

PL/SQL feladat

Adatbázis rendszerek 2

Szabó Larion

NSW74Y

2022

Tartalomjegyzék

1. A feladat	2
1.1. ER modell	3
1.2. Relációs modell	3
2. Táblák létrehozása és feltöltése	4
2.1. A feltöltés eredményei	7
3. Beszúrást és módosítást segítő triggerek	8
3.1. Autók tábla	8
3.2. Tulajdonosok tábla	9
3.3. Felhasználók tábla	10
4. Packagek	12
4.1. Autók táblához tartozó package	12
4.1.1. Példák	15
4.2. Tulajdonosokhoz tartozó package	15
4.2.1. Példák	20

1. fejezet

A feladat

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

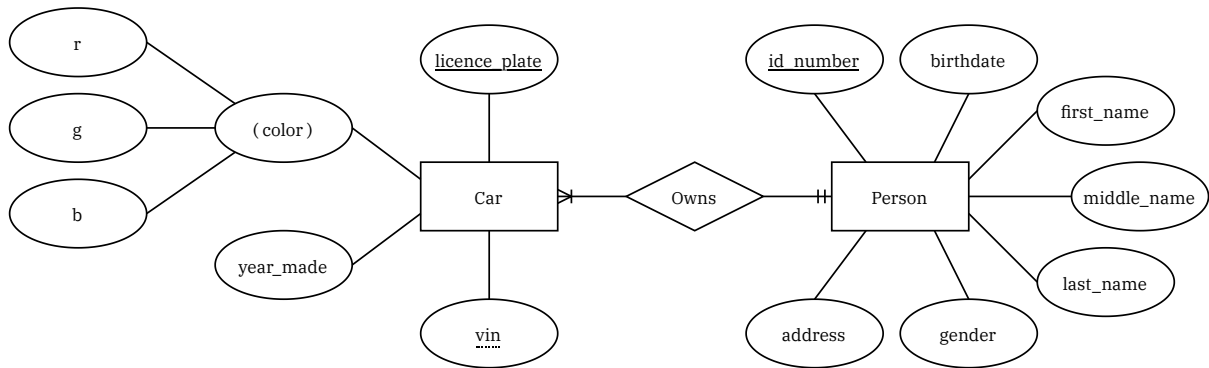
Az alkalmazáshoz szükséges az adminisztrátoroknak biztosítani, hogy saját felhasználói fiókjuk és jelszavuk legyen, utóbbi pedig nem egyszerű szöveggént kerül tárolásra.

- A felhasználónév
 - Ez szolgál az egyedi azonosítóként
- A jelszó
 - Nem egyszerű szöveggént 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.

