

# Rational Data Architect skills series, Part 2: Generate SQL/XML queries with Rational Data Architect

Transform data from relational data sources into XML format

Skill Level: Intermediate

[Torsten Bittner \(tbittner@us.ibm.com\)](mailto:tbittner@us.ibm.com)  
Software Engineer  
IBM

07 Sep 2006

You can use the SQL/XML query language to transform relational data into XML format. Since it is cumbersome to manually write SQL/XML queries, IBM® Rational® Data Architect simplifies this work by automatically generating SQL/XML queries based on graphically defined mappings. Get an introduction to the SQL/XML generation component of Rational Data Architect.

## Section 1. Before you start

Learn how to use Rational Data Architect for SQL/XML query generation in DB2® for Linux®, UNIX®, and Windows®. The SQL/XML query language is used to extract data from relational sources and transform it into XML format.

### About this tutorial

This tutorial shows you, step by step, how to:

- Graphically define a mapping model between a relational database model as the source and an XML schema definition as the target

- Generate an SQL/XML query based on this mapping model
- Execute the SQL/XML query and analyze the result
- Define joins between multiple source columns
- Add transformation functions to the mapping model

## Objectives

After taking this tutorial, you should be able to use Rational Data Architect and its mapping editor to generate and run SQL/XML queries.

## Prerequisites

### Product name change

On December 16th, 2008 IBM announced that as of Version 7.5.1, Rational Data Architect is renamed to [InfoSphere Data Architect](#) to feature its role in InfoSphere Foundation Tools.

This tutorial assumes familiarity with relational databases, preferably DB2. You should also have a good understanding of the XML and XSD standards. Basic knowledge of the SQL/XML query language is required. Basic knowledge of Rational Data Architect is beneficial, but not required. (Consult the article "[Use Rational Data Architect to integrate data sources](#)" (developerWorks, March 2006) and part one of this series, "[Access and integrate enterprise metadata with Rational Data Architect](#)" (developerWorks, July 2006), for reference.)

## System requirements

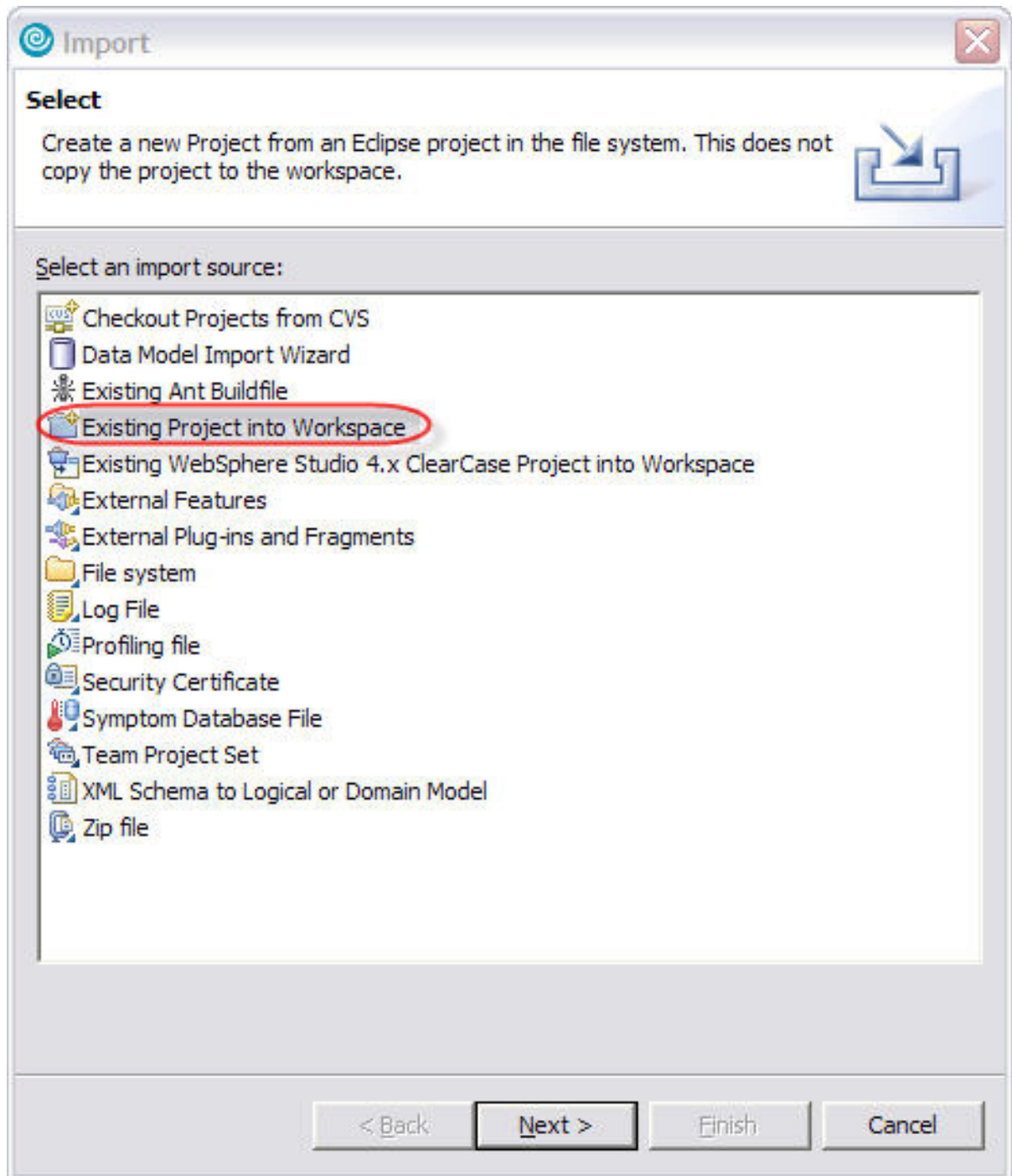
In order to execute the steps that are described in this tutorial, it is necessary to have Rational Data Architect 6.1 and DB2 installed. You can download trial versions of IBM Rational Data Architect 6.1 (see [Resources](#)) and DB2 V8.2 (see [Resources](#)).

### Setup steps

1. Install [DB2 V8.2](#).
2. Install [Rational Data Architect V6.1](#).
3. Unpack the package RDA\_SQLXML.zip into a folder (for example, C:\RDA\_Tutorials). This will create the RDA\_SQLXML folder.

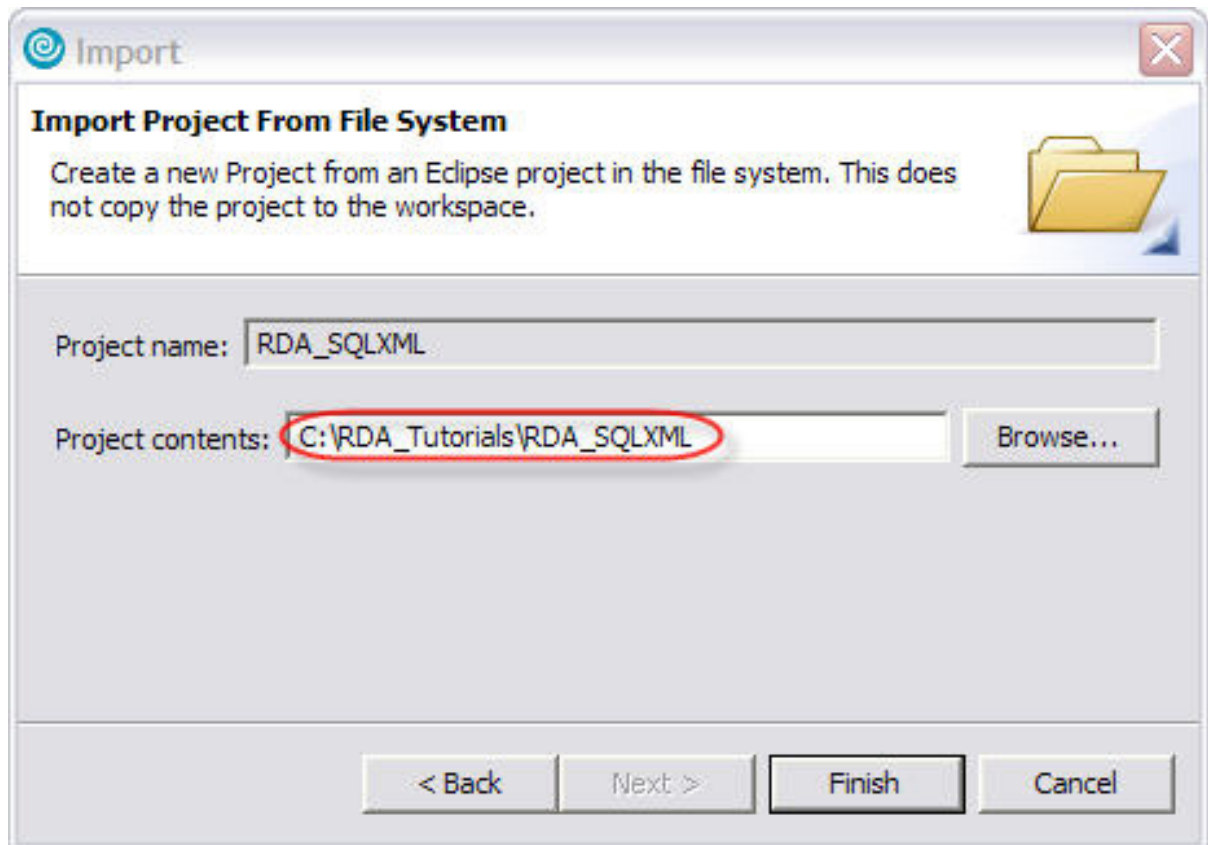
4. Start Rational Data Architect and specify the folder where you unpacked the package as the location for your workspace (for example, C:\RDA\_Tutorials).
5. The folder RDA\_SQLXML that is in the package is a Rational Data Architect data project folder. In Rational Data Architect you have to import it into your workspace. From the File menu, select **Import**.
6. Select the Existing Project into the Workspace Wizard.

### Figure 1. Import Wizard selection



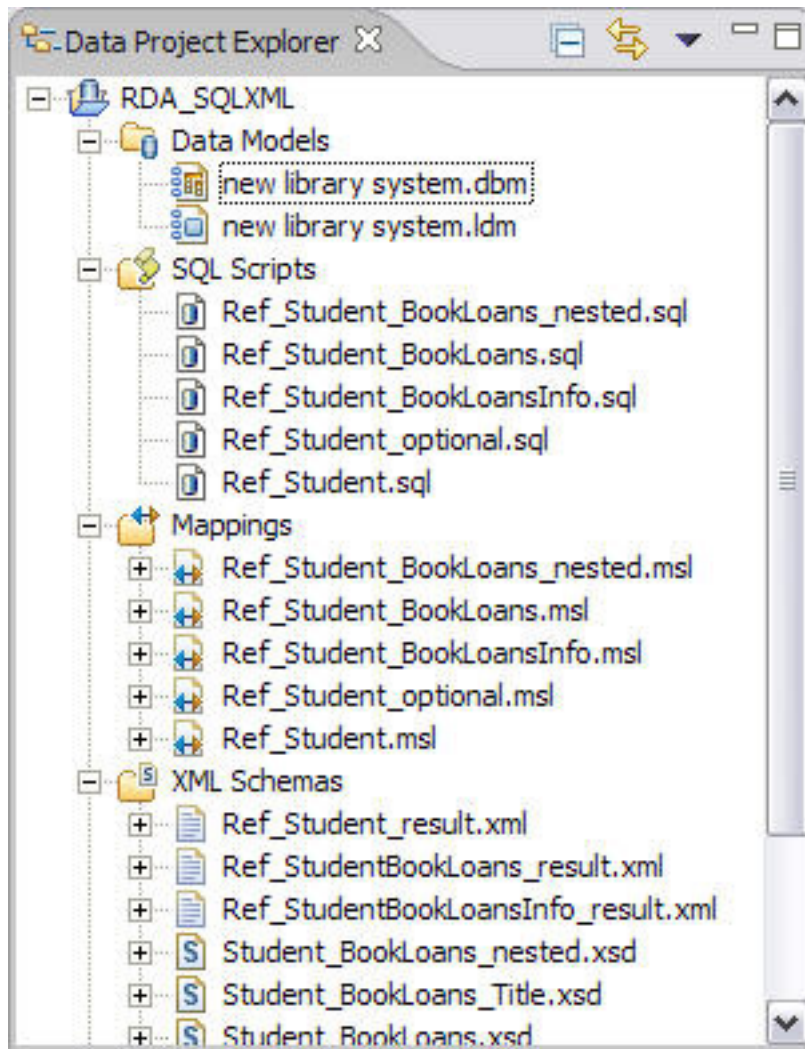
7. Click **Next**. Browse to the location where you extracted RDA\_SQLXML.zip (for example, C:\RDA\_Tutorials).

**Figure 2. Project Import Wizard**



8. Click **Finish**. As a result you see the RDA\_SQLXML project with a set of data models, XSD schemas, and mappings model files in your workspace.

**Figure 3. Data Project Explorer after project import**



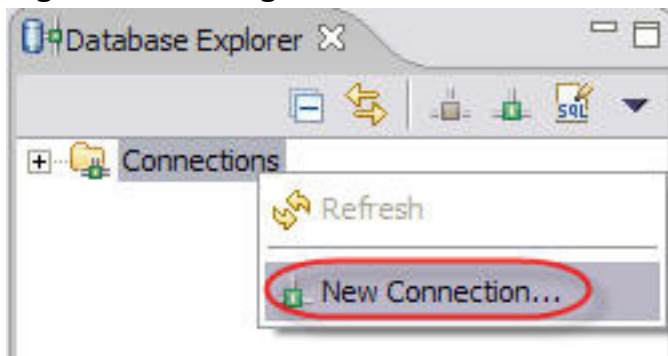
9. Now you need sample data for the relational database. The data will be used to fill the generated XML instance document with data. This data will be extracted from the relational database and inserted into the XML instance document during SQL/XML query execution. The script file `rda_sqlxml_db.sql` creates a DB2 database LIBRARY and inserts a set of sample data. Deploy the script in your DB2 database. Start the DB2 command window (Windows menu **Start > IBM DB2 > Command Line Tools > Command Window**).
10. Change to the folder where you extracted RDA\_SQLXML.zip.
11. Run this command: `db2 -tvf rda_sqlxml_db.sql`. This will create the database LIBRARY, define primary and foreign keys, and insert sample data.

