# ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ

## Катедра „Компютърни системи”

**КУРСОВ ПРОЕКТ**

**ПО БАЗИ ОТ ДАННИ**

**Студент: Иван Иванов Стоилов**

**Фак. №: 121221122** **Група: 40**

| **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. |

1. Да се проектира база от данни и да се представи ER диаграма със съответни CREATE TABLE заявки за средата MySQL.
2. Напишете заявка, в която демонстрирате SELECT с ограничаващо условие по избор.
3. Напишете заявка, в която използвате агрегатна функция и GROUP BY по ваш избор.
4. Напишете заявка, в която демонстрирате INNER JOIN по ваш избор.
5. Напишете заявка, в която демонстрирате OUTER JOIN по ваш избор.
6. Напишете заявка, в която демонстрирате вложен SELECT по ваш избор.
7. Напишете заявка, в която демонстрирате едновременно JOIN и агрегатна функция.
8. Създайте тригер по ваш избор.
9. Създайте процедура, в която демонстрирате използване на курсор.

***Вашата работа трябва да включва: задание, ER-диаграма, CREATE TABLE заявки, всички останали заявки, решения на задачите от 2 до 9 и резултатите от тях.***

**Задача 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, Brands, Models и Orders. Допълнително ще създадем още няколко таблици, необходими за завършване на базата от данни – Payments, Deliveries, Discount, Stock и Clients.

Първата ще отразява съответната група, към която може да се причисли дадена стока. В нея ще се съдържа информация за името на съответната група (Електроника, Домакински електроуреди и др.).

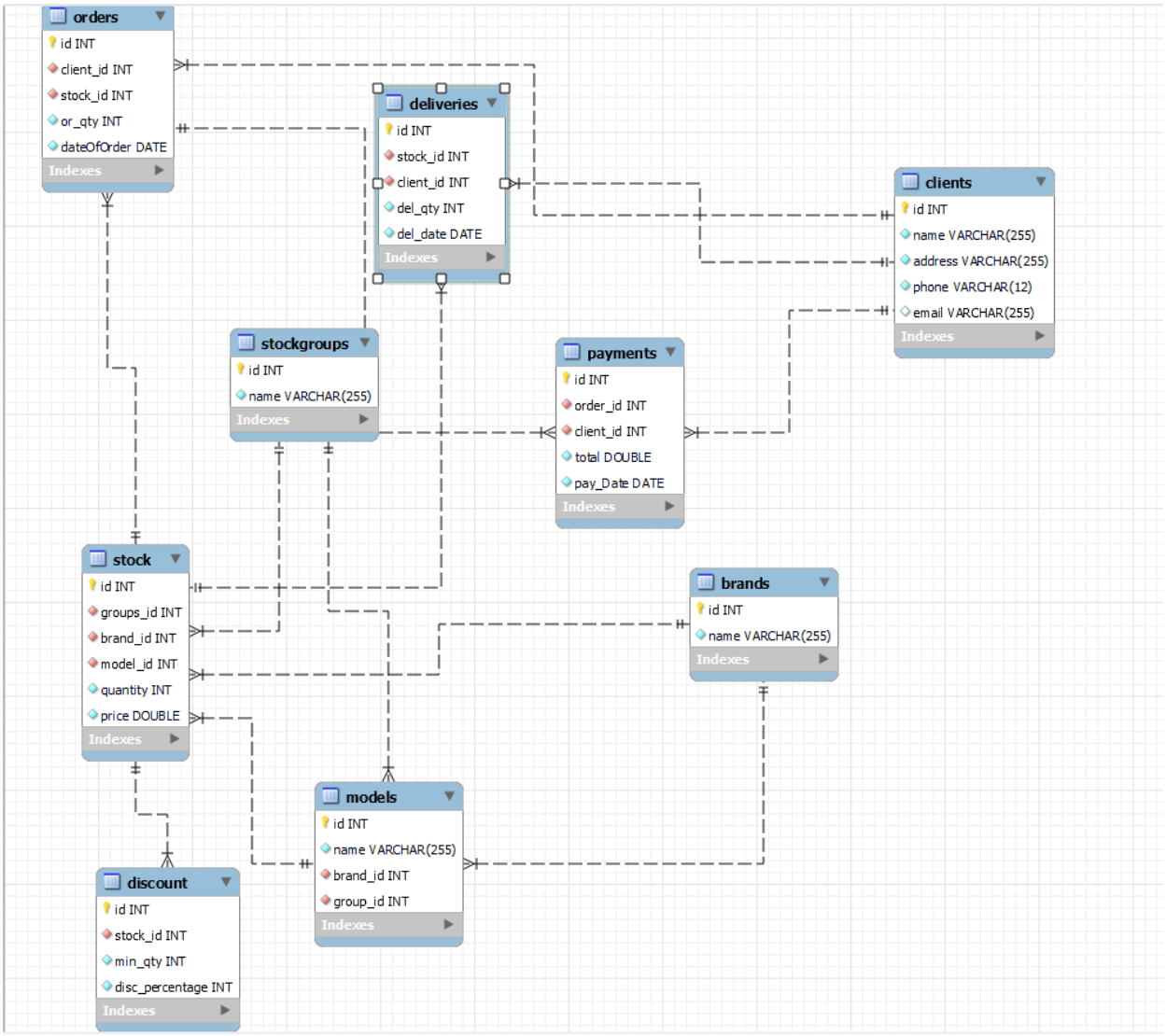
Във втората таблица ще съхраняваме името на съответните марки (Samsung, Bosch, AEG, Liebherr и т.н.). В таблицата Models ще съхраняваме името на модела, като той ще бъде свързан със съответната марка и група към които принадлежи.

