Теория баз данных

Лекция 6. Модель данных «сущность–связь»

Е. П. Моргунов
Сибирский федеральный университет
г. Красноярск
Институт космических и информационных технологий
emorgunov@mail.ru

Версия от 29.03.2018

6.1. Введение

- Чтобы добиться полного понимания характера данных и способов их использования в организации, необходимо применять в процессе обмена информацией между специалистами общую модель, которая не усложнена техническими подробностями и не допускает двойных толкований.
- Структура (схема) БД может быть описана с использованием различных языков и нотаций.
- Наиболее распространенным средством абстрактного представления структур баз данных является ER-модель, или модель «сущность–связь» (entity–relationship model).
- ER-моделирование является представителем так называемого семантического моделирования.
- ER-модель была предложена Питером Ченом в 1976 г.
- Существуют и другие методы семантического моделирования.

6.1. Введение (продолжение)

- С ER-моделью связана и соответствующая технология построения диаграмм сущностей и связей, которые называются ER-диаграммами. В принципе, возможно использование и других нотаций, даже текстового описания.
- ER-моделирование представляет собой нисходящий подход к проектированию базы данных, который начинается с выявления наиболее важных данных, называемых сущностями (entities), и связей (relationships) между данными, которые должны быть представлены в модели. Затем в модель вносятся дополнительные сведения, например, указывается информация о сущностях и связях, называемая *атрибутами* (attributes), а также все ограничения, относящиеся к сущностям, связям и атрибутам.
- Основные концепции ER-модели:
  - Типы сущностей
  - Типы связей
  - Свойства (атрибуты)

6.2. Типы сущностей

- **Тип сущности (entity type)** – группа объектов с одинаковыми свойствами, которая рассматривается в конкретной предметной области как имеющая независимое существование.
- Сущности, выявленные (идентифицированные) в одной и той же предметной области разными разработчиками, могут различаться.
- **Физическое существование:** Работник, Объект недвижимости, Клиент, Деталь, Поставщик, Изделие.
- **Концептуальное существование:** Осмотр объекта недвижимости, Продажа объекта недвижимости, Рабочий стаж.
- **Экземпляр сущности (Entity occurrence)**– однозначно идентифицируемый объект, который относится к сущности определенного типа.
- Если это не приводит к искажению смысла, то вместо терминов *«тип сущности»* и *«экземпляр сущности»* используется более общий термин «*сущность».*

6.2. Типы сущностей (продолжение)

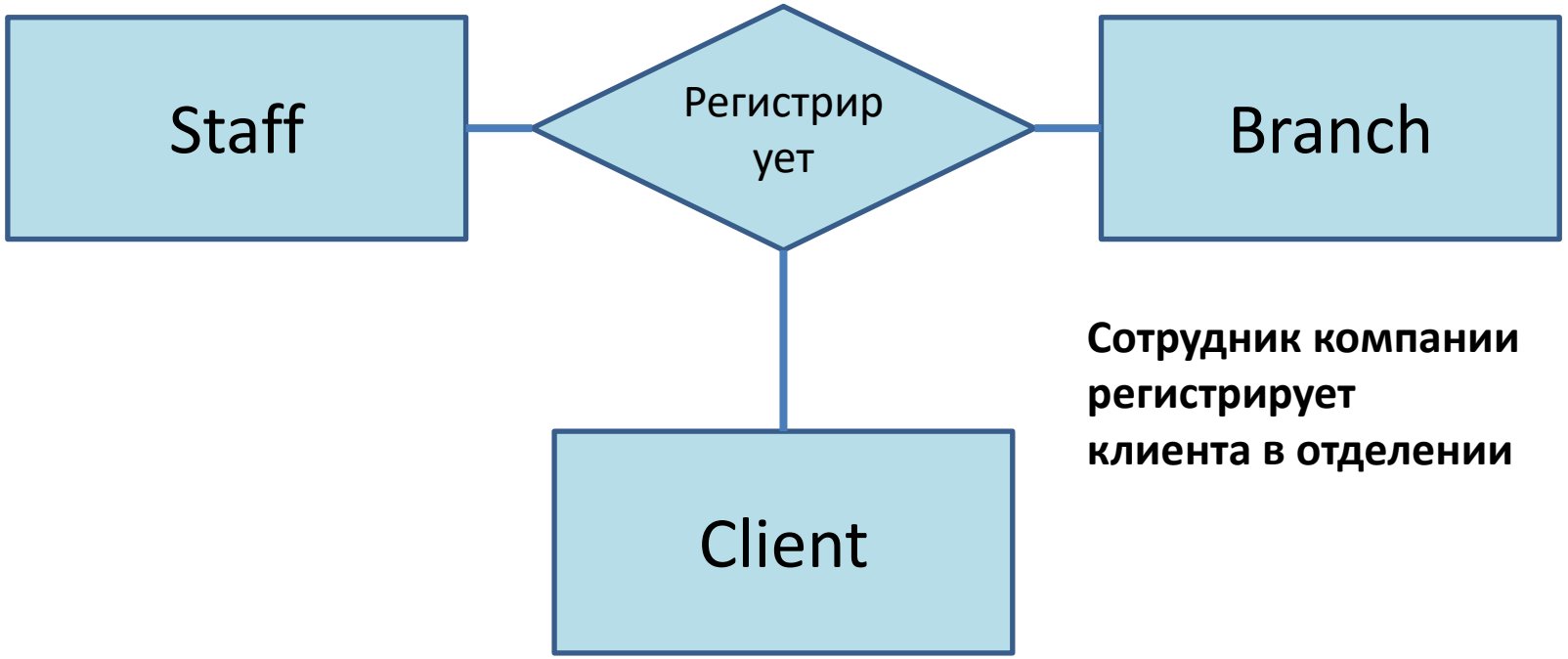
- Каждый тип сущности изображается в виде прямоугольника с именем сущности внутри него.
- В качестве имени обычно применяется существительное в единственном числе.

6.3. Типы связей

- **Тип связей (relationship type)** – набор значимых ассоциаций между разными типами сущностей.
- **Экземпляр связи (relationship occurrence)** – однозначно идентифицируемая ассоциация, которая включает по одному экземпляру сущности из каждого участвующего в связи типа сущности.
- *Экземпляр связи* обозначает все конкретные экземпляры сущностей, участвующие в этой связи.
- Если это не приводит к искажению смысла, то вместо терминов *«тип связи»* и *«экземпляр связи»* используется более общий термин *«связь».*
- Имя связи должно являться глаголом. По возможности в каждой конкретной ER-модели все имена связей должны быть уникальными.
- Связи должны иметь *направление*.

6.3.1. Степень типа связи

- **Степень типа связи (Degree of a relationship type)** – количество типов сущностей, которые охвачены данной связью.
- Если в связи участвует два типа сущностей, то связь называется бинарной, если три типа сущностей – трехсторонней (тернарной).
- Для описания связей со степенью больше двух принято применять термин *сложная связь.*

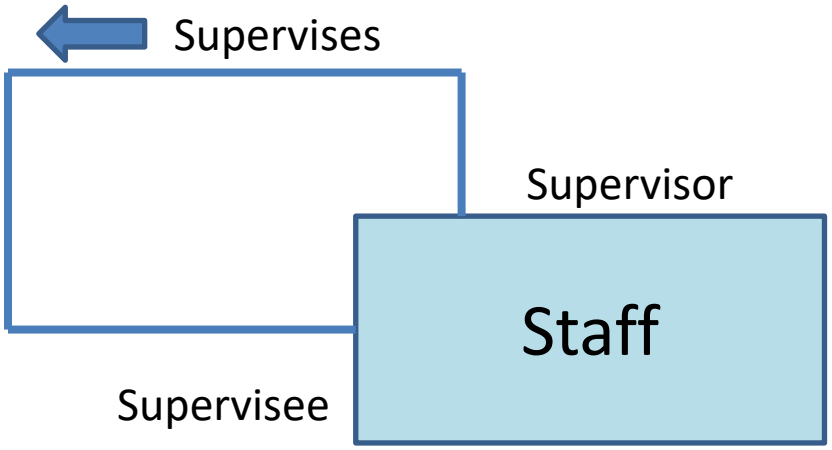


Сотрудник компании регистрирует клиента в отделении

6.3.2. Рекурсивная связь

- **Рекурсивная связь** – связь, в которой *один и тот же тип* сущности участвует более одного раза в *разных ролях.*
- Связям могут присваиваться *ролевые имена* для указания назначения каждой сущности, участвующей в данной связи.

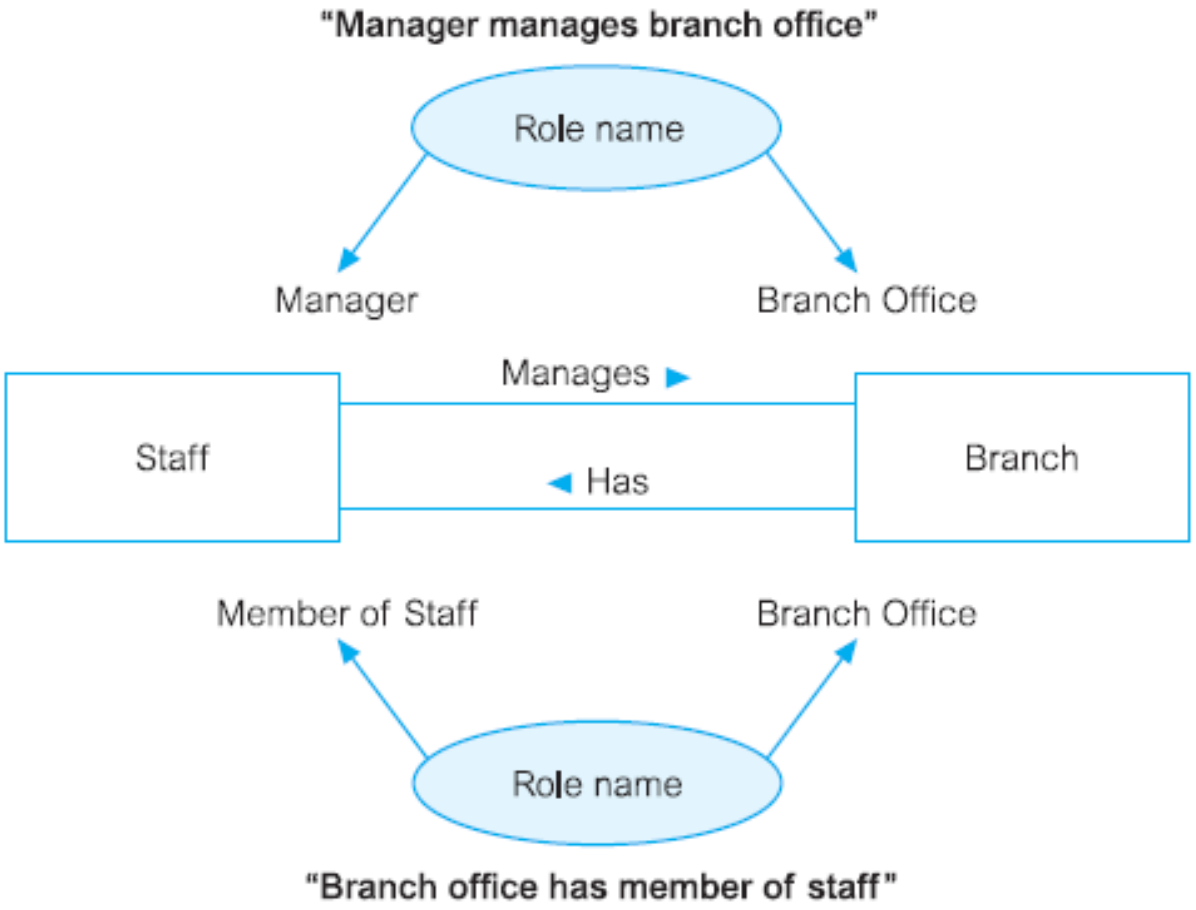
Сотрудник компании (Supervisor) руководит сотрудником компании (Supervisee)



Первый участник связи Supervises с типом сущности Staff получил имя роли Supervisor (Инспектор), а второй получил имя роли — Supervisee (Контролируемый сотрудник).

6.3.2. Рекурсивная связь (продолжение)

- Ролевые имена могут также использоваться, когда две сущности связаны несколькими связями.



6.4. Атрибуты

- **Атрибут (Attribute)** – свойство типа сущности или типа связи.
- Например, сущность staff (Персонал) может быть описана с помощью атрибутов staffNo (Табельный номер работника), name (Имя), position (Должность) и salary (Зарплата).
- Атрибуты содержат значения, которые описывают каждый экземпляр сущности и составляют основную часть информации, сохраняемой в базе данных.
- Связь, которая соединяет сущности, также может иметь атрибуты, аналогичные атрибутам типа сущности.

6.4. Атрибуты (продолжение)

- **Домен атрибута (Attribute domain)** – набор допустимых значений одного или нескольких атрибутов.
- Например, количество комнат в объекте недвижимости может находиться в пределах от 1 до 15 для каждого экземпляра сущности. Следовательно, набор допустимых значений для атрибута rooms (Количество комнат) сущности PropertyForRent можно определить как набор целых чисел от 1 до 15.
- Различные атрибуты могут совместно использовать один и тот же домен.
- Например, атрибуты address (Адрес) типов сущностей Branch (Отделение компании), PrivateOwner (Владелец объекта недвижимости) и BusinessOwner (Владелец делового предприятия) разделяют один и тот же домен, который включает все возможные адреса.
- Атрибуты подразделяются на *простые* и *составные, однозначные* и *многозначные,* а также *производные.*

6.4.1. Простые и составные атрибуты

- **Простой атрибут (Simple attribute)** – атрибут, состоящий из одного компонента с независимым существованием.
- *Простые* атрибуты не могут быть разделены на более мелкие компоненты. Примером простых атрибутов является атрибут position (Должность) или salary (Зарплата) сущности Staff. Простые атрибуты иногда называют *элементарными.*
- **Составной атрибут** (**Composite attribute**) – атрибут, состоящий из нескольких компонентов, каждый из которых характеризуется независимым существованием.
- Некоторые атрибуты могут быть разделены на более мелкие компоненты, которые характеризуются независимым существованием. Например, атрибут address (Адрес) сущности Branch, представляющей отделение компании, со значением '163 Main St, Glasgow, Gil 9QX1 может быть разбит на отдельные атрибуты street ('163 Main St'), city ('Glasgow1) и postcode ('Gil 9QX1).
- Решение о моделировании атрибута address в виде простого атрибута или разбиении его на атрибуты street, city и postcode зависит от того, как рассматривается атрибут address в пользовательском представлении — как единое целое или как набор отдельных компонентов.

6.4.2. Однозначные и многозначные атрибуты

- **Однозначный атрибут (Single-valued attribute)** – атрибут, который содержит одно значение для каждого экземпляра сущности определенного типа.
- Большинство атрибутов являются однозначными. Например, для каждого отдельного экземпляра сущности Branch всегда имеется единственное значение в атрибуте номера отделения компании (branchNo), скажем, 'B003'.
- **Многозначный атрибут (Multi-valued attribute)** – атрибут, который содержит несколько значений для каждого экземпляра сущности определенного типа.
- Например, сущность Branch может иметь несколько значений для атрибута telNo (Номер телефона отделения компании). Следовательно, атрибут telNo в этом случае будет многозначным.
- Многозначный атрибут допускает присутствие определенного количества значений (возможно, в заданных пределах, определяющих максимальное и минимальное количество). Например, атрибут telNo отделения компании может иметь от одного до трех значений.

