**Связи между таблицами**

1. **PolicyType - Policy**
   * Один тип полиса (PolicyType) может быть связан с несколькими полисами (Policy).
   * Связь осуществляется через внешний ключ PolicyTypeId в таблице Policy.
2. **Policy - Contract**
   * Один полис (Policy) может быть связан с несколькими контрактами (Contract).
   * Связь осуществляется через внешний ключ PolicyId в таблице Contract.
3. **Agent - Policy**
   * Один агент (Agent) может быть связан с несколькими полисами (Policy).
   * Связь осуществляется через внешний ключ AgentId в таблице Policy.
4. **Agent - Contract**
   * Один агент (Agent) может быть связан с несколькими контрактами (Contract).
   * Связь осуществляется через внешний ключ AgentId в таблице Contract.
5. **Client - Policy**
   * Один клиент (Client) может быть связан с несколькими полисами (Policy).
   * Связь осуществляется через внешний ключ ClientId в таблице Policy.
6. **Client - Contract**
   * Один клиент (Client) может быть связан с несколькими контрактами (Contract).
   * Связь осуществляется через внешний ключ ClientId в таблице Contract.

**Инфологической** **моделью** **данных** называют обобщенное неформальное описание создаваемой базы **данных**, выполненное с использованием естественного языка, математических фор-мул, таблиц, графиков и других средств, понятных всем людям, работающим.

**Основные компоненты инфологической модели:**

1. **Сущности (Entities)**:
   * Представляют объекты или понятия, которые имеют значение для бизнеса и о которых необходимо хранить информацию. Примеры: Клиент, Полис, Агент.
2. **Атрибуты (Attributes)**:
   * Это характеристики или свойства сущностей. Примеры атрибутов для сущности Клиент: Имя, Фамилия, Адрес, Дата рождения.
3. **Связи (Relationships)**:
   * Определяют, как сущности взаимодействуют друг с другом. Например, связь между Клиентом и Полисом: один клиент может иметь несколько полисов, а каждый полис принадлежит одному клиенту.

**Даталогическое** или логическая **модель** **данных** – это **модель**, ори-ентированная на реализацию **БД** в конкретной СУБД, т.е. это инфологиче-ская **модель**, трансформированная с учетом требований и ограничений кон-кретной СУБД (тип **модели** **данных**, поддерживаемой СУБД, формат **данных**, возможности по обеспечению целостности **данных** и т.д.). Логическая **модель** отражает логические связи между элементами **дан**-**ных** вне зависимости от их содержания и среды хранения.

**Основные компоненты даталогической модели:**

1. **Таблицы (Tables)**:
   * Это структурированные наборы данных, состоящие из строк и столбцов. Каждая таблица соответствует одной сущности из инфологической модели.
2. **Столбцы (Columns)**:
   * Поля в таблице, представляющие атрибуты сущности. Каждый столбец имеет определенный тип данных.
3. **Ключи (Keys)**:
   * **Первичные ключи (Primary Keys)**: Уникальные идентификаторы строк в таблице.
   * **Внешние ключи (Foreign Keys)**: Поля, которые ссылаются на первичные ключи в других таблицах, обеспечивая целостность данных.
4. **Связи (Relationships)**:
   * Связи между таблицами, которые выражаются через первичные и внешние ключи. Эти связи могут быть одного из трех типов: один-к-одному (1:1), один-ко-многим (1

), многие-ко-многим (M

).

1. **Ограничения (Constraints)**:
   * Правила, которые обеспечивают целостность и согласованность данных, такие как уникальность, отсутствие NULL значений, проверка условий и т.д.

### 2 запрос: Объяснение:

**1. SELECT PT.PolicyTypeName, SUM(C.TotalPrice) AS TotalPriceSum**

* **PT.PolicyTypeName**: Выбирается название типа полиса из таблицы PolicyType.
* **SUM(C.TotalPrice) AS TotalPriceSum**: Вычисляется суммарная стоимость всех контрактов для каждого типа полиса. Псевдоним TotalPriceSum используется для удобства в результатах запроса.

**2. FROM dbo.PolicyType AS PT**

* Запрос начинается с таблицы PolicyType, которая обозначена псевдонимом PT.

**3. LEFT JOIN dbo.Policy AS P ON PT.PolicyTypeId = P.PolicyTypeId**

* **LEFT JOIN**: Это соединение всех записей из таблицы PolicyType с соответствующими записями из таблицы Policy по полю PolicyTypeId.
* Если для какого-то типа полиса (из PolicyType) нет соответствующих записей в таблице Policy, то в результирующем наборе все поля из таблицы Policy для этого типа полиса будут заполнены NULL.

**4. LEFT JOIN dbo.Contract AS C ON P.PolicyId = C.PolicyId**

* **LEFT JOIN**: Это соединение всех записей из результата предыдущего соединения (между PolicyType и Policy) с соответствующими записями из таблицы Contract по полю PolicyId.
* Если для какого-то полиса нет соответствующих записей в таблице Contract, то в результирующем наборе все поля из таблицы Contract для этого полиса будут заполнены NULL.

**5. GROUP BY PT.PolicyTypeName**

* Запрос группирует результаты по названию типа полиса.
* Для каждой группы (каждого типа полиса) вычисляется суммарная стоимость контрактов (SUM(C.TotalPrice)).

### 3 запрос: Объяснение:

**1. SELECT C.ClientId, C.Surname, C.Name,**

* **C.ClientId**: Выбирается идентификатор клиента из таблицы Client.
* **C.Surname**: Выбирается фамилия клиента из таблицы Client.
* **C.Name**: Выбирается имя клиента из таблицы Client.

**2. (SELECT SUM(CT.TotalPrice) FROM Contract CT WHERE CT.ClientId = C.ClientId) AS TotalPrice**

* Это подзапрос, который вычисляет суммарную стоимость (SUM(CT.TotalPrice)) всех контрактов для конкретного клиента.
* **FROM Contract CT**: Подзапрос берет данные из таблицы Contract с псевдонимом CT.
* **WHERE CT.ClientId = C.ClientId**: Подзапрос фильтрует строки из таблицы Contract, где ClientId совпадает с ClientId из текущей строки основной таблицы Client.
* **AS TotalPrice**: Результат подзапроса присваивается псевдоним TotalPrice.

**3. FROM Client C**

* Основной запрос выбирает данные из таблицы Client, обозначенной псевдонимом C.

**4. ORDER BY TotalPrice DESC**

* **ORDER BY**: Условие, которое сортирует результаты по столбцу TotalPrice в порядке убывания.
* **DESC**: Сортировка в порядке убывания (от большего к меньшему).

**Объяснение:**

**1. SELECT C.ClientId, C.Surname, C.Name, C.BirthDate**

* **C.ClientId**: Выбирается идентификатор клиента из таблицы Client.
* **C.Surname**: Выбирается фамилия клиента из таблицы Client.
* **C.Name**: Выбирается имя клиента из таблицы Client.
* **C.BirthDate**: Выбирается дата рождения клиента из таблицы Client.

**2. FROM Client C**

* Основной запрос выбирает данные из таблицы Client, обозначенной псевдонимом C.

**3. WHERE YEAR(C.BirthDate) < (**

* Условие WHERE фильтрует строки, выбирая только тех клиентов, год рождения которых меньше некоторого среднего значения, вычисляемого во вложенном подзапросе.