В следващата таблица, която ще създадем - Orders, ще съхраняваме поръчаното количество от дадена стока и датата на поръчката, като всичко това ще бъде свързано и със съответния клиент и със таблицата за наличност на стоките.

Ще създадем и допълнителните таблици, споменати и засегнати по-горе, които са както следва:

- Payments – съдържа информация за дължимата сума, датата на плащане, както и номера на поръчката и клиента.
- Deliveries – съдържа информация за доставеното количество стока, датата на доставка, както и номера на поръчката и клиента.
- Discount – съдържа информация за процента отстъпка, който клиентите получават при поръчано по-голямо количество от дадена стока, минималното количество стока, която трябва да поръчат за да получат отстъпка, както и номера на съответната стока.
- Stock – съдържа информация за наличното количество от дадена стока, цената на съответната стока, както и номерата на групата към която принадлежи, марката и модела на стоката.
- Clients – съдържа информация за клиентите, които поръчват стоки от склада – име на фирмата, адрес за доставка, телефон и e-mail за контакт.

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 warehouse;
CREATE DATABASE warehouse;
USE warehouse;

CREATE TABLE clients(
  id int not null auto_increment PRIMARY KEY,
  name varchar(255) not null,
  address varchar(255) not null,
  phone varchar(12) not null,
  email varchar(255)
  );

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

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

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,
  FOREIGN KEY(brand_id) references brands(id),
  FOREIGN KEY(group_id) references stockGroups(id)
);

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,
  FOREIGN KEY (groups_id) REFERENCES stockGroups(id),
  FOREIGN KEY (brand_id) REFERENCES brands(id),
  FOREIGN KEY (model_id) REFERENCES models(id)
);

CREATE TABLE orders(
  id int not null auto_increment PRIMARY KEY,
  client_id int not null,
  stock_id int not null,
  or_qty int not null,
  dateOfOrder DATE not null,
  FOREIGN KEY (client_id) REFERENCES clients(id),
  FOREIGN KEY (stock_id) references stock(id)
);

CREATE TABLE discount(
  id int not null auto_increment PRIMARY KEY,
  stock_id int not null,
  min_qty int not null,
  disc_percentage int not null,
  FOREIGN KEY (stock_id) references stock(id)
);

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,
  FOREIGN KEY (client_id) REFERENCES clients(id),
  FOREIGN KEY (order_id) references orders(id)
);
```

```
CREATE TABLE deliveries(
  id int not null auto_increment PRIMARY KEY,
  stock_id int not null,
  client_id int not null,
  del_qty int not null,
  del_date DATE not null,
  FOREIGN KEY (client_id) REFERENCES clients(id),
  FOREIGN KEY (stock_id) references stock(id)
);
```

Добавяме и тестови данни в таблиците:

```
INSERT INTO clients (name, address, phone, email)
VALUES ('Technopolis', 'ul. Okolovrasten pat 265', '02 921 1111',
'orders@technopolis.bg'),
            ('TechMart', 'bul. Simeonovsko Shose 138', '02 405 1379',
'orders@techmart.bg'),
        ('Zora', 'ul. Profesor Stoyan Kirkovich 13', '02 887 9495',
'orders@zora.bg');

INSERT INTO stockGroups (name)
VALUES ('Electronics'),
            ('Home Appliances');

