

Міністерство освіти і науки України

НТУУ«Київський політехнічний інститут»

Фізико-технічний інститут

Бази даних

Расчетно-графическая работа

**Виконав:**

Студент III курсу ФТІ

групи ФБ-31

Зацепін Олексій

**Перевірив:**

Коломицев М.В.

### *Варіант 7*

**Предметна область** – Складское підприємство.

**Задачі, що вирішує інформаційна система** – наявність товарів на складі.

**Завдання на обробку даних:**

1. Створити впорядковані списки:

- Товарів за категоріями (по складам);

- Товарів, які є на вибраному складі, але немає на інших складах.

2. обчислення:

- Перевірити, чи достатньо на складі товарів для виконання конкретного замовлення.

3. корекція:

- Зміна залишку товару на складі.

4. Звіти виду:

- "Наявність товарів": Товар - місце зберігання - залишок (з урахуванням одиниці виміру товару).

- "Замовлені товари" Товар - місце зберігання - замовлену кількість.

- "Вага замовлення": номер замовлення - загальна кількість замовлених товарів.

- "Картка товару": номер складу - номер лінії - номер стелажа - одиниця виміру - залишок товару.

Create table Customer(

CustomerID int primary key auto\_increment,

FIOCustomer VARCHAR(20) NOT NULL,

MobileNumber VARCHAR(10) NOT NULL UNIQUE,

Address VARCHAR(40) NOT NULL

);

Create table Distributor(

DistributorID int primary key auto\_increment,

DistributorName varchar(20) not null,

MobileNumber VARCHAR(10) NOT NULL UNIQUE,

Address VARCHAR(40) NOT NULL

);

Create table Product(

ProductID int primary key auto\_increment,

Price decimal not null,

ProductTitle varchar(20) not null,

Category varchar(30) not null,

WeightPerOne double

);

create table StorageHouse(

StorageID int primary key auto\_increment,

Address VARCHAR(40) NOT NULL

);

create table PlaceOnStorage(

PlaceID int,

ProductID int,

StorageID int,

Quantity int not null,

NumberRow mediumint not null,

NumberPosition smallint not null,

primary key(PlaceID,ProductID,StorageID,NumberRow,NumberPosition),

CONSTRAINT ProductPlaceForeign FOREIGN KEY (ProductID) REFERENCES Product(ProductID)

ON DELETE CASCADE on update cascade,

CONSTRAINT HouseForeign FOREIGN KEY (StorageID) REFERENCES StorageHouse(StorageID)

ON DELETE CASCADE on update cascade

);

Create table OrderList(

OrderListID int primary key auto\_increment,

CustomerID int,

ProductID int,

Quantity int not null,

DateOfOrder datetime,

CONSTRAINT OrderListForeign FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID)

ON DELETE CASCADE,

CONSTRAINT ProductForeign FOREIGN KEY (ProductID) REFERENCES Product(ProductID)

ON DELETE CASCADE

);

Create table Distribution(

DistributionID int primary key auto\_increment,

DistributorID int,

ProductID int,

Quantity int not null,

DateOfDistribution datetime,

CONSTRAINT DistributeForeign FOREIGN KEY (DistributorID) REFERENCES Distributor(DistributorID)

ON DELETE CASCADE,

CONSTRAINT ProductDisForeign FOREIGN KEY (ProductID) REFERENCES Product(ProductID)

ON DELETE CASCADE

);

ALTER TABLE Distribution MODIFY COLUMN DateOfDistribution TIMESTAMP DEFAULT CURRENT\_TIMESTAMP;

ALTER TABLE OrderList MODIFY COLUMN DateOfOrder TIMESTAMP default CURRENT\_TIMESTAMP;

ALTER TABLE Product CHANGE COLUMN Price PricePerOne decimal;

DELIMITER //;

CREATE TRIGGER `before\_insert\_customer`

before insert ON Storage\_Company.Customer

FOR EACH ROW

BEGIN

SET @FIO:= NEW.FIOCustomer;

SET @PhoneNumber := NEW.MobileNumber;

IF @FIO not REGEXP '[:alpha:]|[:blank:]' THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'MyError:FIO must contains only letters';

END IF;

IF @PhoneNumber not regexp '^-?[0-9]+$' THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'MyError:Number must contains only digitals';

END IF;

END;

