РЕФЕРАТ

по дисциплине «Информатика»

на тему «Базы данных и их сравнительные характеристики»

## 1. Классификация моделей построения баз данных

В зависимости от архитектуры СУБД делятся на локальные и распределенные СУБД. Все части локальной СУБД размещаются на одном компьютере, а распределенной на нескольких. За несколько десятилетий последовательно появлялись системы (СУБД), основанные на трех базовых моделях данных: иерархической, сетевой и реляционной. Основные определения теории баз знаний и баз данных представлены в таблице 1.

Табл.1. Основные определения

|  |  |
| --- | --- |
| Термин | Определение |
| База данных (БД) | Базами данных называют электронные хранилища информации, доступ к которым осуществляется с помощью одного или нескольких компьютеров. |
| Системы управления базами данных (СУБД) | это программные средства для создания, наполнения, обновления и удаления баз данных. |
| База знаний | Базы знаний это хранилища знаний, представленных в формализованном виде. |
| Система управления базами знаний СУБЗ | это программные средства для создания, наполнения, обновления и удаления баз знаний |

|  |  |
| --- | --- |
| Виды знаний:  Процедурные  Декларативные  Каузальные  Неточные | Знания, отвечающие на вопрос "Как решать поставленную задачу?"  Знания, не содержащие в явном виде процедуры решения задач.  Знания о причинно-следственных связях между объектами предметной области  Знания отличающиеся неполнотой или противоречивостью. |
| Парадигмы решения задач  В СУБД  В СУБЗ | Данные + Алгоритм = Программа решения задачи  Знания + Стратегия вывода = Решение проблемы. |
| Модели знаний  Продукционная  Фреймовая  Семантическая сеть | Знания представленные в формате "ЕСЛИ-ТО"  Знания представленные в виде набора взаимосвязанный фреймов.  Граф, вершины которого соответствуют объектам или понятиям, а дуги определяют отношения между вершинами. |
| Фрейм  Фрейм прототип  Конкретный фрейм | Структурированное описание объекта предметной области состоящее из наименования объекта (имя фрейма), атрибутов объекта (свойств, характеристик) - слоты фрейма.  Это фрейм у которого значения слотов не определены.  Это фрейм прототип с конкретными значениями. |
| Enterprise JavaBeans. | Стандарт для создания средствами языка Java пригодных для многократного использования компонентов, из которых формируются прикладные программы. Компоненты Enterprise JavaBeans облегчают разработку программ, обеспечивающих доступ к хранимой в базе данных информации. |
| Распараллеливание обработки запроса (Intraquery parallelism). | Использование нескольких ЦП для обработки одного запроса. |
| Параллельная обработка запросов (interquery parallelism) | подразумевает параллельную обработку нескольких запросов (на разных ЦП). |
| Уровень изоляции (Isolation level). | Установочный параметр БД, определяющий, в какой степени одновременно обратившиеся к базе данных пользователи могут оказывать влияние на работу друг друга. Как правило, используются три уровня изоляции: завершение чтения (read committed), характеризуется большим количеством одновременно обслуживаемых пользователей и низким уровнем изоляции каждого из них); в установленном порядке (serializable), небольшое число одновременно обслуживаемых пользователей, высокая степень изоляции и повторяющееся чтение (repeatable read), сочетание двух первых уровней. |
| Технология СОМ | COM - Component Object Model - Компонентная модель объектов, предложена корпорацией Микрософт. |
| Технология CORBA | CORBA - Common Object Require Broker Architecture - архитектура с брокером требуемых общих объектов, разработана независимой группой OMG. |
| JDBC (Java Database Connectivity). | Интерфейс взаимодействия с базами данных на языке Java. Этот стандарт, разработанный фирмой Sun Microsystems, определяет способы доступа Java-приложений к данным БД. |
| ODBC (Open Database Connectivity). | Открытый интерфейс взаимодействия с базами данных. Предложенный корпорацией Microsoft стандарт, регулирующий доступ Windows -приложений к базам данных. Стандарт ODBC постепенно заменяется спецификацией OLE DB. |
| OLAP (Online analytical processing). | Оперативный анализ данных. Этот метод обработки применяется с целью ускорения обработки запросов и предусматривает предварительный расчет часто запрашиваемых данных (например, сумм или значений счетчика). |
| OLE DB (Object Linking and Embedding Database). | OLE для баз данных. Новый стандарт Microsoft, регулирующий доступ приложений к базам данных. Имеет расширения для серверов OLAP и предусматривает применение специальных средств обработки мультимедийных данных. |
| Операция соединения (Join). | Процесс, позволяющий объединять данные из двух таблиц посредством сопоставления содержимого двух аналогичных столбцов. |
| SQL (Structured query language). | Язык структурированных запросов, язык S0L. Является принятым в отрасли стандартом для выполнения операций вставки, обновления, удаления и выборки данных из реляционных БД. |
| Хранимая процедура (Stored procedure). | Программа, которая выполняется внутри базы данных и может предпринимать сложные действия на основе информации, задаваемой пользователем. Поскольку хранимые процедуры выполняются непосредственно на сервере базы данных, обеспечивается более высокое быстродействие, нежели при выполнении тех же операций средствами клиента БД. |
| Транзакция (Transaction). | Совокупность операций базы данных, выполнение которых не может быть прервано. Для того чтобы изменения, внесенные в БД в ходе выполнения любой из входящих в транзакцию операций, были зафиксированы в базе данных, все операции должны завершиться успешно. Все базы данных, представленные в нашем обзоре, позволяют использовать транзакции, тогда как БД для настольных систем, например Visual dBase фирмы Inprise или Microsoft Access, не предусматривают применения механизма транзакций. |
| Триггер (Trigger). | Программа базы данных, вызываемая всякий раз при вставке, изменении или удалении строки таблицы. Триггеры обеспечивают проверку любых изменений на корректность, прежде чем эти изменения будут приняты |

