Тема №5

Проект по база от данни за сервиз за битова техника

Изготвил: Стелиана Димитрова – ИС, II курс , ф.н. 4MI0700044

1. Обхват на модела. Дефиниране на задачата.

Базата от данни за сервиз за битова техника ще съхранява информация за техниците и ремонтите, извършени от тях.

В сервиза се пази следната информация за всеки ремонтиран уред: уникален идентификатор, категория уред, производител, модел, име на клиент и годината на производство, ако тя е известна (валидна е всяка година след 1900).

За техниците, работещи в сервиза за битова техника, също се пази информация. Задължително се записва техния уникален идентификатор, име, ЕГН и категории уреди, които може да ремонтира.

Един техник може да извършва много ремонти на уреди. Един уред може да бъде поправян много пъти от различен техник. За всяка поправка се пази дата и цена. За дадена категория уреди може да има няколко техници в сервиза.

2. Множества от същности и техните атрибути

Уред – уникален идентификатор, категория (пералня, телевизор…), производител, модел, име на клиент, година на производство;

Техник – уникален идентификатор, име, ЕГН, категория уред (която може да ремонтира);

Клиент – име, телефон

3. Домейн на атрибутите

Уред – уникален идентификатор: низ, категория (пералня, телевизор…): низ, производител: низ до 50 символа, модел: низ до 20 символа, име на клиент: низ, година на производство: цяло положително число по-голямо от 1900;

Техник – уникален идентификатор: низ, име: низ до 100 символа, ЕГН: низ точно 10 символа , категория уред (която може да ремонтира): низ;

Клиент – име: низ до 100 символа, телефон: низ точно 10 символа;

4. Връзки

Един техник може да извършва много ремонти на уреди. Един уред може да бъде поправян много пъти от различни техници.

За дадена категория уреди може да има няколко техници.

Един клиент може да има много уреди. Един уред е на един клиент.

5. Ограничения по единствена стойност, референтна цялостност и друг тип ограничения

Уред – уникален идентификатор: еднозначно определя уреда

Техник – уникален идентификатор: еднозначно определя техника

Клиент – телефон: еднозначно определя клиента (не може да има двама човека с един и същ номер)

6. Правила и проверки

За уреда – производител: проверка за дължината (до 50 символа)

За уреда – модел: проверка за дължината (до 20 символа)

За уреда – година на производство: проверка за числото (по-голямо от 1900)

За техника – име: проверка за дължината (до 100 символа)