Delimiter //;

CREATE TRIGGER `before\_insert\_distributor`

before insert ON Storage\_Company.Distributor

FOR EACH ROW

BEGIN

SET @title:= NEW.DistributorName;

SET @PhoneNumber := NEW.MobileNumber;

IF @title not REGEXP '[:alpha:]|[:blank:]|[:digit:]' THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'MyError:FIO must contains only letters';

END IF;

IF @PhoneNumber not regexp '^-?[0-9]+$' THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'MyError:Number must contains only digitals';

END IF;

END;

Insert into Customer(FIOCustomer,MobileNumber,Address) values ('Dicson Hanry','2','123fsfa');

Insert into Distributor(DistributorName,MobileNumber,Address) values ('Dicson Hanry','4','123fsfa');

INSERT INTO `Storage\_Company`.`StorageHouse` (`StorageID`, `Address`) VALUES ('1', 'Kiev');

INSERT INTO `Storage\_Company`.`StorageHouse` (`StorageID`, `Address`) VALUES ('2', 'Kharkiv');

INSERT INTO `Storage\_Company`.`StorageHouse` (`StorageID`, `Address`) VALUES ('3', 'Lviv');

INSERT INTO `Storage\_Company`.`StorageHouse` (`StorageID`, `Address`) VALUES ('4', 'Odessa');

INSERT INTO `Storage\_Company`.`StorageHouse` (`StorageID`, `Address`) VALUES ('5', 'Cherkasy');

INSERT INTO `Storage\_Company`.`StorageHouse` (`StorageID`, `Address`) VALUES ('6', 'Vinnitsa');

insert into Product(Price,ProductTitle,WeightPerOne,Category) values

(100.0,"headphones common",100,'accessories'),

(2200.0,"headphones beast",200,'accessories'),

(5.0,"gum love is",30,'gum');

INSERT INTO `Storage\_Company`.`Product` (`ProductID`, `Price`, `ProductTitle`, `Category`, `WeightPerOne`) VALUES ('4', '50', ' \"Roshen Belka\"', 'candy', '20');

UPDATE `Storage\_Company`.`Product` SET `ProductTitle`='\"love is\"' WHERE `ProductID`='3';

INSERT INTO `Storage\_Company`.`PlaceOnStorage` (`PlaceID`, `ProductID`, `StorageID`, `Quantity`, `NumberRow`, `NumberPosition`) VALUES ('7', '1', '4', '25', '1', '1');

INSERT INTO `Storage\_Company`.`PlaceOnStorage` (`PlaceID`, `ProductID`, `StorageID`, `Quantity`, `NumberRow`, `NumberPosition`) VALUES ('1', '1', '1', '20', '1', '1');

INSERT INTO `Storage\_Company`.`PlaceOnStorage` (`PlaceID`, `ProductID`, `StorageID`, `Quantity`, `NumberRow`, `NumberPosition`) VALUES ('2', '1', '1', '30', '1', '2');

INSERT INTO `Storage\_Company`.`PlaceOnStorage` (`PlaceID`, `ProductID`, `StorageID`, `Quantity`, `NumberRow`, `NumberPosition`) VALUES ('3', '1', '2', '20', '1', '1');

INSERT INTO `Storage\_Company`.`PlaceOnStorage` (`PlaceID`, `ProductID`, `StorageID`, `Quantity`, `NumberRow`, `NumberPosition`) VALUES ('4', '2', '3', '20', '1', '1');

INSERT INTO `Storage\_Company`.`PlaceOnStorage` (`PlaceID`, `ProductID`, `StorageID`, `Quantity`, `NumberRow`, `NumberPosition`) VALUES ('5', '3', '1', '200', '1', '3');

INSERT INTO `Storage\_Company`.`PlaceOnStorage` (`PlaceID`, `ProductID`, `StorageID`, `Quantity`, `NumberRow`, `NumberPosition`) VALUES ('6', '4', '1', '20000', '2', '1');

INSERT INTO `Storage\_Company`.`Customer` (`CustomerID`, `FIOCustomer`, `MobileNumber`, `Address`) VALUES ('2', 'Jack Bon', '55000010', 'Kiev');

UPDATE `Storage\_Company`.`Customer` SET `MobileNumber`='222123123', `Address`='Kharkiv' WHERE `CustomerID`='1';

