Практика по базам данных

ОТЧЕТ

Береснёв Руслан

Группа 21.Б10-мм

Предметная область: «Молокозавод»

Реализация: MS SQLServer

Содержание

[ОПИСАНИЕ СИСТЕМЫ 2](#_heading=h.gjdgxs)

[Требования 2](#_heading=h.30j0zll)

[Модель данных 2](#_heading=h.1fob9te)

[Функциональность 3](#_heading=h.3znysh7)

[Серверная часть 3](#_heading=h.2et92p0)

[Клиентская часть 4](#_heading=h.3dy6vkm)

[СКРИПТЫ 5](#_heading=h.1t3h5sf)

[Серверная часть 5](#_heading=h.4d34og8)

[Хранимые процедуры и функции 5](#_heading=h.2s8eyo1)

[Триггеры 6](#_heading=h.17dp8vu)

[Представления 6](#_heading=h.3rdcrjn)

[Клиентская часть 7](#_heading=h.26in1rg)

[ПРИЛОЖЕНИЕ: Создание и заполнение базы данных 10](#_heading=h.lnxbz9)

# ОПИСАНИЕ СИСТЕМЫ

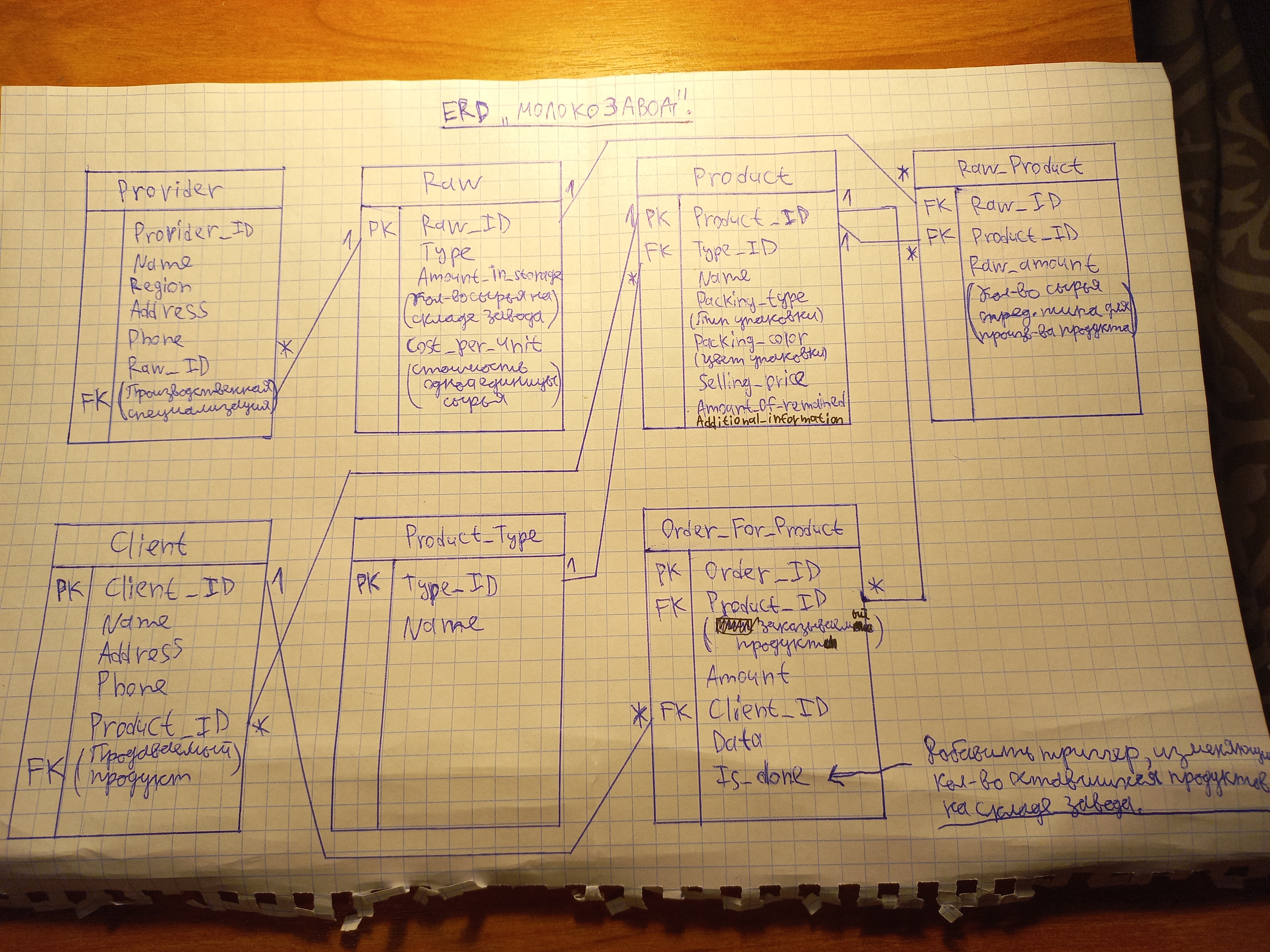
## Требования

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

|  |  |  |
| --- | --- | --- |
| ***Хранимые процедуры\функции*** | *Реализация* | *Комментарии* |
| Добавить поставщика | AddNewProvider() | (имя, регион, адрес, номер телефона, id производимой продукции) |
| Добавить клиента |  |  |
| Создать заказ |  |  |
| Выполнить заказ | TryToCompleteOrder() | (id заказа)  Перевод в состояние «выполнен», если на складе достаточно произведённой продукции |
| Удалить заказ |  |  |
| Получить количество заказов мороженого | GetNumberOfIceCreamOrders() | Возвращает общее число заказов на различное мороженое |
| Изменить тип и количество сырья, нужного для производства конкретного продукта |  |  |
