

Федеральное агентство по образованию Российской Федерации
Государственное образовательное учреждение высшего
профессионального образования Нижегородский государственный
университет им Н. И. Лобачевского

Институт информационных технологий, математики и механики

Отчет по курсу «Технологии баз данных»
Вариант №27
Ювелирная мастерская

Выполнил:
студент института ИТММ

Проверил:
доцент кафедры Программной инженерии
Шапошников Д. Е.

Нижний Новгород
2023 г

Оглавление

[Введение](#)

[Постановка учебной задачи](#)

[Описание предметной области](#)

[Таблицы](#)

[Развитие постановки задачи](#)

[Концептуальная схема](#)

[Структуры базы данных](#)

[Операторы вывода агрегатных состояний](#)

[Триггер для проверки правильности вводимых значений](#)

[Процедура с курсором для модификации определённых записей по условию](#)

[Процедура для удаления родительской записи с соответствующими подчиненными \(дочерними\) записями в другой таблице без использования параметра CASCADE во внешних ключах](#)

[Star схема](#)

Введение

Данные являются неотъемлемой частью современного мира, их используют почти во всех сферах жизни, начиная от бизнеса и заканчивая медициной и наукой. В свою очередь, базы данных являются незаменимым инструментом для хранения, организации и управления данными.

В проекте, связанном с данными и базами данных, мы будем исследовать их основные принципы и технологии, а также разработаем базу данных для конкретной задачи.

Кроме того, мы изучим процесс проектирования базы данных, начиная от определения требований к базе данных и заканчивая созданием схемы базы данных и наполнением ее данными. Все это поможет нам лучше понять, как работают базы данных и как их использовать для решения различных задач.

Постановка учебной задачи

Создать концептуальную схему по выбранной предметной области (с учетом её развития) и на её основе спроектировать структуру реляционной базы данных. Выделить обязательные поля, наложить условия целостности, условия корректности значений, если необходимо.

Создать структуру базы данных (таблицы, представления, внешние ключи). Оформить структуру в виде скрипта на языке SQL.

Создать операторы языка SQL для вывода агрегатных данных (с использованием агрегатных функций и подзапросов).

- a. Должно присутствовать соединение таблиц
- b. Должны присутствовать скалярные функции работы с датой / временем
- c. Должны присутствовать скалярные функции с текстовым поиском
- d. Должен присутствовать поиск родительских записей, у которых нет дочерних

Создать триггер INSERT для проверки правильности вводимых данных (на одно условие).

Создать процедуру с курсором для модификации определённых записей по условию.

Создать процедуру для удаления родительской записи с соответствующими подчиненными (дочерними) записями в другой таблице без использования параметра CASCADE во внешних ключах.

Для одной условно-постоянной сущности создать схему "Star" необязательных атрибутов и создать функцию добавления атрибута и функцию добавления значения

Описание предметной области

Вы работаете в ювелирной мастерской.

Ваша мастерская осуществляет изготовление ювелирных изделий для частных лиц на заказ.

Вы работаете с определенными материалами (платина, золото, серебро, различные драгоценные камни и т.д.).

При обращении к Вам потенциального клиента Вы определяетесь с тем, какое именно изделие ему необходимо.

Все изготавливаемые Вами изделия принадлежат к некоторому типу (серьги, кольца, броши, браслеты), бывают выполнены из определенного материала, имеют некоторый вес и цену (включающую стоимость материалов и работы).

Таблицы

- Изделия (Код изделия, Название, Тип, Код материала, Вес, Цена).
- Материалы (Код материала, Название, Цена за грамм).
- Продажи (Код изделия, Дата продажи, Фамилия покупателя, Имя покупателя, Отчество покупателя).

Развитие постановки задачи

В процессе опытной эксплуатации базы данных выяснилось, что ювелирное изделие может состоять из нескольких материалов. Кроме того, постоянным клиентам мастерская предоставляет скидки.

Внести в структуру таблиц изменения, учитывающие эти факты, и изменить существующие запросы. Добавить новые запросы.

