МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯРОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждениевысшего образования
«Воронежский государственный технический университет»

Кафедра систем автоматизированного проектирования и информационных систем

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ ПО ДИСЦИПЛИНЕ «Технологии проектирования баз данных (SQL/NoSQL)»

для обучающихся по направлению 09.03.02 «Информационных системы и технологии», профиль «Разработка web-ориентированных информационных систем» всех форм обучения

Воронеж 2022

УДК 658.512:621(07)
ББК 30.18:85.1:34.5я7

Составители: Э. Р. Саргсян, Н. А. Рындин

Методические рекомендации по выполнению лабораторных работ по дисциплине «Технологии проектирования баз данных (SQL/NoSQL)» для обучающихсяпо направлению 09.03.02 «Информационных системы и технологии», профиль «Разработка web-ориентированных информационных систем» всех форм обучения / ФГБО ВО «Воронежский государственный техническийуниверситет»; сост.: Э. Р. Саргсян, Н. А. Рындин – Воронеж: Изд-во ВГТУ, 2021. – 33 с.

Приводится описание выполнения лабораторных работ по курсу «Технологии проектирования баз данных (SQL/NoSQL)» для студентов обучающихся по направлению 09.03.02 «Информационных системы и технологии», профиль «Разработка web-ориентированных информационных систем» всех форм обучения

УДК 658.512:621(07)
ББК 30.18:85.1:34.5я7

Рецензент -

Рекомендовано методическим семинаром кафедры САПРИС и методической комиссией ФИТКБ Воронежского государственного технического университета в качестве методических материалов

1. Ранние подходы к организации баз данных

К ранним моделям относят модели, предшествующие реляционной модели данных: иерархическую, сетевую модели и модель на основе инвертированных списков. В явном виде таковых моделей осталось очень мало. Однако эти системы исторически предшествовали реляционным, и для правильного понимания причин повсеместного перехода к реляционным моделям необходимо иметь о них представление. Ниже приведены наиболее общие характеристики ранних систем:

1. Эти системы активно использовались в течение многих лет, дольше, чем используются многие из реляционных СУБД. Некоторые из ранних систем используются даже в наше время, накоплены громадные базы данных, и одной из актуальных проблем информационных систем является использование этих систем совместно с современными системами.
2. Все ранние системы не основывались на каких-либо абстрактных моделях. Понятие модели данных фактически вошло в обиход специалистов в области баз данных только вместе с реляционным подходом. Абстрактные представления ранних систем появились позже на основе анализа и выявления общих признаков у различных конкретных систем.
3. В ранних системах доступ к базам данных производился на уровне записей. Пользователи этих систем осуществляли явную навигацию в базе данных, используя языки программирования, расширенные функциями СУБД. Интерактивный доступ к базе данных поддерживался только путем создания соответствующих прикладных программ с собственным интерфейсом.

4. Навигационная природа ранних систем и доступ к данным на уровне записей заставляли пользователя самого производить всю оптимизацию доступа к базе данных без какой-либо поддержки системы.
5. После появления реляционных систем большинство ранних систем было оснащено «реляционными» интерфейсами. Однако в большинстве случаев это не сделало их по-настоящему реляционными системами, поскольку оставалась возможность манипулировать данными в естественном для них режиме.

Необходимость изучения ранних моделей вызвана еще одним обстоятельством: внутренняя организация реляционных систем во многом основана на использовании методов ранних систем, и некоторое знание в области ранних систем будет полезно для понимания путей развития постреляционных СУБД (например, в основе графовой модели лежит сетевая модель данных).

1.1. Иерархическая модель данных

Иерархическая модель данных была исторически первой структурой баз данных, видимо, из-за того, что древовидные иерархические структуры широко используются в повседневной человеческой деятельности. Это всевозможные классификаторы, ускоряющие поиск информации, иерархические функциональные структуры управления и т.д.

Иерархическая модель данных – представление базы данных в виде древовидной (иерархической) структуры, состоящей из объектов (данных) различных уровней.

В 1968 г. компания IBM предложила систему управления информацией IMS (Information Management System), основанную на иерархической модели данных. В этой модели данные представляются как дерево связанных записей. Имеется один корневой (родительский) тип записи – корень дерева. С ним в подчиненной связи типа 1: N находятся дочерние записи. Связи между записями выражаются в виде отношений предок/потомок, а у каждой записи есть ровно одна родительская запись. Это помогает поддерживать ссылочную целостность: когда запись

удаляется из дерева, все ее потомки должны быть также удалены. Общая схема иерархической модели представлена на рис. 1.1:

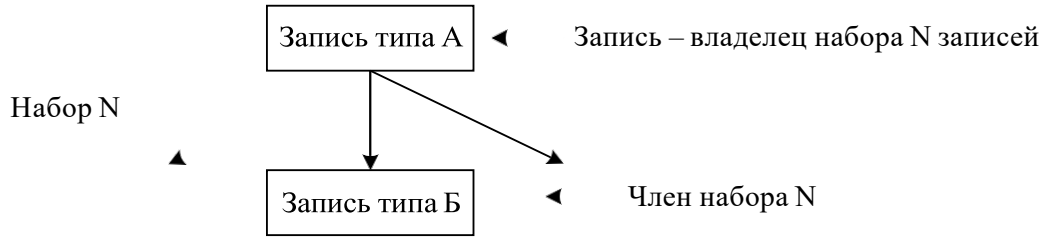


Рис. 1.1. Общая схема иерархической модели данных

Одной из наиболее важных сфер применения первых СУБД было планирование производства в компаниях, занимающихся выпуском продукции. Например, если автомобильная компания хотела выпустить 10000 автомобилей одной модели и 5000 автомобилей другой модели, необходимо знать, сколько деталей следует заказать у своих поставщиков. Чтобы ответить на этот вопрос, необходимо выяснить, из каких частей состоит изделие, затем определить, из каких деталей состоят эти части и т.д. Так, например, автомобиль состоит из двигателя, корпуса и ходовой части, двигатель состоит из клапанов, цилиндров, свеч и т.д. Список составных частей изделия (рис. 1.2) по своей природе является иерархической структурой. Для хранения данных, имеющих такую структуру, и была разработана иерархическая модель данных, представляющая собой упорядоченный граф (дерева).

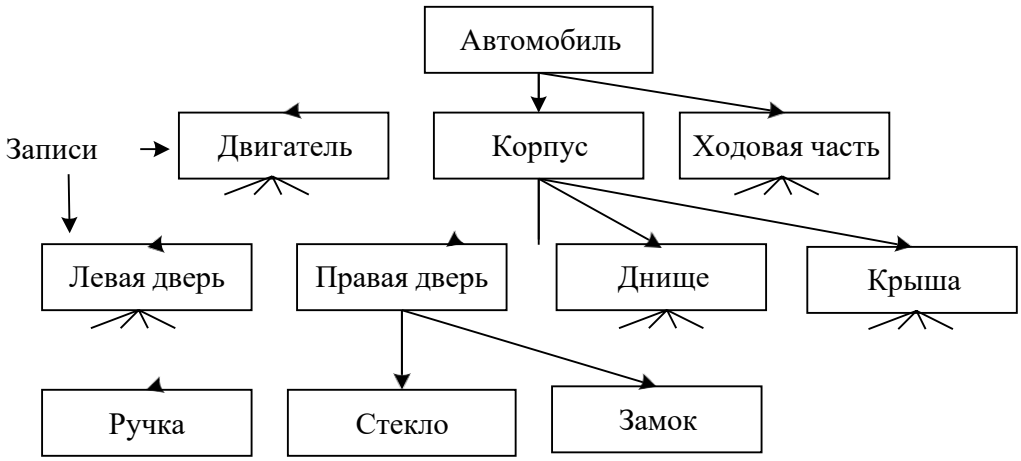


Рис. 1.2. Список составных частей изделия

В этой модели каждая *запись* базы данных представляла конкретную деталь. Между записями существовали *отношения предок/потомок,* связывающие каждую часть с деталями, входящими в нее.

Другими примерами такой иерархии могут быть представление структуры учебной дисциплины (рис. 1.3), где имеются уровни иерархической структуры: дисциплина, раздел, тема, вопрос, или структура организации (директор, заместитель, руководители отделов, сотрудники) (рис.1.4).

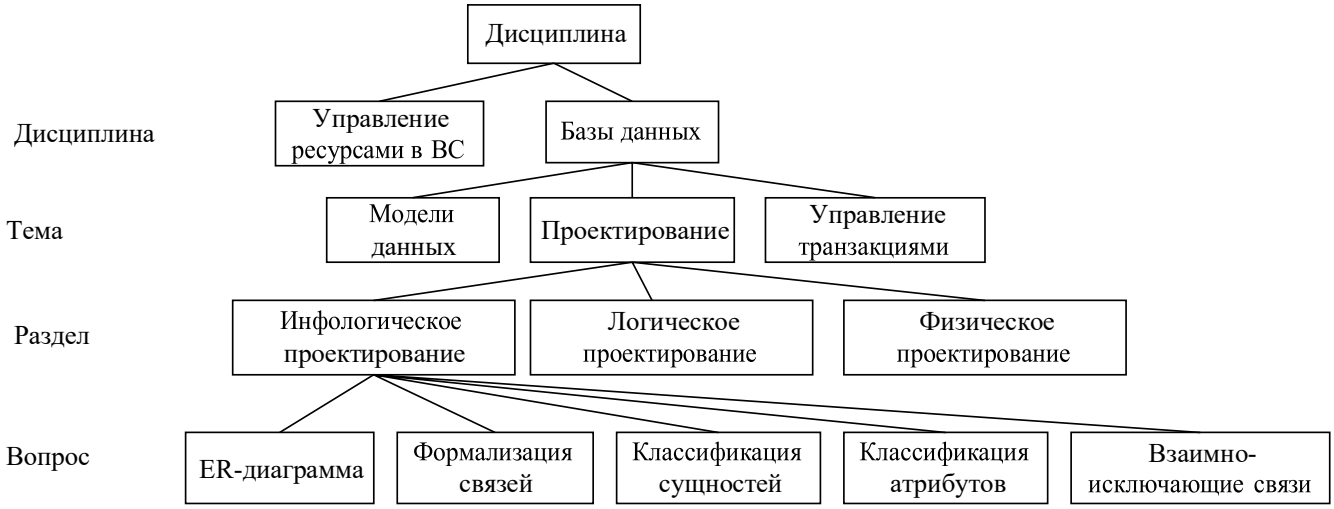


Рис. 1.3. Структура учебной дисциплины

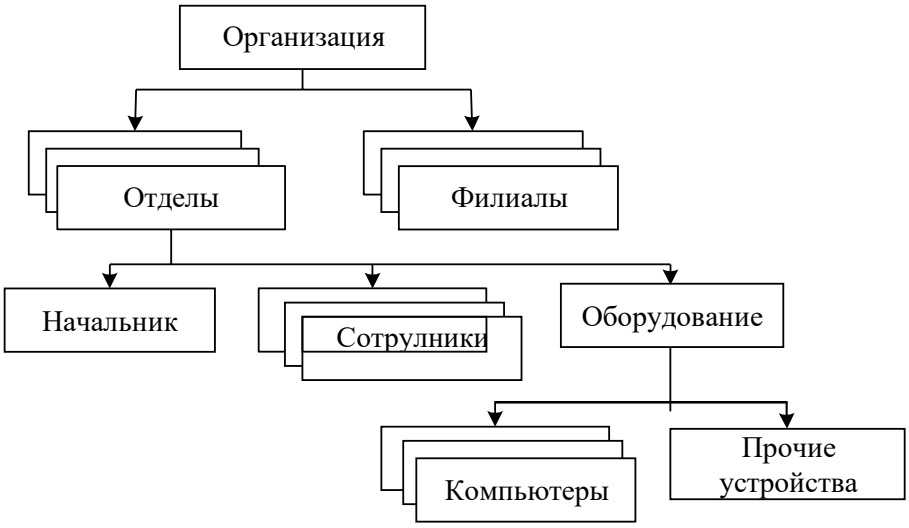


Рис. 1.4. Структура организации

На рис. 1.5 изображена простая иерархическая база данных, в которой фиксируется деятельность исполнителей. Корень дерева представляет собой запись

об исполнителе. Ее потомками являются две записи о счет-фактурах и три записи об оплатах счетов. Структура счета номер 362 уточняется в трех дочерних записях, у счета номер 368 – в одной записи.

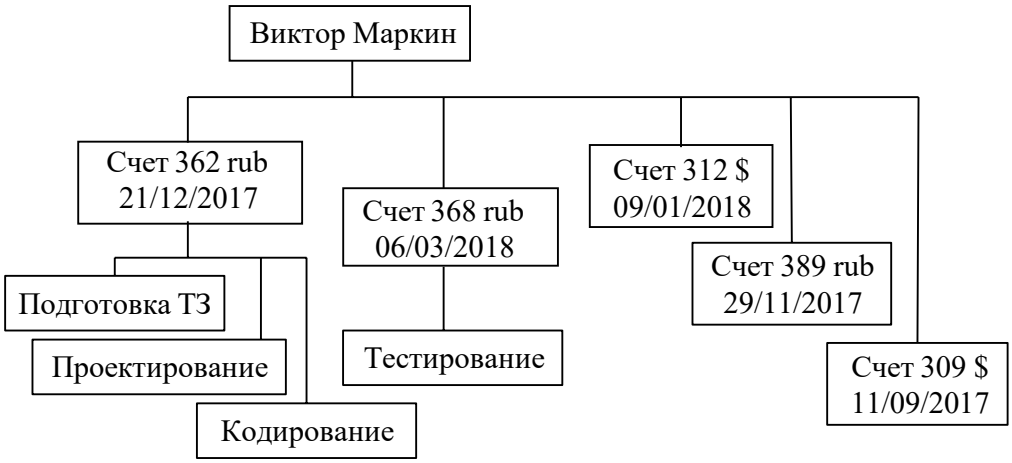


Рис. 1.5. Экземпляры логических записей базы данных организации

В графической диаграмме схемы базы данных вершины используются для интерпретации сущностей, а дуги – для интерпретации связей. При этом возможна ситуация, когда сущность-предок не имеет потомков или имеет их несколько, тогда как у сущности-потомка обязательно должен быть только один предок. Объекты, имеющие общего предка, называются *близнецами*.

Для описания структуры (или, правильнее, схемы) иерархической базы данных на некотором языке программирования используется тип данных *дерево*. Тип данных дерево схож с типами *структура* языков программирования PL/1 или C и *запись* языка Паскаль.

Тип *дерево* является составным. Он может включать в себя подтипы, каждый из которых, в свою очередь является типом *дерево*. Каждый из типов *дерево* состоит из одного *корневого* типа и упорядоченного (возможно, пустого) набора подчиненных типов. Каждый из элементарных типов, включенных в тип *дерево*, является простым или составным типом *запись*. Простая запись состоит из одного типа, например, числового, а составная *запись* объединяет некоторую совокупность типов (целое, строку символов и т.д.).

На рис. 1.6 показан пример дерева, отражающего схему иерархической базы данных. Здесь тип записи «Отдел» является предком для типов записей

«Руководитель» и «Сотрудник», а «Руководитель» и «Сотрудник» – потомки типа записи «Отдел». Между типами поддерживаются связи.

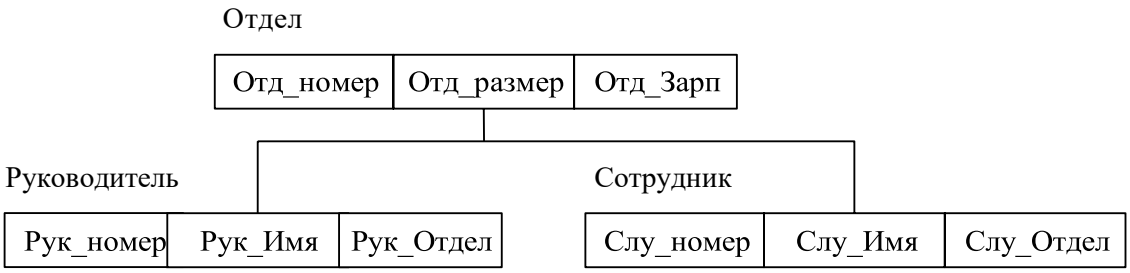


Рис. 1.6. Схема иерархической базы данных организации

Содержимое базы данных с такой схемой могло бы выглядеть, например, как показано на рис. 1.7.

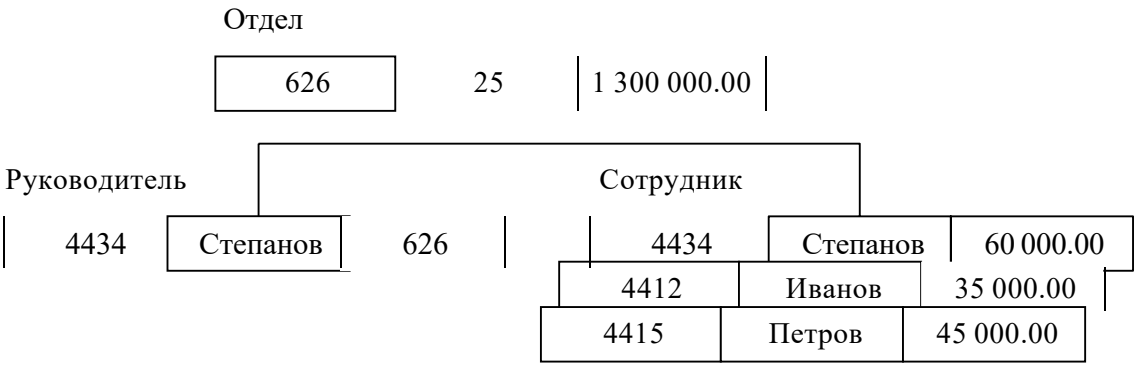


Рис. 1.7. Экземпляры логических записей иерархической базы данных организации

Еще один пример схемы базы данных с иерархической моделью, описывающей структуру больницы, и частично загруженной базы данных с такой схемой показаны на рис. 1.8 - 1.9.

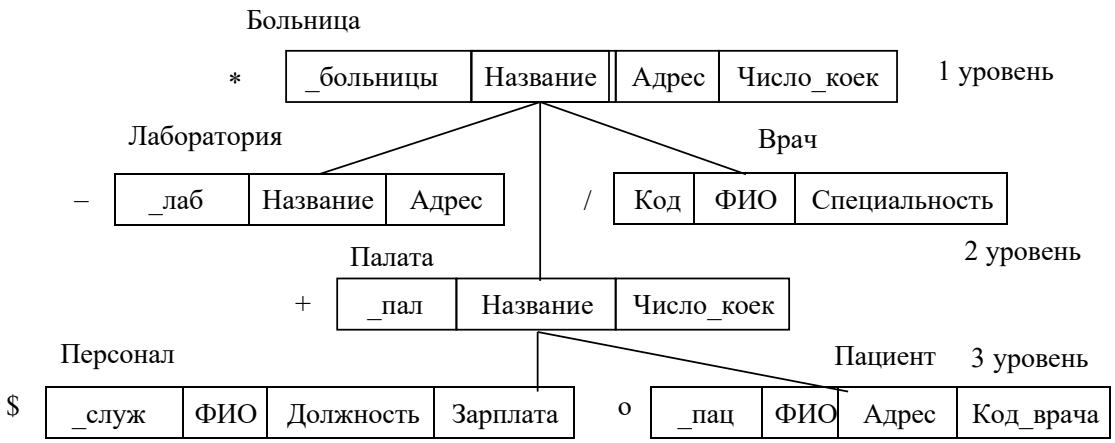


Рис. 1.8. Схема иерархической базы данных больницы

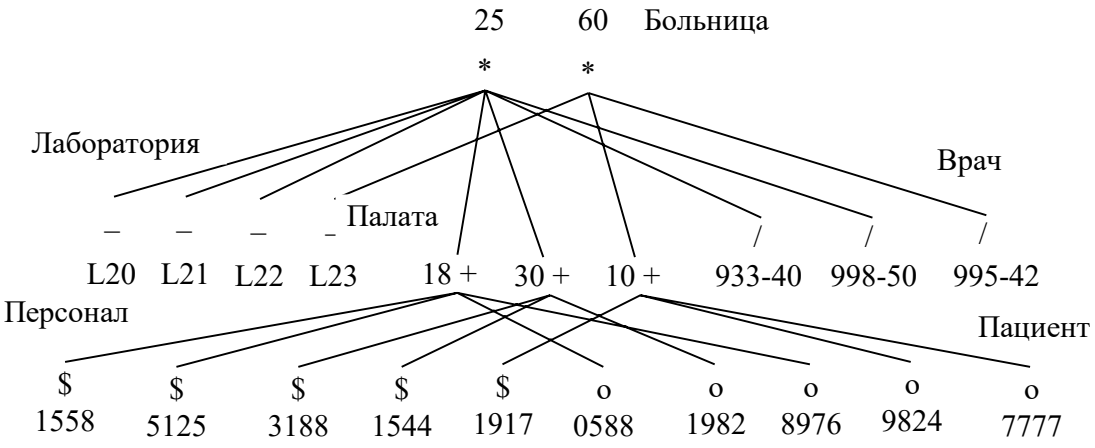


Рис. 1.9. Экземпляры логических записей иерархической базы данных больницы

В иерархических СУБД может использоваться терминология, отличная от приведенной выше. Так в упомянутой системе IMS понятию *запись* соответствует термин *сегмент*, а под *записью базы данных* понимается вся совокупность записей, относящихся к одному экземпляру типа *дерево*.

Иерархические базы данных имеют централизованную структуру, поэтому безопасность данных легко контролировать. К сожалению, определенные знания о физическом порядке хранения записей все же необходимы, так как отношения предок/потомок реализуются в виде физических указателей из одной записи на другую. Это означает, что поиск записи осуществляется методом прямого обхода дерева. Записи, расположенные в одной половине дерева, ищутся быстрее, чем в другой. Отсюда следует необходимость правильно упорядочивать записи, чтобы время их поиска было минимальным.

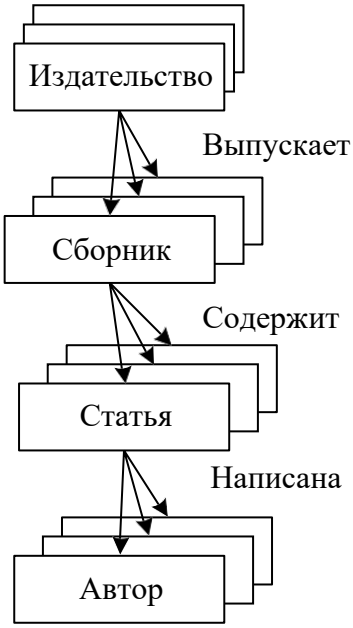
Основные операции манипулирования иерархически организованными данными:

- найти указанное дерево базы данных (например, дерево со значением 40 в поле «№_больницы»);
- перейти от одного дерева к другому;
- перейти от одной записи к другой внутри дерева в порядке иерархии (например, от палаты к первому пациенту);
- перейти от одной записи к другой на одном уровне внутри дерева (например, к следующему пациенту);

- вставить новую запись в указанную позицию;
- удалить текущую запись.

Способы описания схем данных, базирующиеся на иерархической модели, и соответствующие языки манипулирования данными зависят от конкретной реализации.

На рис. 1.10 приведено графическое представления иерархической модели базы данных тематических сборников издательств и описание схемы иерархической модели в системе IMS фирмы IBM.




```
Dbd    Name = Тематические сборники
Segm Name = Издательство
        Field  Name = Название
        Field Name = Адрес
        Field  Name = Счет
Segm  Name =  Сборник, Parent = Издательство
        Field  Name = Название
        Field  Name = Периодичность
        Field  Name = Цена
        Field  Name = Ответственный_редактор
Segm  Name =  Статья, Parent = Сборник
        Field  Name = Название
Segm Name = Автор, Parent = Статья
        Field  Name = ФИО
        Field  Name = Гонорар
```

Рис. 1.10. Описание иерархической базы данных тематических сборников издательств

Язык доступа к данным, который поддерживает IMS, позволяет обращаться к элементам напрямую, зная название и, возможно, дополнительное условие. Например, можно распечатать название всех сборников, ответственным редактором которых является Иванов:

```
Get Unique Сборник Where Ответственный_редактор = «Иванов»
/* получили первый сборник */
While true do
        Print Сборник.Название
        Get next Сборник Where Ответственный_редактор = «Иванов»
End while
```

Выбрав один из сборников в предыдущем примере, можно спуститься «вниз» по иерархии (оператор **Get next Within Parent** позволяет перебрать все элементы–потомки, соответствующие выбранному элементу данного уровня) и просмотреть одну или несколько статей из выбранного сборника.

```
Get Unique Сборник Where Ответственный_редактор = «Иванов»
/* получили первый сборник */
While true do
    Print «Сборник», Сборник.Название
    Get next Within Parent Статья
    While true do
        Print Статья.Название
        Get next Within Parent Статья
    End while
    Get next Сборник Where Ответственный_редактор = «Иванов»
End while
```

Достоинства иерархической модели:

1. *Простота модели.* Принцип построения баз данных в иерархической модели легок для понимания. Иерархия базы данных напоминает структуру компании или генеалогическое дерево.
2. *Использование отношений предок/потомок.* Иерархическая модель позволяет легко представлять отношения предок/потомок, например, «А является частью В» или «А принадлежит В».
3. *Быстродействие.* В иерархической модели отношения предок/потомок реализуются в виде физических указателей из одной записи на другую, поэтому перемещение по базе данных происходит достаточно быстро. Поскольку структура данных в иерархической модели отличается простотой, СУБД может размещать записи предков и потомков на диске рядом друг с другом, что позволяло свести к минимуму количество операций чтения-записи.

Недостатки иерархической модели

1. Операции манипулирования данными в иерархических системах ориентированы прежде всего на поиск информации сверху-вниз, т.е. по данному экземпляру сегмента-отца можно найти все экземпляры сегментов-сыновей. Обратный поиск затруднен, а часто и невозможен. Например, попытка реализовать запрос типа «В скольких сборниках статей опубликовал свои статьи господин Петров?» может оказаться весьма трудной задачей.
2. Дублирование данных на логическом уровне.
3. Для представления связи M:N необходимо дублирование деревьев (рис. 1.11)

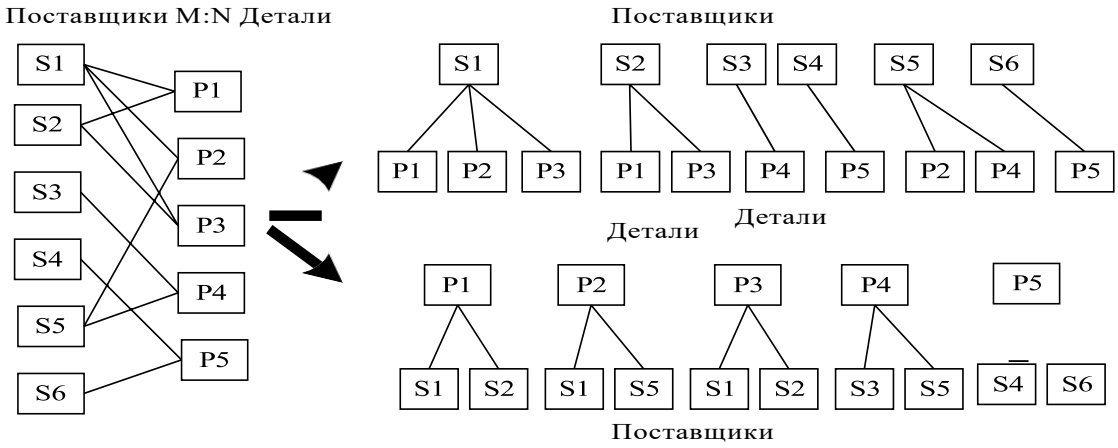


Рис. 1.11. Дублирование деревьев для представления связи M:N

4. В иерархической модели автоматически поддерживается целостность ссылок между предками и потомками по правилу: никакой потомок не может существовать без своего родителя. Целостность по ссылкам между записями, не входящими в одну иерархию, не поддерживается. Поэтому невозможно хранение в базе данных порожденного узла без соответствующего исходного. Аналогично, удаление исходного узла влечет удаление всех порожденных узлов (деревьев), связанных с ним.

Помимо упомянутой СУБД IMS фирмы IBM к иерархическим СУБД относятся также Time-Shared Date Management System (компания Development Corporation), Mark IV MultiAccess Retrieval System (компания Control Data Corporation), System 2000 разработки SAS-Institute, из отечественных в 80-е годы прошлого столетия были широко популярны СУБД Ока, ИНЭС, МИРИС.

Другими примерами использования иерархической модели являются:

- иерархические файловые системы, состоящие из корневого каталога и иерархии подкаталогов и файлов; иерархической базой данных является *Каталог папок Windows*, с которым можно работать, запустив *Проводник ОС Windows*;
- *реестр Windows*, в котором хранится вся информация, необходимая для нормального функционирования компьютерной системы (данные о конфигурации компьютера и установленных драйверах, сведения об установленных программах, настройки графического интерфейса и др.);
- серверы каталогов, такие как LDAP и Active Directory.

Еще одним примером иерархической базы данных является база данных *Доменная система имен компьютеров*, подключенных к Интернету (рис. 1.12).

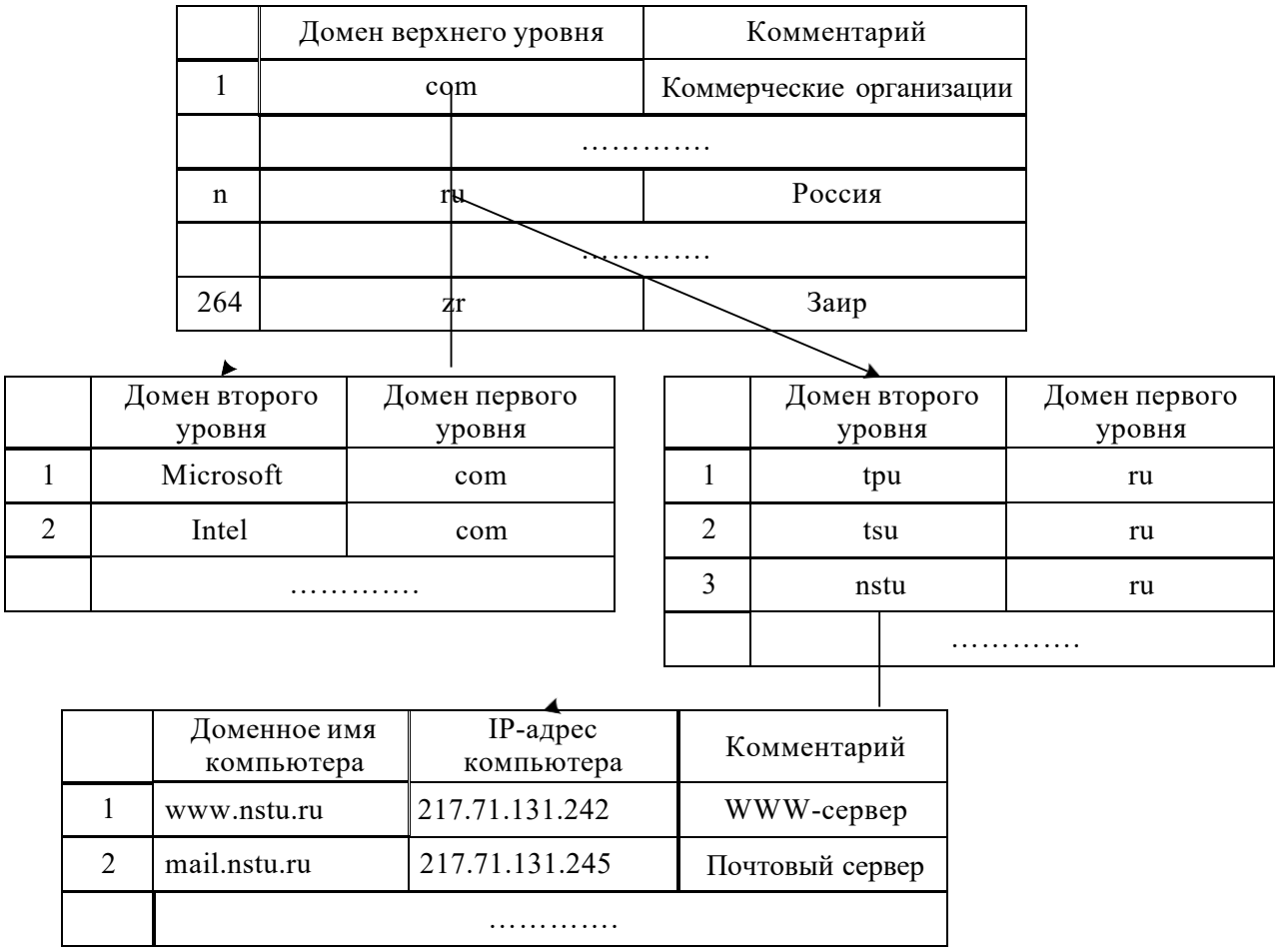


Рис. 1.12. Иерархическая база данных «Доменная система имен»

На верхнем уровне находится табличная база данных, содержащая перечень доменов верхнего уровня, из которых 7 – административные, а остальные 257 – географические. На втором уровне находятся табличные базы данных, содержащие перечень доменов второго уровня для каждого домена первого уровня. На третьем уровне могут находиться табличные базы данных, содержащие перечень доменов третьего уровня для каждого домена второго уровня, и таблицы, содержащие IP-адреса компьютеров, находящихся в домене второго уровня.

Контрольные вопросы и упражнения

1. Каковы структурные элементы иерархической модели данных?
2. Как обеспечивается связь в иерархической модели данных?
3. Как реализуется отношение M:N в иерархической модели данных?
4. Каковы достоинства и недостатки иерархической модели данных?
5. Чем обеспечивается высокая эффективность доступа к данным в иерархической модели данных?
6. Перечислите операции манипулирования иерархически организованными данными.
7. Опишите схему иерархической модели, приведенную на рис. 1.4, на языке СУБД IMS (необходимые поля записей задайте произвольно).
8. Опишите схему иерархической модели, приведенную на рис. 1.8, на языке СУБД IMS (необходимые поля записей задайте произвольно).
9. Как обеспечивается ограничения целостности в иерархической СУБД?
10. Приведите примеры использования иерархических моделей данных.
11. Смоделируйте структуру реляционной базы данных, хранящую схему иерархической модели, приведенную на рис. 1.4.
12. Средствами реляционной базы данных смоделируйте структуру, хранящую схему иерархической модели, приведенную на рис. 1.8.
13. Что понимается под термином сегмент в иерархической модели данных СУБД IMS?

1.2. Сетевая модель данных

Если структура данных оказывалась сложнее, чем простая иерархия, простота организации иерархической базы данных становилась ее недостатком.

Например, в базе данных для хранения заказов один заказ может участвовать в трех различных отношениях предок/потомок, связывающих заказ с клиентом, разместившим его, со служащим, принявшим его, и с заказанным товаром. Такая структура (рис. 1.13) не соответствует иерархической модели.

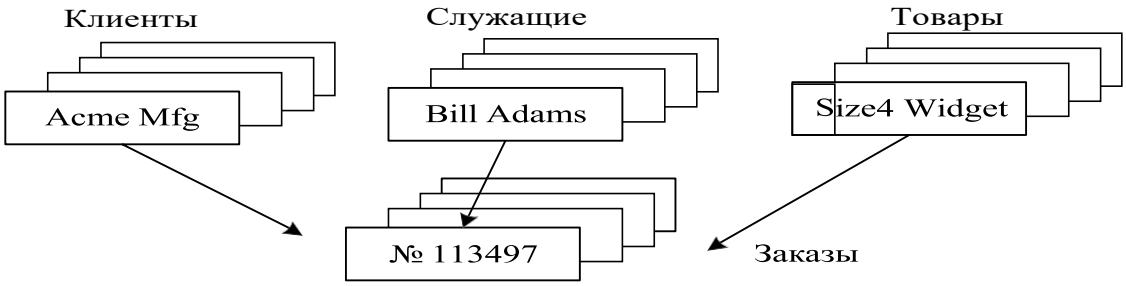


Рис. 1.13. Неиерархическая структура базы данных

Сетевой подход к организации данных является расширением иерархического. Цель разработчиков сетевой модели – создание модели, позволяющей описывать связи M:N, чтобы одна запись могла участвовать в нескольких отношениях предок/потомок (рис. 1.14).

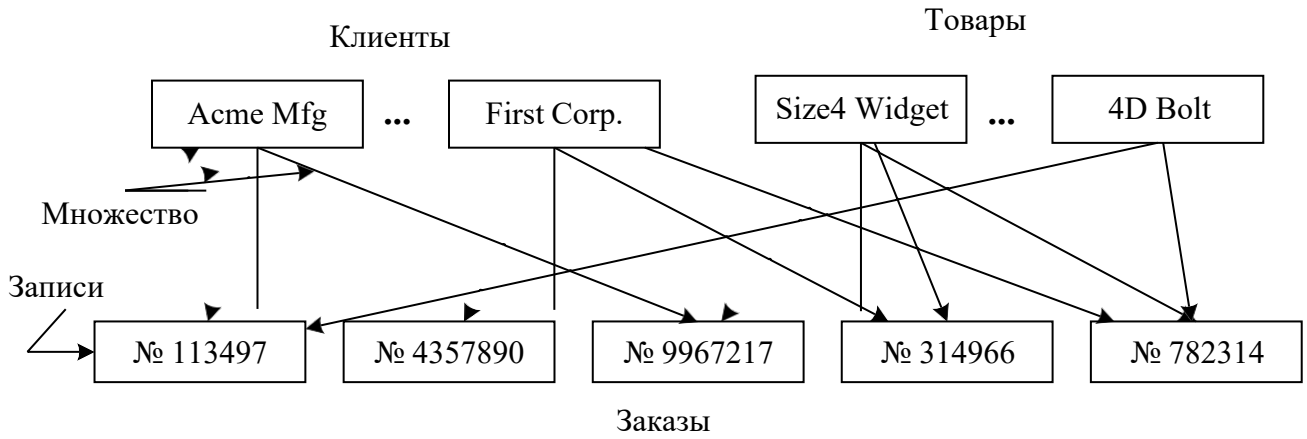


Рис. 1.14. Участие записи в нескольких отношениях предок/потомок

При различных способах реализации сетевых моделей наибольшее распространение получила модель CODASYL (Conference on Data Systems Languages), предложенная группой Data Base Task Group (DBTG) Комитета по языкам программирования, 1971г.

СУБД, построенные на сетевой модели: DMS (компания UNIVAC), DBMS (компания DEC), IDMS (компания Culliname). Из современных сетевых СУБД можно отметить dbVista под DOS и Windows.

Сетевая модель данных базируется также на использовании представления данных в виде графа. С точки зрения теории графов сетевой модели соответствует произвольный граф: если в иерархической модели запись-потомок должна иметь в точности одного предка, то в сетевой модели данных потомок может иметь любое число предков. Вершины графа используются для интерпретации типов объектов, дуги графа используются для интерпретации типов связей между типами объектов.

Структурными элементами сетевой модели данных CODASYL являются элемент данных, агрегат данных, запись, набор записей (рис. 1.15).

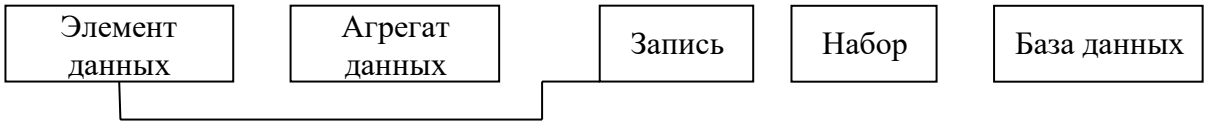


Рис. 1.15. Структурные элементы сетевой модели данных CODASYL

Элемент данных – наименьшая поименованная единица данных (аналог поля в файловых системах). Значение элемента данных может быть числовым (разных типов), символьным, логическим, может быть неопределенным, имя используется для идентификации (табельный номер, шифр детали, год рождения и т.д.).

Агрегат данных – поименованная совокупность элементов данных внутри записи, которую можно рассматривать как единую целую. Имя агрегата используется для его идентификации в схеме структуры более высокого уровня. Агрегат может быть простым, если состоит только из элементов данных, и составным, если включает в другие агрегаты (рис. 1.16).

Дата		
Число	Месяц	Год

Предприятие			
Название	Адрес		
	Почтовый индекс	Город	Улица Номер дома

Рис. 1.16. Примеры агрегатов данных

Различают агрегаты типа *вектор* и типа *повторяющаяся группа*. Если в агрегате, повторяющаяся компонента является простым элементом данных, его называют ***вектором***. Пример вектора – агрегат «Заработная плата», в которой экземпляр элемента данных может повторяться до 12 раз (по числу месяцев в году). Агрегат, повторяющаяся компонента которого представлена совокупностью данных, называется ***повторяющейся группой***. В повторяющуюся группу могут входить отдельные элементы данных, векторы, агрегаты или повторяющиеся группы. На рис. 17 представлен агрегат «Заказ на покупку», имеющий в своем составе повторяющуюся группу «Партия товара».

Заказ на покупку											
Номер заказа	Дата заказа			Партия товара							
	Число	Месяц	Год	Шифр товара	Кол-во	Цена	……	……	Шифр товара	Кол-во	Цена

Рис. 1.17. Пример повторяющейся группы

Запись – поименованная совокупность элементов данных и/или агрегатов, которая не входит в состав никакого другого агрегата (рис. 1.18). Запись может иметь сложную иерархическую структуру, допускает многократное применение агрегации. Имя записи используется для идентификации типа записи в структурах более высокого уровня.

Сотрудник					
Табельный №	ФИО	Дата			Адрес
		День	Месяц	Год	

Рис. 1.18. Пример агрегата *Дата* в записи *Сотрудник*

Связи между записями выполняют так называемые наборы. ***Набор*** – поименованная двухуровневая иерархическая структура, связывающая запись-владельца и записи-членов. Каждый набор должен содержать только один экземпляр записи-владельца и любое количество экземпляров записей-членов (рис. 1.19).

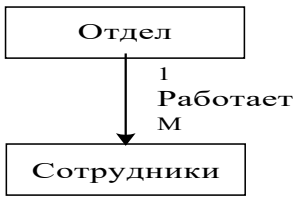


Рис. 1.19. Пример набора, связывающего две записи

Структура сетевой базы данных основана на следующих правилах:

1. База данных содержит любое количество типов записей и типов наборов.
2. Между двумя типами записей может быть определено любое количество типов наборов.
3. Тип записи может быть владельцем и одновременно членом нескольких типов наборов

На рис. 1.20 представлены три типа записей: «Отдел», «Служащие» и «Руководитель» и три типа связей (три набора): «Состоит из служащих», «Имеет руководителя» и «Является служащим».

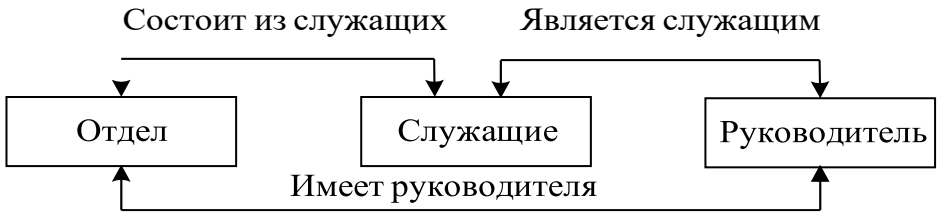


Рис. 1.20. Записи и связи в сетевой модели

Пример схемы сетевой базы данных приведен на рис. 1.21.

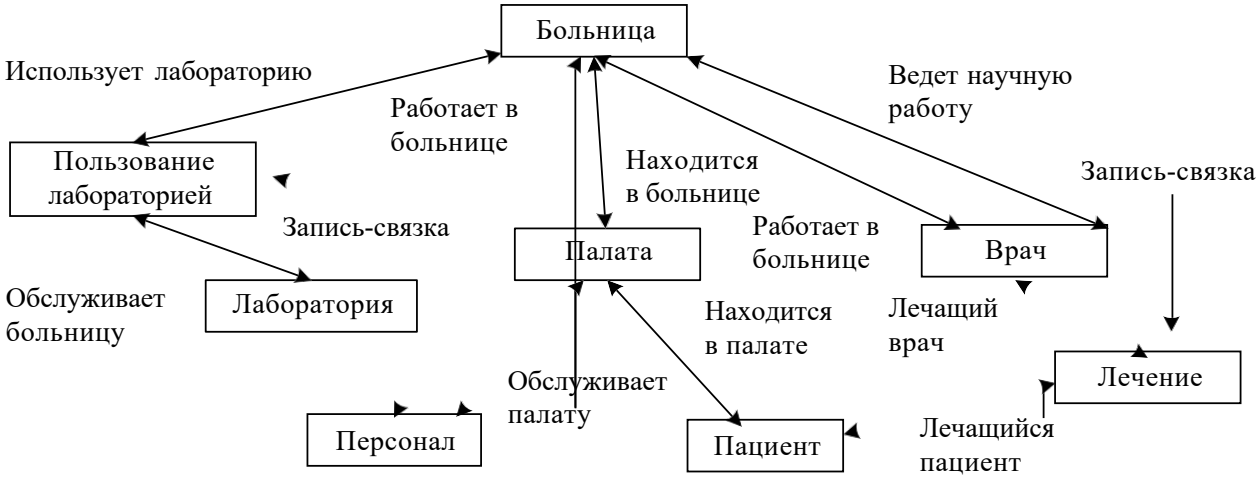


Рис. 1.21. Схема сетевой базы данных

Наборы могут быть нескольких разновидностей:

- С одними и теми же типами записей, но разными типами наборов. На рис. 1.21 присутствуют различные наборы записей между одними и теми же типами записей «Врач» и «Больница» (может быть еще и финансовый аспект).
- Наборы из трех или более записей, в том числе и с обратной связью. На рис. 1.21 это записи «Врач», «Пациент» и «Лечение».
- Так называемые сингулярные наборы (рис. 1.22), у которых нет естественного владельца и в качестве его выступает система.

Рис. 1.22. Сингулярный набор

В дальнейшем такие наборы могут приобрести запись-владельца.

- Наборы между несколькими типами записей, называемые многочленными наборами, которые представляют собой отношение между тремя или более типами записей, один из которых назначается владельцем, а остальные – членами набора. На рис. 1.23 приведен пример многочленного набора, владельцем которого является запись типа «Лектор», а членами – записи типа «Методическая работа», «Научные труды», «Тезисы докладов».

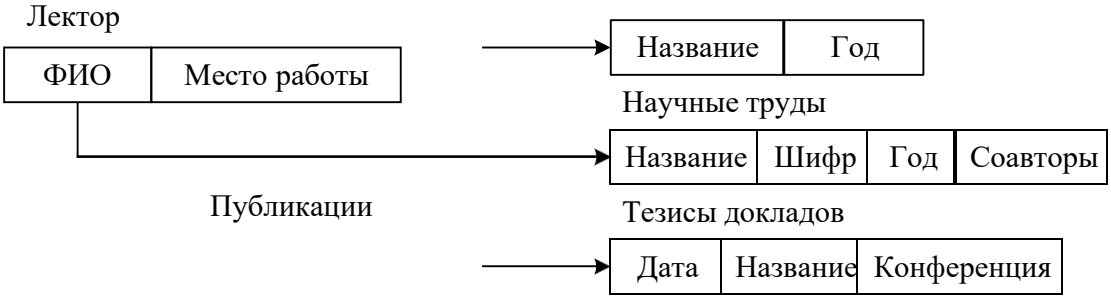
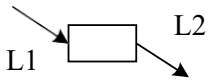


Рис. 1.23. Многочленный набор

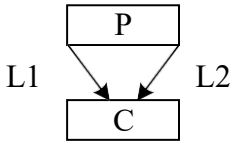
Ограничения на типы записей и связей сетевой модели:

1. Все типы связей должны быть функциональными (1:1, 1:M, M:1).
2. Экземпляр записи может быть членом только одного экземпляра набора среди всех экземпляров набора одного типа (он может входить в состав двух и более экземпляров наборов, но разных типов).
3. Экземпляр записи может быть потомком в одном наборе L1 и предком в другом наборе L2.

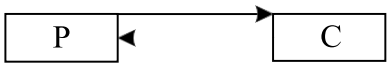


4. Экземпляр записи P может быть предком в любом числе наборов, и аналогично, может быть потомком в любом числе наборов.

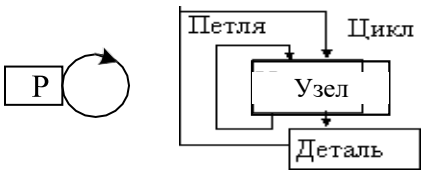
5. Может существовать любое число типов наборов с одним и тем же типом записи предка и одним и тем же типом записи потомка. Например, если L1 и L2 – два типа набора с одним и тем же типом записи предка P и одним и тем же типом записи потомка C, то правила, по которым образуется родство, в разных связях могут различаться.



6. Записи P и C могут быть предком и потомком в одной наборе, и потомком и предком – в другом.



7. Предок и потомок могут быть одного типа записи. Сетевая модель может содержать циклы, когда предшествующая вершина является в то же время предыдущей. Связь записей одного типа называется петлей.



8. Для представления связи M:N вводится дополнительный тип записи и две функциональные связи типа 1:M и M:1. При необходимости запись-связка может содержать дополнительную информацию (рис. 1.24).

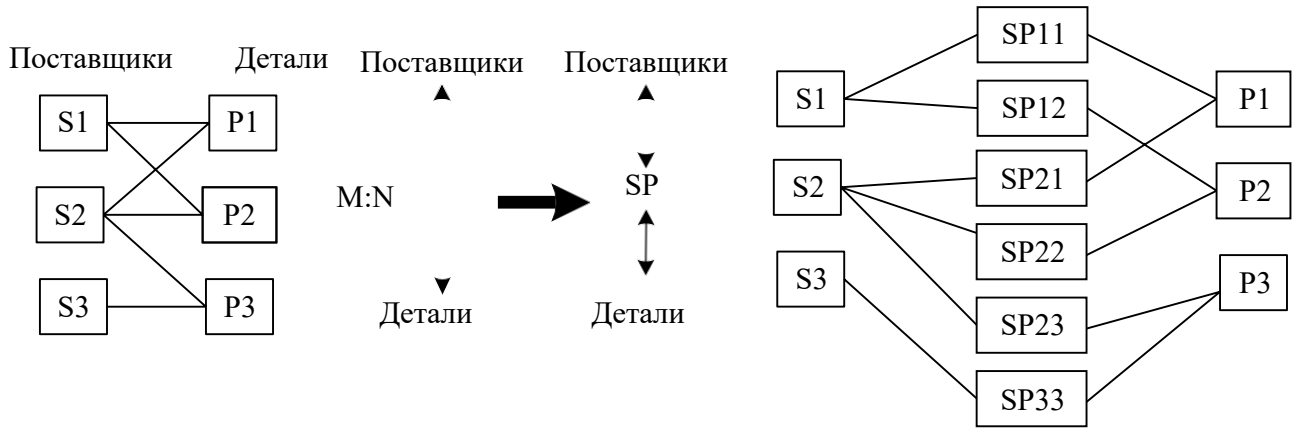


Рис. 1.24. Представление связи M:N

Операции манипулирования данными:

- найти запись в наборе однотипных записей (палату с указанным номером);
- перейти от предка к первому потомку по некоторой связи (к первому врачу некоторой больницы);
- перейти к следующему потомку в некоторой связи (от врача Сидорова к Иванову);
- перейти от потомка к предку по некоторой связи (найти больницу врача Сидорова);
- создать новую запись, уничтожить запись, модифицировать запись;
- включить в связь, исключить из связи;
- переставить в другую связь и т.д.

Если вернуться к примеру с изданием тематических сборников и попытаться расширить его, чтобы он более полно соответствовал реальным взаимоотношениям, то схема сетевой модели будет выглядеть как показано на рис. 1.25.

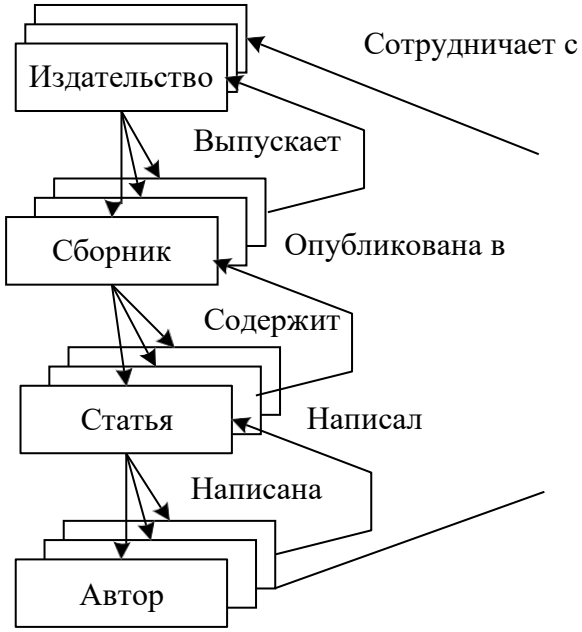


Рис. 1.25. Схема сетевой базы данных тематических сборников издательств

Описание приведенной сетевой схемы на языке CODASYL в этом случае примет вид:

Record name is Издательство

01 Название Type is Character 30

01 Адрес Type is Character 30

01 Счет Type is Picture "999999999"

Record name is Сборник

01 Название Type is Character 30

01 Периодичность Type is fixed

01 Цена Type is fixed

01 Ответственный_редактор Type is Character 20

Record name is Статья

01 Название Type is Character 80

Record name is Автор

01 ФИО Type is Character 20

01 Гонорар Type is fixed

Set name is Выпускает

Owner is Издательство

Member is Сборник

Set name is Содержит

Owner is Сборник

Member is Статья

. . . . . .

Любая сетевая структура может быть приведена к более простому виду путем введения избыточности (рис. 1.26). В некоторых случаях возникающая при этом избыточность мала и является допустимой, в других случаях она может быть чрезмерной.

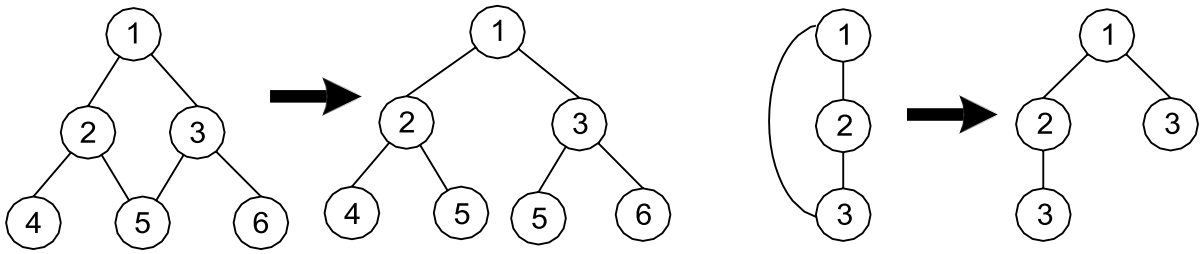


Рис. 1.26. Преобразование сетевой модели путем введения избыточности

Контрольные вопросы и упражнения

1. Каковы структурные элементы сетевой модели данных?
2. Что такое элемент данных? Агрегат? Запись?
3. Перечислите основные виды агрегатов.
4. Как обеспечивается связь между записями в сетевой модели данных?
5. Каковы правила построения сетевой базы данных?
6. Перечислите ограничения на типы записей и связей сетевой модели.
7. Приведите пример многочленного набора данных, отличного от приведенного на рис. 1.23.
8. Приведите пример, в котором между двумя типами записей может быть определено несколько типов наборов.
9. Приведите пример, в котором записи являются предком и потомком в одной наборе, и потомком и предком – в другом.
10. Каковы достоинства и недостатки сетевой модели данных?
11. Перечислите операции манипулирования данными сетевой модели.
12. Опишите схему сетевой модели базы данных, приведенной на рис. 1.21, на

языке CODASYL (необходимые поля записей задайте произвольно).

13. Как задаются ограничения целостности в сетевой СУБД?
14. Средствами реляционной базы данных смоделируйте структуру, хранящую схему сетевой модели, приведенной на рис. 1.21.

1.3. Системы, основанные на инвертированных списках

Сложность практического использования иерархических и сетевых СУБД заставляла искать иные способы представления данных. В конце 60-х годов появились СУБД на основе инвертированных файлов, отличающиеся простотой организации и наличием весьма удобных языков манипулирования данными.

Известными представителями таких систем являются Datacom/DB (компания Applied Data Research, Inc.) и Adabas (компания Software AG). Они применяются, как правило, на больших компьютерах фирмы IBM.

Организация доступа к данным на основе инвертированных списков используется практически во всех современных реляционных СУБД, но в реляционных СУБД пользователи не имеют непосредственного доступа к инвертированным спискам (индексам).

База данных на инвертированных списках похожа на реляционную базу данных, то есть также состоит из таблиц отношений, однако ей присущи важные отличия:

- допускается сложная структура атрибутов (атрибуты не обязательно атомарны);
- строки таблиц (записи) упорядочены в некоторой последовательности, каждой строке присваивается уникальный номер; физическая упорядоченность строк всех таблиц может определяться и для всей бпзы данных (так делается, например, в СУБД Datacom/DB);
- пользователям видны и хранимые таблицы, и пути доступа к ним;
- пользователь может управлять логическим порядком строк в каждой таблице с помощью специального инструмента – индексов; эти индексы автоматически поддерживаются системой и явно видны пользователям.

Некоторые атрибуты могут быть объявлены поисковыми (ключевыми), для каждого из них создается индекс, который содержит упорядоченные значения ключей и указатели на соответствующие записи основной таблицы (инвертированный список). Если таблицу требуется упорядочить по нескольким ключам, то создается несколько индексов. Если, например, в основной таблице с данными, представленной на рис. 1.27, ключевыми атрибутами являются атрибуты «Фамилия» и «Группа», то инвертированные списки выглядят так, как показано на рис. 1.28.

Адреса записей	**Имена столбцов (поля)**		
	Номер зачетной книжки	**Фамилия**	**Группа**
100	052693109	Гриценко	ПМИ-61
220	052693020	Медведев	ПМИ-62
250	052387415	Нотова	ПМ-41
300	050102608	Матвеев	ПМ-41
400	052931417	Мастихин	ПМ-71
440	050102212	Медведев	ПМ-22
530	052693216	Матвеев	ПМИ-62
600	052931712	Паничев	ПМИМ-71
700	052931107	Кочанов	ПМИ-71

Рис. 1.27. Основная таблица с данными

Ключевой атрибут	**Индекс**	**Адреса записей (инвертированные списки)**
Фамилия	Гриценко	100
	Кочанов	700
	Мастихин	400
	Матвеев	300, 530
	Медведев	220, 440
	Нотова	250
	Паничев	600

Группа	ПМИМ-71	600
	ПМИ-61	100
	ПМИ-62	220, 530
	ПМИ-71	700
	ПМ-22	440
	ПМ-41	250, 300
	ПМ-71	400

Рис. 1.28. Индексы и инвертированные списки

Тогда для поиска записей по условию Фамилия = «Медведев» и Группа = «ПМИ-62» достаточно найти пересечение списков с соответствующими индексами: адрес искомой записи = (220, 440) ^ (220, 530) = 220.

Системой поддерживаются два класса операций над данными:

- поиск адреса записи по некоторому пути доступа и некоторому условию,
- обновление, удаление или выборка записи с заданным адресом.

Типичный набор операторов:

- LOCATE FIRST – найти первую запись таблицы T в физическом порядке; возвращается адрес записи;
- LOCATE FIRST WITH SEARCH KEY EQUAL – найти первую запись таблицы T с заданным значением ключа поиска K; возвращается адрес записи;
- LOCATE NEXT – найти первую запись, следующую за записью с заданным адресом в заданном пути доступа; возвращается адрес записи;
- LOCATE NEXT WITH SEARCH KEY EQUAL – найти следующую запись таблицы T в порядке пути поиска с заданным значением K; при этом должно быть установлено соответствие между используемым способом сканирования и ключом K; возвращается адрес записи;
- LOCATE FIRST WITH SEARCH KEY GREATER – найти первую запись таблицы T в порядке ключа поиска K со значением ключевого поля, большим заданного значения K; возвращается адрес записи;

- RETRIVE – выбрать запись с указанным адресом;
- UPDATE – обновить запись с указанным адресом;
- DELETE – удалить запись с указанным адресом;
- STORE – включить запись в указанную таблицу; операция генерирует адрес записи.

К достоинствам рассмотренного метода построения базы данных следует отнести

- более быстрый поиск (особенно поиск уникальной записи по нескольким условиям),
- возможность хранения элементов данных со сложной структурой.

Недостаток модели – отсутствие строгого математического аппарата, отсутствие средств для описания ограничений целостности базы данных, и, как следствие – большая трудоемкость программирования запросов к базе данных. В некоторых системах поддерживаются ограничения уникальности значений некоторых полей, но в основном все возлагается на прикладную программу. Кроме этого, такие СУБД обладают рядом ограничений на количество файлов для хранения данных, количество связей между ними, длину записи и количество ее полей.

Однако, СУБД Adabas хороша в задачах, в которых надо из очень большого объёма данных выбрать по сложному запросу относительно небольшое количество данных (например, найти автомобиль, иномарка, то ли серого, то ли голубого цвета, седан, номер московский, цифровая его часть оканчивается на 78, водитель среднего возраста, в очках).

Контрольные вопросы и упражнения

1. Что представляет собой инвертированный список?
2. Как строится инвертированный список?
3. Перечислите набор операторов систем на основе инвертированных списков.
4. Каковы достоинства и недостатки систем на основе инвертированных списков?

5. Каким образом инвертированные списки используются в реляционных СУБД?

2. Реляционная модель данных

Реляционная модель предложена Э. Коддом (E. Kodd, математик, сотрудник IBM, 1970г., статья «A Relational Model of Data for Large Shared Data Banks») на основе теории отношений (relation) и опирается на систему понятий, важнейшими из которых являются тип данных, домен, атрибут, кортеж, первичный и внешний ключ. Принципы, используемые в реляционной модели, вытекают из понятия n-арного отношения, представляющего собой подмножество декартового произведения.

Факторы, обеспечившие быстрое распространение реляционной модели:

- с прагматической точки зрения база данных представляется в виде двумерных таблиц (отношений), обработка в которых не зависит от организации хранения данных в памяти;
- с математической точки зрения реляционная база данных – конечный набор отношений различной арности, являющихся областью приложений математической логики, теории множеств и общей алгебры; замкнутость реляционной модели (операции над отношениями дают отношение) обеспечивает основу для интерпретации выводимости, избыточности и непротиворечивости данных.
- наличие небольшого набора абстракций, которые позволяют сравнительно просто моделировать большую часть распространенных предметных областей и допускают точные формальные определения.

2.1. Основные понятия реляционной модели данных

Основными понятия реляционных баз данных являются тип данных, домен, атрибут, кортеж, отношение, первичный ключ (рис. 2.1).

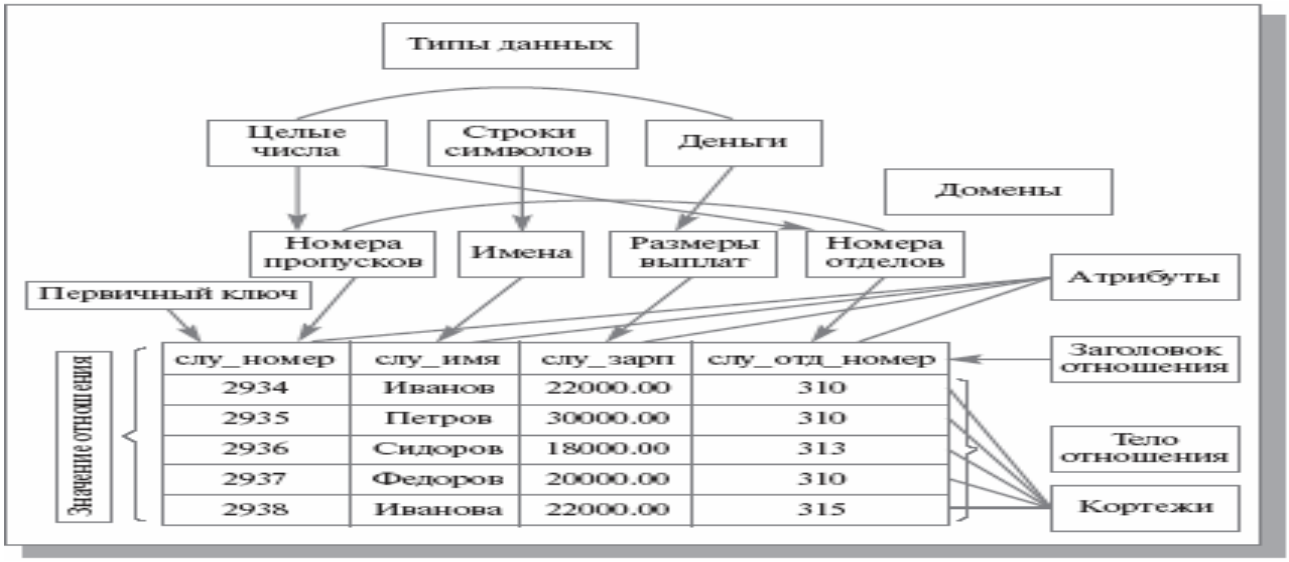


Рис. 2.1. Основные понятия реляционной модели данных

Понятие ***тип данных*** в реляционной модели данных полностью адекватно понятию типа данных в языках программирования. Обычно в современных реляционных базах данных допускается хранение символьных, числовых данных, битовых строк, специализированных числовых данных (таких как «деньги»), а также специальных «темпоральных» (temporal) данных (дата, время, временной интервал). В примере на рис.2.1 представлены данные трех типов: строки символов, целые числа и «деньги».

Домен – допустимое потенциальное множество значений данного типа. В самом общем виде домен определяется заданием некоторого базового типа данных, к которому относятся элементы домена, и произвольного логического выражения, применяемого к элементу типа данных. Если вычисление этого логического выражения дает результат «истина», то элемент данных является элементом домена. Например, домен «Номера пропусков» в приведенном примере определен на базовом типе целых чисел, но в число его значений могут входить только те целые числа, которые являются номерами существующих отделов. Семантическая нагрузка понятия домена состоит в том, что данные считаются сравнимыми только в том случае, когда они относятся к одному домену. В приведенном примере значения доменов «Номера пропусков» и «Номера отделов» относятся к типу целых чисел, но не являются сравнимыми.

Каждый домен описывает некоторый атрибут. ***Атрибут*** – некоторая характеристика объекта (сущности). Атрибуты имеют имена, через которые к ним производится обращение. Имя атрибута должно быть уникальным внутри отношения.

Таким образом, имеется совокупность множеств D_1, D_2,. . . D_n, не обязательно различных, называемых доменами n-арного отношения. Совокупность значений $<d_1, d_2, \ldots d_n,>$, где $d_i \in D_i$, i=1,n называется ***кортежем.*** Множество всех кортежей $<d_1, d_2, \ldots d_n,>$, где $d_i \in D_i$, i=1,n арности n составляет декартово произведение множеств D_1 x D_2 x . . . x D_n. Тогда ***отношение R*** на множествах D_1, D_2,. . . D_n – это некоторое подмножество кортежей арности n декартового произведения доменов D_1 x D_2 x . . . x D_n.

Схема отношения (заголовок отношения) – именованное конечное множество упорядоченных пар вида <A, T>, где A является именем атрибута, а T обозначает имя некоторого базового типа или ранее определенного домена. Иногда схему отношения обозначают множеством атрибутов, предполагая, что атрибут несет информацию о домене.

Сотрудники ({Слу_номер, номера пропусков}, {Сотр_имя, имена},
{Сотр_зарпл, размер з/п}, {Отдел_ном, номера отделов})

Число атрибутов отношения носит название ***степени*** или «арности» схемы отношения (степень отношения **Сотрудники** равна 4).

Мощность тела отношения – число кортежей отношения.

Реляционная база данных – это конечное множество отношений, записываемое в виде:

$\mathbf{R_1(A_{11}, A_{12} \ldots A_{1n})}$ где A_{ij} пара {имя атрибута, имя домена}

. . . . . .

$\mathbf{R_m(A_{m1}, A_{m2} \ldots A_{mz})}$

Обычным житейским представлением отношения является таблица, заголовком которой является схема отношения, а строками – кортежи отношения; в этом случае имена атрибутов именуют столбцы этой таблицы.

С математической точки зрения отношение является множеством и не может содержать совпадающих элементов, а, следовательно, в любой момент времени никакие два кортежа отношения не могут быть дубликатами друг друга. Чтобы это обеспечить, в отношении должен присутствовать некоторый атрибут (или набор атрибутов), однозначно определяющий каждый кортеж отношения и обеспечивающий уникальность кортежей. Атрибут, значение которого идентифицирует кортеж, называется ***ключом*** отношения (в отношении **Сотрудники** это атрибут **Слу_номер**). Если отношение имеет несколько ключей, один из них объявляется ***первичным ключом*** (primary key).

Свойства первичного ключа:

- *уникальность*: в любой момент времени никакие два кортежа отношения не должны иметь одно и то же значение;
- *минимальность*: ни один из атрибутов не может быть исключен из набора атрибутов первичного ключа без нарушения свойств уникальности.

В зависимости от количества атрибутов, ходящих в ключ, различают простые и сложные (составные) ключи.

Простой ключ – ключ, содержащий только один атрибут. Как правило, в качестве него используют самый короткий и простой из возможных типов данных (целочисленный тип), при этом операции, использующие ключ (операции объединения), выполняются значительно быстрее.

Сложный (составной) ключ – ключ, состоящий из нескольких атрибутов.

Суперключ – сложный ключ, с большим числом атрибутов, не удовлетворяющий свойству минимальности. Используется крайне редко, когда избыточность может оказаться полезной пользователю.

С точки зрения информативности атрибута (атрибутов), составляющего первичный ключ, различают искусственные и естественные ключи.

Искусственный* или *суррогатный ключ – ключ создаваемый самой СУБД или пользователем с помощью некоторой процедуры, который сам по себе не содержит информации. Используется для создания уникальности идентификаторов

строк. Им так же заменяют слишком сложные ключи. Как правило, пользователю они не показываются.

Естественный ключ – ключ, содержащий только значимые атрибуты, содержащие информацию.

К достоинствам естественных ключей можно отнести то, что они несут вполне определенную информацию, и их использование не приводит к необходимости добавлять к таблице атрибуты, значения которых для пользователя не несут никакого смысла и используются только для связи между отношениями, что позволяет получить более компактную форму таблиц.

Основным же недостатком естественных ключей является то, что их использование весьма затруднительно в случае изменения предметной области. Значения атрибутов первичного ключа не должно изменяться, и однажды заданное значение первичного ключа для кортежа не может быть изменено. Это требование необходимо для поддержания ссылочной целостности базы данных, которая устанавливается по первичному ключу.

Другим недостатком естественных ключей является то, что, как правило, они являются составными и содержат строковые атрибуты, что сказывается на скорости выполнения операций над данными.

В любой из таблиц может оказаться несколько наборов атрибутов, которые можно выбрать в качестве ключа, такие наборы называются ***потенциальными*** и ***возможными*** ключами.

Вторичные ключи – ключи, отличные от комбинации атрибутов первичного ключа. Они могут не обладать свойством уникальности. Наконец, ***перекрывающиеся ключи*** – сложные ключи, которые имеют один или несколько общих столбцов.

Выбор ключа – это не формальный момент. От того, что выбрано в качестве ключа зависят задаваемые ограничения целостности модели. Если, например, в отношении

Сотрудники (Слу_номер, Сотр_имя, Сотр_зарпл, Отдел_ном)

первичным ключом определен атрибут – **Слу_номер**, это означает, что в организации не должно быть сотрудников с одинаковыми табельными номерами. Если в этом же отношении первичным ключом определен составной ключ **{Слу_номер, Отдел_ном}**, это означает, что в разных отделах могут работать служащие с одинаковыми номерами, но в каждом отделе номера служащих различны.

Для организации взаимосвязи отношений используется понятие внешнего ключа (foreign key). Атрибут называется ***внешним ключом***, если его значения однозначно характеризуют сущности, представленные кортежами некоторого другого отношения, заданные своим первичным ключом (рис. 2.2).

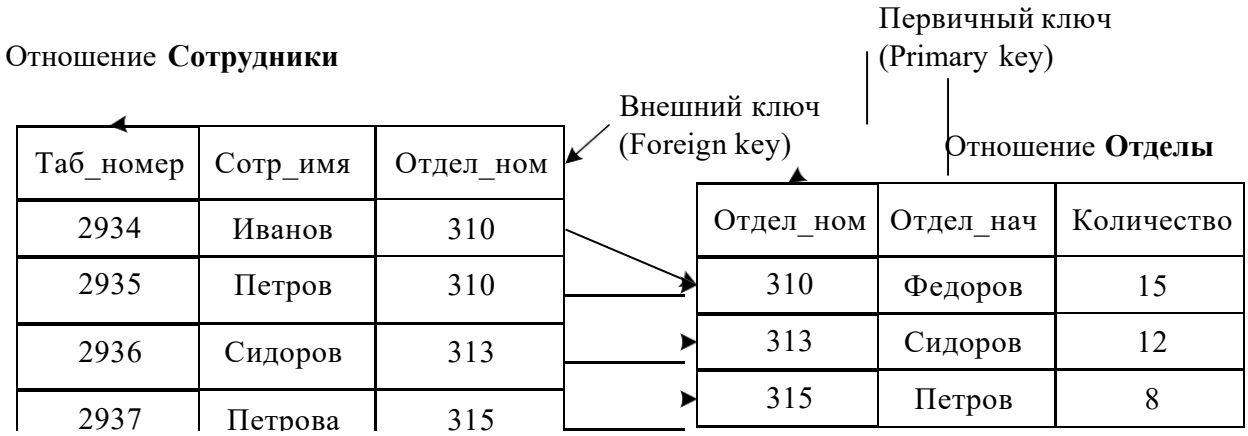


Рис. 2.2. Связь отношений через внешний ключ

При проектировании базы данных следует различать понятия базового и производного отношения. *Базовое отношение* – это отношение, которое содержит один или несколько атрибутов, характеризующих свойства объекта, а также первичный ключ, *производное отношение* – это отношение, которое не характеризует свойства объекта и используется для обеспечения связей между другими таблицами.

2.2. Фундаментальные свойства отношений

1. Отсутствие кортежей-дубликатов.

Данное свойство следует из определения отношения как множества кортежей (в классической теории множеств по определению каждое множество

состоит из различных элементов). Отсюда же вытекает наличие у каждого отношения первичного ключа.

Во многих практических реализациях реляционных СУБД допускается нарушение свойства уникальности кортежей для промежуточных отношений, порождаемых неявно при выполнении запросов. Такие отношения являются не множествами, а мультимножествами.

2. Отсутствие упорядоченности кортежей.

Это свойство также является следствием определения отношения как множества кортежей.

3. Отсутствие упорядоченности атрибутов.

Атрибуты отношений не упорядочены, поскольку по определению схема отношения есть множество пар {имя атрибута, имя домена}.

4. Атомарность значений атрибутов.

Базовым требованием для классических реляционных баз данных является требование нормализованных отношений или отношений, представленных в первой нормальной форме. А отношения, представленные в первой нормальной форме, предполагают атомарность (неделимость) атрибутов. Исходный вариант отношения **Сотрудники** на рис. 2.1 является нормализованным отношением. Потенциальный пример ненормализованного отношения приведен на рис. 2.3. Значением атрибута **Отдел** в нем является отношение.

НОМЕР_ОТДЕЛА	ОТДЕЛ		
	СЛУ_НОМЕР	СЛУ_ИМЯ	СЛУ_ЗАРП
310	2934	Иванов	22000.00
	2935	Петров	30000.00
	2937	Федоров	20000.00
313	2936	Сидоров	18000.00
315	2938	Иванова	22000.00

Рис. 2.3. Пример ненормализованного отношения

Хотя нормализованные отношения обладают некоторыми ограничениями (не любую информацию удобно представлять в виде плоских таблиц), они существенно упрощают манипулирование данными.

Рассмотрим два идентичных оператора занесения кортежа:

- Зачислить сотрудника Кузнецова (Таб. номер 3000, зарплата 115,000) в отдел номер 320
- Зачислить сотрудника Кузнецова (Таб. номер 3001, зарплата 115,000) в отдел номер 310.

Если информация о сотрудниках представлена в виде нормализованного отношения **Сотрудники**, оба оператора будут выполняться одинаково (вставить кортеж в отношение **Сотрудники**). Если же работать с ненормализованным отношением **Отделы**, то первый оператор выразится в занесение кортежа, а второй – в добавление информации о Кузнецове в множественное значение атрибута **Отдел** кортежа с первичным ключом 310.

Наиболее распространенная трактовка реляционной модели данных принадлежит К. Дейту. Согласно Дейту реляционная модель состоит из трех частей, описывающих разные аспекты реляционного подхода:

- структурная часть,
- манипуляционная часть
- целостная часть.

