 ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ

**Катедра „Компютърни системи”**

**КУРСОВ ПРОЕКТ ПО БАЗИ ОТ ДАННИ**

# Студент: Васил Боянов Петров

**ФАК. № 121221084 Група: 43**

**Картина, която съдържа текст

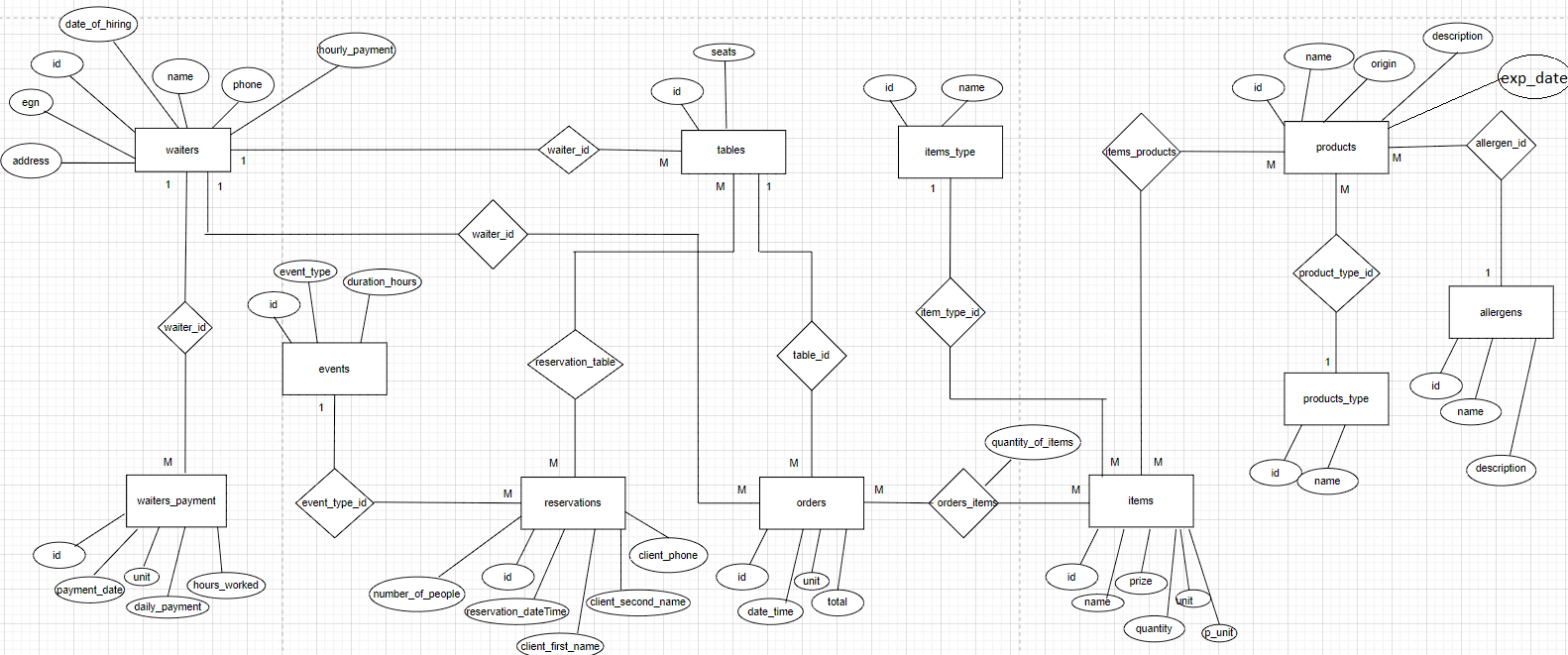
Описанието е генерирано автоматично**

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.

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

Основните обекти, за които ще съхраняваме информация са следните: Products, Items, Allergens, Orders, Waiters, Tables, Reservations, Events. Също така ще създадем и още няколко допълнителни таблици. Една от тях ще се казва Waiters\_payment, в която ще пазим всичко свързано с плащанията на сервитьорите. Други две таблици са: Items\_type и Products\_type, тук ще съхраняваме типа на нашите, съответно артикули и продукти. Останалите допълнителни таблици са следните:

Reservation\_table, Items\_products и Orders\_items. Те ни служат за представянето на връзките (M:M) между отделните основни таблици, като структурата им е идентична, с изключение на Orders\_items, в която ще съхраняваме и количеството на съответния продукт в текущата поръчка (quantity\_of\_item). За проектирането на базата ще използваме модела ER-диаграма (Entity Relationship Diagram):



