Софийски университет „Св. Климент Охридски“

Факултет по математика и информатика

Специалност: Информационни системи,

Курс 3, Група 1

Курсов проект

по Системи за управление на бази от данни

на тема:

Система за управление на кредити в офис

Изготвен от:

Кристина Цекова

1. Описание на множествата същности.

В системата имам общо 5 множества същности.

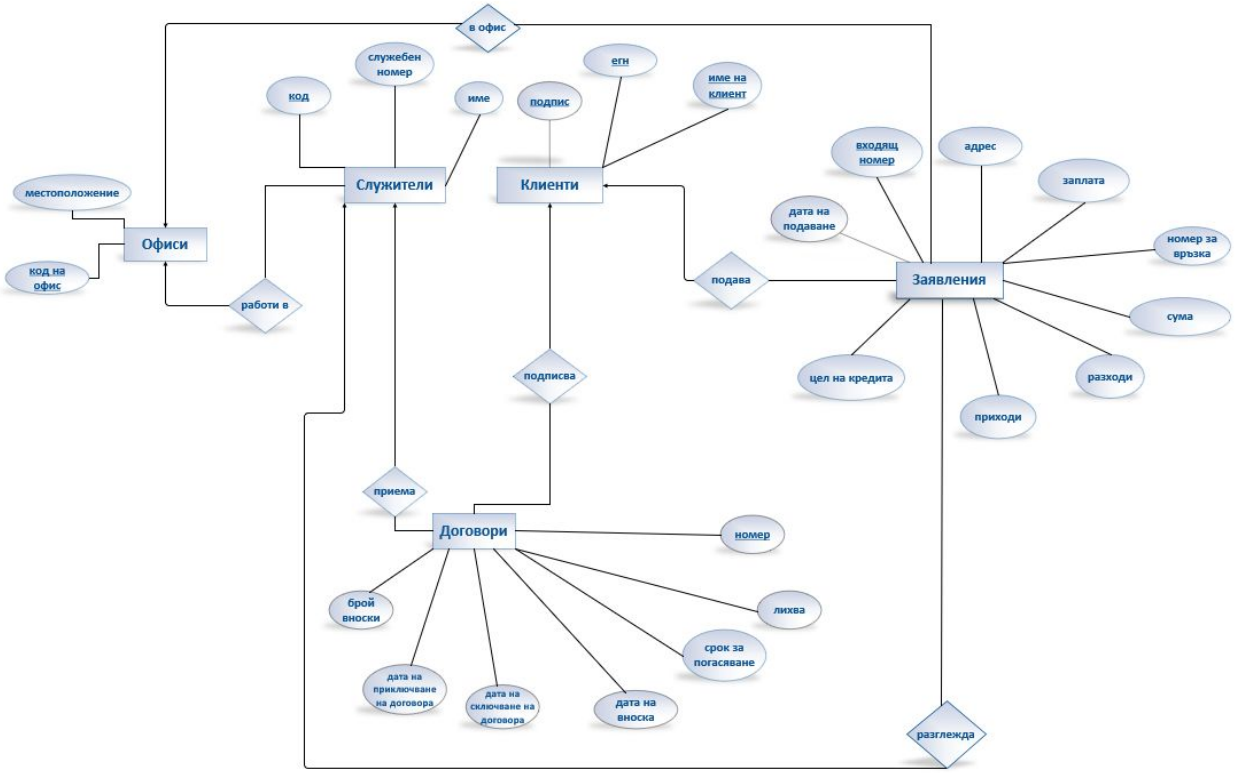
- Служители – представлява множество от същности, което описва служителите, работещи в даден офис и отговарящи за определени клиенти. След като заявление е подадено от клиент в офиса, съответния служител го разглежда и решава дали клиентът е одобрен, или не за отпускане на кредит. Ако той не е одобрен за сумата, която е пожелал, му предлага алтернативен вариант – по – малка сума. Един служител може да разглежда много заявления, но едно заявление може да се разглежда само от един служител. Освен това, договорът, сключен с този клиент, може да се приема само от този служител. Един служител работи в един офис.
    - Атрибути: име на служител, служебен номер, код
    - Ключов атрибут: **код**
    - Домейн на атрибутите: име – символен низ с дължина до 50 символа; номер – символен низ с дължина 10 символа; код – символен низ с дължина 6 символа.
- Клиенти – множество, което описва отделните клиенти. Един клиент може да подава много заявления, но едно заявление може да се попълва само от един клиент. Също така един клиент може да подписва много договори, но един договор се подписва от един клиент.
    - Атрибути: име на клиент, егн, подпис;
    - Ключови атрибути: **име на клиент, егн, подпис;**
    - Домейн на атрибутите: име – символен низ с дължина 50 символа; егн – символен низ с дължина 10 символа; подпис – символен низ с дължина 5 символа.
- Заявления – преставлява множество, описващо какво трябва да включва едно заявление, което всеки клиент, който желае да получи кредит, трябва да попълни и да предаде в съответния офис. Едно заявление може да се подава в един офис.
    - Атрибути: входящ номер, дата на подаване, заплата на клиента, адрес на клиента, номер за връзка, сума, за която кандидатства, разходи, приходи, цел на кредита.;
    - Ключов атрибут: **входящ номер;**
    - Домейн на атрибутите:
      - входящ номер: символен низ с дължина 6 символа;
      - дата на подаване: дата;
      - номер за връзка: символен низ с 10 символа;
      - заплата на клиента: положително число;

- адрес на клиента: символен низ до 50 символа (град/село, улица, блок, номер);
- сума, за която кандидатства: цяло положително число в интервала [100 – 5000];
- разходи: положително число;
- приходи: положително число;
- цел на кредита: цяло положително число в интервала [1 – 3], като 1 – за битови нужди, 2 – за лични нужди, 3 – за погасяване на друг кредит.

