УДК 004.9

Т.Г. Шемсединов

МЕТАМОДЕЛЬ В ЗАДАЧАХ ИНТЕГРАЦИИ ИНФОРМАЦИОННЫХ СИСТЕМ

Аннотация: В статье рассматривается интеграция прикладных информационных систем как постоянный и естественный процесс их жизненного цикла, показывается необходимость применения метамоделей в задачах интеграции и приводятся дополнительные параметры метамоделей, которые обычно не формализуются, или задаются неявно в реляционных и объектных моделях.

Ключевые слова: метамодель, информационные системы, задачи интеграции.

Введение

В последнее время, в начале разработки любой прикладной информационной системы, вопрос интеграции в уже имеющуюся сложную инфраструктуру предприятия стоит острее, чем даже вопрос функционала и производительности. Интеграция стала одним из основных факторов успеха в любой сфере бизнеса, а разработка превратилась из одноразового явления в постоянный процесс, в естественное состояние системы. Стало очевидно, что информационные системы двигаются от статической модели предметной области к динамической модели, изменяющейся в реальном времени. С другой стороны, распространение сетевых средств коммуникации позволило компаниям обмениваться информацией в цифровой форме с партнерами, клиентами, подрядчиками и даже государственными органами [1,2]. В таких условиях, старые методологии разработки, архитектуры, решения и технологии реализации нуждаются в серьезной модернизации.

Постановка задачи

Задачу интеграции информационных систем можно разбить на три составляющих: интеграция внутри предприятия, межкорпоративная интеграция и интеграция в реальном времени. Если первый пункт достаточно проработан и понятен, хотя и не прост, то уже второй приводит к целому ряду проблем: гетерогенность платформ и операционных сред, различия в архитектурах ИС предприятий, различия в подходах к моделированию данных, противоречия в принципах сетевого взаимодействия, несовместимость средств визуализации и, в конце концов, отсутствие понимания между разработчиками и общепринятого подхода к интеграции, постоянные войны стандартов между ведущими ИТ-компаниями. Все это не способствует решению поставленных задач [3,4].

Второй аспект проблемы заключается в том, что разработка информационных систем обычно начинается с одного из слоев: представления, логики или данных (то есть, постановка задачи базируется на описании пользовательского интерфейса, бизнес-логики и процессов или же

структуры базы данных). А ни один из слоев не описывает предметную область с достаточной полнотой, обеспечивающей динамическое связывание нескольких ИС. Дело в том, что и логика, визуальное представление и, тем более, структуры данных, в межкорпоративном взаимодействии очень динамичны и могут меняться не только в процессе проектирования и разработки информационных систем, а и во все время их ежедневной эксплуатации. Интеграция ИС на одном из уровней, и при неполном описании, приводит к тому, что изменение этого уровня в одной из систем, рушит все объединяющие связи. Поэтому, для интеграции ИС необходимо использовать описание на более высоком уровне, например, с использованием метаданных. Таким образом, мы приходим к необходимости интеграции на уровне метамоделей.

Применение метамодели

Метамодель – модель предметной области, описанная на нескольких уровнях абстракции, где каждый высший уровень полностью, целостно и непротиворечиво задает структуру данных, функциональность, отображение и связи низших уровней.

В процессе функционирования три слоя (данные, логика и визуализация) должны динамически (в реальном времени) компилироваться из метамодели предметной области (рис. 1), а интеграция нескольких ИС на уровне метамоделей приводит к отображению интегрирующих факторов на каждый слой соответственно.

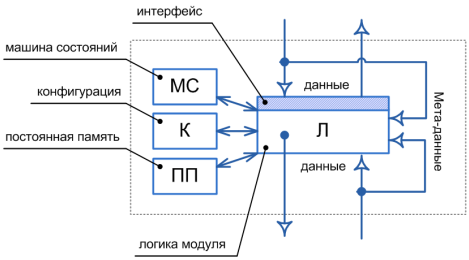

Рис. 1 – Обобщенное звено ИС, управляемое метаданными

Параметры метамодели

Какие же дополнительные параметры необходимы для описания метамодели? Обычно, при моделировании предметной области теряются очень важные параметры модели – атрибуты связей между сущностями предметной области. Дело в том, что в реляционной и объектной моделях, связи не могут иметь атрибутов. Если это необходимо, их вводят, как атрибуты сущностей, состоящих в связях, а что они в действительности принадлежат не сущностям, а связям, “знает” логика приложения,

 ISSN 1560-8956

т.е. эта часть модели описывается в коде, а не в структурах данных. При интеграции же, удаленная система уже не “знает” особенностей, а семантика, описанная не полностью, просто теряется, препятствуя интеграции систем. Какие же могут быть атрибуты связей: класс связи, направление, сила связи, условия действия, время установления, стойкость и т.д. Как в реляционных СУБД, так и в языках программирования, понятие ссылки сводится к идентификации двух связываемых сущностей и не позволяет добавлять дополнительные атрибуты.

