 ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ

Факултет по компютърни системи и технологии

**КУРСОВ ПРОЕКТ**

**ПО БАЗИ ОТ ДАННИ**

**Студент:** Красимир Георгиев Гушмаков

**Фак. №:** 121223008 **Група:** 44

|  |
| --- |
| **Тема №20** |
| Да се разработи база данни за сървърно приложение за мониторинг на различни услуги в мрежа. В базата от данни има пълно описание на всички услуги (хардуер, софтуер, отговорници и т.н.). Клиентските приложения (сървърите с услугите, които работят) изпращат периодично информация за своя статус. Пази се информация за това коя система е дала отказ, кой е репортнал този отказ, за колко време системата е била offline и т.н. |

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.

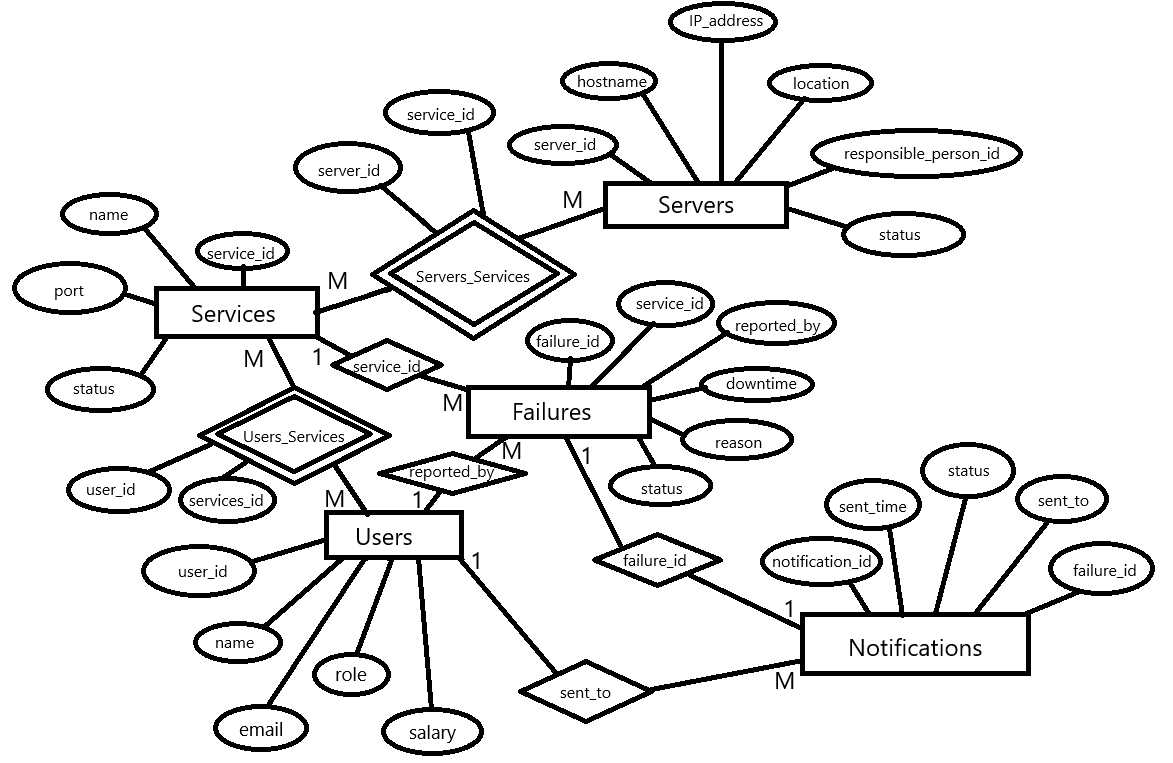
8. Създайте тригер по ваш избор.

