96". Сборник научных трудов в трех томах. Том 2. Казань, 1996. С. 270-274.

[11] Рыбина Г.В. Пышагин С.В. Интеллектуальная поддержка разработки интегрированных экспертных систем // В кн.: КИИ'98 Шестая нац. конференция по искусственному интеллекту с межд. участием. Сборник научных трудов в трех томах. Том 2. Пущино: РАИИ, 1998. С. 419-426

[12]. Левин Д.Е., Пышагин С.В., Рыбина Г.В., Смирнов В.В. Новые возможности инструментального комплекса АТ-ТЕХНОЛОГИЯ, предназначенного для поддержки построения интегрированных экспертных систем // В кн.: КИИ'2000. Седьмая национальная конференция по искусственному интеллекту с международным участием. Труды конференции. М.: Изд-во физико-математической литературы, 2000. Т. 2. С 751-757.

[13]. Рыбина Г.В. Модель диалога в естественно-языковой системе ДИСАР // В кн.: Автоматизированная информационная технология. М.: Энергоатомиздат, 1990, С.29-36.

[14]. Автоматизация анализа научного текста. Киев,"Наукова думка",1984.

[15]. Рыбина Г.В., Пышагин С.В., Смирнов В.В., Левин Д.Е., Душкин Р.В. Инструментальный комплекс АТ-ТЕХНОЛОГИЯ для поддержки разработки интегрированных экспертных систем: Учебное пособие. М.: МИФИ, 2001.

ОРГАНИЗАЦИЯ БАЗ ДАННЫХ ДЛЯ ГИБРИДНЫХ СИСТЕМ МОДЕЛИРОВАНИЯ И ОПТИМИЗАЦИИ

А.В. Жиренкин

Сибирский Государственный Университет Телекоммуникаций и Информатики, РОССИЯ, Новосибирск, тел. (3832) 39-65-34, zhirenkin@katren.ru

Аннотация

В настоящее время в Лаборатории Математического Моделирования и Информационных Сетей Института Вычислительной Математики и Математической Геофизики Сибирского Отделения Российской Академии Наук ведутся работы по созданию гибридных систем моделирования и оптимизации транспортных и коммуникационных сетей. В рамках этой темы возникает вопрос об исследовании и разработке баз данных для подобных систем, включающий в себя создание структуры хранения информации, средств доступа к данным и средств обработки и анализа данных. В данной работе описана модель базы данных, наиболее полно отвечающая предложенным критериям хранения и обработки информации для гибридных систем моделирования.

1. Введение

Классической методикой проектирования баз данных является создание отдельной таблицы для каждой описываемой моделью данных сущности, затем, в процессе нормализации – выделение отдельных таблиц для хранения атрибутов сущности (таблицы-справочники). Такой подход хорошо работает для баз данных с относительно небольшим количеством описываемых объектов (десятки) и при несложных и статичных связях между ними. Однако любое изменение структуры хранимых данных приводит к внесению изменений в структуру таблиц, эти данные отображающих. Несложная на этапе разработки, эта операция становится крайне проблематичной при больших объемах данных и отсутствии у разработчика непосредственного доступа к БД (например, если она стоит у заказчика). Многим, наверное, знакомы громоздкие, медленные и требующие огромного дискового пространства операции по конвертированию БД при переходе на новую версию продукта. Не менее неприятна работа с базами данных, исторически разросшимися до сотен таблиц, структуру, которой сложно даже изобразить в читабельном виде. В связи с этим, встает вопрос – а нельзя ли создать структуру данных, не требующую переделок при появлении новых сущностей, позволяющую хранить произвольную информацию и при этом достаточно простую и эффективную? Чудес, увы, не бывает. Любая универсальная система менее эффективна, чем специализированная. Однако возможно создание решения, сочетающего приемлемую производительность и простоту с достаточной степенью универсальности.

2. Модель базы данных

Сформулируем основные принципы, на которых строиться наша модель данных.

- Каждая сущность, информация о которой хранится в БД – это объект.
- Каждый объект уникален в пределах базы данных и имеет уникальный идентификатор.
- Объект имеет свойства (строковые, числовые, временные, перечислимые), которые описывают атрибуты сущности.
- Объекты могут быть связаны между собой произвольным образом. Связь характеризуется связанными объектами и типом связи. Например, канал может быть связан с пучком, в котором он находится связью типа «канал в пучке» и т.п. Связь в некотором роде аналогична понятию ссылки на таблицу-справочник в традиционной модели базы данных.

Объект может быть хранилищем. В этом случае допускается хранение в нем других объектов (например, пакет в сообщении).

Такая модель, не привязана ни к какой конкретной модели и позволяет реализовать «над собой» практически любую логику. Логика выделяется в отдельный программный слой и, как правило, реализуется на сервере приложений, где по запросу клиента создаются объекты, загружающие информацию о себе из базы данных и реализующие «поведение» объектов реального мира. В то же время, в силу однообразности модели хранения эти объекты несложно создаются на основе базовых классов, инкапсулирующих функциональность по загрузке и сохранению свойств и связей в базе данных. Рассмотрим в деталях каждый из предложенных тезисов базы данных.

2.1. Объекты

Объект, в нашей модели – это скорее логическое понятие, его главное назначение – предоставить уникальный идентификатор, по которому он будет отличаться. Кроме этого, каждый объект обладает типом. Типы объектов мы будем описывать в таблице **ObjType** (Табл. 1).

Таблица 1. Типы объектов

ObjType	
Id INTEGER	Первичный ключ
Code CHAR(10) NOT NULL UNIQUE	Полное название типа.
ItemName CHAR(30)	Краткое название типа.

Типами объектов может быть, например «Гиперсеть», «Пучок», «Канал». Для реализации наследования необходимо всего лишь ввести в ObjType поле ParentId, ссылающееся на Id наследника.

Сами объекты представим в виде таблицы **Objects** (Табл. 2).

Таблица 2. Объекты.

Objects	
Id INTEGER	Первичный ключ
TypeId INTEGER REFERENCES ObjType(Id)	Ссылка на тип объекта
ItemName CHAR(50)	Название объекта

В поле Objects.ItemName хранится строка, описывающая объект и (часто) дублирующая одно из его свойств, либо формируемая как сочетание нескольких свойств (например, тип и количество вершин графа). При работе через сервер приложений формирование этого поля реализуется очень просто, а при выводе в интерфейсе списков объектов получение наименования из этого поля существенно повышает общее быстродействие базы данных. В то же время, в системах, где преимущественно производится ввод информации и мало операций по её выборке можно отказаться от него, что позволит уменьшить накладные затраты на его формирование и обновление. Таким образом, мы получаем возможность, хранить объекты произвольного количества типов. Выборка объектов конкретного типа производится запросом вида:

```
SELECT O.Id, O.ItemName, OT.ItemName
  FROM Objects O
  INNER JOIN ObjType OT ON O.TypeId =
OT.Id
  WHERE OT.Code = 'EMPLOYEE'
```

Однако, наименования явно недостаточно для описания произвольного объекта из реального мира. Поэтому дополним нашу модель группой таблиц для описания свойств объектов.

2.2. Свойства

Свойства объектов отражают реальные атрибуты описываемых ими сущностей. Все они принадлежат к какому-то объекту и содержат некую информацию об атрибуте и его величине, причем список атрибутов общий для всех объектов одного и того же типа. Величина же может выражаться данными различного типа.

Разделим атрибуты на несколько следующих типов:

- Строковые

- Числовые. Различия между целыми и вещественными числами делать не будем, полагая целые подмножеством вещественных. Если задача диктует невозможность применения данного подхода – придется выделять эти атрибуты в различные типы
- Атрибуты, зависящие от времени (дата и время)
- Перечислимые (например, номенклатура канальной аппаратуры)

2.2.1. Строковые атрибуты

Для описания атрибутов заведем таблицу **StrDesc** (Табл. 3). При настройке БД, для каждого типа объектов задается набор свойств данного типа. Так, например, объект типа «HYPERNET» (гиперсеть) может иметь строковые свойства:

«TYPE» - тип сети
«FUNCTION» - функциональность сети
«DESTINATION» - назначение сети

Таблица 3. Описание строковых атрибутов

StrDesc	
Id INTEGER	Первичный ключ
TypeId INTEGER REFERENCES ObjType(Id)	Ссылка на тип объекта
Code CHAR(10)	Краткое название параметра
ItemName CHAR(30)	Полное название параметра.

Сами атрибуты хранятся в таблице **Strings** (Табл. 4).

Таблица 4. Строковые атрибуты.

