НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ

## МАГИСТЪРСКИ ФАКУЛТЕТ

# ДЕПАРТАМЕНТ "ИНФОРМАТИКА"

**ПРОГРАМА „Софтуерни технологии в Интернет”**



### КУРСОВА ЗАДАЧА

**КУРС: INFM111 и INFM160 Програмиране с PL/SQL**

**ТЕМА: База данни за система за управление на рент а кар**

**НА СТУДЕНТА** : Георги Иванов Георгиев

**Фак. №:** f88683

**Дата:** 13/01/2018 г.

**Преподавател:**

/ Д. Пенчева / …………………

София

1. **Описание:**

„Върчуал Рент а кар“ ООД предлага превозни средства под наем. Компанията разполага с три типа МПС - лек автомобил, джип и бус, както и три МПС с 4 вида предназначения - градско пътуване, извънградско пътуване, високо проходими и специално предназначение.

Клиентите мога да резервират МПС от всеки кръгъл час на определена дата до всеки кръгъл час на определена дата, като предоставят име, телефонен номер, ЕГН и опционално имейл.

Минимална резервация – 1 час.

Фирмата изготвя собствени отчети със статистическа и маркетингова цел.

1. **Структура на базата данни:**

Функциониране на системата за Рент а кар е изградено чрез създаването на 5 таблици. Три от тези таблици са свързани със самите МПС. Първата (VEHICLE) е предназначена за видовете превозни средства и техните характеристики. Втората и третата описват поименно и чрез сигнатура съответно типовете (VEHICLE\_TYPE) и предназначението (VEHICLE\_PURPOSE) на превозните средства. Другите две таблици съдържат информация съответно за клиентите (CLIENT) ползващи системата и резервацията (RESERVATION) на МПС.

Таблицата за превозното средство съдържа информация за идентификационния номер на въведеното МПС, тип (автомобил, джип, бус) – вторичен ключ към таблица VEHICLE\_TYPE, предназначение (градско пътуване, извънградско пътуване, високо проходими, специално предназначение) – вторичен ключ към таблица VEHICLE\_PURPOSE и екстри. Номерът на МПС е уникален и се използва за първичен ключ на таблицата.

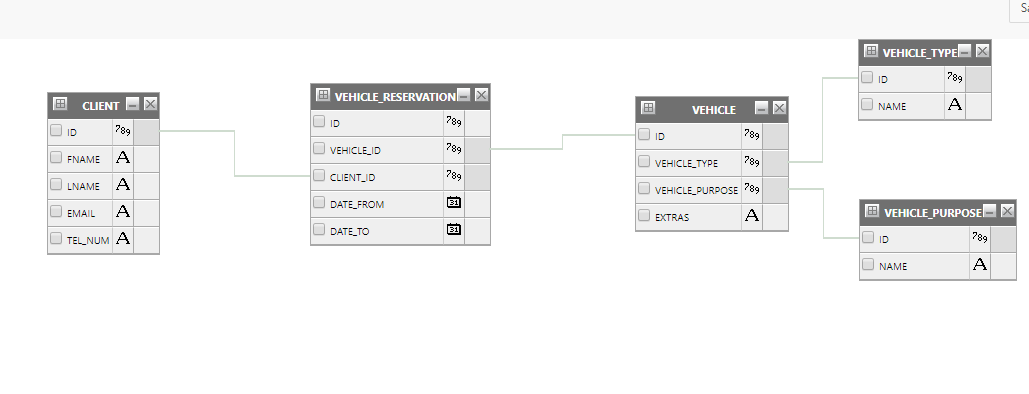
Таблицата за тип МПС съдържа информация за идентификационния номер на типа и името на типа. Номерът е уникален и се използва за първичен ключ на таблицата.

Таблицата за предназначение на МПС съдържа информация за идентификационния номер на предназначението и вида на предназначението. Номерът е уникален и се използва за първичен ключ на таблицата.

Таблицата за клиентите съдържа информация за идентификационния номер на клиента (ЕГН), име, имейл, телефонен номер. ЕГН-то на клиент е уникален и се използва за първичен ключ на таблицата.

Таблицата за резервациите съдържа информация за идентификационен номер на резервацията, номер на МПС – вторичен ключ, ЕГН на клиент, дата и час за начало на резервацията, дата и час за край на резервацията. Идентификационният номер на резервацията е уникален и се използва за първичен ключ на таблицата.

