Теория баз данных

Лекция 8. Методология проектирования баз данных

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

Версия от 3.05.2018

8.1. Введение

Проектирование базы данных является одним из этапов жизненного цикла разработки системы с базами данных.

- Планирование БД
- Определение системы
- Сбор и анализ требований
- Проектирование БД
  - Концептуальное проектирование
  - Логическое проектирование
  - Физическое проектирование
- Прототипирование (и новая итерация проектирования БД)
- Проектирование приложений (параллельно с проектированием БД)
- Реализация
- Преобразование и загрузка данных
- Тестирование
- Функционирование

8.1. Введение (продолжение)

- **Методология проектирования.** Структурированный подход, предусматривающий использование специализированных процедур, технических приемов, инструментов и документации, и предназначенный для поддержки и упрощение процесса проектирования.
- Методология проектирования предусматривает разбиение всего процесса на несколько этапов, каждый из которых, в свою очередь, состоит из нескольких стадий.
- На каждой стадии разработчику предлагается набор технических приемов, позволяющих решать задачи, стоящие перед ним на данной стадии разработки.
- Кроме того, методология предлагает методы планирования, координации, управления, оценки хода разработки проекта, а также структурированный подход к анализу и моделированию всего набора требований, предъявляемых к базе данных, и позволяет выполнить эти действия стандартизированным и организованным образом.

8.1. Введение (продолжение)

- В предлагаемой методологии весь процесс проектирования базы данных подразделяется на три основных стадии:
  - концептуальное проектирование;
  - логическое проектирование;
  - физическое проектирование.
- **Концептуальное проектирование** – процесс конструирования модели данных, используемых на предприятии, независимой от всех деталей реализации.
- Концептуальная модель не зависит от
  - СУБД
  - прикладных программ
  - языков программирования
  - аппаратной платформы
  - вопросов производительности и др.
- Строится на основе требований пользователей.

8.1. Введение (продолжение)

- **Логическое проектирование** – процесс конструирования модели данных, используемых на предприятии, на основе конкретной модели данных, но независимо от конкретной СУБД и других деталей физической реализации.
- Концептуальная модель отображается на логическую на основе выбора целевой модели данных, например, реляционной. Мы знаем тип СУБД (реляционная, сетевая, иерархическая, объектно-ориентированная), но не знаем деталей физической реализации хранения данных, индексов и др.
- **Физическое проектирование** – создание описания конкретной реализации базы данных, размещаемой во внешней памяти в среде целевой СУБД.
- Описываются базовые отношения (base relations), индексы, ограничения целостности.

- Проектирование БД – это итерационный процесс.

8.2. Обзор методологии проектирования

Концептуальное проектирование базы данных

Шаг 1. Построить концептуальную модель данных.

Шаг 1.1. Идентифицировать типы сущностей.

Шаг 1.2. Идентифицировать типы связей.

Шаг 1.3. Идентифицировать атрибуты и связать их с типами сущностей или связей.

Шаг 1.4. Определить домены атрибутов.

Шаг 1.5. Определить атрибуты потенциальных, первичных и альтернативных ключей.

Шаг 1.6. Рассмотреть необходимость использования расширенных концепций моделирования (необязательный шаг).

Шаг 1.7. Проверить модель на предмет избыточности.

Шаг 1.8. Проверить концептуальную модель относительно реализуемости пользовательских транзакций.

Шаг 1.9. Обсудить концептуальную модель с пользователями.

8.2. Обзор методологии проектирования (продолжение)

Логическое проектирование базы данных для реляционной модели

Шаг 2. Построить логическую модель данных.

Шаг 2.1. Создать отношения для логической модели данных.

Шаг 2.2. Проверить отношения с помощью правил нормализации.

Шаг 2.3. Проверить соответствие отношений требованиям пользовательских транзакций.

Шаг 2.4. Проверить ограничения целостности данных.

Шаг 2.5. Обсудить разработанную логическую модель данных с пользователями.

Шаг 2.6. Слияние локальных логических моделей данных в глобальную модель (необязательный шаг).

Шаг 2.7. Проверить возможность расширения модели в будущем.

8.2. Обзор методологии проектирования (продолжение)

Физическое проектирование базы данных для реляционной СУБД

Шаг 3. Перенести логическую модель в среду целевой СУБД.

Шаг 3.1. Спроектировать базовые отношения.

Шаг 3.2. Спроектировать представление производных (derived) данных.

Шаг 3.3. Спроектировать общие ограничения.

Шаг 4. Спроектировать организацию файлов и индексов.

Шаг 4.1. Проанализировать транзакции.

Шаг 4.2. Выбрать способы организации файлов.

Шаг 4.3. Выбрать индексы.

Шаг 4.4. Оценить потребность в дисковой памяти.

Шаг 5. Спроектировать представления пользователей.

Шаг 6. Спроектировать механизмы защиты.

Шаг 7. Рассмотреть необходимость введения контролируемой избыточности.

Шаг 8. Организация мониторинга и настройка функционирования работающей системы.

8.3. Концептуальное проектирование

Шаг 1. Построить концептуальную модель данных.

- Цель: создать концептуальную модель на основе требований к данным предприятия.
- Концептуальная модель включает:
  - типы сущностей
  - типы связей
  - атрибуты и домены атрибутов
  - первичные и альтернативные ключи
  - ограничения целостности

Концептуальное моделирование поддерживается документацией, включая ER-диаграммы и словарь данных.

8.3. Концептуальное проектирование (продолжение)

Шаг 1.1. Идентифицировать типы сущностей.

- На основе пользовательских требований
- Реальные объекты в предметной области

- Проблемы:
  - Пользователи используют примеры и частные случаи, а не обобщенные термины.
  - Вместо наименований должностей используют имена конкретных людей.
  - Трудно решить, считать ли конкретный объект сущностью, связью или атрибутом.
- После выделения каждой сущности ей следует присвоить определенное осмысленное имя, которое обязательно должно быть понятно пользователям. Выбранное имя и описание сущности помещается в словарь данных. Если это возможно, следует установить и внести в документацию данные об ожидаемом количестве экземпляров каждой сущности. Если сущность известна пользователям под разными именами, все дополнительные имена рекомендуется определить как синонимы или псевдонимы и также занести в словарь данных.

8.3. Концептуальное проектирование (продолжение)

Шаг 1.1. Идентифицировать типы сущностей (продолжение).

- **Пример словаря данных с описаниями сущностей**

Имя сущности	Описание	Псевдонимы	Частота
Staff	Общее обозначение для всех сотрудников компании *DreamHome*	Employee	Каждый сотрудник компании работает в одном конкретном отделении
PropertyForRent	Общее обозначение для всех объектов недвижимости	Property	Каждый объект недвижимости имеет одного владельца и передается в аренду в определенном отделении компании, в котором этим объектам недвижимости управляет один сотрудник. Объект недвижимости, сдаваемый в аренду, осматривают многие клиенты, а один клиент берет в аренду на определенный период

8.3. Концептуальное проектирование (продолжение)

Шаг 1.2. Идентифицировать типы связей.

- Искать глаголы в требованиях пользователя
  - Staff *Manages* PropertyForRent
    (Сотрудник компании *управляет* Объектом недвижимости)
  - PrivateQwner *Owns* PropertyForRent
    (Владелец объекта недвижимости *владеет* Объектом недвижимости)
  - PropertyForRent *AssociatedWith* Lease
    (Объект недвижимости *сдается в аренду по* Договору аренды)
- Связи:
  - бинарные
  - множественные
  - рекурсивные (сущность «Работник» имеет атрибут «Код начальника»)
- Нужно использовать ER-диаграммы.
- Определить множественность (кратность) связей (1:M, M:N, 1:1).
- Проверить отсутствие ловушек типа «разветвление» и типа «разрыв».
- Проверить соблюдение требования об участии каждой сущности, по меньшей мере, в одной связи.

8.3. Концептуальное проектирование (продолжение)

Шаг 1.2. Идентифицировать типы связей (продолжение).

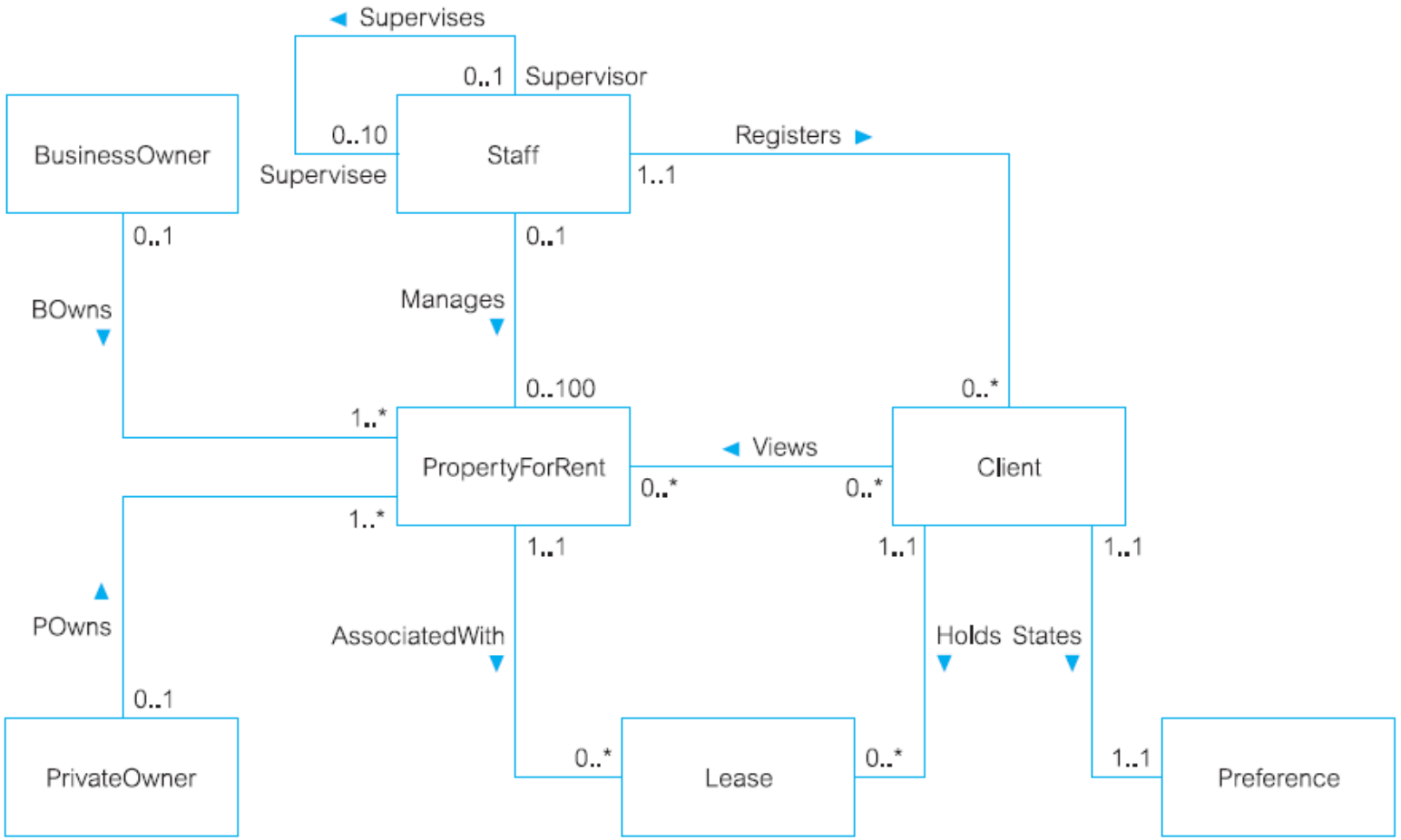
- **Пример словаря данных с описаниями типов связей**

Имя сущности	Кратность	Наименование связи	Кратность	Имя сущности
Staff	0..1	*Manages*	0..100	PropertyForRent
	0..1	*Supervises*	0..10	Staff
PropertyForRent	1..1	*AssociatedWith*	0..*	Lease

8.3. Концептуальное проектирование (продолжение)

Шаг 1.2. Идентифицировать типы связей (продолжение).

- Пример ER-диаграммы



8.3. Концептуальное проектирование (продолжение)

Шаг 1.3. Идентифицировать атрибуты и связать их с сущностями или связями.

- Простые и составные атрибуты
  - Определяем на основе требований пользователя. Например: «Адрес» может быть простым атрибутом, содержащим все элементы адреса. Но можно раздробить этот атрибут на компоненты: «Город», «Улица», «Дом», «Квартира».
- Порожденные (производные) атрибуты
  - Их значения вычисляются на основе значений других атрибутов. Например: «Возраст» и «Дата рождения». Очень часто подобные атрибуты вообще не отображаются в концептуальной модели данных. Однако если производный атрибут показан в модели данных, следует непременно указать, что он является производным.

8.3. Концептуальное проектирование (продолжение)

Шаг 1.3. Идентифицировать атрибуты и связать их с сущностями или связями (продолжение).

- Однозначные и многозначные атрибуты
  - Например: сущность «Работник», многозначный атрибут «Номер телефона».
- Для каждого атрибута нужно записать:
  - имя и описание
  - тип данных и длину
  - все псевдонимы, под которыми он известен
  - простой/составной (если он составной, то указать все входящие в него атрибуты)
  - является ли многозначным
  - является ли вычисляемым (и как это делается)
  - значение по умолчанию (если есть)

8.3. Концептуальное проектирование (продолжение)

Шаг 1.3. Идентифицировать атрибуты и связать их с сущностями или связями (продолжение).

- **Пример словаря данных с описаниями атрибутов**

Имя сущности	Атрибуты	Описание	Тип данных и длина	Пустые значения	Многозначный	. . .
Staff	staffNo	Однозначно определяет сотрудника компании	Строка длиной до 5 символов	Нет	Нет	
	name fName lName	Имя сотрудника компании Фамилия сотрудника компании	Строка длиной до 15 символов Строка длиной до 15 символов	Нет Нет	Нет Нет	
	position	Наименование должности сотрудника компании	Строка длиной до 10 символов	Нет	Нет	

8.3. Концептуальное проектирование (продолжение)

Шаг 1.4. Определить домены атрибутов.

- Домен – некоторое множество значений, элементы которого выбираются для присвоения значений одному или нескольким атрибутам.

