Лекция 3. Понятие модели. Типы связей. Модель сущность-связь. Иерархическая, сетевая и объектно-ориентированная модели данных.

Модель данных - это средство абстракции, позволяющее отобразить информационное содержание данных.

В самом простейшем виде модель данных может быть представлена простым перечнем той информации, которая должна храниться в информационной системе. Но так как данные, представляющие эту информацию, практически всегда структурированы и тесно взаимосвязаны, то необходимо представить каким-либо образом и их структуру.

Иначе говоря, **модель данных представляет собой совокупность правил**, которые могут быть использованы для описания структуры данных. Моделей для отображения одной и той же информации можно придумать множество. Критерием выбора будет **максимум количества информации, извлекаемой на данных**.

Объектом называется элемент информационной системы, сведения о котором мы сохраняем. Объекты могут быть реальными и абстрактными, а также могут представлять некоторую концепцию или событие. Всякий объект представляется посредством своих свойств и находится в некоторых отношениях с другими объектами.

Классом (типом) *объектов* называют совокупность объектов, обладающих одинаковым набором свойств (студент, преподаватель).

Выделенную по некоторым признакам совокупность объектов реального или абстрактного мира, или относящихся к какой-либо области знаний, или необходимых для достижения какой-то цели называют ***предметной областью*** *(БГУ).*

Данные имеют три области своего представления.

Первая область - это *реальный мир.* Объекты являются понятиями реального мира, в котором существуют и имеют некоторые свойства.

Вторая – *область информации* (иначе область идей, которые существуют в голове программиста или разработчика), рассматривает **атрибуты объектов**. ***Атрибут*** - это информационное отражение свойств объекта. Каждый объект характеризуется рядом основных атрибутов. Например, человек может быть охарактеризован такими атрибутами, как ФИО, год рождения, номер паспорта. адрес и т. д.

Третья область - это *область сохраненных данных.* Она может быть бумажным источником или представлять собой память компьютера, в которой используются строки символов или совокупность битов для кодирования элементов информации.

Так как объектов некоторого типа может быть множество, то область представления данных (***область информации***) в свою очередь могут быть разбиты на две подобласти.

Первая дает общую характеристику объекта как некоторого типа в виде **совокупности имен атрибутов** . Для примера выше объект «человек» будет иметь атрибуты ФИО, год рождения, номер паспорта и т. д.

Вторая подобласть состоит из совокупностей значений атрибутов каждого экземпляра объекта в отдельности. Например, Петров (Петров Петр Петрович. 1966, У-НО № 654723); Иванов (Иванов Иван Иванович, 1961,1У-ВЛ№ 865279) в т. д.

Исходя из этого, дадим определение **экземпляру объекта**: ***экземпляром объекта*** называется единичный набор принимаемых элементами данных (атрибутов) значений.

Атрибуты могут быть:

-простыми и составными

-однозначными и многозначными

-сохраняемыми и вычисляемыми

-иметь пустое значение

Составные атрибуты складываются (или могут быть разделены) на нескольких простых, например номер и серия паспорта (используется только как целое). Обычно разложение происходит в иерархическом порядке. Такие составные элементы данных могут использоваться, если ссылки на них возможны только как на целое. Если же возможны ссылки на часть элемента данных, то их лучше разложить на простые (пример адрес, дата рождения).
Многозначными называют атрибуты, которые могут принимать набор значений. Например, телефон и адрес сотрудника.
Вычисляемыми (производными) называют элементы данных, которые могут быть вычислены на основе уже существующих. Например, возраст человека легко можно вычислить из текущей даты и дня рождения, а среднюю зарплату из имеющихся данных о ежемесячной з/п.

Специальное пустое значение (Null) вводится для элементов данных, которые не имеют значений или могут быть опущены при вводе.

Различают следующие три случая:

-не известно, какое значение имеет атрибут (например неизвестна страна происхождения товара либо noname) либо конкретное значение появится позднее (дата окончания учебы, дата закрытия договора);

-элемент данных не может иметь значения в данном конкретном случае (например, ученую степень имеет смысл заполнять только для научных работников и преподавателей);

-известно, что атрибут имеет определенное значение, но оно не важно или было пропущено при вводе (пол сотрудника для библиотеки).

Каждый атрибут связан с набором значений, называемым доменом.

Домен определяет все потенциальные значения, которые могут быть присвоены атрибуту. Таким образом, **домен - это набор значений, который может быть присвоен атрибуту**. Например, **домен «Имена»** содержит все возможные имена людей, а **домен «Номера телефонов»** - все возможные комбинации цифр, соответствующих номерам телефонов. Для доменов могут быть приняты соглашения но написанию (формат), к примеру, для имен вначале должна быть фамилия, затем имя и отчество. Домены нельзя путать с атрибутами. **Атрибуты представляют собой различные интерпретации домена**. Например, атрибуты «Имя студента», «Имя преподавателя» берут свои значения из одного и того же домена «Имена».

В принципе, **домен - что не что иное, как тип данных** в расширенной интерпретации.

Некоторые **атрибуты** обладают важным для построения **информационной модели** свойством. Если известен **факт уникальности значения**, которое принимает некий элемент данных (значение атрибута) некоторого объекта, то можно **однозначно идентифицировать значения**, принимаемые другими элементами данных того же объекта. Например по **номеру и серии паспорта (идентификационному номеру)** можно установить остальные данные о человеке.

Атрибут, по которому можно однозначно идентифицировать остальные элементы данных, называется ключевым.

Ключевой атрибут (ключ) может состоять из **нескольких атрибутов**. Тогда его называют составным или сцепленным.

В общем случае атрибутов, по которым можно однозначно идентифицировать остальные элементы данных, может быть несколько. Тогда один из них выбирают в качестве **первичного ключа**, а остальные будут назваться **потенциальными или альтернативными ключами (серия и номер паспорта и идентификационный номер в организации)**.