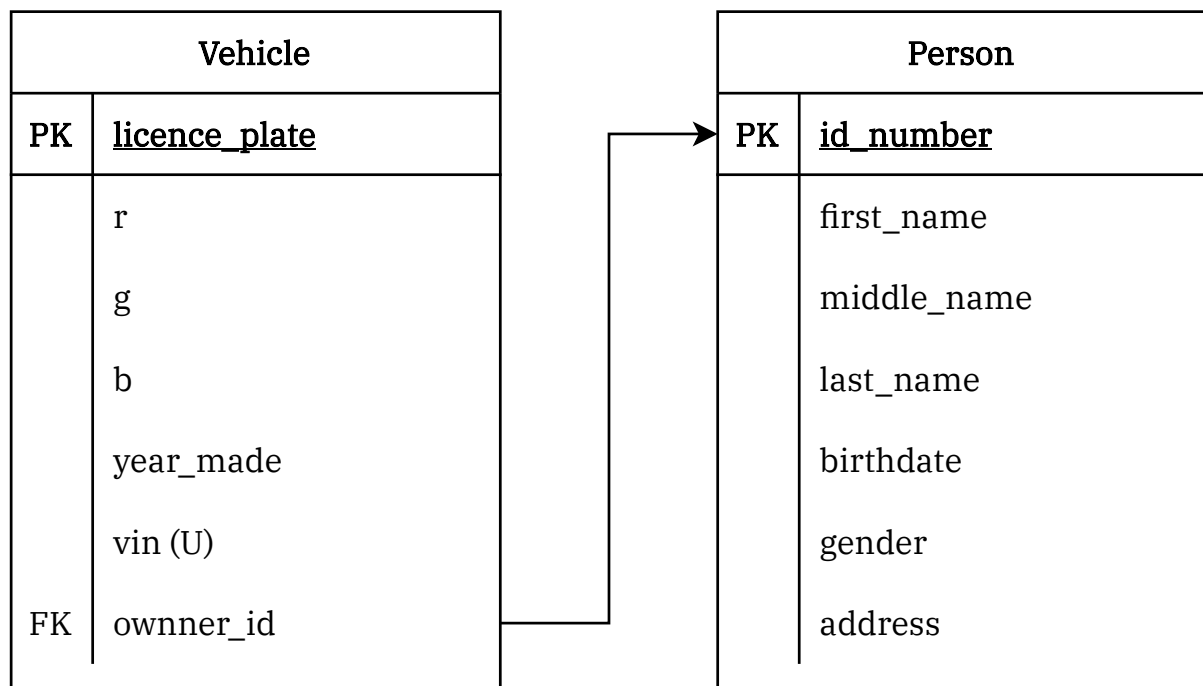
- Car - Autók
 - licence_plate - rendszám
 - color - szín
 - * r - vörös (0-255)
 - * g - zöld (0-255)
 - * b - kék (0-255)
 - year_made - gyártás éve
 - vin - alvázszám
- Person - Tulaj
 - id_number - igazolványszám
 - first_name - keresztnév
 - middle_name - harmadik név
 - last_name - vezetéknév
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

```
1 -- users table
2
3 CREATE TABLE Users (
4     username VARCHAR(255) PRIMARY KEY,
5     password CHAR(128) NOT NULL -- passwords hashed with SHA512
6 );
7
8 -- person table
9
10 CREATE TABLE PERSON (
11     id_number CHAR(8) PRIMARY KEY,
12     first_name VARCHAR(255) NOT NULL,
13     middle_name VARCHAR(255),
14     last_name VARCHAR(255),
15     birthdate DATE NOT NULL,
16     gender INT NOT NULL,
17     address VARCHAR(255) NOT NULL
18 );
19
20 -- car table
21
22 CREATE TABLE Car (
23     licence_plate CHAR(6) PRIMARY KEY,
24     r INTEGER NOT NULL,
25     g INTEGER NOT NULL,
26     b INTEGER NOT NULL,
27     year_made INTEGER NOT NULL,
28     vin INTEGER NOT NULL UNIQUE,
29     owner_id CHAR(8) NOT NULL,
30     CONSTRAINT fk_owner
31         FOREIGN KEY (owner_id)
32         REFERENCES Person(id_number)
33 );
34
35 -- add an admin user
36
37 INSERT INTO Users VALUES (
38     'admin',
39     q'[d404559f
40     602eab6f
41     d602ac76
```

```

42     80dacbfa
43     add13630
44     335e951f
45     097af390
46     0e9de176
47     b6db2851
48     2f2e000b
49     9d04fba5
50     133e8b1c
51     6e8df59d
52     b3a8ab9d
53     60be4b97
54     cc9e81db]' -- 1234
55     -- Perl style string
56 );
57
58 -- fill person table
59
60 INSERT INTO Person VALUES (
61     '123742JD',
62     'Jareth',
63     NULL,
64     'Dupont',
65     (TO_DATE('1990/03/12', 'yyyy/mm/dd')),
66     0,
67     '1212 Faraway, Imaginary lane12.'
68 );
69
70 INSERT INTO Person VALUES (
71     '827364BC',
72     'Bryce',
73     'J.',
74     'Cassidy',
75     (TO_DATE('2000/01/30', 'yyyy/mm/dd')),
76     0,
77     '5502 Lexington, Bullroad 87.'
78 );
79
80 INSERT INTO Person VALUES (
81     '963428CM',
82     'Cristiano',
83     NULL,
84     'Mackay',
85     (TO_DATE('1988/06/20', 'yyyy/mm/dd')),
86     0,
87     '4047 Brighton, Sunnystreet 148b 12/2'
88 );
89
90 INSERT INTO Person VALUES (
91     '987123LP',
92     'Lilly',
93     'Grace',
94     'Parrish',
95     (TO_DATE('1970/04/12', 'yyyy/mm/dd')),
96     1,
97     '1001 London, Park avenue 1.'
98 );
99