- Офиси – множество от същности, което описва офиса, в който работят служителите. В един офис могат да работят много служители. В даден офис постъпват много заявления.
    - Атрибути: код, местоположение;
    - Ключов атрибут: **код**;
    - Домейн на атрибутите:

      - код: символен низ с дължина 6 символа;
      - местоположение: симоволен низ с дължина до 20 символа.
- Договори – множеството, което описва какво включва договор, сключен между клиент и служител. Договорът се подписва от точно един клиент и се предава на точно един служител.
    - Атрибути: номер, дата на сключване, дата на приключване, брой вноски, дата на вноска, лихва, срок за погасяване;
    - Ключов атрибут: **номер;**
    - Домейн на атрибутите:
      - номер: символен низ с дължина 6 символа
      - дата на сключване: дата
      - дата на приключване: дата
      - брой вноски: цяло положително число
      - дата на вноска: дата
      - лихва: положително число;
      - срок за погасяване: цяло положително число в интервала [7 - 252] дни.

2. E\R диаграма на модела на БД



3. Преобразуване от E\R модел към релационен модел

Тъй като всички отношения между множествата същности са много-един или един-много, те могат да бъдат оптимизирани. След оптимизацията им релационния модел изглежда по следния начин:

1. Служители(код, име, служебен номер, код на офис)

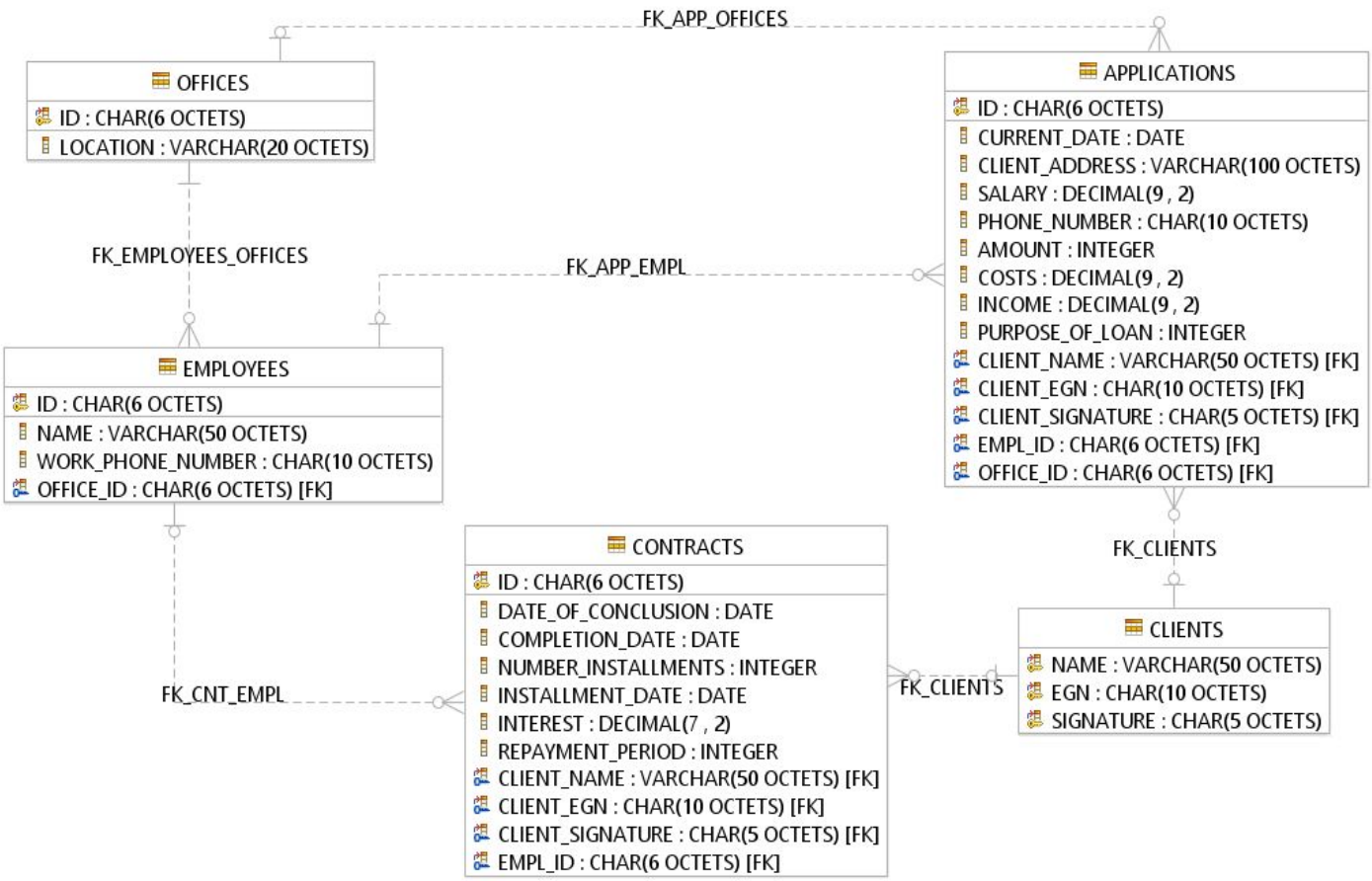
2. Клиенти(име, егн, подпис)

3. Заявления(входящ номер, дата на подаване, адрес, заплата, номер за връзка, сума, разходи, приходи, цел на кредита, име на клиент, егн на клиент, подпис на клиент, код на служител)

4. Договори (номер, дата на сключване, дата на приключване, брой вноски, дата на вноска, лихва, срок за погасяване, име на клиент, егн на клиент, подпис на клиент, код на служител)

5. Офиси(код, местоположение)

4. Картинка на релационния модел от Data studio.



5. Описание на функциите.

В проекта си съм създала общо 4 функции – 2 от тях са скаларни и 2 са таблични.

- Функцията GET_EGN(CL_SIGN CHAR(5)) е скаларна функция, която връща егн на клиент по негов подпис.

```
CREATE FUNCTION GET_EGN(CL_SIGN CHAR(5))
RETURNS VARCHAR(10)
RETURN
    SELECT CLIENT_EGN
    FROM CONTRACTS
    WHERE CLIENT_SIGNATURE = CL_SIGN;

-- Извиквам функцията
VALUES FN71852.GET_EGN('KDD78');
```

Editor | Configuration | Validation | Special Registers | Performance Metrics

Properties | SQL Results

```
1
----------
7801235465
```

- Функцията GET_LOCATION(OFF_ID CHAR(6)) е втората ми скаларна функция, която по номер на офис връща неговото местоположение.

```
-- Скаларна функция за OFFICES - връща местоположението на офиса по даден негов номер
CREATE FUNCTION GET_LOCATION(OFF_ID CHAR(6))
RETURNS VARCHAR(20)
RETURN
    SELECT LOCATION
    FROM OFFICES
    WHERE ID = OFF_ID;

-- Извиквам функцията
VALUES FN71852.GET_LOCATION('OFF004');
```

Editor | Configuration | Validation | Special Registers | Performance Metrics

Properties | SQL Results

```
1
-----
Sofia
```

- Функцията EMPL_DETAILS(E_ID CHAR(6)) е таблична функция, която по подаден номер на служител връща таблица с информация за негово име и номер на офис, за който работи.

```
-- Таблична функция за EMPLOYEES - връща таблица с информация за NAME и OFFICE_ID за служител по неговото егн
CREATE FUNCTION EMPL_DETAILS(E_ID CHAR(6))
RETURNS TABLE(E_NAME VARCHAR(50), OFF_ID CHAR(6))
RETURN
    SELECT NAME, OFFICE_ID
    FROM EMPLOYEES
    WHERE ID = E_ID;

SELECT * FROM TABLE(FN71852.EMPL_DETAILS('EMP220')) E;
```

Editor | Configuration | Validation | Special Registers | Performance Metrics

Properties | SQL Results

```
E_NAME                    OFF_ID
------------------------- ------
Kristina Tsekova Ivanova OFF004
```

- Функция CLIENT_DETAILS() не приема никакви аргументи. Тя връща таблица с информация за името, заплатата, разходите и приходите на клиент, чиято заплата е по – голяма или равна на 900лв.

```
-- Таблична функция за APPLICATIONS - връща таблица с информация за NAME, SALARY, COSTS и INCOME за всички клиенти, които имат заплата >= 900
CREATE FUNCTION CLIENT_DETAILS()
RETURNS TABLE(CL_NAME VARCHAR(50), CL_SALARY DOUBLE, CL_COSTS DOUBLE, CL_INCOME DOUBLE)
RETURN
    SELECT CLIENT_NAME, SALARY, COSTS, INCOME
    FROM APPLICATIONS
    WHERE SALARY >= 900;

SELECT * FROM TABLE(FN71852.CLIENT_DETAILS()) C;
```

Editor | Configuration | Validation | Special Registers | Performance Metrics

Properties | SQL Results

CL_NAME	CL_SALARY	CL_COSTS	CL_INCOME
Teodor Petrov Todorov	1000.0	200.0	900.0
Stoyan Penev Georgiev	900.0	350.0	500.0
Stefani Vasileva Nikolova	1100.0	250.0	850.0
Kristiana Dimitrova Dobreva	950.0	200.0	650.0
Dobrinka Georgieva Tsekova	1000.0	100.0	900.0
Nikoleta Stefanova Rizova	900.0	100.0	800.0
Georgi Petrov Georgiev	1300.0	200.0	1100.0
Gergana Biserova Mironova	1500.0	250.0	1250.0
Asen Petrov Georgiev	950.0	300.0	650.0

6. Описание на тригерите.

