PL/SQL feladat

Adatbázis rendszerek 2

Szabó Larion

NSW74Y

2022

Tartalomjegyzék

1. fejezet

A feladat

A feladat egy olyan adatbázis rendszer létrehozása, amely nemcsak a felhasználók, tulajdonosok és autóik tárolására alkalmas, hanem üzleti logika is található benne, melyet PL/SQL triggerek és tárolt eljárások segítségével implementálunk.

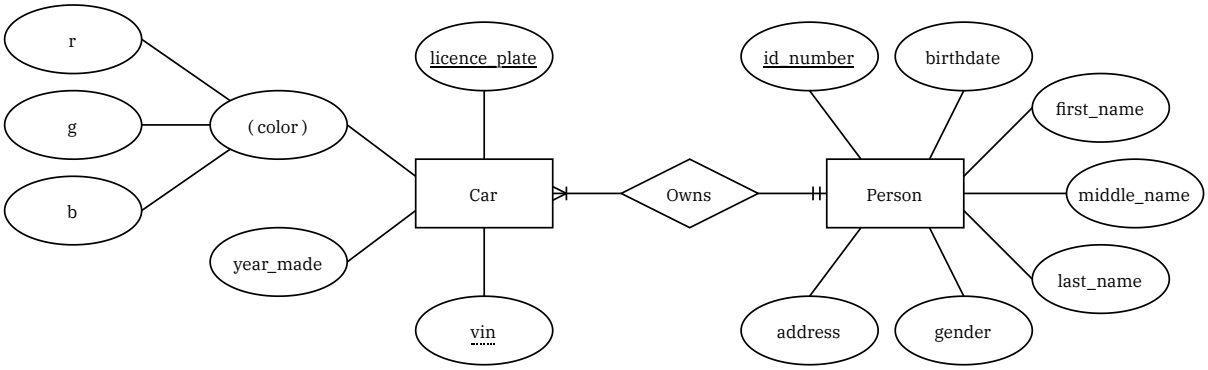
Az alkalmazáshoz szükséges az adminisztrátoroknak biztosítani, hogy saját felhasználói fiókjuk és jelszavuk legyen, utóbbi pedig nem egyszerű szövegként kerül tárolásra.

- A felhasználónév
  - Ez szolgál az egyedi azonosítóként
- A jelszó
  - Nem egyszerű szövegként tárolt
  - Minden felhasználó jelszava SHA-512-es hash algoritmussal kerül titkosításra. Így az nem visszafejthető az adatbázis ismeretekor sem.

Az adatbázis tartalmaz két másik táblát is. Autó tulajdonosokat és az autóikat.

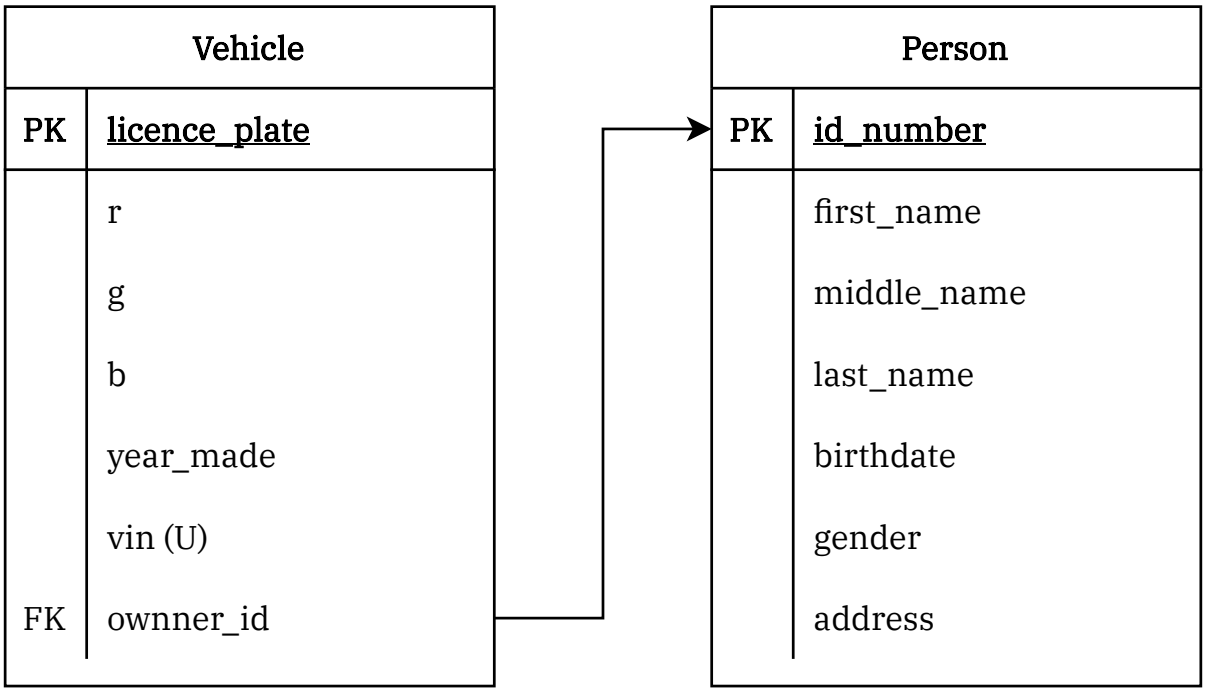
- Car - Autók
  - licence_plate - rendszám
  - color - szín
    - r - vörös (0-255)
    - g - zöld (0-255)
    - b - kék (0-255)
  - year_made - gyártás éve
  - vin - alvázszám
- Person - Tulaj
  - id_number - igazolványszám
  - first_name - keresztnév
  - middle_name - harmadik név
  - last_name - vezetéknév
  - birthdate - születési dátum
  - gender - nem
  - address - cím

