A.II.3.3 Спецификация дизайна данных

В процессе спецификации дизайна генерируются конкретные языковые интерфейсы из семантической модели данных. Эти интерфейсы согласуются с определенными моделями данных, хотя их следует отличать от семантических моделей данных. Различают иерархические, сетевые, реляционные и объектно-ориентированные модели данных. Поскольку иерархические модели данных представляют только исторический интерес, а важность сетевых моделей уменьшается, мы сфокусируемся на реляционных моделях данных и частично рассмотрим объектно-ориентированные модели данных.

На первом шаге информационные объекты из описания требований трансформируются в отношения с соблюдением определенных правил.

На втором шаге отношения оптимизируются с помощью нормализации; устраняются все аномалии операций “вставить”, “изменить” или “удалить”. При этом детализируются исходные отношения, которые перешли из описания требований. Если чрезмерная детализация препятствует эффективности, возможна также денормализация.

На третьем шаге определяются условия целостности, которые могут или передаваться из описания требований и приводиться в соответствие с ограничениями реляционной модели, или генерироваться на этапе спецификации дизайна. Условия целостности, передаваемые из описания требований, нуждаются в дополнительной обработке по причине языковых ограничений реляционной модели. Условия целостности необходимо формулировать на специальном языке манипулирования данными.

На четвертом шаге реляционная схема трансформируется в язык описания данных соответствующей системы управления базами данных. При этом добавляются логические пути доступа для поддержки обработки записей. За этим шагом следует этап внедрения.

Реляционная схема формулируется средствами языка описания данных (Data Description Language — DDL). Использование языка описания данных максимально приближает этап внедрения, когда остальные аспекты ARIS взаимодействуют с аспектом данных. Приложения должны взаимодействовать со схемой базы данных, выраженной средствами DDL.

A.II.3.3.1 Создание отношений

Отношения (R_i) определяются списком наименований атрибутов A_{ij}. Отношения удобно представить в виде таблицы (рис. 68). Математически отношение представляет собой декартово произведение доменов атрибутов.

Придерживаясь относительно простых требований, отношения создаются из описания требований к данным в рамках ERM. Каждый тип сущности и каждый тип связи n:n приводит к созданию нового отношения. Тип связи n:n означает, что максимальное значение по меньшей мере двух кардинальностей соседних типов сущностей равно n.

(1) $R_i (A_{i1}, A_{i2}, ..., A_{iz})$ A_{ij} — Атрибут j в отношении i

Деталь (Номер детали, Наименование, Запас)

Деталь	Номер детали	Наименование	Запас
	4717	Гайка	526
	4728	Болт	768

(2) $R_i \subseteq D_{i1} \times D_{i2} \times ... \times D_{iz}$

где D_{ij} — домен A_{ij}

Рис. 68. Представление отношений

С другой стороны, типы связи 1:n не создают нового отношения. Отношения корректируются с помощью добавления ключевого атрибута к типу сущности, из которого исходит кардинальность, максимальное значение которой равно 1 (рис. 69). Добавленный ключевой атрибут называется внешним ключом.

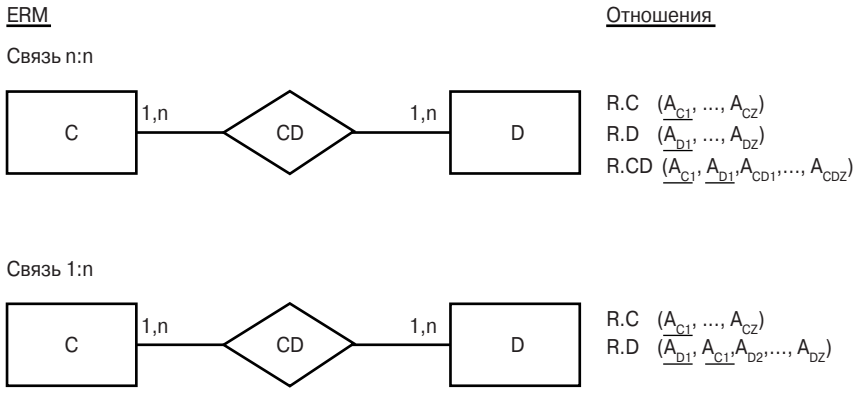

Рис. 69. Создание отношений

В метамодели создания отношений на рис. 70 вводится класс ОТНОШЕНИЕ. Класс ассоциаций СОЗДАНИЕ ОТНОШЕНИЯ обозначает связь с классом ИНФОРМАЦИОННЫЙ ОБЪЕКТ. В соответствии с условиями формирования, информационный объект может иметь 0 или максимум 1 отношение, тогда как одно отношение может характеризовать один или несколько информационных объектов. Атрибут класса ОТНОШЕНИЕ — это собственное имя отношения, которое может совпадать с именем исходного информационного объекта ERM.

