1. FEJEZET

Egyszerű lekérdezések

Elméleti összefoglaló

A SELECT utasítás

A relációs adatbázis-kezelő rendszerek szabványosított nyelve 1986 óta az SQL (Structured Query Language). Ennek legutolsó szabványosított változata az SQL99, illetve a szabványosítás fázisában vannak az adattárházak többdimenziós relációinak lekérdezésére szolgáló multidimenzionális adatbázis-kezelő nyelvek (*lásd* OLAP – On-line Analytical Processing, illetve DMQL – Data Mining Query Language címszavak alatt a szakirodalomban, például [11], [13] és [24]).

Az adatbázisok kezelését az SQL részeit alkotó nyelvek utasításaival végezhetjük. Az adatbázisokat alkotó adattáblák létrehozását, szerkezetük módosítását a DDL (Data Definition Language – adatdefiníciós nyelv), lekérdezését a DQL (Data Query Language – adatlekérdező nyelv), adatokkal való feltöltését és az adatok módosítását a DML (Data Manipulation Language – adatmódosító nyelv), valamint az egyes adattáblák használatának felhasználói jogosítványokkal (úgynevezett jogosultságokkal) való korlátozását a DCL (Data Control Language – adatvezérlő nyelv) végzi.

Az adatlekérdező nyelv legfontosabb utasítása a SELECT, melynek vázlatos felépítése:

```
SELECT SzelekciósLista
  FROM TáblaLista
  [WHERE LogikaiOszlopkifejezés]
  [GROUP BY CsoportosítóOszlopkifejezés-lista]
  [HAVING LogikaiOszlopkifejezés]
  [ORDER BY RendezőOszlopkifejezés-lista];
```

ahol a listaelemeket vesszők választják el. A SELECT utasítás egyes részeit az alábbiakban mutatjuk be.

SzelekciósLista

A *SzelekciósLista* eleme lehet:

- csillag (*) karakter, mely az összes oszlop kijelölésének szimbóluma,
- *Oszlopkifejezés*,
- *Oszlopkifejezés* AS *MásodlagosOszlopnév*,
- *Oszlopkifejezés* AS *"MásodlagosOszlopnév"*.

A fentiekben az *Oszlopkifejezés* oszlopnevekből, konstansokból, műveletekből és oszlopokra vonatkozó egysoros és csoportfüggvényekből álló kifejezés, a *MásodlagosOszlopnév* egy összefüggő (szóközt nem tartalmazó) karaktersorozat (melyet az Oracle nagybetűs alakban jelenít meg), amely az oszlop jelentésére utal, végül a *"MásodlagosOszlopnév"* egy esetlegesen szóközt is tartalmazó karaktersorozat (melyet az Oracle változatlan alakban jelenít meg).

A *SzelekciósLista* kezdődhet a DISTINCT kulcsszóval, ha a többszörös sormegjelenítéseket ki akarjuk szűrni.

TáblaLista

A *TáblaLista* eleme lehet:

- *fizikai, vagy logikai táblanév*,
- *fizikai, vagy logikai táblanév SZÓKÖZ másodlagos táblanév*,
- (*allekérdezés*) *SZÓKÖZ másodlagos táblanév*.

Az allekérdezéssel részletesen a 3. fejezet foglalkozik.

LogikaiOszlopkifejezés

A *LogikaiOszlopkifejezés* oszlopnevekből, konstansokból, műveletekből, egysoros függvényekből és allekérdezésekből álló logikai kifejezés, mely tartalmazhat:

- logikai műveletjeleket
  (AND, OR, NOT),
- hasonlító műveletjeleket
  (>, >=, <, <=, =, <>, !=),
- intervallumvizsgálatot
  (*oszlopkifejezés* BETWEEN *AlsóHatár* AND *FelsőHatár*),
- alsztringvizsgálatot
  (*oszlopkifejezés* LIKE *%alsztring%*),
- allekérdezésre vonatkozó halmazvizsgálatot
  (*oszlopkifejezés* [NOT] IN | ANY | ALL | EXISTS *allekérdezés*),
- NULL-értékre vonatkozó vizsgálatot
  (*oszlopkifejezés* IS NULL | IS NOT NULL).

CsoportosítóOszlopkifejezés-lista

A *CsoportosítóOszlopkifejezés-lista* oszlopnevekből, konstansokból, műveletekből és egysoros függvényekből álló kifejezéseket tartalmazhat.

RendezőOszlopkifejezés-lista

A *RendezőOszlopkifejezés-lista* elemei:

Oszlopkifejezés [ASC | DESC]

ahol az *Oszlopkifejezés* oszlopnevekből, konstansokból, műveletekből és oszlopokra vonatkozó egysoros és csoportfüggvényekből álló kifejezés, vagy az ezeket a szelekciós listában jelölő másodlagos oszlopnév, továbbá az ASC a kódtábla szerint növekvő,-a DESC pedig a csökkenő rendezést írja elő (az ASC az alapértelmezés).

Megjegyzések a SELECT utasításhoz

- A megkülönböztetés érdekében szükség lehet *minősített oszlopnevek* használatára is, melyek alakja: *tulajdonosnév.táblanév.oszlopnév*, vagy *táblanév.oszlopnév* (például scott.emp.job, vagy emp.job).
- Előfordulhat, hogy egy listázásnál szeretnénk az összes oszlopon kívül még néhány kiszámított (pszeudó) oszlopot is megjeleníteni. Ekkor a * szimbólumot minősített névként használjuk (például SELECT emp.*, sal+NVL(comm,0) FROM emp;).
- A szelekciós lista valamely oszlopkifejezése akkor redukálódik egyetlen konstansra, ha azt minden sorba ki szeretnénk íratni. Ilyen eset fordulhat elő akkor, ha a feladat ugyanazt a konstans értéket tartalmazza megszorításként az előállítandó lista minden sorára (például a „listázza ki mindazon clerk foglalkozású..." jellegű feladatok). Az ilyen konstansok visszaírása a listára (kiemelve a tábla fejlécébe) – bár látszólag felesleges helyet foglalnak – lényegesen javítják annak értelmezését (*lásd* később).
- A szelekciós lista tartalmazhat úgynevezett egyértékű allekérdezést is (*lásd* 3. fejezet).
- Célszerű a rendező oszlopkifejezés-lista elemeit a szelekciós listában feltüntetni a lista könnyebb érthetősége érdekében.
- Ha egy SELECT utasítás tartalmaz GROUP BY utasításrészt, akkor a szelekciós listában csak olyan oszlop, vagy olyan oszlopra vonatkozó egysoros függvény szerepelhet, amely oszlop a GROUP BY utasításrészben is szerepel, továbbá csak olyan oszlopra vonatkozó csoportfüggvény szerepelhet, amely oszlop a GROUP BY utasításrészben *nem* szerepel.
- A HAVING utasításrészben álló logikai oszlopkifejezésben csak olyan oszlopkifejezés szerepelhet, amely a GROUP BY utasításrészben is szerepel.
- A rendező oszlopkifejezéssel ellentétben a WHERE, a GROUP BY és a HAVING utasításrészekben másodlagos oszlopnév nem szerepelhet.