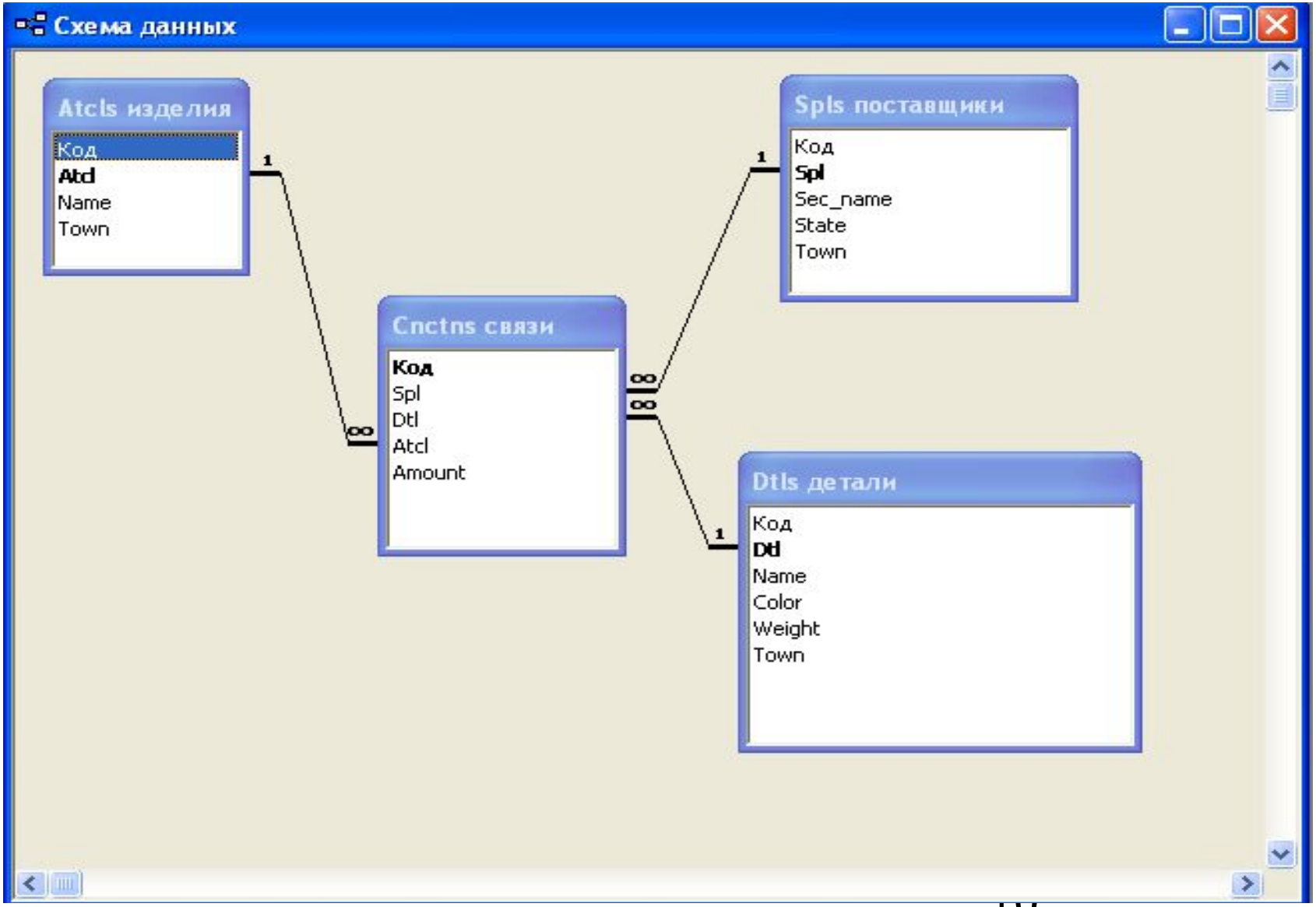
Элемент данных (значение атрибута) является минимальным фрагментом данных. В большинстве случаев он сам по себе информации не несет и приобретает смысл только в связи с другими элементами данных.

Совокупность значений связанных элементов данных называется **записью данных (в РБД - кортежем**). Некоторый объект может быть описан набором элементов данных, между которыми существует естественная связь (серия и номер паспорта). В дальнейшем такие элементы данных мы всегда, будем рассматривать в совокупности и такой вид связи явно выделяться не будет.

Соответствующая модель объектов с составляющими их элементами данных и взаимосвязями называется концептуальной моделью.

Поскольку такая модель позволяет наглядно отобразить структуру данных, то иначе ее называют схемой данных.

Схема данных



ТИПЫ СВЯЗЕЙ

В общем случае связи подразделяются **по количеству связываемых объектов**. Можно выделить **унарные, бинарные, тернарные** и связи более высоких порядков. **Унарная связь** (кроме рекурсивной) обычно смысла не имеет. Наибольший интерес представляют **бинарные связи**, так как связи более высоких порядков редко встречаются, труднореализуемы, нелегки для понимания и всегда могут быть разбиты на совокупности бинарных связей.

Первый тип связи – «один к одному»:
Эта связь означает, что одному экземпляру объекта А может соответствовать один и только один экземпляр связанного с ним объекта В, На практике такой вид связи встречается редко, так как почти всегда два таким образом связываемых объекта можно объединить в один. В качестве примера использования такой связи может послужить бухгалтерская БД с информацией о проводках (одна и та же операция оплаты идет в дебет и кредит).

Второй тип связи - «один ко многим»:
Это означает, что **одному экземпляру объекта А** соответствует **несколько (много) экземпляров объекта В**, Данный вид связи является наиболее распространенным. Например, к одной кафедре может относится множество студентов и преподавателей

Третий тип связи - «многие к одному»:
Это означает, что **нескольким экземплярам объекта А** соответствует **один экземпляр объекта В**. Связи «один ко многим» и «многие к одному» являются обратимыми, поэтому на практике обычно говорят о связи «один ко многим».

Четвертый тип связи - «многие ко многим»:
Это означает, что одному экземпляру объекта А соответствует несколько (много) экземпляров объекта В и, наоборот, каждому экземпляру объекта В может соответствовать несколько экземпляров объекта А. Например, сведения об приписке студентов к факультативным курсам. Один студент может посещать несколько курсов, и к каждому факультативному курсу приписано несколько студентов.

Наряду с взаимосвязями между объектами можно выделить зависимости между атрибутами объектов.

Зависимости бывают **функциональные, транзитивные и многозначные.**

Понятие **функциональной зависимости** является базовым, так как из его основе формулируются определения всех остальных видов зависимостей. **Атрибут В** (название кафедры) **функционально зависит** от **атрибута А** (имя студента), если **каждому** значению **атрибута А** соответствует **одно и только одно значение атрибута В**. То есть если нам известно значение атрибута А (его иначе называют детерминантом), то мы однозначно можем определить значение атрибута В. Однако данному значению атрибута В может соответствовать несколько различных значений атрибута А. Математически функциональная зависимость В от А обозначается записью А → В. Необходимо отметить, что А и В могут быть составными, т. е. состоять из двух и более атрибутов (например для университетской базы при совпадающих названиях кафедр).

Если же между атрибутами A и B существует **функциональная зависимость вида A → B и B → A** (т. е. между A и B имеется взаимно однозначное соответствие), то говорят о **функциональной взаимозависимости (обозначается как A ↔ B или B ↔ A).**

Наличие **функциональной взаимозависимости** между атрибутами A и B (например. хранение в БД наряду с номером зачетки студента номера его паспорта) приводит к образованию **связи «один к одному»** между этими атрибутами.

Функциональная зависимость может быть полной либо частичной.

Частичной зависимостью (частичной функциональной зависимостью) называется зависимость атрибута B (проживание студента в Минске) от части составного атрибута A (адрес студента).