Заявките, с които създаваме базата данни и таблиците са:

CREATE DATABASE restaurant;

CREATE TABLE products\_type(

id INT AUTO\_INCREMENT PRIMARY KEY,

`name` VARCHAR(55) NOT NULL

);

CREATE TABLE allergens(

id INT AUTO\_INCREMENT PRIMARY KEY,

`name` VARCHAR(55) NOT NULL,

`description` TEXT NULL DEFAULT NULL

);

CREATE TABLE products(

id INT AUTO\_INCREMENT PRIMARY KEY,

`name` VARCHAR(55) NOT NULL,

`description` TEXT NULL DEFAULT NULL,

origin VARCHAR(60) NULL DEFAULT NULL,

UNIQUE(`name`, origin),

product\_type\_id INT NOT NULL,

CONSTRAINT FOREIGN KEY (product\_type\_id) REFERENCES products\_type(id) ON DELETE CASCADE ON UPDATE CASCADE,

allergen\_id INT NULL DEFAULT NULL,

CONSTRAINT FOREIGN KEY (allergen\_id) REFERENCES allergens(id) ON UPDATE CASCADE

);

CREATE TABLE items\_type(

id INT AUTO\_INCREMENT PRIMARY KEY,

`name` VARCHAR(55) NOT NULL

);

CREATE TABLE items(

id INT AUTO\_INCREMENT PRIMARY KEY,

`name` VARCHAR(70) NOT NULL UNIQUE,

quantity INT NOT NULL,

unit ENUM('g', 'ml') NOT NULL,

price DOUBLE NOT NULL CONSTRAINT PositivePrice CHECK (price > 0),

p\_unit VARCHAR(3) DEFAULT 'BGN' NOT NULL,

item\_type\_id INT NOT NULL,

CONSTRAINT FOREIGN KEY (item\_type\_id) REFERENCES items\_type(id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE waiters(

id INT AUTO\_INCREMENT PRIMARY KEY,

`name` VARCHAR(255) NOT NULL,

egn CHAR(10) NOT NULL UNIQUE CONSTRAINT EGN CHECK(CHAR\_LENGTH(egn) = 10),

phone VARCHAR(20) NULL DEFAULT NULL,

address VARCHAR(255) NULL DEFAULT NULL,

date\_of\_hiring DATE NOT NULL,

hourly\_payment DOUBLE NOT NULL CONSTRAINT HPMoreThan4 CHECK(hourly\_payment > 4),

unit VARCHAR(3) DEFAULT 'BGN' NOT NULL

);

CREATE TABLE waiters\_payment(

id INT AUTO\_INCREMENT PRIMARY KEY,

payment\_date DATE NOT NULL,

hours\_worked TINYINT NOT NULL CONSTRAINT PositiveHoursWorked CHECK(hours\_worked > 0),

daily\_payment DOUBLE NOT NULL CONSTRAINT PositiveDailyPayment CHECK(daily\_payment >0),

unit VARCHAR(3) DEFAULT 'BGN' NOT NULL,

waiter\_id INT NOT NULL,

CONSTRAINT FOREIGN KEY (waiter\_id) REFERENCES waiters(id) ON DELETE CASCADE ON UPDATE CASCADE,

UNIQUE KEY (payment\_date, waiter\_id)

);

CREATE TABLE `tables`(

id INT AUTO\_INCREMENT PRIMARY KEY,

seats TINYINT NOT NULL,

waiter\_id INT NULL DEFAULT NULL,

CONSTRAINT FOREIGN KEY (waiter\_id) REFERENCES waiters(id) ON DELETE SET NULL ON UPDATE CASCADE

);

CREATE TABLE orders(

id INT AUTO\_INCREMENT PRIMARY KEY,

date\_time DATETIME NOT NULL,

total DOUBLE NOT NULL,

unit VARCHAR(3) DEFAULT 'BGN' NOT NULL,

table\_id INT NOT NULL,

CONSTRAINT FOREIGN KEY (table\_id) REFERENCES `tables`(id) ON DELETE CASCADE ON UPDATE CASCADE,

waiter\_id INT NOT NULL,

CONSTRAINT FOREIGN KEY (waiter\_id) REFERENCES waiters(id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE items\_products(

item\_id INT NOT NULL,

CONSTRAINT FOREIGN KEY (item\_id) REFERENCES items(id) ON DELETE CASCADE ON UPDATE CASCADE,

product\_id INT NOT NULL,

CONSTRAINT FOREIGN KEY (product\_id) REFERENCES products(id) ON DELETE CASCADE ON UPDATE CASCADE,

