Практика по базам данных

ОТЧЁТ

Балашов Илья

243 группа

Предметная область : “Молокозавод”

Содержание

Описание системы 2

Требования 2

Модель данных 2

Функциональность 3

Серверная часть 3

Функции 3

Триггеры 3

Представления 4

Клиентская часть 5

Скрипты 8

Серверная часть 8

Клиентская часть 15

Создание и заполнение базы данных 18

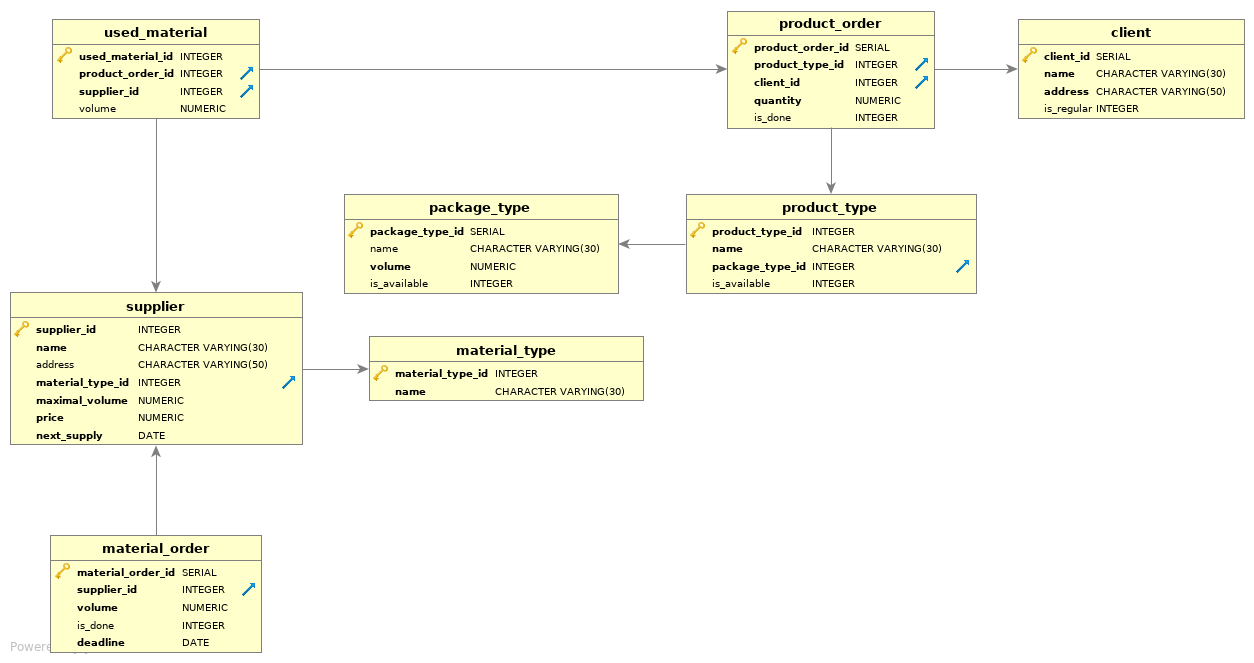
**Описание системы**

**Требования**

Система разрабатывается для внедрения на молокозавод, на котором из поставляемого поставщиками сырья производят потребительские товары. Затем эти товары закупаются клиентами завода – торговыми организациями. Поставщиками являются сельскохозяйственные компании, расположенные в разных районах области, клиентами – городские магазины. Каждый поставщик поставляет некоторое сырье определенных оптовых объемов и стоимости, с определенной регулярностью. В результате переработки получаются конечные продукты разного вида, объемов, в разных упаковках.

Необходимо иметь возможность хранить информацию о поставщиках и клиентах, о сырье, о конечном продукте (включая информацию о сырье, из которого он был произведен), о заводских заказах на сырье и заказах на конечные продукты – от клиентов.

**Модель данных**

****

**Функциональность**

**Серверная часть**

**Функции**

|  |  |  |
| --- | --- | --- |
| **Хранимые функции** | **Реализация** | **Комментарии** |
| Обработка заказа на сырьё | PlaceRawMaterialOrder(материал, поставщик, количество, дедлайн) | - |
| Перевод заказа в статус “выполнен” | - | - |
| Добавление нового поставщика | - | - |
| Среднее количество заказываемого сырья каждого типа | GetAvgOrderNumber() | (тип сырья, средн.количество)  Сортируем по количеству |
| Получение поставщика, который сможет продать нужное сырьё дешевле всех | GetCheapestSupplier(material\_name) | (id\_поставщика, цена) |
| Количество невыполненных заказов на готовую продукцию | - | - |
| Является ли партнер (клиент или поставщик) добросовестным | IsPartnerTrustworthy(partner\_name) | 1 или 0  Поставщики, у которых не более 1 просроченной поставки, и постоянные покупатели |
| ... |  |  |

