Bitemporal modeling with InfoSphere Data Architect, Part 2: Modeling system period temporal requirement

Balaji Kadambi
Rahul Jain

November 23, 2011

Industry regulations and competitive pressures are prompting IT departments to maintain more data for longer periods of time and to provide better ways for business users to analyze past, current, and future events. In order to achieve these goals, firms are developing time-aware applications and queries with temporal databases. Beginning with IBM® InfoSphere® Data Architect version 7.6, you can model relational databases as well as temporal relational databases. This two-part series will demonstrate how you can accelerate temporal data modeling with a DB2 for z/OS version 10 database by adopting InfoSphere Data Architect 7.6.

Introduction

In Part 1 of this article series, you learned how to use InfoSphere Data Architect version 7.6 to model basic business-period temporal data for the Sample Bank. In this continuation of the series, you will learn how to modify an existing data model in order to model bitemporal requirements, using InfoSphere Data Architect version 7.6. Bitemporal modeling provides a lot of flexibility in data design, providing you the ability to query on both system-period and business-period data.

Problem description

Now that you have modeled how to track interest rate changes during the duration of a loan, a new requirement is defined. The Central Bank issues the notification that the benchmark interest rate is changing two weeks before it becomes effective. The home loan department manually updates the loan system with the new home loan interest rate in the loan system, as well as the date that the change is effective.

However, the Central Bank specified an incorrect interest rate due to a miscalculation, so the interest rate specified at the Sample Bank is now also incorrect. As a result, the installment bill that was automatically generated for customers at the beginning of the month was incorrect. Unfortunately, the Central Bank did not notify the Sample Bank of the error more than two weeks before the installment bills were generated. The error was corrected in the Sample Bank system, but incorrect bills were generated. Shortly before the bills were sent to customers, the Sample...
Bank sent out a notice to their customers, informing them of the issue and specifying that the difference would be refunded in the bill that will be issued the next month.

Naturally, some borrowers contacted the home loan department with questions about the incorrect bills. The home loan department must be able to query the system and find the correct interest rate for the borrower when the bill was generated in order to tell customers how much they would be refunded in the bill that is generated in the next month.

To address this new requirement, you will modify the LOAN_INTEREST table in order to track both system-period and business-period data, which allows customer service representatives to query for the data that they need.

You can modify the LOAN_INTEREST table in either of the following ways.

1. Modify the Loan_Model.dbm physical data model that you developed in part 1.
2. Reverse-engineer the database to create, then modify a new model.

**Note:** The following section will briefly describe how to reverse-engineer a database so that you can make changes to existing data sources. If you would like to skip this section, [click here](#).

**Reverse engineer the loan model from the database**

Follow these steps in order to reverse-engineer the loan database to create a new physical data model in the Data Project Explorer.

1. From the main menu, click **File > New > Physical Data Model**. The New Physical Data Model wizard opens.
2. On the Model File page of the wizard, specify a file name, then select the **Create from reverse engineering** option, as shown in Figure 1, then click **Next**.

**Figure 1. Model File option page**

![New Physical Data Model](image)

3. From the **Source** page, select the **Database** option, as shown in Figure 2, then click **Next**.
4. From the **Select Connection** page, create a new connection by clicking the **New** button or select an existing connection as shown in Figure 3, then click **Next**.

**Figure 3. Select Connection page**

5. From the **Select Objects** page, select **LOAN_SCHEMA** as shown in Figure 4, then click **Next**.

**Figure 4. Select Objects page**
6. From the **Database Elements** page, ensure that all of the database element check boxes are selected, as shown in Figure 5, and then click **Finish**.

**Figure 5. Database Elements page**

---

**Inspect the physical data model**

In order to view the physical data model with fresh eyes, you can inspect it in the Data Project Explorer. Notice the BORROWER, LOAN, and LOAN_INTEREST tables under the LOAN_SCHEMA schema, as shown in Figure 6.
Enabling system temporal behavior in a table

In the current model, the corrections that are made to records with regard to interest rates and installment amount in the LOAN_INTEREST table cannot be tracked. In order to track record-level changes in the interest rate for a loan, you should add system-period tracking to the LOAN_INTEREST table.