| … |  |  |

|  |  |  |
| --- | --- | --- |
| ***Триггеры*** | *Реализация* | *Комментарии* |
| Запрет на удаление поставщика | TR\_Del\_Provider | Для сохранности базы поставщиков |
| Запрет на удаление клиента | TR\_Del\_Client | Для сохранности базы клиентов |
| Триггер, срабатывающий, когда количество заказов конкретного продукта превысило 30 | TR\_Ins\_Order\_For\_Product | Добавляет пометку о популярности продукта к дополнительной информации о нём |
| Формирование номера нового заказа при его добавлении |  | с использованием секвенции |
| … |  |  |

|  |  |  |
| --- | --- | --- |
| ***Представления*** | *Реализация* | *Комментарии* |
| Количество вариаций продуктов каждого типа | V\_NumberOfProductVariations | (тип продукта, количество продуктов данного типа) – например, две марки творога |
| Вся информация о заказах |  |  |
| Поставщики и поставляемое ими сырьё | V\_ProvidersAndRaw | (ID поставщика, имя, регион, адрес, номер телефона, тип пост-го сырья, кол-во единиц сырья на складе завода, цена закупки единицы сырья) |
| Клиенты и заказываемые ими продукты |  |  |
| Используемое сырьё и его количество для производства всех сырных продуктов |  |  |
| Список клиентов по количеству заказанных ими за всё время единиц продукции, где учитываются только клиент, заказавшие больше 50 единиц продукции | V\_ClientsAndNumberOfAnyProductionUnits | (Название клиента, общее количество единиц заказанной им продукции) |
| … |  |  |

### Клиентская часть

|  |  |  |  |
| --- | --- | --- | --- |
| *Экранные формы основные* | *дополнительные* | *Реализация*  *(запрос)* | *Что здесь можно использовать из серверной части* |
| Реестр поставщиков |  |  |  |
|  |  | (01) Данные о поставщиках из Зеленодольска |  |
|  |  | (02) Поставщики, у которых цена единицы сырья меньше 50 | V\_ProvidersAndRaw |
|  |  | (03) Информация о поставщиках и поставляемом сырье из Ульяновского района и Зеленодольского района | V\_ProvidersAndRaw |
| Реестр клиентов |  |  |  |
|  |  | (04) Названия всех продуктов и их цены у клиентов, названия которых содержат букву ‘n’ |  |
|  |  | (05) Список названий клиентов, которые заказывают продукцию в бумажном пакете, а также тип данной продукции и её название |  |
|  |  | (06) Сумая маленькая сумма единиц заказанной продукции среди всех клиентов |  |
| Информация о продуктах |  |  |  |
|  |  | (07) Количество вариаций продуктов каждого типа | V\_NumberOfProductVariations |
|  |  | (08) Средняя цена продукции в картонной упаковке |  |
|  |  | (09) Продукция с ценой выше 70, имеющая дополнительную информацию о себе |  |
| Реестр заказов |  |  |  |
|  |  | (10) Номера выполненных заказов и количество заказанной продукции в заказах, где это количество больше 90 (отсортировано по возрастанию количества заказанной продукции) |  |
|  |  | (11) Заказы с непустой информацией, созданные клиентами, располагающимися в квартирах с номером 10 |  |
| Техническая информация |  |  |  |
|  |  | (12) Названия всех продуктов, для производства которых используется молоко |  |

# СКРИПТЫ

## Серверная часть

### Хранимые процедуры и функции

-- Добавление нового поставщика

GO

CREATE PROCEDURE AddNewProvider

@var\_name AS CHAR(20),

@var\_region AS CHAR(20),

@var\_address AS CHAR(20),

@var\_phone AS CHAR(20),

@var\_raw\_ID AS INTEGER

AS

BEGIN

DECLARE @var\_new\_provider\_id INTEGER;

SELECT @var\_new\_provider\_id = MAX(Provider\_ID) + 1 FROM Provider;

INSERT INTO Provider (Provider\_ID, Name, Region, Address, Phone, Raw\_ID)

VALUES (@var\_new\_provider\_id, @var\_name, @var\_region, @var\_address, @var\_phone, @var\_raw\_ID);

END;

-- Пример вызова процедуры:

-- EXECUTE AddNewOwner @var\_name = 'Fine Company', @var\_region = 'Zelenodolsk district', @var\_address = 'Ul. Podgornaya, 1-10', @var\_phone = '+79854563983', @var\_raw\_ID = 1;

-- DROP PROCEDURE AddNewProvider;

-- Выполнить заказ (если на складе завода достаточное количество единиц заказанного продукта)

GO

CREATE PROCEDURE TryToCompleteOrder

@var\_order\_id AS INTEGER

AS

BEGIN

DECLARE @var\_product\_id INTEGER;

SELECT @var\_product\_id = Product\_ID FROM Order\_For\_Product WHERE Order\_ID = @var\_order\_id;

DECLARE @var\_amount INTEGER;

SELECT @var\_amount = Amount FROM Order\_For\_Product WHERE Order\_ID = @var\_order\_id;

DECLARE @var\_amount\_of\_remained INTEGER;

SELECT @var\_amount\_of\_remained = Amount\_Of\_Remained FROM Product WHERE Product\_ID = @var\_product\_id;

IF (@var\_amount\_of\_remained - @var\_amount) >= 0

UPDATE Order\_For\_Product SET Is\_Done = 1 WHERE Order\_ID = @var\_order\_id;

UPDATE Product SET Amount\_Of\_Remained = @var\_amount\_of\_remained - @var\_amount WHERE Product\_ID = @var\_product\_id;

END;

-- Пример вызова процедуры:

-- EXECUTE TryToCompleteOrder @var\_order\_id = 2;

-- DROP PROCEDURE TryToCompleteOrder;

-- Получить количество заказов мороженого

GO

CREATE FUNCTION GetNumberOfIceCreamOrders()

RETURNS INTEGER

BEGIN

DECLARE @ResultInt INTEGER;

SELECT @ResultInt = COUNT(\*) FROM Order\_For\_Product JOIN Product ON Order\_For\_Product.Product\_ID = Product.Product\_ID JOIN Product\_Type ON Product.Type\_ID = Product\_Type.Type\_ID WHERE Product\_Type.Name = 'Ice Cream';

RETURN @ResultInt;