ER modell



1.1. ábra. ER modell

Relációs modell



1.2. ábra. Relációs modell

2. fejezet

Táblák létrehozása és feltöltése

```
-- users table

CREATE TABLE Users (
    username VARCHAR(255) PRIMARY KEY,
    password CHAR(128) NOT NULL -- passwords hashed with SHA512
);

-- person table

CREATE TABLE PERSON (
    id_number CHAR(8) PRIMARY KEY,
    first_name VARCHAR(255) NOT NULL,
    middle_name VARCHAR(255),
    last_name VARCHAR(255),
    birthdate DATE NOT NULL,
    gender INT NOT NULL,
    address VARCHAR(255) NOT NULL
);

-- car table

CREATE TABLE Car (
    licence_plate CHAR(6) PRIMARY KEY,
    r INTEGER NOT NULL,
    g INTEGER NOT NULL,
    b INTEGER NOT NULL,
    year_made INTEGER NOT NULL,
    vin INTEGER NOT NULL UNIQUE,
    owner_id CHAR(8) NOT NULL,
    CONSTRAINT fk_owner
        FOREIGN KEY (owner_id)
        REFERENCES Person(id_number)
);

-- add an admin user

INSERT INTO Users VALUES (
    'admin',
     q'[d404559f
     602eab6f
     d602ac76
```

```
     80dacbfa
     add13630
     335e951f
     097af390
     0e9de176
     b6db2851
     2f2e000b
     9d04fba5
     133e8b1c
     6e8df59d
     b3a8ab9d
     60be4b97
     cc9e81db]' -- 1234
     -- Perl style string
);

-- fill person table

INSERT INTO Person VALUES (
    '123742JD',
    'Jareth',
    NULL,
    'Dupont',
    (TO_DATE('1990/03/12', 'yyyy/mm/dd')),
    0,
    '1212 Faraway, Imaginary lane12.'
);

INSERT INTO Person VALUES (
    '827364BC',
    'Bryce',
    'J.',
    'Cassidy',
    (TO_DATE('2000/01/30', 'yyyy/mm/dd')),
    0,
    '5502 Lexington, Bullroad 87.'
);

INSERT INTO Person VALUES (
    '963428CM',
    'Cristiano',
    NULL,
    'Mackay',
    (TO_DATE('1988/06/20', 'yyyy/mm/dd')),
    0,
    '4047 Brighton, Sunnystreet 148b 12/2'
);

INSERT INTO Person VALUES (
    '987123LP',
    'Lilly',
    'Grace',
    'Parrish',
    (TO_DATE('1970/04/12', 'yyyy/mm/dd')),
    1,
    '1001 London, Park avenue 1.'
);

