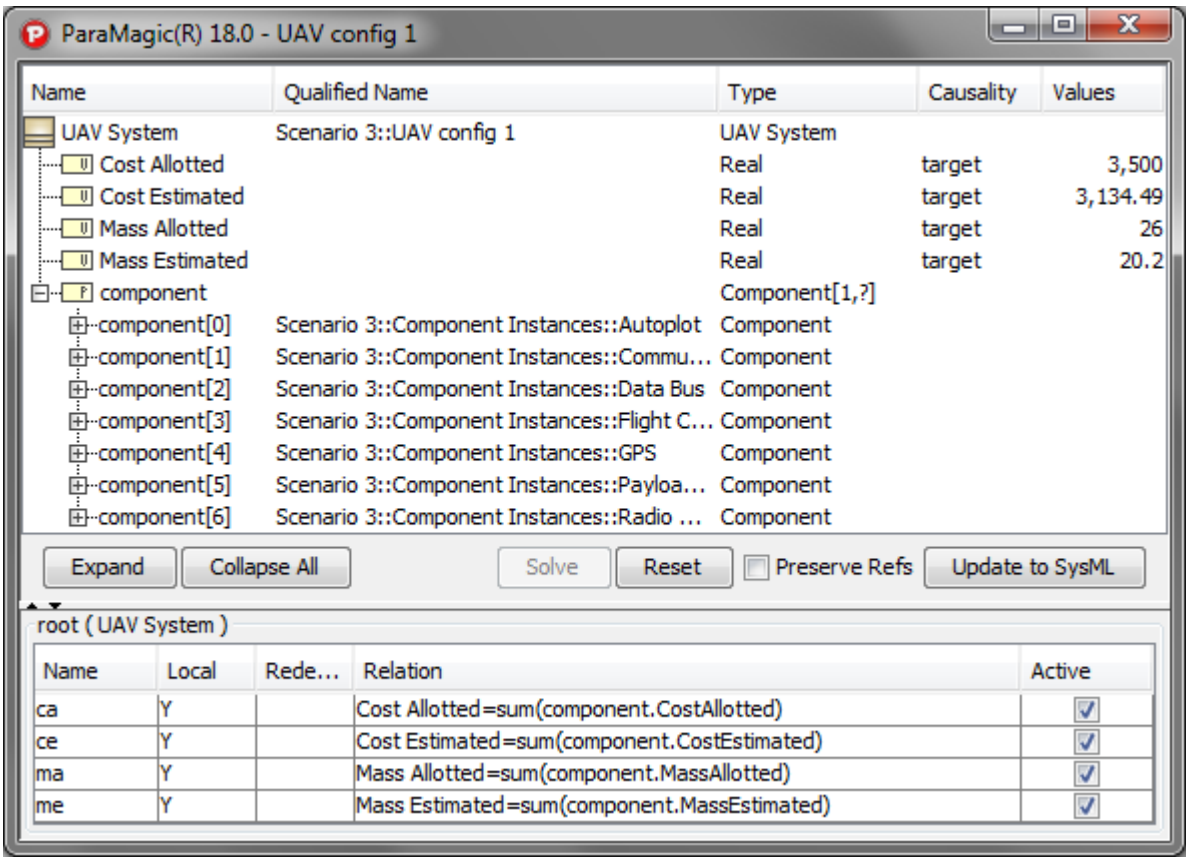


Figure 17 Solutions for cost and mass rollups using ParaMagic

These same general approaches can be extended to additional roll-up calculations, including additional layers of inheritance and multiple inheritance. It is important to evaluate the parametric solver tool capabilities to handle inheritance, recursion and complex aggregates for the class of problems the modeler needs to solve.

Reference – Ways to search the MySQL repository

A few methods have been put in place to find certain MySQL objects in order to use them to generate corresponding objects in SysML. We will outline these in the sections below.

Filtering rows of a table

Firstly, the rows in any table shown under the MySQL tree in SysML may be filtered directly, accessing the menu item by right-clicking the table as shown in Figure 18.