END;

-- Пример вызова функции:

-- SELECT dbo.GetNumberOfIceCreamOrders();

-- DROP FUNCTION GetNumberOfIceCreamOrders;

### Триггеры

-- Запрет на удаление поставщика

GO

CREATE TRIGGER TR\_Del\_Provider ON Provider FOR DELETE

AS ROLLBACK;

-- Проверка: DELETE FROM Provider;

-- Запрет на удаление клиента

GO

CREATE TRIGGER TR\_Del\_Client ON Client FOR DELETE

AS ROLLBACK;

-- Проверка: DELETE FROM Client;

-- После 30-го заказа продукции добавить в графу с дополнительной информацией о ней пометку о том,

-- что данная продукция стала популярной

GO

CREATE TRIGGER TR\_Ins\_Order\_For\_Product ON Order\_For\_Product FOR INSERT

AS

BEGIN

IF (SELECT COUNT(\*) FROM Order\_For\_Product WHERE Order\_For\_Product.Product\_ID = (SELECT Product\_ID FROM INSERTED)) > 30

UPDATE Product SET Additional\_Information = 'Popular product: ' + Additional\_Information WHERE Product.Product\_ID = (SELECT Product\_ID FROM INSERTED);

END;

### Представления

-- Получить количество вариаций продуктов каждого типа

GO

CREATE VIEW V\_NumberOfProductVariations (Product\_Type, Count) AS

SELECT Product\_Type.Name, COUNT(\*) FROM Product, Product\_Type WHERE Product.Type\_ID = Product\_Type.Type\_ID

GROUP BY Product\_Type.Name;

-- DROP VIEW V\_NumberOfProductVariations;

-- Поставщики и поставляемое ими сырьё

GO

CREATE VIEW V\_ProvidersAndRaw (Provider\_ID, Name, Region, Address, Phone, Raw\_Type, Amount\_In\_Storage, Cost\_Per\_Unit) AS

SELECT Provider.Provider\_ID, Provider.Name, Provider.Region, Provider.Address, Provider.Phone, Raw.Type, Raw.Amount\_In\_Storage, Raw.Cost\_Per\_Unit

FROM Provider LEFT JOIN Raw ON Provider.Raw\_ID = Raw.Raw\_ID;

-- DROP VIEW V\_ProvidersAndRaw;

-- Список клиентов по количеству заказанных ими за всё время единиц продукции,

-- где учитываются только клиенты, заказавшие больше 50 единиц продукции

GO

CREATE VIEW V\_ClientsAndNumberOfAnyProductionUnits (Client\_Name, Production\_Unit\_Count) AS

SELECT Client.Name, SUM(Amount) AS Pr\_Un\_C FROM Order\_For\_Product RIGHT JOIN Client ON

Order\_For\_Product.Client\_ID = Client.Client\_ID GROUP BY Client.Name HAVING SUM(Amount) > 50;

-- DROP VIEW V\_ClientsAndNumberOfAnyProductionUnit;

## Клиентская часть

(запросы для экранных форм и отчетов)

-- 1. Поставщики из Зеленодольска

-- SELECT Name, Address FROM Provider WHERE Region = 'Zelenodolsky district';

-- 2. Поставщики, у которых цена единицы сырья меньше 50

-- SELECT Name, Raw\_Type FROM V\_ProvidersAndRaw WHERE Cost\_Per\_Unit < 50;

-- 3. Информация о поставщиках и поставляемом ими сырье из Ульяновского района и Зеленодольского района

/\*

SELECT \* FROM V\_ProvidersAndRaw WHERE Region = 'Ulyanovsk district'

UNION

SELECT \* FROM V\_ProvidersAndRaw WHERE Region = 'Zelenodolsky district';

\*/

-- 4. Названия всех продуктов и их цены у клиентов, в названии которых есть буква 'n'

/\*

SELECT Product.Name, Product.Selling\_Price FROM Client, Product WHERE Client.Product\_ID = Product.Product\_ID

AND Client.Name LIKE '%n%'