**Figure 4. Creating LIBRARY database**

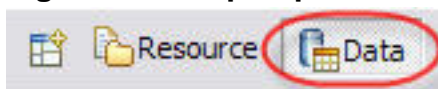


12. Connect to the database LIBRARY in Rational Data Architect. In the Database Explorer, right-click on **Connection** and select **New Connection**. (**Note:** If you don't see the Database Explorer, make sure that you are in the Data Perspective, shown in [Figure 6](#) below.)

**Figure 5. Creating new database connection**

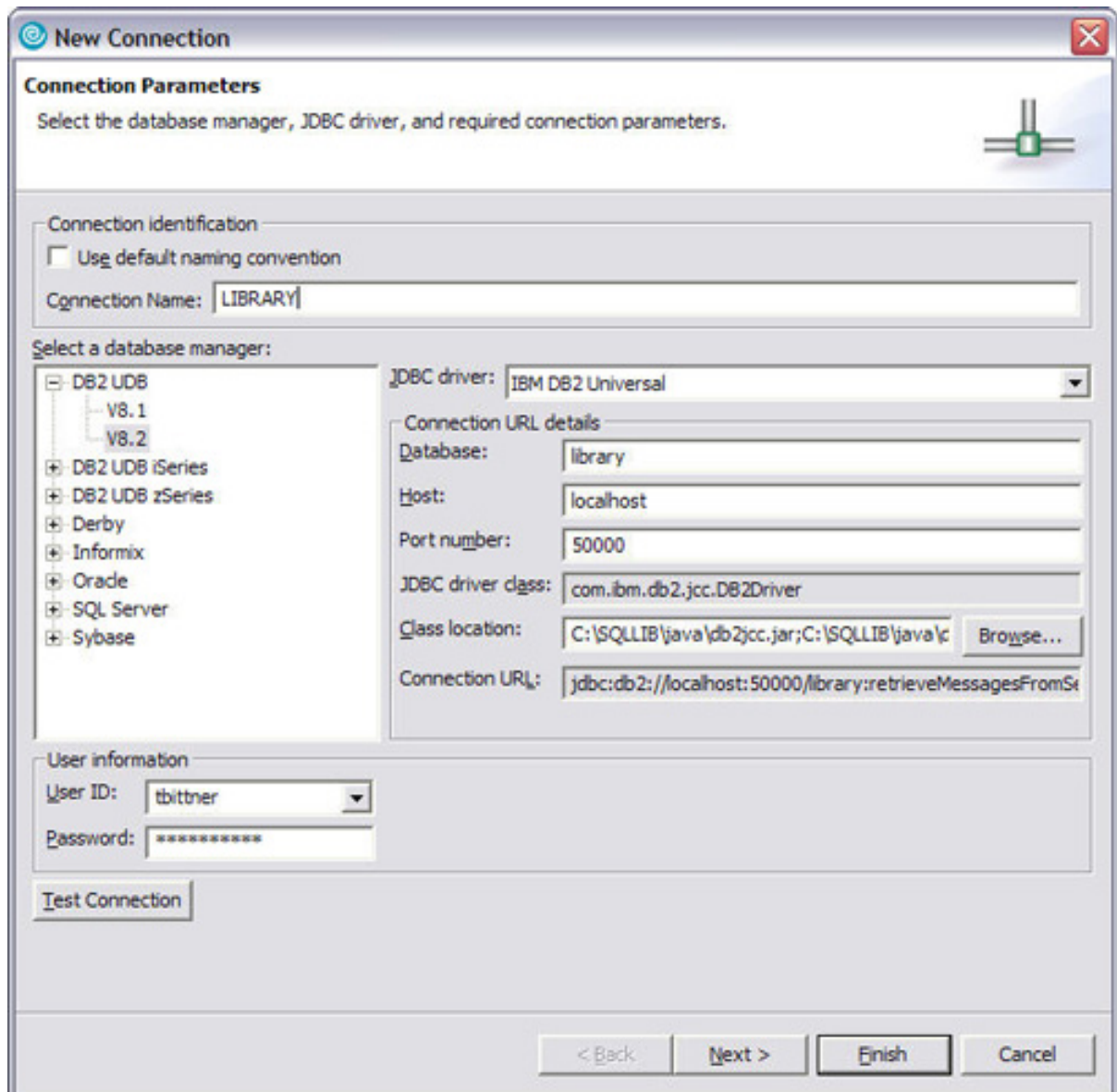


**Figure 6. Data perspective**



13. Specify the connection information according to your environment, similar to [Figure 7](#).

**Figure 7. Database connection settings**



14. Click **Test Connection** to check whether all parameters are set correctly. If the test is successful, click **Finish**.

---

## Section 2. Scenario overview and problem description

The scenario is related to the new school library database system introduced in the tutorial "[Access and integrate enterprise metadata with Rational Data Architect.](#)" In

the tutorial, a new library system is designed based on two existing systems: the school's library and the Santa Clara County's library. For auditing purposes the school wants to keep track of the history of loaned books. The library council decided to stay independent from database vendors by storing the data in standardized XML form.

In order to extract XML data from a relational database, you can use the SQL/XML query language. SQL/XML is an ANSI and ISO standard language for accessing and managing data stored in relational databases. Its syntax is defined in the SQL2003 standard.

Generally, the structure of XML documents is described using an XML Schema Definition. An XML document that conforms to an XSD is also called an XML instance document.

An SQL/XML query also defines the structure of the XML instance document as well as the tables and columns (data source) that are used to fill the XML instance document with data. In that sense, the SQL/XML duplicates the information that is contained by the XSD (target XSD) about the output format of the XML instance document. So SQL/XML queries combine information about document structure and data source definition. This combination makes SQL/XML queries complex. If the database schema or the XML instance document schema or both contain many entities manually writing of SQL/XML queries is cumbersome and error prone.

Rational Data Architect simplifies the creation of SQL/XML queries by providing the user a mapping editor. The mapping editor is used to graphically define mappings from a relational database as the source to an XML Schema Definition as the target. The mappings are interpreted by a query generation component to generate an SQL/XML query. Applying this query to the source database extracts the relational data from the database and creates XML instance data in the structure that is defined by the target XSD.

The mapping editor also allows mappings from a relational data source to a relational data target. In this scenario, the mapping information is interpreted to generate an SQL query to transform the data. More information about SQL generation can be found in the tutorial "[Access and integrate enterprise metadata with Rational Data Architect.](#)"

---

## Section 3. Create a simple mapping model

In this section, create a mapping model. Define the relational data source schema and the target XSD schema, and create mappings between source columns and

target XML elements. Then use these mappings to generate an SQL/XML query.

## Create a new mapping model

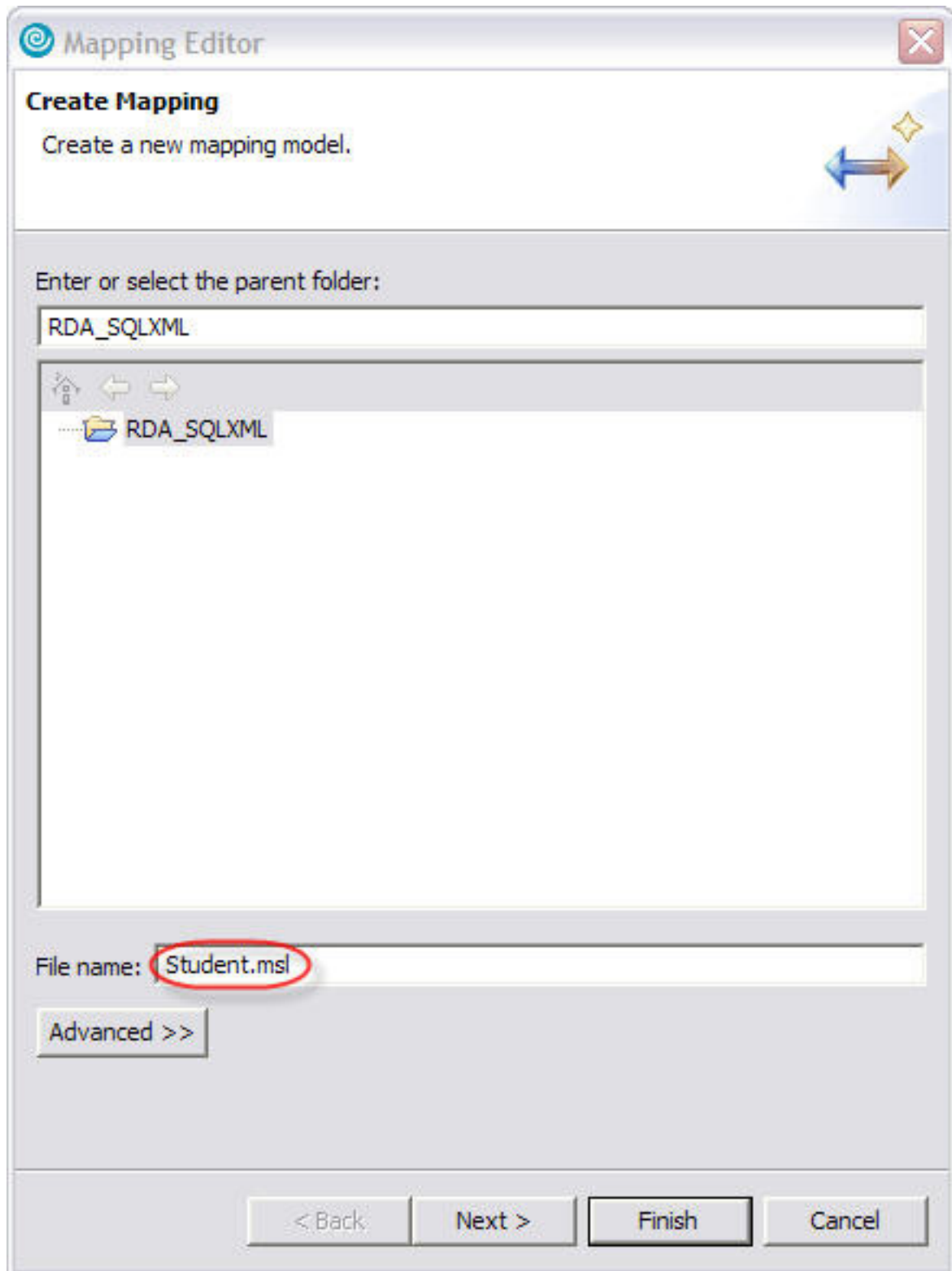
1. Change to the Data perspective
2. In the Data Project explorer, right-click on the folder **Mappings** in the tree, and select **New > Mapping Model**.

**Figure 8. New Mapping Model**



3. In the Mapping Editor wizard, specify the file name `Student.ms1`.

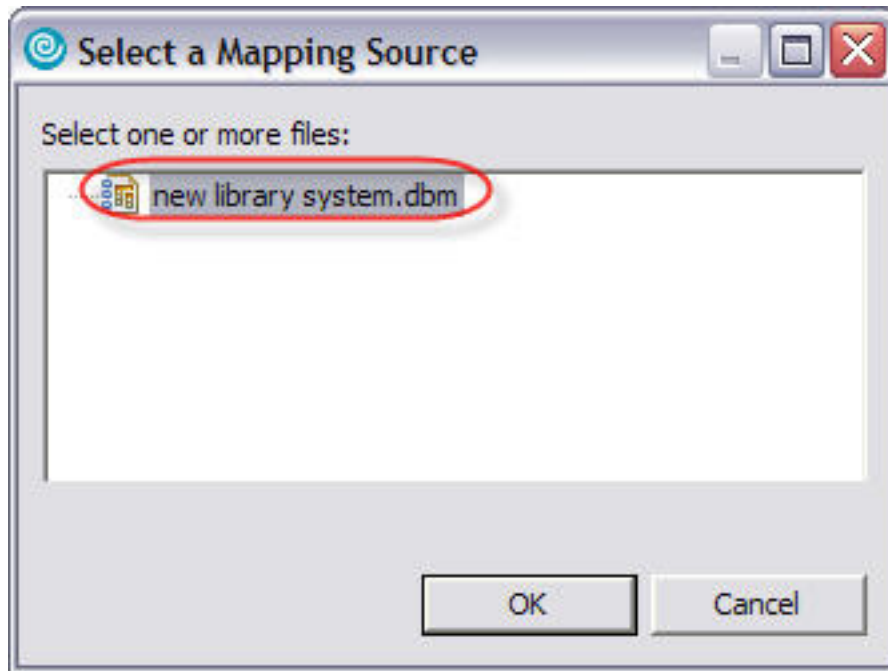
**Figure 9. New Mapping Model Wizard**



4. Click **Next**. Click **Add...** to specify a source model. Select the database

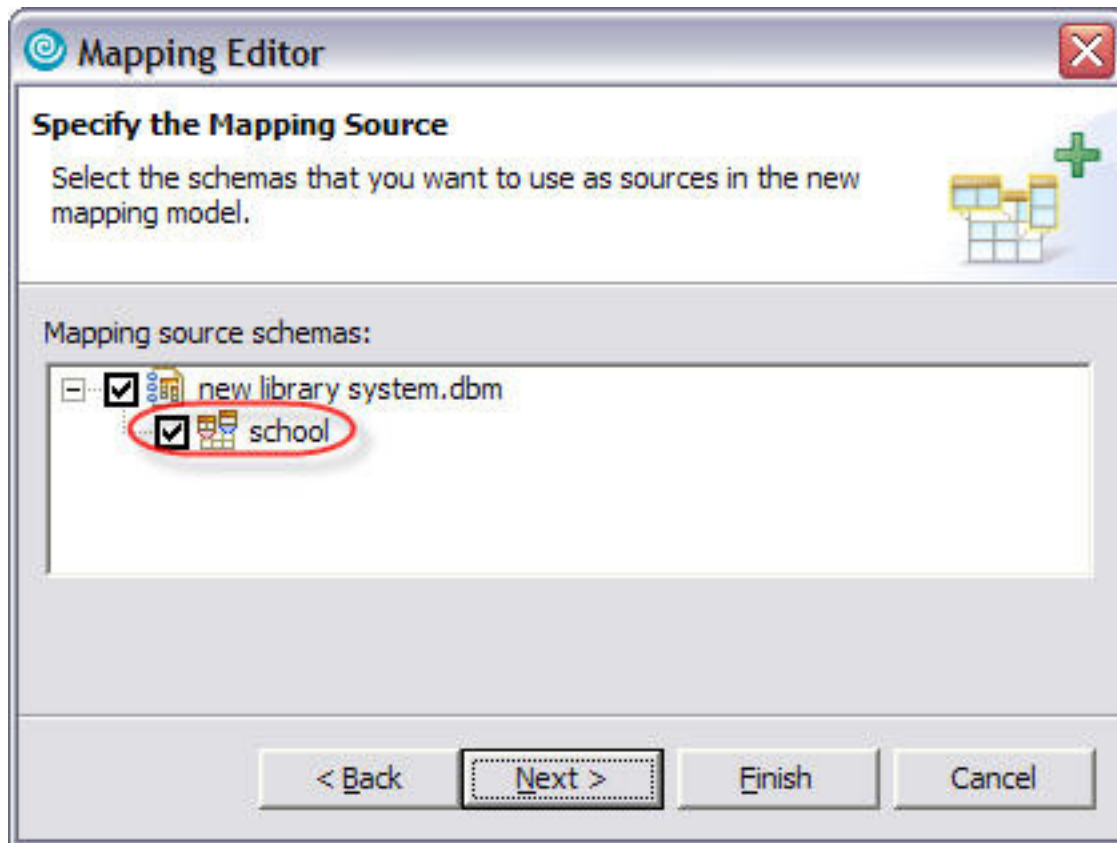
model new library system.dbm as the mapping source.