INSERT INTO `Storage\_Company`.`OrderList` (`CustomerID`, `ProductID`, `Quantity`, `DateOfOrder`) VALUES ('1', '1', '10', '10-10-111');

INSERT INTO `Storage\_Company`.`OrderList` (`CustomerID`, `ProductID`, `Quantity`, `DateOfOrder`) VALUES ('2', '1', '10', '11-12-2000');

INSERT INTO `Storage\_Company`.`Distributor` (`DistributorID`, `DistributorName`, `MobileNumber`, `Address`) VALUES ('2', 'Samsung', '1000000', 'Kiev');

UPDATE `Storage\_Company`.`Distributor` SET `MobileNumber`='2000000', `Address`='Kiev' WHERE `DistributorID`='1';

INSERT INTO `Storage\_Company`.`Distributor` (`DistributorID`, `DistributorName`, `MobileNumber`, `Address`) VALUES ('3', 'Meizu', '0222222', 'Pekin');

INSERT INTO `Storage\_Company`.`Distributor` (`DistributorID`, `DistributorName`, `MobileNumber`, `Address`) VALUES ('4', 'HTC ', '1002000', 'Gongkong');

INSERT INTO `Storage\_Company`.`Distributor` (`DistributorID`, `DistributorName`, `MobileNumber`, `Address`) VALUES ('5', 'Roshen', '2842818', 'Kiev');

INSERT INTO `Storage\_Company`.`Distributor` (`DistributorID`, `DistributorName`, `MobileNumber`, `Address`) VALUES ('6', 'Microsoft', '1234212', 'New York');

INSERT INTO `Storage\_Company`.`Product` (`ProductID`, `PricePerOne`, `ProductTitle`, `Category`, `WeightPerOne`) VALUES ('5', '10000', 'Samsung Galaxy S4', 'smartphone', '130');

INSERT INTO `Storage\_Company`.`Product` (`ProductID`, `PricePerOne`, `ProductTitle`, `Category`, `WeightPerOne`) VALUES ('6', '9000', 'HTC One X', 'smartphone', '140');

INSERT INTO `Storage\_Company`.`Product` (`ProductID`, `PricePerOne`, `ProductTitle`, `Category`, `WeightPerOne`) VALUES ('7', '3000', 'Windows 10', 'OS', '-');

INSERT INTO `Storage\_Company`.`StorageHouse` (`StorageID`, `Address`) VALUES ('7', 'London');

INSERT INTO `Storage\_Company`.`StorageHouse` (`StorageID`, `Address`) VALUES ('8', 'Pekin');

INSERT INTO `Storage\_Company`.`StorageHouse` (`StorageID`, `Address`) VALUES ('9', 'Gongkong');

DELIMITER //

CREATE PROCEDURE `Unique\_Products`(IN city varchar(40))

LANGUAGE SQL

DETERMINISTIC

SQL SECURITY DEFINER

COMMENT 'only at one storage'

BEGIN

SELECT ProductTitle From Product,PlaceOnStorage,StorageHouse

where StorageHouse.StorageID=PlaceOnStorage.StorageID and PlaceOnStorage.ProductID=Product.ProductID

and Address=city

group by ProductTitle

having ProductTitle not in (SELECT ProductTitle From Product,PlaceOnStorage,StorageHouse

where StorageHouse.StorageID=PlaceOnStorage.StorageID and PlaceOnStorage.ProductID=Product.ProductID

and Address<>city );

END//

DELIMITER $$;

call Unique\_Products("Lviv");

Create view Products\_by\_Category as

select Category,ProductTitle from Product

order by Category;

create view Products\_by\_Storage as

select Address,Product.ProductTitle from Product,PlaceOnStorage,StorageHouse

where StorageHouse.StorageID=PlaceOnStorage.StorageID and PlaceOnStorage.PlaceID=Product.ProductID

order by Address;

create view Product\_Card as

select ProductTitle,Price,Quantity,WeightPerOne Gramms\_per\_one,Address,NumberRow,NumberPosition from Product,PlaceOnStorage,StorageHouse

where StorageHouse.StorageID=PlaceOnStorage.StorageID and PlaceOnStorage.ProductID=Product.ProductID

order by ProductTitle;

create view Weight\_Order\_list as

select OrderListID,OrderList.Quantity\*WeightPerOne as FullWeight from OrderList,Product