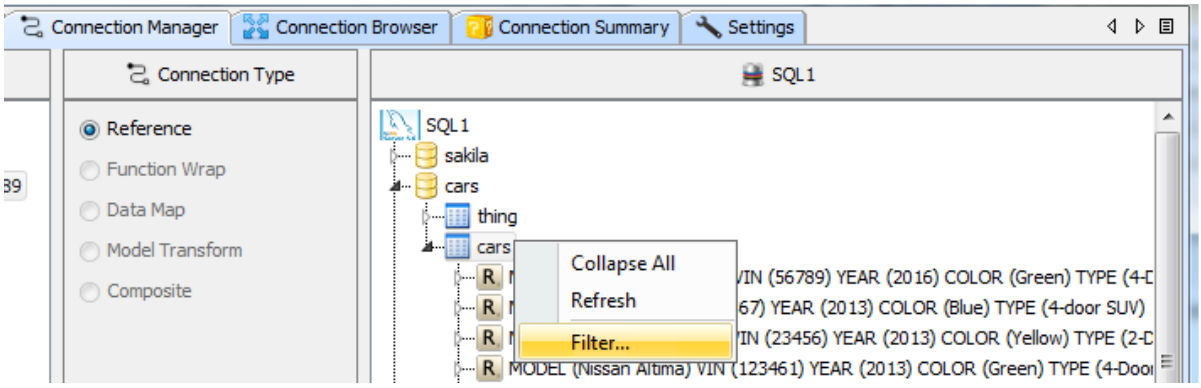


Figure 18 Filtering a MySQL table from the right-click menu

From here we can enter a valid SQL query such as the one shown in Figure 19.

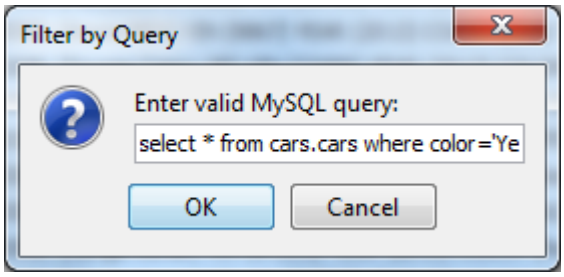


Figure 19 Filtering a MySQL table by valid SQL query, "select * from cars.cars where color='Yellow'"

The table will then be filtered to show only the results of the query above, as shown in Figure 20. After filtering, these rows can be used just as they were in Scenario 1, which can be very helpful in the case of extremely large tables.

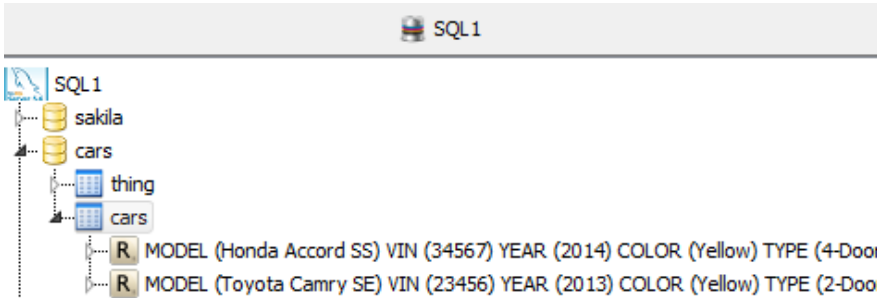


Figure 20 MySQL table after being filtered by query

Simple Search

Another way to search is to open the Search Repository tab on the far right of the MySQL Repository pane on the dashboard.

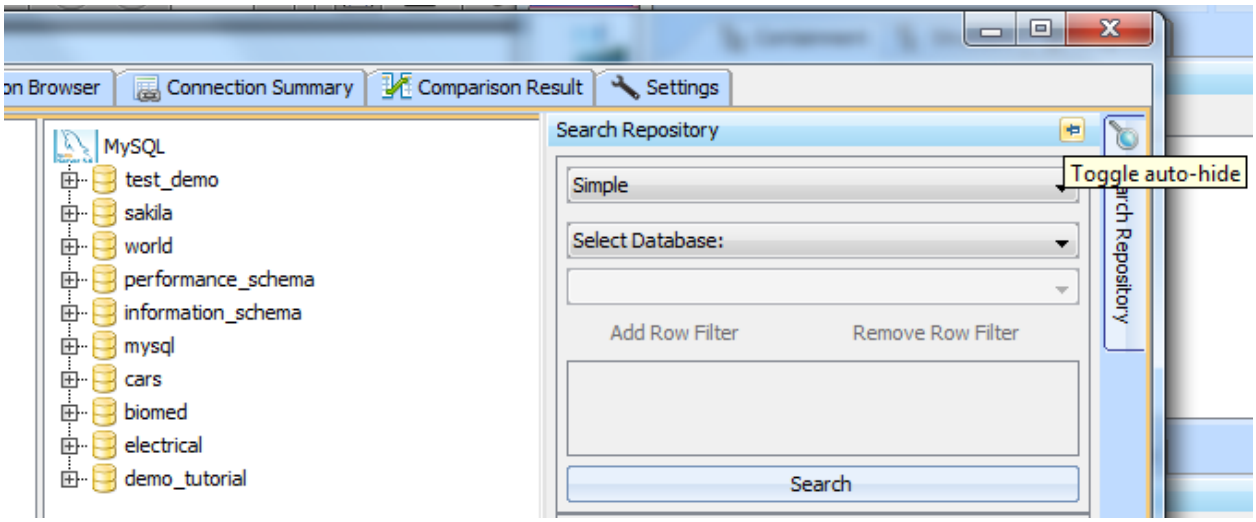


Figure 21 Search Repository tab for a MySQL repository

Using the default Simple search to select a database (schema) and from there a table, as shown on the left and right, respectively, of Figure 22.