**Триггеры**

|  |  |  |
| --- | --- | --- |
| **Триггер** | **Реализация** | **Комментарии** |
| Добавление id при добавлении заказа | - | - |
| Если клиент совершил > 3 заказов или купил > 200 единиц товара, сделать его постоянным клиентом | Tr\_Mark\_Regular\_Client | - |
| Проверка на наличие заказываемого сырья у поставщика | Tr\_Check\_Material\_Availability | Смотрим наличие записи в таблице Suppliers |
| Контроль повторного добавления поставщика | - | - |
| Контроль даты следующей поставки у поставщика | - | Для вновь совершаемого заказа должна быть не раньше текущей даты |
| Вместо удаления типа готового продукта помечаем его недоступным | Tr\_Mark\_Unavailable | Используем поле Is\_Available |
| Запрет на удаление клиента | - | Удержание клиентской базы |
| …. |  |  |

**Представления**

|  |  |  |
| --- | --- | --- |
| **Представление** | **Реализация** | **Комментарии** |
| Невыполненные заказы на готовую продукцию | UnfulfilledProductOrders | - |
| Невыполненные заказы на сырьё | UnfulfilledMaterialOrders | - |
| Клиенты с невыполненными заказами | - |  |
| Постоянные клиенты | - |  |
| Доступные виды тары и товары, которые можно в эту тару упаковать | AvailablePackageOptions | (тип упаковки, тип продукта) |
| Список всех, с кем работает молокозавод | PartnerList | Клиенты и поставщики  (название, адрес) |
| ... |  |  |

**Клиентская часть**

|  |  |  |  |
| --- | --- | --- | --- |
| **Экранная форма,**  **основное** | **Дополнительные элементы** | **Запрос** | **Что используем от серверной части** |
| Реестр магазинов (клиентов) |  |  |  |
|  | Новый клиент |  |  |
|  | Изменить данные клиента |  |  |
|  | Удалить клиента |  |  |
|  | Показать только тех, у которых есть невыполненный заказ | (1) (название магазина, количество невыполненных заказов, постоянный ли клиент) | UnfulfilledProductOrders |
| Реестр заказов готовой продукции |  |  |  |
|  | Новый заказ | (2) Создание нового заказа; при этом успешно сработает триггер, который сделает клиента постоянным | Tr\_Mark\_Regular\_Client |
|  | Просмотр информации о заказе |  |  |
|  | Отметить заказ как выполненный |  |  |
| Реестр поставщиков |  |  |  |
|  | Добавить поставщика |  |  |
|  | Удалить поставщика |  |  |
|  | Сколько в среднем заказывают продукции, которую продаёт поставщика | (3) Среднее число упаковок в заказе | GetAvgOrderNumber |
|  | Изменить данные о поставщике |  |  |
| Реестр заказов на сырьё |  |  |  |
|  | Разместить новый заказ |  |  |
|  | Отложить deadline | (4) Переносит дедлайн на неделю после текущей даты |  |
|  | Показать добросовестных поставщиков и тех, кто в “зоне риска” (т.е у кого 2-3 просроченных заказа) | (5) Возвращает все данные о поставщике | IsPartnerTrustworthy |
|  | Обновить дату следующей поставки |  |  |
|  | Получить информацию о самом выгодном предложении по каждому типу сырья | (6) (id\_поставщика,  название поставщика, название сырья, стоимость) | GetCheapestSupplier |
| Реестр типов упаковки |  |  |  |
|  | Добавить тип |  |  |
|  | Показать товары, которые упаковываются в эту тару |  |  |
|  | Показать самую популярную тару (на которую больше всего заказов) | (7) (id тары, название тары, количество заказов) |  |
|  | Изменить статус доступность упаковки | (8) Делает тару недоступной для покупки |  |
|  | Удалить упаковку |  |  |
| Реестр типов готовой продукции |  |  |  |
|  | Добавить новый тип | - |  |
|  | Добавить новую тару для существующего товара | - |  |
|  | Показать типы товаров (без учета тары), на которые нет текущих заказов | (9) (id товара, имя товара, тип упаковки, наличие) | UnfulfilledProductOrders |
|  | Удалить товар | (10) Удаляет товар (но триггер помечает его недоступным) | Tr\_Mark\_Unavailable |
| Служебные функции |  |  |  |
|  | Список постоянных клиентов и добросовестных поставщиков,  кроме покупателей, у которых нет текущих заказов | (11) (Название компании) | IsPartnerTrustworthy  PartnerList |
|  | Добросовестным клиентам поставить в соответствие  наименование в самом крупном (по количеству) заказе | (12) (название клиента, наименование товара в самом крупном заказе) | IsPartnerTrustworthy |
| ... |  |  |  |

**Скрипты**

**Серверная часть**

-----------------------------------------------

-------- Процедуры и функции --------

-----------------------------------------------

------------------------------------------------

--- Обработка нового заказа на сырьё

------------------------------------------------

CREATE OR REPLACE FUNCTION PlaceRawMaterialOrder

(

material CHARACTER VARYING,

supplier\_name CHARACTER VARYING,

volume NUMERIC,

supply\_deadline DATE

)

RETURNS void AS $$

BEGIN

INSERT INTO Material\_Order(material\_order\_id, supplier\_id, volume, is\_done, deadline)