INSERT INTO brands (name)
VALUES ('Samsung'),
            ('Bosch');

INSERT INTO models (name, brand_id, group_id)
VALUES ('Galaxy S20', 1, 1),
            ('KGN36NLEA', 2, 2);

INSERT INTO stock (groups_id, brand_id, model_id, quantity, price)
VALUES (1, 1, 1, 100, 799.99),
            (2, 2, 2, 50, 1099.99);

INSERT INTO orders (client_id, stock_id, or_qty, dateOfOrder)
VALUES (1, 1, 2, '2022-01-01'),
            (2, 2, 5, '2022-01-20'),
            (3, 2, 7, '2022-01-20');

INSERT INTO discount (stock_id, min_qty, disc_percentage)
VALUES (1, 10, 10),
            (2, 5, 10);

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

INSERT INTO deliveries (client_id, stock_id, del_qty, del_date)
VALUES (1, 1, 1, '2022-01-10'),
            (2, 2, 1, '2022-01-20');
```

**Задача 2.** Напишете заявка, в която демонстрирате SELECT с ограничаващо условие по избор – заявката ще бъде комбинация от **Задача 2** и **Задача 7**. Тя ще покаже всички заявени от клиенти продукти през м.Януари 2022г. с тяхната марка и модел, както и количество на продукта, дата на заявката, дължима сума и информация за клиента.

```
SELECT orders.id as order_ID, clients.name as client_name, clients.address
as clientAddress, clients.phone as clientPhone,
stockgroups.name as StockGroup, CONCAT(brands.name, " ", models.name) as
ProductName, orders.or_qty as Qty,
orders.dateOfOrder, CONCAT(payments.total, " BGN") as total
FROM stock
JOIN orders ON stock.id=orders.stock_id
JOIN clients ON orders.client_id=clients.id
JOIN brands ON stock.brand_id=brands.id
JOIN models ON stock.model_id=models.id
JOIN stockgroups ON stock.groups_id=stockgroups.id
JOIN payments ON orders.id=payments.order_id
WHERE dateOfOrder BETWEEN '2022-01-01' AND '2022-01-31'
```

Резултатът след изпълнението на тази заявка е:

| order_ID | client_name | clientAddress | clientPhone | StockGroup | ProductName | Qty | dateOfOrder | total |
|---|---|---|---|---|---|---|---|---|
| 1 | Technopolis | ul. Okolovrasten pat 265 | 02 921 1111 | Electronics | Samsung Galaxy S20 | 2 | 2022-01-01 | 1599.98 BGN |
| 2 | TechMart | bul. Simeonovsko Shose 138 | 02 405 1379 | Home Appliances | Bosch KGN36NLEA | 5 | 2022-01-20 | 5499.95 BGN |
| 3 | Zora | ul. Profesor Stoyan Kirkovich 13 | 02 887 9495 | Home Appliances | Bosch KGN36NLEA | 7 | 2022-01-20 | 7699.93 BGN |

**Задача 3.** Напишете заявка, в която използвате агрегатна функция и GROUP BY по ваш избор – показва дължимата сума от всеки клиент, заявил стока във склада.

```
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;
```

Резултатът от изпълнението на заявката е:

| name | total_payments |
|---|---|
| Technopolis | 1599.98 BGN |
| TechMart | 5499.95 BGN |
| Zora | 7699.93 BGN |

**Задача 4.** Напишете заявка, в която демонстрирате INNER JOIN по ваш избор - връща информация за датата на поръчката, артикула, който е поръчан, и магазина, който я е поръчал.

```
SELECT orders.id, clients.name, brands.name, models.name,
orders.dateOfOrder
FROM clients
INNER JOIN orders ON clients.id = orders.client_id
INNER JOIN stock ON orders.stock_id = stock.id
INNER JOIN models ON stock.model_id = models.id
INNER JOIN brands ON stock.brand_id = brands.id;
```

Резултатът от изпълнението на заявката е:

| id | name | name | name | dateOfOrder |
|---|---|---|---|---|
| 1 | Technopolis | Samsung | Galaxy S20 | 2022-01-01 |
| 2 | TechMart | Bosch | KGN36NLEA | 2022-01-20 |
| 3 | Zora | Bosch | KGN36NLEA | 2022-01-20 |

**Задача 5.** Напишете заявка, в която демонстрирате OUTER JOIN по ваш избор – заявката ще ни даде списък на всички клиенти в базата данни, които имат поръчки, включително тези, които не са направили никакви поръчки, както и наличностите на заявената стока. Ако няма поръчки за дадения клиент, ще бъде изведена стойност null за колоната "quantity".