В проекта си имам два тригера: единият се активира след промяна на данни, а другият – преди вмъкване на данни.

- Тригерът EMP_NEW_WORK_NUMBER променя таблицата EMPLOYEES_NEW като задава нов номер на служител, чийто номер съвпада с подадения.

```
-- Копиране на таблицата
CREATE TABLE FN71852.EMPLOYEES_NEW LIKE EMPLOYEES@

-- Вмъквам данни в новата дтаблица
INSERT INTO EMPLOYEES_NEW SELECT * FROM FN71852.EMPLOYEES@

SELECT * FROM EMPLOYEES_NEW@

-- Създавам тригера
CREATE TRIGGER FN71852.EMP_NEW_WORK_NUMBER
BEFORE UPDATE ON EMPLOYEES_NEW
REFERENCING NEW AS NEW_ROW
FOR EACH ROW
    SET WORK_PHONE_NUMBER = NEW_ROW.WORK_PHONE_NUMBER@

UPDATE FN71852.EMPLOYEES_NEW
SET WORK_PHONE_NUMBER = '0987652361'
WHERE ID = 'EMP100'@
```

ID	NAME	WORK_PHONE_NUMBER	OFFICE_ID
EMP100	Ivan Georgiev Ivanov	0879876543	OFF001
EMP210	Petar Ivanov Georgiev	0888821573	OFF002
EMP315	Kalin Petrov Todorov	0887435383	OFF003

ID	NAME	WORK_PHONE_NUMBER	OFFICE_ID
EMP100	Ivan Georgiev Ivanov	0987652361	OFF001
EMP210	Petar Ivanov Georgiev	0888821573	OFF002
EMP315	Kalin Petrov Todorov	0887435383	OFF003

- Тригерът TRIG_NEW_CONTRACTS цели да изведе информация за лихвата на кредит, която се променя, по зададен номер на договор.

```
-- Копирам таблицата
CREATE TABLE CONTRACTS_NEW LIKE CONTRACTS@

-- Вмъквам данни в новата таблица
INSERT INTO CONTRACTS_NEW
SELECT * FROM CONTRACTS@

-- Създавам таблица, в която ще се пази резултатът
CREATE TABLE AUDIT_CONTRACTS(CTIME_CONTRACT TIMESTAMP, TEXT VARCHAR(500))@

SELECT * FROM AUDIT_CONTRACTS@

-- Създавам тригер, който извежда информация за лихвата на клиент
CREATE TRIGGER TRIG_NEW_CONTRACTS
    AFTER UPDATE OF INTEREST ON CONTRACTS_NEW
    REFERENCING OLD AS OLD_CONTRACT NEW AS NEW_CONTRACT
    FOR EACH ROW
    BEGIN
      DECLARE V_TEXT VARCHAR(200);
      SET V_TEXT =  ' CONTRACT_NO = ' || OLD_CONTRACT.ID
                        || ' OLD INTEREST = ' || CHAR(OLD_CONTRACT.INTEREST)
                        || ' NEW INTEREST = ' || CHAR(NEW_CONTRACT.INTEREST);
      INSERT INTO AUDIT_CONTRACTS VALUES(CURRENT_TIMESTAMP, V_TEXT);
END@

-- Първи тест
UPDATE CONTRACTS_NEW
    SET INTEREST = INTEREST - 100
    WHERE ID = 'CNT127'@

-- Втори тест
UPDATE CONTRACTS_NEW
    SET INTEREST = INTEREST - 50
    WHERE ID = 'CNT256'@
```

Резултат:

```
CTIME_CONTRACT             TEXT
-------------------------- --------------------------------------------------------------------
2020-12-16 11:53:46.80004   CONTRACT_NO = CNT127 OLD INTEREST = 1189.00 NEW INTEREST = 1089.00
2020-12-16 11:53:58.570025  CONTRACT_NO = CNT256 OLD INTEREST = 806.00 NEW INTEREST = 756.00
```