VALUES

(

(SELECT MAX(material\_order\_id)+1 FROM Material\_Order),

(SELECT supplier\_id FROM Supplier, Material\_Type

WHERE Supplier.Name = supplier\_name

AND Material\_Type.Material\_Type\_ID = Supplier.Material\_Type\_ID

AND Material\_Type.Name = material

LIMIT 1),

volume,

0,

supply\_deadline

);

END;

$$ LANGUAGE plpgsql;

--------- Пример вызова ---------

SELECT PlaceRawMaterialOrder

(

CAST('Клубничный джем' AS CHARACTER VARYING),

CAST('Совхоз им. Ленина' AS CHARACTER VARYING),

10.0,

CAST('02-02-1999' AS DATE)

);

---------------------------------------------------------------------------

--- Среднее количество заказываемого сырья каждого типа

---------------------------------------------------------------------------

CREATE OR REPLACE FUNCTION GetAvgOrderNumber()

RETURNS TABLE (

Material\_Name CHARACTER VARYING,

Avg\_Volume NUMERIC

) AS $$

BEGIN

RETURN QUERY

SELECT Material\_Type.name, AVG(COALESCE(Orders.Volume, 0)) AS Avg\_Volume

FROM

Material\_Type,

(SELECT \* FROM Supplier LEFT JOIN Material\_Order

ON Supplier.Supplier\_ID = Material\_Order.Supplier\_ID) AS Orders

WHERE Material\_Type.Material\_Type\_ID = Orders.Material\_Type\_ID

GROUP BY Material\_Type.name

ORDER BY Avg\_Volume DESC;

END;

$$ LANGUAGE plpgsql;

--------- Пример вызова ---------

SELECT \* FROM GetAvgOrderNumber();

--------------------------------------------------------------------------------------------------------

--- Получение поставщика, который сможет продать нужное сырьё дешевле всех

--------------------------------------------------------------------------------------------------------

CREATE OR REPLACE FUNCTION GetCheapestSupplier(material\_name CHARACTER VARYING)

RETURNS TABLE (

Supplier\_ID INT,

Supplier\_Name CHARACTER VARYING,

Offer\_Name CHARACTER VARYING,

Price NUMERIC

) AS $$

BEGIN

RETURN QUERY

SELECT Supplier.Supplier\_ID, Supplier.Name, Material\_Type.Name, Supplier.Price FROM Supplier, Material\_Type

WHERE Material\_Type.Material\_Type\_ID = Supplier.Material\_Type\_ID

AND Material\_Type.Name = material\_name

ORDER BY Supplier.Price LIMIT 1;

END;

$$ LANGUAGE plpgsql;

--------- Пример вызова ---------

SELECT \* FROM GetCheapestSupplier('Молоко сырое');

--------------------------------------------------------

--- Является ли поставщик добросовестным

--------------------------------------------------------

CREATE OR REPLACE FUNCTION IsPartnerTrustworthy(partner\_name CHARACTER VARYING)

RETURNS INT

AS $$

DECLARE is\_partner\_trustworthy INT;

BEGIN

SELECT COUNT(\*) INTO is\_partner\_trustworthy FROM

(SELECT Supplier.Name AS Name

FROM Supplier JOIN Material\_Order

ON Material\_Order.Supplier\_ID = Supplier.Supplier\_ID

AND Material\_Order.Deadline < CURRENT\_DATE

AND Material\_Order.Is\_Done = 0

GROUP BY Supplier.Name

HAVING COUNT(\*) < 2

UNION

SELECT Client.Name AS Name FROM Client

WHERE Client.Is\_Regular = 1) AS Trustworthy

WHERE Trustworthy.Name = partner\_name;

IF is\_partner\_trustworthy > 0 THEN

RETURN 1;

ELSE

RETURN 0;

END IF;

END;

$$ LANGUAGE plpgsql;

--------- Пример вызова ---------

SELECT IsPartnerTrustworthy('Андреевский'); -- 1

SELECT IsPartnerTrustworthy('Ферма "Ласточка"'); -- 1

SELECT IsPartnerTrustworthy('Совхоз им. Ленина'); -- 0

-------------------------------

-------- Триггеры ---------

-------------------------------

------------------------------------------------

--- Делаем клиента постоянным

------------------------------------------------

CREATE OR REPLACE FUNCTION Mark\_Regular\_Client()

RETURNS TRIGGER

AS $$

DECLARE

client\_orders INT;

client\_quantity INT;

client\_regular INT;

BEGIN

SELECT COUNT(\*), SUM(quantity) INTO client\_orders, client\_quantity FROM Product\_Order

WHERE Client\_ID = NEW.Client\_ID;

SELECT Is\_Regular INTO client\_regular FROM Client

WHERE Client\_ID = NEW.Client\_ID;

IF (client\_orders > 3 OR client\_quantity > 200) AND (client\_regular = 0) THEN

UPDATE Client SET Is\_Regular = 1 WHERE Client\_ID = NEW.Client\_ID;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER Tr\_Mark\_Regular\_Client AFTER INSERT OR UPDATE ON Product\_Order

FOR EACH ROW EXECUTE PROCEDURE Mark\_Regular\_Client();