```

```sql
-- fill car table

INSERT INTO Car VALUES (
    'AAG145',
    156,
    173,
    206,
    1975,
    986123557,
    '123742JD'
);

INSERT INTO Car VALUES (
    'BCG459',
    255,
    30,
    30,
    1987,
    129863256,
    '827364BC'
);

INSERT INTO Car VALUES (
    'CMB973',
    212,
    175,
    55,
    2011,
    981672357,
    '963428CM'
);

INSERT INTO Car VALUES (
    'JDD742',
    100,
    100,
    100,
    2007,
    1293872136,
    '123742JD'
);

INSERT INTO Car VALUES (
    'LPG652',
    255,
    255,
    255,
    2020,
    98671235,
    '987123LP'
);
```

A feltöltés eredményei

LICENCE	R	G	B	VIN	MADE	ID	FIRST N	MIDDLE N	LAST N	BIRTHDATE	G	ADDRESS
AAG145	156	173	206	986123557	1975	123742JD	Jareth	-	Dupont	12-MAR-90	0	1212Faraway, Imaginarylane12.
BCG459	255	30	30	129863256	1987	827364BC	Bryce	J.	Cassidy	30-JAN-00	0	5502 Lexington, Bullroad 87.
CMB973	212	175	55	981672357	2011	963428CM	Cristiano	-	Mackay	20-JUN-88	0	4047 Brighton, Sunnystreet 148b 12/.
JDD742	100	100	100	1293872136	2007	123742JD	Jareth	-	Dupont	12-MAR-90	0	1212Faraway, Imaginarylane12.
LPG652	255	255	255	98671235	2020	987123LP	Lilly	Grace	Parrish	12-APR-70	1	1001 London, Park avenue 1.

2.1. táblázat. A feltöltés eredménye – mindkét tábla

3. fejezet

Beszúrást és módosítást segítő triggerek

Autók tábla

Autó beszúrása esetén nem lehet jövőbeli gyártási évet megadni, valamint a rendszám formátumának teljesíteni kell az alábbi reguláris kifejezést: `[A-Z,a-z]{3}\d{3}`

```
CREATE OR REPLACE TRIGGER trig_car_insert
BEFORE INSERT OR UPDATE ON Car
FOR EACH ROW
DECLARE
    current_year INTEGER;
BEGIN
    select extract(year from sysdate) into current_year from dual;
    IF (:NEW.year_made > current_year)
    THEN
        RAISE_APPLICATION_ERROR(-20002, 'Future year of manufacturing not
    allowed');
    END IF;
    IF NOT REGEXP_LIKE (:NEW.licence_plate,'[A-Z,a-z]{3}\d{3}')
    THEN
        RAISE_APPLICATION_ERROR(-20001, 'Licence plate of invalid format');
    END IF;
END;
/
```

Ilyenkor, ha jövőbeli évet adunk meg gyártási évnek:

```
INSERT INTO Car VALUES (
    'BCG459',
    255,
    30,
    30,
    2100, -- future
    129863256,
    '827364BC'
);
```

Az alábbi hibaüzenetet kapjuk:

```
ORA-20002: Future year of manufacturing not allowed ORA-06512: at "
    SQL_BTJLCMUNQPAXKNGDUQJSHWIOZ.TRIG_CAR_INSERT", line 7
ORA-06512: at "SYS.DBMS_SQL", line 1721
```

Ha nem megfelelő formátumú rendszámot adunk meg:

```
INSERT INTO Car VALUES (
    'GG12',
    255,
    30,
    30,
    1999,
    129863256,
    '827364BC'
);
```

Az alábbi hibaüzenetet kapjuk:

```
ORA-20001: Licence plate of invalid format ORA-06512: at "
    SQL_BTJLCMUNQPAXKNGDUQJSHWIOZ.TRIG_CAR_INSERT", line 11
ORA-06512: at "SYS.DBMS_SQL", line 1721
```

Tulajdonosok tábla

```
CREATE OR REPLACE TRIGGER trig_person_insert
BEFORE INSERT OR UPDATE ON Person
FOR EACH ROW
DECLARE
BEGIN
    IF(:NEW.birthdate > SYSDATE)
    THEN
        RAISE_APPLICATION_ERROR (-20002 , 'Future dates not allowed');
    END IF;
    IF NOT REGEXP_LIKE (:NEW.id_number,'\d{6}[A-Z,a-z]{2}')
    THEN
        RAISE_APPLICATION_ERROR (-20001 , 'ID of invalid format');
    END IF;
END;
/
```