**Figure 10. Mapping source model selection**



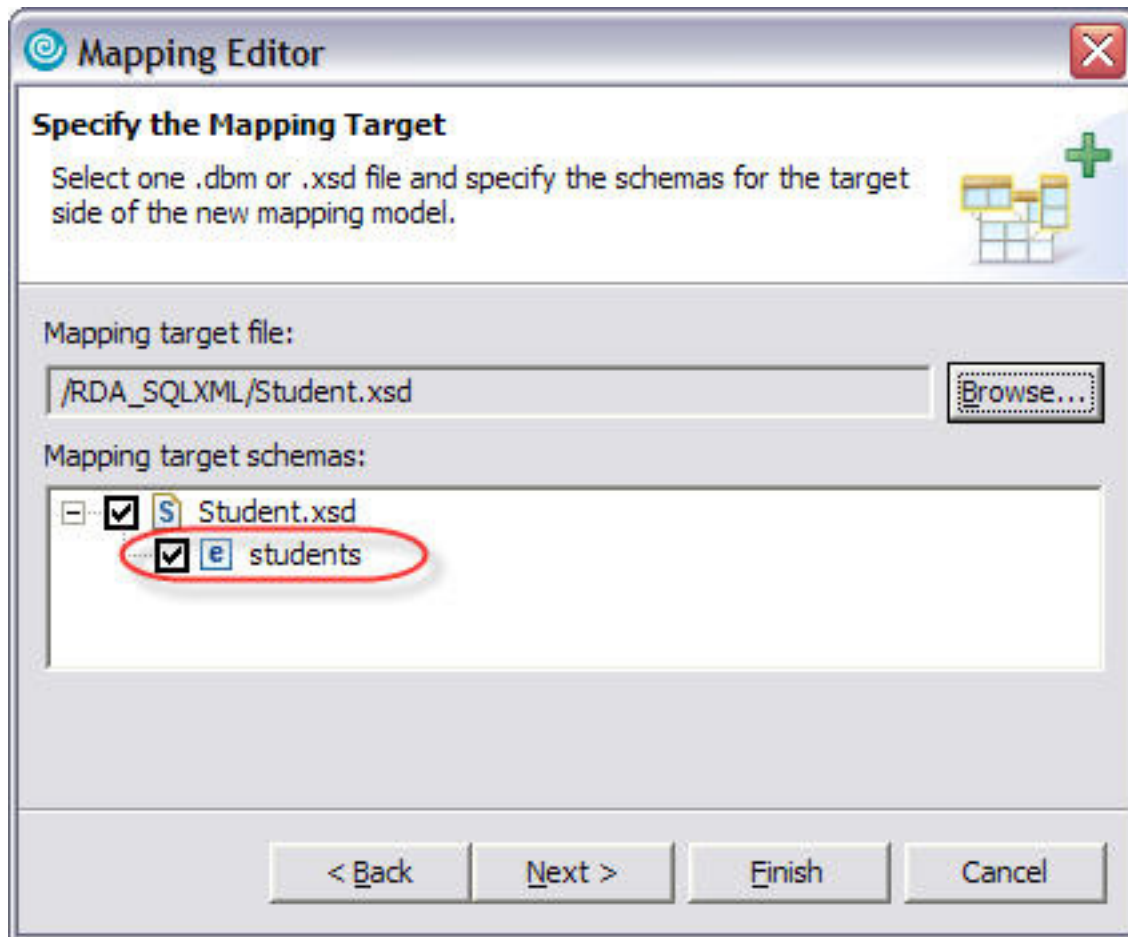
5. Click **OK**. Click **Next** in the mapping editor wizard. Make sure the schema school is selected as the source schema.

**Figure 11. Mapping Source schema selection**



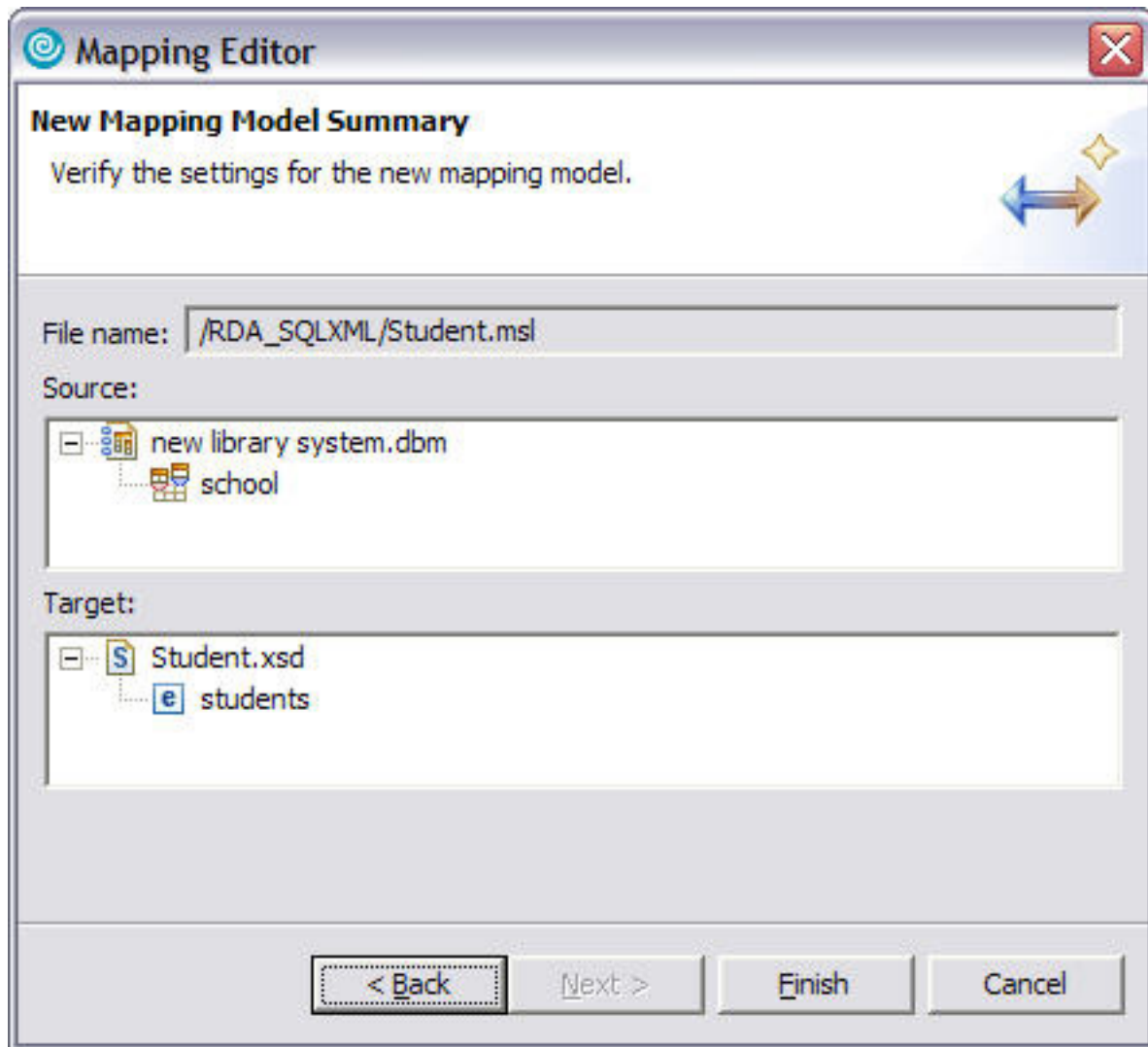
6. Click **Next**. Click the **Browse...** button to specify the XSD document Student.xsd as the target document. Click **OK**. Make sure that the schema students is selected as the target schema.

**Figure 12. Mapping target schema selection**



7. Click **Next** and verify the source and target settings in the mapping model summary.

**Figure 13. Mapping model summary**

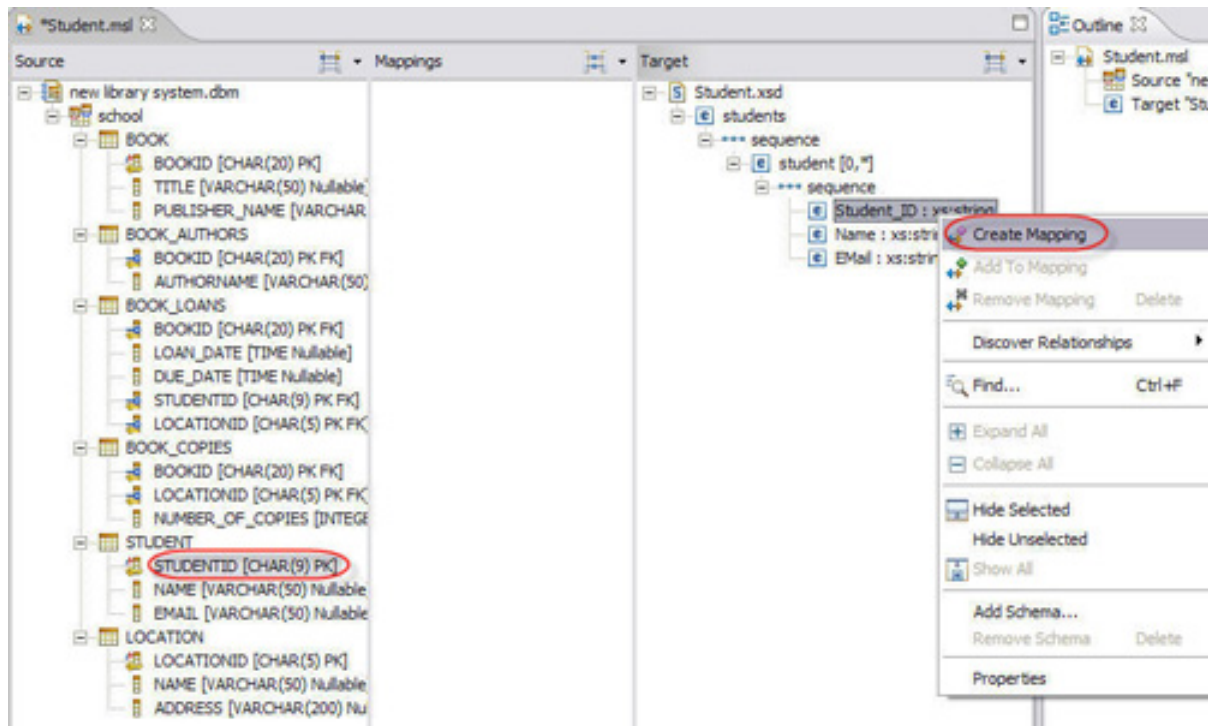


8. Click **Finish**. Now you can see the mapping editor with the selected source and target schemas.

## Adding mappings to the mapping model

1. Now let's add some mappings. Select the column element **STUDENTID** under the table element **STUDENT** on the source side. On the target side, right-click on the XML element **Student\_ID** and choose **Create Mapping** from the context menu. This creates a blue mapping line with a little box in the center. The mapping line always points from the source element to the target element.