ПРИМЕРЫ

- Домен атрибута, включающий допустимые значения табельных номеров (staffNo). Он состоит из строк переменной длины, которые могут включать до пяти символов; первые два символа должны быть буквенными, а следующие — состоять из цифр от 1 до 3, представляющих собой числа от 1 до 999.
- Возможные значения атрибута sex сущности Staff, которые могут быть представлены как «M» или «F». Домен этого атрибута включает две односимвольные строки со значением «M» или «F».

8.3. Концептуальное проектирование (продолжение)

Шаг 1.4. Определить домены атрибутов (продолжение).

- Домены должны содержать следующие данные:
  - набор допустимых значений для атрибута;
  - сведения о размере и формате каждого из атрибутов.
- В доменах может быть указана и другая дополнительная информация, например сведения о допустимых операциях со значениями атрибутов, а также данные о том, какие атрибуты можно использовать для сравнения с другими атрибутами или при построении комбинаций из нескольких атрибутов.
- После определения доменов атрибутов их имена и характеристики помещаются в словарь данных. Одновременно обновляются записи словаря данных, относящиеся к атрибутам: в них заносятся имена назначенных каждому атрибуту доменов вместо обозначения типов данных и информации о размерности.

8.3. Концептуальное проектирование (продолжение)

Шаг 1.5. Определить атрибуты потенциальных, первичных и альтернативных ключей.

- *Потенциальным ключом* называется атрибут или минимальный набор атрибутов заданной сущности, позволяющий однозначно идентифицировать каждый ее экземпляр.
- Для некоторых сущностей возможно наличие нескольких потенциальных ключей. В этом случае среди них нужно выбрать один ключ, который будет называться *первичным ключом.* Все остальные потенциальные ключи будут называться *альтернативными ключами.*
- Рекомендации по выбору ключей:
  - минимальное число атрибутов;
  - мала вероятность изменения значений атрибутов ключа;
  - малая длина текстовых значений, входящих в состав ключа;
  - малое наибольшее значение (для числовых атрибутов ключа);
  - ключ-кандидат, наиболее удобный с точки зрения пользователя.

8.3. Концептуальное проектирование (продолжение)

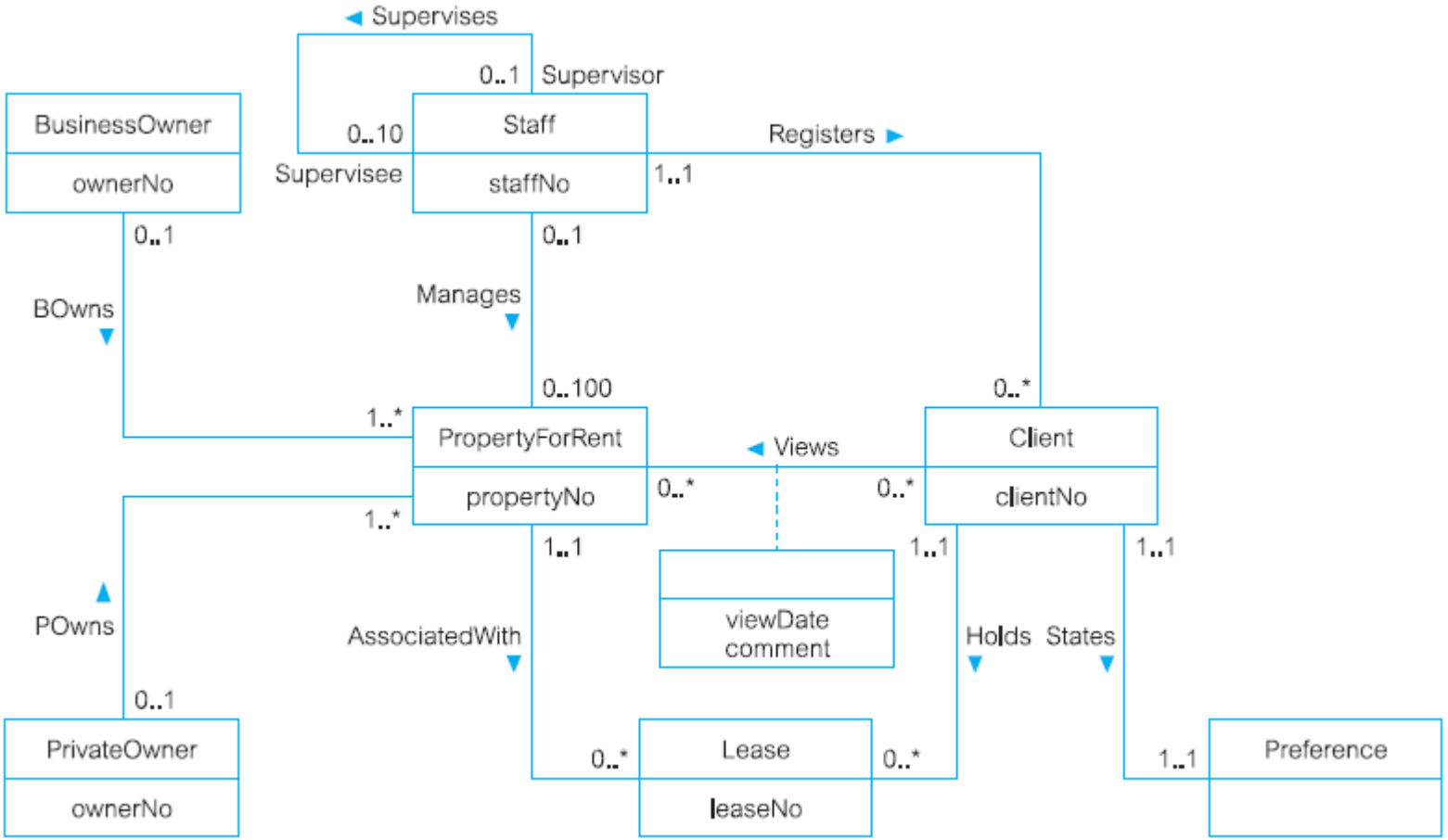
Шаг 1.5. Определить атрибуты потенциальных, первичных и альтернативных ключей (продолжение).

- В процессе определения первичного ключа устанавливается, является ли данная сущность сильной или слабой.
- Если выбрать первичный ключ для данной сущности оказалось возможным, то такую сущность принято называть *сильной.* И наоборот, если выбрать первичный ключ для заданной сущности невозможно, то ее называют *слабой.*
- Определение первичных ключей для слабых сущностей может быть выполнено только на стадии логического проектирования.
- После выбора первичных и альтернативных ключей сущностей (если таковые определены) сведения о них необходимо поместить в словарь данных.

8.3. Концептуальное проектирование (продолжение)

Шаг 1.5. Определить атрибуты потенциальных, первичных и альтернативных ключей (продолжение).

- ER-диаграмма после добавления первичных ключей



8.3. Концептуальное проектирование (продолжение)

Шаг 1.6. Рассмотреть необходимость использования расширенных концепций моделирования (необязательный шаг).

- **Цель.** Рассмотреть необходимость использования таких расширенных понятий моделирования, как уточнение/обобщение, агрегирование и композиция.
- Подход, требующий обобщения, предусматривает необходимость выявить общие особенности разных сущностей для определения обобщающей сущности суперкласса. Агрегирование может применяться для обозначения связи «has-a» (включает) или «is-part-of» (входит в состав) между типами сущностей; в такой связи одна сущность представляет «целое», а другая — ее «часть». Композиция (особый тип агрегирования) может применяться для определения взаимосвязи между типами сущностей, которая обусловливает строгую принадлежность и совпадение срока существования между «целым» и «частью».

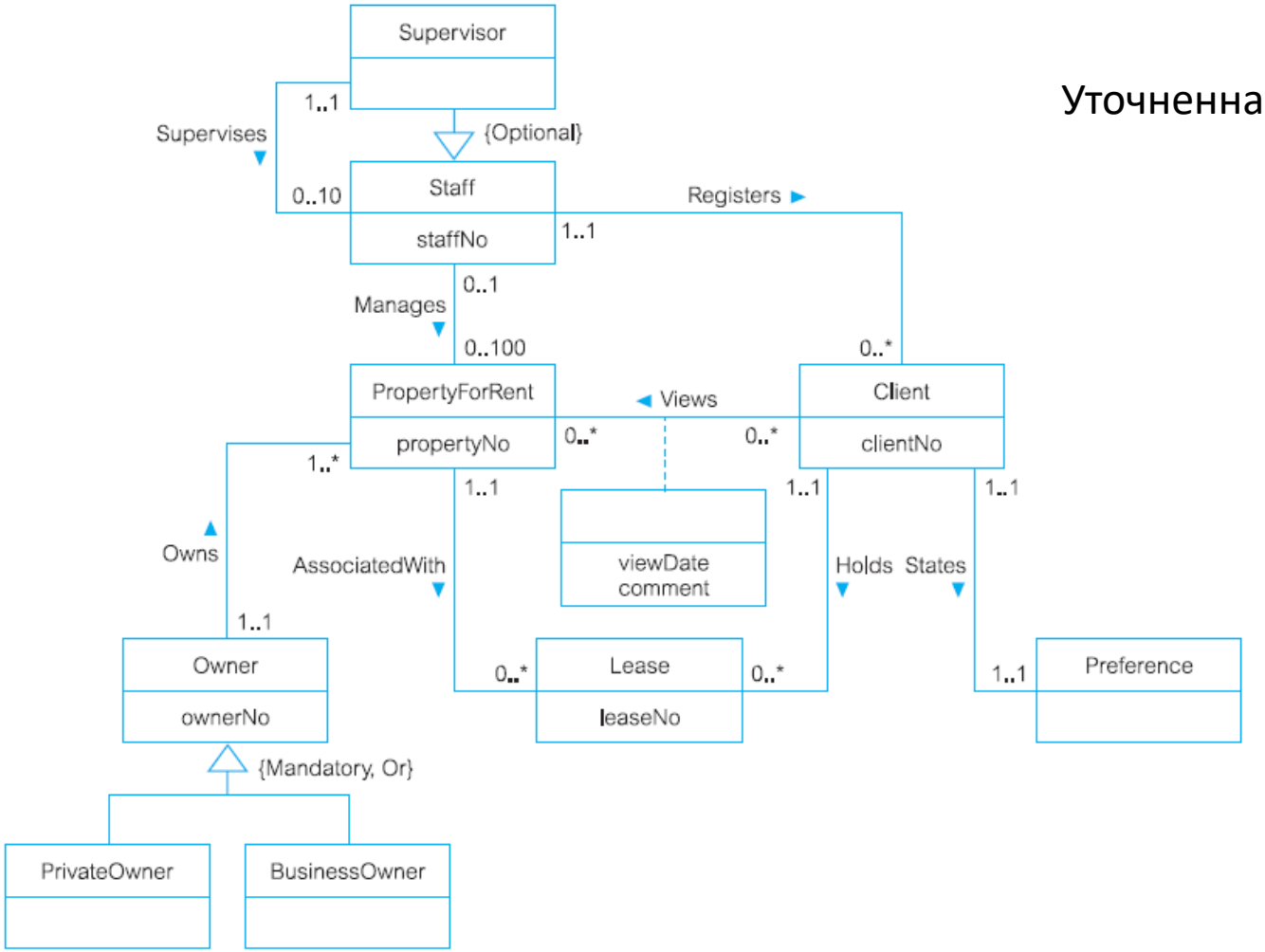
8.3. Концептуальное проектирование (продолжение)

Шаг 1.6. Рассмотреть необходимость использования расширенных концепций моделирования (необязательный шаг) (продолжение).

- Для представления Staff учебного проекта *DreamHome* было решено обобщить две сущности (PrivateOwner и BusinessOwner) для создания суперкласса Owner, содержащего общие атрибуты ownerNo, address и telNo. Связь между суперклассом Owner и его подклассами определяется как обязательная и непересекающаяся (такая связь обозначается как {Mandatory, Or}, поскольку каждый элемент суперкласса Owner должен быть элементом одного из подклассов, но не может одновременно входить в состав обоих подклассов.
- Кроме того, определен один подкласс, являющийся уточнением класса Staff (а именно, Supervisor), который предназначен исключительно для моделирования связи Supervises (Руководит). Связь между суперклассом Staff и подклассом Supervisor является необязательной, поскольку элемент суперкласса Staff не обязательно должен быть элементом подкласса Supervisor.

8.3. Концептуальное проектирование (продолжение)

Шаг 1.6. Рассмотреть необходимость использования расширенных концепций моделирования (необязательный шаг) (продолжение).



Уточненная ER-диаграмма

8.3. Концептуальное проектирование (продолжение)

Шаг 1.7. Проверить модель на предмет избыточности.

- Проверить связи типа 1:1.
- Возможно, что в процессе определения сущностей были обнаружены две сущности, которые соответствуют в данной организации одному и тому же концептуальному объекту. Например, допустим, что обнаружены две сущности (Client и Renter), которые фактически являются одинаковыми; иными словами, Client — это синоним для Renter. В таком случае эти две сущности должны быть объединены.
- Удалить избыточные связи.
- Связь является избыточной, если представленная в ней информация может быть получена с помощью других связей. Разработчик стремится создать минимальную модель данных, а поскольку избыточные связи не нужны, они должны быть удалены. На ER-диаграмме наличие избыточных связей можно относительно легко обнаружить, поскольку оно проявляется в том, что между двумя сущностями имеется несколько путей. Но такая ситуация не всегда означает, что одна из связей является избыточной, так как они могут представлять разные ассоциации между сущностями.

8.3. Концептуальное проектирование (продолжение)

Шаг 1.8. Проверить концептуальную модель относительно реализуемости пользовательских транзакций.

- Для этого должна быть предпринята попытка выполнить все необходимые операции вручную с помощью данной модели.
- Если все транзакции удалось выполнить таким образом, то проверка соответствия концептуальной модели данных требуемым транзакциям считается успешной.
- Но если невозможно провести вручную все транзакции, это означает, что модель данных содержит дефекты, которые должны быть устранены. В таком случае, вероятно, в модели данных не учтены какие-либо сущности, связи или атрибуты.
- Существует два возможных способа проверки того, что локальная концептуальная модель данных поддерживает требуемые транзакции.

1. Описание транзакций.
2. Проверка с применением путей выполнения транзакций.

8.3. Концептуальное проектирование (продолжение)

Шаг 1.8. Проверить концептуальную модель относительно реализуемости пользовательских транзакций (продолжение).