Strings	
Id INTEGER	Первичный ключ
TypeId INTEGER REFERENCES StrDesc(Id)	Ссылка на тип параметра
ObjectId INTEGER REFERENCES Objects(Id)	Ссылка на объект
Value VARCHAR(255)	Значение параметра

Чтобы получить список гиперсетей, имеющих один и тот же тип можно использовать запрос:

```
SELECT O.ItemName, S.Value
 FROM Objects O
  INNER JOIN ObjType OT ON O.TypeId =
OT.Id
  INNER JOIN Strings S ON O.Id = S.ObjectId
  INNER JOIN StrDesc SD ON S.TypeId =
SD.Id
 WHERE OT.Code = 'HYPERNET'
  AND SD.TypeId = OT.Id
  AND SD.Code = 'TYPE'
```

Интересный эффект мы получим, создав у различных типов объектов атрибуты с одинаковыми значениями поля Code. Предположим, что у объектов типов HYPERNET (гиперсеть) и GRAPH (граф) есть строковые атрибуты с одинаковым значением 'TYPE'. Тогда можно одним запросом получить список всех известных в базе графов и сетей с одинаковым типом.

```
SELECT O.ItemName, OT.ItemName, S.Value
 FROM Objects O
  INNER JOIN ObjType OT ON O.TypeId =
OT.Id
  INNER JOIN Strings S ON O.Id = S.ObjectId
  INNER JOIN StrDesc SD ON S.TypeId =
SD.Id
 WHERE SD.Code = 'TYPE'
ORDER BY OT.ItemName, O.ItemName
```

При появлении еще какого-то объекта, имеющего аналогичный атрибут – его наличие будет немедленно учтено и он появится в списке без каких-либо модификаций модели и запросов. Добиться подобного результата в БД с традиционной структурой будет, как минимум, сложно.

2.2.2. Числовые атрибуты

В зависимости от логики баз данных – источников для любого числового типа данных можно выделить отдельную группу таблиц. В остальном, хранение значений этого типа аналогично строковым. Описание атрибутов представлено ниже в виде таблицы **PropDesc** (Табл. 5)

Таблица 5. Описание числовых атрибутов.

PropDesc	
Id INTEGER	Первичный ключ
TypeId INTEGER REFERENCES ObjType(Id)	Ссылка на тип объекта
Code CHAR(10)	Краткое название параметра
ItemName CHAR(30)	Полное название параметра.

А сами атрибуты – в виде таблицы **Properties** (Табл.6). При необходимости возможно до-

полнение таблицы PropDesc дополнительными полями, задающими минимальное и максимальное значение атрибута, значение по умолчанию и т.п. Проверка корректности этих значений может осуществляться в триггере на изменение таблицы **Properties** (Табл. 6)

Таблица 6. Числовые атрибуты.

Properties	
Id INTEGER	Первичный ключ
TypeId INTEGER REFERENCES PropDesc(Id)	Ссылка на тип параметра
ObjectId INTEGER REFERENCES Objects(Id)	Ссылка на объект
Value DECIMAL(20,4)	Значение параметра. На серверах, поддерживающих тип данных MONEY (аналог Currency в Delphi), удобно использовать этот тип.

2.2.3. Атрибуты, зависящие от времени

Многие объекты в процессе своей «жизни» изменяют своё состояние. Так, канал может быть назначен соединению, удален от соединения, снова назначен, узел приведен в аварийное состояние, в ремонте и т.д. Для описания состояний объекта служит таблица **Status** (Табл. 7).

Таблица 7. Описание состояний объекта.

Status	
Id INTEGER	Первичный ключ
TypeId INTEGER REFERENCES ObjType(Id)	Ссылка на тип объекта
Code CHAR(10)	Краткое название состояния
ItemName CHAR(30)	Полное название состояния.

Например, для канала допустимые состояния могут включать:

«CONNECT» - Назначен соединению

«DELETE» - Удален от соединения

При этом возможно реализовать логику, когда любое состояние может возникать многократно, например, узел может быть испорчен, отремонтирован, опять испорчен.

Саму историю состояний можно представить в виде таблицы **History** (Табл. 8).

Таблица 8. История состояний объекта.

