#### 2.1.1.3.1. Ограничительные условия, поддерживающие целостность базы данных

Как следует из определения ссылочной целостности при наличии в ссылочных полях двух таблиц различного представления данных происходит нарушение ссылочной целостности, такое нарушение делает информацию в базе данных недостоверной. Чтобы предотвратить потерю ссылочной целостности, используется механизм каскадных изменений (который чаще врего реализуется специальными объектами СУБД - триггерами). Данный механизм состоит в следующей последовательсности действий:

* при изменении поля связи в записи родительской таблицы следует синхронно изменить значения полей связи в соответствующих записях дочерней таблицы;
* при удалении записи в родительской таблицы следует удалить соответствующие записи и в доцерней таблице.

#### 2.1.1.3.2 Поцесс нормализации

*Нормализация -* процесс приведения реляционных таблиц к стандартному виду. В базе данных могут присутствовать такие проблемы как:

* *Избыточность данных.*Повторение данных в базе данных.
* *Аномалия обновления.* Противоречивость данных, вызванная их избыточностью и частичным обновлением.
* *Аномалия удаления.* Непреднамеренная потеря данных, вызванная удалением других данных.
* *Аномалия ввода.* Невозможность ввести данные в таблицу, вызванная отсутствием других данных.

Для решения этих проблем применяют р*азбиение таблиц* - разделение таблицы на несколько таблиц. Для того чтобы это сделать пользуются нормальными формами или правилами структурирования таблиц.

*Первая нормальная форма*

Реляционная таблица находится в *первой нормальной форме (1НФ),* если значения в таблице являются атомарными для каждого атрибута таблицы, т.е. такими значениями, которые не являются множеством значений или повторяющейся группой. В определении Кодда реляционной модели уже заложено, что реляционные таблицы находились в 1НФ,

*Вторая нормальная форма.*

Реляционная таблица находится во *второй нормальной форме (2НФ),* если никакие неключевые атрибуты не являются функционально зависимыми лишь от части ключа. Таким образом, 2НФ может оказаться нарушена только в том случае, когда ключ составной.

*Функциональная зависимость.*Значение атрибута в кортеже однозначно определяет значение другого атрибута в кортеже.

Более формально можно определить функциональную зависимость следующим образом: если А и В - атрибуты в таблице В, то запись ФЗ (функциональную зависимость): А - " В обозначает, что если два кортежа в таблице В имеют одно и то же значение атрибута А, то они имеют одно и то же значение атрибута В. Это определение такжеприменимо,если А и В - множества столбцов, а не просто отдельные столбцы.

Атрибут в левой части ФЗ называется *детерминантом,* так как его значение определяет значение атрибута в правой части. Ключ таблицы является *детерминантом,* так как его значение однозначно определяет значение каждого атрибута таблицы.

Процесс разбиения на две 2НФ-таблицы состоит из следующих шагов:

1. Создается новая таблица, атрибутами которой будут атрибуты исходной таблицы, входящие в противоречащую правилу ФЗ. Детерминант ФЗ становится ключом новой таблицы.
2. Атрибут, стоящий в правой части ФЗ, исключается из исходной таблицы.
3. Если более одной ФЗ нарушают 2НФ, то шаги 1 и 2 повторяются для каждой такой ФЗ.
4. Если один и тот же детерминант входит в несколько ФЗ, то все функционально зависящие от него атрибуты помещаются в качестве неключевых атрибутов в таблицу, ключом которой будет детерминант.

*Третья нормальная форма*

Реляционная таблица имеет *третью нормальную форму* (ЗНФ), если для любой ФЗ: Х - У Х является ключом. Заметим, что любая таблица, удовлетворяющая ЗНФ, также удовлетворяет и 2НФ. Однако обратное неверно.