\*/

-- 5. Список названий клиентов, которые заказывают продукцию в бумажном пакете, а также тип данной продукции и её название

/\*

SELECT Client.Name, Product\_Type.Name, Product.Name FROM Order\_For\_Product JOIN Client

ON Order\_For\_Product.Client\_ID = Client.Client\_ID JOIN Product ON Order\_For\_Product.Product\_ID = Product.Product\_ID

JOIN Product\_Type ON Product.Type\_ID = Product\_Type.Type\_ID WHERE Packing\_Type = 'Paper bag';

\*/

-- 6. Сумая маленькая сумма единиц заказанной продукции среди всех клиентов

/\*

SELECT MIN(Product\_Units\_Count\_Grouped\_By\_Clients.Units\_Count) FROM

(SELECT Client.Name, SUM(Amount) AS Units\_Count FROM Order\_For\_Product JOIN Client ON

Order\_For\_Product.Client\_ID = Client.Client\_ID GROUP BY Client.Name) Product\_Units\_Count\_Grouped\_By\_Clients;

\*/

-- 7. Количество вариаций продуктов каждого типа

-- SELECT \* FROM V\_NumberOfProductVariations;

-- 8. Средняя цена продукции в картонной упаковке

-- SELECT AVG(CONVERT(DECIMAL, Selling\_Price)) FROM Product WHERE Packing\_Type = 'Carton box';

-- 9. Продукция с ценой выше 70, имеющая дополнительную информацию о себе

/\*

SELECT Price\_More\_70.Name FROM (SELECT \* FROM Product WHERE Selling\_Price > 70) Price\_More\_70

WHERE Price\_More\_70.Additional\_Information IS NOT NULL AND Price\_More\_70.Additional\_Information != '';

\*/

-- 10. Номера выполненных заказов и количество заказанной продукции в заказах, где это количество больше 90

-- (отсортировано по возрастанию количества заказанной продукции)

-- SELECT Order\_ID, Amount FROM Order\_For\_Product WHERE Is\_Done = 1 AND Amount > 90 ORDER BY Amount;

-- 11. Заказы с непустой информацией, созданные клиентами, располагающимися в квартирах с номером 10

/\*

SELECT Order\_For\_Product.Order\_ID FROM Order\_For\_Product, Client WHERE Order\_For\_Product.Client\_ID = Client.Client\_ID

AND Order\_For\_Product.Data IS NOT NULL AND Order\_For\_Product.Data != '' AND Client.Address LIKE '%-10';

\*/

-- 12. Названия всех продуктов, для производства которых используется молоко

/\*

SELECT Product.Name FROM Product JOIN Raw\_Product ON Product.Product\_ID = Raw\_Product.Product\_ID

JOIN Raw ON Raw\_Product.Raw\_ID = Raw.Raw\_ID WHERE Raw.Type = 'Milk';

\*/

# ПРИЛОЖЕНИЕ: Создание и заполнение базы данных

CREATE DATABASE Dairy\_Plant;

GO

USE Dairy\_Plant;

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

-- Создание таблиц и PK

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

-- Таблица для поставщика какого-то одного вида сырья

CREATE TABLE Provider(

Provider\_ID INTEGER NOT NULL,

Name VARCHAR(30) NOT NULL,

Region VARCHAR(30),

Address VARCHAR(50) NOT NULL,

Phone VARCHAR(15) NOT NULL,

Raw\_ID INTEGER NOT NULL,

CONSTRAINT Provider\_PK PRIMARY KEY (Provider\_ID)

)

;

-- Таблица для клиента (магазин), который продаёт один вид продукции

CREATE TABLE Client(

Client\_ID INTEGER NOT NULL,

Name VARCHAR(30) NOT NULL,

Address VARCHAR(50) NOT NULL,

Phone VARCHAR(15) NOT NULL,

Product\_ID INTEGER NOT NULL,

CONSTRAINT Client\_PK PRIMARY KEY (Client\_ID)

)

;