1. Описание транзакции.

Сформировать в виде списка подробные сведения об объектах недвижимости, которыми управляет указанный сотрудник отделения.

- Сведения об объектах недвижимости хранятся в сущности PropertyForRent, а сведения о сотрудниках, которые управляют объектами недвижимости, — в сущности Staff. В данном случае для получения требуемых сведений можно использовать связь Staff *Manages* PropertyForRent.

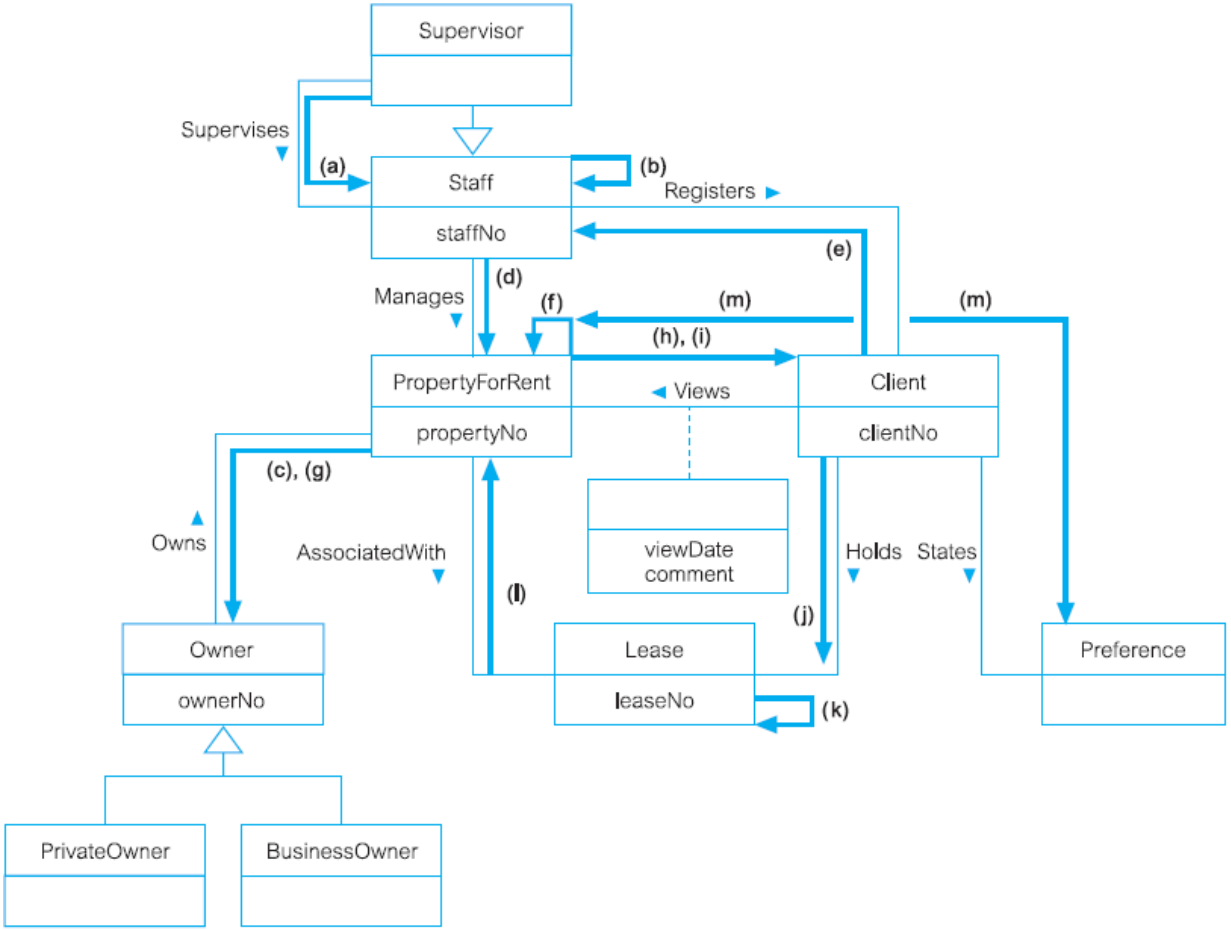
2. Проверка с применением путей выполнения транзакций.

- При этом подходе на ER-диаграмме стрелками соединяются те сущности, которые должны участвовать в каждой конкретной транзакции. если в модели обнаруживаются области, которые, по-видимому, не используются в каких-либо транзакциях, то можно еще раз проверить, должна ли эта информация быть представлена в модели данных. С другой стороны, если некоторые области модели не позволяют предоставить правильный путь выполнения транзакции, то. возможно, придется еще раз убедиться в том, что не пропущены какие-либо важные сущности, связи или атрибуты.

8.3. Концептуальное проектирование (продолжение)

Шаг 1.8. Проверить концептуальную модель относительно реализуемости пользовательских транзакций (продолжение).

2. Проверка с применением путей выполнения транзакций (продолжение).



8.3. Концептуальное проектирование (продолжение)

Шаг 1.9. Обсудить концептуальную модель с пользователями.

- **Цель.** Обсуждение локальных концептуальных моделей данных с конечными пользователями с целью подтверждения того, что данная модель полностью соответствует спецификации требований пользовательского представления.

8.4. Логическое проектирование для реляционной модели данных

Шаг 2. Построить логическую модель данных.

- **Цель.** Преобразовать концептуальную модель данных в логическую модель данных и затем проверить ее на предмет структурной корректности и способности поддерживать требуемые транзакции.
- **Логическое проектирование баз данных.** Процесс конструирования общей информационной модели на основе моделей данных отдельных пользователей. Эта модель является независимой от особенностей реально используемой СУБД и других физических условий.
- Если при проектировании базы данных используется всего одно представление пользователя или несколько представлений, но с применением централизованного подхода для управления ими, тогда шаг 2.6 опускается.
- Если же используется несколько различных представлений пользователей с применением подхода интеграции представлений, тогда шаги 2.1 – 2.5 повторяются для требуемого числа моделей данных, каждая из которых соответствует представлениям различных пользователей о базе данных. А на шаге 2.6 эти модели данных объединяются.

8.4. Логическое проектирование (продолжение)

Шаг 2.1. Создать отношения для логической модели данных.

- **Сильные типы сущностей**
  - Для каждой из них создать отношение, включающее все простые атрибуты. Составные атрибуты разбить на простые.
- **Слабые типы сущностей**
  - Для каждой из них создать отношение, включающее все простые атрибуты. Первичный ключ слабой сущности полностью или частично определяется на основе сильной сущности-владельца.

8.4. Логическое проектирование (продолжение)

Шаг 2.1. Создать отношения для логической модели данных (продолжение).

- **Бинарные связи типа 1:M (родительская и дочерняя сущности)**
  - Сущность, находящаяся на стороне «1» считается родительской, а сущность на стороне «M» – дочерней.
  - Копия атрибутов первичного ключа родительской сущности помещается в отношение, реализующее дочернюю сущность, выполняя роль внешнего ключа.
  - Если связь также имеет атрибуты, то их тоже нужно поместить в дочернее отношение.
  - Например. Для реализации связи Staff *Registers* Client атрибут staffNo первичного ключа отношения Staff помещается в отношение Client, где он выполняет роль внешнего ключа.

8.4. Логическое проектирование (продолжение)

Шаг 2.1. Создать отношения для логической модели данных (продолжение).

- **Бинарные связи типа 1:1**

  В этом случае кардинальность нельзя использовать для определения родительской и дочерней сущностей. Для определения этого используется ограничение «участие», или «членство».

- Обязательное членство на обеих сторонах связи
  - Обе сущности нужно объединить в одно отношение. В качестве первичного ключа выбрать любой из первичных ключей. Второй из них станет альтернативным ключом.
  - Пример: связь Client *States* Preference.

8.4. Логическое проектирование (продолжение)

Шаг 2.1. Создать отношения для логической модели данных (продолжение).

- **Бинарные связи типа 1:1 (продолжение)**
- Обязательное членство на одной стороне связи
  - В качестве родительской сущности выступает та, которая имеет необязательное членство в этой связи, а в качестве дочерней – та, которая имеет обязательное членство.
  - Копия первичного ключа родительской сущности помещается в отношение, представляющее дочернюю сущность.
  - Пример: Если бы в связи Client *States* Preference было бы необязательное участие на стороне сущности Client (т. е. не каждый клиент указывает свои предпочтения), тогда сущность Client была бы родительской, а сущность Preference – дочерней. Тогда копию атрибутов первичного ключа сущности Client (т. е. clientNo) следовало бы поместить в отношение Preference.

8.4. Логическое проектирование (продолжение)

Шаг 2.1. Создать отношения для логической модели данных (продолжение).

- **Бинарные связи типа 1:1 (продолжение)**
- Необязательное членство на обеих сторонах связи
  - Определить родительскую и дочернюю сущности не всегда возможно, поэтому копию первичного ключа одной сущности можно поместить в отношение, представляющее другую сущность.
  - Пример. Связь Staff *Uses* Car.

8.4. Логическое проектирование (продолжение)

Шаг 2.1. Создать отношения для логической модели данных (продолжение).

- **Рекурсивные бинарные связи типа 1:1**
- Обязательное членство на обеих сторонах связи
  - Обе сущности нужно объединить в одно отношение. В качестве первичного ключа выбрать любой из первичных ключей. Второй ключ нужно переименовать и использовать в качестве внешнего ключа.
- Обязательное членство на одной стороне связи
  - Можно либо создать объединенное отношение, включающее две копии первичного ключа, либо создать новое отношение, представляющее связь. Оно будет иметь в составе только две копии первичного ключа, которые будут являться внешними ключами. Одну из копий нужно переименовать.
- Необязательное членство на обеих сторонах связи
  - Создать новое отношение, представляющее связь, как в предыдущем случае.

8.4. Логическое проектирование (продолжение)

Шаг 2.1. Создать отношения для логической модели данных (продолжение).

- **Связи типа суперкласс/подкласс**
- Сущность, соответствующая суперклассу, будет родительской, а сущность, соответствующая подклассу, – дочерней.
- Имеется также возможность преобразовать подобную связь в одно или несколько отношений.
- Выбор наиболее подходящего способа преобразования зависит от многих факторов, таких как:
  - ограничения непересечения и степени участия, которые распространяются на связь типа суперкласс/подкласс;
  - от того, участвуют ли подклассы в разных связях;
  - а также от количества сущностей, участвующих в связи суперкласс/подкласс.

8.4. Логическое проектирование (продолжение)

Шаг 2.1. Создать отношения для логической модели данных (продолжение).

- **Связи типа суперкласс/подкласс (продолжение)**
- Рекомендации по выбору способа преобразования связи суперкласс/подкласс с учетом только ограничений степени участия и непересечения

Ограничение степени участия	Ограничение непересечения	Необходимые отношения
Обязательное {Mandatory}	Пересечение до-пускается {And}	Одно отношение (с одним или несколькими определителями, позволяющими указать тип каждой записи)
Необязательное {Optional}	Пересечение до-пускается {And}	Два отношения: одно —для суперкласса, а второе—для всех подклассов (с одним или несколькими определителями, позволяющими указать тип каждой записи)

8.4. Логическое проектирование (продолжение)

Шаг 2.1. Создать отношения для логической модели данных (продолжение).

- **Связи типа суперкласс/подкласс (продолжение)**
- Рекомендации по выбору способа преобразования связи суперкласс/подкласс с учетом только ограничений степени участия и непересечения

Ограничение степени участия	Ограничение непересечения	Необходимые отношения
Обязательное {Mandatory}	Пересечение не допускается {Or}	Несколько отношений: по одному отношению для каждого сочетания суперкласс/подкласс
Необязательное {Optional}	Пересечение не допускается {Or}	Несколько отношений: одно —для суперкласса, а другие— по одному для каждого подкласса

8.4. Логическое проектирование (продолжение)

Шаг 2.1. Создать отношения для логической модели данных (продолжение).

- **Связи типа суперкласс/подкласс (продолжение)**
- ПРИМЕР. Различные варианты преобразования связи суперкласс/подкласс Owner, основанные на использовании ограничений степени участия и непересечения.
- Вариант 1 – обязательное участие, пересечение допускается
  - AllOwner (ownerNo, address, telNo, fName, INarae, bName, toType, contactName, pOwnerFlag, bownerFlag)
  - Первичный ключ ownerNo
- Вариант 2 – необязательное участие, пересечение допускается
  - Owner (ownerNo, address, telNo)
  - Первичный ключ ownerNo
  - OwnerDetails (ownerNo, fName IName, bName, bType, contactName, pOwnerFlag, bOwnerFlag)
  - Первичный ключ ownerNo
  - Внешний ключ ownerNo ссылается на Owner(ownerNo)

8.4. Логическое проектирование (продолжение)

Шаг 2.1. Создать отношения для логической модели данных (продолжение).

- **Связи типа суперкласс/подкласс (продолжение)**
- ПРИМЕР (продолжение). Различные варианты преобразования связи суперкласс/подкласс Owner, основанные на использовании ограничений степени участия и непересечения.
- Вариант 3 – обязательное участие, пересечение не допускается
  - PrivateOwner (ownerNo, fName, lName, address, telNo)
  - Первичный ключ ownerNo
  - BusinessOwner (ownerNo, bName, bType, contactName, address, telNo}
  - Первичный ключ ownerNo

8.4. Логическое проектирование (продолжение)

Шаг 2.1. Создать отношения для логической модели данных (продолжение).

- **Связи типа суперкласс/подкласс (продолжение)**
- ПРИМЕР (продолжение). Различные варианты преобразования связи суперкласс/подкласс Owner, основанные на использовании ограничений степени участия и непересечения.
- Вариант 4 – необязательное участие, пересечение не допускается
  - Owner (ownerNo, address, telNo)
  - Первичный ключ ownerNo
  - PrivateOwner (ownerNo, fName, lName}
  - Первичный ключ ownerNo
  - Внешний ключ ownerNo ссылается на Owner(ownerNo)
  - BusinessOwner (ownerNo, bName, bType, contactName)
  - Первичный ключ ownerNo
  - Внешний ключ ownerNo ссылается на Owner(ownerNo)

8.4. Логическое проектирование (продолжение)

Шаг 2.1. Создать отношения для логической модели данных (продолжение).

- **Бинарные связи типа M:N**
  - Создать отношение, представляющее связь, и включить в него все атрибуты, описывающие эту связь.
  - Поместить атрибуты, являющиеся первичными ключами сущностей, участвующих в связи, в новое отношение. Они будут внешними ключами.
  - Один или оба этих внешних ключа будут образовывать первичный ключ нового отношения, возможно, с добавлением атрибутов, описывающих саму связь.

8.4. Логическое проектирование (продолжение)

Шаг 2.1. Создать отношения для логической модели данных (продолжение).

- **Бинарные связи типа M:N (продолжение)**
- ПРИМЕР. Связь Client *Views* PropertyPorRent типа. Для отображения этой связи создаются отношения с сильными сущностями Client и PropertyForRent, а также отношение Viewing, соответствующее связи Views.

Client (clientNo, fName, lName, telNo, eMail, prefType, maxRent, staffNo)
Primary Key clientNo
Alternate Key eMail
Foreign Key staffNo **references** Staff(staffNo)

PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent)
Primary Key propertyNo

Viewing (clientNo, propertyNo, dateView, comment)
Primary Key clientNo, propertyNo
Foreign Key clientNo **references** Client(clientNo)
Foreign Key propertyNo **references** PropertyForRent(propertyNo)

8.4. Логическое проектирование (продолжение)

Шаг 2.1. Создать отношения для логической модели данных (продолжение).

- **Сложные типы связей**
- Для каждой сложной связи создается отношение, отображающее эту связь, и в него включаются все атрибуты, входящие в состав рассматриваемой связи.
- В новое отношение передаются для использования в качестве внешних ключей копии атрибутов первичного ключа сущностей, участвующих в сложной связи.
- Все внешние ключи, соответствующие стороне связи «многие» (например, 1..*, 0..*), как правило, образуют также первичный ключ этого нового отношения, возможно, в сочетании с некоторыми другими атрибутами связи.

8.4. Логическое проектирование (продолжение)

Шаг 2.1. Создать отношения для логической модели данных (продолжение).

- **Многозначные атрибуты**
  - Для каждого многозначного атрибута, имеющегося в какой-то сущности, создать новое отношение, представляющее этот атрибут. Включить первичный ключ этой сущности в новое отношение, он будет служить в качестве внешнего ключа.
- ПРИМЕР. В представлении Branch в соответствии с той ситуацией, что каждое отделение может иметь до трех номеров телефонов, атрибут telNo сущности Branch был определен как многозначный.

Branch (branchNo, street, city, postcode)
Primary Key branchNo

Telephone (telNo, branchNo)
Primary Key telNo
Foreign Key branchNo **references** Branch(branchNo)

8.4. Логическое проектирование (продолжение)

Шаг 2.2. Проверить отношения с помощью правил нормализации.

- Идентифицировать функциональные зависимости между атрибутами (чтобы это сделать, нужно понять суть каждого атрибута)
- Для избежания проблем, связанных с избыточностью данных, требуется привести каждое отношение хотя бы к 3NF. Заметим, что процесс формирования отношений из концептуальной модели данных должен порождать отношения, которые сразу находятся в 3НФ.

Шаг 2.3. Проверить соответствие отношений требованиям пользовательских транзакций.

- Это делается вручную, используя отношения, первичные и внешние ключи, ER-диаграммы

8.4. Логическое проектирование (продолжение)

Шаг 2.4. Проверить ограничения целостности данных.

- Ограничения целостности служат для того, чтобы данные не стали неполными, неточными или несогласованными.
- Здесь идет речь о том, какие ограничения целостности нужны, а не о том, как этого достичь.

- **Ограничения целостности бывают следующие:**
- Обязательное наличие допустимых значений
  - Не допускаются значения NULL. Например: каждый работник должен иметь конкретную должность.
- Ограничения домена
  - Каждый атрибут имеет множество допустимых значений. Например: пол может иметь два значения – мужской и женский.
- Множественность связей
  - Например: в каждом отделе предприятия могут работать много работников, но каждый работник работает только в одном отделе.
- Целостность сущностей
  - Первичные ключи не могут содержать значений NULL.

8.4. Логическое проектирование (продолжение)

Шаг 2.4. Проверить ограничения целостности данных (продолжение).

- **Ограничения целостности бывают следующие (продолжение) :**
- Ссылочная целостность
  - Внешний ключ связывает каждую запись в подчиненном отношении с родительской записью в родительском отношении, имеющей соответствующее значение первичного или уникального ключа.
  - Например: отношения «Студенты» и «Успеваемость» связаны по атрибуту «Номер зачетной книжки». Тогда номер зачетной книжки в записях, относящихся к данному студенту и находящихся в отношении «Успеваемость», должен совпадать с номером зачетной книжки в записи об этом студенте, находящейся в отношении «Студенты»

8.4. Логическое проектирование (продолжение)

Шаг 2.4. Проверить ограничения целостности данных (продолжение).

- **Ограничения целостности бывают следующие (продолжение) :**
- Ссылочная целостность (продолжение)

  Проблемы:

  1. Допускаются ли значения NULL в атрибутах внешнего ключа? Если класс членства подчиненной сущности обязательный, тогда не допускаются. Если класс членства необязательный, то допускаются.

  2. Как обеспечить ссылочную целостность? Для связи вида 1:M могут иметь место следующие случаи:
  - вставка записи в дочернее отношение
  - удаление записи из дочернего отношения
  - обновление внешнего ключа в дочернем отношении
  - вставка записи в родительское отношение
  - удаление записи из родительского отношения (стратегии: NO ACTION, CASCADE, SET NULL, SET DEFAULT)
  - обновление первичного ключа в родительском отношении (те же стратегии)

8.4. Логическое проектирование (продолжение)

Шаг 2.4. Проверить ограничения целостности данных (продолжение).

- **Ограничения целостности бывают следующие (продолжение) :**
- Общие ограничения (бизнес-правила)
  - Например: число подчиненных у руководителя не может превышать некоторого предела.

8.4. Логическое проектирование (продолжение)

Шаг 2.5. Обсудить разработанную логическую модель данных с пользователями.

- Если пользователи не считают полученную модель соответствующей тем требованиям, которые были предъявлены к данным предприятия, тогда необходимо повторить некоторые предшествующие шаги

Шаг 2.6. Слияние локальных логических моделей данных в глобальную модель (необязательный шаг).

- Не выполняется, если имеется только одно представление пользователя о данных предприятия или для построения глобальной логической модели был выбран централизованный подход, при котором требования отдельных пользователей сначала объединяются в единый документ, а затем выполняется построение модели.
- Выполняется, если был выбран подход интеграции представлений пользователей о базе данных, когда логические модели отдельных пользователей объединяются в единую логическую модель.

8.4. Логическое проектирование (продолжение)

Шаг 2.6. Слияние локальных логических моделей данных в глобальную модель (необязательный шаг) (продолжение).

Шаг 2.6.1. Объединить локальные логические модели пользователей в глобальную логическую модель.

- Пошаговый подход – добавление локальных логических моделей по одной за один раз.
- Нужно учитывать, что возможны случаи, когда
  - два отношения имеют одинаковые имена, но разную суть (омонимы)
  - два отношения имеют разные имена, но одинаковую суть (синонимы)

Шаг 2.6.2. Выполнить проверку глобальной логической модели.

- Уделить особое внимание только тем фрагментам модели, которые подверглись изменениям.

Шаг 2.6.3. Обсудить глобальную логическую модель с пользователями.

- После этого шага использование термина «локальная логическая модель данных» не требуется, поэтому используется только термин «логическая модель данных».

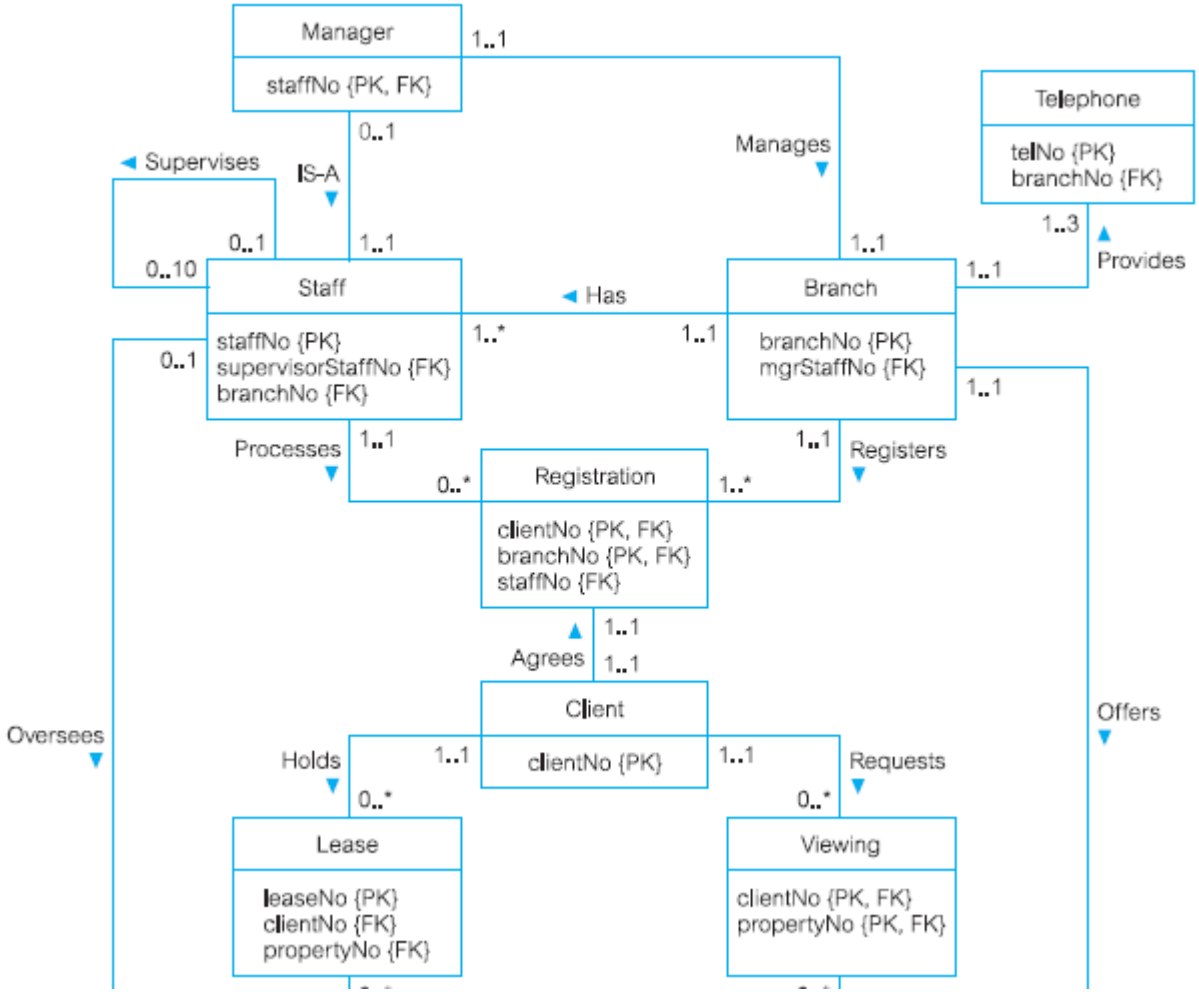
8.4. Логическое проектирование (продолжение)

Шаг 2.7. Проверить возможность расширения модели в будущем.

- Проектировать нужно по возможности так, чтобы логическая модель была расширяемой и могла дополняться с учетом возникновения новых требований бизнеса.

8.4. Логическое проектирование (продолжение)

- Фрагмент логической модели



8.5. Физическое проектирование реляционной базы данных

Результат стадии логического проектирования:

- Логическая модель данных, включающая ER-диаграммы (диаграммы отношений), реляционную схему и документацию, в т. ч. словарь данных.
- На стадии логического проектирования отвечают на вопрос ЧТО должно быть сделано.
- На стадии физического проектирования отвечают на вопрос КАК это должно быть сделано.
- **Физическое проектирование базы данных.** Процесс подготовки описания реализации базы данных во внешней памяти; описание должно включать основные отношения, файловую организацию, индексы, обеспечивающие эффективный доступ к данным, а также все соответствующие ограничения целостности и средства защиты.

8.5. Физическое проектирование (продолжение)

Шаг 3. Перенести логическую модель в среду целевой СУБД.

- **Цель.** Создание базовой функциональной схемы реляционной базы данных на основе глобальной логической модели данных, которая может быть реализована в целевой СУБД.

Шаг 3.1. Спроектировать базовые отношения.

Шаг 3.2. Спроектировать представление производных (derived) данных.

Шаг 3.3. Спроектировать общие ограничения.

8.5. Физическое проектирование (продолжение)

Шаг 3.1. Спроектировать базовые отношения.

- **Цель.** Определение способа представления в целевой СУБД отношений, определенных в глобальной логической модели данных.
- Определение каждого выделенного в глобальной логической модели данных отношения включает следующие элементы:
  - имя отношения;
  - список простых атрибутов, заключенный в круглые скобки;
  - определение первичного ключа и (если таковые существуют) альтернативных (АК) и внешних (FK) ключей;
  - список производных атрибутов и описание способов их вычисления;
  - определение требований ссылочной целостности для любых внешних ключей.

8.5. Физическое проектирование (продолжение)

Шаг 3.1. Спроектировать базовые отношения (продолжение).

- Для каждого атрибута в словаре данных должна присутствовать следующая информация:
  - определение его домена, включающее указание типа данных, размерность внутреннего представления атрибута и любые требуемые ограничения на допустимые значения;
  - принимаемое по умолчанию значение атрибута (необязательно);
  - допустимость значения NULL для данного атрибута.

Реализация базовых отношений

Это выполняется с помощью команды CREATE TABLE.

8.5. Физическое проектирование (продолжение)

Шаг 3.2. Спроектировать представление производных (derived) данных.

- **Цель.** Определение способа представления в целевой СУБД всех производных данных, которые включены в глобальную логическую модель данных.
- Производными, или вычисляемыми, называются атрибуты, значения которых можно определить с использованием значений других атрибутов. Например:
  - количество сотрудников, работающих в конкретном отделении;
  - общая сумма ежемесячной зарплаты всех сотрудников;
  - количество объектов недвижимости, находящихся под управлением определенного сотрудника компании.

8.5. Физическое проектирование (продолжение)

Шаг 3.2. Спроектировать представление производных (derived) данных (продолжение).

- Зачастую производные атрибуты не включают в логическую модель данных, а описывают в словаре данных.
- Если производный атрибут включен в модель, для указания на то, что он является производным, перед его именем ставится косая черта (/).
- На первом этапе проектирования изучается логическая модель данных и словарь данных, а также подготавливается список всех производных атрибутов.
- На этапе физического проектирования базы данных необходимо определить, должен ли производный атрибут храниться в базе данных или вычисляться каждый раз, когда в нем возникает необходимость.

8.5. Физическое проектирование (продолжение)

Шаг 3.2. Спроектировать представление производных (derived) данных (продолжение).

- Проектировщик должен рассчитать следующее:
  - дополнительные затраты на хранение производных данных и поддержание их согласованности с реальными данными, на основе которых они вычисляются;
  - затраты на вычисление производных данных, если их вычисление выполняется по мере необходимости.

  Из этих двух вариантов выбирается наименее дорогостоящий с учетом требований к производительности.

8.5. Физическое проектирование (продолжение)

Шаг 3.2. Спроектировать представление производных (derived) данных (продолжение).

ПРИМЕР. Добавим в отношение Staff дополнительный атрибут noOfProperties, обозначающий количество объектов недвижимости, которыми управляет в настоящее время каждый сотрудник компании.

Staff

staffNo	fName	lName	branchNo	noOfProperties
SL21	John	White	B005	0
SG37	Ann	Beech	B003	2
SG14	David	Ford	B003	1
SA9	Mary	Howe	B007	1
SG5	Susan	Brand	B003	0
SL41	Julie	Lee	B005	1

PropertyForRent

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

8.5. Физическое проектирование (продолжение)

Шаг 3.2. Спроектировать представление производных (derived) данных (продолжение).
ПРОДОЛЖЕНИЕ ПРИМЕРА.

- Дополнительные затраты памяти для этого нового производного атрибута не слишком велики.
- Значение атрибута обновляется каждый раз, когда под управление сотрудника компании передается объект недвижимости или отменяется его назначение для управления каким-то объектом; такие изменения происходят также при удалении объекта недвижимости из списка доступных объектов.
- Но в любом из этих случаев значение атрибута noOfProperties для соответствующего сотрудника компании должно быть увеличено или уменьшено на 1.

8.5. Физическое проектирование (продолжение)

Шаг 3.2. Спроектировать представление производных (derived) данных (продолжение).
ПРОДОЛЖЕНИЕ ПРИМЕРА.

- В процессе проектирования необходимо обеспечить, чтобы эти изменения происходили в каждом из указанных случаев и количество учитываемых объектов оставалось правильным, обеспечивая тем самым целостность базы данных.
- При обращении к этому атрибуту с помощью любого запроса его значение является непосредственно доступным и не требует вычисления.
- С другой стороны, если этот атрибут не хранился бы непосредственно в отношении Staff, то его значение приходилось бы вычислять каждый раз, когда оно потребуется. Для этого необходимо выполнить соединение отношений Staff и PropertyForRent.

8.5. Физическое проектирование (продолжение)

Шаг 3.2. Спроектировать представление производных (derived) данных (продолжение).
ПРОДОЛЖЕНИЕ ПРИМЕРА.

- Таким образом, если запрос указанного типа выполняется часто или считается очень важным с точки зрения производительности, производный атрибут более целесообразно хранить в базе данных, а не вычислять при каждом обращении к его значению.
- Решение, предусматривающее хранение производных атрибутов, является более приемлемым и в том случае, если язык запросов целевой СУБД не позволяет легко реализовать алгоритм вычисления производных атрибутов.
- Принятые решения необходимо задокументировать.

8.5. Физическое проектирование (продолжение)

Шаг 3.3. Спроектировать общие ограничения.

Цель. Реализация ограничений предметной области для целевой СУБД.

- Обновление информации в отношениях может регламентироваться ограничениями предметной области, регулирующими выполнение тех реальных транзакций, которые связаны с проведением таких обновлений.
- Способ реализации указанных ограничений опять-таки будет зависеть от типа выбранной целевой СУБД, поскольку одни системы для реализации ограничений предметной области предоставляют более широкие возможности, чем другие.

8.5. Физическое проектирование (продолжение)

Шаг 3.3. Спроектировать общие ограничения (продолжение).

- ПРИМЕР. В компании *DreamHome* существует правило, согласно которому каждый сотрудник может одновременно заниматься не более чем ста объектами недвижимости. Это ограничение, если СУБД позволяет, можно включить в оператор CREATE TABLE языка SQL для таблицы PropertyForRent с помощью следующей конструкции:

```
CONSTRAINT StaffNotHandlingTooMuch
    CHECK (NOT EXISTS (SELECT staffNo
                       FROM PropertyForRent
                       GROUP BY staffNo
                       HAVING COUNT(*) > 100))
```

8.5. Физическое проектирование (продолжение)

Шаг 3.3. Спроектировать общие ограничения (продолжение).

- Альтернативным методом реализации ограничений является применение триггеров.
- В некоторых системах отсутствует поддержка части или даже всех ограничений предметной области и поэтому такие ограничения приходится предусматривать непосредственно в программном коде самого приложения. Такой подход имеет недостатки.
- Все решения, принятые в связи с реализацией ограничений предметной области, должны быть полностью описаны в сопроводительной документации. Кроме того, в документации должны быть указаны причины выбора именно данного варианта из нескольких возможных.

8.5. Физическое проектирование (продолжение)

Шаг 4. Спроектировать организацию файлов и индексов.

- **Цель.** Определение оптимальной файловой организации для хранения базовых отношений, а также индексов, необходимых для достижения приемлемой производительности; иными словами, определение способа хранения отношений и строк во внешней памяти.

Шаг 4.1. Проанализировать транзакции.
Шаг 4.2. Выбрать способы организации файлов.
Шаг 4.3. Выбрать индексы.
Шаг 4.4. Оценить потребность в дисковой памяти.

8.5. Физическое проектирование (продолжение)

Шаг 4.1. Проанализировать транзакции.

- **Цель.** Определение функциональных характеристик транзакций, которые будут выполняться в проектируемой базе данных, и выделение наиболее важных из них.
- Для того чтобы разрабатываемый физический проект базы данных обладал требуемым уровнем эффективности, необходимо получить максимум сведений о тех транзакциях и запросах, которые будут выполняться в базе данных.

8.5. Физическое проектирование (продолжение)

Шаг 4.1. Проанализировать транзакции (продолжение).

- Для успешного планирования каждой транзакции необходимо знать следующее:
  - транзакции, выполняемые наиболее часто и оказывающие существенное влияние на производительность;
  - транзакции, наиболее важные для работы организации;
  - периоды времени на протяжении суток/недель, в которые нагрузка базы данных возрастает до максимума (называемые *периодами пиковой нагрузки).*
- Эта информация используется для определения компонентов базы данных, которые могут вызвать проблемы производительности.

8.5. Физическое проектирование (продолжение)

Шаг 4.1. Проанализировать транзакции (продолжение).

- Для выделения областей, которые с наибольшей вероятностью могут явиться источником проблем, необходимо выполнить перечисленные ниже действия.
  - Подготовка схемы соответствия путей выполнения транзакций и отношений.
  - Определение отношений, наиболее часто используемых при выполнении транзакций.
  - Анализ интенсивности доступа к данным в некоторых из транзакций, использующих эти отношения.

8.5. Физическое проектирование (продолжение)

Шаг 4.2. Выбрать способы организации файлов.

- **Цель.** Определение наиболее эффективного файлового представления для каждого из основных отношений.
- Одной из главных задач физического проектирования базы данных является обеспечение эффективного хранения данных.
- Во многих случаях реляционная СУБД может предоставлять лишь ограниченный выбор или даже вообще не позволять выбрать определенную файловую организацию.
- Если целевая СУБД не позволяет выбрать определенную файловую организацию, этот этап может быть пропущен.

8.5. Физическое проектирование (продолжение)

Шаг 4.3. Выбрать индексы.

- **Цель.** Определение того, будет ли добавление индексов способствовать повышению производительности системы.
- Индексы создаются по первичным и уникальным ключам.
- Для ускорения доступа к данным можно создавать дополнительные индексы.
- Сопровождение и применение дополнительных индексов приводит к увеличению издержек, поэтому необходимо определить, оправдывают ли они повышение производительности при выборке данных, достигнутое благодаря их использованию.

8.5. Физическое проектирование (продолжение)

Шаг 4.3. Выбрать индексы (продолжение).

- Основные издержки, связанные с применением дополнительных индексов, перечислены ниже.
  - Ввод индексной записи в каждый дополнительный индекс при вставке строки в отношение.
  - Обновление дополнительного индекса при обновлении соответствующей строки в отношении.
  - Увеличение потребности в дисковом пространстве в связи с необходимостью хранения дополнительного индекса.
  - Возможное снижение производительности процесса оптимизации запросов, поскольку оптимизатор запросов должен учесть наличие всех дополнительных индексов и только после этого выбрать оптимальную стратегию выполнения запросов.

8.5. Физическое проектирование (продолжение)

Шаг 4.3. Выбрать индексы (продолжение).

- **Рекомендации по созданию индексов.**

1. Не создавать индекс на небольших отношениях.
2. Ввести дополнительный индекс на внешнем ключе, если с его помощью часто происходит доступ к отношению.
3. Ввести дополнительные индексы на атрибутах, которые часто применяются в следующих конструкциях:
   - критерии выборки или соединения;
   - конструкции ORDER BY;
   - конструкции GROUP BY;
   - другие операции, требующие сортировки (такие как UNION и DISTINCT).

8.5. Физическое проектирование (продолжение)

Шаг 4.3. Выбрать индексы (продолжение).

- **Рекомендации по созданию индексов (продолжение).**

4. Ввести дополнительные индексы на атрибутах, применяемых во встроенных функциях агрегирования, наряду со всеми атрибутами, используемыми для этих встроенных функций.

   Например, чтобы определить среднюю зарплату сотрудников каждого отделения, можно применить следующий запрос SQL:

```
SELECT branchNo, AVG(salary)
FROM Staff
GROUP BY branchNo;
```

8.5. Физическое проектирование (продолжение)

Шаг 4.3. Выбрать индексы (продолжение).

- **Рекомендации по созданию индексов (продолжение).**

4. (Продолжение) Согласно приведенной выше рекомендации, можно предусмотреть создание индекса на атрибуте branchNo, поскольку он участвует в конструкции GROUP BY.

   Но более эффективное решение может предусматривать создание индекса и на атрибуте branchNo, и на атрибуте salary. Это позволяет выполнить в СУБД весь запрос по данным только из самого индекса, без необходимости доступа к файлу данных.

   Такой план выполнения запроса иногда называют планом, предусматривающим использование только индекса, поскольку необходимые результаты могут быть получены с помощью только данных, содержащихся в индексе.

8.5. Физическое проектирование (продолжение)

Шаг 4.3. Выбрать индексы (продолжение).

- **Рекомендации по созданию индексов (продолжение).**

5. Не индексировать атрибут или отношение, которые часто обновляются.
6. Не индексировать атрибут, если в запросах с использованием этого атрибута обычно происходит выборка значительной части (например, 25%) строк кортежей в отношении. В таком случае может оказаться более эффективным поиск во всем отношении, чем поиск с использованием индекса.
7. Не индексировать атрибуты, которые состоят из длинных символьных строк.

8.5. Физическое проектирование (продолжение)

Шаг 4.4. Оценить потребность в дисковой памяти.

- **Цель.** Определить объем дискового пространства, который требуется для базы данных.
- Во многих проектах выдвигается требование, чтобы физическая реализация базы данных могла быть осуществлена на основе существующей конфигурации аппаратных средств.
- Но даже при отсутствии такого требования проектировщик обязан оценить объем дискового пространства, который требуется для хранения файлов базы данных, хотя бы на тот случай, что в связи с этим потребуется приобретение новых аппаратных средств.
- Цель данного этапа состоит в оценке объема дискового пространства, необходимого для поддержки реализации базы данных во внешней памяти.

8.5. Физическое проектирование (продолжение)

Шаг 4.4. Оценить потребность в дисковой памяти (продолжение).

- Как и на предыдущих этапах, проведение оценки потребности в дисковом пространстве в значительной степени зависит от целевой СУБД и от аппаратных средств, которые применяются для поддержки функционирования базы данных.
- Как правило, такая оценка основана на информации о среднем размере каждой строки и количестве строк в отношении.
- Эта оценка должна показывать максимальное количество, но может также оказаться целесообразным проведение анализа интенсивности роста размеров отношения и корректировка итоговых сведений об объеме диска с учетом полученного коэффициента роста для определения возможных размеров базы данных в будущем.

8.5. Физическое проектирование (продолжение)

Шаг 5. Спроектировать представления пользователей.

- **Цель.** Спроектировать пользовательские представления, необходимость в создании которых была выявлена на стадии сбора и анализа требований, составляющей часть жизненного цикла системы с базой данных.
- В многопользовательской СУБД такие представления играют главную роль при определении структуры базы данных и соблюдении правил защиты. Основные преимущества пользовательских представлений:
  - независимость от данных
  - упрощение структуры приложения
  - учет потребностей пользователей.

8.5. Физическое проектирование (продолжение)

Шаг 6. Спроектировать механизмы защиты.

- **Цель.** Проектирование средств защиты для базы данных в соответствии с требованиями пользователей.
- Назначение этого этапа состоит в определении способов реализации требований к защите. Состав средств защиты зависит от конкретной системы. Поэтому проектировщик должен знать, какие возможности предлагает рассматриваемая им целевая СУБД. Реляционные СУБД, как правило, предоставляют средства защиты базы данных следующих типов:
  - защита системы;
  - защита данных.

8.5. Физическое проектирование (продолжение)

Шаг 6. Спроектировать механизмы защиты (продолжение).

- Средства защиты системы регламентируют доступ и эксплуатацию базы данных на уровне системы; к ним, в частности, относится аутентификация пользователя по имени и паролю.
- Средства защиты данных регламентируют доступ и использование объектов базы данных (таких как отношения и представления), а также действия, которые могут быть выполнены пользователями с конкретными объектами.
- Правила доступа можно создавать с помощью команд GRANT и REVOKE.

8.5. Физическое проектирование (продолжение)

Шаг 7. Рассмотреть необходимость введения контролируемой избыточности.

- **Цель.** Определение необходимости ввода контролируемой избыточности за счет ослабления условий нормализации для повышения производительности системы.
- Результатом нормализации является логическая модель базы данных — структурно цельная система с минимальной избыточностью.
- Но в некоторых случаях оказывается, что нормализованная модель не обеспечивает максимальной производительности при обработке данных.
- Следовательно, при некоторых обстоятельствах может оказаться необходимым ради повышения производительности пожертвовать частью той выгоды, которую обеспечивает модель с полной нормализацией.

8.5. Физическое проектирование (продолжение)

Шаг 7. Рассмотреть необходимость введения контролируемой избыточности (продолжение).

- К денормализации следует прибегать лишь тогда, когда нормализованная база данных не удовлетворяет требованиям, предъявляемым к производительности системы.
- Необходимо учесть и следующие особенности:
  - денормализация может усложнить физическую реализацию системы;
  - денормализация часто приводит к снижению гибкости;
  - денормализация может ускорить чтение данных, но при этом замедлить обновление записей.
- Формально *денормализацию* можно определить как модификацию реляционной модели, при которой степень нормализации модифицированного отношения становится ниже, чем степень нормализации, по меньшей мере, одного из исходных отношений.

8.5. Физическое проектирование (продолжение)

Шаг 7. Рассмотреть необходимость введения контролируемой избыточности (продолжение).

- Термин «денормализация» будет также применяться в случае, когда два отношения объединяются в одно и полученное отношение остается нормализованным, однако содержит больше пустых значений, чем исходные отношения.
- Некоторые авторы определяют денормализацию как *модификацию реляционной модели с учетом требований эксплуатации.*
- Существует следующее эмпирическое правило: если производительность системы не удовлетворяет поставленным требованиям и проектируемое отношение имеет низкую частоту операций обновления при большой частоте запросов, тогда денормализация может оказаться оправданной.

8.5. Физическое проектирование (продолжение)

Шаг 7. Рассмотреть необходимость введения контролируемой избыточности (продолжение).

- **ПРИМЕР.** Неявная денормализация.
- Отношение Branch (branchNo, street, city, postcode, mgrStaffNo) не находится в 3НФ, т. к. атрибут city (город) функционально определяется атрибутом postcode (почтовый индекс).
- Иными словами, мы можем определить значение атрибута city, зная значение атрибута postcode.
- Таким образом, отношение Branch находится лишь в 2НФ.
- Чтобы привести это отношение к третьей нормальной форме, его пришлось бы разбить на два отношения:

  Branch (branchNo, street, postcode, mgrStaffNo)

  Postcode (postcode, city)

8.5. Физическое проектирование (продолжение)

Шаг 7. Рассмотреть необходимость введения контролируемой избыточности (продолжение).

- **ПРОДОЛЖЕНИЕ ПРИМЕРА.** Неявная денормализация.
- Однако адрес отделения редко требуется без атрибута city, т.е. без указания города.
- Это означает, что после такого разделения таблиц придется формировать соединение (и выполнять соответствующий запрос) каждый раз, когда потребуется получить полный адрес.
- Учитывая это, можно остановиться на второй нормальной форме и реализовать в базе данных первоначальный вариант отношения Branch.

8.5. Физическое проектирование (продолжение)

Шаг 7. Рассмотреть необходимость введения контролируемой избыточности (продолжение).

- Нельзя сформулировать общие правила определения того, когда действительно требуется денормализация отношений. Рассмотрим несколько наиболее распространенных ситуаций.
- Шаг 7.1. Объединение связей типа «один к одному» (1:1).
- Шаг 7.2. Дублирование неключевых атрибутов в связях «один ко многим» (1:*) для уменьшения количества соединений.
- Шаг 7.3. Дублирование атрибутов внешнего ключа в связях «один ко многим» (1:*) для уменьшения количества соединений.

8.5. Физическое проектирование (продолжение)

Шаг 7. Рассмотреть необходимость введения контролируемой избыточности (продолжение).

- Шаг 7.4. Дублирование атрибутов в связях «многие ко многим» (1:*) для уменьшения количества соединений.
- Шаг 7.5. Введение повторяющихся групп.
- Шаг 7.6. Создание таблиц из данных, содержащихся в других таблицах.
- Шаг 7.7. Секционирование отношений.

8.5. Физическое проектирование (продолжение)

Шаг 7. Рассмотреть необходимость введения контролируемой избыточности (продолжение).

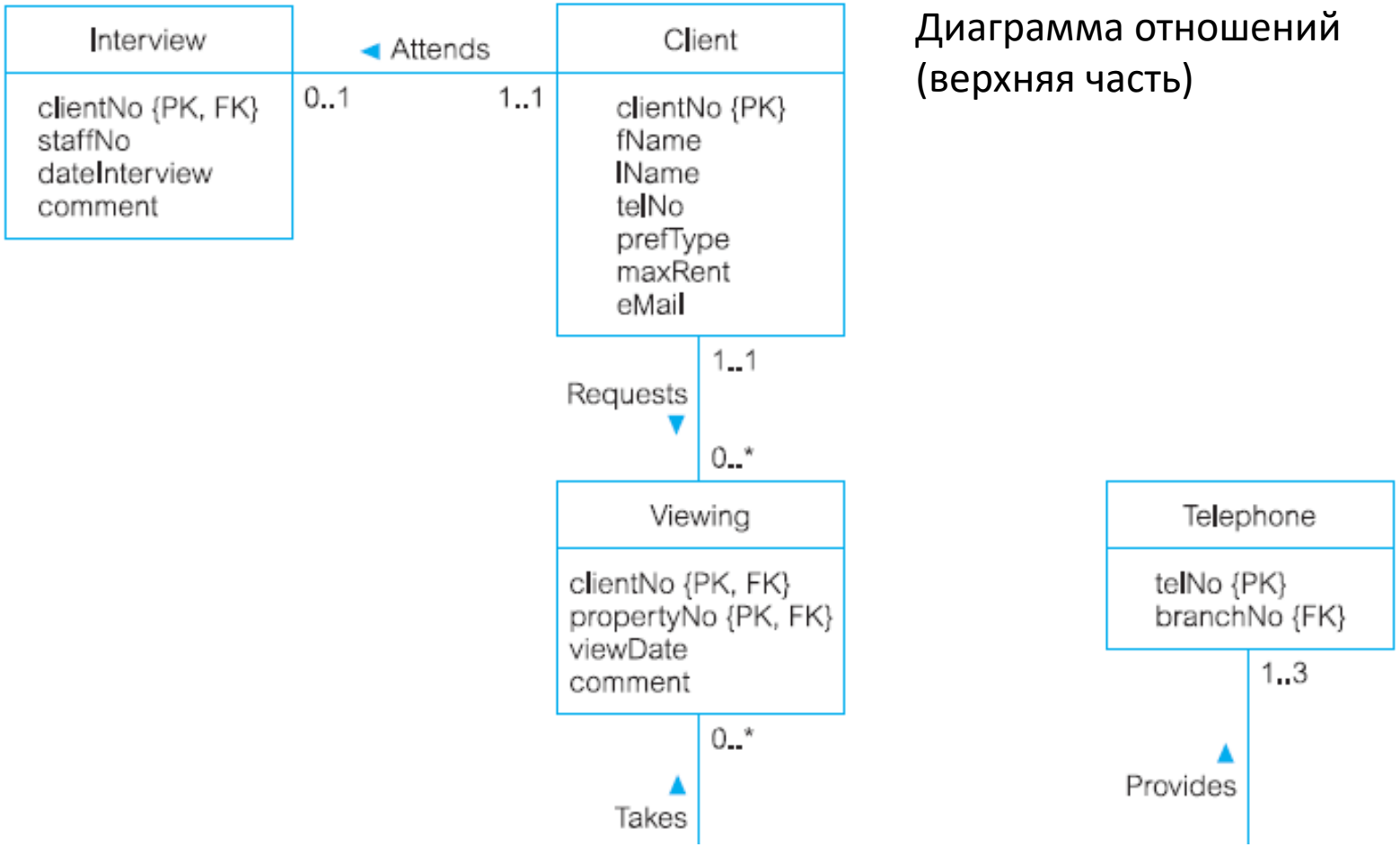
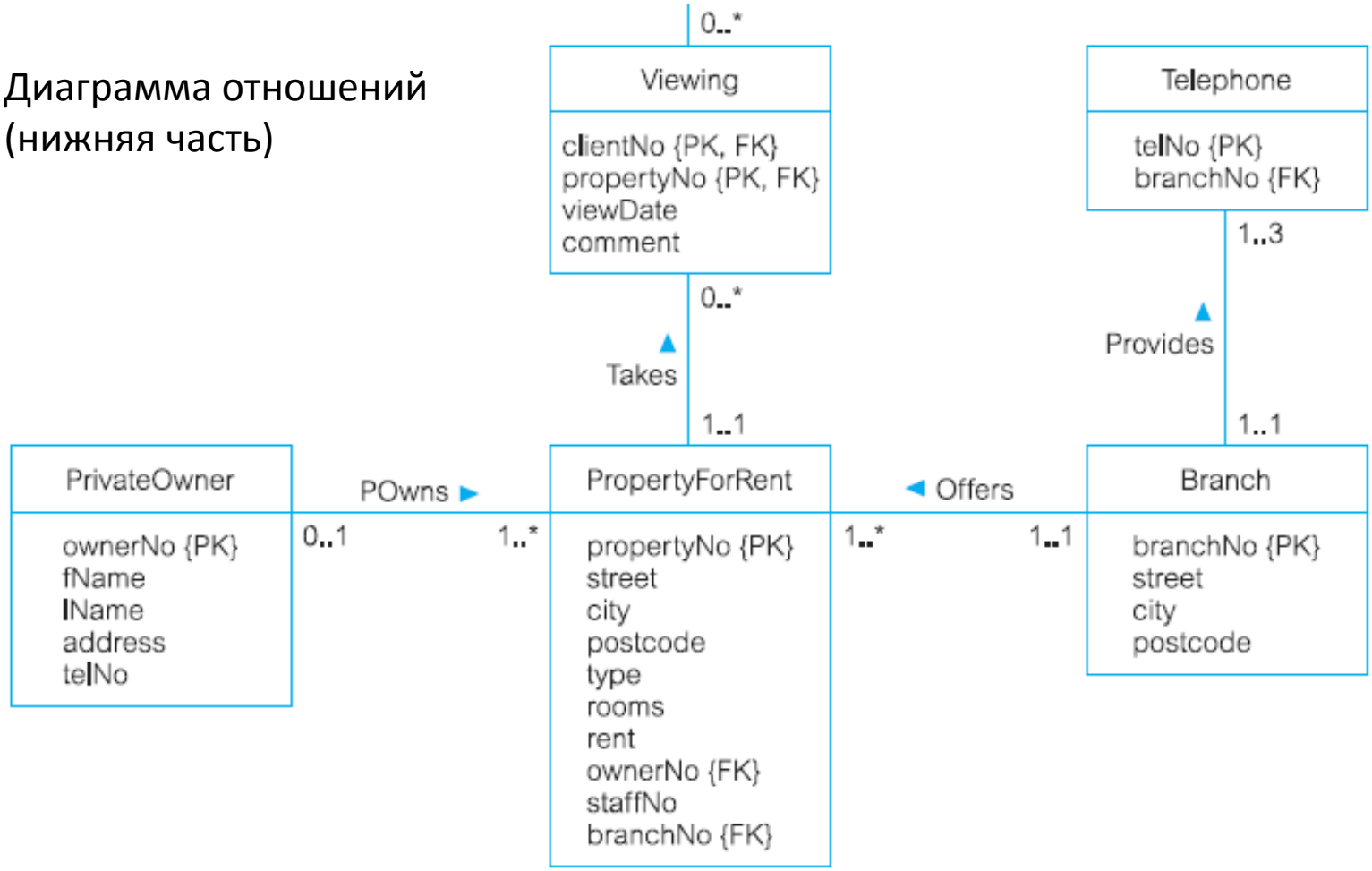


Диаграмма отношений (верхняя часть)

8.5. Физическое проектирование (продолжение)

Шаг 7. Рассмотреть необходимость введения контролируемой избыточности (продолжение).

Диаграмма отношений (нижняя часть)



8.5. Физическое проектирование (продолжение)

Шаг 7. Рассмотреть необходимость введения контролируемой избыточности (продолжение).

Branch

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Telephone

telNo	branchNo
0207-886-1212	B005
0207-886-1300	B005
0207-886-4100	B005
01224-67125	B007
0141-339-2178	B003
0141-339-4439	B003
0117-916-1170	B004
0208-963-1030	B002

Interview

clientNo	staffNo	dateInterview	comment
CR56	SG37	11-Apr-12	current lease ends in June
CR62	SA9	7-Mar-12	needs property urgently

8.5. Физическое проектирование (продолжение)

Шаг 7. Рассмотреть необходимость введения контролируемой избыточности (продолжение).

PropertyForRent

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

Client

clientNo	fName	lName	telNo	prefType	maxRent	eMail
CR76	John	Kay	0207-774-5632	Flat	425	john.kay@gmail.com
CR56	Aline	Stewart	0141-848-1825	Flat	350	astewart@hotmail.com
CR74	Mike	Ritchie	01475-392178	House	750	mritchie01@yahoo.co.uk
CR62	Mary	Tregear	01224-196720	Flat	600	maryt@hotmail.co.uk

8.5. Физическое проектирование (продолжение)

Шаг 7. Рассмотреть необходимость введения контролируемой избыточности (продолжение).

PrivateOwner

ownerNo	fName	lName	address	telNo
CO46	Joe	Keogh	2 Fergus Dr, Aberdeen AB2 7SX	01224-861212
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419
CO40	Tina	Murphy	63 Well St, Glasgow G42	0141-943-1728
CO93	Tony	Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025

Viewing

clientNo	propertyNo	viewDate	comment
CR56	PA14	24-May-13	too small
CR76	PG4	20-Apr-13	too remote
CR56	PG4	26-May-13	
CR62	PA14	14-May-13	
CR56	PG36	28-Apr-13	no dining room

8.5. Физическое проектирование (продолжение)

Шаг 7.1. Объединение связей типа «один к одному» (1:1).

- Этот прием имеет смысл применять в том случае, если наиболее часто генерируются запросы, обращающиеся сразу к нескольким объединяемым таблицам, а наиболее редко — запросы только к отдельным таблицам.
- Рассмотрим, например, связь 1:1 между отношениями Client и Interview. Отношение Client содержит информацию о потенциальных арендаторах объектов недвижимости. Отношение Interview содержит даты собеседований, проведенных сотрудником компании с данным клиентом, а также комментарии, касающиеся клиента.
- Мы можем объединить эти два отношения в новое отношение ClientInterview. Поскольку между отношениями Client и Interview установлена связь 1:1 с возможным отсутствием соответствующей записи в дочернем отношении, объединенное отношение ClientInterview с большой вероятностью будет содержать много пустых полей.

8.5. Физическое проектирование (продолжение)

Шаг 7.1. Объединение связей типа «один к одному» (1:1) (продолжение).

ClientInterview

clientNo	fName	lName	telNo	prefType	maxRent	eMail
CR76	John	Kay	0207-774-5632	Flat	425	john.kay@gmail.com
CR56	Aline	Stewart	0141-848-1825	Flat	350	astewart@hotmail.com
CR74	Mike	Ritchie	01475–392178	House	750	mritchie01@yahoo.co.uk
CR62	Mary	Tregear	01224–196720	Flat	600	maryt@hotmail.co.uk

Рисунок разделен на две части. В нижней части показаны атрибуты исходного отношения Interview.

<u>Обратите внимание на наличие здесь пустых полей.</u>

staffNo	dateInterview	comment
SG37	11-Apr-12	current lease ends in June
SA9	7-Mar-12	needs property urgently

8.5. Физическое проектирование (продолжение)

Шаг 7.2. Дублирование неключевых атрибутов в связях «один ко многим» (1:*) для уменьшения количества соединений.

- На этапе денормализации основной целью является уменьшение количества соединений (или полное их устранение) в запросах, которые выполняются особенно часто или являются важными.
- Например, при извлечении данных по запросу из отношения PropertyForRent (Арендуемый объект недвижимости) очень часто приходится одновременно извлекать имя владельца.

```
SELECT p.*, o.lName
FROM PropertyForRent p, PrivateOwner o
WHERE p.ownerNo = o.ownerNo AND branchNo = 'B003';
```

8.5. Физическое проектирование (продолжение)

Шаг 7.2. Дублирование неключевых атрибутов в связях «один ко многим» (1:*) для уменьшения количества соединений (продолжение).

- Если атрибут IName, обозначающий в данном случае фамилию владельца, будет введен и в отношение PropertyForRent, то не нужно будет ссылаться в запросе на таблицу PrivateOwner, и запрос примет вид:

```
SELECT p.*
FROM PropertyForRent p
WHERE branchNo = 'B003';
```

- В данном примере, поскольку изменение имени владельца недвижимости весьма маловероятно, дублирование поля IName представляется разумным решением.

8.5. Физическое проектирование (продолжение)

Шаг 7.2. Дублирование неключевых атрибутов в связях «один ко многим» (1:*) для уменьшения количества соединений (продолжение).

Частный случай связи 1:M – это использование так называемых поисковых таблиц (lookup table). По-другому их называют еще справочными таблицами (*reference table или pick list*). В типичном случае такая таблица содержит код и его описание. Если определить справочную (родительскую) таблицу для атрибута «Тип собственности», то можно модифицировать таблицу PropertyForRent (дочернюю).

PropertyType

type	description
H	House
F	Flat

PropertyForRent

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	H	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	F	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	F	3	350	CO40		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	F	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	H	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	F	4	450	CO93	SG14	B003

8.5. Физическое проектирование (продолжение)

Шаг 7.2. Дублирование неключевых атрибутов в связях «один ко многим» (1:*) для уменьшения количества соединений (продолжение).

- Использование справочных таблиц дает следующие преимущества:
  - уменьшение размеров дочернего отношения: тип данных, используемый для хранения кода, занимает меньший объем, чем описание;
  - если потребуется изменить описание объекта, достаточно будет внести изменение в одну из записей справочной таблицы и не изменять множество записей в дочернем отношении;
  - справочные таблицы используют для эффективного контроля правильности данных, вводимых пользователем.

8.5. Физическое проектирование (продолжение)

Шаг 7.2. Дублирование неключевых атрибутов в связях «один ко многим» (1:*) для уменьшения количества соединений (продолжение).

- Если таблица поиска используется в часто выполняющихся запросах, а атрибут «Описание», скорее всего, не изменится, то можно рассмотреть вопрос дублирования атрибута «Описание» в дочерней таблице.
- При этом оригинальная поисковая таблица (lookup table) не будет избыточной, т. к. она по-прежнему может использоваться для проверки данных, вводимых пользователем. Но теперь мы устраним необходимость в выполнении соединений таблиц.

PropertyForRent

propertyNo	street	city	postcode	type	description	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	H	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	F	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	F	Flat	3	350	CO40		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	F	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	H	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	F	Flat	4	450	CO93	SG14	B003

8.5. Физическое проектирование (продолжение)

Шаг 7.3. Дублирование атрибутов внешнего ключа в связях «один ко многим» (1:*) для уменьшения количества соединений.

- К часто выполняющимся можно отнести запрос в приложении *DreamHome,* требующий вывести список всех владельцев, которые сдают свою недвижимость в аренду через данное отделение.
- Соответствующая форма SQL этого запроса имеет следующий вид:

```
SELECT o.lName
FROM PropertyForRent p, PrivateOwner o
WHERE p.ownerNo = o.ownerNo AND branchNo = 'B003';
```

8.5. Физическое проектирование (продолжение)

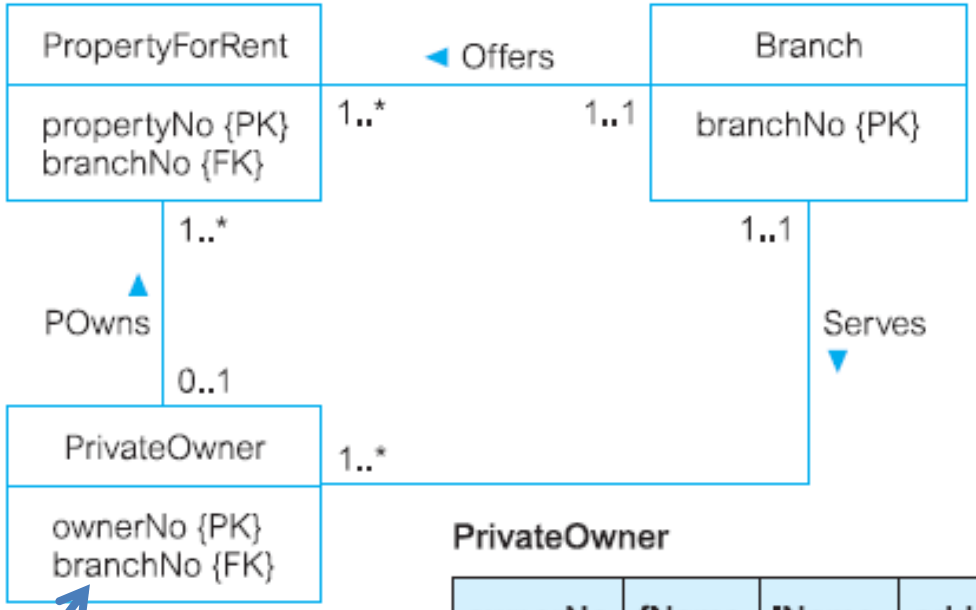
Шаг 7.3. Дублирование атрибутов внешнего ключа в связях «один ко многим» (1:*) для уменьшения количества соединений (продолжение).

- Таким образом, из-за отсутствия прямой связи между отношениями PrivateOwner и Branch для получения списка владельцев приходится извлекать данные о номере отделения (атрибут branchNo) из таблицы PropertyForRent. Необходимость в этом соединении можно устранить из запроса, продублировав внешний ключ branchNo в таблице PrivateOwner.
- Иными словами, необходимо установить прямую связь между таблицами Branch и PrivateOwner. Теперь запрос SQL принимает следующий вид:

```
SELECT o.lName
FROM PrivateOwner o
WHERE branchNo = 'B003';
```

8.5. Физическое проектирование (продолжение)

Шаг 7.3. Дублирование атрибутов внешнего ключа в связях «один ко многим» (1:*) для уменьшения количества соединений (продолжение).



Модифицированная диаграмма отношений и измененное отношение PrivateOwner

Продублированный внешний ключ

PrivateOwner

ownerNo	fName	lName	address	telNo	branchNo
CO46	Joe	Keogh	2 Fergus Dr, Aberdeen AB2 7SX	01224-861212	B007
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419	B003
CO40	Tina	Murphy	63 Well St, Glasgow G42	0141-943-1728	B003
CO93	Tony	Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025	B003

8.5. Физическое проектирование (продолжение)

Шаг 7.3. Дублирование атрибутов внешнего ключа в связях «один ко многим» (1:*) для уменьшения количества соединений (продолжение).

- Если бы владелец мог сдавать свою недвижимость в аренду через различные отделения, рассмотренный способ денормализации стал бы неприменимым.
- В этом случае между таблицами Branch и PrivateOwner пришлось бы создавать связь типа «многие ко многим» (*:*). Отметим также, что в таблицу PropertyForRent включен атрибут branchNo, поскольку для управления данным объектом недвижимости может не быть назначен сотрудник компании (например, если компания только начинает работу с этой недвижимостью).

8.5. Физическое проектирование (продолжение)

Шаг 7.3. Дублирование атрибутов внешнего ключа в связях «один ко многим» (1:*) для уменьшения количества соединений (продолжение).

- Если бы отношение PropertyForRent не включало номер отделения (атрибут branchNo), для доступа к этому атрибуту пришлось бы формировать соединение таблиц PropertyForRent и Staff через атрибут staffNo. В этом случае исходный запрос SQL принял бы вид:

  SELECT o.lName
  FROM Staff s, PropertyForRent p, PrivateOwner o
  WHERE s.staffNo = p.staffNo **AND** p.ownerNo = o.ownerNo **AND** s.branchNo = 'B003';

- Возможность удалить из запроса два соединения может служить хорошим доводом в пользу того, чтобы создать прямую связь между таблицами PrivateOwner и Branch, внедрив внешний ключ branchNo в отношение PrivateOwner.

8.5. Физическое проектирование (продолжение)

Шаг 7.4. Дублирование атрибутов в связях «многие ко многим» (1:*) для уменьшения количества соединений.

- Проектируя логическую модель базы данных, мы отображали каждую связь вида *:* на три отношения: два из них порождались на основе двух оригинальных сущностей, участвующих с этой связи, а третье отношение представляло саму связь между этими сущностями.
- Теперь для восстановления информации, представленной в связи вида *:*, нам придется объединить три таблицы.
- Но в некоторых случаях удается сократить количество таблиц, подлежащих соединению, продублировав атрибуты одной из исходных таблиц в промежуточной таблице.

8.5. Физическое проектирование (продолжение)

Шаг 7.4. Дублирование атрибутов в связях «многие ко многим» (1:*) для уменьшения количества соединений (продолжение).

- Например, была выполнена декомпозиция связи *:* между сущностями Client и PropertyForRent с помощью промежуточного отношения Viewing.
- Предположим, что необходимо учесть следующее требование: сотрудники компании *DreamHome* должны связываться с теми клиентами, которые осмотрели предложенную недвижимость и еще не оставили свои комментарии по поводу увиденного. При этом отметим, что сотруднику при разговоре с клиентами нужен лишь атрибут street, поэтому соответствующий запрос SQL для исходной базы данных принимает следующий вид:

```
SELECT p.street, c.*, v.viewDate
FROM Client c, Viewing v, PropertyForRent p
WHERE v.propertyNo = p.propertyNo AND c.clientNo = v.clientNo  AND
comment IS NULL;
```

8.5. Физическое проектирование (продолжение)

Шаг 7.4. Дублирование атрибутов в связях «многие ко многим» (1:*) для уменьшения количества соединений (продолжение).

- Продублировав атрибут street в промежуточном отношении Viewing, можно убрать из запроса ссылку на отношение PropertyForRent:

```
SELECT c.*, v.street, v.viewDate
FROM Client c, Viewing v
WHERE c.clientNo = v.clientNo AND comment IS NULL;
```

Viewing

clientNo	propertyNo	street	viewDate	comment
CR56	PA14	16 Holhead	24-May-13	too small
CR76	PG4	6 Lawrence St	20-Apr-13	too remote
CR56	PG4	6 Lawrence St	26-May-13	
CR62	PA14	16 Holhead	14-May-13	no dining room
CR56	PG36	2 Manor Rd	28-Apr-13	

8.5. Физическое проектирование (продолжение)

Шаг 7.5. Введение повторяющихся групп.

- Мы убирали повторяющиеся группы полей из логической модели данных, когда приводили объекты базы данных к первой нормальной форме (1НФ). Затем мы выделяли повторяющиеся группы в отдельные отношения, формируя при этом связь 1:* с исходным (родительским) отношением.
- Но иногда для повышения общей производительности системы имеет смысл снова ввести повторяющиеся группы. Так, в отделениях компании *DreamHome* может быть установлено не больше трех номеров телефонов, при этом не каждое отделение обязательно имеет три телефонные линии.
- Исходя из этого, мы создали в логической модели данных таблицу Telephone со связью типа «три к одному» (3:1) с таблицей Branch.

8.5. Физическое проектирование (продолжение)

Шаг 7.5. Введение повторяющихся групп (продолжение).

- Если доступ к этой информации важен или обращение выполняется часто, тогда может оказаться эффективным объединить отношения и записывать номера телефонов в оригинальное отношение Branch, предусмотрев по одному атрибуту для каждого из трех номеров телефонов.
- Такой тип денормализации следует рассматривать лишь при следующих условиях:
  - максимальное количество элементов в повторяющейся группе известно заранее (в нашем примере — максимальное число номеров телефонов, т. е. 3);
  - число элементов в повторяющейся группе постоянно и не изменится со временем (максимальное число телефонных линий постоянно и едва ли изменится);
  - число элементов в повторяющейся группе не слишком велико, обычно не больше 10 (однако, это условие не так важно, как первые два).

8.5. Физическое проектирование (продолжение)

Шаг 7.5. Введение повторяющихся групп (продолжение).

- Иногда в повторяющейся группе полей наиболее важным и часто используемым является поле, соответствующее последнему или текущему значению атрибута, или даже сам факт, что имеет место повторяющаяся группа.
- В предыдущем примере мы могли бы разместить в отношении Branch лишь один номер телефона, оставив остальные номера в отношении Telephone. Это позволило бы избавиться от пустых значений в отношении Branch, поскольку в каждом отделении всегда имеется хотя бы один номер телефона.

Branch

branchNo	street	city	postcode	telNo1	telNo2	telNo3
B005	22 Deer Rd	London	SW1 4EH	0207-886-1212	0207-886-1300	0207-886-4100
B007	16 Argyll St	Aberdeen	AB2 3SU	01224-67125		
B003	163 Main St	Glasgow	G11 9QX	0141-339-2178	0141-339-4439	
B004	32 Manse Rd	Bristol	BS99 1NZ	0117-916-1170		
B002	56 Clover Dr	London	NW10 6EU	0208-963-1030		

8.5. Физическое проектирование (продолжение)

Шаг 7.6. Создание таблиц из данных, содержащихся в других таблицах.

- Бывают случаи, когда отчеты необходимо формировать в самое напряженное (пиковое) время. Для подготовки таких отчетов требуются доступ к производным данным и формирование соединения нескольких отношений, принадлежащих к одному и тому же набору основных отношений.
- Однако данные, включаемые в отчет, во многих случаях относительно неизменны или не должны обязательно отражать самые актуальные данные (т. е. отчет считается приемлемым, если данные получены несколько часов тому назад).

8.5. Физическое проектирование (продолжение)

Шаг 7.6. Создание таблиц из данных, содержащихся в других таблицах (продолжение).

- В таких случаях можно создать единую, в значительной степени денормализованную таблицу, содержащую те поля из разных базовых отношений, на которые ссылается отчет.
- Пользователь будет использовать эту таблицу вместо того, чтобы непосредственно обращаться к базовым отношениям. Такие таблицы чаще всего создаются и заполняются в пакетном режиме в ночное время, когда система загружена незначительно.
- Можно использовать материализованные представления (PostgreSQL).

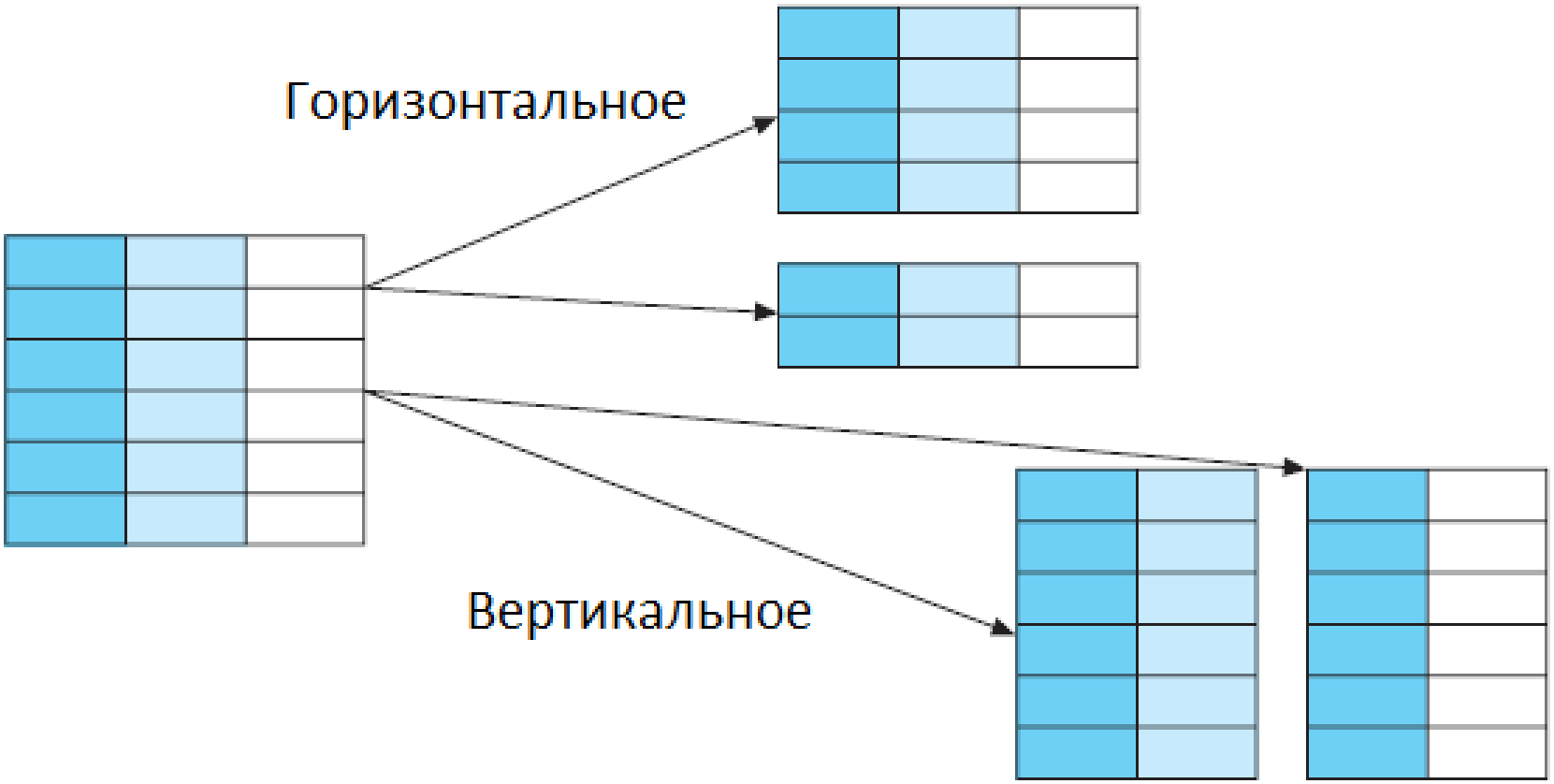
8.5. Физическое проектирование (продолжение)

Шаг 7.7. Секционирование отношений.

- Альтернативным подходом к объединению отношений является их **секционирование (partitioning)**. Оно призвано решать проблему поддержки очень больших отношений (и индексов). При использовании секционирования отношение разбивается на ряд более мелких фрагментов, которыми легче управлять. Они называются **секциями (partitions)**.
- Существуют два вида секционирования: горизонтальное и вертикальное.
- **Горизонтальное секционирование**. Распределяет кортежи отношения между несколькими, возможно, более мелкими, отношениями.
- **Вертикальное секционирование**. Распределяет атрибуты отношения между несколькими, возможно, более мелкими, отношениями. Первичный ключ дублируется, чтобы можно было восстановить оригинальное отношение.

8.5. Физическое проектирование (продолжение)

Шаг 7.7. Секционирование отношений (продолжение).



8.5. Физическое проектирование (продолжение)

Шаг 7.7. Секционирование отношений (продолжение).

- **ПРИМЕР**. Секционирование в СУБД PostgreSQL.
- Создание таблицы, секционируемой по диапазонам:

```
CREATE TABLE measurement
( logdate date NOT NULL,
  peaktemp int,
  unitsales int
)
PARTITION BY RANGE ( logdate );
```

- Создание секции таблицы, секционируемой по диапазонам:

```
CREATE TABLE measurement_y2016m07
PARTITION OF measurement
( unitsales DEFAULT 0 )
FOR VALUES FROM ( '2016-07-01‘ ) TO ( '2016-08-01‘ );
```

8.5. Физическое проектирование (продолжение)

Шаг 7.7. Секционирование отношений (продолжение).

- **ПРИМЕР**. Секционирование в СУБД PostgreSQL.
- Создание таблицы, секционируемой по спискам:

```
CREATE TABLE cities
( city_id bigserial NOT NULL,
  name text NOT NULL,
  population bigint
)
PARTITION BY LIST ( left( lower( name ), 1 ) );
```

- Создание секции таблицы, секционируемой по спискам:

```
CREATE TABLE cities_ab
PARTITION OF cities
( CONSTRAINT city_id_nonzero CHECK ( city_id != 0 ) )
FOR VALUES IN ( 'a', 'b‘ );
```

8.5. Физическое проектирование (продолжение)

Шаг 7.7. Секционирование отношений (продолжение).

- Секции особенно полезны в приложениях, которые хранят и обрабатывают большие объемы данных.
- Предположим, что, например, в базе данных *DreamHome* поддерживается отношение ArchivedPropertyForRent, содержащее сотни тысяч записей, которые хранятся неопределенно долго для аналитических целей. Поиск конкретной записи для какого-то филиала (branch) может занимать довольно долгое время.
- Это время можно сократить, создав горизонтальные секции, по одной для каждого филиала компании.

8.5. Физическое проектирование (продолжение)

Шаг 7.7. Секционирование отношений (продолжение).

- Могут также возникнуть обстоятельства, при которых нам часто требуются некоторые атрибуты очень большого отношения. В таком случае имеет смысл использовать вертикальное секционирование.
- В одну секцию нужно поместить те атрибуты, обращение к которым происходит часто. А в другую секцию – остальные атрибуты. Первичный ключ повторяется в каждой из секций, что позволит восстановить исходное отношение с помощью операции соединения.

8.5. Физическое проектирование (продолжение)

Шаг 7.7. Секционирование отношений (продолжение).

- Преимущества секционирования:

1. *Улучшение балансировки нагрузки.* Секции можно разместить на различных носителях данных, тем самым позволяя организовать параллельный доступ.
2. *Повышение производительности.* Она может быть повышена за счет снижения объема данных, которые нужно просмотреть или обработать, и за счет параллельной работы.
3. *Повышение доступности (availability).* Если секции распределены по различным областям устройств для хранения данных, и одна из таких областей вышла из строя, другие секции все еще могут быть доступными.
4. *Улучшение восстановления после сбоев.* Более мелкие секции можно восстановить эффективнее; организовать их резервное копирование легче, чем резервное копирование очень больших отношений.
5. *Защищенность.* Можно предоставить доступ к конкретной секции только тем пользователям, которым он необходим. Можно организовать различные ограничения доступа к разным секциям.

8.5. Физическое проектирование (продолжение)

Шаг 7.7. Секционирование отношений (продолжение).

- Недостатки секционирования:

1. *Сложность.* Секционирование обычно не является прозрачным для конечных пользователей, поэтому запросы, в которых участвует более одной секции, сложнее написать.
2. *Снижение производительности.* Запросы, в которых объединяются данные из более чем одной секции, могут оказаться медленнее, чем без использования секционирования.
3. *Дублирование.* Вертикальное секционирование связано с дублированием первичного ключа. Это может привести не только к повышенным затратам внешней памяти, но также и к несогласованности данных.

8.5. Физическое проектирование (продолжение)

Общие советы по денормализации

- Рассмотрите последствия денормализации на предыдущих шагах процесса проектирования базы данных. Например, может оказаться необходимым повторно оценить выбор индексов, создаваемых для денормализованных отношений.
- Необходимо продумать технологию поддержания целостности данных. Традиционные решения таковы:

1. *Триггеры.* Могут использоваться для автоматизации обновления производных или дублирующихся данных.
2. *Транзакции.* Обновление денормализованных данных нужно выполнять атомарным образом.
3. *Согласование данных в пакетном режиме.* Запуск пакетных программ, которые согласовывают денормализованные данные, в соответствующее время.

8.5. Физическое проектирование (продолжение)

Общие советы по денормализации (продолжение)

- С точки зрения подержания целостности данных лучшим решением являются триггеры. Однако они могут вызвать проблемы производительности.
- Преимущества денормализации:

1. Может повыситься производительность за счет:
   - предварительного вычисления производных данных;
   - снижения потребности в выполнении соединений таблиц;
   - снижения числа внешних ключей в отношениях;
   - снижения числа индексов;
   - снижения числа отношений.

8.5. Физическое проектирование (продолжение)

Общие советы по денормализации (продолжение)

- Недостатки денормализации:

1. Могут ускориться выборки данных, но при этом может замедлиться их обновление.
2. Всегда выполняется с учетом специфики конкретного приложения. При изменении его специфики денормализацию нужно пересматривать.
3. Может увеличиться размер отношений.
4. В одних случаях реализация может упроститься, а в других – усложниться.
5. Приносится в жертву гибкость.

Все решения, связанные с денормализацией, необходимо тщательно документировать, указывая причину принятия того или иного решения. Необходимо обновлять логическую модель данных соответственно.

8.5. Физическое проектирование (продолжение)

Шаг 8. Организация мониторинга и настройка функционирования работающей системы.

- **Цель.** Обеспечить текущий контроль функционирования работающей системы и добиться повышения ее производительности для устранения последствий принятия неоптимальных проектных решений или учета изменившихся требований.
- Одной из целей физического проектирования является обеспечение эффективного хранения данных и получения доступа к ним.
- Для оценки эффективности можно использовать следующие показатели:

1. Число транзакций, обрабатываемых в единицу времени.
2. Время отклика системы.
3. Объем дисковой памяти для хранения базы данных.

8.5. Физическое проектирование (продолжение)

Шаг 8. Организация мониторинга и настройка функционирования работающей системы (продолжение).

- Настройка работающей системы может принести целый ряд выгод:

1. Настройка системы позволяет устранить потребность в дополнительном оборудовании.
2. Настройка системы дает возможность снизить требования к аппаратному обеспечению и, как результат, снизить расходы на обслуживание базы данных.
3. Точная настройка системы позволяет сократить время ответа на запрос и повысить производительность базы данных, что, в свою очередь, повышает эффективность работы как отдельных сотрудников (пользователей базы данных), так и всей организации в целом.
4. Сокращение времени ответа на запрос улучшает психологическое состояние персонала.
5. Чем короче время ответа базы данных на запрос, тем более удовлетворенным чувствует себя клиент фирмы.

8.6. Полезные ссылки

- Software Ideas Modeler https://www.softwareideas.net/
- Экосистема PostgreSQL/Postgres Pro https://postgrespro.ru/products/ecosystem
-

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 глав 9–10. Запросы к базе данных выполнять с помощью утилиты psql, описанной в главе 2, параграф 2.2.