Концептуальная схема

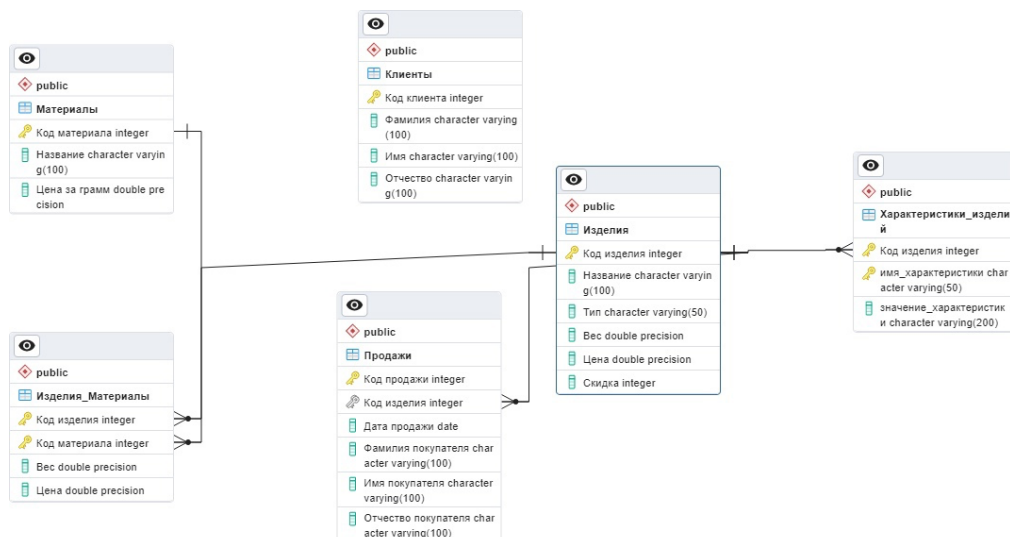
Концептуальная схема базы данных - это модель данных, которая описывает сущности, атрибуты и связи между ними. Она является первым этапом при проектировании базы данных и служит основой для создания более подробных моделей, таких, как логическая и физическая модели.

Концептуальная схема базы данных имеет несколько преимуществ:

Она помогает понять и описать структуру хранимых данных и связи между ними, что позволяет лучше понимать предметную область и требования к базе данных.

Концептуальная схема базы данных позволяет установить связи между сущностями. Это помогает определить, как данные должны быть связаны и как они будут использоваться в дальнейшем.

Она помогает избежать ошибок при проектировании базы данных, таких, как дублирование данных или неправильное определение связей между таблицами.



Структуры базы данных

Таблицы:

1. Изделия

Данный код создает таблицу с шестью столбцами для хранения информации об изделиях, включая их название, тип, стоимость производства, цену продажи и процент скидки. Первичный ключ "Код изделия" гарантирует, что каждая строка в таблице уникальна и может быть легко идентифицирована.

"Код изделия" - это ключ с типом данных "SERIAL", который автоматически генерирует уникальное целочисленное значение для каждой новой строки, добавленной в таблицу. Этот столбец используется для уникальной идентификации каждого изделия в таблице.

Название - этот столбец имеет тип данных "VARCHAR (100)", что означает, что он может хранить до 100 символов переменной длины. Тут хранится название изделия.

Тип - этот столбец имеет тоже тип данных "VARCHAR (50)". Тут хранится тип изделия.

Вес - этот столбец имеет тип данных "FLOAT", что означает, что он может хранить десятичные числа с плавающей точкой. Тут хранится стоимость производства изделия.

Цена - этот столбец также имеет тип данных "FLOAT". Тут хранится цена продажи изделия.

Скидка - этот столбец имеет тип данных "INTEGER" и значение по умолчанию равно 0. Тут хранится процент скидки, применяемый к цене продажи изделия.

```
CREATE TABLE Изделия (  
    "Код изделия" SERIAL PRIMARY KEY,  
    Название VARCHAR(100) NOT NULL,  
    Тип VARCHAR(50) NOT NULL,  
    Вес FLOAT NOT NULL,  
    Цена FLOAT NOT NULL,  
    Скидка INTEGER DEFAULT 0  
);
```

2. Материалы

Данный код создает таблицу с тремя столбцами для хранения информации о материалах, включая их название и цену за грамм. Первичный ключ "Код материала" гарантирует, что каждая строка в таблице уникальна и может быть легко идентифицирована.

"Код материала" - это первичный ключ с типом данных «SERIAL». Столбец используется для уникальной идентификации каждого материала в таблице.

Название - этот столбец имеет тип данных "VARCHAR (100)". Тут хранится название материала.