Имена атрибутов, определяющих отношение, можно взять из описания требований, хотя их можно изменить. Если изменений не происходит, атрибуты создаются с помощью связи СОЗДАНИЕ ОТНОШЕНИЯ между классами ОТНОШЕНИЕ и ИНФОРМАЦИОННЫЙ ОБЪЕКТ. Однако чтобы подчеркнуть самостоятельную роль спецификации дизайна в создании атрибутов,

атрибуты отношения связаны с общими атрибутами из описания требований с помощью класса ассоциаций СВЯЗЬ ОТНОШЕНИЯ С АТРИБУТАМИ.

Если имена не изменяются при переходе от описания требований, отношения создаются в соответствии с требованиями. Многие прикладные системы поддерживают такую автоматическую миграцию из ERM.

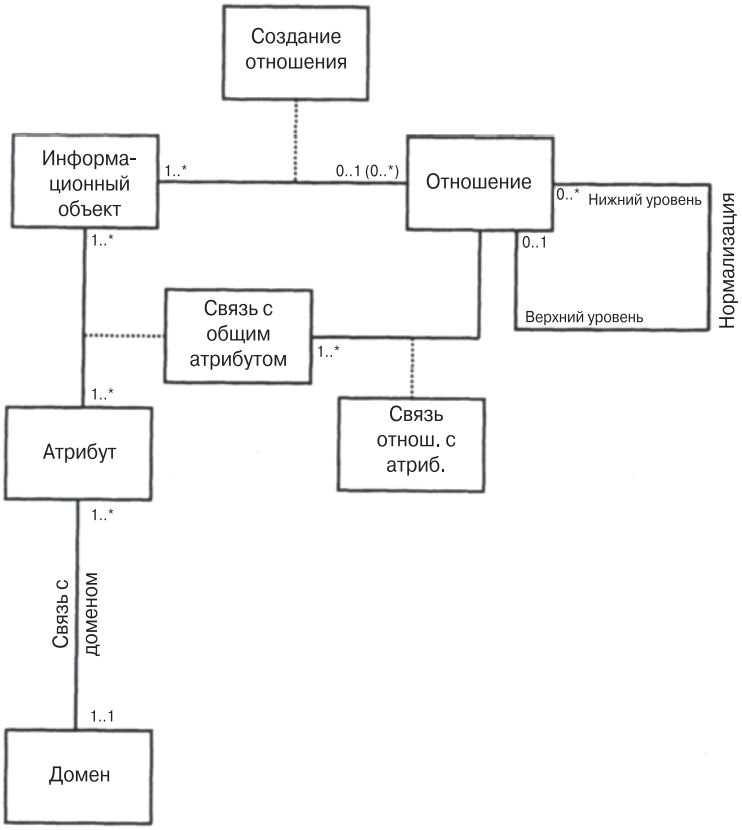

Рис. 70. Метамодель создания отношений

Домены имеют доступ к существующим определениям доменов из описания требований с помощью назначения атрибутов (класс ДОМЕН будет обсуждаться при рассмотрении реляционной модели и условий целостности, относящихся к доменам).

Преобразование типов сущностей и типов связей в отношения реляционной модели не вызывает затруднений. Преобразование сложных объектов связано с необходимостью импортирования процедур или неструктурированных групп данных в среду реляционной модели или даже преобразования модели данных в объектно-ориентированную модель данных.

A.II.3.3.2 Нормализация и денормализация

Трансформация объектов описания требований в исходные отношения реляционной модели данных иногда вызывают нежелательные побочные эффекты — избыточные функциональные зависимости между атрибутами отношений. Эти аномалии устраняются с помощью так называемой нормализации.

