В. Ю. Пирогов

ИНФОРМАЦИОННЫЕ СИСТЕМЫ И БАЗЫ ДАННЫХ

организация и проектирование

В. Ю. Пирогов

ИНФОРМАЦИОННЫЕ СИСТЕМЫ И БАЗЫ ДАННЫХ

организация и проектирование

Рекомендовано УМО в области инновационных
междисциплинарных образовательных программ в качестве учебного пособия
по специальности 010503 «Математическое обеспечение
и администрирование информационных систем»

Санкт-Петербург
«БХВ-Петербург»
2009

УДК 681.3.06
ББК 32.973.26-018.2
П33

Пирогов В. Ю.

П33 Информационные системы и базы данных: организация и проектирование: учеб. пособие. — СПб.: БХВ-Петербург, 2009. — 528 с.: ил. — (Учебная литература для вузов)

ISBN 978-5-9775-0399-0

Излагаются основные вопросы по организации и проектированию информационных систем: классификация, структура, безопасность и принципы проектирования; а также архитектура информационной системы: интерфейсы и протоколы, клиентские приложения. Большое внимание уделяется базам данных и их программному управлению, языкам SQL и QBE. Приводятся примеры новых технологий в области баз данных.

Для студентов вузов, специалистов и руководителей, применяющих информационные системы

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Татьяна Лапина*
Зав. редакцией	*Григорий Добин*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Фото	*Кирилла Сергеева*
Зав. производством	*Николай Тверских*

Рецензенты:

Б. А. Новиков, д. ф.-м. н., профессор, заведующий лабораторией исследования операций математико-механического факультета Санкт-Петербургского государственного университета;

В. А. Костин, к. ф.-м. н., доцент кафедры информатики Санкт-Петербургского государственного университета.

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 30.07.09.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 42,57.
Тираж 1500 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

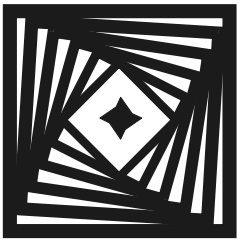
Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0399-0

Оглавление

ГЛАВА 2



Базы данных как часть информационной системы

Глава посвящена рассмотрению такого понятия, как *база данных*. Мы подробно рассматриваем одну из наиболее употребляемых в настоящее время моделей данных — реляционную модель: основные положения, реляционную алгебру и реляционное исчисление, нормальные формы, связи между таблицами, а также место, которое занимает базы данных в структуре информационных систем.

2.1. Базы данных

Общие понятия

Базы данных — что это? Ведь ранее мы говорили об информационных хранилищах. Информационное хранилище является частью информационной системы. Но мы никак не уточняли, какова структура таких хранилищ. В действительности обычному пользователю нет нужды знать эту структуру. При работе с ИС он оперирует профессиональными терминами и понятиями. Например, бухгалтер при работе с бухгалтерской информационной системой имеет дело с такими объектами, как счет, проводка, журнал, платежный документ и т. д. Он не знает, как в действительности устроено информационное хранилище, да и не обязан знать, т. к. каждый должен заниматься своим делом, а поверхностные знания часто приводят к отрицательным результатам.

Таким образом, информационное хранилище является общим понятием, отражающим факт наличия в системе хранимых данных. Для пользователя те объекты, которыми он оперирует при работе с ИС, и есть те самые хранимые данные. С другой стороны, грамотный компьютерщик знает, что все данные хранятся в файлах. Файлы — это те структуры хранения данных, которыми манипулирует операционная система. Но файлы — это всего лишь именован-

ная последовательность хранимых во внешней памяти байтов. Однако при создании информационной системы не слишком удобно оперировать последовательностями байтов. Непосредственно с файлами работают системные программисты. Прикладные программисты работают с другой абстракцией. Эта абстракция находится на среднем уровне между уровнем файловой системы *(физический или внутренний уровень)* и уровнем пользователя *(внешний уровень)*. Назовем этот промежуточный уровень *логическим* или *прикладным уровнем*. Сказанное проиллюстрировано на рис. 2.1.

Итак, для конечного пользователя информационное хранилище состоит из значений тех показателей, которыми он оперирует в силу своей профессиональной деятельности. Для системного программиста информационное хранилище представляет собой набор файлов. Для прикладного же программиста и администратора ИС информационное хранилище представляет собой некоторую абстрактную структуру, отражающую предметную область. Этот уровень и принято описывать с помощью понятия *"база данных"*. Информационное хранилище может состоять из нескольких баз данных, идентифицируемых своим именем. Структура базы данных строится на основе *модели данных*, о которых мы будем говорить в следующем разделе. Важно отметить, что модель данных, по сути, представляет собой некоторый язык, с помощью которого может быть описана предметная область. Такое описание не должно зависеть от аппаратной и программной части системы. В описание должен входить и набор типовых операций над данными.

Рис. 2.1. Три уровня восприятия данных

Определение

Базой данных будем называть именованную часть информационного хранилища, структура которой описывается на языке некоторой модели данных. Описание структуры конкретной базы данных называется *схемой*, *системным каталогом* (или просто *каталогом*) *базы данных* или *словарем базы данных*[1].

[1] Чрезвычайно интересное обсуждение понятия "база данных" можно найти в книге [34].

СУБД

СУБД — это система управления базами данных. В *главе 5* мы подробно остановимся на этом понятии. Но для дальнейшего рассмотрения требуется хотя бы кратко пояснить читателю, что это такое.

Для того чтобы понять, что такое СУБД, обратимся снова к рис. 2.1. Для того чтобы прикладные программисты воспринимали базу данных не в виде набора файлов, а как некоторую структуру, которая описывает предметную область, между ними и файловой системой должна быть некоторая прослойка. Это прослойка представляет собой набор процедур (программный интерфейс), с помощью которого прикладной программист может управлять базой данных. Эту функцию выполняет СУБД. Другими словами, СУБД отделяет прикладного программиста от физической структуры базы данных. В простейшем случае СУБД состоит только из программного интерфейса. Так обстоит дело в некоторых системах программирования. В более сложных системах мы имеем также и интегрированную среду, позволяющую в интерактивном режиме управлять базами данных. Кроме этого, СУБД также предоставляет разработчикам средства безопасности, такие, например, как утилиты для резервного копирования, поддержка параллельной работы приложений, система поддержки целостности баз данных, транзакционные механизмы, парольный вход и разделение доступа, и многое другое *(см. главу 5)*. Большинство современных информационных систем строится именно на основе СУБД.

На рис. 2.2 представлена схема взаимодействия СУБД и прикладного программного обеспечения в схеме построения информационной системы. Тут важно понять, что база данных в определенной модели существует для прикладного программного обеспечения, тогда как программное обеспечение СУБД взаимодействует с данными на уровне файловой системы и системных вызовов операционной системы.

Все СУБД могут быть поделены на настольные и промышленные. Настольные СУБД, такие как Access, FoxPro, предназначены для создания либо автономных информационных систем, либо ИС файл-серверного типа. Промышленные СУБД[1], такие как Oracle, MS SQL Server, PostgreSQL и др., предназначены для построения клиент-серверных информационных систем. СУБД, как правило, предоставляет разработчику язык программирования, который включает в себя специализированный язык управления базами данных. Для наиболее распространенных баз данных *реляционного* типа таким языком является язык SQL.

[1] Другое название промышленных СУБД — серверы баз данных.

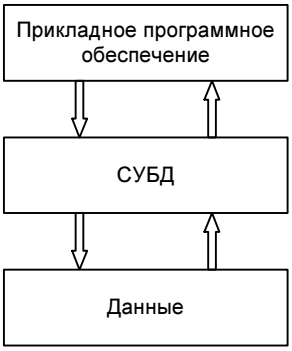


Рис. 2.2. Информационная система на основе СУБД

Использование СУБД при построении информационных систем призвано реализовать *физическую* и *логическую* независимость прикладного программирования от данных. Физическая независимость от данных заключается в том, что работа программного обеспечения ИС не будет зависеть от изменений, которые могут происходить на внутреннем, физическом уровне. Эти изменения могут заключаться, например, в том, что будет изменена файловая система или же в том, что изменится структура тех файлов, которые составляют базу данных.

Логическая независимость прикладного программирования от данных при использовании СУБД в трехуровневой структуре доступа к данным (см. рис. 2.1) заключается, прежде всего, в том, что добавление новых элементов (например, добавление нового столбца в таблицу) в структуру данных никак не влияет на функционирование программного обеспечения. Добиться логической независимости от операций удаления элементов из базы данных в общем случае невозможно, поскольку на удаляемые элементы, как правило, имеются явные ссылки в программном обеспечении.

Модели данных

Рассмотрим основные модели данных, которые используются (или использовались) при создании информационных систем. Перечень существующих моделей данных является в то же время и хронологическими метками истории развития баз данных и информационных систем.

Файловая модель

Иногда файловую модель, на мой взгляд, очень неудачно называют файловой системой. При этом упускается из вида тот факт, что данный термин служит для обозначения организации данных во внешней памяти, которую поддерживают и используют операционные системы. Файловая модель была первой

моделью, используемой при разработке информационных систем[1]. Точнее, модель, как таковая, отсутствовала. Можно сказать, что файловая модель — это модель без СУБД. Прикладные программисты разрабатывали базы данных непосредственно на внутреннем уровне (см. рис. 2.1), т. е. имели дело напрямую с файлами (логический и внутренний уровень совпадали). Другими словами базы данных представляли собой наборы файлов, трактовка внутренней структуры которых принадлежала непосредственно разработчикам данной информационной системы, т. е. была уникальна. Файловая модель обладала рядом недостатков, но, несмотря на это, она дожила до наших дней и иногда используется для разработки небольших однопользовательских информационных систем. Перечислим основные недостатки данной модели.

- Структуру базы данных приходилось разрабатывать каждый раз при разработке информационной системы, что требовало определенных усилий и времени.
- Поскольку алгоритм управления такой базой данных был полностью заложен в программном обеспечении информационной системы, то при необходимости изменить структуру данных каждый раз приходилось вносить изменения и в программное обеспечение. Кроме этого, каждый раз приходилось писать утилиты для преобразования базы данных от одной структуры к другой. Такие утилиты часто были одноразового использования. Все усложняющиеся структуры данных и алгоритмы их обработки приводили в конечном итоге к увеличению программных ошибок.
- Часто даже в одной фирме создавалось несколько информационных систем (для каждого отдела, а зачастую для каждого рода деятельности), каждая из которых оперировала своими структурами данных. Бесконечной головной болью был перенос данных из одной ИС в другую, ведь структуры данных еще и периодически менялись. Так что часть программистов непрерывно писала программы преобразования данных из одного формата в другой.
- Децентрализованное хранение приводило к тому, что в различных отделах приходилось дублировать одни и те же данные, что, во-первых, требовало дополнительных ресурсов (времени и денег), а во-вторых, хранение дополнительных данных требовало и дополнительной внешней памяти. Но головная боль начиналась тогда, когда нарушалась согласованность данных об одном и том объекте, но из разных информационных систем. Для обнаружения и исправления таких ошибок требовались большие усилия.

---

[1] Можно сказать, что первые файловые модели были простой заменой ручных картотек. Каждый ящик картотеки соответствовал одному файлу. Например, в одном файле хранился список работников предприятия, второй файл хранил перечень выпускаемой продукции и т. д.

- Работа того или иного отдела требовала все новых и новых форм отчетности. Но для каждого нового отчета приходилось писать программный код, что увеличивало дополнительную нагрузку программистов. При этом, опять же после изменения структуры данных, все процедуры формирования отчетов приходилось снова переписывать.
- Разные фирмы часто пользовались различными языками программирования, отличающимися друг от друга структурой создаваемых ими файлов. Возникали, таким образом, дополнительные сложности совместимости форматов файлов.

> ***ЗАМЕЧАНИЕ***
>
> Строго говоря, файловую модель данных нельзя назвать моделью в полном смысле этого слова, т. к. здесь мы четко видим зависимость описания данных от таких факторов, как файловая система и программное обеспечение.

Сетевая модель

Сетевая модель относится к ранним моделям данных. Сейчас информации об этой модели данных почти не встретишь, даже такой специалист как К. Дейт при переиздании своего классического труда "Введение в системы баз данных" исключил вопросы, касающиеся сетевой и иерархической модели данных (см. [8, 15]). Впрочем, в последнее время снова заговорили о сетевой модели в связи с распределенными информационными системами, а также с развитием объектных моделей данных.

В 1971 году группа DTBG (Database Task Group) представила в Американский национальный институт стандартов отчет, который послужил в дальнейшем основой для разработки сетевых систем управления базами данных. Стандарт сетевой модели впервые был определен в 1975 году организацией CODASYL (Conference of Data System Languages), которая определила базовые понятия модели и формальный язык описания.