Полная функциональная зависимость определяется зависимостью атрибута B от всего составного атрибута A. (кафедра и имя студента)

Атрибут С зависит от **атрибута А транзитивно** (существует транзитивная зависимость), если для **атрибутов А, В, С** выполняются условия **А → В и В → С**, **но обратная зависимость отсутствует**. Например, ФИО → Ср. балл → Стипендия.

Атрибуты могут быть и **независимыми**. **Два или более атрибута будут взаимно независимыми, если ни один из этих атрибутов не является функционально зависимым от других**.

Проектирование концептуальной модели можно провести на основе анализа существующих зависимостей между атрибутами.

Все атрибуты, характеризующиеся **полной функциональной зависимостью от ключевого атрибута**, будут **атрибутами объекта одного типа**.

Те **атрибуты**, которые характеризуются **транзитивной зависимостью** от **ключевого атрибута**, необходимо выделить и сформировать **отдельные объекты (студент → кафедра → спецкурс)**.

Многозначные зависимости также требуют **дополнительного разбиения по объектам**. Практически процесс формирования объектов на основе анализа зависимостей необходимо продолжать до тех пор, пока все **неключевые атрибуты** будут характеризоваться только **полной функциональной зависимостью от соответствующих ключевых атрибутов**. На первом же шаге проектирования можно значительно сократить множество анализируемых атрибутов, исключив **тривиальные** зависимости, т. е. те зависимости, которые не могут не выполняться. Атрибуты, связанные **тривиальной** зависимостью, необходимо всегда рассматривать **вместе (номер и серия паспорта)**.

Существует большое количество видов концептуальных моделей, позволяющих отобразить семантику данных.

Критериями выбора модели могут служить следующие характеристики:

• выразительность. Модель должна содержать достаточно средств для выражения типов, атрибутов, связей и ограничений;

• формализованность;

• простота и легкость восприятия. Модель должна восприниматься конечными пользователями;

• минимальность. Модель должна обладать минимальным (но достаточным) набором не перекрывающихся по смыслу концепций;

• представительность. Модель должна обладать выразительной диаграммной техникой представления.

Всеми этими характеристиками обладает **модель сущность–связь (ER),** специально разработанная для концептуального моделирования реляционных БД. Большим достоинством ее также является возможность автоматизации процесса проектирования.

Модель сущность - связь (entity-relationship) представляет собой высокоуровневую концептуальную модель данных, созданную Питером Ченом (P. Chen) в 1976 г. в целях упрощения задачи проектирования БД.
Основным объектом в ER модели является сущность, которая представляет собой **реальную вещь** и может быть **реально существующим (физическое существование) или абстрактным (концептуальное существование) объектом**.
Каждая **сущность** имеет **набор атрибутов**, представляющих ее свойства. **Сущность имеет имя - тип**, который представляет **совокупность сущностей с одинаковым набором атрибутов**.

Таким образом, некоторая **сущность** определяется **типом (именем)** и **набором атрибутов**. В свою очередь некоторый **экземпляр сущности** будет характеризоваться **набором значений атрибутов**.

Различаются **сильные и слабые типы сущностей.**
Слабый тип сущности определяется как тип, существование которого зависит от какого-то другого тина сущности, а **сильный** тип сущности - как тип, существование которого не зависит от всех других типов сущностей.

Слабые сущности могут быть **дочерними**, **подчиненными** и **зависимыми**, а **сильные** - **родительскими**, **владельцами** или **доминантными**.

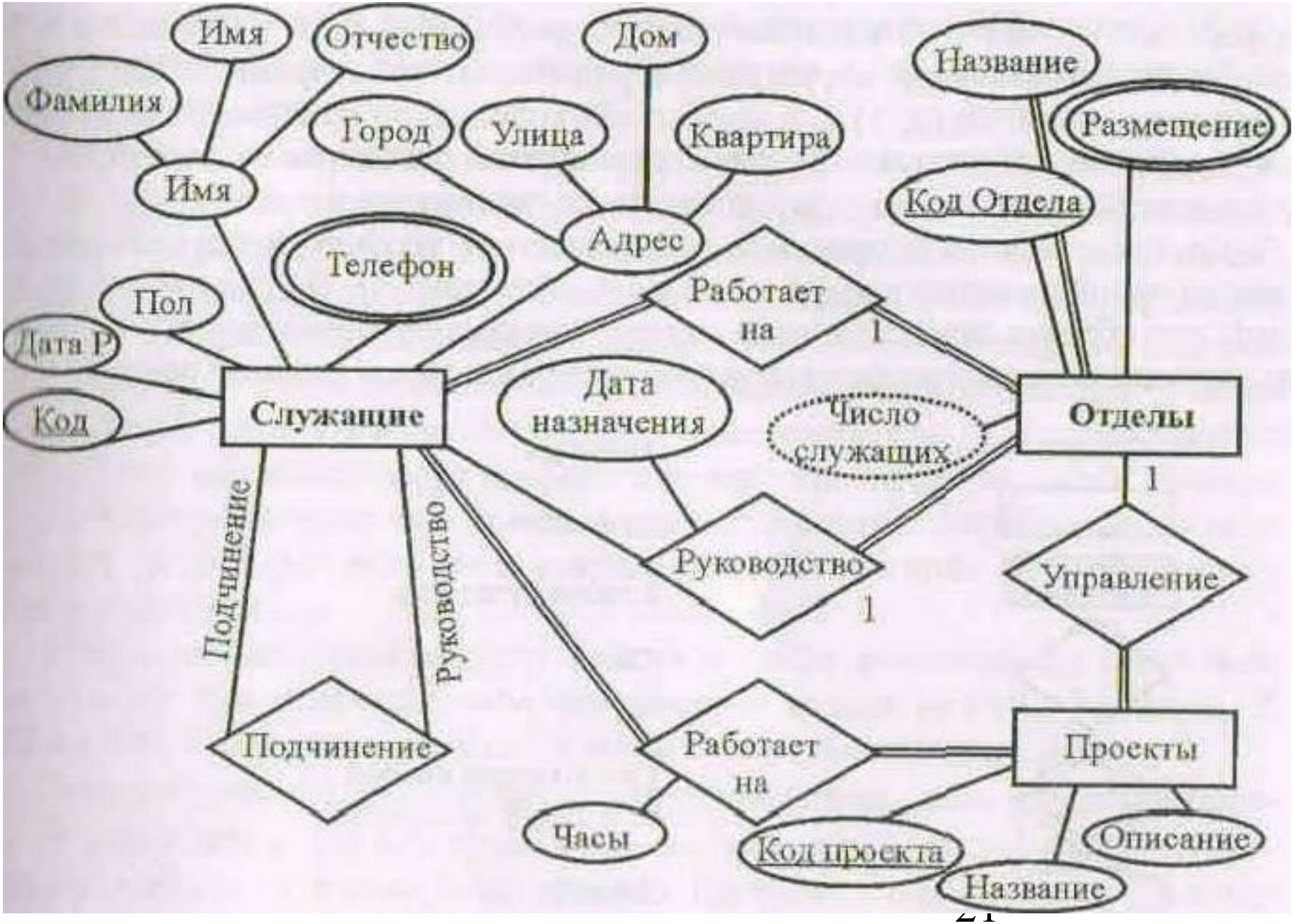
Между типами **сущностей** могут существовать связи. В общем случае связи могут быть унарными (рекурсивная), бинарными, тернарными и быть более высокого порядка (**степень связи**).

