**Endterm for DBMS2**

**DBMS for online-shop**

**Created by:   Zholdasova Madina - 210103149**

**Kusman Alua – 210103375**

**Seisenbay Sabira – 210103272**

**Bekturganova Indira – 210103105**

**Introduction part and Create an ERD part.**

We have created a database of an online clothing shop. Our database shows the all actions of the client to get the stuff they like, i.e. to a greater extent the database is aimed at describing the relationship of the client with the order.

1) For the correct compilation of the database and its ERD, we tried to manually write and analyze the client's business process when buying stuff in real life.

The main thing that is in any store is its stuff, in our case clothes. Any clothing has a type (hoodie, pants, etc.); size; gender, to which it is intended; price and number of copies (because in stores clothes should not be in a single copy).

The goods get into the store when the sellers place them. Sellers have a name and a phone number where clients can contact them.

Next, we have the client and his actions on the platform to receive the stuff:

First, the client enters the online store platform, entering his data there (name, address, phone number) and there he is immediately assigned a unique number (primary key - id).

Second, the customer registers an order, one or more. And each order, in addition to a unique number (primary key – id), has a date (when this order was created) and consists of one or more units of products. Note: one type of product can be ordered several times (for example, two pairs of pants, not just one).

Third, in order to reach the customer, the order must go through a certain path consisting of several steps. First, it must be paid, then packed, then the courier receives it and takes it to the client and the last client sends the order.

2) Now, based on these records, we will highlight the main things that will go as names for tables.

Clothes, Clients, Orders, Seller, Step

And we will write attributes to each table.

Clothing: size, gender, price, quantity;

Client: name, address, phone number;

Orders: registration date;

Seller: name, phone number;

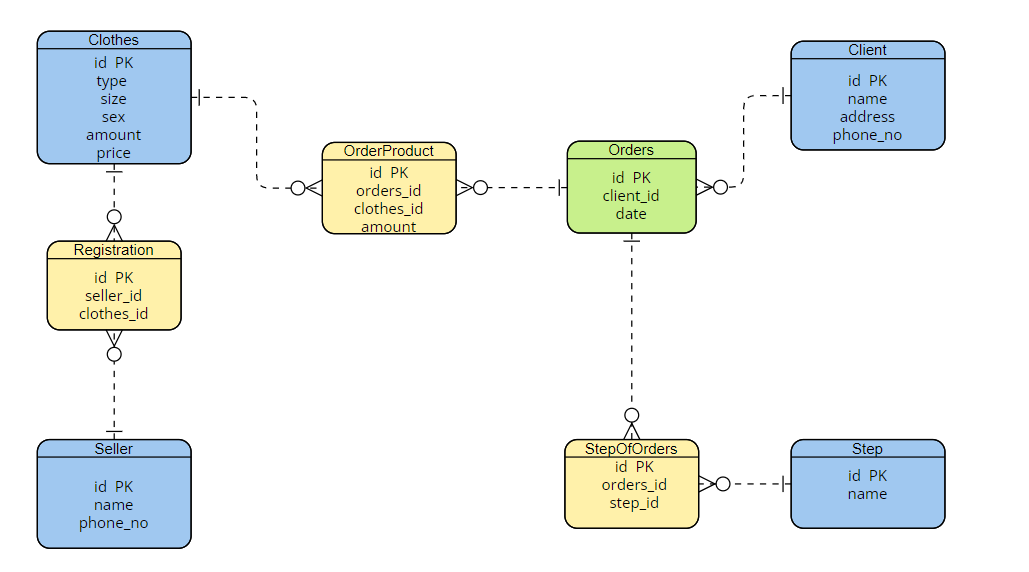
Step: name;

At this stage, omit the id, but by default they are in every table.

3) Next, let's try to determine the relationship between our tables. For example, a Clients-Order relationship: a customer can make many orders, one order is made for only one customer, i.e. the relationship will be One-to-Many. The relationship between the Orders-Step: one order has several steps and one step refers to many orders, i.e. the relationship will be Many-to-Many. Clothes-Orders Relationship: One garment can be included in many orders, and one order can include many clothes, i.e. the relationship will be Many-to-Many. The relationship between the Seller-Clothes relationship: One type of clothes can be registered by many sellers and one seller can register many clothes, i.e. the relationship will be Many-to-Many.

We have presented Many-to-Many relationships in the form of new tables between existing ones.

And thus we have compiled our ERD:



**Normalization part. Why our database follows normal form?**

We have 8 tables in our database. First table Clothes that has following attributes. This table in 1 normal form because each table cell contains only single value, and each column have a unique name. As you see there is no duplicates.

So we can say that table is also in second normal form, because our table follows first normal form and each non-key attribute dependent only on the primary key in our case it is id in Clothes table, no partial dependency. For example,we can find non-key attributes like type,size,sex,amount,price with just knowing id of the clothes.

Our table following third normal form too, because clothes table follows second normal form and all non-key attributes independent of each other, no transitive dependency. For example, we cannot find value of the non-key attribute  knowing the parameter of other non key attribute for example we cannot find amount knowing size of the clothes.

Second table Client that has following attributes. This table in 1 normal form because each table cell contains only single value, and each column have a unique name. As you see there is no duplicates

So we can say that table is also in second normal form, because our table follows first normal form and each non-key attribute dependent only on the primary key in our case it is id in Client table, no partial dependency. For example,we can find non-key attributes like name,address,phone\_no with just knowing id of the client.

Our table following third normal form too, because client table follows second normal form and all non-key attributes independent of each other, no transitive dependency. We cannot find value of the non-key attribute knowing the parameter of other non key attribute. for example, we cannot find address knowing phone\_no of the client.

Third table Seller that has following attributes. This table in first normal form because each table cell contains only single value, and each column have a unique name. As you see there is no duplicates

So we can say that Seller table is also in second normal form, because our table follows first normal form and each non-key attribute dependent only on the primary key in our case it is id in Seller table, no partial dependency. For example,we can find non-key attributes like name,phone\_no with just knowing id of the Seller.

Our table following third normal form too, because client table follows second normal form and all non-key attributes independent of each other, no transitive dependency. We cannot find value of the non-key attribute knowing the parameter of other non key attribute. for example, we cannot find name knowing phone\_no of the seller ,because there can be a people with the same name.

Fourth table Step that has following attributes. This table in 1 normal form because each table cell contains only single value, and each column have a unique name. As you see there is no duplicates

So we can say that Step table is also in second normal form, because our table follows first normal form and each non-key attribute dependent only on the primary key in our case it is id in Step table, no partial dependency. For example,we can find non-key attribute name with just knowing id of the step.

Our table following third normal form too, because Step table follows second normal form and there id no transitive dependency, because we have only one non-key attribute.

Fifth table Registration that has following attributes. This table in 1 normal form because each table cell contains only single value, and each column have a unique name. As you see there is no duplicates

So we can say that Registration table is also in second normal form, because our table follows first normal form and each non-key attribute dependent only on the primary key in our case it is id in Registration table, no partial dependency. For example, we can find non-key attributes like seller\_id,clothes\_id with just knowing Registration id.

Our table following third normal form too, because client table follows second normal form and all non-key attributes independent of each other, no transitive dependency. We cannot find value of the non-key attribute knowing the parameter of other non key attribute. for example, we cannot find seller\_id knowing clothes\_id.

Sixth table OrderProduct that has following attributes. This table in 1 normal form because each table cell contains only single value, and each column have a unique name. As you see there is no duplicates

So we can say that OrderProduct table is also in second normal form, because our table follows first normal form and each non-key attribute dependent only on the primary key in our case it is id in OrderProduct table, no partial dependency. For example, we can find non-key attributes like orders\_id,clothes\_id,amount with just knowing id of the order

Our table following third normal form too, because OrderProduct table follows second normal form and all non-key attributes independent of each other, no transitive dependency. We cannot find value of the non-key attribute knowing the parameter of other non key attribute. for example, we cannot find orders\_id knowing clothes\_id of the Order.

Seventh table StepOfOrders that has following attributes. This table in 1 normal form because each table cell contains only single value, and each column have a unique name. As you see there is no duplicates

So we can say that StepOfOrders table is also in second normal form, because our table follows first normal form and each non-key attribute dependent only on the primary key in our case it is id in Client table, no partial dependency. For example,we can find non-key attributes like orders\_id,step\_id with just knowing StepOfOrders id.

Our table following third normal form too, because StepOfOrders table follows second normal form and all non-key attributes independent of each other, no transitive dependency. We cannot find value of the non-key attribute knowing the parameter of other non key attribute. For example, we cannot find orders\_id knowing step\_id.

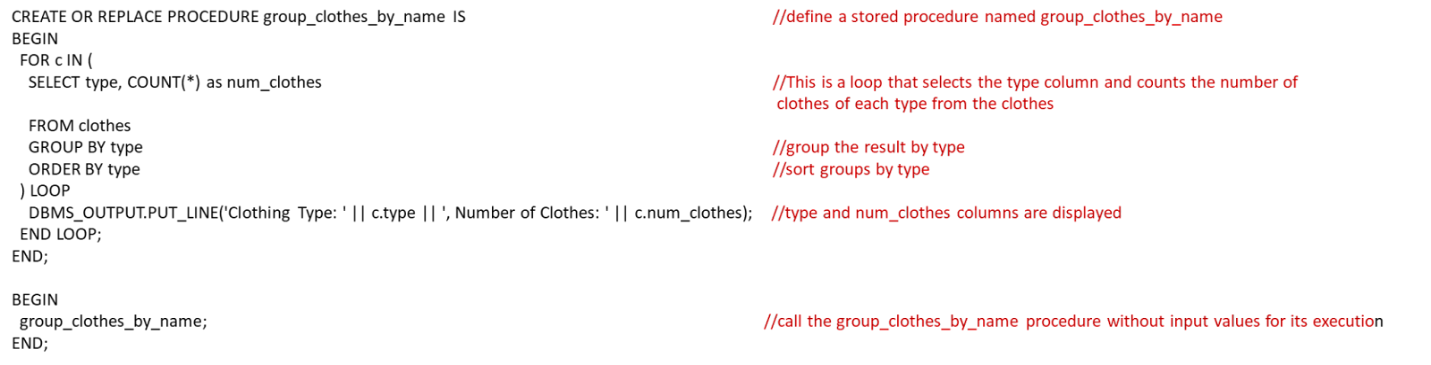
Eighth table Orders that has following attributes. This table in 1 normal form because each table cell contains only single value, and each column have a unique name. As you see there is no duplicates

So we can say that Orders table is also in second normal form, because our table follows first normal form and each non-key attribute dependent only on the primary key in our case it is id in Order table, no partial dependency. For example, we can find non-key attributes like client\_id,date with just knowing id of the Orders.

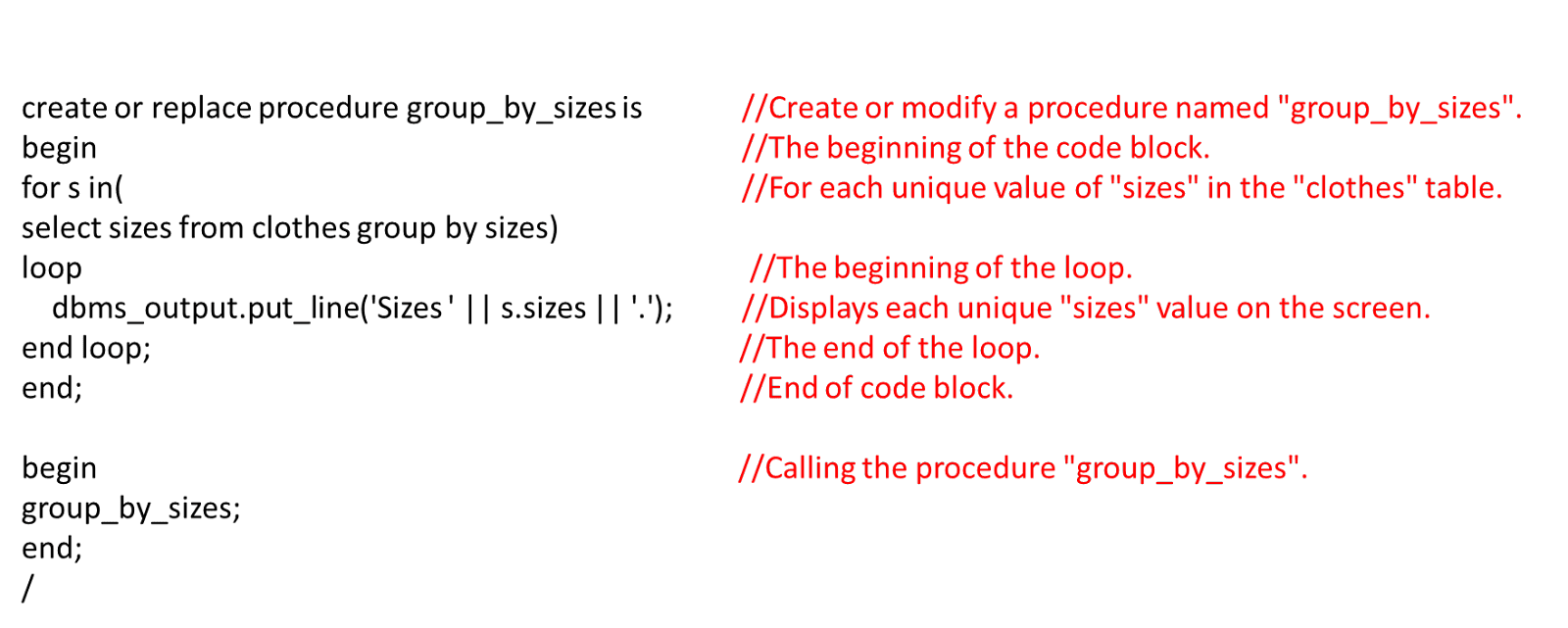
Our table following third normal form too, because Orders table follows second normal form and all non-key attributes independent of each other, no transitive dependency. We cannot find value of the non-key attribute knowing the parameter of other non key attribute. for example, we cannot find client\_id knowing date of the Order.

**PL/SQL part.**

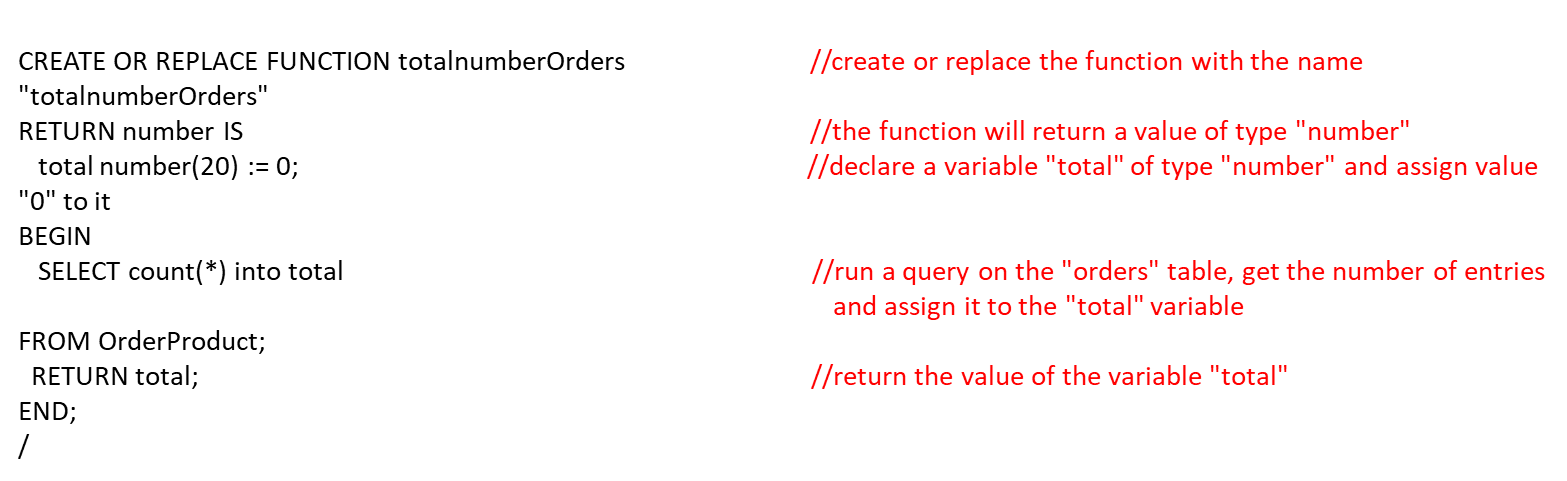
**Procedure 1:**

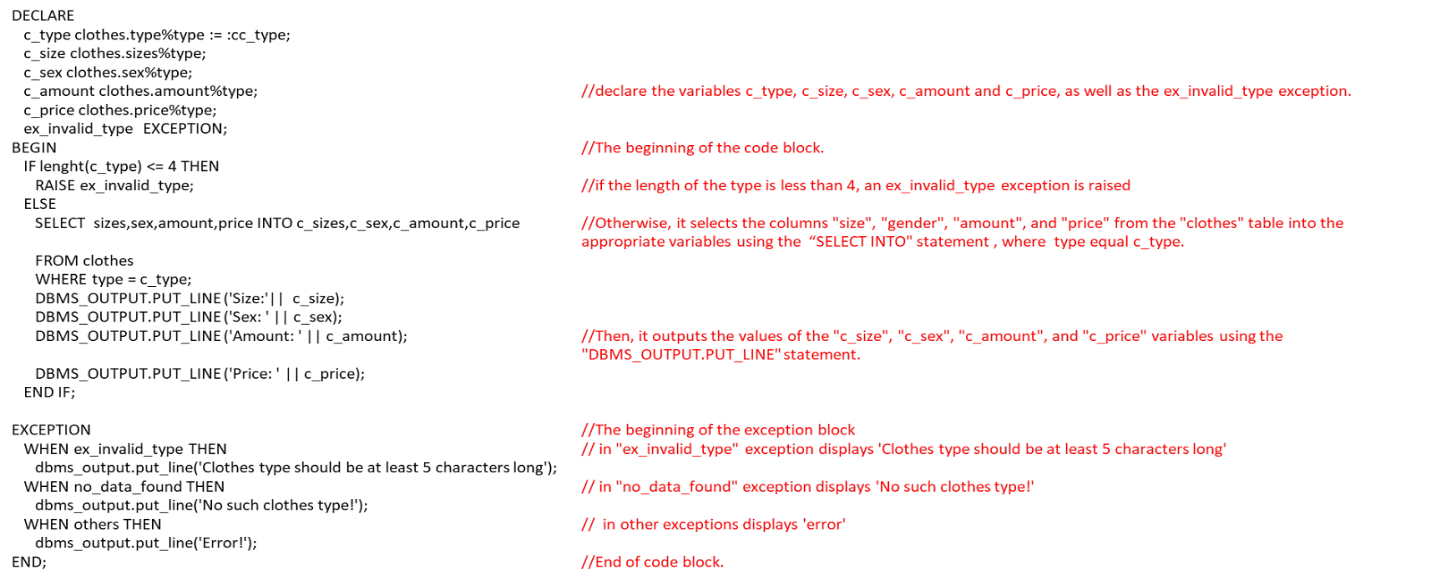


**Procedure 2:**

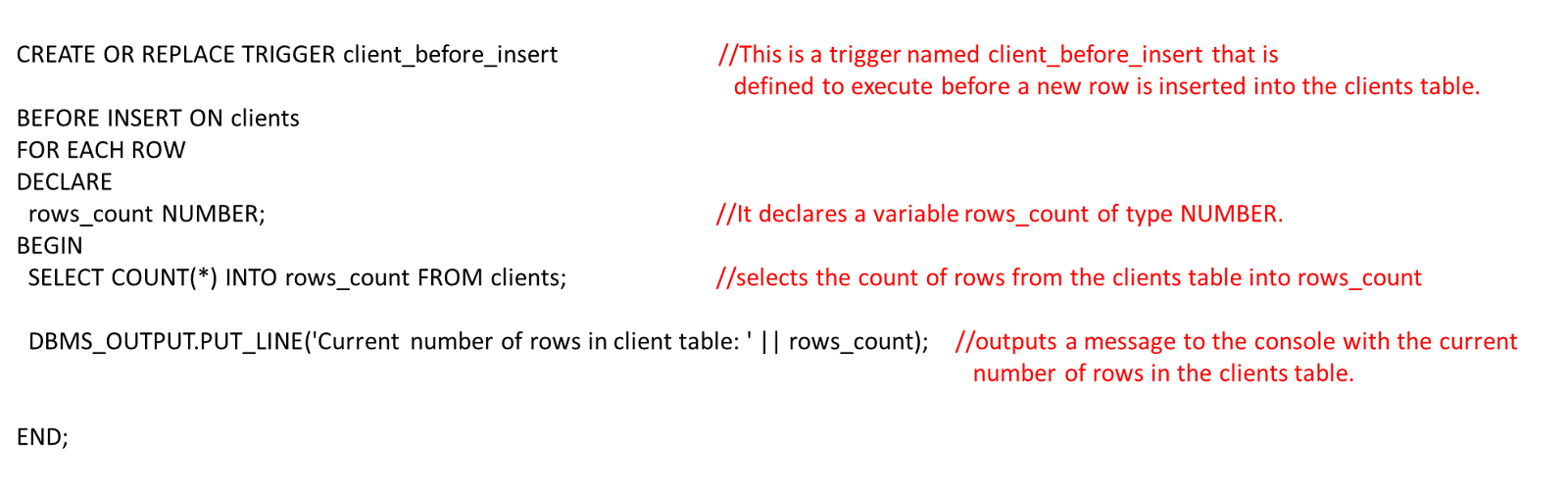


**Function:**



**Exception:**

**Trigger:**



**Tables:**