6.4.3. Производные атрибуты

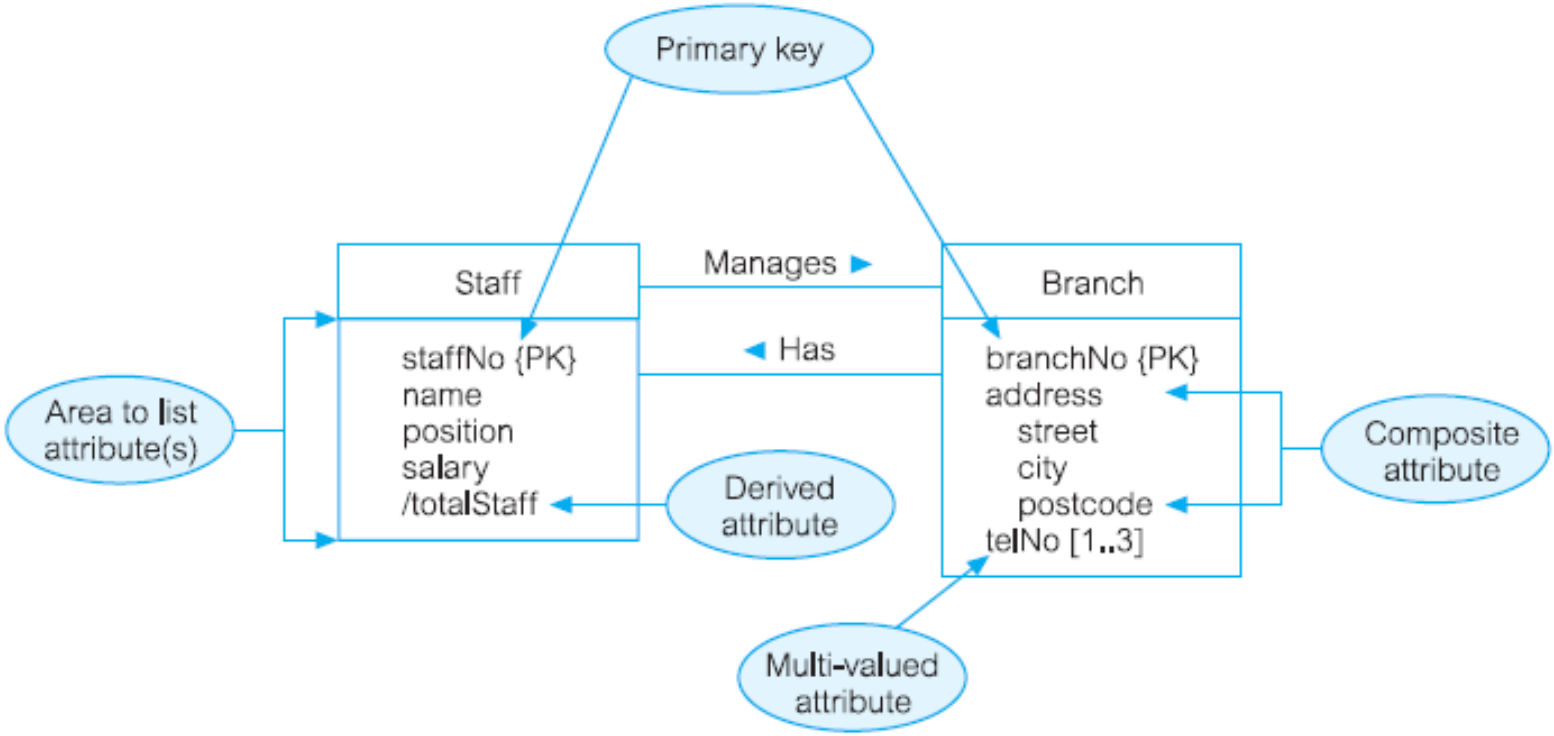
- **Производный** атрибут (**Derived attribute**) – атрибут, который представляет значение, производное от значения связанного с ним атрибута или некоторого множества атрибутов, принадлежащих некоторому (не обязательно данному) типу сущности.
- Например, значение атрибута duration (Срок действия) сущности Lease (Договор аренды) вычисляется на основе атрибутов rentstart (Начало срока аренды) и rentFinish (Конец срока аренды), которые также относятся к типу сущности Lease.
- В некоторых случаях значение атрибута является производным от многих экземпляров сущности одного и того же типа. Например, атрибут totalStaff (Общее количество сотрудников) сущности типа staff может быть вычислен на основе подсчета общего количества экземпляров сущности staff .

6.4.4. Ключи

- **Потенциальный ключ** (**Candidate key**) – атрибут или минимальный набор атрибутов, который однозначно идентифицирует каждый экземпляр типа сущности.
- Потенциальный ключ не может содержать значения NULL.
- **Первичный ключ (Primary key)** – потенциальный ключ, который выбран для однозначной идентификации каждого экземпляра сущности определенного типа.
- Выбор первичного ключа сущности осуществляется с учетом суммарной длины атрибутов, минимального количества необходимых атрибутов в ключе, а также наличия гарантий уникальности его значений в текущий момент времени и в обозримом будущем.
- **Составной ключ (Composite key)** – потенциальный ключ, который состоит из двух или нескольких атрибутов.

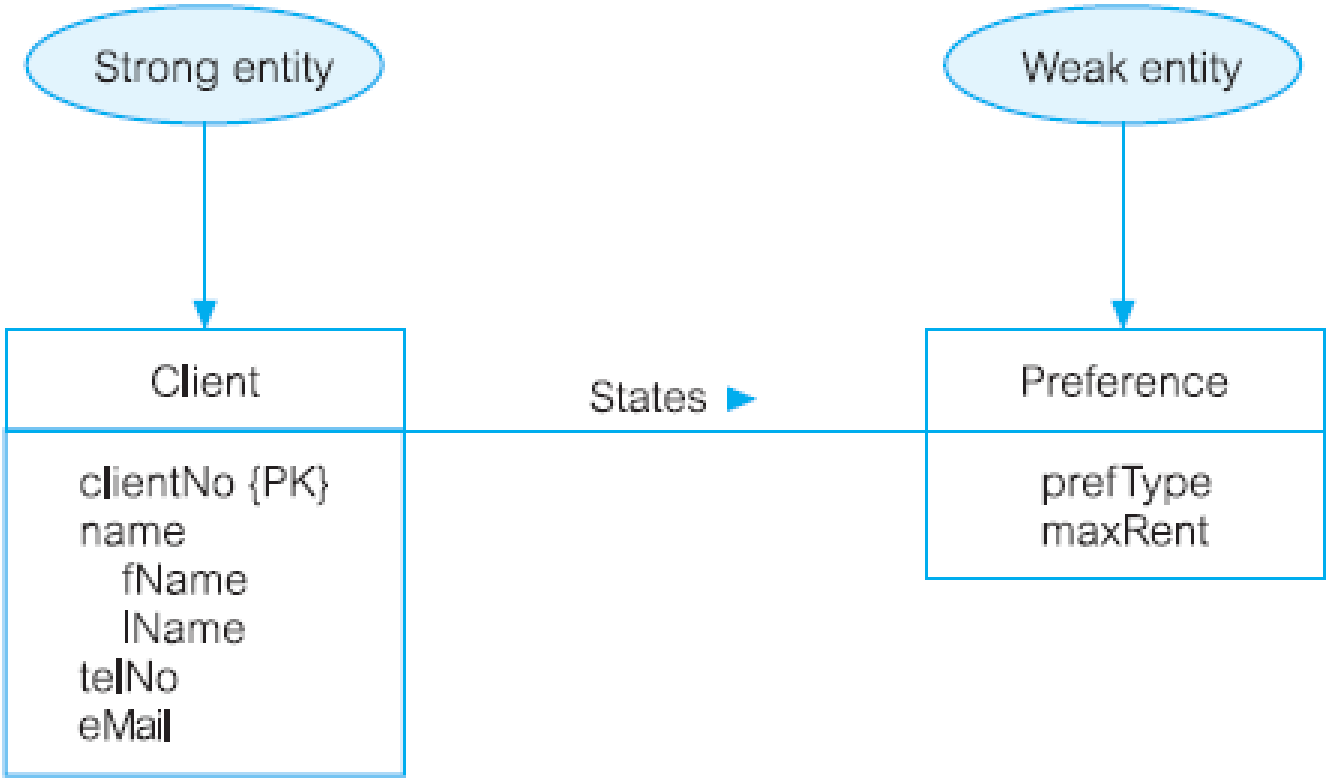
6.4.4. Ключи (продолжение)

- Составной атрибут address сущности Branch: за ним следуют имена атрибутов, входящих в его состав: street, city и postcode.
- Рядом с именами многозначных атрибутов ставится обозначение диапазона возможных значений этого атрибута. Например, для атрибута telNo указан диапазон [1. .3].



6.5. Сущности сильного и слабого типов

- **Сущность сильного типа (Strong entity type)** – тип сущности, существование которого не зависит от какого-то иного типа сущности.
- **Сущность слабого типа (Weak entity type)** – тип сущности/существование которого зависит от какого-то другого типа сущности.



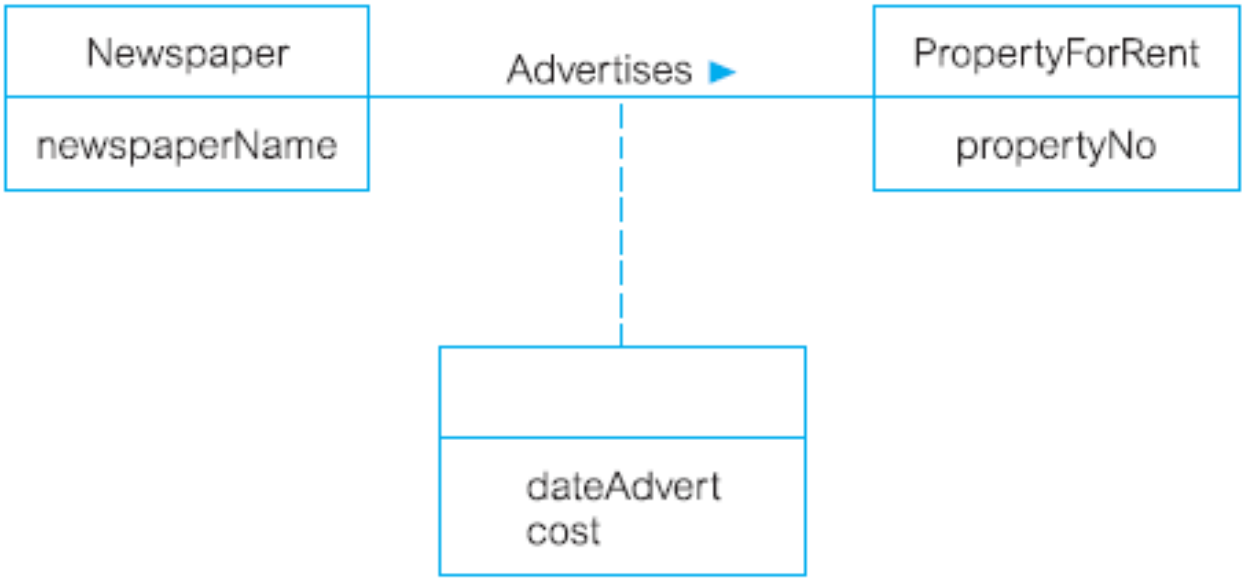
6.5. Сущности сильного и слабого типов (продолжение)

- Характерной особенностью слабой сущности является то, что каждый экземпляр сущности нельзя однозначно идентифицировать с помощью только тех атрибутов, которые относятся к сущности этого типа. Например, обратите внимание, что для сущности Preference нет первичного ключа. Это означает, что каждый экземпляр сущности типа Preference невозможно идентифицировать только с помощью атрибутов этой сущности. Однозначно идентифицировать каждое пожелание клиента можно только, учитывая связь конкретного пожелания с некоторым клиентом, который однозначно идентифицируется с использованием первичного ключа для сущности типа Client, а именно: clientNo. В данном примере сущность Preference рассматривается как зависимая в своем существовании от сущности Client, которая описывает будущего арендатора, желающего арендовать объекты недвижимости конкретного типа и на определенных условиях.
- Сущности слабого типа иногда называют *дочерними, зависимыми* или *подчиненными,* а сущности сильного типа — *родительскими, сущностями-владельцами* или *доминантными.*

6.6. Атрибуты связей

- Чтобы подчеркнуть различие между сущностью и связью, обладающей атрибутом, линия, которая соединяет прямоугольник с именем атрибута (атрибутов) и саму связь, отображается как штриховая.

Газета рекламирует арендуемый объект недвижимости

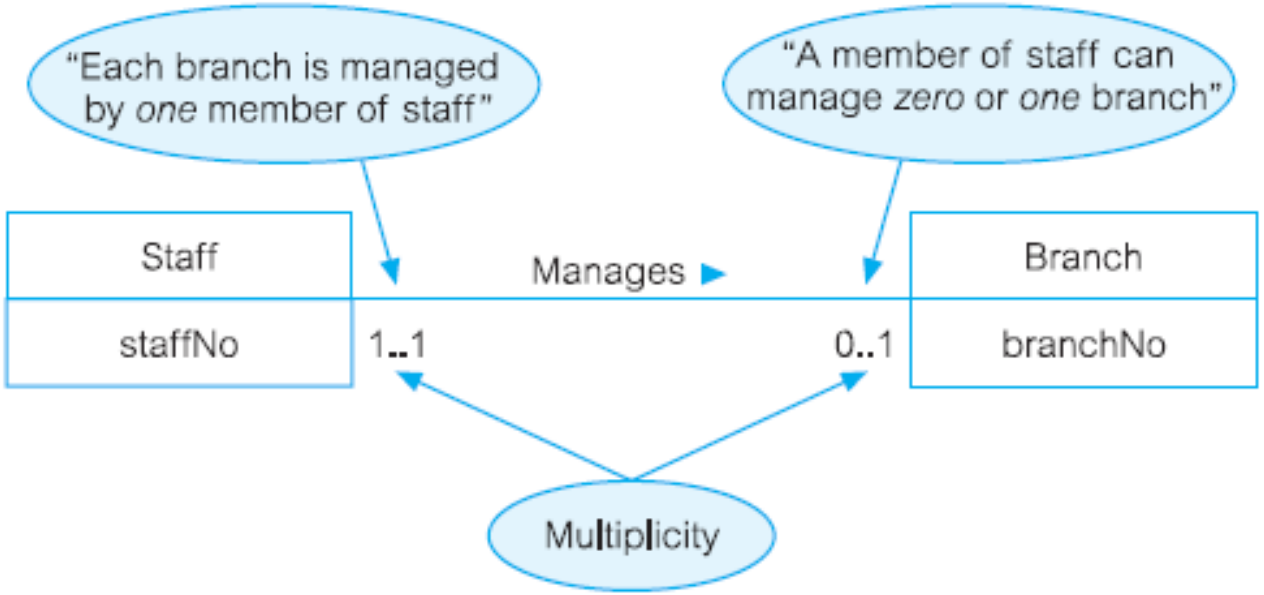


6.7. Структурные ограничения

- Основным типом ограничения, накладываемого на связь, является кратность.
- **Кратность (Multiplicity)** – количество (заданное как одно значение или как диапазон значений), возможных экземпляров сущности некоторого типа, которые могут быть связаны с одним экземпляром сущности другого типа с помощью определенной связи.
- Ограничения кратности описывают способ формирования связи между сущностями. Они отражают требования (или бизнес-правила), установленные пользователем или предприятием.
- Наиболее распространенной степенью связи является двухсторонняя. Двухсторонние связи обычно обозначаются как связи
  - «один к одному» (1:1)
  - **«**один ко многим» (1:*)
  - или «многие ко многим» (*:*).
- Важно отметить, что не все ограничения предметной области могут быть легко представлены в виде ER-модели. Например, с помощью ER-модели сложно представить условие, согласно которому сотрудник компании получает дополнительный день отпуска за каждый год стажа работы в компании.