```

```

100 -- fill car table
101
102 INSERT INTO Car VALUES (
103     'AAG145',
104     156,
105     173,
106     206,
107     1975,
108     986123557,
109     '123742JD'
110 );
111
112 INSERT INTO Car VALUES (
113     'BCG459',
114     255,
115     30,
116     30,
117     1987,
118     129863256,
119     '827364BC'
120 );
121
122 INSERT INTO Car VALUES (
123     'CMB973',
124     212,
125     175,
126     55,
127     2011,
128     981672357,
129     '963428CM'
130 );
131
132 INSERT INTO Car VALUES (
133     'JDD742',
134     100,
135     100,
136     100,
137     2007,
138     1293872136,
139     '123742JD'
140 );
141
142 INSERT INTO Car VALUES (
143     'LPG652',
144     255,
145     255,
146     255,
147     2020,
148     98671235,
149     '987123LP'
150 );

```

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

Beszúrás és módosítást segítő triggererek

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}`

```
1 CREATE OR REPLACE TRIGGER trig_car_insert
2 BEFORE INSERT OR UPDATE ON Car
3 FOR EACH ROW
4 DECLARE
5     current_year INTEGER;
6 BEGIN
7     select extract(year from sysdate) into current_year from dual;
8     IF (:NEW.year_made > current_year)
9     THEN
10         RAISE_APPLICATION_ERROR(-20002, 'Future year of manufacturing not
allowed');
11     END IF;
12     IF NOT REGEXP_LIKE (:NEW.licence_plate, '[A-Z,a-z]{3}\d{3}')
13     THEN
14         RAISE_APPLICATION_ERROR(-20001, 'Licence plate of invalid format');
15     END IF;
16 END;
17 /
```

Ilyenkor, ha jövőbeli évet adunk meg gyártási évnél:

```
1 INSERT INTO Car VALUES (
2     'BCG459',
3     255,
4     30,
5     30,
6     2100, -- future
7     129863256,
8     '827364BC'
9 );
```

Az alábbi hibaüzenetet kapjuk:

```

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

```

Ha nem megfelelő formátumú rendszámot adunk meg:

```

1 INSERT INTO Car VALUES (
2     'GG12',
3     255,
4     30,
5     30,
6     1999,
7     129863256,
8     '827364BC'
9 );

```

Az alábbi hibaüzenetet kapjuk:

```

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

```

Tulajdonosok tábla

```

1 CREATE OR REPLACE TRIGGER trig_person_insert
2 BEFORE INSERT OR UPDATE ON Person
3 FOR EACH ROW
4 DECLARE
5 BEGIN
6     IF(:NEW.birthdate > SYSDATE)
7     THEN
8         RAISE_APPLICATION_ERROR (-20002 , 'Future dates not allowed');
9     END IF;
10    IF NOT REGEXP_LIKE (:NEW.id_number, '\d{6}[A-Z,a-z]{2}')
11    THEN
12        RAISE_APPLICATION_ERROR (-20001 , 'ID of invalid format');
13    END IF;
14 END;
15 /

```

Ha jövőbeli születési dátumot adunk meg:

```

1 INSERT INTO Person VALUES (
2     '987123LP',
3     'Lilly',
4     'Grace',
5     'Parrish',
6     (TO_DATE('2100/04/12', 'yyyy/mm/dd')), -- future
7     1,
8     '1001 London, Park avenue 1.'
9 );

```

Az alábbi hibaüzenetet kapjuk:

```

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

```

```

3 ORA-04088: error during execution of trigger 'WKSP_LARIONDB2.
  TRIG_PERSON_INSERT'
4 ORA-06512: at "SYS.DBMS_SQL", line 1721
5
6
7 3.      'Lilly',
8 4.      'Grace',
9 5.      'Parrish',
10 6.      (TO_DATE('2100/04/12', 'yyyy/mm/dd')),
11 7.      1,