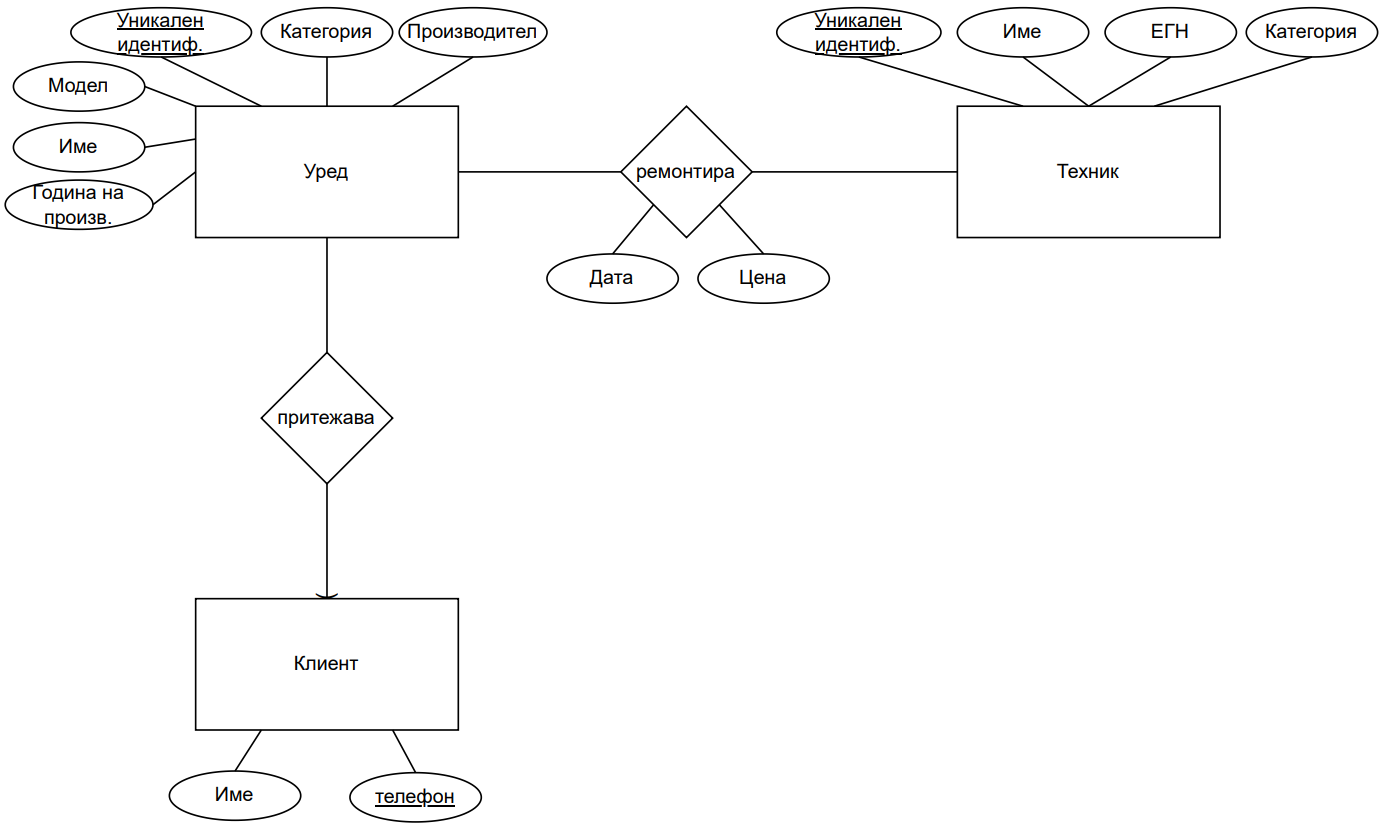
За техника – ЕГН: проверка за дължината (точно 10 символа)

За клиента – име: проверка за дължината (до 100 символа)

За клиента – телефон: проверка за дължината (точно 10 символа)

За всички положителни числа – проверка дали са по-големи от нула

7. E/R модел на данни



8. Релационен модел на данни

В E/R модела на данни няма isa йерархии, затова ще започнем с преобразуването на множествата от същности, след това връзката много – много и накрая връзката много – един. Завършваме с оптимизиране на връзката много – един.

- Преобразуване на множествата от същности:

  Appliance (name, model, applianceId, applianceCategory, manufacturer, yearOfManufacture) <- Уреди

  Technician (name, EGN, texnicianId, texnicianCategory) <- Техници

  Customer (name, phoneNumber) <- Клиенти

- Преобразуване на връзката много – много

  Repair (applianceId, texnicianId, date, price)

- Преобразуване на връзката много – един

  Owns (applianceId, phoneNumber)

- Получаваме следния релационен модел:

  Appliance (name, model, applianceId, applianceCategory, manufacturer, yearOfManufacture)

  Technician (name, EGN, texnicianId, texnicianCategory)

  Customer (name, phoneNumber)

  Repair (applianceId, texnicianId, date, price)

  Owns (applianceId, phoneNumber)

- Оптимизираме връзката много – един и получаваме:

  Appliance (name, model, applianceId, applianceCategory, manufacturer, yearOfManufacture, customerPhoneNumber)

  Technician (name, EGN, texnicianId, texnicianCategory)

  Customer (name, phoneNumber)

  Repair (applianceId, texnicianId, date, price)

  Owns (applianceId, customerPhoneNumber) – отпада - оптимизираме в Appliance

- Окончателно за схемата на базата от данни получаваме:

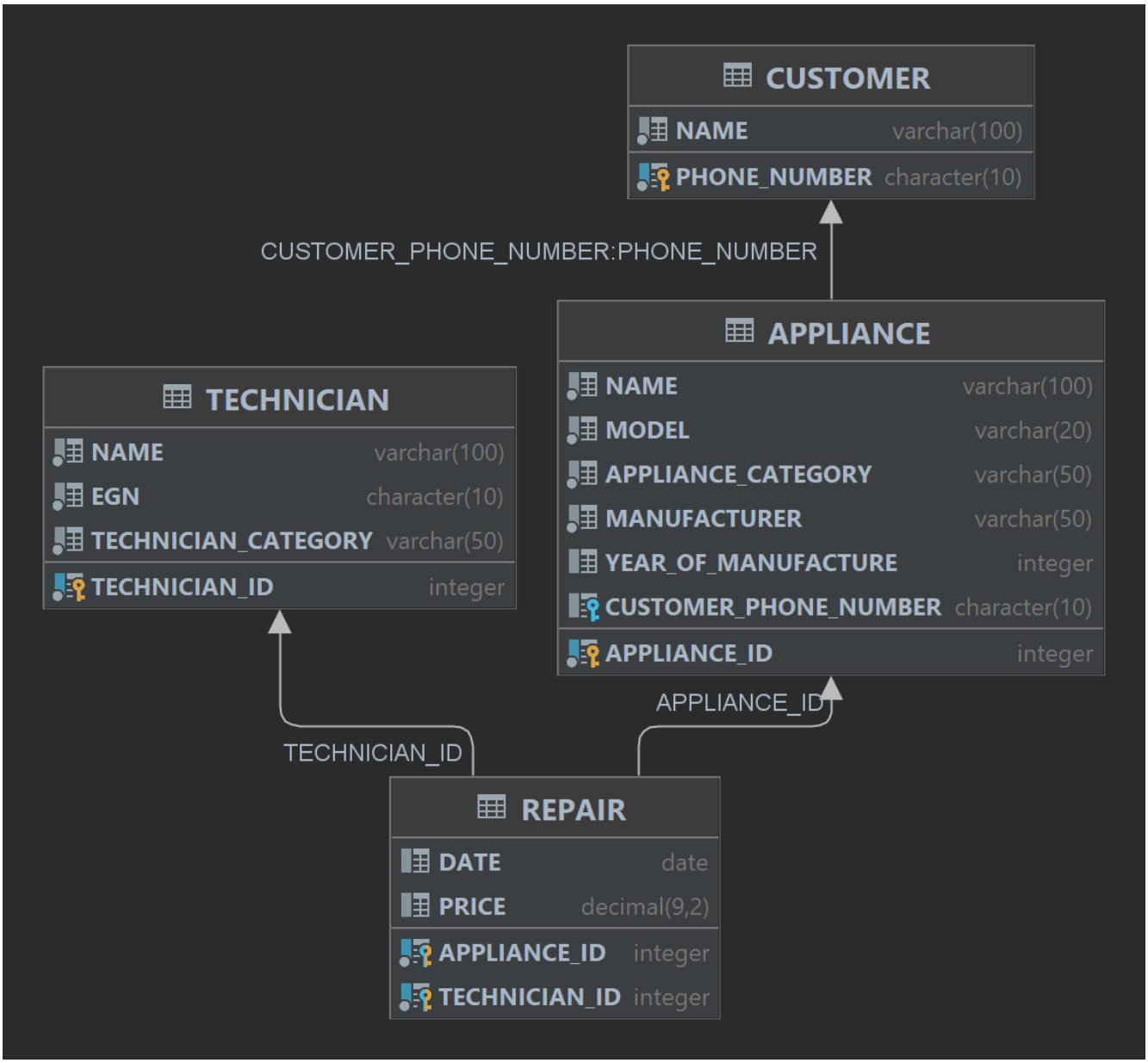
  Appliance (name, model, applianceId, applianceCategory, manufacturer, yearOfManufacture, customerPhoneNumber)

  Technician (name, EGN, texnicianId, texnicianCategory)

  Customer (name, phoneNumber)

  Repair (applianceId, texnicianId, date, price)

9. Схема на базата от данни



10. Описание на функциите

В проектът „Сервиз за битова техника“ са създадени две функции.