1.1. példa

Listázza ki a manager foglalkozású dolgozók nevét, belépési idejét, részlegének azonosítóját a nevek szerint csökkenően rendezve. (A01-01.sql)

1. megoldás

```
SELECT ename          AS "A dolgozó neve",
       'MANAGER     ' AS "foglalkozása",
```

```
        hiredate        AS "belépési dátuma",
        deptno          AS "részlegének azonosítója"
    FROM emp
    WHERE UPPER(job) = 'MANAGER'
    ORDER BY ename DESC;
```

Eredmény

```
A dolgozó  foglalkozása belépési  részlegének azonosítója
---------- ------------ --------- ----------------------
JONES      MANAGER      81-ÁPR-02                    20
CLARK      MANAGER      81-JÚN-09                    10
BLAKE      MANAGER      81-MÁJ-01                    30
```

Megjegyzés

A kiíratás láthatóan nem pontosan a kívánt oszlopfejlécekkel történt. Ennek az az oka, hogy az SQL*Plus-környezet alapértelmezése szerint az oszlopnevek az egyes oszlopok deklarációja szerinti mezőszélességben kerülnek kiírásra. A karakter típusú adatokat (ideértve a dátumadatokat is) balra, a szám típusú adatokat pedig alapértelmezés szerint jobbra igazítja a rendszer. A kiírandó mezőszélesség megváltoztatása, illetve a speciális oszlopfejlécek kiíratása az SQL*Plus formázási utasításaival lehetséges (*lásd* 4. fejezet).

Megjegyezzük, hogy a `HEADING` utasításrészben szereplő fejlécszöveghez egyaránt használhatjuk a " és a ' szimbólumokat.

Ezek után nézzük a jó megoldást.

2. megoldás (A jó megoldás)

```
COLUMN ename    FORMAT A14
COLUMN ename    HEADING "A dolgozó neve"
COLUMN hiredate FORMAT A15
COLUMN hiredate HEADING 'belépési dátuma'
COLUMN deptno   HEADING 'részlegének azonosítója'

SELECT ename,
       'MANAGER     ' AS "foglalkozása",
       hiredate,
       deptno
  FROM emp
  WHERE UPPER(job) = 'MANAGER'
  ORDER BY ename DESC;

CLEAR COLUMNS

SELECT ename,
       'MANAGER     ' AS "foglalkozása",
       hiredate,
```

```
      deptno
FROM emp
WHERE UPPER(job) = 'MANAGER'
ORDER BY ename DESC;
```

Eredménytáblák

```
A dolgozó neve foglalkozása belépési dátuma részlegének azonosítója
-------------- ------------ ---------------- ------------------------
JONES          MANAGER      81-ÁPR-02                              20
CLARK          MANAGER      81-JÚN-09                              10
BLAKE          MANAGER      81-MÁJ-01                              30

ENAME      foglalkozása HIREDATE      DEPTNO
---------- ------------ --------- ----------
JONES      MANAGER      81-ÁPR-02         20
CLARK      MANAGER      81-JÚN-09         10
BLAKE      MANAGER      81-MÁJ-01         30
```

Megjegyzés

A fenti példában a numerikus oszlop fejlécének kiíratásához a számadatok automatikus jobbra tömörítése miatt nem volt szükség a mezőszélesség beállítására.

Ne felejtkezzünk el a listázást követően a formázásnak az alapértelmezés szerinti alakra való visszaállításáról a `CLEAR COLUMNS` parancs segítségével.

Speciális függvények

Az alábbiakban bemutatásra kerülő sorfüggvények elsősorban a táblaadatok megjelenítésére szolgálnak, de bizonyos esetekben a speciális adatok beviteléhez is használhatjuk őket.

NVL függvény

Az `NVL` függvény a `NULL` értéket (a „hiányzó" értéket) tényleges értékké alakítja át. A függvény visszatérési értéke a bal oldali paraméterének aktuális értéke *(operandus)*, ha az nem `NULL`, egyébként a visszaadott érték a *helyettesítő* paraméter értéke lesz.

A függvény alakja:

`NVL(`*operandus*`,` *helyettesítő*`)`

Az *operandus* egy oszlopnév, a *helyettesítő* lehet literál, oszlopnév vagy kifejezés. A *helyettesítő* adattípusának meg kell egyeznie az *operandus* adattípusával. Az `NVL` függvény numerikus, dátum és karakteres adatoknál használható. Például `NVL(fizetés,0)`, `NVL(belépés,sysdate)`, `NVL(munkakör,'még nincs')`.

Az NVL függvény azért szükséges, mert ha egy kifejezésben NULL érték szerepel, akkor a kifejezés értéke is NULL lesz, az NVL függvény használatával azonban az eredmény már értékelhetővé válik.

DECODE függvény

Egy lista eredményének áttekinthetőségét növelhetjük a DECODE függvény használatával, melynek szintaktikája az alábbi:

DECODE(*oszlopkifejezés*, *h1*,*t1* [, *h2*,*t2*]..., *kifejezés*)

A kiértékelés soronként megvizsgálja az aktuális adattáblára az *oszlopkifejezés*-t, mely egy vagy több oszlopnevet is tartalmazó egysoros függvény. Ha az *oszlopkifejezés* értéke *h1*, akkor a függvény értéke *t1* stb., egyébként pedig a *kifejezés* értéke (ahol ez utóbbi hivatkozhat valamely oszlopra, rendszerváltozóra, de lehet konstans kifejezés is).