В структурной части модели фиксируется, что единственной структурой данных, используемой в реляционных базах данных, является нормализованное n-арное отношение.

В манипуляционной части модели утверждаются два фундаментальных механизма манипулирования реляционными базами данных – реляционная алгебра и реляционное исчисление. Первый механизм базируется, в основном, на классической теории множеств и является процедурным языком: задает алгоритм (перечень операций) получения результата, а второй механизм базируется на классическом логическом аппарате исчисления предикатов первого порядка и

является непроцедурным языком, указывая свойства результата, но не указывая алгоритма его достижения.

В целостной части реляционной модели данных фиксируются два базовых требования целостности, которые должны поддерживаться в любой реляционной СУБД.

1. Первое требование называется требованием ***целостности сущностей*** (*entity integrity*). Любой сущности в реляционной базе данных соответствует кортеж отношения. И требование состоит в том, чтобы любой кортеж любого отношения был отличим от любого другого кортежа этого отношения. Это означает, что любое отношение должно обладать первичным ключом.

Данное требование автоматически удовлетворяется, если не нарушаются базовые свойства отношений.

Более строго, требование целостности сущностей полностью звучит следующим образом: в любом отношении должен существовать первичный ключ, и никакое значение первичного ключа в кортежах отношения не должно содержать неопределенных значений.

На практике не все характеристики сущности могут быть известны к тому моменту, когда требуется зафиксировать сущность в базе данных. Простой пример – процедура принятия на работу человека, размер заработной платы которого еще не определен. В этом случае сотрудник отдела кадров, который заносит в отношение **СЛУЖАЩИЕ** кортеж, описывающий нового служащего, просто не может ввести значение атрибута **СЛУ_ЗАРП**, поскольку любое значение будет неверным.

Э.Кодд предложил использовать в таких случаях неопределенные значения. ***Неопределенные значения*** **(*Null-значения)*** не принадлежит никакому типу данных и могут присутствовать среди значений любого атрибута, определенного на любом типе данных, если только это явно не запрещено при определении атрибута.

Поскольку требование целостностей сущности означает, что первичный ключ должен полностью идентифицировать каждую сущность, первичный ключ не допускается наличие неопределенных значений.

В классической реляционной модели это требование распространяется и на *возможные (уникальные) ключи (Unique)*. В SQL-ориентированных СУБД это требование для возможных ключей не поддерживается. Возможный ключ может содержать Null-значения, однако трактовка Null-значений уникальных ключей в различных СУБД может существенно различаться.

2. Второе требование называется ***требованием целостности по ссылкам*** **(***referential integrity)*.

Требование целостности по ссылкам, или требование внешнего ключа состоит в том, что для каждого значения внешнего ключа, появляющегося в ссылающемся отношении, в отношении, на которое ведет ссылка, должен найтись кортеж с таким же значением первичного ключа, либо значение внешнего ключа должно быть неопределенным. Для нашего примера из рис. 2.2 это означает, что если для сотрудника указан номер отдела, то этот отдел должен существовать.

Как ограничения целостности сущностей и ограничения по ссылкам поддерживаются СУБД?

Для соблюдения целостности по сущностям достаточно гарантировать отсутствие в любом отношении кортежей с одним и тем же значением первичного ключа.

Для соблюдения целостности по ссылкам

- при обновлении ссылающегося отношения (вставке новых кортежей или модификации значения внешнего ключа в существующих кортежах) достаточно следить за тем, чтобы не появлялись некорректные значения внешнего ключа;
- при удалении кортежа из отношения для обеспечения целостности но ссылкам на практике существуют три подхода:

- запрещается производить удаление кортежа, на который имеются ссылки; сначала нужно либо удалить ссылающиеся кортежи, либо соответствующим образом изменить значения их внешнего ключа;
- при удалении кортежа, на который имеются ссылки, во всех ссылающихся кортежах значение внешнего ключа автоматически становится неопределенным;
- третий подход (каскадное удаление) состоит в том, что при удалении кортежа из основного отношения с некоторым значением внешнего ключа автоматически удаляются кортежи из связанного с ним отношения с таким же значением первичного ключа.

Э. Коду принадлежат 12 правил, которым должна удовлетворять реляционная база данных:

1. *Правило информации.* Вся информация на логическом уровне представляется только значениями в таблицах без использования указателей и индексов.
2. *Правило гарантированного доступа.* Каждый элемент таблицы должен быть доступен через комбинацию из имени таблицы, имени поля и ключа.
3. *Системная поддержка неопределенных значений.* Для представления отсутствующих данных с любым типом используются Null-значения.
4. *Динамический каталог, создаваемый на основе реляционной модели.* Метаданные (словари) формируются тем же языком, что и данные. Пользователи, обладающие соответствующими правами, могут работать с метаданными с помощью того же реляционного языка, который они применяют для работы с основными данными.
5. *Правило исчерпывающего подъязыка данных.* В базе данных возможно использование нескольких языков взаимодействия с базой данных (например, язык вопросов-ответов), однако один, чаще всего SQL, должен быть главным и поддерживать все следующие элементы:
   - определение данных,
   - определение представлений,

- обработку данных (интерактивную и программную),
- задание условий целостности данных,
- идентификацию прав доступа,
- задание границ транзакций (начало, завершение и отмена).

6. *Правило обновления представления (view).* Все обновляемые представления должна обновлять и система.
7. *Ввод, обновление и удаление данных.* Возможность работать с отношением как с одним операндом должна существовать не только при чтении данных, но и при добавлении, обновлении и удалении данных.
8. *Физическая независимость данных.* Возможное изменение расположения базы данных, способов хранения данных не должно оказывать влияния на прикладные программы и пользователя.
9. *Логическая независимость данных.* При добавлении или удалении элементов (таблиц, полей) в структуре базы данных другие части базы данных остаются неизменными.
10. *Независимость условий целостности.* Должна существовать возможность определять условия целостности в языке реляционной базы данных и хранить их в каталоге, а не в прикладной программе.
11. *Независимость распределения.* В распределенных базах данных расположение данных независимо. Для пользователя такая база данных должна выступать как локальная база данных.
12. *Правило соблюдения правил.* Нельзя обходить ограничения, введенные с помощью языка SQL.

Контрольные вопросы и упражнения

1. Что такое тип данных? Домен*?* Атрибут? Кортеж*?* Схема отношения? Отношение?
2. Каков порядок строк отношения?
3. Какими способами задается домен?
4. Чем отличается домен от типа данных?

5. Что такое схема базы данных?
6. Для чего используются первичные ключи? Как они задаются?
7. В чем разница между первичным и возможным ключом?
8. Какими свойствами должен обладать первичный ключ?
9. Дайте классификацию ключей с точки зрения информативности входящих в них атрибутов.
10. Что такое сурогатный ключ?
11. В чем различие вторичных и первичных ключей?
12. Какова арность отношения P (таблица деталей), используемого при выполнении лабораторных работ?
13. Какова мощность тела отношения SPJ (таблица поставок), используемого при выполнении лабораторных работ?
14. Что такое базовое отношение? Что такое производное отношение?
15. Что такое внешний ключ? Какими свойствами он должен обладать?
16. Какой тип связи получится, если в качестве внешнего ключа будет выступать первичный ключ?
17. Какой тип связи установится при использовании неуникального внешнего ключа?
18. Перечислите фундаментальные свойства отношений.
19. Почему важна атомарность атрибутов?
20. Что понимается под структурной, манипуляционной, целостной частями реляционной модели данных?
21. Какой смысл имеет Null-значение?
22. Каково значение выражения «IsNull(0)»?
23. Каков результат выполнения операции умножения над двумя полями, содержащими значения Null?
24. Как обеспечивается целостность по сущностям?
25. Как обеспечивается целостность по ссылкам?
26. Для таблиц, используемых при выполнении лабораторных работ, определите атрибуты, кортежи, домены и ключи отношений.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Базы данных. Теория и практика: [учебник для вузов по направлению "Информатика и вычислительная техника" и "Информационные системы"] / Б.Я. Советов, В. В.Цехановский, В.Д. Чертовской. – М, 2007
2. Хансен Г., Хансен Д. Базы данных: разработка и управление. – М.: Бином,1999.
3. Базы данных / А.Д. Хоменко, В.М. Цыганков, М.Г. Мальцев. – Санкт- Петербург, Корона, 2002.
4. Рудикова Л.В. Базы данных: разработка приложений. – Санкт-Петербург, БХВ-Петербург, 2006.
5. Кузнецов С.Д. Базы данных. – М.: Академия, 2012.
6. Базы данных. Проектирование, реализация и сопровождение / Т. Конноли,К. Бегг, А. Страчан. – М. – С./П. – К., 1999.
7. Дейт К. Введение в системы баз данных. – М.: Вильямс, 2005.
8. Методы и модели анализа данных: OLAP и Data Mining / А.А. Барсегян, М.С. Куприянов, В.В. Степаненко, И.И. Холод. – Санкт-Петербург, БХВ- Петербург, 2004.
9. Бизнес-аналитика: от данных к знаниям: учебное пособие/ Н.Б. Паклин, В.И. Орешков. – Санкт-Петербург, Питер, 2010.
10. Базы данных. Интеллектуальная обработка информации/ В.В. Корнеев, А. Ф. Гарев, С.В. Васютин, В.В. Райх. – М. – Нолидж, 2000.
11. Клеппман М. Высоконагруженные приложения. Программирование, масштабирование, поддержка. – СПб.: Питер, 2018. – 640 с.: ил. — (Серия «Бестселлеры O’Reilly»).
12. Фаулер, Мартин, Садаладж, Прамодкумар Дж. NoSQL: новая методология разработки нереляционных баз данных. :пер. с англ. – М.: ООО "И.Д. Вильяме", 2013. - 192 с.: ил. – парал. тит. англ.
13. Робинсон Ян, Вебер Джим, Эифрем Эмиль Графовые базы данных: новые возможности для работы со связанными данными / пер. с англ. Р. Н.

Рагимова; науч. ред. А. Н. Кисилев. – 2-е изд. – М.: ДМК Пресс, 2016. – 256 с.: ил.

14. Кайл Бэнкер MongoDB в действии. / Пер. с англ. Слинкина А. А. – М.: ДМК Пресс, 2012. – 394с.: ил.
15. Эрик Редмонд, Джим. Р. Уилсон Семь баз данных за семь недель. Введение в современные базы данных и идеологию NoSQL. Под редакцией Жаклин Картер / пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2013. – 384с.: ил.
16. Карпентер Д., Хьюитт Э. Cassandra. Полное руководство. 2-е изд./ пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2017. – 400с.: ил.

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ ПО ДИСЦИПЛИНЕ «Технологии проектирования баз данных (SQL/NoSQL)»

для обучающихся по направлению 09.03.02 «Информационных системы и технологии», профиль «Разработка web-ориентированных информационных систем» всех форм обучения

Составители:

Саргсян Эрик Ромович

Рындин Никита Александрович

Подписано в печать 04.06.2021

Формат 60x84 1/8 Бумага для множительных аппаратов Уч.-изд. л. 3,3 Усл. печ. л. 3,0.

ФГБОУ ВО «Воронежский государственный технический университет» 396026 Воронеж, Московский просп., 14

Участок оперативной полиграфии издательства ВГТУ 396026 Воронеж, Московский просп., 14