

Adatbázisok II

1. gyakorlat

Követelmények és információk

- **Gyakorlat:** 17. csoport – Péntek 8:00 – 10:00 Microsoft labor (LD-2-124)
- **Gyakorlat vezető:** Holicza Barnabás, **Elérhetőségek:** Teams-en vagy email - arian2@inf.elte.hu
- Jelenléti oktatás, gyakorlat látogatása kötelező (katalógus lesz vezetve minden órán)
- **A gyakorlati jegy megszerzésének feltételei:**
 - A gyakorlatokon való jelenlét (**maximum 3 hiányzás megengedett** --> HKR. 66. § (1) b), c)). Ennél **több hiányzás esetén gyakjegy uv-val sem szerezhető** meg a gyakorlati jegy.
 - A félév során megírandó 2 zh mindegyikén **1-estől különböző jegy megszerzése.**
 - A gyakorlati jegyet a **2 ZH érdemjegyének átlaga** adja a kerekítés szabályai szerint. (4.5 -> 5)
 - Az **egyik zh** (1-es vagy bármilyen más jegy) a **vizsgaidőszak első hetében javítható.**
 - A javítás után a **korábbi jegy és az új érdemjegy átlaga képezi a zh jegyét.**
 - A **javítás után is 1-es zh-val** rendelkező hallgató **gyakorlati jegye elégtelen.**Ennek **kijavítására** gyakorlati jegy uv-val van lehetőség a **vizsgaidőszak második hetében.**

**2 ZH:
7. és a 12.
gyakorlaton**

Adatbázisok elérése I.

- **SQLDeveloper elérhetősége:**

- <https://people.inf.elte.hu/nikovits/AB1/sqldeveloper-21.2.1.204.1703-x64.zip>

- **Az adatbázisok eléréséhez szükséges paraméterek:**

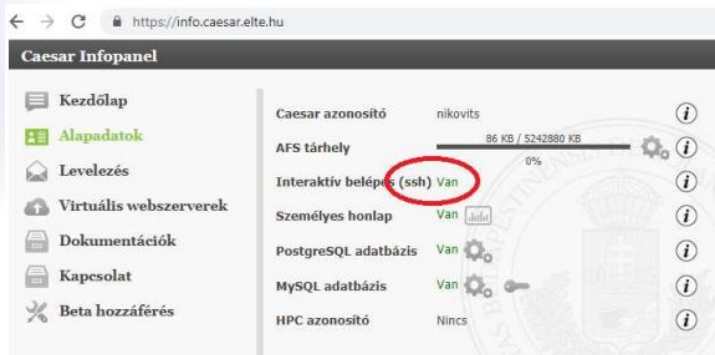
ULLMAN	ARAMIS
host: ullman.inf.elte.hu	host: aramis.inf.elte.hu
port: 1521	port: 1521
service_name: ullman	service_name: aramis

- **Belépési adatok:**

- Kezdetben **neptun kód** a név és jelszó – kezdeti jelszó, lejáratási idő lekérdezhető:
 - `SELECT username, account_status, expiry_date FROM DBA_USERS WHERE username=user;`
- **Jelszó megváltoztatása:**
 - `ALTER USER usernév IDENTIFIED BY új_jelszó;`

Adatbázisok elérése II.

0. lépés

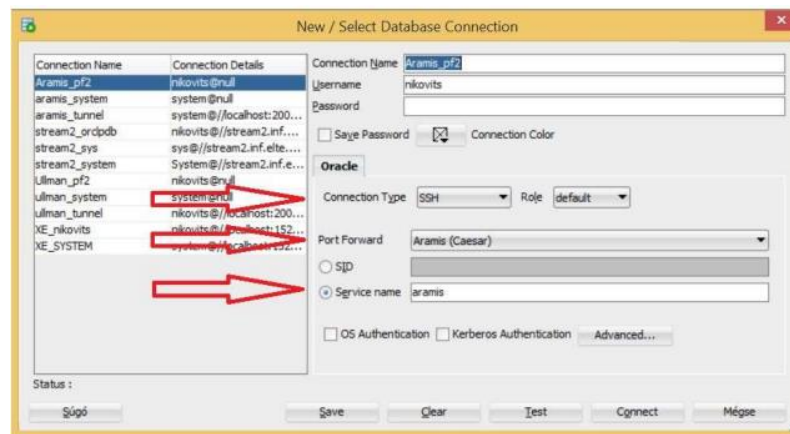
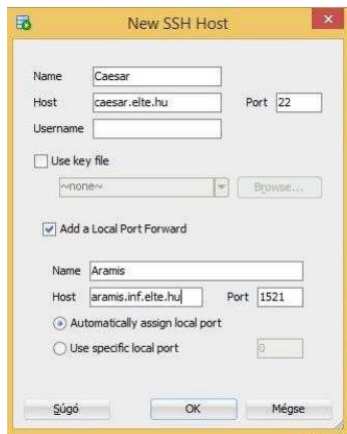
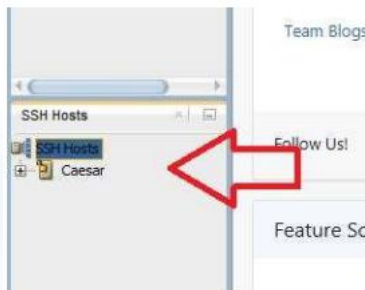