- Тригер, който извиква процедура: TRIG_APP_CLIENT

```
-- Създавам таблица, в която ще запазвам резултата от изпълнението на процедурата
CREATE TABLE FN71852.APP_RESULT(app_id CHAR(6), curr_date DATE, client_addr VARCHAR(100))@

-- Процедура, която въвежда в таблицата APP_RESULT данни за заявление с номер APP357
CREATE PROCEDURE FN71852.APP_INFO()
LANGUAGE SQL
BEGIN
    DECLARE id_app CHAR(6);
    DECLARE cur_date DATE;
    DECLARE client_address VARCHAR(100);

    DECLARE c1 CURSOR FOR SELECT ID, FN71852.APPLICATIONS.CURRENT_DATE, CLIENT_ADDRESS
                    FROM FN71852.APPLICATIONS WHERE ID = 'APP357';
    OPEN c1;

    FETCH FROM c1 INTO id_app, cur_date, client_address;
    INSERT INTO FN71852.APP_RESULT VALUES(id_app, cur_date, client_address);
END@

-- Създавам тригер, който променя адреса на клиент, чиято година от датата на подаване е >= 2017
CREATE TRIGGER FN71852.TRIG_APP_CLIENT
AFTER UPDATE ON APP_RESULT
REFERENCING NEW AS n
FOR EACH ROW
WHEN(YEAR(DATE(n.curr_date)) >= 2017)
BEGIN ATOMIC
    CALL FN71852.APP_INFO();
END@

-- Променя се адресът на заявление с номер APP357
UPDATE APP_RESULT
SET client_addr = 'Bulgaria, Mezdra, Leshtaka street, No. 5'
WHERE app_id = 'APP357'@

-- Проверка дали е направена промяна на адреса
SELECT * FROM FN71852.APP_RESULT@
```

Преди извикването на тригера:

```
APP357 2017-05-15   Bulgaria, Razgrad, Dimitur Blagoev street, No. 30
```

След извикването на тригера:

```
APP_ID CURR_DATE  CLIENT_ADDR
------ ---------- ----------------------------------------
APP357 2017-05-15 Bulgaria, Mezdra, Leshtaka street, No. 5
```

7. Описание на изгледите.

В проекта си имам три изгледа.

- Изгледът CONTRACTS_CONCLUSION връща информация за номер на договор, дата на сключване на договора, име на клиент и брой вноски на клиента, за който е изпълнено, че годината на сключване на договора е от по-голяма или равна на 2010, а също и денят на сключване на договора е между 1 и 15.

```
-- View за таблицата CONTRACTS
CREATE VIEW FN71852.CONTRACTS_CONCLUSION
AS
    SELECT ID, DATE_OF_CONCLUSION, CLIENT_NAME, NUMBER_INSTALLMENTS
    FROM CONTRACTS
    WHERE YEAR(DATE_OF_CONCLUSION) >= '2010'
    AND DAY(DATE_OF_CONCLUSION) >= '01' AND DAY(DATE_OF_CONCLUSION) <= '15';
```

Резултат:

FN71852.CONTRACTS_CONCLUSION

	ID [CHAR(6 OCTETS)]	DATE_OF_CONCLUSION [DATE]	CLIENT_NAME [VARCHAR(50 OCTETS)]	NUMBER_INSTALLMENTS [INTEGER]
1	CNT125	2012-10-08	Krasimira Atanasova Iordanova	12
2	CNT256	2013-11-10	Teodor Petrov Todorov	5
3	CNT345	2019-07-02	Bojidara Ilieva Mineva	6
4	CNT567	2020-03-12	Stefani Vasileva Nikolova	8
5	CNT127	2011-11-11	Kristiana Dimitrova Dobreva	9
6	CNT472	2017-05-15	Kostadin Petrov Kostadinov	1

- Изгледът HIGHEST_CLIENTS_COSTS съдържа информация за номер на заявление, адрес на клиент, както и неговите разходи, които са над 150лв.

```
-- View за таблицата APPLICATIONS
CREATE VIEW HIGHEST_CLIENTS_COSTS
AS
    SELECT ID, CLIENT_ADDRESS, COSTS
    FROM APPLICATIONS
    WHERE COSTS >= 150;
```

Резултат:

FN71852.HIGHEST_CLIENTS_COSTS

	ID [CHAR(6 OCTETS)]	CLIENT_ADDRESS [VARCHAR(100 OCTETS)]	COSTS [DECIMAL(9 , 2)]
1	APP036	Bulgaria, Sofia, Kolkata street, No. 9	150.00
2	APP145	Bulgaria, Burgas, Dinko Petrov street, No. 17	200.00
3	APP215	Bulgaria, Dobrich, Gavril Genov street, No. 2	200.00
4	APP178	Bulgaria, Shumen, Kozlodui street, No. 20	350.00
5	APP234	Bulgaria, Mezdra, Leshtaka street, No. 5	250.00
6	APP214	Bulgaria, Vratsa, Hadzi Dimitur street, No. ...	200.00
7	APP456	Bulgaria, Razgrad, Dinko Petrov street, No....	150.00
8	APP357	Bulgaria, Razgrad, Dimitur Blagoev street, ...	200.00
9	APP156	Bulgaria, Burgas, Zdravets street, No. 38	200.00
1...	APP289	Bulgaria, Sofia, Strupets street, No. 47	200.00
1...	APP109	Bulgaria, Vratsa, Dunav street, No. 59	250.00
1...	APP807	Bulgaria, Mezdra, Al. Stamboliiski street, N...	300.00

- Изгледът NULL_NUM_EMPL дава информация за номера и името на служител, чийто служебен телефонен номер е null.

```
-- View за таблицата EMPLOYEES
CREATE VIEW NULL_NUM_EMPL
AS
    SELECT ID, NAME
    FROM EMPLOYEES
    WHERE WORK_PHONE_NUMBER = '';
```

Резултат:

FN71852.NULL_NUM_EMPL

	ID [CHAR(6 OCTETS)]	NAME [VARCHAR(50 OCTETS)]
1	EMP220	Kristina Tsekova Ivanova
2	EMP843	Stefan Metodiev Stefanov
3	EMP345	Ivaila Georgieva Marinova

8. Описание на процедурите.