History	
Id INTEGER	Первичный ключ
StatusId INTEGER REFERENCES Status(Id)	Ссылка на тип состояния
ObjectId INTEGER REFERENCES Objects(Id)	Ссылка на объект
ItemDate DATETIME	Дата возникновения состояния

При изменении состояния объекта в **History** (Табл. 8) добавляется запись со ссылкой на это состояние и датой его возникновения.

2.2.4. Перечислимые атрибуты

Последним видом атрибутов будут перечислимые типы, которые могут принимать одно из заранее заданного набора значений. Для этих данных нам понадобится не две, как для остальных, а три таблицы. Первая таблица **EnumDesc** (Табл. 9) задаёт, какие перечисления допустимы для выбранного типа объекта:

Таблица 9. Описание допустимых перечислений.

EnumDesc	
Id INTEGER	Первичный ключ
TypeId INTEGER REFERENCES ObjType(Id)	Ссылка на тип объекта
Code CHAR(10)	Краткое название состояния
ItemName CHAR(30)	Полное название состояния.

Вторая – **EnumValues** (Табл. 10) – определяет возможные значения для перечислимого типа.

Таблица 10. Значения для перечислимого типа

EnumValues	
Id INTEGER	Первичный ключ
DescId INTEGER REFERENCES EnumDesc(Id)	Ссылка на тип перечисления
Code CHAR(10)	Краткое название значения
ItemName CHAR(30)	Полное название значения.

И третья – **Enums** (Табл. 11) непосредственно хранит значения, связанные с объектом.

Таблица 11. Перечислимые атрибуты.

Enums	
Id INTEGER	Первичный ключ
ValueId INTEGER REFERENCES EnumValues(Id)	Ссылка на значение
ObjectId INTEGER REFERENCES Objects(Id)	Ссылка на объект

Таким образом, мы получили возможность описать произвольный набор свойств любой сущности, данные о которой хранятся в реальной базе данных и отобразить её на единообразно хранимые объекты. Однако, на настоящий момент данная модель еще не пригодна для решения сколько-либо сложных задач, поскольку реальные объекты любой предметной области имеют разнообразные связи друг с другом. Определим их в рамках нашей модели.

2.3. Связи

Сущности реального мира имеют между собой множество различных связей, определяющих их роль и место в этом мире. В традиционной методологии построения БД связи отображаются на ссылки (REFERENCES) между таблицами, в которых хранится описание соответствующих сущностей. Такой подход обеспечивает высокую эффективность БД и возможность автоматической проверки и поддержания целостности связей (механизмы referential integrity), однако жестко ограничивает связи, хранимые в БД её структурой. В нашей модели структуры данных не предусмотрено отдельных полей для хранения ссылок на другие таблицы. Вместо них введем дополнительные таблицы. Первая из них – **LinkType** (Табл.12)- описывает возможные типы связей, которые могут быть установлены между объектами.

Таблица 12. Типы связей.

LinkType	
Id INTEGER	Первичный ключ
Code CHAR(10)	Краткое название связи
ItemName CHAR(30)	Полное название связи.

Например, канал может быть связан с пучком, в котором он находится связью типа «CANBUNCH» (канал в пучке). В то же время он может быть связан с отделом связью «MAIN» (магистральный канал).

Сами связи представляются в таблице **Links** Табл.13.

Таблица 13. Связи.

Links	
Id INTEGER	Первичный ключ
ParentId INTEGER REFERENCES Objects(Id)	Ссылка на первый из связываемых объектов
ChildId INTEGER REFERENCES Objects(Id)	Ссылка на второй из связываемых объектов
TypeId INTEGER REFERENCES LinkType(Id)	Ссылка на тип связи.

Таким образом, чтобы получить канал в определенном пучке необходимо выполнить запрос:

```
SELECT O.ItemName
FROM Objects O
INNER JOIN Links L ON O.Id = L.ParentId
INNER JOIN LinkType LT ON LT.Id =
L.TypeId
WHERE LT.Code = 'CANAL'
AND L.ChildId = :FirmId -- первичный ключ
объекта-пучок
```

А все каналы в данном пучке можно получить запросом:

```
SELECT Employee.ItemName
FROM Objects Canals
  INNER JOIN Links L ON L.ChildId = Ca-
nals.Id
  INNER JOIN LinkType LT ON L.TypeId =
LT.Id
  INNER JOIN Objects Bunchs ON L.ParentId =
Bunchs.Id
  INNER JOIN ObjType OT ON Bunchs.TypeId
= OT.Id
WHERE OT.Code = 'BUNCH'
  AND LT.Code = 'CANBUNCH
  AND Department.ItemName = 'Искомый пучок'
```

Тут сразу видна слабость предлагаемой схемы – связи никак не контролируются, и ничто не гарантирует, что "каналом" в пучке не окажется узел. Поскольку описание связей между объектами является "сердцем" нашей модели, необходимо ввести в неё механизм, контроли-

рующий допустимость устанавливаемой связи. Таким механизмом может служить таблица допустимых связей – **AllowedLinks** (Табл.14).