UNIQUE(item\_id, product\_id)

);

CREATE TABLE orders\_items(

order\_id INT NOT NULL,

CONSTRAINT FOREIGN KEY (order\_id) REFERENCES orders(id) ON DELETE CASCADE ON UPDATE CASCADE,

item\_id INT NOT NULL,

CONSTRAINT FOREIGN KEY (item\_id) REFERENCES items(id) ON DELETE CASCADE ON UPDATE CASCADE,

quantity\_of\_item TINYINT NOT NULL DEFAULT 1,

UNIQUE(order\_id, item\_id)

);

CREATE TABLE `events`(

id INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,

event\_type ENUM("Рожден ден", "Бал", "Юбилей", "Вечеря", "Обяд", "Закуска", "Сватба",

"Друго събитие") NOT NULL UNIQUE,

duration\_hours INT NOT NULL

);

CREATE TABLE reservations(

id INT AUTO\_INCREMENT PRIMARY KEY,

reservation\_dateTime DATETIME NOT NULL CONSTRAINT CHECK (TIME(reservation\_dateTime) > '08:00:00' OR TIME(reservation\_dateTime) < '23:00:00'),

client\_first\_name VARCHAR(60) NOT NULL,

client\_second\_name VARCHAR(60) NOT NULL,

client\_phone VARCHAR(15) NOT NULL,

number\_of\_people TINYINT NOT NULL,

event\_type\_id INT NOT NULL,

CONSTRAINT FOREIGN KEY (event\_type\_id) REFERENCES `events`(id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE reservation\_table(

reservation\_id INT NOT NULL,

CONSTRAINT FOREIGN KEY (reservation\_id) REFERENCES reservations(id) ON DELETE CASCADE ON UPDATE CASCADE,

table\_id INT NOT NULL,

CONSTRAINT FOREIGN KEY (table\_id) REFERENCES `tables`(id) ON DELETE CASCADE ON UPDATE CASCADE,

UNIQUE KEY (reservation\_id, table\_id)

);

Добавяме и тестови данни в таблиците:

INSERT INTO products\_type(`name`)

VALUES

('Зеленчуци'),

('Плодове'),

('Месо'),

('Морска храна'),

('Яйца'),

('Млечни продукти'),

('Подправки'),

('Ядки'),

('Зърнени култури');

INSERT INTO allergens (`name`, `description`) VALUES

('Глутен', 'Протеин, съдържащ се в пшеницата, ечемика и ръжта'),

('Соя', 'Бобово растение, често използвано като съставка в много преработени храни'),

('Фъстъци', 'Вид ядка, която може да предизвика тежки алергични реакции при някои хора'),

('Дървесни ядки', 'Различни видове ядки, като бадеми, кашу и орехи'),

('Лактоза', 'Често срещана съставка в много млечни продукти, включително сирене, кисело мляко и сладолед'),

('Яйца', 'Често срещана съставка в много печени изделия, включително торти и бисквити'),

('Морски дарове', 'Различни видове морски дарове, като скариди, раци и омари'),

('Риба', 'Често срещана съставка в много видове кухня'),

('Сусам', 'Семе, което обикновено се използва като съставка в много видове хляб и

сладкиши'),

('Пшеница', 'Универсална съставка, която се използва в голямо разнообразие от храни, включително хляб, макаронени изделия, зърнени храни и печени изделия'),

('Сулфити', 'Вид консервант, който често се използва в много видове преработени храни');

INSERT INTO products (`name`, `description`, origin, product\_type\_id, allergen\_id) VALUES

('Моркови', 'Кореноплоден зеленчук, който често се консумира суров или сготвен', 'Европа', 1, NULL),

('Домати', 'Плод, който често се използва като зеленчук в кулинарията', 'Европа', 1, NULL),

('Чушки', 'Група зеленчуци с разнообразни цветове и вкусове', 'Европа', 1, NULL),

('Лук', 'Луковичен зеленчук, който често се използва като ароматизатор в кулинарията', 'Европа', 1, NULL),

('Картофи', 'Кореноплоден зеленчук с нишесте, който често се вари, пече или пържи', 'Европа', 1, NULL),

('Краставици', 'Вид тиква, която често се консумира сурова или маринована', 'Азия', 1, NULL), …