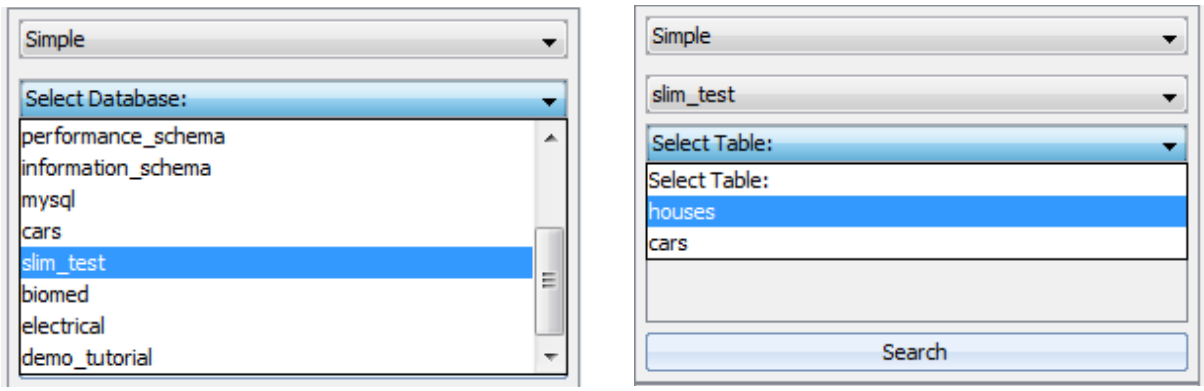


Figure 22 Simple search for SQL: Select Database (schema), left; Select Table, right

From here, all rows of the selected table are shown (Figure 23)

Figure 23 Results of

To filter the table's rows, now simple row filters can be inserted, such as "beds >= 2" as shown with the results of the filter in Figure 24

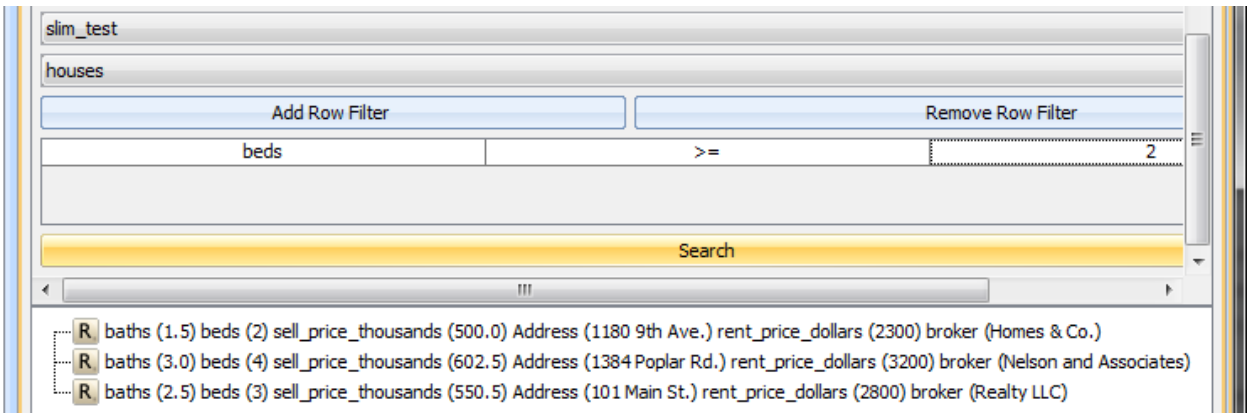


Figure 24 Filtering beds >= 2 and results

Add as many filters as you like, such as the new filter added in Figure 25, without having to write the SQL syntax out. This method is particularly suited for those who aren't fluent in the SQL language.