```

Ha nem megfelelő formátumú személyi igazolvány számot adunk meg:

```

1 INSERT INTO Person VALUES (
2     '98712LPG', -- bad format
3     'Lilly',
4     'Grace',
5     'Parrish',
6     (TO_DATE('1970/04/12', 'yyyy/mm/dd')),
7     1,
8     '1001 London, Park avenue 1.'
9 );

```

Az alábbi hibaüzenetet kapjuk:

```

1 ORA-20001: ID of invalid format
2 ORA-06512: at "WKSP_LARIONDB2.TRIG_PERSON_INSERT", line 9
3 ORA-04088: error during execution of trigger 'WKSP_LARIONDB2.
  TRIG_PERSON_INSERT'
4 ORA-06512: at "SYS.DBMS_SQL", line 1721
5
6
7 7.      1,
8 8.      '1001 London, Park avenue 1.'
9 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.

```

1 CREATE OR REPLACE TRIGGER trig_user_insert
2 BEFORE INSERT OR UPDATE ON Users
3 FOR EACH ROW
4 DECLARE
5 BEGIN
6     IF (LENGTH(RTRIM(:NEW.password)) <> 128)
7     THEN
8         SELECT standard_hash (RTRIM(:NEW.password), 'SHA512') INTO :NEW.
          password FROM dual;
9     END IF;
10 END;
11 /

```

Példa:

```
1 INSERT INTO Users VALUES (  
2     'administrator',  
3     'super_secure_password2022'  
4 );  
5  
6 SELECT * FROM Users;
```

Az eredmény:

USERNAME	PASSWORD
administrator	487E262B0600EEC35293233FFD86F5054C47482821F47F12610E31F90- FBE4377DFE067A18D610EC32DD6EB94038B8BB7081BC53E397617565- 784A4B67C46E018

4. fejezet

Packagek

Autók táblához tartozó package

```
1 CREATE OR REPLACE PACKAGE p_cars AS
2     TYPE c_tab IS TABLE OF Car%ROWTYPE;
3     FUNCTION oldest_car RETURN Car%ROWTYPE;
4     FUNCTION cars_of(p_id VARCHAR2) RETURN c_tab PIPELINED;
5     PROCEDURE insert_car(
6         licence_plate  VARCHAR2,
7         r              NUMBER,
8         g              NUMBER,
9         b              NUMBER,
10        year_made      NUMBER,
11        vin            NUMBER,
12        owner_id       VARCHAR2
13    );
14    PROCEDURE insert_car(
15        licence_plate  VARCHAR2,
16        hex_color      VARCHAR2,
17        year_made      NUMBER,
18        vin            NUMBER,
19        owner_id       VARCHAR2
20    );
21    PROCEDURE insert_car(
22        licence_plate  VARCHAR2,
23        r              NUMBER,
24        g              NUMBER,
25        b              NUMBER,
26        year_made      NUMBER,
27        vin            NUMBER,
28        owner          Person%ROWTYPE
29    );
30    PROCEDURE insert_car(
31        licence_plate  VARCHAR2,
32        hex_color      VARCHAR2,
33        year_made      NUMBER,
34        vin            NUMBER,
35        owner          Person%ROWTYPE
36    );
37 END p_cars;
```

```

38 /
39
40 CREATE OR REPLACE PACKAGE BODY p_cars AS
41     TYPE color IS RECORD (r NUMBER, g NUMBER, b NUMBER);
42
43     FUNCTION hex_to_rgb(hex VARCHAR2) RETURN color
44     IS
45         rgb color;
46     BEGIN
47         rgb.r := to_number(substr(hex,1,2),'xx');
48         rgb.g := to_number(substr(hex,3,2),'xx');
49         rgb.b := to_number(substr(hex,5,2),'xx');
50
51         DBMS_OUTPUT.PUT_LINE(rgb.r || ' ' || rgb.g || ' ' || rgb.b);
52         return rgb;
53     END;
54
55     FUNCTION oldest_car RETURN Car%ROWTYPE
56     IS
57         oldest Car%ROWTYPE;
58     BEGIN
59         SELECT *
60         INTO oldest
61         FROM Car
62         ORDER BY year_made
63         FETCH FIRST 1 ROWS ONLY;
64         return oldest;
65     END;
66
67     FUNCTION cars_of(p_id VARCHAR2) RETURN c_tab PIPELINED
68     IS
69     BEGIN
70         FOR l_c IN (SELECT * FROM Car WHERE owner_id = p_id)
71         LOOP
72             PIPE ROW(l_c);
73         END LOOP;
74         RETURN;
75     END;
76
77     PROCEDURE insert_car(
78         licence_plate VARCHAR2,
79         r              NUMBER,
80         g              NUMBER,
81         b              NUMBER,
82         year_made      NUMBER,
83         vin            NUMBER,
84         owner_id       VARCHAR2
85     )
86     IS
87     BEGIN
88         INSERT INTO Car VALUES (
89             licence_plate,
90             r,
91             g,
92             b,
93             year_made,
94             vin,
95             owner_id

```

```

96         );
97     END;
98
99
100    PROCEDURE insert_car(
101        licence_plate    VARCHAR2,
102        hex_color         VARCHAR2,
103        year_made         NUMBER,
104        vin               NUMBER,
105        owner_id          VARCHAR2
106    )
107    IS
108        rgb_val color;
109    BEGIN
110        rgb_val := hex_to_rgb(hex_color);
111        INSERT INTO Car VALUES (
112            licence_plate,
113            rgb_val.r,
114            rgb_val.g,
115            rgb_val.b,
116            year_made,
117            vin,
118            owner_id
119        );
120    END;
121
122    PROCEDURE insert_car(
123        licence_plate    VARCHAR2,
124        r                NUMBER,
125        g                NUMBER,
126        b                NUMBER,
127        year_made         NUMBER,
128        vin              NUMBER,
129        owner             Person%ROWTYPE
130    )
131    IS
132    BEGIN
133        insert_car(
134            licence_plate,
135            r,
136            g,
137            b,
138            year_made,
139            vin,
140            owner.id_number
141        );
142    END;
143
144    PROCEDURE insert_car(
145        licence_plate    VARCHAR2,
146        hex_color         VARCHAR2,
147        year_made         NUMBER,
148        vin              NUMBER,
149        owner             Person%ROWTYPE
150    )
151    IS
152    BEGIN
153        insert_car(

```

```

154         licence_plate,
155         hex_color,
156         year_made,
157         vin,
158         owner.id_number
159     );
160 END;
161 END p_cars;
162 /

```

Példák

Autó beillesztése tárolt eljárással:

```

1 BEGIN
2     p_cars.insert_car(
3         'AGV123',
4         'FFFF00',
5         1994,
6         1231215,
7         '123742JD'
8     );
9 END;

```

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

```

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

```

Tulajdonosokhoz tartozó package

```

1 CREATE OR REPLACE PACKAGE p_person AS
2     TYPE p_tab IS TABLE OF Person%ROWTYPE;
3     FUNCTION get_age(birthdate IN DATE) RETURN NUMBER;
4     FUNCTION average_age RETURN NUMBER;
5     FUNCTION oldest RETURN Person%ROWTYPE;
6     FUNCTION under_age(age NUMBER) RETURN p_tab PIPELINED;
7     PROCEDURE insert_person(
8         id_number      VARCHAR2,
9         first_name     VARCHAR2,
10        middle_name    VARCHAR2,
11        last_name       VARCHAR2,
12        birthdate       DATE,
13        gender          NUMBER,
14        address         VARCHAR2
15    );
16    PROCEDURE insert_person(
17        id_number      VARCHAR2,
18        p_name         VARCHAR2,
19        birthdate       DATE,
20        gender          NUMBER,
21        address         VARCHAR2
22    );
23    PROCEDURE insert_person(
24        id_number      VARCHAR2,
25        first_name     VARCHAR2,

```