INSERT INTO items\_type(`name`) VALUES

('Тестени изделия'),

('Предястия'),

('Основни ястия'),

('Десерти'),

('Алкохолни напитки'),

('Безалкохолни напитки');

INSERT INTO items(`name`, quantity, unit ,prize, item\_type\_id) VALUES

('Филия хляб', 70, ‘g’, 0.15, 1),

('Питка със сирене', 110, ‘g’, 1.50, 1),

('Пърленка', 90, ‘g’, 1.10, 1),

('Шопска салата', 300, ‘g’, 5.50, 2),

('Гръцка салата', 350, ‘g’, 7.00, 2),

('Цезар салата', 250, ‘g’, 6.00, 2),

('Млечна салата', 100, ‘g’, 2.50, 2),

('Таратор', 150, ‘g’, 2.00, 2),

('Пълнена чушка със сирене', 150, ‘g’, 3.00, 2),

('Пататник', 200, ‘g’, 4.50, 2),

('Пилешки хапки', 150, ‘g’, 7.00, 2),

…

INSERT INTO items\_products VALUES

(1, 48),

(2, 48),

(2, 27),(3, 48),(4, 6),(4, 2),(4, 3),(4, 27),(4, 32),(4, 34),(5, 2),(5, 6),

(5, 3),(5, 4),(5, 34),(5, 49),(5, 32),(6, 7),(6, 1),(6, 27),(6, 30),(6, 49),

(7, 6),(7, 49),(7, 24),(7, 34),(8, 6),(8, 49),(8, 24),(9, 3),(9, 27),(10, 5),

(10, 4),(10, 35),(10, 30),(10, 48),(10, 34),(10, 32),(10, 49),(10, 37),(11, 15),(11, 48),

…

INSERT INTO waiters (`name`, egn, phone, address, date\_of\_hiring, hourly\_payment) VALUES

('Мария Петрова', '0243064568', '0888193275', 'бул. България 25', '2020-03-10', 8.20),

('Димитър Георгиев', '0343206467', NULL, 'ул. Христо Ботев 12', '2019-08-05', 13.00),

('София Димитрова', '9903148563', '0891849302', 'ул. Георги Бенковски 2', '2022-02-28', 7.00),

('Николай Лазаров', '9810294577', '0892927321', NULL, '2021-06-10', 7.80),

('Гергана Стоянова', '0145144567', NULL, 'бул. Цариградско шосе 55', '2020-01-01', 9.00),

('Станислава Маринова', '9805134547', '0894820472', 'ул. Любен Каравелов 8', '2021-12-01', 8.50);

('Иван Иванов', '9503028563', '0888123456', 'ул. България 15, София', '2020-05-01', 10.50);

('Петър Петров', '9901267504', '0899654321', 'бул. Цар Борис III 25, Варна', '2019-01-15', 9.00);

('Мария Георгиева', '9812064567', NULL, 'ул. Гео Милев 18, Пловдив', '2021-09-01', 11.00);

INSERT INTO tables (seats, waiter\_id) VALUES

(2, NULL),

(2, NULL),

(4, NULL),

(8, 5),

(4, NULL),

(4, NULL),

(6, NULL),

(10, 8),

(10, 2);

INSERT INTO orders(date\_time, table\_id, waiter\_id) VALUES

('2021-09-01', 1, 2),

('2021-09-04', 2, 7),

('2021-09-06', 4, 6),

('2021-09-09', 7, 5),

('2021-09-11', 6, 9),

('2021-09-13', 3, 3);

INSERT INTO orders\_items VALUES

(1, 1, 6),(1, 4, 1),(1, 5, 1),(1, 13, 1),(1, 20, 1),(1, 27, 2),\

(2, 3, 2),(2, 2, 2),(2, 6, 2),(2, 5, 1),(2, 4, 1),(2, 12, 1),(2, 22, 1), (2, 17, 1),(2, 16, 1),(2, 29, 4),

(3, 1, 10),(3, 7, 2),(3, 8, 2),(3, 14, 10),(3, 15, 10),(3, 23, 2),(3, 24, 2),(3, 36, 2),(3, 30, 2),

(4, 1, 25),(4, 9, 5),(4, 11, 5),(4, 21, 4),(4, 18, 3),(4, 22, 3),(4, 26, 5),(4, 27, 8),(4, 28, 2),(5, 3, 6),(5, 2, 6),(5, 4, 3),(5, 7, 3),(5, 12, 2),(5, 13, 2),(5, 19, 2),(5, 26, 1),(5, 34, 1)(6, 1, 16),(6, 8, 3),(6, 9, 3),(6, 10, 2),(6, 18, 4),(6, 20, 2),(6, 17, 2),(6, 23, 3),(6, 33, 3),(6, 30, 3),(6, 28, 2);