Ha jövőbeli születési dátumot adunk meg:

```
INSERT INTO Person VALUES (
    '987123LP',
    'Lilly',
    'Grace',
    'Parrish',
    (TO_DATE('2100/04/12', 'yyyy/mm/dd')), -- future
    1,
    '1001 London, Park avenue 1.'
);
```

Az alábbi hibaüzenetet kapjuk:

```
ORA-20002: Future dates not allowed
ORA-06512: at "WKSP_LARIONDB2.TRIG_PERSON_INSERT", line 5
```

```
ORA-04088: error during execution of trigger 'WKSP_LARIONDB2.
    TRIG_PERSON_INSERT'
ORA-06512: at "SYS.DBMS_SQL", line 1721


3.      'Lilly',
4.      'Grace',
5.      'Parrish',
6.      (TO_DATE('2100/04/12', 'yyyy/mm/dd')),
7.      1,
```

Ha nem megfelelő formátumú személyi igazolvány számot adunk meg:

```
INSERT INTO Person VALUES (
    '98712LPG', -- bad format
    'Lilly',
    'Grace',
    'Parrish',
    (TO_DATE('1970/04/12', 'yyyy/mm/dd')),
    1,
    '1001 London, Park avenue 1.'
);
```

Az alábbi hibaüzenetet kapjuk:

```
ORA-20001: ID of invalid format
ORA-06512: at "WKSP_LARIONDB2.TRIG_PERSON_INSERT", line 9
ORA-04088: error during execution of trigger 'WKSP_LARIONDB2.
    TRIG_PERSON_INSERT'
ORA-06512: at "SYS.DBMS_SQL", line 1721


7.      1,
8.      '1001 London, Park avenue 1.'
9. );
```

Felhasználók tábla

A jelszavaknak hashelve kell bekerülniük, ezért, ha nem hashelt a bemenő jelszó, akkor az SHA-512 algoritmussal hashelésre kerülnek. Erre a STANDARD_HASH beépített algoritmus szolgál. A bejövő szöveg CHAR(128) típusú, ezért a valódi hossz meghatározáshoz az utána levő üres karaktereket az RTRIM függvénnyel le kell vágni.

```
CREATE OR REPLACE TRIGGER trig_user_insert
BEFORE INSERT OR UPDATE ON Users
FOR EACH ROW
DECLARE
BEGIN
    IF (LENGTH(RTRIM(:NEW.password)) <> 128)
    THEN
        SELECT standard_hash (RTRIM(:NEW.password), 'SHA512') INTO :NEW.
    password FROM dual;
    END IF;
END;
/
```

Példa:

```
INSERT INTO Users VALUES (
    'administrator',
    'super_secure_password2022'
);

SELECT * FROM Users;
```

Az eredmény:

USERNAME	PASSWORD
administrator	487E262B0600EEC35293233FFD86F5054C47482821F47F12610E31F90-FBE4377DFE067A18D610EC32DD6EB94038B8BB7081BC53E397617565-784A4B67C46E018

