Базы данных

Серебрякова Т.А.

Введение

- **Базой данных** является представленная в объективной форме совокупность самостоятельных материалов (статей, расчетов, нормативных актов, судебных решений и иных подобных материалов), систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины Существует множество других определений, отражающих скорее субъективное мнение тех или иных авторов о том, что означает этот термин в их понимании. Так или иначе все определения сводятся к понятию *«(взаимосвязанная) совокупность хранимых данных»*, однако общепризнанная единая формулировка определения отсутствует. Наиболее часто используются следующие отличительные признаки:
- **База данных хранится и обрабатывается в вычислительной системе**. Таким образом, любые внекомпьютерные хранилища информации (архивы, библиотеки, картотеки и т. п.) базами данных не являются.
- **Данные в базе данных хорошо структурированы (систематизированы)** с целью обеспечения возможности их эффективного поиска и обработки. Под структурированностью в данном случае понимается явное выделение составных частей (элементов), связей между ними, а также типизация элементов и связей, при которой с каждым типом элемента или связи соотносится определённая семантика и допустимые операции, а эффективность определяется тем, как соотносятся гибкость и мощность возможностей (поиска и обработки) с затратами усилий и ресурсов.

Существующие модели данных

Реляционная модель данных была предложена в 1969 г. сотрудником фирмы **IBM Е.Ф. Кодцом** (Dr. **Codd E.F.).** Она представляет собой набор **"плоских файлов " -** таблиц, называемых **"отношениями ",** к которым применимы операции **реляционной алгебры** для реализации автоматизированного ответа на запросы пользователей системы .

Иерархическая модель - модель организации данных, представляющая собой древовидный **граф,** состоящий из ряда типов записей **("типов данных ")** и связей между ними **("отношений "** или "**характеристики отношений**"), причем один из типов записей определяется как корневой или входной, а остальные связаны с ним или друг с другом отношениями "один-ко-многим" или (реже) "один-к-одному".

Сетевая модель - модель организации данных, подобная иерархической **,** но отличающаяся от нее тем, что каждая запись может вступать в любое количество поименованных связей с другими записями как исходная или порожденная, или как то и другое (см. двунаправленная связь).

Существующие модели данных

Ядром любой базы данных является *модель данных*. Модель данныхпредставляет собой множество структур данных, ограничений целостности иопераций манипулирования данными. С помощью модели данных могут бытьпредставлены объекты предметной области и взаимосвязи между ними.

Модель данных - совокупность структур данных и операций их обработки. По способу установления связей между данными СУБД основывается наиспользовании трёх основных видов модели: иерархической, сетевой илиреляционной; на комбинации этих моделей или на некотором их подмножестве.

Сетевая модель

- Стандарт сетевой модели впервые был определен в 1975 году организацией CODASYL (Conference of Data System Languages), которая определила базовые понятия модели и формальный язык описания. На разработку этого стандарта большое влияние оказал американский ученый Ч.Бахман

Базовыми объектами сетевой модели являются:

- элемент данных;
- агрегат данных;
- запись;
- набор данных.

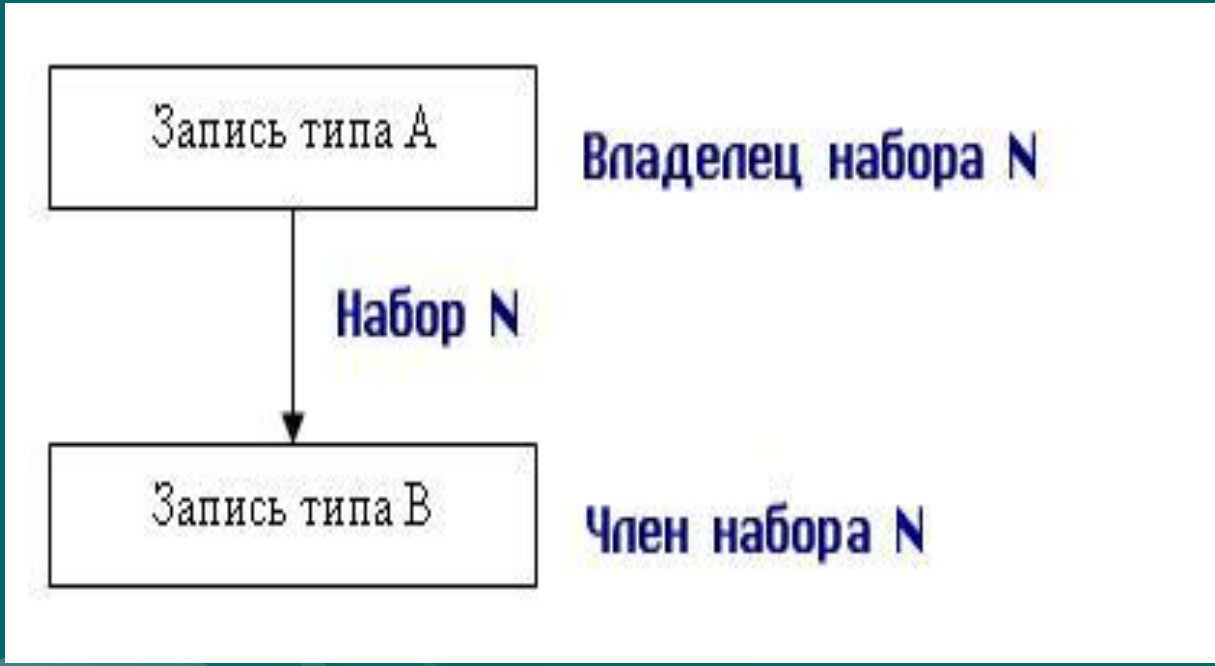
- *Элемент данных* — то же, что и в иерархической модели, то есть минимальная информационная единица, доступная пользователю с использованием СУБД.

- *Агрегат данных* соответствует следующему уровню обобщения в модели. В модели определены агрегаты двух типов: агрегат типа *вектор* и агрегат типа *повторяющаяся группа.*

- *Записью* называется совокупность агрегатов или элементов данных, моделирующая некоторый класс объектов реального мира. Понятие записи соответствует понятию «сегмент» в иерархической модели. Для записи, так же как и для сегмента, вводятся понятия типа записи и экземпляра записи.

- *Набор.* Набором называется двухуровневый граф, связывающий отношением «одии-комногим» два типа записи.

Пример набора:



Набор фактически отражает иерархическую связь между двумя типами записей. Родительский тип записи в данном наборе называется владельцем набора, а дочерний тип записи — членом того же

- Для любых двух типов записей может быть задано любое количество наборов, которые их связывают. Фактически наличие подобных возможностей позволяет промоделировать отношение «многие-ко-многим» между двумя объектами реального мира, что выгодно отличает сетевую модель от иерархической. В рамках набора возможен последовательный просмотр экземпляров членов набора, связанных с одним экземпляром владельца набора.

- Между двумя типами записей может быть определено любое количество наборов: например, можно построить два взаимосвязанных набора. Существенным ограничением набора является то, что один и тот же тип записи не может быть одновременно владельцем и членом набора.

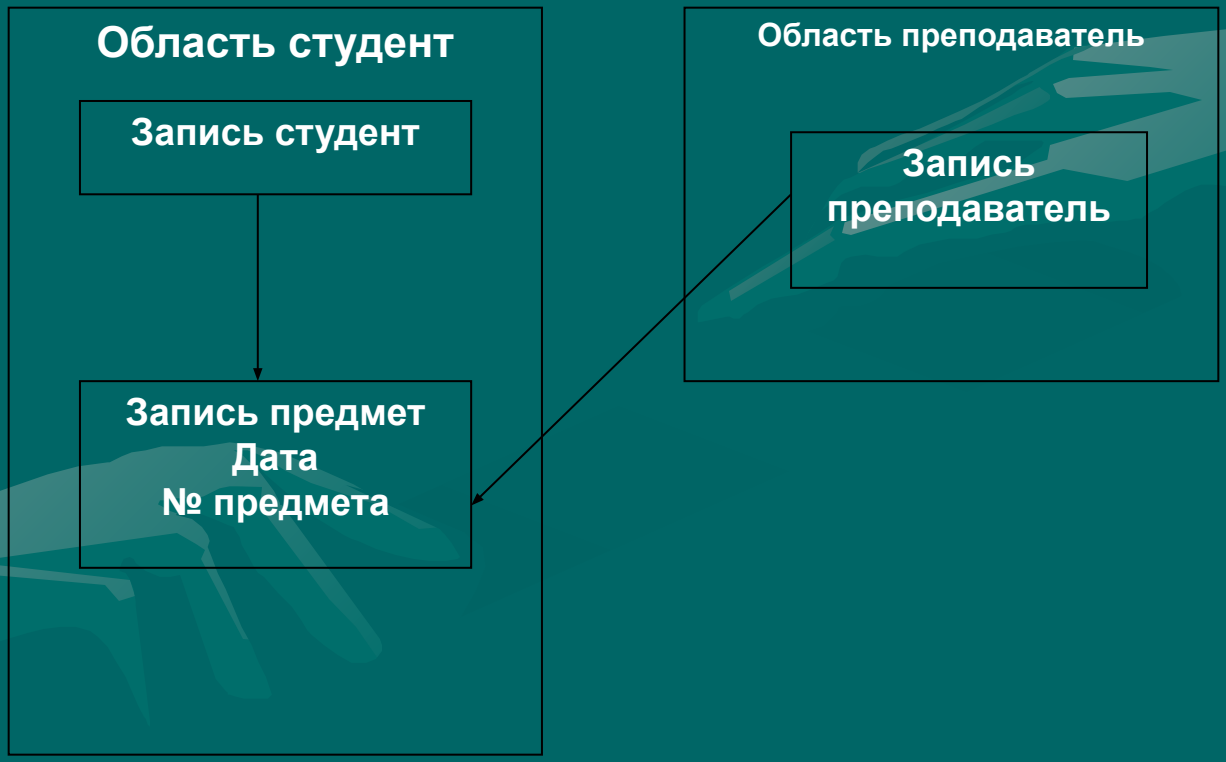
Сетевые модели данных можно разделить на:

- ***Простые*** **-** структура данных, в которой все бинарные отношения имеют мощность один-ко-многим.

- ***Сложные*** **-** структура данных, в которой одно или несколько бинарных отношений имеют мощность много-ко-многим.

В сетевой модели данных объекты предметной области объединяются в сеть, в узлах которой размещены объекты, а ребра отображают их связи.

Графически сетевую модель можно представить с помощью прямоугольников и стрелок:



Операции над данными в сетевой модели

- **ДОБАВИТЬ** - внести запись в БД и, в зависимости от режима включения, либо включить ее в групповое отношение, где она объявлена подчиненной, либо не включать ни в какое групповое отношение.
- **ВКЛЮЧИТЬ В ГРУППОВОЕ ОТНОШЕНИЕ** - связать существующую подчиненную запись с записью-владельцем.
- **ПЕРЕКЛЮЧИТЬ** - связать существующую подчиненную запись с другой записью-владельцем в том же групповом отношении.
- **ОБНОВИТЬ** - изменить значение элементов предварительно извлеченной записи.
- **ИЗВЛЕЧЬ** - извлечь записи последовательно по значению ключа, а также используя групповые отношения - от владельца можно перейти к записям - членам, а от подчиненной записи к владельцу набора.
- **УДАЛИТЬ** - убрать из БД запись. Если эта запись является владельцем группового отношения, то анализируется класс членства подчиненных записей. Обязательные члены должны быть предварительно исключены из группового отношения, фиксированные удалены вместе с владельцем, необязательные останутся в БД.
- **ИСКЛЮЧИТЬ ИЗ ГРУППОВОГО ОТНОШЕНИЯ** - разорвать связь между записью-владельцем и записью-членом.

Достоинства сетевой модели:

- наличие успешных реализаций систем управления базами данных, обеспечивающих эту сетевую модель (как и в иерархической модели);

- простота реализации часто встречающихся в реальном мире взаимосвязей "многие ко многим".

***Недостатки модели*:**

- Основной недостаток сетевой модели состоит в ее сложности;
- Прикладной программист должен детально знать логическую структуру базы данных;
- Трудности осуществления навигации среди различных экземпляров наборов и экземпляров записей;
- Возможная потеря независимости данных при реорганизации базы данных;
- Представление, используемое прикладной программой, сложнее, чем в иерархической модели.

Иерархическая структура

К основным понятиям иерархической структуры относятся уровень, элемент или узел и связь. Узел - это совокупность атрибутов, описывающих некоторый объект. На схеме иерархического дерева узлы представляются вершинами графа. Каждый узел на более низком уровне связан только с одним узлом, находящимся на более высоком уровне. Иерархическое дерево имеет только одну вершину (корень дерева), не подчиненную никакой другой вершине и находящуюся на самом верхнем (первом) уровне. Зависимые (подчиненные) узлы находятся на втором, третьем и так далее уровнях. Количество деревьев в базе данных определяется числом корневых записей.

К каждой записи базы данных существует только один (иерархический) путь от корневой записи.

Иерархическая база данных

Иерархической базой данных называется множество отношений и веерных отношений, для которых соблюдаются два ограничения

- 1. Существует единственное отношение, называемое корневым, которое не является зависимым ни в одном веерном отношении.
- 2. Все остальные отношения (за исключением корневого) являются зависимыми отношениями только в одном веерном отношении.

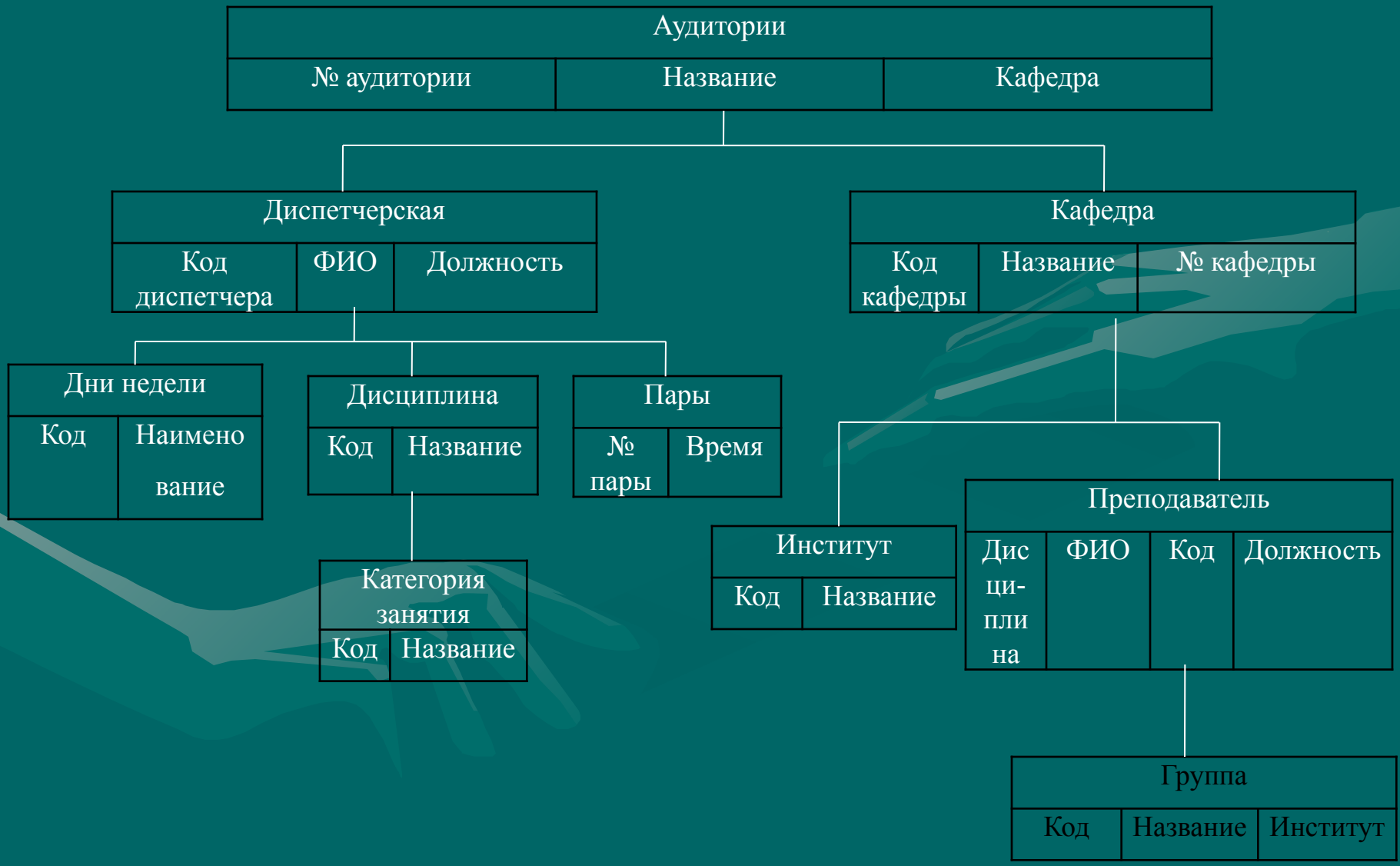
Описание иерархическая модели

Иерархическая модель данных имеет много общих черт с сетевой моделью данных, хронологически она появилась даже раньше, чем сетевая.

Иерархическая модель представляет информационные отображения объектов реального мира – сущности и их связи в виде ориентированного графа или дерева.

В иерархической модели данных допускается отображение одной предметной области в несколько иерархических баз данных.

Пример иерархической модели



О иерархической модели можно говорить как о модели ориентированных деревьев. В иерархической модели каждому узлу дерева соответствует свой тип записи. Очевидно, что каждый тип записи может присутствовать только в одном месте иерархии, поскольку в противном случае нарушается ацикличность. Каждая стрелка в дереве соответствует отношению ``один-ко-многим''. Иерархическая модель требует фактически, чтобы все сущности, кроме корневой были бы зависимыми. Чтобы ввести независимую сущность необходимо поставить ее в начало нового дерева, а на ее место поставить фиктивную сущность, которая не будет содержать данных, но будет содержать ссылку на экземпляр новой корневой сущности. Тип записи, соответствующий такой фиктивной сущности называется виртуальным.

Инфологическая модель данных "Сущность-связь"

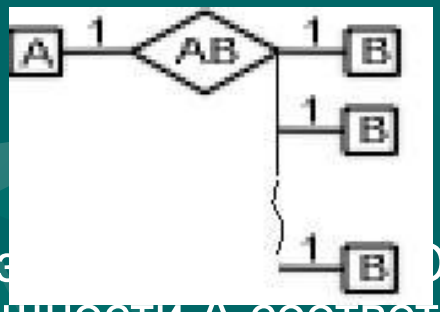
- Наиболее близка к концептуальной модели, модель “Сущность-связь”, хоть и значительно более ущербная с точки зрения пользователя. Основными конструктивными элементами инфологических моделей, являются сущности, связи между ними и их свойства.

Инфологическая модель

- ***Сущность*** – любой различимый объект. Самолет, машина, крыло, колесо – это сущности. Как и в концептуальной модели есть тип сущности и его экземпляр. Например, тип сущности – машина, а экземпляр – Москвич.
- ***Атрибут*** – поименованная характеристика сущности. Например, у машины есть атрибуты: мотор, кузов, шасси и т.д. Атрибуты используются для определения того, какая информация должна быть собрана о сущности. Любой атрибут может быть сущностью, в зависимости от точки зрения на него. Так ошейник – это сущность, но на собаке – это уже ее атрибут.
- ***Связь*** – ассоциирование двух или более сущностей.

Инфологическая модель

- **Первый тип связи** – связь ОДИН-К-ОДНОМУ (1:1): в каждый момент времени каждому представителю (экземпляру) сущности А соответствует 1 или 0 представителей сущности В. Например, работник и его ставка. В концептуальной модели можно было бы наследовать от типа работник тип строчка в ведомости, где добавить свойство сумма зарплаты, тогда указав должность работника можно узнать какие зарплаты получают работники, занимающие или занимавшие эту должность.



- **Второй тип** – связ ОГИМ (1:М): одному представителю сущности А соответствуют 0, 1 или несколько представителей сущности В.

Инфологическая модель

- ***Ключ*** – минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности. Как правило – это первичный ключ в таблице базы данных. Теперь о ***внешних ключах***:

  Если сущность С связывает сущности А и В, то она должна включать внешние ключи, соответствующие первичным ключам сущностей А и В.

  Если сущность В обозначает сущность А, то она должна включать внешний ключ, соответствующий первичному ключу сущности А.

Три основных класса сущностей

- ***Стержневая сущность*** (***стержень***) – это независимая сущность. Например, при описании накладной, стержневой сущностью является шапка накладной.
- ***Ассоциативная сущность*** (***ассоциация***) – это связь вида "многие-ко-многим". Например, товар в накладной – это связь с шапкой накладной и справочником наименований товара, справочником единиц измерения.
- ***Характеристическая сущность*** (***характеристика***) – это связь вида "многие-к-одной" или "одна-к-одной" между двумя сущностями (частный случай ассоциации). Единственная цель характеристики в рамках рассматриваемой предметной области состоит в описании или уточнении некоторой другой сущности. Это что-то вроде перечисления. Например, Список поставщиков – это список указателей на отдельные записи из справочника организаций. При указании поставщика в накладной, Вы выбираете его из списка поставщиков, но реально указываете организацию из справочника организаций. Просто организация может быть и поставщиком, и покупателем, и налоговым органом, но Вам удобнее будет выбирать из более короткого списка.

Реляционная модель

Общая характеристика реляционной модели данных

Основы реляционной модели данных были впервые изложены в статье Е.Кодда в 1970 г. Эта работа послужила стимулом для большого количества статей и книг, в которых реляционная модель получила дальнейшее развитие. Наиболее распространенная трактовка реляционной модели данных принадлежит К.Дейту. Согласно Дейту, реляционная модель состоит из трех частей:

- Структурной части.
- Целостной части.
- Манипуляционной части.

Реляционная модель

- ***Структурная часть*** описывает, какие объекты рассматриваются реляционной моделью. Постулируется, что единственной структурой данных, используемой в реляционной модели, являются нормализованные n-арные отношения.
- ***Целостная часть*** описывает ограничения специального вида, которые должны выполняться для любых отношений в любых реляционных базах данных. Это ***целостность сущностей*** и ***целостность внешних ключей***.
- ***Манипуляционная часть*** описывает два эквивалентных способа манипулирования реляционными данными - ***реляционную алгебру*** и ***реляционное исчисление***.

Реляционная модель

Типы данных

Любые данные, используемые в программировании, имеют свои типы данных.

Важно! Реляционная модель требует, чтобы типы используемых данных были *простыми*.

Для уточнения этого утверждения рассмотрим, какие вообще типы данных обычно рассматриваются в программировании. Как правило, типы данных делятся на три группы:

- Простые типы данных.
- Структурированные типы данных.
- Ссылочные типы данных.

Реляционная модель

Простые типы данных

Простые, или атомарные, типы данных не обладают внутренней структурой. Данные такого типа называют ***скалярами***. К простым типам данных относятся следующие типы:

- Логический.
- Строковый.
- Численный.

Различные языки программирования могут расширять и уточнять этот список, добавляя такие типы как:

- Целый.
- Вещественный.
- Дата.
- Время.
- Денежный.
- Перечислимый.
- Интервальный.

Конечно, понятие атомарности довольно относительно. Так, строковый тип данных можно рассматривать как одномерный массив символов, а целый тип данных - как набор битов. Важно лишь то, что при переходе на такой низкий уровень теряется ***семантика (смысл) данных***. Если строку, выражающую, например, фамилию сотрудника, разложить в массив символов, то при этом теряется смысл такой строки как единого целого.

Реляционная модель

Структурированные типы данных предназначены для задания сложных структур данных. Структурированные типы данных конструируются из составляющих элементов, называемых компонентами, которые, в свою очередь, могут обладать структурой. В качестве структурированных типов данных можно привести следующие типы данных:

- Массивы
- Записи (Структуры)

Реляционная модель

- ***Ссылочный тип данных*** (***указатели***) предназначен для обеспечения возможности указания на другие данные. Указатели характерны для языков процедурного типа, в которых есть понятие области памяти для хранения данных. Ссылочный тип данных предназначен для обработки сложных изменяющихся структур, например деревьев, графов, рекурсивных структур.

Реляционная модель

Отношения, атрибуты, кортежи отношения

Определения и примеры

Фундаментальным понятием реляционной модели данных является понятие ***отношения***.

Определение 1. ***Атрибут отношения*** *есть пара вида <Имя_атрибута : Имя_домена>.*

Имена атрибутов должны быть уникальны в пределах отношения. Часто имена атрибутов отношения совпадают с именами соответствующих доменов.

Определение 2. ***Отношение***, определенное на множестве доменов $D_1, D_2, \ldots, D_n$ (не обязательно различных), содержит две части: заголовок и тело.

Реляционная модель

Достоинства модели:

- Простота представления данных;
- Запросы не строятся на основе заранее определенной структуры - могут быть сформулированы на непроцедурном языке;
- Независимость данных;
- Реляционная модель хранения данных основана на хорошо проработанной теории отношений;
- При проектировании базы данных применяются строгие методы, построенные на использовании реляционной алгебры;
- Простота внесения изменений в базу данных.

Реляционная модель

Недостатки модели

- Невозможность представления объектов с отношением «многие-ко-многим» в одной таблице;
- Значительно большее время реакции на запросы;
- Больший объем внешней памяти.

Нормализация отношений

Отношение называется **нормализованным**, если значение каждого атрибута в каждом кортеже является атомарным (неделимым).

В реляционной модели данных поддерживаются только нормализованные отношения по следующим причинам:

- такой подход не налагает ограничений на то, что можно описывать с помощью нормализованных отношений;
- полученное упрощение в структуре данных ведет к соответствующим упрощениям в операторах манипулирования данными.

Нормализация – это разбиение таблицы на две или более, обладающих лучшими свойствами при включении, изменении и удалении данных. Окончательная цель нормализации сводится к получению такого проекта базы данных, в котором ***каждый факт появляется лишь в одном месте***, т.е. исключена избыточность информации. Это делается не столько с целью экономии памяти, сколько для исключения возможной противоречивости хранимых данных и предсказуемости поведения системы во время эксплуатации. Последний факт полезен для понимания структуры данных пользователем, а значит ускорения обучаемости и исключения случайных ошибок в работе.

Первая нормальная форма

Определение. Отношение R находится в 1НФ тогда и только тогда, когда все входящие в него значения (домены) содержат только атомарные значения.

Это значит, что любое нормализованное отношение находится в 1 НФ.

Вторая нормальная форма

Определение. Отношение находится во 2НФ, если оно находится в 1НФ и каждый неключевой атрибут функционально полно зависит от составного ключа.

Чтобы отношение привести ко 2НФ, необходимо:

- построить его проекцию, исключив атрибуты, которые не находятся в полной функциональной зависимости от составного ключа;
- построить проекцию (в общем случае не одну), использовав часть составного ключа и атрибуты, функционально зависящие от этой части составного ключа.

Третья нормальная форма

- **Определение.** Отношение R находится в 3НФ, если оно находится в 2НФ и каждый неключевой атрибут нетранзитивно зависит от первичного ключа.
- Отношение, находящееся в 2НФ и не находящееся в 3НФ, всегда может быть преобразовано в эквивалентную совокупность отношений 3НФ. Для преобразования отношения к 3НФ необходимо построить несколько отношений.

Четвертая нормальная форма

Р.Фейгин определил четвертую нормальную форму (4НФ), в которой находятся некоторые отношения ЗНФ.

4НФ применяется к схемам отношений с многозначными зависимостями. 4НФ запрещает хранить независимые элементы, когда между этими элементами существует связь типа "многие-ко-многим". 4НФ требует, чтобы такие элементы запоминались в отдельных отношениях.

Определение. Говорят, что отношение R находится в 4НФ, если оно находится в НФБК и в нем отсутствуют невависимые многозначные вависимости, т.е. все невависимые многозначные зависимости выделены в отдельные отношения с одним и тем же ключом.

Сравнение нормализованных и ненормализованных моделей

Критерий	Отношения слабо нормализованы (1НФ, 2НФ)	Отношения сильно нормализованы (3НФ)
Адекватность базы данных предметной области	ХУЖЕ (-)	ЛУЧШЕ (+)
Легкость разработки и сопровождения базы данных	СЛОЖНЕЕ (-)	ЛЕГЧЕ (+)
Скорость выполнения вставки, обновления, удаления	МЕДЛЕННЕЕ (-)	БЫСТРЕЕ (+)
Скорость выполнения выборки данных	БЫСТРЕЕ (+)	МЕДЛЕННЕЕ (-)

При проектировании БД требуется различать взаимосвязи:

между объектами

между атрибутами одного объекта

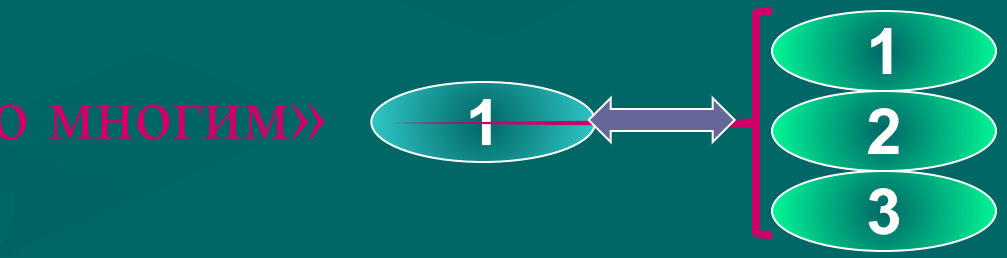
и между атрибутами различных объектов

Взаимосвязь показывает взаимодействие 2-х множеств различных объектов.

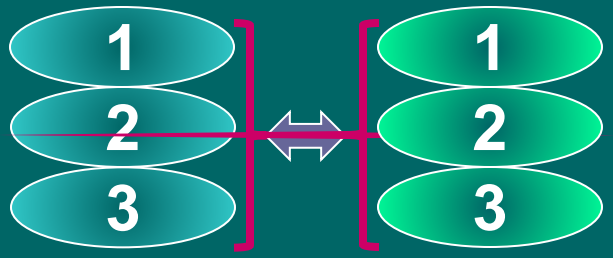
Различают взаимосвязи типа:

«один к одному»

«один ко многим»



«многие ко многим»



Если объект второго множества взаимодействует с одним конкретным объектом первого множества, то тогда делается ВЫВОД:

Тип этой связи - один-ко-многим.

Если объект второго множества взаимодействует со многими объектами первого множества, то тогда делается ВЫВОД:

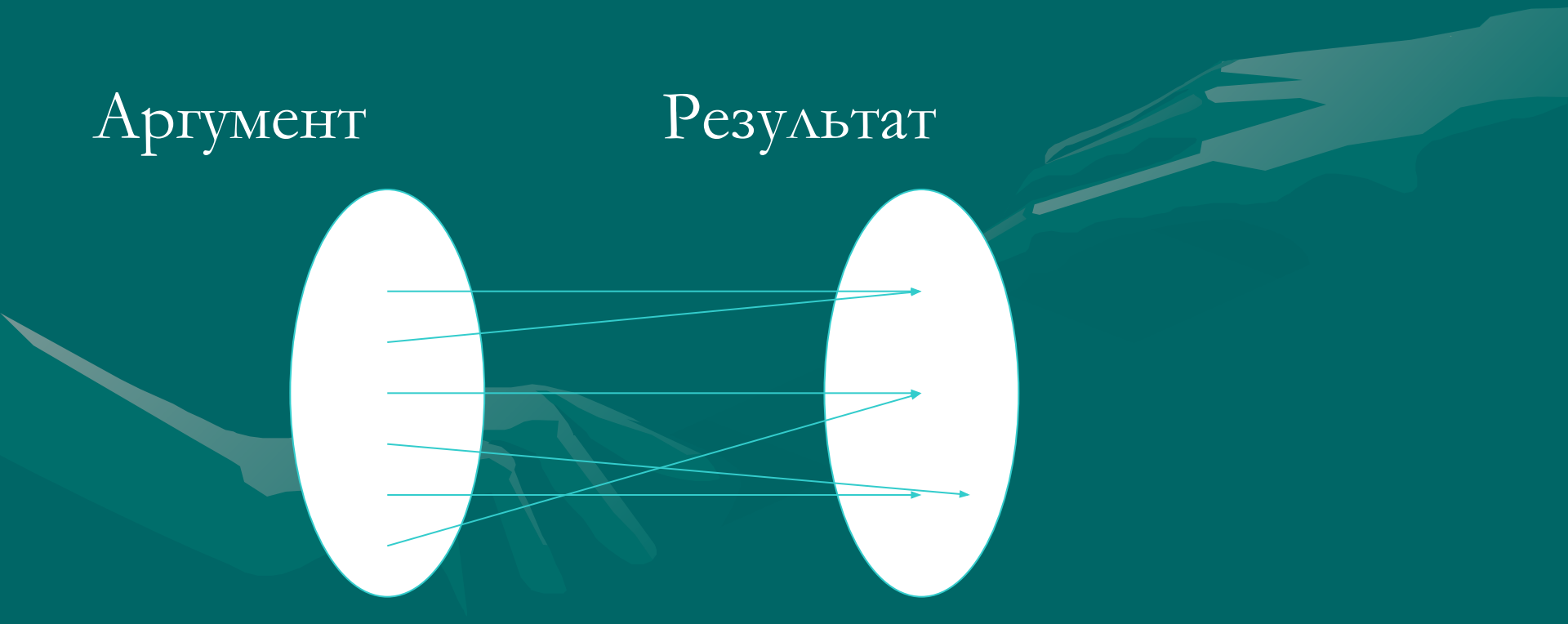
Тип этой связи - многие-ко-многим.

Если же объекты первого и второго множеств одинаково взаимодействуют друг с другом в одиночку, то делается ВЫВОД:

Тип этой связи - один-к-одному.

Отображение «Многие к одному»