-- Таблица для сырья одного конкретного типа

CREATE TABLE Raw(

Raw\_ID INTEGER NOT NULL,

Type VARCHAR(20) NOT NULL,

Amount\_In\_Storage INTEGER DEFAULT 0 NOT NULL,

Cost\_Per\_Unit INTEGER NOT NULL,

CONSTRAINT Raw\_PK PRIMARY KEY (Raw\_ID),

CONSTRAINT Amount\_In\_Storage CHECK (Amount\_In\_Storage >= 0),

CONSTRAINT Cost\_Per\_Unit CHECK (Cost\_Per\_Unit >= 0)

)

;

-- Таблица для возможных типов производимой заводом продукции

CREATE TABLE Product\_Type(

Type\_ID INTEGER NOT NULL,

Name VARCHAR(30) NOT NULL,

CONSTRAINT Type\_PK PRIMARY KEY (Type\_ID)

)

;

-- Таблица для конкретного продукта (разные вариации продуктов одного типа, например)

CREATE TABLE Product(

Product\_ID INTEGER NOT NULL,

Type\_ID INTEGER NOT NULL,

Name VARCHAR(30) NOT NULL,

Packing\_Type VARCHAR(20),

Packing\_Color VARCHAR(20),

Selling\_Price INTEGER NOT NULL,

Amount\_Of\_Remained INTEGER DEFAULT 0 NOT NULL,

Additional\_Information VARCHAR(50),

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

CONSTRAINT Selling\_Price CHECK (Selling\_Price >= 0),

CONSTRAINT Amount\_Of\_Remained CHECK (Amount\_Of\_Remained >= 0)

)

;

-- Таблица для клиентских заказов на определёную произведённую продукцию

CREATE TABLE Order\_For\_Product(

Order\_ID INTEGER NOT NULL,

Product\_ID INTEGER NOT NULL,

Amount INTEGER NOT NULL,

Client\_ID INTEGER NOT NULL,

Data VARCHAR(100),

Is\_Done INTEGER DEFAULT 0 NOT NULL,

CONSTRAINT Order\_PK PRIMARY KEY (Order\_ID),

CONSTRAINT Amount CHECK (Amount >= 0),

CONSTRAINT Is\_Done CHECK (IS\_Done IN (0, 1))

)

;

CREATE TABLE Raw\_Product(

Raw\_ID INTEGER NOT NULL,

Product\_ID INTEGER NOT NULL,

Raw\_Amount INTEGER NOT NULL,

CONSTRAINT Raw\_Amount CHECK (Raw\_Amount >= 0),

)

;

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

-- Создание FK

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

ALTER TABLE Provider ADD CONSTRAINT FK\_Provider\_Raw

FOREIGN KEY (Raw\_ID)

REFERENCES Raw(Raw\_ID)

;

ALTER TABLE Client ADD CONSTRAINT FK\_Client\_Product

FOREIGN KEY (Product\_ID)

REFERENCES Product(Product\_ID)

;

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

FOREIGN KEY (Type\_ID)

REFERENCES Product\_Type(Type\_ID)

;

ALTER TABLE Order\_For\_Product ADD CONSTRAINT Order\_For\_Product\_Product

FOREIGN KEY (Product\_ID)

REFERENCES Product(Product\_ID)

;

ALTER TABLE Order\_For\_Product ADD CONSTRAINT Order\_For\_Product\_Client

FOREIGN KEY (Client\_ID)

REFERENCES Client(Client\_ID)

;

ALTER TABLE Raw\_Product ADD CONSTRAINT Raw\_Product\_Raw

FOREIGN KEY (Raw\_ID)

REFERENCES Raw(Raw\_ID)

;

ALTER TABLE Raw\_Product ADD CONSTRAINT Raw\_Product\_Product

FOREIGN KEY (Product\_ID)

REFERENCES Product(Product\_ID)

;

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

-- Создание индексов для некоторых важных полей таблиц

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

CREATE INDEX Index\_Provider\_Name ON Provider(Name);

CREATE INDEX Index\_Provider\_Region ON Provider(Region);

CREATE INDEX Index\_Client\_Name ON Client(Name);

CREATE INDEX Index\_Raw\_Type ON Raw(Type);