"Цена за грамм" - этот столбец имеет тип данных «FLOAT». Тут хранится цена материала за грамм.

```
CREATE TABLE Материалы (  
    "Код материала" SERIAL PRIMARY KEY,  
    Название VARCHAR(100) NOT NULL,  
    "Цена за грамм" FLOAT NOT NULL  
);
```

3. Продажи

Данный код создает таблицу с шестью столбцами для хранения информации о продажах изделий, включая информацию о конкретном изделии, дате продажи и информацию о покупателе (фамилия, имя, отчество). Первичный ключ "Код продажи" гарантирует, что каждая строка в таблице уникальна и может быть легко идентифицирована. Связь "Код изделия" с таблицей "Изделия" гарантирует целостность данных и соответствие продаж конкретным изделиям.

"Код продажи" - это первичный ключ с типом данных "SERIAL", Этот столбец используется для уникальной идентификации каждой продажи в таблице.

"Код изделия" - этот столбец имеет тип данных "INTEGER" и не может содержать пустых значений "NOT NULL". Он связан с первичным ключом столбца "Код изделия" в таблице "Изделия" через фразу "REFERENCE Изделия ("Код изделия")". Это означает, что каждая продажа связана с конкретным изделием из таблицы "Изделия". Ключевое слово "ON DELETE CASCADE" используется для того, чтобы при удалении изделия из таблицы "Изделия", соответствующие записи в таблице "Продажи" также будут удалены.

"Дата продажи" - этот столбец имеет тип данных "DATE" и не может содержать пустых значений "NOT NULL". В этом столбце хранится дата продажи изделия.

"Фамилия покупателя" - этот столбец имеет тип данных "VARCHAR (100)". В этом столбце хранится фамилия покупателя.

"Имя покупателя" - этот столбец имеет тип данных "VARCHAR (100)". В этом столбце хранится имя покупателя.

"Отчество покупателя" - этот столбец имеет тип данных "VARCHAR (100)". В этом столбце хранится отчество покупателя.

```
CREATE TABLE Продажи (  
  "Код продажи" SERIAL PRIMARY KEY,  
  "Код изделия" INTEGER NOT NULL REFERENCES Изделия("Код изделия") ON DELETE CASCADE,  
  "Дата продажи" DATE NOT NULL,  
  "Фамилия покупателя" VARCHAR(100) NOT NULL,  
  "Имя покупателя" VARCHAR(100) NOT NULL,  
  "Отчество покупателя" VARCHAR(100) NOT NULL  
);
```

4. Изделия_Материалы

Данный код создает таблицу с пятью столбцами для хранения информации о связи между изделиями и материалами, включая информацию о стоимости материала, цене продажи изделия и связанных с этими изделиями и материалами ключах "Код изделия" и "Код материала". Связи "Код изделия" и "Код материала" с таблицами "Изделия" и "Материалы" соответственно гарантируют целостность данных и соответствие материалов конкретным изделиям. Составной первичный ключ гарантирует уникальность каждой строки в таблице.

"Код изделия" - этот столбец имеет тип данных "INTEGER" и не может содержать пустых значений "NOT NULL". Он связан с первичным ключом столбца "Код изделия" в таблице "Изделия" через фразу "REFERENCES Изделия ("Код изделия»)".

"Код материала" - этот столбец имеет тип данных "INTEGER" и не может содержать пустых значений "NOT NULL". Он связан с первичным ключом столбца "Код материала" в таблице "Материалы" через фразу "REFERENCES Материалы ("Код материала»)".

Вес - этот столбец имеет тип данных "FLOAT" и не может содержать пустых значений "NOT NULL". В этом столбце хранится стоимость материала, необходимого для производства изделия.

Цена - этот столбец имеет тип данных "FLOAT" и не может содержать пустых значений "NOT NULL". В этом столбце хранится цена продажи изделия после учета стоимости материала и других затрат.

PRIMARY KEY ("Код изделия", "Код материала") - это составной первичный ключ таблицы, который гарантирует уникальность каждой строки в таблице в сочетании с ключами "Код изделия" и "Код материала". Это означает, что каждое изделие может использовать каждый материал только один раз в этой таблице.

```
CREATE TABLE Изделия_Материалы (  
    "Код изделия" INTEGER NOT NULL REFERENCES Изделия("Код изделия") ON DELETE CASCADE,  
    "Код материала" INTEGER NOT NULL REFERENCES Материалы("Код материала") ON DELETE CASCADE,  
    Вес FLOAT NOT NULL,  
    Цена FLOAT NOT NULL,  
    PRIMARY KEY ("Код изделия", "Код материала")  
);
```

5. Клиенты

Данный код создает таблицу с четырьмя столбцами для хранения информации о клиентах, включая их фамилию, имя и отчество. Первичный ключ "Код клиента" гарантирует, что каждая строка в таблице уникальна и может быть легко идентифицирована.