4. fejezet

Packagek

Autók táblához tartozó package

```
CREATE OR REPLACE PACKAGE p_cars AS
    TYPE c_tab IS TABLE OF Car%ROWTYPE;
    FUNCTION oldest_car RETURN Car%ROWTYPE;
    FUNCTION cars_of(p_id VARCHAR2) RETURN c_tab PIPELINED;
    PROCEDURE insert_car(
        licence_plate   VARCHAR2,
        r               NUMBER,
        g               NUMBER,
        b               NUMBER,
        year_made       NUMBER,
        vin             NUMBER,
        owner_id        VARCHAR2
    );
    PROCEDURE insert_car(
        licence_plate   VARCHAR2,
        hex_color       VARCHAR2,
        year_made       NUMBER,
        vin             NUMBER,
        owner_id        VARCHAR2
    );
    PROCEDURE insert_car(
        licence_plate   VARCHAR2,
        r               NUMBER,
        g               NUMBER,
        b               NUMBER,
        year_made       NUMBER,
        vin             NUMBER,
        owner           Person%ROWTYPE
    );
    PROCEDURE insert_car(
        licence_plate   VARCHAR2,
        hex_color       VARCHAR2,
        year_made       NUMBER,
        vin             NUMBER,
        owner           Person%ROWTYPE
    );
END p_cars;
```

```
/

CREATE OR REPLACE PACKAGE BODY p_cars AS
    TYPE color IS RECORD (r NUMBER, g NUMBER, b NUMBER);

    FUNCTION hex_to_rgb(hex VARCHAR2) RETURN color
    IS
        rgb color;
    BEGIN
        rgb.r := to_number(substr(hex,1,2),'xx');
        rgb.g := to_number(substr(hex,3,2),'xx');
        rgb.b := to_number(substr(hex,5,2),'xx');

        DBMS_OUTPUT.PUT_LINE(rgb.r || ' ' || rgb.g || ' ' || rgb.b);
        return rgb;
    END;

    FUNCTION oldest_car RETURN Car%ROWTYPE
    IS
        oldest Car%ROWTYPE;
    BEGIN
        SELECT *
        INTO oldest
        FROM Car
        ORDER BY year_made
        FETCH FIRST 1 ROWS ONLY;
        return oldest;
    END;

    FUNCTION cars_of(p_id VARCHAR2) RETURN c_tab PIPELINED
    IS
    BEGIN
        FOR l_c IN (SELECT * FROM Car WHERE owner_id = p_id)
        LOOP
            PIPE ROW(l_c);
        END LOOP;
        RETURN;
    END;

    PROCEDURE insert_car(
        licence_plate   VARCHAR2,
        r               NUMBER,
        g               NUMBER,
        b               NUMBER,
        year_made       NUMBER,
        vin             NUMBER,
        owner_id        VARCHAR2
    )
    IS
    BEGIN
        INSERT INTO Car VALUES (
            licence_plate,
            r,
            g,
            b,
            year_made,
            vin,
            owner_id
```

```
        );
    END;


    PROCEDURE insert_car(
        licence_plate   VARCHAR2,
        hex_color       VARCHAR2,
        year_made       NUMBER,
        vin             NUMBER,
        owner_id        VARCHAR2
    )
    IS
        rgb_val color;
    BEGIN
        rgb_val := hex_to_rgb(hex_color);
        INSERT INTO Car VALUES (
            licence_plate,
            rgb_val.r,
            rgb_val.g,
            rgb_val.b,
            year_made,
            vin,
            owner_id
        );
    END;

    PROCEDURE insert_car(
        licence_plate   VARCHAR2,
        r               NUMBER,
        g               NUMBER,
        b               NUMBER,
        year_made       NUMBER,
        vin             NUMBER,
        owner           Person%ROWTYPE
    )
    IS
    BEGIN
        insert_car(
            licence_plate,
            r,
            g,
            b,
            year_made,
            vin,
            owner.id_number
        );
    END;

    PROCEDURE insert_car(
        licence_plate   VARCHAR2,
        hex_color       VARCHAR2,
        year_made       NUMBER,
        vin             NUMBER,
        owner           Person%ROWTYPE
    )
    IS
    BEGIN
        insert_car(
```

```
            licence_plate,
            hex_color,
            year_made,
            vin,
            owner.id_number
        );
    END;
END p_cars;
/
```

Példák

Autó beillesztése tárolt eljárással:

```
BEGIN
    p_cars.insert_car(
        'AGV123',
        'FFFF00',
        1994,
        1231215,
        '123742JD'
    );
END;
```

Azon autók, melyek a 123742JD személyi igazolvány számú tulajdonoshoz tartoznak:

```
SELECT * FROM TABLE(p_cars.cars_of('123742JD'));
```

Tulajdonosokhoz tartozó package

```
CREATE OR REPLACE PACKAGE p_person AS
    TYPE p_tab IS TABLE OF Person%ROWTYPE;
    FUNCTION get_age(birthdate IN DATE) RETURN NUMBER;
    FUNCTION average_age RETURN NUMBER;
    FUNCTION oldest RETURN Person%ROWTYPE;
    FUNCTION under_age(age NUMBER) RETURN p_tab PIPELINED;
    PROCEDURE insert_person(
        id_number       VARCHAR2,
        first_name      VARCHAR2,
        middle_name     VARCHAR2,
        last_name       VARCHAR2,
        birthdate       DATE,
        gender          NUMBER,
        address         VARCHAR2
    );
    PROCEDURE insert_person(
        id_number       VARCHAR2,
        p_name            VARCHAR2,
        birthdate       DATE,
        gender          NUMBER,
        address         VARCHAR2
    );
    PROCEDURE insert_person(
        id_number       VARCHAR2,
        first_name      VARCHAR2,
```

```
        middle_name     VARCHAR2,
        last_name       VARCHAR2,
        birthdate       DATE,
        gender          VARCHAR2,
        address         VARCHAR2
    );
    PROCEDURE insert_person(
        id_number       VARCHAR2,
        p_name            VARCHAR2,
        birthdate       DATE,
        gender          VARCHAR2,
        address         VARCHAR2
    );
    PROCEDURE delete_person(
        v_first_name       VARCHAR2,
        v_middle_name     VARCHAR2,
        v_last_name       VARCHAR2
    );
    PROCEDURE delete_person(
        v_first_name       VARCHAR2,
        v_last_name       VARCHAR2
    );
    PROCEDURE delete_person(p_name VARCHAR2);
END p_person;
/

CREATE OR REPLACE PACKAGE BODY p_person AS
    FUNCTION get_age(birthdate IN DATE) RETURN NUMBER
    IS
        age NUMBER;
    BEGIN
        age := (sysdate - birthdate) / 365;
        return age;
    END;
    FUNCTION average_age RETURN NUMBER
    IS
        l_avg NUMBER;
        l_p_count NUMBER;
    BEGIN
        FOR l_bd IN (SELECT birthdate FROM Person)
        LOOP
            l_avg := l_avg + get_age(l_bd.birthdate);
        END LOOP;
        SELECT COUNT(*) INTO l_p_count FROM Person;
        l_avg := l_avg / l_p_count;
        RETURN l_avg;
    END;
    FUNCTION oldest RETURN Person%ROWTYPE
    IS
        l_oldest Person%ROWTYPE;
    BEGIN
        SELECT *
        INTO l_oldest
        FROM Person
        ORDER BY birthdate
        FETCH FIRST 1 ROWS ONLY;
        return l_oldest;
    END;
```

```
    FUNCTION under_age(age NUMBER) RETURN p_tab PIPELINED
    IS
    BEGIN
        FOR l_p IN (SELECT * FROM Person WHERE get_age(Person.birthdate) <
    age)
        LOOP
            PIPE ROW(l_p);
        END LOOP;
        RETURN;
    END;
    PROCEDURE insert_person(
        id_number       VARCHAR2,
        first_name      VARCHAR2,
        middle_name     VARCHAR2,
        last_name       VARCHAR2,
        birthdate       DATE,
        gender          NUMBER,
        address         VARCHAR2
    )
    IS
    BEGIN
        INSERT INTO Person VALUES(
            id_number,
            first_name,
            middle_name,
            last_name,
            birthdate,
            gender,
            address
        );
    END;
    PROCEDURE insert_person(
        id_number       VARCHAR2,
        p_name            VARCHAR2,
        birthdate       DATE,
        gender          NUMBER,
        address         VARCHAR2
    )
    IS
        v_array apex_application_global.vc_arr2;
    BEGIN
        v_array := apex_util.string_to_table(p_name, ' ');
        IF (v_array.count = 2)
        THEN
            insert_person(
                id_number,
                v_array(1),
                NULL,
                v_array(2),
                birthdate,
                gender,
                address
            );
        ELSIF (v_array.count = 3)
        THEN
            insert_person(
                id_number,
                v_array(1),
```

```
                v_array(2),
                v_array(3),
                birthdate,
                gender,
                address
            );
        ELSE
            RAISE_APPLICATION_ERROR(-20003, 'Person name of invalid format'
);
        END IF;
    END;
    PROCEDURE insert_person(
        id_number       VARCHAR2,
        first_name      VARCHAR2,
        middle_name     VARCHAR2,
        last_name       VARCHAR2,
        birthdate       DATE,
        gender          VARCHAR2,
        address         VARCHAR2
    )
    IS
        v_gender NUMBER;
    BEGIN
        IF (gender = 'MALE')
        THEN
            v_gender := 0;
        ELSIF (gender = 'FEMALE')
        THEN
            v_gender := 1;
        ELSE
            RAISE_APPLICATION_ERROR(-20004, q'[Person"s gender of invalid
format]');
        END IF;
        insert_person(
                id_number,
                first_name,
                middle_name,
                last_name,
                birthdate,
                v_gender,
                address
            );
    END;
    PROCEDURE insert_person(
        id_number       VARCHAR2,
        p_name            VARCHAR2,
        birthdate       DATE,
        gender          VARCHAR2,
        address         VARCHAR2
    )
    IS
        v_array apex_application_global.vc_arr2;
    BEGIN
        v_array := apex_util.string_to_table(p_name, ' ');
        IF (v_array.count = 2)
        THEN
            insert_person(
                id_number,
```

```
                v_array(1),
                NULL,
                v_array(2),
                birthdate,
                gender,
                address
            );
        ELSIF (v_array.count = 3)
        THEN
            insert_person(
                id_number,
                v_array(1),
                v_array(2),
                v_array(3),
                birthdate,
                gender,
                address
            );
        ELSE
            RAISE_APPLICATION_ERROR(-20003, 'Person name of invalid format'
    );
        END IF;
    END;
    PROCEDURE delete_person(
        v_first_name VARCHAR2,
        v_last_name VARCHAR2
    )
    IS
        v_rownum NUMBER;
    BEGIN
        SELECT COUNT(*)
        INTO v_rownum
        FROM Person
            WHERE Person.first_name = v_first_name
                AND Person.middle_name IS NULL
                AND Person.last_name = v_last_name;
        IF (v_rownum > 1)
        THEN
            RAISE_APPLICATION_ERROR(-20004, 'Person ambiguous');
        END IF;
        DELETE FROM Person WHERE
            Person.first_name = v_first_name
            AND Person.last_name = v_last_name;
    END;
    PROCEDURE delete_person(
        v_first_name       VARCHAR2,
        v_middle_name     VARCHAR2,
        v_last_name       VARCHAR2
    )
    IS
        v_rownum NUMBER;
    BEGIN
        IF (v_middle_name IS NULL)
        THEN
            delete_person(v_first_name, v_last_name);
        END IF;
        SELECT COUNT(*)
        INTO v_rownum
```

```
        FROM Person
            WHERE Person.first_name = v_first_name
                AND Person.middle_name = v_middle_name
                AND Person.last_name = v_last_name;
        IF (v_rownum > 1)
        THEN
            RAISE_APPLICATION_ERROR(-20004, 'Person ambiguous');
        END IF;
        DELETE FROM Person
            WHERE Person.first_name = v_first_name
            AND Person.middle_name = v_middle_name
            AND Person.last_name = v_last_name;
    END;
    PROCEDURE delete_person(p_name VARCHAR2)
    IS
        v_array apex_application_global.vc_arr2;
    BEGIN
        v_array := apex_util.string_to_table(p_name, ' ');
        IF (v_array.count = 2)
        THEN
            delete_person(
                v_array(1),
                v_array(2)
            );
        ELSIF (v_array.count = 3)
        THEN
            delete_person(
                v_array(1),
                v_array(2),
                v_array(3)
            );
        ELSE
            RAISE_APPLICATION_ERROR(-20003, 'Person name of invalid format'
    );
        END IF;
    END;
END p_person;
/
```

Példák

Tulajdonos beillesztése tárolt eljárással:

```
BEGIN
p_person.insert_person(
    '897616LF',
    'John Rockefeller',
    (TO_DATE('1965/01/30', 'yyyy/mm/dd')),
    'MALE',
    '5502 Lexington, Bullroad 87.'
);
END;
```

Eredmény:

id	first name	middle name	last name	birthdate	g	address
897616LF	John	-	Rockefeller	01/30/1965	0	5502 Lexington, Bullroad 87.

Tulajdonos életkora:

```
1    SELECT p_person.get_age((TO_DATE('1965/01/30', 'yyyy/mm/dd'))) FROM
     DUAL;
```

Eredmény:

```
57.282900082445459157787924911212582445 5
```

40 alatti tulajdonosok:

```
1    SELECT * FROM TABLE(p_person.under_age(40));
```

Eredmény:

id	first name	middle name	last name	birthdate	g	address
123742JD	Jareth	-	Dupont	03/12/1990	0	1212 Faraway, Imaginary lane 12.
827364BC	Bryce	J.	Cassidy	01/30/2000	0	5502 Lexington, Bullroad 87.
963428CM	Cristiano	-	Mackay	06/20/1988	0	4047 Brighton, Sunnystreet 148b 12/2

Átlagéletkor:

```
1    SELECT P_PERSON.average_age() FROM DUAL;
```

Eredmény:

```
39.532364313800101471334348046676813 8
```

Tulajdonos törlése:

```
1 BEGIN
2     p_person.delete_person('John Rockefeller');
3 END;
```