Таблица 14. Допустимые связи.

AllowedLinks	
Id INTEGER	Первичный ключ
ParentId INTEGER REFERENCES ObjType(Id)	Ссылка на тип первого из связываемых объектов
ChildId INTEGER REFERENCES ObjType(Id)	Ссылка на тип второго из связываемых объектов
TypeId INTEGER REFERENCES LinkType(Id)	Ссылка на тип связи.

На вставку и изменение записей в **Links** (Табл.13) создается несложный триггер, который проверяет типы объектов для создаваемой связи и их допустимость по **AllowedLinks** (Табл.14). Так как разрешенные типы связей хранятся в таблице, мы получаем гибкий механизм для настройки БД под требования конкретной задачи. Если требуется задать новый тип связи между объектами – достаточно лишь добавить в **AllowedLinks** (Табл.14) запись с этим типом связи и типами объектов. Хотелось отметить, что предлагаемая модель позволяет легко определить иерархические данные. Для этого надо лишь задать связь типа "находится в". Выбирая какой тип объекта должен быть первым (ParentId), а какой вторым (ChildId) в связи, лучше придерживаться единой системы. При проведении анализа это безразлично, однако, для уменьшения путаницы лучше в качестве первого выбирать объект, который бы оказался на стороне «один» отношения «один-ко-многим» в реальной базе данных. Если объекты связаны по принципу «многие-ко-многим» - надо просто выработать для себя единую методику и в дальнейшем ей следовать.

3. Заключение

Преимущества предложенной модели:

- Возможность определить в рамках модели произвольный объект из реального мира.
- Структура ее независима от данных и не требует пересмотра при появлении новых сущностей.
- Большие возможности для анализа и простота его проведения по сравнению с анализом базы данных, построенной по классической схеме.

Недостатки модели:

- **Ослабленный контроль целостности данных**. Нельзя осуществить полный контроль ввода объектов при незаполненном атрибуте, либо неуказанной связью с другим объектом. *Обойти это можно путем создания приложений, работающих с системой, не допуская прямого доступа к системе.*
- **Сложность в понимании структуры и выборке данных.** Сама структура модели не ориентированна на бизнес- логику. *Преодолением данной проблемы может послужить создание набора представлений.*
- **Пониженное быстродействие при выборке**. Для увеличения процесса выборки, возможно автоматическое формирование в базе краткого описания объекта, необходимого для поиска его в списке себе подобных и использование преимущественно именно этого поля. *Практика показывает, что вывод имени объекта и двух- трех атрибутов вполне достаточен для его выбора.*

Литература

[1].В.К. Попков Математические модели живучести сетей связи, Новосибирск, ВЦ СОРАН, 1990, стр. 3-94

[2].В.К. Попков Диалоговая система моделирования и оптимизации сетей электросвязи. *В кн. Проблемы функционирования информационных сетей электросвязи.* Новосибирск, 1991, стр.238-346

[3].Дейт, К. Дж. Введение в системы баз данных: [Пер. с англ.]/К. Дейт.- 6-е изд.- М. и др.: Вильямс, 1999.- 846 с.; 24 см.- (Серия "Системное программирование") - Перевод изд.: An introduction to database systems C. J. Date (Reading, Mass. etc.).- ISBN 5-8459-0019-0

[4].Реализация баз данных Microsoft SQL Server 7.0: Офиц. пособие Microsoft для самостоят. подгот. : [Сертификац. экзамен 70-019 : Пер. с англ.].- М.: Рус. ред., 2000.- XXIX, 475 с.: ил.; 24 см.- (Учебный курс) - Перевод изд.: Designing and Implementing Data Warehouses with Microsoft SQL Server 7.0 (Microsoft press).- ISBN 5-7502-0113-9