2. lépés

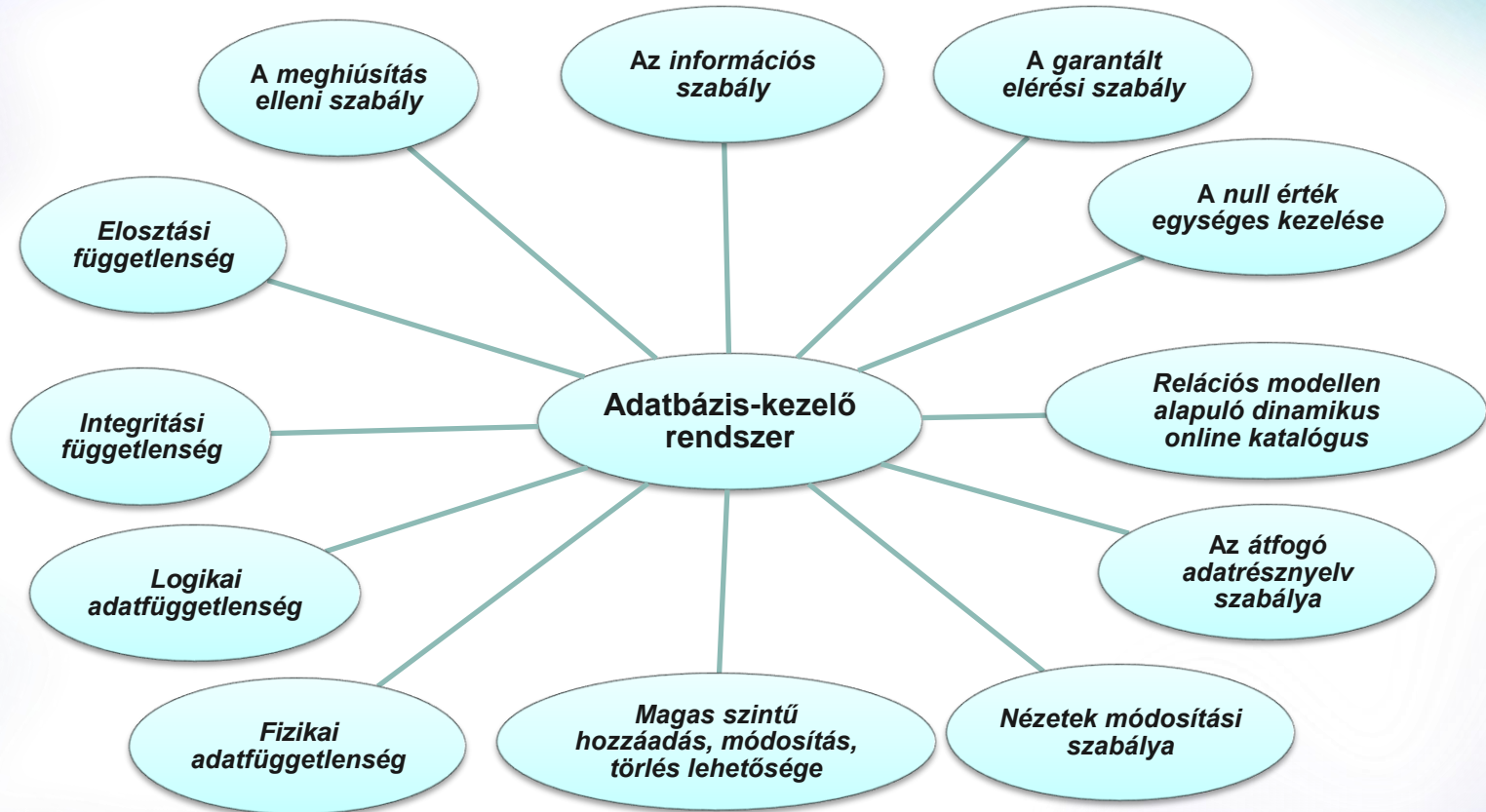


3. lépés

1. lépés



Codd 12 szabálya

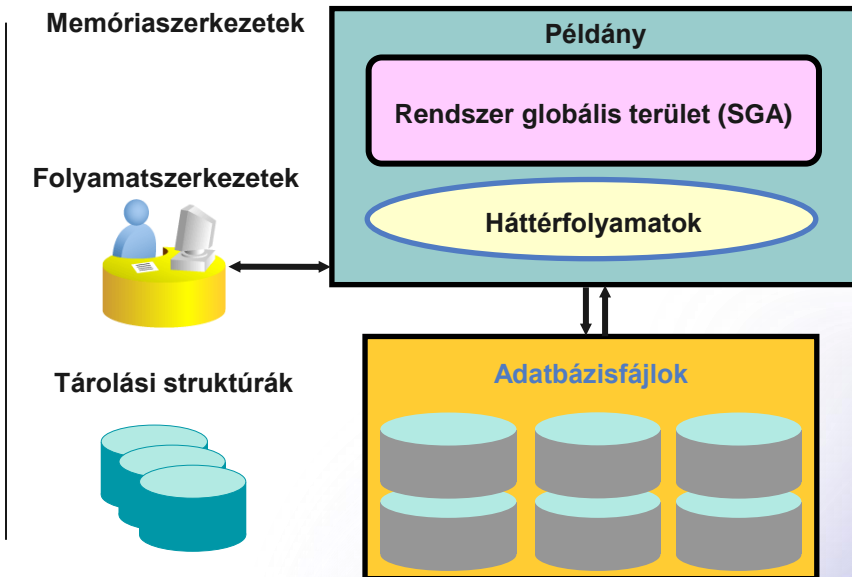


Oracle adatbázis architektúra

- **Egy Oracle szerver:**

- Egy adatbázis-kezelő rendszer, amely nyílt, átfogó és integrált megközelítést kínál az információkezeléshez
- Egy Oracle példányból és egy Oracle adatbázisból áll

- **Adatbázis struktúra:**



Adatszótár nézetek

	Ki kérdezheti le	Tartalom	Részthalmaza	Megjegyzések
DBA_	DBA	Minden	N/A	Tartalmazhat további oszlopokat, amelyek csak az adatbázis adminisztrátorok (DBA) számára hasznosak
ALL_	Mindenki	Minden, amit a felhasználó jogosultságai alapján láthat	DBA_ nézetek	Tartalmazza a felhasználó saját objektumait
USER_	Mindenki	Minden, amit a felhasználó birtokol	ALL_ nézetek	Általában megegyezik az ALL_ nézettel, azzal a különbséggel, hogy hiányzik a TULAJDONOS (OWNER) oszlop.

Példa:

a

```
SELECT table_name, tablespace_name FROM
user_tables;
```

b

```
SELECT sequence_name, min_value, max_value,
increment_by FROM all_sequences WHERE
sequence_owner IN ('MDSYS', 'XDB');
```

c

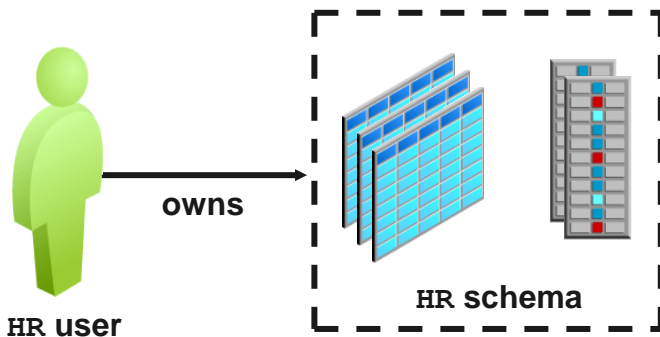
```
SELECT USERNAME, ACCOUNT_STATUS FROM
dba_users WHERE ACCOUNT_STATUS = 'OPEN';
```

d

```
DESCRIBE dba_indexes;
```

Séma és a séma objektumok

- Mi az a séma?