--------- Пример вызова ---------

INSERT INTO Product\_Order(product\_type\_id, client\_id, quantity, is\_done)

VALUES(2, 2, 200, 0);

------------------------------------------------------------------------------

--- Проверка на наличие заказываемого сырья у поставщика

------------------------------------------------------------------------------

CREATE OR REPLACE FUNCTION Check\_Material\_Availability()

RETURNS TRIGGER

AS $$

BEGIN

IF NEW.Volume > (SELECT maximal\_volume FROM Supplier

WHERE Supplier\_ID = NEW.Supplier\_ID) THEN

RAISE EXCEPTION 'Ordered volume % from supplier % is higher than possible',

NEW.Volume, NEW.Supplier\_ID;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER Tr\_Check\_Material\_Availability BEFORE INSERT OR UPDATE ON Material\_Order

FOR EACH ROW EXECUTE PROCEDURE Check\_Material\_Availability();

--------- Пример вызова ---------

INSERT INTO Material\_Order(Supplier\_ID, Volume, Is\_Done, Deadline)

VALUES(2, 250, 0, '2048-02-02');

-----------------------------------------------------------------------------

--- Проверка на наличие заказываемого сырья у поставщика

-----------------------------------------------------------------------------

CREATE OR REPLACE FUNCTION Mark\_Unavailable()

RETURNS TRIGGER

AS $$

BEGIN

UPDATE Product\_Type SET Is\_Available = 0 WHERE Product\_Type\_ID = OLD.Product\_Type\_ID;

RETURN NULL;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER Tr\_Mark\_Unavailable BEFORE DELETE ON Product\_Type

FOR EACH ROW EXECUTE PROCEDURE Mark\_Unavailable();

--------- Пример вызова ---------

DELETE FROM Product\_Type WHERE Name = 'Творог 5%';

-------------------------------------

-------- Представления --------

-------------------------------------

----------------------------------------------------------------

--- Невыполненные заказы на готовую продукцию

----------------------------------------------------------------

CREATE VIEW UnfulfilledProductOrders AS

SELECT

Product\_Order.Product\_Order\_Id,

Product\_Type.Name AS Product\_Name,

Client.Client\_ID AS Client\_ID,

Client.Name AS Client\_Name,

Product\_Order.quantity

FROM Product\_Order, Client, Product\_Type

WHERE Product\_Order.Is\_Done = 0

AND Client.Client\_ID = Product\_Order.Client\_ID

AND Product\_Type.Product\_Type\_ID = Product\_Order.Product\_Type\_ID;

--------- Пример вызова ---------

SELECT MAX(quantity) FROM UnfulfilledProductOrders;

------------------------------------------------

--- Невыполненные заказы на сырьё

------------------------------------------------

CREATE VIEW UnfulfilledMaterialOrders AS

SELECT

Material\_Order.Material\_Order\_ID,

Supplier.Name AS Supplier\_Name,

Material\_Order.Volume,

Material\_Order.Deadline

FROM Material\_Order JOIN Supplier

ON Material\_Order.Supplier\_ID = Supplier.Supplier\_ID

WHERE Material\_Order.Is\_Done = 0;

--------- Пример вызова ---------

SELECT MAX(volume) FROM UnfulfilledMaterialOrders;

------------------------------------------------------------------------------------------------

--- Доступные виды тары и товары, которые можно в эту тару упаковать

------------------------------------------------------------------------------------------------

CREATE VIEW AvailablePackageOptions AS

SELECT

Package\_Type.name AS Package,

Package\_Type.Volume AS Volume,

Product\_Type.name AS Product

FROM Package\_Type LEFT JOIN Product\_Type

ON Package\_Type.Package\_Type\_ID = Product\_Type.Package\_Type\_ID;

--------- Пример вызова ---------

SELECT \* FROM AvailablePackageOptions;

---------------------------------------------------------

--- Список всех, с кем работает молокозавод

---------------------------------------------------------

CREATE VIEW PartnerList AS

SELECT

Supplier.Name AS Name,

Supplier.Address AS Address

FROM Supplier JOIN Material\_Order

ON Material\_Order.Supplier\_ID = Supplier.Supplier\_ID

UNION

SELECT

Client.Name AS Name,

Client.Address AS Address

FROM Client

--------- Пример вызова ---------

SELECT \* FROM PartnerList WHERE LEFT(Name, 1) = 'П';

-----------------------------------------

--- Удаление представлений

-----------------------------------------

/\*

DROP VIEW PartnerList;

DROP VIEW AvailablePackageOptions;

DROP VIEW UnfulfilledMaterialOrders;

DROP VIEW UnfulfilledProductOrders;

\*/

**Клиентская часть**

--- (1) Показать только тех, у которых есть невыполненный заказ

SELECT Client\_Data.Name, Orders.Unfulfilled\_Count, Client.Is\_Regular

FROM

Client,

(SELECT Client\_ID, Name, Is\_Regular FROM Client) AS Client\_Data,

(SELECT Client\_ID, COUNT(\*) AS Unfulfilled\_Count

FROM UnfulfilledProductOrders GROUP BY Client\_ID) AS Orders

WHERE Client\_Data.Client\_ID = Client.Client\_ID

AND Orders.Client\_ID = Client\_Data.Client\_ID;