CASE kifejezés

A CASE kifejezés segítségével létrehozhatunk olyan pszeudooszlopot, melynek az értéke valamilyen kifejezéstől függ. Ennek szintaktikája:

```
CASE
  WHEN LogikaiKifejezés  THEN VisszatérőKifejezés
  [WHEN LogikaiKifejezés  THEN VisszatérőKifejezés]...
  [ELSE VisszatérőKifejezés]
END
```

illetve

```
CASE oszlopkifejezés
  WHEN érték  THEN VisszatérőKifejezés
  [WHEN érték  THEN VisszatérőKifejezés]...
  [ELSE VisszatérőKifejezés]
END
```

A CASE kifejezés használatakor az első esetben a rendszer megkeresi az első WHEN-THEN párost, és megvizsgálja a *LogikaiKifejezés* értékét. Ha ez igaz (TRUE), akkor a függvény értéke a *VisszatérőKifejezés* lesz, ha hamis (FALSE), akkor megy tovább a következő WHEN-THEN párosra. Amennyiben ezek egyikében sem talált egyezőséget, a függvény értéke az ELSE ág *VisszatérőKifejezése* lesz. A *LogikaiKifejezés* természetesen tartalmazhat oszlopkifejezést is.

A CASE kifejezés második alakjánál az az aktuális *VisszatérőKifejezés*, amelyikhez tartozó WHEN melletti értéket felveszi a CASE mellett álló *oszlopkifejezés*.

Megjegyezzük, hogy a CASE kifejezés matematikai értelemben függvénynek is tekinthető, hiszen az *oszlopkifejezést* mint paramétert leképezi a *VisszatérőKifejezés* értékére.

Megjegyzés

A DECODE függvény és a CASE kifejezés sok tekintetben hasonlóak. Kölcsönösen egymásba ágyazhatók, és mindkettőnél ügyelni kell, hogy az összehasonlított, és a visszaadott értékek azonos típusúak legyenek, illetve ez utóbbi lehet NULL érték is. Az összehasonlításnál ügyeljünk arra, hogy ha a vizsgált oszlopkifejezés esetleges NULL értékére szeretnénk feltételt adni, akkor az NVL függvénnyel egy olyan értéket rendeljünk a NULL értékhez, mely az értéktartományos kívül esik (*lásd* az alábbi példát).

A CASE kifejezés többlettudása többek között annak köszönhető, hogy a benne szereplő *LogikaiKifejezés* tartalmazhat úgynevezett allekérdezést is (*lásd* 3. fejezet).

1.2. példa

Listázza ki a dolgozók nevét, munkakörét és jutalékát oly módon, hogy akinek nincs jutaléka, annál azt írja ki, hogy "Nem jár jutalék".

1. megoldás (DECODE függvénnyel)

```
SELECT ename          AS Neve,
       job            AS Munkakör,
       DECODE(NVL(comm,-1),
         -1,  'Nem jár jutalék',
         comm)
                      AS Jutalék
  FROM emp;
```

2. megoldás (CASE kifejezéssel – oszlopkifejezéssel)

```
SELECT ename          AS Neve,
       job            AS Munkakör,
       CASE NVL(comm,-1)
         WHEN -1   THEN 'Nem jár jutalék'
         ELSE TO_CHAR(comm)
       END            AS Jutalék
  FROM emp;
```

3. megoldás (CASE kifejezéssel – oszlopkifejezés nélkül)

```
SELECT ename          AS Neve,
       job            AS Munkakör,
       CASE
         WHEN NVL(comm,-1) = -1
           THEN 'Nem jár jutalék'
         ELSE TO_CHAR(comm)
       END            AS Jutalék
  FROM emp;
```

Eredmény (mindhárom esetben)

```
NEVE       MUNKAKÖR  JUTALÉK
---------- --------- ----------------------------------------
KING       PRESIDENT Nem jár jutalék
BLAKE      MANAGER   Nem jár jutalék
CLARK      MANAGER   Nem jár jutalék
JONES      MANAGER   Nem jár jutalék
MARTIN     SALESMAN  1400
ALLEN      SALESMAN  300
TURNER     SALESMAN  0
JAMES      CLERK     Nem jár jutalék
WARD       SALESMAN  500
FORD       ANALYST   Nem jár jutalék
SMITH      CLERK     Nem jár jutalék
SCOTT      ANALYST   Nem jár jutalék
ADAMS      CLERK     Nem jár jutalék
MILLER     CLERK     Nem jár jutalék

14 sor kijelölve.
```

Megjegyzés

- Az `NVL` függvénnyel a `comm` oszlop `NULL` értékéhez a `-1` értéket rendeltük, mert ez kívül esik a `comm` megengedett értéktartományán. Ha például a `0` értéket használtuk volna hozzárendelési értékként, akkor `Turner` esetén is a `"Nem jár jutalék"` üzenet jelent volna meg, holott e dolgozónak általában lehet jutaléka, csak éppen ebben a hónapban nem volt (azaz `0` volt).
- Figyeljünk fel arra, hogy a `CASE` használata esetén szükség volt a `TO_CHAR` konvertáló függvény használatára is.

Karakterkezelő függvények

Az SQL karakterkezelő függvényei igen sokoldalúak; alapvetően megjelenítéshez használjuk (*lásd* 1.5 példa: Helytakarékos listázás, vagy 1.8. példa: Formázott kiíratás az SQL-ben), de szövegkeresére éppúgy alkalmasak (*lásd* 1.3. és 1.4. példa: Szövegkeresés), mint diagramok kvázigrafikus megjelenítésére (*lásd* 1.6. és 1.7. példa: „Grafikus" megjelenítés).

A karakterkezelő függvények bemenő és kimenő paramétere karakter, karaktersorozat (kivétel a `CHR` és a `LENGTH` függvény).

Az Oracle-rendszer telepítésénél célszerű az alapértelmezett (`default`) adatbázis karakterkészletet választani (*lásd* 1. melléklet), és feltétlenül kerüljük az `Unicode` (UTF8) karakterkészlet beállítást, mivel még a 9.2-es verzióban is a karakterkezelő függvények ez utóbbit hibásan kezelik. (Ezen kívül hibásan kezeli még az SQL*Plus `COLUMN` utasítása is, *lásd* 4. fejezet.)

