ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ

КАТЕДРА КОМПЮТЪРНИ СИСТЕМИ

КУРСОВА ПРОЕКТ

Дисциплина: „База Данни”

Тема № 30
Да се разработи база данни за система за склад на търговска верига за битова техника, в който стоките са записани по групи, марки, модели, количества и цени. Потребителите да могат да заявяват дадени стоки и да получават определени предварително дефинирани отстъпки за по-големи количества. Допълнете таблиците с необходима информация по ваш избор.

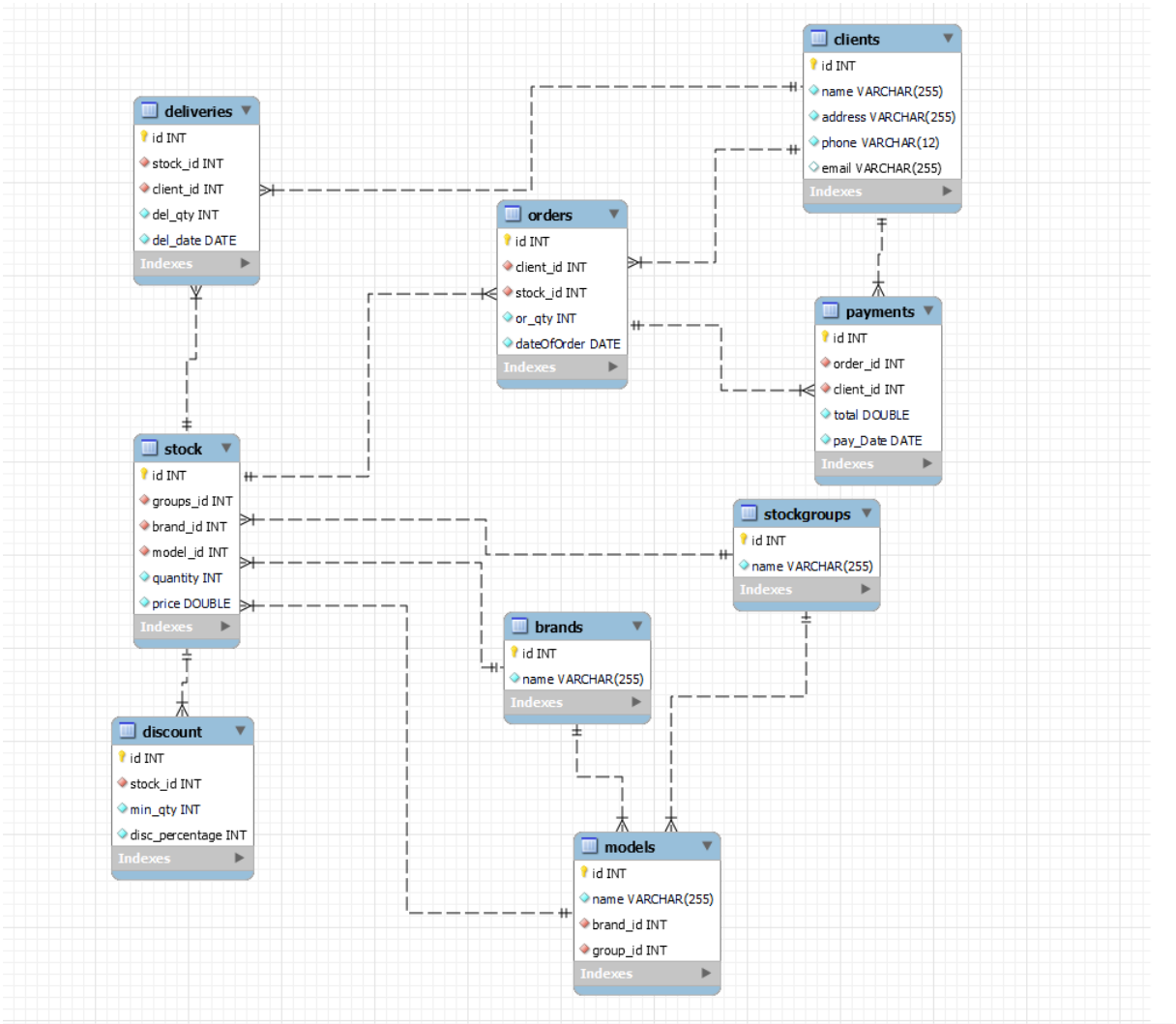
Изготвил:
Фак. № 121221101 Група: 45**Б**

Ръководител:
гл.ас. д-р Петко Данов

София, 2023

1.Да се проектира база от данни и да се представи ER диаграма със съответни CREATE TABLE заявки за средата MySQL:

Основните обекти, за които трябва да съхраняваме информация, според заданието са: stockgroups, models и brands. Допълнително
създаваме още няколко таблици. Първата stock, ще пази наличните количества за дадена стока, както и нейната цена. Втората таблица ще бъде за клиентите. В нея ще се съхраняват всички данни за клиентите. Третата таблица ще е за поръчките, като в нея ще имаме полета за дата на поръчката както и за количество. Четвъртата таблица е за плащанията и там ще се пази платената сума и датата на плащането. Петата таблица е за отстъпките и последната е за доставките, като в нея имаме едно поле от тип ENUM в което ще се пази статуса на поръчката.
За проектирането на базата ще използваме модела ER-диаграма (Entity Relationship Diagram):



Заявките, с които създаваме базата данни и таблиците са:

```
DROP DATABASE IF EXISTS store ;
CREATE DATABASE store;
USE store;


CREATE TABLE stockGroups(
  id int not null auto_increment PRIMARY KEY,
  name VARCHAR(255) NOT NULL
)ENGINE = InnoDB;

CREATE TABLE brands(
  id int not null auto_increment PRIMARY KEY,
  name VARCHAR(255) NOT NULL
)ENGINE = InnoDB;

CREATE TABLE models(
  id int not null auto_increment PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  brand_id int not null,
  group_id int not null,
  constraint foreign key(brand_id) references brands(id),
  constraint foreign key(group_id) references stockGroups(id)
)ENGINE = InnoDB;

CREATE TABLE stock(
  id int not null auto_increment PRIMARY KEY,
  groups_id int not null,
  brand_id int not null,
  model_id int  not null,
  quantity int not null,
  price double not null,
  constraint foreign key (groups_id) references stockGroups(id),
  constraint foreign key (brand_id) references brands(id),
  constraint foreign key (model_id) references models(id)
)ENGINE = InnoDB;

CREATE TABLE clients(
  id int not null auto_increment PRIMARY KEY,
  name varchar(255) not null,
  address varchar(255) not null,
  phone varchar(20) not null,
  email varchar(255)
  )ENGINE = InnoDB;

CREATE TABLE orders(
  id int not null auto_increment PRIMARY KEY,
  client_id int not null,
  stock_id int not null,
  order_quantity int not null,
```

```
  dateOfOrder DATE not null,
  constraint foreign key (client_id) references clients(id),
  constraint foreign key (stock_id) references stock(id)
)ENGINE = InnoDB;

CREATE TABLE payments(
  id int not null auto_increment PRIMARY KEY,
  order_id int not null,
  client_id int not null,
  total double not null,
  pay_Date DATE not null,
  constraint foreign key (client_id) references clients(id),
  constraint foreign key (order_id) references orders(id),
  constraint foreign key(order_id) references orders(id)
)ENGINE = InnoDB;

CREATE TABLE discount(
  id int not null auto_increment PRIMARY KEY,
  stock_id int not null,
  min_quantity int not null,
  discount_percentage int not null,
 constraint foreign key (stock_id) references stock(id)
)ENGINE = InnoDB;

CREATE TABLE deliveries(
  id int not null auto_increment PRIMARY KEY,
  stock_id int not null,
  client_id int not null,
  delivery_quantity int not null,
  status ENUM ('At warehouse','Delivering','Delivered'),
  delivery_date DATE not null,
  constraint foreign key(client_id) REFERENCES clients(id),
  constraint foreign key (stock_id) references stock(id)
)ENGINE = InnoDB;
```