--- (2) Новый заказ на готовую продукцию (на сервере сработает триггер)

INSERT INTO Product\_Order(product\_type\_id, client\_id, quantity, is\_done)

VALUES(2, 3, 240, 0);

--- (3) Среднее число упаковок в заказе (в целом)

SELECT AVG(Avg\_Volume) FROM GetAvgOrderNumber();

--- (4) Отложить дедлайн на неделю после текущей даты

UPDATE Material\_Order SET Deadline = CURRENT\_DATE + INTERVAL '1 week'

WHERE Material\_Order\_ID = 7;

--- (5) Показать добросоветсных партнеров и тех поставщиков, кто в "зоне риска"

SELECT \* FROM Supplier

WHERE IsPartnerTrustworthy(Supplier.Name) = 1

OR Supplier.Supplier\_ID IN (SELECT Supplier.Supplier\_ID AS ID

FROM Supplier JOIN Material\_Order

ON Material\_Order.Supplier\_ID = Supplier.Supplier\_ID

AND Material\_Order.Deadline < CURRENT\_DATE

AND Material\_Order.Is\_Done = 0

GROUP BY Supplier.Supplier\_ID

HAVING COUNT(\*) >= 2 AND COUNT(\*) <= 3);

--- (6) Получить информацию о самом выгодном предложении по каждому типу сырья

CREATE OR REPLACE FUNCTION GetCheapestOffers(material\_ids INT[])

RETURNS TABLE (

Offer\_ID INT,

Offer\_Supplier CHARACTER VARYING,

Material\_Name CHARACTER VARYING,

Offer\_Price NUMERIC

)

AS $$

DECLARE

element INT;

BEGIN

FOREACH element IN ARRAY material\_ids LOOP

RETURN QUERY SELECT Supplier\_ID, Supplier\_Name, Offer\_Name, Price

FROM GetCheapestSupplier((SELECT Name FROM Material\_Type WHERE Material\_Type\_ID = element)) LIMIT 1;

END LOOP;

END;

$$ LANGUAGE plpgsql;

SELECT \* FROM GetCheapestOffers(ARRAY(SELECT Material\_Type\_ID FROM Material\_Type));

--- (7) Показать самую популярную тару

SELECT Package\_Type.Package\_Type\_ID, Package\_Type.Name, COUNT(\*)

FROM Package\_Type, Product\_Type, Product\_Order

WHERE Product\_Type.Product\_Type\_ID = Product\_Order.Product\_Type\_ID

AND Package\_Type.Package\_Type\_ID = Product\_Type.Package\_Type\_ID

GROUP BY Package\_Type.Package\_Type\_ID, Package\_Type.Name

ORDER BY COUNT(\*) DESC

LIMIT 1;

--- (8) Изменить статус доступности упаковки

UPDATE Package\_Type SET Is\_Available = 0

WHERE Package\_Type.Package\_Type\_ID = (SELECT Package\_Type\_ID FROM Package\_Type WHERE Name = 'Тетра пак');

--- (9) Показать типы товаров (без учета тары), на которые нет текущих заказов

SELECT \* FROM Product\_Type

WHERE Product\_Type.Name NOT IN (SELECT Product\_Name FROM UnfulfilledProductOrders);

--- (10) Удалить тип товара (задействуется триггер и вместо удаление пометит товар недоступным)

DELETE FROM Product\_Type WHERE Name = 'Творог 5%';

/\* Без триггера запрос был бы некорректным из-за наличия внешних ключей,

ссылающихся на строку \*/

--- (11) Список постоянных клиентов и добросовестных поставщиков,

--- кроме покупателей, у которых нет текущих заказов

SELECT DISTINCT Name FROM PartnerList WHERE IsPartnerTrustworthy(PartnerList.Name) = 1

EXCEPT

SELECT DISTINCT Name FROM Client

WHERE Client.Client\_ID NOT IN (SELECT UnfulfilledProductOrders.Client\_ID FROM UnfulfilledProductOrders);

--- (12) Добросовестным клиентам поставить в соответствие

--- наименование в самом крупном (по количеству) заказе

SELECT

Client.Name,

(SELECT Product\_Type.Name FROM Product\_Order, Product\_Type

WHERE Product\_Order.Client\_ID = Client.Client\_ID

AND Product\_Type.Product\_Type\_ID = Product\_Order.Product\_Type\_ID

ORDER BY Product\_Order.Quantity DESC

LIMIT 1) AS Biggest\_Order

FROM Client

WHERE IsPartnerTrustworthy(Client.Name) = 1

**Создание и заполнение базы данных**

-- Database: dairy\_plant

----------------------------------------

--- Создание базы данных

----------------------------------------

/\*

CREATE DATABASE dairy\_plant

WITH OWNER = postgres

ENCODING = 'UTF8'

TABLESPACE = pg\_default

LC\_COLLATE = 'ru\_RU.UTF-8'

LC\_CTYPE = 'ru\_RU.UTF-8'

CONNECTION LIMIT = -1;

\*/

-- DROP DATABASE dairy\_plant;

----------------------------------------

--- Создание таблиц и PK

----------------------------------------

CREATE TABLE Material\_Type(

Material\_Type\_ID INTEGER NOT NULL,

Name VARCHAR(30) NOT NULL,

CONSTRAINT Material\_Type\_PK PRIMARY KEY(Material\_Type\_ID)

)

;

CREATE TABLE Product\_Type(

Product\_Type\_ID INTEGER NOT NULL,

Name VARCHAR(30) NOT NULL,

Package\_Type\_ID INTEGER NOT NULL,

Is\_Available INTEGER DEFAULT 1 CHECK(Is\_Available in (0, 1)),

CONSTRAINT Product\_Type\_PK PRIMARY KEY(Product\_Type\_ID)

)

;

CREATE TABLE Client(

Client\_ID SERIAL,

Name VARCHAR(30) NOT NULL,

Address VARCHAR(50) NOT NULL,

Is\_Regular INTEGER DEFAULT 0 CHECK(Is\_Regular in (0, 1)),

CONSTRAINT Client\_PK PRIMARY KEY(Client\_ID)

)

;

CREATE TABLE Supplier(

Supplier\_ID INTEGER NOT NULL,

Name VARCHAR(30) NOT NULL,

Address VARCHAR(50),

Material\_Type\_ID INTEGER NOT NULL,

Maximal\_Volume DECIMAL NOT NULL,

Price DECIMAL NOT NULL,

Next\_Supply DATE DEFAULT CURRENT\_DATE NOT NULL,

CONSTRAINT Supplier\_PK PRIMARY KEY(Supplier\_ID)

)

;

CREATE TABLE Material\_Order(

Material\_Order\_ID SERIAL,

Supplier\_ID INTEGER NOT NULL,

Volume DECIMAL NOT NULL,

Is\_Done INTEGER CHECK(Is\_Done in (0, 1)),

Deadline DATE DEFAULT CURRENT\_DATE NOT NULL,

CONSTRAINT Material\_Order\_PK PRIMARY KEY(Material\_Order\_ID)

)

;

CREATE TABLE Product\_Order(

Product\_Order\_ID SERIAL,

Product\_Type\_ID INTEGER NOT NULL,

Client\_ID INTEGER NOT NULL,

Quantity DECIMAL NOT NULL,

Is\_Done INTEGER CHECK(Is\_Done in (0, 1)),

CONSTRAINT Product\_Order\_PK PRIMARY KEY(Product\_Order\_ID)

)

;

CREATE TABLE Used\_Material(

Used\_Material\_ID INTEGER NOT NULL,

Product\_Order\_ID INTEGER NOT NULL,

Supplier\_ID INTEGER NOT NULL,

Volume DECIMAL,

CONSTRAINT Used\_Material\_PK PRIMARY KEY(Used\_Material\_ID)

)

;

CREATE TABLE Package\_Type(

Package\_Type\_ID SERIAL,

Name VARCHAR(30),

Volume DECIMAL NOT NULL,

Is\_Available INTEGER DEFAULT 1 CHECK(Is\_Available in (0, 1)),

CONSTRAINT Package\_Type\_PK PRIMARY KEY(Package\_Type\_ID)

)

;

----------------------------------------

--- Создание foreign key

----------------------------------------

ALTER TABLE Product\_Order ADD CONSTRAINT FK\_Product\_Order\_Type

FOREIGN KEY (Product\_Type\_ID)

REFERENCES Product\_Type(Product\_Type\_ID)

;

ALTER TABLE Product\_Order ADD CONSTRAINT FK\_Product\_Order\_Client

FOREIGN KEY (Client\_ID)

REFERENCES Client(Client\_ID) ON DELETE CASCADE

;

ALTER TABLE Material\_Order ADD CONSTRAINT FK\_Material\_Order\_Supplier

FOREIGN KEY (Supplier\_ID)

REFERENCES Supplier(Supplier\_ID) ON DELETE CASCADE

;

ALTER TABLE Used\_Material ADD CONSTRAINT FK\_Used\_Material\_Product\_Order

FOREIGN KEY (Product\_Order\_ID)

REFERENCES Product\_Order(Product\_Order\_ID) ON DELETE CASCADE

;

ALTER TABLE Used\_Material ADD CONSTRAINT FK\_Used\_Material\_Supplier

FOREIGN KEY (Supplier\_ID)

REFERENCES Supplier(Supplier\_ID) ON DELETE CASCADE

;

ALTER TABLE Product\_Type ADD CONSTRAINT FK\_Product\_Type\_Package\_Type

FOREIGN KEY (Package\_Type\_ID)

REFERENCES Package\_Type(Package\_Type\_ID) ON DELETE CASCADE

;

ALTER TABLE Supplier ADD CONSTRAINT FK\_Supplier\_Material\_Type

FOREIGN KEY (Material\_Type\_ID)

REFERENCES Material\_Type(Material\_Type\_ID) ON DELETE CASCADE

;

----------------------------------------

--- Создание индексов

----------------------------------------

CREATE INDEX Supplier\_Index ON Supplier(Material\_Type\_ID, Price, Next\_Supply);

CREATE INDEX Product\_Order\_Index ON Product\_Order(Product\_Type\_ID, Quantity);

CREATE INDEX Material\_Order\_Index ON Material\_Order(Supplier\_ID, Volume, Deadline);

CREATE INDEX Product\_Type\_Index ON Product\_Type(Package\_Type\_ID);

CLUSTER Product\_Type USING Product\_Type\_Index;

----------------------------------------

--- Заполнение данными

----------------------------------------

INSERT INTO Material\_Type(Material\_Type\_ID, Name) VALUES (1, 'Молоко сырое');

INSERT INTO Material\_Type(Material\_Type\_ID, Name) VALUES (2, 'Клубничный джем');

INSERT INTO Material\_Type(Material\_Type\_ID, Name) VALUES (3, 'Сахар');

INSERT INTO Material\_Type(Material\_Type\_ID, Name) VALUES (4, 'Эмульгатор');

INSERT INTO Material\_Type(Material\_Type\_ID, Name) VALUES (5, 'Ванилин');

INSERT INTO Material\_Type(Material\_Type\_ID, Name) VALUES (6, 'Изюм');

INSERT INTO Material\_Type(Material\_Type\_ID, Name) VALUES (7, 'Закваска');

INSERT INTO Supplier(Supplier\_ID, Name, Address, Material\_Type\_Id, Maximal\_Volume, Price, Next\_Supply)

VALUES (1, 'Совхоз им. Ленина', 'Гатчинский район', 1, 100, 30, '2019-07-20');

INSERT INTO Supplier(Supplier\_ID, Name, Address, Material\_Type\_Id, Maximal\_Volume, Price, Next\_Supply)

VALUES (2, 'Совхоз им. Ленина', 'Гатчинский район', 2, 231, 70, '2019-06-15');

INSERT INTO Supplier(Supplier\_ID, Name, Address, Material\_Type\_Id, Maximal\_Volume, Price, Next\_Supply)

VALUES (3, 'Ферма "Ласточка"', 'Волосовский район', 2, 122, 69, '2019-05-10');

INSERT INTO Supplier(Supplier\_ID, Name, Address, Material\_Type\_Id, Maximal\_Volume, Price, Next\_Supply)

VALUES (4, 'ИП Петрович А.А.', 'Кингисеппский район', 4, 1000, 10, '2019-06-01');

INSERT INTO Supplier(Supplier\_ID, Name, Address, Material\_Type\_Id, Maximal\_Volume, Price, Next\_Supply)

VALUES (5, 'Совхоз №15', 'Тихвинский район', 1, 1321, 31, '2019-05-31');

INSERT INTO Supplier(Supplier\_ID, Name, Address, Material\_Type\_Id, Maximal\_Volume, Price, Next\_Supply)

VALUES (6, 'Совхоз им. Ленина', 'Гатчинский район', 5, 12312, 12, '2019-05-31');

INSERT INTO Supplier(Supplier\_ID, Name, Address, Material\_Type\_Id, Maximal\_Volume, Price, Next\_Supply)

VALUES (7, 'ООО "Легион"', 'Всеволжский район', 3, 1333, 35, '2019-06-01');

INSERT INTO Supplier(Supplier\_ID, Name, Address, Material\_Type\_Id, Maximal\_Volume, Price, Next\_Supply)

VALUES (8, 'ООО "Легион"', 'Всеволжский район', 5, 666, 12, '2019-05-30');

INSERT INTO Supplier(Supplier\_ID, Name, Address, Material\_Type\_Id, Maximal\_Volume, Price, Next\_Supply)

VALUES (9, 'Ферма "Ласточка"', 'Волосовский район', 7, 23, 67, '2019-05-08');

INSERT INTO Package\_Type(Name, Volume) VALUES ('Тетра пак', 1.0);

INSERT INTO Package\_Type(Name, Volume) VALUES ('Пюр пак', 0.97);

INSERT INTO Package\_Type(Name, Volume) VALUES ('Фольга оберточная', 0.180);

INSERT INTO Package\_Type(Name, Volume) VALUES ('Полиэтиленовая пленка', 0.25);

INSERT INTO Package\_Type(Name, Volume, Is\_Available) VALUES ('Стаканчик пластиковый', 0.2, 0);

INSERT INTO Package\_Type(Name, Volume) VALUES ('Стаканчик пластиковый', 0.5);

INSERT INTO Product\_Type(Product\_Type\_ID, Name, Package\_Type\_ID) VALUES (0, 'Молоко 1.5%', 1);

INSERT INTO Product\_Type(Product\_Type\_ID, Name, Package\_Type\_ID) VALUES (1, 'Молоко 1.5%', 2);

INSERT INTO Product\_Type(Product\_Type\_ID, Name, Package\_Type\_ID) VALUES (2, 'Молоко 2.5%', 1);

INSERT INTO Product\_Type(Product\_Type\_ID, Name, Package\_Type\_ID) VALUES (3, 'Молоко 3%', 2);

INSERT INTO Product\_Type(Product\_Type\_ID, Name, Package\_Type\_ID) VALUES (4, 'Творог обезжиренный', 3);

INSERT INTO Product\_Type(Product\_Type\_ID, Name, Package\_Type\_ID) VALUES (5, 'Творог обезжиренный', 4);

INSERT INTO Product\_Type(Product\_Type\_ID, Name, Package\_Type\_ID) VALUES (6, 'Творог 5%', 4);

INSERT INTO Product\_Type(Product\_Type\_ID, Name, Package\_Type\_ID) VALUES (7, 'Творог 5% c джемом', 5);

INSERT INTO Product\_Type(Product\_Type\_ID, Name, Package\_Type\_ID) VALUES (8, 'Творожная масса с изюмом', 4);

INSERT INTO Client(Client\_ID, Name, Address, Is\_Regular) VALUES (1, 'Андреевский', 'Петергоф, Ботаническая 66к.3', 1);

INSERT INTO Client(Client\_ID, Name, Address) VALUES (2, 'Лента', 'Петергоф, Гостилицкое шоссе 58');

INSERT INTO Client(Client\_ID, Name, Address) VALUES (3, 'Перекрёсток', 'СПб, Лиговский проспект 30А');

INSERT INTO Client(Client\_ID, Name, Address) VALUES (4, 'Перекрёсток', 'Петергоф, Гостилицкая улица 2А');

INSERT INTO Client(Client\_ID, Name, Address) VALUES (5, 'Пятерочка', 'Петергоф, Чичеринская улица 2');

INSERT INTO Client(Client\_ID, Name, Address, Is\_Regular) VALUES (6, 'Кафе "Яма"', 'Петергоф, Ульяновская улица 28', 1);

INSERT INTO Product\_Order(Product\_Type\_ID, Client\_ID, Quantity, Is\_Done)

VALUES (3, 3, 50, 0);

INSERT INTO Product\_Order(Product\_Type\_ID, Client\_ID, Quantity, Is\_Done)

VALUES (6, 5, 100, 1);

INSERT INTO Product\_Order(Product\_Type\_ID, Client\_ID, Quantity, Is\_Done)

VALUES (4, 3, 75, 0);

INSERT INTO Product\_Order(Product\_Type\_ID, Client\_ID, Quantity, Is\_Done)

VALUES (3, 2, 100, 0);

INSERT INTO Product\_Order(Product\_Type\_ID, Client\_ID, Quantity, Is\_Done)

VALUES (2, 1, 42, 1);

INSERT INTO Material\_Order(Supplier\_ID, Volume, Is\_Done, Deadline) -- 1

VALUES (4, 100, 0, '2019-04-15');

INSERT INTO Material\_Order(Supplier\_ID, Volume, Is\_Done, Deadline) -- 2

VALUES (1, 1000, 1, '2019-05-21');

INSERT INTO Material\_Order(Supplier\_ID, Volume, Is\_Done, Deadline) -- 3

VALUES (1, 100, 0, '2019-06-01');

INSERT INTO Material\_Order(Supplier\_ID, Volume, Is\_Done, Deadline) -- 4

VALUES (3, 100, 0, '2019-04-20');

INSERT INTO Material\_Order(Supplier\_ID, Volume, Is\_Done, Deadline) -- 5

VALUES (2, 230, 1, '2018-10-10');

INSERT INTO Material\_Order(Supplier\_ID, Volume, Is\_Done, Deadline) -- 6

VALUES (2, 130, 0, '2019-04-15');

INSERT INTO Material\_Order(Supplier\_ID, Volume, Is\_Done, Deadline) -- 7

VALUES (2, 130, 0, '2019-04-14');

INSERT INTO Used\_Material(Used\_Material\_ID, Product\_Order\_ID, Supplier\_ID, Volume)

VALUES (1, 1, 1, 50);

INSERT INTO Used\_Material(Used\_Material\_ID, Product\_Order\_ID, Supplier\_ID, Volume)

VALUES (2, 2, 1, 900);

INSERT INTO Used\_Material(Used\_Material\_ID, Product\_Order\_ID, Supplier\_ID, Volume)

VALUES (3, 2, 9, 100);

INSERT INTO Used\_Material(Used\_Material\_ID, Product\_Order\_ID, Supplier\_ID, Volume)

VALUES (4, 3, 5, 70);

INSERT INTO Used\_Material(Used\_Material\_ID, Product\_Order\_ID, Supplier\_ID, Volume)

VALUES (5, 3, 9, 5);

INSERT INTO Used\_Material(Used\_Material\_ID, Product\_Order\_ID, Supplier\_ID, Volume)

VALUES (6, 4, 1, 100);

INSERT INTO Used\_Material(Used\_Material\_ID, Product\_Order\_ID, Supplier\_ID, Volume)

VALUES (7, 5, 1, 42);

----------------------------------------

--- Удаление таблиц

----------------------------------------

/\*

DROP TABLE Material\_Order;

DROP TABLE Used\_Material;

DROP TABLE Product\_Order;

DROP TABLE Product\_Type;

DROP TABLE Client;

DROP TABLE Supplier;

DROP TABLE Material\_Type;

DROP TABLE Package\_Type;

\*/