Az alábbi karakterkezelő függvények mind SQL-függvények, így nem csupán az SQL*Plus-környezetben használhatók (részletesen *lásd* [16]):

LOWER({*oszlop* | *kifejezés*})	Karakterlánc minden betűjét kisbetűvé alakítja át.
UPPER({*oszlop* | *kifejezés*})	Karakterlánc minden betűjét nagybetűvé alakítja át.
INITCAP({*oszlop* | *kifejezés*})	Minden szó első betűjét nagybetűvé, a többit kisbetűvé alakítja át.
CONCAT({*oszlop1* | *kifejezés1*}, {*oszlop2* | *kifejezés2*})	A két megadott karakterláncot összefűzi. Hatása azonos az összefűzés (||) operátorral.
SUBSTR({*oszlop* | *kifejezés*}, *m* [, *n*])	A karakterlánc *m*-edik pozíciójától kezdődően *n* karaktert ad vissza (*n* hiánya esetén az összeset).
LENGTH({*oszlop* | *kifejezés*})	A karakterlánc hosszát adja vissza.
INSTR({*oszlop* | *kifejezés*}, *minta* [, *m* [, *n*]])	A *minta* karaktersorozatnak az első paraméterként megadott karaktersorozatbeli pozícióját adja vissza. Ha a visszatérő érték nulla, akkor nem talált. Az opcionális paraméterek jelentése: *m*: kezdőpozíció, *n*: *n*-edik előfordulás:
LPAD({*oszlop* | *kifejezés*}, *n* [, '*kitöltő*'])	Jobbra igazítja a karakterláncot, és balról kiegészíti a *kitöltő* karakterrel oly módon, hogy a hossza éppen *n* legyen. A *kitöltő* karakter hiánya esetén szóköz karaktert használ.
RPAD({*oszlop* | *kifejezés*}, *n* [, '*kitöltő*'])	Balra igazítja a karakterláncot, és jobbról kiegészíti a *kitöltő* karakterrel oly módon, hogy a hossza éppen *n* legyen. A *kitöltő* karakter hiánya esetén szóköz karaktert használ.
CHR(*karakterkód*)	A decimálisan megadott kódú karaktert adja vissza. Néhány gyakran használt karakter kódja: soremelés (LF): 10, (vezérlő karakter) tabulálás (TAB): 9, (vezérlő karakter) szóköz (SPACE): 32.

1.3. példa (Szövegkeresés)

Az alábbi példában bemutatjuk a SUBSTR függvény használatát.

1. A SUBSTR függvény alkalmazása adattábla oszlopon hosszmegadás nélkül

```
SELECT ename, SUBSTR(ename,1)
  FROM emp
  WHERE deptno = 10;
```

Eredmény

```
ENAME      SUBSTR(ENA
---------- ----------
KING       KING
```

```
CLARK      CLARK
MILLER     MILLER
```

2. Alkalmazás adattábla oszlopon hosszmegadással

```
SELECT ename, SUBSTR(ename,3,4)
  FROM emp
  WHERE deptno = 10;
```

Eredmény

```
ENAME      SUBS
---------- ----
KING       NG
CLARK      ARK
MILLER     LLER
```

3. Alkalmazás karakteres konstanson

```
SELECT SUBSTR('ABRAKADABRA',3,5)
  FROM dual;
```

Eredmény

```
SUBST
-----
RAKAD
```

1.4. példa (Szövegkeresés)

Az alábbi példában bemutatjuk az `INSTR` függvény használatát.

1. Az `INSTR` függvény használata az opcionális paraméterek nélkül

1A. Alkalmazás adattábla oszlopon

```
SELECT ename, INSTR(ename,'AR')
  FROM emp;
```

Eredmény

```
ENAME      INSTR(ENAME,'AR')
---------- -----------------
KING                       0
BLAKE                      0
CLARK                      3
JONES                      0
MARTIN                     2
ALLEN                      0
TURNER                     0
JAMES                      0
```

```
WARD                    2
FORD                    0
SMITH                   0
SCOTT                   0
ADAMS                   0
MILLER                  0

14 sor kijelölve.
```

1B. Alkalmazás karakteres konstanson

```
SELECT INSTR('ABRADAKADRA','AD')
  FROM dual;
```

Eredmény

```
INSTR('ABRADAKADRA','AD')
-------------------------
                        4
```

2. Az INSTR függvény használata az opcionális paraméterekkel

2A. Az első előfordulás az első karaktertől

```
SELECT INSTR('ABRADAKADRA','AD',1)
  FROM dual;
```

Eredmény

```
INSTR('ABRADAKADRA','AD',1)
---------------------------
                          4
```

2B. A második előfordulás az első karaktertől

```
SELECT INSTR('ABRADAKADRA','AD',1,2)
  FROM dual;
```

Eredmény

```
INSTR('ABRADAKADRA','AD',1,2)
-----------------------------
                            8
```

2C. Az első előfordulás az ötödik karaktertől

```
SELECT INSTR('ABRADAKADRA','AD',5)
  FROM dual;
```

Eredmény

```
INSTR('ABRADAKADRA','AD',5)
---------------------------
                          8
```

2D. A második előfordulás az ötödik karaktertől

```
SELECT INSTR('ABRADAKADRA','AD',5,2)
  FROM dual;
```

Eredmény

```
INSTR('ABRADAKADRA','AD',5,2)
-----------------------------
                            0
```

1.5. példa (Helytakarékos listázás)

Listázza ki az 1.2. példa feladatának megoldását „helytakarékos" módon úgy, hogy a szöveges megjegyzés kiírásának csak 16 karaktert engedélyez (a SUBSTR függvény használatával).