```

26         middle_name      VARCHAR2,
27         last_name        VARCHAR2,
28         birthdate        DATE,
29         gender           VARCHAR2,
30         address          VARCHAR2
31     );
32     PROCEDURE insert_person(
33         id_number         VARCHAR2,
34         p_name            VARCHAR2,
35         birthdate        DATE,
36         gender           VARCHAR2,
37         address          VARCHAR2
38     );
39     PROCEDURE delete_person(
40         v_first_name      VARCHAR2,
41         v_middle_name     VARCHAR2,
42         v_last_name       VARCHAR2
43     );
44     PROCEDURE delete_person(
45         v_first_name      VARCHAR2,
46         v_last_name       VARCHAR2
47     );
48     PROCEDURE delete_person(p_name VARCHAR2);
49 END p_person;
50 /
51
52 CREATE OR REPLACE PACKAGE BODY p_person AS
53     FUNCTION get_age(birthdate IN DATE) RETURN NUMBER
54     IS
55         age NUMBER;
56     BEGIN
57         age := (sysdate - birthdate) / 365;
58         return age;
59     END;
60     FUNCTION average_age RETURN NUMBER
61     IS
62         l_avg NUMBER;
63         l_p_count NUMBER;
64     BEGIN
65         FOR l_bd IN (SELECT birthdate FROM Person)
66         LOOP
67             l_avg := l_avg + get_age(l_bd.birthdate);
68         END LOOP;
69         SELECT COUNT(*) INTO l_p_count FROM Person;
70         l_avg := l_avg / l_p_count;
71         RETURN l_avg;
72     END;
73     FUNCTION oldest RETURN Person%ROWTYPE
74     IS
75         l_oldest Person%ROWTYPE;
76     BEGIN
77         SELECT *
78         INTO l_oldest
79         FROM Person
80         ORDER BY birthdate
81         FETCH FIRST 1 ROWS ONLY;
82         return l_oldest;
83     END;

```

```

84 FUNCTION under_age(age NUMBER) RETURN p_tab PIPELINED
85 IS
86 BEGIN
87     FOR l_p IN (SELECT * FROM Person WHERE get_age(Person.birthdate) <
age)
88     LOOP
89         PIPE ROW(l_p);
90     END LOOP;
91     RETURN;
92 END;
93 PROCEDURE insert_person(
94     id_number      VARCHAR2,
95     first_name     VARCHAR2,
96     middle_name    VARCHAR2,
97     last_name      VARCHAR2,
98     birthdate      DATE,
99     gender         NUMBER,
100    address        VARCHAR2
101 )
102 IS
103 BEGIN
104     INSERT INTO Person VALUES(
105         id_number,
106         first_name,
107         middle_name,
108         last_name,
109         birthdate,
110         gender,
111         address
112     );
113 END;
114 PROCEDURE insert_person(
115     id_number      VARCHAR2,
116     p_name         VARCHAR2,
117     birthdate      DATE,
118     gender         NUMBER,
119     address        VARCHAR2
120 )
121 IS
122     v_array apex_application_global.vc_arr2;
123 BEGIN
124     v_array := apex_util.string_to_table(p_name, ' ');
125     IF (v_array.count = 2)
126     THEN
127         insert_person(
128             id_number,
129             v_array(1),
130             NULL,
131             v_array(2),
132             birthdate,
133             gender,
134             address
135         );
136     ELSIF (v_array.count = 3)
137     THEN
138         insert_person(
139             id_number,
140             v_array(1),

```

```

141         v_array(2),
142         v_array(3),
143         birthdate,
144         gender,
145         address
146     );
147     ELSE
148         RAISE_APPLICATION_ERROR(-20003, 'Person name of invalid format'
149 );
150     END IF;
151     END;
152     PROCEDURE insert_person(
153         id_number      VARCHAR2,
154         first_name     VARCHAR2,
155         middle_name    VARCHAR2,
156         last_name      VARCHAR2,
157         birthdate      DATE,
158         gender         VARCHAR2,
159         address        VARCHAR2
160     )
161     IS
162         v_gender NUMBER;
163     BEGIN
164         IF (gender = 'MALE')
165             THEN
166                 v_gender := 0;
167             ELSIF (gender = 'FEMALE')
168                 THEN
169                     v_gender := 1;
170             ELSE
171                 RAISE_APPLICATION_ERROR(-20004, q'[Person"s gender of invalid
format]');
172             END IF;
173             insert_person(
174                 id_number,
175                 first_name,
176                 middle_name,
177                 last_name,
178                 birthdate,
179                 v_gender,
180                 address
181             );
182     END;
183     PROCEDURE insert_person(
184         id_number      VARCHAR2,
185         p_name         VARCHAR2,
186         birthdate      DATE,
187         gender         VARCHAR2,
188         address        VARCHAR2
189     )
190     IS
191         v_array apex_application_global.vc_arr2;
192     BEGIN
193         v_array := apex_util.string_to_table(p_name, ' ');
194         IF (v_array.count = 2)
195             THEN
196                 insert_person(
197                     id_number,

```

```

197         v_array(1),
198         NULL,
199         v_array(2),
200         birthdate,
201         gender,
202         address
203     );
204     ELSIF (v_array.count = 3)
205     THEN
206         insert_person(
207             id_number,
208             v_array(1),
209             v_array(2),
210             v_array(3),
211             birthdate,
212             gender,
213             address
214         );
215     ELSE
216         RAISE_APPLICATION_ERROR(-20003, 'Person name of invalid format'
217 );
218     END IF;
219 END;
220
221 PROCEDURE delete_person(
222     v_first_name VARCHAR2,
223     v_last_name VARCHAR2
224 )
225 IS
226     v_rownum NUMBER;
227 BEGIN
228     SELECT COUNT(*)
229     INTO v_rownum
230     FROM Person
231     WHERE Person.first_name = v_first_name
232           AND Person.middle_name IS NULL
233           AND Person.last_name = v_last_name;
234     IF (v_rownum > 1)
235     THEN
236         RAISE_APPLICATION_ERROR(-20004, 'Person ambiguous');
237     END IF;
238     DELETE FROM Person WHERE
239         Person.first_name = v_first_name
240         AND Person.last_name = v_last_name;
241 END;
242
243 PROCEDURE delete_person(
244     v_first_name VARCHAR2,
245     v_middle_name VARCHAR2,
246     v_last_name VARCHAR2
247 )
248 IS
249     v_rownum NUMBER;
250 BEGIN
251     IF (v_middle_name IS NULL)
252     THEN
253         delete_person(v_first_name, v_last_name);
254     END IF;
255     SELECT COUNT(*)
256     INTO v_rownum

```

```

254     FROM Person
255         WHERE Person.first_name = v_first_name
256             AND Person.middle_name = v_middle_name
257             AND Person.last_name = v_last_name;
258     IF (v_rownum > 1)
259     THEN
260         RAISE_APPLICATION_ERROR(-20004, 'Person ambiguous');
261     END IF;
262     DELETE FROM Person
263         WHERE Person.first_name = v_first_name
264             AND Person.middle_name = v_middle_name
265             AND Person.last_name = v_last_name;
266 END;
267 PROCEDURE delete_person(p_name VARCHAR2)
268 IS
269     v_array apex_application_global.vc_arr2;
270 BEGIN
271     v_array := apex_util.string_to_table(p_name, ' ');
272     IF (v_array.count = 2)
273     THEN
274         delete_person(
275             v_array(1),
276             v_array(2)
277         );
278     ELSIF (v_array.count = 3)
279     THEN
280         delete_person(
281             v_array(1),
282             v_array(2),
283             v_array(3)
284         );
285     ELSE
286         RAISE_APPLICATION_ERROR(-20003, 'Person name of invalid format'
287     );
288     END IF;
289 END p_person;
290 /

```

Példák

Tulajdonos beillesztése tárolt eljárással:

```

1 BEGIN
2 p_person.insert_person(
3     '897616LF',
4     'John Rockefeller',
5     (TO_DATE('1965/01/30', 'yyyy/mm/dd')),
6     'MALE',
7     '5502 Lexington, Bullroad 87.'
8 );
9 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:

57.2829000824454591577879249112125824455

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, Sunny-street 148b 12/2

Átlagéletkor:

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

Eredmény:

39.5323643138001014713343480466768138

Tulajdonos törlése:

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