Сетевая модель данных опирается на математическую теорию направленных графов. Базовыми элементами сетевой модели являются:

- *элемент данных* — минимальная информационная единица, доступная пользователю;
- *агрегат данных* — именованная совокупность элементов данных внутри записи или другого агрегата. Агрегат бывает двух видов — агрегат типа *"вектор"* и агрегат типа *"повторяющаяся группа"*. Например, агрегат <город, улица, дом, квартира>, которому можно присвоить имя **Адрес**, является агрегатом типа "вектор". Примером агрегата типа "повторяющаяся группа" может служить агрегат <месяц, сумма> с названием **Зарплата**. Агрегат "повторяющаяся группа" характеризуется числом повторений. В данном примере это число повторений равно 12;

- *запись* — совокупность агрегатов или элементов данных, отражающих некоторую сущность предметной области. Например, записью будет <Фамилия, Зарплата>, где Фамилия — это элемент данных, а Зарплата — агрегат. Данную запись можно назвать **Зарплата сотрудника**;
- *тип записей* — эта совокупность подобных записей. Например, в предыдущем случае типом записи будет совокупность всех записей **Зарплата сотрудника**, выражающая множество сотрудников некоторого отдела. Тип записей представляет (моделирует) некоторый класс реального мира;
- *набор* — именованная двухуровневая иерархическая структура, которая содержит запись владельца и запись (или записи) членов. Наборы отражают связи "один-ко-многим" и "один-к-одному" между двумя типами записей. На рис. 2.3 представлен пример набора. Здесь **Отдел** — запись–владелец, **Сотрудник** — запись-член. *Тип набора* определяет связь между двумя типами записей. Каждый экземпляр типа набора содержит один экземпляр записи владельца и произвольное количество записей членов (для связи типа "один-ко-многим"). Обратите внимание, как на рис. 2.3 обозначается связь "один-ко-многим". Разветвление на конце указывает на множество экземпляров некоторого объекта. Такой способ обозначения связей мы будем использовать и далее. Среди всех наборов в сетевой модели допускается существование наборов, не имеющих владельцев. Такие наборы называются *сингулярными*. Владельцами сингулярных наборов формально считается система. Сингулярные наборы предназначены для доступа к экземплярам отдельных записей.

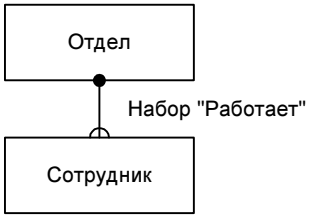


Рис. 2.3. Набор в сетевой модели данных

Резюмируя вышесказанное, будем говорить, что структура базы данных в сетевой модели задается типами записей и типами наборов.

Отметим некоторые особенности построения сетевой модели:

- база данных может состоять из произвольного количества записей и наборов различных типов;
- связь между двумя записями может выражаться произвольным количеством наборов;

- в любом наборе может быть только один владелец;
- тип записи может быть владельцем в одних типах наборов и членом в других типах наборов;
- тип записи может не входить ни в один из определенных типов наборов.

Для управления сетевой базой данных используется специальный язык, который можно разбить на следующие разделы:

- язык описания данных в сетевой модели:
  - описание базы данных (размещение);
  - описание элементов, агрегатов и записей;
  - описание наборов;
- язык манипулирования данными:
  - навигационные операции. С помощью операций навигации (группы операций FIND), двигаясь по связям, можно переходить от одной текущей записи к другой. Соответственно операции модификации осуществляются над текущей записью;
  - операции модификации, которые осуществляют:
    - добавление новых экземпляров отдельных типов записей;
    - добавление экземпляров новых наборов;
    - удаление экземпляров записей и наборов;
    - модификацию отдельных составляющих внутри конкретных экземпляров записей.

Иерархическая модель

Исторически *иерархическая модель* появилась раньше сетевой. Она наиболее проста из всех моделей данных. Самой известной иерархической системой, позволяющей создавать иерархические базы данных, является система IMS (Information Management System) фирмы IBM, используемая в свое время для поддержки лунного проекта "Аполлон". Появление иерархической модели связано с тем, что в реальном мире очень многие связи соответствуют иерархии, когда один объект выступает как родительский, а с ним может быть связано множество подчиненных объектов.

Основными информационными единицами в иерархической модели являются: база данных (БД), *сегмент*[1] и *поле*. Поле данных определяется как минимальная, неделимая единица данных, доступная пользователю с помощью

[1] Вместо термина "сегмент" используется также термин "запись".

СУБД. Выделяют также *тип поля*, представляющий собой совокупность полей одного типа. Сегмент состоит из конкретных экземпляров полей. *Тип сегмента* — совокупность входящих в него типов полей. Иерархическая модель представляет собой неориентированный граф, в вершинах которого располагаются сегменты (или типы сегмента). Дуги, соединяющие узлы, представляют собой связи или типы связей. Особенностью такой модели является то, что каждый сегмент может иметь не более одного предка, произвольное количество потомков и, по крайней мере, одно поле. Сегмент, который не имеет потомков, называют *листовым сегментом*. Иерархическое дерево начинается с одного сегмента, называемого *корневым сегментом*. Очень важно, что каждый сегмент должен иметь свое уникальное имя или идентификатор.

На рис. 2.4 схематически представлена иерархическая структура. Узлы (сегменты) соединены друг с другом связующими дугами. Сегмент `A` является корневым сегментом. Сегменты `B`, `E`, `H`, `J`, `I` являются листовыми сегментами. Каждый сегмент при этом может содержать произвольное количество полей.

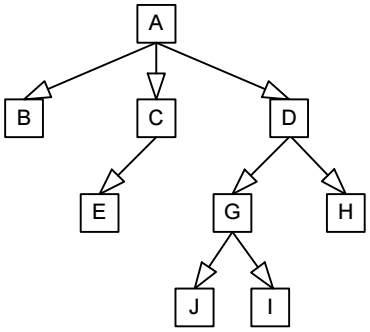


Рис. 2.4. Иерархическая структура

Для иерархической модели данных выделяют два языковых средства: язык описания данных и язык модификации данных. Описание базы данных предполагает описание всех ее сегментов и установление связей между ними.

Иерархическая модель довольно удобна для представления предметных областей, т. к. иерархические отношения довольно часто встречаются между сущностями реального мира. Но иерархическая модель не поддерживает отношения "многие-ко-многим", когда множество объектов одного типа связаны с множеством объектов другого типа. Предположим, что требуется построить модель отношения между множеством собственников жилья и множеством квартир. Если основной вопрос будет заключаться в определении того, каким жильем владеет тот или иной собственник, то естественно взять в качестве родительских узлов данные о собственнике. При этом каждый сегмент — собственник будет связан с `N` узлами — квартирами. Таким образом,

по собственнику мы легко найдем все квартиры, которые находятся в его собственности. Однако проблема заключается в том, что у одной и той же квартиры может быть несколько собственников. То есть одна и та же квартира может встречаться в разных деревьях. В результате решения таких задач, как получение списка всех квартир или получение всех собственников конкретной квартиры, будут уже не столь очевидными. Кроме того, сложной выглядит даже операция удаления из базы конкретной квартиры, поскольку для этого придется просматривать все деревья. Можно, конечно, построить параллельно деревья, в которых родительскими сегментами будут данные о квартирах, а порождаемыми сегментами — данные о владельцах, но в результате мы получим еще избыточность данных, что породит дополнительную проблему их согласованности.

Основной единицей обработки в иерархической модели является сегмент. К сегментам могут применяться такие операции, как запомнить, модифицировать, удалить, извлечь, найти. Операция поиска сводится к одной из возможных процедур обхода дерева. Иерархические СУБД поддерживают, обычно, правило: никакой сегмент не может существовать без своего родителя (исключая корневой сегмент). Подобные правила, поддерживаемые СУБД, называют *ограничениями целостности*.

В качестве примера реально работающей иерархической модели данных можно привести устройство реестра, поддерживаемое операционными системами семейства Windows.

Реляционная модель

Основателем реляционной модели данных является сотрудник фирмы IBM Э. Ф. Кодд. В статье "A Relation Model of Data for Large Shared Data Banks", которая вышла в 1970 году, он показал, что любое представление данных может быть сведено к совокупности двумерных таблиц, которые в математике называются отношениями (relations, отсюда термин *"реляционный"*). Мы подробно рассмотрим реляционную модель данных в *разд. 2.2*.

Объектная и объектно-реляционная модели

Объектные СУБД призваны интегрировать свойства баз данных и объектных языков программирования. В 1993 году был принять стандарт объектных баз данных ODMG-93 (Object Database Management Group — группа управления объектными базами данных, образована в 1991 году). В последнее время ведущие производители реляционных СУБД ради повышения конкурентоспособности своих продуктов пытаются внести в них элементы объектного проектирования. Мы подробнее поговорим об объектных и объектно-реляционных моделях в *главе 5*.