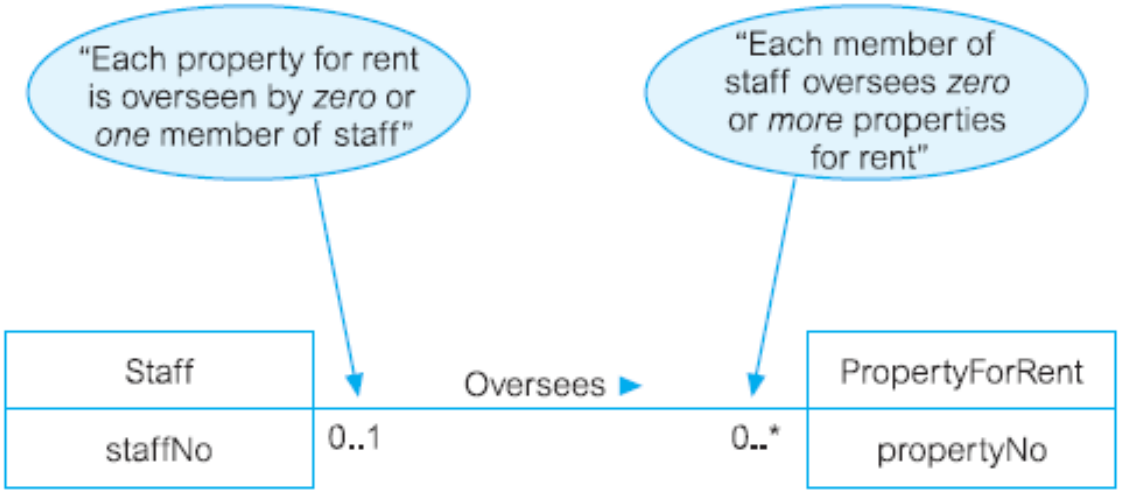
6.7.1. Связь «один к одному»

- ER-диаграмма связи Staff Manages Branch.
- Чтобы показать, что под управлением любого сотрудника компании может находиться нуль или одно отделение, на этой схеме рядом с сущностью Branch помещено обозначение 0. .1.
- А для указания на то, что в любом отделении всегда имеется один менеджер, рядом с обозначением сущности Staff помещено обозначение 1. .1.



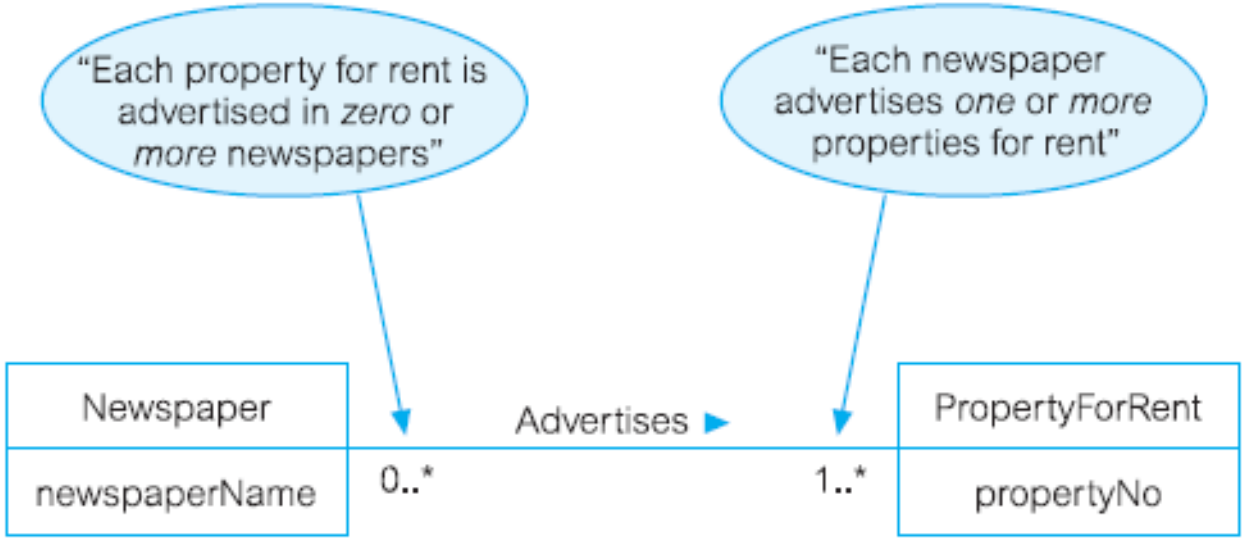
6.7.2. Связь «один ко многим»

- ER-диаграмма связи Staff Oversees PropertyForRent.
- Для указания того, что количество арендуемых объектов недвижимости, находящихся под управлением любого сотрудника компании, может составлять от нуля и больше, на этой схеме рядом с изображением сущности PropertyForRent помещено обозначение 0. . *.
- А для указания на то, что каждым арендуемым объектом недвижимости управляет нуль или один сотрудник компании, рядом с изображением сущности Staff помещено обозначение 0. . 1.
- Для связей типа 1:* должно быть выбрано имя, которое имеет смысл, если связь рассматривается в направлении от «одного» ко «многим».



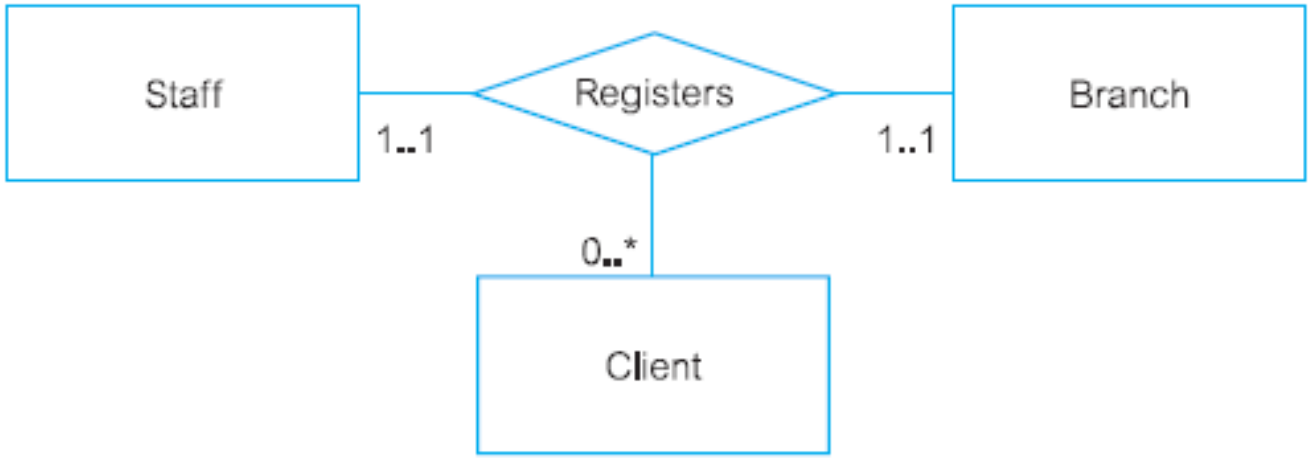
6.7.3. Связь «многие ко многим»

- ER-диаграмма связи Newspaper *Advertises* PropertyForRent.
- Для указания на то, что каждая газета может публиковать рекламу об одном или нескольких объектах недвижимости, сдаваемых в аренду, рядом с изображением сущности PropertyForRent показано обозначение 1..*.
- Для указания на то, что количество газет, в которых публикуется реклама о сдаваемых в аренду объектах недвижимости, может составлять нуль и больше, рядом с изображением сущности Newspaper помещено обозначение 0. . *.
- Для связи *:* может быть выбрано имя, которое имеет смысл при рассмотрении этой связи либо в том, либо в ином направлении.



6.7.4. Кратность сложных связей

- **Кратность сложной связи** – количество (заданное как одно значение или как диапазон значений) экземпляров сущности определен него типа в n-арной связи, определяемое после фиксации остальных (n – 1) значений.



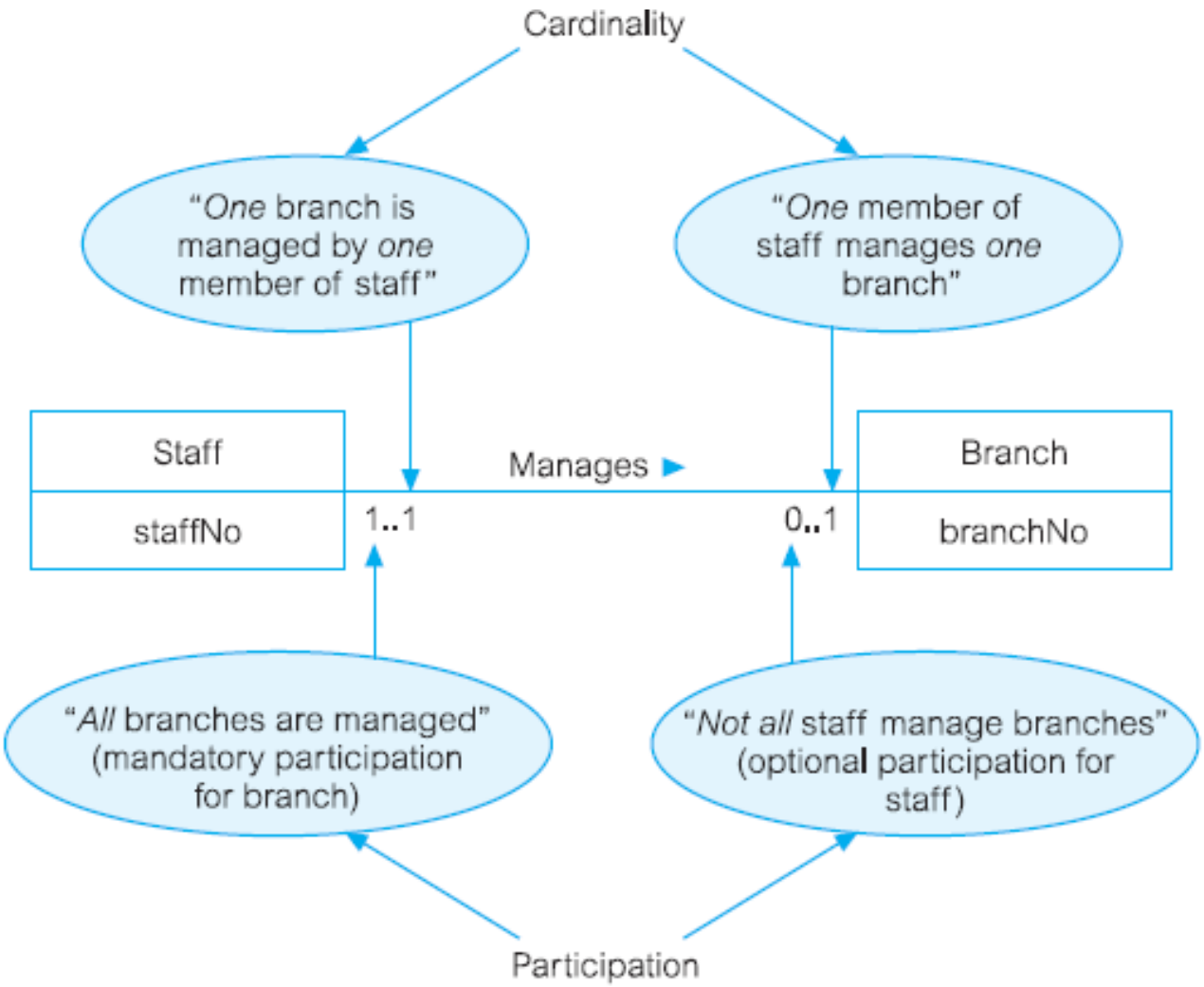
6.7.5. Ограничения кардинальности и степени участия

- Ограничения кратности фактически состоят из двух отдельных ограничений, известных как *кардинальность* и *степень участия.*
- **Кардинальность (Cardinality)** – определяет максимальное количество возможных экземпляров связи для каждой сущности, участвующей в связи конкретного типа.
- Понятие *кардинальности двухсторонней связи* является обобщением понятия *кратности,* которое выше рассматривалось как связь «один к одному» (1:1), «один ко многим» (1:*) и «многие ко многим» (*:*).
- **Степень участия (Participation)** – определяет, участвуют ли в связи все или только некоторые экземпляры сущности.
- Ограничение степени участия определяет, должны ли участвовать в конкретной связи все экземпляры сущности (такое условие принято называть *обязательным участием)* или только некоторые экземпляры (такое условие называется *необязательным участием).*

6.7.5. Ограничения кардинальности и степени участия (продолжение)

- Степень участия сущностей в связи проявляется в виде минимальных значений для диапазонов кратности на каждой стороне связи. Необязательное участие представляется минимальным значением 0, а обязательное участие – минимальным значением 1.
- Важно учитывать, что степень участия конкретной сущности в связи представлено минимальным значением на *противоположной* стороне этой связи, т. е. минимальным значением кратности, стоящим возле связанной сущности.