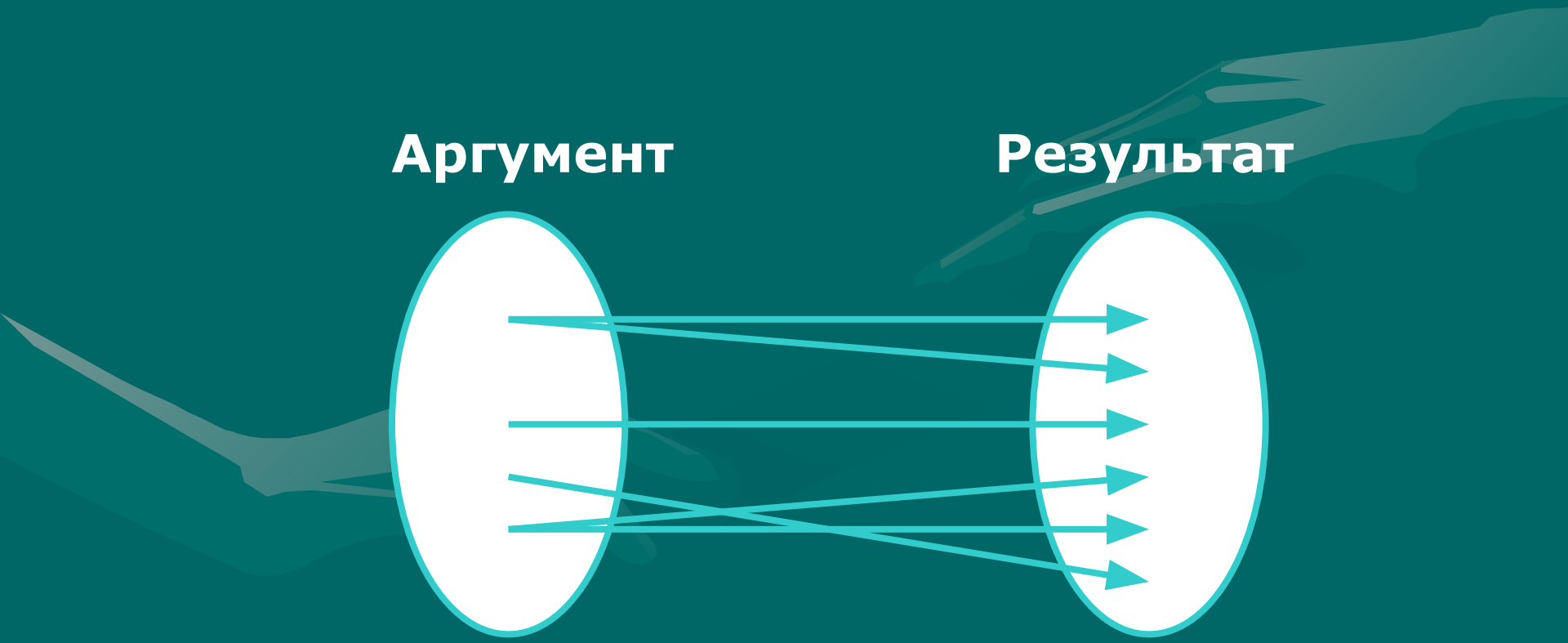
Отображение имеет тип *Многие к одному*, если оно является функцией



Из значения аргумента выходит одна стрелка

Отображение «Один ко многим»

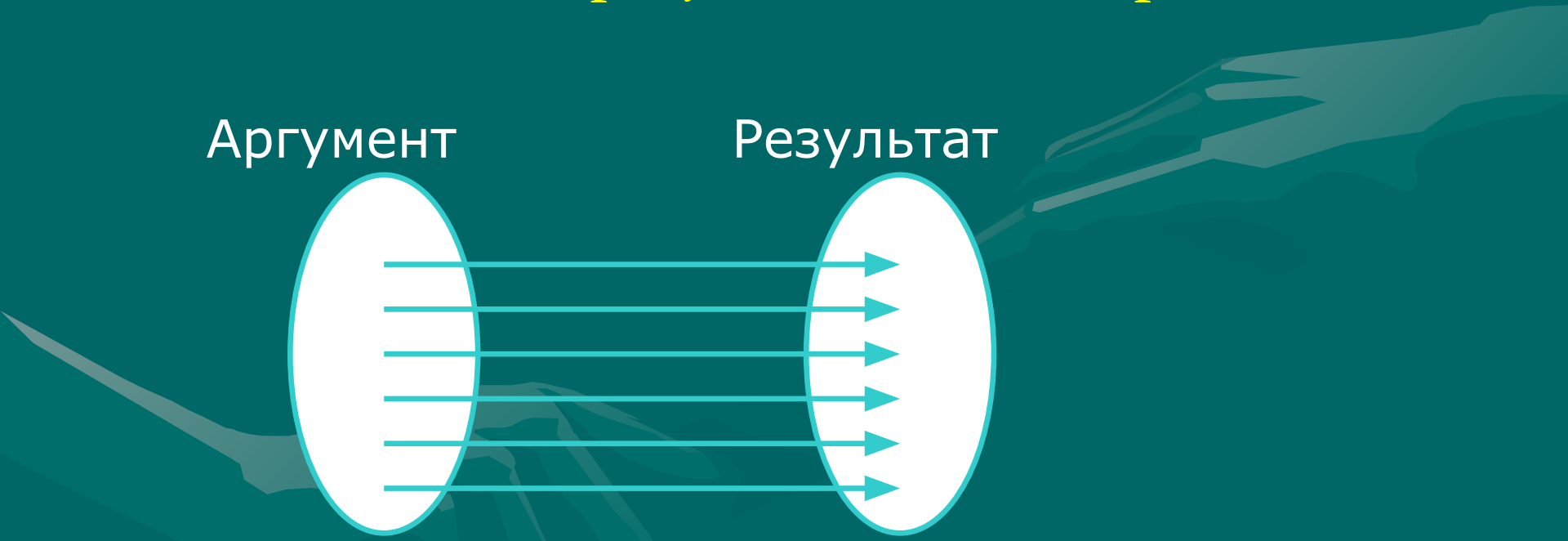
Отображение имеет тип *Один ко многим*, если для каждого значения результата отображения имеется только одно значение аргумента. При этом одно значение аргумента может отображаться в несколько значений результата



В каждое значение результата входит одна стрелка

Отображение «один к одному»

Отображение имеет тип ***Один к одному***, если каждому значению аргумента соответствует одно значение результата и наоборот.



Из каждого значения аргумента выходит ровно одна стрелка и в каждое значение результата входит тоже ровно одна стрелка

Целостность данных

Правила, обеспечивающие поддержание установленных межтабличных связей при вводе или удалении записей. Если наложены условия целостности данных, Access не позволяет добавлять в связанную таблицу записи, для которых нет соответствующей записей в главной таблице, или же изменять записи в главной таблице таким образом, что после этого в связанной таблице появятся записи, не имеющие соответствующих главных записей, а также удалять записи в главной таблице, для которых имеются подчиненные записи в связанной таблице».

Целостность данных

Правила, обеспечивающие поддержание установленных межтабличных связей при вводе или удалении записей. Если наложены условия целостности данных, Access не позволяет добавлять в связанную таблицу записи, для которых нет соответствующей записей в главной таблице, или же изменять записи в главной таблице таким образом, что после этого в связанной таблице появятся записи, не имеющие соответствующих главных записей, а также удалять записи в главной таблице, для которых имеются подчиненные записи в связанной таблице».

Параметры целостности:

Обеспечение целостности

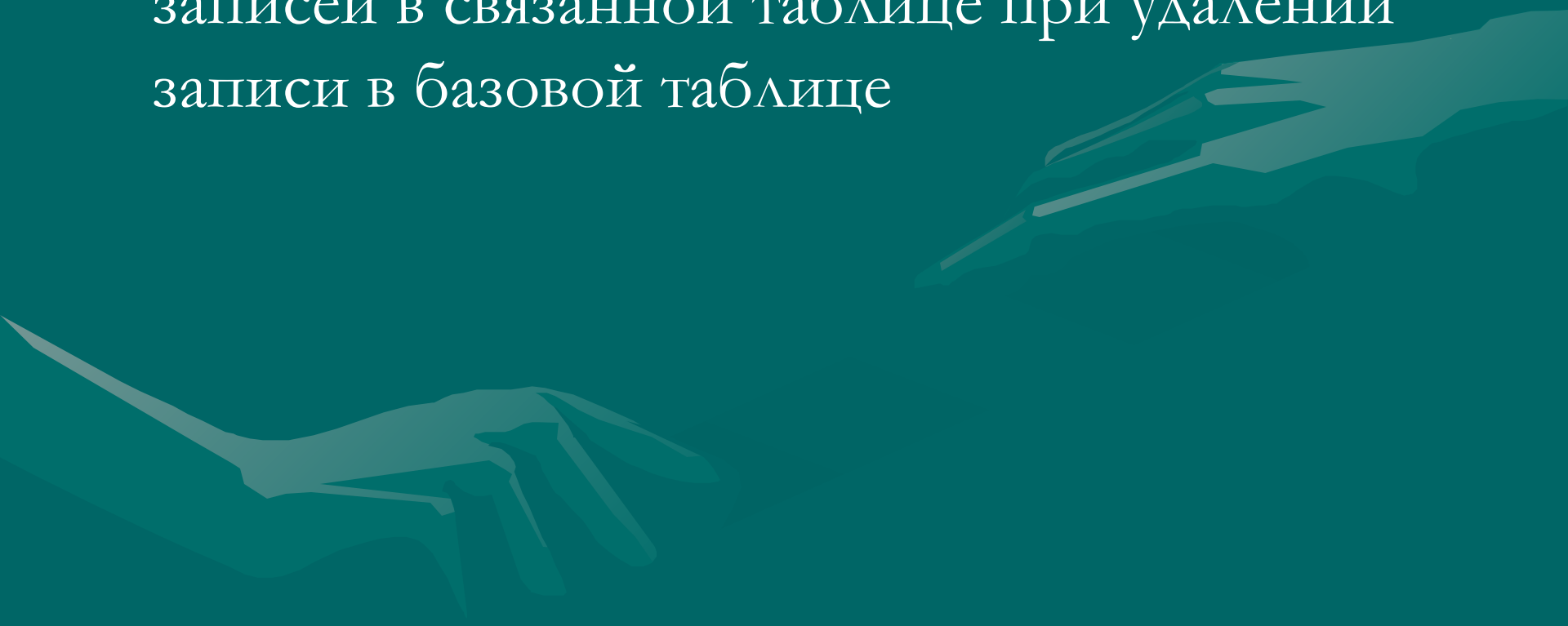
Этот параметр устанавливается только в том случае, если соответствующее поле главной таблицы является первичным ключом, связанные поля имеют один и тот же тип данных или обе таблицы содержатся в одной БД

Каскадное обновление связанных полей:

Для автоматического обновления соответствующих значений в связанной таблице при изменении значения ключевого поля в базовой таблице

Каскадное удаление связанных записей:

Для автоматического удаления связанных записей в связанной таблице при удалении записи в базовой таблице



Главная таблица: типы связи

Если оба связываемых атрибута не являются ключевыми, то главной будет таблица, от которой пользователь начинает протягивать связь. В этом случае тип связи не устанавливается.

Главная таблица: типы связи

Если в связываемых таблицах ровно один из двух связываемых атрибутов объявлен ключевым, то главной будет та таблица, к которой относится ключевой атрибут. В этом случае при установлении обеспечения целостности данных получается связь типа «один ко многим».

Главная таблица: типы связи

Если в связываемых таблицах оба связываемых атрибута объявлены ключевыми, то главная таблица назначается пользователем (протягиванием в нужном направлении связи между атрибутами). При установлении обеспечения целостности данных получается связь типа «один к одному»

Связь

«Связь, это пара таблиц, в каждой из которых выделено по набору атрибутов, с указанием типа соединения и параметров целостности»
А. Г. Гейн

Свойство связи

1. Если связь имеет тип «один к одному», то каждая строка главной таблицы связана не более чем с одной строкой подчиненной таблицы и каждая строка подчиненной таблицы связана в точности с одной строкой главной таблицы.

Свойство связи

2. Если связь имеет тип «один ко многим», то каждая строка подчиненной таблицы связана в точности с одной строкой главной таблицы, но каждая строка главной таблицы может быть связана с несколькими строками подчиненной таблицы.

Свойство связи

3. Если одной записи в главной таблице могут соответствовать несколько записей связанной таблицы, и наоборот, одной записи в подчиненной таблице могут соответствовать несколько записей главной таблицы, такая запись называется «**многие ко многим**». Две таблицы, находящиеся в отношении «многие ко многим» могут быть связаны с помощью третьей (промежуточной) таблицы, в которой присутствуют по одному атрибуту, в точности повторяющие один из атрибутов связанной и главной таблицы. Промежуточная таблица должна быть связана с двумя другими таблицами по данным атрибутам связью «один к одному» или «один ко многим».

Языки манипулирования данными

- SQL (обычно произносимый как "СИКВЭЛ" или "ЭСКЮЭЛЬ") символизирует собой *Структурированный Язык Запросов*. Это - язык, который дает Вам возможность создавать и работать в реляционных базах данных, являющихся наборами связанной информации, сохраняемой в таблицах.
- Стандарт SQL и QBE определяется ANSI (*Американским Национальным Институтом Стандартов*) и в данное время также принимается ISO (*Международной Организацией по Стандартизации*)

Обзор языка SQL

- Язык SQL (Structured Query Language - структурированный язык запросов) представляет собой стандартный высокоуровневый язык описания данных и манипулирования ими в системах управления базами данных (СУБД), построенных на основе реляционной модели данных.
- Язык SQL был разработан фирмой IBM в конце 70-х годов. Первый международный стандарт языка был принят международной стандартизирующей организацией ISO в 1989 г., а новый (более полный) - в 1992 г.. В настоящее время все производители реляционных СУБД поддерживают с различной степенью соответствия стандарт SQL92.
- Единственной структурой представления данных (как прикладных, так и системных) в реляционной базе данных (БД) является двумерная таблица. Любая таблица может рассматриваться как одна из форм представления теоретико-множественного понятия **отношение** (relation), отсюда название модели данных - реляционная.

В настоящее время наибольшее распространение получили реляционные SQL СУБД двух групп:
1)мощные крупные коммерческие СУБД, ориентированные на хранение 2)огромных объемов информации (от гигабайт);
мобильные компактные свободно распространяемые (в том числе и в исходных кодах) СУБД, использование которых оправдано и для БД объемом всего лишь в десятки килобайт.

Наиболее известными СУБД первой группы являются:

- Sybase SQLserver фирмы Sybase, Inc.;
- Oracle фирмы Oracle Corporation;
- Ingres фирмы Computer Associates International;
- Informix фирмы Informix Corporation.

К наиболее популярным СУБД второй группы относятся:

- PostgreSQL организации PostgreSQL;
- microSQL фирмы Hughes Technologies Pty. Ltd.;
- mySQL фирмы T.C.X DataKonsult AB.

Основы синтаксиса языка SQL

Программа на языке SQL представляет собой простую линейную последовательность операторов языка SQL. Язык SQL в своем чистом виде операторов управления порядком выполнения запросов к БД (типа циклов, ветвлений, переходов) не имеет.

Основы синтаксиса языка SQL

Операторы языка SQL строятся с применением:

- зарезервированных ключевых слов;
- идентификаторов (имен) таблиц и столбцов таблиц;
- логических, арифметических и строковых выражений, используемых для формирования критериев поиска информации в БД и для вычисления значений ячеек результирующих таблиц;
- идентификаторов (имен) операций и функций, используемых в выражениях.

Основы синтаксиса языка SQL

Допустимыми разделителями лексических единиц в операторе являются:

- один или несколько пробелов,
- один или несколько символов табуляции,
- один или несколько символов новая строка.

Типы данных языка SQL

Базовыми принято считать следующие типы данных:

- INT[(*len*)] - целое число длиной 4 байта, представляемое при выводе максимально *len* цифрами;
- SMALLINT[(*len*)] - целое число длиной 2 байта, представляемое при выводе максимально *len* цифрами;
- FLOAT[(*len*,*dec*)] - действительное число, представляемое при выводе максимально *len* символами с *dec* цифрами после десятичной точки;
- CHAR(*size*) - строка символов фиксированной длины размером *size* символов;
- VARCHAR(*size*) - строка символов переменной длины максимальным размером до *size* символов;
- BLOB (Binary Large OBject) - массив произвольных (двоичных) байтов (максимальный размер зависит от реализации, обычно это 65535 байт); этот тип данных может использоваться, например, для хранения изображений;
- DATE - астрономическая дата;
- TIME - астрономическое время.

Типы данных языка SQL

Символьные константы (типа CHAR и VARCHAR) записываются как последовательности символов, заключенные в одиночные апострофы, например brass (латунь).
Десятичные константы (типа FLOAT) могут записываться в научной нотации как последовательности следующих компонент:

- знак числа;
- десятичное число с точкой;
- символ e;
- знак (+ или -) показателя степени;
- целое число, играющее роль показателя степени числа 10

Обзор Язык манипулирования данными QBE

Разработанный модуль, предназначенный для формирования исполняемых запросов, создания хранимых запросов на основе языка Query-by-Example и его расширения на универсальные схемы баз данных . Предлагаемый модуль оформлен как plug-in к СУБД Caché.

Обзор Язык манипулирования данными QBE

- QBE обладает высоким быстродействием.
- Интерфейс выполнен с использованием технологий CSP. Программирование на QBE осуществляется посредством таблиц-шаблонов, которые формируются в соответствии со схемой базы
- Предусмотрены две возможности. Можно создать SQL-фразу и использовать ее, а можно сгенерировать запрос на Caché Object Script с прямым доступом к глобалам.
- Полученные первые экспериментальные результаты показали, что QBE запросы, формируемые как методы в COS, работают быстрее, чем соответствующие SQL-запросы.
- Имеющаяся возможность просмотреть SQL-фразу эквивалентную QBE-запросу может быть использована для быстрого обучения QBE и как средство дополнительного контроля правильности запроса.

Этапы проектирования

Концептуальное проектирование - сбор, анализ и редактирование требований к данным. Для этого осуществляются следующие мероприятия:

- обследование предметной области, изучение ее информационной структуры
- выявление всех фрагментов, каждый из которых харакетризуется пользовательским представлением, информационными объектами и связями между ними, процессами над информационными объектами
- моделирование и интеграция всех представлений

По окончании данного этапа получаем концептуальную модель, инвариантную к структуре базы данных. Часто она представляется в виде модели "сущность-связь".

Логическое проектирование - преобразование требований к данным в структуры данных. На выходе получаем СУБД-ориентированную структуру базы данных и спецификации прикладных программ. На этом этапе часто моделируют базы данных применительно к различным СУБД и проводят сравнительный анализ моделей.

Физическое проектирование - определение особенностей хранения данных, методов доступа и т.д.

Различие уровней представления данных на каждом этапе проектирования

КОНЦЕПТУАЛЬНЫЙ УРОВЕНЬ сущности атрибуты связи	Представление аналитика
ЛОГИЧЕСКИЙ УРОВЕНЬ записи элементы данных связи между записями	Представление программиста
ФИЗИЧЕСКИЙ УРОВЕНЬ группирование данных индексы методы доступа	Представление администратора

Этапы проектирования

I этап. Постановка задачи.

На этом этапе формируется задание по созданию БД. В нем подробно описывается состав базы, назначение и цели ее создания, а также перечисляется, какие виды работ предполагается осуществлять в этой базе данных (отбор, дополнение, изменение данных, печать или вывод отчета и т. д).

Этапы проектирования

II этап. Анализ объекта.

На этом этапе рассматривается, из каких объектов может состоять БД, каковы свойства этих объектов. После разбиения БД на отдельные объекты необходимо рассмотреть свойства каждого из этих объектов, или, другими словами, установить, какими параметрами описывается каждый объект. Все эти сведения можно располагать в виде отдельных записей и таблиц. Далее необходимо рассмотреть тип данных каждой отдельной единицы записи. Сведения о типах данных также следует занести в составляемую таблицу.

Этапы проектирования

III этап. Синтез модели.

На этом этапе по проведенному выше анализу необходимо выбрать определенную модель БД. Далее рассматриваются достоинства и недостатки каждой модели и сопоставляются с требованиями и задачами создаваемой БД. После такого анализа выбирают ту модель, которая сможет максимально обеспечить реализацию поставленной задачи. После выбора модели необходимо нарисовать ее схему с указанием связей между таблицами или узлами.

Этапы проектирования

IV этап. Выбор способов представления информации и программного инструментария.

После создания модели необходимо, в зависимости от выбранного программного продукта, определить форму представления информации.

В большинстве СУБД данные можно хранить в двух видах:

- с использованием форм;
- без использования форм.
- **Форма** – это созданный пользователем графический интерфейс для ввода данных в базу.

Этапы проектирования

V этап. Синтез компьютерной модели объекта.

В процессе создания компьютерной модели можно выделить некоторые стадии, типичные для любой СУБД.

- **Стадия 1.** Запуск СУБД, создание нового файла базы данных или открытие созданной ранее базы.
- **Стадия 2.** Создание исходной таблицы или таблиц.

Создавая исходную таблицу, необходимо указать имя и тип каждого поля. Имена полей не должны повторяться внутри одной таблицы. В процессе работы с БД можно дополнять таблицу новыми полями. Созданную таблицу необходимо сохранить, дав ей имя, уникальное в пределах создаваемой базы.

Этапы проектирования

V этап. Синтез компьютерной модели объекта.

- **Стадия 3.** Создание экранных форм.

  Первоначально необходимо указать таблицу, на базе которой будет создаваться форма. Ее можно создавать при помощи мастера форм, указав, какой вид она должна иметь, или самостоятельно. При создании формы можно указывать не все поля, которые содержит таблица, а только некоторые из них. Имя формы может совпадать с именем таблицы, на базе которой она создана. На основе одной таблицы можно создать несколько форм, которые могут отличаться видом или количеством используемых из данной таблицы полей. После создания форму необходимо сохранить. Созданную форму можно редактировать, изменяя местоположение, размеры и формат полей.

- **Стадия 4.** Заполнение БД.

  Процесс заполнения БД может проводиться в двух видах: в виде таблицы и в виде формы. Числовые и текстовые поля можно заполнять в виде таблицы, а поля типа MEMO и OLE – в виде формы.

Этапы проектирования

VI этап. Работа с созданной базой данных.

Работа с БД включает в себя следующие действия:

- **поиск необходимых сведений;**
- **сортировка данных;**
- **отбор данных;**
- **вывод на печать;**
- **изменение и дополнение данных.**


Тема
закончена