"Код клиента" - это первичный ключ с типом данных "SERIAL". Этот столбец используется для уникальной идентификации каждого клиента в таблице.

Фамилия - этот столбец имеет тип данных "VARCHAR (100)". В этом столбце хранится фамилия клиента.

Имя - этот столбец имеет тип данных "VARCHAR (100)". В этом столбце хранится имя клиента.

Отчество - этот столбец имеет тип данных "VARCHAR (100)". В этом столбце хранится отчество клиента.

```
CREATE TABLE Клиенты (  
    "Код клиента" SERIAL PRIMARY KEY,  
    Фамилия VARCHAR(100) NOT NULL,  
    Имя VARCHAR(100) NOT NULL,  
    Отчество VARCHAR(100) NOT NULL  
);|
```

Операторы вывода агрегатных состояний

Вывести общую стоимость продаж за месяц.

Запрос суммирует значения столбца "Цена" из таблицы "Продажи", связанной со столбцом "Код изделия" таблицы "Изделия". Далее, запрос фильтрует данные, выбирая только те, которые соответствуют условию "Дата продажи" больше или равно '2020-05-01' и меньше '2023-06-01'.

Таким образом, данный запрос возвращает сумму цен продаж изделий за период с 1 мая 2020 года по 31 мая 2023 года.


Если данные в таблицах "Продажи" и "Изделия" были заполнены корректно и без ошибок, а также если данные в столбце "Цена" были заданы в правильном формате, то данный запрос должен работать корректно и вернуть результат в виде единственной строки с одним столбцом, содержащим сумму цен продаж.

Запрос


```
1 SELECT SUM("Цена") FROM Продажи
2 JOIN Изделия ON Продажи."Код изделия" = Изделия."Код изделия"
3 WHERE "Дата продажи" >= '2020-05-01' AND "Дата продажи" < '2023-06-01';
```

История запросовData OutputСообщенияNotifications


≡+





▼





▼









	sum double precision 
1	1

Вывести количество продаж по типам изделий:

Запрос выбирает значения столбца "Тип" из таблицы "Изделия" и подсчитывает количество продаж для каждого типа изделий из таблицы "Продажи", связанной со столбцом "Код изделия" таблицы "Изделия". Далее, запрос группирует данные по столбцу "Тип" таблицы "Изделия".

Таким образом, данный запрос возвращает количество продаж для каждого типа изделий.

Если данные в таблицах "Продажи" и "Изделия" были заполнены корректно и без ошибок, то данный запрос должен работать корректно и вернуть результат в виде таблицы с двумя столбцами: "Тип" (тип изделия) и "count" (количество продаж).

```
1 SELECT Изделия.Тип, COUNT(*) FROM Продажи
2 JOIN Изделия ON Продажи."Код изделия" = Изделия."Код изделия"
3 GROUP BY Изделия.Тип;
```

История запросов Data Output Сообщения Notifications

	Тип character varying	count bigint	
1	Камень	2	

Вывести изделия, сделанные из материала с названием, содержащим слово «Каменюка».

Запрос выбирает значения столбца "Название" из таблицы "Изделия" и значения столбца "Название" из таблицы "Материалы". Затем запрос объединяет таблицы "Изделия", "Изделия_Материалы" и "Материалы", связывая их по соответствующим ключам "Код изделия" и "Код материала".

Далее, запрос фильтрует данные, выбирая только те, которые соответствуют условию "Название" материала содержит строку "Каменюка".

Таким образом, данный запрос возвращает названия изделий и материалов, которые содержат строку "Каменюка" в названии материала.