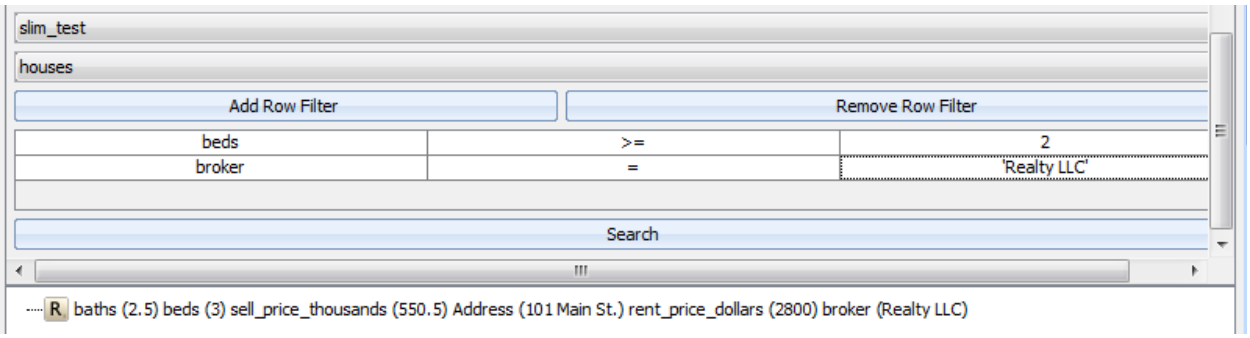


Figure 25 Filtering both beds >= 2 and broker = 'Realty LLC' and the results

Advanced search

The third alternative, recommended for those comfortable writing SQL queries already, is to select Advanced Query to search the entire database, entering a complex query as shown in Figure 26.

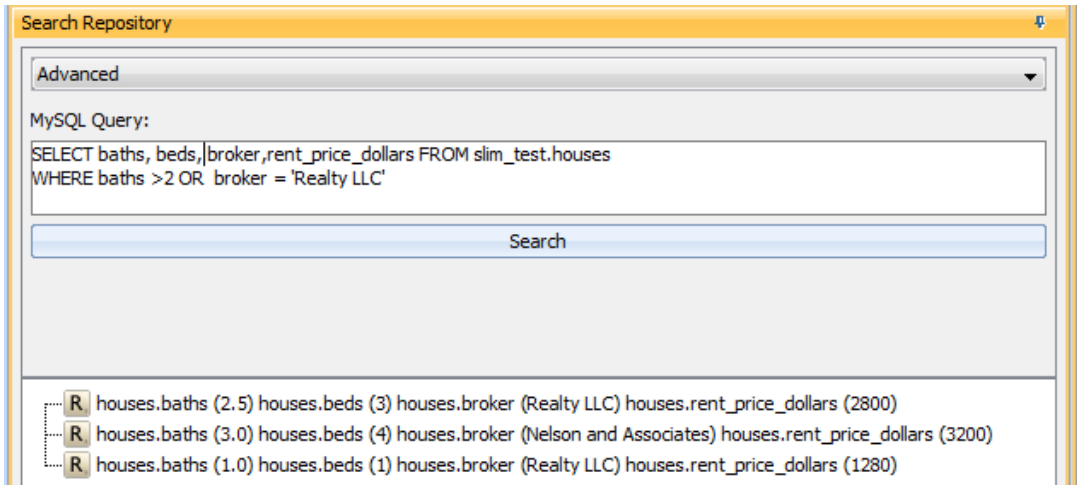


Figure 26 Filtering a MySQL table with an advanced query in the Search Repository tab

Summary

The intent of this technical note has been to demonstrate three basic scenarios for integrating MySQL databases with SysML models using the Syndeia and ParaMagic plugins for MagicDraw & SysML, as well as to show the search and filtering capabilities of Syndeia with MySQL. In the first scenario, MySQL tables and their rows were used to generate SysML blocks and instances while simultaneously creating connections between the two. Then, changes were made to values both in rows of the MySQL table and the block specializations / instances in SysML and synced in both directions. In the second scenario, the reverse was demonstrated, where SysML blocks and instances were used to generate a table in MySQL. In the third and final scenario, the instances generated from MySQL rows in the first scenario were used to perform mass and cost rollup calculations using SysML parametrics and the ParaMagic solver plugin. In the last reference section, three distinct methods of filtering and searching the MySQL repository using Syndeia were demonstrated, including filtering the rows of a table directly with a SQL query, and using the Search Repository tab to perform either a simple search that doesn't require knowledge of SysML, or advanced searches for those familiar with writing complex queries in SQL. This Technote has demonstrated the ability to connect tables in MySQL databases with systems engineering models in SysML, and ensure that values of key parameters such as mass and cost stay in sync between different software environments during concurrent development.

About the Author

Rose Yntema (rose.yntema@intercax.com) is Applications Engineer for Intercax LLC, Atlanta, GA, where she applies MBSE techniques to complex systems in areas such as aerospace, energy, defense, and telecommunications. She is actively involved in the development of SysML parametric modeling and simulation software. Rose earned her M.S. (2012) in Electrical and Computer Engineering from the Georgia Institute of Technology, and Sc.B. (2010) in Electrical Engineering from Brown University.

For further information, visit us at www.intercax.com or contact us at info@intercax.com.