Также и для сущностей, следует различать **связи и типы связей**.

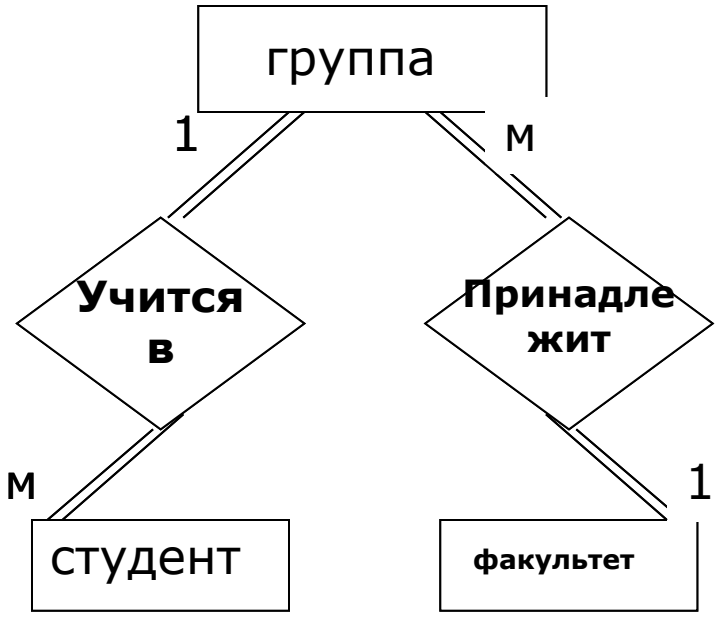
В свою очередь, **атрибуты** также могут иметь **связи**. На **связи** накладываются **ограничения по степени участия и по показателю кардинальности**. **Показатель кардинальности** соответствует **типу связи** в ее обычной формулировке, т. е, 1 : 1, 1 : M, M : 1, M : N.

Если **ER модель** составляется для некоторого **предприятия**, то **показатели кардинальности** прежде всего будут определяться **производственными правилами (бизнес-правилами),** установленными на данном предприятии.

Термин **«бизнес-правило»** получил широкое распространение, поскольку при моделировании любой предметной области важнейшей задачей является выделение и учет всех без исключения **отношений между объектами** (если, конечно, это позволяет сама модель). **Связям** могут также присваиваться **ролевые имена** для **однозначного определения назначения каждой связи**. Если назначение каждой связи определено недвусмысленно, то ролевые имена не указываются. **Степень участия** может быть **полной и частичной**. **Полной** она будет в том случае, если для существования объекта, участвующего в связи, требуется существование другого объекта (иначе - **зависимость существования**). Например, если правила предприятия требуют, чтобы любой служащий работал в некотором отделе, экземпляр сущности «Служащий» будет существовать, только если он участвует в связи «Работает на» с сущностью «Отдел». В то же время только некоторые служащие могут руководить отделами. Значит, участие сущности «Служащий» в связи «Руководство» будет только **частичным**.

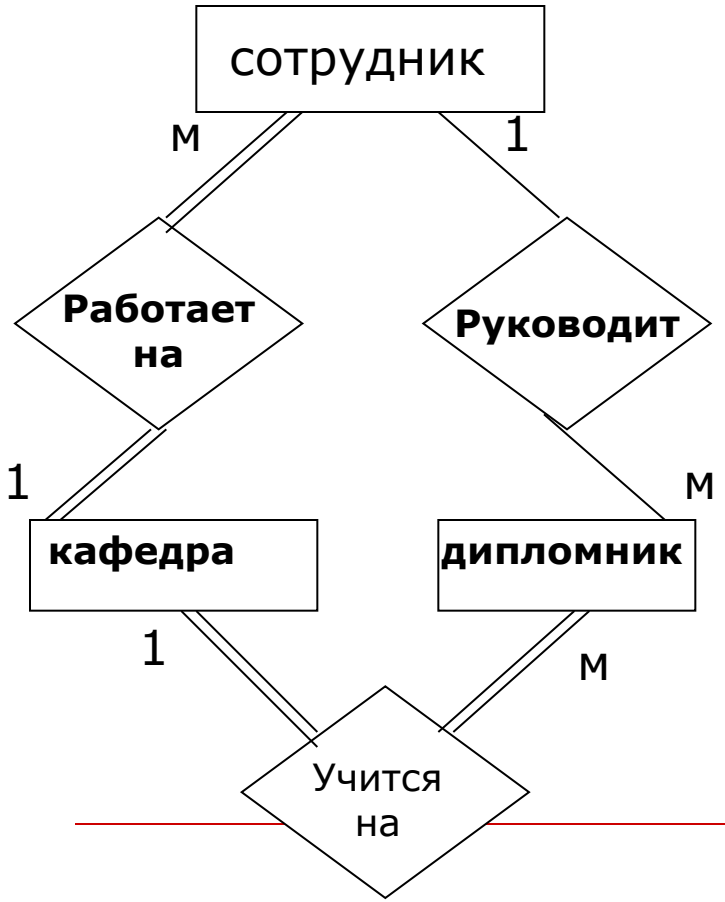

Имя
Отчество
Дом
Название
Фамилия
Размещение
Город
Улица
Квартира
Имя
Код Отдела
Адрес
Телефон
Работает на
1
Пол
Дата Р
Служащие
Дата назначения
Число служащих
Отделы
Код
1
Подчинение
Руководство
Руководство
Управление
1
Подчинение
Работает на
Проекты
Часы
Код проекта
Описание
Название

К **недостаткам** этой модели можно отнести возможность появления «**ловушек**» при недостаточном определении связей между объектами. К таким ловушкам можно отнести **ловушки разветвления и разрыва.**



Ловушка разветвления имеет место в том случае, если модель отображает **связи между типами сущностей**, но **путь** между отдельными сущностями этих типов указан **неоднозначно**. Например, ловушка разветвления может возникать, когда **две и более связи «один ко многим» разветвляются из одной сущности**. В данном примере схема данных не позволяет однозначно ответить на вопрос к каким группам относятся студенты. Устранить такую ловушку можно с помощью реорганизации связей между типами сущностей.