INSERT INTO waiters\_payment(payment\_date, hours\_worked, waiter\_id) VALUES

('2022-01-10', 8, 1),

('2022-01-12', 7, 1),

('2022-01-14', 6, 1),

('2022-01-16', 5, 1),

('2022-01-23', 8, 1),

('2022-02-14', 9, 1),

('2022-02-17', 8, 1),

('2022-02-19', 8, 1),

('2022-02-25', 7, 1),

('2022-01-10', 8, 2),

('2022-01-12', 7, 2),

('2022-01-14', 6, 2),

('2022-01-16', 5, 2),

('2022-01-23', 8, 2),

('2022-02-14', 9, 2),

('2022-02-17', 8, 2),

('2022-02-19', 8, 2),

('2022-02-25', 7, 2),

…

INSERT INTO reservations(reservation\_dateTime, client\_first\_name, client\_second\_name, client\_phone, number\_of\_people, event\_type\_id) VALUES

('2023-04-06 12:00:00', 'Иван', 'Иванов', '0888643255', 2, 2),

('2023-04-06 12:00:00', 'Петър', 'Петров', '0897320121', 4, 2),

('2023-04-06 18:00:00', 'Мария', 'Иванова', '0876923757', 8, 3),

('2023-04-06 18:00:00', 'Георги', 'Георгиев', '0896128462', 15, 3),

('2023-04-06 20:00:00', 'Анна', 'Петрова', '0876923786', 20, 4),

('2023-04-06 16:00:00', 'Васил', 'Петров', '0876823532', 25, 3);

INSERT INTO `events` VALUES

(NULL, "Закуска", 2),

(NULL, "Обяд", 3),

(NULL, "Вечеря", 3),

(NULL, "Рожден ден", 5),

(NULL, "Бал", 6),

(NULL, "Юбилей", 7),

(NULL, "Сватба", 8),

(NULL, "Друго събитие", 4);

**Задача 2. Напишете заявка, в която демонстрирате SELECT с  
ограничаващо условие по избор –** ще изведем информация за всички продукти, на които произхода им е от Европа:

SELECT \* FROM products

WHERE origin = 'Европа';

Картина, която съдържа текст

Описанието е генерирано автоматично

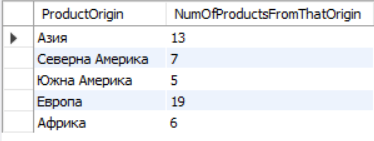
**Задача 3. Напишете заявка, в която използвате агрегатна функция и GROUP  
BY по ваш избор -**  ще изведем произходите на продуктите и тяхната съответна бройка:

SELECT origin AS ProductOrigin,

COUNT(\*) AS NumOfProductsFromThatOrigin

FROM products

GROUP BY origin;



**Задача 4. Напишете заявка, в която демонстрирате INNER JOIN по ваш  
избор –** заявката ще извежда информация за всеки един артикул и неговия тип, за целта ще обединим информация от таблиците items и items\_type:

SELECT items.name AS Item,

items.quantity AS QuantityOfItem,

items.unit AS UnitOfItem,

items.prize AS PrizeOfItem,

items.p\_unit AS PrizeUnitOfItem,

items\_type.name AS TypeOfItem

FROM items JOIN items\_type

ON items.item\_type\_id = items\_type.id;

Картина, която съдържа маса

Описанието е генерирано автоматично

**Задача 5. Напишете заявка, в която демонстрирате OUTER JOIN по ваш  
избор -** заявката ще извежда информацията от таблицата tables, като също така ще се изведе и името на сервитьора, който има персонална задача да обслужва тази маса, за целта ще използваме LEFT OUTER JOIN, като ще се покаже информацията и за масите, които нямат строго опреден сервитьор за тях:

SELECT `tables`.id AS TableNumber,

waiters.name AS WaiterName

FROM `tables` LEFT JOIN waiters

ON `tables`.waiter\_id = waiters.id;

Картина, която съдържа маса

Описанието е генерирано автоматично

**Задача 6. Напишете заявка, в която демонстрирате вложен SELECT по ваш  
избор -** със следващата заявка ще изведем информация за поръчките и съответните артикули, които са включени в тях. За целта ще използваме таблиците orders и items, както и свързващата ги таблица orders\_items:

SELECT orders\_items.order\_id AS OrderNumber,

items.`name` AS ItemInTheOrder,

orders\_items.quantity\_of\_item AS QuantityOfItem

FROM orders\_items JOIN items

ON orders\_items.order\_id IN (

SELECT id

FROM orders

)

AND orders\_items.item\_id = items.id;

Картина, която съдържа маса

Описанието е генерирано автоматично

**Задача 7. Напишете заявка, в която демонстрирате едновременно JOIN и  
агрегатна функция -** ще изведем информация за всеки един сервитьор и неговото

месечно възнаграждение, подредено по намаляващ ред и ще ограничим резултата до първите 9 записа:

SELECT waiter\_id AS WaiterId, `name` AS WaiterName, MONTH(payment\_date) AS Month, SUM(daily\_payment) AS MontlyEarnings

FROM waiters\_payment JOIN waiters

ON waiter\_id = waiters.id

GROUP BY waiter\_id, MONTH(payment\_date)

ORDER BY MontlyEarnings DESC

LIMIT 9;

Картина, която съдържа маса

Описанието е генерирано автоматично

**Задача 8. Създайте тригер по ваш избор –** ще създадем тригер, който преди въвеждане на данни в таблицата reservations, ще проверява дали е възможно тази резервация да бъде направена, според вече наличните резервации в таблицата:

delimiter |

DROP TRIGGER IF EXISTS constraint\_for\_reservation;

CREATE TRIGGER constraint\_for\_reservation BEFORE INSERT ON reservations

FOR EACH ROW

BEGIN

DECLARE bef\_event\_seats TINYINT;

DECLARE res\_id INT;

DECLARE diff\_hours TINYINT;

DECLARE bef\_event\_type\_id INT;

DECLARE bef\_event\_hours TINYINT;

DECLARE count\_of\_rows INT;

DECLARE iterator INT;

DECLARE date\_time\_res\_closing DATETIME;

DECLARE event\_seats TINYINT;

DECLARE flag TINYINT;

DECLARE new\_event\_hours TINYINT;

DECLARE not\_avail\_seats INT;

SELECT duration\_hours INTO new\_event\_hours

FROM `events`

WHERE id = NEW.event\_type\_id;

SET @result = 0;

SET @currSeats = NEW.number\_of\_people;

SET @bef\_seats\_sum = 0;

SET flag = 1;

SET not\_avail\_seats = 0;

SET date\_time\_res\_closing= STR\_TO\_DATE(CONCAT(DATE(DATE\_ADD( NEW.reservation\_dateTime, INTERVAL 1 DAY)), ' ', '02:00:00'), '%Y-%m-%d %H:%i:%s');

IF (DATE\_ADD(NEW.reservation\_dateTime, INTERVAL new\_event\_hours HOUR) > date\_time\_res\_closing)

THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = "Съжалявам, но ресторанта ще бъде затворен докато вашето събитие приключи!";

END IF;

CALL available\_tables(NEW.reservation\_dateTime, NEW.id, @currSeats, @result);

CREATE TEMPORARY TABLE tempRes(

id INT,

difference\_hours INT,

bef\_e\_type\_id INT

)ENGINE = MEMORY;

CREATE TEMPORARY TABLE tempTabBefEvents(

tempTabBefEvenetId INT,

tempTabSeats TINYINT

)ENGINE = MEMORY;

IF (@result = 0)

THEN

INSERT INTO tempRes

SELECT id, TIMESTAMPDIFF(HOUR, reservation\_dateTime, NEW.reservation\_dateTime), event\_type\_id

FROM reservations

WHERE DATE(reservation\_dateTime) = DATE(NEW.reservation\_dateTime)

ORDER BY CASE WHEN reservation\_dateTime <= NEW.reservation\_dateTime THEN 0 ELSE 1 END ASC,

ABS(TIMESTAMPDIFF(HOUR, reservation\_dateTime, NEW.reservation\_dateTime)) ASC, reservation\_dateTime ASC;

SELECT COUNT(id) INTO count\_of\_rows

FROM tempRes;

SET iterator = 1;

res\_loop: WHILE (iterator >= 1 AND iterator <= count\_of\_rows)

DO

SELECT id, difference\_hours, bef\_e\_type\_id INTO

res\_id , diff\_hours, bef\_event\_type\_id

FROM tempRes

LIMIT 1;

DELETE FROM tempRes

WHERE id = res\_id;