Първата е създадена с цел да извлече информация за уредите на един клиент. Тя е наречена F_GET_ALL_APPLIANCES() и представя имената на уредите, техните модели, дата на ремонт и името на техника, който ги е ремонтирал. Функцията приема телефонен номер на клиента (P_PHONE_NUMBER) и връща таблица с четири колони: NAME (име на уреда), MODEL (модел на уреда), DATE_OF_REPAIR (дата на ремонт) и TECHNICIAN_NAME (име на техниката). За да я извикаме използваме заявка към таблицата.

```sql
CREATE FUNCTION F_GET_ALL_APPLIANCES(P_PHONE_NUMBER CHAR(10))
RETURNS TABLE(NAME VARCHAR(100), MODEL VARCHAR(20), DATE_OF_REPAIR DATE,
TECHNICIAN_NAME VARCHAR(100))
RETURN
    SELECT A.NAME, A.MODEL, R.DATE, T.NAME
    FROM APPLIANCE A, REPAIR R, TECHNICIAN T
    WHERE A.CUSTOMER_PHONE_NUMBER = P_PHONE_NUMBER
    AND A.APPLIANCE_ID = R.APPLIANCE_ID
    AND R.TECHNICIAN_ID = T.TECHNICIAN_ID;

SELECT *
FROM TABLE(FN4MI0700044.F_GET_ALL_APPLIANCES('0899000112')) T;
```

	NAME	MODEL	DATE_OF_REPAIR	TECHNICIAN_NAME
1	HP Spectre x360 Laptop	Spectre x360	2023-03-30	Antonio Stefanov
2	Nikon D850 DSLR Camera	D850	2023-03-21	Ivan Ivanov
3	Samsung Galaxy Z Fold 3	Z Fold 3	2023-03-20	Elena Stoyanova

Втората функция се казва F_AVG_PRICE_FOR_CATEGORY(). Тя приема параметър P_CATEGORY от тип VARCHAR(50) и връща резултат от тип INT. Функцията е създадена с цел да изчисли средната цена на ремонтите за уреди от определена категория. Използва блок от BEGIN и END, в който се дефинира променлива P_AVG_PRICE от тип DECIMAL(9, 2), която ще съхранява средната цена. След това правим заявка SELECT AVG(PRICE) INTO P_AVG_PRICE, където се изчислява средната цена на ремонтите за уреди от зададената категория. Използва се подзаявка за извличане на APPLIANCE_ID на уредите, които са от желаната категория. Накрая функцията връща средната цена P_AVG_PRICE като резултат.

```sql
CREATE OR REPLACE FUNCTION F_AVG_PRICE_FOR_CATEGORY (P_CATEGORY
VARCHAR(50))
RETURNS INT
BEGIN
    DECLARE P_AVG_PRICE DECIMAL(9, 2);

    SELECT AVG(PRICE) INTO P_AVG_PRICE
    FROM REPAIR
    WHERE APPLIANCE_ID IN ( SELECT APPLIANCE_ID
                            FROM APPLIANCE
                            WHERE APPLIANCE_CATEGORY = P_CATEGORY);

    RETURN P_AVG_PRICE;
END;
```

```
VALUES FN4MI0700044.F_AVG_PRICE_FOR_CATEGORY('TV');
```

	1
1	127

11. Описание на процедурите

Първата ни процедура P_NEW_PRICE_AFTER_2020 е създадена за да увеличи цените на ремонтите за уреди, произведени след 2020 г., да изведе информация за тези ремонти, както и да покаже съобщение кои клиенти трябва да бъдат уведомени за увеличението на цените. Процедурата използва курсор за обхождане на резултатите от заявката и взима данните от три свързани таблици - APPLIANCE, REPAIR и CUSTOMER. Чрез FETCH извличаме първият ред от резултатите. След това се използва цикъл за обхождане на резултатите от курсора. Ако годината на производство на уреда е след 2020 г., цената на ремонта се увеличава с 20% и се извежда информация за ремонта, както и уведомление към клиента. Цикълът продължава с извличането на следващите редове, докато няма повече резултати. Накрая курсорът се затваря.

```
CREATE OR REPLACE PROCEDURE P_NEW_PRICE_AFTER_2020()
BEGIN
    DECLARE V_MODEL VARCHAR(30);
    DECLARE V_YEAR INT;
    DECLARE V_PRICE DECIMAL(9,2);
    DECLARE V_NAME VARCHAR(50);
    DECLARE V_PHONE_NUMBER CHAR(10);
    DECLARE SQLCODE INT;

    DECLARE C1 CURSOR WITH RETURN FOR SELECT A.MODEL,
A.YEAR_OF_MANUFACTURE, R.PRICE, C.NAME, A.CUSTOMER_PHONE_NUMBER
                                       FROM APPLIANCE A, REPAIR R,
CUSTOMER C
                                       WHERE A.CUSTOMER_PHONE_NUMBER =
C.PHONE_NUMBER
                                       AND R.APPLIANCE_ID =
A.APPLIANCE_ID;

    OPEN C1;
    FETCH C1 INTO V_MODEL, V_YEAR, V_PRICE, V_NAME, V_PHONE_NUMBER;

    WHILE SQLCODE = 0 DO
        IF V_YEAR > 2020 THEN
```

```
            SET V_PRICE = V_PRICE * 1.2;

            CALL DBMS_OUTPUT.PUT_LINE('Model - ' || V_MODEL || ', Price
of repair - ' || V_PRICE);
            CALL DBMS_OUTPUT.PUT_LINE('Notify ' || V_NAME || ',whose
phone number is ' || V_PHONE_NUMBER || ', that the price is going up
because of the new model.');
            CALL DBMS_OUTPUT.PUT_LINE('---------------------------------
--------------------------------------------------------------------------
------------------');

        END IF;

        FETCH C1 INTO V_MODEL, V_YEAR, V_PRICE, V_NAME, V_PHONE_NUMBER;
    END WHILE;

CLOSE C1;
END;

CALL FN4MI0700044.P_NEW_PRICE_AFTER_2020();
```

```
Model - R101AE, Price of repair - 98.40
Notify Nikol Stefanova,whose phone number is 0890412337, that the price is going up because of the new model.
--------------------------------------------------------------------------------------------------------------
Model - R101AE, Price of repair - 54.00
Notify Nikol Stefanova,whose phone number is 0890412337, that the price is going up because of the new model.
--------------------------------------------------------------------------------------------------------------
Model - R101AE, Price of repair - 138.00
Notify Nikol Stefanova,whose phone number is 0890412337, that the price is going up because of the new model.
--------------------------------------------------------------------------------------------------------------
Model - V11, Price of repair - 126.00
Notify Martin Georgiev,whose phone number is 0890111223, that the price is going up because of the new model.
--------------------------------------------------------------------------------------------------------------
Model - 13, Price of repair - 84.60
Notify Stanimir Petrov,whose phone number is 0892333445, that the price is going up because of the new model.
--------------------------------------------------------------------------------------------------------------
Model - XPS 13, Price of repair - 132.00
Notify Iliya Dimitrov,whose phone number is 0870111223, that the price is going up because of the new model.
--------------------------------------------------------------------------------------------------------------
Model - XPS 13, Price of repair - 90.60
Notify Iliya Dimitrov,whose phone number is 0870111223, that the price is going up because of the new model.
--------------------------------------------------------------------------------------------------------------
```

Процедурата с име P_DAYS_OFF приема параметри за брой дни, начална дата на отпуската, id на техник и връща новата дата, на която ще бъде готов уреда, чрез параметър с ключовата дума OUT. Процедурата използва курсор за извличане на резултатите от заявката за ремонти, които ще се извършат от определен техник след началната дата на отпуската. В цикъла се изчислява новата дата (P_NEW_DATE), като се