- **Процедура с курсор и входни и изходни параметри**

Най – напред си създавам таблица CONTRACTS_COPY, с която да работя и в която да променям данни, за да си запазя оригиналните данни. Добавям си ограниченията – първичен ключ и външни ключове.
Процедурата DECREASE_CLIENT_INTEREST приема един входен параметър - contract_id, и два изходни: client_name и interest.
Целта на процедурата е по подаден номер на договор, да се промени лихвата на клиента с този договор. Като ако номерът е четен, то лихвата се намалява със 100лв. Ако той е нечетен – с 20 лв.
Създавам си и курсор, който взима името на клиент и лихвата на кредита от таблицата CONTRACTS_COPY, където номерът на договора съвпада с входния параметър. След тези промени данните се записват в новата таблица.

Резултат:

```
-- Извикване на процедурата
CALL FN71852.DECREASE_CLIENT_INTEREST('CNT234', ?, ?)@
```

Editor | Configuration | Validation | Special Registers | Performance Metrics

Properties | SQL Results

```
Name         Type   Data type Value  Value (OUT)
------------ ------ --------- ------ ------------------------
CONTRACT_ID  INPUT  CHAR      CNT234
CLIENT_NAME  OUTPUT VARCHAR           Lidiya Mihaleva Velikova
INTEREST     OUTPUT DECIMAL           39.00
```

```
-- Създавам си таблица, за да не променям оригиналните данни
CREATE TABLE CONTRACTS_COPY LIKE FN71852.CONTRACTS@

-- Добавям си ключовете
ALTER TABLE CONTRACTS_COPY ADD PRIMARY KEY ID REFERENCES EMPLOYEES(ID)@
ALTER TABLE CONTRACTS_COPY ADD FOREIGN KEY (CLIENT_NAME, CLIENT_EGN, CLIENT_SIGNATURE) REFERENCES CLIENTS(NAME, EGN, SIGNATURE)@
ALTER TABLE CONTRACTS_COPY ADD FOREIGN KEY EMPL_ID REFERENCES EMPLOYEES(ID)@

-- Въвеждам данните от старата таблица в новата
INSERT INTO CONTRACTS_COPY (SELECT * FROM FN71852.CONTRACTS)@

-- Проверявам дали данните са в новата таблица
SELECT * FROM FN71852.CONTRACTS_COPY@

-- Създавам процедура, която по подаден номер на договор, извежда информация за името на клиента и неговата лихва
-- Целта й е да се намалява лихвата на клиенти в зависимост от номера на договора. Ако той е четен, лихвата се намалява със 100лв. Иначе - с 20лв.
CREATE PROCEDURE DECREASE_CLIENT_INTEREST(IN contract_id CHAR(6), OUT client_name VARCHAR(50), OUT interest DECIMAL(7, 2))
RESULT SETS 1
LANGUAGE SQL
BEGIN
    DECLare contract ANCHOR ROW CONTRACTS_COPY;

    -- С курсора взимам името и лихвата от таблицата
    DECLARE cursor1 CURSOR FOR SELECT CLIENT_NAME, INTEREST
    FROM CONTRACTS_COPY
    WHERE ID = contract_id;

    -- Проверявам дали номерът е четен
    IF(MOD(SUBSTR(contract_id, 4), 2) = 0 ) THEN
        UPDATE CONTRACTS_COPY
        SET INTEREST = INTEREST - 100 WHERE ID = contract_id;
    -- Иначе номерът е нечетен
    ELSE
        UPDATE CONTRACTS_COPY
        SET INTEREST = INTEREST - 20 WHERE ID = contract_id;
    END IF;
    OPEN cursor1;
    FETCH FROM cursor1 INTO client_name, interest;
END@
```