SELECT duration\_hours INTO bef\_event\_hours

FROM `events`

WHERE id = bef\_event\_type\_id;

IF (diff\_hours > 0 AND diff\_hours >= bef\_event\_hours)

THEN

IF (flag = 1)

THEN

CALL events\_sum(@bef\_seats\_sum, res\_id);

IF (@currSeats + @bef\_seats\_sum >= NEW.number\_of\_people)

THEN

SET flag = 0;

END IF;

END IF;

ELSE IF (diff\_hours <= 0 AND ABS(diff\_hours) >= new\_event\_hours)

THEN

IF (flag = 1)

THEN

CALL events\_sum(@bef\_seats\_sum, res\_id);

IF (@currSeats + @bef\_seats\_sum >= NEW.number\_of\_people)

THEN

LEAVE res\_loop;

END IF;

END IF;

ELSE

SELECT number\_of\_people INTO event\_seats

FROM reservations

WHERE id = res\_id;

SET not\_avail\_seats = not\_avail\_seats + event\_seats;

IF (not\_avail\_seats + NEW.number\_of\_people > (SELECT SUM(seats) FROM `tables`))

THEN

DROP TEMPORARY TABLE IF EXISTS tempRes;

DROP TEMPORARY TABLE IF EXISTS tempTables;

DROP TEMPORARY TABLE IF EXISTS tempTabBefEvents;

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = "Нямаме толкова свободни места за вашата резервация за този час, те вече са запазени за друга такава!";

END IF;

INSERT INTO tempTabBefEvents (tempTabBefEvenetId)

SELECT table\_id

FROM reservation\_table

WHERE reservation\_id = res\_id;

END IF;

END IF;

SET iterator = iterator + 1;

END WHILE;

IF ((iterator - 1 = count\_of\_rows AND flag = 1))

THEN

DROP TEMPORARY TABLE IF EXISTS tempRes;

DROP TEMPORARY TABLE IF EXISTS tempTables;

DROP TEMPORARY TABLE IF EXISTS tempTabBefEvents;

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = "Нямаме толкова много свободни места за вашата резервация за този час!";

END IF;

END IF;

DROP TEMPORARY TABLE IF EXISTS tempRes;

END;

|

delimiter ;

Ще създадем и тригер, който след успешно въведена резервация в таблицата reservations, ще въвежда в таблицата reservation\_table, съответната резервация заедно с нейните резервирани маси:

delimiter |

DROP TRIGGER IF EXISTS insert\_into\_reservations\_tables;

CREATE TRIGGER insert\_into\_reservations\_tables AFTER INSERT ON reservations

FOR EACH ROW

BEGIN

CALL update\_reser\_tab(NEW.id, NEW.number\_of\_people);

END;

|

delimiter ;

**Задача 9. Създайте процедура, в която демонстрирате използване на  
курсор –** ще създадем няколко процедури, които ще ни помогнат в процеса на резервации, като първата от тях ще бъде процедура, която проверява дали има достатъчно свободни маси за съответната резервация:

delimiter |

DROP PROCEDURE IF EXISTS available\_tables;

CREATE PROCEDURE available\_tables(IN res\_dateTime DATETIME, IN res\_id INT,

INOUT res\_seats TINYINT, OUT res TINYINT)

BEGIN

DECLARE finished INT;

DECLARE currTableId INT;

DECLARE currAvaSeats1 TINYINT;

DECLARE tableCursor CURSOR FOR

SELECT id

FROM `tables`

ORDER BY seats;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;

SET currAvaSeats1 = 0;

SET finished = 0;

CREATE TEMPORARY TABLE tempTables(

table\_id INT,

table\_seats TINYINT

)ENGINE = MEMORY;

OPEN tableCursor;

table\_loop: WHILE(finished = 0)

DO

FETCH tableCursor INTO currTableId;

IF(finished = 1)

THEN

LEAVE table\_loop;

END IF;

INSERT INTO tempTables

SELECT id, seats

FROM `tables`

WHERE id = currTableId

AND id NOT IN (

SELECT table\_id

FROM reservation\_table

WHERE reservation\_id IN (

SELECT id

FROM reservations

WHERE DATE(reservation\_dateTime) = DATE(res\_dateTime)

)

);

SELECT SUM(table\_seats) INTO currAvaSeats1

FROM tempTables;

END WHILE;

CLOSE tableCursor;

SET currAvaSeats1 = IFNULL(currAvaSeats1, 0);