1. megoldás (DECODE függvénnyel)

```
SELECT ename          AS Neve,
       job            AS Munkakör,
       DECODE(NVL(comm,-1),
         -1,  'Nem jár jutalék',
         SUBSTR(TO_CHAR(comm),1,16))
                      AS Jutalék
  FROM emp;
```

2. megoldás (CASE kifejezéssel)

```
SELECT ename          AS Neve,
       job            AS Munkakör,
       CASE NVL(comm,-1)
         WHEN -1 THEN 'Nem jár jutalék'
         ELSE SUBSTR(TO_CHAR(comm),1,16)
       END            AS Jutalék
  FROM emp;
```

Eredmény (mindkét esetben)

```
NEVE       MUNKAKÖR  JUTALÉK
---------- --------- ----------------
KING       PRESIDENT Nem jár jutalék
BLAKE      MANAGER   Nem jár jutalék
CLARK      MANAGER   Nem jár jutalék
JONES      MANAGER   Nem jár jutalék
```

```
MARTIN      SALESMAN  1400
ALLEN       SALESMAN  300
TURNER      SALESMAN  0
JAMES       CLERK     Nem jár jutalék
WARD        SALESMAN  500
FORD        ANALYST   Nem jár jutalék
SMITH       CLERK     Nem jár jutalék
SCOTT       ANALYST   Nem jár jutalék
ADAMS       CLERK     Nem jár jutalék
MILLER      CLERK     Nem jár jutalék

14 sor kijelölve.
```

1.6. példa („Grafikus" megjelenítés)

Listázza ki az összes alkalmazott nevét és fizetését egy oszlopban a fizetésük szerint csökkenően. A fizetést jelenítse meg úgy, hogy minden 1000 USD-t egy # szimbólum jelöljön. Legyen a két oszlop között az elválasztó jel a kettőspont. A megjelenítendő oszlopnak adjon értelmes másodlagos nevet. (Használja az LPAD és RPAD függvényeket.)

1. megoldás (Az RPAD függvény és „konstans" oszlop használatával)

```
SELECT RPAD(ename,6) ||
       RPAD(':', ROUND((sal/1000),0)+1, '#')
         AS "Az alkalmazottak és fizetésük"
  FROM emp
  ORDER BY sal DESC;
```

Eredmény

```
Az alkalmazottak és fizetésük
------------------------------------------------------------------------
KING  :#####
FORD  :###
SCOTT :###
JONES :###
BLAKE :###
CLARK :##
ALLEN :##
TURNER:##
MILLER:#
MARTIN:#
WARD  :#
ADAMS :#
JAMES :#
SMITH :#
```

```
14 sor kijelölve.
```

Megjegyzés

- Az RPAD függvényt ekkor a következőképpen használtuk. „Oszlopnévként" a „:" konstanst megadva, természetesen a „:" konstans oszlopértékeket kaptuk. Mivel kitöltő karakterként a „#" konstanst írtuk elő, ezért a második paraméterként megadott mezőhossz (melyhez természetesen hozzáadtunk 1-et a „:" oszlopérték miatt) tulajdonképpen egy „grafikus" függvényábrázolást eredményez.
- Az RPAD(ename,6) megjelenítéséből látható, hogy a kitöltő karakter megadásának elhagyása esetén, az alapértelmezés a szóköz karakter.
- A fenti megjelenítést javíthatjuk egyrészt a nevek mellé helyezett elválasztó szóközökkel, másrészt a SUBSTR függvény mezőszélesség-korlátozó hatásával.

2. megoldás

```
SELECT SUBSTR(RPAD(ename,6) || RPAD(' ',5) ||
              RPAD(':', ROUND((sal/1000),0)+1, '#'), 1, 30)
         AS "Az alkalmazottak és fizetésük"
  FROM emp
  ORDER BY sal DESC;
```

Eredmény

```
Az alkalmazottak és fizetésük
------------------------------
KING       :######
FORD       :####
SCOTT      :####
JONES      :####
BLAKE      :####
CLARK      :##
ALLEN      :##
TURNER     :##
MILLER     :#
MARTIN     :#
WARD       :#
ADAMS      :#
JAMES      :#
SMITH      :#

14 sor kijelölve.
```

1.7. példa („Grafikus" megjelenítés)

Oldja meg az előző feladatot oly módon, hogy a fizetés grafikus kiíratása után kiírja azt numerikusan is, továbbá az egyes dolgozók munkakörét is. A lista formázása legyen „helytakarékos".

1. megoldás

```
SELECT SUBSTR(RPAD(ename,6) || RPAD(' ',7) ||
              RPAD(': ', ROUND((sal/1000),0)+2, '#'), 1, 25) ||
              LPAD(sal, 8)
             AS "Dolgozó neve : fizetése",
       job  AS "munkaköre"
  FROM emp
  ORDER BY sal DESC;
```

Eredmény

```
Dolgozó neve : fizetése          munkaköre
-------------------------------- ---------
KING         : #####  5000      PRESIDENT
FORD         : ###    3000      ANALYST
SCOTT        : ###    3000      ANALYST
JONES        : ###    2975      MANAGER
BLAKE        : ###    2850      MANAGER
CLARK        : ##     2450      MANAGER
ALLEN        : ##     1600      SALESMAN
TURNER       : ##     1500      SALESMAN
MILLER       : #      1300      CLERK
MARTIN       : #      1250      SALESMAN
WARD         : #      1250      SALESMAN
ADAMS        : #      1100      CLERK
JAMES        : #       950      CLERK
SMITH        : #       800      CLERK

14 sor kijelölve.
```

2. megoldás

```
SELECT SUBSTR(RPAD(ename,6) || RPAD(' ',7) ||
              RPAD(': ', ROUND((sal/1000),0)+2, '#') ||
              RPAD(' ', 5-ROUND((sal/1000),0)) ||
              LPAD(sal, 6), 1, 27)
             AS "Dolgozó neve :   fizetése",
       job  AS "munkaköre"
  FROM emp
  ORDER BY sal DESC;
```

Eredmény

```
Dolgozó neve :   fizetése   munkaköre
-------------------------- ---------
KING         : #####  5000  PRESIDENT
FORD         : ###    3000  ANALYST
SCOTT        : ###    3000  ANALYST
JONES        : ###    2975  MANAGER
BLAKE        : ###    2850  MANAGER
CLARK        : ##     2450  MANAGER
ALLEN        : ##     1600  SALESMAN
TURNER       : ##     1500  SALESMAN
MILLER       : #      1300  CLERK
MARTIN       : #      1250  SALESMAN
WARD         : #      1250  SALESMAN
ADAMS        : #      1100  CLERK
JAMES        : #       950  CLERK
SMITH        : #       800  CLERK

14 sor kijelölve.
```

1.8. példa (Adatformázás SQL-ben)

Listázza a fizetés szerint csökkenően rendezve az eladók (`salesman`) és a hivatalnokok (`clerk`) nevét, munkakörét, fizetését, jutalékát, valamint a jutalék/fizetés arányukat.