Если данные в таблицах "Изделия", "Материалы" и "Изделия_Материалы" были заполнены корректно и без ошибок, то данный запрос должен работать корректно и вернуть результат в виде таблицы с двумя столбцами: "Название" изделия и "Название" материала, удовлетворяющих заданному условию.

```
1 SELECT Изделия."Название", Материалы."Название" FROM Изделия
2 JOIN Изделия_Материалы ON Изделия."Код изделия" = Изделия_Материалы."Код изделия"
3 JOIN Материалы ON Изделия_Материалы."Код материала" = Материалы."Код материала"
4 WHERE Материалы."Название" LIKE '%Каменюка%';
```

История запросов Data Output Сообщения Notifications

	Название character varying	Название character varying
1	вапвамама	Каменюка
2	ваапв	Каменюка

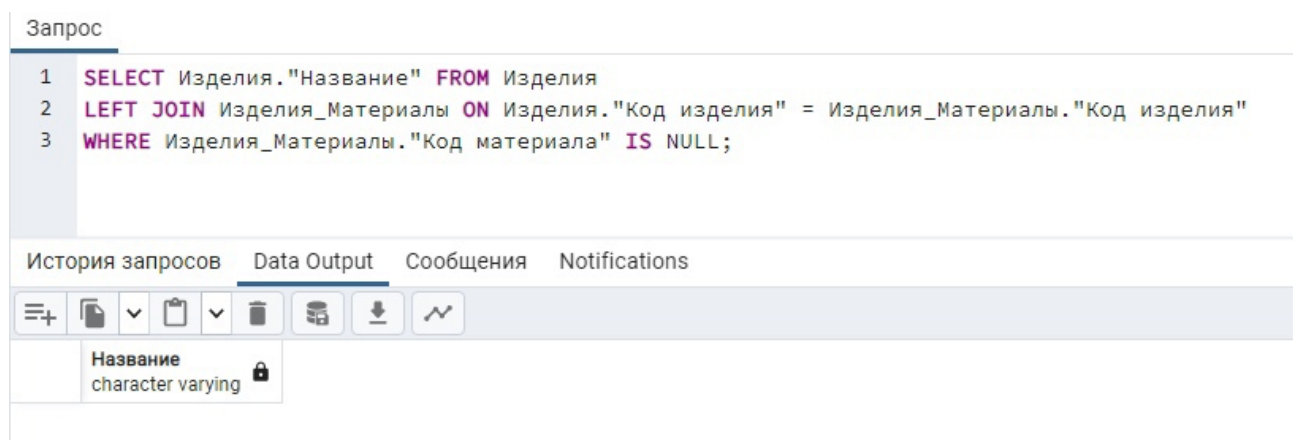
Вывести изделия, у которых нет материалов (их нет).

Запрос выбирает значения столбца "Название" из таблицы "Изделия". Затем запрос объединяет таблицы "Изделия" и "Изделия_Материалы", связывая их по соответствующим ключам "Код изделия".

Далее, запрос фильтрует данные, выбирая только те, для которых не существует записи в таблице "Изделия_Материалы", связанной с соответствующим кодом изделия. Это определяется условием "Код материала" IS NULL.

Таким образом, данный запрос возвращает названия изделий, которые не имеют связанных материалов.

Если данные в таблицах "Изделия" и "Изделия_Материалы" были заполнены корректно и без ошибок, то данный запрос должен работать корректно и вернуть результат в виде таблицы со столбцом "Название" изделий, не имеющих связанных материалов.



Триггер для проверки правильности вводимых значений

```
1 CREATE OR REPLACE FUNCTION check_цена() RETURNS TRIGGER AS $$
2 BEGIN
3     IF NEW."Цена" < 0 THEN
4         RAISE EXCEPTION 'Цена не может быть отрицательной';
5     END IF;
6     RETURN NEW;
7 END;
8 $$ LANGUAGE plpgsql;
9
10 CREATE TRIGGER "триггер_цена" BEFORE INSERT OR UPDATE ON Изделия FOR EACH ROW EXECUTE FUNCTION check_цена();
```

	Код изделия [PK] integer	Название character varying	Тип character varying	Вес double precision	Цена double precision	Скидка integer
1	1	ваапв	Камень	30000	25	0
2	2	вапвамама	Камень	3000000000	1	0
3	3	ваапапа	Камень	2500000	2	0
4+	[default]	вайа	Зеркало	12	-5	[default]

	Код изделия [PK] integer	Название character varying	Тип character varying	Вес double precision	Цена double precision	Скидка integer
1	1	ваапв	Камень	30000	25	0
2	2	вапвамама	Камень	3000000000	1	0
3	3	ваапапа	Камень	2500000	2	0
4+	[default]	вайа	Зеркало	12	-5	[default]

❗ Цена не может быть отрицательной CONTEXT: функция PL/pgSQL "check_цена()", строка 4, оператор RAISE

Данный код представляет собой создание функции и триггера базы данных для таблицы "Изделия".

Функция `check_Цена()` предназначена для проверки значений столбца "Цена" при вставке или обновлении записей в таблице "Изделия". Функция возвращает значение NEW, если значение "Цена" больше или равно 0, иначе вызывается исключение с сообщением "Цена не может быть отрицательной".

Создание триггера "триггер_цена" обеспечивает автоматическую проверку значений столбца "Цена" перед вставкой или обновлением записей в таблице "Изделия". Триггер "триггер_цена" вызывает функцию `check_цена()` перед каждой вставкой или обновлением записей в таблице "Изделия".

Таким образом, данный код обеспечивает проверку значения столбца "Цена" при вставке или обновлении записей в таблице "Изделия" и автоматическое вызов функции `check_Цена()`. Если значение столбца "Цена" меньше 0, то триггер вызовет исключение и вставка или обновление записей не будет выполнено.

Процедура с курсором для модификации определённых записей по условию

```
1 CREATE OR REPLACE FUNCTION update_цена(тип_изделия VARCHAR(50), старая_цена FLOAT, новая_цена FLOAT) RETURNS VOID AS $$
2 DECLARE
3     курсор CURSOR FOR SELECT "Код изделия" FROM Изделия WHERE Тип = тип_изделия AND Цена = старая_цена;
4 BEGIN
5     FOR запись IN курсор LOOP
6         UPDATE Изделия SET Цена = новая_цена WHERE "Код изделия" = запись."Код изделия";
7     END LOOP;
8 END;
9 $$ LANGUAGE plpgsql;
```

	Код изделия [PK] integer	Название character varying	Тип character varying	Вес double precision	Цена double precision	Скидка integer
1	1	ваапв	Камень	30000	25	0
2	2	вапвамама	Камень	30000000000	1	0
3	3	ваапапа	Камень	2500000	1	0

≡+

	Код изделия [PK] integer	Название character varying	Тип character varying	Вес double precision	Цена double precision	Скидка integer
1	1	ваапв	Камень	30000	25	0
2	2	вапвамама	Камень	30000000000	2	0
3	3	ваапапа	Камень	2500000	2	0

Данный код представляет собой создание функции базы данных для таблицы "Изделия".

Функция `update_цена()` предназначена для обновления значений столбца "Цена" для записей в таблице "Изделия" с заданными значениями типа изделия, старой цены и новой цены.

Функция использует курсор для выборки всех записей в таблице "Изделия", у которых тип изделия и старая цена соответствуют заданным значениям.

Далее, функция обновляет значения столбца "Цена" для каждой выбранной записи в таблице "Изделия", устанавливая новое значение цены.

Таким образом, данный код обеспечивает обновление значений столбца "Цена" для записей в таблице "Изделия" с заданными значениями типа изделия, старой цены и новой цены. Если записи не найдены, то функция не выполняет никаких действий.

Для вызова функции необходимо передать значения параметров `тип_изделия`, `старая_цена` и `новая_цена`.

Пример вызова функции:

```
SELECT update_цена('тип изделия', 100.00, 110.00);
```

Данный запрос обновит значения столбца "Цена" для всех записей в таблице "Изделия" с типом изделия "тип изделия" и старой ценой 100.00 на новую цену 110.00.

Процедура для удаления родительской записи с соответствующими подчиненными (дочерними) записями в другой таблице без использования параметра CASCADE во внешних ключах

```
CREATE OR REPLACE FUNCTION delete_изделие(код INTEGER) RETURNS VOID AS $$  
BEGIN  
    DELETE FROM Продажи WHERE "Код изделия" = код;  
    DELETE FROM Изделия_Материалы WHERE "Код изделия" = код;  
    DELETE FROM Изделия WHERE "Код изделия" = код;  
END;  
$$ LANGUAGE plpgsql;
```

Данный код представляет собой создание функции базы данных для удаления записей в таблицах "Изделия", "Изделия_Материалы" и "Продажи".

Функция delete_изделие() предназначена для удаления записей из указанных таблиц, связанных с заданным кодом изделия.

Функция использует три оператора DELETE для удаления записей из таблиц "Продажи", "Изделия_Материалы" и "Изделия", соответствующих заданному коду изделия.

Таким образом, данный код обеспечивает удаление записей из таблиц "Изделия", "Изделия_Материалы" и "Продажи", связанных с заданным кодом изделия.

Для вызова функции необходимо передать параметр код, соответствующий удаляемому изделию.

Пример вызова функции:

```
SELECT delete_изделие(123);
```

Данный запрос удалит все записи в таблицах "Изделия", "Изделия_Материалы" и "Продажи", связанные с изделием, у которого "Код изделия" равен 123.

Star схема

Запрос

```
1 CREATE TABLE Характеристики_ювелирных_изделий (  
2 "Код изделия" INTEGER,  
3 имя_характеристики VARCHAR(50),  
4 значение_характеристики VARCHAR(200),  
5 PRIMARY KEY ("Код изделия", имя_характеристики),  
6 FOREIGN KEY ("Код изделия") REFERENCES Ювелирные_изделия ("Код изделия") ON DELETE CASCADE  
7 );  
8  
9 CREATE OR REPLACE FUNCTION добавить_характеристику_ювелирного_изделия("Код изделия" INTEGER, имя_характеристики VARCHAR(50), значение_характеристики VARCHAR(200))  
10 RETURNS VOID AS $$  
11 BEGIN  
12 INSERT INTO Характеристики_ювелирных_изделий ("Код изделия", имя_характеристики, значение_характеристики)  
13 VALUES ("Код изделия", имя_характеристики, значение_характеристики);  
14 END;  
15 $$ LANGUAGE plpgsql;  
16  
17 CREATE OR REPLACE FUNCTION изменить_значение_характеристики_ювелирного_изделия("Код изделия" INTEGER, имя_характеристики VARCHAR(50), значение_характеристики VARCHAR(200))  
18 RETURNS VOID AS $$  
19 BEGIN  
20 UPDATE Характеристики_ювелирных_изделий  
21 SET значение_характеристики = значение_характеристики || ' ' || изменить_значение_характеристики_ювелирного_изделия.значение_характеристики  
22 WHERE "Код изделия" = изменить_значение_характеристики_ювелирного_изделия."Код изделия" AND имя_характеристики = изменить_значение_характеристики_ювелирного_изделия.имя_характеристики;  
23 END;  
24 $$ LANGUAGE plpgsql;
```

Данный код представляет собой создание таблицы "Характеристики_ювелирных_изделий" и функций базы данных для добавления и изменения характеристик ювелирных изделий.

Таблица "Характеристики_ювелирных_изделий" имеет три столбца: "Код изделия", "имя_характеристики" и "значение_характеристики". Первые два столбца образуют первичный ключ, а столбец "Код изделия" является внешним ключом, связывающим таблицу с таблицей "Ювелирные_изделия".

Функция "добавить_характеристику_ювелирного_изделия" предназначена для добавления новых характеристик для указанного ювелирного изделия. Функция использует оператор INSERT для добавления новых записей в таблицу "Характеристики_ювелирных_изделий", указывая значения для всех трех столбцов: "Код изделия", "имя_характеристики" и "значение_характеристики".

Функция "изменить_значение_характеристики_ювелирного_изделия" предназначена для изменения значения характеристики для указанного ювелирного изделия. Функция использует оператор UPDATE для обновления значения столбца "значение_характеристики" в таблице "Характеристики_ювелирных_изделий" для записи, у которой "Код изделия" и "имя_характеристики" соответствуют заданным значениям. Новое значение характеристики передается в качестве параметра функции "изменить_значение_характеристики_ювелирного_изделия".

Таким образом, данный код обеспечивает добавление и изменение характеристик ювелирных изделий в таблице "Характеристики_ювелирных_изделий". Для вызова функций необходимо передать значения параметров "Код изделия", "имя_характеристики" и "значение_характеристики" в соответствующие функции.

Пример вызова функции

"добавить_характеристику_ювелирного_изделия":

```
SELECT добавить_характеристику_ювелирного_изделия(123,  
'материал', 'золото');
```

Данный запрос добавит новую характеристику "материал" со значением "золото" для ювелирного изделия с "Кодом изделия", равным 123.

Пример вызова функции

"изменить_значение_характеристики_ювелирного_изделия":

```
SELECT  
изменить_значение_характеристики_ювелирного_изделия(123,  
'материал', 'серебро');
```

Данный запрос изменит значение характеристики "материал" с "золото" на "серебро" для ювелирного изделия с "Кодом изделия", равным 123.