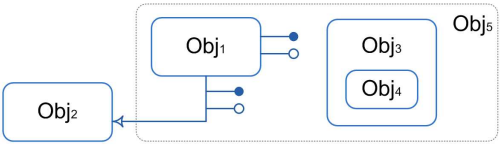

Рис. 2 – Пример объектов, описываемых метамоделью

Теряется, так же, и целостность объектов. Реляционная и объектная декомпозиция размещает атрибуты одной сущности во многих таблицах или многих классах, однако, это приводит к тому, что обратная операция в большинстве случаев уже не возможна без человека. Часть атрибутов описаны как поля таблицы, другие же – вынесены в разные таблицы, которые ссылаются на описываемую сущность. Но связи в реляционной модели используются не только для внутреннего моделирования объектов и их сложных свойств, но и для описания внешних отношений между разными объектами. Таким образом, по связи невозможно однозначно определить, является ли дочерняя таблица частью родительской сущности или отдельной сущностью. На практике приходится даже раскрашивать ER-диаграммы и структуры баз данных, выделяя цветами группы таблиц описывающих одни сущности. Но масштабность задач требует вводить не только такую одноранговую группировку, предметная область может быть разделена на зоны, что задается как префиксы к именам таблиц или группы таблиц помещаются в разные базы данных.

Таким образом, метамодель должна обеспечивать две принципиально разных возможности: связь между объектами и включение одного объекта в другой, как составной части. Кроме того, включение объектов так же должно быть разделено на включение “сильное” (вложенный объект может существовать только внутри родительского) и включение “слабое” (вложенный объект временно находится внутри родительского контейнера).

Современные СУБД не предоставляют возможности записывать источник получения данных в полях таблиц, хотя данные могут быть взяты из импорта, введены вручную, получены из внешней системы и т.д. Приходится выходить из положения, добавляя поле, описывающее источник данных другого поля. Реляционная модель не предоставляет возможно-

сти формально указать, что одно поле является атрибутом другого поля. Задача "понимания"этой логики ложится на приложение. Чтобы упростить операции можно применять суффиксы и префиксы в именовании полей, например поле Frequency содержит значение частоты, FrequencySource – указывает на источник данных и т.д. Аналогичным образом, может храниться и время фиксации измеренной величины (например: FrequencyTime). Однако, часто может быть необходимо хранить одно или несколько предыдущих значений. Для одного значения можно добавить поле (PreviousFrequency), для хранения истории требуется введение отдельной таблицы, и вот логика уже определяется приложением, т.е. на уровне СУБД невозможно явно указать, что таблица хранит историю параметра.

Теряются, так же, точность, весомость и степень доверия к атрибутам сущностей. Сохраняя значения атрибутов, необходимо определять в метаданных, что они поступали, например, с разных датчиков из разных систем, имеющих различную погрешность. Таким образом, к полю, хранящему значение, имеет смысл прикреплять поле-спутник, для указания точности (например, максимальное возможное значения отклонения или среднеквадратическое отклонение). Сопутствующие величины так же часто теряются, например, для производственных систем, могут быть метаданные: количество проведенных изменений для получения среднего значения, хранимого в поле; доверительный интервал значения; вероятностные характеристики; которые могут изменяться в зависимости от пути поступления данных в базу; время события получения данных и т.д. Реляционные СУБД не поддерживают также операций над нечеткими величинами, их приходится хранить в нескольких полях, а обрабатывать программно. Таким образом, модель предметной области размывается между структурой базы данных, сервером приложений и клиентом, что приводит к несовместимости с другой системой.

Теряется и семантика групповых связей, когда несколько связей описывают один аспект предметной области, а инструментов для объединения связей в группы нет. Сложности есть и в формальном описании сложных свойств, когда свойство распадается на несколько полей, а описать их группировку или определить ее невозможно. В таких случаях нужно использовать метаданные, которые будут интерпретированы в сервере приложений или в клиенте.

Выводы

Итак, все приведенные выше случаи говорят о большом количестве параметров предметной области, которые описываются реляционной и объектной моделями не полно, что и является препятствием для взаимодействия систем. Введение же метамоделей, позволяет задавать несколько слоев абстракции, описывающих ключевые для задач интеграции параметры и, тем самым, повышать гибкость при объединении, масштабировании и модернизации ИС. Метамодели, интерпретируемые динамически, позволяют так же генерировать интерфейсы связей между ИС

в реальном времени, строить пользовательские интерфейсы “на лету” и модифицировать структуры хранения данных без внесения изменений в программный код систем.

Литература

1. Дон Тапскотт. Электронно-цифровое общество. Рефл-бук, 1999, 408 с.
2. Под редакцией Н. Ермошкина. Демистификация ИТ. Что на самом деле информационные технологии дают бизнесу. 2006, Альпина Бизнес Букс, 304 с.
3. Don Tapscott. Grown Up Digital: How the Net Generation is Changing Your World. 2008.
4. Беккер Й., Вилков Л., Таратухин В., Кугелер М., Розерман М. Менеджмент процессов. М.: Эксмо, 2007. 360 с.

Отримано 01.12.2010 р.