6.7.5. Ограничения кардинальности и степени участия (продолжение)



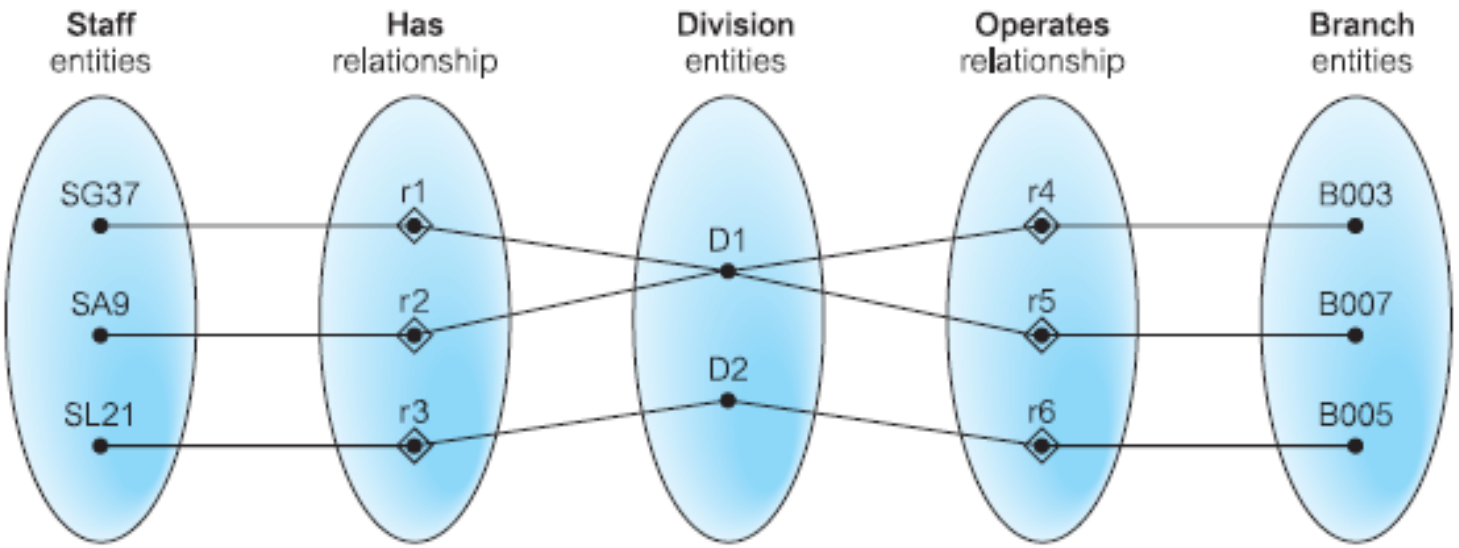
6.8. Проблемы ER-моделирования

- Эти проблемы принято называть *ловушками соединения* (connection trap). Они обычно возникают вследствие неправильной интерпретации смысла некоторых связей.
- Два вида ловушек соединения:
  - ловушка типа *«разветвление»* (fan trap)
  - ловушка типа *«разрыв»* (chasm trap)

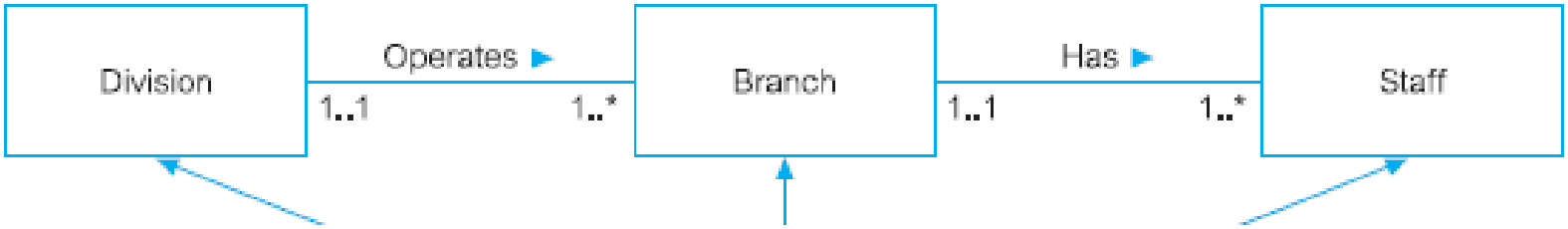
6.8.1. Ловушка типа «разветвление»

- Ловушка типа «разветвление» – имеет место в том случае, когда модель отображает связь между *типами* сущностей, но путь между отдельными сущностями [этого типа определен неоднозначно.
- Имеет место, когда две или несколько связей типа 1:* исходят из одной сущности.
- На основании этой модели можно сделать вывод, что один отдел (Division) может состоять из нескольких отделений компании (Branch) и в нем может работать многочисленный штат сотрудников.
- Проблемы начинаются при попытках выяснить, *в каком отделении компании работает каждый из сотрудников отдела*.

6.8.1. Ловушка типа «разветвление» (продолжение)



Устранить эту проблему можно путем перестройки ER-модели для представления правильного взаимодействия этих сущностей



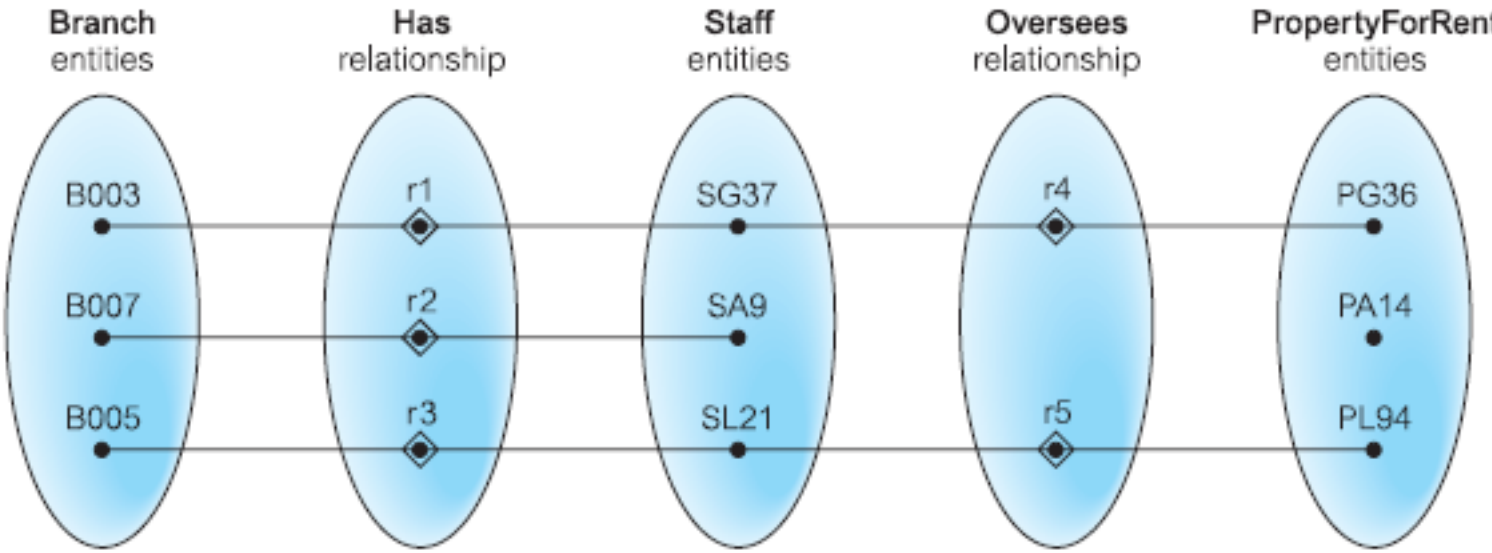
Реструктуризация модели устраняет ловушку типа «разветвление»

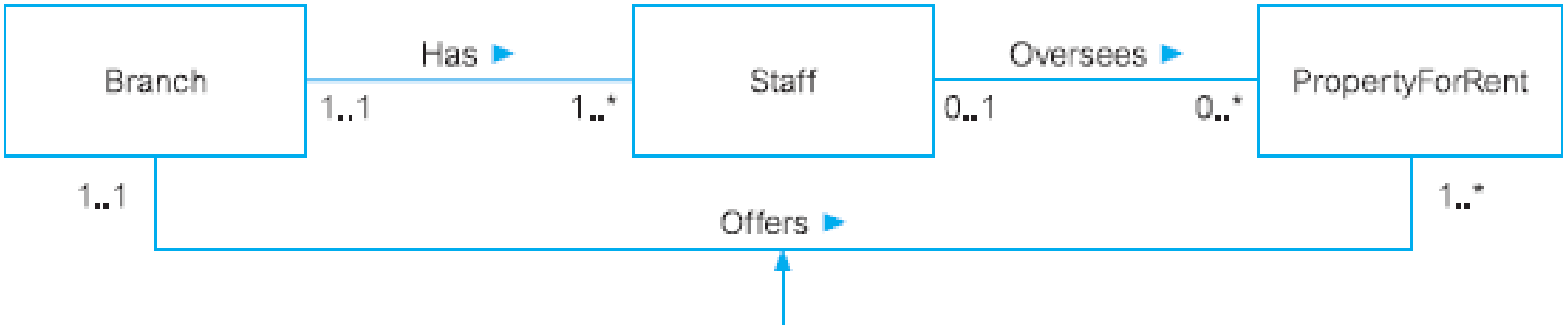
6.8.2. Ловушка типа «разрыв»

- Ловушка типа «разрыв» – имеет место в том случае, когда в модели предполагается наличие связи между *типами* сущностей, но не существует пути между отдельными сущностями этих типов.
- Может возникать, если существует одна или несколько связей с минимальной кратностью, равной нулю (которая обозначает необязательное участие), и эти связи составляют часть пути между взаимосвязанными сущностями.
- Проблемы начинаются при попытках выяснить, *какое отделение компании отвечает за работу с объектом под конкретным номером*.
- Кратность сущностей Staff и PropertyForRent в связи Oversees имеет минимальное значение, равное нулю, а это означает, что некоторые объекты недвижимости не могут быть связаны с отделением компании с помощью информации о сотрудниках. Поэтому для разрешения этой проблемы следует ввести недостающую связь *Offers* между сущностями Branch и PropertyForRent.

6.8.2. Ловушка типа «разрыв» (продолжение)

Схема семантической сети ER-модели

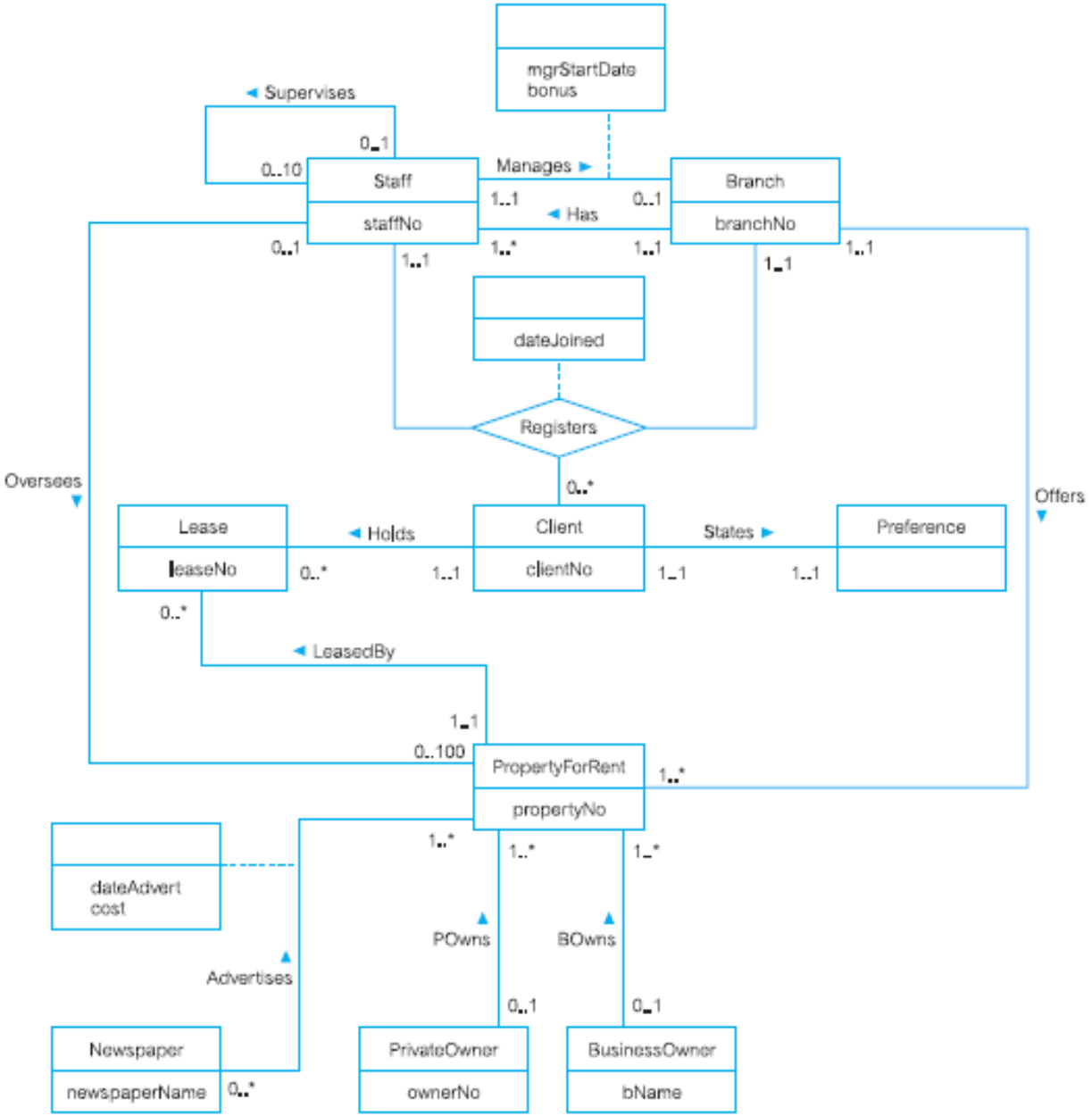




Добавление связи Offers устраняет ловушку типа «разрыв»

6.9. Общий вид ER-диаграммы

ER-диаграмма представления Branch



6.10. Расширенная модель «сущность–связь»

- ER-модель, оснащенная дополнительными семантическими концепциями, называется *расширенной моделью «сущность–связь»* (Enhanced Entity-Relationship — EER).
- Наиболее важные и полезные дополнительные концепции EER-модели:
  - уточнение/обобщение (specialization/generalization)
  - агрегирование (aggregation)
  - композиция (composition)

6.10.1. Уточнение/обобщение

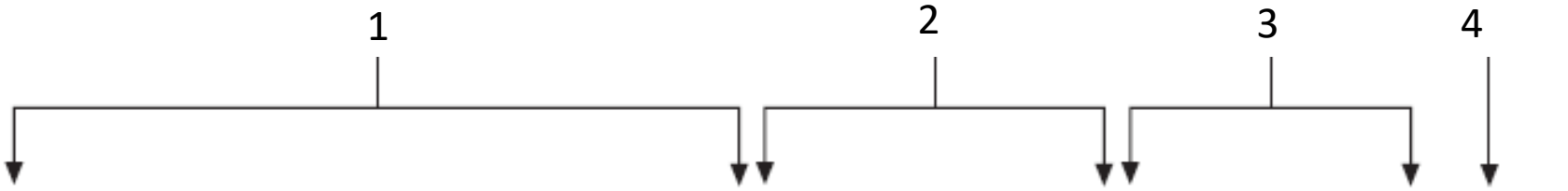
- Понятие уточнения/обобщения (specialization/generalization) связано со специальными типами сущностей, которые принято называть *суперклассами* и *подклассами,* а также с процессом *наследования атрибутов*.
- **Суперкласс (Superclass).** Тип сущности, включающий одну или несколько различных группировок ее экземпляров, которые должны быть представлены в модели данных.
- **Подкласс.** Различимая группировка экземпляров некоторого типа сущности, которая должна быть представлена в модели данных.
- Например, сущности, принадлежащие к типу сущности Staff, можно разбить на категории Manager (Менеджер), SalesPersonnel (Работник торговли) и Secretary (Секретарь). Иными словами, сущность Staff является суперклассом подклассов Manager, SalesPersonnel и Secretary.