IF (currAvaSeats1 < res\_seats)

THEN

DELETE FROM tempTables;

SET res = 0;

SET res\_seats = currAvaSeats1;

ELSE

SET res = 1;

END IF;

END;

|

delimiter ;

Втората процедура, ще ни връща сумата на местата на свободните маси за съответната резервация:

delimiter |

DROP PROCEDURE IF EXISTS events\_sum;

CREATE PROCEDURE events\_sum(OUT sum\_of\_seats INT, IN res\_id INT)

BEGIN

INSERT INTO tempTables

SELECT id, seats

FROM `tables`

WHERE id IN (

SELECT table\_id

FROM reservation\_table

WHERE reservation\_id = res\_id

)

AND id NOT IN (

SELECT tempTabBefEvenetId

FROM tempTabBefEvents

);

SELECT SUM(table\_seats) INTO sum\_of\_seats

FROM tempTables;

END;

|

delimiter ;

Третата процедура, ще въвежда масите за съответната резервация в таблицата reservation\_table:

delimiter |

DROP PROCEDURE IF EXISTS update\_reser\_tab;

CREATE PROCEDURE update\_reser\_tab(IN new\_res\_id INT, IN res\_seats TINYINT)

BEGIN

DECLARE tempTabId INT;

DECLARE tempTabSeats TINYINT;

DECLARE currSeats TINYINT;

DECLARE table\_id\_min\_seats INT;

DECLARE table\_min\_seats TINYINT;

DECLARE tabCursor CURSOR FOR

SELECT table\_id, table\_seats FROM tempTables

ORDER BY table\_seats DESC;

SET currSeats = 0;

OPEN tabCursor;

tab\_loop: WHILE (TRUE)

DO

FETCH tabCursor INTO tempTabId, tempTabSeats;

IF (currSeats + tempTabSeats >= res\_seats)

THEN

SET table\_id\_min\_seats = tempTabId;

IF (currSeats + tempTabSeats - res\_seats != 1 AND

currSeats + tempTabSeats != res\_seats)

THEN

SELECT table\_id, table\_seats INTO table\_id\_min\_seats, table\_min\_seats

FROM tempTables

WHERE table\_seats + currSeats >= res\_seats

ORDER BY table\_seats ASC

LIMIT 1;

IF (res\_seats < 5 AND currSeats + table\_min\_seats > 2 \* res\_seats)

THEN

DROP TEMPORARY TABLE IF EXISTS tempRes;

DROP TEMPORARY TABLE IF EXISTS tempTables;

DROP TEMPORARY TABLE IF EXISTS tempTabBefEvents;

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT= "Съжалявам, но свободните маси, които имаме в момента, са повече от два пъти по-големи от това, което се опитвате да резервирате!";

END IF;

END IF;

INSERT INTO reservation\_table

VALUES (new\_res\_id, table\_id\_min\_seats);

LEAVE tab\_loop;

ELSE

INSERT INTO reservation\_table

VALUES (new\_res\_id, tempTabId);

END IF;

SET currSeats = currSeats + tempTabSeats;

END WHILE;

CLOSE tabCursor;

DROP TEMPORARY TABLE IF EXISTS tempTables;

DROP TEMPORARY TABLE IF EXISTS tempTabBefEvents;

END;

|

delimiter ;

И последната четвърта процедура, ще изтрива резервации от таблицата reservations:

delimiter |

DROP PROCEDURE IF EXISTS delete\_reservation;

CREATE PROCEDURE delete\_reservation(IN res\_id INT)

BEGIN

DELETE FROM reservations

WHERE id = res\_id;

END;

|

delimiter ;

Ще тестваме със следните заявки:

INSERT INTO reservations(reservation\_dateTime, client\_first\_name, client\_second\_name, client\_phone, number\_of\_people, event\_type\_id)

VALUES ('2023-04-06 15:00:00', 'Стоян', 'Стефанов', '0878668293', 5, 3);

INSERT INTO reservations(reservation\_dateTime, client\_first\_name, client\_second\_name, client\_phone, number\_of\_people, event\_type\_id)

VALUES ('2023-04-06 13:00:00', 'Стоян', 'Стефанов', '0878668293', 70, 3);

INSERT INTO reservations(reservation\_dateTime, client\_first\_name, client\_second\_name, client\_phone, number\_of\_people, event\_type\_id)

VALUES ('2023-04-06 22:00:00', 'Стоян', 'Стефанов', '0878668293', 10, 4);

Резултатът от следните заявки:





