ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ

Катедра - “Компютърни системи”

КУРСОВ ПРОЕКТ
ПО БАЗИ ОТ ДАННИ

Студент: Владислав Валентинов Атанасов
ФАК. номер: 121222190
Група: 39

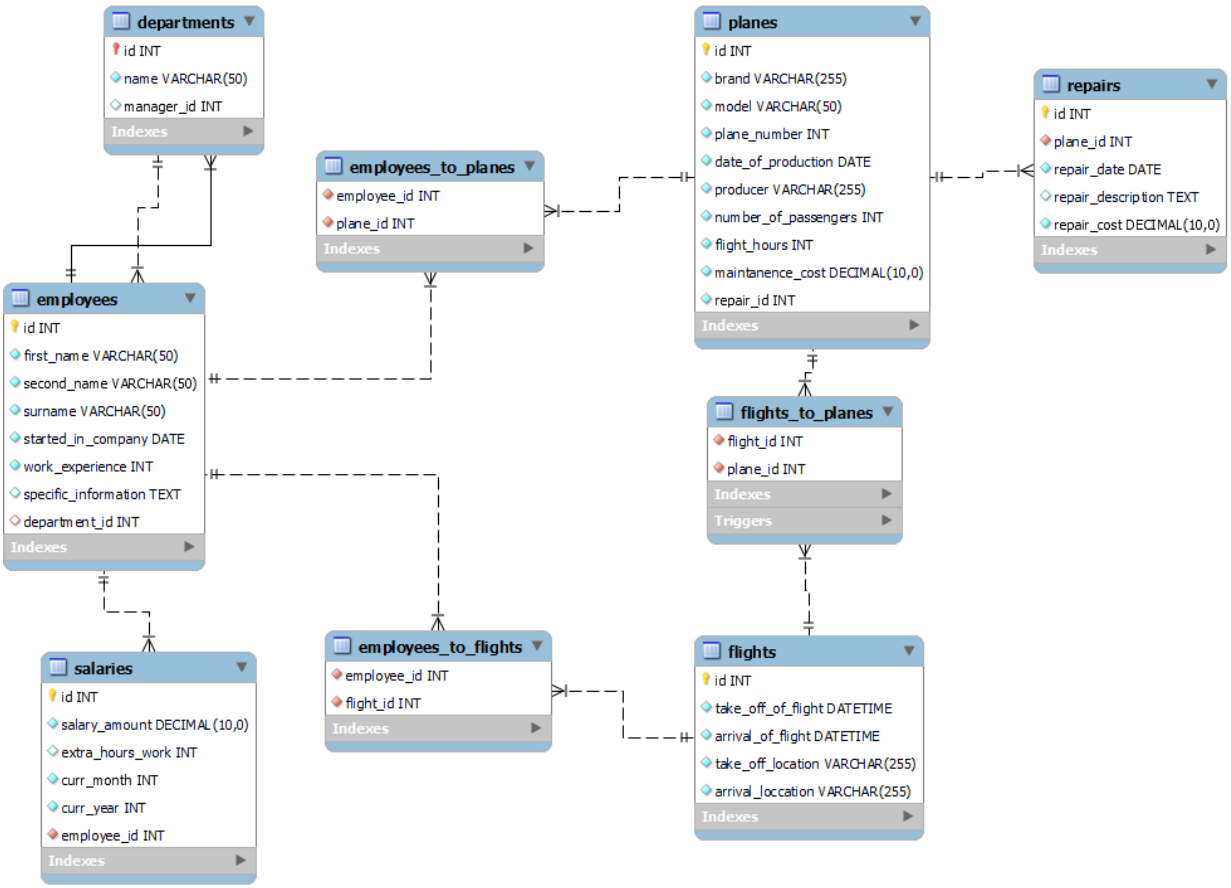
Тема № 26
Проектирайте база данни за система за малко частно летище. В летището има различни отдели, всеки отдел се ръководи от мениджър. Отделите са: администрация, общи работници, стюардеси, пилоти. В базата се пази информация за самолетите, които са собственост на летището, разходите за тяхната поддръжка и история на извършените ремонти. За всеки самолет се съхранява марка, модел, номер, дата на производство, производител, брой пътници, които побира, брой летателни часове. Системата пази информация за всички полети (минали и предстоящи). За всеки полет се съхранява номер на полета, номер на самолет, пилот/и, стюардеси, дата и час на излитане, дата и час на кацане, начална и крайна дестинация.