6.10.1. Уточнение/обобщение (продолжение)

- Связь между суперклассом и любым из его подклассов называется *связью суперкласс/подкласс.* Например, связью суперкласс/подкласс является связь Staff/Manager.
- Каждый элемент подкласса является также элементом суперкласса. Иными словами, сущность, которая относится к подклассу, является той же сущностью, которая относится и к суперклассу, но выполняет в суперклассе и подклассе разные роли. Связь между суперклассом и подклассом является связью «один к одному» (1:1).
- Подклассы могут перекрываться. Например, Manager и SalesPersonnel представляют собой перекрывающиеся подклассы суперкласса Staff, т. е. менеджер может быть еще и торговым работником.
- С другой стороны, не каждый элемент суперкласса должен быть элементом одного из подклассов. Например, в компании могут быть сотрудники, не имеющие такой конкретной должности, как менеджер или торговый работник.

6.10.1. Уточнение/обобщение (продолжение)

Отношение AllStaff содержит данные обо всем персонале



staffNo	name	position	salary	mgrStartDate	bonus	sales Area	car Allowance	typing Speed
SL21	John White	Manager	30000	01/02/95	2000			
SG37	Ann Beech	Assistant	12000					
SG66	Mary Martinez	Sales Manager	27000			SA1A	5000	
SA9	Mary Howe	Assistant	9000					
SL89	Stuart Stern	Secretary	8500					100
SL31	Robert Chin	Snr Sales Asst	17000			SA2B	3700	
SG5	Susan Brand	Manager	24000	01/06/91	2350			

1 – атрибуты, характерные **для всех** сотрудников
2 – атрибуты, характерные для менеджеров отделений (Manager)
3 – атрибуты, характерные для торговых работников (SalesPersonnel)
4 – атрибут, характерный для работников секретариата (Secretary)

Обратите внимание на пустые поля в таблице.

6.10.1. Уточнение/обобщение (продолжение)

- Суперклассы и подклассы могут применяться для того, чтобы в процессе разработки не приходилось описывать экземпляры разных сущностей, которые могут характеризоваться различными атрибутами, в пределах одной сущности (см. пример на предыдущем слайде).
- Концепции суперклассов и подклассов могут быть введены в ER-модель по двум основным причинам.

1. Они позволяют не описывать несколько раз аналогичные сущности, благодаря чему экономится время проектировщика, а ER-диаграммы становятся более удобными для восприятия.
2. Они позволяют ввести в проект больший объем семантической информации в форме, знакомой для широкого круга пользователей. Например, утверждения, что *«Менеджер* IS-A *(является) сотрудником компании»* и *«Квартира* IS-A *(является) одним из типов объектов недвижимости»,* позволяют сообщить важную семантическую информацию в краткой форме.

6.10.1. Уточнение/обобщение (продолжение)

- Представитель подкласса *наследует* все атрибуты суперкласса, а также обладает атрибутами, характерными только для подкласса.
- Например: представитель подкласса SalesPersonnel наследует все атрибуты суперкласса Staff (такие как staffNo, name, position и salary), а также обладает атрибутами, характерными только для подкласса (salesArea и carAllowance).
- Подкласс, как таковой, также является сущностью и поэтому может, в свою очередь, включать один или несколько подклассов. Сущность вместе с ее подклассами, подклассами своих подклассов и т.д. составляет так называемую *иерархию типов.*
- Подкласс, принадлежащий к нескольким суперклассам, называется *общим подклассом (***shared subclass***).* Иными словами, элемент общего подкласса должен быть элементом всех взаимосвязанных суперклассов. Следовательно, атрибуты суперклассов наследуются общим подклассом, который может также иметь свои собственные дополнительные атрибуты. Этот процесс называется *множественным наследованием.*

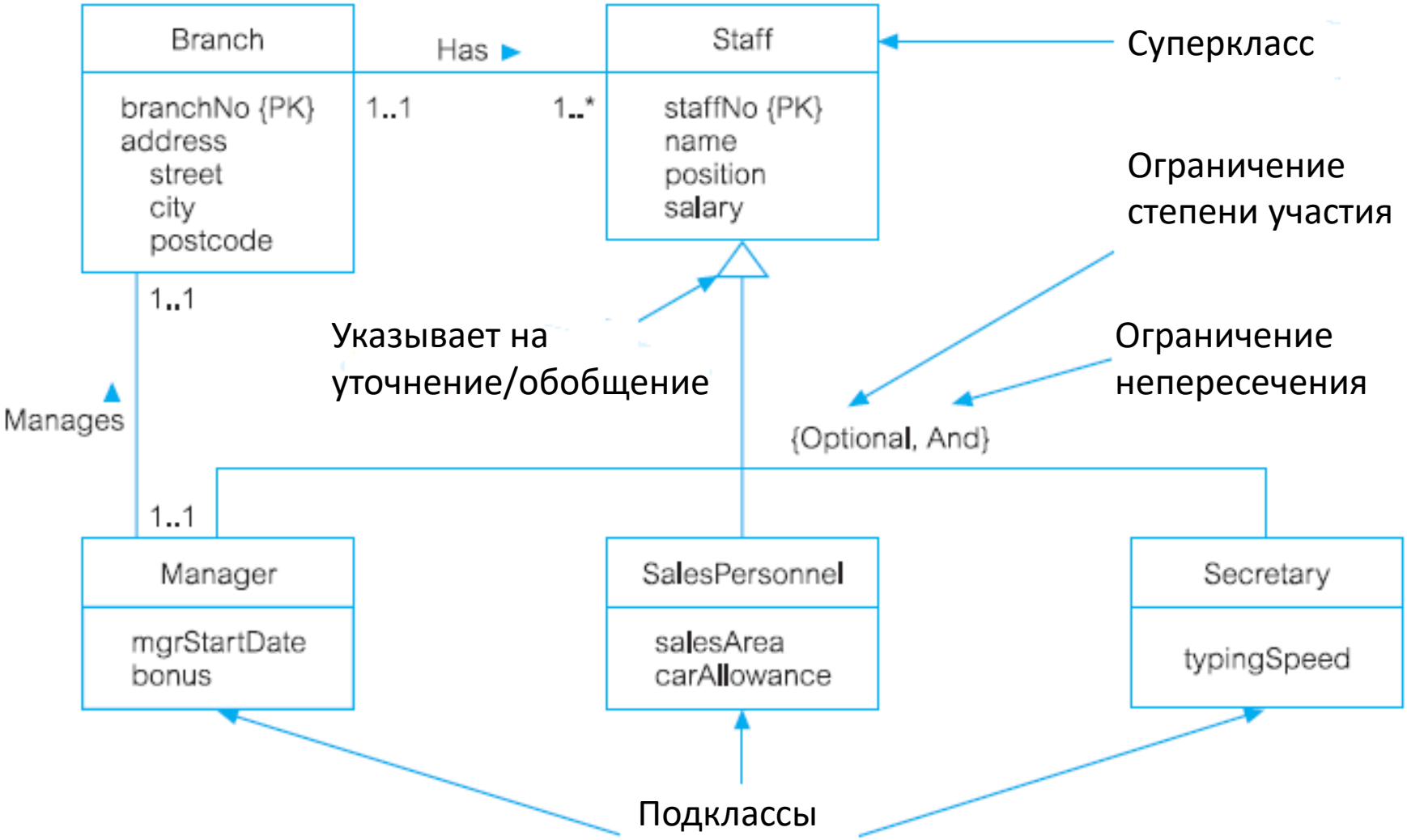
6.10.1. Уточнение/обобщение (продолжение)

- **Уточнение** (**Specialization**). Процесс выявления различий между членами сущности путем выявления их отличительных особенностей.
- Уточнение представляет собой нисходящий подход к выявлению множества суперклассов и относящихся к ним подклассов. Набор подклассов определяется на основе некоторых отличительных особенностей сущностей суперкласса. После выявления множества подклассов сущности определенного типа можно определить атрибуты, характерные для каждого подкласса (если это потребуется), а также выявить все связи между каждым подклассом и другими типами сущностей или подклассов (в случае необходимости).
- Например, рассмотрим модель, в которой все сотрудники компании представлены в виде одной сущности Staff. Применяя процесс уточнения сущности Staff, разработчик должен выявить различия между экземплярами этой сущности, рассматривая их изначально как экземпляры с различимыми атрибутами и/или связями. Сотрудники компании, выполняющие функции менеджера, торгового работника и секретаря, имеют различимые атрибуты, что позволяет выделить подклассы Manager, SalesPersonnel и Secretary специализированного суперкласса Staff.

6.10.1. Уточнение/обобщение (продолжение)

- **Обобщение (Generalization).** Процесс минимизации различий между сущностями путем выявления их общих характеристик.
- Процесс обобщения основан на восходящем подходе, который приводит к созданию обобщенного суперкласса на основе первоначальных типов сущностей.
- Например, рассмотрим модель, в которой категории сотрудников компании Manager, SalesPersonnel и Secretary представлены как различные типы сущностей. Применяя к этим сущностям процесс обобщения, разработчик должен выявить подобие между ними, например общие атрибуты и связи. Поскольку эти сущности имеют некоторые атрибуты, общие для всего персонала, то разработчик может сделать заключение, что сущности Manager, SalesPersonnel и Secretary являются подклассами обобщенного суперкласса staff.
- Поскольку процесс обобщения может рассматриваться как обратный по отношению к процессу уточнения, весь этот подход к моделированию известен под общим названием *уточнение/обобщение.*

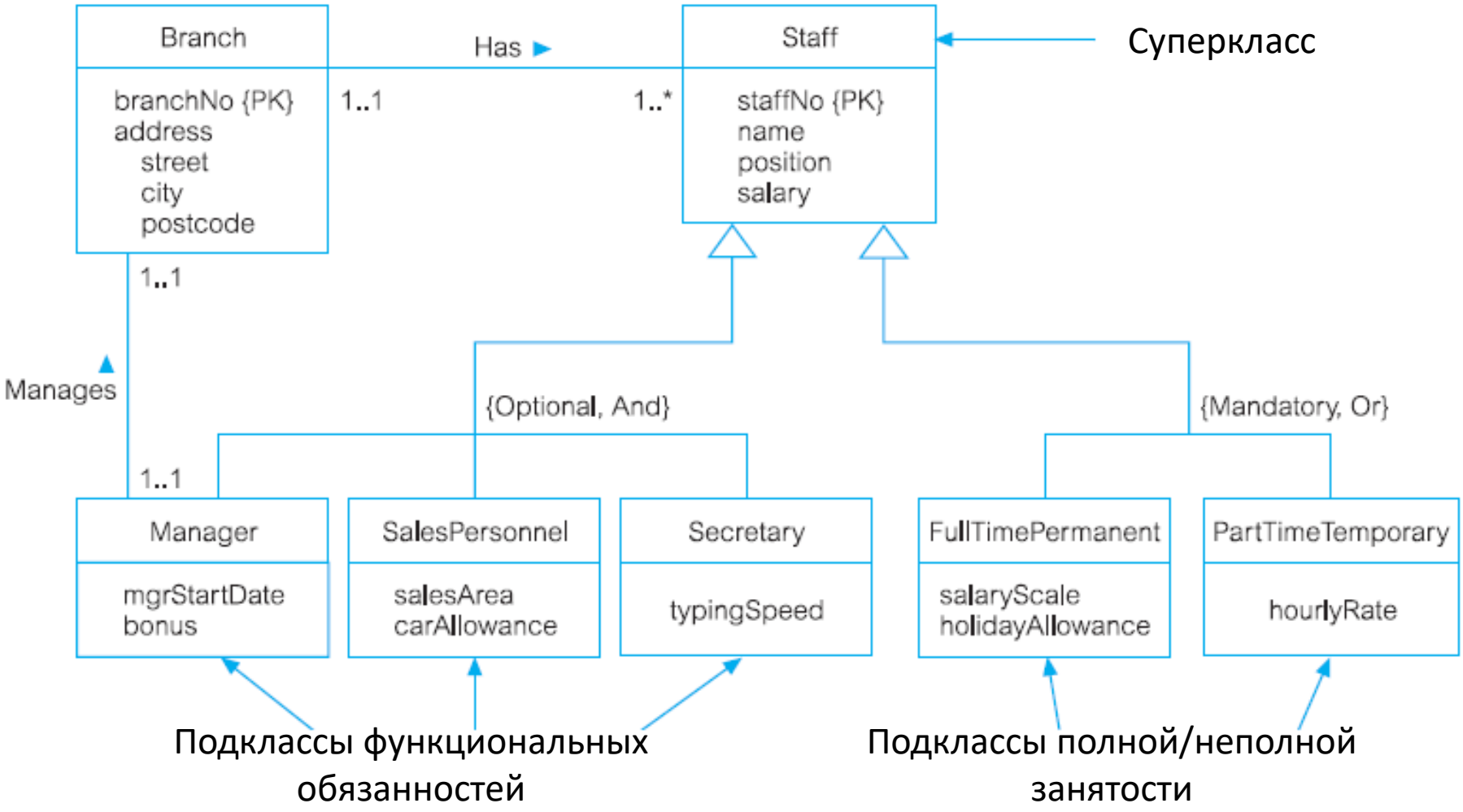
6.10.1. Уточнение/обобщение (продолжение)



6.10.1. Уточнение/обобщение (продолжение)

- Уточнение одной и той же сущности может проводиться на основе разных отличительных характеристик.
- Например, в результате проведения другого процесса уточнения сущности Staff могут быть выявлены подклассы FullTimePermanent (Постоянный сотрудник) и Part Time Temporary (Временный сотрудник), которые позволяют учесть различия между трудовыми договорами, по которым происходит прием на работу сотрудников компании.