```
SELECT clients.name, stock.id, stock.quantity
FROM clients
LEFT OUTER JOIN orders ON clients.id = orders.client_id
LEFT OUTER JOIN stock ON orders.stock_id = stock.id;
```

Резултатът от изпълнението на заявката е:

| name | id | quantity |
|---|---|---|
| Technopolis | 1 | 100 |
| TechMart | 2 | 50 |
| Zora | 2 | 50 |

**Задача 6.** Напишете заявка, в която демонстрирате вложен SELECT по ваш избор - ще изведе клиентско ID и средна стойност на плащанията за всеки клиент.

```
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;
```

Резултатът от изпълнението на заявката е:

| client_id | average_payment |
|---|---|
| 1 | 1599.98 |
| 2 | 5499.95 |
| 3 | 7699.93 |

**Задача 7.** Напишете заявка, в която демонстрирате едновременно JOIN и агрегатна функция – както споменахме по-горе, тази задача е комбинирана със **задача 2** поради появилата се идея за заявка във тази задача и последвалата необходимост да се демонстрира едновременното използване на SELECT, JOIN и агрегатна функция.

**Задача 8.** Създайте тригер по ваш избор - за всеки нов запис в "orders", тригерът ще обнови количеството на продукта в таблицата "stock" чрез изваждане на количеството, посочено в новия запис от "orders".

```
DELIMITER |
DROP TRIGGER IF EXISTS update_stock_on_order;
CREATE TRIGGER update_stock_on_order AFTER INSERT ON orders
FOR EACH ROW
BEGIN
    UPDATE stock
    SET quantity = quantity - NEW.or_qty
    WHERE id = NEW.stock_id;
END;
| DELIMITER ;
```

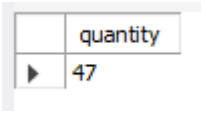
За да тестваме тригера пишем една допълнителна INSERT заявка, която ще промени наличността на артикул с номер 2.

```
INSERT INTO orders (client_id, stock_id, or_qty, dateOfOrder)
VALUES (1, 2, 3, '2022-02-01');
```

От тестовите данни се вижда, че въведената наличност е 50бр. След изпълнението на тригера и заявката, за проверка пишем проста SELECT заявка:

```
SELECT quantity FROM stock WHERE id = 2;
```

Резултатът от нея е:



| | quantity |
|---|---|
| ▶ | 47 |

**Задача 9.** Създайте процедура, в която демонстрирате използване на курсор . Тази процедура приема име на клиент като входен параметър и използва курсор за да изведе информацията за всички поръчки на клиента заедно с информацията за моделите и марките на стоките, които са били поръчани.

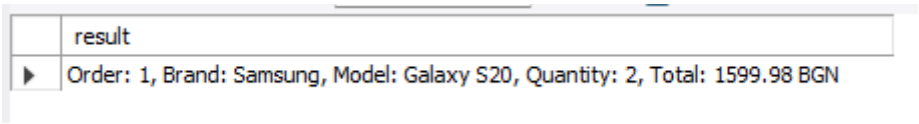
```
DELIMITER |
DROP PROCEDURE IF EXISTS show_client_orders;
CREATE PROCEDURE show_client_orders(IN client_name VARCHAR(255))
BEGIN
DECLARE total_due DOUBLE;
DECLARE order_id, client_id, stock_id, or_qty, model_id, brand_id INT;
DECLARE model_name, brand_name VARCHAR(255);
DECLARE done INT DEFAULT FALSE;
DECLARE order_cursor CURSOR FOR
SELECT o.id, o.client_id, o.stock_id, o.or_qty, 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, or_qty, 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.or_qty) 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: ', or_qty, ', Total: ', total_due, ' BGN') as
result
GROUP BY order_id;
SET total_due = 0;
FETCH order_cursor INTO order_id, client_id, stock_id, or_qty, model_id,
brand_id;
END WHILE;

CLOSE order_cursor;

END |
DELIMITER ;
```

За да използваме процедурата, трябва да я извикаме по следния начин:

```
CALL show_client_orders('Technopolis');
```

Резултатът от изпълнението на процедурата е:



| | result |
|---|---|
| ▶ | Order: 1, Brand: Samsung, Model: Galaxy S20, Quantity: 2, Total: 1599.98 BGN |

Показва номера на клиента, марката и модела на стоката, поръчаното количество и дължимата сума от клиента.