where OrderList.ProductID=Product.ProductID;

create view Product\_Balance as

select ProductTitle,Address,Quantity as Balance from Product,PlaceOnStorage,StorageHouse

where StorageHouse.StorageID=PlaceOnStorage.StorageID and PlaceOnStorage.ProductID=Product.ProductID;

create view Ordered\_goods as

select ProductTitle,Address,OrderList.Quantity from OrderList,Product,PlaceOnStorage,StorageHouse

where StorageHouse.StorageID=PlaceOnStorage.StorageID and PlaceOnStorage.ProductID=Product.ProductID

and OrderList.ProductID=Product.ProductID;

select \* from Weight\_Order\_list;

select \* from Product\_Card;

select \* from Products\_by\_Storage;

select \* from Products\_by\_Category;

select \* from Ordered\_goods;

delimiter $$

CREATE TRIGGER `before\_insert\_new\_orderList`

before insert ON Storage\_Company.OrderList

FOR EACH ROW

BEGIN

declare q int;

set @Quantity=NEW.Quantity;

set @productID=NEW.ProductID;

if @Quantity>all(select Quantity from Product,PlaceOnStorage

where Product.ProductID=PlaceOnStorage.ProductID and ProductID=@productID) then

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Not enought product';

END IF;

if @productID not in (select Product.ProductID from Product) then

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'This product doesnt exist';

END IF;

END;

drop procedure if exists `cancel\_Order`;

DELIMITER //

create procedure `cancel\_Order` (IN id int)

language sql

SQL security invoker

BEGIN

declare customerID int;

declare q int;

declare q1 int;

declare prodID int;

declare city varchar(40);

declare cityID int;

select CustomerID into customerID from OrderList where OrderListID=id;

select Address into city from Customer where CustomerID=customerID;

select StorageID into cityID from StorageHouse where Address=city;

select Quantity into q from PlaceOnStorage where StorageID=cityID and ProductID=prodID;

select Quantity into q1 from OrderList where `ProductID`=@productID

and`StorageID`=cityID;

UPDATE `Storage\_Company`.`PlaceOnStorage`

SET `Quantity`=q+q1 WHERE `ProductID`=@productID

and`StorageID`=cityID;

end//

delimiter $$

CREATE TRIGGER `after\_insert\_new\_orderList`

after insert ON Storage\_Company.OrderList

FOR EACH ROW

BEGIN

declare q int;

declare city varchar(30);

declare cityID int;

set @Quantity=NEW.Quantity;

set @productID=NEW.ProductID;

set @customer=NEW.CustomerID;

select Address into city from Customer where CustomerID=@customer;

select StorageID into cityID from StorageHouse where Address=city;

select Quantity into q from PlaceOnStorage where StorageID=cityID and ProductID=@productID;

UPDATE `Storage\_Company`.`PlaceOnStorage`

SET `Quantity`=q-@Quantity WHERE `ProductID`=@productID

and`StorageID`=cityID;

END;

drop trigger `allocateSpace`;

DELIMITER //;

CREATE TRIGGER `allocateSpace`

before insert ON Storage\_Company.PlaceOnStorage

FOR EACH ROW

BEGIN

declare numR int;

declare numPos int;

declare q int;

SET @strID:= NEW.StorageID;

SET @prodID := NEW.ProductID;

SET @q=NEW.Quantity;

SET @numR=NEW.NumberRow;

SET @numPos=NEW.NumberPosition;

/\* if @prodID in (select ProductID from PlaceOnStorage where StorageID=@strID) then

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'haha';

select Quantity into q from PlaceOnStorage where StorageID=@strID and ProductID=@prodID;

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'haha';

UPDATE `Storage\_Company`.`PlaceOnStorage` SET `Quantity`=q+@q WHERE

`ProductID`=@prodID and`StorageID`=@strID;

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'There this product in this storage, so quantity are added';

end if;\*/

while(@numPos in (select NumberPosition from PlaceOnStorage where NumberRow=@numR)) do

#SET @numR=FLOOR((RAND() \* (100-1))+1);

SET @numPos=FLOOR(RAND() \* (100-1));

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'There this product in this storage, so quantity are added';

end while;

#SET NEW.NumberRow=@numR;

SET NEW.NumberPosition=@numPos;

END;