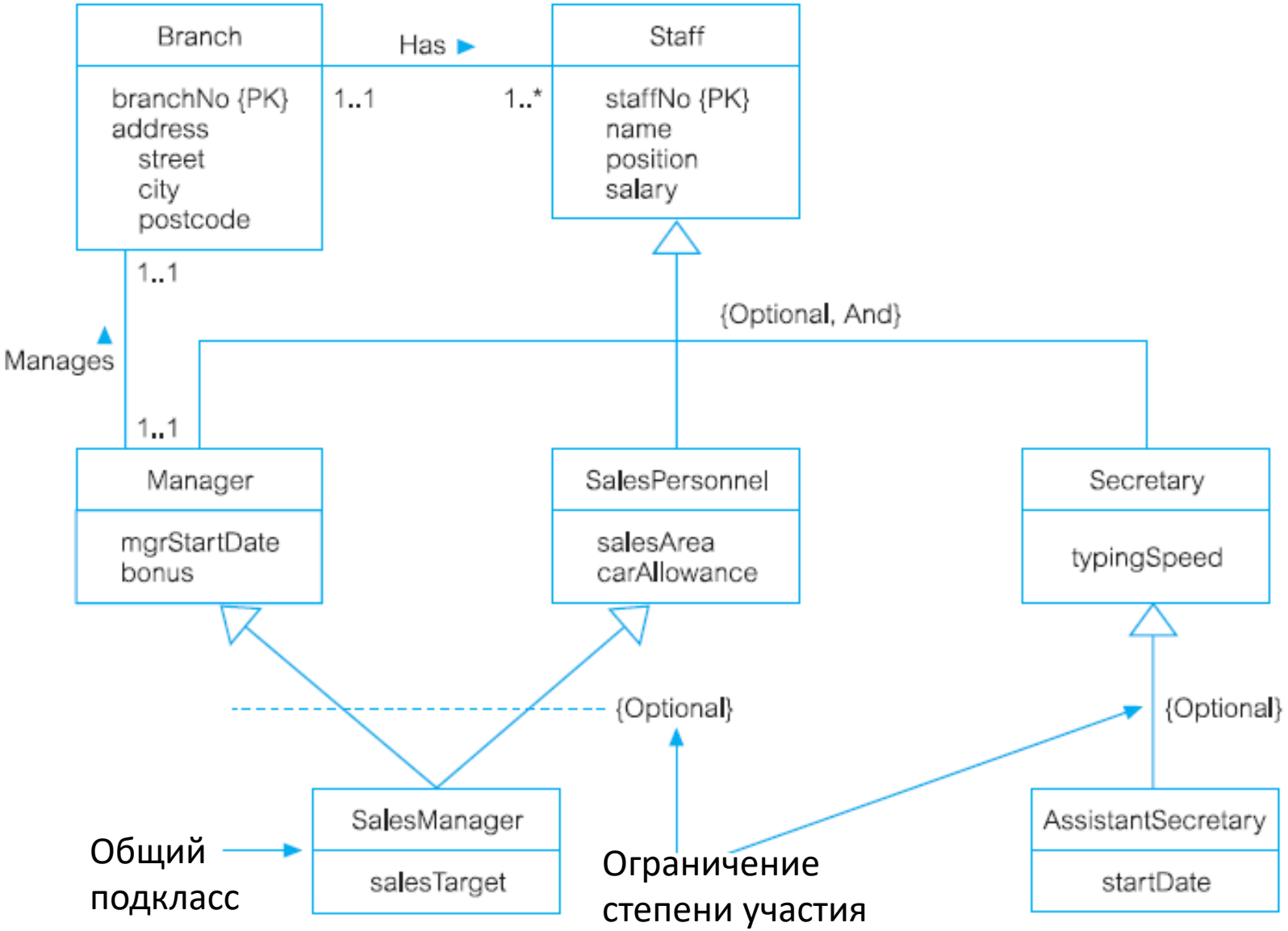
6.10.1. Уточнение/обобщение (продолжение)



6.10.1. Уточнение/обобщение (продолжение)

- Суперкласс, его подклассы, подклассы этих подклассов и т. д. составляют так называемую *иерархию типов.*
- В рассматриваемой компании имеются сотрудники, принадлежащие к общему подклассу (shared subclass) – SalesManager (Менеджер торгового отдела), а подкласс Secretary имеет собственный подкласс Assistant Secretary (Заместитель секретаря). Иными словами, представитель общего подкласса SalesManager должен быть членом подклассов SalesPersonnel и Manager, а также суперкласса Staff. Следовательно, подкласс SalesManager наследует атрибуты суперкласса Staff (stiaffNo, name, position и salary) и атрибуты подклассов SalesPersonnel (salesArea и carAllowance) и Manager (mgrStartDate и bonus), а также имеет собственный дополнительный атрибут salesTarget (торговая специализация).
- В свою очередь, AssistantSecretary является подклассом класса Secretary, который является подклассом класса Staff. Это означает, что представитель подкласса AssistantSecretary должен быть членом подкласса Secretary и суперкласса Staff. Поэтому подкласс AssistantSecretary наследует атрибуты суперкласса Staff (staffNo, name, position и salary) и атрибут подкласса Secretary (typingSpeed), а также имеет собственный дополнительный атрибут startDate.

6.10.1. Уточнение/обобщение (продолжение)



6.10.1.1. Ограничения процесса уточнения/обобщения

- В процессе уточнения/обобщения могут применяться ограничения двух типов, а именно: ограничения степени участия (participation constraint) и ограничения непересечения (disjoint constraint).
- **Ограничение степени участия**. Определяет, должен ли каждый член суперкласса быть отнесен к какому-то подклассу.
- Ограничение степени участия может быть обязательным или необязательным.
- Связь суперкласс/под класс с обязательным участием указывает на то, что каждый элемент суперкласса должен быть также элементом какого-то подкласса. Для указания на обязательное участие в фигурных скобках рядом с треугольником, указывающим на суперкласс, должно быть приведено ключевое слово «Mandatory» (Обязательный).
- Связь суперкласс/подкласс с необязательным участием указывает на то, что некоторые члены суперкласса могут не принадлежать ни к одному из подклассов. Для указания на то, что применяется необязательное ограничение степени участия, в фигурных скобках под треугольником, который указывает на суперкласс, должно быть помещено ключевое слово «Optional» (Необязательный).

6.10.1.1. Ограничения процесса уточнения/обобщения (продолжение)

- **Ограничение непересечения.** Описывает связь между членами подклассов и указывает, может ли член суперкласса принадлежать только к *одному* или *нескольким* подклассам.
- Ограничение непересечения может применяться, только если суперкласс имеет несколько подклассов. Если подклассы являются непересекающимися, то каждый экземпляр сущности может быть членом только одного из подклассов.
- Для представления непересекающейся связи суперкласс/подкласс вслед за ограничением степени участия в фигурных скобках должно быть помещено ключевое слово «Or».
- Если подклассы, которые входят в иерархию уточнения/обобщения, не являются непересекающимися (в таком случае их называют *пересекающимися),* то любой экземпляр сущности может быть членом нескольких подклассов.
- Для представления пересекающейся связи суперкласс/подкласс в фигурных скобках вслед за ограничением степени участия должно быть помещено ключевое слово «And».

6.10.1.1. Ограничения процесса уточнения/обобщения (продолжение)

- Для иерархий, имеющих единственный подкласс на некотором уровне, нет необходимости включать ограничение непересечения, и по этой причине для подклассов SalesManager и AssistantSecretary предусмотрено только ограничение степени участия.
- Ограничения непересечения и степени участия, характерные для процессов уточнения и обобщения, являются различными, поэтому при их совместном использовании связи между подклассами могут подразделяться на четыре категории:
  - «обязательный и непересекающийся»
  - «необязательный и непересекающийся»
  - «обязательный и пересекающийся»
  - «необязательный и пересекающийся»

6.10.1.2. Пример 1

- Рассмотрим сущность Staff, которая представляет всех сотрудников компании. Но в спецификации требований к данным для представления Branch учебного проекта DreamHome упомянуты две категории функциональных обязанностей, а именно: Manager и Supervisor (Инспектор). При выборе наилучшего способа моделирования данных о сотрудниках компании могут рассматриваться следующие три варианта.

1. Представление всех сотрудников компании с помощью обобщенной сущности staff.
2. Создание трех отдельных сущностей Staff, Manager и Supervisor.
3. Трактовка сущностей Manager и Supervisor как подклассов суперкласса Staff.

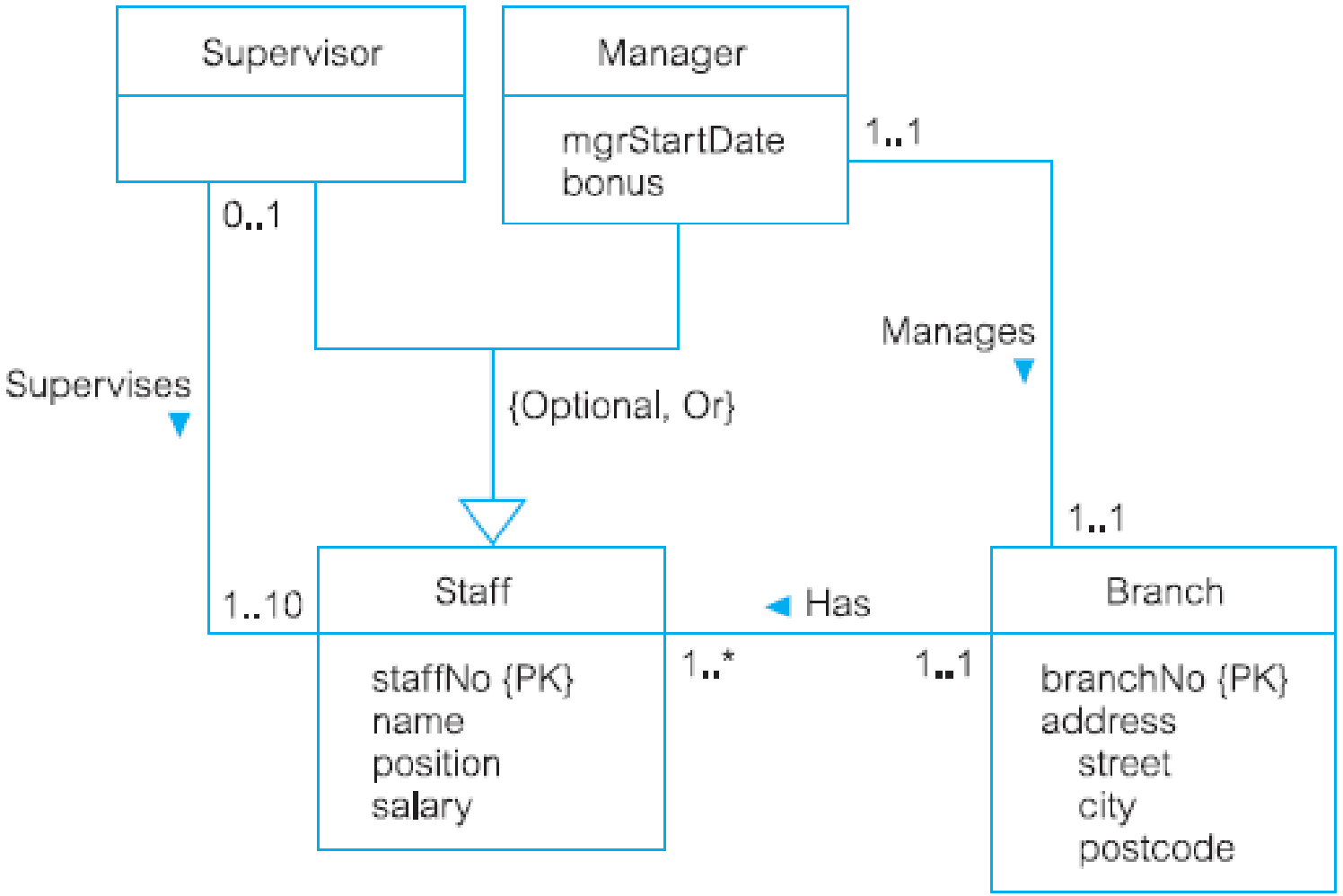
6.10.1.2. Пример 1 (продолжение)

- Выбранный вариант зависит от степени общности атрибутов и связей, относящихся к каждой сущности. Допустим, что все атрибуты сущности Staff представлены в сущностях Manager и Supervisor, в том числе взят тот же первичный ключ — staffNo. Кроме того, сущность Supervisor не имеет дополнительных атрибутов, представляющих соответствующие функциональные обязанности. С другой стороны, сущность Manager имеет два дополнительных атрибута: mgrStartDate и bonus. К тому же сущности Manager и Supervisor относятся к разным связям — Manager *Manages* Branch (Менеджер управляет отделением) и Supervisor *Supervises* Staff (Инспектор управляет персоналом). С учетом этой информации был выбран третий вариант и созданы подклассы Manager и Supervisor суперкласса Staff.
- Следует отметить, что на этой EER-диаграмме подклассы показаны над суперклассами. Но взаимное расположение подклассов и суперклассов на схеме не имеет значения; важно только, чтобы треугольник уточнения/обобщения указывал на суперкласс.

6.10.1.2. Пример 1 (продолжение)

- Иерархия уточнения/обобщения сущности Staff является необязательной и непересекающейся (что указано с помощью ключевых слов {Optional, Or}), поскольку:
  - не все сотрудники компании являются менеджерами или инспекторами,
  - кроме того, ни один сотрудник компании не может одновременно занимать должности менеджера и инспектора.
- Такая форма особенна удобна для отображения общих атрибутов, относящихся к этим подклассам и суперклассу Staff, а также для индивидуальных связей, относящихся к каждому подклассу, а именно: связей Manager *Manages* Branch и Supervisor *Supervises* Staff.

6.10.1.2. Пример 1 (продолжение)



6.10.1.3. Пример 2

- Рассмотрим иерархию уточнения/обобщения связи между владельцами объектов недвижимости. В спецификации требований к данным для представления Branch описаны два типа владельцев: PrivateOwner (Владелец недвижимости) и BusinessOwner (Владелец предприятия).
- В данном случае также имеются три варианта выбора модели представления данных о владельцах недвижимости.

1. Сущности PrivateOwner и BusinessOwner рассматриваются как две отдельные сущности.
2. Владельцы обоих типов отображаются в виде обобщенной сущности Owner (Владелец).
3. Сущности PrivateOwner и BusinessOwner представляются как подклассы суперкласса Owner.

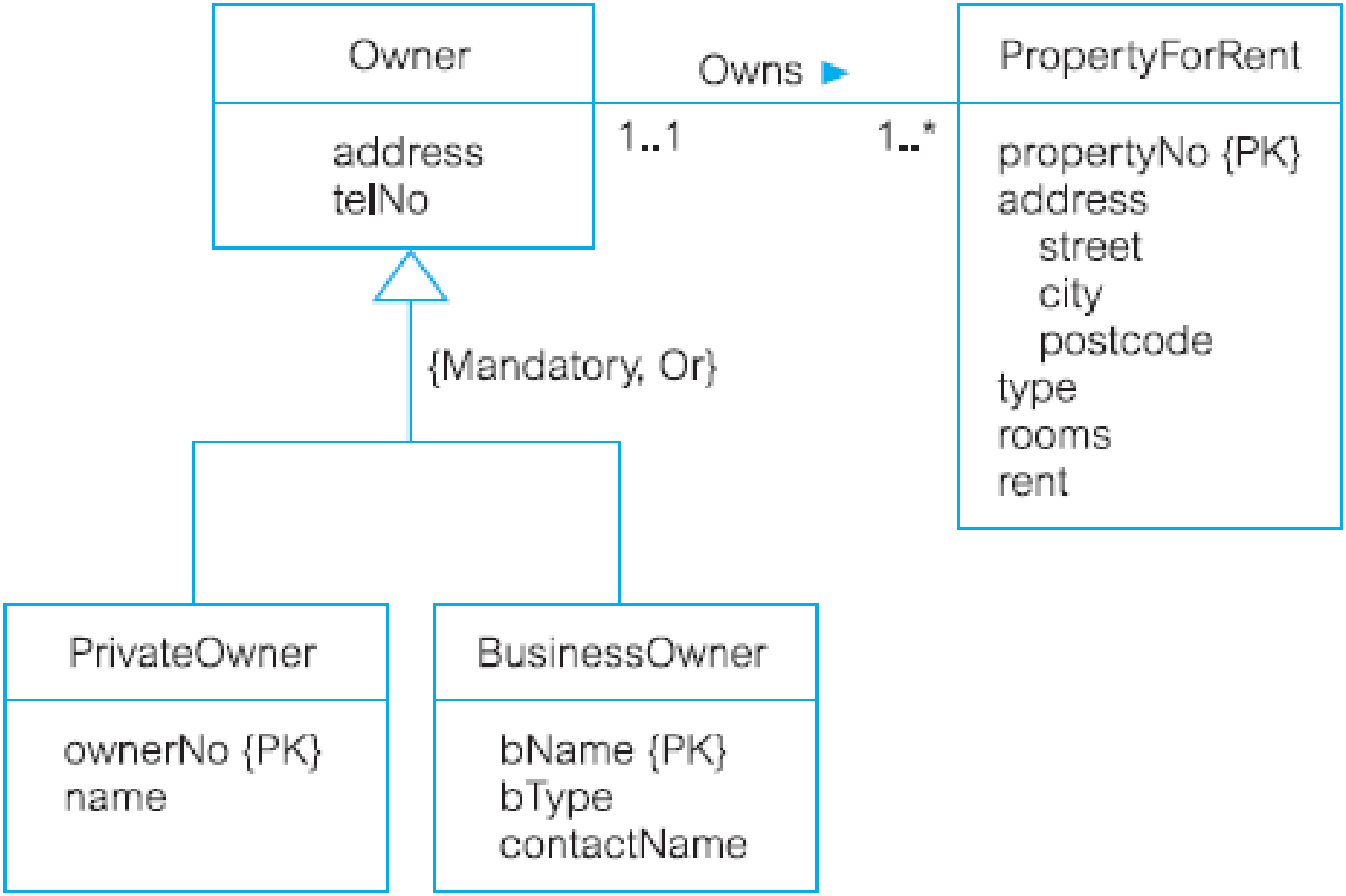
6.10.1.3. Пример 2 (продолжение)

- Прежде чем принять решение о том, какой из этих вариантов следует выбрать, необходимо изучить атрибуты и связи, которые относятся к каждой из этих сущностей. Сущности PrivateOwner и BusinessOwner обладают общими атрибутами (address (адрес) и telNo (номер телефона)) и имеют аналогичную связь с объектом недвижимости, сдаваемым в аренду (PrivateOwner *POwns* PropertyForRent и BusinessOwner *BOwns* PropertyForRent). Однако владельцы обоих типов имеют также разные атрибуты; например, сущность PrivateOwner имеет индивидуальные атрибуты ownerNo и name, а сущность BusinessOwner — индивидуальные атрибуты bName, bType и contactName. В этом случае решено создать суперкласс Owner с подклассами PrivateOwner и BusinessOwner.

6.10.1.3. Пример 2 (продолжение)

- Иерархия уточнения/обобщения сущности Owner является обязательной и непересекающейся (как указано с помощью ключевых слов {Mandatory, Or}), поскольку владелец должен быть либо владельцем недвижимости, либо владельцем предприятия, но не владельцем того и другого одновременно.
- В данной схеме решено связать суперкласс Owner с сущностью PropertyForRent с помощью связи *Owns* (Владеет).

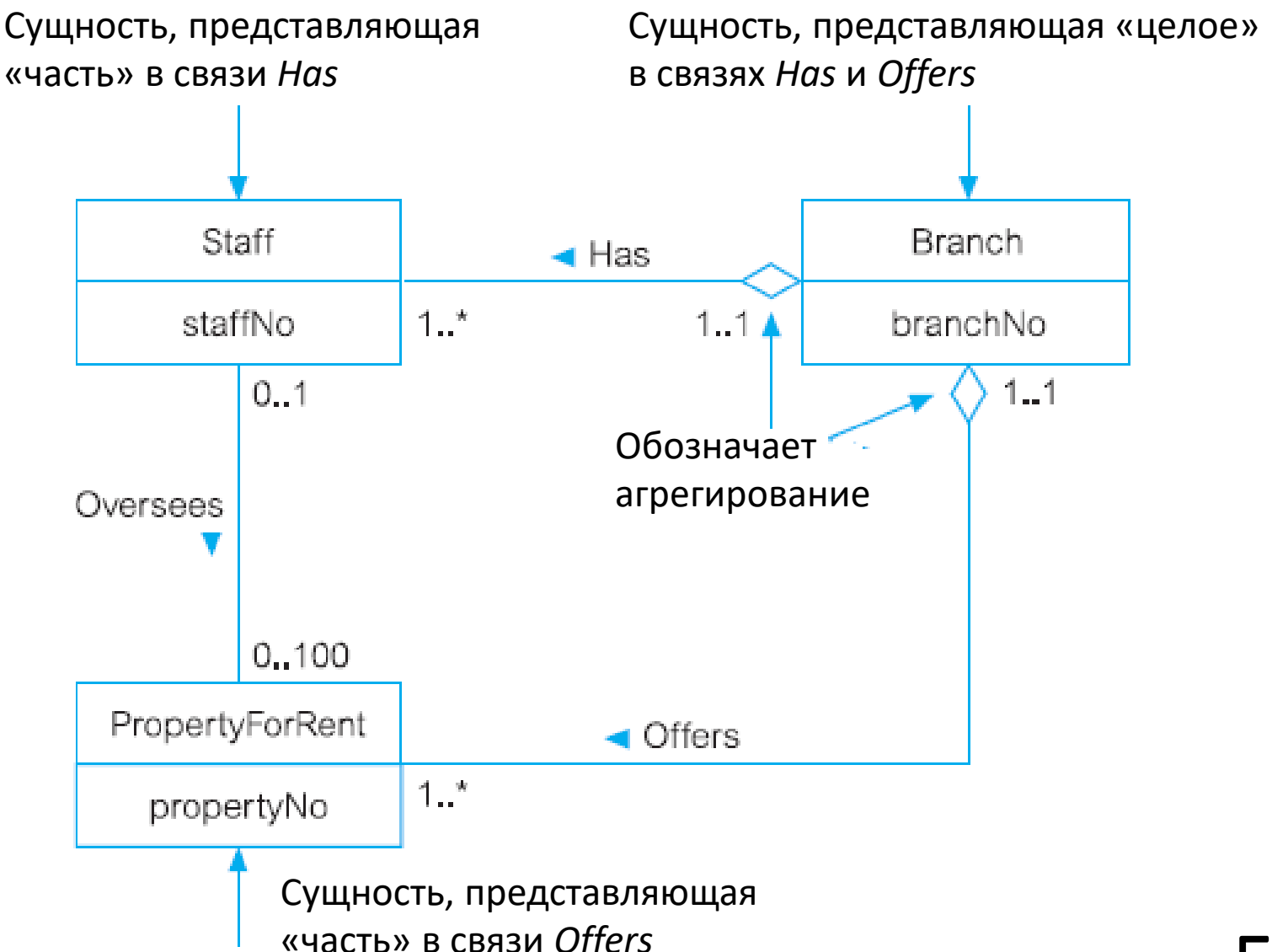
6.10.1.3. Пример 2 (продолжение)



6.10.2. Агрегирование

- **Агрегирование.** Представляет связь «has-a» (включает) или «is-part-of» (входит в состав) между типами сущностей, один из которых представляет «целое», а другой — «часть».
- Связь обычно применяется для обозначения соотношений между двумя типами сущностей, которые концептуально находятся на одном и том же уровне.
- Но иногда возникает необходимость моделировать связь «has-a» или «is-part-of», в которой одна сущность представляет собой более крупную сущность («целое»), состоящую из меньших сущностей («частей»). Связь такого особого вида называется агрегированием.
- Агрегирование не изменяет сути связи, с помощью которой происходит перемещение от целого к его частям, а также не определяет зависимости между временем существования целого и его частей.
- Примером агрегирования является связь *Has,* которая определяет соответствие между сущностями Branch («целое») и Staff («часть»).

6.10.2. Агрегирование (продолжение)



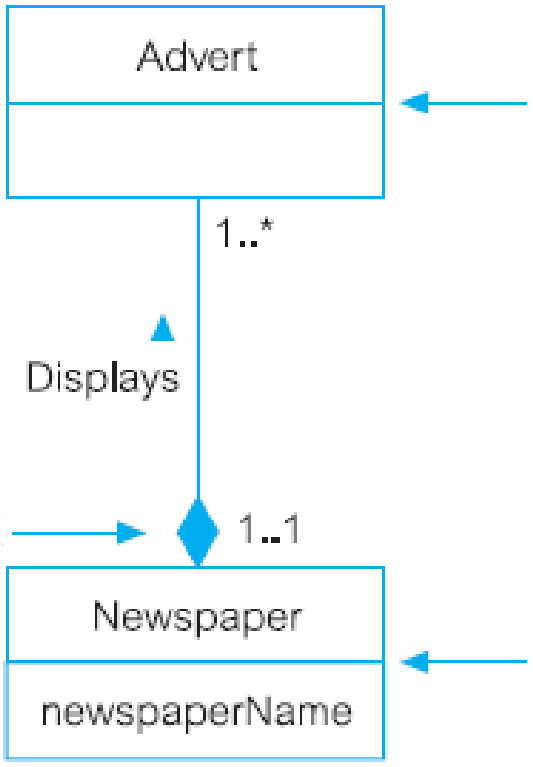
6.10.3. Композиция

- **Композиция**. Особая форма агрегирования, представляющая зависимость между сущностями, которая характеризуется полной принадлежностью и совпадением срока существования между «целым» и «частью».
- Агрегирование представляет исключительно концептуальную связь, но в его определении ничего не говорится о том, по каким признакам «целое» отличается от «части». Тем не менее для использования в моделировании данных предусмотрена одна из разновидностей агрегирования, называемая *композицией,* которая характеризуется полной принадлежностью и совпадением срока существования между «целым» и «частью».
- В композиции «целое» отвечает за размещение его «частей», а это означает, что композиция должна управлять созданием и разрушением своих «частей». Иными словами, любой объект в любой момент времени может входить в состав только одной композиции.

6.10.3. Композиция (продолжение)

- В качестве иллюстрации рассмотрим такой пример композиции, как связь Displays (Публикует), которая устанавливает соотношение между сущностью Newspaper (Газета) и сущностью Advert (Рекламное объявление). Поскольку данная связь рассматривается как композиция, тем самым подчеркивается факт, что сущность Advert («часть») принадлежит одной и только одной сущности Newspaper («целому»).
- В этом и состоит отличие композиции от агрегирования, в котором любая часть может быть общей для многих целых. Например, сущность Staff может быть «частью» одной или нескольких сущностей Branch.
- Агрегирование и композицию следует использовать, только если есть необходимость выделить такие особые связи между типами сущностей, как «has-a» или «is-part-of», которые подразумевают, что процессы создания, обновления и удаления этих тесно связанных сущностей должны охватывать обоих участников связи (и «целое», и «часть»).

6.10.3. Композиция (продолжение)



Сущность, представляющая «часть» в связи *Displays*

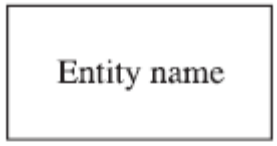
Показывает композицию

Сущность, представляющая «целое» в связи *Displays*

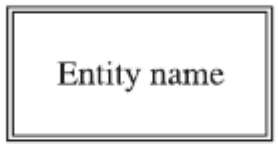
6.11. Альтернативные нотации

- Мы использовали нотацию (систему обозначений), основанную на языку UML (Unified Modeling Language).
- Существуют другие нотации:
  - Нотация П. Чена
  - Нотация «вороньи лапки» (Crow’s Feet)

6.11.1. Нотация П. Чена



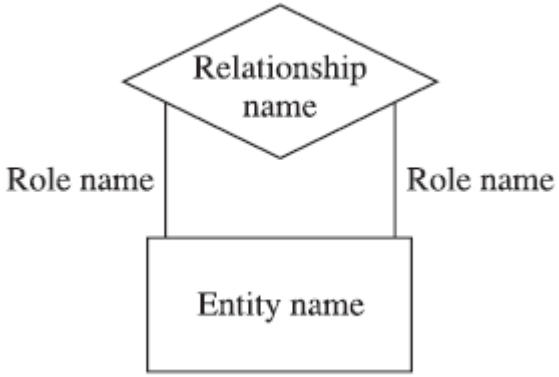
Сильная сущность



Слабая сущность

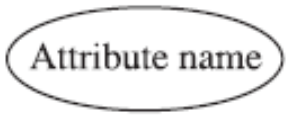
Связь

Связь, ассоциированная со слабой сущностью

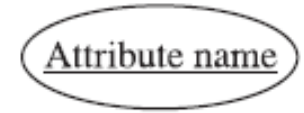


Рекурсивная связь с именами ролей, указывающими, какие функции выполняет сущность в связи

6.11.1. Нотация П. Чена (продолжение)

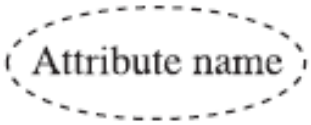


Атрибут

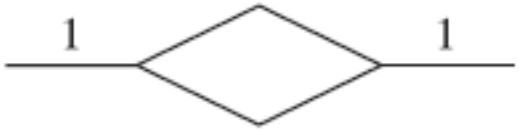


Атрибут первичного ключа

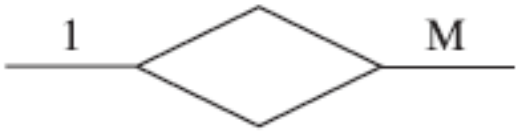
Многозначный атрибут



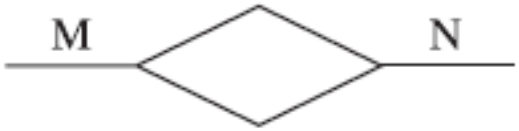
Производный атрибут



Связь «один к одному» (1:1)



Связь «один ко многим» (1:M)



Связь «многие ко многим» (M:N)

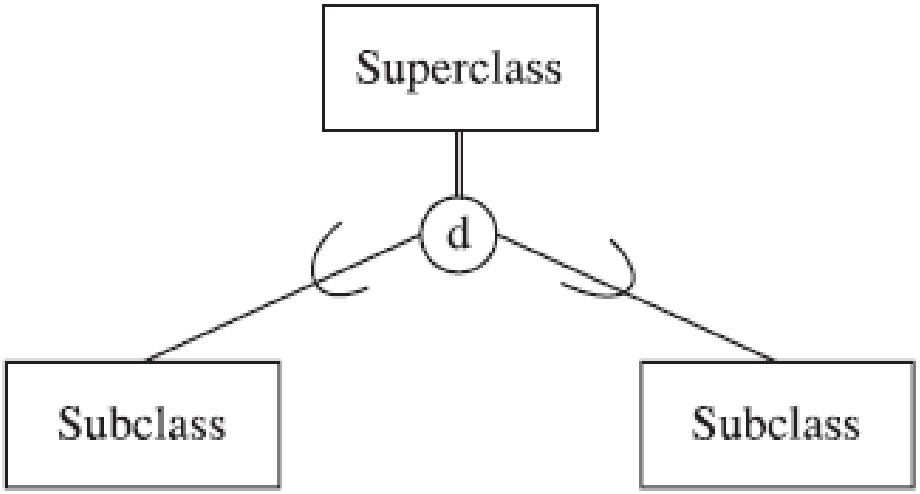
6.11.1. Нотация П. Чена (продолжение)

Связь «один ко многим» с обязательным участием сущностей A и B

Связь «один ко многим» с необязательным участием сущности A и обязательным участием сущности B