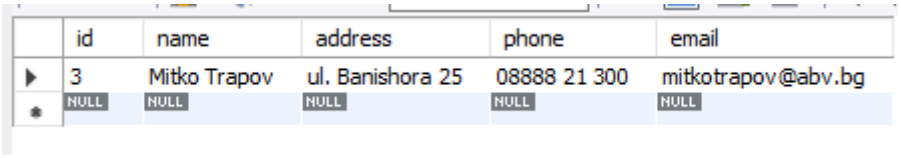
Добавяме и тестови данни в таблиците:

```
INSERT INTO clients (name, address, phone, email)
VALUES ('Ivan Georgiev', 'ul. Slavqnska 7', '088 528 5767', 'ivangeorgiev@abv.bg'),
            ('Petur Ivanov', 'ul. Ivan Vazov 13', '08888 45 500', 'peturivanov@abv.bg'),
            ('Mitko Trapov', 'ul. Banishora 25', '08888 21 300', 'mitkotrapov@abv.bg'),
            ('Georgi Nikolaev', 'ul. Aleksandur Stamboliiski 62 ', '0878 396 5765',
'georginikolaev@abv.bg');

INSERT INTO stockGroups (name)
VALUES ('Washing Mashines'),
            ('TVs'),
            ('Laptops'),
```

```
    ('Fridges');

INSERT INTO brands (name)
VALUES ('Miele'),
       ('Sony'),
       ('Apple'),
       ('Beko');

INSERT INTO models (name, brand_id, group_id)
VALUES ('A++', 1, 1),
       ('Bravia', 2, 2),
       ('MacBook',3,3),
       ('NoFrost',4,4);

INSERT INTO stock (groups_id, brand_id, model_id, quantity, price)
VALUES (1, 1, 1, 222, 900),
       (2, 2, 2, 333, 2000),
       (3, 3, 3, 555, 3000),
       (4, 4, 4, 444, 4000);

INSERT INTO orders (client_id, stock_id, order_quantity, dateOfOrder)
VALUES (1, 1, 10, '2023-05-07'),
       (2, 2, 11, '2023-11-14'),
       (3, 3, 12, '2023-04-05'),
       (4, 4, 5, '2023-07-05');

INSERT INTO discount (stock_id, min_quantity, discount_percentage)
VALUES (1, 10, 15),
       (2, 5, 10),
       (3, 15, 25),
       (4, 20, 35);

INSERT INTO payments (order_id, client_id, total, pay_Date)
SELECT orders.id, orders.client_id, stock.price * orders.order_quantity, CURDATE()
FROM orders
INNER JOIN stock ON orders.stock_id = stock.id;


INSERT INTO deliveries (client_id, stock_id, delivery_quantity,status, delivery_date)
VALUES (1, 1, 5,'Delivered', '2023-11-22'),
       (2, 2, 10,'At warehouse', '2023-12-23'),
       (3, 3, 21,'Delivering', '2023-01-23'),
       (4, 4, 21,'Delivering', '2023-01-23');
```

Задача 2. Напишете заявка, в която демонстрирате SELECT с ограничаващо условие по избор – Ще изведем информацията за клиентите с име : „Mitko Trapov”

```
/* 2 */
SELECT * from clients where name="Mitko Trapov";
```



id	name	address	phone	email
3	Mitko Trapov	ul. Banishora 25	08888 21 300	mitkotrapov@abv.bg
NULL	NULL	NULL	NULL	NULL

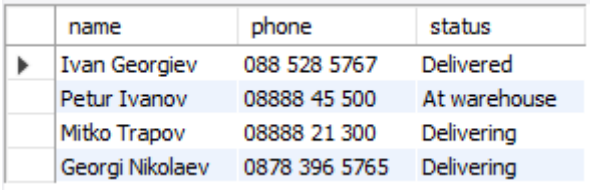
Задача 3. Напишете заявка, в която използвате агрегатна функция и GROUP BY по ваш избор – Ще изведем имената на клиентите и сумата на плащанията им.

```
/* 3 */
SELECT clients.name, CONCAT(SUM(payments.total), " BGN") AS total_payments
FROM clients
JOIN payments ON payments.client_id = clients.id
GROUP BY clients.name;
```

name	total_payments
Ivan Georgiev	9000 BGN
Petur Ivanov	22000 BGN
Mitko Trapov	36000 BGN
Georgi Nikolaev	20000 BGN

Задача 4. Напишете заявка, в която демонстрирате INNER JOIN по ваш избор – Тук ще изведем клиентите и техните телефонни номера както и статуса на поръчките им.

```
/* 4 */
SELECT c.name, c.phone, d.status  FROM clients as c
JOIN deliveries as d on d.client_id = c.id;
```



name	phone	status
Ivan Georgiev	088 528 5767	Delivered
Petur Ivanov	08888 45 500	At warehouse
Mitko Trapov	08888 21 300	Delivering
Georgi Nikolaev	0878 396 5765	Delivering

Задача 5. Напишете заявка, в която демонстрирате OUTER JOIN по ваш избор – Ще напишем заявка с която да изведем модела на дадена техника, както и нейния производител.

```
/* 5 */
SELECT m.name as model, b.name as brand FROM models as m
LEFT JOIN brands as b ON m.brand_id = b.id;
```

model	brand
A++	Miele
Bravia	Sony
MacBook	Apple
NoFrost	Beko

Задача 6. Напишете заявка, в която демонстрирате вложен SELECT по ваш избор – Ще създадем заявка, с която ще изведем модела на техниките и съответно какви са.

```
/* 6 */
SELECT m.name as model_name, sp.name as stockGroup FROM models as m
JOIN stockGroups as sp ON m.id IN(
select group_id FROM models as m WHERE m.group_id = sp.id);
```

stockGroup	model_name
Washing Mashines	A++
TVs	Bravia
Laptops	MacBook
Fridges	NoFrost