CREATE INDEX Index\_Product\_Type\_Name ON Product\_Type(Name);

CREATE INDEX Index\_Product\_Name On Product(Name);

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

-- Заполнение таблиц тестовыми данными

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

INSERT INTO Product\_Type(Type\_ID, Name) VALUES (1, 'Kefir');

INSERT INTO Product\_Type(Type\_ID, Name) VALUES (2, 'Yoghurt');

INSERT INTO Product\_Type(Type\_ID, Name) VALUES (3, 'Curd');

INSERT INTO Product\_Type(Type\_ID, Name) VALUES (4, 'Cheese');

INSERT INTO Product\_Type(Type\_ID, Name) VALUES (5, 'Butter');

INSERT INTO Product\_Type(Type\_ID, Name) VALUES (6, 'Ice Cream');

INSERT INTO Product(Product\_ID, Type\_ID, Name, Packing\_Type, Packing\_Color, Selling\_Price, Amount\_Of\_Remained, Additional\_Information) VALUES (1, 1, 'Kefir "Svezheye segondnya"', 'Carton box', 'Blue', 75, 19, 'Kefir with extended shelf life');

INSERT INTO Product(Product\_ID, Type\_ID, Name, Packing\_Type, Packing\_Color, Selling\_Price, Amount\_Of\_Remained, Additional\_Information) VALUES (2, 2, 'Yoghurt "Mepica"', 'Glass bottle', 'Transparent', 48, 100, 'Yoghurt with fruit pieces');

INSERT INTO Product(Product\_ID, Type\_ID, Name, Packing\_Type, Packing\_Color, Selling\_Price, Amount\_Of\_Remained, Additional\_Information) VALUES (3, 3, 'Curd "Krasnoye selo"', 'Paper bag', 'Green', 110, 120, 'Skim curd');

INSERT INTO Product(Product\_ID, Type\_ID, Name, Packing\_Type, Packing\_Color, Selling\_Price, Amount\_Of\_Remained) VALUES (4, 4, 'Cheese "Ranabeyevo"', 'Paper bag', 'Brown', 150, 200);

INSERT INTO Product(Product\_ID, Type\_ID, Name, Packing\_Type, Packing\_Color, Selling\_Price, Amount\_Of\_Remained) VALUES (5, 5, 'Butter "Butter Station"', 'Paper bag', 'Yellow', 80, 250);

INSERT INTO Product(Product\_ID, Type\_ID, Name, Packing\_Type, Packing\_Color, Selling\_Price, Amount\_Of\_Remained) VALUES (6, 6, 'Ice cream "Zelenodolsk kholod"', 'Carton box', 'White', 105, 100);

INSERT INTO Raw(Raw\_ID, Type, Amount\_In\_Storage, Cost\_Per\_Unit) VALUES (1, 'Milk', 100, 90);

INSERT INTO Raw(Raw\_ID, Type, Amount\_In\_Storage, Cost\_Per\_Unit) VALUES (2, 'Carton boxes', 2000, 10);

INSERT INTO Raw(Raw\_ID, Type, Amount\_In\_Storage, Cost\_Per\_Unit) VALUES (3, 'Glass Bottles', 590, 30);

INSERT INTO Raw(Raw\_ID, Type, Amount\_In\_Storage, Cost\_Per\_Unit) VALUES (4, 'Paper Bag', 5000, 10);

INSERT INTO Raw(Raw\_ID, Type, Amount\_In\_Storage, Cost\_Per\_Unit) VALUES (5, 'Cream', 250, 150);

INSERT INTO Provider(Provider\_ID, Name, Region, Address, Phone, Raw\_ID) VALUES (1, 'Milk and Co.', 'Zelenodolsky district', 'Ul. Gvardii, 1-10', '111-110', 1);

INSERT INTO Provider(Provider\_ID, Name, Region, Address, Phone, Raw\_ID) VALUES (2, 'Best Boxes', 'Zelenodolsky district', 'Ul. Pridorozhnaya, 14-56', '+79378998910', 2);