**Figure 14. Defining a single mapping**

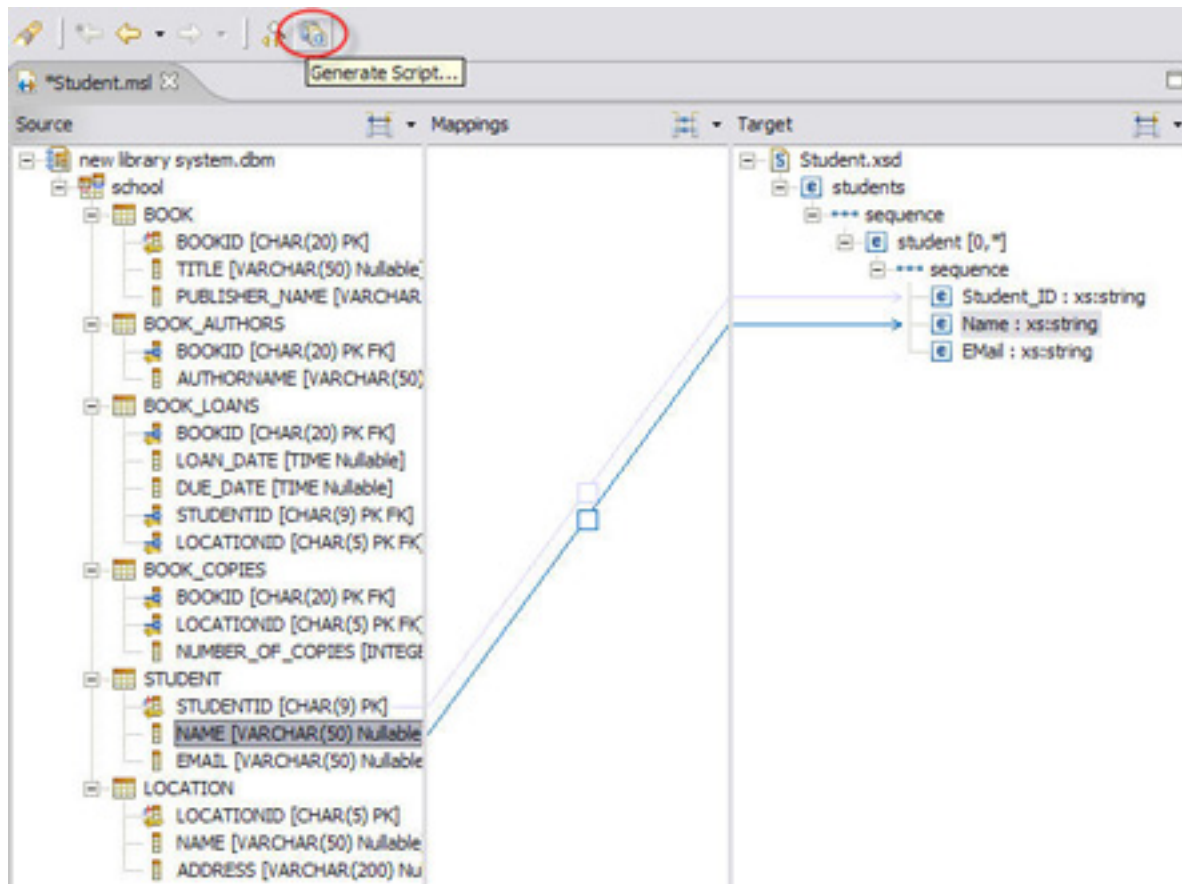


2. Use the same method to create mappings from the column NAME in the STUDENT table to the XML element Name.

## Section 4. Generating the SQL/XML query

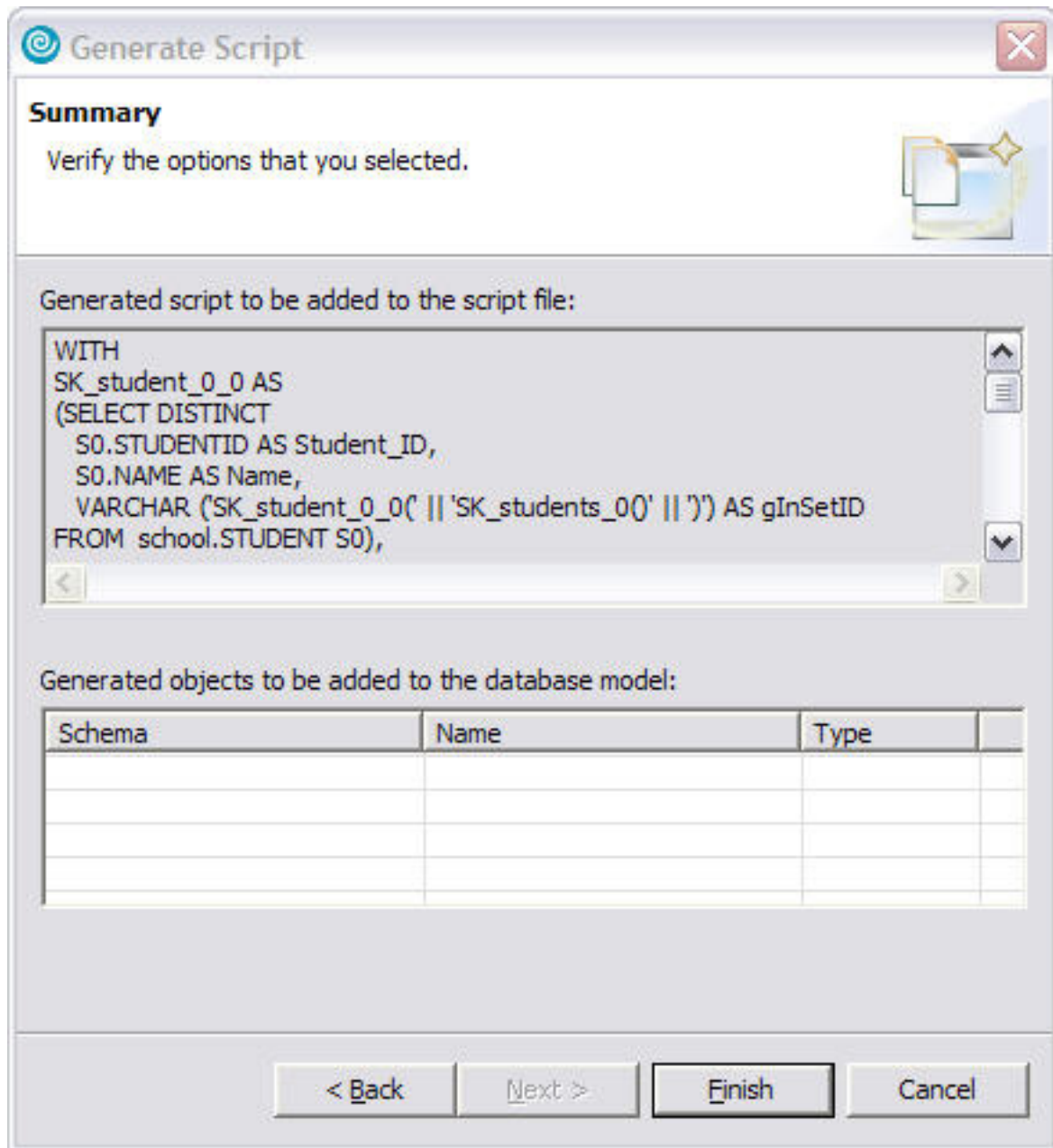
1. Click on the **Generate Script...** button in the toolbar.

**Figure 15. Generate script button**



2. When you are asked to save resources, click **OK** to save your changes to the mapping model. Unsaved changes are not considered during SQL/XML query generation.
3. In the first page of the Generate Script Wizard, accept all default values, including the file name Student.sql, and click **Next**. The summary page shows a preview of the query and a list of generated objects to be added to the database model, if any.

**Figure 16. Generate Script Wizard summary**



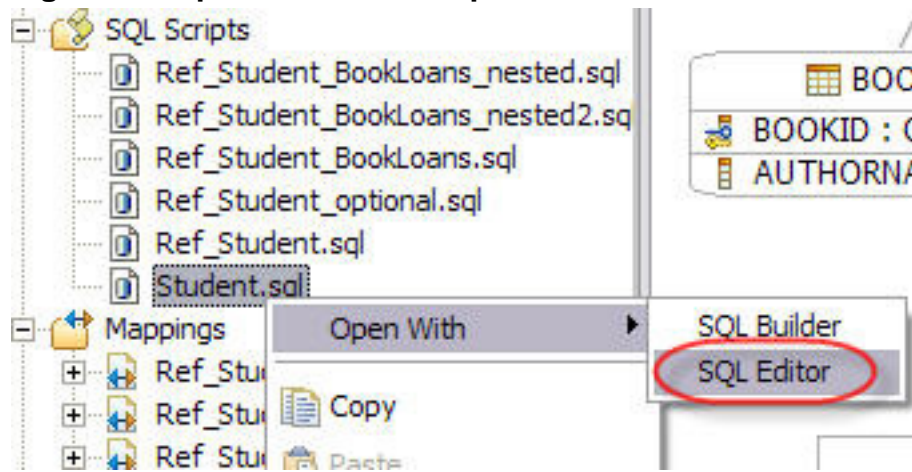
4. Click **Finish**.

---

## Section 5. Execute the generated SQL/XML query

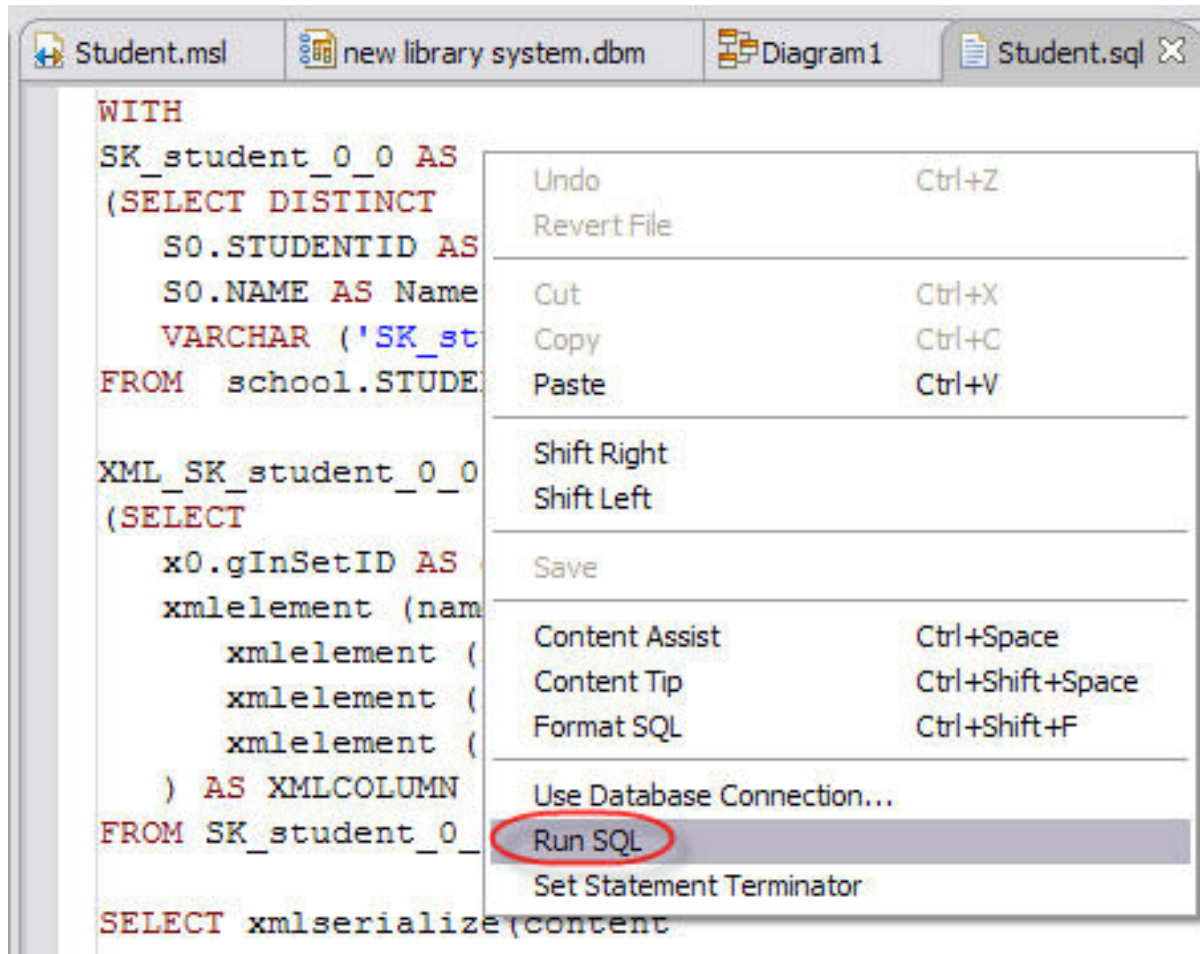
1. Open the generated query Student.sql with the SQL Editor.

**Figure 17. Open SQL/XML script with SQL Editor**



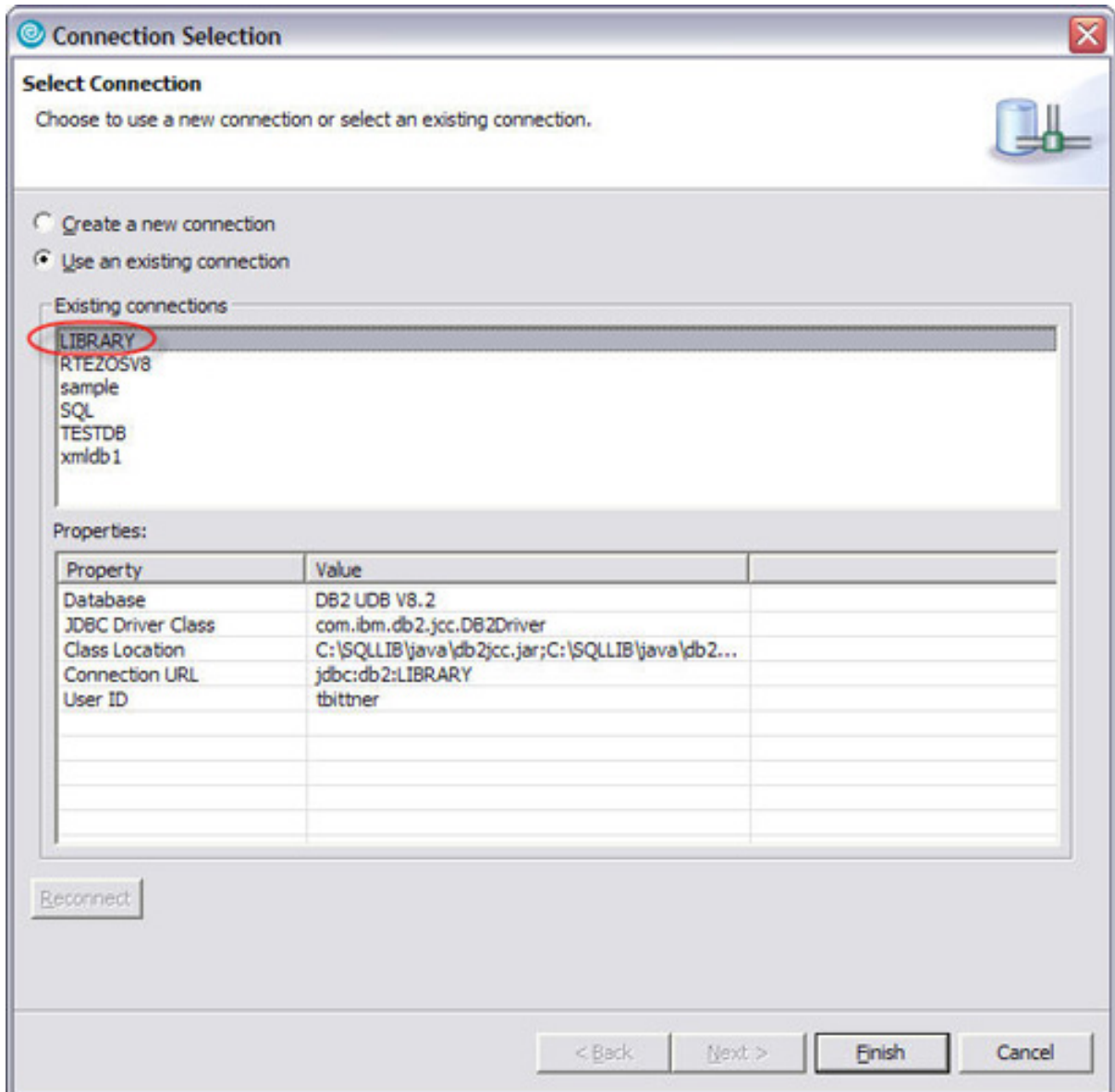
2. Right-click in the SQL Editor text window, and select **Run SQL**.

