Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Проектирование и разработка баз данных интернет-приложений**

Студент: Козека Е. М.

ФИТ 3 курс 4 группа

Преподаватель: Нистюк О. А.

Минск 2025

**Лабораторная работа №2. Реализация базы данных (вариант 14)**

**Задание 1.** Создать базу данных для СУБД SQL Server по разработанному в Лабораторной работе 1 варианту. Создать необходимые объекты (представления, индексы, последовательности и пр.).

**Таблицы**:

table **PRODUCT\_CATEGORIES** -- *информация о категориях товара*

(

**category\_id** int primary key, -- *ID категории товара*

**category\_name** varchar(200) not null -- *название категории товара*

);

table **PRODUCTS** -- *информация о товарах*

(

**product\_id** int primary key, -- *ID товара*

**product\_name** varchar(200) not null, -- *наименование товара*

**category\_id** int, -- *ID категории товара*

**price** int not null, -- *цена товара*

**description** varchar(255), -- *описание товара*

**image\_url** varchar(200), -- *изображение товара*

**quantity** int default 0, -- *количество товара*

-- *внешний ключ для связи с таблицей PRODUCT\_CATEGORIES по столбцу category\_id*

foreign key (category\_id) references PRODUCT\_CATEGORIES (category\_id)

);

table **CUSTOMERS** -- *информация о покупателях*

(

**customer\_id** int primary key, -- *ID покупателя*

**name** varchar(100) unique, -- *ФИО покупателя*

**email** varchar(100), -- *электронная почта покупателя*

**phone\_number** varchar(20), -- *номер телефона покупателя*

**address** varchar(200) -- *адрес покупателя*

);

table **ORDER\_STATUSES** -- *информация о статусах заказов*

(

**status\_id** int primary key, -- *ID статуса заказа*

**status\_name** varchar(50) not null -- *наименование статуса заказа*

);

table **ORDERS** -- *информация о заказах*

(

**order\_id** int primary key, -- *ID заказа*

**customer\_id** int, -- *ID покупателя*

**order\_date** date, -- *дата оформления*

**total\_amount** int, -- *сумма заказа*

**status\_id** int, -- *ID статуса заказа*

-- *внешний ключ для связи с таблицей CUSTOMERS по столбцу customer\_id*

foreign key (customer\_id) references CUSTOMERS (customer\_id),

-- *внешний ключ для связи с таблицей ORDER\_STATUSES по столбцу status\_id*

foreign key (status\_id) references ORDER\_STATUSES (status\_id)

);

table **COMPOSITION\_OF\_ORDERS**  -- *информация о* *составе заказа*

(

**composition\_id** int primary key, -- *ID**состава заказа*

**order\_id** int, -- *ID заказа*

**product\_id** int, -- *ID товара*

**quantity** int, -- *количество*

-- *внешний ключ для связи с таблицей ORDERS по столбцу order\_id*

foreign key (order\_id) references ORDERS (order\_id),

-- *внешний ключ для связи с таблицей PRODUCTS по столбцу product\_id*

foreign key (product\_id) references PRODUCTS (product\_id)

);

table **CARTS** -- *информация о корзинах*

(

**cart\_id** int primary key, -- *ID корзины*

**customer\_id** int, -- *ID покупателя*

-- *внешний ключ для связи с таблицей CUSTOMERS по столбцу customer\_id*

foreign key (customer\_id) references CUSTOMERS (customer\_id)

);

table **CART\_ITEMS** -- *информация о товарах в корзине*

(

**cart\_item\_id** int primary key, -- *ID товара в корзине*

**cart\_id** int, -- *ID корзины*

**product\_id** int, -- *ID товара*

**quantity** int, -- *количество*

-- *внешний ключ для связи с таблицей CARTS по столбцу cart\_id*

foreign key (cart\_id) references CARTS (cart\_id),

-- *внешний ключ для связи с таблицей PRODUCTS по столбцу product\_id*

foreign key (product\_id) references PRODUCTS (product\_id)

);

**Индексы**:

Индексы для оптимизации запросов:

* idx\_products\_product\_name;
* idx\_product\_categories\_category\_name;
* idx\_customer\_phone\_number.