1. **Диаграма на връзките между таблиците:**



1. **Създаване на таблиците и тяхното пълнене**

Създаване на таблица VEHICLE\_TYPE:

CREATE TABLE "VEHICLE\_TYPE"

( "ID" NUMBER(3,0),

"NAME" VARCHAR2(100) NOT NULL,

CONSTRAINT "VEHICLE\_TYPE\_PK" PRIMARY KEY ("ID")

USING INDEX ENABLE

);

Примерни данни за таблица VEHICLE\_TYPE:

|  |  |
| --- | --- |
| ID | NAME |
| 0 | CAR |
| 1 | SUV |
| 2 | BUS |
|  |  |

Заявки за напълване на примерните данни:

INSERT INTO VEHICLE\_TYPE (ID, NAME) values (0, 'CAR');

INSERT INTO VEHICLE\_TYPE (ID, NAME) values (1, 'SUV');

INSERT INTO VEHICLE\_TYPE (ID, NAME) values (2, 'BUS');

Създаване на таблица VEHICLE\_PURPOSE:

CREATE TABLE "VEHICLE\_PURPOSE"

( "ID" NUMBER(3,0),

"NAME" VARCHAR2(100) NOT NULL,

CONSTRAINT "VEHICLE\_PURPOSE\_PK" PRIMARY KEY ("ID")

USING INDEX ENABLE

);

Примерни данни за таблица VEHICLE\_PURPOSE:

|  |  |
| --- | --- |
| ID | NAME |
| 0 | CITY |
| 1 | LONG DISTANCE |
| 2 | HIGHLANDER |
| 3 | SPECIAL |
|  |  |

Заявки за напълване на примерните данни:

INSERT INTO VEHICLE\_PURPOSE (ID, NAME) values (0, 'CITY');

INSERT INTO VEHICLE\_PURPOSE (ID, NAME) values (1, 'LONG DISTANCE');

INSERT INTO VEHICLE\_PURPOSE (ID, NAME) values (2, 'HIGHLANDER');

INSERT INTO VEHICLE\_PURPOSE (ID, NAME) values (3, 'SPECIAL');

Създаване на таблица VEHICLE:

CREATE TABLE "VEHICLE"

( "ID" NUMBER(5,0),

"VEHICLE\_TYPE" NUMBER NOT NULL,

"VEHICLE\_PURPOSE" NUMBER NOT NULL,

"EXTRAS" VARCHAR2(100) NOT NULL,

"PRICE" NUMBER(5,2),

CONSTRAINT "VEHICLE\_PK" PRIMARY KEY ("ID")

USING INDEX ENABLE

);

ALTER TABLE "VEHICLE" ADD CONSTRAINT "FK\_VEHICLE\_TYPE" FOREIGN KEY ("VEHICLE\_TYPE")

REFERENCES "VEHICLE\_TYPE" ("ID") ENABLE;

ALTER TABLE "VEHICLE" ADD CONSTRAINT "FK\_VEHICLE\_PURPOSE" FOREIGN KEY ("VEHICLE\_PURPOSE")

REFERENCES "VEHICLE\_PURPOSE" ("ID") ENABLE;

Примерни данни за таблица VEHICLE:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | VEHICLE\_TYPE | VEHICLE\_PURPOSE | EXTRAS | PRICE |
| 1 | 0 | 1 | CLIMA, PARKTRONIC | 20 |
| 2 | 1 | 1 | DUAL ZONE CLIMATRONIC | 37 |
| 3 | 2 | 2 | CAMERA, NAVIGATION | 49 |
| 4 | 1 | 2 | CLIMA, PANORAMIC ROOF | 17 |
| 5 | 1 | 1 | SPORTS PACKAGE | 98 |
| 6 | 1 | 1 | CHAMPAGNE | 73 |

Заявки за напълване на примерните данни:

INSERT INTO VEHICLE (ID, VEHICLE\_TYPE, VEHICLE\_PURPOSE, EXTRAS)

VALUES (\*, \*, \*, '\*');

\* specific vehicle data

UPDATE VEHICLE

SET PRICE = {0}

WHERE ID = {1}

{0} – specified vehicle price, {1}- specified vehicle ID

--Real code

DECLARE

TYPE extrasarray IS VARRAY(6) OF VARCHAR2(50);

extras extrasarray;

BEGIN

extras := extrasarray('CLIMA, PARKTRONIC', 'DUAL ZONE CLIMATRONIC', 'CAMERA, NAVIGATION', 'CLIMA, PANORAMIC ROOF', 'SPORTS PACKAGE', 'CHAMPAGNE');

FOR i IN 1..6

LOOP

INSERT INTO VEHICLE (ID, VEHICLE\_TYPE, VEHICLE\_PURPOSE, EXTRAS)

VALUES (i, ROUND(DBMS\_RANDOM.VALUE(0,2)), ROUND(DBMS\_RANDOM.VALUE(0,3)), extras(i));

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('VEHICLES INSERTED!');

END;

UPDATE VEHICLE

SET PRICE = 37

WHERE ID = 2

Създаване на таблица CLIENT:

CREATE TABLE "CLIENT"

( "ID" NUMBER(10,0),

"FNAME" VARCHAR2(20),

"LNAME" VARCHAR2(20),

"EMAIL" VARCHAR2(100),

"TEL\_NUM" VARCHAR2(100),

CONSTRAINT "CLIENT\_PK" PRIMARY KEY ("ID")

USING INDEX ENABLE

);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | FNAME | LNAME | EMAIL | TEL\_NUM |
| 1000000001 | Georgi | Georgiev | 1@random.com | 1231 |
| 1000000002 | Ivan | Kirov | 2@random.com | 1232 |
| 1000000003 | Qvor | Panchev | 3@random.com | 1233 |
| 1000000004 | Antonio | Zografsky | 4@random.com | 1234 |
| 1000000005 | Borislav | Radoslavov | 5@random.com | 1235 |
| 1000000006 | Delyan | Sarafov | 6@random.com | 1236 |