Perform the following steps to add system-period columns to the table.

1. Select the LOAN_INTEREST table in the Data Project Explorer. The properties of the table display in the Properties view.
2. Click the Columns tab to add system-period columns to the table. Select the System time period check box, as shown in Figure 7.
3. Review the following changes to the physical data model.
   - System-period tracking is enabled.
   - Three new columns were automatically added: SYSTEM_START_TIME, SYSTEM_END_TIME, and TRANS_ID.
   - The LOAN.INTEREST_HIST history table was automatically created, containing the same columns as the LOAN_INTEREST table.

These modifications are shown in Figure 8.
4. Click the **Versioning** tab of the LOAN_INTEREST table properties view, shown in Figure 9, to see that the new LOAN_INTEREST_HIST history table is linked to the LOAN_INTEREST table.

---

**Figure 8. Data Project Explorer view of the modified loan physical data model**
Note: The history table is automatically updated whenever the temporal table is updated, and any modifications that you make to the LOAN_HISTORY table are automatically applied to its related history table. The history table is automatically deleted when the versioning is removed on the temporal table. If you want to remove versioning for a system-period temporal table, you can select the **Do not establish versioning relationship with a history table** option.

5. Because you have specified that the LOAN_INTEREST table has both system period and business period properties, the table is considered a **bitemporal** table. In order to deploy your changes, you should compare the updated model to the source database, then generate the DDL script that you can share with your database administrator.

**Propagate changes to the database**

1. Use the Compare editor to compare and merge your changes, then generate the delta DDL.
   Right-click the **LOAN_SCHEMA** schema, then select **Compare With -> Original Source** as shown in Figure 10. If the data model was not reverse engineered, then select the **Another Data Object** option during compare and select the **LOAN_SCHEMA** object on the Data Source Explorer.
2. In the Filtering Criteria window, keep the default options and click OK to compare the model to the source database.

3. The Compare editor opens. Inspect the result of the comparison in the Compare editor to see the following differences, also shown in Figure 11.
   - The LOAN_INTEREST_HIST table is associated with the LOAN_INTEREST temporal table. This update is not present in the database.
   - The LOAN_INTEREST table has new system-period columns, which are not present in the database.

4. Specify that you want to merge the system-period properties into the source database. Select the temporalPeriods row in the Compare editor, then click the Copy from Left to Right button as shown in Figure 12.
Figure 12. Merging changes in the Compare editor

![Image of the Compare editor with merged changes]

The history table and system-period tables are copied to the right side of the Compare editor.

5. Click the **Generate Right Delta DDL** button, shown in Figure 13.

Figure 13. Generating deployable delta DDL in the Compare editor

![Image of the Compare editor with generated DDL]

6. Save the delta DDL so that you can share it with your database administrator. Your database administrator will deploy the changes for you on the database server.

**Inspect the deployed bitemporal model on the database server**

After the database administrator runs the DDL to deploy your changes, you can view the changes in the Data Source Explorer.

1. Expand the `LOAN_SCHEMA` schema to locate the new `LOAN_INTEREST_HIST` table, and then verify the changes to the `LOAN_INTEREST` table. The updated schema is shown in Figure 14.
2. When you inspect the properties of the LOAN_INTEREST table, you see that the bitemporal (system-time period and business-time period) properties are enabled, as shown in Figure 15.
3. In the **Versioning** tab, you verify that the LOAN_INTEREST_HIST table is linked to the LOAN_INTEREST table, as shown in Figure 16.

**Figure 16. Verifying the history table for the LOAN_INTEREST table**

---

**Conclusion**

In this article, you learned how to model bitemporal properties within tables. With little effort, you were able to add a system period to a table and create an associated history table. When you compared and merged changes, you generated a delta DDL script to deploy your changes to the target database.

With InfoSphere Data Architect version 7.6, you can accelerate the temporal data design and deployment process. The learning curve time is reduced via a simple interface that helps modelers to understand database-specific features. It's easy to deploy your temporal changes with a simple delta DDL generator that places your changes into a simple script that runs on your database server.
Acknowledgements

Thanks to Erin Wilson for the valuable suggestions and the review of this article.

© Copyright IBM Corporation 2011
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)