- Séma objektumok

- Az Oracle adatbázisban az **adatbázis séma** logikai adatstruktúrák vagy **séma objektumok** gyűjteménye.
- Az adatbázis séma egy adatbázis felhasználó tulajdonában van, és ugyanaz a neve, mint a megadott **felhasználóneve**.

- Séma objektumok elérése

Database Instance: orcl.oracle.com

[Home](#) [Performance](#) [Administration](#) [Maintenance](#)

Schema

Database Objects

[Tables](#)
[Indexes](#)
[Views](#)
[Synonyms](#)
[Sequences](#)
[Database Links](#)
[Directory Objects](#)
[Reorganize Objects](#)

Users & Privileges

[Users](#)
[Roles](#)
[Profiles](#)
[Audit Settings](#)

Programs

[Packages](#)
[Package Bodies](#)
[Procedures](#)
[Functions](#)
[Triggers](#)
[Java Classes](#)
[Java Sources](#)

Materialized Views

[Materialized Views](#)
[Materialized View Logs](#)
[Refresh Groups](#)

XML Database

[Configuration](#)
[Resources](#)
[Access Control Lists](#)
[XML Schemas](#)
[XMLType Tables](#)
[XMLType Views](#)

BI & OLAP

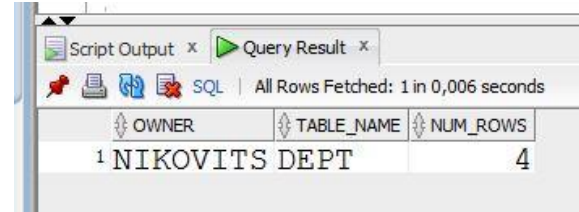
[Dimensions](#)
[Cubes](#)
[OLAP Dimensions](#)
[Measure Folders](#)

Táblák

```
CREATE TABLE dept  
(deptno NUMBER(2), dname VARCHAR2(42), loc VARCHAR2(39));
```

```
SELECT owner, table_name, num_rows  
FROM DBA_TABLES  
WHERE owner='NIKOVITS' AND table_name='DEPT';
```

```
(!) ANALYZE TABLE DEPT COMPUTE STATISTICS;  
(!) ANALYZE TABLE DEPT DELETE STATISTICS;
```

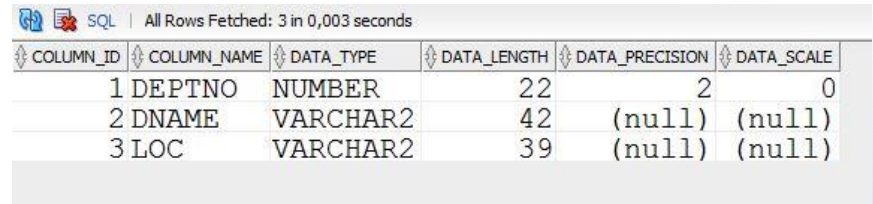


The screenshot shows a SQL query result window with the title 'Query Result'. It displays the results of a query on the DBA_TABLES view, filtered by owner='NIKOVITS' and table_name='DEPT'. The results are shown in a table with three columns: OWNER, TABLE_NAME, and NUM_ROWS. The single row returned is 'NIKOVITS DEPT' with 4 rows.

OWNER	TABLE_NAME	NUM_ROWS
NIKOVITS	DEPT	4

```
CREATE TABLE dept  
(deptno NUMBER(2),  
dname VARCHAR2(42),  
loc VARCHAR2(39));
```

```
SELECT column_id, column_name, data_type,  
       data_length, data_precision, data_scale  
FROM DBA_TAB_COLUMNS  
WHERE owner='NIKOVITS' AND table_name='DEPT';
```



The screenshot shows a SQL query result window with the title 'Query Result'. It displays the results of a query on the DBA_TAB_COLUMNS view, filtered by owner='NIKOVITS' and table_name='DEPT'. The results are shown in a table with six columns: COLUMN_ID, COLUMN_NAME, DATA_TYPE, DATA_LENGTH, DATA_PRECISION, and DATA_SCALE. The three rows returned are for DEPTNO, DNAME, and LOC.

COLUMN_ID	COLUMN_NAME	DATA_TYPE	DATA_LENGTH	DATA_PRECISION	DATA_SCALE
1	DEPTNO	NUMBER	22	2	0
2	DNAME	VARCHAR2	42	(null)	(null)
3	LOC	VARCHAR2	39	(null)	(null)

Nézetek és szinonimák

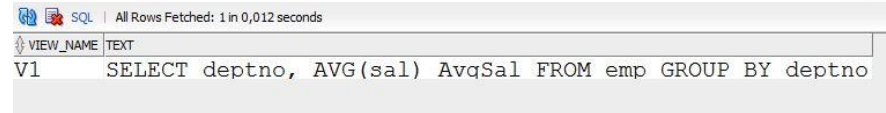
CREATE VIEW v1 AS

SELECT deptno, AVG(sal) AvgSal FROM emp GROUP BY deptno;

SELECT view_name, text

FROM DBA_VIEWS

WHERE owner='NIKOVITS' AND view_name='V1';



SQL | All Rows Fetched: 1 in 0,012 seconds

VIEW_NAME	TEXT
V1	SELECT deptno, AVG(sal) AvgSal FROM emp GROUP BY deptno

CREATE SYNONYM syn1 FOR v1;

SELECT * FROM DBA_SYNONYMS

WHERE owner='NIKOVITS' AND synonym_name='SYN1';



SQL | All Rows Fetched: 1 in 0,009 seconds

OWNER	SYNONYM_NAME	TABLE_OWNER	TABLE_NAME	DB_LINK
NIKOVITS	SYN1	NIKOVITS	V1	(null)

SELECT * FROM syn1 WHERE deptno > 10;

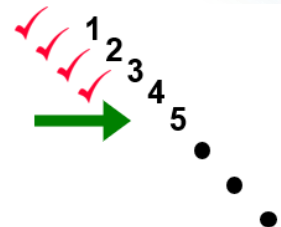


SQL | All Rows

DEPTNO	AVGSAL
30	1800
20	2175

Szekvenciák

- **A szekvencia** egy olyan mechanizmus, amely automatikusan generálja az egymást követő egész számokat egy meghatározott minta szerint.
 - A szekvenciának **van egy neve**, amelyet a következő érték kérésénél használnak.
 - A szekvencia **nincs összekapcsolva** semmilyen konkrét táblával vagy oszloppal.
 - A szekvencia **növekvő vagy csökkenő** irányú lehet.
 - A számok közötti intervallum bármilyen méretű lehet.
 - A szekvencia **újraindulhat**, ha elér egy meghatározott limitet.



```
CREATE SEQUENCE seq1  
MINVALUE 1 MAXVALUE 100 INCREMENT BY 5  
START WITH 50 CYCLE;
```

```
SELECT * FROM DBA_SEQUENCES  
WHERE sequence_name='SEQ1';
```

Következő érték a szekvenciából:

```
INSERT INTO dept VALUES(seq1.NEXTVAL, 'IT', 'Budapest');
```

Aktuális érték a szekvenciából:

```
INSERT INTO emp(deptno, empno, ename, job, sal)  
VALUES(seq1.CURRVAL, 1, 'Tailor', 'SALESMAN', 100);
```

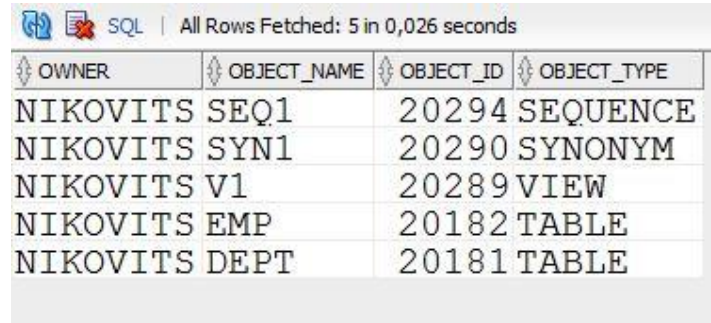
Aktuális érték a szekvenciából:

```
INSERT INTO emp(deptno, empno, ename, job, sal)  
VALUES(seq1.CURRVAL, 2, 'Sailor', 'SALESMAN', 200);
```

SQL All Rows Fetched: 1 in 0,003 seconds									
SEQUENCE_OWNER	SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	CYCLE_FLAG	ORDER_FLAG	CACHE_SIZE	LAST_NUMBER	
NIKOVITS	SEQ1	1	100	5	Y	N	20	50	

Objektumok listája

```
SELECT owner, object_name, object_id, object_type  
FROM DBA_OBJECTS  
WHERE owner='NIKOVITS, and created > sysdate - 1;
```



The screenshot shows a SQL query result in a database client window. The window title is "SQL" and it indicates "All Rows Fetched: 5 in 0,026 seconds". The result is displayed as a table with four columns: OWNER, OBJECT_NAME, OBJECT_ID, and OBJECT_TYPE. The data rows are as follows:

OWNER	OBJECT_NAME	OBJECT_ID	OBJECT_TYPE
NIKOVITS	SEQ1	20294	SEQUENCE
NIKOVITS	SYN1	20290	SYNONYM
NIKOVITS	V1	20289	VIEW
NIKOVITS	EMP	20182	TABLE
NIKOVITS	DEPT	20181	TABLE

PL/SQL ismétlés I.

```
SET SERVEROUTPUT ON;
```

```
DECLARE
```

```
-- Változók, típusok, altípusok
v_number NUMBER := 10;
v_text VARCHAR2(50) := 'Hello World';
SUBTYPE t_percentage IS NUMBER(3,2);
v_percent t_percentage := 50.5;
```

```
-- Rekord típus
```

```
TYPE t_employee IS RECORD (
    emp_id NUMBER,
    emp_name VARCHAR2(50),
    emp_salary NUMBER
);
```

```
v_emp t_employee;
```

```
-- Kollekciónk: Asszociatív tömb (Index by table)
```

```
TYPE t_assoc_array IS TABLE OF VARCHAR2(50) INDEX BY PLS_INTEGER;
v_assoc_arr t_assoc_array;
```

```
-- Dinamikus tömb (Nested Table)
```

```
TYPE t_nested_table IS TABLE OF NUMBER;
v_nested_table t_nested_table := t_nested_table();
```

```
-- Dinamikus tömb (Varray)
```

```
TYPE t_varray IS VARRAY(5) OF VARCHAR2(50);
v_varray t_varray := t_varray();
```

```
-- Képernyőre írás
```

```
PROCEDURE print_message(p_message IN VARCHAR2) IS
BEGIN
    DBMS_OUTPUT.PUT_LINE(p_message);
END;
```

Syntax of PL/SQL Block Structure:

```
DECLARE --optional
    <declarations>
```

```
BEGIN --mandatory
    <executable statements. At least one executable statement is mandatory>
```

```
EXCEPTION --optional
    <exception handler>
```

```
END; --mandatory
```

```
-- Adatbekérés a felhasználótól (Ebben az esetben csak inicializálás)
```

```
PROCEDURE get_user_input IS
```

```
    v_input VARCHAR2(50);
```

```
BEGIN
```

```
    v_input := 'Sample Input'; -- Simulált adatbekérés
```

```
    DBMS_OUTPUT.PUT_LINE('User input: ' || v_input);
```

```
END;
```

```
-- Kurzor használat
```

```
CURSOR c_emp IS
```

```
    SELECT empno, ename, sal FROM emp WHERE deptno = 10;
```

```
-- Kivétel kezelés
```

```
EXCEPTION
```

```
    WHEN NO_DATA_FOUND THEN
```

```
        DBMS_OUTPUT.PUT_LINE('No data found!');
```

```
    WHEN OTHERS THEN
```

```
        DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
```


PL/SQL ismétlés II.

BEGIN

-- Vezérlő utasítások: IF-ELSE, WHILE, FOR

IF v_number > 5 THEN
 print_message('v_number is greater than 5');

ELSE
 print_message('v_number is 5 or less');
END IF;

-- WHILE ciklus

WHILE v_number > 0 LOOP
 v_number := v_number - 1;
 DBMS_OUTPUT.PUT_LINE('Countdown: ' || v_number);
END LOOP;

-- FOR ciklus

FOR i IN 1..3 LOOP
 DBMS_OUTPUT.PUT_LINE('Loop iteration: ' || i);
END LOOP;

-- Kurzor használat rekordokkal

OPEN c_emp;
FETCH c_emp INTO v_emp.emp_id, v_emp.emp_name, v_emp.emp_salary;
IF c_emp%FOUND THEN
 DBMS_OUTPUT.PUT_LINE('Employee fetched: ' || v_emp.emp_name);
ELSE
 DBMS_OUTPUT.PUT_LINE('No employees found');
END IF;
CLOSE c_emp;

-- Asszociatív tömb feltöltése és használata

v_assoc_arr(1) := 'First';
v_assoc_arr(2) := 'Second';
DBMS_OUTPUT.PUT_LINE('Assoc array element 1: ' || v_assoc_arr(1));

-- Dinamikus tömb (Nested Table) kezelése

v_nested_table.EXTEND(3);
v_nested_table(1) := 100;
v_nested_table(2) := 200;
v_nested_table(3) := 300;
DBMS_OUTPUT.PUT_LINE('Nested Table element 2: ' || v_nested_table(2));

-- Dinamikus tömb (Varray) kezelése

v_varray.EXTEND(2);
v_varray(1) := 'First Varray Element';
v_varray(2) := 'Second Varray Element';
DBMS_OUTPUT.PUT_LINE('Varray element 1: ' || v_varray(1));

-- Kollekciónak metódusok használata

IF v_nested_table.EXISTS(1) THEN
 DBMS_OUTPUT.PUT_LINE('Nested Table element 1 exists');
END IF;

-- Adatbekérés és hiba kezelése

BEGIN
 get_user_input;
EXCEPTION
 WHEN OTHERS THEN
 DBMS_OUTPUT.PUT_LINE('Error occurred during input.');

END;

/

Feladatok

Határidő: a következő gyakorlat kezdete!

- **Kötelező feladat az 1. gyakorlathoz**

- Hozzunk létre mindkét adatbázisban (ullman, aramis) egy GYAK01 nevű táblát, aminek egy oszlopa van, és az a NIKOVITS felhasználó olyan tábláinak nevét tartalmazza, amelyek nevében van 'B' betű.

- **Vigyázat !!!** Az alábbiakban ha egy objektumnak vagy egy felhasználónak a neve kisbetűvel szerepel a feladat szövegében, az nem jelenti, hogy ez az adatszótárban is kisbetűvel van tárolva.

Pl. orauser - ORAUSER felhasználó, emp - EMP tábla.

CREATE table proba(o integer); --> A tábla neve az adatszótárakban 'PROBA' lesz.

CREATE table "proba"(o integer); --> A tábla neve az adatszótárakban 'proba' lesz.

Adatbázis objektumok (DBA_OBJECTS)

- Az alábbi lekérdezések segítenek feltérképezni, hogy milyen objektumok vannak egy Oracle adatbázisban, ki a tulajdonosuk, mikor hozták létre azokat, stb. A kérdések után zárójelben az elvárt végeredmény oszlopai szerepelnek.

1. Futtassuk a következő SQL utasítást: SELECT sysdate FROM dual;

- Kinek a tulajdonában van a DUAL nevű tábla? [owner, object_name]
- --> Futtassuk a következő utasításokat:
- SELECT * FROM user_tables; (hozzunk létre egy tetszőleges táblát, majd futtassuk ismét)
- SELECT * FROM all_tables;
- Kinek a tulajdonában van az ALL_TABLES nevű tábla/nézet ? [owner, object_name, object_type]

2. Futtassuk a következő utasításokat:

- `SELECT COUNT(*) FROM all_tables;`
- `SELECT COUNT(*) FROM dba_tables;`
- Miért kapunk különböző értékeket? (lásd a jogosultságokat)
- Kinek a tulajdonában van a DBA_TABLES nevű, illetve a DUAL nevű szinonima? [owner, object_name, object_type]
- Az 1. és 2. feladat megmagyarázza, hogy miért tudjuk elérni a DUAL táblát, illetve a DBA_TABLES nézetet anélkül, hogy minősítenénk őket a tulajdonos nevével így -> tulajdonos.objektum.

3. Az adatbázisban sokféle objektum van: tábla, nézet, index, szinonima ...

- Milyen típusú objektumai vannak az ORAUSER nevű felhasználónak az adatbázisban? [object_type]

4. Hány különböző fajta objektum (tábla, index stb.) van a jelenlegi adatbázisban? [darab]

5. Melyek ezek a típusok? Soroljuk fel egy lekérdezéssel. [object_type]

6. Kik azok a felhasználók, akiknek több mint 10 féle objektumuk van? [owner]

7. Kik azok a felhasználók, akiknek van triggere (TRIGGER vagy trigger ?!) és nézete (VIEW) is? [owner]

--> vigyázzunk az objektumok nevénél a kisbetű/nagybetű-re.