**Figure 18. Running SQL/XML script**



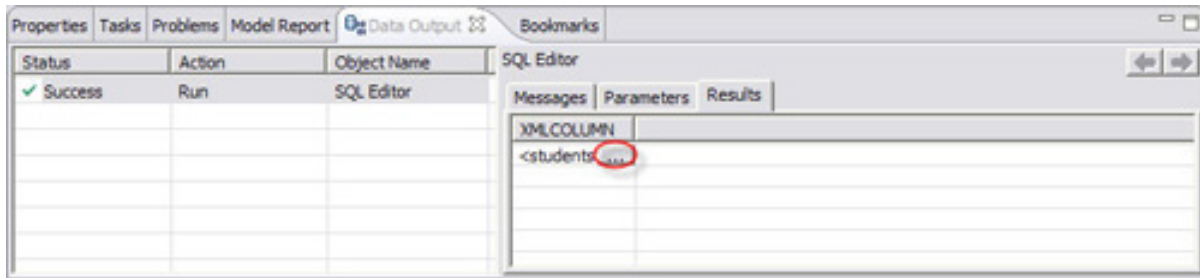
- From the list of existing connections, select the **LIBRARY** connection. (Note: You might have to click the Reconnect button, if you previously disconnected from the database.)

**Figure 19. Selecting database connection for SQL/XML query execution**



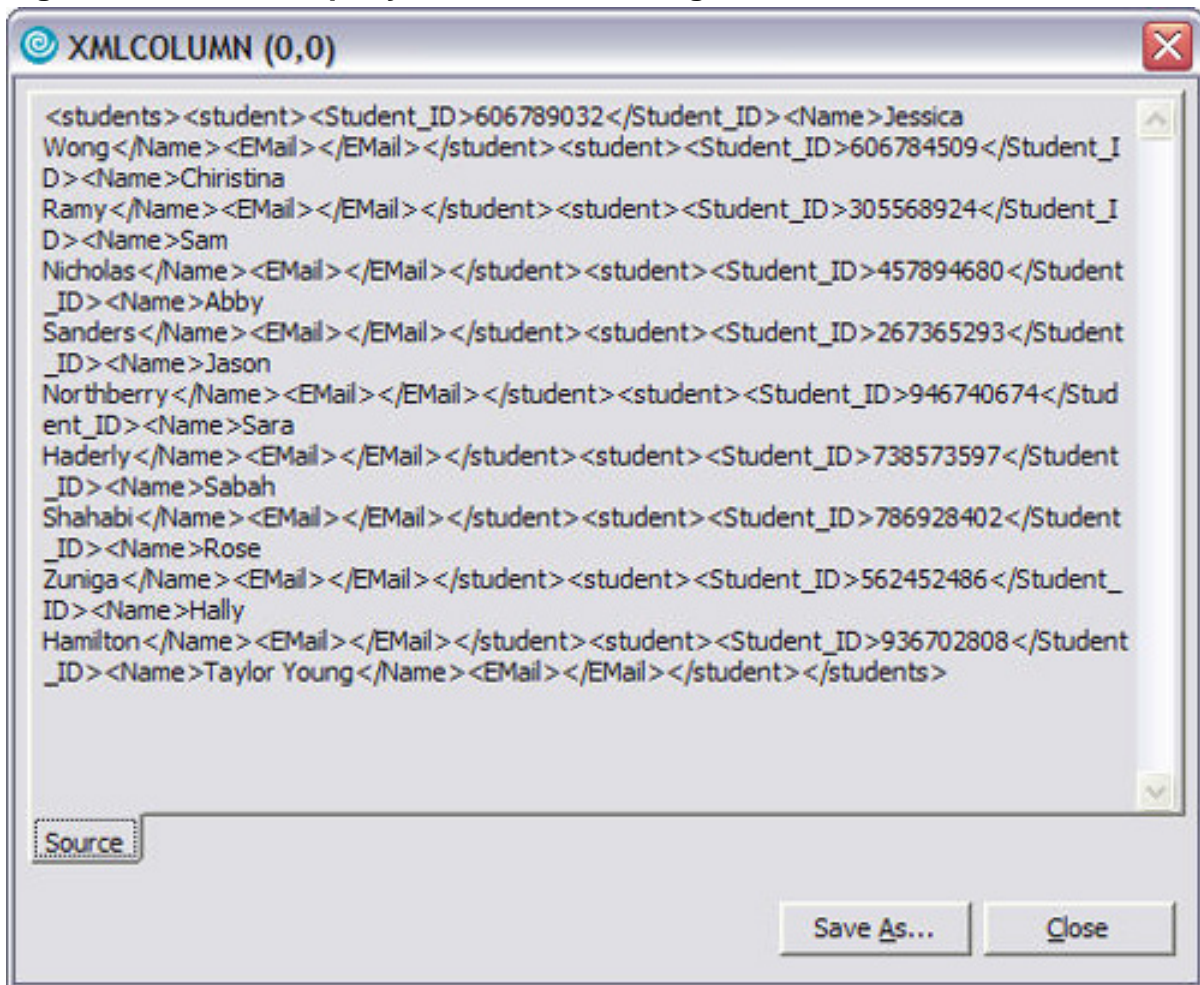
4. Click **Finish**. The result of the query shows up in the Data Output View that is located in the lower, right-hand corner. Click on the ellipsis button (...) next to <students>.

**Figure 20. SQL/XML query execution result**

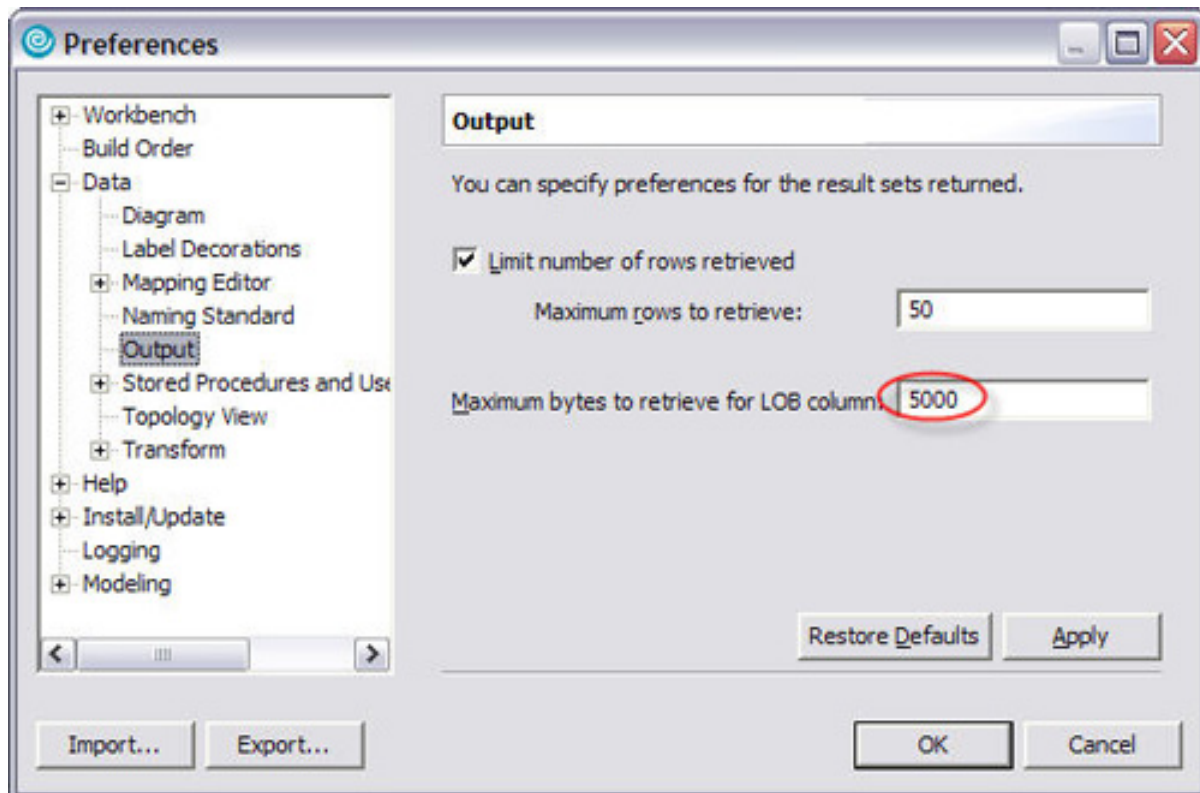


5. This opens up the XML data that is extracted from the relational database LIBRARY. (**Note:** If the text of the XML document is cut off before the document ends, increase the preference value for Maximum bytes to retrieve for LOB column.) Go to the menu **Window > Preferences**. Navigate to the category **Data**, then to **Output**, and change the value to 5000. Click **OK**. Rerun the script.

**Figure 21. SQL/XML query execution resulting XML data**



**Figure 22. Preference page settings for maximum bytes**



6. The XML data that is returned from the database does not contain any line breaks. This makes it harder to read the document. You can use the "Save as" button to save the XML data in a file and properly format it using a text editor. For this tutorial, a properly formatted XML result file is provided with Ref\_Student\_result.xml. **Note:** The XML result that is returned from the database is not a well-formed XML document because it is missing the required XML tag `<?xml version=1.0>` at the beginning.

Looking at the resulting Ref\_Student\_result.xml file, notice that it contains the empty XML element Email for each student element. The reason for this is that in the XSD Student.xsd, the element Email is defined as required, but in the mapping editor no source column was defined for the element Email. In the next section, see how to prevent this.

---

## Section 6. Modifying the target XML schema

Let's modify Student.xsd to make the element EMail optional and regenerate the SQL/XML query.

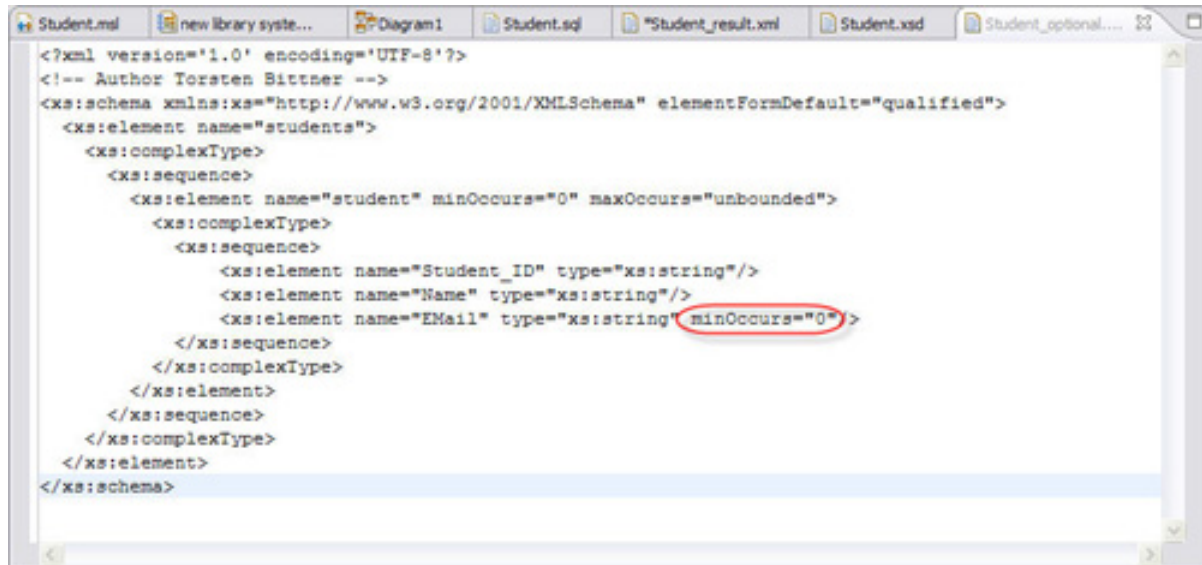
1. Open the XML schema Student.xsd with the Text Editor.

**Figure 23. Open Student.xsd with text editor**



2. Modify Student.xsd, as shown in Figure 24 (add the attribute minOccurs="0" to the element EMail). Then save it, replacing the original file.

**Figure 24. Modify Student.xsd**



4. Regenerate the query, and name the file `Student_optional.sql`.
  5. Execute the query. Open the resulting XML data. Notice that the element `EEmail` does not show up anymore.
- 

## Section 7. Joining data

The SQL/XML query defined in the first example was fairly simple. In order to exploit more of the capability that SQL/XML offers, let's join data using the following example.

1. Create a new mapping model (as shown in [Figure 8](#)).
2. A faster way to specify a mapping source and target model is to add them to the model using drag and drop. In the first page of the mapping editor wizard, enter the file name `Student_BookLoans.msl`, and press **Finish**.