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.

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

От заданието се изисква да се създадат таблиците: ”departments”, ”planes”, ”flights”, “repairs“. Създават се и допълнителни таблици: ”employees”, която съхранява информация за всеки служител(собствено, бащино и фамилно име, кога е започнал да работи в компанията, колко години стаж има, някаква специфична информация за служителя и към кой отдел е) и ”salaries”, която съхранява информация за заплатите на всеки служител (размер на месечната заплата, колко извънредни часове е работил даденият служител, месецът и годината, за която става плащането).

ER-диаграма (Entity-Relationship Diagram) на базата:



CREATE TABLE заявки, с които създаваме базата:

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

```sql
CREATE TABLE departments(
    id INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
    name VARCHAR(50) NOT NULL,
    manager_id INT DEFAULT NULL
);

CREATE TABLE employees(
    id INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
    first_name VARCHAR(50) NOT NULL,
    second_name VARCHAR(50) NOT NULL,
    surname VARCHAR(50) NOT NULL,
    started_in_company DATE NOT NULL,
    work_experience INT NOT NULL,
    specific_information TEXT NULL,
    department_id INT DEFAULT NULL,
    FOREIGN KEY (department_id) REFERENCES departments(id)
);

ALTER TABLE departments ADD CONSTRAINT FK_employee_id
FOREIGN KEY (id) REFERENCES employees(id);

CREATE TABLE planes(
    id INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
    brand VARCHAR(255) NOT NULL,
    model VARCHAR(50) NOT NULL,
    plane_number INT NOT NULL,
    date_of_production DATE NOT NULL,
    producer VARCHAR(255) NOT NULL,
    number_of_passengers INT NOT NULL,
    flight_hours INT NOT NULL,
    maintanence_cost DECIMAL NOT NULL,
    repair_id INT NOT NULL
);

CREATE TABLE repairs(
    id INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
    plane_id INT NOT NULL,
    repair_date DATE NOT NULL,
    repair_description TEXT,
    repair_cost DECIMAL NOT NULL,
    CONSTRAINT FOREIGN KEY (plane_id) REFERENCES planes(id)
);
```

```
CREATE TABLE flights(
    id INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
    take_off_of_flight DATETIME NOT NULL,
    arrival_of_flight DATETIME NOT NULL,
    take_off_location VARCHAR(255) NOT NULL,
    arrival_loccation VARCHAR(255) NOT NULL
);

CREATE TABLE flights_to_planes (
    flight_id INT NOT NULL,
    plane_id INT NOT NULL,
    CONSTRAINT FOREIGN KEY (plane_id) REFERENCES planes(id),
    CONSTRAINT FOREIGN KEY (flight_id) REFERENCES flights(id)
);

CREATE TABLE employees_to_flights (
    employee_id INT NOT NULL,
    flight_id INT NOT NULL,
    CONSTRAINT FOREIGN KEY (employee_id) REFERENCES employees(id),
    CONSTRAINT FOREIGN KEY (flight_id) REFERENCES flights(id)
);

CREATE TABLE employees_to_planes (
    employee_id INT NOT NULL,
    plane_id INT NOT NULL,
    CONSTRAINT FOREIGN KEY (employee_id) REFERENCES employees(id),
    CONSTRAINT FOREIGN KEY (plane_id) REFERENCES planes(id)
);

CREATE TABLE salaries (
    id INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
    salary_amount DECIMAL NOT NULL,
    extra_hours_work INT NULL,
    curr_month INT NOT NULL,
    curr_year INT NOT NULL,
    employee_id INT NOT NULL,
    CONSTRAINT FOREIGN KEY (employee_id) REFERENCES employees(id),
    UNIQUE KEY (employee_id, curr_month, curr_year)
);
```

INSERT заявки, с които добавяме тестови данни в базата:

INSERT INTO employees (first_name, second_name, surname, started_in_company,

```
  work_experience, specific_information)
VALUES
('John', 'Doe', 'Smith', '2020-05-15', 2, NULL),
('Jane', 'Doe', 'Smith', '2019-10-20', 13, 'Experienced pilot'),
('Alice', 'Johnson', 'Brown', '2021-01-10', 1, NULL),
('Marry', 'Hana', 'Green', '2020-05-15', 2, NULL),
('Harry', 'Doe', 'Thomsan', '2019-10-20', 3, NULL),
('Elizabeth', 'Huffer', 'Summer', '2021-01-10', 1, NULL);

INSERT INTO departments (name, manager_id)
VALUES
('Pilots', 1),
('Workers',  2),
('Administration', 3),
('Stewards', 4);

UPDATE employees
SET department_id =
  CASE
    WHEN first_name = 'John' AND second_name = 'Doe'
      AND surname = 'Smith' THEN 1
    WHEN first_name = 'Jane' AND second_name = 'Doe'
      AND surname = 'Smith' THEN 2
    WHEN first_name = 'Alice' AND second_name = 'Johnson'
      AND surname = 'Brown' THEN 3
    WHEN first_name = 'Marry' AND second_name = 'Hana'
      AND surname = 'Green' THEN 4
    WHEN first_name = 'Harry' AND second_name = 'Doe'
      AND surname = 'Thomsan' THEN 2
    WHEN first_name = 'Elizabeth' AND second_name = 'Huffer'
      AND surname = 'Summer' THEN 4
    ELSE department_id
  END;

INSERT INTO planes (brand, model, plane_number, date_of_production, producer,
  number_of_passengers, flight_hours, maintanence_cost, repair_id)
VALUES
('Boeing', '737', 1001, '2019-01-20', 'Boeing Company', 150, 5000, 2500.00, 1),
```

```
('Airbus', 'A320', 2001, '2020-05-15', 'Airbus SE', 180, 4500, 2800.00, 2),
('Boeing', '747', 3001, '2018-11-10', 'Boeing Company', 300, 7000, 4000.00, 3);

INSERT INTO flights (take_off_of_flight, arrival_of_flight,
  take_off_location, arrival_loccation)
VALUES
('2024-04-01 08:00:00', '2024-04-01 10:30:00', 'New York', 'Los Angeles'),
('2024-04-02 10:00:00', '2024-04-02 13:30:00', 'London', 'Paris'),
('2024-04-03 12:30:00', '2024-04-03 15:00:00', 'Tokyo', 'Beijing');

INSERT INTO salaries (id, salary_amount,
  extra_hours_work, curr_month, curr_year, employee_id)
VALUES
(1, 8000.00, 11, 4 , 2024, 1),
(2, 4000.00, 7, 4 , 2024, 2),
(3, 4500.00, NULL, 4 , 2024, 3),
(4, 5000.00, NULL, 4 , 2024, 4),
(5, 4000.00, 5, 4 , 2024, 5),
(6, 4500.00, 9, 4 , 2024, 6);

INSERT INTO employees_to_flights (employee_id, flight_id)
VALUES
(2, 1),
(3, 1),
(6, 1),
(2, 2),
(3, 2),
(6, 2);

INSERT INTO employees_to_planes (employee_id, plane_id)
VALUES
(2, 1),
(3, 1),
(6, 1),
(2, 2),
(3, 2),
(6, 2);
```

```
INSERT INTO flights_to_planes (flight_id, plane_id)
VALUES
(1, 1),
(1, 1),
(1, 1),
(2, 2),
(2, 2),
(2, 2);

INSERT INTO repairs (plane_id, repair_date, repair_description, repair_cost)
VALUES
(1, '2023-05-15', 'Engine maintenance', 5000.00),
(2, '2023-06-20', 'Replacement of avionics system', 8000.00),
(3, '2023-07-10', 'Repair of landing gear', 6000.00),
(1, '2023-08-05', 'Repair of hydraulic system', 4000.00),
(3, '2023-09-12', 'Structural repair', 7000.00);
```

2. Напишете заявка, в която демонстрирате SELECT с ограничаващо условие по избор.