добавят дните от параметъра P_NUMBER_OF_DAYS. След което се извежда се информация за уреда и новата дата на вземането му.

```
CREATE OR REPLACE PROCEDURE P_DAYS_OFF(IN P_NUMBER_OF_DAYS INT, IN
P_FROM_DATE DATE, IN P_TECHNICIAN_ID INT, OUT P_NEW_DATE DATE)
BEGIN
    DECLARE V_MODEL VARCHAR(30);
    DECLARE V_DATE DATE;

    DECLARE SQLCODE INT;

    DECLARE C1 CURSOR WITH RETURN FOR SELECT A.MODEL, R.DATE
                                      FROM APPLIANCE A, REPAIR R
                                      WHERE R.TECHNICIAN_ID =
P_TECHNICIAN_ID
                                      AND R.DATE >= P_FROM_DATE
                                      AND A.APPLIANCE_ID =
R.APPLIANCE_ID;

    OPEN C1;
    FETCH C1 INTO V_MODEL, V_DATE;

    WHILE SQLCODE = 0 DO
        SET P_NEW_DATE = DATE(V_DATE) + P_NUMBER_OF_DAYS DAYS;

        CALL DBMS_OUTPUT.PUT_LINE(V_MODEL || ' will be ready on ' ||
P_NEW_DATE);

        FETCH C1 INTO V_MODEL, V_DATE;
    END WHILE;

CLOSE C1;
END;

BEGIN
    DECLARE NEW_DATE DATE;
    CALL FN4MI0700044.P_DAYS_OFF(20, '2023-03-01' , 141, NEW_DATE);
END;
```

```
D850 will be ready on 04/10/2023
Switch will be ready on 03/21/2023
Video Doorbell Pro will be ready on 03/30/2023
```

Последната процедура с име P_TECHNICIAN_INFO приема id на техника и извежда информацията за него, включително име, ЕГН, id, категорията, която ремонтира, и брой на техническите ремонти, които е извършил. Процедурата включва обработка на грешка при ненамерен техник. Имаме 2 заявки – първата се използва за извличане на броя ремонти на дадения техник, а втората прави проверка дали техникът съществува. Използваме EXIT HANDLER FOR NOT_FOUND, който се активира в случай, че техникът не е намерен (SQLSTATE '21000'). Процедурата извършва проверка дали техникът съществува и, ако не, генерира грешка чрез SIGNAL NOT_FOUND. Ако техникът съществува, извлича се информацията за него и се извежда в конзолата. Примери за извикване на процедурата са в края на кода, където се извикват с идентификаторите 141 и 133. В първия случай техникът съществува, а във втория - не.

```
CREATE OR REPLACE PROCEDURE P_TECHNICIAN_INFO(IN P_TECHNICIAN_ID INT)
BEGIN
    DECLARE V_NAME VARCHAR(100);
    DECLARE V_EGN CHAR(10);
    DECLARE V_CATEGORY VARCHAR(50);

    DECLARE V_ID_COLLECTION INT;
    DECLARE V_APP_COUNT INT;

    DECLARE r_error INT DEFAULT 0;
    DECLARE SQLCODE INT DEFAULT 0;
    DECLARE NOT_FOUND CONDITION FOR SQLSTATE '21000';

    DECLARE EXIT HANDLER FOR NOT_FOUND
    BEGIN
        SET r_error = SQLCODE;
        CALL DBMS_OUTPUT.PUT_LINE('ERROR OCCURS - TECHNICIAN IS NOT
FOUND: ' || r_error);
    END;

    SELECT COUNT(*) INTO V_APP_COUNT
    FROM  REPAIR R
    WHERE R.TECHNICIAN_ID = P_TECHNICIAN_ID;

    SELECT COUNT(*) INTO V_ID_COLLECTION
    FROM TECHNICIAN
    WHERE TECHNICIAN_ID = P_TECHNICIAN_ID;

    IF(V_ID_COLLECTION = 0) THEN
        SIGNAL NOT_FOUND;
    ELSE
        SELECT NAME, EGN, TECHNICIAN_CATEGORY INTO V_NAME, V_EGN,
V_CATEGORY
        FROM TECHNICIAN
        WHERE TECHNICIAN_ID = P_TECHNICIAN_ID;
```

```
        CALL DBMS_OUTPUT.PUT_LINE('TECHNICIAN NAME: ' || V_NAME);
        CALL DBMS_OUTPUT.PUT_LINE('TECHNICIAN EGN: ' || V_EGN);
        CALL DBMS_OUTPUT.PUT_LINE('TECHNICIAN ID: ' || P_TECHNICIAN_ID);
        CALL DBMS_OUTPUT.PUT_LINE('TECHNICIAN CATEGORY: ' ||
V_CATEGORY);
        CALL DBMS_OUTPUT.PUT_LINE('TECHNICAL REPAIRS: ' || V_APP_COUNT);
    END IF;

END;

CALL FN4MI0700044.P_TECHNICIAN_INFO( 141);
CALL FN4MI0700044.P_TECHNICIAN_INFO( 133);
```

```
FN4MI0700044> CALL FN4MI0700044.P_TECHNICIAN_INFO( 141)
[2024-01-21 22:51:58] completed in 143 ms
TECHNICIAN NAME: Ivan Ivanov
TECHNICIAN EGN: 8123999999
TECHNICIAN ID: 141
TECHNICIAN CATEGORY: Dishwasher
TECHNICAL REPAIRS: 5
FN4MI0700044> CALL FN4MI0700044.P_TECHNICIAN_INFO( 133)
[2024-01-21 22:51:59] completed in 36 ms
ERROR OCCURS - TECHNICIAN IS NOT FOUND: -438
```

12. Описание на тригерите

В проекта са създадени 2 тригера.

Първият е наречен TRIG_AFTER_UPDATE_REPAIR_PRICE и се изпълнява след актуализация на PRICE в таблицата REPAIR. Този тригер следи за увеличение на цената на ремонт и записва информацията в таблицата REPAIR_PRICE_INCREASE_LOG (тя е създадена преди създаването на тригера). Предназначена е за съхранение на информацията за увеличенията на цените на ремонтите. Когато новата цена (N.PRICE) е по-голяма от старата цена (O.PRICE), тригерът извършва вмъкване на запис в таблицата REPAIR_PRICE_INCREASE_LOG. Имаме 2 примера, в които се увеличава цената на ремонта за уред с APPLIANCE_ID = 280 с 50 и след това се намалява цената за уред с APPLIANCE_ID = 290 с 50. В резултат от тях виждаме, че се добавя само увеличението.

```
CREATE TABLE REPAIR_PRICE_INCREASE_LOG (
    APPLIANCE_ID INT NOT NULL,
    TECHNICIAN_ID INT NOT NULL,
```

```
    OLD_PRICE DECIMAL(9, 2) NOT NULL,
    NEW_PRICE DECIMAL(9, 2) NOT NULL
);

CREATE OR REPLACE TRIGGER TRIG_AFTER_UPDATE_REPAIR_PRICE
AFTER UPDATE OF PRICE ON REPAIR
REFERENCING NEW AS N OLD AS O
FOR EACH ROW
WHEN (N.PRICE > O.PRICE)
    INSERT INTO REPAIR_PRICE_INCREASE_LOG(APPLIANCE_ID, TECHNICIAN_ID,
OLD_PRICE, NEW_PRICE)
    VALUES (N.APPLIANCE_ID, N.TECHNICIAN_ID, O.PRICE, N.PRICE);

UPDATE REPAIR
SET PRICE = PRICE + 50
WHERE APPLIANCE_ID = 280;

UPDATE REPAIR
SET PRICE = PRICE - 50
WHERE APPLIANCE_ID = 290;

SELECT * FROM REPAIR_PRICE_INCREASE_LOG;
```

	APPLIANCE_ID	TECHNICIAN_ID	OLD_PRICE	NEW_PRICE
1	280	9	12.00	62.00
2	280	173	120.50	170.50

Вторият е TRIG_AFTER_INSERT_TECHNICIAN и се изпълнява след вмъкване на нов запис в таблицата TECHNICIAN. За всеки нов ред (текущия) тригерът извиква процедурата P_TECHNICIAN_INFO, като предоставя TECHNICIAN_ID на въведения техник като параметър. За него се извежда информация като име, ЕГН, id, категория и брой на техническите ремонти.

```
CREATE OR REPLACE TRIGGER TRIG_AFTER_INSERT_TECHNICIAN
AFTER INSERT ON TECHNICIAN
REFERENCING NEW AS N
FOR EACH ROW
BEGIN
  DECLARE V_TECHNICIAN_ID INT;

  SET V_TECHNICIAN_ID = N.TECHNICIAN_ID;
  CALL FN4MI0700044.P_TECHNICIAN_INFO(V_TECHNICIAN_ID);
END;
```

```
INSERT INTO TECHNICIAN(NAME, EGN, TECHNICIAN_CATEGORY)
VALUES ('John Doee', 6898127643, 'Washing machine');
```

```
TECHNICIAN NAME: John Doee
TECHNICIAN EGN: 6898127643
TECHNICIAN ID: 180
TECHNICIAN CATEGORY: Washing machine
TECHNICAL REPAIRS: 0
```