- **Процедура с прихващане на изключение**

Отново най – напред си създавам таблица - EMPL_ID_CHANGE, в която ще запазвам новата информация за служител.
Имам общо две променливи, които са от тип INTEGER и показват наличието на грешка.
Декларирам си и условията за съответните SQLSTATE – ове, които отговарят на двете грешки. Имам и променливи, в които се запазва резултатът.
Създавам курсор, който взима номер, номер на офис и служебен номер на служител от таблицата EMPLOYEES.
След това декларирам типовете condition handlers.
Идеята на процедурата е да се извежда информация за номера на служителя, номера на офиса, към който принадлежи и служебния му телефонен номер. Ако се появи грешка, тя ще бъде прихваната, като ако се активира UNDO handler-а, т.е. намерил е null стойност, ще се прекрати обхождането на редовете и ще се изведе резултатът до тук.

```
 -- Създавам таблица, в която ще се извежда резултатът от процедурата
CREATE TABLE FN71852.EMPL_ID_CHANGE(CTIME TIMESTAMP, MESSAGE VARCHAR(2000))@

CREATE PROCEDURE FN71852.EMPL_ITERATE()
LANGUAGE SQL
BEGIN ATOMIC -- Връща целия резултат или, ако някоя команда даде грешка, връща всичко преди нея (Заради UNDO)
    DECLARE null_value  INTEGER DEFAULT 0;
    DECLARE out_of_range INTEGER DEFAULT 0;

    -- Променливи, в които ще се запазва резултатът
    DECLARE emp_id CHAR(6) DEFAULT '';
    DECLARE emp_officeID CHAR(12) DEFAULT '';
    DECLARE emp_workPhone CHAR(10) DEFAULT 0;

    -- Декларирам условията за съответните sqlstates
    DECLARE null_not_allowed  CONDITION FOR SQLSTATE '22004';  -- null стойности не са позволени
    DECLARE out_range CONDITION FOR SQLSTATE '02000';  -- аргумент от substr е извън допустимата дължина

    -- Курсор, с който ще обхождам редовете в таблицата EMPLOYEES
    DECLARE c1 CURSOR FOR SELECT ID, OFFICE_ID, WORK_PHONE_NUMBER FROM FN71852.EMPLOYEES;

    -- Декларирам типовете condition handlers
    DECLARE UNDO HANDLER FOR null_not_allowed  SET null_value  = 1;
    DECLARE CONTINUE HANDLER FOR out_range SET out_of_range = 1;

    OPEN c1;
        loop: LOOP
            FETCH c1 INTO emp_id, emp_officeID, emp_workPhone; -- обхождам emp_id, emp_officeID, emp_workPhone
            IF null_value  = 1 OR out_of_range = 1 THEN LEAVE loop; --OR invalid_value = 1
            ELSEIF emp_id = 'EMP210' THEN ITERATE loop;
            END IF;

            -- Вмъквам в новосъздадената таблица новите стойности на emp_id, emp_officeID, emp_workPhone
            INSERT INTO FN71852.EMPL_ID_CHANGE(CTIME, MESSAGE) VALUES (CURRENT_TIMESTAMP, 'ID:' || emp_id || ' ' || 'OFFICE ID:' ||
                                emp_officeID || ' ' || 'PHONE_NUMBER:' || emp_workPhone || ' ');
        END loop;
    CLOSE c1;
END@
```

Примерен резултат:

```
-- Извиквам процедурата.
SELECT * FROM FN71852.EMPL_ID_CHANGE@
CALL FN71852.EMPL_ITERATE()@
SELECT * FROM FN71852.EMPL_ID_CHANGE@
```

Editor | Configuration | Validation | Special Registers | Performance Metrics

Properties | SQL Results

CTIME	MESSAGE
2021-01-13 15:50:46.35199	ID:EMP100 OFFICE ID:OFF001 PHONE_NUMBER:0879876543
2021-01-13 15:50:46.352375	ID:EMP315 OFFICE ID:OFF003 PHONE_NUMBER:0887435383
2021-01-13 15:50:46.352444	ID:EMP220 OFFICE ID:OFF004 PHONE_NUMBER:
2021-01-13 15:50:46.352548	ID:EMP426 OFFICE ID:OFF005 PHONE_NUMBER:0987645383
2021-01-13 15:50:46.352628	ID:EMP635 OFFICE ID:OFF006 PHONE_NUMBER:0943882157
2021-01-13 15:50:46.352702	ID:EMP843 OFFICE ID:OFF007 PHONE_NUMBER:
2021-01-13 15:50:46.352776	ID:EMP159 OFFICE ID:OFF008 PHONE_NUMBER:0889675645
2021-01-13 15:50:46.35285	ID:EMP589 OFFICE ID:OFF009 PHONE_NUMBER:0876215341
2021-01-13 15:50:46.352924	ID:EMP345 OFFICE ID:OFF010 PHONE_NUMBER:
2021-01-14 21:29:44.233621	ID:EMP100 OFFICE ID:OFF001 PHONE_NUMBER:0879876543
2021-01-14 21:29:44.233918	ID:EMP315 OFFICE ID:OFF003 PHONE_NUMBER:0887435383
2021-01-14 21:29:44.234045	ID:EMP220 OFFICE ID:OFF004 PHONE_NUMBER:

- **Процедура с курсор и while цикъл**

Най-напред си създавам три масива, в които ще пазя информация съответно за: адрес на клиент, заплатата на клиент, размера на кредита, за който кандидатства.

След това си създавам и една таблица - CLIENT_INFO, в която ще се запазва резултатът от процедурата.

Имам и курсор, който е константен и с негова помощ взимам информацията от таблицата APPLICATIONS, но само информацията за клиентите, за които целта на кредита е: DAILY NEEDS /1/.

Създавам процедурата CLIENT_APP_INFO, която не приема параметри. В началото декларирам три променливи от тип масивите и една променлива – clientApp, която сочи към ред от таблицата APPLICATIONS. Освен тези неща, имам и променлива SQLCODE, която ще следи за грешки, както и променлива appID, която да сочи към номера на заявлението.