Нормализация — процесс преобразования реляционных отношений к виду, отвечающему так называемым нормальным формам. Хотя процесс нормализации разработан именно для реляционных баз данных, его рассматривают как общую процедуру улучшения структур данных и применяют также для других моделей данных. В этой книге каждый отдельный шаг нормализации освещается только на уровне определения. Более детально эта тема рассматривается в других публикациях (*см. Schlageter/Stucky, Datenbanksysteme 1983, p. 183; Wedekind, Datenbanksysteme I 1991, p. 200; Vossen, Datenbank-Management-Systeme 1995, pp. 249–270*). Более того, мы рассмотрим только нормальные формы от первой по третью (так называемые нормальные формы Бойса-Кодда), так как нормальные формы от четвертой и выше используются редко.

Начнем со следующего примера (*см. Schlageter/Stucky, Datenbanksysteme 1983, p. 162*), который относится к проектной организации.

ПЕРВАЯ НОРМАЛЬНАЯ ФОРМА (1 NF)

(1,1) R_EMPLOYEE (**EMP_NO**, NAME, ADDRESS, PROFESSION, **DEPT_NO**)
(1,2) R_PROJECT (**P_NO**, P_NAME, P_DESCR, P_MGR)
(1,3) R_EMP_PROJ (P_NO, EMP_NO, PH_NO, PERCENT_WORK_HOURS)
(1,4) R_DEPT (DEPT_NO, DEPT_MGR, BUILD_NO, JANITOR)

ВТОРАЯ НОРМАЛЬНАЯ ФОРМА (2 NF)

(2,1) R_EMPLOYEE* (EMP_NO, NAME, PH_NO, ADDRESS, PROFESSION, DEPT_NO)
(2,3) R_EMP_PROJ* (P_NO, EMP_NO, PERCENT_WORK_HOURS)

ТРЕТЬЯ НОРМАЛЬНАЯ ФОРМА (3 NF)

(3,4) R_DEPT* (DEPT_NO, DEPT_MGR, BUILD_NO)
(3,5) R_BUILD (BUILD_NO, JANITOR)

Определения

- Отношение R находится в первой нормальной форме (1 NF), если каждый атрибут может содержать только одно значение.
- Отношение находится во второй нормальной форме (2 NF), если оно соответствует первой нормальной форме и каждый неключевой атрибут функционально зависит от ключевого атрибута.
- Отношение находится в третьей нормальной форме (3 NF), если оно соответствует второй нормальной форме и ни один из неключевых атрибутов не зависит транзитивно от ключевого атрибута.

Рассмотрим процесс нормализации на предложенном выше примере.

Отношения, находящиеся в первой нормальной форме, содержат аномалии, которые устраняются при переходе ко второй нормальной форме.

- Имеет место **аномалия вставки**. Например, когда в базу добавляются данные о новом сотруднике, который еще не занят ни в одном проекте. В этом случае номер телефона (PH_NO) не может относиться к новому сотруднику, так как PH_NO находится только среди атрибутов отношения "сотрудник–проект" (R_EMP_PROJ).
- Когда проект закончен и отношение (1,3) следует удалить, появляется **аномалия удаления**. Будет удален и номер телефона сотрудника, хотя номер телефона остается адекватным.
- Имеет место также **аномалия изменения.** Например, если номер телефона сотрудника, занятого в нескольких проектах, изменяется, то многие кортежи отношения R_EMP_PROJ нуждаются в изменении. Происходит массовая корректировка записей базы данных, хотя по сути изменяется *единичный* факт.

Эти аномалии устраняются, когда отношения (1,1) и (1,3) преобразуются во вторую нормальную форму. Учитывая, что номер телефона PH_NO зависит только от ключевого атрибута отношения (1,1) EMP_NO, он добавляется к атрибутам этого отношения. Одновременно номер телефона исключается из отношения (1,3). Отношения (1,2) и (1,4) уже находятся во второй нормальной форме.

Когда нанимается новый охранник (janitor), все кортежи отношения (1,4), относящиеся к департаментам здания, в котором работает охранник, подлежат корректировке. Атрибут JANITOR не зависит непосредственно от DEPT_NO (код департамента), а зависит от BUILD_NO (код здания). Поэтому в третьей нормальной форме вместо отношения (1,4) присутствуют два отношения: отношение (3,4) получено из отношения (1,4) с помощью исключения атрибута JANITOR, а отношение (3,5) добавлено. Отношения (1,2), (2,1) и (2,3) уже находятся в третьей нормальной форме.

Технически процесс нормализации заключается в том, что исходные отношения дробятся. Степень измельчения отношений зависит от их исходных качеств. Если исходное состояние — это так называемое универсальное отношение, где объекты из описания требований не отсортированы, процесс нормализации означает полное изменение структуры. С другой стороны, если описание требований выполнено с использованием, например, ERM, информационные объекты находятся на достаточно высоком уровне нормализации.

Имеет смысл проверить систему с помощью нормализации, даже если информационные объекты спроектированы тщательно, ключевые атрибуты определены и неключевые атрибуты будут добавлены позже.

При дроблении класса ОТНОШЕНИЕ в процессе нормализации исходные отношения превращаются в новые отношения. Учитывая, что один информационный объект может вести ко многим отношениям, кардинальность для класса ОТНОШЕНИЕ превращается в (0..*), как показано в скобках на рис. 70.

Связь между отношением реляционной модели и отношением нижнего или верхнего уровня нормализации отображает класс НОРМАЛИЗАЦИЯ.

Описание требований и исходные отношения корректируются при создании новых отношений в процессе нормализации. Однако это ведет не к расширению диаграммы классов, изображенной на рис. 70, а только к созданию новых экземпляров класса СВЯЗЬ ОТНОШЕНИЯ С АТРИБУТАМИ.

A.II.3.3.3 Условия целостности

Выполнение условий целостности данных необходимо для того, чтобы база данных корректно моделировала объективную реальность (*см. Blaser/Jarke/Lehmann, Datenbanksprachen und Datenbankbenutzung 1987, p. 586*).

Поскольку таблицы реляционных моделей не слишком хорошо отображают семантические факты, определяются условия целостности с помощью языка манипулирования данными. Условия целостности могут также определяться внутри прикладной программы. Учитывая принцип локальности и преимущества централизованного контроля целостности данных, имеет смысл включить условия целостности в модель данных. Современные базы данных позволяют хранить много функций (*см. Dittrich/Gatziu, Aktive Datenbanksysteme 1996*).

Условия целостности ссылаются на семантическое содержание и внедрение модели данных. Условия целостности тесно связаны с базой данных и не касаются вопросов пользовательского интерфейса. Вот почему в данной работе мы фокусируемся на условиях семантической целостности.

Условия состоятельности (условия ссылочной целостности) касаются атрибутов, экземпляров кортежей и отношений, производных от типов связей (*см. Blaser/Jarke/Lehmann, Datenbanksprachen und Datenbankbenutzung 1987, p. 588*).

Язык структурированных запросов SQL (Structured Query Language) — универсальный компьютерный язык, применяемый для создания, модификации и управления данными в реляционных базах данных. В SQL условия целостности определяются с помощью команд ASSERT и проверяются при выполнении соответствующих действий.

Например, чтобы при удалении кода сотрудника (EMP_NO) в отношении R_EMPLOYEE связь с навыками сотрудника (в отношении OWNS, которое ссылается на отношение SKILLS) также удалялась, следует использовать следующее определение триггера (*см. Blaser/Jarke/Lehmann, Datenbanksprachen und Datenbankbenutzung 1987, p. 592*).

```
DDEFINE TRIGGER T1
ON DELETE OF EMPLOYEE (EMP_NO):
DELETE OWNS
WHERE OWNS.EMP_NO = EMPLOYEE.EMP_NO.
```

На рис. 71 иллюстрируется несколько примеров определения условий целостности с помощью команд ASSERT языка SQL.

Объяснение	Команды SQL
1) Условие касается атрибута. Значение EMP_NO должно иметь 4 знака.	ASSERT IB1 ON EMPLOYEE: EMP_NO BETWEEN 0001 AND 9999
2) Условие касается многих атрибутов записи. SALARY_SUM (зарплата департамента) должна быть меньше ANNUAL_BUDGET.	ASSERT IB2 ON DEPARTMENT: SALARY_SUM < ANNUAL_BUDGET
3) Условие касается многих экземпляров одного типа записи (отношения). Зарплата одного сотрудника не может превышать больше чем на 20% среднюю зарплату по департаменту.	ASSERT IB3 ON EMPLOYEE X: SALARY * 1,2 ≤ (SELECT AVG(SALARY) from EMPLOYEE WHERE DEPART = X.DEPART)
4) Условие касается многих экземпляров во многих отношениях. Значение SALARY_SUM для департамента должно быть равно сумме значений SALARY всех сотрудников департамента.	ASSERT IB4 ON DEPARTMENT X: SALARY_SUM = (SELECT SUM(SALARY) from EMPLOYEE WHERE DEPART = X.DEPARTNO)

Рис. 71. Условия целостности
(Reuter, Sicherheits- und Integritatsbedingungen 1987, pp. 381, 385)

На рис. 72 иллюстрируются ключевые взаимосвязи, порождаемые условиями целостности. Левый фланг рис. 72, содержащий классы ОТНОШЕНИЕ, АТРИБУТ и ДОМЕН, представляет собой отправную позицию для обсуждения вопросов целостности. Класс ТИП ЦЕЛОСТНОСТИ описывает различные типы условий целостности (*см Reuter, Sicherheits- und Integritatsbedingungen 1987, p. 380*). Условия целостности различают по их рангу, т.е. по типу и количеству объектов, которых касается условие целостности (например, см. рис. 71). Кроме того, условия целостности различают по времени проверки (или условие проверяется всегда, или после выполнения определенного количества операций), по способу проверки (условия состояния или условия перехода) или по вызываемым действиям.

На рис. 72 каждое конкретное условие целостности относится к одному типу целостности. Условие целостности может касаться одного или многих отношений и ассоциаций атрибутов одного или многих отношений. Пределы значений атрибутов проверяются с помощью доменов.

A.II.3.3.4 Логические пути доступа

Выполнение вложенных запросов SQL может привести к потере эффективности базы данных. Чтобы избежать потери эффективности, следует разработать утилиты (обслуживающие программы) для обеспечения доступа к отдельным записям или группам записей базы данных. В частности, следует избегать необходимости последовательного просмотра таблиц.

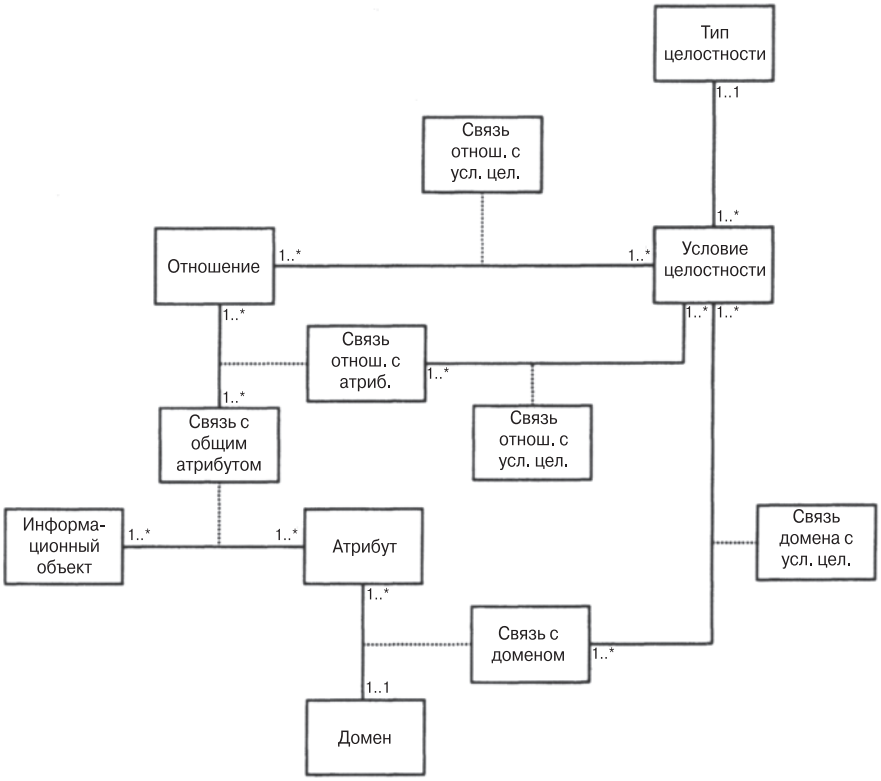

Рис. 72. Метамодель условий целостности

Как правило, утилиты обеспечивают доступ к записям по их ключам и доступ к группам записей в определенной последовательности (сортировка). Логические пути доступа для первичных ключей классифицируются на основании последовательных организационных форм, древовидных индексов или распределенных организационных форм. Пути доступа для вторичных ключей создаются с помощью инвертированных списков (таблиц индексов).