1. megoldás

```
SELECT ename     AS DolgozóNeve,
       job       AS Munkaköre,
       sal       AS Fizetése,
       comm      AS Jutaléka,
       CASE NVL(comm,0)
         WHEN 0  THEN NULL
         ELSE NVL(comm,0)/sal
       END       AS Jutalékaránya
  FROM emp
  WHERE UPPER(job) IN ('SALESMAN','CLERK')
  ORDER BY sal DESC;
```

Eredmény

```
DOLGOZÓNEV MUNKAKÖRE   FIZETÉSE   JUTALÉKA JUTALÉKARÁNYA
---------- --------- ---------- ---------- -------------
ALLEN      SALESMAN        1600        300         .1875
TURNER     SALESMAN        1500          0
MILLER     CLERK           1300
```

```
MARTIN      SALESMAN        1250        1400          1.12
WARD        SALESMAN        1250         500            .4
ADAMS       CLERK           1100
JAMES       CLERK            950
SMITH       CLERK            800

8 sor kijelölve.
```

2. megoldás (Formázott kiírással)

```
SELECT RPAD(ename,11)          AS "DolgozóNeve",
       SUBSTR(job,1,9)         AS "Munkaköre",
       LPAD(sal,8)             AS "Fizetése",
       LPAD(comm,8)            AS "Jutaléka",
       CASE NVL(comm,0)
         WHEN 0  THEN NULL
         ELSE TO_CHAR(NVL(comm,0)/sal,'999999990.99')
       END                     AS "JutalékAránya"
  FROM emp
  WHERE UPPER(job) IN ('SALESMAN','CLERK')
  ORDER BY sal DESC;
```

Eredmény

```
DolgozóNeve Munkaköre Fizetése Jutaléka JutalékAránya
----------- --------- -------- -------- -------------
ALLEN       SALESMAN      1600      300          0.19
TURNER      SALESMAN      1500        0
MILLER      CLERK         1300
MARTIN      SALESMAN      1250     1400          1.12
WARD        SALESMAN      1250      500          0.40
ADAMS       CLERK         1100
JAMES       CLERK          950
SMITH       CLERK          800

8 sor kijelölve.
```

Dátumok és számok formázott megjelenítése

Az aktuális dátumot (és természetesen az időt is) a rendszerdátumból (`sysdate`) állíthatjuk elő. Karakterláncból (vagy számból) dátumformátumot a `TO_DATE` függvénnyel, dátumból vagy számból formázott karakterláncot pedig a `TO_CHAR` függvénnyel állíthatunk elő. (E függvények SQL-eszközök, tehát nem csupán SQL*Plus-környezetben használhatók.) A dátumértékek között végezhetünk kivonást az eltelt idő meghatározására. (Mindezt részletesen *lásd* [12] és [16]. Az SQL*Plus formázási lehetőségeit a 4. fejezetben mutatjuk be.)

Használatuk szintaktikája:

TO_DATE({*karakterlánc* | *szám*} [,'*formátummaszk*'])

TO_CHAR({*dátum* | *szám*} [,'*formátummaszk*'])

ahol:

- formátummaszk hiányában dátum esetén az alapértelmezett dátumforma szerint, szám esetén pedig az ábrázoláshoz szükséges méret szerint történik a konverzió,
- a formátummaszk elemei az alábbi táblázatok dátum- és számformátumaiból állíthatók össze.

Formátummaszkok:

A TO_DATE és a TO_CHAR függvényekben egyaránt használható *dátumformátum* elemek (a felsorolás nem teljes):

Formátum	Leírás
YYYY	A teljes évszám.
YEAR	A teljes évszám betűkkel.
MM	A hónap neve két számjeggyel.
MONTH	A hónap teljes neve.
MON	A hónap nevének három nagybetűs rövidítése.
mon	A hónap nevének három kisbetűs rövidítése.
WW	A hét sorszáma az évben.
W	A hét sorszáma a hónapban.
DDD	A nap sorszáma az évben.
DD	A nap sorszáma a hónapban.
D	A nap sorszáma a héten.
DY	A hét napjának hárombetűs rövidítése.
DAY	A nap teljes neve.
HH	Az óra (1-12).
HH12	Az óra (1-12).
HH24	Az óra (1-24).
MI	A perc.
SS	A másodperc.
SSSSS	Az éjfél óta eltelt másodpercek száma.
/,-:._ *szóköz*	A dátumelemek között használható elválasztó karakterek (a TO_CHAR elhelyezi, a TO_DATE figyelmen kívül hagyja).

A TO_CHAR függvényben alkalmazható *számformátum* elemek (ez sem teljes):

Formátum	Leírás
9999000	A nullák vagy kilencesek száma határozza meg a megjeleníthető számjegyek számát (a mellékelt példa 7 jegyű egész szám kiíratását engedélyezi, az első három helyiértéken vezető nullával).
99999	5 értékes számjegy, vezető nullák nem jelennek meg.
09999	5 értékes számjegy, ugyanennyi vezető nullával együtt.
0999.999	4 egészjegy, vezető nullával, három tizedessel.
S9999	Az S karakter helyén az előjel jelenik meg.
$9999	Minden szám elé $ jel kerül.
L999	Az L karakter helyén a helyi pénznem jelenik meg.
.	A tizedespont jele.
,	Az ezres csoportosítás elválasztó jele.

Rendszerdátum és -idő formátumának beállítása

A rendszerdátum és -idő formátumának beállítása az

```
ALTER SESSION SET NLS_DATE_FORMAT = 'formátummaszk';
```

utasítással történhet, ahol a *formátummaszk* a fenti dátumformátum elemekből állítható össze.
Rendszerdátum formátumának beállítása nap felbontásban (egy lehetséges eset)

```
ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY.MM.DD';
```

Rendszerdátum formátumának beállítása másodperc felbontásban (egy lehetséges eset)

```
ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY-MM-DD HH24:MI:SS';
```

Dátum és idő megjelenítése és használata

A következő mintapéldákban bemutatjuk a dátum és idő kezelésében legfontosabb négy függvény, a TO_DATE, a TO_CHAR, a TO_NUMBER konverziós függvények, valamint az aktuális dátumot (és időt) megadó sysdate dátumfüggvény használatát (részletesebben *lásd* [16] és [12]).

1.9. példa

Ebben a példában bemutatjuk az aktuális dátum és idő formátummaszkos megjelenítését.

1. Az aktuális dátum a rendszer dátumformátumában

```
SELECT RPAD('sysdate', 13)  AS "Dátumfüggvény",
       LPAD(sysdate, 25)    AS "A rendszer dátumformátuma"
```

```
    FROM dual;
```

Eredmény

```
Dátumfüggvény A rendszer dátumformátuma
------------- ------------------------
sysdate                        04-DEC-19
```

2. Az aktuális dátum a magyar dátumformátumban

```
SELECT RPAD('YYYY.MM.DD.', 15)
         AS "A formátummaszk",
       LPAD(TO_CHAR(sysdate,'YYYY.MM.DD.'), 23)
         AS "A magyar dátumformátuma"
  FROM dual;
```

Eredmény

```
A formátummaszk A magyar dátumformátuma
--------------- -----------------------
YYYY.MM.DD.                  2004.12.19.
```

3. Az alapértelmezett aktuális rendszerdátum és -idő

```
SELECT RPAD('YYYY.MM.DD, HH24:MI:SS', 22)
         AS "A formátummaszk",
       RPAD(TO_CHAR(sysdate,'YYYY.MM.DD, HH24:MI:SS'), 21)
         AS "Az aktuális dátum-idő"
  FROM dual;
```

Eredmény

```
A formátummaszk        Az aktuális dátum-idő
---------------------- --------------------
YYYY.MM.DD, HH24:MI:SS 2004.12.19, 13:07:52
```

4. Az aktuális idő

```
SELECT RPAD('HH:MI:SS', 15)                  AS "A formátummaszk",
       LPAD(TO_CHAR(sysdate,'HH:MI:SS'), 8)  AS "Idő(12)"
  FROM dual;
```

Eredmény

```
A formátummaszk Idő(12)
--------------- --------
HH:MI:SS        01:11:24
```

5. Az aktuális idő 24 órás formátumban

```
SELECT RPAD('HH24/MI', 15)                   AS "A formátummaszk",
       LPAD(TO_CHAR(sysdate,'HH24/MI'), 7)   AS "Idő(24)"
  FROM dual;
```

Eredmény

```
A formátummaszk Idő(24)
--------------- -------
HH24/MI           13/13
```

1.10. példa

E példában bemutatjuk a TO_DATE függvény használatát.

0. A rendszerdátum formátumának beállítása

```
ALTER SESSION SET NLS_DATE_FORMAT = 'YY-MM-DD';
```

Eredmény

```
A munkamenet módosítva.
```

1. A TO_DATE függvény használata formátummaszkkal karakterlánc esetén

```
SELECT RPAD('''20041214'',YYYYMMDD', 24)
         AS "Karakterlánc és formátum",
       LPAD(TO_DATE('20041214','YYYYMMDD'), 16)
         AS "Konvertált dátum"
  FROM dual;
```

Eredmény

```
Karakterlánc és formátum Konvertált dátum
------------------------ ----------------
'20041214',YYYYMMDD             04-DEC-14
```

2. A TO_DATE függvény használata formátummaszkkal szám esetén

```
SELECT RPAD('20041214,YYYYMMDD', 17)
         AS "Szám és formátum",
       LPAD(TO_DATE(20041214,'YYYYMMDD'), 16)
         AS "Konvertált dátum"
  FROM dual;
```

Eredmény

```
Szám és formátum  Konvertált dátum
----------------- ----------------
20041214,YYYYMMDD         04-DEC-14
```

1.11. példa

Az alábbiakban összefoglaljuk a dátumformák és a -függvények használatát. Az egyes dátumkiíró utasítások és futási eredményük értékelésénél vegyük figyelembe a rendszer aktuális dátumformátumát!

1. Állítsuk be a rendszerdátum formátumát évszázadot NEM tartalmazó alakra, és adjunk meg évszázadhiányos, múlt századi dátumot

```
ALTER SESSION SET NLS_DATE_FORMAT = 'YY-MON-DD';

SELECT RPAD('81-JAN-10',14)           AS "Megadott dátum",
       LPAD (TO_DATE('81-JAN-10'),16) AS "Konvertált dátum"
  FROM dual;
```

Eredmény

```
A munkamenet módosítva.
Megadott dátum Konvertált dátum
-------------- ----------------
81-JAN-10             81-JAN-10
```

2. Írassuk a fenti dátumot teljes évszázados formátumban

```
SELECT RPAD('81-JAN-10',14)           AS "Megadott dátum",
       LPAD (TO_CHAR(TO_DATE('81-JAN-10'),'YYYY-MON-DD'),16)
                                      AS "Konvertált dátum"
  FROM dual;
```

Eredmény

```
Megadott dátum Konvertált dátum
-------------- ----------------
81-JAN-10           2081-JAN-10
```

Megjegyzés
Tehát az évszázad nélkül megadott dátum 21. századi dátumot jelent, ami kiderül abból, ha évszázadot tartalmazó formátumban íratjuk ki.

3. Az eltelt évek számának meghatározása

```
SELECT LPAD(TO_NUMBER(TO_CHAR(TO_DATE('81-JAN-10'), 'YYYY'))
            - TO_NUMBER(TO_CHAR(sysdate, 'YYYY')), 14)
         AS "Dátum - MaiNap"
  FROM dual;
```

Eredmény

```
Dátum - MaiNap
--------------
            77
```

Megjegyzés
A 2081–2004 valóban 77 év. Ez számértékileg helyes, egyébként nem.

4. Váltsunk át a magyar dátumformára, és most is kérdezzük le ugyanazt a dátumot

```
ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY.MM.DD';

SELECT RPAD('81-JAN-10',14)              AS "Megadott dátum",
       LPAD (TO_CHAR(TO_DATE('81-JAN-10'),'YYYY-MON-DD'),16)
                                         AS "Konvertált dátum"
  FROM dual;
```

Eredmény

```
A munkamenet módosítva.
Megadott dátum Konvertált dátum
-------------- ----------------
81-JAN-10           0081-JAN-10
```

Megjegyzés

Tehát az évszázad nélkül megadott dátum magyar dátumformátummal 0081 lesz.