INSERT INTO Provider(Provider\_ID, Name, Region, Address, Phone, Raw\_ID) VALUES (3, 'Shiny Glass Company', 'Ulyanovsk district', 'Ul. Podgornaya, 1-3', '256-256', 3);

INSERT INTO Provider(Provider\_ID, Name, Region, Address, Phone, Raw\_ID) VALUES (4, 'New Village', 'Ulyanovsk district', 'Ul. Vosstaniya, 100-89', '+79836457814', 4);

INSERT INTO Provider(Provider\_ID, Name, Region, Address, Phone, Raw\_ID) VALUES (5, 'Best Country', 'Smolensk region', 'Ul. Nagornaya, 10-9', '+77634567489', 4);

INSERT INTO Raw\_Product(Raw\_ID, Product\_ID, Raw\_Amount) VALUES (1, 1, 1);

INSERT INTO Raw\_Product(Raw\_ID, Product\_ID, Raw\_Amount) VALUES (2, 1, 1);

INSERT INTO Raw\_Product(Raw\_ID, Product\_ID, Raw\_Amount) VALUES (5, 1, 1);

INSERT INTO Raw\_Product(Raw\_ID, Product\_ID, Raw\_Amount) VALUES (1, 3, 1);

INSERT INTO Raw\_Product(Raw\_ID, Product\_ID, Raw\_Amount) VALUES (4, 3, 1);

INSERT INTO Raw\_Product(Raw\_ID, Product\_ID, Raw\_Amount) VALUES (5, 3, 1);

INSERT INTO Client(Client\_ID, Name, Address, Phone, Product\_ID) VALUES (1, 'Svezheye segondnya', '14 prospect VO, 1-10', '+79878735467', 1);

INSERT INTO Client(Client\_ID, Name, Address, Phone, Product\_ID) VALUES (2, 'Mepica', 'Ul. Nikolaevskaya 8-10', '+79879035467', 2);

INSERT INTO Client(Client\_ID, Name, Address, Phone, Product\_ID) VALUES (3, 'Krasnoye selo', 'Ul. Zarechnaya 9-11', '+79768374678', 3);

INSERT INTO Client(Client\_ID, Name, Address, Phone, Product\_ID) VALUES (4, 'Ranabeyevo', 'Ul. Zarechnaya 101-1', '+79898374678', 4);

INSERT INTO Client(Client\_ID, Name, Address, Phone, Product\_ID) VALUES (5, 'Butter Station', 'Ul. Mostovaya 91-3', '+79454563456', 5);

INSERT INTO Client(Client\_ID, Name, Address, Phone, Product\_ID) VALUES (6, 'Zelenodolsk kholod', 'Ul. Moldovskaya 3-46', '+79274563456', 6);

INSERT INTO Order\_For\_Product(Order\_ID, Product\_ID, Amount, Client\_ID, Data, Is\_Done) VALUES (1, 1, 2500, 1, 'Very big order', 1);

INSERT INTO Order\_For\_Product(Order\_ID, Product\_ID, Amount, Client\_ID, Is\_Done) VALUES (2, 6, 100, 6, 0);

INSERT INTO Order\_For\_Product(Order\_ID, Product\_ID, Amount, Client\_ID, Data, Is\_Done) VALUES (3, 3, 10, 3, 'Small order for newly opened store', 1);

INSERT INTO Order\_For\_Product(Order\_ID, Product\_ID, Amount, Client\_ID, Is\_Done) VALUES (4, 2, 250, 2, 0);

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

-- Удаление индексов

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

/\*

DROP INDEX Index\_Provider\_Name;

DROP INDEX Index\_Provider\_Region;

DROP INDEX Index\_Client\_Name;

DROP INDEX Index\_Raw\_Type;

DROP INDEX Index\_Product\_Type\_Name;

DROP INDEX Index\_Product\_Name;

\*/

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

-- Удаление таблиц

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

/\*

DROP TABLE Raw\_Product;

DROP TABLE Provider;

DROP TABLE Order\_For\_Product;

DROP TABLE Raw;

DROP TABLE Client;

DROP TABLE Product;

DROP TABLE Product\_Type;

\*/