Ловушка разрыва появляется в том случае, когда в модели предполагается наличие связи между типами сущностей, но не существует пути между отдельными сущностями этих типов.



Ловушка разрыва может возникать между тремя и более сущностями, связанными между собой отношениями «один ко многим» при наличии **связи с частичным участием.**

Из схемы видно, что на кафедре работает несколько сотрудников и за ними закреплены определенные дипломники. Но не за каждым сотрудником должен быть закреплен дипломник и не все дипломники должны быть закреплены за сотрудниками в данный момент времени. Такая схема не позволяет ответить на вопрос, какие дипломники числятся за кафедрой. Такая ловушка устраняется введением дополнительной связи между типами сущностей.

МОДЕЛИ ДАННЫХ

Хранимые в базе данные имеют определенную логическую структуру, т.е. описываются некоторой моделью представления данных (в дальнейшем просто моделью данных).

К числу классических моделей данных относятся базы данных, **построенных на инвертированных списках, иерархическая, сетевая и реляционная.**

В последние годы появились и стали внедряться на практике **постреляционная, многомерная и объектно-ориентированная модели**.

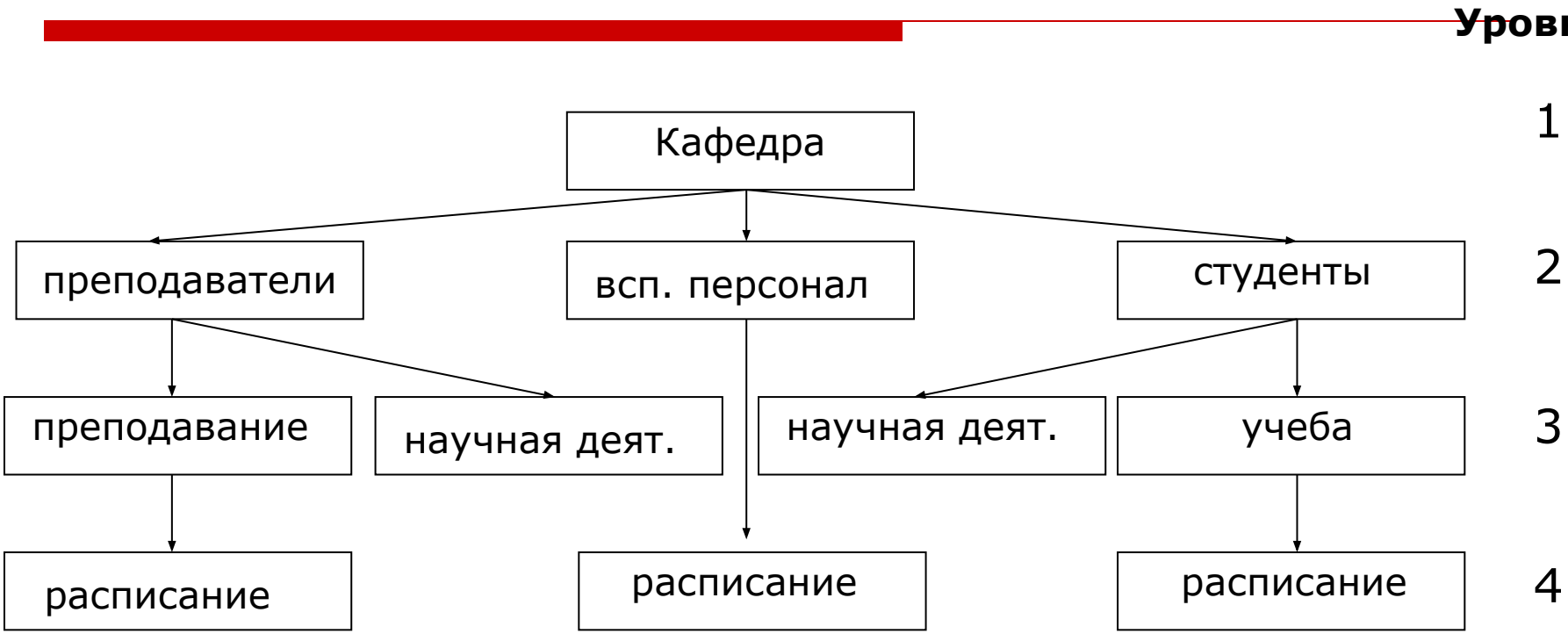
Разрабатываются также всевозможные системы, основанные на моделях данных, расширяющих известные модели, например **объектно-реляционная**.

ИЕРАРХИЧЕСКАЯ МОДЕЛЬ

Метрически первой была разработана и воплощена иерархическая модель данных. В 1960 г. разработана БД для большой ЭВМ IBM system 360. первой коммерческой СУБД стала СУБД IMS фирмы IBM. Среди отечественных можно назвать СУБД ОКА.

Иерархическая модель данных представляется связным **графом типа дерева**, вершины (типы) которого расположены на разных иерархических уровнях. При этом **одна вершина, расположенная на самом верху дерева**, называется **корнем** и не подчиняется ни одной вершине, а **все остальные связаны с одной и только одной вершиной, расположенной на более высоком уровне**. **Элементы**, расположенные в конце ветви, т. с. **не имеющие порожденных**, называются **листьями**. **Потомков** одного и того же типа называют **близнецами**. Элементы могут быть простыми, а также могут представлять собой записи. **Иерархическая БД** представляет собой упорядоченную **совокупность экземпляров типа «дерево»,** содержащих в свою очередь **совокупность экземпляров типа «запись».** Обход всех элементов производится сверху вниз и слева направо.

ИЕРАРХИЧЕСКАЯ МОДЕЛЬ ДАННЫХ



Иерархической модели присущи связи **«один к одному» и «один ко многим».** Связь **«многие ко многим» не может быть реализована**, так как вершина не может иметь более одного родителя. Видно, что иерархической модели данных присуще дублирование информации. К достоинствам иерархической модели данных относятся эффективность использования памяти ЭВМ и неплохие показатели времени выполнения основных операций с данными.

К недостаткам можно отнести:

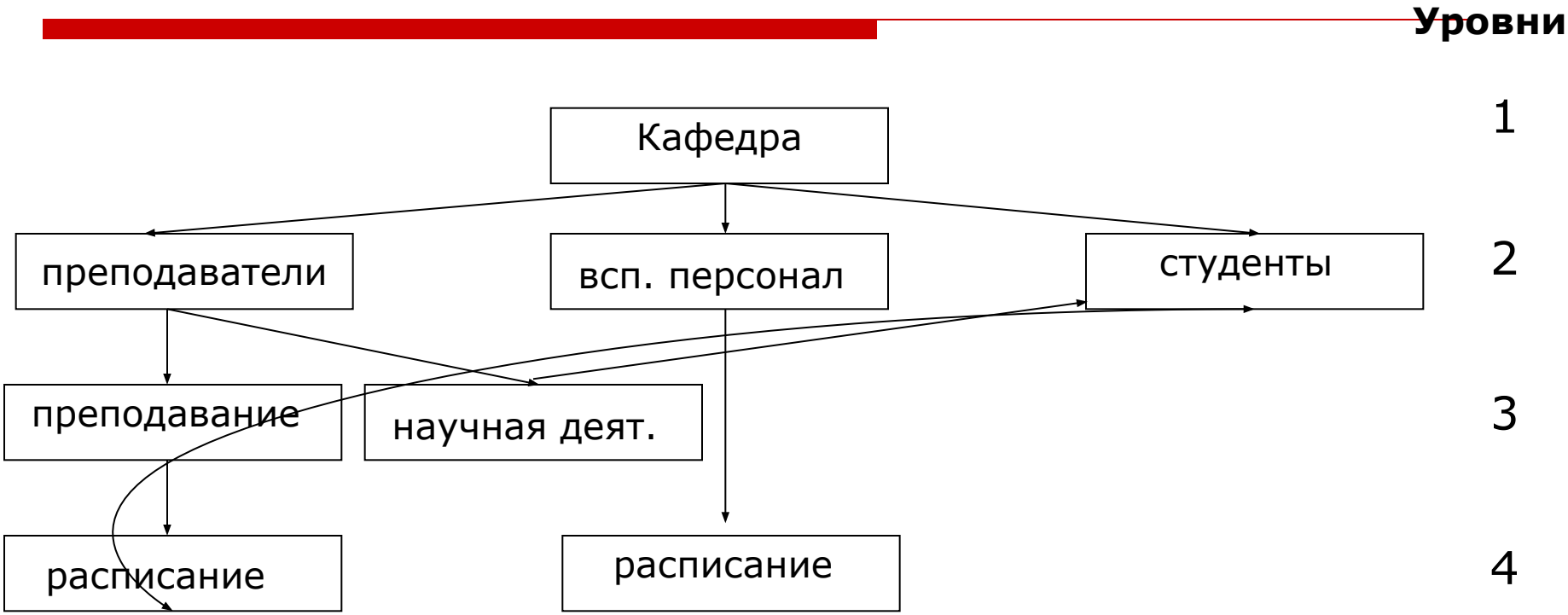
- невозможность установления связи М : М;
- сильную логическую зависимость данных. Добавление или удаление вершины невозможно без разрушения всей структуры данных;
- избыточность информации;
- затрудненный поиск «снизу вверх». Это приводит к простому перебору всех значений в БД.

СЕТЕВАЯ МОДЕЛЬ ДАННЫХ

Впервые сетевые СУБД появились в 1970 г. с появлением миникомпьютеров DEC. Это IDMS, VAX_DMBS, db_Vistall и наша СЕТЬ.

Если в отношении между данными порожденный элемент имеет более одного исходного элемента, то такое отношение уже нельзя описать как древовидное. Его описывают а виде **сетевой структуры**. Сетевая модель данных позволяет отображать взаимосвязи элементов данных в виде **произвольного графа**, обобщая тем самым иерархическую модель данных В **сетевой** модели данные представлены в виде **записей и связей**. Запись может иметь множество как подчиненных ей записей, так и связей, которым она подчинена. Такая сеть позволяет реализовывать отношение М : М на всех связях, С некоторой избыточностью сетевые модели всегда можно разложить на несколько иерархических, вводя дополнительные вершины. Сетевая модель данных существенно уменьшает дублирование информации. К другим достоинствам этой модели данных можно отнести большую гибкость системы. В ней можно использовать не только предопределенные связи , но и добавлять новые (наряду с узлами) в процессе работы. Разрешены также симметричные запросы (сверху вниз и снизу вверх), выполняющиеся по схожим алгоритмам.

СЕТЕВАЯ МОДЕЛЬ ДАННЫХ



К **недостаткам сетевой модели** данных можно отнести;
• недостаточную скорость выполнения поиска;
• некоторую избыточность информации (для каждой записи дублируются на подчиненные и родительские элементы);
• громоздкое и труднообозримое представление данных;
• ослабления контроля целостности вследствие допустимости установления произвольных связей между записями.

Общие недостатки:
-Слишком сложно пользоваться;

-Фактически необходимы знания о физической организации;

-Прикладные системы зависят от этой организации;

-Их логика перегружена деталями организации доступа к БД.

ОБЪЕКТНО-ОРИЕНТИРОВАННАЯ МОДЕЛЬ ДАННЫХ

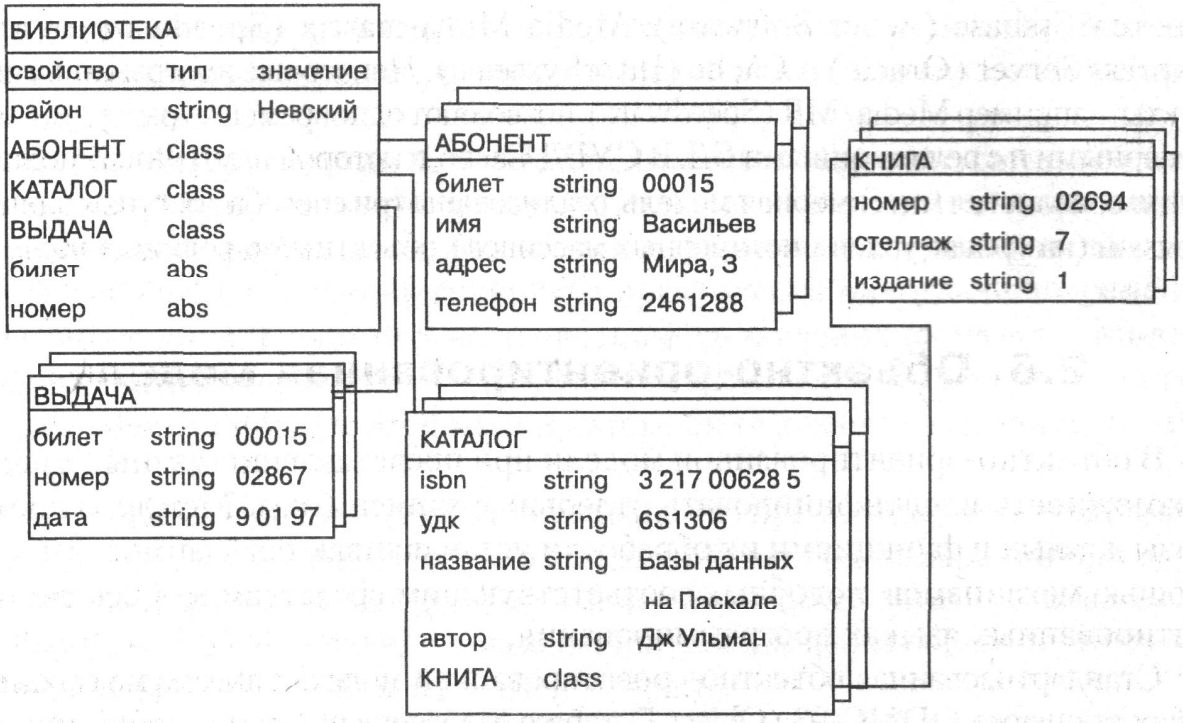
Стандарт ODMG-93 (Object Database Management Group)

Структура объектно-ориентированной БД графически представима в виде **дерева, узлами которого являются объекты**.
Свойства объектов описываются некоторым стандартным типом (например, строковым —string) или типом, конструируемым пользователем (определяется как **class**).

