Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ

УНИВЕРСИТЕТ»

«Проектирование и разработка баз данных интернет-приложений»

Отчёт по лабораторной работе №2

Реализация базы данных.

Выполнил: Сосновец М.И.

ФИТ 3 курс 4 группа

Преподаватель: Нистюк О.А.

Минск 2025

**Описание таблиц СУБД Oracle**

Таблица «Customer» хранит информацию о клиентах. В нее входят следующие столбцы:

Таблица 1.1 – Столбцы таблицы **Customer**

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Описание | Тип | Ограничения |
| ID | Уникальный идентификатор клиента | NUMBER | PRIMARY KEY |
| PRODUCT\_ID | Идентификатор продукта, связанный с клиентом | NUMBER | FOREIGN KEY (ссылка на Product.id) |
| NAME | Имя клиента | VARCHAR2(100) | NOT NULL |
| EMAIL | Email клиента | VARCHAR2(255) | UNIQUE, NOT NULL |
| PASSWORD | Пароль клиента | VARCHAR2(100) | NOT NUL |

Таблица «Product» предназначена для хранения информации о продуктах. В неё входят следующие столбцы:

Таблица 1.2 – Столбцы таблицы **Product**

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Описание | Тип | Ограничения |
| ID | Уникальный идентификатор продукта | NUMBER | PRIMARY KEY |
| NAME | Название товара | VARCHAR2(100) | NOT NULL |
| DESCRIPTION | Описание товара | CLOB | - |
| PRICE | Цена товара | NUMBER (10,2) | NOT NULL |
| SELLER\_ID | Идентификатор продавца | NUMBER | FOREIGN KEY (Seller.id) |

Таблица «Orders» предназначена для хранения информации о заказах. В неё входят следующие столбцы:

Таблица 1.3 – Столбцы таблицы **Orders**

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Описание | Тип | Ограничения |
| ID | Уникальный идентификатор заказа | NUMBER | PRIMARY KEY |
| CUSTOMER\_ID | Идентификатор клиента | NUMBER | FOREIGN KEY (Customer.id) |
| ORDER\_DATE | Дата заказа | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |
| STATUS | Статус заказа | VARCHAR2(50) | - |

Таблица «Administrator» предназначена для хранения информации о заказах. В неё входят следующие столбцы:

Таблица 1.4 – Столбцы таблицы **Administrator**

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Описание | Тип | Ограничения |
| ID | Уникальный идентификатор администратора | NUMBER | PRIMARY KEY |
| NAME | Имя администратора | VARCHAR2(100) | NOT NULL |
| EMAIL | Электронная почта администратора | VARCHAR2(255) | UNIQUE |
| PASSWORD | Пароль администратора | VARCHAR2(100) | NOT NULL |

Таблица «Seller» предназначена для хранения информации о заказах. В неё входят следующие столбцы:

Таблица 1.5 – Столбцы таблицы **Seller**

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Описание | Тип | Ограничения |
| ID | Уникальный идентификатор продавца | NUMBER | PRIMARY KEY |
| NAME | Имя продавца | VARCHAR2(100) | NOT NULL |
| EMAIL | Электронная почта продавца | VARCHAR2(255) | UNIQUE |
| PASSWORD | Пароль продавца | VARCHAR2(100) | NOT NULL |

**Описание объектов СУБД Oracle**

**Последовательность "** **CUSTOMER\_SEQ":**

Эта последовательность используется для генерации уникальных идентификаторов (ID) для новых записей в таблице Customer. Она гарантирует, что каждому новому клиенту будет присвоен уникальный ID, который используется в качестве первичного ключа для таблицы.

|  |
| --- |
| create sequence customer\_seq  start with 1000  increment by 10  minvalue 1000  maxvalue 999999  cache 20; |

**Последовательность "** **PRODUCT\_SEQ":**

Эта последовательность используется для генерации уникальных идентификаторов для новых записей в таблице **Product**. Она помогает автоматизировать процесс создания уникальных значений для столбца id в таблице товаров.

|  |
| --- |
| create sequence product\_seq  start with 500  increment by 5  minvalue 500  maxvalue 5000  cycle  cache 10; |

**Последовательность "** **ORDER\_SEQ":**

Последовательность order\_seq используется для генерации уникальных идентификаторов для записей в таблице Orders. Этот идентификатор необходим для уникального обозначения каждого заказа, который размещает клиент.

|  |
| --- |
| create sequence order\_seq  start with 1  increment by 10  noorder  minvalue 1  maxvalue 1000000  nocycle  cache 20; |

**Последовательность "** **ADMINISTRATOR\_SEQ":**

Эта последовательность используется для генерации уникальных идентификаторов для записей в таблице Administrator. Она нужна для создания уникальных ID для каждого администратора.

|  |
| --- |
| create sequence administator\_seq  start with 1  increment by 2  minvalue 1  maxvalue 100  nocycle  cache 5; |

**Последовательность "** **SELLER\_SEQ":**

Последовательность seller\_seq используется для генерации уникальных идентификаторов для записей в таблице **Seller**. Она необходима для создания уникальных значений для столбца id каждого продавца.

|  |
| --- |
| create sequence seller\_seq  start with 100  increment by 7  minvalue 100  maxvalue 99999  nocycle  cache 15; |

**Индекс "** **IDX\_CUSTOMER\_EMAIL ":**

Индекс idx\_customer\_email создается для ускорения поиска записей в таблице Customer по столбцу email. Этот индекс полезен, когда часто выполняются запросы на поиск клиентов по их электронной почте. Использование индекса значительно снижает время выполнения таких запросов, улучшая производительность.

|  |
| --- |
| create index idx\_customer\_email on Customer(email); |

**Индекс "** **IDX\_PRODUCT\_NAME ":**

Индекс idx\_product\_name создается для ускорения поиска записей в таблице Product по столбцу name. Это важно, если часто выполняются запросы, которые фильтруют или сортируют товары по их имени. Индекс улучшает производительность таких операций, обеспечивая быстрый доступ к строкам таблицы с нужным значением в столбце name.

|  |
| --- |
| create index idx\_product\_name on Product(name); |

**Индекс "** **IDX\_ORDER\_STATUS ":**

Индекс idx\_order\_status создается для ускорения поиска записей в таблице Orders по столбцу status. Это полезно, когда необходимо быстро найти заказы с определенным статусом, например, все заказы со статусом «Pending», «Shipped», «Delivered». Индекс помогает ускорить такие запросы, делая их более эффективными.

|  |
| --- |
| create index idx\_order\_status on Orders(status); |

**Представление "** **VV\_ORDERS ":**

Представление vw\_orders предназначено для отображения информации о заказах с объединением данных из таблиц Orders, Customer и Product. Это представление позволяет быстро получить сводную информацию о каждом заказе, включая данные о клиенте, продукте, дате заказа и текущем статусе заказа.

|  |
| --- |
| create view vw\_orders as  select  o.Id as Order\_Id,  c.Name as Customer\_Name,  p.Name as Product\_Name,  o.Order\_Date,  o.Status  from Orders o  join Customer c on o.Customer\_Id = c.Id  join Product p on c.Product\_Id = p.Id; |

**Представление "** **VV\_SELLERS\_PRODUCTS":**

Представление vw\_sellers\_products предназначено для отображения товаров, связанных с продавцами. Оно объединяет таблицы Seller и Product, предоставляя информацию о продавцах и их товарах, включая цену товара. Это представление полезно для получения информации о товарах, которые продаются каждым продавцом.

|  |
| --- |
| create view vw\_sellers\_products as  select  s.Name as Seller\_Name,  p.Name as Product\_Name,  p.Price  from Seller s join Product p on s.Id = p.Seller\_Id; |

**Процедура "** **ADDCUSTOMERS ":**

Процедура AddCustomer используется для добавления нового клиента в таблицу Customer. Она принимает в качестве параметров имя клиента, его электронную почту, пароль и идентификатор продукта, который клиент покупает. Процедура генерирует новый идентификатор для клиента с помощью последовательности customer\_seq.

|  |
| --- |
| create or replace procedure AddCustomer (  p\_name in varchar2,  p\_email in varchar2,  p\_password in varchar2,  p\_product\_id in number  ) as  begin  insert into Customer (Id, Name, Email, Password, Product\_Id)  values (customer\_seq.nextval, p\_name, p\_email, p\_password, p\_product\_id);  commit;  end; |

**Процедура "** **UPDATEORDERSTATUS ":**

Процедура UpdateOrderStatus используется для обновления статуса заказа в таблице Orders. Она принимает в качестве параметров идентификатор заказа и новый статус, который должен быть присвоен заказу.

|  |
| --- |
| create or replace procedure UpdateOrderStatus (  p\_order\_id in number,  p\_new\_status in varchar2  ) AS  begin  update orders  set Status = p\_new\_status  where Id = p\_order\_id;  commit;  end; |

**Процедура "** **DELETEPRODUCT":**

Процедура DeleteProduct используется для удаления товара из таблицы Product. Она принимает в качестве параметра идентификатор товара, который необходимо удалить.

|  |
| --- |
| create or replace procedure DeleteProduct (  p\_product\_id IN number  ) as  begin  delete from Product where Id = p\_product\_id;  commit;  end; |

**Функция "** **ADDPRODUCT":**

Функция AddProduct используется для добавления нового товара в таблицу Product. Она принимает параметры для имени товара, его описания, цены и идентификатора продавца. После добавления товара функция возвращает его уникальный идентификатор, сгенерированный с помощью последовательности product\_seq.

|  |
| --- |
| create or replace function AddProduct (  p\_name in varchar2,  p\_description in varchar2,  p\_price in number,  p\_seller\_id in number  ) return number is  new\_product\_id number;    pragma autonomous\_transaction;  begin  insert into Product (Id, Name, Description, Price, Seller\_Id)  values (product\_seq.nextval, p\_name, p\_description, p\_price, p\_seller\_id)  returning Id into new\_product\_id;    commit;    return new\_product\_id;  exception  when others then  rollback;  return null;  end; |

**Функция "** **GETCUSTOMERORDERTOTAL":**

Функция GetCustomerOrderTotal вычисляет общую стоимость всех заказов, сделанных определенным клиентом. Она принимает в качестве параметра идентификатор клиента и возвращает общую сумму стоимости всех его заказов.

|  |
| --- |
| create or replace function GetCustomerOrderTotal (  p\_customer\_id in number  ) return number is  total number;  begin  select sum(p.Price) into total from orders o  join Customer c on o.Customer\_Id = c.Id  join Product p on c.Product\_Id = p.Id  where o.Customer\_Id = p\_customer\_id;  return NVL(total, 0);  end; |

**Функция "** **SELLEREXISTS":**

Функция SellerExists проверяет, существует ли продавец с указанным адресом электронной почты. Функция возвращает 1, если продавец с таким email существует, и 0, если не существует.

|  |
| --- |
| create or replace function SellerExists (  p\_email in varchar2  ) return number is  exists\_flag number;  begin  select count(\*) into exists\_flag from Seller  where Email = p\_email;  if exists\_flag > 0 then  return 1;  else  return 0;  end if;  end; |

**Описание таблиц СУБД SQL Server**

Таблица «Customer» хранит информацию о клиентах. В нее входят следующие столбцы:

Таблица 1.6 – Столбцы таблицы **Customer**

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Описание | Тип | Ограничения |
| ID | Уникальный идентификатор клиента | INT IDENTITY | PRIMARY KEY,  IDENTITY(1000,10) |
| PRODUCT\_ID | Идентификатор продукта, связанный с клиентом | INT | FOREIGN KEY (REFERENCES Product.id) |
| NAME | Имя клиента | VARCHAR(100) | - |
| EMAIL | Email клиента | VARCHAR(255) | UNIQUE |
| PASSWORD | Пароль клиента | VARCHAR(100) | - |

Таблица «Product» предназначена для хранения информации о продуктах. В неё входят следующие столбцы:

Таблица 1.7– Столбцы таблицы **Product**

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Описание | Тип | Ограничения |
| ID | Уникальный идентификатор продукта | INT IDENTITY | PRIMARY KEY,  IDENTITY(500,5) |
| NAME | Название товара | VARCHAR(100) | - |
| DESCRIPTION | Описание товара | TEXT | - |
| PRICE | Цена товара | DECIMAL (10,2) | NOT NULL |
| SELLER\_ID | Идентификатор продавца | INT | FOREIGN KEY (REFERENCES Seller.id) |

Таблица «Orders» предназначена для хранения информации о заказах. В неё входят следующие столбцы:

Таблица 1.8 – Столбцы таблицы **Orders**

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Описание | Тип | Ограничения |
| ID | Уникальный идентификатор заказа | INT IDENTITY | PRIMARY KEY, IDENTITY(1,10) |
| CUSTOMER\_ID | Идентификатор клиента | INT | FOREIGN KEY (REFERENCES Customer.id) |
| ORDER\_DATE | Дата заказа | DATETIME | DEFAULT (GETDATE()) |
| STATUS | Статус заказа | VARCHAR(50) | - |

Таблица «Administrator» предназначена для хранения информации о заказах. В неё входят следующие столбцы:

Таблица 1.9 – Столбцы таблицы **Administrator**

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Описание | Тип | Ограничения |
| ID | Уникальный идентификатор администратора | INT IDENTITY | PRIMARY KEY,  IDENTITY(1,2) |
| NAME | Имя администратора | VARCHAR(100) | - |
| EMAIL | Электронная почта администратора | VARCHAR(255) | UNIQUE |
| PASSWORD | Пароль администратора | VARCHAR(100) | - |

Таблица «Seller» предназначена для хранения информации о заказах. В неё входят следующие столбцы:

Таблица 1.10 – Столбцы таблицы **Seller**

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Описание | Тип | Ограничения |
| ID | Уникальный идентификатор продавца | INT IDENTITY | PRIMARY KEY,  IDENTITY(100,7) |
| NAME | Имя продавца | VARCHAR(100) | - |
| EMAIL | Электронная почта продавца | VARCHAR2(255) | UNIQUE |
| PASSWORD | Пароль продавца | VARCHAR2(100) | - |

**Описание объектов СУБД SQL SERVER**

**Индекс "** **IDX\_CUSTOMER\_EMAIL ":**

Индекс idx\_customer\_email создается для ускорения поиска записей в таблице Customer по столбцу email. Этот индекс полезен, когда часто выполняются запросы на поиск клиентов по их электронной почте. Использование индекса значительно снижает время выполнения таких запросов, улучшая производительность.

|  |
| --- |
| CREATE INDEX idx\_customer\_email ON Customer(email); |

**Индекс "** **IDX\_PRODUCT\_NAME ":**

Индекс idx\_product\_name создается для ускорения поиска записей в таблице Product по столбцу name. Это важно, если часто выполняются запросы, которые фильтруют или сортируют товары по их имени. Индекс улучшает производительность таких операций, обеспечивая быстрый доступ к строкам таблицы с нужным значением в столбце name.

|  |
| --- |
| CREATE INDEX idx\_product\_name ON Product(name); |

**Индекс "** **IDX\_ORDER\_STATUS ":**

Индекс idx\_order\_status создается для ускорения поиска записей в таблице Orders по столбцу status. Это полезно, когда необходимо быстро найти заказы с определенным статусом, например, все заказы со статусом «Pending», «Shipped», «Delivered». Индекс помогает ускорить такие запросы, делая их более эффективными.

|  |
| --- |
| CREATE INDEX idx\_order\_status ON Orders(status); |

**Представление "** **VV\_ORDERS ":**

Представление vw\_orders предназначено для отображения информации о заказах с объединением данных из таблиц Orders, Customer и Product. Это представление позволяет быстро получить сводную информацию о каждом заказе, включая данные о клиенте, продукте, дате заказа и текущем статусе заказа.

|  |
| --- |
| create view vw\_orders as  select  o.id as order\_id,  c.name as customer\_name,  p.name as product\_name,  o.order\_date,  o.status  from orders o  join customer c on o.customer\_id = c.id  join product p on c.product\_id = p.id; |

**Представление "** **VV\_SELLERS\_PRODUCTS":**

Представление vw\_sellers\_products предназначено для отображения товаров, связанных с продавцами. Оно объединяет таблицы Seller и Product, предоставляя информацию о продавцах и их товарах, включая цену товара. Это представление полезно для получения информации о товарах, которые продаются каждым продавцом.

|  |
| --- |
| create view vw\_sellers\_products as  select  s.name as seller\_name,  p.name as product\_name,  p.price  from seller s join product p on s.id = p.seller\_id; |

**Процедура "** **ADDCUSTOMERS ":**

Процедура AddCustomer используется для добавления нового клиента в таблицу Customer. Она принимает в качестве параметров имя клиента, его электронную почту, пароль и идентификатор продукта, который клиент покупает. Процедура генерирует новый идентификатор для клиента с помощью последовательности customer\_seq.

|  |
| --- |
| CREATE PROCEDURE dbo.addproduct (  @p\_name VARCHAR(100),  @p\_description TEXT,  @p\_price DECIMAL(10,2),  @p\_seller\_id INT,  @new\_product\_id INT OUTPUT  )  AS  BEGIN  INSERT INTO product (name, description, price, seller\_id)  VALUES (@p\_name, @p\_description, @p\_price, @p\_seller\_id);    SET @new\_product\_id = SCOPE\_IDENTITY();  END;  GO |

**Процедура "** **UPDATEORDERSTATUS ":**

Процедура UpdateOrderStatus используется для обновления статуса заказа в таблице Orders. Она принимает в качестве параметров идентификатор заказа и новый статус, который должен быть присвоен заказу.

|  |
| --- |
| create procedure updateorderstatus (  @p\_order\_id int,  @p\_new\_status varchar(50)  )  as  begin  update orders  set status = @p\_new\_status  where id = @p\_order\_id;  end;  go |

**Процедура "** **DELETEPRODUCT":**

Процедура DeleteProduct используется для удаления товара из таблицы Product. Она принимает в качестве параметра идентификатор товара, который необходимо удалить.

|  |
| --- |
| create procedure deleteproduct (  @p\_product\_id int  )  as  begin  delete from product where id = @p\_product\_id;  end;  go |

**Функция "** **ADDPRODUCT":**

Функция AddProduct используется для добавления нового товара в таблицу Product. Она принимает параметры для имени товара, его описания, цены и идентификатора продавца. После добавления товара функция возвращает его уникальный идентификатор, сгенерированный с помощью последовательности product\_seq.

|  |
| --- |
| create or replace function AddProduct (  p\_name in varchar2,  p\_description in varchar2,  p\_price in number,  p\_seller\_id in number  ) return number is  new\_product\_id number;    pragma autonomous\_transaction;  begin  insert into Product (Id, Name, Description, Price, Seller\_Id)  values (product\_seq.nextval, p\_name, p\_description, p\_price, p\_seller\_id)  returning Id into new\_product\_id;    commit;    return new\_product\_id;  exception  when others then  rollback;  return null;  end; |

**Функция "** **GETCUSTOMERORDERTOTAL":**

Функция GetCustomerOrderTotal вычисляет общую стоимость всех заказов, сделанных определенным клиентом. Она принимает в качестве параметра идентификатор клиента и возвращает общую сумму стоимости всех его заказов.

|  |
| --- |
| create function getcustomerordertotal (  @p\_customer\_id int  )  returns decimal(10,2)  as  begin  declare @total decimal(10,2);    select @total = sum(p.price)  from orders o  join customer c on o.customer\_id = c.id  join product p on c.product\_id = p.id  where o.customer\_id = @p\_customer\_id;  return isnull(@total, 0);  end;  go |

**Функция "** **SELLEREXISTS":**

Функция SellerExists проверяет, существует ли продавец с указанным адресом электронной почты. Функция возвращает 1, если продавец с таким email существует, и 0, если не существует.

|  |
| --- |
| create function sellerexists (  @p\_email varchar(255)  )  returns bit  as  begin  declare @exists\_flag bit;  select @exists\_flag = case when count(\*) > 0 then 1 else 0 end  from seller  where email = @p\_email;  return @exists\_flag;  end;  go |