*Критерийнормальной формы Бойса-Кодда (НФБК)* утверждает, что таблица удовлетворяет ЗНФ, если в ней нет транзитивных зависимостей. Транзитивная зависимость возникает, если неключевой атрибут функционально зависит от одного или более неключевых атрибутов. То есть этот критерий учитывает следующие два случая:

1. Неключевой атрибут зависит от ключевого атрибута, входящего в составной ключ (критерий нарушения 2НФ).
2. Ключевой атрибут, входящий в составной ключ, зависит от неключевого атрибута.

Таким образом, если таблица удовлетворяет НФБК, то она также удовлетворяет ЗНФ в смысле транзитивных зависимостей и 2НФ.

*Четвертая нормальная форма*

Таблица имеет *четвертую нормальную форму (4НФ),* если она имеет ЗНФ и не содержит многозначных зависимостей. Поскольку проблема многозначных зависимостей возникает в связи с многозначными атрибутами, то мы можем решить проблему, поместив каждый многозначный атрибут в свою собственную таблицу вместе с ключом, от которого атрибут зависит.

*Пятая нормальная форма.*

Пятая нормальная форма (5НФ) была предложена для того, чтобы исключить аномалии, связанные с особым типом ограничительных условий, называемых со*вместными зависимостями.* Эти зависимости имеют в основном теоретический интерес и сомнительную практическую ценность. Следовательно, пятая нормальная форма в действительности не имеет практического применения.

*Нормальная форма область/ключ.*

Таблица имеет *нормальную форму область/ключ (НФОК),* если любое ограничительное условие в таблице является следствием определений областей и ключей. Однако не был дан общий метод приведения таблицы к НФОК.

В качестве примера, рассмотрим структуру реляционной базы данных, описывающей "отношения" пациентов и докторов в произвольной клинике (область приложения примера выбрана из-за того, что в сертификационных тестах Oracle аналогичные примеры встречаются очень часто). Пусть существует некая клиника, основные характеристики которой описываются в таблице CLINICS, в данной клинике работают доктора, основные характеристики которых описывает таблица DOCTORS. Данные пациентов клиники хранятся в таблице PATIENTS. Взаимосвязи между таблицами представлены на рис.10. (Для упрощения предполагается, что у доктора может быть несколько пациентов, которые не являются пациентами других докторов, для реализации реальной картины, когда один пациент может относиться к нескольким разным докторам, между таблицами DOCTORS и PATIENTS необходимо включить дополнительную связывающую таблицу).

|  |  |  |
| --- | --- | --- |
| № | Наименование столбца | Описание |
| Таблица CLINICS | | |
| 1 | CS\_NNN | Индекс |
| 2 | CS\_REG\_NUMBER | Регистрационный номер |
| 3 | CS\_CITY\_NNN | Ссылка на справочник городов и регионов |
| 4 | CS\_NAME | Наименование клиники |
| 5 | CS\_ADDRESS | Адрес клиники |
| 6 | CS\_PHONE\_NUMBER | Номер телефона |
| 7 | CS\_TYPE | Профиль клиники |
| Таблица DOCTORS | | |
| 1 | DC\_NNN | Индекс |
| 2 | DC\_NAME | Ф.И.О. доктора |
| 3 | DC\_CS\_NNN | Ссылка на таблицу CLINICS |
| 4 | DC\_DIPLOM\_NUMBER | Номер диплома |
| 5 | DC\_SPECIALTY\_NNN | Ссылка на справочник специальностей |
| 6 | DC\_SHTAT\_NNN | Ссылка на штатное расписание |
| 7 | DC\_CALENDAR\_NNN | Ссылка на расписание приема |
| Таблица PATIENTS | | |
| 1 | PT\_NNN | Индекс |
| 2 | PT\_REG\_NUMBER | Регистрационный номер |
| 3 | PT\_NAME | Ф.И.О. пациента |
| 4 | PT\_ADDRESS | Адрес пациента |
| 5 | PT\_POLIS\_NUMBER | Номер полиса |
| 6 | PT\_PHONE\_NUMBER | Номер телефона |
| 7 | PT\_BIRTHDATE | Дата рождения |
| 8 | PT\_FIRST\_VISIT | Дата первого визита |
| 9 | PT\_LAST\_VISIT | Дата последнего визита |
| 10 | PL\_PT\_NNN | Ссылка на таблицу платежей |
| Таблица PAYMENTS | | |
|  | PL\_NNN | Индекс |
|  | PL\_ACCOUNT\_NUMBER | Номер расчетного счета |
|  | PL\_PAY\_NNN | Ссылка на таблицу расчетов |

Представленная структура, конечно, не обладает функциональной полнотой с точки зрения проектирования АИС клиники, с ее помощью мы лишь рассмотрим различные типы отношений в реляционных СУБД.