Извеждаме информация за служителите получили над 5000 лева заплата за текущия месец.

```
SELECT e.first_name, e.surname,
   d.name AS dept_name, s.curr_month, s.salary_amount
FROM employees e
JOIN departments d ON e.department_id = d.id
JOIN salaries s ON e.id = s.employee_id
WHERE s.curr_month = MONTH(NOW())
AND s.curr_year = YEAR(NOW())
AND s.salary_amount >= 5000;
```

	first_name varchar	surname varchar	dept_name varchar	curr_month int	salary_amount newdecimal
1	John	Smith	Pilots	4	8000
2	Marry	Green	Stewards	4	5000

3. Напишете заявка, в която използвате агрегатна функция и GROUP BY по ваш

избор.
Извеждаме информация за средната заплата на служителите по отдели.

```
SELECT d.name AS dept_name, AVG(s.salary_amount) AS avg_salary
FROM employees e
JOIN departments d ON e.department_id = d.id
JOIN salaries s ON e.id = s.employee_id
GROUP BY d.name;
```

	dept_name varchar	avg_salary newdecimal
1	Pilots	8000.0000
2	Workers	4000.0000
3	Administration	4500.0000
4	Stewards	4750.0000

4. Напишете заявка, в която демонстрирате INNER JOIN по ваш избор.
Извеждаме информация за заплатата на всеки служител.

```
SELECT e.first_name, e.surname, s.salary_amount
FROM employees e INNER JOIN salaries s
ON e.id = s.employee_id;
```

	first_name varchar	surname varchar	salary_amount newdecimal
1	John	Smith	8000
2	Jane	Smith	4000
3	Alice	Brown	4500
4	Marry	Green	5000
5	Harry	Thomsan	4000
6	Elizabeth	Summer	4500

5. Напишете заявка, в която демонстрирате OUTER JOIN по ваш избор.
Извеждаме информация за заплатите на всеки служител, който не е мениджър.

```
SELECT e.first_name, e.surname,
  d.name AS manager_dept, s.salary_amount
```

```
FROM employees e
LEFT JOIN departments d ON e.department_id = d.id
LEFT JOIN salaries s ON e.id = s.employee_id
WHERE e.id != d.manager_id;
```

	first_name varchar	surname varchar	manager_dep' varchar	salary_amount newdecimal
1	Harry	Thomsan	Workers	4000
2	Elizabeth	Summer	Stewards	4500

6. Напишете заявка, в която демонстрирате вложен SELECT по ваш избор.
Извеждаме информация за броят на стюартите на полет с “id” номер 2.

```
SELECT COUNT(*) AS num_steward_on_flight_2
FROM (
  SELECT employee_id
  FROM employees_to_flights
  WHERE flight_id = 2) AS flight_2
JOIN employees e ON flight_2.employee_id = e.id
JOIN departments d ON e.department_id = d.id
WHERE e.id != d.manager_id
AND d.name = 'Stewards';
```

	num_steward_on_flight bigint
1	1

7. Напишете заявка, в която демонстрирате едновременно JOIN и агрегатна функция.
Извеждаме информация за това на колко полета е бил всеки служител

```
SELECT e.first_name, e.surname,
  COUNT(ef.flight_id) AS num_flights
FROM employees e
LEFT JOIN employees_to_flights ef ON e.id = ef.employee_id
JOIN departments d ON e.department_id = d.id
WHERE e.id != d.manager_id
GROUP BY
```

```
  e.first_name,
  e.surname;
```

	first_name varchar	surname varchar	num_flights bigint
1	Harry	Thomsan	0
2	Elizabeth	Summer	2

8. Създайте тригер по ваш избор

Тригерът отговаря за автоматизацията на увеличаването на летателните часове на самолетите при добавяне на нов полет в базата данни.

```
DROP TRIGGER IF EXISTS increment_flight_hours;
DELIMITER |
CREATE TRIGGER increment_flight_hours AFTER INSERT ON flights_to_planes
FOR EACH ROW
BEGIN
  DECLARE flight_duration INT;

  SELECT TIMESTAMPDIFF(HOUR, f.take_off_of_flight, f.arrival_of_flight)
    INTO flight_duration
  FROM flights f
  WHERE f.id = NEW.flight_id;

  UPDATE planes
  SET flight_hours = flight_hours + flight_duration
  WHERE id = NEW.plane_id;
END
|
DELIMITER ;
```

Тестови заявки за тригера:

```
INSERT INTO flights (take_off_of_flight, arrival_of_flight, take_off_location,arrival_location)
VALUES
  ('2024-04-23 18:00:00', '2024-04-23 20:00:00', 'Tokyo', 'Seoul');

INSERT INTO flights_to_planes (flight_id, plane_id) VALUES (1, 1);
```

Успехно е увеличен броят летателни часове на самолет с “id” номер 1

	plane_id int	* flight_hours int
> 1	1	5002

9. Създайте процедура, в която демонстрирате използване на курсор.
Процедурата добавя към основната заплата на всеки служител заплащане спрямо извънредните му часове работа през текущия месец на база отдела, в който работи и годините му стаж спрямо начална ставка (base_bonus). Заработените часове извънреден труд се нулират при изпълнение на процедурата.
Тази процедура се изпълнява автоматично на първо число от всеки месец (или когато е предвидено да се изплащат заплатите) посредством създадено събитие (EVENT).

```
SET @base_bonus = 50;
DROP PROCEDURE IF EXISTS update_salary_based_on_extra_work;
DELIMITER |
CREATE PROCEDURE update_salary_based_on_extra_work(IN base_bonus INT)
BEGIN
  DECLARE done INT DEFAULT FALSE;
  DECLARE emp_id INT;
  DECLARE sal_amount DECIMAL;
  DECLARE extra_hours INT;
  DECLARE dept_id INT;
  DECLARE exp_years INT;
  DECLARE dept_multiplier FLOAT;
  DECLARE exp_multiplier FLOAT DEFAULT 1.0;

  DECLARE cur_salary CURSOR FOR
    SELECT s.employee_id, s.salary_amount,
      s.extra_hours_work, e.department_id, e.work_experience
    FROM salaries s
    JOIN employees e ON s.employee_id = e.id
    WHERE s.extra_hours_work IS NOT NULL;

  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
```

```
OPEN cur_salary;

salary_loop: LOOP
  FETCH cur_salary INTO emp_id, sal_amount, extra_hours, dept_id, exp_years;
  IF done THEN
    LEAVE salary_loop;
  END IF;

  CASE dept_id
    WHEN 1 THEN
      SET dept_multiplier = 2.5;
    WHEN 2 THEN
      SET dept_multiplier = 1.0;
    WHEN 3 THEN
      SET dept_multiplier = 1.1;
    WHEN 4 THEN
      SET dept_multiplier = 1.75;
  END CASE;

  IF exp_years >= 10 THEN
    SET exp_multiplier = 1.5;
  ELSEIF exp_years >= 5 THEN
    SET exp_multiplier = 1.3;
  ELSEIF exp_years >= 3 THEN
    SET exp_multiplier = 1.1;
  END IF;

  UPDATE salaries
  SET salary_amount = sal_amount + (base_bonus * dept_multiplier * exp_multiplier)
  WHERE employee_id = emp_id;

  UPDATE salaries
  SET salary_amount = 0
  WHERE employee_id = emp_id;

END LOOP;
```

```
    CLOSE cur_salary;
END |
DELIMITER ;
```

Таблица “salaries” преди извикване на процедурата:

	id int	salary_amoun decimal(10,0)	extra_hours_work int	curr_mon int	curr_year int	employee_ int
1	1	8000	11	4	2024	1
2	2	4000	7	4	2024	2
3	3	4500	(NULL)	4	2024	3

Таблица “salaries” след извикване на процедурата:

	id int	salary_amoun decimal(10,0)	extra_hours_work int	curr_mon int	curr_year int	employee_ int
1	1	8125	0	4	2024	1
2	2	4075	0	4	2024	2
3	3	4500	(NULL)	4	2024	3

- **Събитие (EVENT) при което се изпълнява процедурата.**

```
CREATE EVENT IF NOT EXISTS update_salaries_event
ON SCHEDULE
  EVERY 1 MONTH
  STARTS CONCAT(YEAR(CURRENT_DATE()), '-',
    LPAD(MONTH(CURRENT_DATE()), 2, '0'), '-01')
DO
  CALL update_salary_based_on_extra_work(@base_bonus);

SET GLOBAL event_scheduler = ON;
```