5. Az eltelt évek számának meghatározása

```
SELECT LPAD(TO_NUMBER(TO_CHAR(TO_DATE('81-JAN-10'), 'YYYY'))
            - TO_NUMBER(TO_CHAR(sysdate, 'YYYY')), 14)
         AS "Dátum - MaiNap"
  FROM dual;
```

Eredmény

```
Dátum - MaiNap
--------------
         -1923
```

Megjegyzés

Tehát 1923 év telt el 81. jan. 10. óta. Számértékileg ez is helyes, de egyébként nem.

6. Állítsuk vissza a rendszer dátumformáját alapértelmezésre

```
ALTER SESSION SET NLS_DATE_FORMAT = 'YY-MON-DD';
SELECT sysdate FROM dual;
```

Eredmény

```
A munkamenet módosítva.
SYSDATE
---------
04-DEC-24
```

Megjegyzés

- Ha a rendszerdátum évszázadmegadást nem tartalmaz, akkor a hiányos dátumot 21. századiként (tehát a `81-JAN-10` alakot `2081-JAN-10` dátumként), ha pedig évszázadmegadást tartalmaz, akkor 1. századiként értelmezi (tehát a `81-JAN-10` alakot `0081-JAN-10` dátumként).
- A fentiek tanulsága az, hogy fokozottan ügyelni kell a hiányos (évszázadmegadást nem tartalmazó) dátumforma használatára, mivel annak értelmezése függ az aktuális rendszerdátum beállításától. Ha a rendszerdátum formátumának aktuális állapotától függetlenül akarunk korábbi évszázadi dátumot megadni, akkor azt célszerű az évszázad megjelölésével tenni (például `'1981.JAN.10'` alakban).
- Figyeljünk fel arra, hogy a `TO_DATE` függvény, mely alapfunkciójaként a karakteres megadású dátumot dátumtípusra konvertálja, automatikus dátumkonverziót is végez a mindenkori rendszerdátum formátumára.

1.12. példa

Ebben a példában bemutatjuk az eltelt idő meghatározásának módját, ha a felbontást napokban, hónapokban, illetve években szeretnénk megkapni.

1. Az eltelt idő meghatározása (az eredmény napokban)

1A. módszer

```
SELECT LPAD(ROUND(sysdate - TO_DATE('2003-DEC-19')), 21)
       AS "Eltelt idő (napokban)"
  FROM dual;
```

1B. módszer

```
SELECT LPAD(ROUND(sysdate - TO_DATE('2003-DEC-19','YYYY-MM-DD')), 21)
       AS "Eltelt idő (napokban)"
  FROM dual;
```

1C. módszer

```
SELECT LPAD(ROUND(sysdate - TO_DATE(20031219,'YYYYMMDD')), 21)
       AS "Eltelt idő (napokban)"
  FROM dual;
```

Eredmény (mindhárom módszer esetén)

```
Eltelt idő (napokban)
---------------------
                  368
```

2. Az eltelt idő meghatározása (az eredmény hónapokban)

```
SELECT LPAD(ROUND(MONTHS_BETWEEN(sysdate, TO_DATE('2003-DEC-19'))), 23)
       AS "Eltelt idő (hónapokban)"
  FROM dual;
```

Eredmény

```
Eltelt idő (hónapokban)
-----------------------
                     12
```

Megjegyzés
A MONTHS_BETWEEN függvény az eltelt idő értéket törthónapként (azaz pontos értékként) adja vissza.

3. Az eltelt idő meghatározása (az eredmény években)

3A. módszer

```
SELECT LPAD(TO_NUMBER(TO_CHAR(sysdate, 'YYYY'))
          - TO_NUMBER(TO_CHAR(TO_DATE('2003-DEC-19'),'YYYY')), 20)
         AS "Eltelt idő (években)"
  FROM dual;
```

3B. módszer

```
SELECT  LPAD(ROUND(MONTHS_BETWEEN(sysdate,  TO_DATE('2003-DEC-19'))/12),
20)
         AS "Eltelt idő (években)"
  FROM dual;
```

Eredmény (mindkét módszer esetén)

```
Eltelt idő (években)
--------------------
                   1
```

1.13. példa

E példában az aktuális és az eltelt idő megjelenítését mutatjuk be. Ez utóbbit oly módon, hogy a felbontás legyen másodperc, illetve nap.

1. A dátum-idő kiíratása a rendszer dátumformátumában

```
ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY/MM/DD.HH24:MI:SS';
SELECT sysdate  AS "Dátum-idő"
  FROM dual;
```

Eredmény

```
Dátum-idő
-------------------
2004/12/19.22:47:49
```

2. A dátum-idő kiíratása formátummaszk szerint

```
SELECT RPAD('YYYY-MON-DD.HH:MI:SS', 20)
         AS "A formátummaszk",
       TO_CHAR(sysdate,'YYYY-MON-DD.HH:MI:SS')
         AS "Dátum-idő"
  FROM dual;
```

Eredmény

```
A formátummaszk      Dátum-idő
-------------------- --------------------
YYYY-MON-DD.HH:MI:SS 2004-DEC-19.10:47:59
```

3. Az eltelt idő meghatározása (az eredmény másodpercekben)

```
ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY-MM-DD.HH24:MI:SS';
SELECT LPAD((sysdate - TO_DATE('2004.12.19,23:08:05'))*86400, 27)
         AS "Eltelt idő (másodpercekben)"
  FROM dual;
```

Eredmény

```
Eltelt idő (másodpercekben)
---------------------------
                         18
```

Megjegyzés

Ha a rendszer dátumformátumát kiegészítjük az óra-perc-másodperc időformátummal, akkor egyrészt a `sysdate` dátumfüggvény felbontása is megnő (az általa visszaadott aktuális dátumérték kiegészül a rendszer dátumformátuma szerinti időadatokkal), másrészt két dátumérték kivonása az eltelt időt is másodperc felbontással, ám törtnapban adja meg. Annak érdekében, hogy az eltelt idő dimenziója másodperc legyen, a kapott értéket meg kell szorozni a napban lévő másodpercek számával (24*60*60 = 86 400).

4. Az eltelt idő meghatározása (az eredmény napokban)

```
ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY-MM-DD.HH24:MI:SS';
SELECT LPAD(ROUND(sysdate - TO_DATE('2004.12.17,23:08:05')), 21)
         AS "Eltelt idő (napokban)"
  FROM dual;
```

Eredmény

```
Eltelt idő (napokban)
---------------------
                    2
```