Перед тем, как перейти к рассмотрению вопросов стандартизации и целостности данных в РСУБД несколько рекомендаций по выбору наименований таблиц и полей. Внимательно взглянув на описание таблиц можно заметить, что генерация наименований таблиц и столбцов подчиняется некоторой синтаксической конструкции, которая в общем виде может быть представлена следующим образом:

Для таблиц:

<Псевдоним АИС>\_<Псевдоним модуля АИС>\_:\_<Псевдоним подмодуля>\_<Имя таблицы>

Например, если бы мы разрабатывали АИС клиники c сокращенным названием CSL, то все таблицы входящие в данную систему было бы целесообразно называть

CSL\_<имя модуля>\_<имя таблицы>.

Для столбцов:

<Псевдоним таблицы>\_<имя столбца>.

Например, как показано на рис.10. Регистрационный номер пациента храниться в поле PT\_REG\_NUMBER, таблицы PATIENTS, имеющий псевдоним PT.

Конечно, использование этих не хитрых правил не является обязательным, но позволяет значительно облегчить читаемость разработанной информационной структуру. Предположите, как было бы все, если бы поля таблиц назывались P111, P112 и т.п., а ведь такие вещи встречаются практически очень часто, например в FoxPro 2.6.

Перейдем к рассмотрению вопросов стандартизации и обеспечения ссылочной целостности реляционных таблиц.

**Преобразование отношений**

Поля таблиц могут находиться между собой в одном из следующих отношений:

один-к-одному, один-ко-многим, многие-ко-многим и рекурсивных, определения которых приведены в табл.1. Рассмотрим преобразование отношений на примере АИС "ДОКТОР-ПАЦИЕНТ" (рис.10).

Отношение один-к-одному представляет собой такое отношение, при котором каждой записи в таблице А соответствует единственная запись в таблице В (рис.11). Применение такого типа отношений встречается крайне редко и предназначено в основном для функционального разделения информации на несколько таблиц, т.е. когда не хотят, чтобы таблица БД "распухала" от второстепенной информации. На рис.10 представлено, как используя отношение один-к-одному таблица PATIENTS преобразована в две таблицы: PATIENTS\_REG и PATIENTS\_KART (на рисунке показаны только основные атрибуты таблиц). Также необходимо принимать во внимание, что БД использующие такие отношения не могут быть полностью нормализованы.

Отношение один-ко-многим можно без преувеличения назвать основным типом отношений использующемся при проектировании современных БД, так как позволяет представлять иерархические структуры данных. Под данным отношение понимается такое отношение, когда одной записи в родительской таблице соответствуют записи в дочерней таблице (причем число соответствующих записей выражается рядом натуральных чисел 0,1,2,:N и т.п.) (рис.12). Отношения один-ко-многим могут быть жесткими и нежесткими. Для жестких отношений должно выполнять требование, что каждой записи в родительской таблице должна соответствовать хотя бы одна запись в дочерней таблице.

Отношение многие-ко-многим представляет собой отношение при котором записям родительской таблицы соответствуют записи дочерней таблицы, а ряду записей дочерней таблицы соответствуют записи в родительской таблицы (рис.13). Использование такого типа отношений крайне ограничено, не только из-за того, что некоторые БД его вообще не поддерживают на уровне индексов и ссылочной целостности, но и потому, что практически любое отношение многие-ко-многим может быть заменено одним или более отношением один-ко-многим (посмотрите на пример на рис.13. и так не когда не делайте).

Другим важным типом отношений - является рекурсивное отношение, т.е. такое отношение которое описывает связи между записями внутри одной таблицы БД, т.е. оно связывает объектное множество с ним самим. Пример рекурсивных отношений показан на рис.14., который иллюстрирует, что доктора Петров А.А. и Васин Н.Н. находятся в зависимости от доктора Сидорова В.Н.. В зависимости от функционального назначения этого отношения оно может иллюстрировать, например, что они являются пациентами доктора Сидорова В.Н., или Сидорова В.Н. является по отношению к ним начальником и т.п. Данный тип отношений позволят реализовать древовидную структуру функциональных отношений, например, структуру организации.

Учитывая требования ссылочной целостности и нормализации на основе применения рассмотренных выше типов отношений осуществляется преобразование функциональной модели бизнес - процессов и реаляционную модель. Итогом этапа является диаграмма "Сущность-связь" (часто называемая CASE диаграмма, ER-диаграама, рис.10).

**Замечания**

Например:

1)Что такое традиционная база данных ?  
Бывают сетевые, иерархические и реляционные БД. Последние, в свою очередь делятся на СУБД для решения задач оперативного управления транзакциями (OLTP) и системы принятия решений (DSS).  
Почему "традиционная база данных"- это база в разделямых файлах? 2) "Ведь в реляционной безе данных проблемы синхронизации данных не возникает вовсе" - очень опасно так говорить. Если USER читает не с начала,- то он может спутать это с синхронизацией транзакций,- а это проблема ключевая в рел-ых СУБД.  
3)Подсистемы RDBMS очень схожи с соответствующими подсистемами ОС и сильно  
интегрированы с предоставляемыми базовой ОС сервисными функциями - я бы так  
не говорил.  
4) , и администратор базы данных должен будет восстанавливать часть или всю  
БД, используя файлы резервных копий (если их сделали!) - кого-копии или  
администраторов, когда они не делают копии?  
5)SYS или SYSTEM, с парролем: master или manager - а также Amum13:). (на  
самом деле - CHANGE\_ON\_INSTALL и MANAGER)  
Что касаемо, сосбтвенно, описания конкретно Oracle, - то вроде ничего  
странного я там не видел.

#### 2.1.1.3.3 Преобразование функциональной модели в реляционную.

В разделе 1.3. нами были рассмотрены основные этапы разработки автоматизированной информационной системы, в разделе 1.3.1 мы разработали функциональную модель АИС, теперь, после того как мы рассмотрели основные оложения терии баз данных, пришло время заняться непосредственно формализацией выделенных бизнес-процессов, операций и т.п. Результатом первого этапа проектирования АИС является функциональная модель системы содержащая множество объектов (процессов, операций), их атрибутов.

Объектное множество с атрибутами может быть преобразовано в реляционную таблицу с именем объектного множества в качестве имени таблицы и атрибутами объектного множества в качестве атрибутов таблицы. Если некоторый набор этих атрибутов может быть использован в качестве ключа таблицы, то он выбирается ключом таблицы. В противном случае мы добавляем к таблице атрибут, значения которого будут однозначно определять объекты-элементы исходного объектного множества, и который, таким образом, может служить ключом таблицы.