В спецификации дизайна обуславливается, какой тип поддержки требуется для определенных атрибутов. Класс ТИП ЛОГИЧЕСКОГО ПУТИ ДОСТУПА характеризует различные типы поддержки. Связь между атрибутом отношения и типом пути доступа характеризует логический путь доступа. Атрибут определенного отношения участвует в определении многих путей доступа.

Пути доступа в спецификации дизайна певоначально определяются в соответствии с общими предположениями относительно количества кортежей в таблице и ожидаемого количества обращений к таблице. Когда база данных создана и показатели эффективности известны, утилиты доступа могут детализироваться дальше.

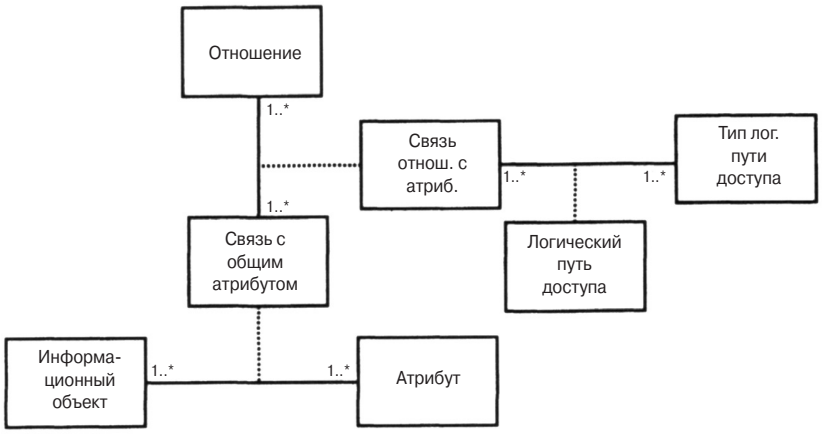

Рис. 73. Логические пути доступа

A.II.3.3.5 Схема базы данных

На заключительном этапе спецификации дизайна структуры данных переводятся на язык определения данных конкретной системы баз данных, например, ORACLE, INFORMIX, CA-INGRES. Учитывая, что реляционные модели определены математически, а условия целостности на языке SQL стандартизованы, перевод схемы реляционной базы данных на язык конкретной системы баз данных не вызывает затруднений.

Если предприятие использует несколько различных систем управления базами данных (СУБД), нейтральная схема реляционной базы данных (рис. 74) может быть преобразована в несколько конкретных схем. Поставщики программного обеспечения, предлагающие продукты для различных баз данных, часто встречаются с такой ситуацией. Конкретные схемы баз данных включают конкретные определения логических путей доступа и условия целостности.

A.II.3.3 Внедрение модели данных

Начнем со спецификации дизайна модели данных, т.е. традиционной схемы базы данных с ее отношениями, атрибутами и условиями целостности. В спецификации дизайна определяются также логические пути доступа к определенным ассоциациям атрибутов с учетом частоты обращений и запросов.

В процессе внедрения концептуальные схемы моделируются во внутренние схемы, которые описывают те же фрагменты реальности, что и концептуальная схема. Семантический аспект не рассматривается на этом этапе. Моделирование внутренней схемы из концептуальной схемы возможно без знания семантического окружения.

Задача администраторов базы данных — структурировать внутреннюю схему, чтобы создать эффективные структуры базы данных с помощью ин-

формационных технологий. Администраторы должны знать параметры использования и отслеживать время отклика на запросы к базе данных от различных приложений. При этом администраторы могут ничего не знать о содержании приложений.