8. Kik azok a felhasználók, akiknek van nézete, de nincs triggere? [owner]

9. Kik azok a felhasználók, akiknek több mint n táblájuk, de maximum m indexük van? [owner]

(n és m értékét adjuk meg úgy, hogy kb. 1-15 között legyen a sorok száma, pl. n=20, m=15)

10. Melyek azok az objektum típusok, amelyek tényleges tárolást igényelnek, vagyis tartoznak hozzájuk adatblokkok?
[object_type] --> Az olyan objektumoknak, amik nem igényelnek tényleges tárolást, pl. nézet, szinonima, csak a definíciója tárolódik az adatszótárban. A megoldáshoz a data_object_id oszlopot vizsgáljuk meg.
11. Melyek azok az objektum típusok, amelyek nem igényelnek tényleges tárolást, vagyis nem tartoznak hozzájuk adatblokkok? [object_type] --> Az utóbbi két lekérdezés metszete nem üres. Vajon miért?
12. Keressük meg az utóbbi két lekérdezés metszetét. [object_type] --> Ezek olyan objektum típusok, amelyekből előfordul adatblokkokkal rendelkező és adatblokkokkal nem rendelkező is.

Táblák oszlopai (DBA_TAB_COLUMNS)

- Az alábbi kérdésekkel egy tábla oszlopait vizsgálhatjuk meg részletesen, vagyis az oszlop nevét, sorszámát (hányadik oszlop), típusát, azt hogy elfogadja-e a NULL értéket, van-e alapértelmezett értéke, stb.
13. Hány oszlopa van a nikovits.emp táblának? [darab]
 14. Milyen típusú a nikovits.emp tábla 6. oszlopa? [data_type]
 15. Adjuk meg azoknak a tábláknak a tulajdonosát és nevét, amelyeknek van 'Z' betűvel kezdődő oszlopa.
[owner, table_name]
 16. Adjuk meg azoknak a tábláknak a tulajdonosát és nevét, amelyeknek legalább 8 darab dátum típusú oszlopa van.
[owner, table_name]

17. Adjuk meg azoknak a tábláknak a tulajdonosát és nevét, amelyeknek 1. es 4. oszlopa is VARCHAR2 típusú, az oszlop hossza mindegy. [owner, table_name]
 18. Írjunk meg egy PLSQL procedúrát, amelyik a paraméterül kapott karakterlánc alapján kiírja azoknak a tábláknak a nevét és tulajdonosát, amelyek az adott karakterlánccal kezdődnek.
(Ha a paraméter kisbetűs, akkor is működjön a procedúra!)
- A fenti procedúra segítségével írjuk ki a Z betűvel kezdődő táblák nevét és tulajdonosát.

```
CREATE OR REPLACE PROCEDURE table_print(p_kar VARCHAR2) IS
```

```
...
```

```
SET SERVEROUTPUT ON
```

```
EXECUTE table_print('Z');
```

Példa

Futtassuk le az alábbi SQL és PL/SQL utasításokat és nézzük meg, hogy mi kerül a táblába.

```
CREATE TABLE test1(col1 INTEGER PRIMARY KEY, col2 VARCHAR2(20)); /
                                                                    BEGIN
                                                                    FOR i IN 1..14 LOOP
                                                                    INSERT INTO test1 VALUES(null, 'trigger'||to_char(i,'FM09'));
                                                                    END LOOP;
                                                                    INSERT INTO test1 VALUES(seq1.currval + 1, 'sequence + 1');
                                                                    COMMIT;
                                                                    END;
                                                                    /
CREATE SEQUENCE seq1
MINVALUE 1 MAXVALUE 100 INCREMENT BY 5 START WITH 50 CYCLE;

CREATE OR REPLACE TRIGGER test1_bir -- before insert row
BEFORE INSERT ON test1
FOR EACH ROW
WHEN (new.col1 is null)
BEGIN
    :new.col1 := seq1.nextval;
END;

SELECT * FROM test1 ORDER BY col2;

DROP TABLE test1; -- a trigger is törölődni fog
DROP sequence seq1; -- a szekvencia nincs a táblához kötve
```