13. Описание на изгледите

Изгледът с име TECHNICIAN_INFO представлява виртуална таблица, в която се представя информация за техниците. Той използва функции и операции за обработка на данни, за да предостави нужната информация. След името на изгледа в скобите се задават имената на колоните, а след това чрез SELECT извличаме информацията. Възрастта на техниците се изчислява като се използва EXTRACT и CAST за извличане на годината от текущата дата и конвертиране на двуцифрената година от ЕГН към цяло число. След това двете се изваждат и получаваме годините. За колоната AVG_SALARY се използва функцията FN4MI0700044.F_AVG_PRICE_FOR_CATEGORY за изчисляване на средната цена на ремонтите за категорията на техника. Накрая чрез заявка извличаме редовете и колоните от изгледа.

```
CREATE VIEW TECHNICIAN_INFO (TECHNICIAN_NAME, TECHNICIAN_AGE, CATEGORY,
AVG_SALARY)
AS
SELECT T.NAME, EXTRACT(YEAR FROM CURRENT_DATE) - CAST(CONCAT('19',
SUBSTR(T.EGN, 1, 2)) AS INT) AS , T.TECHNICIAN_CATEGORY,
       FN4MI0700044.F_AVG_PRICE_FOR_CATEGORY(T.TECHNICIAN_CATEGORY)
FROM TECHNICIAN T;

SELECT * FROM TECHNICIAN_INFO;
```

	TECHNICIAN_NAME	TECHNICIAN_AGE	TECHNICIAN_CATEGORY	AVG_SALARY
1	Vanian Karnolski	52	Mixer	58
2	Antonio Stefanov	52	Oven	98
3	Dmitry Bodurov	49	Electric Shaver	61
4	Preslav Marinov	52	Air Conditioner	170
5	Borislav Georgiev	43	Refrigerator	112
6	Alek Maximov	52	Washing Machine	75
7	Miroslav Dimitrov	57	Laptop	110
8	Dimitar Dimitrov	59	Dishwasher	62
9	Aleksandar Dimitrov	35	Blender	93
10	Plamen Cholakov	52	Electric Kettle	73

Изгледът с име V_CUSTOMER_APPLIANCE_INFO представлява виртуална таблица, която комбинира информацията от две базови таблиците - CUSTOMER и APPLIANCE. Той предоставя информация за връзката между клиенти и техните уреди. Чрез SELECT избираме необходимите колони от двете таблици. Създаваме връзка между CUSTOMER и APPLIANCE чрез телефонния номер на клиента и клиентския телефонен номер на уреда. Накрая чрез заявка извличаме всички редове и колони от изгледа V_CUSTOMER_APPLIANCE_INFO, където категорията на уреда е 'Oven'.

```sql
CREATE VIEW V_CUSTOMER_APPLIANCE_INFO (CUSTOMER_NAME, PHONE_NUMBER,
APPLIANCE_NAME, APPLIANCE_CATEGORY, APPLIANCE_ID)
AS
SELECT C.NAME as CUSTOMER_NAME, C.PHONE_NUMBER, A.NAME as
APPLIANCE_NAME, A.APPLIANCE_CATEGORY, A.APPLIANCE_ID
FROM CUSTOMER C, APPLIANCE A
WHERE C.PHONE_NUMBER = A.CUSTOMER_PHONE_NUMBER;

SELECT *
FROM V_CUSTOMER_APPLIANCE_INFO
WHERE APPLIANCE_CATEGORY = 'Oven';
```

	CUSTOMER_NAME	PHONE_NUMBER	APPLIANCE_NAME	APPLIANCE_CATEGORY	APPLIANCE_ID
1	Steliana Dimitrova	0884312345	Frigidaire Microwave Oven	Oven	20
2	Dimitar Yordanov	0893312335	Toshiba Microwave Oven	Oven	160
3	Rumiana Dimitrova	0887712348	KitchenAid Dual Convection Countertop Toaste…	Oven	50
4	Todor Ivanov	0878899001	Breville Smart Oven	Oven	410