### Figure 26. New mapping model `Student_BookLoans.msl`

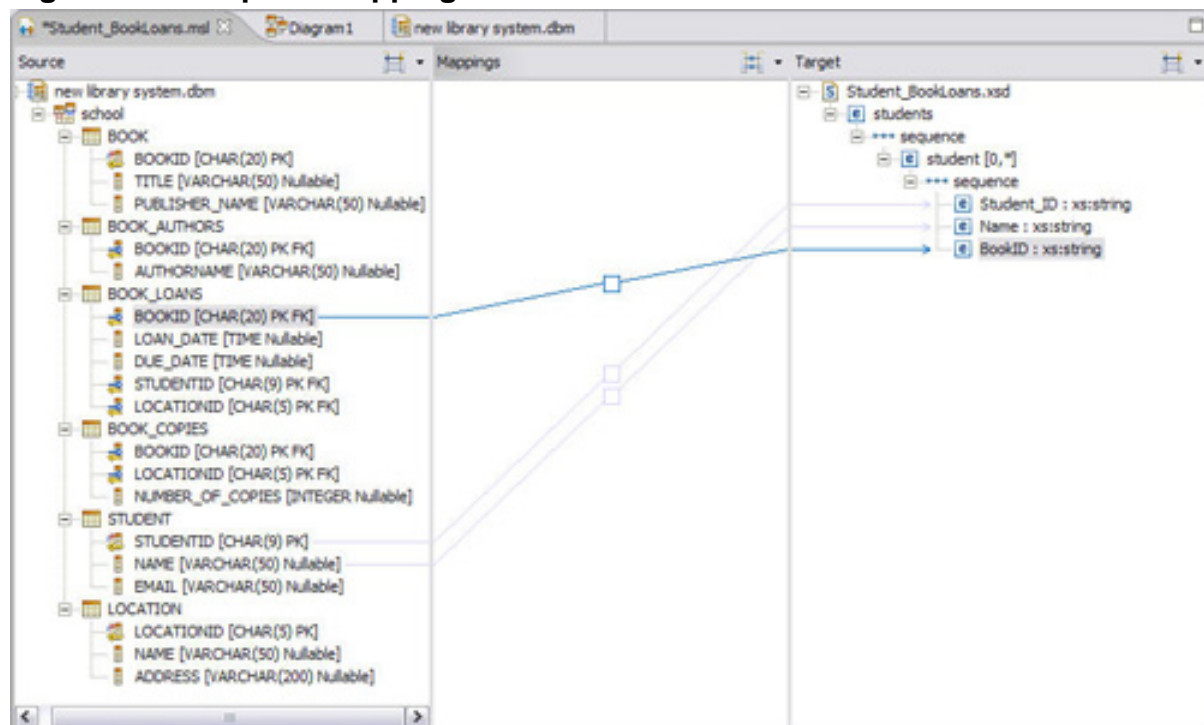


3. Now that you see the mapping editor with an empty source and target

pane, drag the file new library system.dbm from the Data Project Explorer folder Data Models and drop it into the source pane.

4. Drag the file Student\_BookLoans.xsd from the Data Project Explorer folder XML Schemas and drop it into the target pane.
5. Create mappings from the column STUDENTID in the table STUDENT to the XML element Student\_ID, column NAME in the table STUDENT to the XML element Name, and column BOOKID in the table BOOK\_LOANS to the XML element BookID.

**Figure 27. Complete mapping Student\_BookLoans.msl**



6. Generate the SQL/XML query and save it as Student\_BookLoans.sql.
7. Execute the query.

Looking at the resulting Ref\_Student\_BookLoans.xml ([Listing 1](#)), notice that the BookID values contained in each student element correspond to the BookID of the BOOK\_LOANS table. This shows that the data has been joined properly between the STUDENT and BOOK\_LOANS tables, generating a list of students and their corresponding loaned books. Note that the students Jessica Wong and Rose Zuniga show up twice. This is because both of them borrowed two books.

### **Listing 1. XML data resulting from SQL/XML query Student\_BookLoans.sql**

```

<students>
  <student>
    <Student_ID>606789032</Student_ID>
    <Name>Jessica Wong</Name><BookID>0-06-0522003      </BookID>
  </student>
  <student>
    <Student_ID>738573597</Student_ID>
    <Name>Sabah Shahabi</Name>
    <BookID>0-06-0522233      </BookID>
  </student>
  <student>
    <Student_ID>946740674</Student_ID>
    <Name>Sara Haderly</Name>
    <BookID>0-06-0522233      </BookID>
  </student>
  <student>
    <Student_ID>267365293</Student_ID>
    <Name>Jason Northberry</Name>
    <BookID>0-06-0782005      </BookID>
  </student>
  <student>
    <Student_ID>786928402</Student_ID>
    <Name>Rose Zuniga</Name>
    <BookID>0-23-7522030      </BookID>
  </student>
  <student>
    <Student_ID>457894680</Student_ID>
    <Name>Abby Sanders</Name>
    <BookID>0-26-0522003      </BookID>
  </student>
  <student>
    <Student_ID>606789032</Student_ID>
    <Name>Jessica Wong</Name>
    <BookID>0-39-8960431      </BookID>
  </student>
  <student>
    <Student_ID>786928402</Student_ID>
    <Name>Rose Zuniga</Name>
    <BookID>0-39-8960431      </BookID>
  </student>
</students>

```

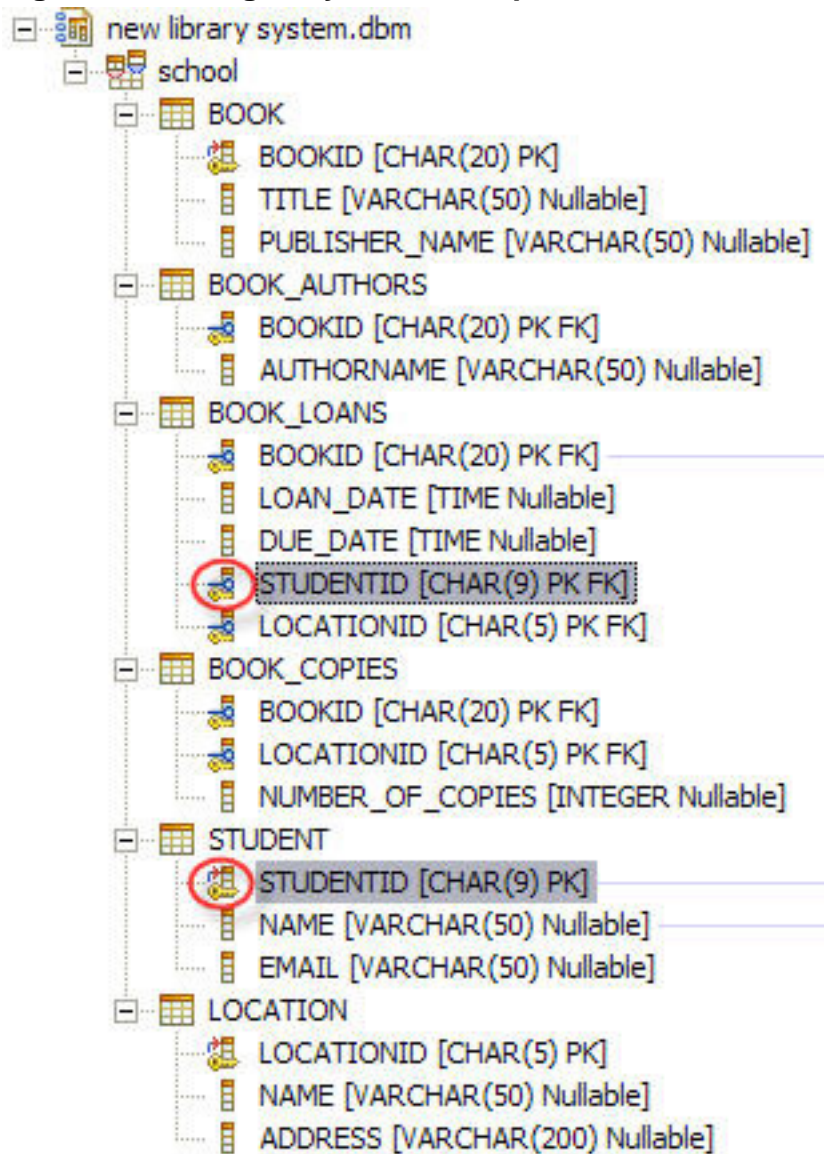
In the next section, you can look at joining data in further detail.

---

## Section 8. Working with mapping groups

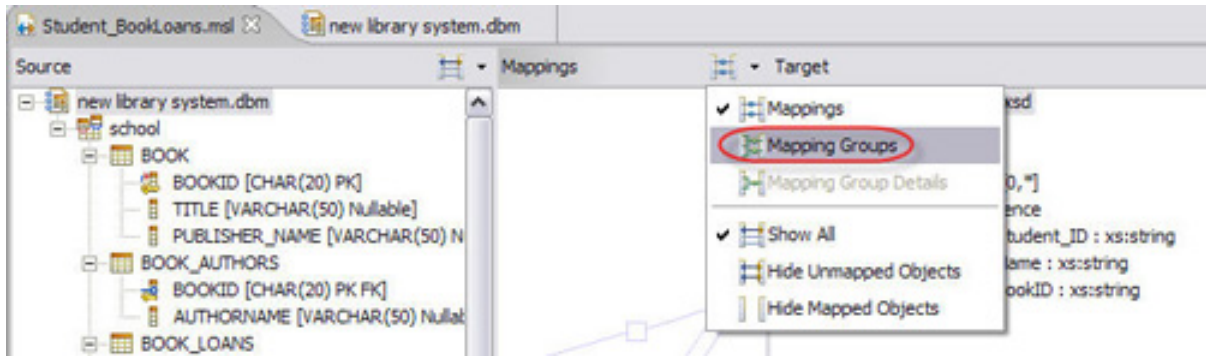
When mapping the column `BOOK_LOANS.BOOKID` to the XML element `BookID`, the RDA mapping editor automatically created a join between `BOOK_LOANS.STUDENTID` and `STUDENT.STUDENTID`.

The question is why this join was created automatically. The reason is that there is a foreign key relationship defined between these two columns in the database model `new library system.dbm`. The mapping editor visualizes the relationship with the key-icons next to the column names.

**Figure 28. Foreign key relationship between STUDENTID columns**

1. In order to see the join, you have to switch the mapping editor to view Mapping Groups, as shown in [Figure 29](#).

**Figure 29. Switch to mapping groups view**

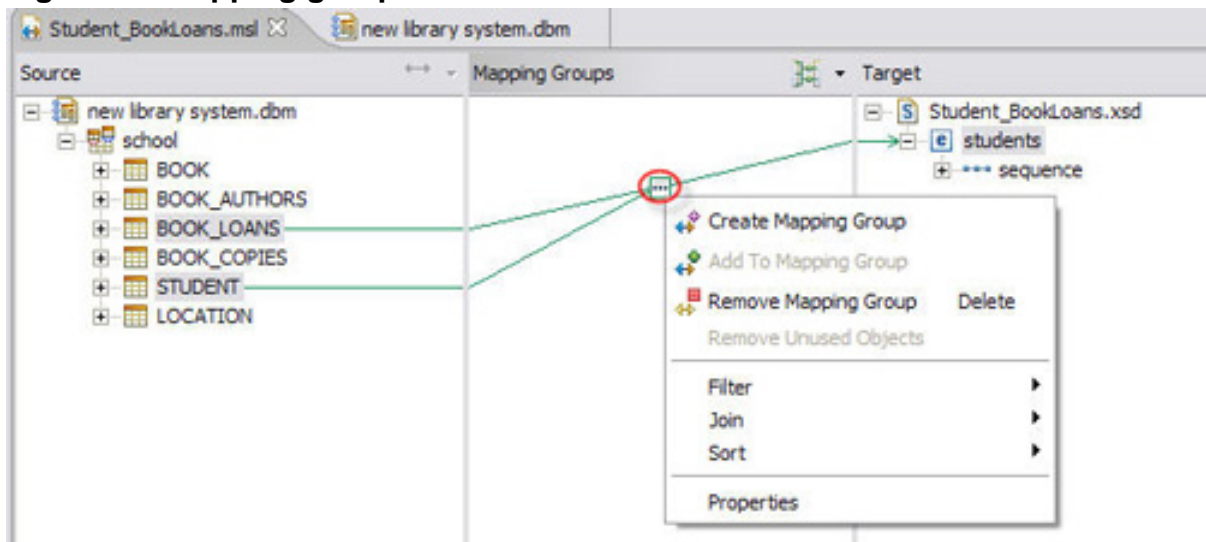


Once you've done that, you'll see a smaller amount of green mapping lines. These lines point to tables that contain mapped columns on the source side (BOOK\_LOANS and STUDENT). On the target side, they point to the parent element of mapped elements (students).

The mapping group view has the following functionalities:

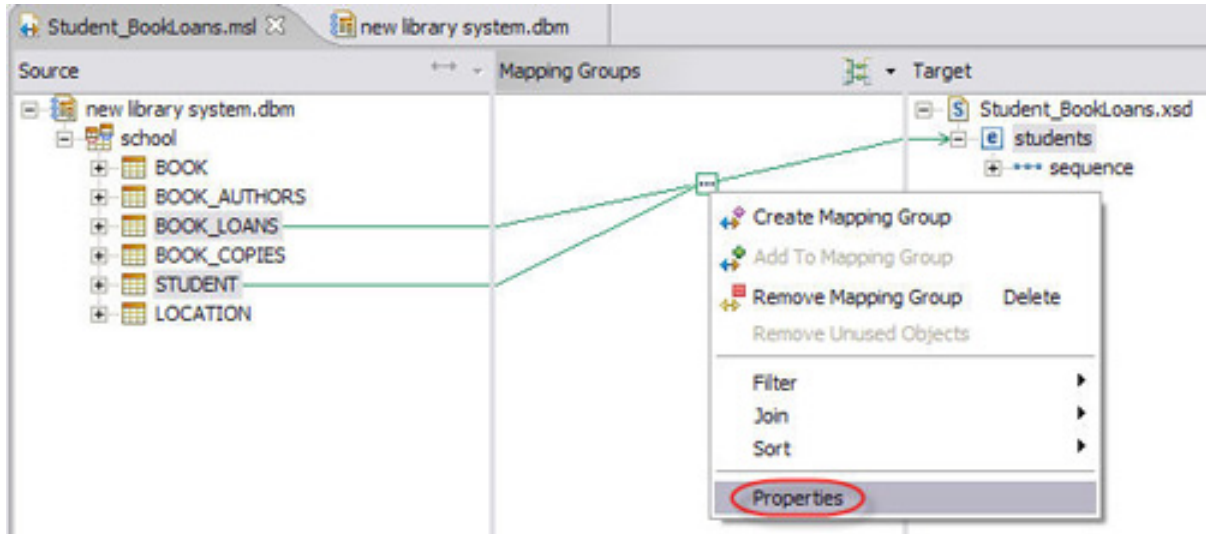
- Reduced amount of mapping lines -- You can look at the mappings you defined on a higher level (relational table level to XSD parent element level, for example)
  - Adding/removing joins between tables
  - Adding/removing filters
  - Adding/removing sorts
2. To perform these action, right-click on the green box in the middle of the mapping group line.

**Figure 30. Mapping group actions**



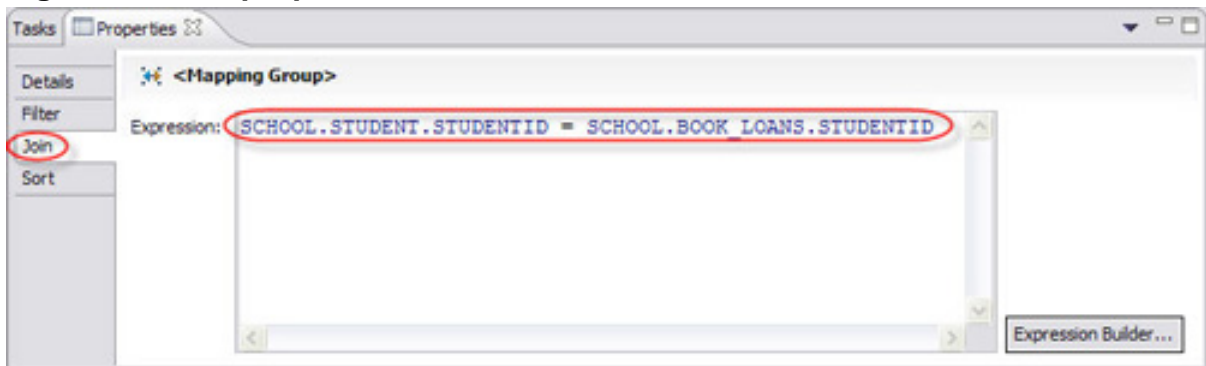
3. The Properties view also shows which of these refinements are associated to the mapping. In order to see the defined join, right-click on the green box, and select **Properties**.

**Figure 31. Mapping group properties**



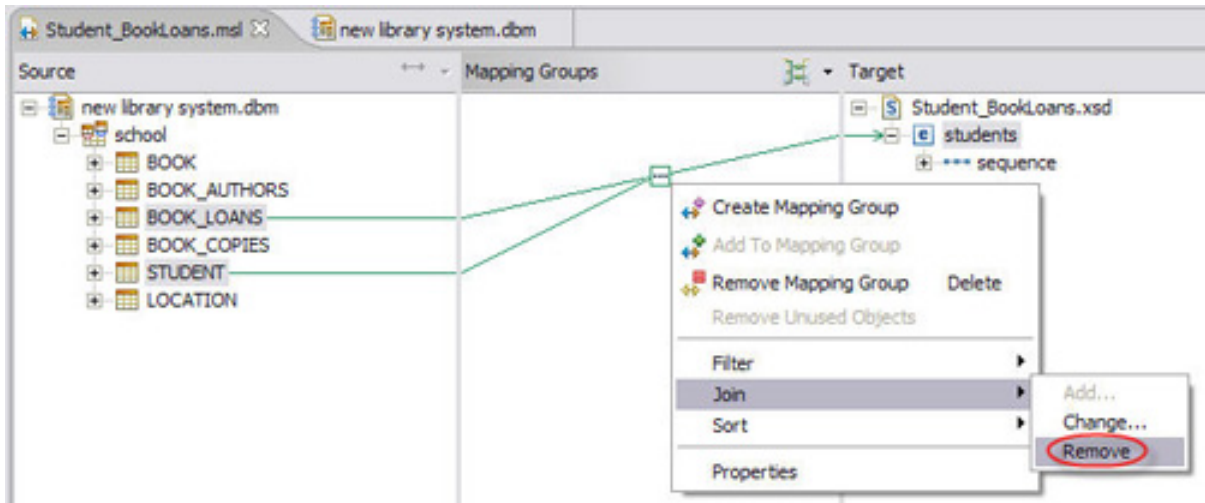
4. Select the tab **Join**. Notice the join that has been defined automatically because of the foreign key relationship between the STUDENTIDs.

**Figure 32. Join properties for STUDENTID**



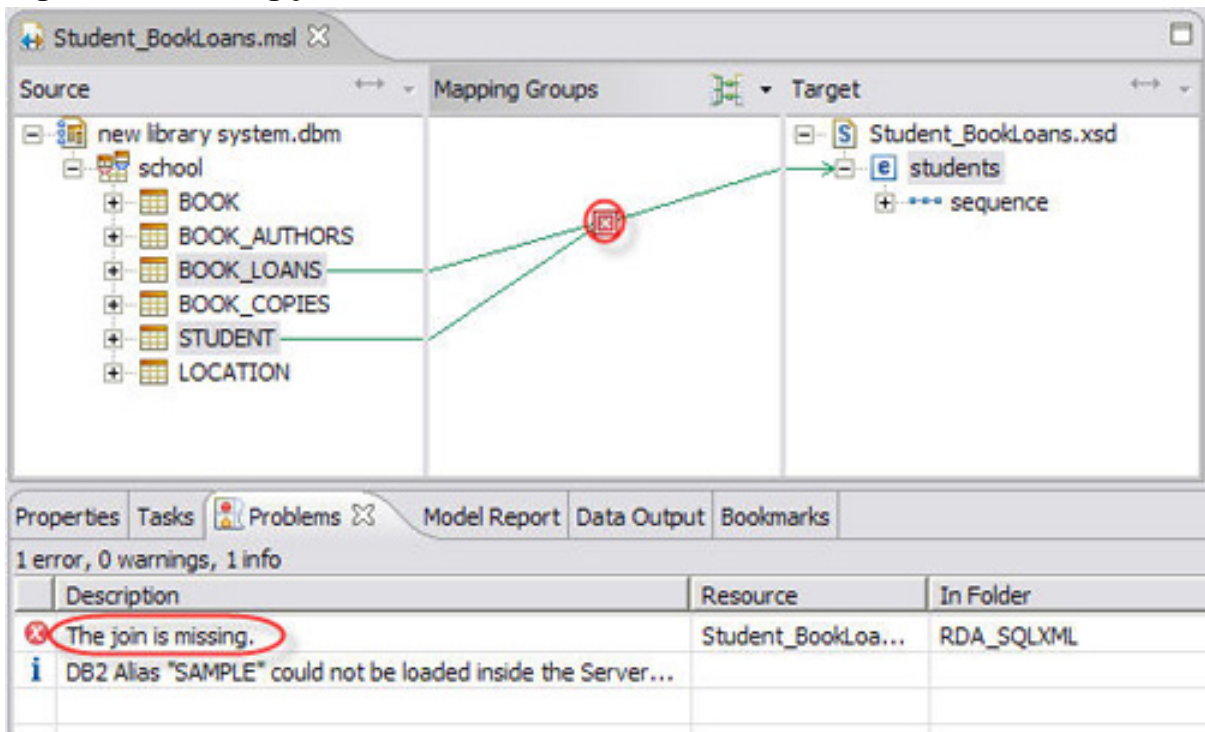
5. In order to remove the join, right-click on the green mapping group box, and select **Join > Remove**. Notice that the three dots in the mapping group box disappear.

**Figure 33. Remove the join**

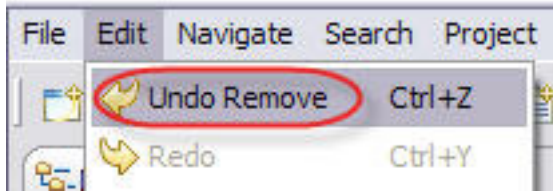


6. Save the mapping model. Notice that a red cross appears in the box, indicating that the join between the two tables is missing. It is not possible to generate the SQL/XML script without defining a join. This problem is also indicated in the Eclipse Problems view.

**Figure 34. Missing join**



7. Fix the problem of the missing join by undoing the remove action (Menu **Edit > Undo Remove**).

**Figure 35. Undoing remove join action**

8. Save the mapping model. The problem disappears from the Eclipse Problems view.

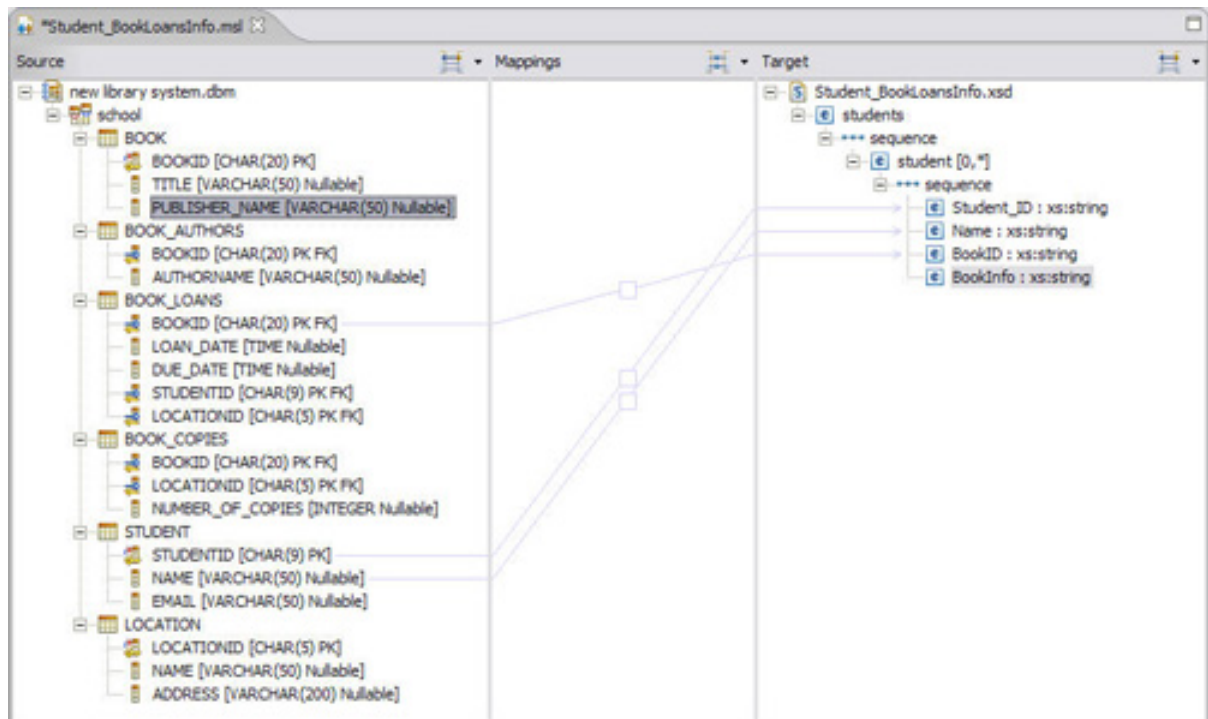
---

## Section 9. Adding transformations

When the structure of the source data is different than the structure of the target data, transformations are required. With the mapping editor, you can add transformation functions to each mapping line. The following example shows how to add a string concatenation function to the mapping model.

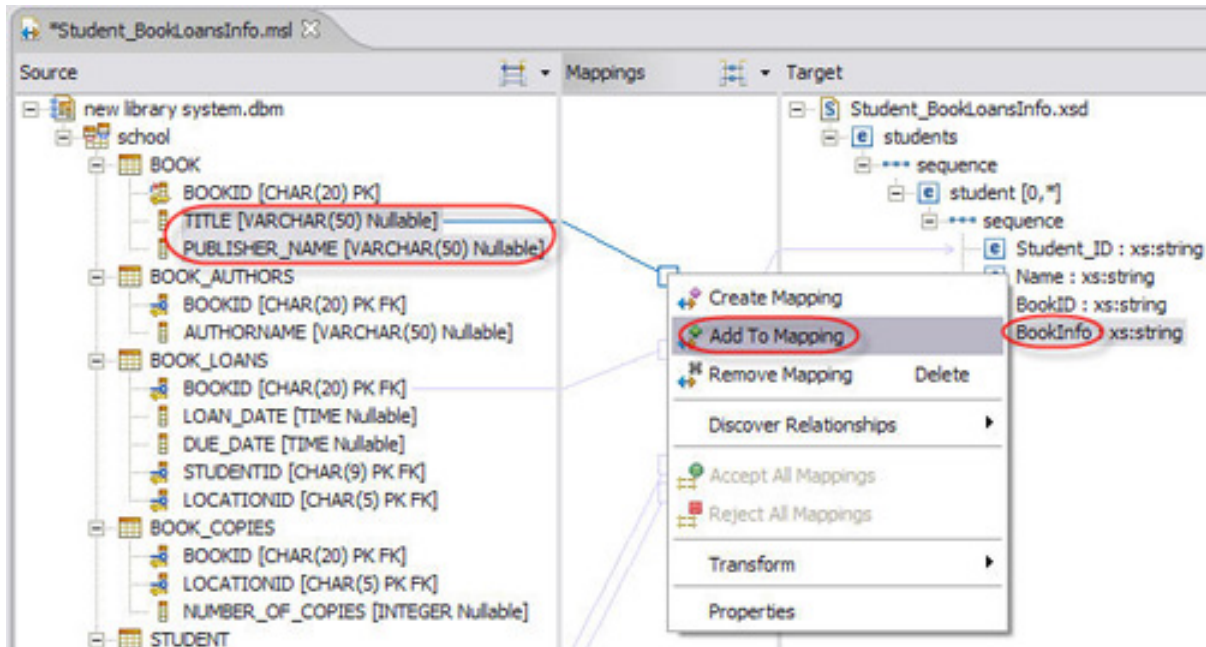
1. Create a new Mapping Model (as shown in [Figure 8](#))  
Student\_BookLoansInfo.msl with new library system.dbm as the source and Student\_BookLoansInfo.xsd as the target.
2. Create mappings from STUDENT.STUDENTID to the element Student\_ID, BOOK\_LOANS.BOOKID to BookID, and STUDENT.NAME to Name.

**Figure 36. Mapping of BookInfo**



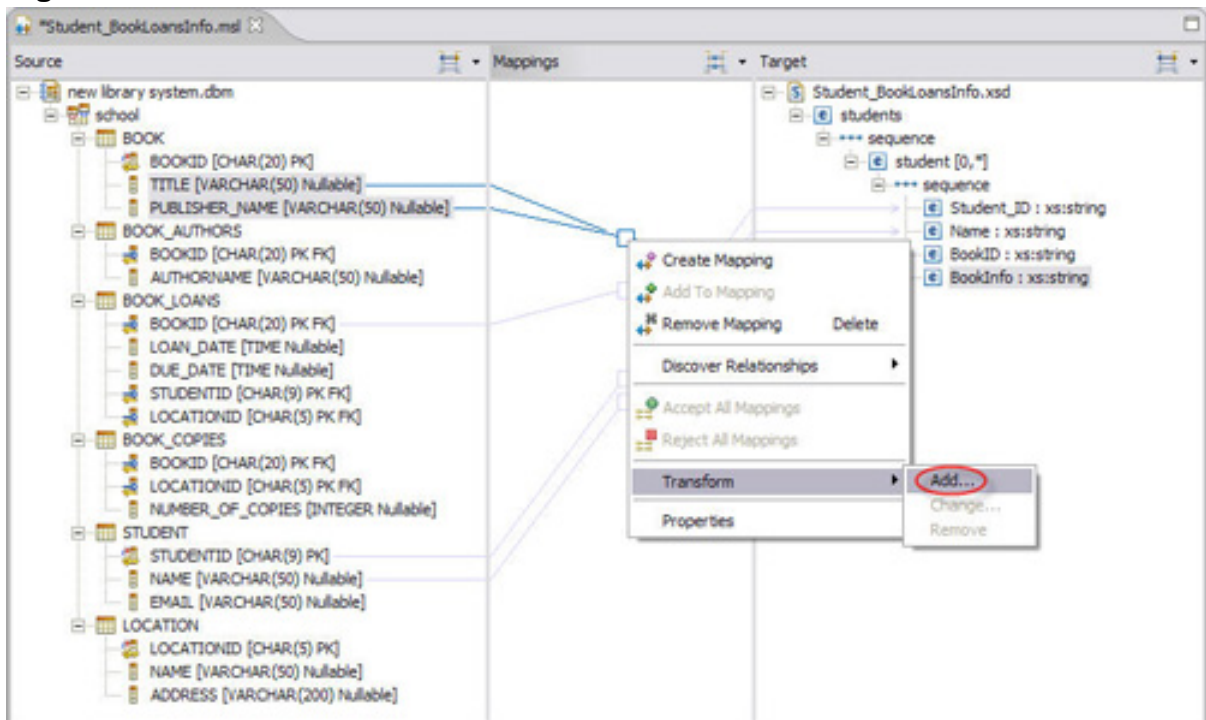
3. You want to use the target XML element BookInfo to store book title and the publisher name. In order to do so, create a mapping from BOOK.TITLE to BookInfo.
4. Select the mapping line from BOOK.TITLE to BookInfo, and also select the column BOOK.PUBLISHER\_NAME on the source side (keep the CTRL key pressed when selecting the element).
5. Right-click on the mapping line, and select **Add to Mapping** from the context menu.

**Figure 37. Add PUBLISHER\_NAME to mapping**



6. This will result in two mapping lines connecting from the source to the target element. Right-click the mapping line, and select **Transform > Add** from the context menu.

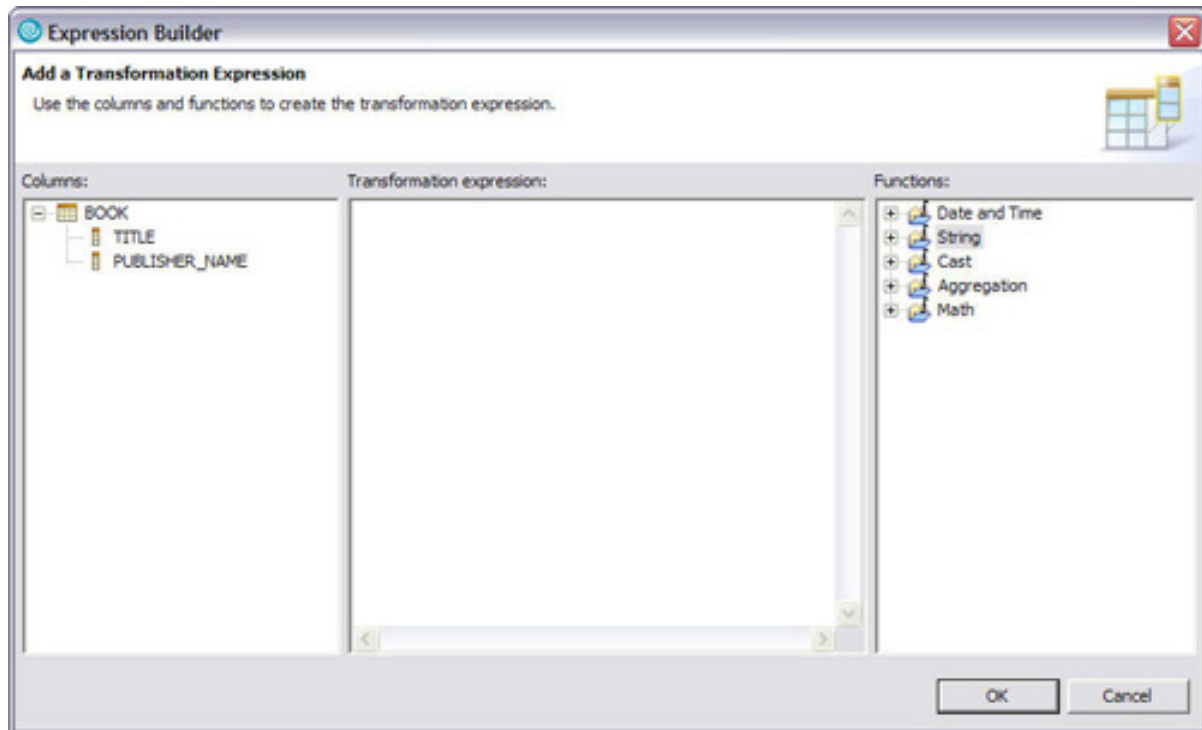
**Figure 38. Add transformation**



7. The Expression Builder dialog comes up to assist with building the

transformation. It shows the source columns that are available for the transformation on the left-hand side. On the right, the Expression Builder shows a list of common SQL transformation functions. To add source columns and transformation functions to the transformation expression text box, double-click on them.

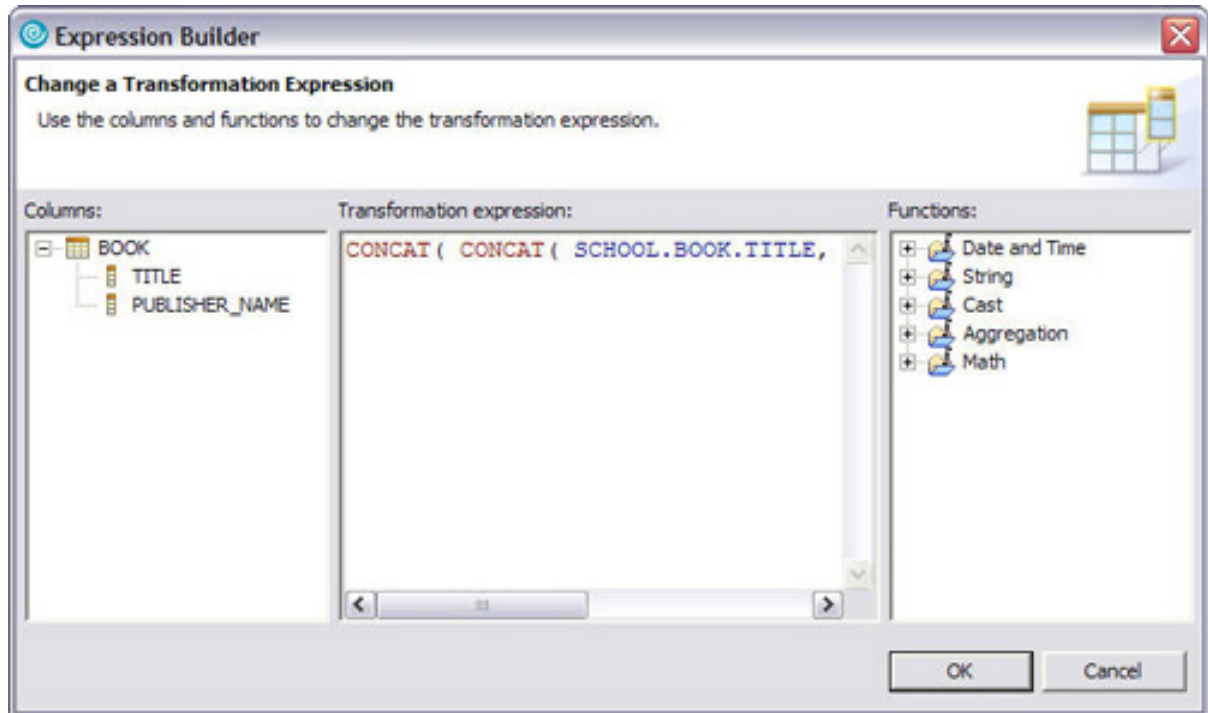
**Figure 39. Expression Builder**



8. For the BookInfo target element, you want to concatenate the title and the publisher name. In between, you want to add a forward slash (/) as a separator. For the concatenation, use the DB2 SQL function `concat(String1, String2)`. This DB2 SQL function is not a standardized SQL function -- that's why it does not show up in the list of available functions.
9. Manually add the following transformation into the transformation expression text field:

```
CONCAT( CONCAT( SCHOOL.BOOK.TITLE, ' / ' ), SCHOOL.BOOK.PUBLISHER_NAME )
```

**Figure 40. Expression Builder with transformation**



10. Generate the SQL/XML query. Note that the transformation function was added to the query.

**Figure 41. Student\_BookLoansInfo.sql with transformation**

```

WITH
SK_student_0_0 AS
(SELECT DISTINCT
  S1.STUDENTID AS Student_ID,
  S1.NAME AS Name,
  S0.BOOKID AS BookID,
  CONCAT( CONCAT( S2.TITLE, ' / ' ), S2.PUBLISHER NAME ) AS BookInfo,
  VARCHAR ('SK_student_0_0(' || 'SK_students_0()' || ')') AS gInSetID
FROM school.BOOK_LOANS S0,
  school.STUDENT S1,
  school.BOOK S2
WHERE S1.STUDENTID = S0.STUDENTID AND S2.BOOKID = S0.BOOKID),

XML_SK_student_0_0 AS
(SELECT
  x0.gInSetID AS gInSetID,
  xmlelement (name "student",
    xmlelement (name "Student_ID", x0.Student_ID),
    xmlelement (name "Name", x0.Name),
    xmlelement (name "BookID", x0.BookID),
    xmlelement (name "BookInfo", x0.BookInfo)
  ) AS XMLCOLUMN
FROM SK_student_0_0 x0)

SELECT xmlserialize(content
  xmlelement (name "students",
    Table0.XMLCOLUMN
  ) AS CLOB) AS XMLCOLUMN
FROM
(SELECT xmlagg (x1.XMLCOLUMN) AS XMLCOLUMN
FROM XML_SK_student_0_0 x1
) AS Table0;

```

- Execute the query. In the result (see also Ref\_StudentBookLoansInfo\_result.xml) you'll see that the book title and the publishers name were correctly inserted into the XML document.

## Listing 2. Query result

```

...
<student>
  <Student_ID>606789032</Student_ID>
  <Name>Jessica Wong</Name>
  <BookID>0-06-0522003      </BookID>
  <BookInfo>C++ Crash Course / Westly Publishing</BookInfo>
</student>
...

```

---

## Section 10. Summary

In this tutorial, you learned how to use Rational Data Architect to generate SQL/XML queries. You used the mapping editor to graphically define mappings from a relational source schema to an XML target schema. The mapping information was used to generate an SQL/XML query that extracts data from the relational database and populates it in XML form. You executed the generated query and analyzed the resulting XML data.

The tutorial also showed how to use mapping groups to analyze and create joins between multiple source columns. Finally, you added transformation functions to the mapping model.

Rational Data Architect can be used to quickly generate complex SQL/XML queries. The automatic query generation significantly reduces the time it takes to manually create SQL/XML queries.

## Downloads

Description	Name	Size	Download method
Package required to run through tutorial	RDA_SQLXML.zip	20KB	<a href="#">HTTP</a>

[Information about download methods](#)

# Resources

## Learn

- "[Access and integrate metadata with Rational Data Architect](#)" (developerWorks, July 2006): Understand how RDA capabilities can enhance the federation capabilities of WebSphere Information Integrator.
- "[Use Rational Data Architect to integrate data sources](#)" (developerWorks, March 2006): Explore a tool-supported process for federation design in five steps.
- [developerWorks resource page for DB2 for Linux, UNIX, and Windows](#): Read articles and tutorials and connect to other resources to expand your DB2 skills.
- [developerWorks Architecture zone](#): Get the resources you need to advance your skills in the architecture arena.

## Get products and technologies

- Download a free trial version of [Rational Data Architect](#).
- Download a free trial version of [DB2 Enterprise 9](#).
- Download a free trial version of [DB2 Enterprise Server Edition, V8.2](#).
- Now you can use DB2 for free. Download [DB2 Express-C](#), a no-charge version of DB2 Express Edition for the community that offers the same core data features as DB2 Express Edition and provides a solid base to build and deploy applications.

## Discuss

- [Participate in the discussion forum for this content](#).
- Check out [developerWorks blogs](#) and get involved in the [developerWorks community](#).

## About the author

Torsten Bittner



Torsten Bittner works as a software engineer in Information Management, IBM Software Group. He carries a diploma degree in computer science from the University of Rostock, Germany. His development responsibilities include the Rational Data Architect mapping editor discovery and the query generation component.