Отварям курсора и в един цикъл въвеждам информация от главната таблица в масивите, които съм създала.

Накрая, въвеждам данните в таблицата CLIENT_INFO.

```
-- Създавам три асоциативни масива,за да изведа данни за служител.
CREATE TYPE FN71852.clientAddressArr AS VARCHAR(200) ARRAY[VARCHAR(100)]@ -- с помощта на този масив извеждам адреса на клиента
CREATE TYPE FN71852.clientSalArr AS VARCHAR(200) ARRAY[VARCHAR(100)]@     -- този масив съдържа заплатата на клиента
CREATE TYPE FN71852.clientAmountArr AS VARCHAR(200) ARRAY[VARCHAR(100)]@  -- тук пазя информация за размера на кредита

 -- Създавам таблица, в която ще се извежда резултатът от процедурата
CREATE TABLE CLIENT_INFO(CTIME TIMESTAMP, MESSAGE VARCHAR(1000))@

-- Добавям курсор, който не се променя и извежда всички данни за клиент, чиято цел на кредита е '1' - DAILY NEEDS
CREATE VARIABLE appIndex CURSOR CONSTANT (CURSOR FOR SELECT * FROM FN71852.APPLICATIONS WHERE PURPOSE_OF_LOAN = 1)@

CREATE PROCEDURE CLIENT_APP_INFO()
BEGIN
    DECLARE clientApp ANCHOR ROW FN71852.APPLICATIONS; -- clientApp сочи към ред от таблицата APPLICATIONS
    DECLARE clientAddrHash FN71852.clientAddressArr;   -- clientAddrHash е от тип clientAddressArr (асоциативния масив)
    DECLARE clientSalHash FN71852.clientSalArr;        -- clientSalHash е от тип clientSalArr (асоциативния масив)
    DECLARE clientAmountHash FN71852.clientAmountArr;  -- clientAmountHash е от тип clientAmountArr (асоциативния масив)

    DECLARE SQLCODE INT; -- SQLCODE показва наличието на грешка
    DECLARE appID ANCHOR FN71852.APPLICATIONS.ID; -- appID сочи към ID от таблицата APPLICATIONS

    OPEN appIndex;
    -- За завлението със съответен номер въвеждаме информация за име, заплата и размер на кредита на даден клиент
    FETCH appIndex INTO clientApp;
    WHILE SQLCODE = 0 DO -- докато няма грешка
        SET clientAddrHash[clientApp.ID] = clientApp.CLIENT_ADDRESS;
        SET clientSalHash[clientApp.ID] = clientApp.SALARY;
        SET clientAmountHash[clientApp.ID] = clientApp.AMOUNT;
        FETCH appIndex INTO clientApp;
    END WHILE;

    -- Извеждаме информация за името и заплатата и размер на кредита на даден клиент по номер на заявлението
    SET appID = ARRAY_FIRST(clientAddrHash);
    WHILE appID IS NOT NULL DO
        INSERT INTO FN71852.CLIENT_INFO VALUES (CURRENT_TIMESTAMP, 'ID: ' || appID || ' Address: ' || clientAddrHash[appID]
                        || ' Salary: ' || clientSalHash[appID] || ' Amount: ' || clientAmountHash[appID]);
        SET appID = ARRAY_NEXT(clientAddrHash, appID);
    END WHILE;
END@
```

Резултат:

```
-- Извиквам процедурата.
SELECT * FROM FN71852.CLIENT_INFO@
CALL FN71852.CLIENT_APP_INFO()@
SELECT * FROM FN71852.CLIENT_INFO@
```

Editor | Configuration | Validation | Special Registers | Performance Metrics

Properties | SQL Results

CTIME	MESSAGE
2021-01-14 21:30:06.135162	ID: APP021 Address: Bulgaria, Varna, Liulin street, No. 6 Salary: 800.00 Amount: 1000
2021-01-14 21:30:06.136141	ID: APP156 Address: Bulgaria, Burgas, Zdravets street, No. 38 Salary: 750.00 Amount: 400
2021-01-14 21:30:06.136212	ID: APP214 Address: Bulgaria, Vratsa, Hadzi Dimitur street, No. 89 Salary: 950.00 Amount: 2500
2021-01-14 21:30:06.136269	ID: APP215 Address: Bulgaria, Dobrich, Gavril Genov street, No. 2 Salary: 1000.00 Amount: 1500
2021-01-14 21:30:06.136371	ID: APP234 Address: Bulgaria, Mezdra, Leshtaka street, No. 5 Salary: 1100.00 Amount: 3000
2021-01-14 21:30:06.136535	ID: APP357 Address: Bulgaria, Razgrad, Dimitur Blagoev street, No. 30 Salary: 650.00 Amount: 100
2021-01-14 21:30:06.136587	ID: APP765 Address: Bulgaria, Veliko Turnovo, Georgi Damqnov street, No. 5 Salary: 800.00 Amount: 500
2021-01-14 21:30:06.136632	ID: APP807 Address: Bulgaria, Mezdra, Al. Stamboliiski street, No. 67 Salary: 950.00 Amount: 900