Заявки за напълване на примерните данни:

INSERT INTO CLIENT (ID, FNAME, LNAME, EMAIL, TEL\_NUM)

values (\*, '\*', '\*', '\*', '\*');

\* specific client data

--REAL CODE

DECLARE

TYPE fnamesarray IS VARRAY(6) OF VARCHAR2(20);

fnames fnamesarray;

TYPE lnamesarray IS VARRAY(6) OF VARCHAR2(20);

lnames lnamesarray;

TYPE emailsarray IS VARRAY(6) OF VARCHAR2(20);

emails emailsarray;

TYPE telarray IS VARRAY(6) OF VARCHAR2(20);

tels telarray;

BEGIN

fnames := fnamesarray('Georgi', 'Ivan', 'Qvor', 'Antonio', 'Borislav', 'Delyan');

lnames := lnamesarray('Georgiev', 'Kirov', 'Panchev', 'Zografsky', 'Radoslavov', 'Sarafov');

emails := emailsarray('1@random.com', '2@random.com', '3@random.com', '4@random.com', '5@random.com', '6@random.com');

tels := telarray('1231', '1232', '1233', '1234', '1235', '1236');

FOR i IN 1..6

LOOP

INSERT INTO CLIENT (ID, FNAME, LNAME, EMAIL, TEL\_NUM)

values (1000000000 + i, fnames(i), lnames(i), emails(i), tels(i));

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('CLIENTS INSERTED!');

END;

Създаване на таблица VEHICLE\_RESERVATION:

CREATE TABLE "VEHICLE\_RESERVATION"

( "ID" VARCHAR(40),

"VEHICLE\_ID" NUMBER(5,0),

"CLIENT\_ID" NUMBER(10,0),

"DATE\_FROM" TIMESTAMP,

"DATE\_TO" TIMESTAMP,

CONSTRAINT "VEHICLE\_RESERVATION\_PK" PRIMARY KEY ("ID")

USING INDEX ENABLE

);

ALTER TABLE "VEHICLE\_RESERVATION" ADD CONSTRAINT "FK\_VR\_V" FOREIGN KEY ("VEHICLE\_ID")

REFERENCES "VEHICLE" ("ID") ENABLE;

ALTER TABLE "VEHICLE\_RESERVATION" ADD CONSTRAINT "FK\_VR\_CL" FOREIGN KEY ("CLIENT\_ID")

REFERENCES "CLIENT" ("ID") ENABLE;

Примерни данни за таблица VEHICLE\_RESERVATION:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | VEHICLE\_ID | CLIENT\_ID | DATE\_FROM | DATE\_TO |
| 1001 | 2 | 1000000005 | 05-JAN-18 10.00.00.000000 AM | 05-JAN-18 12.00.00.000000 PM |
| 1001 | 3 | 1000000005 | 05-JAN-18 10.00.00.000000 AM | 06-JAN-18 10.00.00.000000 AM |
| 1001 | 3 | 1000000002 | 06-JAN-18 10.00.00.000000 AM | 07-JAN-18 12.00.00.000000 PM |
| 1001 | 4 | 1000000003 | 06-JAN-18 10.00.00.000000 AM | 08-JAN-18 10.00.00.000000 AM |
| 1001 | 5 | 1000000005 | 05-JAN-18 10.00.00.000000 AM | 12-JAN-18 12.00.00.000000 PM |
| 1001 | 4 | 1000000002 | 05-JAN-18 10.00.00.000000 AM | 05-JAN-18 12.00.00.000000 PM |

Заявки за напълване на примерните данни:

INSERT INTO VEHICLE\_RESERVATION (ID, VEHICLE\_ID, CLIENT\_ID, DATE\_FROM, DATE\_TO)

values (\*, \*, \*, TIMESTAMP'\*', TIMESTAMP'\*');

\* specific reservation data

--REAL CODE

DECLARE

TYPE datefromarray IS VARRAY(6) OF TIMESTAMP;

datefroms datefromarray;

TYPE datetoarray IS VARRAY(6) OF TIMESTAMP;

datetos datetoarray;

random\_id VARCHAR(40);

BEGIN

datefroms := datefromarray(TIMESTAMP'2018-01-05 10:00:00', TIMESTAMP'2018-01-05 10:00:00', TIMESTAMP'2018-01-06 10:00:00', TIMESTAMP'2018-01-06 10:00:00', TIMESTAMP'2018-01-05 10:00:00', TIMESTAMP'2018-01-05 10:00:00');

datetos := datetoarray(TIMESTAMP'2018-01-05 12:00:00', TIMESTAMP'2018-01-06 10:00:00', TIMESTAMP'2018-01-07 12:00:00', TIMESTAMP'2018-01-08 10:00:00', TIMESTAMP'2018-01-12 12:00:00', TIMESTAMP'2018-01-05 12:00:00');

FOR i IN 1..6

LOOP

random\_id := random\_uuid();

INSERT INTO VEHICLE\_RESERVATION (ID, VEHICLE\_ID, CLIENT\_ID, DATE\_FROM, DATE\_TO)

values (random\_id, ROUND(DBMS\_RANDOM.VALUE(1, 6)), ROUND(DBMS\_RANDOM.VALUE(1000000001, 1000000006)), datefroms(i), datetos(i));

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('RESERVATIONS INSERTED!');

END;

1. Задачи
2. Напишете функция, която връща броя на резервациите по дадено ID на МПС. Да се обработи грешка при въвеждане на невалидно или несъществуващо ID.

CREATE OR REPLACE FUNCTION vehicle\_res\_number(vehicleid NUMBER)

RETURN NUMBER

IS

id\_found BOOLEAN := FALSE;

resnum NUMBER;

CURSOR VEHICLES IS SELECT \* FROM VEHICLE;

INVALID\_ID EXCEPTION;

NO\_SUCH\_ID EXCEPTION;

BEGIN

BEGIN

IF vehicleid < 0 THEN

RAISE INVALID\_ID;

END IF;

FOR REC IN VEHICLES LOOP

IF REC.ID = vehicleid THEN

id\_found := TRUE;

EXIT;

END IF;

END LOOP;

IF id\_found = FALSE THEN

RAISE NO\_SUCH\_ID;

END IF;

SELECT COUNT(VEHICLE\_ID) INTO resnum

FROM VEHICLE\_RESERVATION

WHERE VEHICLE\_ID = vehicleid;

EXCEPTION

WHEN INVALID\_ID THEN

DBMS\_OUTPUT.PUT\_LINE('Vehicle id is invalid!');

WHEN NO\_SUCH\_ID THEN

DBMS\_OUTPUT.PUT\_LINE('Vehicle id does not exist!');

END;

RETURN resnum;

END;

Примерно изпълнение:

DECLARE

rescount NUMBER;

BEGIN

rescount := vehicle\_res\_number(13);

DBMS\_OUTPUT.PUT\_LINE(rescount);

END;

-- Vehicle id does not exist!

DECLARE

rescount NUMBER;

BEGIN

rescount := vehicle\_res\_number(-1);

DBMS\_OUTPUT.PUT\_LINE(rescount);

END;

-- Vehicle id is invalid!

DECLARE

rescount NUMBER;

BEGIN

rescount := vehicle\_res\_number(3);

DBMS\_OUTPUT.PUT\_LINE(rescount);

END;

-- 2

1. Напишете процедура, която показва дали в даден интервал определено МПС е заето. Обработете грешка свързана с коректността на параметрите, а именно началният период на търсене трябва да е преди крайния.

CREATE OR REPLACE PROCEDURE is\_vehicle\_reserved(vehicleid NUMBER, starttime TIMESTAMP, endtime TIMESTAMP)

IS

counter NUMBER;

INVALID\_INPUT\_PARAMETERS EXCEPTION;

BEGIN

IF starttime >= endtime THEN

RAISE INVALID\_INPUT\_PARAMETERS;

END IF;

SELECT COUNT(\*) INTO counter FROM VEHICLE\_RESERVATION

WHERE VEHICLE\_ID = vehicleid

AND ((DATE\_FROM = starttime AND DATE\_TO = endtime) OR

(DATE\_FROM > starttime AND DATE\_TO < endtime) OR

(DATE\_FROM <= starttime AND DATE\_TO > starttime AND DATE\_TO < endtime) OR

(DATE\_FROM >= starttime AND DATE\_FROM < endtime AND DATE\_TO >= endtime) OR

(DATE\_FROM > starttime AND DATE\_TO > endtime));

IF counter != 0 THEN

DBMS\_OUTPUT.PUT\_LINE(vehicleid || ' is already reserved');

ELSE

DBMS\_OUTPUT.PUT\_LINE(vehicleid || ' is free');

END IF;

EXCEPTION

WHEN INVALID\_INPUT\_PARAMETERS THEN

DBMS\_OUTPUT.PUT\_LINE('Invalid input parameters');

END;

Примерно изпълнение:

BEGIN

is\_vehicle\_reserved(2, '05-JAN-18 10.00', '05-JAN-18 11.00');

END;

-- 2 is already reserved

BEGIN

is\_vehicle\_reserved(2, '05-JAN-18 10.00', '05-JAN-18 10.00');

END;

-- Invalid input parameters

1. Напишете процедура, която извежда името на клиента с най-много резервации и броят на резервациите му.

CREATE OR REPLACE PROCEDURE max\_res\_client

IS

counter NUMBER := 0;

currRes NUMBER;

cl\_name CLIENT.FNAME%TYPE;

CURSOR CLS IS SELECT \* FROM CLIENT;

BEGIN

FOR rec IN CLS LOOP

SELECT COUNT(CLIENT\_ID) INTO currRes FROM VEHICLE\_RESERVATION WHERE CLIENT\_ID = rec.ID;

IF currRes > counter THEN

counter := currRes;

cl\_name := rec.FNAME;

END IF;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Client ' || cl\_name || ' has reserved the most. Total number of reservation is ' || counter);

END;

Примерно изпълнение:

BEGIN

max\_res\_client();

END;

-- Client Borislav has reserved the most. Total number of reservation is 3

1. Напишете тригер, който генерира произволен UUID, като ID на резервация

CREATE or REPLACE FUNCTION random\_uuid RETURN VARCHAR2

IS

v\_uuid VARCHAR2(40);

BEGIN

SELECT regexp\_replace(rawtohex(sys\_guid()), '([A-F0-9]{8})([A-F0-9]{4})([A-F0-9]{4})([A-F0-9]{4})([A-F0-9]{12})', '\1-\2-\3-\4-\5') INTO v\_uuid FROM dual;

RETURN v\_uuid;

END;

CREATE OR REPLACE TRIGGER gen\_res\_id

BEFORE INSERT

ON VEHICLE\_RESERVATION

FOR EACH ROW

DECLARE

random\_id VARCHAR2(40);

BEGIN

random\_id := random\_uuid();

:NEW.ID := random\_id;

END;

Примерно изпълнение:

INSERT INTO VEHICLE\_RESERVATION (VEHICLE\_ID, CLIENT\_ID, DATE\_FROM, DATE\_TO)

VALUES (1, 1000000001, '07-JAN-18 10.00', '09-JAN-18 10.00');

Проверка на новия запис:

SELECT \* FROM VEHICLE\_RESERVATION;

1. Напишете функция, която приема два параметъра от тип VEHICLE\_TYPE и връща кой от двата е предпочитан повече спрямо направените резервации и с колко. Обработете грешка в случай на грешно въведен тип. Типът се въвежда по име.

--Спомагателна функция

CREATE OR REPLACE FUNCTION v\_type\_exists(v\_type VARCHAR2)

RETURN BOOLEAN

IS

v\_type\_num NUMBER := 0;

res BOOLEAN := FALSE;

BEGIN

SELECT COUNT(NAME) INTO v\_type\_num FROM VEHICLE\_TYPE WHERE NAME = v\_type;

IF v\_type\_num > 0 THEN

res := TRUE;

DBMS\_OUTPUT.PUT\_LINE('FOUND');

END IF;

RETURN res;

END;

--Примерно изпълнение:

DECLARE

res BOOLEAN;

BEGIN

res := v\_type\_exists('SUV');

END;

--Спомагателна функция

CREATE OR REPLACE FUNCTION v\_type\_counter(v\_type\_name VARCHAR2)

RETURN NUMBER

IS

res NUMBER := 0;

BEGIN

SELECT COUNT(vr.ID) INTO res

FROM VEHICLE\_RESERVATION vr

INNER JOIN VEHICLE v ON vr.VEHICLE\_ID = v.ID

INNER JOIN VEHICLE\_TYPE vt ON v.VEHICLE\_TYPE = vt.ID

WHERE vt.NAME = v\_type\_name;

RETURN res;

END;

--Примерно изпълнение:

BEGIN

DBMS\_OUTPUT.PUT\_LINE(v\_type\_counter('CAR'));

END;

--Главна функция

CREATE OR REPLACE FUNCTION preferred\_vehichle\_type (v\_type\_one VARCHAR2, v\_type\_two VARCHAR2)

RETURN VARCHAR2

IS

one\_found BOOLEAN := FALSE;

two\_found BOOLEAN := FALSE;

count\_one NUMBER := 0;

count\_two NUMBER := 0;

dif NUMBER := 0;

res VARCHAR2(100) := '';

BEGIN

one\_found := v\_type\_exists(v\_type\_one);

two\_found := v\_type\_exists(v\_type\_two);

IF one\_found = FALSE OR two\_found = FALSE THEN

RETURN 'Invalid vehicle type entered!';

END IF;

count\_one := v\_type\_counter(v\_type\_one);

count\_two := v\_type\_counter(v\_type\_two);

IF count\_one > count\_two THEN

dif := count\_one - count\_two;

res := 'Type ' || v\_type\_one || ' has ' || dif || ' more reservations than type ' || v\_type\_two;

ELSIF count\_one < count\_two THEN

dif := count\_two - count\_one;

res := 'Type ' || v\_type\_two || ' has ' || dif || ' more reservations than type ' || v\_type\_one;

ELSIF count\_one = count\_two THEN

res := 'Types are equally preferred!';

END IF;

RETURN res;

END;

--Примерно изпълнение

BEGIN

DBMS\_OUTPUT.PUT\_LINE(preferred\_vehichle\_type('SUV', 'CAR'));

END;

BEGIN

DBMS\_OUTPUT.PUT\_LINE(preferred\_vehichle\_type('CAR','SUV'));

END;

BEGIN

DBMS\_OUTPUT.PUT\_LINE(preferred\_vehichle\_type('SUV','CAR'));

END;

BEGIN

DBMS\_OUTPUT.PUT\_LINE(preferred\_vehichle\_type('CAR','A'));

END;

1. Да се напише процедура, която резервира МПС. Да се обработят изключения за невалидно въведен интервал и зает ресурс. Използвайте процедурата от задача 2 като я адаптирате към функция.

CREATE OR REPLACE FUNCTION is\_v\_reserved(vehicleid NUMBER, starttime TIMESTAMP, endtime TIMESTAMP)

RETURN BOOLEAN

IS

res BOOLEAN := FALSE;

counter NUMBER;

BEGIN

IF starttime <= endtime THEN

SELECT COUNT(\*) INTO counter FROM VEHICLE\_RESERVATION

WHERE VEHICLE\_ID = vehicleid

AND ((DATE\_FROM = starttime AND DATE\_TO = endtime) OR

(DATE\_FROM > starttime AND DATE\_TO < endtime) OR

(DATE\_FROM <= starttime AND DATE\_TO > starttime AND DATE\_TO < endtime) OR

(DATE\_FROM >= starttime AND DATE\_FROM < endtime AND DATE\_TO >= endtime) OR

(DATE\_FROM > starttime AND DATE\_TO > endtime));

IF counter != 0 THEN

DBMS\_OUTPUT.PUT\_LINE(vehicleid || ' is already reserved');

res := TRUE;

ELSE

DBMS\_OUTPUT.PUT\_LINE(vehicleid || ' is free');

res := FALSE;

END IF;

ELSE

DBMS\_OUTPUT.PUT\_LINE('Invalid input parameters');

res := FALSE;

END IF;

RETURN res;

END;

CREATE OR REPLACE PROCEDURE reserve\_vehicle(v\_id NUMBER, c\_id NUMBER, d\_from TIMESTAMP, d\_to TIMESTAMP)

IS

v\_reserved BOOLEAN := FALSE;

BEGIN

v\_reserved := is\_v\_reserved(v\_id, d\_from, d\_to);

IF v\_reserved = FALSE THEN

INSERT INTO VEHICLE\_RESERVATION (VEHICLE\_ID, CLIENT\_ID, DATE\_FROM, DATE\_TO) VALUES (v\_id, c\_id, d\_from, d\_to);

DBMS\_OUTPUT.PUT\_LINE('Insert successful!');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Exiting!');

END IF;

END;

--Примерно изпълнение:

BEGIN

reserve\_vehicle(3, 1000000005, '08-JAN-18 10.00', '09-JAN-18 12.00');

END;

1. Напишете процедура, която връща най-резервираният тип МПС на даден клиент.

CREATE OR REPLACE PROCEDURE most\_reserved\_type\_by\_client(c\_id NUMBER)

IS

m\_type NUMBER := 0;

counter NUMBER := 0;

currRecRes Number := 0;

c\_name VARCHAR2(100);

v\_name VARCHAR2(100);

CURSOR reserv IS SELECT vr.CLIENT\_ID, v.VEHICLE\_TYPE FROM VEHICLE\_RESERVATION vr INNER JOIN VEHICLE v ON v.ID = vr.VEHICLE\_ID WHERE CLIENT\_ID = c\_id;

BEGIN

FOR rec IN reserv LOOP

SELECT COUNT(vr.ID) INTO currRecRes

FROM VEHICLE\_RESERVATION vr

INNER JOIN VEHICLE v ON v.ID = vr.VEHICLE\_ID

WHERE vr.CLIENT\_ID = rec.CLIENT\_ID

AND v.VEHICLE\_TYPE = rec.VEHICLE\_TYPE;

IF counter < currRecRes THEN

counter := currRecRes;

m\_type := rec.VEHICLE\_TYPE;

END IF;

END LOOP;

SELECT NAME INTO v\_name

FROM VEHICLE\_TYPE

WHERE ID = m\_type;

SELECT FNAME INTO c\_name

FROM CLIENT

WHERE ID = c\_id;

DBMS\_OUTPUT.PUT\_LINE('Vehicle type ' || v\_name || ' is most reserved by client ' || c\_name);

END;

--Примерно изпълнение:

BEGIN

most\_reserved\_type\_by\_client(1000000002);

END;

1. Да се напише пакет, който съдържа всички създадени функции и процедури до момента.

CREATE OR REPLACE PACKAGE MANAGE\_VEHICLE\_RESERVATION AS

FUNCTION vehicle\_res\_number(vehicleid NUMBER) RETURN NUMBER;

PROCEDURE is\_vehicle\_reserved(vehicleid NUMBER, starttime TIMESTAMP, endtime TIMESTAMP);

PROCEDURE max\_res\_client;

FUNCTION random\_uuid RETURN VARCHAR2;

FUNCTION v\_type\_exists(v\_type VARCHAR2) RETURN BOOLEAN;

FUNCTION v\_type\_counter(v\_type\_name VARCHAR2) RETURN NUMBER;

FUNCTION preferred\_vehichle\_type (v\_type\_one VARCHAR2, v\_type\_two VARCHAR2) RETURN VARCHAR2;

FUNCTION is\_v\_reserved(vehicleid NUMBER, starttime TIMESTAMP, endtime TIMESTAMP) RETURN BOOLEAN;

PROCEDURE reserve\_vehicle(v\_id NUMBER, c\_id NUMBER, d\_from TIMESTAMP, d\_to TIMESTAMP);

PROCEDURE most\_reserved\_type\_by\_client(c\_id NUMBER);

END MANAGE\_VEHICLE\_RESERVATION;

CREATE OR REPLACE PACKAGE BODY MANAGE\_VEHICLE\_RESERVATION AS

FUNCTION vehicle\_res\_number(vehicleid NUMBER)

RETURN NUMBER

IS

id\_found BOOLEAN := FALSE;

resnum NUMBER;

CURSOR VEHICLES IS SELECT \* FROM VEHICLE;

INVALID\_ID EXCEPTION;

NO\_SUCH\_ID EXCEPTION;

BEGIN

BEGIN

IF vehicleid < 0 THEN

RAISE INVALID\_ID;

END IF;

FOR REC IN VEHICLES LOOP

IF REC.ID = vehicleid THEN

id\_found := TRUE;

EXIT;

END IF;

END LOOP;

IF id\_found = FALSE THEN

RAISE NO\_SUCH\_ID;

END IF;

SELECT COUNT(VEHICLE\_ID) INTO resnum

FROM VEHICLE\_RESERVATION

WHERE VEHICLE\_ID = vehicleid;

EXCEPTION

WHEN INVALID\_ID THEN

DBMS\_OUTPUT.PUT\_LINE('Vehicle id is invalid!');

WHEN NO\_SUCH\_ID THEN

DBMS\_OUTPUT.PUT\_LINE('Vehicle id does not exist!');

END;

RETURN resnum;

END;

PROCEDURE is\_vehicle\_reserved(vehicleid NUMBER, starttime TIMESTAMP, endtime TIMESTAMP)

IS

counter NUMBER;

INVALID\_INPUT\_PARAMETERS EXCEPTION;

BEGIN

IF starttime >= endtime THEN

RAISE INVALID\_INPUT\_PARAMETERS;

END IF;

SELECT COUNT(\*) INTO counter FROM VEHICLE\_RESERVATION

WHERE VEHICLE\_ID = vehicleid

AND ((DATE\_FROM = starttime AND DATE\_TO = endtime) OR

(DATE\_FROM > starttime AND DATE\_TO < endtime) OR

(DATE\_FROM <= starttime AND DATE\_TO > starttime AND DATE\_TO < endtime) OR

(DATE\_FROM >= starttime AND DATE\_FROM < endtime AND DATE\_TO >= endtime) OR

(DATE\_FROM > starttime AND DATE\_TO > endtime));

IF counter != 0 THEN

DBMS\_OUTPUT.PUT\_LINE(vehicleid || ' is already reserved');

ELSE

DBMS\_OUTPUT.PUT\_LINE(vehicleid || ' is free');

END IF;

EXCEPTION

WHEN INVALID\_INPUT\_PARAMETERS THEN

DBMS\_OUTPUT.PUT\_LINE('Invalid input parameters');

END;

PROCEDURE max\_res\_client

IS

counter NUMBER := 0;

currRes NUMBER;

cl\_name CLIENT.FNAME%TYPE;

CURSOR CLS IS SELECT \* FROM CLIENT;

BEGIN

FOR rec IN CLS LOOP

SELECT COUNT(CLIENT\_ID) INTO currRes FROM VEHICLE\_RESERVATION WHERE CLIENT\_ID = rec.ID;

IF currRes > counter THEN

counter := currRes;

cl\_name := rec.FNAME;

END IF;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Client ' || cl\_name || ' has reserved the most. Total number of reservation is ' || counter);

END;

FUNCTION random\_uuid

RETURN VARCHAR2

IS

v\_uuid VARCHAR2(40);

BEGIN

SELECT regexp\_replace(rawtohex(sys\_guid()), '([A-F0-9]{8})([A-F0-9]{4})([A-F0-9]{4})([A-F0-9]{4})([A-F0-9]{12})', '\1-\2-\3-\4-\5') INTO v\_uuid FROM dual;

RETURN v\_uuid;

END;

FUNCTION v\_type\_exists(v\_type VARCHAR2)

RETURN BOOLEAN

IS

v\_type\_num NUMBER := 0;

res BOOLEAN := FALSE;

BEGIN

SELECT COUNT(NAME) INTO v\_type\_num FROM VEHICLE\_TYPE WHERE NAME = v\_type;

IF v\_type\_num > 0 THEN

res := TRUE;

DBMS\_OUTPUT.PUT\_LINE('FOUND');

END IF;

RETURN res;

END;

FUNCTION v\_type\_counter(v\_type\_name VARCHAR2)

RETURN NUMBER

IS

res NUMBER := 0;

BEGIN

SELECT COUNT(vr.ID) INTO res

FROM VEHICLE\_RESERVATION vr

INNER JOIN VEHICLE v ON vr.VEHICLE\_ID = v.ID

INNER JOIN VEHICLE\_TYPE vt ON v.VEHICLE\_TYPE = vt.ID

WHERE vt.NAME = v\_type\_name;

RETURN res;

END;

FUNCTION preferred\_vehichle\_type (v\_type\_one VARCHAR2, v\_type\_two VARCHAR2)

RETURN VARCHAR2

IS

one\_found BOOLEAN := FALSE;

two\_found BOOLEAN := FALSE;

count\_one NUMBER := 0;

count\_two NUMBER := 0;

dif NUMBER := 0;

res VARCHAR2(100) := '';

BEGIN

one\_found := v\_type\_exists(v\_type\_one);

two\_found := v\_type\_exists(v\_type\_two);

IF one\_found = FALSE OR two\_found = FALSE THEN

RETURN 'Invalid vehicle type entered!';

END IF;

count\_one := v\_type\_counter(v\_type\_one);

count\_two := v\_type\_counter(v\_type\_two);

IF count\_one > count\_two THEN

dif := count\_one - count\_two;

res := 'Type ' || v\_type\_one || ' has ' || dif || ' more reservations than type ' || v\_type\_two;

ELSIF count\_one < count\_two THEN

dif := count\_two - count\_one;

res := 'Type ' || v\_type\_two || ' has ' || dif || ' more reservations than type ' || v\_type\_one;

ELSIF count\_one = count\_two THEN

res := 'Types are equally preferred!';

END IF;

RETURN res;

END;

FUNCTION is\_v\_reserved(vehicleid NUMBER, starttime TIMESTAMP, endtime TIMESTAMP)

RETURN BOOLEAN

IS

res BOOLEAN := FALSE;

counter NUMBER;

BEGIN

IF starttime <= endtime THEN

SELECT COUNT(\*) INTO counter FROM VEHICLE\_RESERVATION

WHERE VEHICLE\_ID = vehicleid

AND ((DATE\_FROM = starttime AND DATE\_TO = endtime) OR

(DATE\_FROM > starttime AND DATE\_TO < endtime) OR

(DATE\_FROM <= starttime AND DATE\_TO > starttime AND DATE\_TO < endtime) OR

(DATE\_FROM >= starttime AND DATE\_FROM < endtime AND DATE\_TO >= endtime) OR

(DATE\_FROM > starttime AND DATE\_TO > endtime));

IF counter != 0 THEN

DBMS\_OUTPUT.PUT\_LINE(vehicleid || ' is already reserved');

res := TRUE;

ELSE

DBMS\_OUTPUT.PUT\_LINE(vehicleid || ' is free');

res := FALSE;

END IF;

ELSE

DBMS\_OUTPUT.PUT\_LINE('Invalid input parameters');

res := FALSE;

END IF;

RETURN res;

END;

PROCEDURE reserve\_vehicle(v\_id NUMBER, c\_id NUMBER, d\_from TIMESTAMP, d\_to TIMESTAMP)

IS

v\_reserved BOOLEAN := FALSE;

BEGIN

v\_reserved := is\_v\_reserved(v\_id, d\_from, d\_to);

IF v\_reserved = FALSE THEN

INSERT INTO VEHICLE\_RESERVATION (VEHICLE\_ID, CLIENT\_ID, DATE\_FROM, DATE\_TO) VALUES (v\_id, c\_id, d\_from, d\_to);

DBMS\_OUTPUT.PUT\_LINE('Insert successful!');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Exiting!');

END IF;

END;

PROCEDURE most\_reserved\_type\_by\_client(c\_id NUMBER)

IS

m\_type NUMBER := 0;

counter NUMBER := 0;

currRecRes Number := 0;

c\_name VARCHAR2(100);

v\_name VARCHAR2(100);

CURSOR reserv IS SELECT vr.CLIENT\_ID, v.VEHICLE\_TYPE FROM VEHICLE\_RESERVATION vr INNER JOIN VEHICLE v ON v.ID = vr.VEHICLE\_ID WHERE CLIENT\_ID = c\_id;

BEGIN

FOR rec IN reserv LOOP

SELECT COUNT(vr.ID) INTO currRecRes

FROM VEHICLE\_RESERVATION vr

INNER JOIN VEHICLE v ON v.ID = vr.VEHICLE\_ID

WHERE vr.CLIENT\_ID = rec.CLIENT\_ID

AND v.VEHICLE\_TYPE = rec.VEHICLE\_TYPE;

IF counter < currRecRes THEN

counter := currRecRes;

m\_type := rec.VEHICLE\_TYPE;

END IF;

END LOOP;

SELECT NAME INTO v\_name

FROM VEHICLE\_TYPE

WHERE ID = m\_type;

SELECT FNAME INTO c\_name

FROM CLIENT

WHERE ID = c\_id;

DBMS\_OUTPUT.PUT\_LINE('Vehicle type ' || v\_name || ' is most reserved by client ' || c\_name);

END;

END MANAGE\_VEHICLE\_RESERVATION;

BEGIN

MANAGE\_VEHICLE\_RESERVATION.max\_res\_client;

END;

1. Допълнителна информация

Кодовете към проекта могат да бъдат намерени на следната гитхъб страница:

<https://github.com/Georgegig/nbu-rent-a-car>