**Представления**:

-- *количество проданных товаров по периодам*

view SALES\_BY\_PERIOD\_view with schemabinding as

select convert(varchar(7), o.order\_date, 120) as

month, p.product\_name, sum(co.quantity) as total\_quantity

from dbo.ORDERS o

join dbo.COMPOSITION\_OF\_ORDERS co

on o.order\_id = co.order\_id

join dbo.PRODUCTS p

on co.product\_id = p.product\_id

group by convert(varchar(7), o.order\_date, 120), p.product\_name;

-- *популярные товары*

create view POPULAR\_PRODUCTS\_view with schemabinding as

select p.product\_name, sum(co.quantity) as total\_quantity

from dbo.COMPOSITION\_OF\_ORDERS co

join dbo.PRODUCTS p on co.product\_id = p.product\_id

group by p.product\_name;

-- *общее количество товара*

create view TOTAL\_AMOUNT\_OF\_PRODUCTS\_view with schemabinding as

select product\_name, quantity from dbo.PRODUCTS;

**Процедуры**:

-- *процедура для просмотра количества проданных товаров по периодам*

create procedure SalesByPeriod

as

begin

select \* from SALES\_BY\_PERIOD\_view;

end;

-- *процедура для просмотра популярных товаров*

create procedure DisplayPopularProducts

as

begin

select \* from POPULAR\_PRODUCTS\_view;

end;

-- *процедура для вывода общего количества товара*

create procedure DisplayTotalAmountOfProducts

as

begin

select \* from TOTAL\_AMOUNT\_OF\_PRODUCTS\_view;

end;

-- *процедура для вывода информации о существующих категориях*

create or alter procedure DisplayProductCategories

as

begin

declare @category\_name varchar(100);

declare cur\_product\_categories cursor for

select category\_name from PRODUCT\_CATEGORIES;

open cur\_product\_categories;

fetch next from cur\_product\_categories into @category\_name;

while @@fetch\_status = 0

begin

print 'Category Name: ' + @category\_name;

fetch next from cur\_product\_categories into @category\_name;

end;

close cur\_product\_categories;

deallocate cur\_product\_categories;

end;

-- *процедура для вывода информации обо всех заказах*

create or alter procedure DisplayOrderInformation

as

begin

set nocount on;

begin try

select c.name as

customer\_name,

p.product\_name,

o.order\_date,

o.total\_amount,

os.status\_name as order\_status

from CUSTOMERS c

join ORDERS o on c.customer\_id = o.customer\_id

join COMPOSITION\_OF\_ORDERS co

on o.order\_id = co.order\_id

join PRODUCTS p

on co.product\_id = p.product\_id

join ORDER\_STATUSES os

on o.status\_id = os.status\_id

order by o.order\_date;

end try

begin catch

print 'Ошибка при выполнении процедуры';

throw;

end catch;

end;

-- *процедура для вывода пользователей*

create or alter procedure DisplayCustomerInformation

as

begin

set nocount on;

begin try

select customer\_id, name, email, phone\_number, address from CUSTOMERS;

end try

begin catch

print 'Ошибка при выполнении процедуры'; throw;

end catch;

end;

**Функция**:

-- *функция для вывода общей суммы всех заказов клиента*

create or alter function GetTotalOrderAmount(@CustomerId int)

returns int

as

begin

declare @TotalAmount int;

select @TotalAmount = sum(total\_amount) from ORDERS

where customer\_id = @CustomerId;

return coalesce(@TotalAmount, 0);

end;

**Последовательность**:

-- *последовательность для ID заказов*

create sequence OrderSeq

as int -- *состоит из целых чисел*

start with 1 -- *начинается с 1*

increment by 1; -- *шаг = 1*

**Триггер**:

-- *обновление суммы заказа на основе всех товаров в заказе*

create trigger trg\_UpdateTotalAmount on COMPOSITION\_OF\_ORDERS

after insert, update

as

begin

set nocount on;

update ORDERS set total\_amount = (

select sum(p.price \* c.quantity)

from COMPOSITION\_OF\_ORDERS c

join PRODUCTS p on c.product\_id = p.product\_id

where c.order\_id = ORDERS.order\_id

)

where order\_id in (select order\_id from inserted);

end;

**Задание 2.** Создать базу данных для СУБД Oracle по разработанному в Лабораторной работе 1 варианту. Создать необходимые объекты (представления, индексы, последовательности и пр.).

