



Рисунок 1. Даталогическая схема данных к ТЗ

Сущности 1.1 “Номенклатура” и 1.2 “Каталог номенклатуры / дерево категорий” отражены в таблицах **Products** и **Categories** соответственно.

Вложенность таблицы **Categories** реализована через сущность *parent_id*, ссылающуюся на первичный ключ *id*. Такой дизайн подразумевает возможность иметь условно неограниченный уровень вложенности.

Сущность 1.3 “Клиенты” отражена в таблице **Clients**.

Сущность 1.4 “Заказы покупателей” отражена в таблицах **Orders** и **Order_items**. Такой дизайн позволяет добавлять произвольное (ненулевое) количество товаров в заказ.

Внешние ключи отображены стрелками.

2.1. Получение информации о сумме товаров заказанных под каждого клиента (Наименование клиента, сумма)

В подзапросе получение сумм с агрегацией по **Order_items.order_id**. В основном запросе получение сумм с агрегацией по **Orders.client_name**.

```
WITH subq AS (  
    SELECT  
        Order_items.order_id,  
        SUM(Products.price * Order_items.amount) AS order_price  
    FROM  
        Order_items JOIN Products  
        ON Order_items.product_id = Products.id  
    GROUP BY  
        Order_items.order_id  
)  
  
SELECT  
    Orders.client_name,  
    SUM(subq.order_price) AS total_sum  
FROM  
    Orders JOIN subq  
    ON Orders.id = subq.order_id  
GROUP BY  
    Orders.client_name
```

2.2. Найти количество дочерних элементов первого уровня вложенности для категорий номенклатуры.

В рекурсивном подзапросе построение каталога с учетом вложенности. В основном запросе подсчет дочерних элементов. LEFT JOIN используется для включения в результат тех элементов, у которых нет потомков.

Сортировка результатов по массиву *path* позволяет корректно учесть вложенность категорий.

Поле *level* отображает уровень вложенности категории.

```
WITH RECURSIVE category_tree AS (  
    SELECT  
        id,  
        name,  
        parent_id,  
        ARRAY[id] AS path,  
        1 AS level  
    FROM categories  
    WHERE parent_id IS NULL  
  
    UNION ALL  
  
    SELECT  
        categories.id,  
        categories.name,  
        categories.parent_id,  
        category_tree.path || categories.id,  
        category_tree.level + 1  
    FROM  
        categories JOIN category_tree  
        ON categories.parent_id = category_tree.id  
)  
  
SELECT  
    category_tree.name,  
    category_tree.level AS nesting_level,  
    COUNT(categories.id) AS first_order_children_count  
FROM  
    category_tree LEFT JOIN categories  
    ON category_tree.id = categories.parent_id  
GROUP BY  
    category_tree.id,  
    category_tree.name,  
    category_tree.level,  
    category_tree.path  
ORDER BY  
    category_tree.path
```