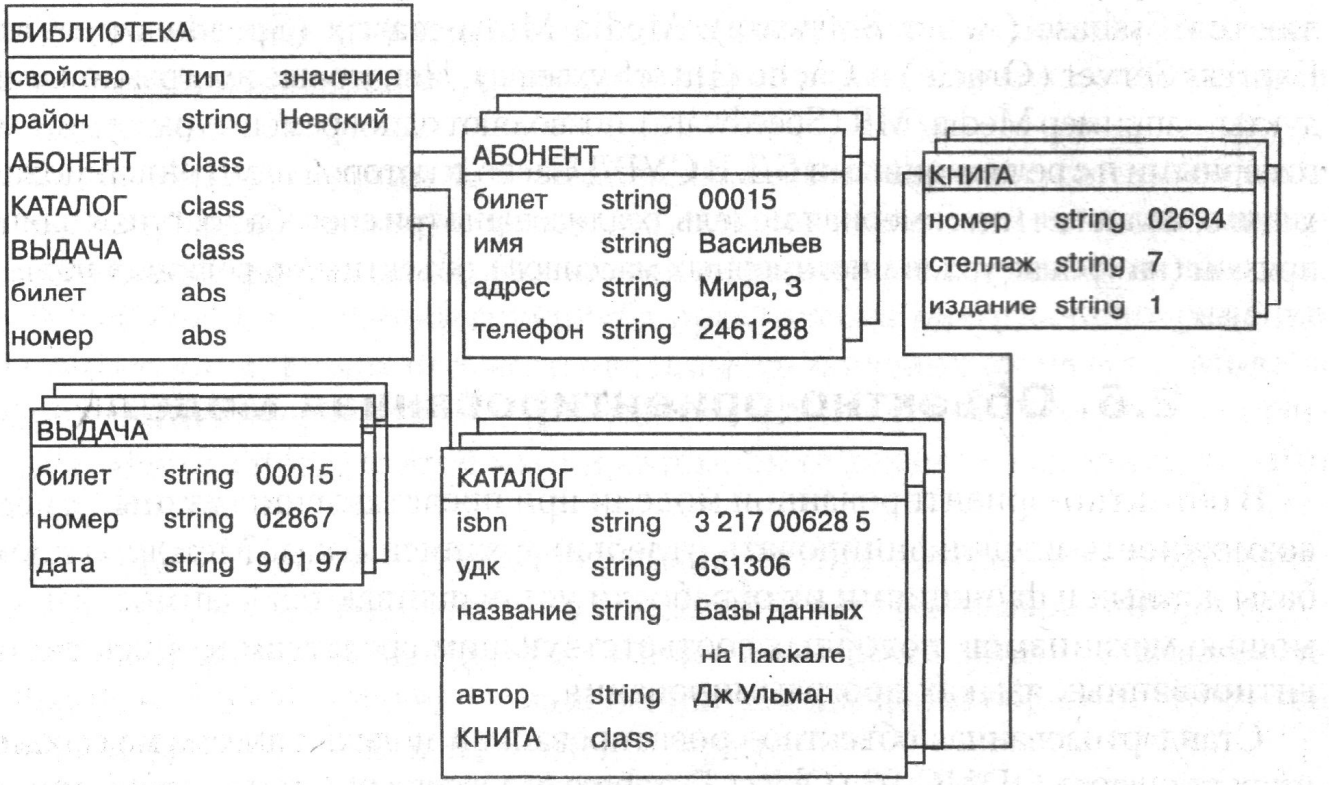
Значением свойства типа string является строка символов.
Значение свойства типа class есть объект, являющийся экземпляром соответствующего класса.

Каждый объект-экземпляр класса считается потомком объекта, в котором он определен как свойство. **Объект-экземпляр класса принадлежит своему классу и имеет одного родителя. Родовые отношения в БД образуют связную иерархию объектов.**

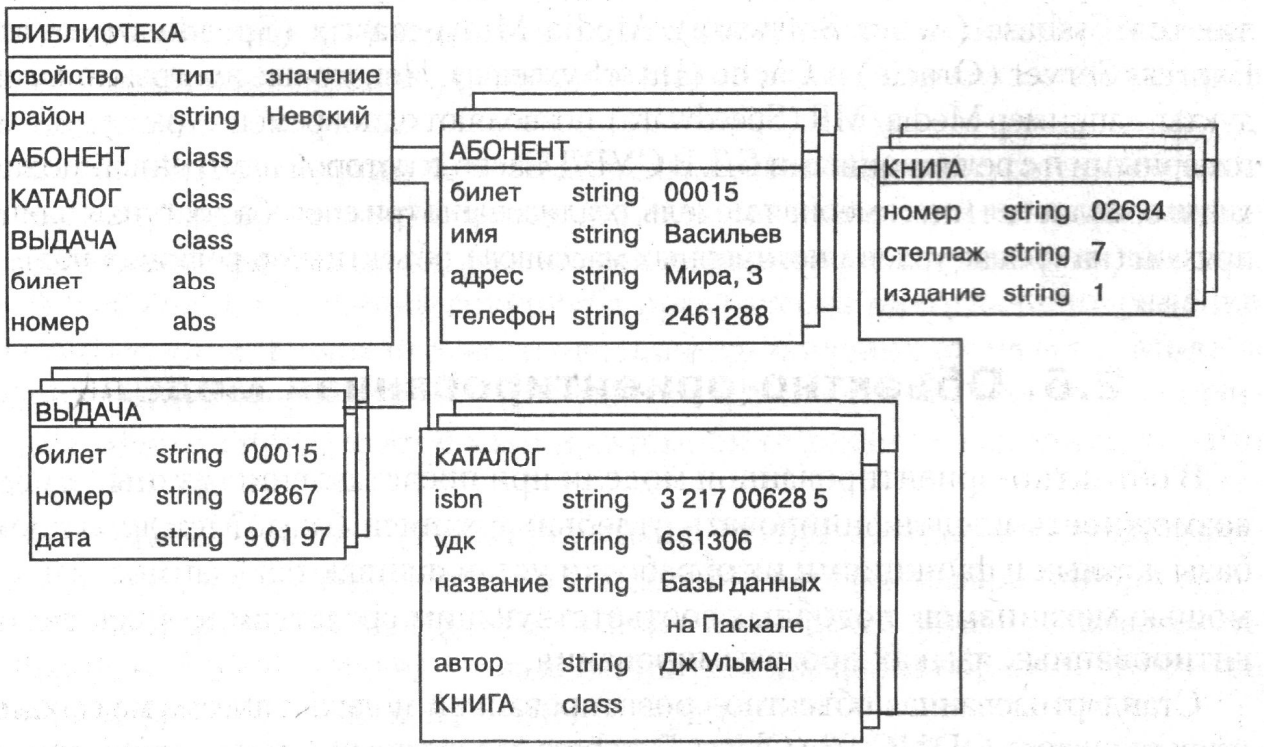
Здесь объект типа **БИБЛИОТЕКА** является родительским для объектов-экземпляров классов **АБОНЕНТ, КАТАЛОГИ, ВЫДАЧА**. Различные объекты типа **КНИГА** могут иметь одного или разных родителей. Объекты типа **КНИГА**, имеющие одного и того же родителя, должны различаться по крайней мере инвентарным номером (уникален для каждого экземпляра книги), но имеют одинаковые значения свойств *isbn, удк, название* и *автор.*
Логическая структура объектно-ориентированной БД внешне похожа на структуру **иерархической БД**. Основное отличие между ними состоит в методах манипулирования данными.