**Таблицы**:

table **PRODUCT\_CATEGORIES** -- *информация о категориях товара*

(

**category\_id** number primary key, -- *ID категории товара*

**category\_name** varchar(200) not null -- *название категории товара*

);

table **PRODUCTS** -- *информация о товарах*

(

**product\_id** number primary key, -- *ID товара*

**product\_name** varchar(200) not null, -- *наименование товара*

**category\_id** number, -- *ID категории товара*

**price** number not null, -- *цена товара*

**description** varchar(255), -- *описание товара*

**image\_url** varchar(200), -- *изображение товара*

**quantity** number default 0, -- *количество товара*

-- *внешний ключ для связи с таблицей PRODUCT\_CATEGORIES по столбцу category\_id*

foreign key (category\_id) references PRODUCT\_CATEGORIES (category\_id)

);

table **CUSTOMERS** -- *информация о покупателях*

(

**customer\_id** number primary key, -- *ID покупателя*

**name** varchar2(100) unique, -- *ФИО покупателя*

**email** varchar2(100), -- *электронная почта покупателя*

**phone\_number** varchar2(20), -- *номер телефона покупателя*

**address** varchar2(200) -- *адрес покупателя*

);

table **ORDER\_STATUSES** -- *информация о статусах заказов*

(

**status\_id** number primary key, -- *ID статуса заказа*

**status\_name** varchar2(50) not null -- *наименование статуса заказа*

);

table **ORDERS** -- *информация о заказах*

(

**order\_id** number primary key, -- *ID заказа*

**customer\_id** number, -- *ID покупателя*

**order\_date** date, -- *дата оформления*

**total\_amount** number, -- *сумма заказа*

**status\_id** number, -- *ID статуса заказа*

-- *внешний ключ для связи с таблицей CUSTOMERS по столбцу customer\_id*

foreign key (customer\_id) references CUSTOMERS (customer\_id),

-- *внешний ключ для связи с таблицей ORDER\_STATUSES по столбцу status\_id*

foreign key (status\_id) references ORDER\_STATUSES (status\_id)

);

table **COMPOSITION\_OF\_ORDERS**  -- *информация о* *составе заказа*

(

**composition\_id** number primary key, -- *ID**состава заказа*

**order\_id** number, -- *ID заказа*

**product\_id** number, -- *ID товара*

**quantity** number, -- *количество*

-- *внешний ключ для связи с таблицей ORDERS по столбцу order\_id*

foreign key (order\_id) references ORDERS (order\_id),

-- *внешний ключ для связи с таблицей PRODUCTS по столбцу product\_id*

foreign key (product\_id) references PRODUCTS (product\_id)

);

table **CARTS** -- *информация о корзинах*

(

**cart\_id** number primary key, -- *ID корзины*

**customer\_id** number, -- *ID покупателя*

-- *внешний ключ для связи с таблицей CUSTOMERS по столбцу customer\_id*

foreign key (customer\_id) references CUSTOMERS (customer\_id)

);

table **CART\_ITEMS** -- *информация о товарах в корзине*

(

**cart\_item\_id** number primary key, -- *ID товара в корзине*

**cart\_id** number, -- *ID корзины*

**product\_id** number, -- *ID товара*

**quantity** number, -- *количество*

-- *внешний ключ для связи с таблицей CARTS по столбцу cart\_id*

foreign key (cart\_id) references CARTS (cart\_id),

-- *внешний ключ для связи с таблицей PRODUCTS по столбцу product\_id*

foreign key (product\_id) references PRODUCTS (product\_id)

);

**Индексы**:

Индексы для оптимизации запросов:

* idx\_products\_product\_name;
* idx\_product\_categories\_category\_name;
* idx\_customer\_phone\_number.

**Представления**:

-- *количество проданных товаров по периодам*

create materialized view SALES\_BY\_PERIOD\_view

refresh complete on commit

as

select

to\_char(o.order\_date, 'YYYY-MM') as month,

p.product\_name, sum(co.quantity) as total\_quantity

from ORDERS o

join COMPOSITION\_OF\_ORDERS co on o.order\_id = co.order\_id

join PRODUCTS p on co.product\_id = p.product\_id

group by to\_char(o.order\_date, 'YYYY-MM'), p.product\_name

order by month desc;

-- *популярные товары*

create materialized view POPULAR\_PRODUCTS\_view

refresh complete on commit

as

select p.product\_name, sum(co.quantity) as total\_quantity

from COMPOSITION\_OF\_ORDERS co

join PRODUCTS p on co.product\_id = p.product\_id

group by p.product\_name

order by total\_quantity desc;

-- *общее количество товара*

create materialized view TOTAL\_AMOUNT\_OF\_PRODUCTS\_view

refresh complete on commit

as

select product\_name, quantity from PRODUCTS;

**Процедуры**:

-- *процедура для просмотра количества проданных товаров по периодам*

create or replace procedure SalesByPeriod

as

begin

for rec in (select \* from SALES\_BY\_PERIOD\_view)

loop

dbms\_output.put\_line('Month: ' || rec.month ||

', Product Name: ' || rec.product\_name ||

', Total Quantity: ' || rec.total\_quantity);

end loop;

end;

-- *процедура для просмотра популярных товаров*

create or replace procedure DisplayPopularProducts

as

begin

for rec in (select \* from POPULAR\_PRODUCTS\_view)

loop

dbms\_output.put\_line('Product Name: ' || rec.product\_name ||

', Total Quantity: ' || rec.total\_quantity);

end loop;

end;

-- *процедура для вывода общего количества товара*

create or replace procedure DisplayTotalAmountOfProducts

as

begin

for rec in (select \* from TOTAL\_AMOUNT\_OF\_PRODUCTS\_view

where product\_name not like 'Product%')

loop

dbms\_output.put\_line('Product Name: ' || rec.product\_name ||

', Quantity: ' || rec.quantity);

end loop;

end;

-- *процедура для вывода информации о существующих категориях*

create or replace procedure DisplayProductCategories

as

begin

for rec in (select \* from PRODUCT\_CATEGORIES)

loop

dbms\_output.put\_line('Category Name: ' || rec.category\_name);

end loop;

end;

-- *процедура для вывода информации обо всех заказах*

create or replace procedure DisplayOrderInformation

as

begin

for rec in (

select c.name as customer\_name, p.product\_name, o.order\_date, o.total\_amount,

os.status\_name as order\_status

from CUSTOMERS c

join ORDERS o on c.customer\_id = o.customer\_id

join COMPOSITION\_OF\_ORDERS co on o.order\_id = co.order\_id

join PRODUCTS p on co.product\_id = p.product\_id

join ORDER\_STATUSES os on o.status\_id = os.status\_id

)

loop

dbms\_output.put\_line('Customer Name: ' || rec.customer\_name ||

', Product Name: ' || rec.product\_name || ', Order Date: ' ||

to\_char(rec.order\_date, 'DD-MON-YYYY') || ', Total Amount: ' ||

rec.total\_amount || ', Order Status: ' || rec.order\_status);

end loop;

exception

when others then

dbms\_output.put\_line('ERROR');

rollback;

raise;

end;

-- *процедура для вывода пользователей*

create or replace procedure DisplayCustomerInformation

as

begin

for rec in (select \* from customers)

loop

dbms\_output.put\_line('Customer ID: ' || rec.customer\_id || ', Name: ' || rec.name ||

', Email: ' || rec.email || ', Phone Number: ' || rec.phone\_number ||

', Address: ' || rec.address);

end loop;

exception

when others then

dbms\_output.put\_line('ERROR');

rollback;

raise;

end;

**Функция**:

-- функция для вывода общей суммы всех заказов клиента

create or replace function GetTotalOrderAmount

(CustomerId in number)

return number

is

TotalAmount number;

begin

SELECT COALESCE(SUM(total\_amount), 0) INTO TotalAmount

FROM KEM.ORDERS WHERE customer\_id = CustomerId;

RETURN TotalAmount;

end;

**Последовательность**:

-- *последовательность для ID заказов*

create sequence OrderSeq

start with 1 -- *начинается с 1*

increment by 1 -- *шаг = 1*

nocache -- *без кеширования*

nocycle; -- *нецикличная*

**Триггер**:

-- *обновление суммы заказа на основе всех товаров в заказе*

create or replace trigger trg\_UpdateTotalAmount

after insert or update on KEM.COMPOSITION\_OF\_ORDERS

declare

begin

update KEM.ORDERS o

set o.total\_amount = (

select coalesce(sum(p.price \* c.quantity), 0)

from KEM.COMPOSITION\_OF\_ORDERS c

join KEM.PRODUCTS p on c.product\_id = p.product\_id

where c.order\_id = o.order\_id

)

where o.order\_id in

(select distinct order\_id from KEM.COMPOSITION\_OF\_ORDERS);

end;