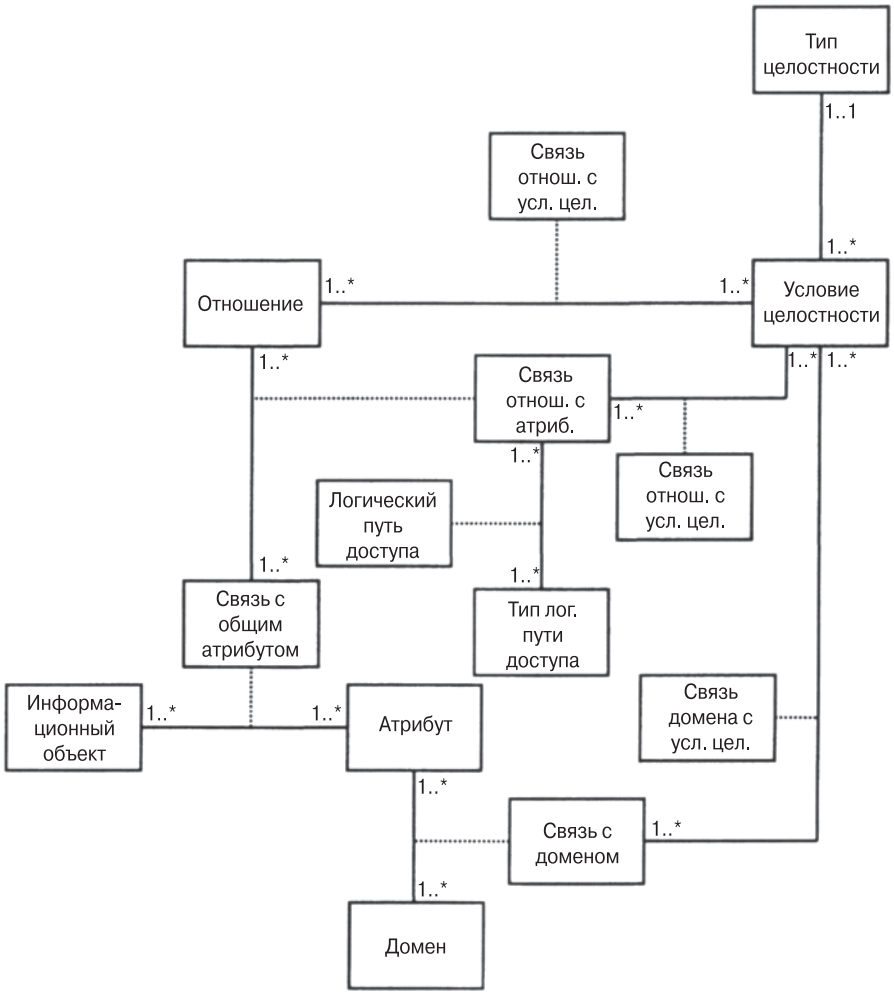

Рис. 74. Нейтральная схема реляционной базы данных

Автономия этапа внедрения поддерживается использованием специфической корпоративной терминологии, которая моделируется на логическом уровне спецификации дизайна. Например, понятия ОТНОШЕНИЕ и АТРИБУТ на этапе внедрения (рис. 75) подчиняются терминам ТИП ЗАПИСИ и ПОЛЕ. Термин ТИП ЗАПИСИ обозначает тип записи, который характеризуется определенной комбинацией атрибутов. Класс СТРАНИЦА включает различные типы записей. Задача администратора состоит в том, чтобы оптимизировать базу данных, размещая часто используемые типы записей физически близко друг к другу.

Внедрение отличается от спецификации дизайна тем, что на этапе внедрения последовательность полей можно изменить, поля переименовать, данные сжать или использовать специфические форматы полей. Более того, создаются виртуальные поля, т.е. задаются правила преобразования полей, если содержание полей создается из содержания других полей (например, итоговые поля).

Возможно различие между отношениями и записями, если отношения разбиваются на многие записи или многие отношения объединяются в одну физическую запись.

Условия состоятельности и целостности, определенные на уровне спецификации дизайна, на этапе внедрения определяются на физическом уровне в виде процедур. Кроме того, логическим путям доступа ставятся в соответствие конкретные методы физического доступа или определяются дополнительные пути физического доступа.

В добавление к именам ЗАПИСЬ и ПОЛЕ, имеющим аналоги в спецификации дизайна, вводятся классы СТРАНИЦА и ЗОНА (см. рис. 75), т.е. создаются дополнительные организационные элементы для оптимизации структур распределения памяти. Исходные организационные элементы лежат в основании физического размещения данных и доступа к данным, находящимся на внешних устройствах.

Потенциал оптимизации страниц и зон очевиден, если речь идет о достоверности данных. Так, эффективнее иметь доступ ко многим записям, размещенным на одной странице, чем обрабатывать записи, распределенные между несколькими страницами. Аналогично, упорядоченный доступ к страницам эффективнее, чем доступ к страницам в произвольном порядке.