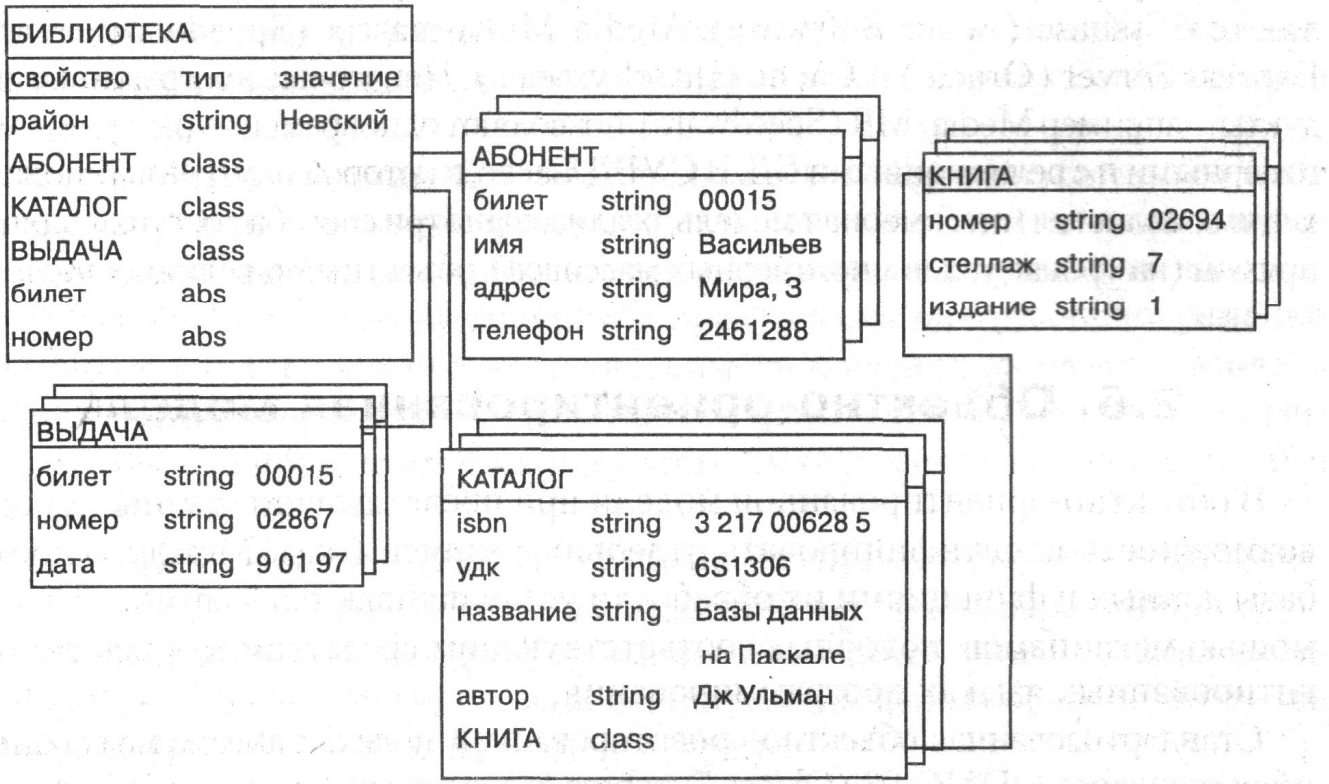
Инкапсуляция ограничивает область видимости имени свойства пределами того объекта, в котором оно определено. Так, если в объект типа КАТАЛОГ добавить свойство, задающее телефон автора книги и имеющее название *телефон,* то мы получим одноименные свойства у объектов АБОНЕНТ и КАТАЛОГ. Смысл такого свойства будет определяться тем объектом, в который оно инкапсулировано.



Наследование, наоборот, распространяет область видимости свойства на всех потомков объекта. Так, всем объектам типа КНИГА, являющимся потомками объекта типа КАТАЛОГ, можно приписать свойства объекта-родителя: *isbn, удк, название* и *автор.* Если необходимо расширить действие механизма наследования на объекты, не являющиеся непосредственными родственниками (например, между двумя потомками одного родителя), то в их общем предке определяется абстрактное свойство типа *abs*.



Полиморфизм в объектно-ориентированных языках программирования означает способность одного и того же программного кода работать с разнотипными данными. Другими словами, он означает допустимость в объектах разных типов иметь методы (процедуры или функции) с одинаковыми именами. Во время выполнения объектной программы одни и те же методы оперируют с разными объектами в зависимости от типа аргумента. Применительно к нашей объектно-ориентированной БД полиморфизм означает, что объекты класса КНИГА, имеющие разных родителей из класса КАТАЛОГ, могут иметь разный набор свойств. Следовательно, программы работы с объектами класса КНИГА могут содержать полиморфный код.



Основным ***достоинством*** объектно-ориентированной модели данных в сравнении с реляционной является возможность отображения информации о сложных взаимосвязях объектов.

Объектно-ориентированная модель данных позволяет **идентифицировать отдельную запись базы данных и определять функции их обработки.**

Недостатками объектно-ориентированной модели являются высокая понятийная сложность, неудобство обработки данных и низкая скорость выполнения запросов.

G-Base (Grapael), GemStone (Servio-Logic and OGI), Statice (Symbolics), ObjectStore (Object Design), Objectivity /DB (Objectivity), Versant (Versant Technologies), O2 (Ardent Software), ODB-Jupiter, Iris, Orion and Postgres.