## 2. Иерархическая модель

Первые иерархические и сетевые СУБД были созданы в начале 60-х годов. Причиной послужила необходимость управления миллионами записей (связанных друг с другом иерархическим образом), например при информационной поддержке лунного проекта Аполлон. Среди реализуемых на практике СУБД этого типа преобладает система IMS (Information Management System компании IBM) (На данный момент это самая распространенная СУБД из всех данного типа). Применяются и другие иерархические системы: TDMS (Time-Shared Date Management System) компании Development Corporation; Mark IV Multi - Access Retrieval System компании Control Data Corporation; System - 2000 разработки SAS-Institute.

Отношения в иерархической модели данных организованы в виде совокупностей деревьев, где дерево - структура данных, в которой тип сегмента потомка связан только с одним типом сегмента предка. Графически: Предок - точка на конце стрелки, а Потомок - точка на острие стрелки. В базах данных определено, что точки - это типы записей, а стрелки представляют отношения один - к - одному или один - ко - многим.

К ограничения иерархической модели данных можно отнести:

1. Отсутствует явное разделение логических и физических характеристик модели;
2. Для представления неиерархических отношений данных требуются дополнительные манипуляции;
3. Непредвиденные запросы могут требовать реорганизации базы данных.

## 3. Сетевая модель

Сети - естественный способ представления отношений между объектами. Они широко применяются в математике, исследованиях операций, химии, физике, социологии и других областях знаний. Сети обычно могут быть представлены математической структурой, которая называется направленным графом. Направленный граф имеет простую структуру. Он состоит из точек или узлов,соединенных стрелками или ребрами. В контексте моделей данных узлы можно представлять как типы записей данных, а ребра представляют отношения один-к -одному или один-ко-многим. Структура графа делает возможными простые представления иерархических отношений (таких, как генеалогические данные) .

Сетевая модель данных - это представление данных сетевыми структурами типов записей и связанных отношениями мощности один-к-одному или один-ко-многим. В конце 60-х конференция по языкам систем данных (Conference on Data Systems Languages, CODASYL) поручила подгруппе, названной Database Task Group (DTBG), разработать стандарты систем управления базами данных. На DTBG оказывала сильное влияние архитектура, использованная в одной из самых первых СУБД, Iategrated Data Store (IDS), созданной ранее компанией General Electric.Это привело к тому, что была рекомендована сетевая модель.

Документы Database Task Group (DTBG) (группа для разработки стандартов систем управления базами данных) от 1971 года остается основной формулировкой сетевой модели, на него ссылаются как на модель CODASYL DTBG. Она послужила основой для разработки сетевых систем управления базами данных нескольких производителей. IDS (Honeywell) и IDMS (Computer Associates) - две наиболее известных коммерческих реализации. В сетевой модели существует две основные структуры данных: типы записей и наборы:

* Тип записей. Совокупность логически связанных элементов данных.
* Набор. В модели DTBG отношение один-ко-многим между двумя типами записей.
* Простая сеть. Структура данных, в которой все бинарные отношения имеют мощность один-ко-многим.
* Сложная сеть. Структура данных, в которой одно или несколько бинарных отношений имеют мощность многие-ко-многим.
* Тип записи связи. Формальная запись, созданная для того, чтобы преобразовать сложную сеть в эквивалентную ей простую сеть.

В модели DBTG возможны только простые сети, в которых все отношения имеют мощность один-к-одному или один-ко-многим. Сложные сети, включающие одно или несколько отношений многие-ко-многим, не могут быть напрямую реализованы в модели DBTG. Следствием возможности создания искусственных формальных записей является необходимость дополнительного объема памяти и обработки, однако при этом модель данных имеет простую сетевую форму и удовлетворяет требованиям DBTG.

## 4. Реляционная модель

В 1970-1971 годах Е.Ф. Кодд опубликовал две статьи, в которых ввел реляционную модель данных и реляционные языки обработки данных - реляционную алгебру и реляционное исчисление.

* Реляционная алгебра - Процедурный язык обработки реляционных таблиц.
* Реляционное исчисление - Непроцедурный язык создания запросов.

Все существующие к тому времени подходы к связыванию записей из разных файлов использовали физические указатели или адреса на диске. В своей работе Кодд продемонстрировал, что такие базы данных существенно ограничивают число типов манипуляций данными. Более того, они очень чувствительны к изменениям в физическом окружении. Когда в компьютерной системе устанавливался новый накопитель или изменялись адреса хранения данных, требовалось дополнительное преобразование файлов. Если к формату записи в файле добавлялись новые поля, то физические адреса всех записей файла изменялись. То есть такие базы данных не позволяли манипулировать данными так, как это позволяла бы логическая структура. Все эти проблемы преодолела реляционная модель, основанная на логических отношениях данных.

Существует два подхода к проектированию реляционной базы данных.

* Первый подход заключается в том, что на этапе концептуального проектирования создается не концептуальная модель данных, а непосредственно реляционная схема базы данных, состоящая из определений реляционных таблиц, подвергающихся нормализации.
* Второй подход основан на механическом преобразовании функциональной модели, созданной ранее, в нормализованную реляционную модель. Этот подход чаще всего используется при проектировании больших, сложных схем баз данных, необходимых для корпоративных информационных систем.

Табл.1. Основные определения реляционных СУБД

|  |  |
| --- | --- |
| Термин | Определение |
| Реляционная модель данных | Организует и представляет данные в виде таблиц или реляций. |
| Реляционная база данных (РБД, RDBMS). | База данных, построенная на реляционной модели. |
| Реляция (таблица-элементарная информационная единица) | Двумерная таблица, содержащая строки и столбцы данных. |
| Степень реляции. | Количество атрибутов реляции. При том необходимо помнить, что никакие два атрибута реляции не могут иметь одинаковых имен. |
| Кортежи | Строки реляции (таблицы), соответствуют объекта, конкретному событию или явлению. |
| Атрибуты | Столбцы таблицы, характеризующие признаки, параметры объекта, события, явления. |
| Область атрибута | Набор всех возможных значений, которые могут принимать атрибуты. Если в процессе работы возникает ситуация, что атрибут неприменим или значения одного или нескольких атрибутовстроки пока неизвестны, то строка запишется в базуданных с пустыми значениямиэтих атрибутов (NULL строка). |
| Пустое значение | Значение, приписываемое атрибуту в кортеже, если атрибут неприменим или его значение неизвестно |
| Ключ | Любой набор атрибутов, однозначно определяющий каждый кортеж реляционной таблицы. |
| Ключ реляции | Ключ также можно описать как минимальное множество атрибутов, однозначно определяющих (или функционально определяющих)каждое значение атрибута в кортеже. |
| Составной ключ | Ключ содержащий два или более атрибута. |
| Потенциальный ключ | В любой данной реляционной таблице может оказаться более одного набора атрибутов. Обычно в качестве первичного ключа выбирают потенциальный ключ, которым проще всего пользоваться при повседневной работе по вводу данных. |
| Первичный ключ. | Поле или набор полей, однозначно идентифицирующий запись. |
| Внешний ключ. | Набор атрибутов одной таблицы, являющийся ключом другой (или той же самой) таблицы; используется для определения логических связей между таблицами. Атрибуты внешнего ключа не обязательно должны иметь те же имена, что и атрибуты ключа, которым они соответствуют. |
| Рекурсивный внешний ключ. | Внешний ключ, ссылающийся на свою собственную реляционную таблицу. |
| Родительская реляция (таблица) | Таблица, поля которой входят в другую таблицу. |
| Дочерняя реляция (таблица) | Таблица, поля которой используют информацию из полей другой таблицы, являющейся по отношению к данной родительской. |
| Отношение один-к-одному | Когда одной записи в родительской таблицы соответствует одна запись в дочерней таблице |
| Отношение один-ко-многим | Когда одной записи в родительской таблицы соответствует несколько записей в дочерней таблице |
| Отношение многие-ко-многим | Когда многим записям в родительской таблицы соответствуют несколько записей в дочерней таблице |
| Рекурсивное отношение. | Отношение, связывающее объектное множество с ним самим. |
| View (Представления) | Информационная единица РБД (по структуре аналогичная таблице), записи которой сформированы в результате выполнения запросов к другим таблицам. |
| Ссылочная целлостность | Адекватное воспроизведение записей в ссылочных полях таблиц. |
| Триггер | Средство обеспечения ссылочной целостности на основе механизма каскадных изменений. |
| Индекс | Механизмы быстрого доступа к хранящимся в таблицах данных путем их предварительной сортировки. |
| Транзакция | Такое воздействие на СУБД, которое переводит ее из одного целостного состояния в другое. |

## 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 всего реализуется специальными объектами СУБД - триггерами). Данный механизм состоит в следующей последовательности 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 дочерней таблице.

## Процесс 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 разбиение 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.

Критерий нормальной 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 совместными 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).

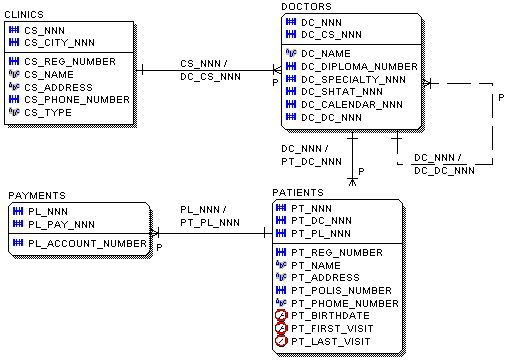


Рис. 2. Диаграмма, иллюстрирующая отношения таблиц АИС.

|  |  |  |
| --- | --- | --- |
| № | 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.2. 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.2).

Отношение один-к-одному представляет собой такое отношение, при котором каждой записи в таблице А соответствует единственная запись в таблице В (рис.1). 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.

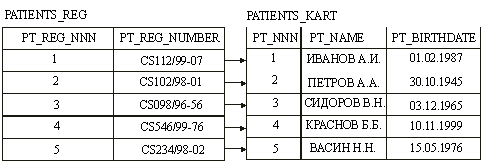


Рис.1. Отношение один к одному.

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.2). 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.

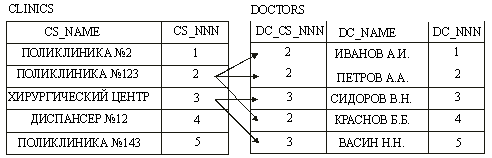


Рис.2. Отношение один ко многим.

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.3. и так не когда не делайте).

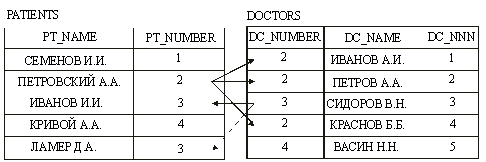


Рис.3. Отношение многие ко многим.

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.4., 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.

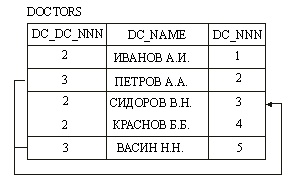


Рис.4. Отношение многие ко многим.

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.2).

## Преобразование функциональной модели в реляционную.

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.5).



Рис.5. Отношение один к одному.

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.



Рис.6. Отношение один ко многим.

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.

база данных реляционный



Рис.7. Отношение 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**

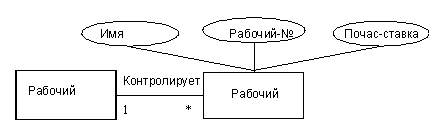


Рис.8. Рекурсивные отношения.

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.

## 5. 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.

## 6. Язык манипуляции данными (ЯМД)

Язык манипуляции данными (ЯМД) обеспечивает эффективные команды манипуляции сетевой системой базы данных. ЯМД позволяет пользователям выполнять над базой данных операции в целях получения информации, создания отчетов, а также обновления и изменения содержимого записей.

Основные команды ЯМД можно классифицировать следующим образом: команды передвижения, команды извлечения, команды обновления записей, команды обновления наборов.

Табл.2. Основные типы команд ЯМД.

|  |  |
| --- | --- |
| Наименование типа команд | Назначение |
| Команды передвижения. | Команды, применяемые для поиска записей базы данных. |
| Команды извлечения. | Команды, применяемые для извлечения записей базы данных. |
| Команды обновления записей. | Команды, применяемые для изменения значений записей. |
| Команды обновления наборов. | Команды, применяемые для добавления, изменения или удаления экземпляров наборов. |

**Список литературы**

1. Попов И.Г., Мамонов С.Г. Информационные системы. М.: Инфра, 2007.

2. Абросимов А.Г. Бородинова М.А. Теория экономических информационных систем. Учебное пособие - Самара. Изд-во Самарск.гос. экон. акад., 2007.

3. Информационные системы. Учебник /Петров В.Н. – СПб.: Питер, 2008.

4. Информационное обеспечение систем управления. Учебное пособие/Голенищев Э.П., Клименко И.В. - Ростов н/Д: Феникс, 2009.

5. Интеллектуальные информационные системы в экономике. Учебное пособие/Тельнов Ю.Ф. Издание третье, расширенное и доработанное. Серия «Экономика и бизнес». – Москва.: СИНТЕГ, 2009.