Язык описания хранения данных (Data Storage Description Language — DSDL) связывает внутренние и внешние модели, внедряя структуры распределения памяти. Физические пути доступа моделируются с помощью конкретных индексных таблиц, цепочек хэш-функций.

Физические пути доступа определяются на уровне связи записи и поля. Они или отображают логические пути доступа из спецификации дизайна, или создаются на этапе внедрения, когда появляются детальные сведения о параметрах эффективности.

Определения на уровне физической структуры данных подчиняются цели независимости данных. Изменения физических устройств или программного обеспечения должны влиять только на внедрение, но не на концептуальную схему базы данных. Одна концептуальная схема базы данных может включать несколько внутренних схем баз данных. С другой стороны, возможна корректировка концептуальной схемы базы данных без изменения ее физической схемы.

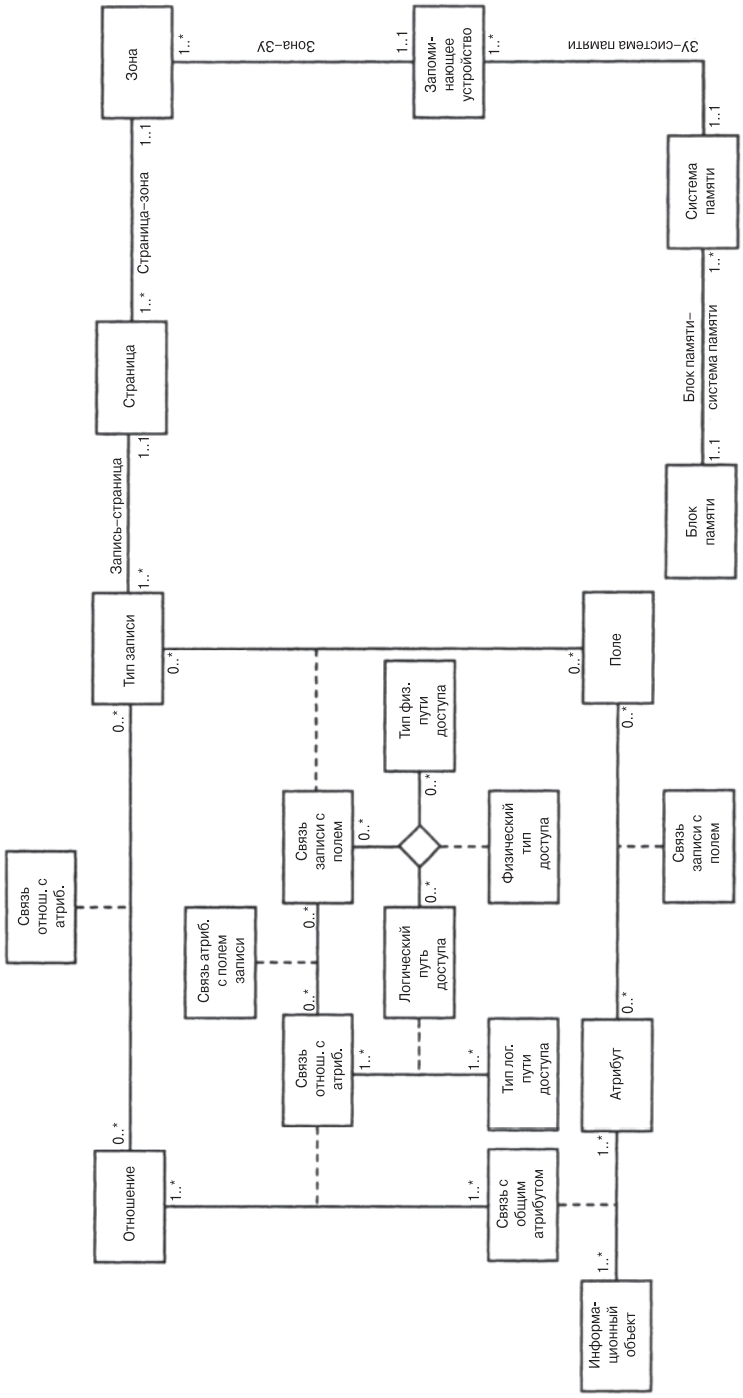

Рис. 75. Внедрение реляционной базы данных