Связь «один ко многим» с необязательным участием сущностей A и B

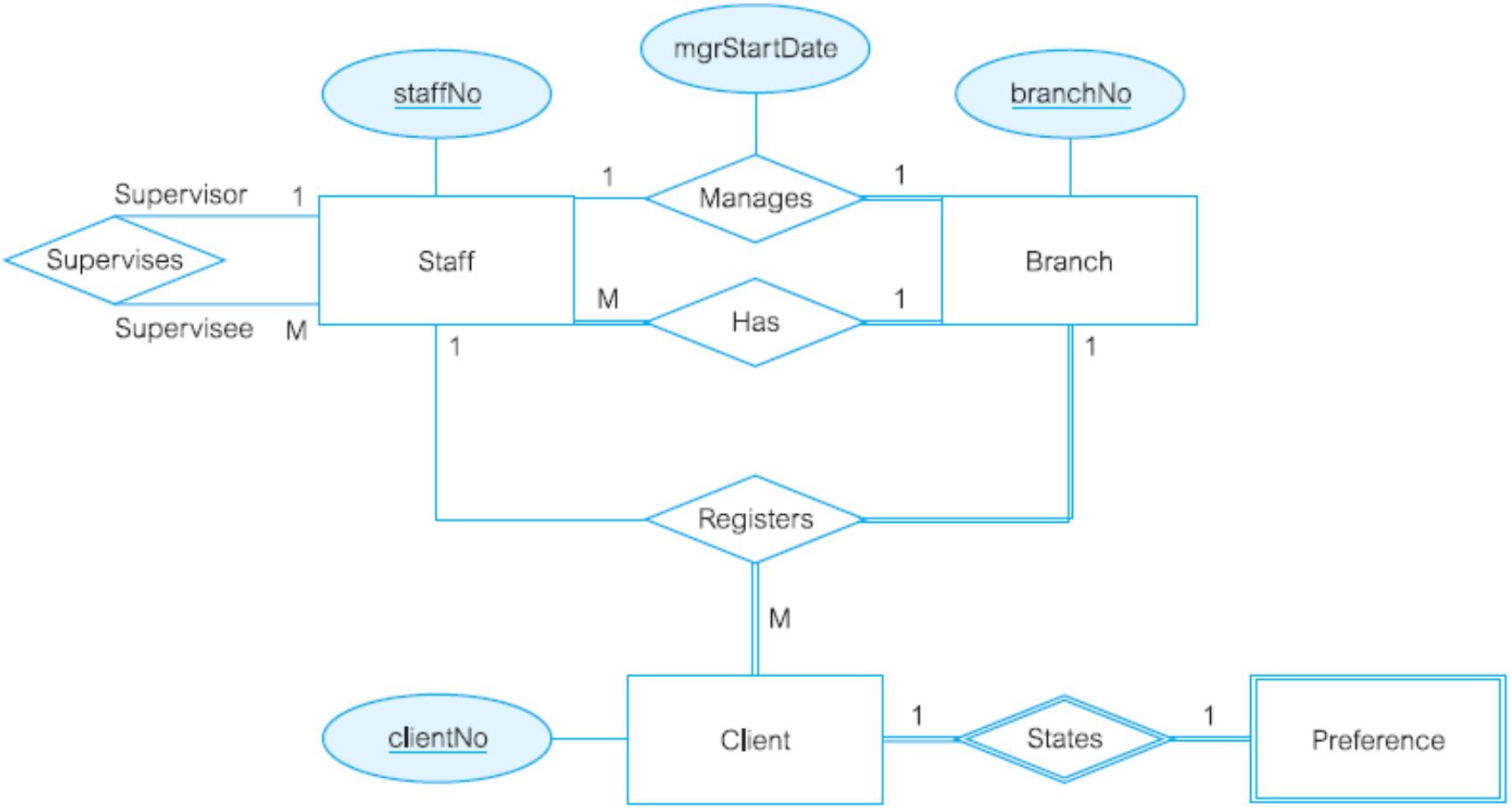
6.11.1. Нотация П. Чена (продолжение)



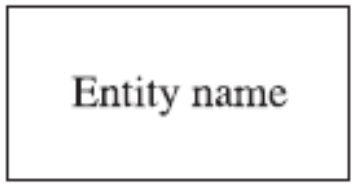
Уточнение/обобщение

Если в кружке стоит буква «d», связь является непересекающейся,
а если в кружке стоит буква «o», связь не является непересекающейся.
Двойная линия от суперкласса к кружку обозначает обязательное участие,
одинарная линия обозначает необязательное участие.

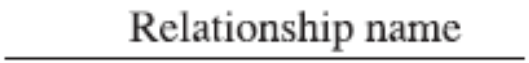
6.11.1.1. Нотация П. Чена (пример)



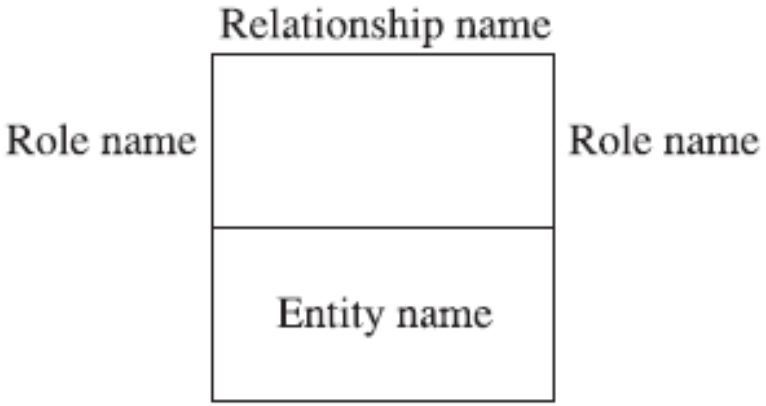
6.11.2. Нотация «вороньи лапки»



Сущность

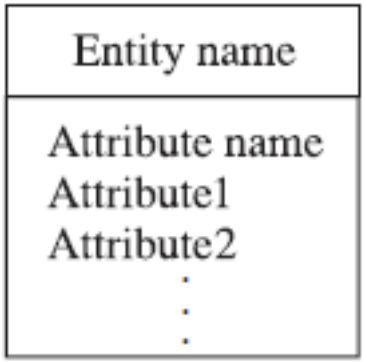


Связь

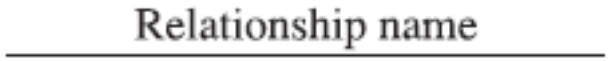


Рекурсивная связь с именами ролей, указывающими, какие функции выполняет сущность в связи

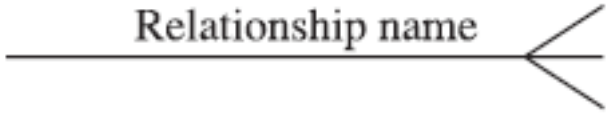
6.11.2. Нотация «вороньи лапки» (продолжение)



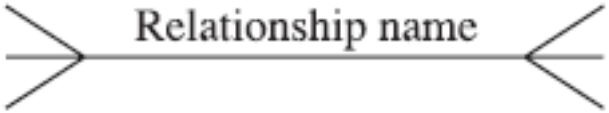
Атрибуты перечисляются в нижней части обозначения сущности.
Атрибуты первичного ключа подчеркиваются.
Многозначный атрибут помещается в фигурные скобки ({}).



Связь «один к одному»

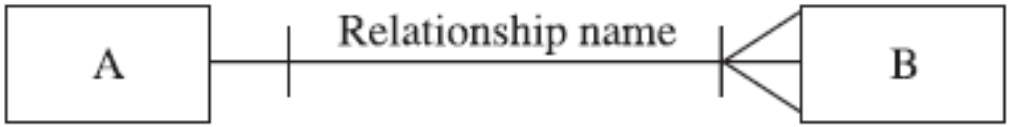


Связь «один ко многим»

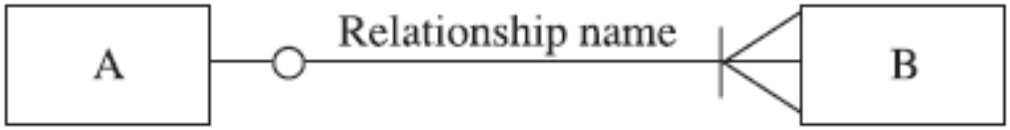


Связь «многие ко многим»

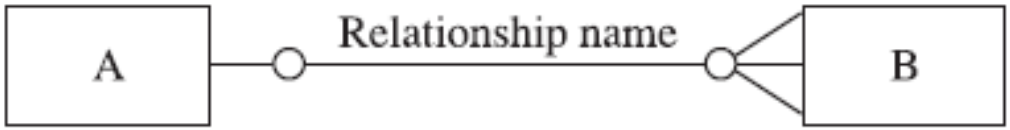
6.11.2. Нотация «вороньи лапки» (продолжение)



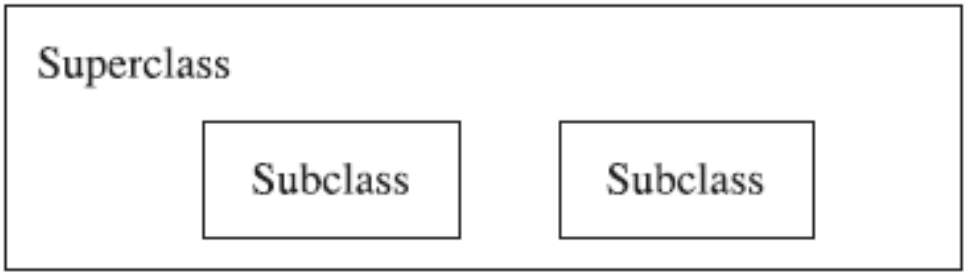
Связь «один ко многим» с обязательным участием сущностей A и B



Связь «один ко многим» с необязательным участием сущности A и обязательным участием сущности B

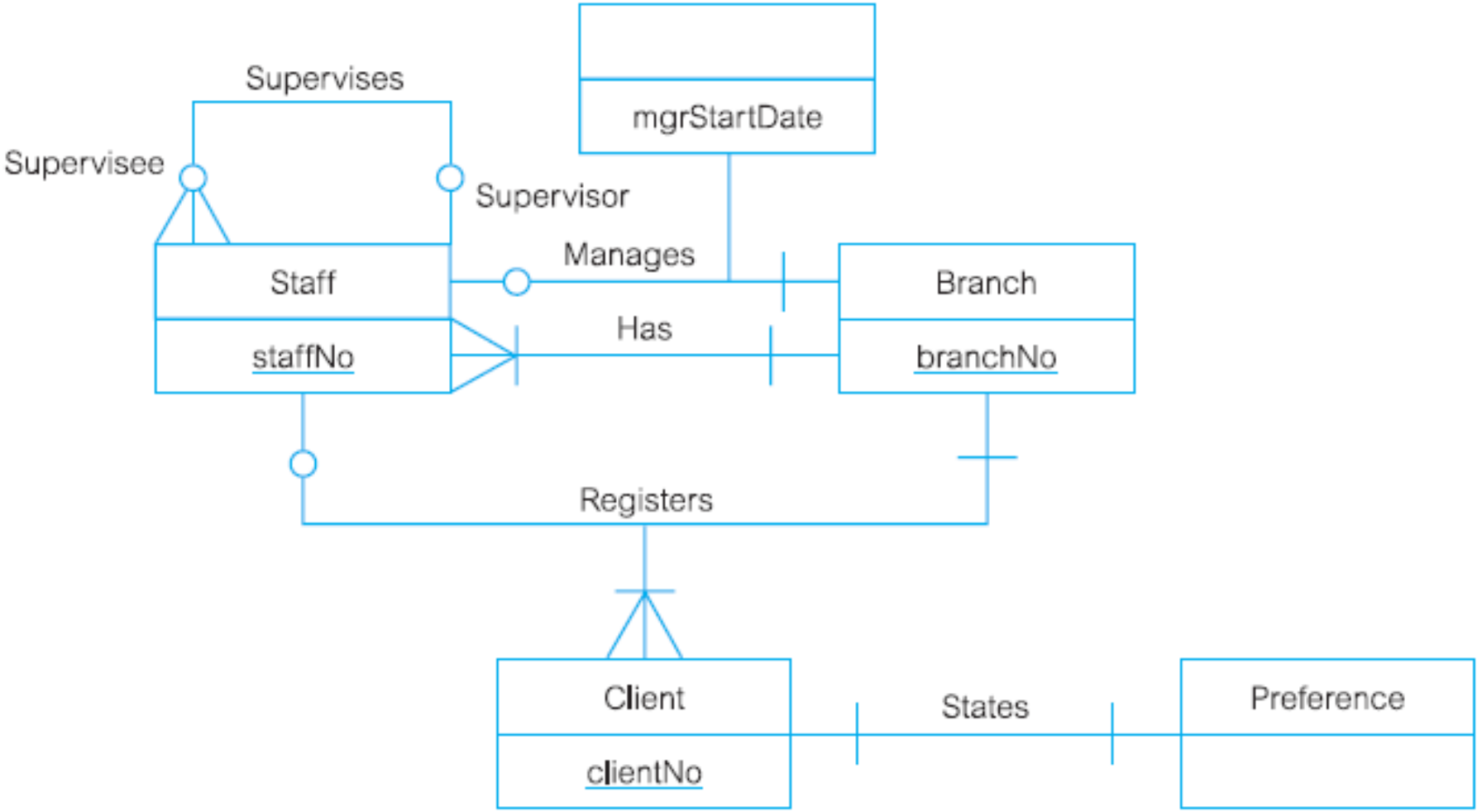


Связь «один ко многим» с необязательным участием сущностей A и B



Для представления иерархии уточнения/обобщения используются прямоугольники, заключенные в прямоугольник

6.11.2.1. Нотация «вороньи лапки» (пример)



6.12. Полезные ссылки

- Software Ideas Modeler https://www.softwareideas.net/

Литература

1. Гарсиа-Молина, Г. Системы баз данных. Полный курс : пер. с англ. / Гектор Гарсиа-Молина, Джеффри Ульман, Дженнифер Уидом. – М. : Вильямс, 2003. – 1088 с.
2. Грофф, Дж. SQL. Полное руководство : пер. с англ. / Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель. – 3-е изд. – М. : Вильямс, 2015. – 960 с.
3. Дейт, К. Дж. Введение в системы баз данных : пер. с англ. / Крис Дж. Дейт. – 8-е изд. – М. : Вильямс, 2005. – 1328 с.
4. **Коннолли, Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика : пер. с англ. / Томас Коннолли, Каролин Бегг. – 3-е изд. – М. : Вильямс, 2003. – 1436 с.**
5. Кузнецов, С. Д. Основы баз данных : учеб. пособие / С. Д. Кузнецов. – 2-е изд., испр. – М. : Интернет-Университет Информационных Технологий ; БИНОМ. Лаборатория знаний, 2007. – 484 с.
6. Лузанов, П. PostgreSQL для начинающих / П. Лузанов, Е. Рогов, И. Лёвшин ; Postgres Professional. – М., 2017. – 146 с.
7. Моргунов, Е. П. Язык SQL. Базовый курс : учеб.-практ. пособие. / Е. П. Моргунов ; под ред. Е. В. Рогова, П. В. Лузанова ; Postgres Professional. – М., 2017. – 257 с.
8. PostgreSQL [Электронный ресурс] : официальный сайт / The PostgreSQL Global Development Group. – http://www.postgresql.org.
9. Postgres Professional [Электронный ресурс] : российский производитель СУБД Postgres Pro : официальный сайт / Postgres Professional. – http://postgrespro.ru.

Задание

Для выполнения практических заданий необходимо использовать книгу:

Моргунов, Е. П. Язык SQL. Базовый курс : учеб.-практ. пособие / Под ред. Е. В. Рогова, П. В. Лузанова ; Postgres Professional. – М., 2017. – 257 с. https://postgrespro.ru/education/books/sqlprimer

1. Изучить материал главы 8. Запросы к базе данных выполнять с помощью утилиты psql, описанной в главе 2, параграф 2.2.