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 да съхраняваме информация, според заданието са: Users, Servers и Services. В таблицата Users ще се съдържа информация за имената на потребителите, техният имейл адрес, тяхната заплата, както и за тяхната роля, която те изпълняват в сървъра. В таблицата Servers ще се съдържа информация за името на сървъра, ip-адреса му, локацията му, актуалния му статус, както и кой човек отговаря за поддръжката му. В таблицата Services ще се съхранява информация за името на самата услуга, актуалния и статус, както и за порта, чрез който се свързва със сървъра. Освен тези три основни таблици ще създадем други две, които ще се казват Failures и Notifications. Първата от които ще съхранява информация за това колко време даден услуга е била неактивен, причината за тази неактивност, актуалния статус на проблема, уникалния номер на услугата, която е имала проблем, както и от кого е бил открит този проблем. Втората от тези две таблици ще съдържа в себе си информация за времето по което е изпратено известието за проблема, актуалния статус на това известие, дали то е получено или все още не е, до кого е изпратено, както и уникалния номер на проблема, с което е свързано даденото известие. И последните две таблици, които ще създадем за цялостното изпълнение на задачата са двете свързващи таблици Users\_Services, съдържаща уникалните номера на потребителите и на услугите, които те ползват, и Servers\_Services, съдържаща уникалните номера на сървърите и услугите, които те предлагат. Тези две последни таблици реализира връзката Много към Много както на потребителите с услугите, така и на сървърите с услугите. За проектирането на базата ще използваме модела ER-диаграма (Entity Relationship Diagram).



**Заявките, с които създаваме базата данни и таблиците са:**

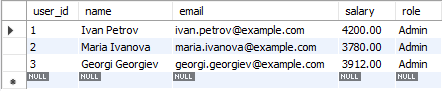
|  |
| --- |
| **create database ServerApplication;**  **use ServerApplication;**  **create table users(**  **user\_id int primary key auto\_increment,**  **name varchar(50),**  **email varchar(50),**  **salary decimal(10,2),**  **role varchar(50)**  **);**  **create table servers(**  **server\_id int primary key auto\_increment,**  **hostname varchar(50),**  **ip\_address varchar(20),**  **location varchar(50),**  **status varchar(20),**  **responsible\_person\_id int,**  **constraint foreign key (responsible\_person\_id) references users(user\_id) on delete cascade on update cascade**  **);**  **create table services(**  **service\_id int primary key auto\_increment,**  **name varchar(50),**  **status varchar(20),**  **port varchar(20)**  **);**  **create table failures(**  **failure\_id int primary key auto\_increment,**  **downtime time,**  **reason varchar(50),**  **status varchar(20),**  **service\_id int,**  **reported\_by int,**  **constraint foreign key (service\_id) references services(service\_id) on delete cascade on update cascade,**  **constraint foreign key (reported\_by) references users(user\_id) on delete cascade on update cascade**  **);**  **create table notifications(**  **notification\_id int primary key auto\_increment,**  **sent\_time time,**  **status varchar(20),**  **sent\_to int,**  **failure\_id int unique,**  **constraint foreign key (sent\_to) references users(user\_id) on delete cascade on update cascade,**  **constraint foreign key (failure\_id) references failures(failure\_id) on delete cascade on update cascade**  **);**  **create table users\_services(**  **user\_id int,**  **service\_id int,**  **primary key(user\_id, service\_id),**  **constraint foreign key (user\_id) references users(user\_id) on delete cascade on update cascade,**  **constraint foreign key (service\_id) references services(service\_id) on delete cascade on update cascade**  **);**  **create table servers\_services(**  **server\_id int,**  **service\_id int,**  **primary key(server\_id, service\_id),**  **constraint foreign key (server\_id) references servers(server\_id) on delete cascade on update cascade,**  **constraint foreign key (service\_id) references services(service\_id) on delete cascade on update cascade**  **);** |