**Преобразование отношений**

Поля таблиц могут находиться между собой в обном из следующих отношений: один-к-одному, один-ко-многим, многие-ко-многим и рекурсивных, определения которых представлены в табл.1. Прежде чем рассмотреть реализацию и преобразование отношений более подробно, обсудим реторический вопрос о правилах именования таблиц и столбцов. Как мы уже ранее отмечали, что практически любая АИС имеет модульную структуру и соответствено, в каждый модель входит определенное число таблиц. Пусть имеется модуль "Операционный День", условно назовем его OPDAY, тогда удобно, что все таблицы данного модуля наименовались следующим образовам OPDAY\_CUSTOMERS (ТАБЛИЦА КЛИЕНТОВ), OPDAY\_ACCOUNT (таблица счетов) и т.п. При наменовании столбцов таблицы желательно придерживаться следующего подхода: <краткое наименование таблицы>\_<наименование столбца>. Например, для таблицы OPDAY\_CUSTOMERS наименование столбцов удобно реализовать следующим образом CUST\_NNN (порядковый номер записи), CUST\_FIO (фио клиента), CUST\_ACCOUNT\_NNN (ссылка на таблицу счетов) и т.п. Практически в каждой организации, занимающейся разработкой АИС существуют свои нормы к наименованию модулей, таблиц, столбцов и объектов базы данных, однако общие принципы во многом схожи с приведенным в данных примерах. Теперь рассмотрим основные принципы преобразования отношений:

Отношение один-к-одному.

Рассмотрим пример установки отношений клиентов и счетов в АБС (см. рис.6).

Отношение ИМЕЕТ ТЕКУЩИЙ СЧЕТ представляет собой связь один-к-одному. Это означает, что клиент имеет не более одного текущего счета и каждым текущим счетом пользуется только один клиент. Если мы решим, что ключами являются №-КЛИЕНТА для CUSTOMER (КЛИЕНТ) и №-ТЕКУЩЕГО-СЧЕТА для ACCOUNT\_NUMBER (ТЕКУЩИЙ СЧЕТ), то мы получим две реляционные таблицы, каждая из которых состоит из одного столбца.

CUSTOMER (CUST\_NNN)

ACCOUNT (ACCOUNT\_NUMBER)

Для того чтобы показать связь между этими двумя таблицами, мы должны включить ссылку на ACCOUNT\_NUMBER в таблицу CUSTOMER и и ссылку на СUST\_NNN в таблицу ACCOUNT. Каждый из этих столбцов будет внешним ключом, указывающим на другую таблицу.

CUSTOMER (CUST\_NNN, CUST\_ACCOUNT\_NUMBER )

Внешний ключ: CUST\_ACCOUNT\_NUMBER ссылается на ACCOUNT\_NUMBER.

ACCOUNT (ACCOUNT\_NUMBER, ACCOUNT\_CUST\_NNN)

Внешний ключ: ACCOUNT\_CUST\_NNN ссылается на CUST\_NNN.

Резюме: отношение один-к-одному преобразуется путем помещения одного из объектных множеств в качестве атрибута в таблицу второго объектного множества. Его выбор определяется потребностями конкретного приложения. Во многих случаях оба варианта приемлемы.

*Отношение один-ко-многим*.

Предположим, что отношение ИМЕЕТ-ТЕКУЩИЙ-СЧЕТ имеет мощность "много" со стороны ACCOUNT.

Это означает, что у клиента может быть несколько текущих счетов, но каждым текущим счетом по-прежнему пользуется только один клиент. Таким образом, в любом отношении один-ко-многим в. таблицу, описывающую объект, мощность со стороны которого равна "многим", включается столбец, являющийся внешним ключом, указывающим на другой объект.

*Отношение многие-ко-многим.*

Отношение ИМЕЕТ-ТЕКУЩИЙ-СЧЕТ имеет мощность многие-ко-многим.

Таким образом, мы предполагаем, что у клиента может быть несколько текущих счетов, и что каждым текущим счетом могут пользоваться несколько клиентов. Для того чтобы преобразовать отношение многие-ко-многим целесообразно создать таблицу пересечения, представляющую элементы двух других таблиц, находящихся в отношении многие-ко-многим.

*Рекурсивные отношения*

Объектное множество WORKER(РАБОЧИЙ), дважды встречающееся на диаграмме, и это одно и то же объектное множество в обоих случаях. Обе копии объектного множества WORKER(РАБОЧИЙ) имеют одни и те же атрибуты. В этой модели два экземпляра объектного множества WORKER(РАБОЧИЙ) использованы для удобства, чтобы показать отношение SUPERVISES(КОНТРОЛИРУЕТ), существующее между объектами WORKER(РАБОЧИЙ) и WORKER(РАБОЧИЙ). Это отношение называется *рекурсивным,* поскольку оно связывает объектное множество с ним самим. В данном случае отношение мощности один-ко-многим означает, что одному работнику подчиняются несколько других работников.

WORKER (WORKER-ID, NAME, HOURLY-RATE, WORKER-ID)

Чтобы преобразовать объектное множество WORKER вместе с его атрибутами и отношением SUPERVISES в реляционную таблицу нужно изменить имя второго атрибута WORKER-ID на имя, соответствующее отношению SUPERVISES, которое оно представляет. SUPV-ID.

WORKER (WORKER-ID, NAME, HOURLY-RАТЕ, SUPV-ID)

Внешний ключ: SUPV-ID ссылается на WORKER

SUPV-ID - это рекурсивный внешний ключ, поскольку он ссылается на WORKER-ID, то есть ключ своей собственной таблицы. Таким образом, в результате преобразования рекурсивных отношений появляются рекурсивные внешние ключи.

Функциональные зависимости, определенные для реляционной модели, являются атрибутами отношения один-к-одному или один-ко-многим.

Описанный процесс преобразования каждой из этих конструкций в атрибуты реляционных таблиц гарантирует, что они будут зависеть только от ключевых атрибутов. Таким образом, каждая полученная реляционная таблица будет иметь ЗНФ. Многозначные атрибуты реляционной модели встречаются только в отношениях многие-ко-многим. Поскольку они преобразуются в реляционные таблицы, обладающие составными ключами из ключевых атрибутов отдельных объектных множеств, то они гарантированно имеют 4НФ.

#### 2.1.2. Понятие языка определения данных (ЯОД - DBTG)

Язык - средство, при помощи которого определяется структура данных или схема, а также происходит запоминание данных и манипуляция ими. Язык, которым определяется схема, называется *языком определения данных (ЯОД),*а язык, используемый для запоминания данных и манипуляции ими, называется *языком манипуляции данными (ЯМД).*

Процедура применения ЯОД и определения схемы такова:

1. Создается концептуальная модель данных.
2. Концептуальная модель данных преобразуется в диаграмму сетевой структуры данных.
3. Проверяется, существуют ли между типами записей отношения один-ко-многим. Они могут быть непосредственно реализованы в виде наборов DBTG.
4. Если есть отношения мощности многие-ко-многим, то каждое из них преобразуется в два набора путем создания записи связи.
5. Если есть n-арные отношения, то они преобразуются в бинарные отношения.
6. Применяется ЯОД для реализации схемы.

Схема состоит из следующих частей:

1. *Раздел схемы.* Раздел схемы DBTG, задающий имя схемы.
2. *Раздел записей.* Раздел схемы DBTG, определяющий каждую запись: ее элементы данных и ее адрес.
3. *Раздел наборов.* Раздел схемы DBTG, определяющий наборы, включая типы записей владельцев и членов.

Подсхемы - это в основном, подмножества схемы. В подсхеме могут быть сгруппированы элементы данных, которые не были сгруппированы в схеме; записи и наборы могут быть переименованы и порядок описаний может быть изменен.

Принятого стандарта DBTG для подсхемы не существует; однако, обычно используются следующие отделы:

1. Отдел заголовка, позволяющий присвоить имя подсхеме и указать связанную с ней схему.
2. Отдел преобразования, в котором при желании производится замена имен из схемы на нужные в подсхеме.
3. Структурный отдел, в котором задается, какие записи, элементы данных и наборы из схемы должны присутствовать в подсхеме. Этот отдел состоит из разделов записей и наборов.

*Раздел записей подсхемы.* Раздел структурного отдела, в котором задаются записи, элементы данных и типы данных подсхемы.

*Раздел наборов подсхемы.* Раздел структурного отдела, в котором задаются наборы, которые должны быть включены в подсхему.

Подсхема позволяет пользователю строить из предопределенной схемы схему, соответствующую нуждам конкретного приложения.