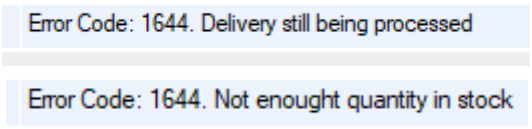
Задача 7. Напишете заявка, в която демонстрирате едновременно JOIN и агрегатна функция – Ще напишем заявка с която да изведем id на клиента и средната стойност на плащанията му.

```
/* 7 */
SELECT p.client_id, AVG(p.total) AS average_payment
FROM
  (SELECT client_id, SUM(total) AS total
   FROM payments
   GROUP BY client_id) AS p
   GROUP BY p.client_id;
```

client_id	average_payment
1	9000
2	22000
3	36000
4	20000

Задача 8. Създайте тригер по ваш избор – Ще създадем два тригера. Единият ще хвърля съобщение за грешка при опит за изтриване на поръчката, ако тя не е доставена все още. Вторият ще хвърля грешка ако количеството на поръчаните стоки не е налично в склада, тоест ако искаме 10 телевизора, а имаме само 5 налични, ще получим грешка.

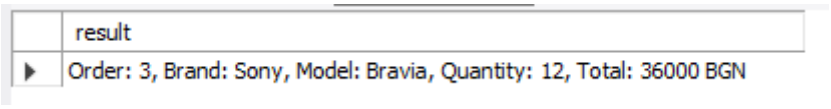
```
DROP TRIGGER if exists fin_del;
delimiter |
CREATE TRIGGER fin_del BEFORE DELETE ON deliveries
FOR EACH ROW
BEGIN
IF(OLD.status != 'Delivered')
THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT =  'Delivery still being
processed';
end if;
END;
|
delimiter ;
DROP TRIGGER if exists notenought_qty;
delimiter |
CREATE TRIGGER notenought_qty BEFORE INSERT ON orders
FOR EACH ROW
BEGIN
DECLARE qty_instock INT;
SET qty_instock = (SELECT stock.quantity from stock where id = NEW.stock_id);
IF(NEW.order_quantity>qty_instock)
THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT =  'Not enought quantity in
stock';
end if;
END;
|
```



Задача 9. Създайте процедура, в която демонстрирате използване на курсор – Ще създадем процедура, която ще извежда поръчките на клиента, който е подаден като входен параметър.

```
DROP PROCEDURE IF EXISTS client_orders_select;
DELIMITER |
CREATE PROCEDURE client_orders_select(IN client_name VARCHAR(255))
BEGIN
DECLARE total_due DOUBLE;
DECLARE order_id, client_id, stock_id, order_quantity, model_id, brand_id INT;
DECLARE model_name, brand_name VARCHAR(255);
DECLARE done INT DEFAULT FALSE;
```

```
DECLARE order_cursor CURSOR FOR
SELECT o.id, o.client_id, o.stock_id, o.order_quantity, m.id, b.id
FROM orders o
INNER JOIN stock s ON o.stock_id = s.id
INNER JOIN models m ON s.model_id = m.id
INNER JOIN brands b ON m.brand_id = b.id
INNER JOIN clients c ON o.client_id = c.id
WHERE c.name = client_name;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN order_cursor;

SET total_due = 0;
SELECT concat('Orders for client: ', client_name) as result;

FETCH order_cursor INTO order_id, client_id, stock_id, order_quantity, model_id, brand_id;
WHILE NOT done DO
SELECT m.name INTO model_name FROM models m WHERE m.id = model_id;
SELECT b.name INTO brand_name FROM brands b WHERE b.id = brand_id;
SELECT (s.price * o.order_quantity) INTO total_due FROM orders o INNER JOIN stock s
ON o.stock_id = s.id WHERE o.id = order_id;
SELECT concat('Order: ', order_id, ', Brand: ', brand_name, ', Model: ', model_name, ',
Quantity: ', order_quantity, ', Total: ', total_due, ' BGN') as result
GROUP BY order_id;
SET total_due = 0;
FETCH order_cursor INTO order_id, client_id, stock_id, order_quantity, model_id, brand_id;
END WHILE;

CLOSE order_cursor;

END |
DELIMITER ;

CALL client_orders_select('Mitko Trapov')
```



	result
▶	Order: 3, Brand: Sony, Model: Bravia, Quantity: 12, Total: 36000 BGN