**Добавяме и тестови данни в таблиците:**

|  |
| --- |
| **-- Вмъкване на данни в таблицата users**  **INSERT INTO users (user\_id, name, email, salary, role)**  **VALUES (1, 'Ivan Petrov', 'ivan.petrov@example.com', 4200.00, 'Admin'),**  **(2, 'Maria Ivanova', 'maria.ivanova@example.com',3780.00, 'Admin'),**  **(3, 'Georgi Georgiev', 'georgi.georgiev@example.com', 3912.00, 'Admin'),**  **(4, 'Elena Stoyanova', 'elena.stoyanova@example.com', 3160.00, 'Technician'),**  **(5, 'Petar Dimitrov', 'petar.dimitrov@example.com', 2950.00, 'Technician'),**  **(6, 'Stoyan Kostov', 'stoyan.kostov@example.com', 3004.00, 'Technician'),**  **(7, 'Kristina Hristova', 'kristina.hristova@example.com', 0.00, 'User'),**  **(8, 'Dimitar Nikolov', 'dimitar.nikolov@example.com', 0.00, 'User'),**  **(9, 'Nikolay Kolev', 'nikolay.kolev@example.com', 0.00, 'User'),**  **(10, 'Viktor Andreev', 'viktor.andreev@example.com', 0.00, 'User');**  **-- Вмъкване на данни в таблицата servers**  **-- Само потребители с роля "Admin" или "Technician" могат да отговарят за сърварите**  **INSERT INTO servers (server\_id, hostname, ip\_address, location, status, responsible\_person\_id)**  **VALUES (1, 'server1', '192.168.1.1', 'Sofia', 'Active', 1),**  **(2, 'server2', '192.168.1.2', 'Plovdiv', 'Inactive', 2),**  **(3, 'server3', '192.168.1.3', 'Varna', 'Active', 3),**  **(4, 'server4', '192.168.1.4', 'Burgas', 'Active', 4),**  **(5, 'server5', '192.168.1.5', 'Ruse', 'Inactive', 5),**  **(6, 'server6', '192.168.1.6', 'Stara Zagora', 'Active', 6);**  **-- Вмъкване на данни в таблицата services**  **INSERT INTO services (service\_id, name, status, port)**  **VALUES (1, 'Web Server', 'Running', '80'),**  **(2, 'Database Server', 'Running', '3306'),**  **(3, 'Email Server', 'Down', '25'),**  **(4, 'VPN Service', 'Running', '1194'),**  **(5, 'File Server', 'Running', '21'),**  **(6, 'Monitoring Service', 'Down', '8080'),**  **(7, 'DNS Server', 'Running', '53'),**  **(8, 'Firewall', 'Running', '443'),**  **(9, 'SSH Service', 'Running', '22'),**  **(10, 'Backup Service', 'Down', '9090');**  **-- Вмъкване на данни в таблицата failures**  **-- Само потребители с роля "User" могат да докладват за откази на системата**  **INSERT INTO failures (failure\_id, downtime, reason, status, service\_id, reported\_by)**  **VALUES (1, '00:30:00', 'Network Issue', 'Resolved', 3, 9),**  **(2, '01:15:00', 'Hardware Failure', 'Unresolved', 6, 7),**  **(3, '00:45:00', 'Software Bug', 'Resolved', 10, 10),**  **(4, '02:00:00', 'Power Outage', 'Unresolved', 3, 8),**  **(5, '00:20:00', 'DDoS Attack', 'Resolved', 8, 7),**  **(6, '01:10:00', 'Overload', 'Unresolved', 2, 8),**  **(7, '00:50:00', 'Configuration Error', 'Resolved', 4, 10),**  **(8, '01:30:00', 'Security Breach', 'Unresolved', 7, 9),**  **(9, '02:45:00', 'Network Congestion', 'Resolved', 5, 7),**  **(10, '00:55:00', 'Storage Failure', 'Unresolved', 1, 10);**  **-- Вмъкване на данни в таблицата notifications**  **-- Известие за проблем в системата могат да получат само потребители с роля "Admin" или "Technician"**  **INSERT INTO notifications (notification\_id, sent\_time, status, sent\_to, failure\_id)**  **VALUES (1, '00:31:00', 'Sent', 3, 1),**  **(2, '01:16:00', 'Sent', 6, 2),**  **(3, '00:46:00', 'Sent', 4, 3),**  **(4, '02:01:00', 'Failed', 3, 4),**  **(5, '00:21:00', 'Sent', 2, 5),**  **(6, '01:11:00', 'Sent', 2, 6),**  **(7, '00:51:00', 'Failed', 4, 7),**  **(8, '01:31:00', 'Sent', 1, 8),**  **(9, '02:46:00', 'Sent', 5, 9),**  **(10, '00:56:00', 'Failed', 1, 10);**  **-- Вмъкване на данни в таблицата users\_services**  **INSERT INTO users\_services (user\_id, service\_id)**  **VALUES (1, 3), (2, 7), (3, 1), (4, 9), (5, 6),**  **(9, 6), (7, 7), (1, 8), (7, 9), (3, 10),**  **(10, 3), (3, 9), (8, 1), (4, 10), (2, 5),**  **(1, 6), (6, 7), (8, 8), (6, 9), (10, 7);**  **-- Вмъкване на данни в таблицата servers\_services**  **INSERT INTO servers\_services (server\_id, service\_id)**  **VALUES (1, 1), (2, 2), (3, 3), (4, 4), (5, 5),**  **(6, 6), (1, 7), (2, 8), (3, 9), (4, 10);** |

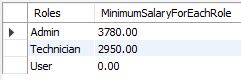
**Задача 2. Напишете заявка, в която демонстрирате SELECT с ограничаващо условие по избор –** ще изведем информация за всички потребители, които притежават роля “Admin”.

|  |
| --- |
| **SELECT \* FROM users**  **WHERE role like '%Admin%';** |

****

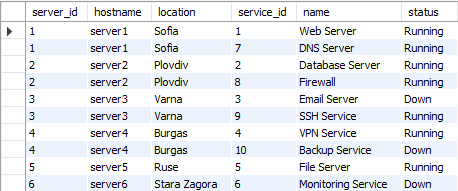
**Задача 3. Напишете заявка, в която използвате агрегатна функция и GROUP BY по ваш избор –** ще изведем минималните заплати на всяка една различна роля, които те получават за поддръжката на сървърите и техните услуги.

|  |
| --- |
| **SELECT role as Roles, min(salary) as MinimumSalaryForEachRole**  **FROM users**  **GROUP BY role;** |



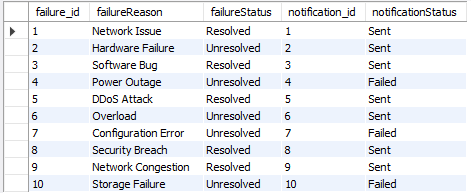
**Задача 4. Напишете заявка, в която демонстрирате INNER JOIN по ваш избор –** ще напишем заявка, която да изведе всички сървъри и поддържаните от тях услуги. Информацията се съхранява в таблиците servers, services и servers\_services.

|  |
| --- |
| **SELECT servers.server\_id, servers.hostname, servers.location,**  **services.service\_id, services.name, services.status**  **FROM servers\_services**  **JOIN servers ON servers\_services.server\_id = servers.server\_id**  **JOIN services ON servers\_services.service\_id = services.service\_id;** |

****

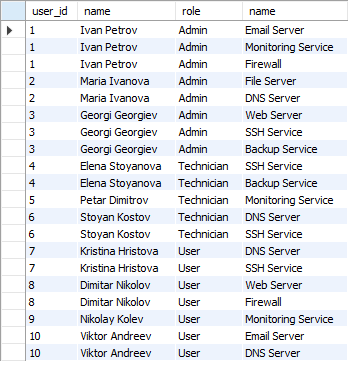
**Задача 5. Напишете заявка, в която демонстрирате OUTER JOIN по ваш избор –** ще напишем заявка, с която ще изведем информация за всички проблеми, които възникват по време на използването на различните сървърни услуги и известията, които се изпращат за оправянето им. Информацията се съдържа в таблиците failurеs и notifications. За свързването ще използваме RIGHT OUTER JOIN и по този начин, ако има проблем, за които все още няма пуснато известие, той би трябвало също да присъства в таблицата.

|  |
| --- |
| **SELECT failures.failure\_id, failures.reason as failureReason, failures.status as failureStatus,**  **notifications.notification\_id, notifications.status as notificationStatus**  **FROM failures**  **RIGHT OUTER JOIN notifications ON failures.failure\_id = notifications.failure\_id;** |

****

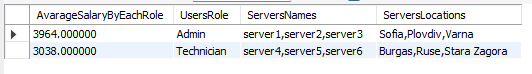
**Задача 6. Напишете заявка, в която демонстрирате вложен SELECT по ваш избор –** ще създадем заявка, с която ще изведем всички потребители заедно с услугите, които те активно използват. Информацията се намира в таблиците users и services. Те не са свързани директно, а чрез трета таблица - users\_services, която налага използването на вложен SELECT, за да извлечем информацията.

|  |
| --- |
| **SELECT users.user\_id, users.name, users.role, services.name**  **FROM users**  **JOIN services**  **ON users.user\_id IN (**  **SELECT user\_id**  **FROM users\_services**  **WHERE users\_services.service\_id = services.service\_id**  **);** |

****

**Задача 7. Напишете заявка, в която демонстрирате едновременно JOIN и агрегатна функция –** ще изведем средната заплата според различните потребителски роли, самите видове потребителски ролите, както и имената на сървърите и локациите им. Ще използваме GROUP\_CONCAT за последните две изброени колони с цел да обединим всички имена на сървъри и техните локации в един общ ред, според зависи дали те се обслужват от потребители с ролята Admin или с ролята Technician.

|  |
| --- |
| **SELECT avg(users.salary) as AvarageSalaryByEachRole, users.role as UsersRole,**  **group\_concat(servers.hostname) as ServersNames, group\_concat(servers.location) as ServersLocations**  **FROM users**  **JOIN servers ON users.user\_id = servers.responsible\_person\_id**  **GROUP BY role;** |

****

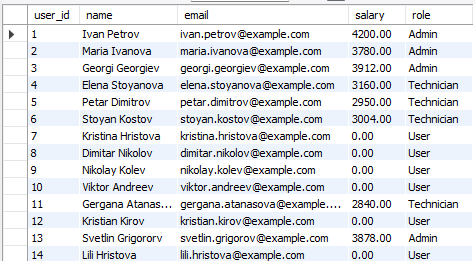
**Задача 8. Създайте тригер по ваш избор –** ще създадем един тригер, който има за цел да гарантира, че **всички нови потребители с роля "User" винаги имат заплата 0.00**, дори ако при INSERT е подадена друга стойност.

|  |
| --- |
| **DELIMITER //**  **CREATE TRIGGER users\_salary\_limit**  **BEFORE INSERT ON users**  **FOR EACH ROW**  **BEGIN**  **IF NEW.role = 'User' AND NEW.salary <> 0 THEN SET NEW.salary = 0;**  **END IF;**  **END //**  **DELIMITER ;** |

Ще тестваме тригера със следна INSERT заявка:

|  |
| --- |
| **INSERT INTO users(user\_id, name, email, salary, role) VALUES**  **(11, "Gergana Atanasova", "gergana.atanasova@example.com", 2840.00, "Technician"),**  **(12, "Kristian Kirov", "kristian.kirov@example.com", 940.00, "User"),**  **(13, "Svetlin Grigororv", "svetlin.grigorov@example.com", 3878.00, "Admin"),**  **(14, "Lili Hristova", "lili.hristova@example.com", 1430.00, "User");**  **SELECT \* FROM users;** |

Резултат след извеждането на всички записи от таблицата users чрез заявка SELECT \* FROM.



**Задача 9. Създайте процедура, в която демонстрирате използване на курсор –** ще създадем процедура, която извежда в нова таблица всички потребители с роля Technician, тяхната заплата, категорията на заплатата им, както и часът, в които е направена последната справка за тази информация. За и+зпълнението на задачата ще трябва предварително да си създадем нова таблица с име technician\_reports, в която ще бъдат вкарани всички записи, отговарящи на условията, зададени в процедурата.

|  |
| --- |
| **CREATE TABLE technician\_reports (**  **technician\_id INT,**  **technician\_name VARCHAR(50),**  **salary DECIMAL(10, 2),**  **salary\_category VARCHAR(20),**  **report\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP**  **);** |

Това е кода на съответната процедура:

|  |
| --- |
| **DELIMITER //**  **CREATE PROCEDURE list\_technicians\_salaries()**  **BEGIN**  **DECLARE done INT DEFAULT FALSE;**  **DECLARE tech\_name VARCHAR(50);**  **DECLARE tech\_salary DECIMAL(10,2);**  **DECLARE tech\_id INT;**  **DECLARE salary\_category VARCHAR(20);**  **DECLARE tech\_cursor CURSOR FOR**  **SELECT user\_id, name, salary FROM users WHERE role = 'Technician';**  **DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;**  **OPEN tech\_cursor;**  **read\_loop: LOOP**  **FETCH tech\_cursor INTO tech\_id, tech\_name, tech\_salary;**  **IF done THEN**  **LEAVE read\_loop;**  **END IF;**  **IF tech\_salary < 2900 THEN**  **SET salary\_category = 'Low';**  **ELSEIF tech\_salary BETWEEN 2900 AND 3100 THEN**  **SET salary\_category = 'Medium';**  **ELSE**  **SET salary\_category = 'High';**  **END IF;**  **INSERT INTO technician\_reports (technician\_id, technician\_name, salary, salary\_category)**  **VALUES (tech\_id, tech\_name, tech\_salary, salary\_category);**  **END LOOP;**  **CLOSE tech\_cursor;**  **END //**  **DELIMITER ;** |

След което извикваме съответната процедура и извеждаме всички записи от новосъздадената таблица technician\_reports чрез заявката SELECT \* FROM.

|  |
| --- |
| **CALL list\_technicians\_salaries();**  **SELECT \* FROM technician\_reports;** |

Ето го и крайният резултат от процедурата, в която демонстрираме използването на курсор и новосъздадената таблица:

