Itroduction

ERD

Normal forms

Coding parts(function,procedure,exception,triggers

Introduction

777-pham

Idea: We know that the transition from offline sales to online is now happening at a rapid pace.Therefore, our task is to make it easier for the seller. Of all the necessary online services, the most important is the database system. So in this project we plan to create a database system in which the seller could sell his goods online.

9 entities: Customer

Product

Staff

Supplier

Basket

Item 1

Item 2

Delivery

Payment

Normal Forms:

In our tables we contain all three normal forms

1)In any tables you can see it's in first normal form,

because each table cell contain a single value and

each record (primary key) is unique.

2)As it should be in second normal form, in any tables

Single Column Primary Key does not functionally dependant

on any subset of candidate key relation and it is in first

normal form.

3)In third normal form has to be no transitive functional

dependencies. In our tables we divided Basket and Item \_1,

Supplier and Item \_2 to remove transitive functional dependency.

In other tables we do not have this type of functional dependency.

Since in each basket (NO\_order) the order number can have several products, we moved and created a new entity named ITEM\_1. The same scheme works with the supplier table.

Procedures

1)The procedure is the cost of the products that were ordered with the quantity. We multiply the quantity of each items by the price and group by order ID

CREATE OR REPLACE PROCEDURE basket\_total(

v\_No\_order IN item\_1.NO\_order%TYPE,

v\_order\_total OUT NUMBER

)

IS

BEGIN

SELECT SUM(item\_1.quantity \* product.price) INTO v\_order\_total

FROM item\_1

JOIN product ON item\_1.product\_id = product.product\_id

WHERE item\_1.No\_order = v\_NO\_order

GROUP BY item\_1.NO\_order;

DBMS\_OUTPUT.PUT\_LINE('Total sum for items ' v\_No\_order ': $' v\_order\_total);

END;

DECLARE

v\_order\_total NUMBER;

BEGIN

basket\_total(79, v\_order\_total);

DBMS\_OUTPUT.PUT\_LINE('Total sum for order 79: $' v\_order\_total);

END;

2)Thanks to this procedure , users can see how many items are in his ıtem\*basket. This makes it very convenient for the user to navigate how many items are in the basket. create or replace PROCEDURE check\_item\_basket(p\_item\_1\_id IN NUMBER)

IS

v\_basket\_count NUMBER;

BEGIN

SELECT quantity INTO v\_basket\_count

FROM item\_1

WHERE item\_1\_id = p\_item\_1\_id;

IF SQL%ROWCOUNT = 1 THEN

DBMS\_OUTPUT.PUT\_LINE('The basket with ID ' p\_item\_1\_id ' contains ' v\_basket\_count ' items.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('The basket with ID ' p\_item\_1\_id ' not found.');

END IF;

END;

1) Function to get customer information by ID

CREATE OR REPLACE FUNCTION get\_customer\_info(p\_customer\_id IN NUMBER)

RETURN customer%ROWTYPE

AS

v\_customer customer%ROWTYPE;

BEGIN

SELECT \*

INTO v\_customer

FROM customer

WHERE customer\_id = p\_customer\_id;

RETURN v\_customer;

END;

DECLARE

v\_customer customer%ROWTYPE;

BEGIN

v\_customer := get\_customer\_info(1);

DBMS\_OUTPUT.PUT\_LINE('Name: ' v\_customer.name);

DBMS\_OUTPUT.PUT\_LINE('Email: ' v\_customer.email);

END;

2) Function for getting staff who works in the same speciality:

create or replace FUNCTION get\_staff(same\_speciality IN VARCHAR2)

RETURN staff%ROWTYPE

AS

v\_staff staff%ROWTYPE;

BEGIN

SELECT \*

INTO v\_staff

FROM staff

WHERE speciality = same\_speciality;

RETURN v\_staff;

END;

/

DECLARE

v\_staff staff%ROWTYPE;

BEGIN

v\_staff := get\_staff('Occupational Therapist');

DBMS\_OUTPUT.PUT\_LINE(v\_staff.staff\_id ' - ' v\_staff.name ' - ' v\_staff.speciality);

END;

2) Function which counts the number of records

create or replace function count\_customers

return number

is

v\_customers\_count number;

begin

select count(\*) into v\_customers\_count

from customer;

return v\_customers\_count;

end;

DECLARE

Res NUMBER;

BEGIN

Res := count\_customers();

DBMS\_OUTPUT.PUT\_LINE('Count of customers is ' || Res);

END;

/

4) Function to get product information by ID

CREATE OR REPLACE FUNCTION get\_product\_info(p\_product\_id IN NUMBER)

RETURN product%ROWTYPE

AS

v\_product product%ROWTYPE;

BEGIN

SELECT \*

INTO v\_product

FROM product

WHERE product\_id = p\_product\_id;

RETURN v\_product;

END;

DECLARE

v\_product product%ROWTYPE;

BEGIN

v\_product := get\_product\_info(1);

DBMS\_OUTPUT.PUT\_LINE('Name: ' v\_product.name);

DBMS\_OUTPUT.PUT\_LINE('Price: ' v\_product.price);

END;

Triggers:

1)Trigger which will count row of this table

where insertig new parametres(In TABLE STAFF):

Create or replace trigger rowcount\_staff

Before insert on staff

for each row

DECLARE

row\_count NUMBER;

begin

Select count(\*) into row\_count from staff;

if inserting then

dbms\_output.put\_line('Row count of this table:' || row\_count);

end if;

end;

insert into staff

(staff\_id, name, email, speciality, password, phone)

values(51, 'Sabina Turgan', 'sab@mail.ru', 'Adviser', 'Sab123', '7777777777')

delete from staff where staff\_id = 51

2)This trigger will works when we get products

(means availibility turns to 'true' from 'false')

it's helps us to update expiration date with today's date:

create or replace trigger expiration\_date

before update of availability on product

for each row

begin

if :new.availability = 'true' then

:NEW.expiration\_date := SYSDATE;

end if;

end;

update product

set availability = 'true'

where product\_id = 50;

update product

set expiration\_date = '12/10/2021'

where product\_id = 50;

select \* from product where product\_id = 50

3)Also similar as previous trigger, will write expiration date with today's date when we

insert new product:

create or replace trigger expiration\_date\_insert

before insert on product

for each row

begin

if :new.availability = 'true' then

:new.expiration\_date := CURRENT\_DATE;

end if;

end;

drop trigger expiration\_date\_insert

insert into product(product\_id, name, price, age\_group, description, category, availability, unity)

values(51, 'ospamoks', '2000', '18', 'antibiotics', 'virus', 'true', 'mg')

select \* from product where product\_id = 51

delete from product where product\_id = 51

Exceptions

1)This exception is related to the trigger. with this exception, the administrator cannot enter medicines into the system whose name contains less than 5 letters. when he tries to enter, an error message will come out: the product name must contain more than 5 letters

create or replace trigger CHARACTER

before insert on product

for each row

declare

invalid\_name exception;

begin

IF length(:NEW.name) < 5 THEN

RAISE invalid\_name;

END IF;

DBMS\_OUTPUT.PUT\_LINE ('Enter name of product: ' || :NEW.name);

EXCEPTION

WHEN invalid\_name THEN

dbms\_output.put\_line('Product name must be bigger than 5 letters');

WHEN others THEN

dbms\_output.put\_line('Error!');

END;

insert into product(product\_id, name, price, age\_group, description, category, availability, unity)

values(52, 'osp', '2000', '18', 'antibiotics', 'virus', 'true', 'mg')

2) The main ideology of this exception is that a person cannot order an unlimited number of goods at the same time. And we will introduce a restriction on this. if the number of products exceeds a certain time, an error message appears:"There can be a maximum of 10 items in your cart"

create or replace trigger ITEM

before insert on item\_1

for each row

declare

too\_many\_item exception;

begin

IF :NEW.quantity > 11 THEN

RAISE too\_many\_item;

END IF;

EXCEPTION

WHEN too\_many\_item THEN

dbms\_output.put\_line('There can be no more than 10 items in your basket');

WHEN others THEN

dbms\_output.put\_line('Error! Try again');

END;

Triggers

1)Trigger which will count row of this table

where insertig new parametres(In TABLE BASKET):

create or replace trigger rowcount\_basket

Before insert on basket

for each row

DECLARE

row\_count NUMBER;

begin

Select count(\*) into row\_count from basket;

if inserting then

dbms\_output.put\_line('Row count of this table:' || row\_count);

end if;

end;

1)Trigger which will count row of this table

where insertig new parametres(In TABLE CUSTOMER):

create or replace trigger rowcount\_customer

before insert on customer

for each row

declare

row\_count number;

begin

select count(\*) into row\_count from customer;

if inserting then

dbms\_output.put\_line('Row count of this table:' || row\_count);

end if;

end;

1)Trigger which will count row of this table

where insertig new parametres(In TABLE DELIVERY):

create or replace trigger rowcount\_delivery

Before insert on delivery

for each row

DECLARE

row\_count NUMBER;

begin

Select count(\*) into row\_count from delivery;

if inserting then

dbms\_output.put\_line('Row count of this table:' || row\_count);

end if;

end;

1)Trigger which will count row of this table

where insertig new parametres(In TABLE ITEM\_1):

create or replace trigger rowcount\_item1

Before insert on item\_1

for each row

DECLARE

row\_count NUMBER;

begin

Select count(\*) into row\_count from item\_1;

if inserting then

dbms\_output.put\_line('Row count of this table:' || row\_count);

end if;

end;

1)Trigger which will count row of this table

where insertig new parametres(In TABLE ITEM\_2):

create or replace trigger rowcount\_item2

Before insert on item\_2

for each row

DECLARE

row\_count NUMBER;

begin

Select count(\*) into row\_count from item\_2;

if inserting then

dbms\_output.put\_line('Row count of this table:' || row\_count);

end if;

end;

1)Trigger which will count row of this table

where insertig new parametres(In TABLE PAYMENT):

create or replace trigger rowcount\_payment

Before insert on payment

for each row

DECLARE

row\_count NUMBER;

begin

Select count(\*) into row\_count from payment;

if inserting then

dbms\_output.put\_line('Row count of this table:' || row\_count);

end if;

end;

1)Trigger which will count row of this table

where insertig new parametres(In TABLE PRODUCT):

create or replace trigger rowcount\_product

Before insert on product

for each row

DECLARE

row\_count NUMBER;

begin

Select count(\*) into row\_count from product;

if inserting then

dbms\_output.put\_line('Row count of this table:' || row\_count);

end if;

end;

1)Trigger which will count row of this table

where insertig new parametres(In TABLE SUPPLIER):

create or replace trigger rowcount\_supplier

Before insert on supplier

for each row

DECLARE

row\_count NUMBER;

begin

Select count(\*) into row\_count from supplier;

if inserting then

dbms\_output.put\_line('Row count of this table:' || row\_count);

end if;

end;

Overall, this online pharmacy database project provides a solid foundation for managing the information related to pharmacy products, customers, orders, and staff, and can be expanded upon to include additional functionality as needed.